One Analog Way, Wilmington, MA 01887 U.S.A. • Tel: 781.935.5565 • www.analog.com

Preliminary Technical Data

System Development User Guide for the RF Agile Transceiver Family

ADRV9001 SYSTEM DEVELOPMENT USER GUIDE OVERVIEW

The ADRV9001 is family designator assigned to the System Development User Guide (UG-1828 for new ADRV9002, ADRV9003, ADRV9004, and upcoming additional family members).

The ADRV9001 System Development User Guide covers:

- ADRV9002 integrated dual RF transceiver
- ADRV9003 integrated single RF transceiver (excludes DPD)
- ADRV9004 integrated dual RF transceiver (excludes DPD)

This user guide provides details on functionality across the entire family. Some family members do not include all the features or functions. Refer to the individual product data sheet for each product available features and functions.

TABLE OF CONTENTS

HOW TO USE THIS DOCUMENT

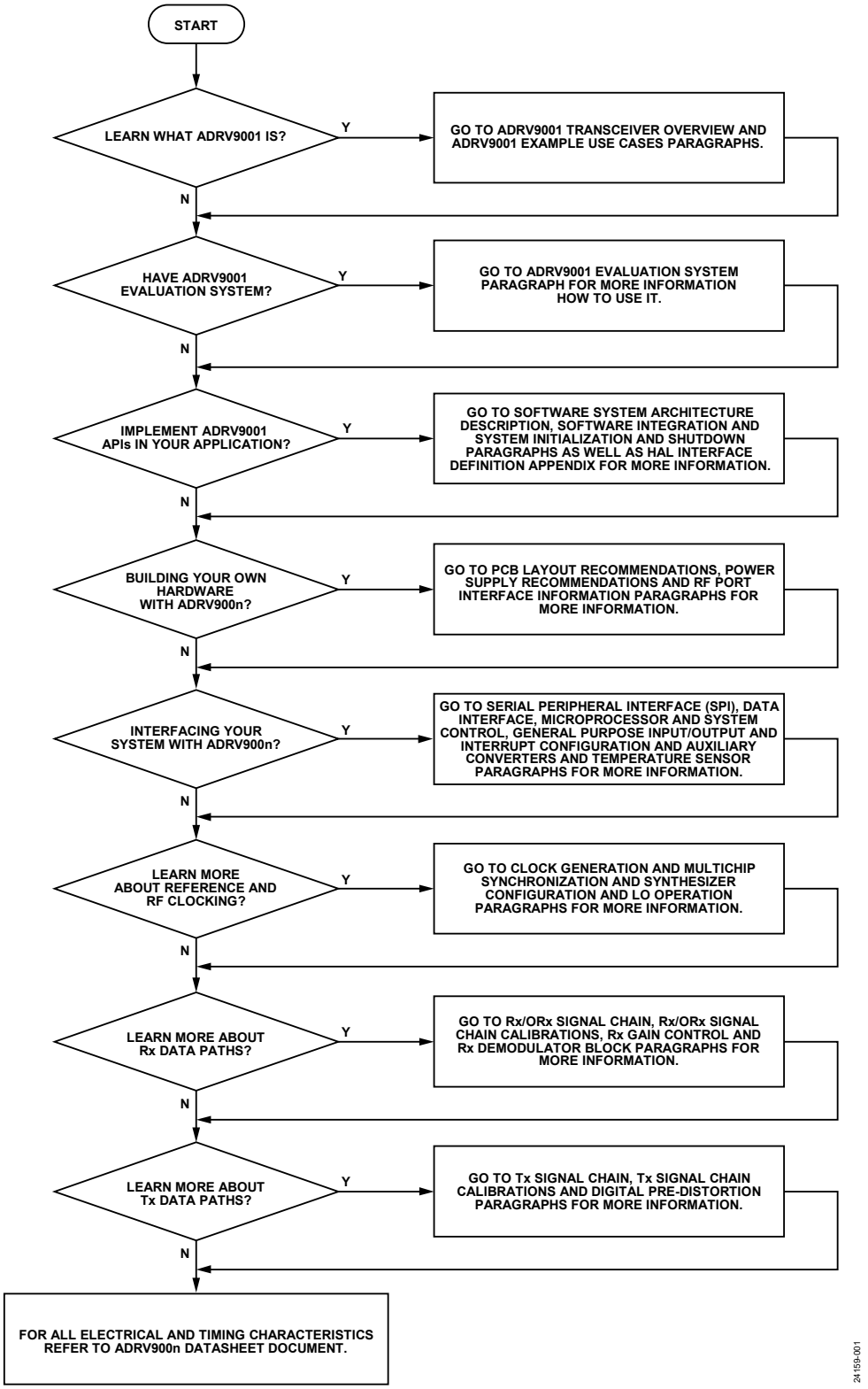


Figure 1. Document Flowchart for Document Navigation

BLOCK DIAGRAM

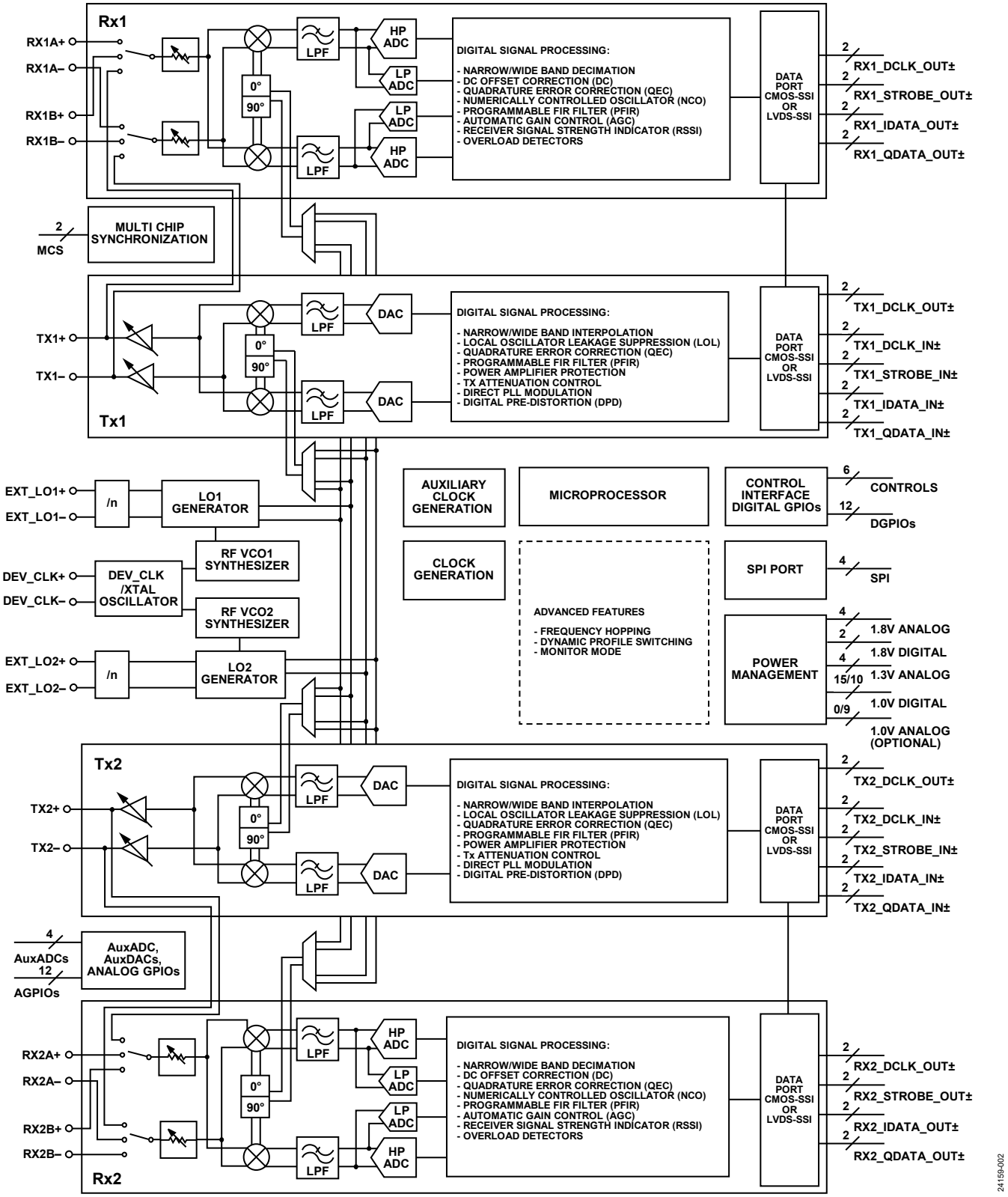


Figure 2. ADRV9002 Block Diagram

PRODUCT HIGHLIGHTS

ADRV9002

The ADRV9002 delivers a versatile combination of high performance and low power consumption required by battery powered radio equipment and can operate in both frequency division duplex (FDD) and time division duplex (TDD) modes. The ADRV9002 operates from 30 MHz to 6000 MHz covering the VHF, licensed and unlicensed cellular bands, and ISM bands. The IC is capable of supporting both narrowband and wideband standards up to 40 MHz bandwidth on both receiver and transmitter.

The transceiver consists of direct conversion signal paths with state-of-the-art noise figure and linearity. Each complete receiver and transmitter sub-system includes DC offset correction, quadrature error correction, and programmable digital filters, eliminating the need for these functions in the digital baseband. In addition, several auxiliary functions such as an auxiliary ADC, auxiliary DACs, and GPIOs are integrated to provide additional monitoring and control capability.

The fully integrated phase locked loops (PLLs) provide high performance, low power fractional-N frequency synthesis for the transmitter, receiver, and clock sections. Careful design and layout techniques have been implemented to provide the isolation demanded in high performance Mobile Radio applications.

All VCO and loop filter components are integrated to minimize the external component count. The LOs have flexible configuration options and include fast lock modes.

The transceiver includes low power sleep and monitor modes to save power, which extends battery life of portable devices while continuing to monitor communication.

The fully integrated low power digital predistortion (DPD) is supported by ADRV9002. It can linearize wideband signals as well as it has been optimized for narrowband type signals to enable linearization of high efficiency power amplifiers. In use cases where the integrated DPD is used, main receivers are used as a power amplifier observation path.

Power supply for ADRV9002 is distributed across four or five different voltage supplies: 2 or 3 analog and 2 digital. The analog supplies are 1.8 V, 1.3V, and 1.0V (in internal LDO bypass mode). 1.3V domain feeds directly some blocks and also internal LDO regulators for some functions to maximum performance. 1.8 V analog domain is used to optimize transmitter and auxiliary converter performance. The digital processing blocks are supplied by a 1.0V source. In addition, a 1.8 V supply is used to supply all GPIO and interface ports that connect with the baseband processor.

High data rate and low data rate interfaces are supported using configurable CMOS or LVDS Synchronous Serial Interface choice.

The core of the ADRV9002 is controlled via a standard 3 or 4-wire serial port. All software control is communicated via this interface. There is also a control interface that uses GPIO lines to provide hardware control to and from the device. These pins can be configured to provide dedicated sets of functions for different application scenarios.

The block diagram in Figure 2 shows a high level view of the functions in the ADRV9002. Descriptions of each block with setup and control details are provided in subsequent sections of this document.

BANDWIDTH AND SAMPLE RATE SUPPORT

The ADRV9002 supports the reception and transmission of channels up to 40 MHz bandwidth. Standard sample rates of 24 kHz (typically for narrowband FM waveforms), 144 kHz and 288 kHz (typically for TETRA signals), and 1.92 MHz, 3.84 MHz, 7.68 MHz, 15.36 MHz, 23.04 MHz, 30.72 MHz, and 61.44 MHz (typically for LTE signals) are available.

In addition, the ADRV9002 supports an almost continuous range of sample rates between 24 kHz and 61.44 MHz. Some sample rates cannot be supported due to internal clocking constraints.

Sample rate scaling is accomplished by enabling or disabling decimation or interpolation filters in the digital signal chain.

Data Interfaces

The ADRV9002 supports both CMOS and LVDS electrical interfaces for its data lanes. All data lanes support both electrical interfaces, but concurrent operation of both interfaces is not supported. Each receive and transmit channel has a dedicated set of lanes for transferring information.

The CMOS bus speed is limited to 80 MHz. Two operating modes are available for the CMOS-SSI electrical interface. For low sample rates, a mode in which 32 bits (16 bits of I and Q data each) are serialized over a single lane, with two additional lanes total required for a clock (SDR or DDR) and a frame synchronization signal, supports a maximum sample rate of 2.5 MHz.

For sample rates above 2.5 MHz, single channel data is serialized over four lanes, with two additional lanes total required for a clock (SDR or DDR) and a frame synchronization signal, supporting a maximum sample rate of 20 MHz.

The LVDS electrical interface supports two modes of operation. The 32 total bits of I and Q data are serialized over one LVDS lane (32 bits composed of 16 bits of I and 16 bits of Q data) or two LVDS-SSI lanes (each dedicated to 16 bits of I or Q data), with two additional lanes total required for a DDR clock and a frame synchronization signal. Sample rates ranging from 24 kHz to 61.44 MHz are supported via the LVDS-SSI interface, resulting in a maximum lane rate of 983.04 MHz.

Note that in LVDS-SSI mode, 12-bit I and Q words are supported for most sample rates.

RF LO Frequency Range and Multiplexing

The ADRV9002 supports a RF LO range from 30 MHz to 6 GHz. RF LOs can be generated via two internal PLLs, or applied externally to the device. When LOs are provided from an external source, double or more of the desired frequency must be applied to the ADRV9002 to allow for the generation of quadrature signals internally.

An LO multiplexing scheme exists on the ADRV9002, that allows for the routing of either of the RFPLLs to any of the transmit or receive channels. The RF channels and RFPLLs can operate concurrently and independently, off a common reference clock, thus enabling: FDD operation, single or dual frequency repeater operation, multi-band TDD operation, and diversity operation amongst various other configurations.

Frequency Hopping

The ADRV9002 supports various forms of frequency hopping, with the main distinguishing factor between them being frequency transition time. RFPLL phase noise, and QEC and LOL algorithm performance may degrade as a function of decreasing frequency transition time.

A fast frequency hopping (FFH) mode exists that supports 64 hop frequencies or less, that are pre-loaded by the user onto the ADRV9002 at power-up. In this mode, the 64 frequencies are cycled through in a circular buffer fashion. Hopping between the frequencies in FFH mode is triggered via a GPIO pin toggle. An API command with SPI transaction can also trigger a frequency hop, albeit with a longer frequency transition time.

A random order FFH mode is also supported, whereby a finite set of frequencies already pre-loaded onto the ADRV9002 can be hopped between in a random manner dictated by the user. Selecting the next frequency to hop to is accomplished by asserting a frequency index word onto the GPIO bus. Alternatively, the API can be used to select the next frequency index, albeit with a longer frequency transition time.

In addition to FFH mode, the ADRV9002 supports other frequency hopping modes where the desired hop frequencies need not be pre-loaded into on-board memory. In these modes, desired hop frequencies can be streamed in via the API. Frequency transition times in these modes are greater than that available in FFH mode.

Note that all frequency hopping modes are available for use in conjunction with the monitor mode described in the Power Consumption Modes section.

Profile Switching

The ADRV9002 supports rapid switching between different RF channel profiles. A transmit or receive RF channel profile contains settings such as bandwidth, sample rate, filtering, input port selection, AGC settings, and algorithm configuration. The profile switching mode enables the support of waveforms that vary modulation schemes and bandwidths dynamically.

Low IF Reception

The receive digital datapath on the ADRV9002 contains an optional digital mixer that is driven by a programmable NCO. The RX LO is offset from the frequency of the desired channel, and then the digital mixer and NCO are used to down convert signal to base-band before being processed by their baseband processor.

There are several advantages to offset the RX LO from the frequency of the desired channel: Impairments that exist about the RX LO, such as LO-leakage, can be avoided. The effect of flicker noise from base-band circuits can be mitigated since the received signal is offset from DC in the analog signal path. Also, image rejection can be improved if the RX LO is offset enough from the desired channel, such that the image frequency lies in the attenuation region of the user's external RF filter.

The low IF reception mode is targeted predominately towards low bandwidth channels, which supports offsets range of $\pm$ 20 MHz about the receiver LO.

Receive Dynamic Range and Blocking

As depicted in Figure 2, the ADRV9002 receive path consists of an input mixer, followed by a base-band filter that drives an ADC. A highly programmable digital decimation and filtering datapath follows the ADC. RF analog gain control is provided in analog attenuator, and additional gain is provided in the digital datapath via AGC loops.

The ADC in the receive chain possesses a high dynamic range. Assuming a mixer gain of 0 dB, the ADC's noise and maximum input power referred to the RF input are -142 dBm/Hz and 8.6 dBm, respectively. These levels translate into a dynamic range in excess of 150 dB on a per Hertz basis. Taking into account the digital filtering and AGC loops, an even greater dynamic range can be achieved.

Given the high dynamic range of the receiver ADC, very little channelization or blocker filtering occurs in the analog signal chain since the ADC can simultaneously absorb weak signals and large blockers. Blocker suppression and channelization are then achieved in the digital signal path.

If reciprocal mixing of the RX LO phase noise by a large blocker close to the desired channel significantly degrades blocking performance, a lower phase noise external LO source can be used in place of the on-board RFPLLs.

The receive path also contains two types of ADCs connected to the chip's RF front end, that allow for the trade-off between power consumption and dynamic range: a high performance ADC, and low power ADC that possess degraded dynamic range. Users can trade-off receive channel dynamic range and power consumption by selecting between either set of ADCs.

Power Consumption Modes

The ADRV9002 provides users with various levels of power control. Power scaling on individual analog signal path blocks can be performed to trade-off power and performance. In addition, enabling and disabling various blocks in TDD RX and TX frames to reduce power can be customized, at the expense of RX/TX or TX/RX turnaround time.

A specialized "RX Monitor mode" exists that allows the ADRV9002 to autonomously poll a region of the spectrum for the presence of a signal, while in a low power state. In this mode, the chip continuously cycles through sleep-detect-sleep states controlled by an internal state machine. Power savings are achieved by ensuring that the sleep duty cycle is greater than the "detect" duty cycle.

In the "sleep" state, the chip is in a minimal power consumption configuration where few functions are enabled. After a pre-determined period, the chip enters the "detect" state. In this state, the chip enables a receiver and performs a power measurement over a bandwidth and at a RX LO frequency determined by the user. If the measured power level in the bandwidth is greater than a user-determined threshold, the "Monitor Mode" state machine exits its cycle. Following the loop exit, an interrupt is provided via a GPIO pin to the user's baseband processor, and the entire receiver analog and digital chains within the ADRV9002 are powered up, assuming that normal signal reception resumes due to the detection of a channel.

If the power measured over the bandwidth is less than the user-determined threshold, the chip resumes its sleep-detect-sleep cycle. The sleep-detect duty cycle and durations, power measurement threshold, and RX LO are user-programmable, and are set before enabling monitor mode.

Note that frequency hopping can be combined with monitor mode, allowing the ADRV9002 to dynamically change the RX LO while performing the power measurement function.

ADRV9003

The ADRV9003 delivers all features offered by ADRV9002 transceiver. Differences between ADRV9002 and ADRV9003 include:

- RF IOs. The ADRV9003 offers two receivers and one transmitter.
- Digital predistortion functionality is not supported by the ADRV9003.

ADRV9004

The ADRV9004 delivers all features offered by ADRV9002 transceiver. Differences between ADRV9002 and ADRV9004 include:

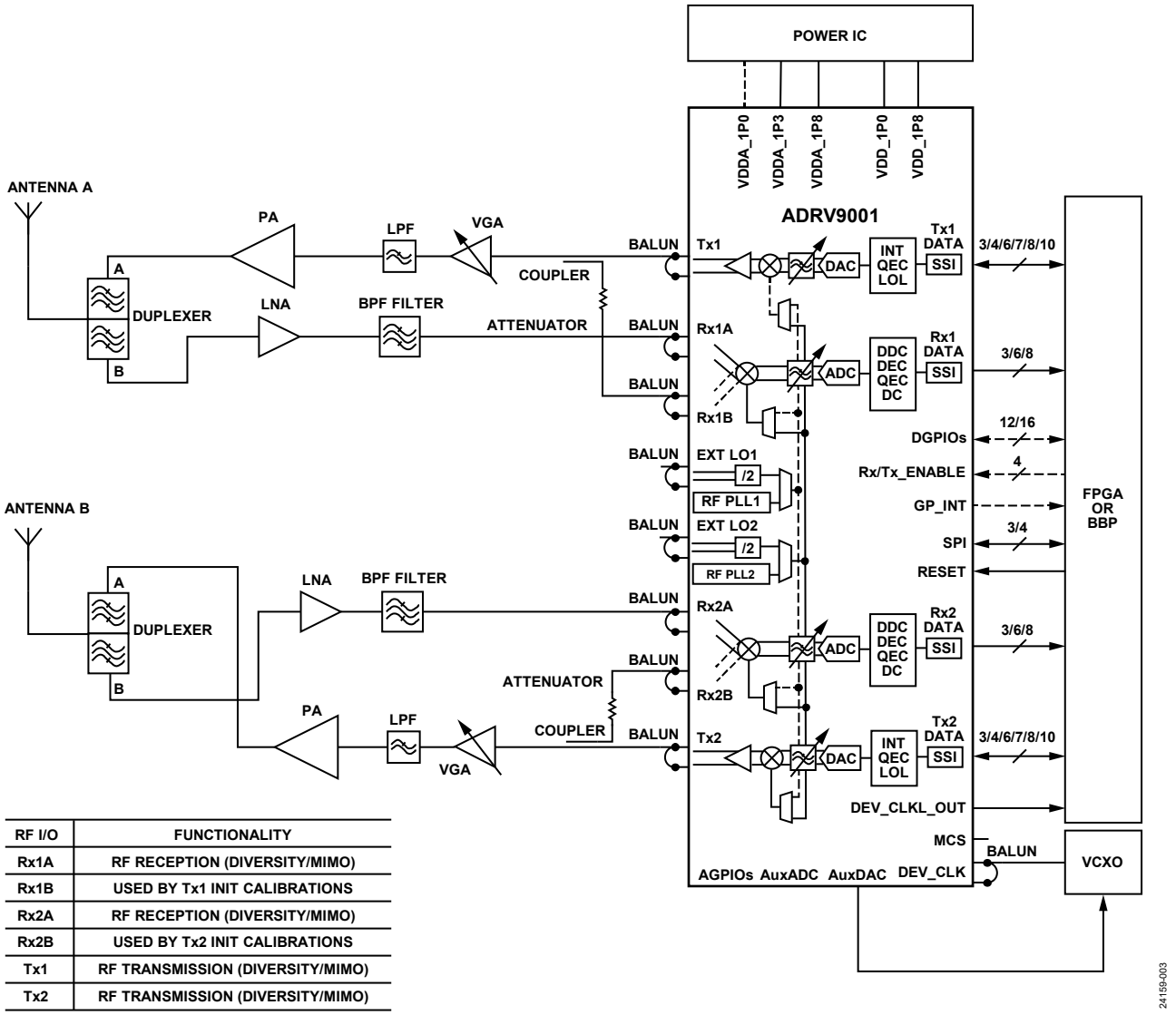
- Digital predistortion functionality is not supported by the ADRV9004.

ADRV9001 EXAMPLE USE CASES

The intention of this section is to provide the reader with the overall idea how ADRV9001 integrated transceiver can operate as an RF front end in different applications. The provided list is not exhaustive, and there are other applications in which the ADRV9001 can serve.

Each example is accompanied with a table that explains the main limitations and highlights what the customer should look for when implementing the ADRV9001 in the end application.

ADRV9001 IN A SINGLE-BAND 2RT2R FDD TYPE SMALL-CELL APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RF RECEPTION (DIVERSITY/MIMO)
Rx1B	USED BY Tx1 INIT CALIBRATIONS
Rx2A	RF RECEPTION (DIVERSITY/MIMO)
Rx2B	USED BY Tx2 INIT CALIBRATIONS
Tx1	RF TRANSMISSION (DIVERSITY/MIMO)
Tx2	RF TRANSMISSION (DIVERSITY/MIMO)

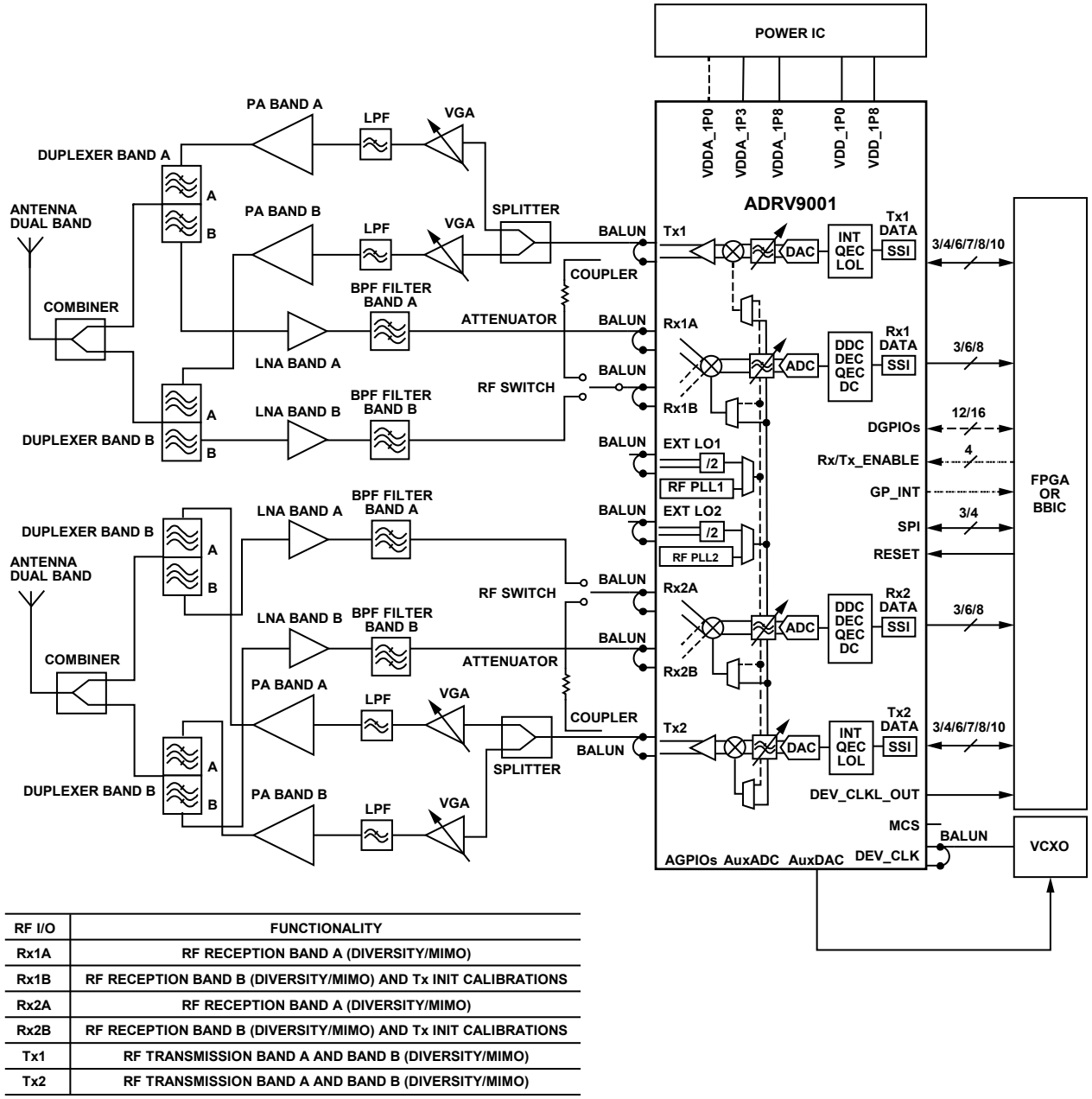
Figure 3. ADRV9001 in Single-Band 2T2R FDD Type Small Cell Application

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF to bits signal chains that can serve as the RF front end in small cell type applications. The ADRV9001 dual Rx and Tx signal chains allow the user to implement MIMO or diversity in their system. The ADRV9001 internal AGC can be used to autonomously monitor and set appropriate gain levels for Rx signal chains. For non-time critical FDD type applications, control of the ADRV9001 TRx can be done through API commands that use the SPI interface.

Table 1. Constrains and Limitations in Single-Band 2T2R FDD Type Small-Cell Application

Functionality	Constrains and Limitations
Receiver Signal Path	The user must ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, the RxB inputs are used only during initialization calibrations. Ensure that the appropriate attenuation is present in line to prevent Rx being overloaded by Tx signal.
Transmit Signal Path	The user must ensure that appropriate level of isolation between Tx1 and Tx2 as well as Rx to Tx is provided at the system level.
LO Generation	In FDD type Small Cell application, ADRV9001 can use its internal LO to generate RF LO1 for uplink (Rx1 and Rx2) and RF LO2 for downlink (Tx1 and Tx2). It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2x RF LO can be used for uplink and External LO2 operating at 2x RF LO can be used for downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as -50dBc and 3rd harmonic can be as high as −9 dBc. Therefore the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality is not available when ADRV9001 operates in 2R2T FDD mode.
Calibrations	During Rx initialization sequence user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence user needs to ensure that the power amplifier is powered down to avoid unwanted emission of transmitter calibration tones at the antenna. No transmitter tracking calibrations are available when ADRV9001 operates in 2R2T FDD mode.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock and control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power-up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN DUAL-BAND 2RT2R FDD TYPE SMALL-CELL APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RF RECEPTION BAND A (DIVERSITY/MIMO)
Rx1B	RF RECEPTION BAND B (DIVERSITY/MIMO) AND Tx INIT CALIBRATIONS
Rx2A	RF RECEPTION BAND A (DIVERSITY/MIMO)
Rx2B	RF RECEPTION BAND B (DIVERSITY/MIMO) AND Tx INIT CALIBRATIONS
Tx1	RF TRANSMISSION BAND A AND BAND B (DIVERSITY/MIMO)
Tx2	RF TRANSMISSION BAND A AND BAND B (DIVERSITY/MIMO)

Figure 4. ADRV9001 in Dual-Band 2T2R FDD Type Small-Cell Application

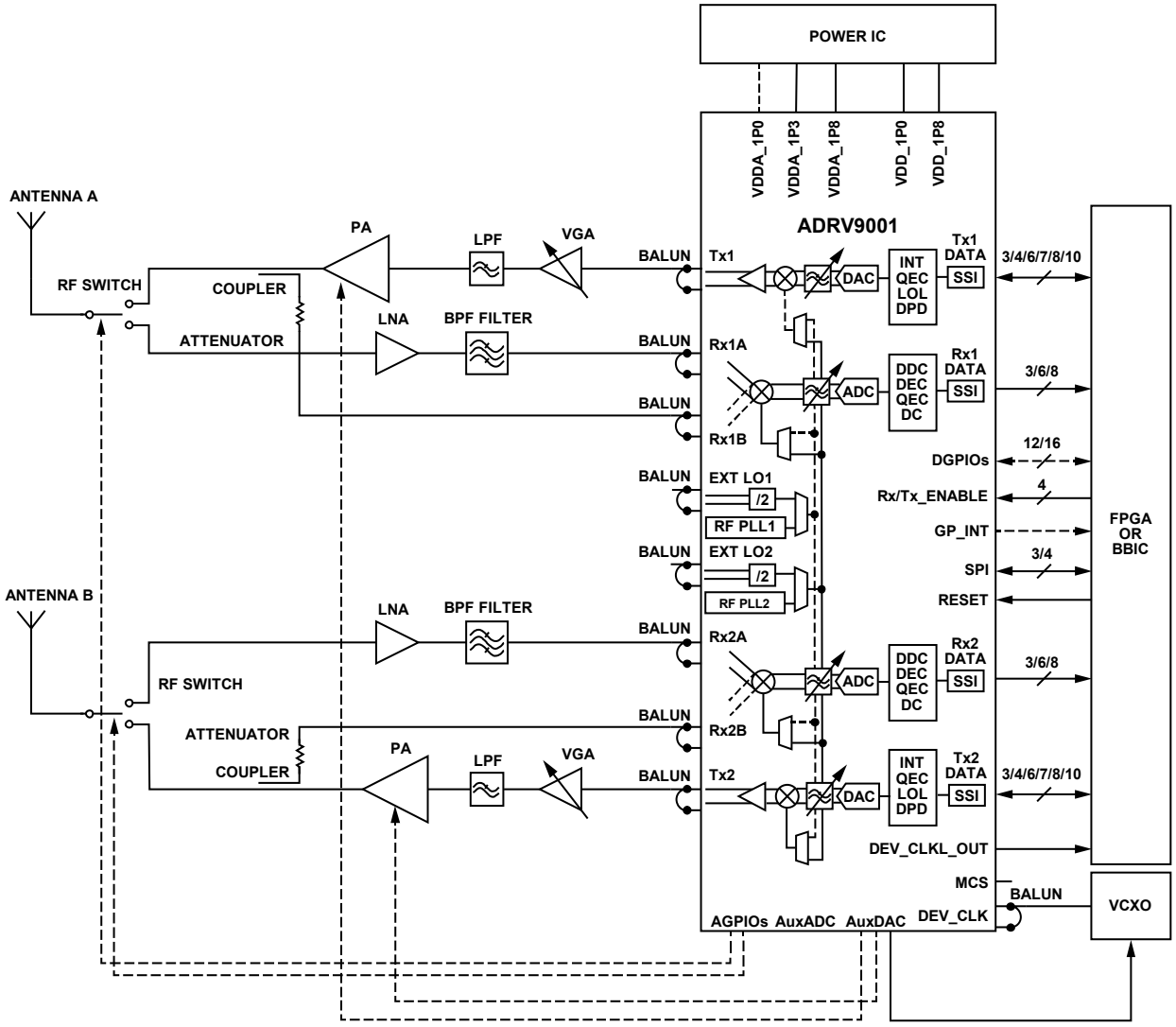
Dual-Band 2T2R FDD Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete dual and RF-to-bits signal chain that can serve as RF front end in small cell type applications. Note that in proposed solution, only one band can be used at the time. ADRV9001 dual Rx and Tx signal chains enables user to implement MIMO or diversity in their system. ADRV9001 internal AGC can be used to autonomously monitor and set appropriate gain level for Rx signal chains. For none time critical FDD type applications control of the ADRV9001 TRx can be done thru API commands that use SPI interface.

Table 2. Constrains and Limitations in Dual-Band 2T2R FDD Type Small-Cell Application

Functionality	Constrains and Limitations
Rx Signal Path	The user must ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, RxB inputs are used to work with Rx Band B signals as well as during initialization calibrations. In this scenario, RF Balun selected for RxB inputs must work with both Band B and Tx Bands. The user must ensure that appropriate attenuation is present in line to prevent Rx being overloaded by Tx signal.
Tx Signal Path	The user must ensure that appropriate level of isolation between Tx1 and Tx2 as well as Rx to Tx is provided at the system level.
LO Generation	In FDD type small cell applications, ADRV9001 can use its internal LO to generate RF LO1 for uplink (Rx1 and Rx2) and RF LO2 for downlink (Tx1 and Tx2). It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2× RF LO can be used for uplink and External LO2 operating at 2× RF LO can be used for downlink. It should be noted that only one set of Rx inputs can be used at the time. This system can operate with two different FDD bands but only one of those bands can be active at particular moment in time.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality is not available when ADRV9001 operates in 2R2T FDD mode.
Calibrations	During Rx initialization sequence user needs to ensure that there are no signals at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During the transmitter initialization sequence, the user needs to ensure that the power amplifier is powered down to avoid unwanted emission of transmitter calibration tones at the antenna. No transmitter tracking calibrations are available when ADRV9001 operates in 2R2T FDD mode.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide a reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before the ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN SINGLE-BAND 2T2R TDD TYPE SMALL-CELL APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RF RECEPTION (DIVERSITY/MIMO)
Rx1B	USED BY Tx1 INIT CALIBRATIONS
Rx2A	RF RECEPTION (DIVERSITY/MIMO)
Rx2B	USED BY Tx2 INIT CALIBRATIONS
Tx1	RF TRANSMISSION (DIVERSITY/MIMO)
Tx2	RF TRANSMISSION (DIVERSITY/MIMO)

Figure 5. Single-Band 2T2R FDD Type Small-Cell Application

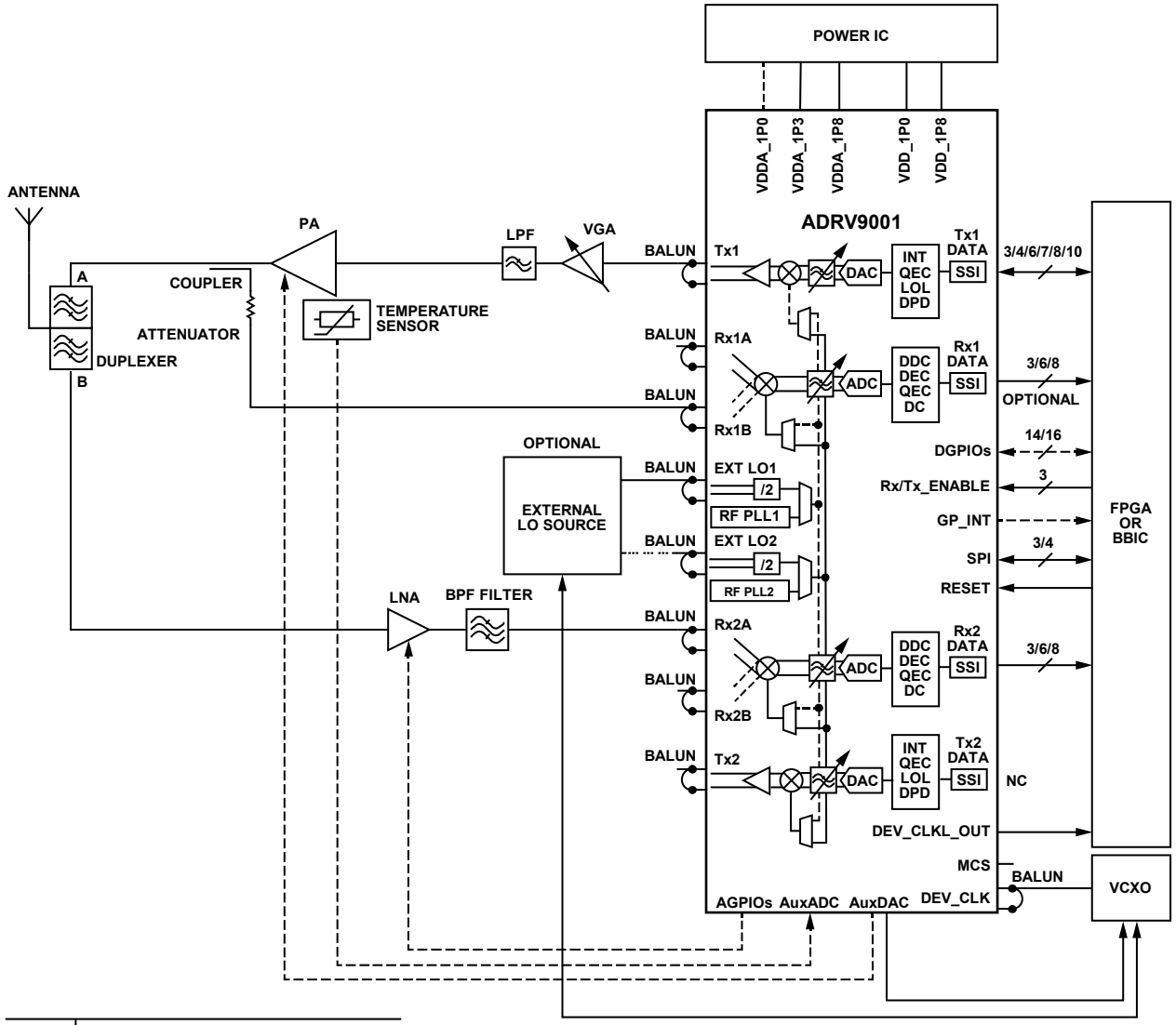
Single-Band 2T2R TDD Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in TDD type small cell type applications. ADRV9001 dual Rx and Tx signal chains enables user to implement MIMO or diversity in their system. In TDD type applications, internal DPD block can be used to linearize external power amplifier and improve overall system efficiency. ADRV9001 internal AGC can be used to autonomously monitor and set appropriate gain level for Rx signal chains. For time critical TDD type applications control of the ADRV9001 TRx can be done by toggling control lines. ADRV9001 can control external Rx/Tx switch using its analog GPIOs as well as provide power amplifier bias voltage by utilizing AuxDAC outputs.

Table 3. Constrains and Limitations in Single-Band 2T2R FDD Type Small-Cell Application

Functionality	Constrains and Limitations
LO Generation	In TDD type small cell applications, ADRV9001 can use its internal LO to generate RF LO1 for both uplink and downlink. It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2× RF LO can be used for both uplink and downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates a square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality can be used in the 2R2T TDD mode. Maximum channel bandwidth that DPD can support is limited by ADRV9001 RF bandwidth divided by 3 or by 5. The DPD operation can be performed by ADRV9001 or observation receiver data can be sent to the baseband processor via the receiver data port during transmit operation. The receiver path used during DPD operation to perform transmitter observation is also used by the transmitter tracking calibrations. In case of external DPD, the user must ensure that access to the receiver path during transmit slots is time-shared between DPD operation and transmitter calibrations.
Calibrations	During Rx initialization sequence user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence, the user needs to ensure that Power Amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. ADRV9001 needs to access Rx datapath during Tx time slots for Tx tracking calibration to operate. If user use Tx observation path with DPD functionality performed by baseband processor, then access to the Rx datapath during Tx slots must be time-shared between DPD operation and Tx calibrations.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	The ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN 1T1R FDD WITH DPD TYPE APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	NOT USED
Rx1B	RF RECEPTION
Rx2A	RF RECEPTION
Rx2B	NOT USED
Tx1	RF TRANSMISSION
Tx2	NOT USED

Figure 6. ADRV9001 in 1T1R FDD with DPD Type Application

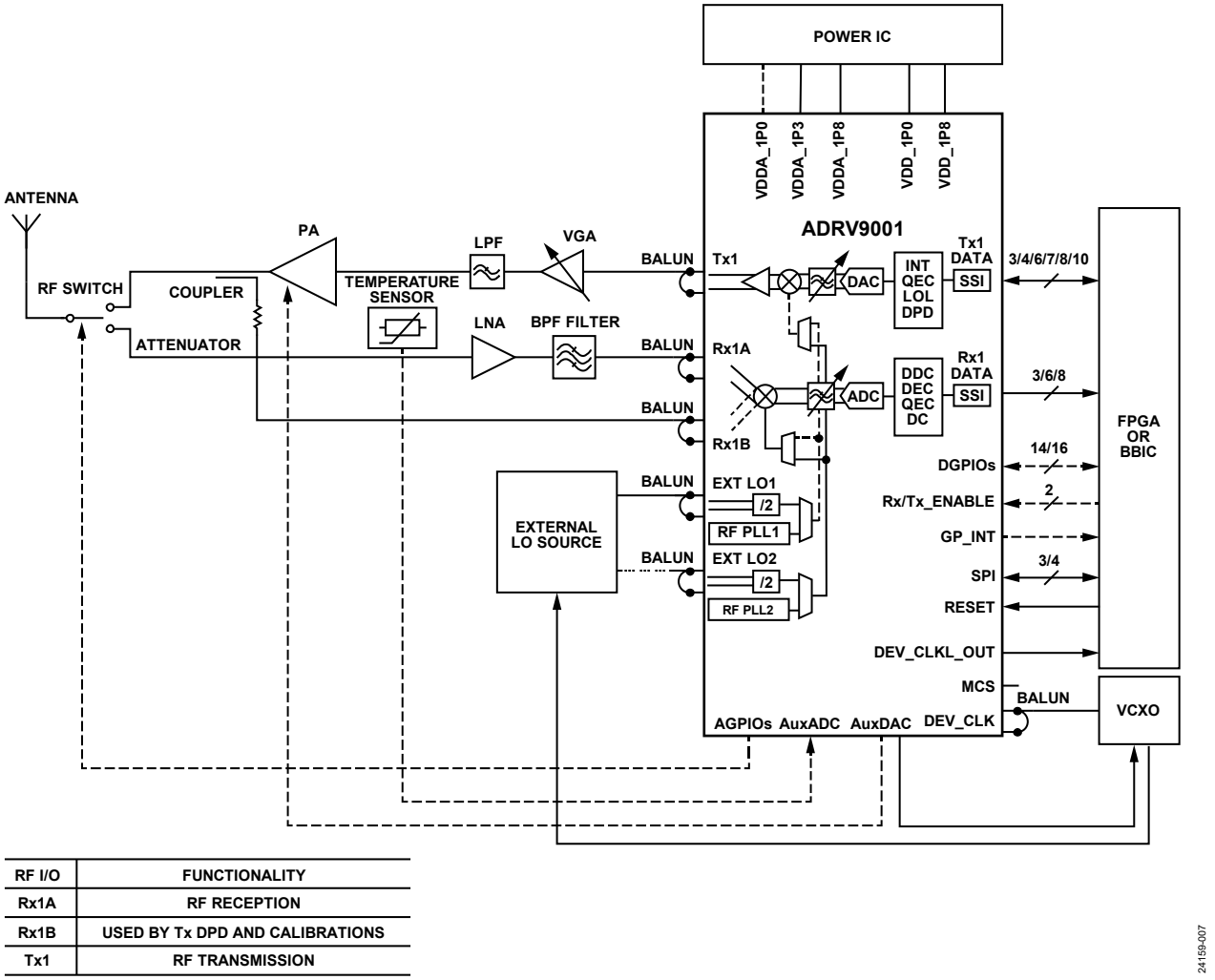
1T1R FDD with DPD Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in FDD type applications that requires DPD. Internal DPD block can be used to linearize external power amplifier and improve overall system efficiency. For systems that demand superior LO phase noise performance, ADRV9001 allows user to apply eternal RF LO. ADRV9001 internal AGC can be used to autonomously monitor and set the appropriate gain level for Rx signal chain. ADRV9001 can control external LNA using its analog GPIOs as well as provide power amplifier bias voltage by utilizing AuxDAC outputs.

Table 4. Constrains and Limitations in 1T1R FDD with DPD Type Application

Functionality	Constrains and Limitations
LO Generation	In 1T1R FDD+DPD type applications, ADRV9001 can use its internal LO to generate RF LO1 for uplink and RF LO2 for downlink. For applications with stringent RF LO requirements, the user can use external LO inputs. External LO1 operating at 2× RF LO can be used for uplink and separate external LO2 operating at 2× RF LO for downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality can be used in the 1T1R FDD mode with second Tx being disabled. Maximum channel bandwidth that DPD can support is limited by ADRV9001 RF bandwidth divided by 3 or by 5. The DPD operation can be performed by ADRV9001 or Rx data can be sent to baseband processor via Rx data port serving as observation Rx. Rx path used during DPD operation to perform Tx observation is also used by the Tx tracking calibrations. In case of external DPD, user must ensure that access to the Rx path during Tx slots is time-shared between external DPD operation and internal Tx calibrations.
Calibrations	During Rx initialization sequence, the user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones that are present at Rx input. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence, user needs to ensure that Power Amplifier is power down to avoid unwanted emission of Tx calibration tones at the antenna. ADRV9001 needs to access Rx datapath during Tx time slots for Tx tracking calibration to operate. If user use Tx observation path with DPD functionality performed by baseband processor, then access to the Rx datapath during Tx slots must be time-shared between DPD operation and Tx calibrations.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components (for example, RF Switch, LNA) or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias or control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide a reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN TETRA TYPE PORTABLE RADIO APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RF RECEPTION
Rx1B	USED BY Tx DPD AND CALIBRATIONS
Tx1	RF TRANSMISSION

Figure 7. ADRV9001 in TETRA Type Portable Radio Application

TETRA Type Portable Radio Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in TETRA type applications. Internal DPD block can be used to linearize external power amplifier and improve overall system efficiency. For systems that demand superior LO phase noise performance, ADRV9001 allows user to apply eternal RF LO. ADRV9001 internal AGC can be used to autonomously monitor and set appropriate gain level for Rx signal chain. For time critical TDD type applications control of the ADRV9001 TRx can be done by toggling control lines. ADRV9001 can control external Rx/Tx switch using its analog GPIOs as well as provide power amplifier bias voltage by utilizing AuxDAC outputs.

Table 5. Constrains and Limitations in TETRA Type Portable Radio Application

Functionality	Constrains and Limitations
LO Generation	In Portable Radio, TETRA type application, ADRV9001 can use its internal LO to generate RF LO1 for both uplink and downlink. For applications with stringent RF LO requirements, the user can use external LO inputs. External LO1 operating at 2× RF LO can be used for both uplink and downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates a square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.

Functionality	Constrains and Limitations
DPD	The DPD functionality can be used in the 1T1R TDD mode. Maximum channel bandwidth that DPD can support is limited by ADRV9001 RF bandwidth divided by 3 or by 5. The DPD operation can be performed by ADRV9001 or Rx data can be sent to baseband processor via Rx data port during Tx operation. Rx path used during DPD operation to perform Tx observation is also used by the Tx tracking calibrations. In case of external DPD, user must ensure that access to the Rx path during Tx slots is time-shared between external DPD operation and Tx calls.
Calibrations	During Rx initialization sequence, the user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at the LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence, user needs to ensure that Power Amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. ADRV9001 needs to access Rx datapath during Tx time slots for Tx tracking calibration to operate. If user use Tx observation path with DPD functionality performed by baseband processor, then access to the Rx datapath during Tx slots must be time-shared between DPD operation and Tx calibrations.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components (for example, RF Switch) or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN DMR TYPE PORTABLE RADIO APPLICATION

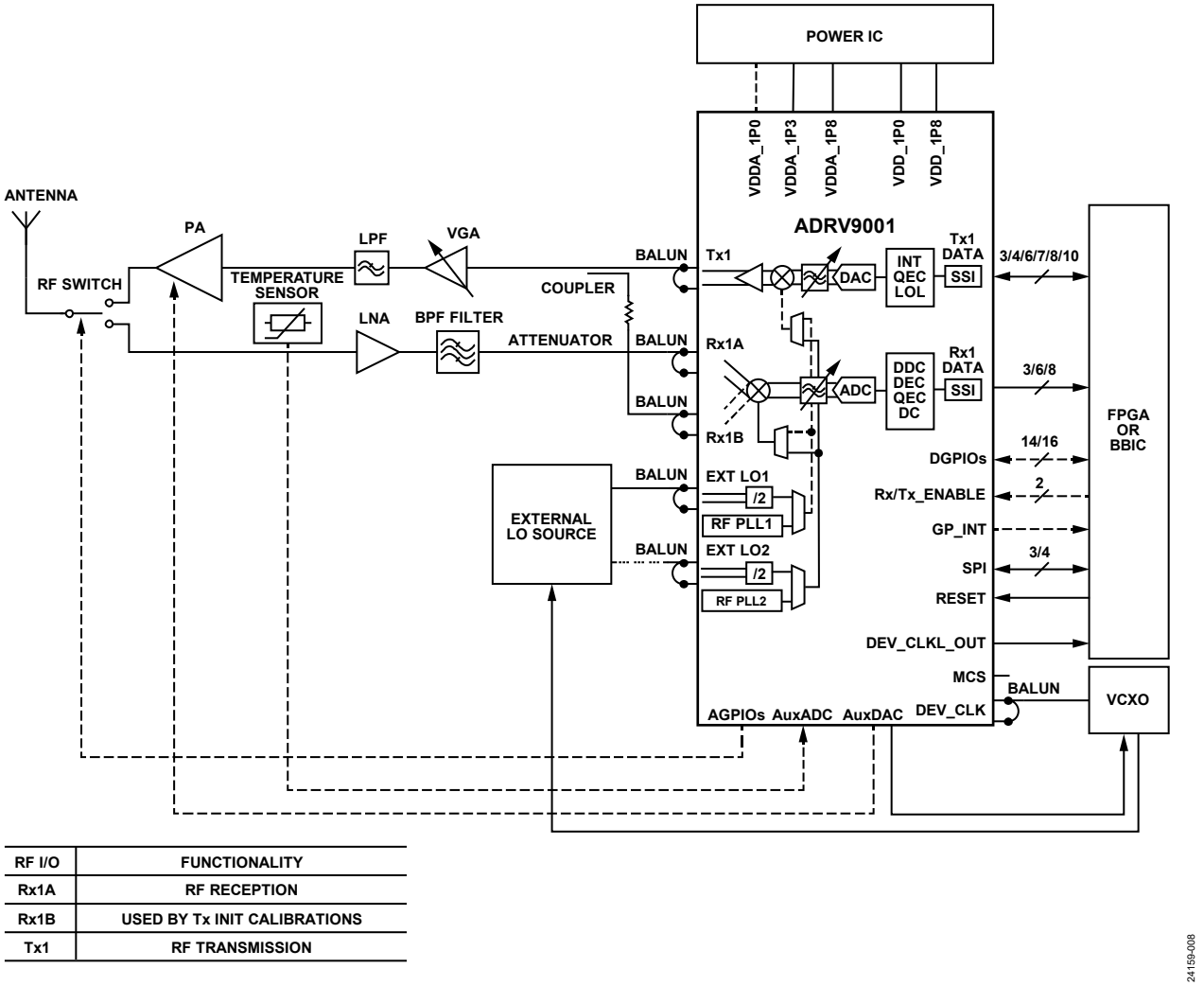


Figure 8. ADRV9001 in DMR Type Portable Radio Application

DMR Type Portable Radio Overview

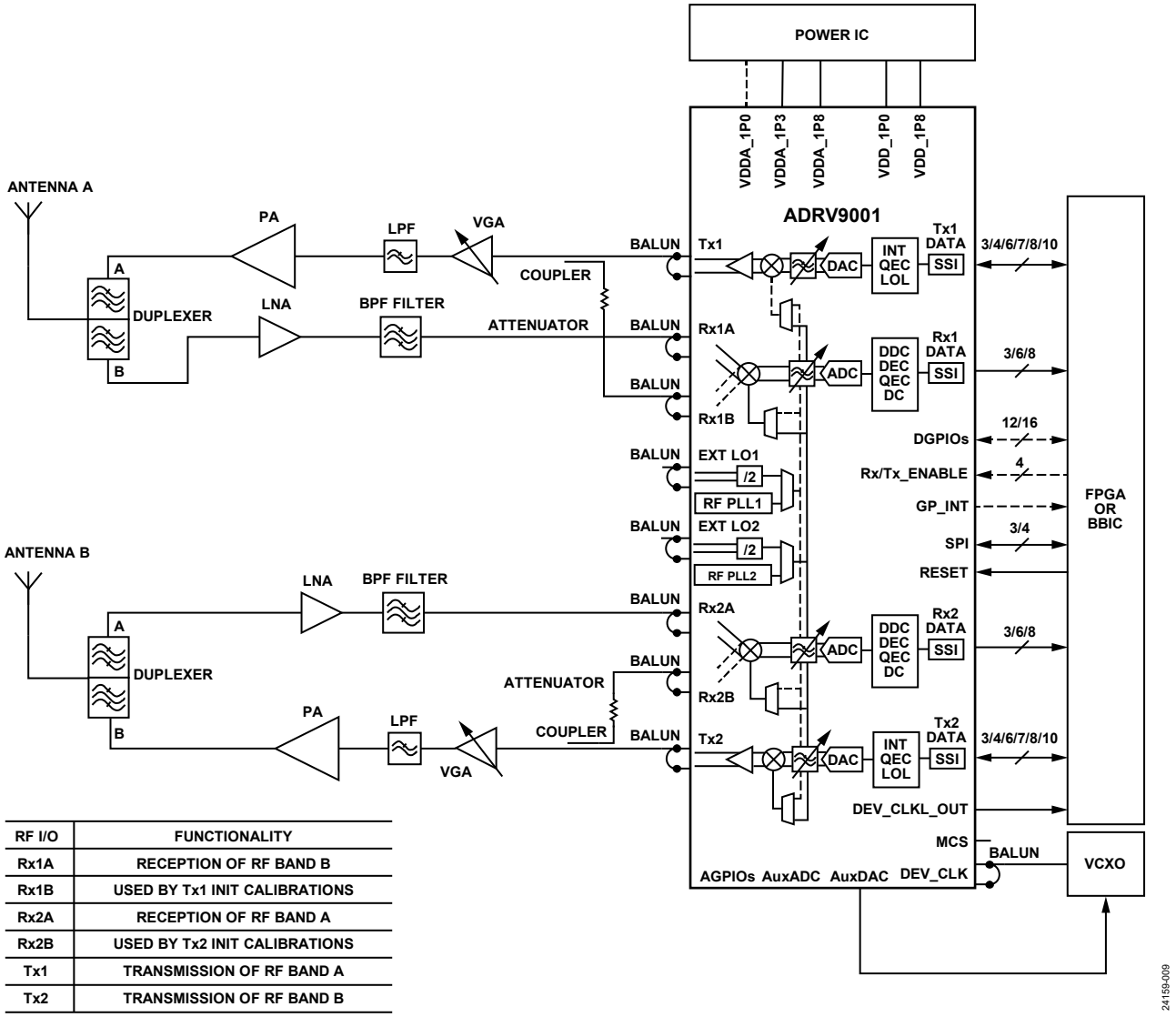
With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in DMR type applications. For systems that demand superior LO phase noise performance, ADRV9001 allows user to apply eternal RF LO. ADRV9001 internal AGC can be used to autonomously monitor and set the appropriate gain level for the Rx signal chain. For time critical TDD type applications control of the ADRV9001 TRx can be done by toggling control lines. ADRV9001 can control external Rx/Tx switch using its analog GPIOs as well as provide power amplifier bias voltage by utilizing AuxDAC outputs.

Table 6. Constrains and Limitations in DMR type Portable Radio Application

Functionality	Constrains and Limitations
Rx Signal Path	User have to ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, RxB input is used only during initialization calibrations. The LNA connected to the Rx1A should be powered down during Tx slots to ensure proper operation of the Tx calibration path (connected to the Rx1B). The user must ensure that appropriate attenuation is present in the line to prevent Rx being overloaded by Tx signal.
LO Generation	In Portable Radio, DMR type application, ADRV9001 can use its internal LO to generate RF LO1 for both uplink and downlink. For applications with stringent RF LO requirements, the user can use external LO inputs. External LO1 operating at 2× RF LO can be used for both uplink and downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates a square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path

Functionality	Constrains and Limitations
	must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
FDPD	The DPD functionality is not available when ADRV9001 operates in 1T1R mode.
Calibrations	During Rx initialization sequence user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence, user needs to ensure that Power Amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. For Tx tracking calibrations to operate, ADRV9001 needs to access Rx datapath during Tx time slots to operate.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components (for example, RF Switch) or read back digital logic levels from external components.
DGPIOs	For DMR type applications, ADRV9001 supports RF Monitor mode of operation. DGPIO pins are used to: sent wake up signal to baseband processor, allow baseband processor to move ADRV9001 into Monitor mode using hardware pins (instead API command). Digital GPIOs can also be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 2 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN FDD TYPE REPEATER APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RECEPTION OF RF BAND B
Rx1B	USED BY Tx1 INIT CALIBRATIONS
Rx2A	RECEPTION OF RF BAND A
Rx2B	USED BY Tx2 INIT CALIBRATIONS
Tx1	TRANSMISSION OF RF BAND A
Tx2	TRANSMISSION OF RF BAND B

Figure 9. ADRV9001 in FDD Type Repeater Application with Baseband Processor Analyzing Traffic Data

ADRV9001 in FDD Type Repeater Application with Baseband Processor

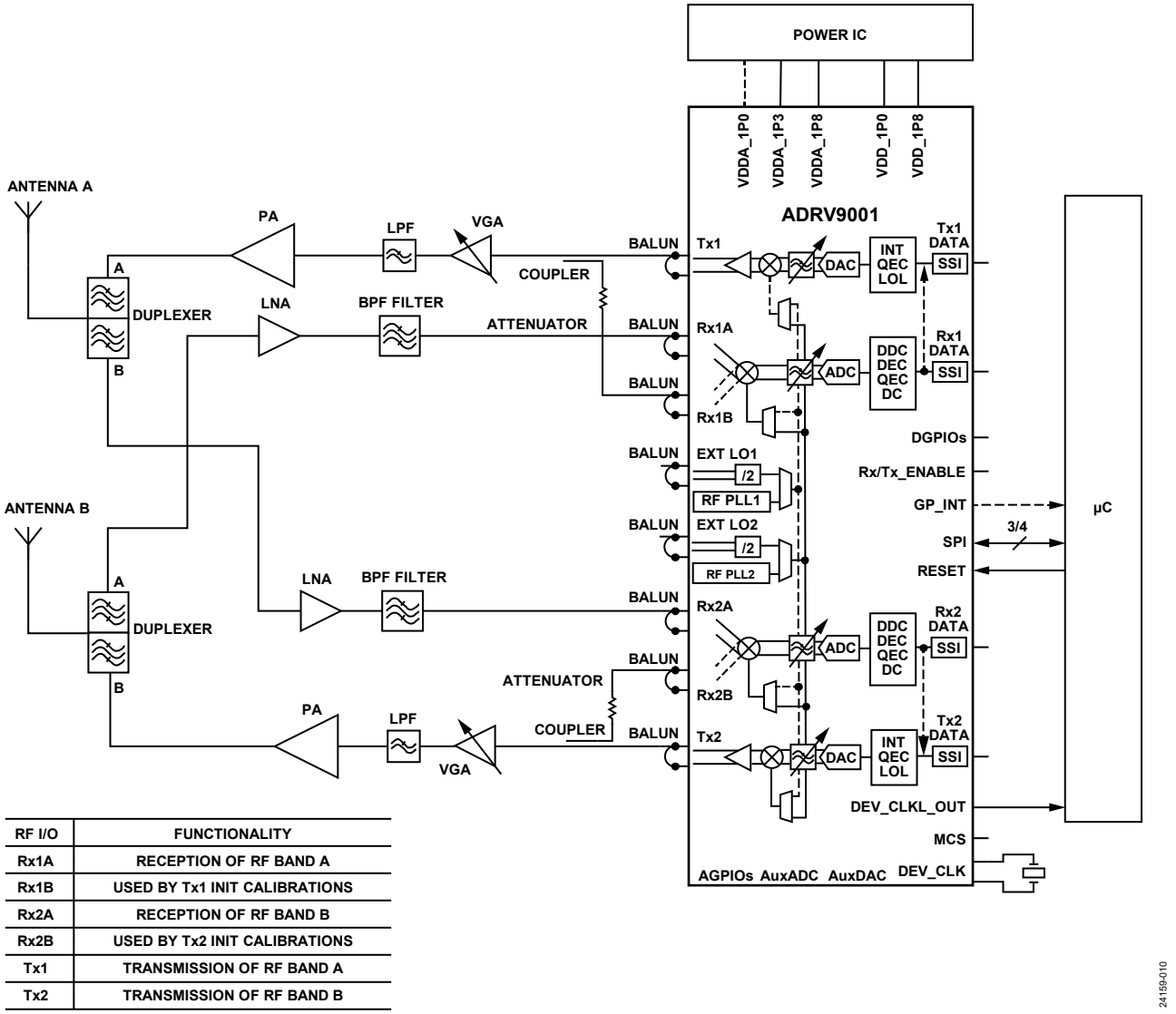
With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in repeater or frequency translator type applications. ADRV9001 internal AGC can be used to autonomously monitor and set the appropriate gain level for Rx signal chains. For none time critical FDD type applications control of the ADRV9001 TRx can be done thru API commands that use SPI interface.

Table 7. Constrains and Limitations in FDD Type Repeater Application with Baseband Processor Analyzing Traffic Data

Functionality	Constrains and Limitations
Rx Signal Path	The user must ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, RxB inputs are used only during initialization calibrations. The user must ensure that appropriate attenuation is present in line to prevent Rx being overloaded by Tx signal.
Tx Signal Path	The user must ensure that appropriate level of isolation between Tx1 and Tx2 as well as Rx to Tx is provided at the system level.
LO Generation	In FDD type Repeater application, ADRV9001 can use its internal LO to generate RF LO1 for uplink (example: Tx1 and Rx1) and RF LO2 for downlink (example: Tx2 and Rx2). It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2× RF LO can be used for uplink and External LO2 operating at 2× RF LO can be used for downlink.

Functionality	Constrains and Limitations
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates a square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality is not available when ADRV9001 operates in 2R2T FDD mode.
Calibrations	During Rx initialization sequence user needs to ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence the user must ensure that the power amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. No Tx tracking calibrations are available when ADRV9001 operates in 2R2T FDD mode.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN A FDD TYPE REPEATER APPLICATION USING INTERNAL LOOPBACKS



RF I/O	FUNCTIONALITY
Rx1A	RECEPTION OF RF BAND A
Rx1B	USED BY Tx1 INIT CALIBRATIONS
Rx2A	RECEPTION OF RF BAND B
Rx2B	USED BY Tx2 INIT CALIBRATIONS
Tx1	TRANSMISSION OF RF BAND A
Tx2	TRANSMISSION OF RF BAND B

Figure 10. ADRV9001 in FDD Type Repeater Application Without Baseband Processor Analyzing Traffic Data

FDD Type Repeater Without Baseband Processor Overview

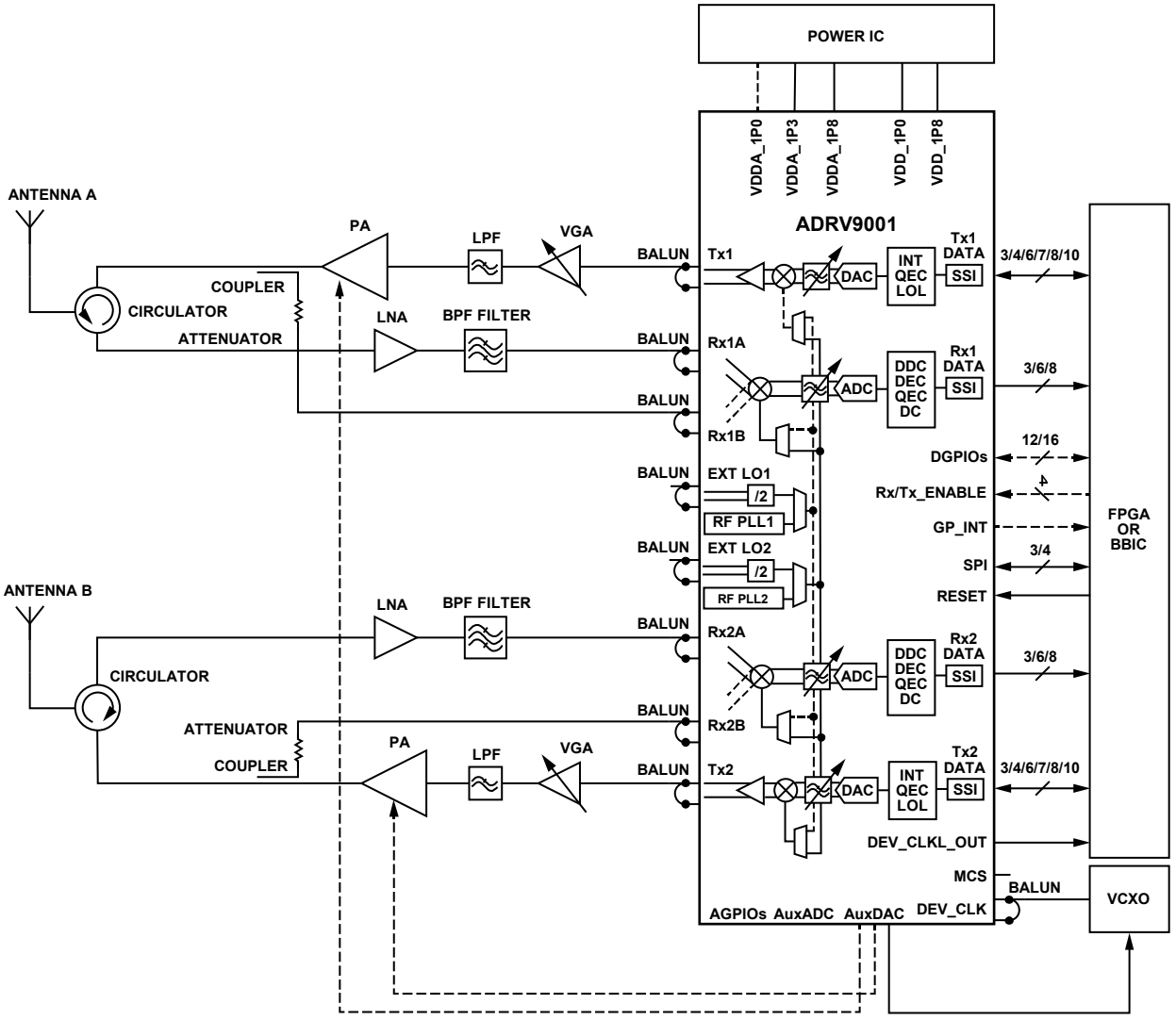
With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-RF signal chain that can serve as repeater or frequency translator. ADRV9001 internal AGC can be used to autonomously monitor and set the appropriate gain level for Rx signal chains. For none time critical FDD type applications control of the ADRV9001 TRx can be done thru API commands that use SPI interface. Support of external crystal enables very compact solution where ADRV9001 provides clock for microprocessor that programs and monitors ADRV9001 operation.

Table 8. Constrains and Limitations in FDD Type Repeater Application Without Baseband Processor Analyzing Traffic Data

Functionality	Constrains and Limitations
Rx Signal Path	The user must ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, RxB inputs are used only during initialization calibrations. The user must ensure that appropriate attenuation is present in line to prevent Rx being overloaded by Tx signal.
Tx Signal Path	The user must ensure that appropriate level of isolation between Tx1 and Tx2 as well as Rx to Tx is provided at the system level.
LO Generation	In FDD type Repeater application, ADRV9001 can use its internal LO to generate RF LO1 for uplink (example: Tx1 and Rx1) and RF LO2 for downlink (example: Tx2 and Rx2). It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2× RF LO can be used for uplink and External LO2 operating at 2× RF LO can be used for downlink.

Functionality	Constrains and Limitations
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality is not available when ADRV9001 operates in 2R2T FDD mode.
Calibrations	During Rx initialization sequence user must ensure that there are no signals at the Rx input (external LNA should be disabled) and appropriate termination should be present at the LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence the user must ensure that the power amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. No Tx tracking calibrations are available when ADRV9001 operates in 2R2T FDD mode.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Unused in this application example.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN TDD TYPE REPEATER APPLICATION



RF I/O	FUNCTIONALITY
Rx1A	RECEPTION FROM ANTENNA A
Rx1B	USED BY Tx1 DPD AND CALIBRATIONS
Rx2A	RECEPTION FROM ANTENNA B
Rx2B	USED BY Tx2 DPD AND CALIBRATIONS
Tx1	TRANSMISSION ON ANTENNA A
Tx2	TRANSMISSION ON ANTENNA B

Figure 11. ADRV9001 in TDD Type Repeater Application with Baseband Processor Analyzing Traffic Data

TDD Type Repeater Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end in TDD type repeater or frequency translator applications. In TDD type applications, internal DPD block can be used to linearize external power amplifier and improve overall system efficiency. ADRV9001 internal AGC can be used to autonomously monitor and set appropriate gain level for Rx signal chains. FPGA or baseband processor is responsible for appropriate time alignment of Rx and Tx time slots. Control of the ADRV9001 Rx and Tx signal chains can be done by toggling control lines. ADRV9001 can provide power amplifier bias voltage by utilizing AuxDAC outputs.

Table 9. Constrains and Limitations in TDD Type Repeater Application with Baseband Processor Analyzing Traffic Data

Functionality	Constrains and Limitations
LO Generation	In TDD type Repeater application, ADRV9001 can use its internal LO to generate RF LO1 for both uplink and downlink. It is also possible to use external LO inputs in this mode of operation. External LO1 operating at 2× RF LO can be used for both uplink and downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality can be used in the 2R2T TDD mode. Maximum channel bandwidth that DPD can support is limited by ADRV9001 RF bandwidth divided by 3 or by 5. The DPD operation can be performed by ADRV9001 or ORx data can be sent to baseband processor via Rx data port during Tx operation. Rx path used during DPD operation to perform Tx observation is also used by the Tx tracking calibrations. In case of external DPD, user must ensure that access to the Rx path during Tx slots is time-shared between DPD operation and Tx calibrations.
Calibrations	During Rx initialization sequence user must ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence the user must ensure that the power amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. ADRV9001 must access Rx datapath during Tx time slots for Tx tracking calibration to operate. If user use Tx observation path with DPD functionality performed by baseband processor then access to the Rx datapath during Tx slots must be time-shared between DPD operation and Tx calibrations.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to control the VCXO responsible for generating the ADRV9001 device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	If there is no need for multichip synchronization, the ADRV9001 can be initialized using API functions only.

ADRV9001 IN RADAR TYPE APPLICATION

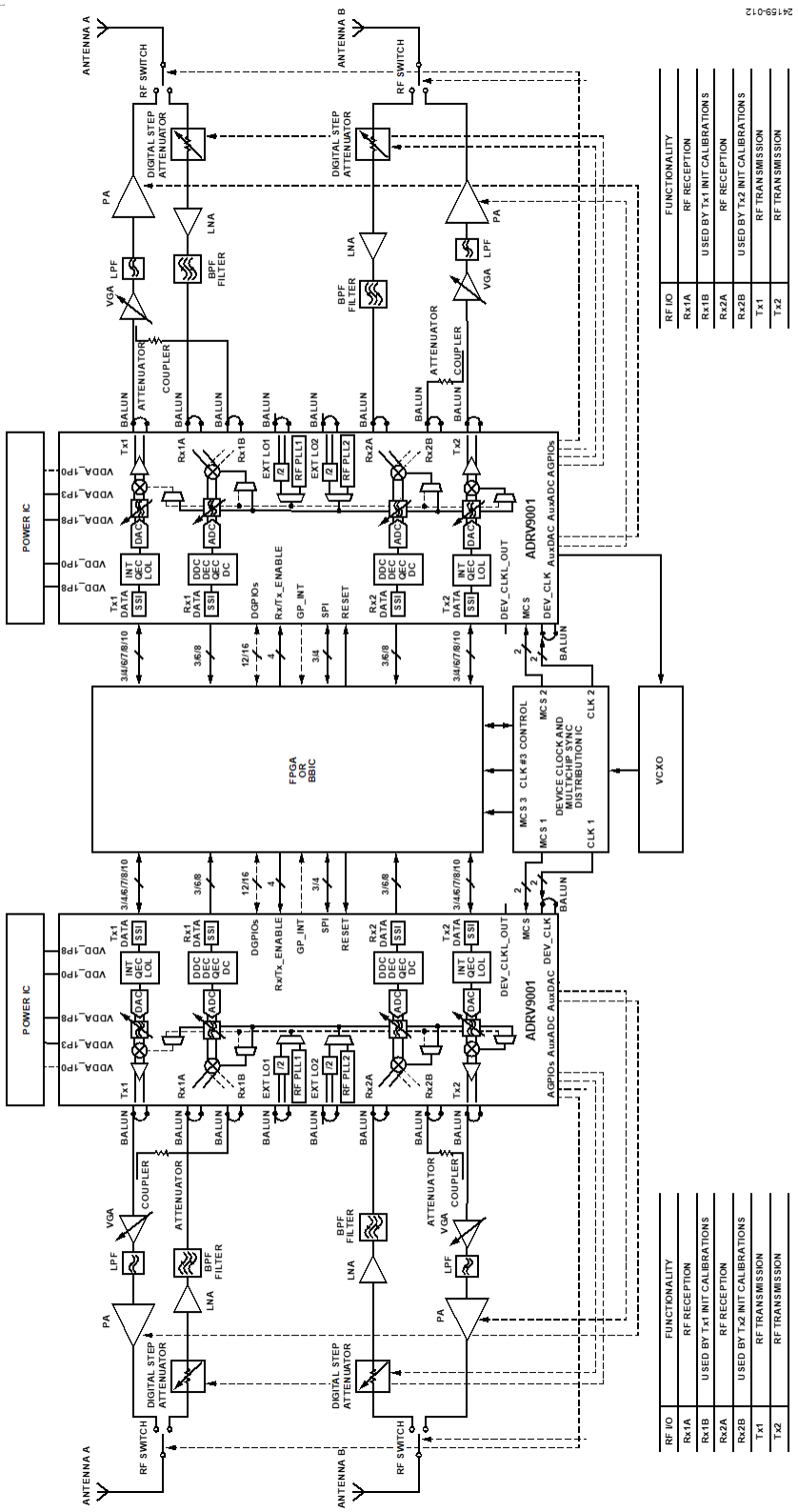


Figure 12. ADRV9001 in Radar Type Application

Radar Type Application Overview

With a minimum number of external components, the ADRV9001 transceiver can be used to build complete RF-to-bits signal chain that can serve as RF front end building block in Radar type applications. ADRV9001 internal AGC can be used to autonomously monitor and set the appropriate gain level for Rx signal chains. Internal AGC can use analog GPIO interface to control external DSA in the Rx signal chains. For time critical TDD type applications control of the ADRV9001 TRx can be done by toggling control lines. ADRV9001 can control external Rx/Tx switch using its analog GPIOs as well as provide power amplifier bias voltage by utilizing AuxDAC outputs. Multi Chip Sync signal together with DEV_CLK can be used to synchronize multiple ADRV9001 in the end system.

Table 10. Constrains and Limitations in Radar Type Application

Functionality	Constrains and Limitations
Rx Signal Path	The user must ensure that appropriate level of isolation between Rx1 and Rx2 as well as Rx to Tx is provided at the system level. In the previously described example, RxB input is used during Tx observation. The LNA connected to the Rx1A should be powered down during Tx slots to ensure proper operation of the Tx observation path (connected to the Rx1B). The user must ensure that appropriate attenuation is present in line to prevent Rx input being overloaded by Tx signal.
LO Generation	In Radar type application, ADRV9001 can use its internal LO to generate RF LO1 for both uplink and downlink. For applications with stringent RF LO requirements, user can use external LO inputs. External LO1 operating at 2× RF LO can be used for both uplink and downlink.
RF Front End	For LO generation, the ADRV9001 uses internal VCO that generates square wave type signal. A square wave LO would produce harmonics. For example: depending of RF matching used on the RF ports user 2nd LO harmonic can be as high as −50 dBc and 3rd harmonic can be as high as −9 dBc. Therefore, the RF filtering on the Rx and Tx path must ensure that signals at the LO harmonic frequencies (up to 9th in some cases) are not affecting overall system performance.
DPD	The DPD functionality can be used in the 2R2T TDD mode. Maximum channel bandwidth that DPD can support is limited by ADRV9001 RF bandwidth divided by 3 or by 5. The DPD operation can be performed by ADRV9001 or ORx data can be sent to baseband processor via Rx data port during Tx operation. Rx path used during DPD operation to perform Tx observation is also used by the Tx tracking calibrations. In case of external DPD, user must ensure that access to the Rx path during Tx slots is time-shared between DPD operation and Tx calibrations.
Calibrations	During Rx initialization sequence user must ensure that there are no signals present at the Rx input (external LNA should be disabled) and appropriate termination should be present at LNA output to avoid reflections of Rx calibration tones. The maximum input signal amplitude must not exceed −82 dBm/MHz for wideband modes, TBD dBm/MHz for narrowband modes. During Tx initialization sequence the user must ensure that the power amplifier is powered down to avoid unwanted emission of Tx calibration tones at the antenna. ADRV9001 must access Rx datapath during Tx time slots for Tx tracking calibration to operate. If user use DPD in its system then access to Rx datapath during Tx slots must be time-shared between DPD operation and Tx calibrations.
AGPIOs	Analog GPIOs (operating at 1.8 V level) can be used as read or write digital levels of in the end user system. AGPIOs can be used to control states of external components (for example, RF Switch) or read back digital logic levels from external components.
DGPIOs	Digital GPIOs can be used to perform real-time monitoring of states of internal ADRV9001 blocks. Digital GPIOs operating as inputs can allow user to control Rx gain, Tx attenuation, AGC operation and other elements of ADRV9001 TRx. Depending on the ADRV9001 operation up to 4 GPIOs may be used by data port interface.
AuxADC	AuxADC can be used to monitor analog voltage (for example, temperature sensor). Maximum AuxADC input voltage must not exceed 0.9 V.
AuxDAC	AuxDAC can be used to: control VCXO responsible for generating Device clock, generate pre-configured ramp up/down signal that can be used to control power amplifier bias, control any circuitry that requires analog control voltage up to 1.8 V.
DEV_CLK_OUT	ADRV9001 provides divided down version of DEV_CLK reference clock input signal on the DEV_CLK_OUT output. This output is intended to provide reference clock signal to the digital components in the overall system. This output can be configured to be active after power up and before ADRV9001 configuration stage.
Multichip Sync	ADRV9001 allows the user to synchronize multiple transceivers used in single system. ADRV9001 provides the capability to accept an external reference clock and synchronize operation with other devices using simple control logic. Logical pulses applied at MCS input align each device's data clock with a common reference. Relationship of MCS pulse to the DEV_CLK edge at the ADRV9001 pins must be preserved. For correct operation, it is critical to match the length of PCB traces that carry DEV_CLK and MCS signals to each ADRV9001 device.

SOFTWARE SYSTEM ARCHITECTURE DESCRIPTION

This section provides information about the device driver Application Programming Interface (API) software developed by ADI for the ADRV9001 transceivers, as well as outlines the overall architecture, folder structure, and methods for using API software on the customer platform.

Note:

- This document does not explain the API library functions. Detailed information regarding the API functions is in the doxygen document included with the SDK (ADRV9001_API.chm) located at /pkg/production/.
- ADRV9001_API.chm is in compressed HTML format. For security reasons, .chm files can only be opened from a local drive. If you attempt to open from a network drive, the file may look empty.
- The ADRV9001 is baseline device for the product family; therefore, all API and evaluation systems use this product number to delineate the product.

For users who are new to this product family, we provide a simple flow chart which can guide you through evaluation, testing and development of our platform and your own system:

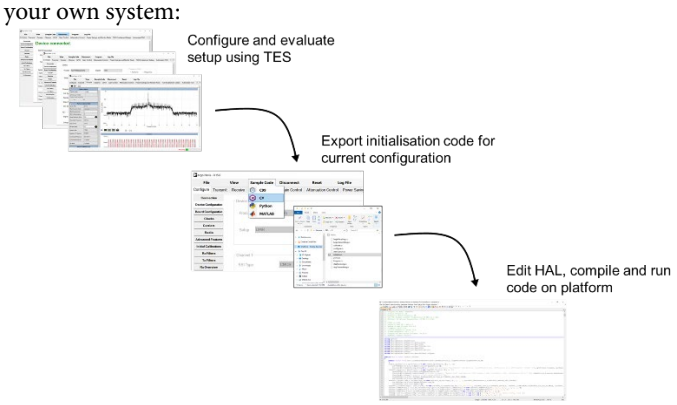


Figure 13: Development Flow Chart

SOFTWARE ARCHITECTURE

Figure 14. illustrates the software architecture for the ADRV9001 evaluation platform that Analog Devices provides. This chapter will take a high-level look at the changes that need to be made to this architecture once the baseline setup for the evaluation board has been explained.

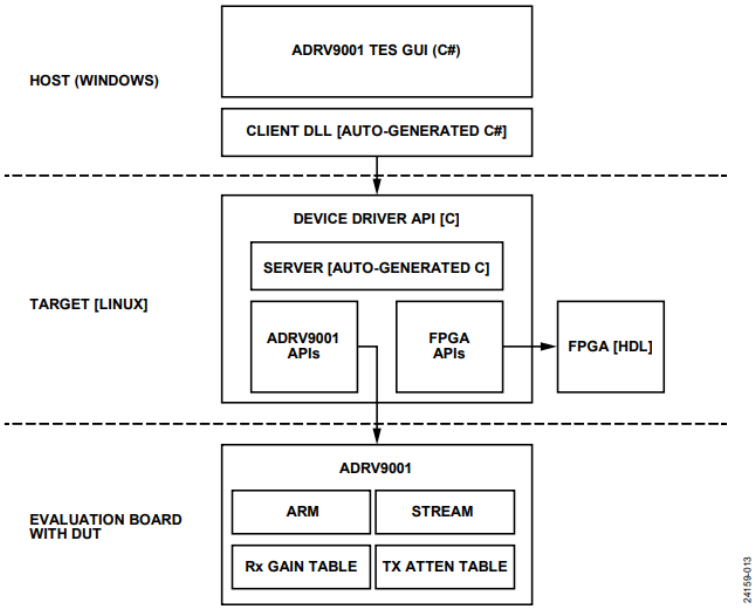


Figure 14: ADRV9001 API Software Architecture (ADI Evaluation Platform)

Layering the system as shown in Figure 14 allows for great flexibility regarding system platform. Any customer beginning their development using the TES software package will be able to swap out elements in the Target layer of this system architecture with only minor edits to the codebase.

FOLDER STRUCTURE

Source files are provided by ADI in the folder structure shown in Figure 15. Each subfolder is explained in the following sections. ADI understands that the developer may desire to use a different folder structure. Whereas Analog Devices provides ADRV9001 API source code releases in the folder structure shown below, the developer may organize the ADRV9001 API into a custom folder organization, if required. This operation, however, does not permit the developer the right to modify the content of the ADRV9001 API source code, with the exception of the customer HAL placeholder files, which will be detailed later in this chapter and the SOFTWARE INTEGRATION chapter.

Figure 15. API Folder Structure

/c_src/common

Common code shared between all the devices, this contains error handling facilities, logging facilities, and HAL access facilities. Note: this is not a folder customers should edit when implementing their own HAL, the files under common\ are designed to work in tandem with any HAL provided to the codebase and as such do not need to be modified. Nor do any new files need to be added here.

/c_src/devices

The device folder includes the main API code for the ADRV9001 transceiver as well as auxiliary devices APIs used for the demo of ADRV9001. The /adrv9001 folder contains the high-level function prototypes, data types, macros, and source code used to build the final user software system. The user is strictly forbidden from modifying the files contained in the /adrv9001 or other devices in this section, Software support is not provided when these files have been modified. Analog Devices maintains this code. The only exception is that user may modify #define macros in adi_adrv9001_user.h, such as modifying polling timeouts and interval settings for various functions.

/c_src/platforms

The /platforms folder provides the means for a developer to insert custom platform hardware driver code for system integration with the ADRV9001 API. A description regarding the HAL interface is contained later in this document. The adi_platform.c/.h files contain function pointers and the required prototypes necessary for the ADRV9001 API to work correctly. Modification of the function prototypes in adi_platform.c is forbidden. The developer is responsible for implementing the code for each function to ensure the correct hardware drivers are called for the user's platform hardware. In the example code provided by ADI in the customer\ folder, there are placeholder functions left empty for the customer to fill in their platform-specific code.

/c_src/third_party

This section contains third party APIs used to help the FPGA control the system. This includes a JSON parser and a FMC FRU info manipulator for example.

CUSTOMISING THE SYSTEM ARCHITECTURE AND FILE STRUCTURE

Given the system architecture and file set are uniquely suited the evaluation platform, these will likely need to change when a customer progresses to developing their own unique solution. This paragraph is designed to give customers at this stage of development a good starting point to build from, as well as build up to more in-depth details which will be provided in the next chapter (SOFTWARE INTEGRATION). Figure 16 provides an illustration of the steps needed to begin development on a custom platform. The goal here is to migrate from the evaluation platform to a custom, bespoke platform. Doing so requires that the evaluation platform APIs and HDL are taken out of the TARGET compilation and replaced with the APIs specific to the bespoke platform. To accommodate this migration, the TES must be used in a slightly different way. Rather than connecting directly to the TARGET platform via a Client-Server setup, customers must instead use the TES package to generate C code, make minor edits to the generated files and then deploy the files to the target platform.

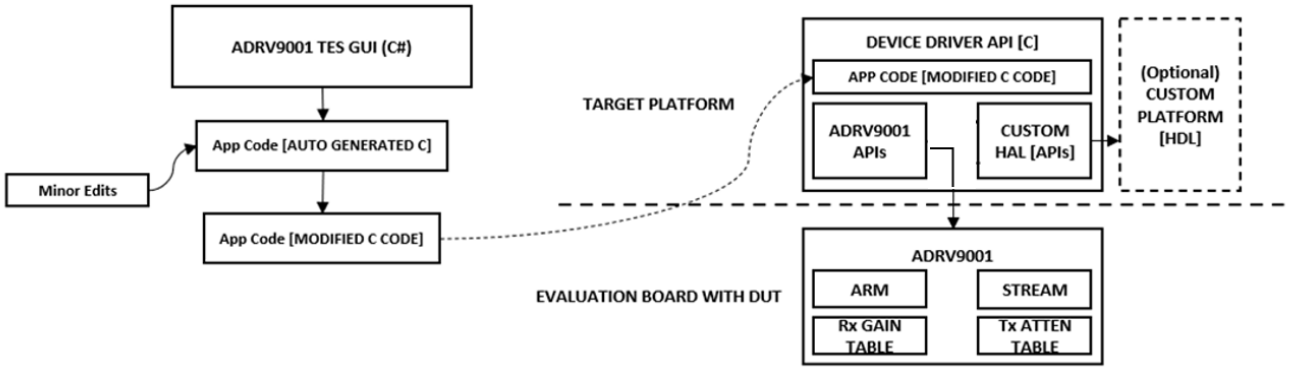


Figure 16: Modified ADRV9001 API Software Architecture

These "Minor Edits" listed in Figure 16 consist mostly of removing function calls targeted at hardware components that the customer has no intention of using. For example, by default the generated code will seek to initialize the Evaluation Platform by making calls to "adi_fpga9001_" functions. Given that customers will be migrating away from this platform, it is necessary to remove those function calls. In a similar fashion, any initialization functions needed for the customer platform must be inserted into the generated C code where they are necessary.

The majority of the “Minor Edits” occur in the “main.c” and “initialize.c” files of the generated codebase. For example, near the beginning of the main(…) function, the linux_uio_initialize(…) function is called. This function is defined in linux_uio_init.c, found under platforms\linux_uio\. Inspecting this function, the following code snippet is found early in definition:

```
if (NULL != fpga9001)
{
   fpga9001->common.devHalInfo = linux_uio_fpga9001_open();
   if (NULL != adrv9001)
   {
      ((adi_adrv9001_hal_linux_uio_Cfg_t *)adrv9001->common.devHalInfo)->fpga9001 = fpga9001;
   }
   if (NULL == fpga9001->common.devHalInfo)
   {
      return -1;
   }
}
```

In more simple English, this code snippet uses the linux_uio FPGA codebase to define an FPGA structure if the fpga pointer argument was not NULL. The application uses this structure to interact with the DMA, the SSI ports and other elements of functionality. Figure 17 highlights the areas of the file structure that customers will focus on during their platform development phase. The edits required for custom platform development involve replacing functions such as linux_uio_initialize(…) with custom code that performs the same tasks relative to the platform device in these highlighted folders.

The adi_platform.c file is where the Hardware Abstraction Layer (HAL) is chosen. In the case of the default HAL provided by Analog Devices, it is also implemented in the adi_platform.c file, however no edits should be made to the default HAL stored under linux_uio\. Customers have been given placeholder files in the customer\ folder. Here all necessary functions have been declared and left empty for customers to fill with their platform-specific code.

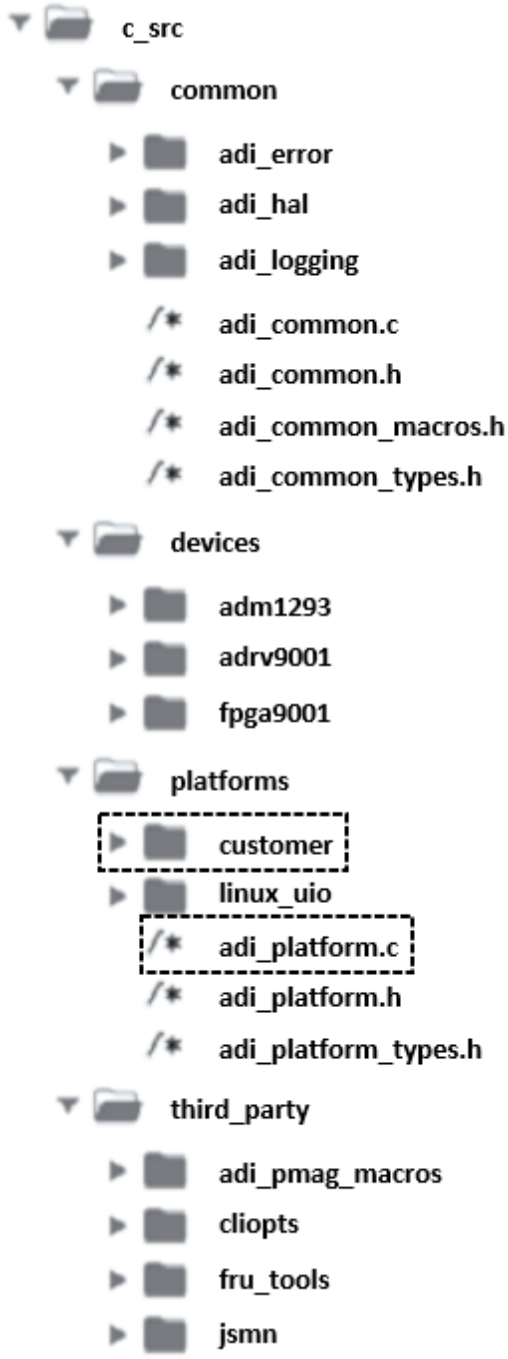


Figure 17. API Folder Structure with Customer Interaction Points Highlighted

More details are provided on the customer\ folder in the Software Integration chapter, which goes into more specifics on the HAL. At this point in development it is recommended to read the files under the highlighted directories in Figure 17 (primarily the customer\ folder) before proceeding to edit them.

SOFTWARE INTEGRATION

The ADRV9001 API package was developed using the ZC706 Evaluation platform. This section describes how to use the provided ADRV9001 API in a custom hardware/software environment. This is readily accomplished because the API was developed abiding by C99 constructs while maintaining Linux system call transparency. The TES package can be used to produce C99 code to replicate a customer's application while maintaining agnostic processor and operating system integration with the ADRV9001 API code.

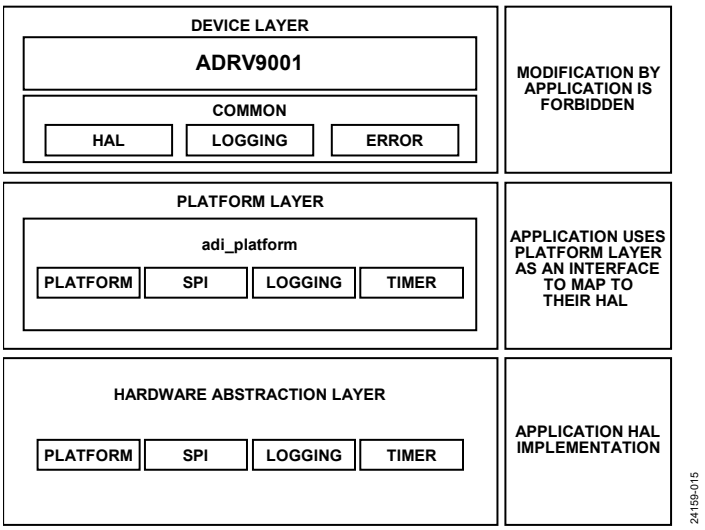


Figure 18. Evaluation System Software Stack

HARDWARE ABSTRACTION LAYER

Users who develop code to target custom hardware platforms use different drivers for the peripherals such as the SPI and GPIO compared to the drivers chosen for the ADI evaluation platform. The hardware abstraction layer (HAL) interface is a set of function pointers that the ADRV9001 API uses when it needs to access the target platform hardware. Each device driver (ADM1293, ADRV9001, FPGA9001) has its own HAL, defined in their respective adi_<device>_hal.h files. This allows a user to select only the components which are desired. The user is responsible for implementing the interface defined in each HAL in order to use the corresponding device driver with their specific platform. The adi_platform.c file maps function pointers to the platform-specific HAL implementations. The implementation of this interface is platform dependent and needs to be implemented by the end user. The function pointers associated with the user HAL layer needs to be set in adi_platform.c.

Below is a code snippet from the beginning of adi_platform.c:

```
#include "adi_platform.h"

#ifndef CUSTOMER_PLATFORM

#include "adi_linux_uio_logging.h"
#include "adi_linux_uio_timer.h"
#include "adi_adm1293_hal_linux_uio.h"
#include "adi_adrv9001_hal_linux_uio.h"
#include "adi_fpga9001_hal_linux_uio.h"

/* Logging interface */
int32_t(*adi_hal_LogWrite)(void *devHalCfg, uint32_t logLevel, const char *comment, va_list args) = linux_uio_LogWrite;

/* Timer interface */
```

```
int32_t(*adi_hal_Wait_us)(void *devHalCfg, uint32_t time_us) = linux_uio_TimerWait_us;
…
```

The file continues from this point to set the functions which handle the I2C reading and writing, the SPI reading and writing, assigning the RESET pin, etc. Taking note of the second line of the file, a customer need only define a CUSTOMER_PLATFORM variable for the pre-processor to use their own platform code. Doing so will change the operation of the adi_platform.c file as follows:

```
#else

#include "adi_common_hal_customer.h"
#include "adi_adm1293_hal_customer.h"
#include "adi_adrv9001_hal_customer.h"
#include "adi_fpga9001_hal_customer.h"

/* Logging interface */
int32_t(*adi_hal_LogWrite)(void *devHalCfg, uint32_t logLevel, const char *comment, va_list args) = customer_LogWrite;

/* Timer interface */
int32_t(*adi_hal_Wait_us)(void *devHalCfg, uint32_t time_us) = customer_TimerWait_us;
…
```

Once done, the adi_platform.c code will automatically switch to using the placeholder customer code under the customer\ folder.

Below is provided a code snippet from adi_adrv9001_hal_customer.c, located under customer\adrv9001\. Here can be seen each function that is required by the HAL to support a customer-specific platform. It is advisable to have read the example HAL implementation provided under the linux_uio\ folder to gain an understanding of the purpose of each function, as well as the acceptable return values. Function names may be modified, if desired, provided the pointer assignments in adi_platform.c are updated accordingly. However, the function parameters may not be modified, as this would cause compilation errors.

```
#include "adi_adrv9001_hal_customer.h"

int32_t customer_adi_adrv9001_hal_open(void *devHalCfg)
{
    /* Customer code goes here */
    return 0;
}

int32_t customer_adi_adrv9001_hal_close(void *devHalCfg)
{
    /* Customer code goes here */
    return 0;
}

int32_t customer_adi_adrv9001_hal_spi_write(void *devHalCfg, const uint8_t txData[], uint32_t numTxBytes)
{
    /* Customer code goes here */
```

```
    return 0;
}

int32_t customer_adi_adrv9001_hal_spi_read(void *devHalCfg, const uint8_t txData[], uint8_t rxData[], uint32_t numTxRxBytes)
{
    /* Customer code goes here */
    return 0;
}

int32_t customer_adi_adrv9001_hal_resetbPin_set(void *devHalCfg, uint8_t pinLevel)
{
    /* Customer code goes here */
    return 0;
}
```

As customers go through their platform development it is recommended that they remain aware of all changes made to the HAL implementation with each release of the SDK. Being familiar with README.md and CHANGELOG.md under the production\ folder is the best way to keep a customer system up to date.

The host can then use the compile_on_platform.py script to compile initialization code for the ADRV9001 on the chosen platform though SSH. Run the script using Python3.X and follow the provided instructions.

/c_src/platforms/customer/

In Figure 19 is shown an expansion of the customer/ folder. This folder contains all the necessary files for building a custom platform with support for the hardware present on the ADRV9001 Eval Board.

Figure 19: Customer Folder Expanded

Most of these files can safely be left at their defaults, the majority of function calls defined in this section of the codebase are optional. For example, the adm1293\ folder houses code used to interact with the ADM1293 device (used for power monitoring) via I2C, however this device is only used by TES to provide power measurements. It has no essential function to the operation of the Eval Board. Similarly, common\ houses the code for a Timer(…) and a Log(…) function, both of which are more useful than they are essential. The fpga9001\ folder houses functions designed at accessing and managing memory on your platform, as well as handling the SSI type configuration, DMA and RAM reading. These functions will be necessary if your custom application requires a platform connected to the SSI ports with access to RAM, DMA and register memory, however not every application will need such a complex platform. Simpler applications which do not need access to these elements of functionality can safely ignore this folder.

The only essential folder is the adrv9001\ folder, which houses the code necessary for instantiating and deconstructing the customer HAL, the SPI functionality and the RESET pin control. These are the minimum functions required in order to interact with the ADRV9001 device provided you can power it. In an upcoming section, an example customer HAL file is provided designed to operate the ADRV9001 device using a Raspberry Pi.

SPI Access

As stated in the previous section, programming the API to use custom SPI code in execution is as easy as calling #define CUSTOMER_PLATFORM in adi_platform.c, however some thought should be given to the SPI code provided. A common issue encountered at this stage in development is a lack of response from the device over the SPI lines, sometimes accompanied by an "ARM Boot Up Timed Out" error message. These issues can often be very confusing, as they happen directly after the Hw_Open() runs successfully.

The issue arises due to a discrepancy between the "standard" bitfield control of SPI settings (CPOL, CPHA, etc.) and the approach Analog Devices takes with SPI control of our devices. Where most devices use normal Binary encodings for their SPI control (00, 01, 10, 11), ADI uses Grey Code for SPI control (00, 01, 11, 10).

As for register access, as has been stated previously, we do not provide the register map for this device to customers. All register access is handled via the APIs. This means that as Register maps change and update, customer code remains valid simply by updating the SDK.

Confirming the SPI operation for custom platforms is possible with a few simple tests. Having read the SPI section of this User Guide, we know that the ADRV9001 transceivers use 3-Byte interactions for SPI communication: {Command}{Address}{Data}. Knowing this, we can monitor the SPI interactions using a scope and record the interactions between the default platform (ZC706 or ZCU102) and the ADRV9001 device. We can do the same for any custom platform. Provided the device setup is identical between default platform and custom platform, the SPI interactions should also be identical.

In the absence of any external equipment, there are also APIs one can use to verify the SPI operation. One such API is adi_adrv9001_spi_Verify(…), which performs the following functions:

1. Reads readonly register to check SPI read operation.
2. Writes scratchpad register with 10110110, reads back the data.
3. Writes scratchpad register with 01001001, reads back the data.

APIs such as these are often described in our documentation as "This function is a helper function and does not need to be called directly by the user", however there's nothing a user from calling them early in their platform setup regardless.

Raspberry Pi HAL

Below we provide some rudimentary examples of how a customer might choose to fill the above listed functions for a Raspberry Pi platform as an example. The Broadcom SPI library for the Raspberry Pi is used to handle all configurations and interactions with the ADRV9001 product. Given that the Raspberry Pi's connectivity is limited to the GPIO pin headers, individual pins are used for the Chip Enable signal and the RESET signal. This is done to allow for fine control on the interaction timings.

In the _hal_open(…) function, the Raspberry Pi's SPI interface is initialized and configured to match the desired behavior and the Chip Enable pin is configured as an output. The _hal_close(…) function, by contrast, simply ends all SPI activity.

Next are the _hal_spi_write(…) and _hal_spi_read(…) functions. In accordance with the workings of the bcm2835 library being used, these functions write data to the ADRV9001 product after driving the Chip Enable line low, In the case of the _hal_spi_read(…) function, data is also accepted from the device on the DO line (MISO on the Raspberry Pi).

Lastly is the _hal_resetPin(…) function. When provided with a value (1 or 0) this function drives the RESET pin high or low.

```
#include "adi_adrv9001_hal_customer.h"
#include "bcm2835.h"  //rpi spi library

#define CE_PIN RPI_BPLUS_GPIO_J8_37
#define RESET_PIN RPI_BPLUS_GPIO_J8_36

int32_t customer_adi_adrv9001_hal_open(void *devHalCfg)
{
   /* Customer code goes here */
   if (!bcm2835_init())
   {
     printf("bcm2835_init failed. Are you running as root??\n");
     return 1;
   }
```

```
  if (!bcm2835_spi_begin())
  {
    printf("bcm2835_spi_begin failed. Are you running as root??\n");
    return 1;
  }

  bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_MSBFIRST);          // The default
  bcm2835_spi_setDataMode(BCM2835_SPI_MODE0);                       // The default
  bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_32768);  // 12kHz
  bcm2835_spi_chipSelect(BCM2835_SPI_CS_NONE);                      //
  bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0, HIGH);         // The default

  // Set the CE pin to be an output
  bcm2835_gpio_fsel(CE_PIN, BCM2835_GPIO_FSEL_OUTP);
  bcm2835_gpio_write(CE_PIN, HIGH);
  delay(1);

  return 0;
}

int32_t customer_adi_adrv9001_hal_close(void *devHalCfg)
{
  /* Customer code goes here */
  bcm2835_spi_end();
  return 0;
}

int32_t customer_adi_adrv9001_hal_spi_write(void *devHalCfg, const uint8_t txData[], uint32_t numTxBytes)
{
  /* Customer code goes here */

  bcm2835_gpio_write(CE_PIN, LOW);
  delay(1);

  bcm2835_spi_begin();
  bcm2835_spi_transfern(txData, numTxBytes);
  bcm2835_spi_end();

  delay(1);
  bcm2835_gpio_write(CE_PIN, HIGH);

  return 0;
}

int32_t customer_adi_adrv9001_hal_spi_read(void *devHalCfg, const uint8_t txData[], uint8_t rxData[], uint32_t numTxRxBytes)
{
  /* Customer code goes here */

  bcm2835_gpio_write(CE_PIN, LOW);
  delay(1);

  bcm2835_spi_begin();
  bcm2835_spi_transfernb(txData, rxData, numTxRxBytes);
  bcm2835_spi_end();
```

```
    delay(1);
    bcm2835_gpio_write(CE_PIN, HIGH);

    return 0;
}

int32_t customer_adi_adrv9001_hal_resetbPin_set(void *devHalCfg, uint8_t pinLevel)
{
    /* Customer code goes here */
    if (pinLevel == 1){
        bcm2835_gpio_write(RESET_PIN, HIGH);
    }else{
        bcm2835_gpio_write(RESET_PIN, LOW);
    }
    return 0;
}
```

In this example of the Raspberry Pi platform the Makefile must also be altered to suit. As this example uses the bcm2835 library, an additional link must be made in the Makefile, as shown here:

-lm -lbcm2835 -lpthread

This link will allow the compiler to access the bcm2835 library, provided it is installed correctly on the Raspberry Pi. Similar measures will need to be taken on a customer platform if additional libraries are needed for control of their platform.

Once the auto-generated code has been verified as operational, and then modified to suit the customer's platform, it is safe to remove all files, folders and function calls (including print statements that are not needed) containing code that does not pertain to the customer's platform. Doing so requires that the customer is **certain** that all calls to the removed functions have been replaced by their bespoke code *or* removed altogether. Once finished, again review the Makefile and remove any unnecessary links and includes from the compilation process.

API Error Handling and Debug

Logging Functions

The API provides a simple logging feature function that may be enabled for debugging purposes. Available logging levels are given by adi_common_LogLevel_e as shown in Table 11.

Table 11. Logging Level

Function Name	Purpose
ADI_LOGLEVEL_TRACE	Log everything in exhaustive detail. Used only for development
ADI_LOGLEVEL_DEBUG	Log diagnostic information
ADI_LOGLEVEL_INFO	Log state changes in the application
ADI_LOGLEVEL_WARN	Log bad, but recoverable events
ADI_LOGLEVEL_ERROR	Log events that cannot be recovered from
ADI_LOGLEVEL_FATAL	Log events that are likely to be fatal
ADI_LOGLEVEL_NONE	Disable all logging

When logging is enabled, the APIs log various messages to the system via the HAL. This feature requires an implementation for the customer_LogWrite function. To enable logging, set #define ADI_COMPILED_LOGLEVEL to one of the defined log levels other than ADI_LOGLEVEL_NONE. Any log initialization (e.g., opening files) must be done by the application before calling any APIs, or messages will not be logged properly. Log levels increase in severity numerically. When a given level is used, messages logged with a log level at least as severe (greater than or equal to) as the set level will be published. Less severe logging calls will be compiled out of the API. By default, ADI_COMPILED_LOGLEVEL is set to ADI_LOGLEVEL_WARN, which means that only messages with severities of ADI_LOGLEVEL_WARN, ADI_LOGLEVEL_ERROR, and ADI_LOGLEVEL_FATAL will be included. Log levels are passed to

customer_LogWrite as it may be desirable to log messages of differing severity in different manners (e.g., insert a message prefix for each level, log errors to standard error, etc.).

Error Handling

Each ADRV9001 API function returns an int32_t value representing a recovery action, with 0 being no action or success. Recovery actions are divided into:

- Warning actions are those that do not have an impact at the time of executing the device API but can cause performance issues or logging problems. The value of this action is positive.
- Error actions are those that cause API not to be able to run and an action is required for API to go back to a good state. The value of this action is negative.

DEVELOPING THE APPLICATION

The user application needs to allocate the init (adi_adrv9001_Init_t) and device (adi_adrv9001_Device_t) structures. Users may want to consider allocating memory from the heap for the adi_adrv9001_Device_t and adi_adrv9001_Init_t as the structures have members expected to be on the order of TBD KB. As part of the SDK, there is a memory_profile.py script designed to profile the storage of application variables. Customers who are concerned about memory usage or limitations may find this script beneficial. There is a section of the README.md under the production\ folder dedicated to "Running the memory profiler".

Note also, the ARM binary and Stream Image are now factored out into separate files, which should help mitigate memory concerns on the part of the customer.

The adi_adrv9001_Init_t structure is used to contain the customer profile initialization settings to configure an ADRV9001 device. This init structure is passed to the ADRV9001 API init functions during the initialization phase. This structure contains the device profile settings, system clock settings, data interface settings, and ADRV9001 specific SPI slave controller settings. The application layer passes a pointer to an instance of the adi_adrv9001_Init_t structure for a particular ADRV9001 device to handle the majority of the device core initialization. After initialization is complete, the adi_adrv9001_Init_t structure may be deallocated if desired.

The adi_adrv9001_Device_t data structure contains information for a particular ADRV9001 device, including devHalInfo, error and caching structures. To support multiple ADRV9001 devices, the Application would need to instantiate multiple adi_adrv9001_Device_t structures to describe each physical ADRV9001 device. Multiple ADRV9001 devices can have their own adi_adrv9001_Init_t or can share a common adi_adrv9001_Init_t if they are to be configurated identically.

devHalInfo

devHalInfo is a structure that allows the user to define and pass any platform hardware settings to the platform HAL layer functions. The common device structure adi_common_Device_t contains devHalInfo. devHalInfo is passed to the platform specific HAL function as a void *devHalCfg. ADRV9001 API functions shall not read or write the devHalInfo but pass it as parameter to all HAL function calls.

The application developer must define devHalInfo per system HAL implementation requirements. The Application developer may implement any structure to pass any hardware configuration information that the hardware requires between the application layer and platform layer. For example, devHalInfo contains SPI chip select information to be used for the physical ADRV9001 device.

Note that the API functions are shared across all instances of physical ADRV9001 devices. The devHalInfo structure defined by the developer identifies which physical ADRV9001 device is targeted (SPI chip select) when a particular ADRV9001 API function is called. The developer may need to store other hardware information unique to a particular ADRV9001 device in this structure such as timer instances or log file information.

Note for ADRV9001 API there is a requirement that only one thread may control and configure a specific device instance at any given time.

devStateInfo

The devStateInfo member is of type adi_adrv9001_Info_t and is a runtime state container for the ADRV9001 API. The application layer must allocate memory for this structure, on the order of TBD KB, but only the ADRV9001 API writes to the structure. The application layer should allocate the devStateInfo structure with all zeroes. The API uses the devStateInfo structure to keep up with the current state of the API (has it been initialized, ARM loaded, etc.), as well as a debug store for any run-time data, such as error codes, error sources, and so forth. It is not intended for the application layer to access the devStateInfo member directly because API functions are provided to access information of the last error.

Private vs. Public API Functions

The API is made up of multiple .c and .h files. Since the API is written in C, there are no language modifiers to identify a function as private or public as commonly used in object-oriented languages. Per the ADI coding standard, public API functions are denoted by the

function name prepended with "adi_adrv9001_" (for example, adi_adrv9001_Rx_Gain_Set()). Private helper functions lack the adi_ prefix and are not intended to be called by the customer application.

Most functions in the ADRV9001 API are prefixed with "adi_adrv9001_" and are for public use. However, many of these functions are never called directly from the application. Utility functions that abstract some common operations, specifically initialization of the ADRV9001, are provided in adi_adrv9001_utilities.c. For this reason, much of the initialization and other helper functions have been separated from the top-level adi_adrv9001.c/ adi_adrv9001.h files to help the developer focus on the functions most commonly used by the Application.

Include Files

For each major function block, there are generally three files: adi_[feature].c, adi_[feature].h and adi_[feature]_types.h. The ADRV9001 API places typedef definitions in files with '_types' suffix such as adi_adrv9001_types.h. These _types.h files are included within their corresponding .h files and do no need to be manually included in the application layer code.

Note that the adi_adrv9001_user.h contain the #defines for API timeouts and SPI read intervals which may be set as needed by the customer platform. The ADRV9001 user files are the only API files that the developer may change.

Restrictions

Analog Devices maintains the code in the /c_src/devices/* folders. Modification of this code by Application developers is forbidden. Direct SPI read/write operation is forbidden when configuring an ADRV9001 or any other ADI devices used for the evaluation of ADRV9001. Developers should only use the high level API functions defined in the public *.h files. Developers should not directly use any SPI read/write functions in the Application for ADRV9001 configuration or control. ADI does not support any application containing SPI writes that are reverse engineered from the original ADRV9001 API.

Delays, Waits, and Sleeps

A subset of ADRV9001 APIs require delays to allow the hardware to complete internal configurations. These ADRV9001 APIs request the system to perform a wait or sleep by calling the HAL interface function adi_common_Wait_us. If the target platform's HAL interface implementation chooses to implement a thread-sleep, it is not permitted for the application to call another API targeting the same ADRV9001 device. The application is required to wait/sleep and the API to complete before continuing with the configuration of the device.

Wait/sleep periods are defined in adi_adrv9001_user.h. The timeout period values are the recommended period required to complete the operation. Modifying these values is not recommended and may impact performance. During this time-out period, the status of ADRV9001 is polled. The frequency of the polling the status during this timeout period may be modified by the user by adjusting the value of the polling interval.

SYSTEM INITIALIZATION AND SHUTDOWN

A graphical user interface (GUI) based transceiver evaluation software (TES) is provided to user to initialize and interact with the ADRV9001 device. Through this TES, user could provide high level system configuration parameters such as signal bandwidth, sample rate and initial gain control settings to initialize the device. The TES uses the user provided parameters to set up an initialization C structure and then makes multiple API calls in a proper order to initialize the device. During the normal operation of the device, the TES allows further user interaction with the device, such as adjusting the transmit/receive gain on the fly. When the operation is completed, the user can safely shut down the device through TES. Figure 20 describes the high level flow of the device operation sequence and the user interaction through TES.

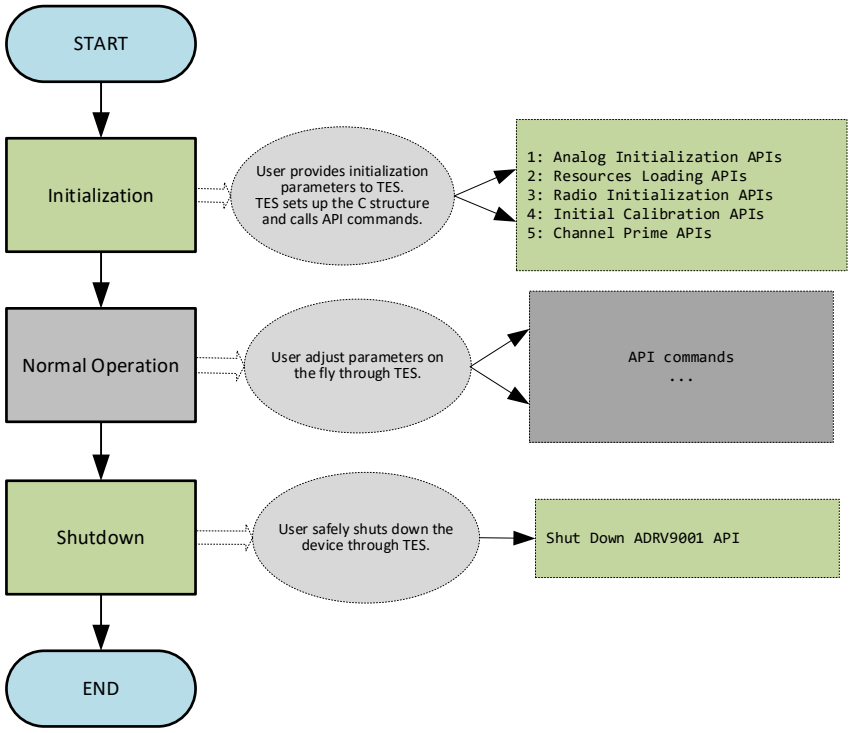


Figure 20. High Level Flowchart of the Device Operation and User Interaction Through TES

As indicated in Figure 20, the purpose of this section is to provide user information about the initialization and shutdown process for the ADRV9001 device utilizing the APIs developed by ADI. Figure 20 listed all high level operations used for initialization and shutdown through one or a set of APIs. In the later sections, the major steps associated with each high level operation are further discussed. Note with Software Development Kit (SDK) provided to the user, The ADRV9001 device can be initialized through the user's own software program independent of TES. However, the same API calling procedure described in this document should be followed.

Note all the information discussed in this section is subject to change over the time. It is not the intention of this section to explain every related API function. Detailed information regarding the API functions can be found in the ADRV9001 Device API doxygen document. In addition, this section does not describe API integration and the hardware abstraction Interface. Details of such can be found in the Software Integration section. To find more details about the TES, refer to the Transceiver Evaluation Software (TES) section.

TES CONFIGURATION AND INITIALIZATION

The TES provides a **Config** tab that contains all the setup options for the ADRV9001. Under the **Config** tab, the user could configure each channel of the device for a desired profile under the **Device Configuration** subtab, which sets high level parameters such as duplex mode, data port sample rates and RF channel bandwidth. Then the user could further initialize the options used by the device during startup under other subtabs, such as the carrier frequencies, ADC type and initial calibrations. Note GUI design could change significantly over the time, see the ADRV9001 Evaluation System section for up-to-date information.

Based on the parameters set by the user, an initialization structure, adi_adrv9001_Init_t, is formulated by TES to contain all the required settings to configure the device. This structure contains the system configuration setting, the system clock settings, transmit/receive data structure settings and Programmable FIR filter settings. Please refer to the doxygen document for more details.

After all tabs are configured, the user must press the **Program** button in TES. This kicks off initialization programming. TES sends a series of API commands that are executed by a dedicated Linux application on the platform. This initialization structure is passed to the

ADRV9001 API initialization functions during the initialization phase. When programming is completed, the system is fully calibrated. With a few additional API calls, the device is ready to operate.

The TES also provides the capability of generating a MATLAB code or python (.py) script or C code which includes all high level API initialization calls. Those automatically generated codes or scripts can be given a file name and stored in a location of the user's choice for future use.

API INITIALIZATION SEQUENCE

As aforementioned, the initialization sequence is comprised of a serial of API calls intermixed with user-defined function calls specific to the hardware platform. The API functions perform all the necessary tasks for device configuration, calibration and control. The following diagram describes the state machine of the device from power up to RF enabled.

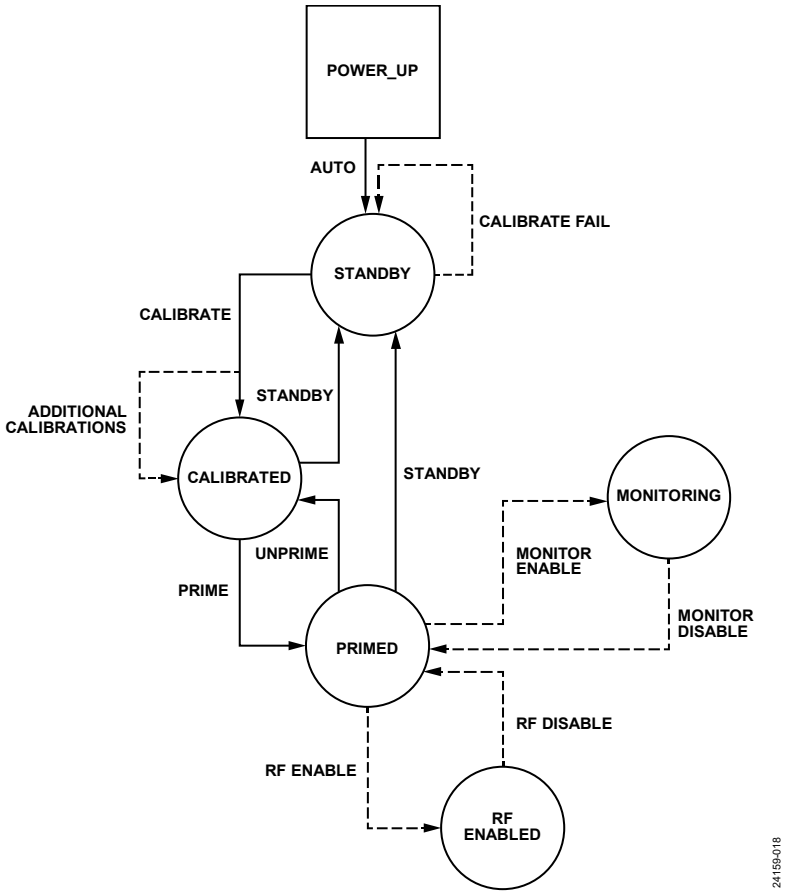


Figure 21. Device State Machine

As shown in Figure 21, after power-up, the device automatically enters the standby state, then the device initialization and calibration begin. If it is successful, the device moves to the calibrated state, otherwise, it remains in standby state. Note when TES completes programming, the device is in the calibrated state. After the device is calibrated, it must be further moved to the primed state, which indicates that the device is ready for operation. Then, through SPI or PIN mode, the device can be moved to the RF enabled state by enabling the transmit/receive channels so transmission and reception can start. Optionally, for power saving, the device can also enter the monitoring state from the primed state. Refer to Power Saving and Monitor Mode section for details. In TES, after programming, playing the receiver or transmitter moves the device from the calibrated state to the primed state and then to the RF enabled state.

This section mainly discusses the device initialization procedure from the standby state to the RF_ENABLED state. The related high level API functions are discussed briefly in the following subsections. Refer to the doxygen document for details of each API function.

Note for MIMO systems with multiple inputs and outputs channels, multiple ADRV9001 devices might be involved. To synchronize among all the devices, it requires a common device clock (DEV_CLOCK) and a multichip synchronization (MCS) signal so that all the internally generated analog and digital clocks are aligned among all the devices. In addition, the MCS is used to synchronize the device and baseband processor data interface for all devices. For simplicity, in the following descriptions, MCS operations are omitted from the initialization steps. Refer to Microprocessor and System Control section in the user guide for more details.

Analog Initialization

Analog initialization API adi_adrv9001_InitAnalog() is the very first API call to configure the device after all dependent data structures have been initialized. It mainly sets the master bias, validates the profile settings and configures the analog clocks.

Resource Loading

After analog initialization, a set of APIs are used to load required resources such as stream image, ARM image, programmable FIR (PFIR) coefficients and so on. It also enables the internal microprocessor and initialize digital clocks.

The major APIs as shown in Figure 22. The order of the major API calls is from the left to the right sequentially. The functionality of each API is explained in the box below it.

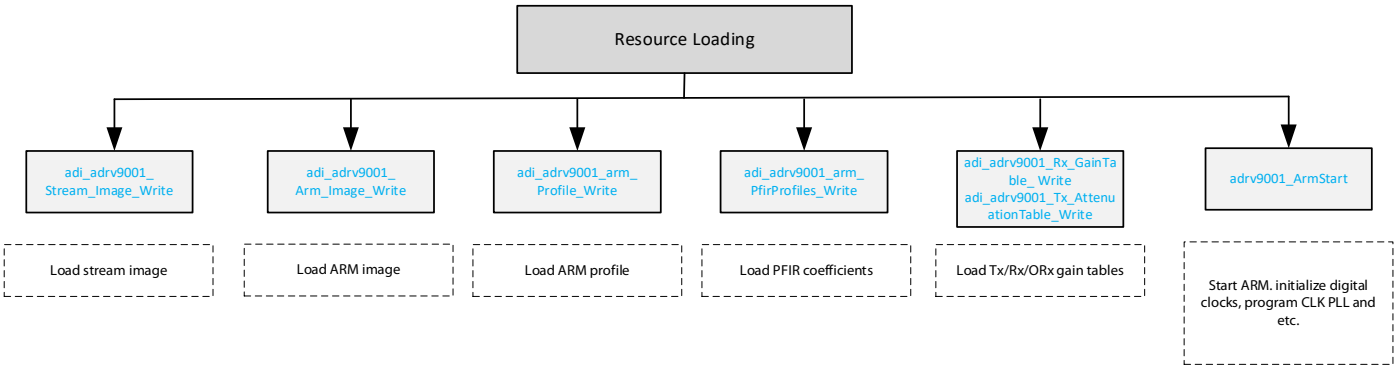


Figure 22. Load Resources and Digital Initialization

Radio Initialization

After digital initialization, the next step is radio initialization through a set of API calls, which is used to load any radio configuration data not passed by profile before performing initial calibrations, such as GPIO configuration, PLL loop filter configuration, carrier frequencies, TDD timing parameters, power management configurations, MCS delay configurations and etc.

The major APIs are shown in Figure 23.

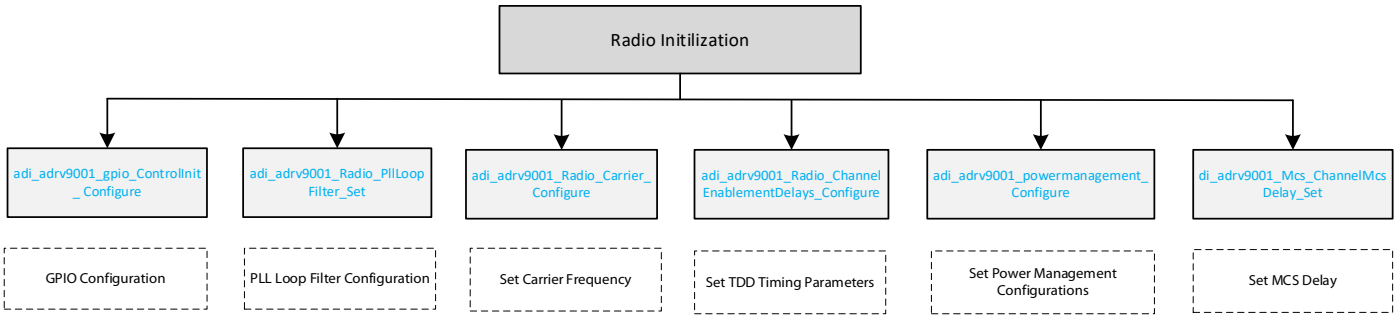


Figure 23. Radio Control Initialization

Calibrations Initialization

The next step in initialization is to perform initial calibrations through API call adi_adrv9001_cals_InitCals_Run() based on calibration mask. To understand calibration mask, see the Transmitter/Receiver/Observation Receiver Signal Chain Calibrations section. When initial calibrations are correctly performed, the channel state is transitioned from standby to the calibrated state as shown in Figure 21.

Prime and RF Enable

The last step in initialization is to move the device from calibrate to the primed state through API call adi_adrv9001_Radio_Channel_Prime(). The primed state indicates that the system is ready for operation when the transmit and receive channels are enabled by the user.

Note after the channel is primed, in order to start the normal transmit or reception activities, it must be further transitioned from primed state to RF enabled state. This can be accomplished by a set of API calls. There are two modes for channel enabling, which are PIN mode and SPI mode.

PIN Mode

1. Call adi_adrv9001_Radio_ChannelEnableMode_Set( ) to set the PIN mode
2. Toggle the pins (for example, Rx1_ENABLE and Tx1_ENABLE pins for Channel 1) to transition the channel to the RF enabled state.

SPI Mode

1. Call adi_adrv9001_Radio_ChannelEnableMode_Set( ) to set the SPI mode
2. Call adi_adrv9001_Radio_Channel_EnableRf( ) to transition the channel to the RF enabled state.

SHUTDOWN SEQUENCE

After completing all the operations, call API adi_adrv9001_Shutdown() through TES to safely shut down the ADRV9001 device. It performs a hardware reset to reset the ADRV9001 device into a safe state for shutdown or re-initialization.

SERIAL PERIPHERAL INTERFACE (SPI)

The SPI bus provides the mechanism for digital control by a baseband processor. Each SPI register is 8 bits wide, and each register contains control bits, status monitors, or other settings that control all functions of the device. This section is mainly an information-only section meant to give the user an understanding of the hardware interface used by the baseband processor to control the device. All control functions are implemented using the API detailed within this document. The following sections explain the specifics of this interface.

SPI CONFIGURATION

Users can configure SPI settings for the device with different SPI controller configurations by configuring member values of the adi_adrv9001_SpiSettings_t data structure. The adi_adrv9001_SpiSettings_t data structure contains:

```
typedef struct adi_adrv9001SpiSettings
{
    uint8_t msbFirst;
    uint8_t enSpiStreaming;
    uint8_t autoIncAddrUp;
    uint8_t fourWireMode;
    adi_adrv9001_CmosPadDrvStr_e cmosPadDrvStrength;
} adi_adrv9001_SpiSettings_t;
```

The parameters for this structure are listed in Table 12.

Table 12. SPI Settings Data Structure

Structure Member	Value	Function	Default
MSBFirst	0x00	Least significant bit first.	0x01
	0x01	Most significant bit first.	
enSpiStreaming	0x00	Disable SW feature. Section Multi-Byte Data Transfer (SPI Streaming) describes this mode of operation.	0x00
	0x01	Enable SW feature to improve SPI throughput. Section Multi-Byte Data Transfer (SPI Streaming) describes this mode of operation. Not Recommended since most registers in ADRV9001 API are not consecutive.	
autoIncAddrUp	0x00	Auto-decrement. Functionality intended to be used with SPI Streaming. Sets address auto-decrement -> next addr = addr -1	0x01
	0x01	Auto-increment. Functionality intended to be used with SPI Streaming. Sets address auto-increment -> next addr = addr +1	
fourWireMode	0x00	SPI hardware implementation. Use 3-wire SPI (SDIO pin is bidirectional). Figure 23 shows example of SPI 3-wire mode of operation. NOTE: ADI's FPGA platform always uses 4-wire mode.	0x01
	0x01	SPI hardware implementation. Use 4-wire SPI. Figure 21 and Figure 22 show examples of SPI 4 –wire mode of operation. NOTE: Default mode for ADI's FPGA platform is 4-wire mode.	
cmosPadDrvStrength	0x00	5 pF load @ 75 MHz	0x01
	0x01	100 pF load @ 20 MHz	

Any value that is not listed in the table is invalid.

For more details, refer to ADRV9001_API doxygen file provided in ADRV9001 SDK package.

SPI BUS SIGNALS

The SPI bus consists of the following signals:

- SCLK
- CSB
- SDIO and SDO

SCLK

SCLK is the serial interface reference clock driven by the baseband processor (uses the SPI_CLK pin). It is only active while CSB is low. The minimum SCLK frequency is 1 kHz. The maximum SCLK frequency is 50 MHz.

CSB

CSB is the active-low chip select that functions as the bus enable signal driven from the baseband processor to the device (uses the SPI_EN pin). CSB is driven low before the first SCLK rising edge and is normally driven high again after the last SCLK falling edge. The device ignores the clock and data signals while CSB is high. CSB also frames communication to and from the device and returns the SPI interface to the ready state when it is driven high.

Forcing CSB high in the middle of a transaction aborts part or all of the transaction. If the transaction is aborted before the instruction is complete or in the middle of the first data word, the transaction is aborted and the state machine returned to the ready state. Any complete data byte transfers prior to CSB deserting are valid, but all subsequent transfers in a continuous SPI transaction are aborted.

SDIO and SDO

When configured as a 4-wire bus, the SPI uses two data signals: SDIO and SDO. SDIO is the data input line driven from the baseband processor (uses the SPI_DIO pin) and SDO is the data output from the device to the baseband processor in this configuration (uses the SPI_DO pin). When configured as a 3-wire bus, SDIO is used as a bidirectional data signal that both receives and transmits serial data. The SDO port is disabled in this mode.

The data signals are launched on the falling edge of SCLK and sampled on the rising edge of SCLK by both the baseband processor and the device. SDIO carries the control field from the baseband processor to the device during all transactions, and it carries the write data fields during a write transaction. In a 3-wire SPI configuration, SDIO carries the returning read data fields from the device to the baseband processor during a read transaction. In a 4-wire SPI configuration, SDO carries the returning data fields to the baseband processor.

The SDO and SDIO pins transition to a high-impedance state when the CSB input is high. The device does not provide any weak pull-ups or pull-downs on these pins. When SDO is inactive, it is floated in a high-impedance state. If a valid logic state on SDO is required at all time, an external weak pull-up/down (10 kΩ value) should be added on the PCB.

SPI DATA TRANSFER PROTOCOL

The SPI is a flexible, synchronous serial communication bus allowing seamless interfacing to many industry standard microcontrollers and microprocessors. The serial I/O is compatible with most synchronous transfer formats, including both the Motorola SPI and Intel SSR protocols. The control field width for this device is limited to 16 bits, and multi-byte IO operation is allowed. This device cannot be used to control other devices on the bus; it only operates as a slave.

There are two phases to a communication cycle. Phase 1 is the control cycle, which is the writing of a control word into the device. The control word provides the serial port controller with information regarding the data field transfer cycle, which is Phase 2 of the communication cycle. The Phase 1 control field defines whether the upcoming data transfer is read or write. It also defines the register address being accessed.

Phase 1 Instruction Format

The 16-bit control field contains the following information:

MSB	D14:D0
R/Wb	A<14:0>

R/Wb—Bit 15 of the instruction word determines whether a read or write data transfer occurs after the instruction byte write. Logic high indicates a read operation; logic zero indicates a write operation.

D14:D0—Bits A<14:0> specify the starting byte address for the data transfer during Phase 2 of the I/O operation.

All byte addresses, both starting and internally generated addresses, are assumed to be valid. That is, if an invalid address (undefined register) is accessed, the IO operation continues as if the address space were valid. For write operations, the written bits are discarded, and read operations result in logic zeros at the output.

Single-Byte Data Transfer

When enSpiStreaming = 0, a single-byte data transfer is chosen. In this mode, CSB goes active-low, the SCLK signal activates, and the address is transferred from the baseband processor to the device.

In LSB mode, the LSB of the address is the first bit transmitted from the baseband processor, followed by the next 14 bits in order from next LSB to MSB. The next bit signifies if the operation is read (set) or write (clear). If the operation is a write, the baseband processor transmits the next 8 bits LSB to MSB. If the operation is a read, the device transmits the next 8 bits LSB to MSB. Once the final bit is transferred, the data lines return to their idle state and the CSB line must be driven high to end the communication session.

In MSB mode, the first bit transmitted is the R/Wb bit that determines if the operation is a read (set) or write (clear). The MSB of the address is the next bit transmitted from the baseband processor, followed by the remaining 14 bits in order from next MSB to LSB. If the operation is a write, the baseband processor transmits the next 8 bits MSB to LSB. If the operation is a read, the device transmits the next 8 bits MSB to LSB. Once the final bit is transferred, the data lines return to their idle state and the CSB line must be driven high to end the communication session.

Multibyte Data Transfer

When enSpiStreaming = 1, a multi-byte data transfer is allowed. In this mode, data transfers across the bus as long as the CSB pin is low. The autoIncAddrUp controls how the address changes for subsequent writes or reads. When autoIncAddrUp = 1, the address increments from the starting address for each subsequent data transfer until CSB is driven high. If the last register address is reached, the next address accessed is 0x000. When autoIncAddrUp = 0, the address decrements from the starting address for each subsequent data transfer. If address 0x000 is reached, the next address that is accessed is the last register location defined in the register map. The register address 0x000 is used to setup SPI interface as well as functionality to soft reset the device. Uncontrolled data written to the register address 0x000 can cause SPI misconfigurations or can reset the device. It is strongly recommended that any data transfer using Multi-Byte Data feature to be controlled so that 0x000 is only written once at startup.

For multi-byte data transfers in LSB mode, the LSB of the address is the first bit transmitted from the baseband processor, followed by the next 14 bits in order from next LSB to MSB. The next bit signifies if the operation is read (set) or write (clear). If the operation is a write, the baseband processor transmits the next 8 bits LSB to MSB. After the MSB is received, the address increments or decrements based on the autoIncAddrUp parameter. The baseband processor, then continues to transfer data in 8-bit words, LSB to MSB, until the operation is terminated by CSB being driven high. If the operation is a read, the device transmits the next 8 bits LSB to MSB. It then changes the address and continues to transfer data in 8-bit words, LSB to MSB, until the operation is terminated by CSB being driven high.

For multi-byte data transfers in MSB mode, the same process is followed, except the first bit transferred indicates if the operation is read (set) or write (clear). The starting address is then transmitted by the baseband processor MSB to LSB, followed by the data transfer, MSB to LSB. Address increment or decrement is still controlled by the autoIncAddrUp parameter.

Using multi-byte data transfer mode provides little benefit because most registers in the device are not consecutive. It is up to the user to determine if multi-byte data transfer enhances device control in their end application compared to the single command format.

Example: LSB-First Multibyte Transfer, Auto-Incrementing Address

To complete a 4-byte write starting at register address 0x02A and ending with register 0x02D in LSB-first format, follow these instructions when programming the master:

- Make sure that fourWireMode = 1 – the device is configured to work with 4-wire interface.
- Make sure that MSBFirst = 0 - SPI operates in LSB first mode.
- Make sure that autoIncAddrUp = 1 - the address pointer automatically increments.
- Make sure that enSpiStreaming = 1 - a multi-byte data transfer is allowed.
- Force the CSB line low and keep it low until the last byte is transferred.
- Send the instruction word 0101 0100 0000 000_0 (the last 0 indicates a write operation) to select 0x02A as the starting address.
- Use the next 32 clock cycles to send the data to be written to the registers, LSB to MSB for each 8-bit word.
- Make sure the CSB line is driven high after the last bit has been sent to 0x02D to end the data transfer.

Example: MSB-First Multibyte Transfer, Autodecrementing Address

To complete a 4-byte write starting at register address 0x02A and ending with register 0x027 in LSB-first format, follow these instructions when programming the master:

- Make sure that fourWireMode = 1 – the device is configured to work with 4-wire interface.
- Make sure that MSBFirst = 1 - SPI operates in MSB first mode.
- Make sure that autoIncAddrUp = 0 - the address pointer automatically decrements.
- Make sure that enSpiStreaming = 1 - a multi-byte data transfer is allowed.

- Force the CSB line low and keep it low until the last byte is transferred.
- Send the instruction word 0_000 0000 0010 1010 (the first 0 indicates a write operation) to select 0x02A as the starting address.
- Use the next 32 clock cycles to send the data to be written to the registers, MSB to LSB for each 8-bit word.
- Make sure the CSB line is driven high after the last bit has been sent to 0x027 to end the data transfer.

TIMING DIAGRAMS

The diagrams in Figure 24 and Figure 25 illustrate the SPI bus waveforms for a single-register write operation and a single-register read operation, respectively. In the first figure, the value 0x55 is written to register 0x00A. In the second value, register 0x00A is read, and the value returned by the device is 0x55. If the same operations were performed with a 3-wire bus, the SDO line in Figure 24 would be eliminated, and the SDIO and SDO lines in Figure 25 would be combined on the SDIO line. Note that both operations use MSB-first mode and all data is latched on the rising edge of the SCLK signal.

Users should be advised, register 0x00A is not user accessible, the SPI write and read operation in Figure 24 and Figure 25 is only for the SPI timing diagram demonstration purpose. Users can use the scratch register 0x009 for the SPI read/write test.

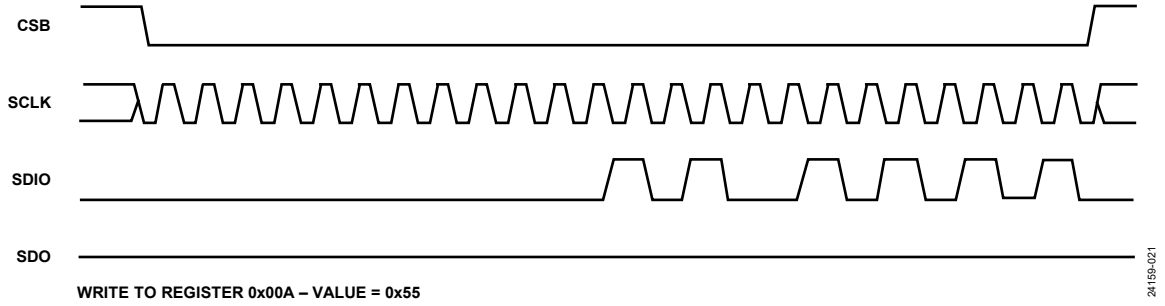


Figure 24. Nominal Timing Diagram, SPI Write Operation

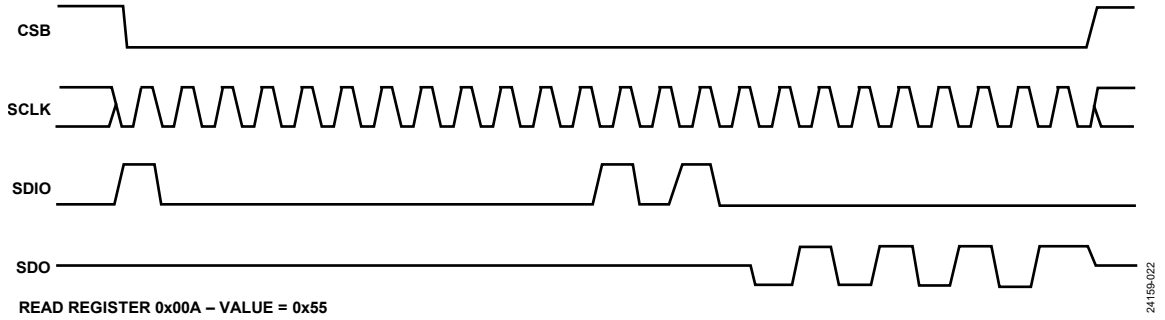


Figure 25. Nominal Timing Diagram, SPI Read Operation

Table 13 lists the timing specifications for the SPI bus. The relationship between these parameters is shown in Figure 26. This diagram shows a 3-wire SPI bus timing diagram with the device returning a value of 0xD4 from a test register and timing parameters marked. Note that this is a single read operation, so the bus-ready parameter after the data is driven from the device (t_{HZS}) is not shown in the diagram.

Table 13. SPI Bus Timing Constraint Values

Parameter	Min	Typical	Max	Description
t_{CP}	28 ns			SCLK Period, 3-wire mode
	22ns			SCLK period, 4-wire mode
t_{MP}	10 ns			SCLK pulse width
t_{SC}	3 ns			CSB setup time to first SCLK rising edge
t_{HC}	0 ns			Last SCLK falling edge to CSB hold
t_S	2 ns			SDIO data input setup time to SCLK
t_H	0 ns			SDIO data input hold time to SCLK
t_{CO}	3 ns		15 ns	SCLK falling edge to output data delay (3-wire mode)
	3ns		10 ns	SCLK falling edge to output data delay (3-wire mode)
t_{HZM}	t_H		t_{CO} (max)	Bus turnaround time after baseband processor drives the last address bit
t_{HZS}	0 ns		t_{CO} (max)	Bus turnaround time after device drives the last data bit

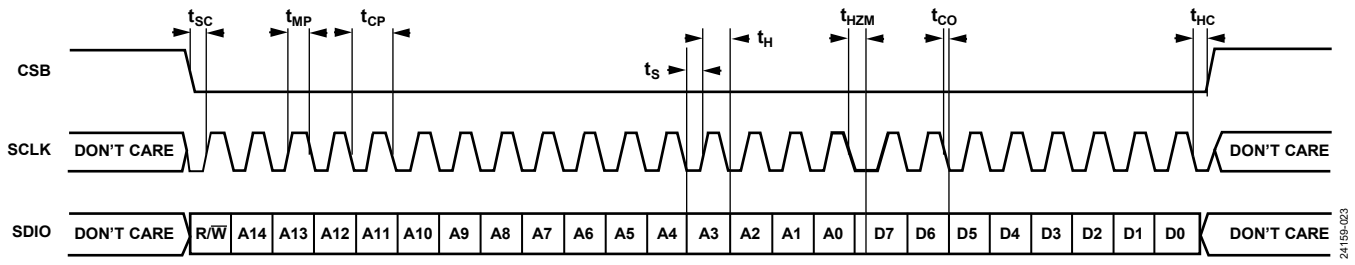


Figure 26. 3-Wire SPI Timing with Parameter Labels, SPI Read

SPI TEST

ADRV9001 has a scratch SPI resister 0x009 for users to perform the SPI read/write validation. Users can follow below procedure to quickly check the SPI function before the relative BBIC drivers are ready.

- Power on the ADRV9001 properly
- Toggle the resetb pin to reset the ADRV9001
- Write register 0x0 with value 0x3C to set ADRV9001 SPI to 4 wire mode, or with value 0x24 to set ADRV9001 SPI to 3 wire mode
- Write whatever value to scratch register 0x009, then read register 0x009 to validate if the read value is the write one

Users should use the oscilloscope to probe the SPI bus signal and check if the SPI master follow the timing diagrams in Figure 24 and Figure 25 when above SPI validation can't pass.

Users can also use the API adi_adrv9001_spi_Verify() to validate the SPI after the adi_adrv9001_spi_Configure() is set if the BBIC has the available drivers.

DATA INTERFACE

GENERAL DESCRIPTION

This document defines the synchronous serial interface (SSI) which transfer data between the ADRV9001 and a baseband processor . ADRV9001 SSI consists of two receive channels and two transmit channels, the channels are independent and can be configured as CMOS signals (CSSI) for applications that have narrow RF signal bandwidths and low data rate or as LVDS signals (LSSI) for applications that require high speed, low noise and longer distance data transfer.

The CSSI supports below two modes of operation and can be operated as either in single data rate (SDR) or double data rate (DDR) data transfer, the maximum clock frequency is 80 MHz.

- One lane data mode, I/Q data or other format data are serialized onto one single lane.
- Four lanes data mode, which is valid only when ADRV9001 transmit or receive I/Q samples and I/Q samples are 16 bits wide. In four-lane data mode, each sample is split into 8 bits block of data and sent over one data lane.

The LSSI also supports two modes of operation, the LSSI always operates in DDR data transfer, the maximum clock frequency is up to 491.52 MHz.

- I/Q in one lane (one-lane mode)
  - With I-Q data samples of 16 bits (total of 32 bits for each transfer)
- I/Q in separate lanes (two-lane mode)
  - With I and Q data samples of 16bits
  - With I and Q data samples of 12 bits

ADRV9001 SSI has various and flexible work modes to support all kinds of system scenarios, users can choose their appropriate work modes according to the interface sample/symbol rate and bit width. Table 14 lists the ADRV9001 SSI work modes and the maximum support I/Q sample rate.

Table 14. ADRV9001 SSI Work Modes

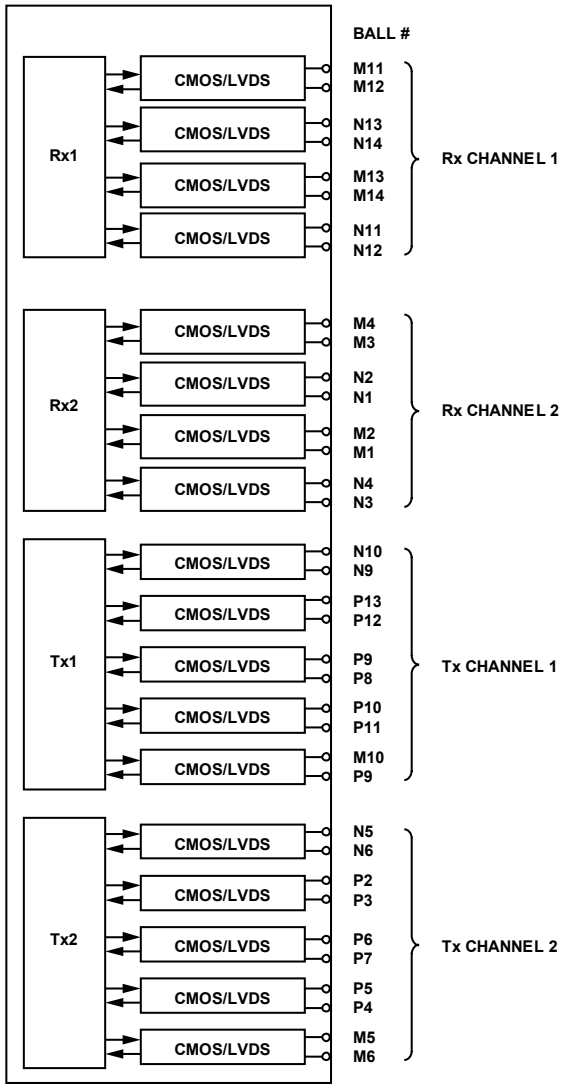
SSI Modes	Data Lanes Per Channel	Serialization Factor Per Data Lane	Maximum Data Lane Rate (MHz)	Maximum Clock Rate (MHz)	Maximum Sample Rate for I/Q (MHz)	Data Type
CSSI 1-Lane	1	32	80	80	2.5	SDR
CSSI 1-Lane	1	32	160	80	5	DDR
CSSI 1-Lane[1]	1	16/8/2	80-SDR/160-DDR	80	Not Applicable	SDR/DDR
CSSI 4-Lane	4	8	80	80	10	SDR
CSSI 4-Lane	4	8	160	80	20	DDR
LSSI 1-Lane	1	32	983.04	491.52	30.72	DDR
LSSI 2-Lane	2	16	983.04	491.52	61.44	DDR
LSSI 2- Lane[2]	2	12	737.28	368.64	61.44	DDR

[1] ADRV9001 data port transmit/receive data symbols, refer CSSI Data Symbols Transmit and Receive.
[2] For User's LVDS data lane rate limitation applications, RX samples are rounded from 16 bits to 12 bits. Tx Sample are extended from 12bits to 16bits.

The following sections explain the details of the signals that make up the SSI and their properties when configured for each mode.

ELECTRICAL SPECIFICATION

ADRV9001 SSI can operate in standard single ended CMOS compatible mode or Low-voltage Differential Signal (LVDS) compatible mode, CMOS SSI and LVDS SSI share the IO pads of ADRV9001. Figure 27 describes the four channels with their corresponding IOs in CMOS and LVDS modes.



	PIN	LVDS-SSI	CMOS-SSI 1-LANE	CMOS-SSI 4-LANE
Rx CHANNEL 1	M11	RX1_DCLK_OUT+	RX1_DCLK_OUT	RX1_DCLK_OUT
	M12	RX1_DCLK_OUT–		
	N13	RX1_STROBE_OUT+	RX1_STROBE_OUT	RX1_STROBE_OUT
	N14	RX1_STROBE_OUT–		
	M13	RX1_IDATA_OUT+		RX1_IDATA1_OUT
	M14	RX1_IDATA_OUT–	RX1_DATA_OUT	RX1_IDATA0_OUT
	N11	RX1_QDATA_OUT+		RX1_QDATA3_OUT
	N12	RX1_QDATA_OUT–		RX1_QDATA2_OUT
Rx CHANNEL 2	M4	RX2_DCLK_OUT+	RX2_DCLK_OUT	RX2_DCLK_OUT
	M3	RX2_DCLK_OUT–		
	N2	RX2_STROBE_OUT+	RX2_STROBE_OUT	RX2_STROBE_OUT
	N1	RX2_STROBE_OUT–		
	M2	RX2_IDATA_OUT+		RX2_IDATA1_OUT
	M1	RX2_IDATA_OUT–	RX2_DATA_OUT	RX2_IDATA0_OUT
	N4	RX2_QDATA_OUT+		RX2_QDATA3_OUT
	N3	RX2_QDATA_OUT–		RX2_QDATA2_OUT
Tx CHANNEL 1	N10	TX1_DCLK_IN+	TX1_DCLK_IN	TX1_DCLK_IN
	N9	TX1_DCLK_IN–		
	P13	TX1_STROBE_IN+	TX1_STROBE_IN	TX1_STROBE_IN
	P12	TX1_STROBE_IN–	TX1_DCLK_OUT	TX1_DCLK_OUT
	P9	TX1_IDATA_IN+		TX1_IDATA1_IN
	P8	TX1_IDATA_IN–	TX1_DATA_IN	TX1_IDATA0_IN
	P10	TX1_QDATA_IN+		TX1_QDATA3_IN
	P11	TX1_QDATA_IN–		TX1_QDATA2_IN
	M10	TX1_DCLK_OUT+	TX1_DCLK_OUT	TX1_DCLK_OUT
	M9	TX1_DCLK_OUT–	TX1_DCLK_OUT	TX1_DCLK_OUT
Tx CHANNEL 2	N5	TX2_DCLK_IN+	TX2_STROBE_IN	TX2_DCLK_IN+
	N6	TX2_DCLK_IN–		
	P2	TX2_STROBE_IN+	TX2_STROBE_IN	TX2_STROBE_IN
	P3	TX2_STROBE_IN–	TX2_DCLK_OUT	TX2_DCLK_OUT
	P6	TX2_IDATA_IN+		TX2_IDATA1_IN
	P7	TX2_IDATA_IN–	TX2_DATA_IN	TX2_IDATA0_IN
	P5	TX2_QDATA_IN+		TX2_QDATA3_IN
	P4	TX2_QDATA_IN–		TX2_QDATA2_IN
	M5	TX2_DCLK_OUT+	TX2_DCLK_OUT	TX2_DCLK_OUT
	M6	TX2_DCLK_OUT–	TX2_DCLK_OUT	TX2_DCLK_OUT

24159-024

Figure 27. ADRV9001 SSI I/Os Mapping

CMOS SSI electrical specification is shown in Table 15. For good performance, the CMOS outputs should drive minimal capacitive loads. The CMOS output drive strength can be increased for capacitive loads, bigger than 10 pF to increase the edge rate of output signal during the transitional period, the maximum capacitive load can reach to 30 pF at 80 MHz clock data rate.

In LVDS mode, an external 100 Ω differential termination resistor is required for each LVDS pair, and the termination resistors should be located as close as possible to the LVDS receiver. ADRV9001 LVDS in circuit has optional internal 100 Ω termination resistor which can be enabled for LSSI, but ADRV9001 LVDS output circuit does not have internal termination resistors, users should develop appropriate LVDS termination resistors in LVDS receiver. The default LVDS out circuit produces 350 mV peak at 1.2 V common mode level, output swing level can be increased to 450 mV for longer trace. LVDS SSI electrical specification is shown in Table 16.

It is recommended to keep trace lengths of SSI Clock, Strobe, Data signals into one Transmit or Receive channel as equal as possible. ADRV9001 SSI has configurable delay cells on LVDS/CMOS in and out circuits which can allow users to small adjust the phase relationship between strobe/data and clock, the adjustable phase delay cell is approximate 90 ps per step for LVD mode and 170 ps per step for CMOS mode, the maximum adjustable step is 7.

Table 15. CSSI Electrical Specification

Symbol	Parameter	Min	Typ	Max	Units
VDIGIO_1P8	Interface power supply voltage	1.71	1.8	1.89	V
V_{IH}	Input voltage high	VDIGIO_1P8 × 0.65		VDIGIO_1P8 + 0.18	V
V_{IL}	Input voltage low	-0.3		VDIGIO_1P8 × 0.35	V
V_{OH}	Output voltage high	VDIGIO_1P8 – 0.45		VDIGIO_1P8	V
V_{OL}	Output voltage low			0.45	V
f_{CLK}	Clock frequency			80	MHz
C_L @ 80 MHz	Load capacitance supported for an 80 MHz clock waveform		10	30	pF

Table 16. LSSI Electrical Specification

Symbol	Parameter	Conditions	Min	Typ	Max	Units
VDIGIO_1P8	Interface power supply voltage		1.71	1.8	1.89	V
V_I	Input voltage range		825		1675	mV
V_{IDTH}	Input differential threshold		−100		+100	mV
R_{IN}	Receiver differential input impedance			100		Ω
V_{OH}	Output voltage high	R_{LOAD} = 100 Ω ± 1%			1390	mV
V_{OL}	Output voltage low	R_{LOAD} = 100 Ω ± 1%	1000			mV
$\|V_{OD}\|$	Output differential voltage	R_{LOAD} = 100 Ω ± 1%		360		mV
V_{OS}	Output offset voltage	R_{LOAD} = 100 Ω ± 1%	1150	1200	1250	mV
R_O	Output impedance, single ended		80	100	120	Ω
I_{SA}, I_{SB}	Output current	Driver shorted to ground			17	mA
I_{SAB}	Output current	Drivers shorted together			4.1	mA
	Clock signal duty cycle	500 MHz	45	50	55	%
T_R, T_F	Output Rise/Fall Time	300 mVp swing		0.371		nsec

CMOS SYNCHRONOUS SERIAL INTERFACE (CMOS-SSI)

One-Lane Mode CSSI Interface

Receive CSSI Interface

The one-lane mode receive CSSI interfaces of each channel (Rx1 and Rx2) are a 3-wire digital interface consisting of:

- RX_DCLK_OUT: is an output clock synchronizing data and strobe output signals.
- RX_STROBE_OUT: is an output signal indicating the first bit of the serial data sample.
- RX_DATA_OUT: is an output serial data stream.

The I and Q samples are serialized out starting with configurable I or Q first and MSB or LSB first, Figure 28 illustrates the receive CSSI interface (Rx1 and Rx2) for a 16-bit I/Q data sample with I sample and MSB first configuration.

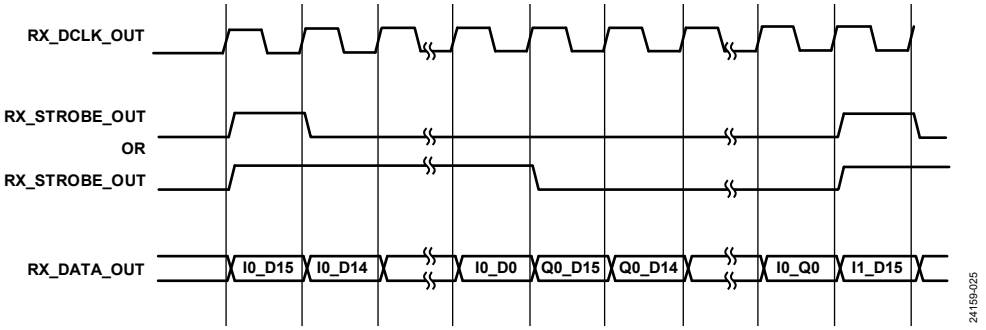


Figure 28. Receive CSSI Timing for 16-Bit I/Q Data Sample (I and MSB First)

The RX_STROBE_OUT signal is aligned with the first bit of the serialized data (I and Q), and can be configured to be high:

For one clock cycle at start of I and Q sample transmit. In the case a 16-bit data sample, RX_STROBE is high for one clock cycle and low for 31 clock cycles.

For I data duration and low for Q data duration. In the case of a 16-bit data sample, RX_STROBE is high for 16 clock cycles (I data sample) and low for 16 clock cycles (Q data sample).

The Transmit CSSI Interface

The one-lane mode transmit CSSI interface of each channel (Tx1 and Tx2) is a 4-wire digital interface consisting of:

- TX_DCLK_IN: is an input clock synchronized to the data and strobe inputs.
- TX_STROBE_IN: is an input signal indicating the first bit of the serial data sample.
- TX_DATA_IN: is an input serial data stream.
- TX_DCLK_OUT: is an optional output reference clock that is provided to the baseband processor to generate all the above signals, the baseband processor can also use RX_DCLK_OUT as the reference clock when its clock rate is equal with Transmit SSI clock rate.

The I and Q samples can be deserialized starting with configurable I or Q first and MSB or LSB first, Figure 29 illustrates the Transmit CSSI interface (Tx1 and Tx2) for a 16-bit I/Q data sample with I sample and MSB first configuration.

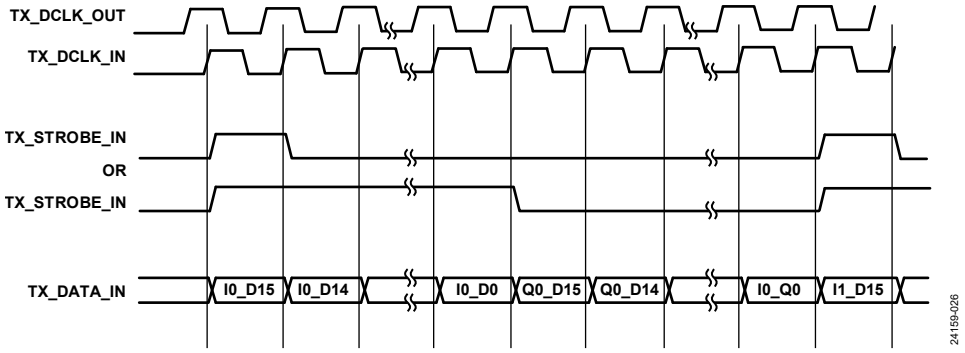


Figure 29. Transmit CSSI Timing for 16-Bit I/Q Data Sample (I and MSB First)

The TX_STROBE_IN signal is aligned with the first bit of the serialized data (I and Q), and can be configured to be high:

- For one clock cycle at start of I and Q sample transmit. In the case a 16-bit data sample, the TX_STROBE is high for one clock cycle and low for 31 clock cycles.
- For I data duration and Low for Q data duration. In the case of a 16-bit data sample, TX_STROBE is high for 16 clock cycles (I data sample) and low for 16 clock cycles (Q data sample).

CSSI Data Symbols Transmit and Receive

The previous sections described data transfer with I/Q format with 16bit width. When the ADRV9001 internal modulation/demodulation is enabled (refer to the Transmitter Signal Chain and Rx Demodulator sections), the data transfer between ADRV9001 and baseband processor would be 2 bits or 16 bits I only data (denoted as symbol to differentiate with I/Q complex samples). In a symbol format mode, raw data are transferred through this interface using different data size. The CSSI interface supports three additional data formats:

- 2 bits of data
- 8 bits of data
- 16 bits of data

Data with a size of two bits could be transferred over a CSSI with an 8-bit data format with six dummy bits. The clock and strobe behavior are similar to the I/Q format described in previous sections.

Figure 30 illustrates the receive CSSI interface (Rx) for 2-bit data symbols.

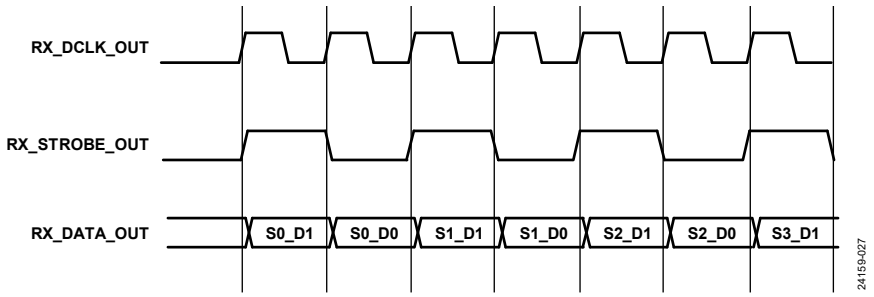


Figure 30. Receive CSSI Timing for 2-Bit Symbols (MSB First)

Figure 31 illustrates the transmit CSSI interface (Tx) for 2-bit data symbols.

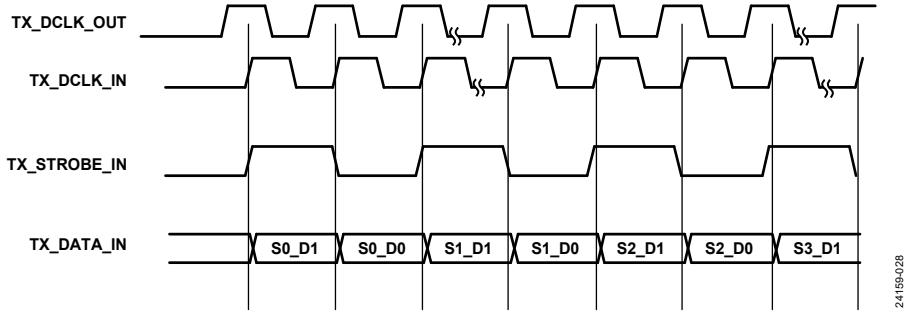


Figure 31. Transmit CSSI Timing for 2-Bit Symbols (MSB First)

Figure 32 illustrates the receive CSSI interface (Rx) for 8-bit data symbols.

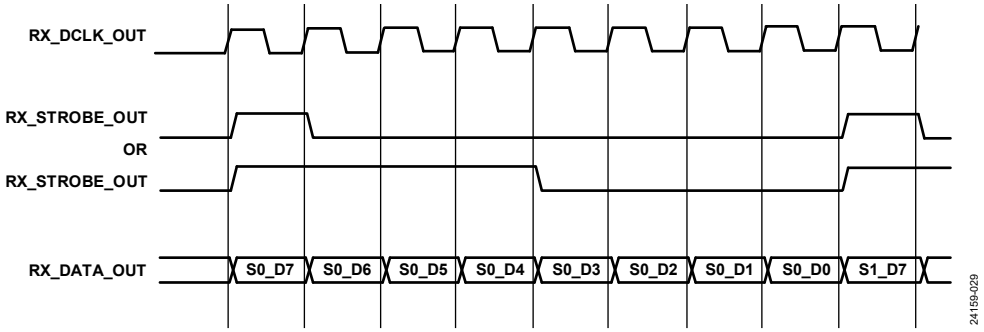


Figure 32. Receive CSSI Timing for 8-Bit Symbols (MSB First)

Figure 33 illustrates the transmit CSSI interface (Tx) for a 8-bit data symbols.

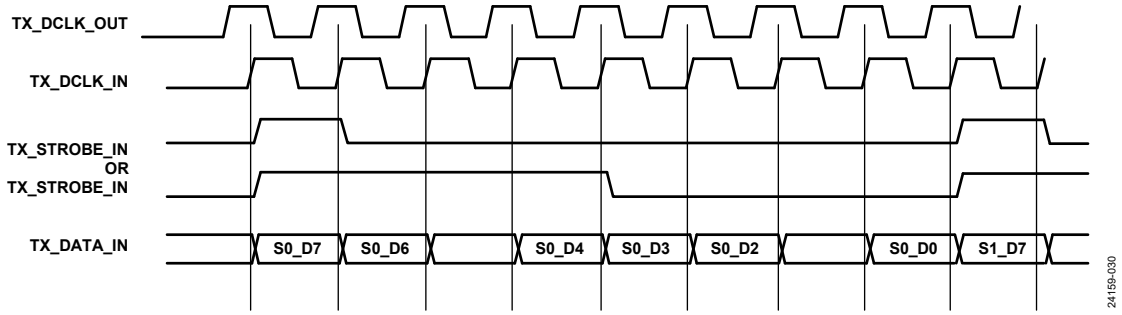


Figure 33. Transmit CSSI Timing for 8-Bit Symbols (MSB First)

Figure 34 illustrates the receive CSSI interface (Rx) for 16-bit data symbols.

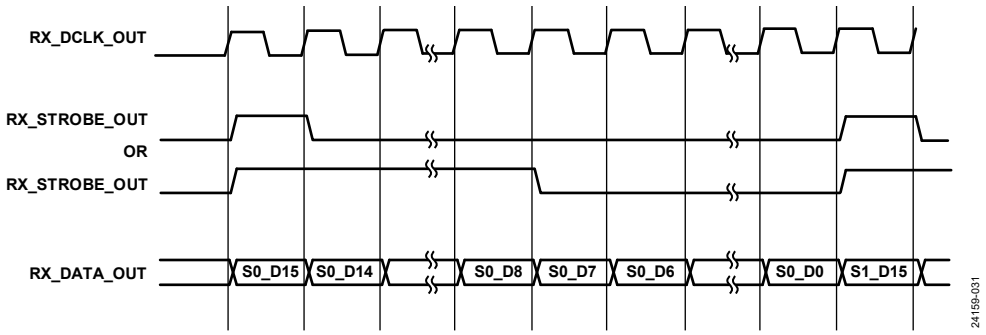


Figure 34. Receive CSSI Timing for 16-Bit Symbols (MSB First)

Figure 35 illustrates the transmit CSSI interface (Tx) for a 16-bit data symbols.

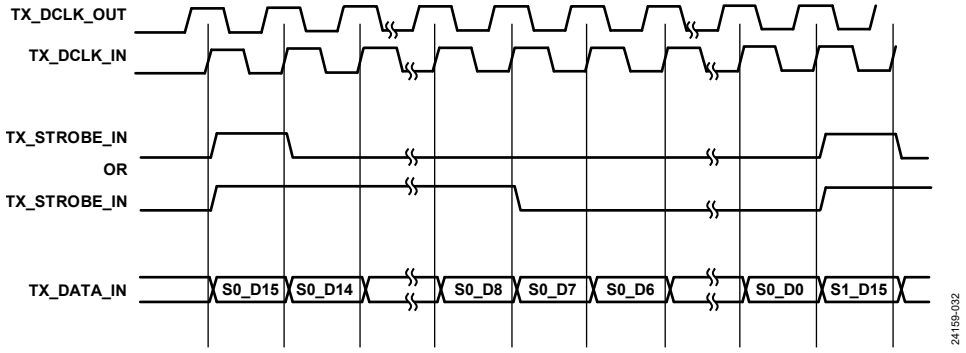


Figure 35. Transmit CSSI Timing for 16-Bit Symbols (MSB First)

Receive CSSI Interface with 2×, 4×, and 8× Data Clock Rates

ADRV9001 receive CSSI supports the 2 times, 4 times, or 8 times of the data clock rate for some applications.

Figure 36, Figure 37, and Figure 38 illustrate the receive CSSI interface (Rx1 and Rx2) for 16-bit I/Q data sample with 2×, 4×, and 8× clock rates. The strobe pulse validates the start of the 32-bit I and Q samples, the remaining data bits are ignored.

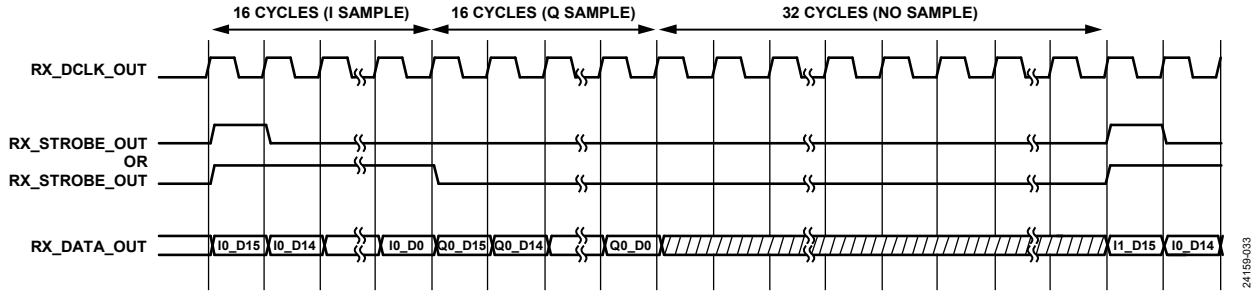


Figure 36. Receive CSSI Timing with 2× Data Clock Rate for 16-Bit I/Q Data Sample (MSB First), 32 Cycles

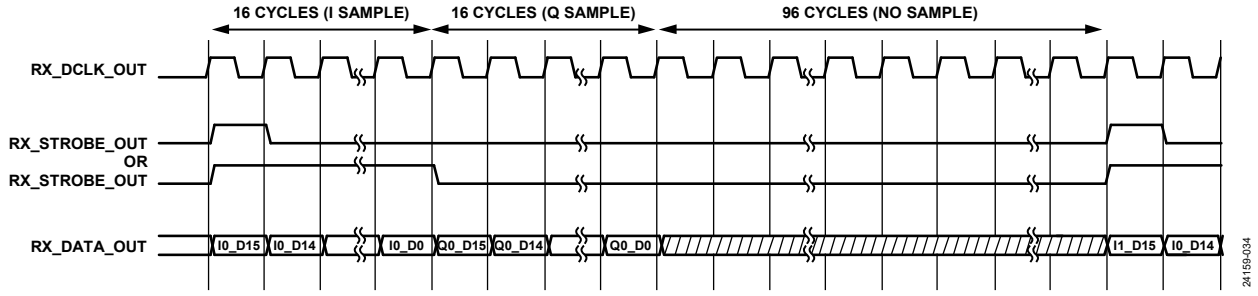


Figure 37. Receive CSSI Timing with 4× Data Clock Rate for 16-Bit I/Q Data Sample (MSB First), 96 Cycles

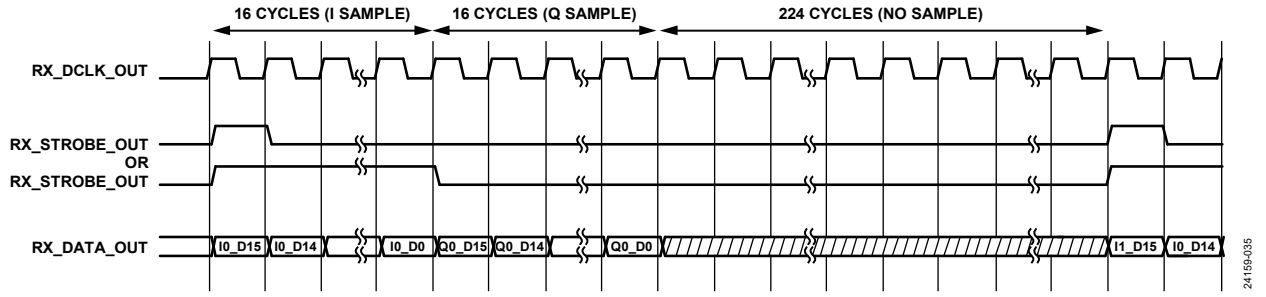


Figure 38. Receive CSSI timing with 8× Data Clock Rate for 16-Bit I/Q Data Sample (MSB First), 224 Cycles

Figure 39, Figure 40, and Figure 41 illustrate the Receive CSSI interface (Rx1 and Rx2) in frequency deviation mode with 16-bit data symbol with 2×, 4×, and 8× clock rates. The strobe pulse validates the start of the 16bits data symbol, the remaining data bits are ignored.

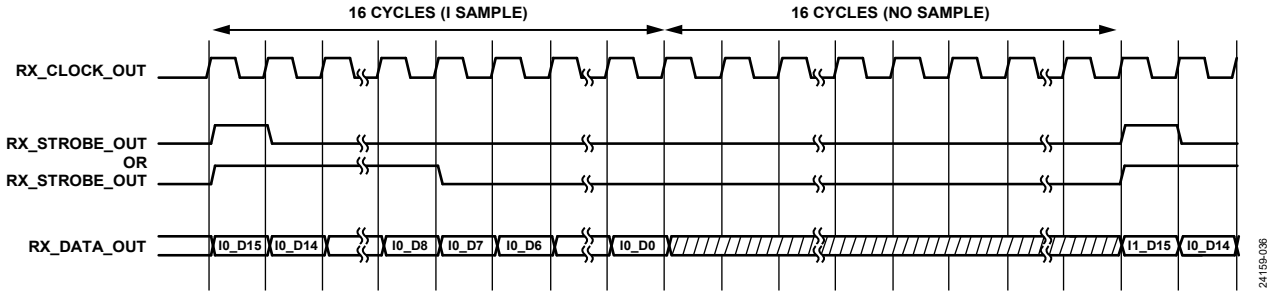


Figure 39. Receive CSSI Timing with 2× Data Clock Rate for 16-Bit Data Symbol (MSB First)

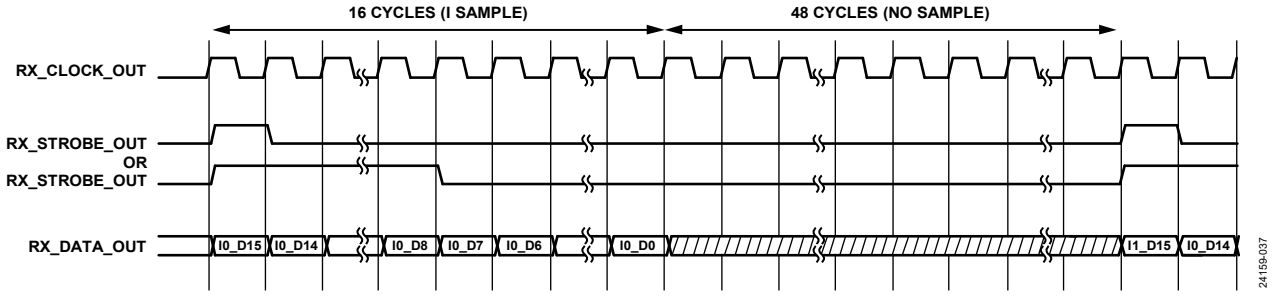


Figure 40. CSSI Receive Timing with 4× Data Clock Rate for 16-Bit Data Symbol (MSB First)

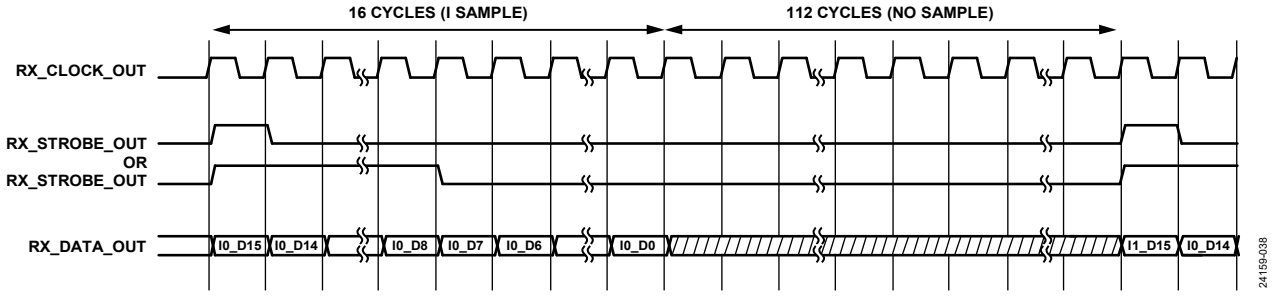


Figure 41. Receive CSSI Timing with 8× Data Clock Rate for 16-Bit Data Symbol (MSB First)

Four-Lane Mode CSSI Interface

The four-lane mode receive CSSI interface of each channel (Rx1 and Rx2) are a 6-wire digital interface consisting of:

- RX_DCLK_OUT: is an output clock synchronous data and strobe output signals.
- RX_STROBE_OUT: is an output signal indicating the first bit of the serial data sample.
- RX_IDATA0_OUT: is an output serial data stream of I sample low byte.
- RX_IDATA1_OUT: is an output serial data stream of I sample high byte.
- RX_QDATA2_OUT: is an output serial data stream of Q sample low byte.
- RX_QDATA3_OUT: is an output serial data stream of Q sample high byte.

Figure 42 illustrates the receive CSSI interface (Rx1 and Rx2) for a four-lane format with MSB first configuration.

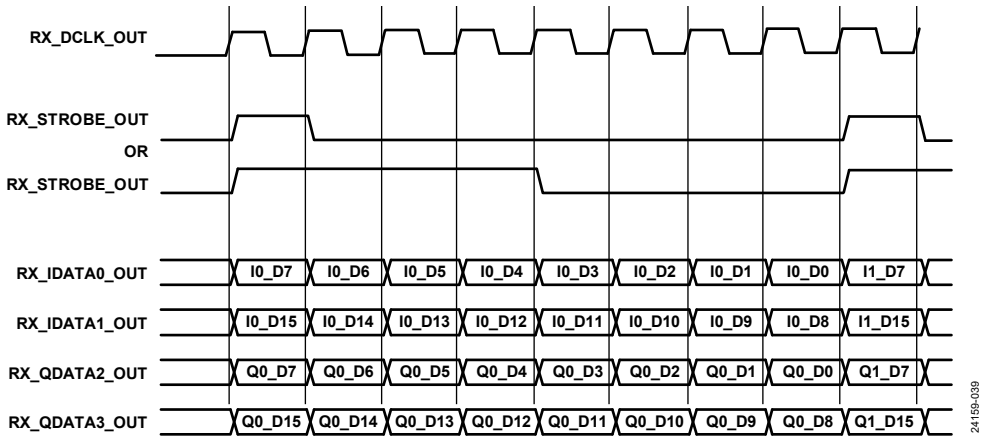


Figure 42. Four-Lane Mode Receive CSSI Timing for 16-Bit I/Q Data Sample (MSB First)

The four-lane mode CSSI transmit interface of each channel (Tx1 and Tx2) is a 7-wire digital interface consisting of:

- TX_DCLK_IN: is an input clock synchronized to the data and strobe inputs.
- TX_STROBE_IN: is an input signal indicating the first bit of the serial data sample.
- TX_IDATA0_IN: is an input serial data stream of I sample low byte.
- TX_IDATA1_IN: is an input serial data stream of I sample high byte.
- TX_QDATA2_IN: is an input serial data stream of Q sample low byte.
- TX_QDATA3_IN: is an input serial data stream of Q sample high byte.
- TX_DCLK_OUT: is an optional output reference clock that is provided to the baseband processor to generate all the above signals, the baseband processor can also RX_DCLK_OUT as the reference clock when its clock rate is equal with transmit SSI clock rate.

Figure 43 illustrates the transmit CSSI interface (Tx1 and Tx2) for a four-lane format with MSB first configuration.

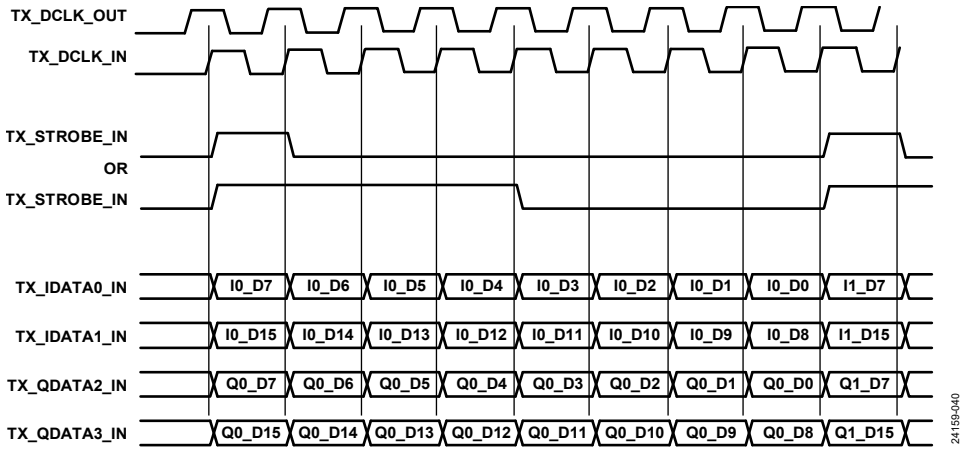


Figure 43. Four-Lane Mode Transmit CSSI Timing for 16-Bit I/Q Data Sample (MSB First)

Transmit and Receive CSSI Using DDR Clock

Transmit and receive CSSI can be operated in either SDR or DDR data transfer.

Figure 44 illustrates the Rx CMOS SSI interface with DDR clock in relation with strobe/data. Each edge of the clock (positive and negative) corresponds to a data sample. The RX DDR Clock can be generated in phase with the data/strobe or delayed quarter cycle of the clock period, the optional delayed clock helps to ease the timing interface of the baseband processor to meet the setup/hold on the baseband processor).

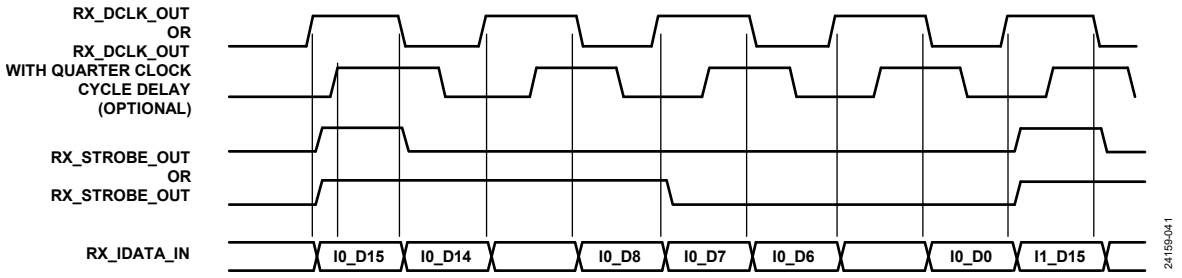


Figure 44. Receive CSSI DDR Clock Relation with Strobe/Data

Figure 45 illustrates the transmit CMOS SSI interface with DDR clock in relation with strobe/data, with respect to ADRV9001. Each edge of the clock (positive and negative) samples the corresponding strobe/data sample based on the interface setup/hold timing.

When the baseband processor drives out the transmit SSI clock, strobe and data to ADRV9001, the output DDR clock can be in-phase with the strobe/data or delayed quarter cycle of the clock period, it's up to the user, but the relation between transmit DDR clock and strobe/data must meet the ADRV9001 setup and hold timing specification.

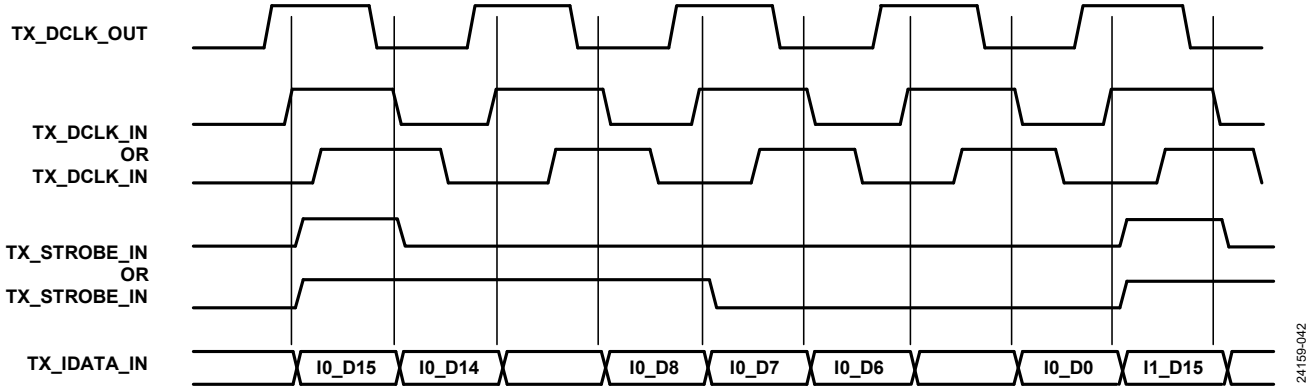


Figure 45. Transmit CSSI DDR Clock Relation with Strobe/Data

Figure 46 and Figure 46 illustrate the timing diagram example for four-lane mode receive, transmit CSSI with DDR clock, 16-bit I/Q sample.

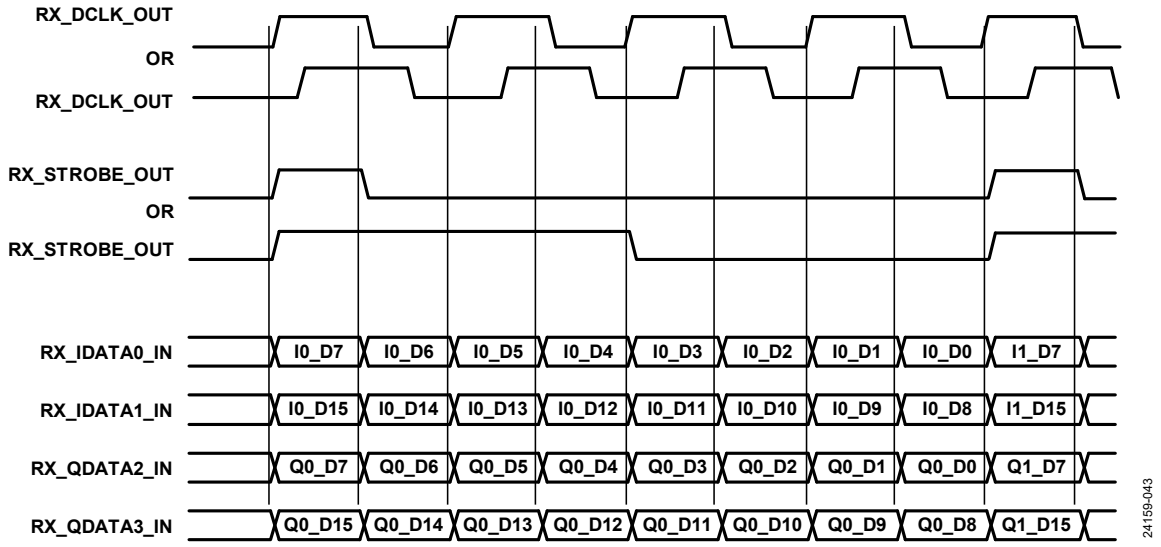


Figure 46. Four-Lane Mode Receive CSSI DDR Timing for 16-Bit I/Q Data Sample

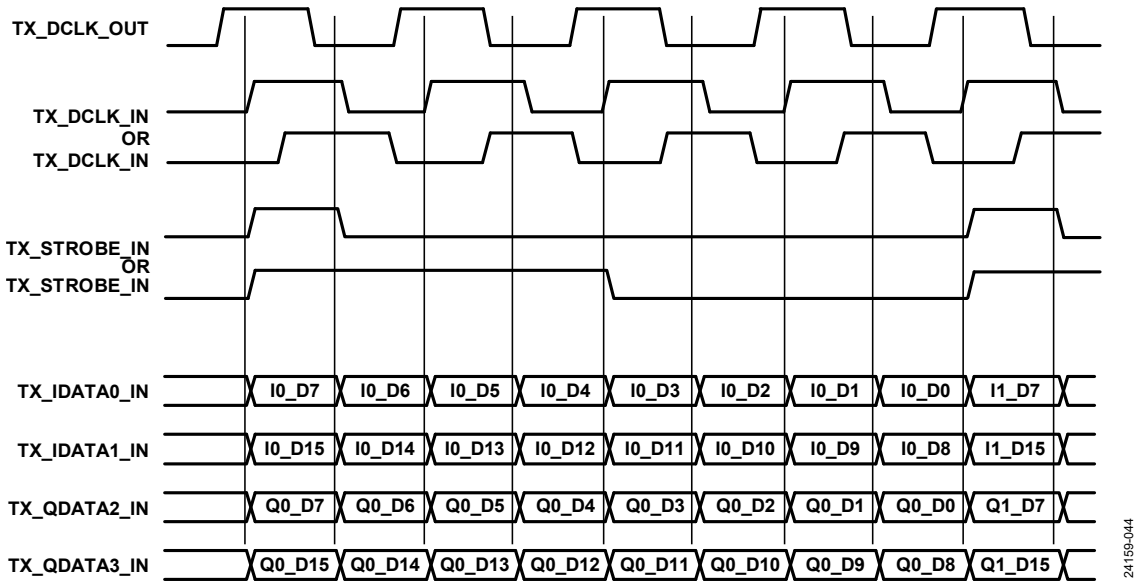


Figure 47. Four-Lane Mode Transmit CSSI DDR Timing for 16-Bit I/Q Data Sample

LVDS SYNCHRONOUS SERIAL INTERFACE (LVDS-SSI)

Receive LSSI Interface

The LSSI receive interfaces of each channel (Rx1 and Rx2) are a 8-wire LVDS interface consisting of:

- RX_DCLK_OUT (±): is a differential output clock.
- RX_STROBE_OUT (±): is a differential output signal indicating the first bit of the serial data sample.
- RX_IDATA_OUT (±): is a differential output serial I data stream.
- RX_QDATA_OUT (±): is a differential output serial Q data stream.

Receive LSSI Interface with Separate Lanes for I and Q

Figure 48 illustrates the receive LSSI interface (Rx1 and Rx2) for a 16-bit I/Q data sample with MSB first configuration. Figure 48 illustrates the receive LSSI interface for a 12-bit I/Q data sample.

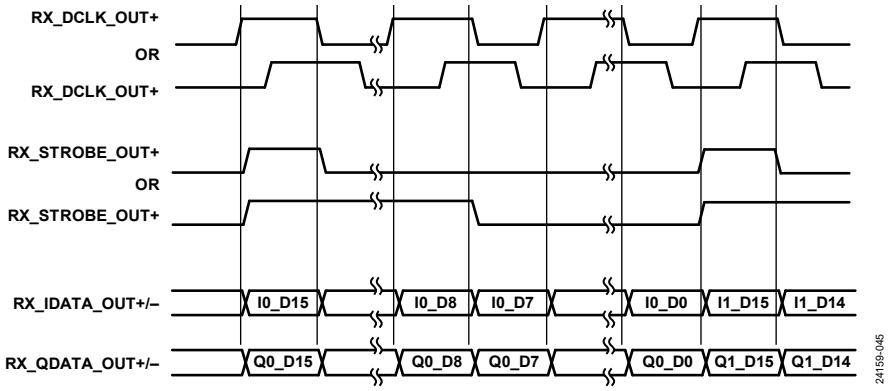


Figure 48. Receive LSSI Timing for 16-Bit I/Q Data Sample over Two Lanes (MSB First)

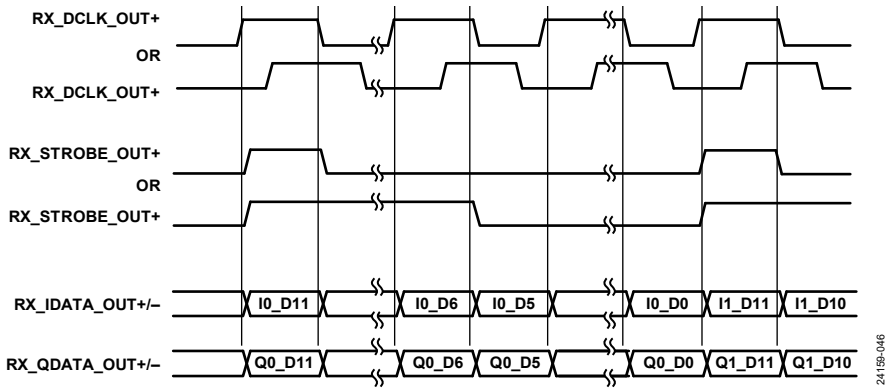


Figure 49. Receive LSSI Timing for 12-Bit I/Q Data Sample over Two Lanes (MSB First)

The RX_STROBE signal is aligned with the first bit of the serialized data (I and Q), and can be configured to be high:

- For a half clock cycle at the start of I and Q sample transmit. In the case of a 16-bit data sample, RX_STROBE is high for a half clock cycle and low for a half and 15 clock cycles. In the case of a 12-bit data sample, RX_STROBE is high for a half clock cycle and low for a half and 11 clock cycles.
- For half of I and Q data duration. In the case of a 16-bit data sample, the RX_STROBE is high for 4 clock cycles, and low for 4 clock cycles (Q data sample). In the case of a 12bit data sample, the RX_STROBE is high for 3 clock cycles and low for 3 clock cycles.

In 12-bit I/Q mode, 16-bit samples from the receive datapath are cut to 12 bits for LSSI, a configurable option for the user to choose the 12-bit is from LSB or MSB of the 16-bit sample data.

Receive LSSI Interface with One Lane for I and Q

In this mode, only one lane is used to transfer I and Q data samples. The I/Q data bits are serialized with configurable I or Q first and MSB or LSB first. The STROBE signal can be configured to high for a half clock cycle to indicate the start of I and Q symbols or for half of I and Q data duration to distinguish between I data and Q data.

Figure 50 illustrates the one-lane receive LSSI interface (Rx1 and Rx2) for a 16-bit I/Q data sample with I sample and MSB first configuration.

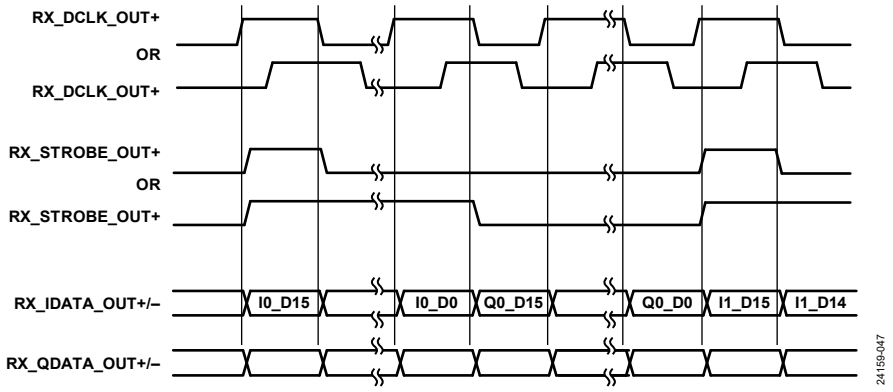


Figure 50. Receive LSSI Timing for 16-Bit I/Q Data Sample over One-Lane (I and MSB First)

Transmit LSSI Interface

The transmit LSSI interface of each channel (Tx1 and Tx2) is an 8-wire digital interface consisting of:

- TX_DCLK_IN (±): is a differential input clock synchronized to the data and strobe inputs.
- TX_STROBE_IN (±): is a differential input signal indicating the first bit of the serial data sample.
- TX_IDATA_IN (±): is a differential input serial I data stream.
- TX_QDATA_IN (±): is a differential input serial Q data stream.

An additional port might be used as a reference clock for the baseband processor to generate above Transmit LSSI clock, Strobe and Data signal, the user could use RX1_DCLK_OUT or RX2_DCLK_OUT as a reference clock if these clock frequencies are equal to the TX clock frequency.

An optional LVDS port (alternative function of Digital GPIO) can also be configured as an output LVDS pad used as a reference clock TX_DCLK_OUT (±) for the baseband processor, the user could use TX_DCLK_OUT to generate above LSSI clock, strobe and data signal.

Transmit LSSI Interface with Separate Lanes for I and Q

Figure 51 illustrates the transmit LSSI interface (Tx1 and Tx2) for a 16-bit I/Q data sample with MSB first configuration.

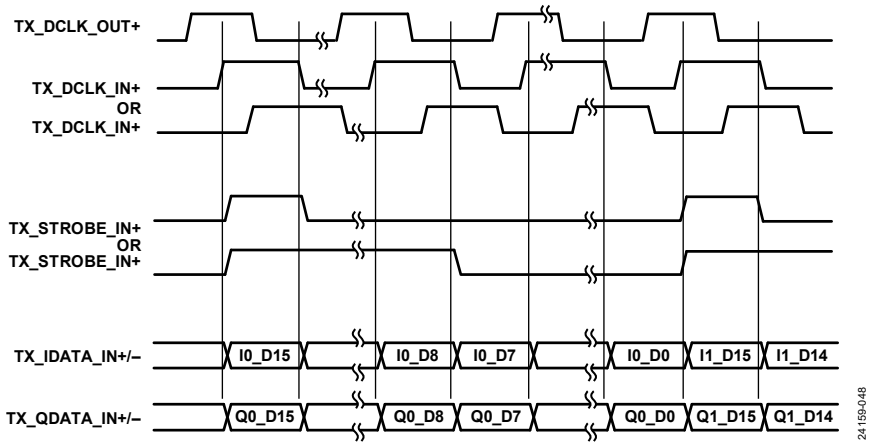


Figure 51. Transmit LSSI Timing for 16-Bit I/Q Data Sample on Separate Lanes

Figure 52 illustrates the Transmit LSSI interface (Tx1 and Tx2) for a 12-bit I/Q data sample with MSB first configuration.

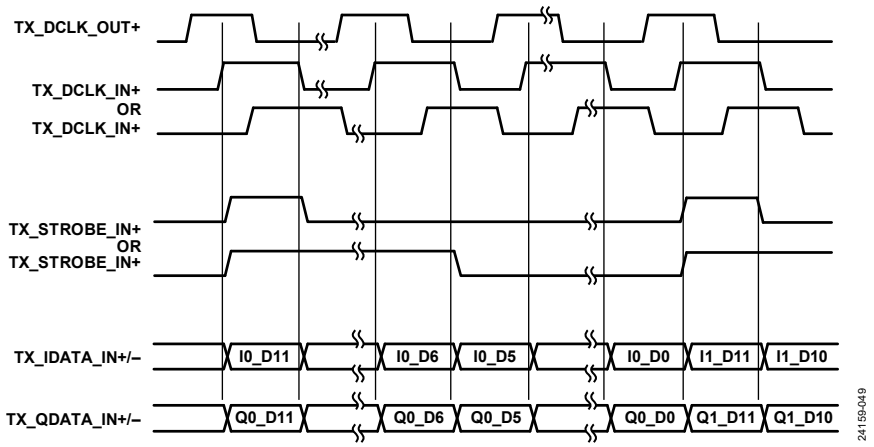


Figure 52. Transmit LSSI Timing for 12-Bit I/Q Data Sample on Separate Lanes

The TX_STROBE signal is aligned with the first bit of the serialized data (I & Q), and can be configured to be high:

- For a half clock cycle at start of I and Q sample transmit. In the case a 16-bit data sample, the TX_STROBE is high for a half clock cycle and low for a half and 15 clock cycles. In the case of a 12-bit data sample, the TX_STROBE is high for a half clock cycle and low for a half and 11 clock cycles.
- For half of I and Q data duration. In the case of a 16-bit data sample, the TX_STROBE is high for 4 clock cycles, and low for 4 clock cycles (Q data sample). In the case of a 12-bit data sample, the TX_STROBE is high for 3 clock cycles and low for 3 clock cycles.

In 12-bit I/Q mode, 12-bit samples from LSSI are extended to 16 bits by padding four bits zero in LSB for the following transmit datapath process.

Transmit LSSI Interface with One Lane for I and Q

In this mode, only one lane is used to transfer I and Q data samples. The I/Q data bits can be deserialized with configurable I or Q first and MSB or LSB first. The STROBE signal can be configured to high for a half clock cycle to indicate the start of I and Q symbols or for half of I and Q data duration to distinguish when I Data and Q Data.

Figure 53 illustrates the one lane LSSI interface (Tx1 and Tx2) for a 16-bit I/Q data sample with I sample and MSB first configuration.

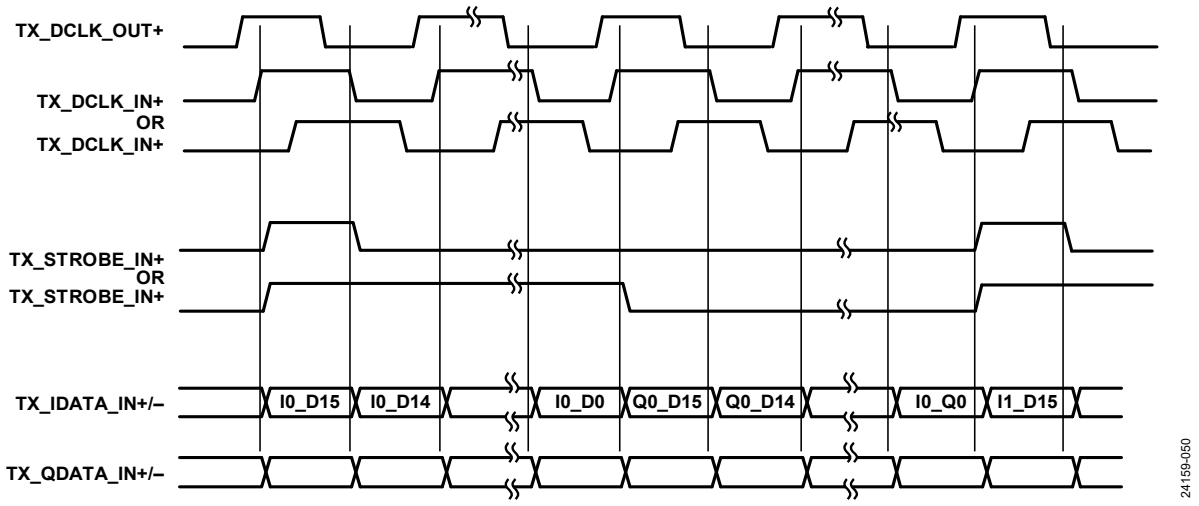


Figure 53. Transmit LSSI Timing for 16-Bit I/Q Data Sample Sharing One Lane

Receive LSSI Interface with 2×, 4×, and 8× Data Clock Rates

ADRV9001 receive LSSI supports the 2 times, 4 times, or 8 times of the data clock rate for some applications, which is similar with the Receiver CSSI mode, refer the timing diagrams in Receive CSSI Interface with 2×, 4×, and 8× Data Clock Rates.

ENHANCED RX SSI MODE

The Rx SSI LVDS two lane and CMOS one lane mode have two enhanced modes to support 22bit data samples and 15 bit data samples in I/Q mode.

In 22bit data samples case, the 32 bit interface data bus has the following fields:

- 22bits of data sample (I/Q from Rx Data path: unrounded data samples: RxDataPathI/Q[21:0])
- 1bit = 0 (Constant)
- 1bit Gain Change (Slicer or Index Gain Change flag)
- 8bits Gain (Slicer or Index Gain)

which produce the following Interface data with 32bit data format for the CMOS and LVDS SSI:

- LVDS 32bit: 2 lanes (I & Q) of 32 bit each
  - LSSI_DATA_I/Q [31:0] = {RxDataPathI/Q[21:0], b0, Gain_Change, Gain [7:0]}
- CMOS 64bit: 1 Lane (I & Q)
  - CSSI_DATA [63:0] = {RxDataPathI[21:0] , b0, Gain_Change, Gain[7:0],
    RxDataPathQ[21:0] , b0, Gain_Change, Gain[7:0] }

In 15bit data samples case, the 16bit interface data bus has the following fields:

- 15bits of data sample (15 bit I/Q rounded from 22 bit Rx Data path samples)
- 1bit Gain Change (Slicer or Index Gain Change flag)

which produce the following Interface data with 16 bit data format for the CMOS and LVDS SSI:

- LVDS 16bit: 2 lanes (I & Q) of 16 bit each
  - LSSI_DATA_I/Q [15:0] = {RxDataPathI/Q[21:0], b0, Gain_Change, Gain [7:0]}
- CMOS 32bit: 1 Lane (I & Q)
  - CSSI_DATA [31:0] = {RxDataPathI rounded[14:0] , Gain_Change, RxDataPathQ rounded[14:0], Gain_Change }

Some of the other basic configuration modes, such as MSB/LSB first option, I or Q first option (for CMOS 1 lane), Long/Short strobe option are similar to previous SSI LVDS/CMOS 16bit operation.

POWER SAVING FOR LSSI

In TDD mode, the LVDS SSI pads can be powered down/up dynamically based on the Tx_Enable and Rx_Enable level to save power, three LSSI power down modes are defined for different user's requirement which are shown in Table 17. API adi_adrv9001_Ssi_PowerDown_Set is used to set the power down mode for specified channel.

Table 17 LSSI power down mode

LSSI Power Down Mode	Description
ADI_ADRV9001_SSI_POWER_DOWN_DISABLED	All SSI PADS are powered up in PRIMED
ADI_ADRV9001_SSI_POWER_DOWN_MEDIUM	RX_DCLK_OUT and TX_DCLK_OUT SSI pads are powered up, TX_DCLK_IN and all Tx/Rx STROBE and DATA SSI pads are powered down in PRIMED
ADI_ADRV9001_SSI_POWER_DOWN_HIGH	All SSI pads are powered down in PRIMED

SSI TIMING PARAMETERS

Receive SSI and transmit SSI timing diagram are shown in Figure 54 and Figure 55. The preliminary timing specification for CMOS SSI is described in Table 18 and the preliminary timing specification for LVDS SSI is described in Table 19.

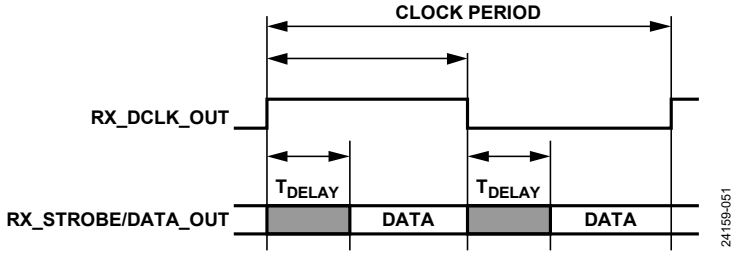


Figure 54. Receive SSI Timing Diagram

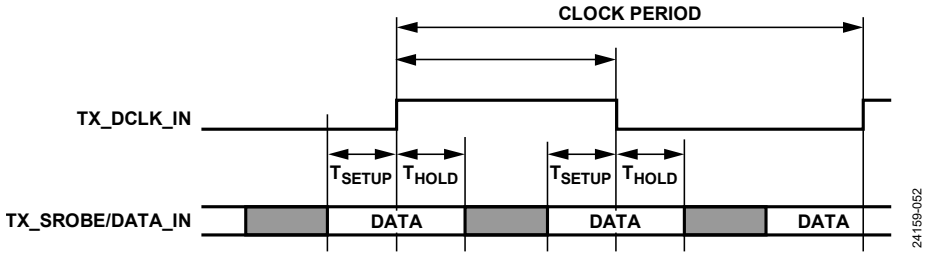


Figure 55. Transmit SSI Timing Diagram

Table 18. CMOS SSI Timing Specification

CMOS SSI	Timing	Description
CMOS Rx t_{DELAY} Maximum	4.5 ns	Clock to strobe/data delay
CMOS Tx t_{SETUP} Minimum	2 ns	Strobe/data setup to clock
CMOS Tx t_{HOLD} Minimum	2 ns	Strobe/data hold after clock

Table 19. LVDS SSI Timing Specification

LVDS SSI	Timing	Description
Rx t_{DELAY} (Maximum)	200 ps	Clock to strobe/data delay
Tx t_{SETUP} (Minimum)	250 ps	Strobe/data setup to clock
Tx t_{HOLD} (Minimum)	550 ps	Strobe/data hold after clock

API PROGRAMMING

The ADRV9001 SSI configuration is performed in chip initialization stage and based on the following data structure.

```
typedef struct adi_adrv9001_SsiConfig
{
    adi_adrv9001_SsiType_e              ssiType;
```

```
    adi_adrv9001_SsiDataFormat_e        ssiDataFormatSel;
    adi_adrv9001_SsiNumLane_e           numLaneSel;
    adi_adrv9001_SsiStrobeType_e        strobeType;
    uint8_t                             lsbFirst;
    uint8_t                             qFirst;
    adi_adrv9001_SsiTxRefClockPin_e     txRefClockPin;
    bool                                lvdsIBitInversion;
    bool                                lvdsQBitInversion;
    bool                                lvdsStrobeBitInversion;
    uint8_t                             lvdsUseLsbIn12bitMode;
    bool                                lvdsRxClkInversionEn;
    bool                                cmosDdrPosClkEn;
    bool                                cmosClkInversionEn;
    bool                                DdrEn;
    bool                                rxMaskStrobeEn;
} adi_adrv9001_SsiConfig_t;
```

In the data structure, the previously mentioned SSI modes are defined for each Tx/RX channel, Table 20 lists the SSI configuration parameters and some default values, users can find the detail data structure and enumerator description in API Doxygen help file.

Table 20 SSI Configuration Parameters

Parameter	Type	Description	Note
ssiType	enum	Sets SSI type	
ssiDataFormatSel	enum	Set SSI data format	
numLaneSel	enum	Set SSI number of lanes	
strobeType	enum	Set SSI strobe type	
lsbFirst	uint8_t	Set LSB first	Default '0', MSB first
qFirst	uint8_t	Set Q data first	Default '0', I data first
txRefClockPin	enum	Set TX SSI reference clock output (TX_DCLK_OUT) options	
lvdsIBitInversion	bool	Set LVDS SSI I bit differential pads polarity inversion	Default 'false'
lvdsQBitInversion	bool	Set LVDS SSI Q bit inversion	Default 'false", Rx SSI ignores this field, I/Q lanes share the configuration of "lvdsIBitInversion"
lvdsStrobeBitInversion	bool	Set LVDS SSI strobe bit inversion	Default 'false"
lvdsUseLsbIn12bitMode	uint8_t	Set LVDS 12 bit mode	Default '0', LVDS SSI uses 16 bit mode
lvdsRxClkInversionEn	bool	Set LVDS RX SSI clock inversion enable	Default 'false'
cmosDdrPosClkEn	bool	Set CMOS DDR positive clock enable	Default 'false'
cmosClkInversionEn	bool	Set CMOS DDR clock inversion enable	Default 'false'
DdrEn	bool	Set DDR mode enable	
rxMaskStrobeEn	bool	Set Rx Strobe Mask, mask the Rx SSI Strobe when interface rate is multi times of sample rate, refer Figure 36	Default 'false'

Figure 44 illustrates the Rx CMOS SSI interface with DDR clock in relation with strobe/data. To make sure the BBIC can get the best setup/hold timing margin for RX CMOS DDR SSI, with Table 20 RX CMOS DDR relative default SSI configurations (cmosDdrPosClkEn=false, cmosClkInversionEn=false), the Rx CMOS SSI output clock/strobe/data phase timing diagram is shown in Figure 56. We would recommend users using this default RX CMOS SSI DDR configuration.

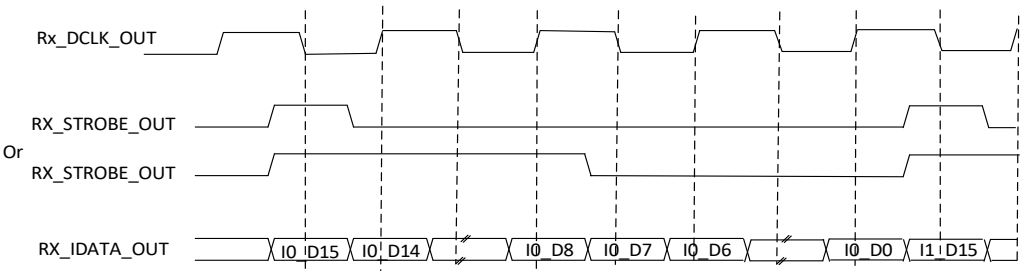


*Figure 56 RX CMOS DDR SSI Default(**Recommend)** output (cmosDdrPosClkEn=**False**, cmosClkInversionEn=**False**)*

Users can also manually edit the CMOS DDR relative configurations based on their BBIC requirements, Figure 57, Figure 58, Figure 59 shows the corresponding Rx output Clock/Strobe/Data timing diagram with different CMOS DDR configuration.

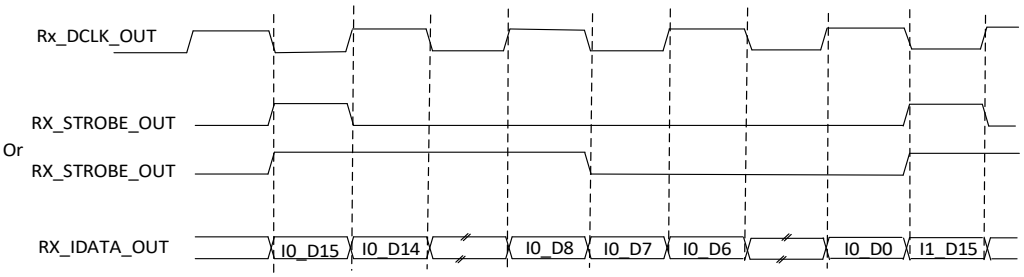


*Figure 57 RX CMOS DDR SSI output (cmosDdrPosClkEn=**True**, cmosClkInversionEn=**False**)*

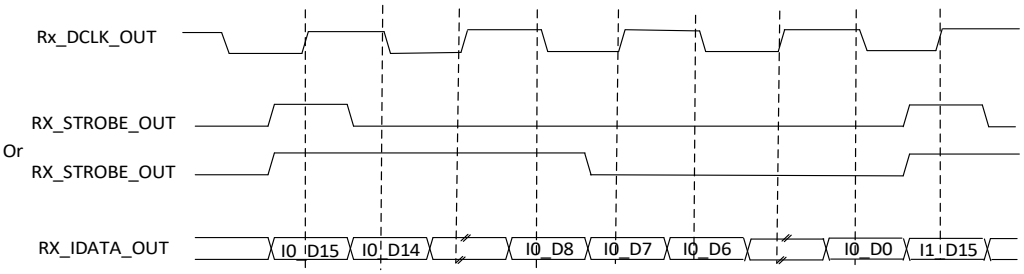


*Figure 58 RX CMOS DDR SSI output (cmosDdrPosClkEn=**False**, cmosClkInversionEn=**True**)*

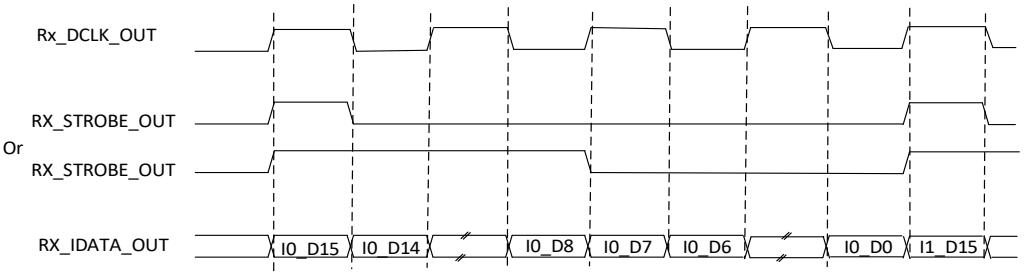


*Figure 59 RX CMOS DDR SSI output (cmosDdrPosClkEn=**True**, cmosClkInversionEn=**True**)*

A set of API commands are provided to set and inspect the SSI test/debug functions, which are summarized in Table 21.

Table 21. SSI Test/Debug API List

SSI Function Name	Description
adi_adrv9001_Ssi_Rx_TestMode_Configure	Configures the SSI test mode for the specified Rx channel
adi_adrv9001_Ssi_Tx_TestMode_Configure	Configures the SSI test mode for the specified Tx channel
adi_adrv9001_Ssi_Tx_TestMode_Status_Inspect	Inspects the SSI test mode status for the specified Tx channel
adi_adrv9001_Ssi_Loopback_Set	Set the enabledness of Rx to Tx SSI interface loopback
adi_adrv9001_Ssi_Delay_Configure	Programs the SSI delay configuration
adi_adrv9001_Ssi_Delay_Inspect	Gets the SSI delay configuration from ADRV9001 device
adi_adrv9001_Ssi_PowerDown_Set	Set the power down mode for the specified channel and SSI type

CSSI/LSSI TESTABILITY AND DEBUG

ADRV9001 SSI has built-in test pattern generator and test pattern checker which can help users to quickly test and debug the SSI interface between the ADRV9001 and the baseband processor. Figure 60 illustrates the ADRV9001 SSI testability and debug diagram with a baseband processor.

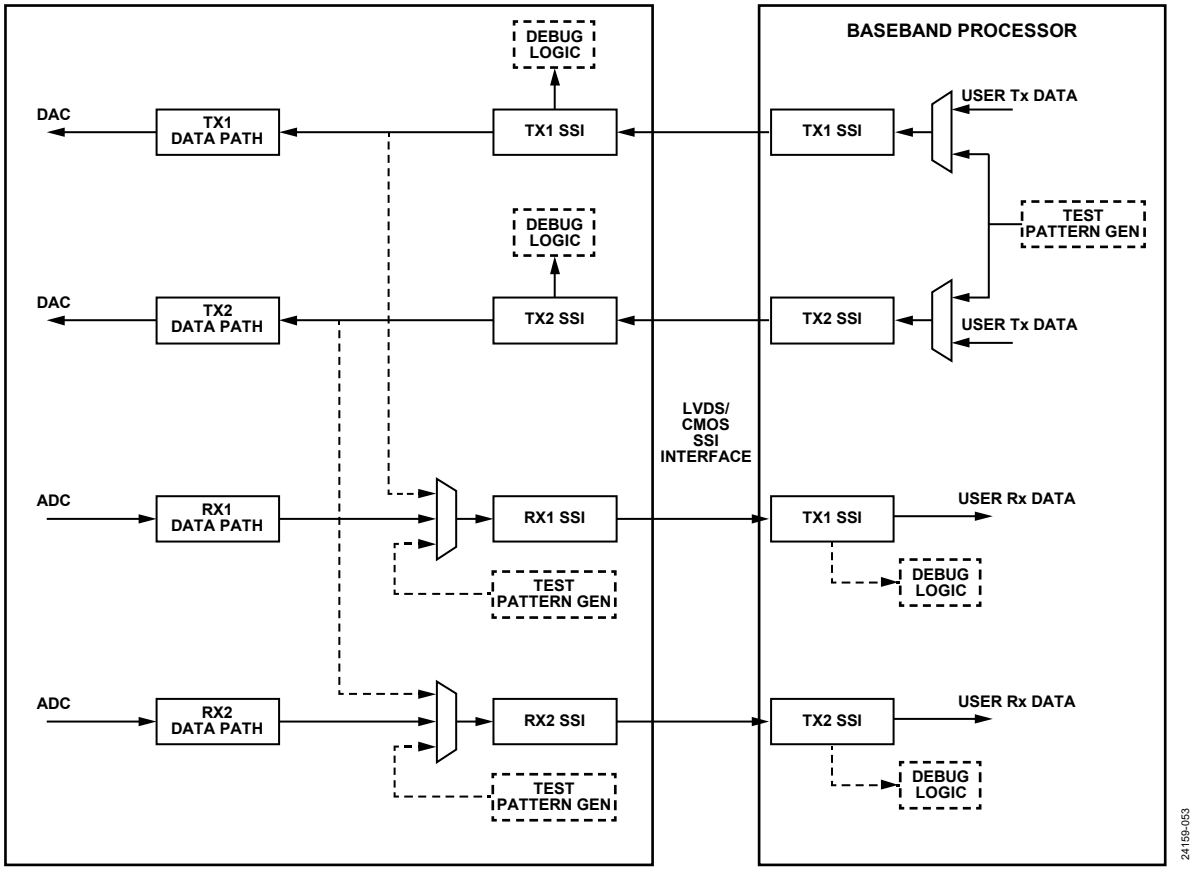


Figure 60. ADRV9001 SSI Testability and Debug Diagram

The ADRV9001 receive SSI can replace the receive channel data with fixed pattern, ramp or PRBS (LSSI only) pattern and to baseband processor if enable the receive debug function by calling adi_adrv9001_Ssi_Rx_TestMode_Configure(), users can check the specified test pattern at their SSI output to test if the receive SSI from ADRV9001 to BBIC works well.

The data structure adi_adrv9001_RxSsiTestModeCfg_t is used to enable and configure the RX SSI test pattern.

```
typedef struct adi_adrv9001_RxSsiTestModeCfg
{
   adi_adrv9001_SsiTestModeData_e testData;   /*!< Type of data to transmit over SSI */
   uint32_t fixedDataPatternToTransmit;       /*!< Value of Fixed pattern to transmit over interface. For various SSI data format:
                        CMOS: Pattern is truncated to bit3  - bit0 value is transmitted on RxSSI I and Q each nibble  (where applicable)
                        LVDS: Pattern is truncated to bit15 - bit0 value transmitted on RxSSI I and Q (where applicable) */
} adi_adrv9001_RxSsiTestModeCfg_t;
```

The enum adi_adrv9001_SsiTestModeData_e enable and choose the specified test pattern and lists in Table 22.

Table 22 Definition of adi_adrv9001_SsiTestModeData_e

ENUM	Description
ADI_ADRV9001_SSI_TESTMODE_DATA_NORMAL	No test mode enabled
ADI_ADRV9001_SSI_TESTMODE_DATA_FIXED_PATTERN	Fixed patten mode
ADI_ADRV9001_SSI_TESTMODE_DATA_RAMP_NIBBLE	nibble ramp mode (CSSI only) ,

	I Q data is same and each 4bits of the samples keep ramping (e.g. sample0-0x0000, sampe1-0x1111, sample2-0x2222, …)
ADI_ADRV9001_SSI_TESTMODE_DATA_RAMP_16_BIT	16 bits ramp mode, I Q data is same, 16bits of the samples keep ramping (e.g. sample0-0x0000, sample1-0x0001, sample2-0x0002,…)
ADI_ADRV9001_SSI_TESTMODE_DATA_PRBS15	PRBS15 mode (LSSI only)
ADI_ADRV9001_SSI_TESTMODE_DATA_PRBS7	PRBS7 mode (LSSI only)

Enhanced Rx SSI 32bit mode, the test pattern generation debug modes are similar to existing 16bit SSI interface debug operations. In CMOS mode, for fixed pattern, the RX SSI 64 bits test pattern = { fixedDataPatternToTransmit [15:0], fixedDataPatternToTransmit [15:0], fixedDataPatternToTransmit [15:0], fixedDataPatternToTransmit [15:0] }. For 16bit ramp mode, the Rx SSI 64bits test pattern = {RampPattern[15:0], RampPattern[15:0], RampPattern[15:0], RampPattern[15:0] }.

Similarly, in LVDS mode, the Rx SSI 32 bits fixed test pattern for I and Q is { fixedDataPatternToTransmit [15:0], fixedDataPatternToTransmit [15:0]} , and ramp patter for I and Q is {RampPattern[15:0], RampPattern[15:0] }. PRBS pattern will not be supported in this enhanced Rx SSI 32bit mode.

The ADRV9001 transmit SSI has a ramp and PRBS (LSSI only) pattern checker, users can configure ADRV9001 TX SSI test mode and transmit ramp or PRBS pattern via SSI to ADRV9001 to verify if SSI works well, or users can also transmit a fixed pattern and configure the ADRV9001 with the specified fixed pattern to verify the SSI work status. Users can call API adi_adrv9001_Ssi_Tx_TestMode_Configure to enable and configure the test mode, and transmit the corresponding test patterns to ADRV9001 via Tx SSI, then call the adi_adrv9001_Ssi_Tx_TestMode_Status_Inspect to get the ADRV9001 TX SSI test mode status.

Similarly, data structure adi_adrv9001_TxSsiTestModeCfg to enable and configure the ADRV9001 TX SSI test pattern checker. BBIC transmits relative test patterns and the format should follow the description in Table 22. For the fixed pattern mode transmit, BBIC should put bit31-16 of fixedDataPatternToCheck on TX SSI I data and bit15-0 on Q data.

```
typedef struct adi_adrv9001_TxSsiTestModeCfg
{
   adi_adrv9001_SsiTestModeData_e testData;   /*!< Type of data to receive over SSI and check */
   uint32_t fixedDataPatternToCheck;          /*!< Value of Fixed pattern to check against pattern received over interface */
} adi_adrv9001_TxSsiTestModeCfg_t;
```

The ADRV9001 transmit SSI data output can be loopback to receive SSI data input by API adi_adrv9001_Ssi_Loopback_Set when transmit and receive SSI runs at same clock rate, users can use their pattern generator and checker to verify if the whole system SSI works well. Users should be noticed that both ADRV9001 TX and RX radio state should be in “RF_ENABLED” state (to make sure the TX/RX SSI is enabled) when set the SSI loopback test function.

As mentioned previously, the SSI clock, strobe and data have programmable delay, the delay can be configured by adi_adrv9001_Ssi_Delay_Configure, the unit for the SSI delay is step. This helps users to meet the timing spec that described in SSI Timing Parameters.

MICROPROCESSOR AND SYSTEM CONTROL

ADRV9001 supports quick configuration from idle states of operation and quick transition between receive and transmit states. Those transitions are handled by internal blocks called stream processors. Stream processor is a processor within the ADRV9001 device assigned to perform a series of configuration tasks upon an external request. Upon a request from the user, the stream processor performs a series of actions defined in the image loaded into the ADRV9001 during initialization process.

The stream processor therefore has streams (series of tasks) for:

- Tx1 Enable/Tx1 disable
- Tx2 Enable/Tx2 disable
- Rx1 Enable/Rx1 disable
- Rx2 Enable/Rx2 disable

Enabling and disabling paths is done typically using pins, however can also be controlled over the SPI bus using API command. The stream is not limited to path enabling events and can react to other events such as a DGPIO input signal.

ADRV9001 is flexible in its configuration, and therefore, the stream is flexible. In the same way as the initialization structures change with profile, so the stream processor image must change with configuration, for example, the stream that enables Rx1 differs depending on whether a narrowband or a wideband setup is chosen. For this reason, it is necessary to use a stream image for each configuration of the device. In this way, when the user saves configuration files (.c) using the ADRV9001 TES, a stream image is also saved automatically. This stream file should then be used when using these configuration files.

Figure 61 describes the general ecosystem of ADRV9001. On the right-hand side (data side), ADRV9001 interfaces with the BBIC and on the left-hand side (antenna side), it interfaces with the RF components. The following section describes control of the ADRV9001 datapaths.

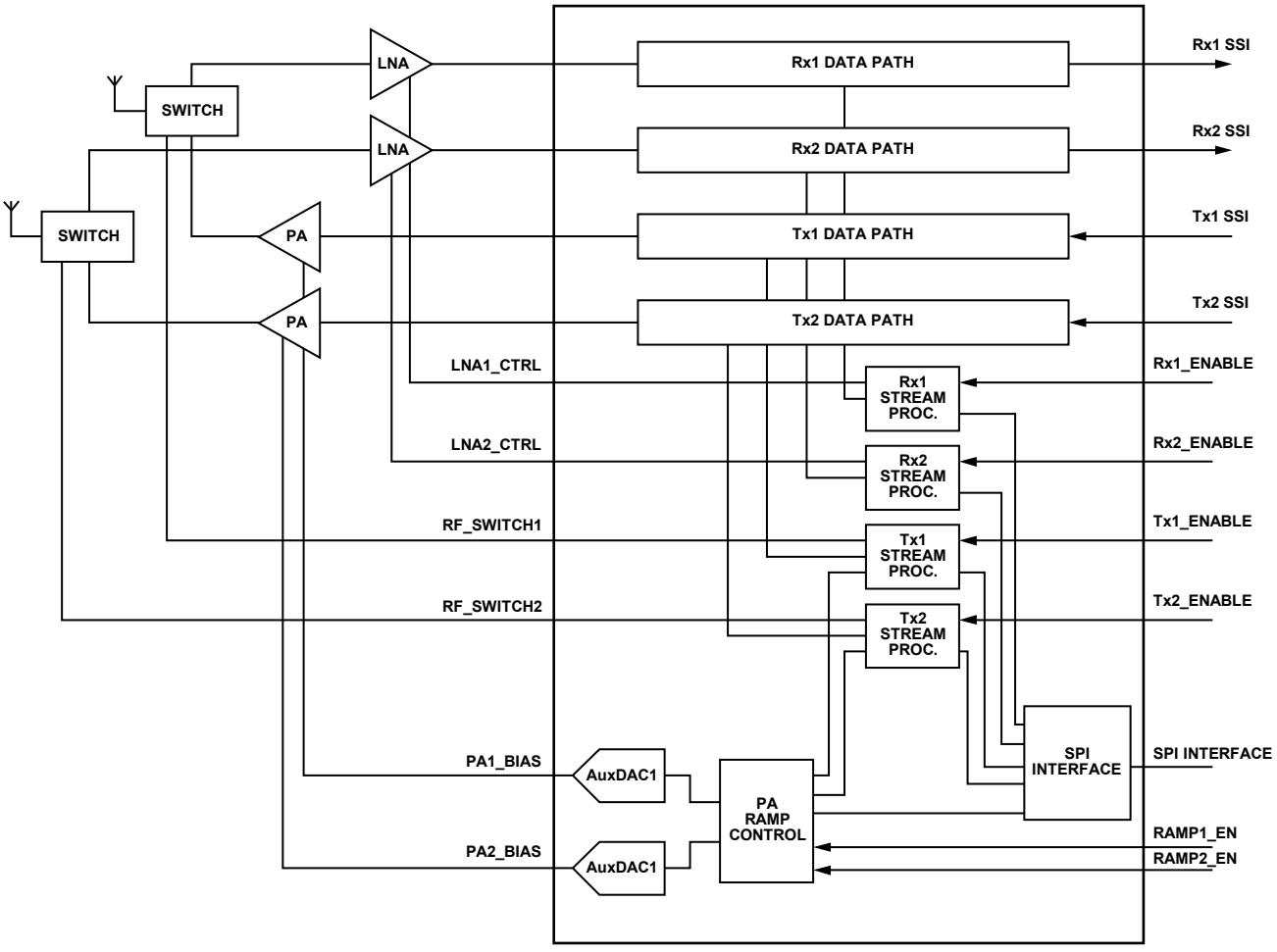


Figure 61. Datapath Control Signals

SYSTEM CONTROL

The datapaths within the ADRV9001 can be controlled either through the API or through ENABLE pin controls. In the case of API control, this is reliant on the SPI communication bus and thus for critical time alignment of powering on/off chains, pin control is recommended. Each datapath is independently controlled, with the following enable signals defined:

Table 23. Data Path Enable Signals.

Enable Signal	Data Path
RX1_ENABLE	Rx1 datapath
RX2_ENABLE	Rx2 datapath
TX1_ENABLE	Tx1 datapath
TX2_ENABLE	Tx2 datapath

For ADRV9001 to receive and react to control signals it must be moved to the primed state. The primed state indicates that the system is ready for operation when the transmit and receive channels are enabled by the user. After the channel is primed, in order to start transmit or reception activities, it must be further transitioned from the primed state to the RF_ENABLED state. This can be accomplished by a set of API calls.

PIN Mode

1. Call adi_adrv9001_Radio_ChannelEnableMode_Set( ) to set the PIN mode.
2. Toggle corresponding ENABLE pin to transition the channel to the RF_ENABLED state.

SPI Mode

1. Call adi_adrv9001_Radio_ChannelEnableMode_Set( ) to set the SPI mode.
2. Call adi_adrv9001_Radio_Channel_EnableRf( ) to transition the channel to RF_ENABLED state.

After pin or SPI/API mode is executed, the ADRV9001 enables the requested channels. The channels remain active until further instruction through a pin command or SPI/API command.

TIMING PARAMETERS CONTROL

ADRV9001 has integrated stream processors to handle various external and internal events that are required to be serviced in real time. Those stream processors coupled with programmable delayed enable modules relieve the system firmware (running on integrated microprocessor) from managing all those critical events by providing a quick and parallel response to external and internal events. This configuration allows the 4 channels (Tx1, Tx2, Rx1, and Rx2) to operate independently from each other by using their own dedicated stream processor.

ADRV9001 can support different applications, each with its own unique challenges. A set of programmable timing parameters for both transmitter and receiver are provided to users to meet their particular timing requirements in various TDD applications. Understanding the ADRV9001 timing parameters is crucial to ensure all TDD events taking place at an accurate time order, as expected by the user. In addition, configuring timing parameters in an optimal way by taking advantage of the multiple power saving modes ADRV9001 offered could improve the overall system power consumption performance significantly.

Timing Definition

Before explaining the typical values for each delay, this chapter will attempt to visualize and explain each delay and how it pertains to the physical component. Inspect Figure 62 for a visual representation of the most pertinent delays in a system. Note that the length of each arrow ***does not*** represent the length of each delay in time, these are just representations of what each delay is in terms of the hardware. Timing information will be provided later in this chapter.

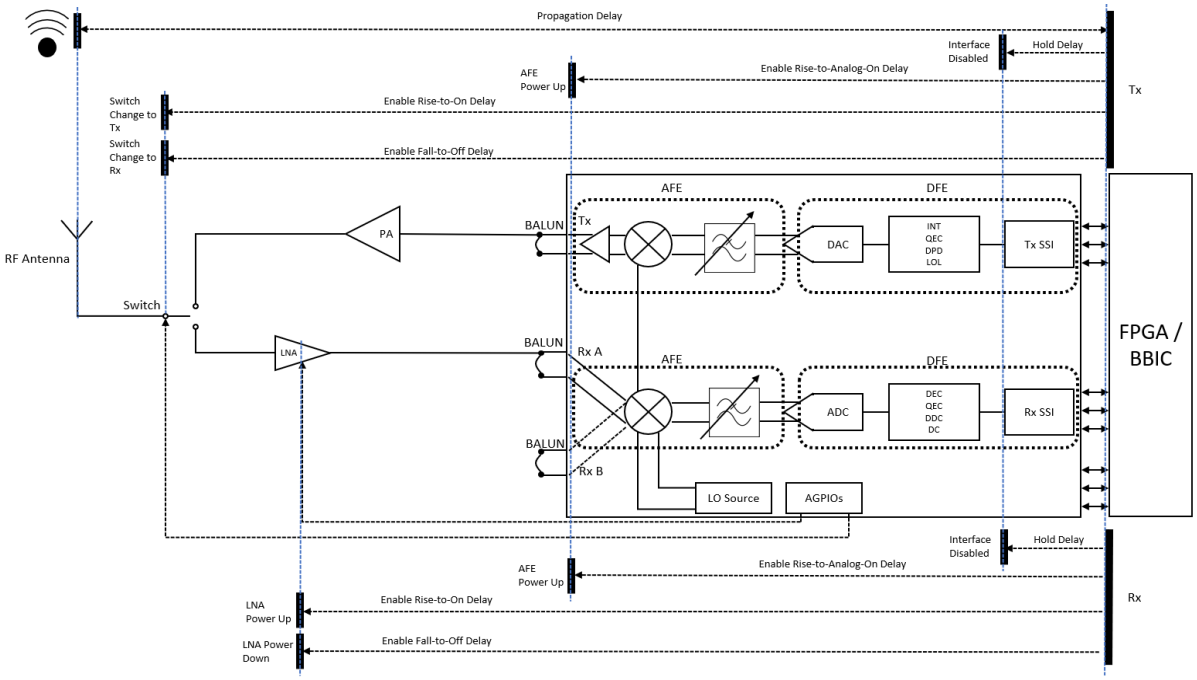


Figure 62: Visualization of Timing Parameters

Broadly speaking, the delays present in the system can be described as follows:

- ***Enable Setup Delay*** is the time taken for ADRV9001 to power up its analog front end. This may or may not include PLL tuning time based on the use case, for example, when Tx and Rx share the same IO but at different frequency, PLL tuning is needed at the frame boundary.
- ***Propagation Delay*** is the delay of data from antenna to RF interface in either direction. Given that this delay encompasses external components it is impossible to determine a priori what this delay should be, it simply must be measured. Naturally, this delay is also very setup dependent and board layout dependent. It does not need to be provided to ADRV9001, but it can be used to derive other parameters required by the ADRV9001.
- ***Enable Rise-to-On Delay*** is the delay between TX_ENABLE rising edge and Rx/Tx switch switching to the Tx channel. Conversely, it is also the delay between RX_ENABLE rising edge and the external Rx LNA powering up. It should align with the desired time when the first symbol is on air. Typically, this delay is equal in length to the propagation delay, however if the setup uses Guard Data to envelop the Frame on Air this extra Guard time must also be accounted for. This delay will not apply to every setup, however. Some applications give control of the antenna switch to the user, for example hand-held radio applications often allow users to control whether to Transmit or Receive data on a single antenna. If ADRV9001 is not controlling antenna switch, this parameter is not needed except to determine other parameters.
- ***Enable Rise-to-Analog-On Delay*** is the delay between TX_ENABLE / RX_ENABLE rising edge and analog power up beginning. This is a user defined parameter, the purpose of which is to align the analog power up with the Frame on Air. Note that this parameter is used to aid in saving power in setups where the propagation delay is quite long. If Tx propagation delay is long, the analog power up can be delayed for power saving or to keep Tx AFE powered down during Rx frame, and vice versa. If the propagation delay is small, this should be set to 0. If this parameter is greater than its max bound, the antenna switch / LNA power on time could be delayed.
- ***Enable Guard Delay*** is the guard time at the beginning of the RF frame. This part of the Frame on Air does not contain useful data, instead it is used as a barrier between the end of device setup and start of data transmission. Any distortions or noise applied to the guard data by the transmitter device will not affect error rates at the receiver. Not every application uses guard data, however for those that do it is worth noting that the Guard Delay only accounts for the Guard data at the *beginning* of the frame, not the end. The Guard data at the end of the frame is accounted for with the Hold delay. Guard Delay is reserved for future use, should be set to 0 currently.
- ***Enable Hold Delay*** is the delay between RX_ENABLE falling edge and masking off datapath data sent over interface. Similarly, it is also the delay between the falling edge of TX_ENABLE and the Tx interface being disabled.
- ***Enable Fall-to-Off Delay*** is the delay between RX_ENABLE falling edge and the powering down the external Rx LNA. If ADRV9001 not controlling external Rx LNA power, this parameter can still be used to delay analog power down. ADRV9001

forces it to 0 currently, meaning the external LNA is disabled at the same time as Rx is disabled. It is also the delay between TX_ENABLE falling edge and Rx/Tx switch switching to Rx channel and the Tx AFE powering down. Even if ADRV9001 is not controlling antenna switch, this parameter is still needed to delay analog power down.

- ***Internal Path Delay*** is the delay between the SSI port and the RF Port for either the Tx or Rx signal chains. It does not include any external components, and TES will calculate it automatically for the user. As part of the design of a custom setup, users are advised to measure the entire Propagation Delay of their setup to ensure it is larger than the Internal Path Delay measured by our software.

While the existence of these delays will be common across the Tx and Rx signal chains, their sizes, uses and applications will vary. The following sections will detail how each of these delays presents in the Tx and Rx signal chains respectively, as well as detail design choices that must be made around them.

Transmit Timing Definition

Transmit timing parameters define the events that take place in order from the start of transmission at the ADRV9001 data port to the end of transmission when the transmit burst is sent through the antenna to the air.

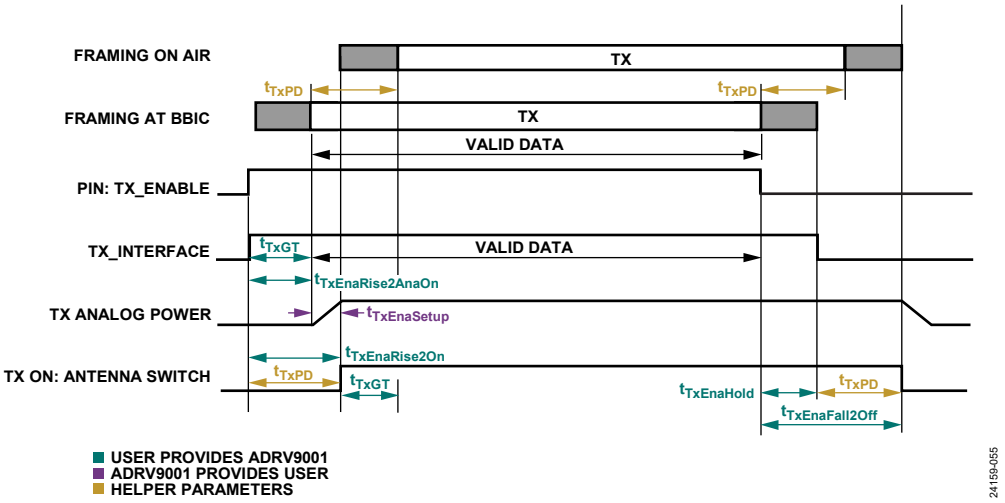


Figure 63. Transmitter Timing Parameters (t_{TxPD} > $t_{TxEnaSetup}$)

As shown in Figure 63, a transmit burst consists a series of valid transmit data with user's option of padding guard data at the beginning and end of the valid data. Based on the timing parameters configured by the user, it is user's decision if full or partial of the guard data should be transmitted to the air and user's responsibility to make sure that the guard data usage is compliant with the standard requirement. The transmit enable pin is controlled by user to signal the start and end of a transmit burst at the data port. Based on the transmitter enable signal and a set of transmit timing parameters configured by user, ADRV9001 further controls the transmitter interface, transmit internal analog components, as well as the antenna switch (if it is controlled by ADRV9001 instead of user) to make sure that the transmit burst is on air at deterministic time as desired by user.

Transmit timing parameters in Figure 63 can be categorized into three types: ADRV9001 parameter (ADRV9001 provides to user), user parameter (user provides to ADRV9001), and helper parameters (determined by user which are not needed to provide to ADRV9001 but could be used by the user to derive other required timing parameters). Table 24 further explains all these timing parameters. All bounds specified in Table 24 are suggestions for optimal operation, no hardware or software restrictions prevent users from setting values that are out of bounds. The maximum programmable parameter value is specified in later sections.

Table 24. Transmit Timing Parameters Description

Tx Timing Parameters	Provided By	Bounds	Comments
enableSetupDelay ($t_{TxEnaSetup}$)	ADRV9001 Parameter	Min: N/A Max: N/A	No PLL retuning @ frame boundary: 8 µs (analog power-up time) PLL tuning @frame boundary: 758 µs (Analog Power-Up time + PLL Tuning time) (The PLL tuning time 750 µs refers to the case when internal LO is used. When external LO is used, users should calculate and use thei own PLL tuning time. Note the time required for PLL tuning is continuously improving in the future.)

Tx Timing Parameters	Provided By	Bounds	Comments
propagationDelay (t_{TxPD})	Helper Parameter	Min: N/A Max: N/A	This parameter should be measured by user and it is profile dependent and board layout dependent. It does not need to provide to ADRV9001. It can be used to derive other parameters required by the ADRV9001.
enableRiseToOnDelay ($t_{TxEnaRise2On}$)	User Parameter	Min: 0 Typical: t_{TxPD} Max: t_{TxGT}+ t_{TxPD}	@ min bound: antenna switch occurs $t_{TxEnaSetup}$ + some margin after Tx enable rising edge @ typical value: all symbols sent over interface (including guard symbols) make it onto the air @ max bound: no guard symbols are transmitted over the air
enableRiseToAnalogOnDelay ($t_{TxEnaRise2AnaOn}$)	User parameter	Min: 0 Max: $t_{TxEnaRise2On} - t_{TxEnaSetup}$	If Tx propagation delay is long, the analog power up can be delayed for power saving or to keep Tx analog powered down during Rx frame. If Tx propagation delay is small, this should be set to 0. If this parameter is greater than its max bound, the antenna switch time could be delayed.
enableGuardDelay (t_{TxGT})	User parameter	Min: TBD Max: TBD	TBD
enableHoldDelay ($t_{TxEnaHold}$)	User parameter	Min: 0 Max: None. Must be optimized to be minimal.	Tx_enable falling edge should ideally come as last valid data is sent over interface. This can be used to disable Tx algorithms whose performance may be degraded if guard symbols are used. Interface can be kept on even after Tx_enable falling edge to allow transmission of user guard symbols.
enableFallToOffDelay ($t_{TxEnaFall2Off}$)	User parameter	Min: $t_{TxEnaHold}$ Max: None. Must be optimized to be minimal. (Recommended Max: $t_{TxEnaHold}$ + t_{TxPD} Note $t_{TxEnaHold}$ is forced to 0 currently.)	@ min bound: antenna switch occurs soon after Tx interface is disabled. It should always occur prior to powering down of Tx analog. Not all symbols in the path make it on air. @ max bound: antenna is switched away from Tx channel just as last user data has propagated to antenna.

Design Strategies for Transmit Timing Parameters

Use Case 1: $t_{TXPD} > t_{TXENASETUP}$

In this case, because the propagation delay is larger than the transmit analog setup delay, user may choose to delay powering up analog front end while the data is propagating through the digital datapath as shown in Figure 63. This could achieve better power savings. For example, if the user measures the propagation delay as 2.5 ms, whereas the enableSetupDelay provided by ADRV9001 is 8 µs, analog front end could be off to avoid burning power for the first 2.492 ms of the propagation time. Having the analog front end powered up early could also be a liability. For example, if the transmit propagation path delay is longer than the guard time between receive and transmit frames, the transmit enable rising edge may occur in the middle of an receive frame. In this case, the user may want to keep the analog front end of the transmitter channel powered down until the end of the receive frame. In such a case, the enableRiseToAnalogOnDelay should be set to some value less than or equal to

propagationDelay – enableSetupDelay

Set enableRiseToOnDelay equal to the propagationDelay or enableRiseToAnalogOnDelay + enableSetupDelay. The transmit enable rising edge should occur enableRiseToOnDelay before on air transmit begins. The transmit interface could be set high at the same time as transmit enable to start transmitting guard symbols.

When the frame ends, the transmit enable falling edge should ideally occur right as the last valid data that must be demodulated by a receiver sent over interface. Interface can be held on for some time longer to allow guard data to be sent across by setting enableHoldDelay to a value greater than zero (note currently enableHoldDelay is forced to 0 by ADRV9001). The parameter, enableFallToOffDelay, determines how much after the transmit enable falling edge, the antenna is switched away from the transmit channel. It must always be set to a value greater than or equal to enableHoldDelay. If both values are set equal, for example, both are set to 0, the interface turns off first, then analog powers down. To ensure that all the data that was sent over the interface makes it onto the air, enableFallToOffDelay should be set greater than or equal to

enableHoldDelay + propagationDelay

If it is greater, then zeros are transmitted after all the data sent over the interface has been propagated.

Use Case 2: $t_{TxPD} < t_{TxEnaSetup}$

In this case, as shown in Figure 64 the time taken for data to propagate from the digital interface to antenna is very small, that is, t_{TxPD} is smaller than the time to setup the analog front end $t_{TxEnaSetup}$, in such a case, enableRiseToAnalogOnDelay can be set to 0, so that analog

power up begins immediately after TX_ENABLE rising edge. The parameter, enableRiseToOnDelay, could also be set to 0, in this case, the antenna is switched to a transmit channel, as soon as analog power up completes. For a more deterministic delay between transmit enable rising edge and antenna switch time, enableRiseToOnDelay should be set to a value greater than or equal to

enableRiseToAnalogOnDelay + enableSetupDelay

In such a case, after raising Tx_enable, some guard data must be sent over the interface to make sure all valid transmit data is transmitted on air. Based on the length of the user guard time and transmit timing parameter configurations, only a part or none of the guard data is transmitted to the air.

When the frame ends, enableFallToOffDelay could be set in a similar way as discussed in Use Case 1.

Note ADRV9001 currently is not controlling the antenna switch, therefore it is the user's responsibility to switch the antenna on and off at the accurate time. As a recommendation, the antenna should be switched on after analog power up and switched off before analog power down.

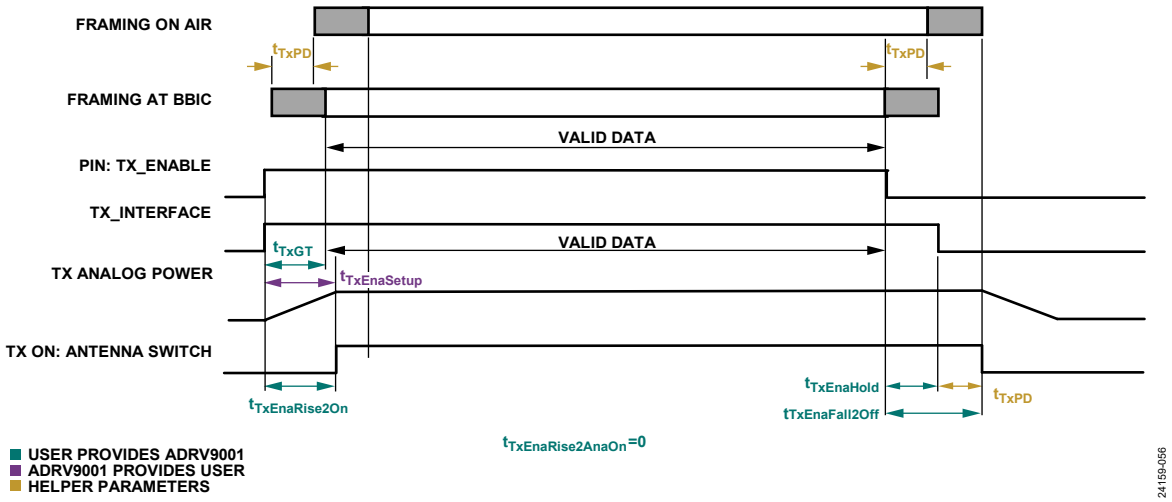


Figure 64. Transmit Timing Parameters ($t_{TxPD} < t_{TxEnaSetup}$)

Receive Timing Definition

Receive timing parameters define the events that take place in order from the start of reception at the air to the end of reception when the receive burst is sent through the ADRV9001 data port to the BBIC.

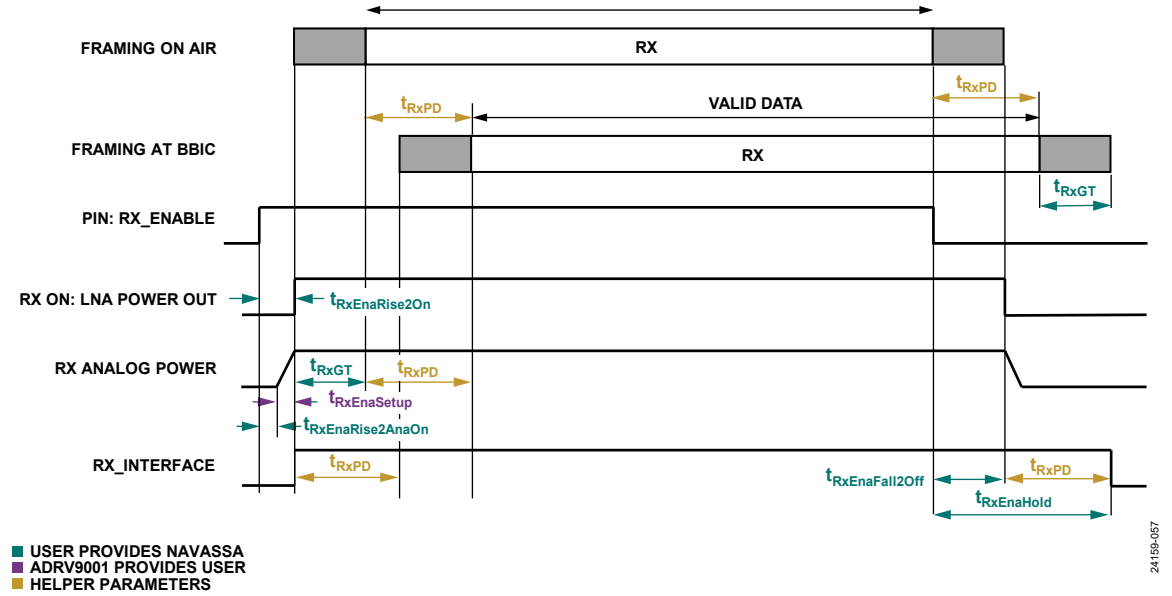


Figure 65. Receive Timing Parameters

As shown in Figure 65, similarly, a receive burst is composed of a series of valid receive data with user's option of padding guard data at the beginning and end of the valid data. Similar to transmit, based on the timing parameters configured by the user, it is the user's

decision if full or partial of the guard data should be received and it is the user's responsibility to make sure that the guard data usage is compliant with the standard requirement. The RX_ENABLE pin is controlled by the user to signal ADRV9001 the start and end of a receive burst at the air (Note RX_ENABLE should rise before the start of the receive burst at air to allow powering up analog front end.). Based on the RX_ENABLE signal and a set of receive timing parameters configured by the user, ADRV9001 further controls receive analog components, receive interface, and the external LNA (if it is controlled by ADRV9001 instead of user) to make sure that the received burst is sent to BBIC at the deterministic time as desired by user.

Similar to transmit timing parameters, as shown in Figure 63, receive timing parameters can be categorized into three types: ADRV9001 parameter (ADRV9001 provides to user), user parameter (user provides to ADRV9001) and helper parameters (determined by user which are not needed to provide to ADRV9001 but could be used by the user to derive other required timing parameters).

All the parameters used in Figure 65 are explained further in Table 25. All bounds specified in Table 25 are suggestions for optimal operation, no hardware or software restrictions prevent a customer from setting values that are out of bounds. The maximum programmable parameter value is specified in later sections.

Table 25. Receive Timing Parameters Description

Delay	Provided By	Bounds	Comments
enableSetupDelay ($t_{RxEnaSetup}$)	ADRV9001 Parameter	Min: N/A Max: N/A	No PLL tuning @ frame boundary: 8 µs (analog power-up time) PLL tuning @frame boundary: 758 µs (Analog Power-Up Time + PLL Tuning Time) (The PLL tuning time 750 µs refers to the case when internal LO is used. When external LO is used, users should calculate and use their own PLL tuning time. Note the time required for PLL tuning is continuously improving in the future.).
propagationDelay (t_{RxPD})	Helper Parameter	Min: N/A Max: N/A	This parameter should be measured by user and it is profile dependent and board layout dependent. It does not need to provide to ADRV9001, however, it can be used to derive values for other parameters required by ADRV9001.
enableRiseToAnalogOnDelay ($t_{RxEnaRise2AnaOn}$)	User Parameter	Min: 0 Max: duration of power up tasks in power savings or frequency hopping modes.	Will only be set to non-zero values if using power savings or frequency hopping. See later sections to determine ho to choose a non-zero value.
enableRiseToOnDelay ($t_{RxEnaRise2On}$)	User Parameter	Min: $t_{RxEnaRise2AnaOn}$ Typ: $t_{RxEnaRise2AnaOn}$ + $t_{RxEnaSetup}$ Max: None. Must be optimized to be minimal.	If set to $t_{RxEnaRise2AnaOn}$, the actual delay is $t_{RxEnaRise2AnaOn}$ + $t_{RxEnaSetup}$.
enableGuardDelay (t_{RxGT})	User Parameter	Min: TBD Max: TBD	TBD
enableFallToOffDelay ($t_{RxEnaFall2Off}$)	User Parameter	Min: 0 Max: None. Must be optimized to be minimal.	Ideally, RX_ENABLE falling edge arrives when the last valid data is received over the air. By setting this value greater than 0, ADRV9001 can continue receiving guard symbols, while signaling to certain algorithms or other systems that the valid data for the frame has already been received.
enableHoldDelay ($t_{RxEnaHold}$)	User Parameter	Min: $t_{RxEnaFall2Off}$ Max: None. Must be optimized to be minimal. (Recommended Max: $t_{RxEnaFall2Off}$ + t_{RxPD} Note $t_{RxEnaFall2Off}$ is forced to 0 currently.)	The interface is disabled only after analog power down has completed. @ min bound: Some of the data received at the antenna may not make it over the interface. @ max bound: Digital datapath and Rx SSI interface remains enabled until last received data is propagated to the interface.

Design Strategy for Receive Timing Parameters

As described, ADRV9001 provides user enableSetupDelay which is the time required to power up the receiver front end. By knowing that, user could set the RX_ENABLE pin high at least enableRiseToOnDelay in advance as shown in Figure 64. In regular TDD mode, that is, no power savings or frequency hopping, enableRiseToAnalogOnDelay should always be set to 0, so that analog power up begins

immediately after receive enable rising edge (Note Figure 64 describes receive timing parameters in a general case with enableRiseToAnalogOnDelay not equal to 0.). The parameter enableRiseToOnDelay could also be set to 0, in this case, the LNA is powered up as soon as analog power up completes. For a more deterministic delay between RX_ENABLE rising edge and LNA power up time, enableRiseToOnDelay can be set to a value greater than or equal to

enableRiseToAnalogOnDelay + enableSetupDelay

Once timing on air is established, the user may choose to raise RX_ENABLE, sometime before the start of the actual frame. As soon as the Rx analog power up completes, the digital interface turns on, however, if the path has a long propagation delay, the initial data coming off the interface are not the data received over the air.

When the frame ends, users may wish to continue receiving for a while, however, ADRV9001 may wish to stop all the tracking algorithms to avoid any performance degradation. This can be achieved by bringing the RX_ENABLE signal low as soon as the frame ends, but setting the enableFallToOffDelay equal to the time user wish to continue receiving data (Note enableFallToOffDelay is forced to 0 currently by ADRV9001.). This time should be no larger than the guard time before the next frame. The longer this value, the later the next Rx_enable rising edge can occur. In cases where the receive path has a large propagation delay, users may wish to turn off the receiver analog front end, so that users may commence a transmit frame, but still leave the digital datapath and interface on so that data already received over the air may be sent over the interface. The enableHoldDelay parameter is used for this purpose. It must always be set at least the enableFallToOffDelay. In order to receive all the data already received over the air, it should be set to

enableFallToOffDelay + propagationDelay

Guard/Hold Times Between Edges of TX_ENABLE and RX_ENABLE

By understanding the transmit and receive timing parameters discussed separately in the previous sections, the minimum guard/hold time design between the rising and falling edges of TX_ENABLE and RX_ENABLE in a TDD system are further discussed in this section. Six scenarios are considered, as follows:

- Guard time between TX_ENABLE falling edge and RX_ENABLE rising edge
- Guard time between RX_ENABLE falling edge and TX_ENABLE rising edge
- Guard time between TX_ENABLE falling edge and TX_ENABLE rising edge
- Guard time between RX_ENABLE falling edge and RX_ENABLE rising edge
- Hold time between TX_ENABLE rising edge and TX_ENABLE falling edge
- Hold time between RX_ENABLE rising edge and RX_ENABLE falling edge

The user should always set the guard/hold timer greater than the minimum requirement. Note no hardware or software restriction prevents user from raising TX_ENABLE/RX_ENABLE at any time. Correct operation cannot be guaranteed if rules described in the following sections are violated.

Guard Time Between TX_ENABLE Falling Edge and RX_ENABLE Rising Edge

The guard time between TX_ENABLE falling edge and RX_ENABLE rising edge is for making sure that the transmitter analog front end and the receiver analog front end are not powered up simultaneously. As discussed in previous sections, after TX_ENABLE falling edge, it takes $t_{TxEnaFall2Off}$ to power off the transmitter analog front end. Therefore, the earliest time the receiver analog front end can be powered up is $t_{TxEnaFall2Off}$ after the TX_ENABLE falling edge. Because it takes $t_{RxEnaRise2On}$ to power up the receiver analog front end starting from the RX_ENABLE rising edge, the minimum guard time is $t_{TxEnaFall2Off} - t_{RxEnaRise2On}$ if $t_{TxEnaFall2Off}$ is greater than $t_{RxEnaRise2On}$. In the case of $t_{TxEnaFall2Off}$ is less than $t_{RxEnaRise2On}$ (this could be possible when power saving modes are enabled as discussed in later sections), RX_ENABLE rising edge could happen $t_{RxEnaRise2On} - t_{TxEnaFall2Off}$ before TX_ENABLE falling edge. Figure 66 describes both cases.

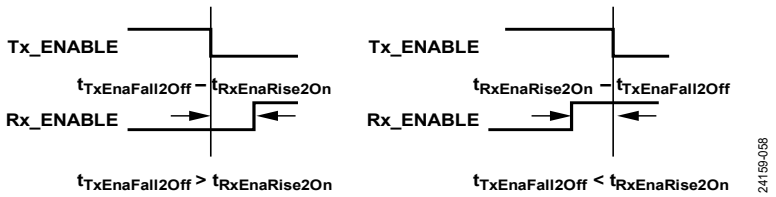


Figure 66. Minimum Guard Time Between TX_ENABLE Falling Edge and RX_ENABLE Rising Edge

Guard Time Between RX_ENABLE Falling Edge and TX_ENABLE Rising Edge

Similarly, the guard time between RX_ENABLE falling edge and TX_ENABLE rising edge is for making sure that the Rx analog front end and the Tx analog front end are not powered up simultaneously. As discussed in previous sections, after RX_ENABLE falling edge, it takes $t_{RxEnaFall2Off}$ to power off the Rx analog front end. Therefore, the earliest time Tx analog front end can be powered up is $t_{RxEnaFall2Off}$ after the RX_ENABLE falling edge. Because it takes $t_{TxEnaRise2On}$ to power up the transmitter analog front end starting from the TX_ENABLE

rising edge, the minimum guard time is $t_{RxEnaFall2Off} - t_{TxEnaRise2On}$ if $t_{RxEnaFall2Off}$ is greater than $t_{TxEnaRise2On}$. In the case of $t_{RxEnaFall2Off}$ is less than $t_{TxEnaRise2On}$, TX_ENABLE rising edge could happen $t_{TxEnaRise2On} - t_{RxEnaFall2Off}$ before TX_ENABLE falling edge. Figure 67 describes both cases.

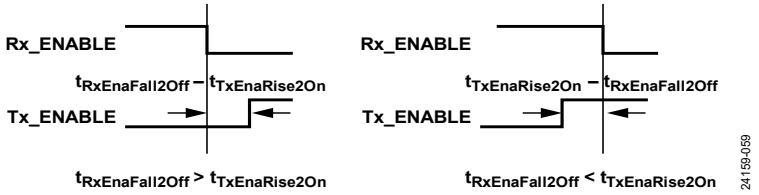


Figure 67. Minimum Guard Time Between RX_ENABLE Falling Edge and TX_ENABLE Rising Edge

Guard Time Between TX_ENABLE Falling Edge and TX_ENABLE Rising Edge

The guard time between TX_ENABLE falling edge and TX_ENABLE rising edge is for making sure that the interface is turned off at the end of the previous frame before it turns on again for the next frame. In addition, it must also make sure that the analog front end has been powered off in the previous frame prior to powering up again in the new frame. Because it takes $t_{TxEnaHold}$ to turn off the transmit interface after the TX_ENABLE falling edge, the next TX_ENABLE rising edge must come after a delay of at least $t_{TxEnaHold}$. This ensures that the interface is turned off at the end of the previous frame before it turns on again for the next frame. Since it takes $t_{TxEnaFall2Off}$ to power down the transmitter analog front end after the TX_ENABLE falling edge, the next TX_ENABLE rising edge must come after a delay of at least equal to $t_{TxEnaFall2Off} - t_{TxEnaRise2AnaOn}$. This ensures that the analog front end has been powered off in the previous frame prior to powering up again in the new frame. If the timing parameters are set appropriately, these two conditions are almost identical. If they are not identical for some reason, the guard time should be set as the maximum of $t_{TxEnaHold}$ and $t_{TxEnaFall2Off} - t_{TxEnaRise2AnaOn}$. Figure 68 describes this scenario.

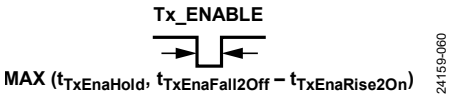


Figure 68. Minimum Guard Time Between TX_ENABLE Falling Edge and TX_ENABLE Rising Edge

Guard Time Between RX_ENABLE Falling Edge and RX_ENABLE Rising Edge

The guard time between the RX_ENABLE falling edge and RX_ENABLE rising edge is for making sure that the interface is turned off at the end of the previous frame before it turns on again for the next frame. Because it takes $t_{RxEnaHold}$ to turn off the receive interface after the RX_ENABLE falling edge, the next RX_ENABLE rising edge must come after a delay of at least $t_{RxEnaHold}$. This ensures that the interface is turned off at the end of the previous frame before it turns on again for the next frame. Because the analog powers down before the interface, the analog front end is guaranteed to power down prior to being powered up at the start of the next frame if this condition is met. Figure 69 describes this scenario.

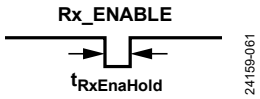


Figure 69. Minimum Guard Time Between RX_ENABLE Falling Edge and RX_ENABLE Rising Edge

Hold Time Between TX_ENABLE Rising Edge and TX_ENABLE Falling Edge

After a TX_ENABLE rising edge, its falling edge must come after a delay of at least $t_{TxEnaRise2AnaOn}$ or $t_{TxEnaRise2On}$ (if controlling antenna switch). In order to actually transmit, the channel must be on for a duration longer than its propagation delay. This can be achieved, either by making sure TX_ENABLE is high for longer than the propagation delay, or by ensuring the $t_{TxEnaHold}$ and $t_{TxEnaFall2Off}$ are longer than t_{TxPD}. Figure 70 describes this scenario.

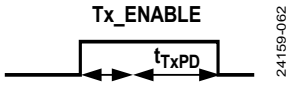


Figure 70. Minimum Hold Time between TX_ENABLE Rising Edge and TX_ENABLE Falling Edge

Hold Time Between RX_ENABLE Rising Edge and RX_ENABLE Falling Edge

After a RX_ENABLE rising edge, its falling edge must come after a delay of at least $t_{RxEnaRise2AnaOn}$ or $t_{RxEnaRise2On}$ (if controlling LNA power). In order to actually receive data, the channel must be on for a duration longer than its propagation delay. This can be achieved, either by making sure RX_ENABLE is high for longer than the propagation delay or by ensuring the $t_{RxEnaHold}$ is longer than t_{RxPD}. Figure 71 describes this scenario.

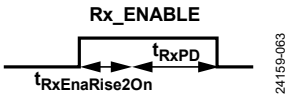


Figure 71. Minimum Hold Time Between RX_ENABLE Rising Edge and RX_ENABLE Falling Edge

Timing Parameters with Power Savings Modes

ADRV9001 offers several channel power savings modes (Power Saving Mode 0, Power Saving Mode 1, and Power Saving Mode 2) that trade off better power savings with longer transition time to turn on and turn off a transmit or receive channel. Please refer to the Power Saving and Monitor Mode section in this User Guide for more details about power saving modes. In order to take advantage of these power saving modes, the timing parameters must be set appropriately.

Note the minimum guard time discussed above does not consider the time takes to power down transmit or receive analog by assuming it is insignificant. But it is highly recommended to allow extra time to make sure analog power up happens only after analog power down is fully completed. The analog power down time is usually much less than the analog power up time.

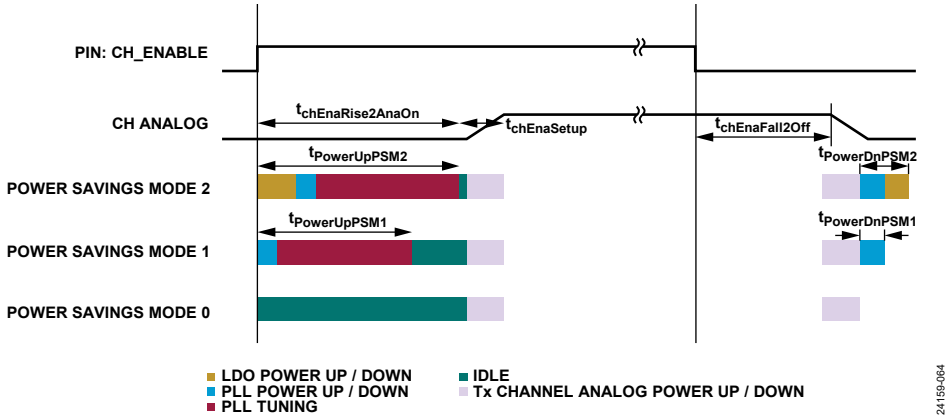


Figure 72. Channel Power-Up and Power-Down Sequences in Different Power Savings Modes

Figure 72 shows the sequence of events taken to power up or power down a transmit or receive channel in the various channel power savings modes. It can be seen in Power Savings Mode 1 and Power Savings Mode 2, the enableRiseToAnalogOnDelay is used to power up additional entities that may have been powered down at the end of the previous frame. (Note in Power Savings Mode 1 and Power Savings Mode 2, PLL is powered down at the end of the previous frame. Therefore, when it is turned on at the start of the new frame, PLL tuning is required.) Thus, the enableRiseToAnalogOnDelay must be set long enough to allow these power up procedures to complete. If the additional power-up procedures in Power Savings Mode 2 takes $t_{PowerUpPSM2}$ to complete, the ADRV9001 prevents the system from entering Power Savings Mode 2, unless enableRiseToAnalogOnDelay is set greater than $t_{PowerUpPSM2}$. Similarly, the same is true for Power Savings Mode 1, the ADRV9001 prevents the system from entering Power Savings Mode 1, unless enableRiseToAnalogOnDelay is set greater than $t_{PowerUpPSM1}$. In Power Savings Mode 0, which is the default mode, there are no additional power up procedures, thus there are no additional restrictions on enableRiseToAnalogOnDelay other than those already specified in earlier sections.

If switching dynamically between several power savings modes, user should set the enableRiseToAnalogOnDelay to satisfy the restrictions of the highest power savings mode. Figure 71 shows that there is a longer idle time when switching to a lower power savings mode. The parameter enableRiseToAnalogOnDelay cannot be changed dynamically, thus the timing of the TX_ENABLE/RX_ENABLE rising edge relative to the on air time should also remain the same even when dynamically switching between different power savings modes.

In certain use cases, when transmit and receive are using the same LO but at different frequencies, if the transition times between transmit and receive frames are always long enough, PLL tuning is performed at the start of the frame. This is not related to any specific power saving mode and PLL tuning happens even in Power Saving Mode 0. The timing diagram looks like Figure 73.

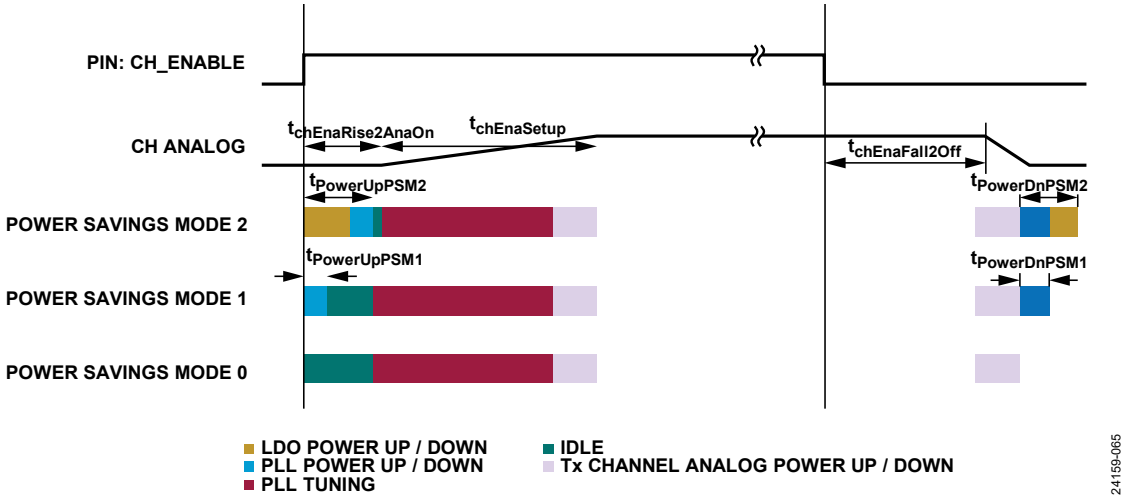


Figure 73. Channel Power-Up and Power-Down Sequence in Different Power Savings Modes (PLL Retune @ Frame Boundary Case)

In this case, in all power saving modes, the PLL tuning is performed during enableSetupDelay instead of enableRiseToAnalogOnDelay. Therefore, enableSetupDelay is much longer as it must allow time to tune the PLL. This means that the additional power up durations $t_{PowerUpPSM}$ are much shorter and thus higher power savings can be achieved while setting the enableRiseToAnalogOnDelay to a much smaller value.

All above description is for internal LOs scenarios, if ADRV9001 is configured with external LO mode, users take the responsibility to configure or re-tune the external PLLs, ADRV9001 channel power up and power down sequence in different power saving modes are same with Figure 71, users should make sure the external LOs are ready before the enableRiseToAnalogOnDelay is expiry.

Impact of Power Savings on Timing Parameter Selection

As explained in the previous section, certain power savings modes cannot be entered if the enableRiseToAnalogOnDelay for that channel is not greater than the duration of the additional power up procedures needed in that mode.

For transmit channels, if the propagation delay is quite large, the enableRiseToAnalogOnDelay chosen may already be larger than the longest power up procedure duration, that is, $t_{PowerUpPSM2}$. In this case, there is no impact to the selection of the timing parameters.

For receive channels, or transmit channels with short propagation delays, the enableRiseToAnalogOnDelay must be chosen larger than $t_{PowerUpPSM1}$ to enter Power Savings Mode 1 and larger than $t_{PowerUpPSM2}$ to enter Power Savings Mode 2 and higher. The enableRiseToOnDelay, if it is being used, must also increase as it must always be larger than enableRiseToAnalogOnDelay. However, none of the other timing parameters are affected by the power savings mode.

At the end of the frame, the power-down procedures take some small but finite time. For receiver channels with large propagation delay, this may have no impact because the digital datapath might be on for a long time after the analog has powered down.

For transmit channels or receive channels with short propagation delays, the minimum period between the channel enable falling edge and the next rising edge must be enableHoldDelay plus the additional time needed for the extra power down procedures ($t_{PowerUpPSM1}$, $t_{PowerUpPSM2}$). This prevents PLL or LDO from beginning power up in the new frame even before it has finished powering down in the old one.

Hardware and Software Restrictions for Timing Parameters

As previously mentioned, the bounds provided for each of these timing parameters and the guard times between rising and falling edges of the receiver and transmitter enable signals are only guidelines. There are almost no hardware or software restrictions preventing users from setting these parameters anyway they like including harmful or useless ways. There are in place a few restrictions, however, which are outlined as follows:

- All timing parameters that must be provided by user have to be within the range of 0 ms to 91 ms. These bounds are specified, assuming the delay generation blocks run at 184.32 MHz (system clock). If operating at a different frequency, the maximum bound scales accordingly. For example, if using a 160 MHz clock, the max delay is 91 ms/184.32 × 160 = 79 ms).
- For all channels the enableRiseToOnDelay must be greater than or equal to the enableRiseToAnalogOnDelay, provided the enableRiseToOnDelay parameter is being used, that is, ADRV9001 is controlling antenna switch and/or LNA power.
- For transmitter channels, the enableHoldDelay must be less than or equal to the enableFallToOffDelay.
- For receiver channels, the enableFallToOffDelay must be less than or equal to the enableHoldDelay.

- For a specific channel, Power Savings Mode 2 or higher is disallowed when the enableRiseToAnalogOnDelay is less than $t_{PowerUpPSM2}$.
- For a specific channel, Power Savings Mode 1 or higher is disallowed when the enableRiseToAnalogOnDelay is less than $t_{PowerUpPSM1}$.

API Programming and Default Values for Timing Parameters

A set of API commands are provided to the user to configure timing parameters. Because the timing parameters are related to the channel power saving mode, users should set the channel power saving mode first before configuring the timing parameters. API Command adi_adrv9001_arm_ChannelPowerSaving_Configure( ) is provided to the user to set the channel power saving mode for a specified channel when the channel is in the calibrated, primed, or RF_ENABLED state. After that, users could use API Command adi_adrv9001_Radio_ChannelEnablementDelays_Configure( ) to configure the timing parameters for the selected channel. The following data structure holds all the ADRV9001 required timing parameters:

```
typedef struct adi_adrv9001_ChannelEnablementDelays
{
    uint32_t riseToOnDelay;         /* Delay from rising edge until antenna switch (Tx) or LNA
(Rx) is powered up */
    uint32_t riseToAnalogOnDelay;   /* Delay from rising  edge until Tx/Rx analog power up
procedure commences */
    uint32_t fallToOffDelay;        /* Delay from falling edge until antenna switch (Tx) or LNA
(Rx) is powered down */
    uint32_t guardDelay;            /* Reserved for future use*/
    uint32_t holdDelay;             /* Delay from falling edge until the Tx/Rx interface is
disabled */
} adi_adrv9001_ChannelEnablementDelays_t
```

Note guardDelay is reserved for future use and forced to 0 by ADRV9001 for both transmit and receive channels. In addition to that, for the transmit channel, holdDelay is also reserved for future use and forced to 0. For the receive channel, fallToOffDelay is also reserved for future use and forced to 0. API Command adi_adrv9001_Radio_ChannelEnablementDelays_Configure( ) should be called when the channel is in the standby or calibrated state.

To set all those timing parameters properly, the user should have prior knowledge about ADRV9001 timing parameters (ADRV9001 provides to user) as well as helping parameters such as the transmit and receive propagation delay. The prior timing parameters include enableSetupDelay, propagationDelay, and maximum intended power savings mode, $t_{PowerUpPSM1}$ and $t_{PowerUpPSM2}$.

Table 26 summarizes all these timing parameters for both transmit and receive. Note all timing parameters specified in units of time assume a system clock frequency of 184.32 MHz. If using a different system clock frequency, it must be adjusted by

scaleFactor = 184.32 (MHz)/*system clock Frequency*

Table 26. Prior Tx/Rx Timing Parameters

	No PLL Retuning at Frame Boundary (Use Case in Figure 71)	**PLL Retuning at frame boundary (Use Case in Figure 72)**
enableSetupDelay	Analog Power-Up*scaleFactor	PLL Tuning + Analog Power-Up *scaleFactor
propagationDelay	From user's own measurement	Same as No PLL tuning case
$t_{PowerUpPSM1}$	PLL Tuning + PLL Power-Up *scaleFactor	PLL Power-Up *scaleFactor
$t_{PowerUpPSM2}$	PLL Tuning + LDO Tuning + PLL Power-Up *scaleFactor	LDO Tuning + PLL Power-Up *scaleFactor

The system clock Freq depends on the profile and user could find the corresponding value under **TDD Enablement Delays** tab in TES. In addition to that, TES also displays the timing parameters provided by ADRV9001 to help determine the prior transmit/receive timing parameters as described in Figure 74 which shows the picture of TES where those timing parameters and the system clock for the current user selected profile are located.

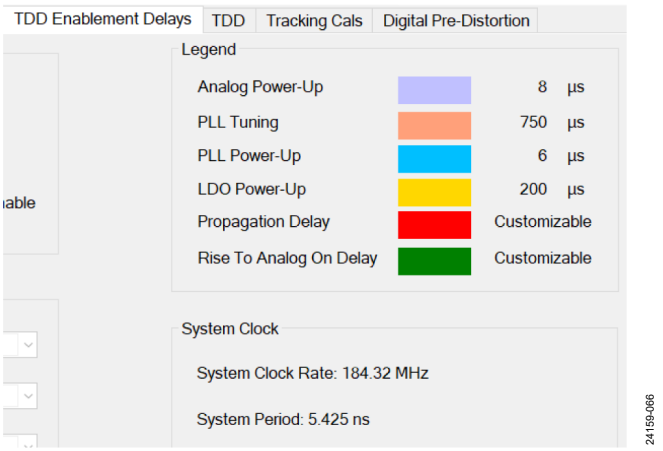


Figure 74. ADRV9001 Provided Timing Parameters and the System Clock for the Selected Profile in TES

Based on the information provided in Figure 74, user can further configure the ADRV9001 required timing parameters.

Default Timing Parameters for Transmit Channels

Figure 75 shows the ADRV9001 transmitter required timing parameters and their minimum, maximum, and default values as well as some recommendations are summarized in Table 27.

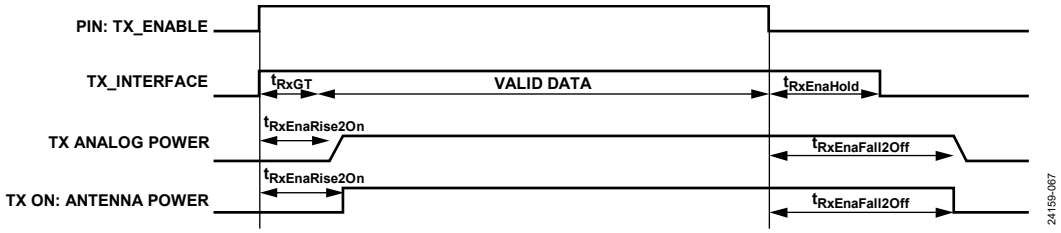


Figure 75. Transmit Timing Parameters

Table 27. ADRV9001 User Provided Transmit Timing Parameters

Timing Parameter	Min Value	Max Value	Comment
enableRiseToAnalogOnDelay ($t_{TxEnaRise2AnaOn}$)	Max of the following values: 0, propagationDelay – enableSetupDelay, $t_{PowerUpPSM}$ (for the maximum intended power savings mode)	91 ms/scaleFactor	Default = min
enableRiseToOnDelay ($t_{TxEnaRise2On}$)	enableRiseToAnalogOnDelay + enableSetupDelay	91 ms/scaleFactor	Default = min Not needed if not controlling LNA power
enableGuardDelay (t_{TxGT}) (Not Used Currently)	0	91 ms/scaleFactor	Default = min Can increase to non-zero if performance degradation is observed and the channel is transmitting for some time before the start of the actual frame (Forced to be 0 currently by ADRV9001).
enableHoldDelay ($t_{TxEnaHold}$) (Not Used Currently)	0	91 ms/scaleFactor	Default = min Can increase if performance degradation is observed and the channel is transmitting for some time after the end of the actual frame (Forced to be 0 currently by ADRV9001).
enableFallToOffDelay ($t_{TxEnaFall2Off}$)	enableHoldDelay	enableHoldDelay + propagationDelay	Default = max Can decrease if not all data sent through interface must be transmitted.

Default Timing Parameters for Receiver Channels

Figure 76 shows the ADRV9001 receiver required timing parameters and their minimum, maximum, and default values as well as some recommendations are summarized in Table 28.

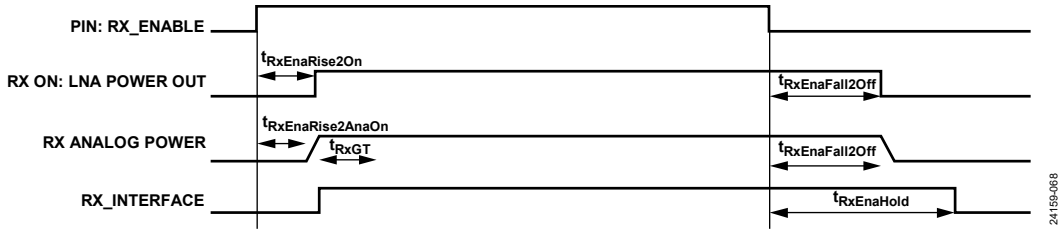


Figure 76. Receiver Timing Parameters

Table 28. User Provided Receiver Timing Parameters

Timing Parameter	Min Value	Max Value	Comment
enableRiseToAnalogOnDelay ($t_{RxEnaRise2AnaOn}$)	Max of the following values: 0 $t_{PowerUpPSM}$ (for the maximum intended power savings mode)	91 ms/scaleFactor	Default = min
enableRiseToOnDelay ($t_{RxEnaRise2On}$)	enableRiseToAnalogOnDelay + enableSetupDelay	91 ms/scaleFactor	Default = min Not needed if not controlling LNA power.
enableGuardDelay (t_{RxGT}) (not used currently)	0	91 ms/scaleFactor	Default = min Can increase if performance degradation is observed and the channel is receiving for some time before the start of the actual frame (Forced to be 0 currently by ADRV9001).
enableFallToOffDelay ($t_{RxEnaFall2Off}$) (Not Used Currently)	0	91 ms/scaleFactor	Default = min Can increase if performance degradation is observed and the channel is receiving for some time after the end of the actual frame. Can still be used if not controlling LNA power down. (Forced to be 0 currently by ADRV9001).
enableHoldDelay ($t_{RxEnaHold}$)	enableFallToOffDelay	enableFallToOffDelay + propagationDelay	Default = max Can decrease if not all data received on air must be sent over interface.

When ADRV9001 calculates the default values, it uses the transmit/receive propagation delay internally characterized for different profiles. User should measure the propagation delay for the entire system to help determine all the required timing parameters accurately. This may include power amplifier, LNA, filters and anything else that might be between chip and the antenna. During the measurement, the user could set all the timing parameters to be the default values for simplification and also it is important to use the same profile and configurations as the actual application being deployed.

Example Rx propagation delay: A typical system level measurement can be done with an RF switch that is controllable from BBIC. User can switch off the switch and turn on receiving signal and switch on the switch and start recording data from the interface. This signal can be a pattern from a signal generator with an external sync. Then BBIC could use the external sync to start recording and obtain a very accurate measurement.

After calculating all the timing parameters required by ADRV9001, user could configure them through TES as shown in Figure 77.

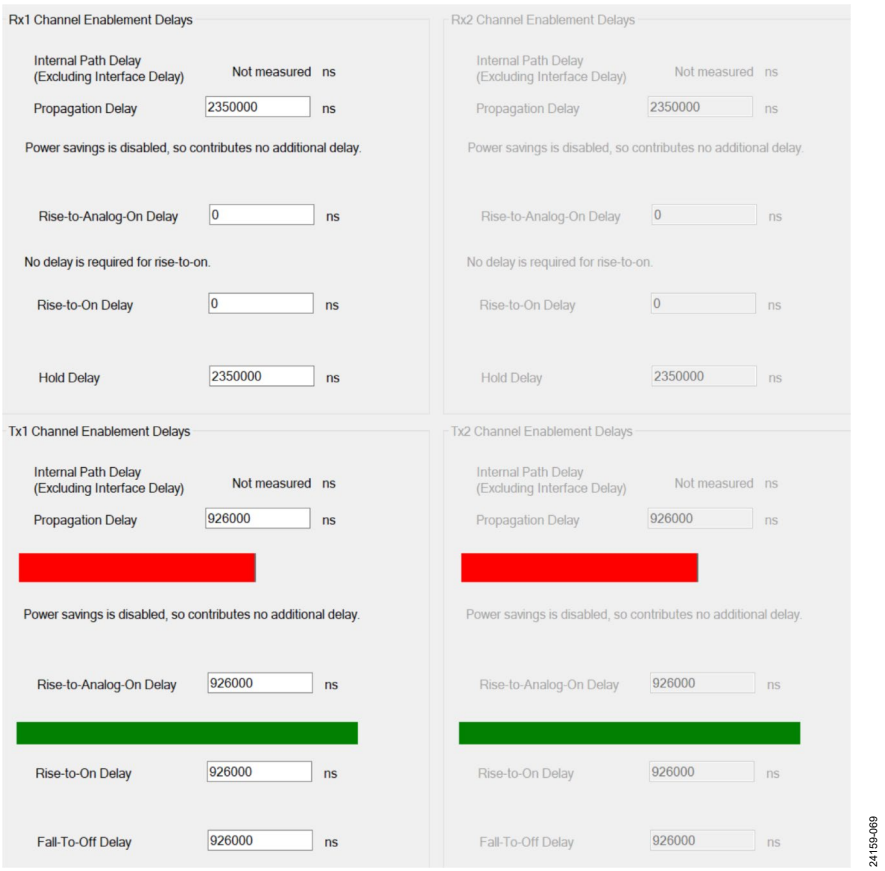


Figure 77. Timing Parameters Configuration in TES

As shown in Figure 77, only relevant channels are enabled for timing parameters configuration. User should enter all the values in ns. The propagation delay is a helper parameter, which is not needed by ADRV9001. It helps to set other timing parameters ADRV9001 requires.

As aforementioned, API Command adi_adrv9001_Radio_ChannelEnablementDelays_Configure( ) can also be used to set the timing parameters. As a summary, all the APIs provided for timing parameters are listed in Table 29. Refer to the API doxygen document for more details.

Table 29. A List of Timing Parameters Related APIs

API Function Name	Description
adi_adrv9001_arm_ChannelPowerSaving_Configure	Configures the channel power saving settings for the specified channel.
adi_adrv9001_arm_ChannelPowerSaving_Inspect	Inspects the channel power saving settings for the specified channel.
adi_adrv9001_Radio_ChannelEnablementDelays_Configure	Configures channel enable delays for the specified channel.
adi_adrv9001_Radio_ChannelEnablementDelays_Inspect	Inspects channel enable delays for the specified channel.

Pin control mode timing measurement

ADRV9001 receives Tx/Rx Enable control signals and the related user's timing parameters to perform the Tx/Rx RF On/Off operation in pin control mode. As previously mentioned the enableSetupDelay (EnaSetup) is the time taken for ADRV9001 to power up its analog front end, and ADRV9001 will also take small time to finish the operations to power down its analog front end at the falling edge of the Tx/Rx Enable signals, users should be advised the programable "EnaRise2AnaOn" and "EnaFall2Off" will delay the Adrv9001 analog components power on and off respectively. The enableSetupDelay varies with different ADRV9001 internal processor clock rate. Users can enable the ADRV9001 internal stream status to GPIO debug function to accurately measure the Tx/Rx enable control signal and when they are actually taken effective by ADRV9001 internal stream processor.

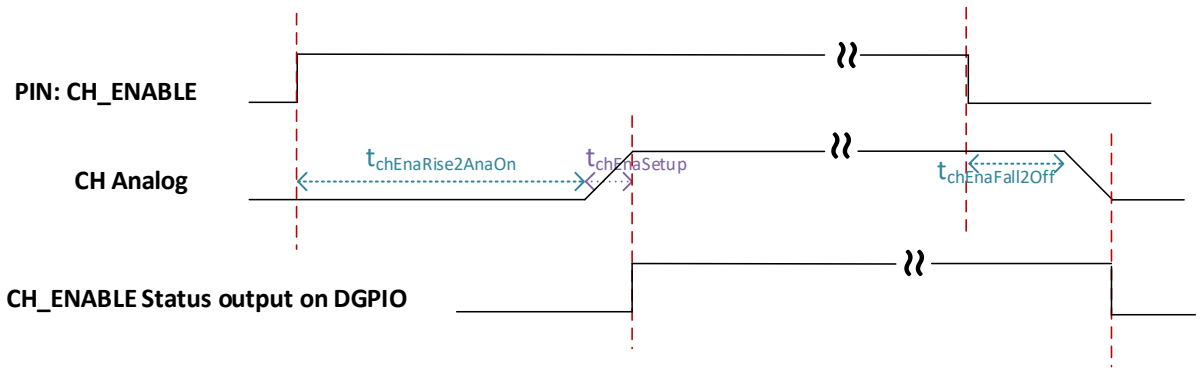


Figure 78 Channel Enable control status output on DGPIO

Figure 78 shows the channel enable signal and its corresponding active output timing on DGPIO, the DGPIO high indicates the start of the channel analog front end fully up to the end of channel analog front end fully off. If the EnaRise2AnaOn is set to 0, the time from CH_ENABLE rising edge to DGPIO rising edge is the "enableSetupDelay". Similarly, if set EnaFall2Off to 0, the time from CH_ENABLE falling edge to DGPIO falling edge will be the time taken for ADRV9001 to fully power down the analog front end.

This stream status to GPIO debug function can be enabled by API adi_adrv9001_Stream_C0_Gpio_Debug_Set() in chip "Standby" or "Calibrated" state, the RX1_ENABLE output status will be mapped to DPGIO_0, TX1_ENABLE output status to DGPIO_1, RX2_ENABLE output status to DGPIO_2 and TX2_ENABLE output status to DGPIO_3, there is no users' option to configure these DGPIOs mapping, and once the debug function is enabled, the DGPIO_0~DPGIO_3 will not be available for other functions, no matter how many channels are configured in the profile. We recommend users using this debug function to do the Tx/Rx ENABLE timing measurement for a given profile, and disable this function to release the DGPIOs usage once users are done with the measurement.

CLOCK GENERATION

CLOCK GENERATION

In ADRV9001 all clocks for the converters and main digital are generated by CLKGEN. CLKGEN receives from two clocks, a high performance (HP) clock PLL and a low power (LP) PLL. The high performance clock PLL has a programmable frequency range of 7.2 GHz to 8.8 GHz. The low power clock PLL can generate a programmable range of 3.3 GHz to 5 GHz frequency. CLKGEN also has the clocks be divided and retimed with reset pulses from the clock PLLs.

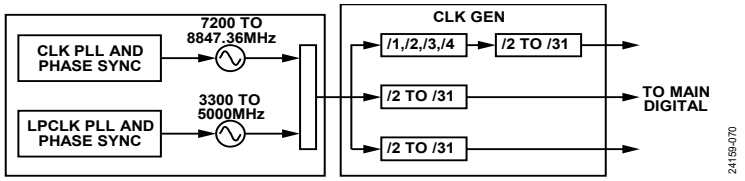


Figure 79. ADRV9001 Clock Generation

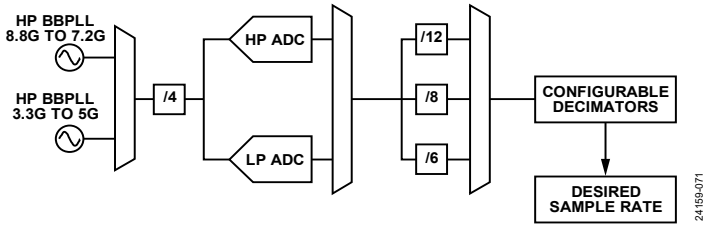


Figure 80. CLKPLL Can Be Programmed to Provide Arbitrary Clock Speed

Low Power Clock PLL (LP CLKPLL)

By default, LP CLKPLL works at a fixed frequency at 4423.68 MHz. However, the user can set it to have a tuning operation range. The operating frequency range of LP CLKPLL is from 3.3 GHz to 5 GHz. User only must provide their final sampling frequency at the interface, and the final frequency of LP CLKPLL are determined internally. This is all done before the chip is programmed. A profile is generated based on the user's provided sampling frequency.

Note LP CLKPLL uses less power than HP CLKPLL but produces more jitter noise. User must take this trade-off into consideration for their end application. Note that, in most profiles, LP CLKPLL meets the performance requirements.

Table 30 lists the supported data sample rate with different standards by LP CLKPLL. Table 31 lists the supported data lane rate by LP CLKPLL. Note the LTE 40 MHz at 16-bit is not supported by the LP CLKPLL.

Table 30. Sample Rate Supported By LP CLKPLL

Standard	Sample Rate
DMR I/Q	2.40E + 04
TETRA	1.44E + 05
TETRA	2.88E + 05
LTE 1.5	1.92E + 06
LTE 3	3.84E + 06
LTE 5	7.68E + 06
LTE 10	1.54E + 07
LTE 15	2.30E + 07
LTE 20	3.07E + 07
LTE 40 @12 bits	6.14E + 07

Table 31. Supported Data Lane Rate By LP CLKPLL

Standard	Serialization Factor Per Data Lane	Data Lane Rate
DMR/P25 Direct Modulation	2	9.60E +03
P25 Direct Modulation	2	1.20E +04
FM Direct Modulation	16	1.28E +05
DMR I/Q	32	7.68E + 05
	8	1.92E + 05
	16	3.84E + 05

Standard	Serialization Factor Per Data Lane	Data Lane Rate
FM Direct Modulation	16	1.54E + 06
TETRA	32	4.61E + 06
	8	1.15E + 06
	16	2.30E + 06
TETRA	32	9.22E + 06
	8	2.30E + 06
	16	4.61E + 06
LTE 1.5	32	6.14E + 07
	8	1.54E + 07
	16	3.07E + 07
LTE 3	8	3.07E + 07
	16	6.14E + 07
LTE 5	8	6.14E + 07
	16	1.23E + 08
LTE 10	16	2.46E + 08
LTE 15	16	3.69E + 08
LTE 20	16	4.92E + 08
LTE 40 @12 bits	12	7.37E +08

Arbitrary Sample Rate

With a programmable frequency range of both HP CLK PLL and LP CLK PLL as mentioned previously, ADRV9001 supports arbitrary sample rate (ASR) mode, which provides user a great flexibility to configure the desired sample rates in their applications. ASR mode supports an almost continuous range of rates up to 61.44 MHz with a list of dead zones mainly due to the limitations in decimation/interpolation rate supported in the data path. The following table summarizes the current dead zone frequency ranges:

Table 32. Dead Zone Frequency Ranges

Dead Zone (CLK PLL Limitation)	Lower Bound (MHz)	Upper Bound
1	50	160/3
2	100/3	40
3	25	80/3
4	50/3	20
5	12.5	40/3
6	25/3	10
7	6.25	20/3
8	25/6	5
9	3.125	10/3
10	25/12	2.5
11	1.5625	5/3
12	25/24	10/9
13	0.7812500	5/6
14	0	1/48

Note that there are a total of 14 different dead zones. For each one, the sample rates user cannot configure are between the lower bound and upper bound with lower bound frequency and upper bound frequency excluded, which means dead zone = (lower bound, upper bound).

When sample rate is greater than 160/3 MHz, HP CLK PLL should always be used. Due to the complexity of data path, users do not have the freedom to select their own CLK PLL clock frequencies and datapath configurations. Users should only provide their desired sample rate and the API will determine the appropriate ADRV9001 profile.

MULTICHIP SYNCHRONIZATION

INTRODUCTION

Multi-chip synchronization (MCS) is necessary when application requires deterministic latency between data paths, such as MIMO applications, which will require multiple ADRV9001 devices. MCS is the solution for this problem to have the data in multiple channels aligned in time. Certain applications not only require the delay to be deterministic but also require phase to be the same. ADRV9001 will also support PLL phase synchronization as one of the operation modes.

THEORY OF OPERATION

Figure 81, illustrates the synchronization between multiple ADRV9001 devices using a shared input pin, MCS. The MCS signal is generated by the external clock chip (for example, AD9528) using device clock DEV_CLK and captured by each ADRV900x device using negative or positive edge of DEV_CLK to meet setup and hold time with good margins. Each ADRV900x device uses this sampled MCS to synchronize all internally generated clocks which make them aligned between all devices internal clocks.

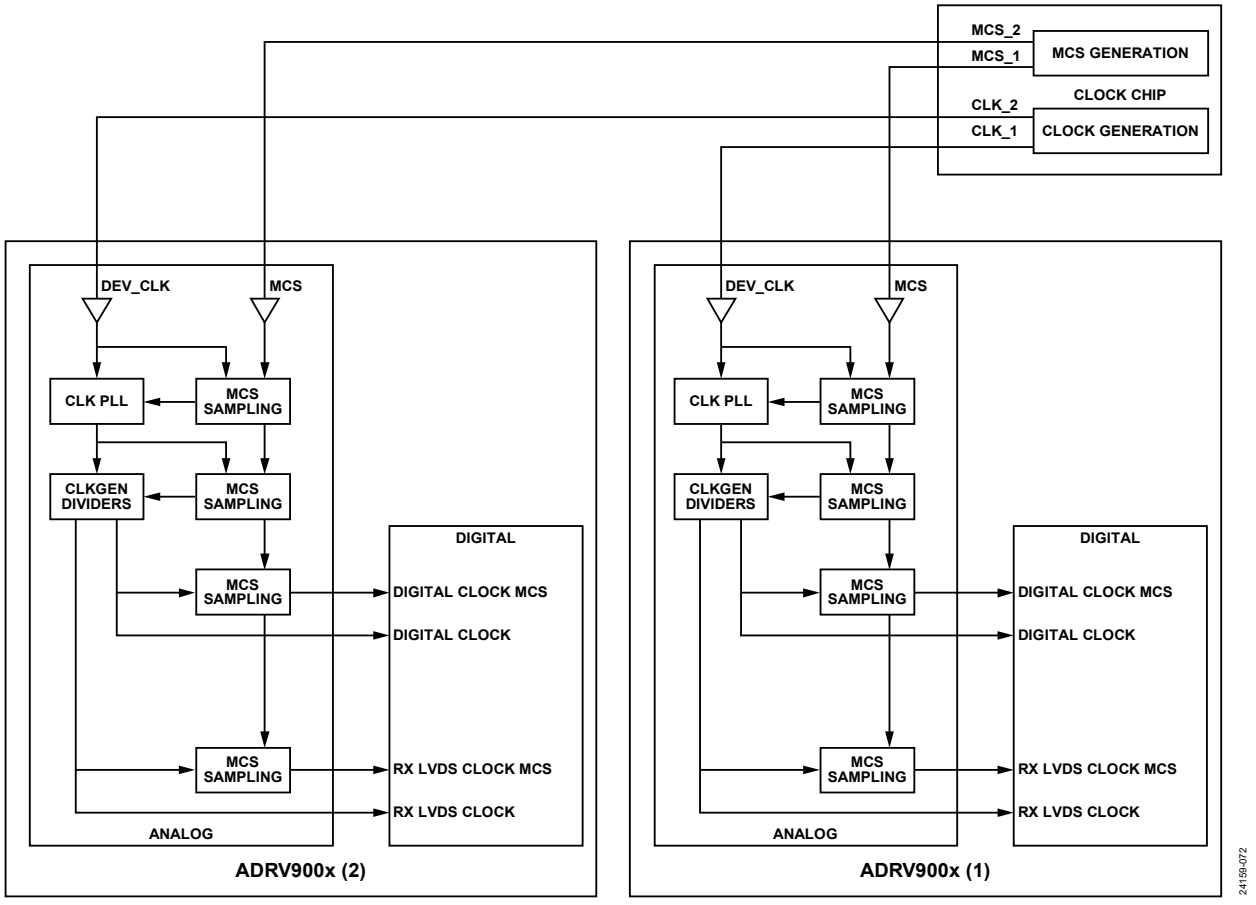


Figure 81 Multi-chip Synchronization System Diagram

Sampled MCS

MCS pulse signal will be sampled internally by ADRV9001 by the DEV_CLK signal rising or falling edge. Figure 82 shows the example that MCS pulse being sampled by the rising edge of the DEV_CLK. This process guarantees that the sampled MCS signal, which is used to synchronize all ADRV9001 devices are time aligned.

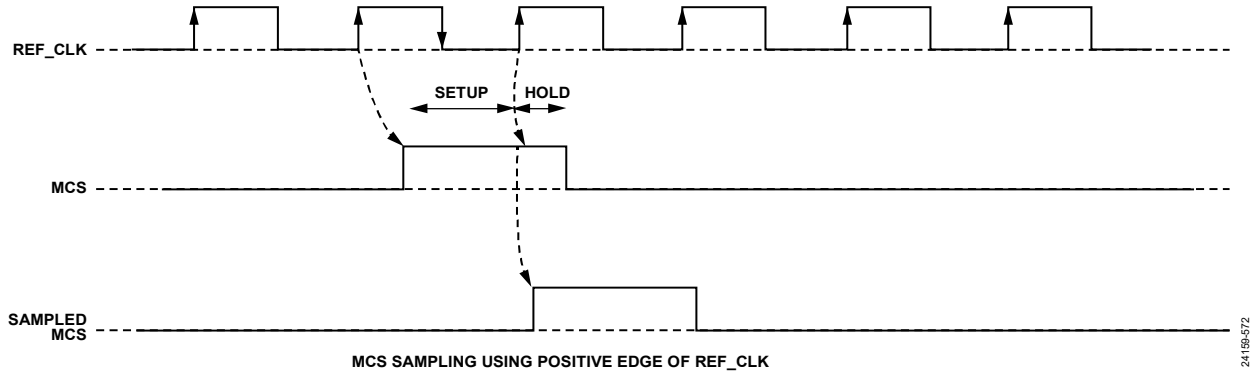


Figure 82 Sample MCS signal at rising edge of DEV_CLK

An external clock module is required to synchronize multiple ADRV9001 devices. Each ADRV9001 will receive a DEV_CLK and an MCS signal. The MCS signals should arrive at all ADRV9001 devices within one DEV_CLK cycle, for the reason that it needs to be sampled by the DEV_CLK mentioned above. For this reason, we recommend the layout to have equal-length traces between the external clock module and each of the ADRV9001 devices. User will need to carefully tune the external clock module so that the pulses will arrive at all ADRV9001 devices within one clock cycle time.

Setup time means the MCS positive edge has to arrive at least 5ns before DEV_CLK positive edge.

Hold time means the MCS negative edge has to arrive at least 5ns after DEV_CLK positive edge.

Setup/hold time are still being characterized and this is a preliminary result.

MCS pulses

The Figure 83 shows the MCS signal required to be received by ADRV9001. There are a total of 6 pulses. First 4 pulses are for the analog clock divider synchronization, and the last 2 are for the digital clock divider synchronization. Together they will synchronize all internal components of ADRV9001.

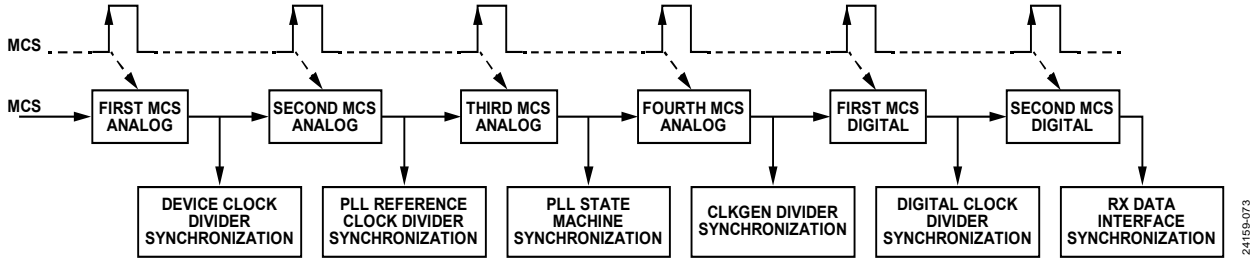


Figure 83 MCS pulses for analog and digital synchronization

Pulse width and delay

Table 33 shows the minimum pulse width of each MCS pulse, as well as the wait time required after each pulse. The user should use this reference to design MCS pulse generation.

Table 33. Minimum Time Requirement for MCS Pulse Width and Wait Time

Pulse No.	Pulse Width (No. of Reference Clock Cycles)	Wait Time After the Pulse Tn (µs)
1	≥2	>1
2	≥2	>1
3	≥2	>1
4	≥2	>100
5	≥2	>100
6	≥2	>1

Digital Only and Phase Synchronization

Enabling MCS will guarantee the delay between the RF ports to SSI interface (or reverse direction) to be deterministic, across all ADRV9001 devices. This ensures data will have deterministic delay (are synchronized) across all channels that have MCS enabled.

Additionally ADRV9001 also provide phase synchronization for the PLLs across multiple devices. User can choose to enable this option so that not only the data are synchronized in time, but also the phase of the PLLs are also synchronized.

Note if choosing MCSMODE_DIGITAL mode, which does not guarantee phase synchronization, the process is done only once and that's after CALIBRATED state. This means after all MCS pulses are sent and all ADRV9001 components are synchronized, no more pulse is needed. MCS is complete and no longer needs to run again unless chip is reset.

If MCSMODE_DIGITAL_AND_RFPLL_PHASE mode is selected, same as previous, MCS will be done at initialization stage and once complete it will no longer require MCS pulses. After that phase synchronization will take place but this does not require MCS to rerun. Whenever PLL changes, the phase sync will need to rerun to ensure the phase between all channels are synchronized.

To select one of the modes user can use the struct below:

```
typedef enum adi_adrv9001_McsMode
{
   ADI_ADRV9001_MCSMODE_DISABLE = 0,            /*!< Multi Chip Synchronization disabled */
   ADI_ADRV9001_MCSMODE_DIGITAL,                /*!< Multi Chip Synchronization for digital*/
   ADI_ADRV9001_MCSMODE_DIGITAL_AND_RFPLL_PHASE   /*!< Multi Chip Synchronization for digital and RFPLL phase */
} adi_adrv9001_McsMode_e;
```

Frequency Hopping

In the case of frequency hopping, user can choose one of the options above, to enable MCSMODE_DIGITAL to have deterministic delay, or to add additional phase synchronization. For the first option, user should only consider the PLL settling time, since there is no additional phase synchronization required. For the second option, there will be additional time consumed and this depends on the reference clock speed and the LO frequencies that user will use for frequency hopping.

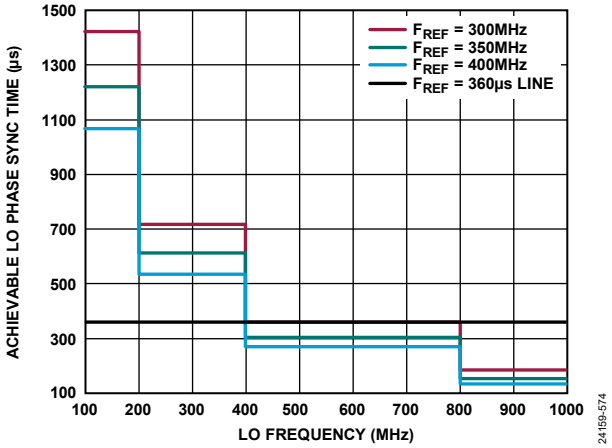


Figure 84. PLL Phase Synchronization Time vs LO frequency and Fref

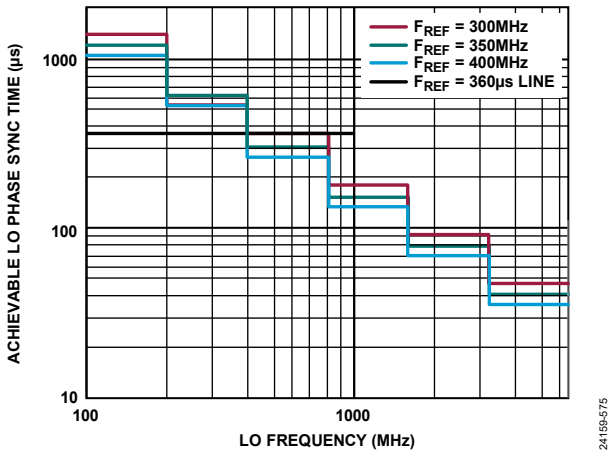


Figure 85. Theoretical Phase Sync Time up to 6GHz

Figure 84 shows the phase synchronization timing when it is required. Note the higher the LO frequency and reference clock speed, the lower the time it requires for phase synchronization. User should take this additional timing into consideration when designing frequency hopping with phase synchronization.

Figure 85 shows the theoretical phase synchronization timing to up to 6 GHz. The actual time will vary but should be close to the theoretical limit.

For frequency hopping, if user doesn't want phase synchronization, then user can select MCS only mode. In this case user will not care about phase synchronization time as it is not enabled. User will not have additional timing consideration because all MCS is done at the initial stage not at the hop stage.

If user decides to have phase synchronized, then user should select MCS and phase sync mode. In this case user should take phase synchronization as additional timing into account. As frequency hops the PLL phase will change and it will take the additional time to do phase sync as mentioned above. This happens at the hop stage not at the initial stage.

MCS SUBSTATES (INTERNAL MCS STATE TRANSITION)

MCS Ready Substate

- Definition: ADRV9001 clock is switched from CLK PLL to Reference clock. MCS is initialized. Ready to receive MCS pulses.
- For current release 0.13, if the MCS command is sent with power saving mode > 0, ADRV9001 will return error
- All ADRV9001 chips can enter MCS Ready asynchronously. This means BBIC will wait for the ready status notified by all ADRV9001 chips before issuing MCS pulses.

MCS Transition Substate

- Definition: ADRV9001 is in MCS pulse 1-6 transition but not finished.
- BBIC or clock chip sends MCS pulses to all ADRV9001 chips synchronously. BBIC monitors MCS status and restart MCS pulses if needed to all ADRV9001 chips.
- Internally ADRV9001 keeps monitoring MCS status. After detecting the 5th MCS pulse, switch reference clock to clock PLL.

MCS Done Substate

- Definition: MCS procedure is done. ADRV9001 is ready to move to PRIMED state
- After received the 6th MCS, the substate is changed to MCS Done. ADRV9001 waits for toPrimed command or MCS command again to re-run MCS.
- If MCS is not disabled, toCalibrated command would bring ADRV9001 back to MCS Done.

PROCEDURE

Before issuing MCS pulses, ADRV9001 needs to be in CALIBRATED state. BBIC can monitor MCS synchronization status by calling adi_adrv9001_Mcs_Status_Get. After issuing one or all of the MCS pulses, this function can be used to check the synchronization status of the analog and digital subsystems.

1. Entering MCS Ready substate

Make all ADRV9001 chips and channels of interest to MCS READY substate by issuing adi_adrv9001_Radio_ToMcsReady(). This function will transition all ADRV9001 channels from the CALIBRATED state to MCS READY substate. This is necessary before issuing all MCS pulses.

2. Configure and send MCS pulses

BBIC or clock chip will configure MCS with 6 pulses as inputs to ADRV9001 MCS pin. The timing for these pulses is specified as in Table 33. This will be a BBIC specific function implemented by the user. adi_fpga9001_Mcs_Configure() and adi_fpga9001_Mcs_Start() are examples in the SDK provided for the EVB system FPGA board. Here user needs to specify the pulse width and wait time and make sure they don't exceed the minimum requirement. While MCS is running, ADRV9001 should be in MCS transition substate.

3. MCS done

After all 6 pulses are received, MCS will be finished. At this point, ADRV9001 should be in MCS DONE substate.

SAMPLE DELAY AND READ DELAY

ADRV9001 also supports the scenario where data coming from/going to SSI interface for multiple channels have different delays. This is typically more important on the transmit side where data coming to SSI interface have different delays. On the receive side, the delay can be manipulated by the baseband processor.

To mitigate this delay difference, ADRV9001 provides the measurement from the last MCS pulse edge to the Tx strobe for different channels. This measurement will be effective for all channels associated to MCS, as Tx strobe of each channel may have a different delay relative to the MCS pulse signal. The function that achieves this is adi_adrv9001_Mcs_TxMcsToStrobeSampleLatency_Get, which takes the channel number and provides the latency, in samples, from MCS to the Tx strobe for the specified channel.

Once the latency is measured for all channels, let's call this MCS_to_Strobe latency, then user can calculate the Sample Delay and Read Delay separately.

Sample Delay – this delay will help delay the ADRV9001 internal FIFO read point so that if MCS_to_Strobe is too big, then it will delay the FIFO so that FIFO saves less irrelevant information.

Read Delay – this delay will help delay FIFO read time. Once information is fed in the FIFO, this value will guarantee the FIFO has samples for all channels before reading.

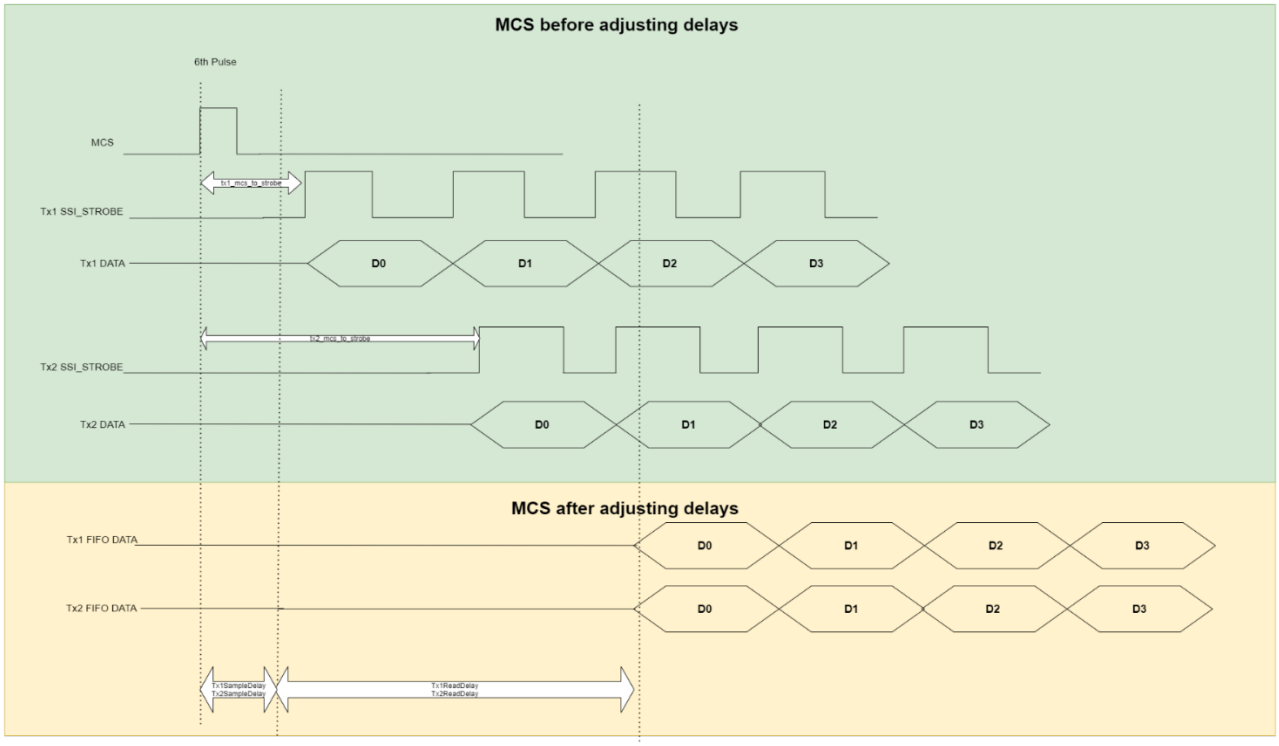


Figure 86 Tx MCS to Strobe Timing Diagram

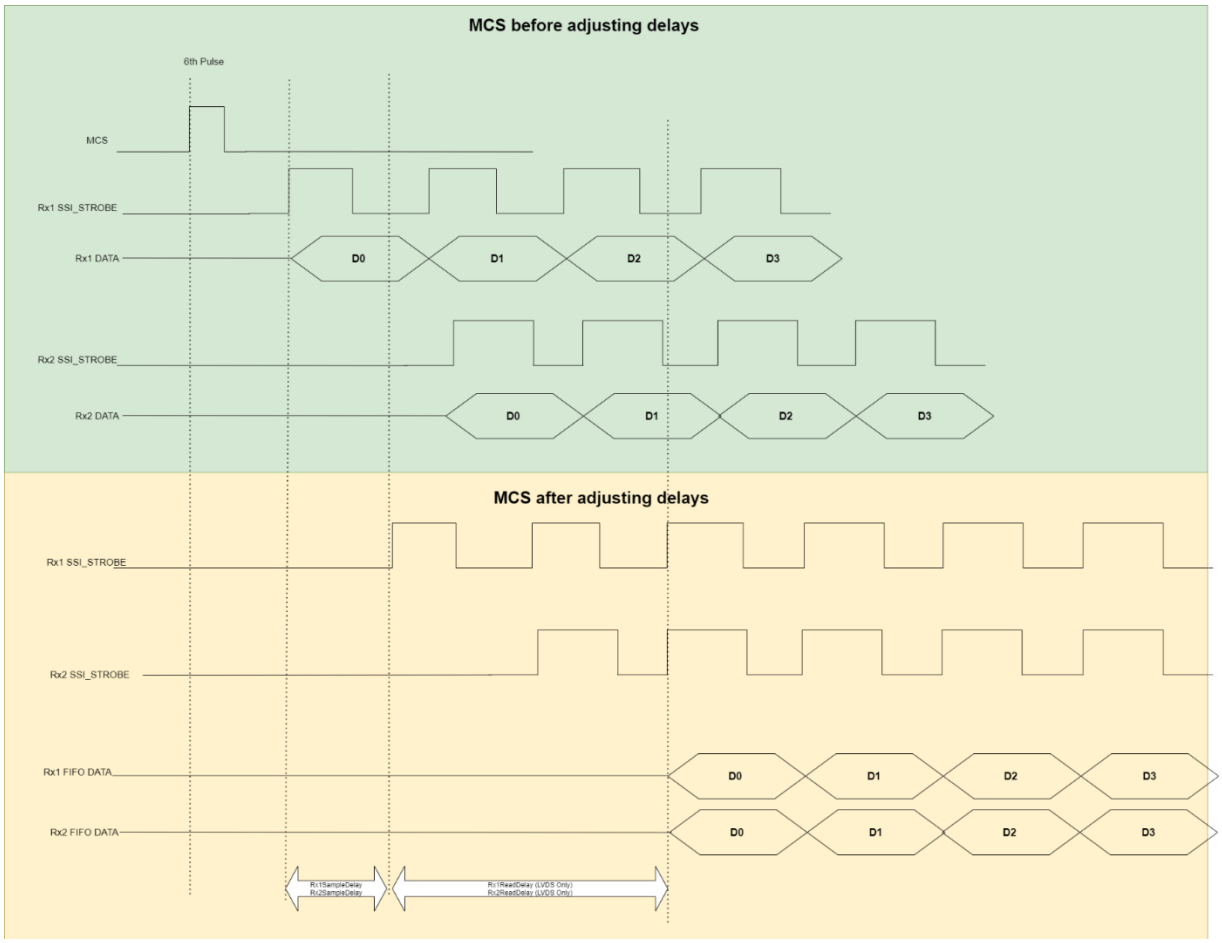


Figure 87 Rx MCS to Strobe Timing Diagram

User needs to calculate them in the following way:

Tx:

- Each channel is independent from other channels
    - sampleDelay = MCS_to_Strobe latency – 1
    - readDelay = 4
- Syncing Multiple channels: (ex. TX1 and TX2, or multiple Devices)
    - sampleDelay = Min (all MCS_to_Strobe latency) – 1
    - readDelay = Max (all MCS_to_Strobe latency) - Min (all MCS_to_Strobe latency) + 4

Rx:

- Each channel is independent from other channels
    - sampleDelay = UserDefined
    - readDelay (LVDS Only)= UserDefined (minimum of 1 to compensate for clock timing in LVDS from analog, imposed by API)
- Syncing Multiple channels: (ex. RX1 and RX2, or multiple Devices)
    - sampleDelay = UserDefined
    - readDelay (LVDS Only)= UserDefined (minimum of 1 to compensate for clock timing in LVDS from analog, imposed by API)

Once sample delay and read delay are calculated, user needs to set them by adi_adrv9001_Mcs_ChannelMcsDelay_Set.

Note that the MCS_to_Strobe delay difference between different channels should be within 2 samples.

Note on the Rx side, functions to calculate sample delay and read delay are user defined. We recommend user to use the default values provided in the software. And we also provide user to do adjustment using the above method. Fundamentally this is controlled by the user from the baseband processor, and user will have full control on how data is timed. In contrast, for Tx side, ADRV9001 will not have control how data will be passed in, therefore will try to adjust the different delays and try to align the data in time.

PHASE SYNCHRONIZATION

The preliminary characterization of the phase synchronization is shown below. More updates will be provided in future releases.

The following plots are trying to show the effect of the phase synchronization function. The test is done with two Rx signals coming from the same signal source. Signal is using CW tones. The reference clock frequency is set to 38.4 MHz. We use LTE61.44 profile for this test. The test is run from 100 MHz to 3GHz with a 100 MHz step, using internal LOs. We also used 5 different ADRV9001 chips for the test.

Figure 88, shows the effect of phase synchronization for the receiving signals. Signals are captured at the interface for phase error calculation

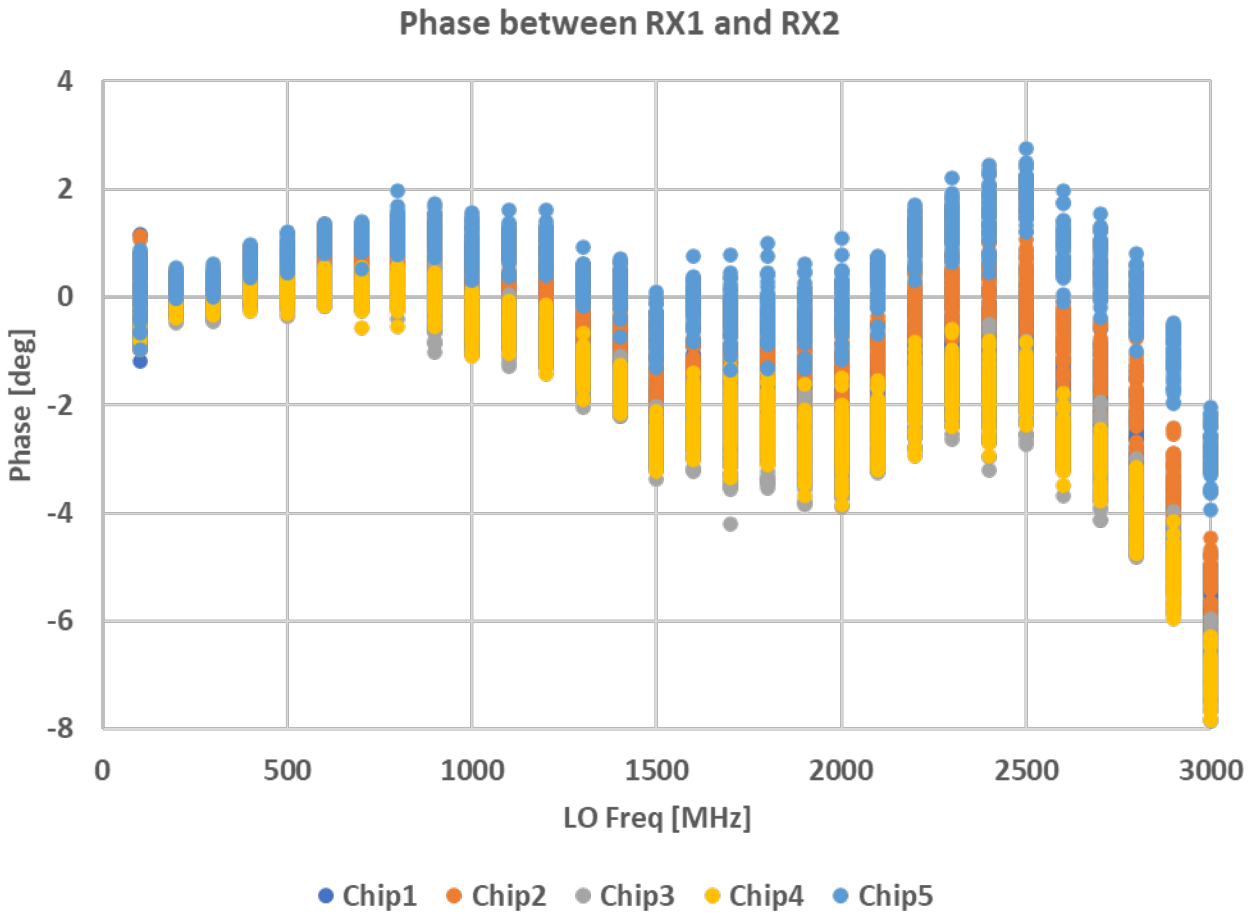


Figure 88 Phase Error after synchronization between two Receivers

Figure 89 shows the phase error after synchronization of the two LOs, these are internal register reads but should be accurate.

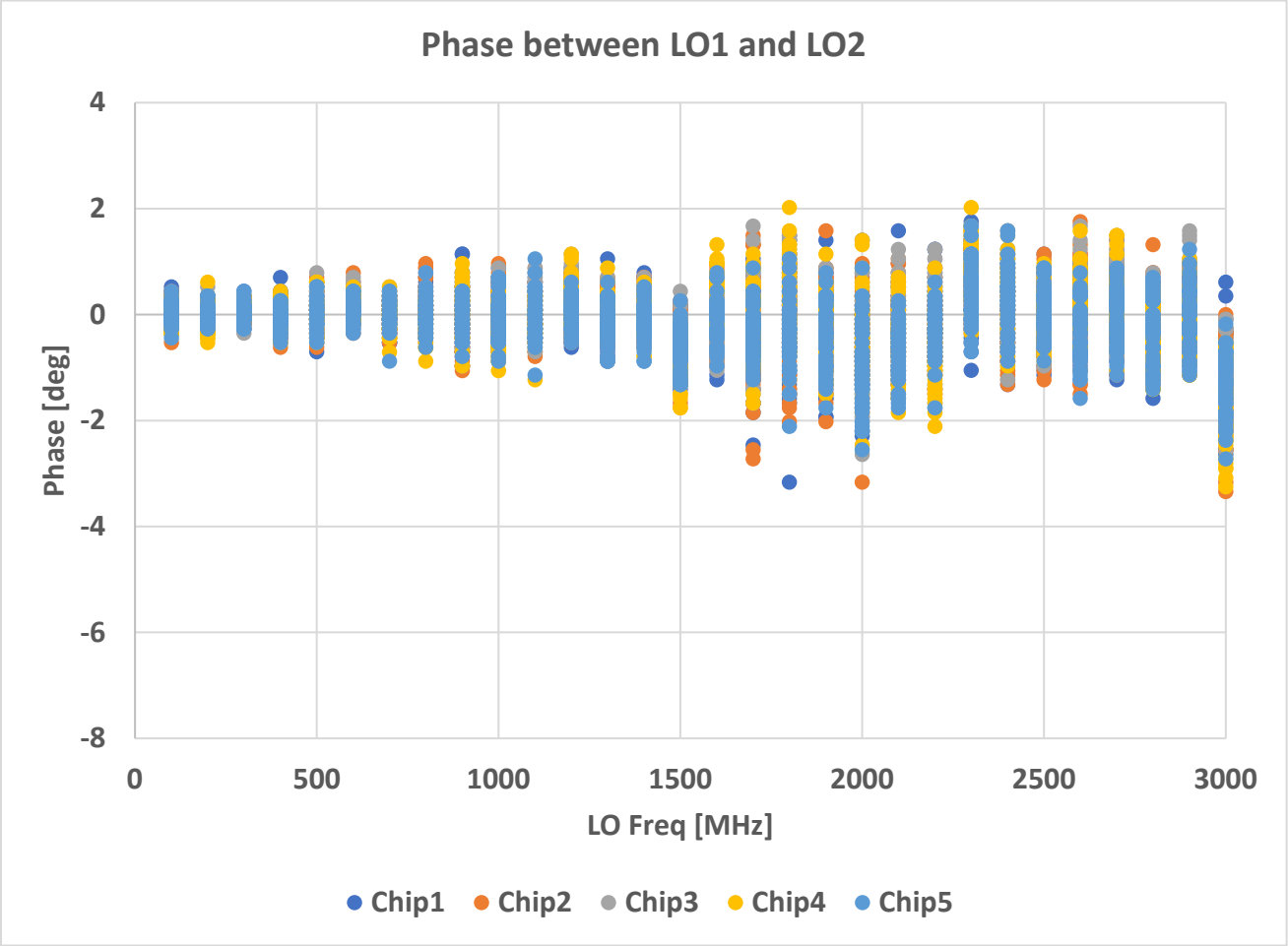


Figure 89 Phase Error after synchronization between two LOs

SYNTHESIZER CONFIGURATION AND LO OPERATION

The ADRV9001 family devices employ four phase-locked loop (PLL) synthesizers: clock, RF (×2), and auxiliary. Each PLL is based on a fractional-N architecture and consists of a common block made up of a reference clock divider, phase frequency detector, charge pump, loop filter, feedback divider, and digital control block, and either a 1 or 4 core voltage-controlled oscillator (VCO). The VCO has a tuning range of 6.5 GHz to 13 GHz. Each PLL drives its own local oscillator (LO) generator: RF LOGEN, aux LOGEN, and CLKGEN. The output of the LOGEN block is a divided version of the VCO frequency. No external components are required to cover the entire frequency range of the device. The reference frequency for the PLL is scaled from the reference clock applied to the device. Figure 90 illustrates synthesizer interconnection and clock/LO distribution block diagram.

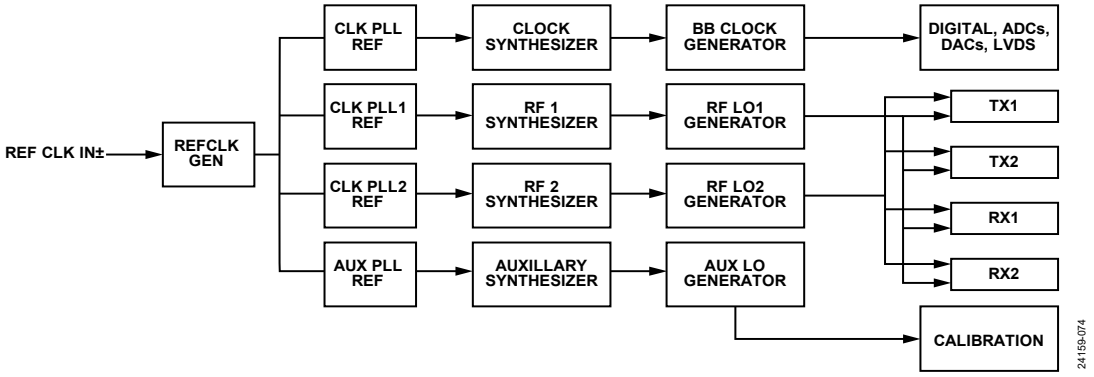


Figure 90. Synthesizer Interconnection and Clock/LO Distribution Block Diagram

Each receiver channel can be used as an observation receiver (ORx) for transmitter channels as shown in Figure 91.

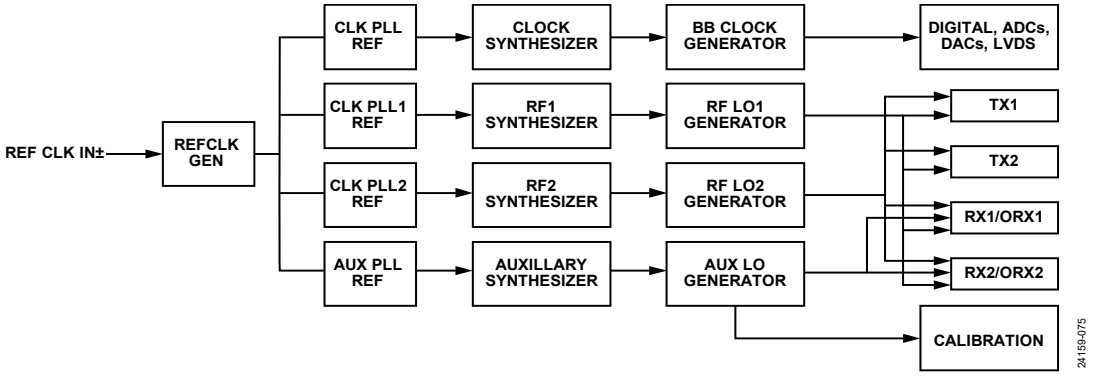


Figure 91. Synthesizer Interconnection and Clock/LO Distribution Diagram (Receiver Channels Configured as Observation Receivers for Transmitter Channels)

CLOCK SYNTHESIZER

The clock synthesizer is used to generate all the clocking signals necessary to run the device. The synthesizer uses a single core VCO block. The reference frequency for the clock PLL is scaled from the device clock by the reference clock generator. Reconfiguration of the clock synthesizer is typically not necessary after initialization. The most direct approach to configuration is to follow the recommended programming sequence and use provided API functions to set the clock synthesizer to the desired mode of operation. The clock generation block of the clock synthesizer provides clock signals for the high speed digital clock, receiver ADC sample and interface clocks, transmitter DAC sample and interface clocks, and LVDS interface clocks.

RF SYNTHESIZER

The device contains two RF PLLs. Each RF PLL uses the PLL block common to all synthesizers in the device and employs a 4 core VCO block which provides a 6 dB phase noise improvement over the single core VCO. As with the other synthesizers in the device, the reference for RF PLL 1 and RF PLL 2 are sourced from the reference generation block of the device. The RF PLLs are also fractional-N architectures with a programmable modulus. The default modulus of 8,388,473 is programmed to provide an exact frequency on at least a 5 kHz raster using certain reference clocks which are integer multiples of 38.40 Hz. The RF LO frequency is derived by dividing down the VCO output in the LOGEN block. The tunable range of the RF LO is 30 MHz to 6000 MHz.

A switching network is implemented in the device to provide flexibility in LO assignment for the two RF LO sources. The switching network is shown in Figure 92 and Figure 93.

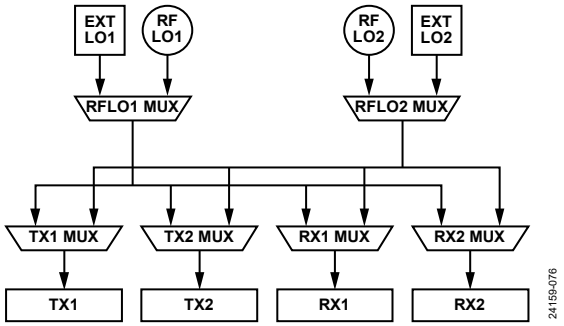


Figure 92. LO Switching Network

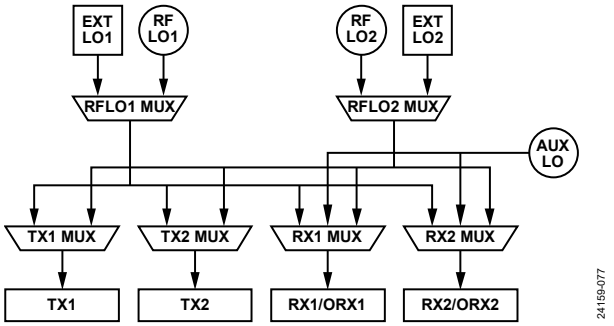


Figure 93. LO Switching Network (Receiver Channels Configured as Observation Receivers for Transmitter Channels)

Note that depending on the application, user has the ability to select best phase noise or best power saving options for better optimization. The option, best phase noise, only works for LO frequency under 1 GHz.

AUXILIARY SYNTHESIZER

An auxiliary synthesizer is integrated to generate the signals necessary to calibrate the device. This synthesizer uses a single core VCO. The reference frequency for the auxiliary synthesizer is scaled from the device clock via the reference clock generation system. The output signal is connected to a switching network and injected into the various circuits to calibrate filter bandwidth corners, or into the receive signal chain as an offset LO. Calibrations are executed during the initialization sequence at startup. There should be no signal present at the receiver/observation receiver input during tone calibration time. Calibrations are fully autonomous. During the calibration, the auxiliary synthesizer is controlled solely by the internal ARM microprocessor and does not require any user interactions. The auxiliary LO signal is also available as an LO source for the observation receiver mixers.

EXTERNAL LO

The device is provisioned with two external LO ports. These ports are available as a pair of balls and are configured to be input for external LO signals.

External LO can receive a signal between 60 MHz and 12 GHz through a matched differential impedance of 100 Ω and delivers a programmable signal between 30 MHz and 6 GHz as the LO for transmitters and receivers in the device. Amplitude must be maintained between ±6 dBm. For more information refer to External LO Port Impedance Matching Network paragraph.

Singled-ended external LO in is also supported. The matched singled-ended impedance is 50 Ω. On-chip duty cycle correction circuit can correct limited range of external LO duty cycle error if it is not 50%.

The user can also enable external LO with 1× divider for up to 1 GHz.

Single-Ended vs. Differential External LO

Note the current eval board only supports differential external LO, however user is not restricted to use single-ended LO in their end system. User must change clocks.ext1LoType and clocks.ext2LoType from 0 to 1. This can be found in the Enum below

```
enum adi_adrv9001_ExtLoType {
    ADI_ADRV9001_EXT_LO_DIFFERENTIAL = 0,
    ADI_ADRV9001_EXT_LO_SINGLE_ENDED = 1}
```

Note only frequencies from 500 MHz to 1000 MHz are supported for single-ended mode. Figure 94 shows the RF LO generation diagram.

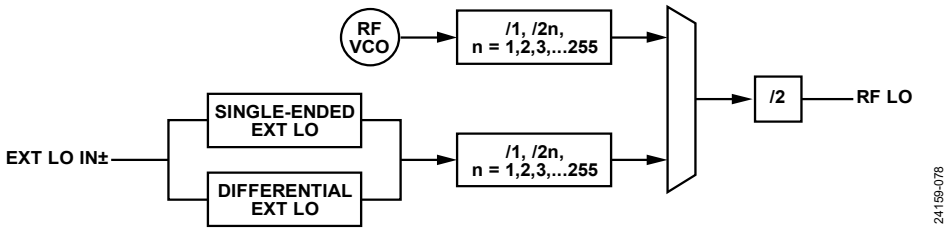


Figure 94. RF LO Generation Diagram

RF PLL Loop Filter Recommendations

For optimal phase noise and EVM performance, a lookup table of RF PLL loop filter bandwidth settings is implemented in ADRV9001 firmware. ADRV9001 automatically selects best RF PLL loop filter configuration based on LO frequency. Alternatively, user can program its own RF PLL loop filter bandwidth following instruction outlined in Loop Filter Configuration paragraph.

PLL Phase Noise

The Figure 95 shows the typical PLL phase noise contributors. For low offset frequencies reference clock dominates the phase noise, and for high offset frequencies, VCO noise dominates the phase noise. User can optimize the phase noise by:

- Provide better reference clock source
- Provide higher reference clock frequency (PFD)
- Adjust loop filter bandwidth to trade-off between close-in band and far-out band noise

When changing the loop filter bandwidth, typically consideration is the wider the bandwidth, the better close-in band noise, but the worse the far-out band noise. User should trade-off between the two to find the optimal setting for the specific application.

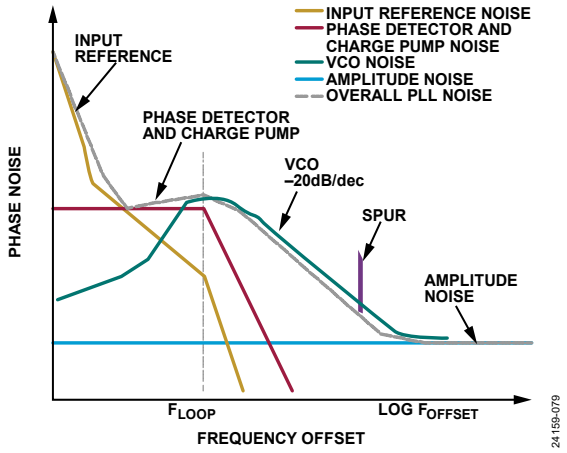


Figure 95. PLL Phase Noise Contributors

Following is an example that PLL phase noise is highly dependent on the PFD frequency (REF_CLK). With a higher PFD frequency, a better phase noise can be achieved.

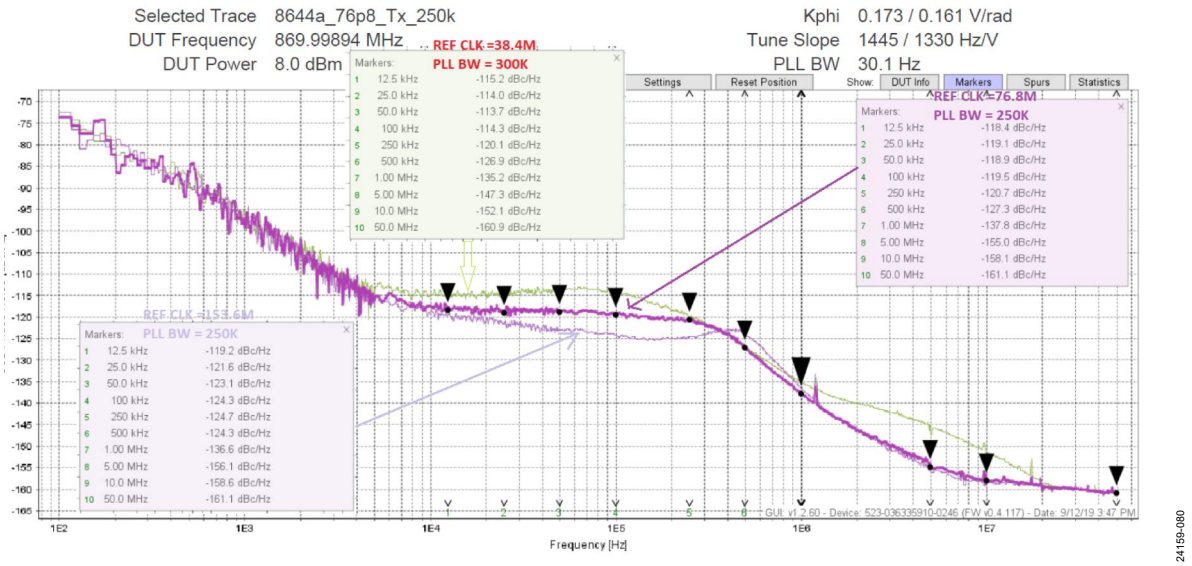


Figure 96. Effect of PFD (REF_CLK) Frequency on PLL Phase Noise

Another example is shown below where a trade-off can be made between close-in phase noise and far-end phase noise by adjusting the loop filter bandwidth. As stated previously the higher the loop filter bandwidth, the better the close-in noise but with the scarification of the far-end noise, and vice versa.

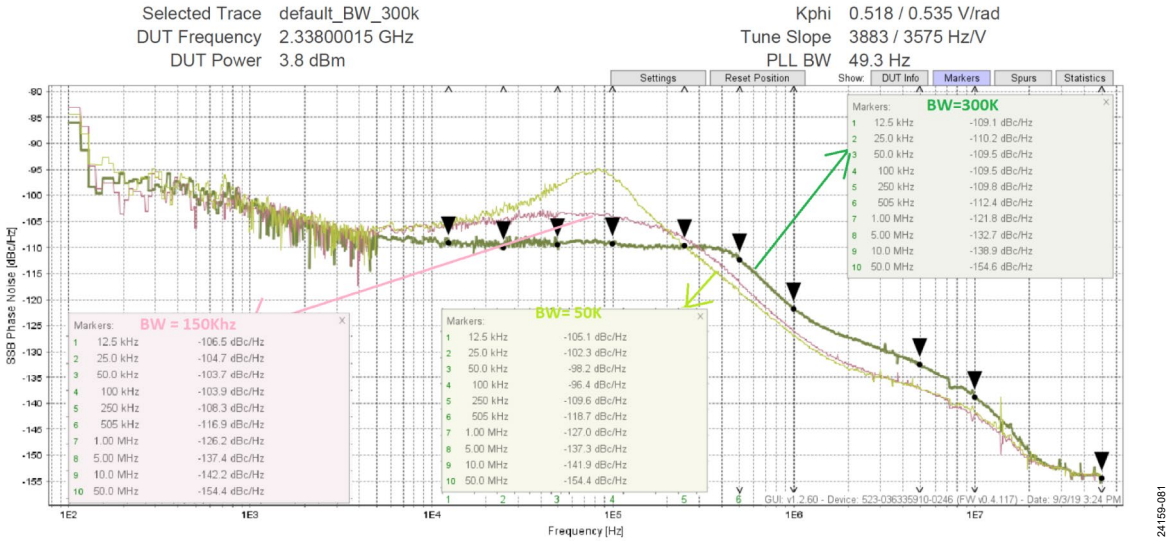


Figure 97. Effect of Loop Filter Bandwidth on PLL Phase Noise

API OPERATION

Data Structure and Enums

Table 34. Data Structures Related to LO Operation

Data Structure	Description
adi_adrv9001_Device_t	Data structure to hold ADRV9001 device instance settings.
adi_adrv9001_Carrier_t	Carrier structure for carrier configuration.
adi_common_Port_e	Enumeration of port types.
adi_common_ChannelNumber_e	Enumeration of channel numbers.
adi_adrv9001_LoGenOptimization_e	Enum of LO Optimization.
adi_adrv9001_PllCalibration_e	Enum for PLL calibration mode.
adi_adrv9001_Carrier_t	Carrier frequency configuration
adi_adrv9001_Pll_e	Enum of PLL selections.
adi_adrv9001_PllLoopFilterCfg_t	Data structure to hold Synthesizer PLL Loop filter settings.

API Commands

More detailed information including parameters, return values is provided in the doxygen document supplied with SDK package.

Table 35. API Commands Related with LO Configuration Settings

API Function	Description
adi_adrv9001_Radio_Channels_EnableRf()	Enable or disable RF channel (Transition the specified channel between RF_ENABLED and PRIMED states) (this function only works if channel is in SPI mode)
adi_adrv9001_Radio_Channel_Prime()	Prime the specified channel (Transition the specified channel between CALIBRATED state to PRIMED states)
adi_adrv9001_Radio_PllStatus_Get()	Checks if the PLLs are locked.
adi_adrv9001_Radio_PllLoopFilter_Set()	Configures the loop filter for the specified PLL.
adi_adrv9001_Radio_PllLoopFilter_Get()	Gets the loop filter configuration for the specified PLL.
adi_adrv9001_Radio_Carrier_Configure()	Sets the carrier configuration for the given channel.
adi_adrv9001_Radio_Carrier_Inspect()	Inspects carrier configuration.

LO Change Procedure

To set the LO frequency to a particular channel, user must:

1. Verify the internal ARM microprocessor has been initialized.
2. If device is on RF_ENABLED state, user needs to first set it to PRIMED state
   a. If in SPI mode, this can be done by calling adi_adrv9001_Radio_Channels_EnableRf().
   b. If in PIN mode, this can be done by moving TX_ENABLE or RX_ENABLE to LOW state.
   c. Once in PRIMED state, user needs to set to CALIBRATED state, calling adi_adrv9001_Radio_Channel_Prime().
3. Once device is in CABLIRATED state, Set the LO frequency by calling adi_adrv9001_Radio_Carrier_Configure().
4. Lastly user needs to bring the device back to RF_ENABLED state. First bring it back to PRIMED, then to RF_ENABLED, calling the same functions above but in reverse order.

User does not have specific control over CLK_PLL. Configuration of CLK_PLL is done at initialization time.

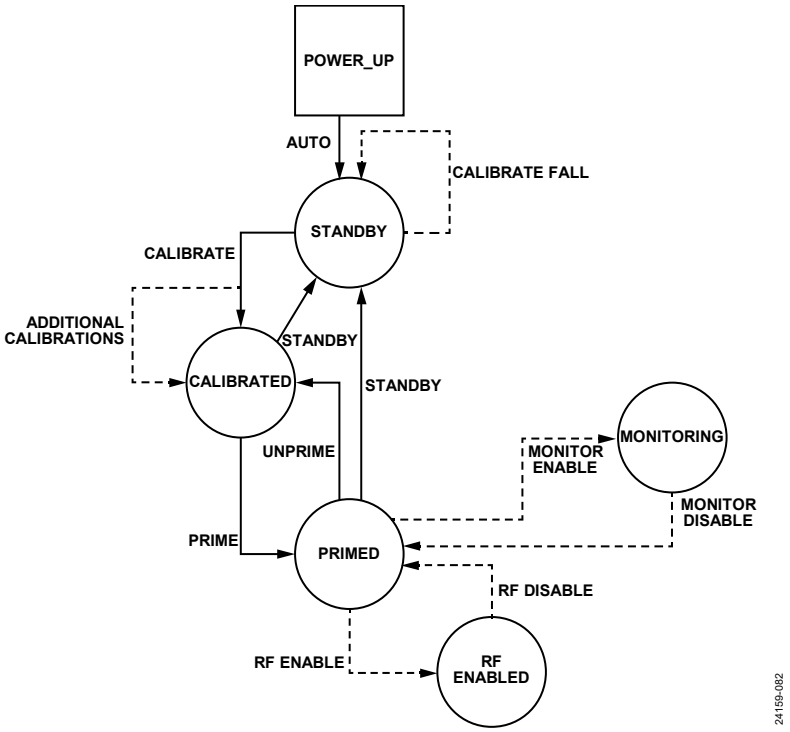


Figure 98. Device State Machine

Note, when changing the LO, user needs to make sure all channels that utilize the LO is configured. For example, if Tx1 and Rx1 are using LO1, and user wants to change LO1, then user needs to configure both Tx1 and Rx1 to achieve that. If all channels (Tx1 Tx2 Rx1 Rx2) are using the same LO, and user needs to change the LO, then user needs to configure all channels (Tx1 Rx2 Rx1 Rx2) to achieve that.

Loop Filter Configuration

Currently, the loop filter is hard-coded in adrv9001_RadioCtrlInit. This function is called at initialization time. However, user does have access to a public API, adi_adrv9001_Radio_PllLoopFilter_Set to manually change the loop filter settings. This function must be called at CALIBRATED state, similar to setting the LO frequency.

FREQUENCY HOPPING

Before delving into the frequency hopping feature details, the user is recommended to read the Multichip Synchronization section and the Timing Parameters Control section of this document.

Frequency hopping allows the user to quickly switch radio signals among different frequency channels. For ADRV9001, this is achieved by retuning the PLL before switching to the frequency channel. There are two local oscillators (LO) inside ADRV9001, therefore we can ping pong between the two LOs. This means while one LO is being used for on-air signal transmitting for one frequency, the other LO can be used to prepare for the next frequency. This makes very fast frequency hopping possible. Besides ping pong between two LOs, ADRV9001 also support single LO for frequency hopping. This allows LO to be retuned while it's off air. This way the user can separately hop Rx1/Tx1 and Rx2/Tx2 as well. More of the operation modes will be described in later sections.

This section explains the key parameters of frequency hopping, namely the HOP signal and Tx and Rx Setup signals. They play a key role in understanding how frequency hopping in ADRV9001 operates and crucial to understand more complicated timing configurations.

Channel use cases and the modes of operation for frequency hopping are also shown. ADRV9001 supports frequency hopping for Tx only, Rx only and TRx. The propagation delay for the data path must be considered as well, as they will affect the channel use case option.

The proposed modes of operation are based on the allowed time for the PLLs to retune. The number of LOs, number of allowed channels, and calibration modes are different depending on the different modes of operation.

The concept of frequency hopping table is explained. All frequency parameters will be provided in a frequency hopping table in all modes of operation. The required parameters of a frequency hopping entry are shown. The different modes of indexing the table are shown as well.

KEY SIGNALS

The frequency hopping framework involves reconfiguring the analog and digital components to hop to different frequencies.

Figure 99 shows a typical frequency hopping timing diagram. In this diagram, we have the HOP signal and the Tx and Rx setup signals, frequency select, and the frames on the air. Tx and Rx setup signals are Tx and Rx ENABLE hardware signals. In the context of frequency hopping, they are repurposed as Tx and Rx setup signals. More information on this is shown in the later sections

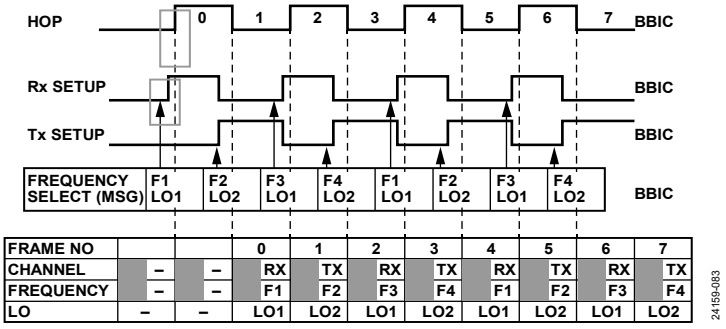


Figure 99. Typical Timing Diagram for Frequency Hopping

Figure 100, shows a Tx/Rx frequency hopping using LO muxing. LO muxing is one of the FH modes which will be discussed in later sections. In this diagram we observe Tx/Rx setup signals and sampled at HOP edges, and then indicate the next frame will be Tx or Rx frame respectively.

It is worth noting that in LO muxing mode, the information of which channel the frame needs to be at frame n (Tx or Rx), needs to be given at (n-2) frame time.

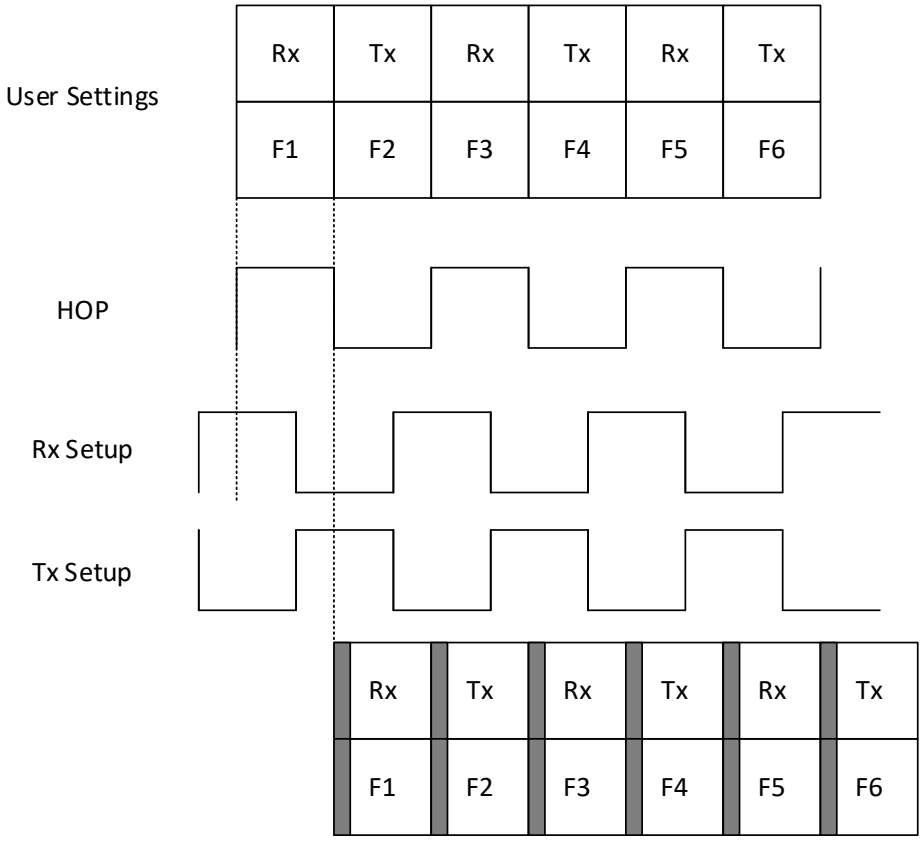


Figure 100 Tx-Rx Frequency Hopping Using LO Muxing

Figure 101, shows a Tx/Rx frequency hopping using LO retune. In contrast, the frame will the information of which channel the frame needs to be at frame n (Tx or Rx), needs to be given at (n-1) frame time.

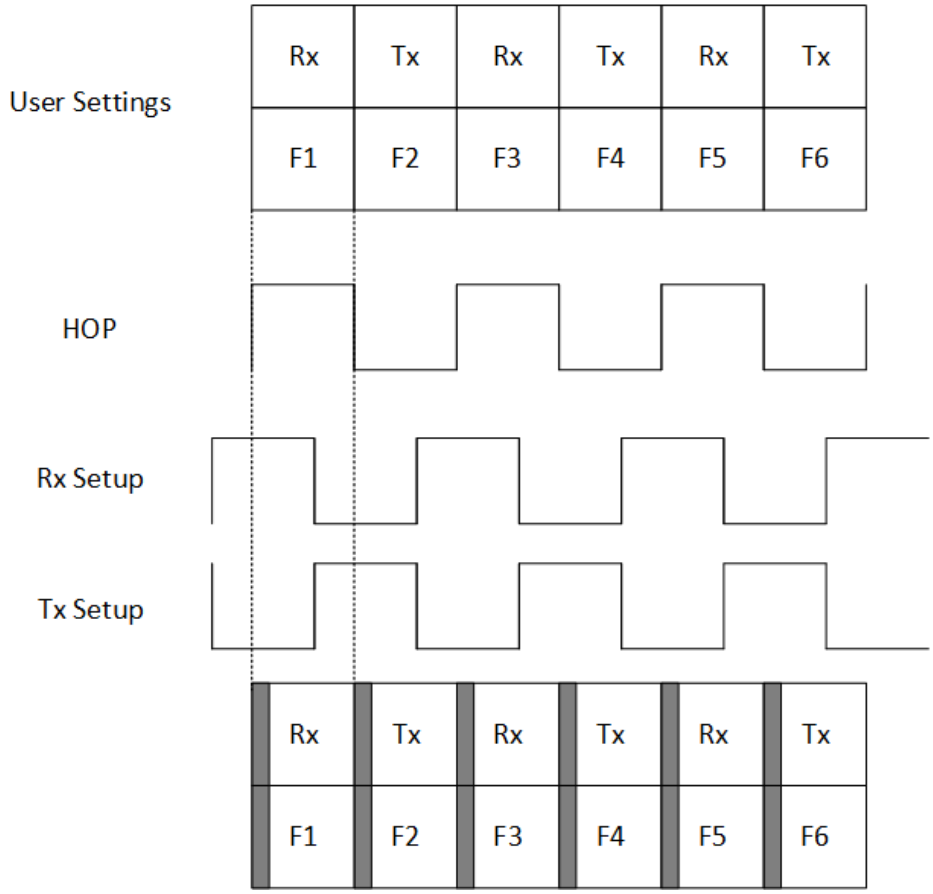


Figure 101 Tx-Rx Frequency Hopping Using LO Retune

Hop Signal and Hop Frame

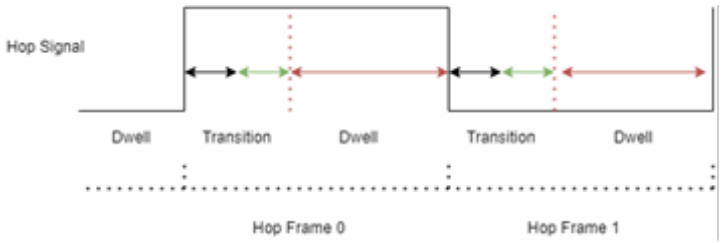


Figure 102. Hop Signal and Hop Frame

In frequency hopping, the period in which a channel is enabled is defined as the hop frame. During each hop frame, data can be operated on a new carrier frequency with either Rx or Tx. A hop frame is made up of a transition and a dwell period. The transition period is the setup time for the hop frame.

The **transition** period is, at a minimum, the time required to setup the Rx or Tx channel and switch to a new operating frequency (first portion of transition time in Figure 102). However, it can be whatever length the user decides for their framing requirements (second portion of transition time in Figure 102).

The **dwell** period is the 'on air' time, where a channel is in RF enabled state. The dwell period can be any length of time the user requires to operate on a frequency.

HOP signal marks the beginning and the end of a Transmit or a Receive frame. The HOP signal is triggered by a DGPIO pin, which can be assigned from any of the available DGPIO pins. Each edge of the HOP signal marks both the possible start and end of a hop frame.

Channel Setup Signal

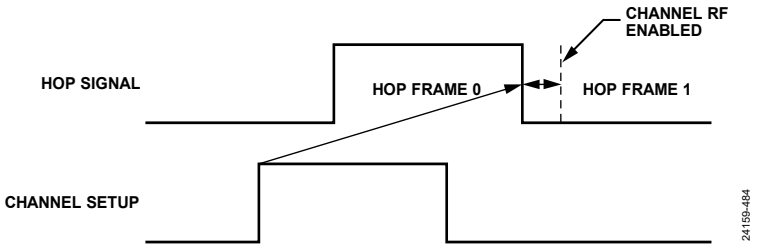


Figure 103 Channel Setup Signal for LO Muxing

The channel enable pins (namely Tx/Rx_ENABLE pins), used in non-FH operation, are repurposed in frequency hopping to signal if an upcoming HOP frame is operating on an Rx or Tx channel. These pins are redefined as Rx/Tx setup; however, they are the same dedicated channel enable hardware pins, which are used to enable an Rx or Tx channel in non-frequency hopping mode.

As seen in Figure 103, this is the LO muxing mode, the channel setup is used to signal a channel is enabled one frame in advance. For example, the channel setup signal is high prior to the start of Hop Frame 0, but that channel is not enabled until Hop Frame 1.

For LO retune mode, we still use the HOP signal to sample Tx/Rx setup signal. However the RF channel is enabled in HOP frame 0 instead of HOP frame 1, shown in Figure 104.

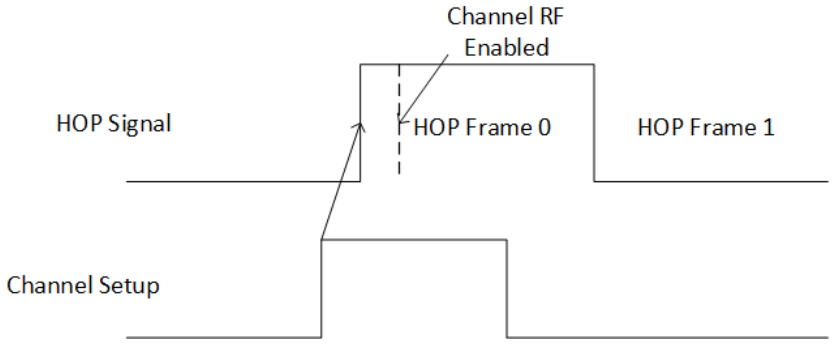


Figure 104 Channel Setup Signal for LO Retune

The frequency information comes from the BBIC. Before each Tx or Rx setup, ADRV9001 expects to get some message (this may come in various forms which is discussed in later sections) which indicates a frequency. Prior to each hop, the channel (Tx or Rx) information and the frequency information are obtained.

Note Tx setup signal has special meaning. In LO muxing mode, Tx setup falling edge indicates the start of the interface. In LO retune mode, Tx setup rising edge indicates the start of the interface. More information will be shown in details.

MODES OF OPERATION

ADRV9001 allows the user to achieve various framing requirements by providing three modes of operation.

Table 36. Frequency Hopping Modes of Operation

Mode	Transition Time	Total frame duration (transition + dwell)	PLLs	PLL Return Time	PLL Cal Mode	Channel
LO mux with hop table preprocess	< channel setup + lo retune	> 13 us	2 LOs	2 transitions + 1 dwell	Fast Cal mode	Single (1T1R)
LO mux with hop table real time update	< channel setup + lo retune	> 25 us	2 LOs	1 transitions + sub 1 dwell	Fast Cal mode	Single/Dual (1T1R or 2T2R diversity)
LO retune with hop table real time update	> channel setup + lo retune	> max(25 us, LO_retune_time)	1 LO	Sub 1 transitions	Fast Cal mode	Single/Dual (1T1R or 2T2R diversity)

Currently, all modes of frequency hopping operation within ADRV9001 use a fast PLL calibration. The modes are differentiated by the user's transition time and dwell time requirements. ADRV9001 defines two modes of PLL usage: LO Muxing and LO Retune.

Note there are two modes of PLL calibration, one is fast one is normal. In terms of phase noise, there is no difference between the fast and normal modes. Normal mode tracks temperature over time where fast mode does not. Therefore fast mode takes less time. For frequency

hopping, typically the frames are fairly short, therefore fast should be sufficient. The PLL calibration mode can be set with adi_adrv9001_PllCalibration structure.

LO Muxing

For short transition times, ADRV9001 requires two LOs to be used in a ping pong operation. This means while one PLL is used for one frame, the other PLL is being retuned for the next frame. During the transition time, the LOs are swapped.

LO muxing always uses 1 frame delay, examples shown in Figure 105.

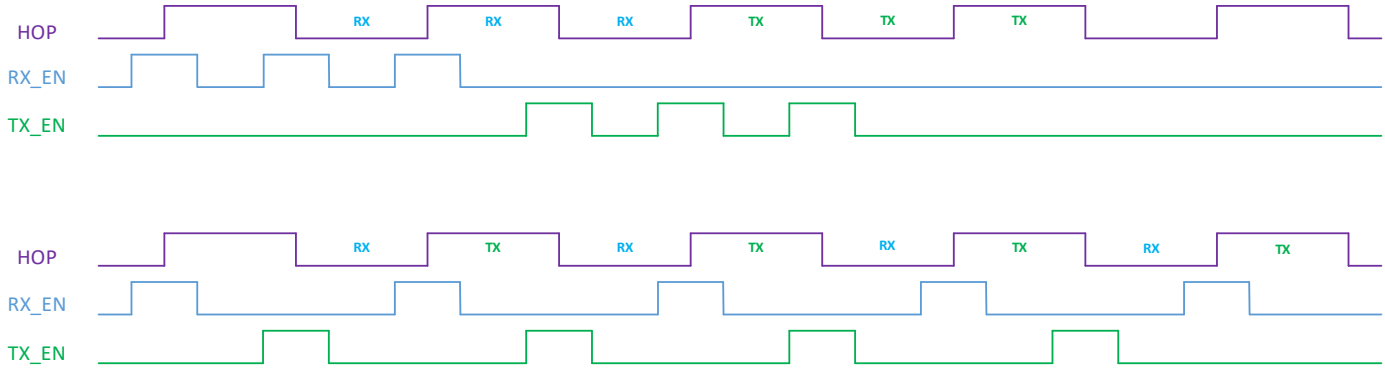


Figure 105 LO Muxing Timing Diagram

LO Retune

For longer transition times, the PLL can be retuned for the starting frame within the transition time, if that time is greater than the sum of the PLL lock time and channel bring up time.

LO retune always uses 0 frame delay, examples shown in Figure 106.

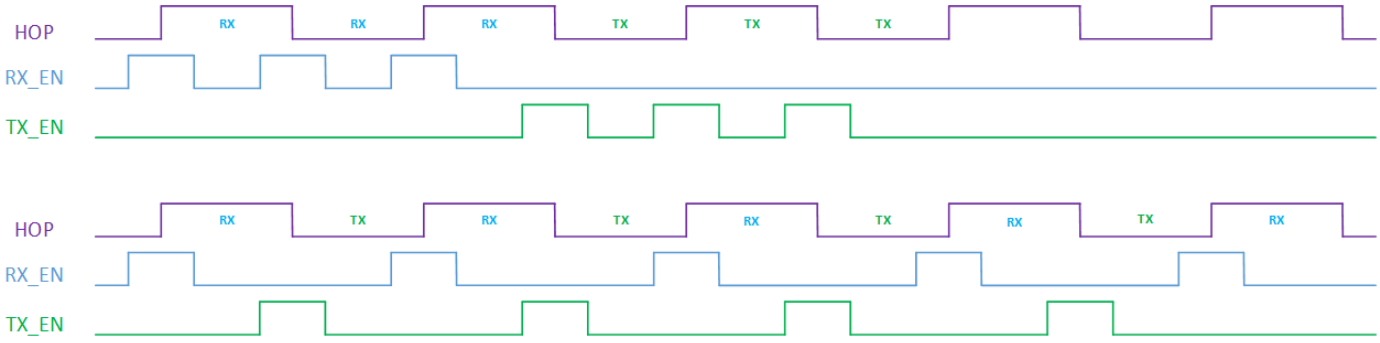


Figure 106 LO Retune Timing Diagram

PLL lock time is discussed in section LO Retune.

Currently user should rely on the actual test result on the LO retune time, with the evaluation platform, as currently the channel bring up time is difficult to quantify because they differ in different cases.

Pre-Process Mode vs. Real-Time Process Mode

Pre-process Mode refers to the frequency hopping tables being sent to ADRV9001 and processed before frequency hopping operation begins.

Real-time Process Mode refers to the frequency hopping table not being processed at initialization stage but at the hopping stage. At each hop, the next entry in the table is read and processed. This allows the user to update the frequency hopping tables on the fly.

These two modes of operations are referring to the frequency hopping tables, and they will be explained in further detail in the frequency hopping table sections

CHANNEL AND PROFILE SELECTION

Depending on the operation mode, operations of dual channel (2T2R, 2T, or 2R) diversity, or single channel (1T1R, 1T, or 1R) is supported.

Table 37. Channel Selection

Channel Use Case	Profile Propagation Delay Requirements	Number of Channels	Selectable Mode
TRx	Propagation delay must be less than the duration of a single hop frame.	1T1R	All modes
		2T2R	LO mux with hop table real time update
			LO retune with hop table real time update
Rx	Propagation delay can be greater than the duration of a hop frame.	1R	All modes
		2R	LO mux with hop table real time update
			LO retune with hop table real time update
Tx	Propagation delay can be greater than the duration of a hop frame.	1T	All modes
		2T	LO mux with hop table real time update
			LO retune with hop table real time update

FREQUENCY HOPPING OPERATION RANGES

Operation Ranges

- The full Rx gain range available in regular mode is supported in frequency hopping mode.
- The full Tx attenuation range available in regular mode is support in frequency hopping mode.

FREQUENCY HOPPING TABLE

Hopping Table Definition and Entries

All modes of frequency hopping require the use of a frequency hopping table. A frequency hopping table is a list of frequency and other operational parameters for each hop frame. At initialization, the user will provide ADRV9001 a frequency hopping table, as well as the number of entries, or frequencies, within that table.

ADRV9001 supports the loading of two tables (table A and table B), each with a minimum length of 1, and a maximum length of **64 entries**. (a total of 128 hop entries/frequencies, if two tables are loaded).

An entry in the frequency hopping table is defined as follows.

Table 38. Hop Table Entry

Parameter	Descriptions
hopFrequencyHz	Operating frequency, in Hz.
rx1OffsetFrequencyHz	The intermediate frequency, if a frame is Rx.
rx2OffsetFrequencyHz	
rxGainIndex	Starting gain index, if frame is Rx.
TxAttenuation_mdB	Starting attenuation level, in mdB, if frame is Tx.

The channel(s) which will be enabled each hop frame is not known at initialization. During frequency hopping operation, the user will use the channel setup signals to inform ADRV9001 on which channel to enable. Therefore, each entry in the hop table contains parameters for both Rx and Tx and it processes the entry appropriately, depending on whether an upcoming hop frame is operating on Rx or Tx.

Note there will be examples provided in the TES. And the ranges of the fields should be the same as non-frequency hopping mode.

Frequency Hopping Table Modes

ADRV9001 supports utilizing the frequency hopping tables in several different ways. This section details the available methods of updating, indexing, and switching between frequency hopping tables.

During operation, an entry in the table is read at the channel setup rising edge. Therefore, like the channel information, the hop entry is read one hop frame in advance. This gives time to prepare the channel and retune the LO.

Automatic Increment

The standard mode of operation in ADRV9001 is automatic increment mode. In this mode, frequency hopping begins at index 0, and continues to iterate over the table until the number of table entries has been reached. Once reached, it wraps back around to index 0.

A hop frame entry is read at the channel setup rising edge, and the index to the table is incremented at the hop edge.

Figure 107, shows an example of automatic increment mode. This example has a 3-entry table. Reading of the entries is done by firmware at the HOP edges. Before first rising edge of HOP signal table should be written into ADRV9001 memory by BBIC, and at the HOP edge the table should be ready to be read by firmware.

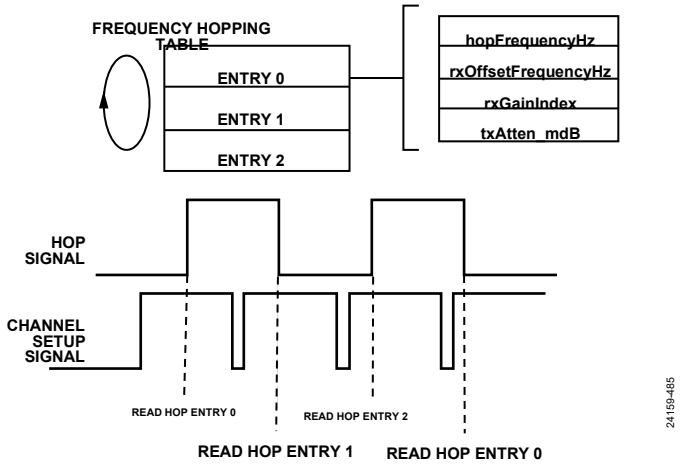


Figure 107. Automatic Table Increment

Index by Pin

Rather than automatically incrementing through a frequency hopping table, the user can use up to 6 DGPIO pins to index any valid entry in the hop table.

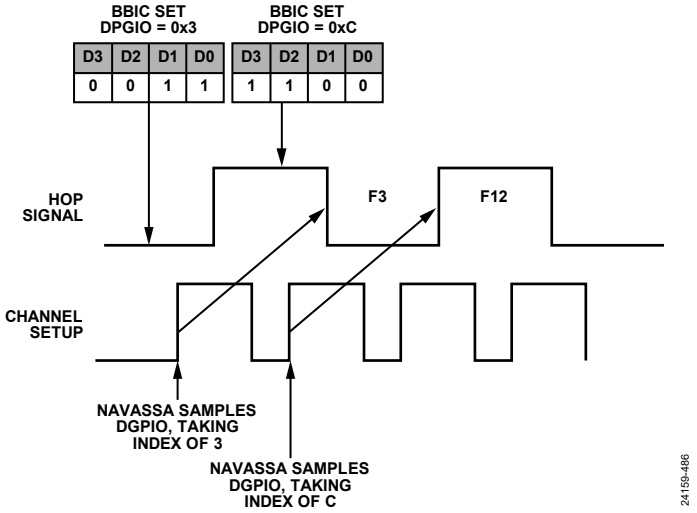


Figure 108. Index by Using DGPIO Pins

At initialization, the user can assign up to 6 DPGIO pins to provide 64 possible indices to a frequency hopping table.

During operation, the user should set the DGPIO pins prior to the upcoming HOP signal edge. User should make sure the DGPIO signals are set and stable by the time HOP signal edge comes. ADRV9001 will sample the DGPIO pins at the HOP signal edge.

The following restrictions apply to this mode;

- Each DGPIO pin represents a bit in the index, the ADRV9001 samples each assigned pin to form a full index to the table.
- The ADRV9001 expects that the lower DPGIO number is the LSB of the index.
- Adjacent DGPIO pins must be assigned (for example, cannot assign DGPIO-0 to Bit 0, and DPGIO-11 to Bit 1, and so on).
- The ADRV9001 samples the DPGIO pins at every channel setup rising edge. It will use any index if the sampled index is deemed valid (meaning, provide a valid index to the frequency hopping table).

Switching Between Two Frequency Hopping Tables

In any frequency hopping mode, ADRV9001 allows the user to switch between either of the frequency hopping tables.

- When a table is switched, the index to the new table is set to 0.
- The new table is read at the next channel setup rising edge.

Automatic Table Ping Pong

Automatic ping pong mode works similarly to regular automatic increment mode. However, once the end of one table has been reached, ADRV9001 will automatically switch to the other table. In this way, the user could continuously increment through, at most, 128 table entries/hop frequencies.

Unlike the manual switch, this mode does not require any external signal from the user. ADRV9001 will handle the switching between the two tables automatically.

Automatic Table with Manual Switch

The user can switch between the two frequency hopping tables at any time using an API or by a DGPIO pin.

The DGPIO pin for hop table switch can be configured at initialization.

When the hop table select pin is low, frequency hopping table A is set as the active table, when the hop table select pin is set high, frequency hopping table B is set as the active table.

The falling or rising edge of HOP Table Select signals to ADRV9001 to switch to the A or B table, respectively.

As per Figure 109, the first channel setup rising edge, after the hop table switch, reads from the newly selected table.

In the example from Figure 109, the new information read from the table will appear on air at frame 4.

Like the PIN method, the user can issue an API to switch between the two tables.

HOP table select pin is sampled at the channel setup rising edge. User must set HOP table select pin prior to the upcoming channel setup rising edge. ADRV9001 resets the frequency hopping table index to zero upon switch. In other words, when the user switches between tables, the new table always starts at entry 0.

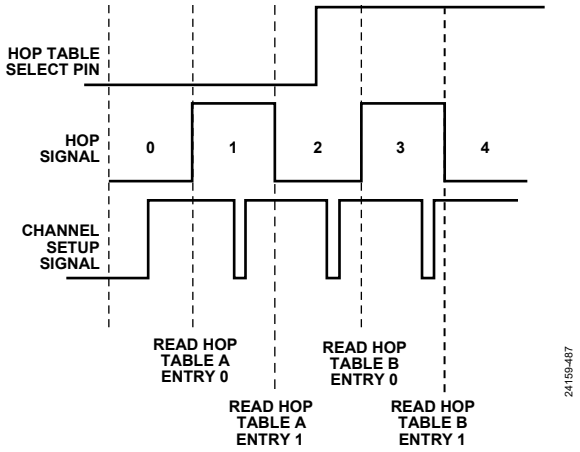


Figure 109. Manual Table Switch

To utilize this feature, user needs to configure structure adi_adrv9001_FhCfg_t->hopTableSwitchGpioConfig.

Index by Pin with Manual Switch

Like the automatic increment mode, the user can use a DGPIO pin or API to switch between tables which indexing the table by pin. ADRV9001 cannot automatically ping pong between tables when the user is in index by pin mode. When user selects index by pin mode, it is no longer a concept of completion of the table, as in the automatic increment mode.

Frequency Hopping Table Real-Time Process

If ADRV9001 is configured for real-time process, the frequency will be configured at real time (Hop edge). This, along with the use of two frequency hopping tables, gives user flexibility to load in new frequency hopping tables during frequency hopping operation.

Example 1: Load New Frequencies with Automatic Ping Pong

In this example, shown in Figure 110, we demonstrate how new frequencies will be loaded on the fly in automatic ping pong mode. Suppose user has already configured the frequency hopping mode to automatic ping pong mode. And table A has a single entry loaded before frequency hopping operation. The following rising/falling edges refer to HOP signal.

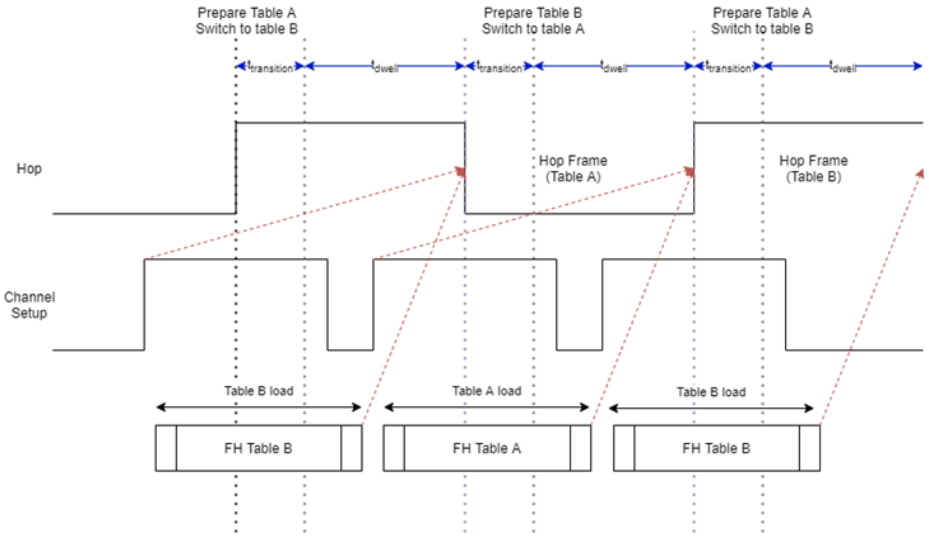


Figure 110. Example 1 Load New Frequencies with Automatic Ping Pong

- At the first rising edge, the ADRV9001 reads the hop table entry from table A (the single entry) and prepares that frequency for the next frame. At this point, the ADRV9001 switches to table B, which currently has no frequency information. Therefore, ADRV9001 will not read from table B yet at his point, it reads from table B at the upcoming HOP edge (the first falling edge).
- The first hop frame starts. During this frame time, BBIC will load a single entry into table B. BBIC must ensure the entry is loaded into ADRV9001 memory prior to the next HOP edge (the first falling edge).
- At the next HOP edge (the first falling edge), ADRV9001 reads the new entry from table B and switches to table A. This frame (from the first falling edge to the second rising edge) uses table B entry.
- At the next HOP edge (the second rising edge), at this point BBIC can load a new frequency entry into table A. This frame (from second rising edge to second falling edge) uses table B. And the process is repeated.

By following this process, the user can provide a new frequency on the fly to ADRV9001 at each HOP edge.

Note that the user does not need to load just a single entry. With automatic ping pong mode, user can load from 1 to 64 entries. However, the user should ensure that the frame timing allows the amount of time required to load the table prior to the automatic switch. We support 1, 4 and 8 entries, the minimum frame length for single entry is 46 us, and minimum frame length for 4 entries is 22 us and, and for 8 entries is 20 us.

Example 2: Loading a Larger Set of Frequencies with Manual Table Switch

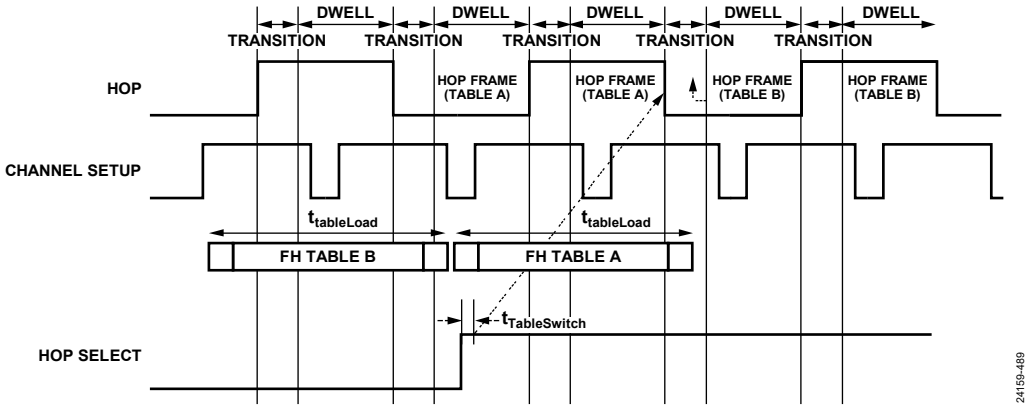


Figure 111. Loading a Larger Set of Frequencies with Manual Table Switch

In this example, shown in Figure 111, we demonstrate how to load a larger set of frequencies on the fly, then use a DGPIO pin to switch to the new table. This is similar to the first example, however it requires an additional signal from the user, namely Hop Select signal to switch to the new table. User will configure the table mode to automatic increment or index by pin.

User will load a larger frequency hopping table, which will be used in multiple hop frames. Once table loading into memory is complete, user then sets the Hop Select DGPIO pin, allowing ADRV9001 to start reading from the second table at the next Hop edge.

- User should ensure the Hop Select Pin is set prior to the appropriate channel setup rising edge.
- User can use the Hop Select table pin to force the switch to the second table at any time. ADRV9001 does not require the switch to happen after completion on the first table.
- This example is also applicable to automatic ping pong mode, as long as user ensures the second hop table is loaded in prior to the completion of the first table.

Frequency Hopping Table Timing

Time used to load a frequency hopping table in ADRV9001 memory and receive the acknowledgement is still being characterized. This information will be updated in future releases.

FREQUENCY HOPPING CALIBRATIONS

When frequency hopping is enabled, ADRV9001 calibrates over a range of frequencies. Since a frequency hopping table can be loaded after initial calibrations, and a new one can be loaded during operation, ADRV9001 calibrates over 42 discrete regions, from 30MHz to 6GHz, to allow the user to operate with any frequency within this range.

The frequency regions are as follows:

Table 39 Frequency Hopping Calibration - Frequency Regions (30MHz - 6GHz)

Region index	Calibration frequency (MHz)	Range (MHz)
0	150	30 <= f < 250
1	350	250 <= f < 450
2	550	450 <= f < 650
3	750	650 <= f < 850
4	950	850 <= f < 1050
5	1150	1050 <= f < 1250
6	1350	1250 <= f < 1450
7	1550	1450 <= f < 1650
8	1750	1650 <= f < 1850
9	1950	1850 <= f < 2050
10	2150	2050 <= f < 2250
11	2350	2250 <= f < 2450
12	2550	2450 <= f < 2650
13	2750	2650 <= f < 2850
14	2950	2850 <= f < 3050
15	3150	3050 <= f < 3250
16	3350	3250 <= f < 3400

17	3450	3400 <= f < 3500
18	3550	3500 <= f < 3600
19	3650	3600 <= f < 3700
20	3750	3700 <= f < 3800
21	3850	3800 <= f < 3900
22	3950	3900 <= f < 4000
23	4050	4000 <= f < 4100
24	4150	4100 <= f < 4200
25	4250	4200 <= f < 4300
26	4350	4300 <= f < 4400
27	4450	4400 <= f < 4500
28	4550	4500 <= f < 4600
29	4650	4600 <= f < 4700
30	4750	4700 <= f < 4800
31	4850	4800 <= f < 4900
32	4950	4900 <= f < 5000
33	5050	5000 <= f < 5100
34	5150	5100 <= f < 5200
35	5250	5200 <= f < 5300
36	5350	5300 <= f < 5400
37	5450	5400 <= f < 5500
38	5550	5500 <= f < 5600
39	5650	5600 <= f < 5700
40	5750	5700 <= f < 5800
41	5850	5800 <= f < 5900
42	5950	5900 <= f < 6000

To reduce the calibration time, the user can reduce the number of regions they calibrate over by setting the minimum and maximum operating frequency in the configuration.

Navassa will store calibration results for the above frequency regions. During operation, when Navassa reads the upcoming hop frequency from the table, it will map it to the appropriate region and apply the algorithm coefficients to HW.

FREQUENCY HOPPING TIMING

In this section, we will show the timing information for different frequency hopping use cases. It is recommended to also read the Timing Parameters Control in this document because many of the timing parameters are explained there.

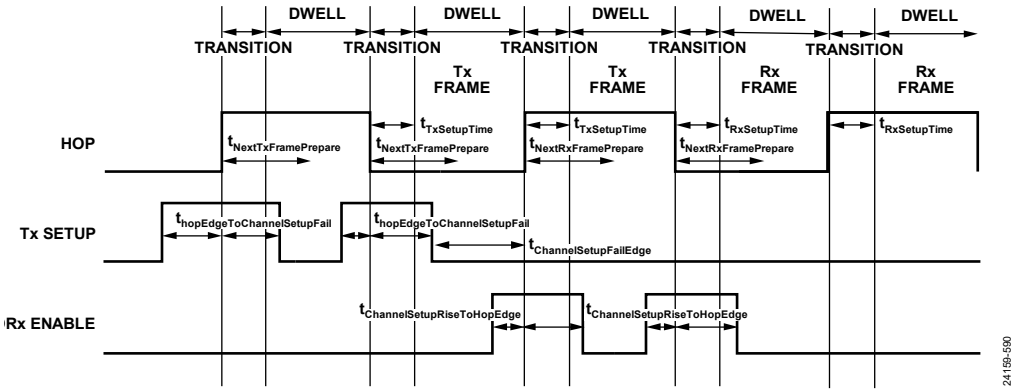


Figure 112. Frequency Hopping Minimum Timing

Table 40. Frequency Hopping Timing Parameters Minimum Timing

Timing Parameter	Description	Time (µs @ 184.32 MHz)
tNextTxFramePrepare	Time required by internal controller to prepare the next Tx frame.	Min: 13 µs
tNextRxFramePrepare	Time required by internal controller to prepare the next Rx frame.	Min: 13 µs
tTxSetupTime (1)	Time taken from hop edge to Tx power up if no LO retune is required and ttxRiseToAnaOn is zero.	Min: 6.7 µs
tTxSetupTime (2)	Time for a consecutive Tx frame. Performs Tx attenuation ramp and LO muxing. The Tx analog is not fully powered up or down and digital remains on.	Min: 4 µs
tRxSetupTime (1)	Time taken from hop edge to Rx power up if no LO retune is required and trxRiseToAnaOn is zero.	Min: 6.7 µs
tRxSetupTime (1)	Time taken for a consecutive Rx frame.	Min: 2 µs
thopEdgeToChannelSetupFall	Minimum time between the hop edge and when the channel setup can go low. This restriction only applies for Tx.	0.42 µs
tChannelSetupRiseToHopEdge	Minimum time required between the channel setup rising edge and the hop edge.	10 µs (at least >3 µs – this time will be improved in future releases)
tChannelSetupFallToHopEdge	Minimum time required between the channel setup falling edge and the hop edge. This restriction only applies to Tx.	0.42 µs

Tx Timing

For LO Muxing mode, Tx setup falling edge starts the Tx interface. And for LO retune mode, Tx setup rising edge starts the Tx interface.

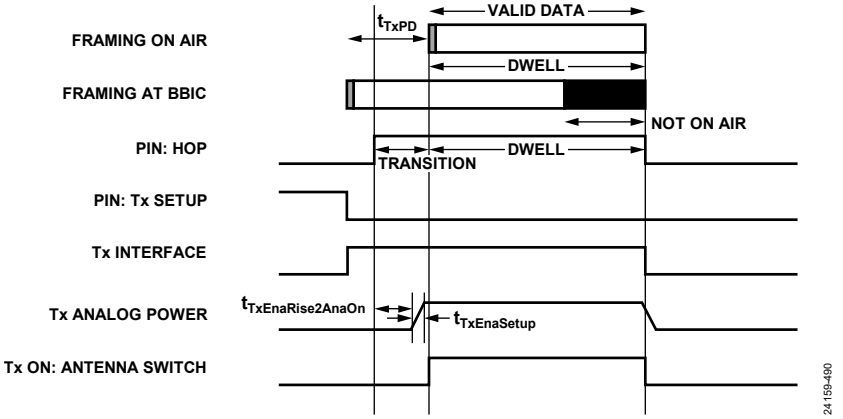


Figure 113. Frequency Hopping Typical Tx Timing for LO Muxing

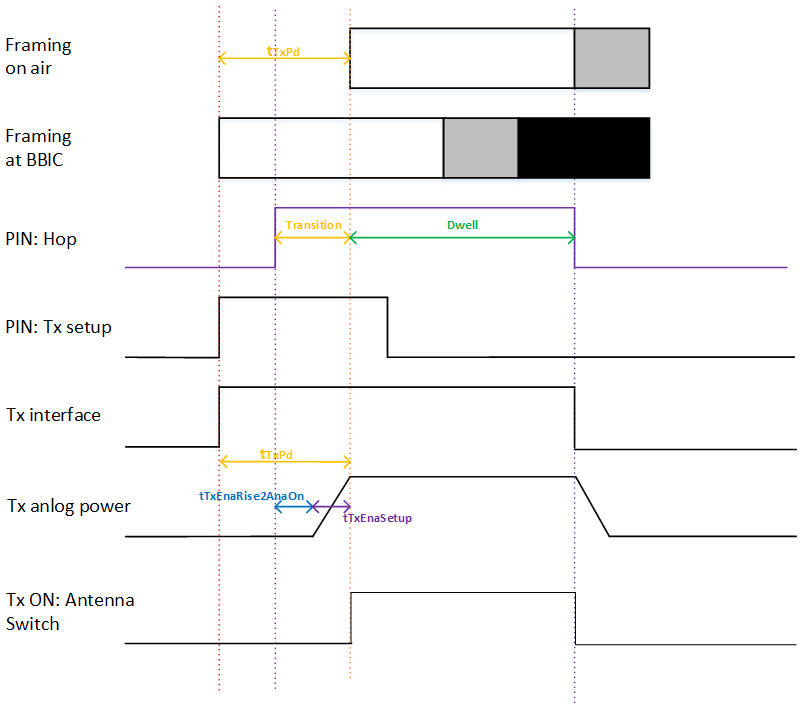


Figure 114 Frequency Hopping Typical Tx Tioming for LO Retune

Table 41 Tx Timing Parameters

Delay Parameter	Descriptions	Bounds	Notes
enableSetupDelay (ttxEnaSetup)	Time taken for ADRV9001 to power up the analog front end. This time may nor may not include PLL retuning time.	N/A	No PLL retuning @ hop edge: 5 µs PLL retuning @ frame boundary: LO_retune_time
propagationDelay	Delay from ADRV9001 digital interface to antenna.	N/A	This time is not used for any delays in ADRV9001. The user should enable the interface and begin data transmission enough time prior to the end of the transition time to account for the propagation delay.
enableRiseToOnDelay	Delay between hop edge and antenna switching to Tx channel.	Min: 0 Max: Transition time	If ADRV9001 is not controlling the antenna switch, this parameter is not needed except to determine other parameters. Typically, the enableRiseToAnalogOnDelay + enableSetupDelay will be the transition time, or transition time + guard time.
enableRiseToAnalogOnDelay (tTxEnaRiseToAnaOn)	Delays the power up of the AFE relative to the hop edge.	Min: 0 Max: enableRiseToOnDelay - enableSetupDelay	This parameter can be used to delay the Tx analog power up if the transition time is greater than the analog power up time. This delay can also be used to delay the analog power up depending on propagation delay requirements.
enableGuardDelay	Not used in frequency hopping. Delay from hop edge to first valid data arriving over interface.	NA	Not used
enableHoldDelay	Not used in frequency hopping. Delay from hop edge and Tx interface being disabled.	NA	Hop edge indicates the end of the frame on air. The datapath and interface are not kept on beyond this point.

Delay Parameter	Descriptions	Bounds	Notes
enableFallToOffDelay	Delay between the hop edge and powering down the analog and Tx antenna switch. Not used in frequency hopping.	NA	Hop edge indicates the end of frame on air. The analog and Tx antenna switch can be powered down immediately.

Rx Timing

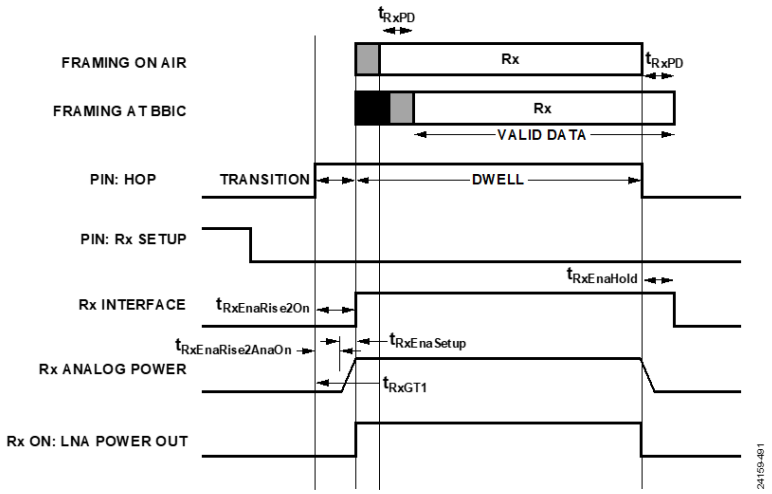


Figure 115. Rx Timing

Table 42. Rx Timing Parameters

Delay Parameter	Descriptions	Bounds	Notes
enableSetupDelay (trxEnaSetup)	Time taken for ADRV9001 to power up the AFE. This time may or may not include PLL retuning time.	N/A	No PLL retuning @ hop edge: 7 µs PLL retuning @ frame boundary: LO_retune_time
propagationDelay	Delay from antenna to Rx interface.	N/A	This parameter will be dynamic profile dependent and board layout dependent. Not necessary to configure ADRV9001, but may be necessary to derive other timing parameters
enableRiseToOnDelay	Delay between hop edge and the LNA power up. If ADRV9001 is not controlling LNA powerup, this variable is not needed.	Min: enableRiseToOnDelay + enableSetupDelay Max: -	
analogGuardTime	Minimum time between the hop edge and analog power up to prevent Rx and Tx FE being powered up at the same time.	N/A	Min/max: 0.15 µs
enableRiseToAnalogOnDelay (tRxEnaRiseToAnaOn)	Delays the power up of the AFE relative to the hop edge.	Min: analogGuardTime Max: -	The minimum time of this delay is the analogGuardTime.
enableGuardDelay (tRxGT1)	Delay between hop edge and first valid data received over the air.	Min: enableRiseToOnDelay + enableSetupDelay Max: 0	Not used currently but can be used to delay starting of the tracking algorithms until the first valid data is received over the air.

Delay Parameter	Descriptions	Bounds	Notes
enableHoldDelay (tRxEnaHold)	Delay between the hop edge and disabling of the Rx interface.	propagationDelay	At the end of an Rx frame, or a sequence of Rx frames, the interface can be left on, even into the next frame, to allow for propagation delay.
enableFallToOffDelay	Delay between hop edge and powering down the LNA. Not used in frequency hopping.	Min: 0 Max: 0	Hop edge indicates the end of frame on air. The analog and Rx LNA can be powered down immediately.

TRx Timing

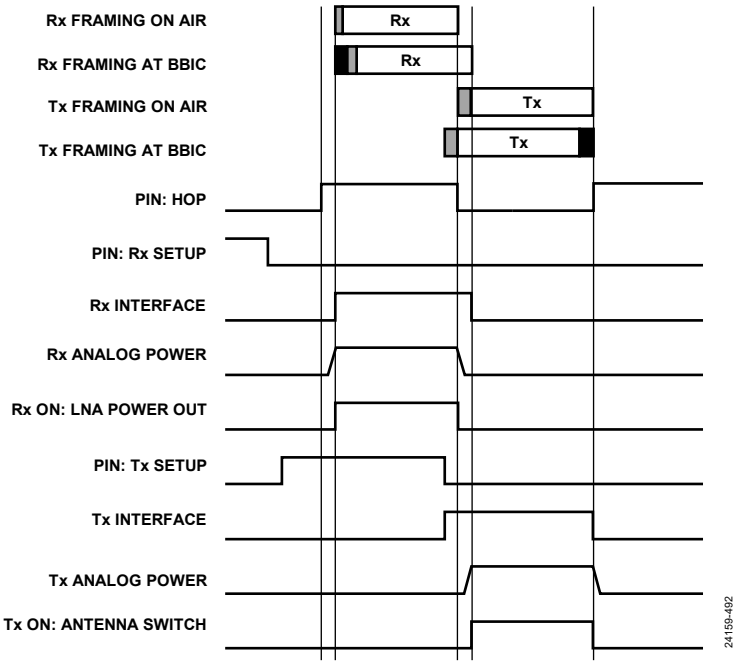


Figure 116. TRx Timing

For TRx operation, because a hop edge can mark both the start and end of an Rx or Tx frame, the ADRV9001 guarantees that the Rx front end and Tx front end are not powered up at the same time.

To achieve this, ADRV9001 enforces a minimum setting for RxRiseToAnaOn, specified by analogGuardTime, to ensure that the Rx front end is powered up after the Tx front end is powered down.

No minimum setting for TxRiseToAnaOn is required. This is because the Rx front end is always powered down before the Tx front end power up routine starts, and no extra delay is required.

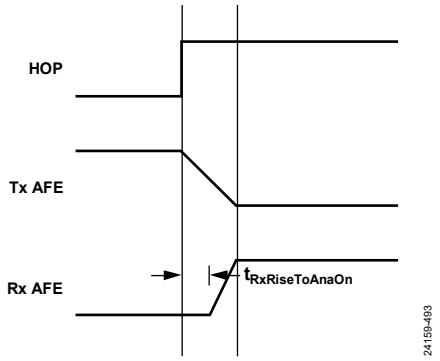


Figure 117. TxRiseToAnaOn

PLL Retune

In PLL retune mode, the time to the start of the Rx/Tx analog front end being powered up is the maximum of $t_{pllRetune}$ and $t_{chRiseToAnaOn}$.

Note for the current release PLL retune time is still being characterized. This information will be updated in future releases.

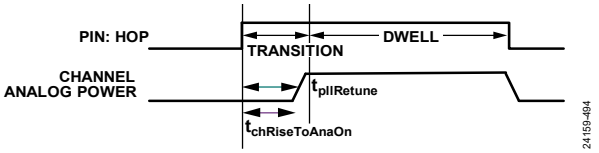


Figure 118. PLL Retune

Table 43. PLL Retune Time (Limited Testing, as PLL Characterization is Still Ongoing)

	PLL Retune Time (µs)	
Device Clock (MHz)	**Fast**	**Normal**
30	91	394
38.4	77	353
50	56	195
100	27	185
150	21	180
200	20	176
245.76	17	172
300	15	165

ADDITIONAL FREQUENCY HOPPING OPERATIONS

Rx Only with Long Propagation Delay

The ADRV9001 supports the scenario when Rx propagation delay is greater than the duration of a hop frame. This however is only supported with Rx only modes. Tx only mode is described in the next section.

To achieve this, set the timing parameter, enableHoldDelay, to the propagation delay to keep the Rx datapath and interface alive after the frame, or series of frames, on air has ended.

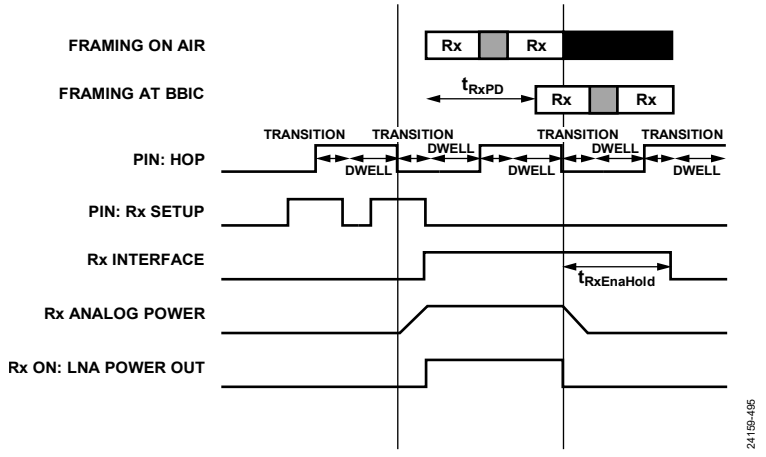


Figure 119. Rx Only with Long Propagation Delay

Tx Only with Long Propagation Delay

The ADRV9001 also supports Tx only case where the propagation delay is greater than the duration of a hop frame.

To achieve this, the user can specify a parameter as a part of the frequency hopping configuration, to delay the powering up of the Tx delay the powering up of the Tx analog, in terms of hop frames. The user can then time the enabling of the interface, along with the using the txRiseToAnalogOn timing parameters, to fine tune the delay.

The delay parameter specifies the delay in terms of hop frames after the first Tx setup rising edge has been received. By design, the ADRV9001 enforces a minimum delay for both Tx and Rx of 1. However, for Tx only, this delay can be greater than 1. If the user sets it to 0, the ADRV9001 defaults the delay to 1.

After the first Tx setup rising edge, the user must then continuously send a pulse train of Tx setup signals until the hop edge in which the Tx frame begins on air. After this, the user can maintain any number of consecutive Tx frames, as long as the Tx setup is continuously toggled. After the pulse train of Tx setup stops (meaning a hop edge without a preceding Tx setup rising edge), the Tx channel is powered down at the next hop edge. A subsequent Tx setup begins the process again.

This example shows a four-frame delay, with the Tx frame starting on air at the fifth frame after the first Tx setup rising edge.

To account for the transition period between consecutive frames, the user may want to pad their valid data with guard symbols. In the example, this is marked by the grey boxes. If the user desires, they can pad their data to keep the transition and dwell times consistent. This is because the transition required for the first Tx frame is greater than that required for consecutive frames.

The user also has the option to program the riseToAnalogOn timing parameter to 0, indicating the analog powerup can be powered up immediately after the hop edge. Then they can time their data to the antenna based on when they drop the Tx setup rising edge and start transmitting data.

For LO muxing case, Tx setup falling edge marks the beginning of the interface. txAnalogPowerOnFrameDelay starts when hop signal first samples a High of Tx setup signal with value of 3, then decrement to 0 with edge coming hop edge. When reaching 0, it waits another frame before Tx analog is powered on. The value will remain 0 until Tx is sampled with low, and the value will be set back to 4

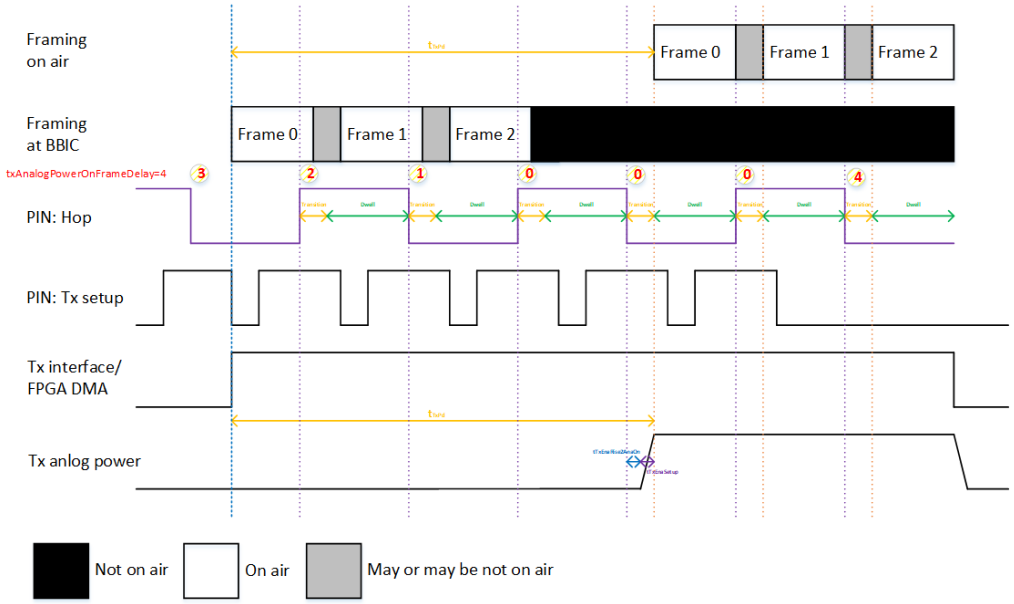


Figure 120. Tx Only with Long Propagation Delay for LO Muxing

For LO retune case, the difference is Tx setup rising edge now marks the beginning of the interface. txAnalogPowerOnFrameDelay starts when hop signal first samples a High of Tx setup signal with value of 3, then decrement to 0 with edge coming hop edge. When reaching 0, Tx analog is powered on. The value will remain 0 until Tx is sampled with low, and the value will be set back to 4.

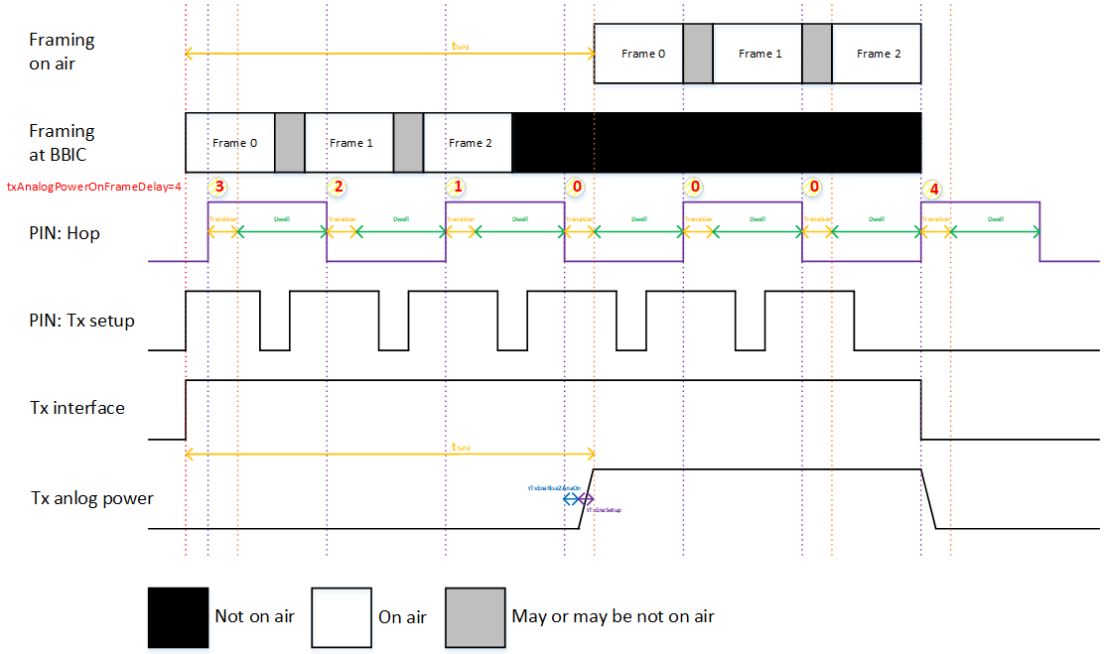


Figure 121 Tx Only with Long Propagation Delay for LO Retune

Tx Only with Short Propagation Delay

There is a restriction to how long before the hop edge the Tx setup falling edge must come. For profiles with very short propagation delays, this means the interface will be on longer than required for valid data to reach the analog front end.

Figure 122, shows one option that user can pad their data with zeros or a known pattern and start transmitting as soon as the interface is enabled. This way although interface is on earlier, the transmitted data will be a known pattern when it reaches analog front end.

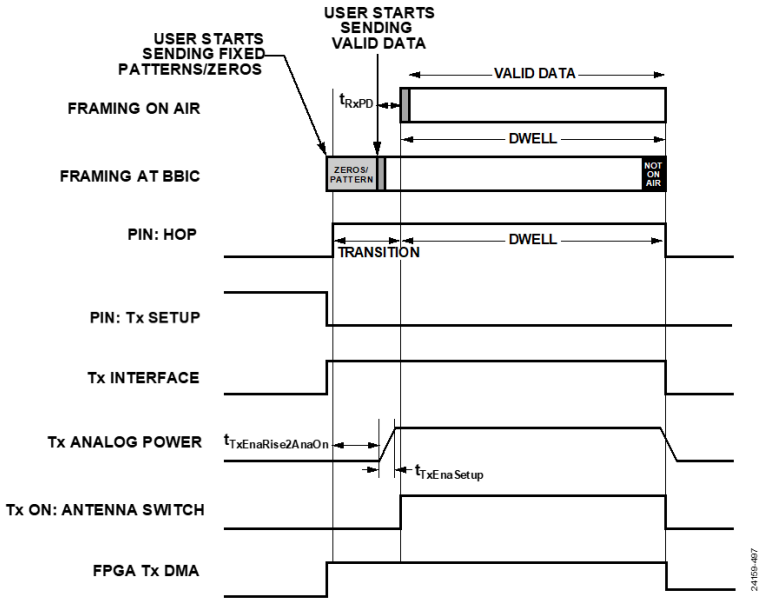


Figure 122. Tx Only with Short Propagation Delay, Padded Data Method

Figure 123, shows another althernative method, where user can hold off from their transmission until $t_{propagationDelay}$ prior to the analog front end being enabled. Here the FPGA Tx DMA is involved to delay the transmitting data until analog front end is ready.

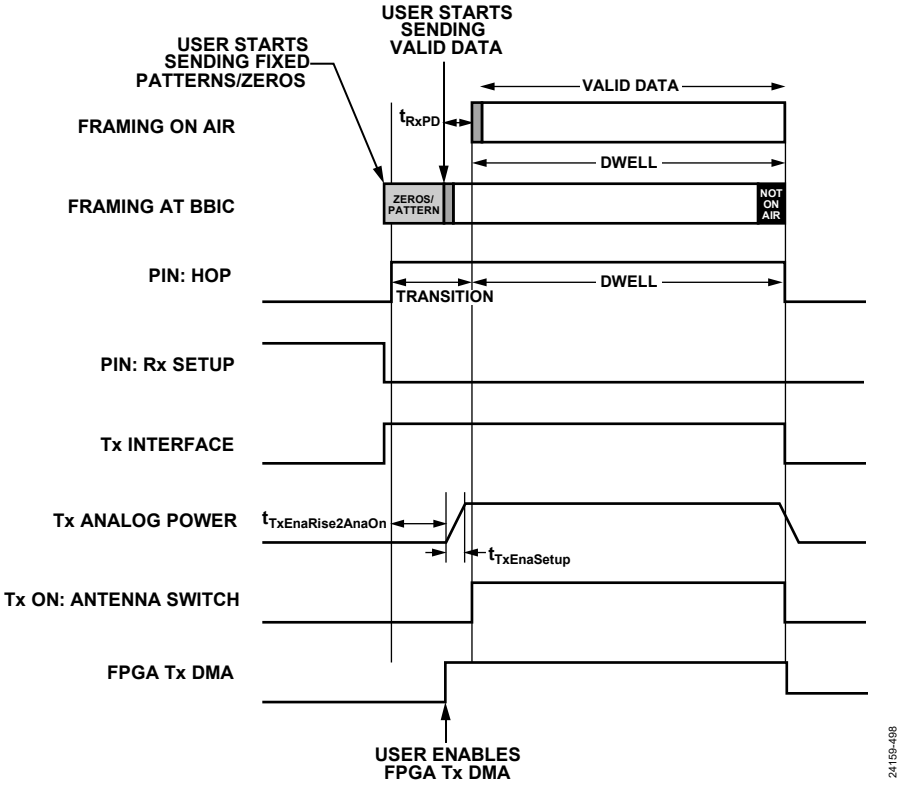


Figure 123. Tx Only with Short Propagation Delay, FPGA Delay

ORx Operation

ADRV9001 supports ORx operation in frequency hopping mode. ORx in frequency hopping works in the same way as normal TDD mode. During the actual Tx frame, the user can set the ORx enable signal high to enable ORx.

The user must then set ORx low (disable ORx) before the end of the frame. This is required regardless of whether the next frame is Tx or Rx.

The user must take the ORx setup and bring-down time into consideration, as well as the propagation delay of their profile.

Table 44. ORx Timing Parameters Time Required

Timing Parameter	Time Required (us)	
	Narrowband	Wideband
tOrxRiseToOn	8	9
tOrxFallToOff	6	5

Table 45. ORx Timing Parameters Timing Restrictions

Timing Parameter	Timing Restriction
tHopEdgeToORxRise	Must be greater than the transition time, or tTxRiseToOn
tORxFallToHopEdge	Enough time for the ORx disable to complete. At least TBD us, however it is recommended to add some guard time to that (for example, 1 µs)

When using ORx in frequency hopping, the user must take into consideration the following parameters when determining their frame duration:

- ORx enable rising edge time (tOrxRiseToOn)
- ORx enable falling edge time (tOrxFallToOff)
- Propagation delay (tpropagationDelay)

The total time the ORx is enabled should be at least (tOrxRiseToOn +tpropagationDelay) otherwise no valid data will be received.

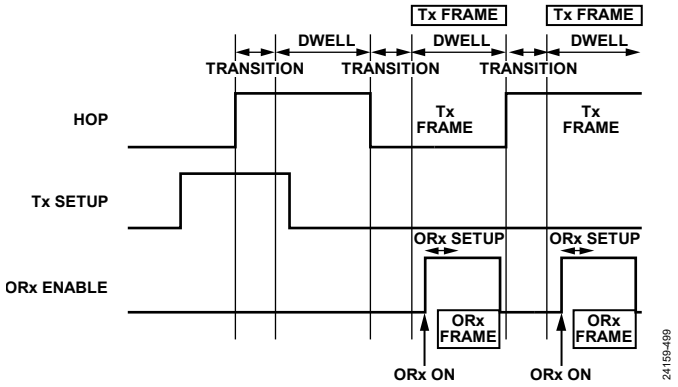


Figure 124. ORx Timing with Tx Setup

DIVERSITY MODE

The ADRV9001 supports diversity mode in frequency hopping, however, there are more timing considerations in which the user must consider.

Primarily, there is increased time required for the ADRV9001 to prepare for an upcoming frame if two channels are enabled. This increased time must be taken into consideration. This time is currently being characterized and more information will be updated in future releases.

To enable two channels for diversity, the operation is like a 1T1R use case. However, the user must control the setup signal for both channels. Prior to the first hop edge, the user must set channel 1 setup rising edge and channel 2 setup rising edge high. There is no restriction on when these signals come relative to each other, however the last one set should be 1us prior to the hop edge (see frequency hopping timing overview).

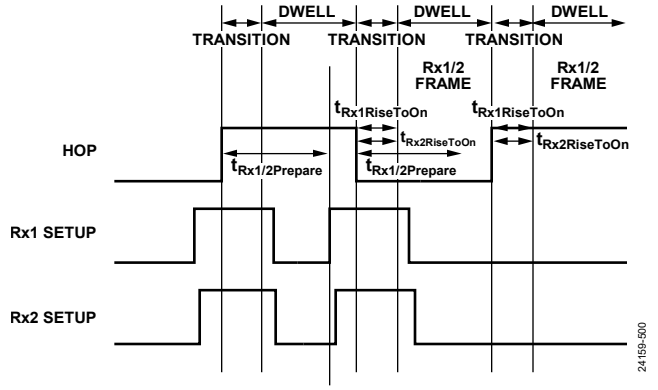


Figure 125. Diversity Timing Example, Rx1/Rx2

FREQUENCY HOPPING WITH RX/ORX GAIN CONTROL

In frequency hopping operation, the user can configure the Rx gain control to either be manual gain control, or AGC.

The user can configure the AGC as in non-FH mode.

When configuring the AGC, the user can either set the AGC to either be reset at the start of each Rx frame to a starting gain index, or to continue from the previous Rx frame.

AGC Configuration

In the ADRV9001, the AGC configuration has the following field,

resetOnRxOn

If the user desires the AGC to continue from the previous Rx frame, they should set *resetOnRxOn* to false. If they want the AGC to reset to a starting gain index each frame, they should set *resetOnRxOn* to true.

In frequency hopping, if the AGC is configured and the *resetOnRxOn* field is true, then the starting Rx gain for each frame will be taken from the frequency hopping table (see *Frequency hopping table* section) **not from the *resetOnRxonGainIndex* field in the AGC configuration.**

Manual Gain Control

In this mode, ADRV9001 will program the manual gain index for each frame based on the gain information in the frequency hopping table (see *Frequency hopping table* section).

ORx Gain Control

The ORx gain operates in manual gain control. In this mode, the user should set their desired starting gain index prior to enabling ORx. They can update the gain throughout the ORx frame.

INTEGRATION WITH OTHER ADVANCED FEATURES

Frequency Hopping with MCS

Frequency hopping with MCS is described in MCS chapter section: Frequency Hopping.

Frequency Hopping with DPD

Frequency Regions

ADRV9001 supports Frequency Hopping to work together with DPD. ADRV9001 divides frequencies into 8 frequency regions. User can specify the start and end frequencies of 7 regions. There is one last region (8th region) to "catch all", which captures the remaining range of the frequencies. For example if user has 6 regions, they can specify 5 regions and the rest will be in the 8th region.

DPD will calculate its coefficients based on the specified regions. This means when transmitting on frequency that is within certain region, DPD will operate only for that region. This includes the capture of the data, calculating the coefficients and lastly update the coefficients in the actuator.

API

To define frequency regions, user need to provide start and end frequencies in adi_adrv9001_DpdFhRegion.

```
typedef struct adi_adrv9001_DpdFhRegions
{
    uint64_t startFrequency_Hz; //!< Carrier frequency greater than or equal to this is
included in the region
    uint64_t endFrequency_Hz;   //!< Carrier frequency less than this is included in the
region
} adi_adrv9001_DpdFhRegions_t
```

User then needs to configure DPD frequency hopping regions in adi_adrv9001_dpd_fh_regions_Configure(), which takes an array of the frequency regions.

```
int32_t adi_adrv9001_dpd_fh_regions_Configure(adi_adrv9001_Device_t *adrv9001,
                                              adi_common_ChannelNumber_e channel,
                                              adi_adrv9001_DpdFhRegions_t dpdFhRegions[],
                                              uint32_t size);
```

TRANSMITTER SIGNAL CHAIN

The ADRV9001 device integrates dual direct-conversion (Zero-IF) transmitters. It supports both time division duplexing (TDD) and frequency division duplexing (FDD) modes and is capable of transmitting both narrowband (NB) and wideband (WB) signals. It supports a wide range of applications such as DMR, P25 and TETRA as examples of NB standards and LTE as an example of WB standards.

In general, each transmitter consists of an independent I and Q signal path with separate digital filters, DACs, analog transmit low pass filters (LPF) and up-conversion mixers. After mixers, an analog attenuator is employed to control the transmitter output signal power.

Data from baseband processor is input to the transmitter signal path via synchronous serial interface (SSI). The serial data is converted to parallel format through the deframer and then the data is processed by interpolation filters. There are several signal conditioning functions, such as transmitter gain control, power amplifier protection, power amplifier, digital pre-distortion (DPD), transmitter quadrature error correction (QEC) and transmitter LO leakage (LOL) handling, before the data is passed on to the DACs. The DAC outputs are filtered by LPF, upconverted to RF via the mixer and attenuated through the analog attenuator to prepare for RF transmission. The ADRV9001 device also supports FM/FSK modulation for some NB applications which will be discussed later.

The high level block diagram of the transmitter signal path is shown in Figure 126:

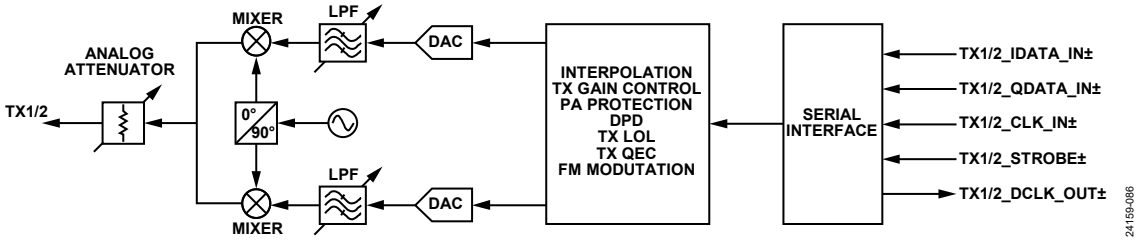


Figure 126. High Level Block Diagram of TX Signal Chain

DATA INTERFACE

The transmitter data interface supports several different interface rates and configurations. It has a total of 5 differential pairs (that is a total of 10 wires). The interface is operated single-ended in CMOS synchronous serial interface (CSSI) mode and differential in LVDS synchronous serial interface (LSSI) mode.

The CSSI interface has one or four data wires as well as one strobe, input and output clock wire. Depending on the number of data wires, the data interface is referred to as CSSI one-lane or CSSI four-lane, respectively. The output clock is passed to the baseband processor to generate the data, strobe and clock signals.

In one-lane operation, the I- and Q-symbols are interleaved and sequentially transmitted over the CSSI interface. Each symbol can consist of 2, 8, 16 bits or 32 bits. The four-lane interface only supports 16-bit symbols. They are separated into two 8-bit words for each I and Q stream, and then sent over the 4 data wires of the CSSI four-lane interface.

For the LSSI interface, there is a separate data lane for the I- and Q-data. There is also a mode where the I-data and Q-data can be interleaved and transferred over a single data lane. The bit-width of the data symbols can be 12, 16 or 32 bits. In addition to the data lanes, the interface has a strobe, an input clock and an optional output clock signal. The output clock signal can be used by the baseband processor to generate the above data, clock and strobe signals.

As mentioned previously, the ADRV9001 supports many NB and WB standards. Depending on the selected standard and the specific symbol rate chosen via the API profile, the interface clock rate can vary significantly. Please note that the CSSI interface is a slow-speed interface and is not able to cover this entire frequency range. Please refer to Data Interface section in this User Guide for more information.

DATAPATH

The high level datapath is shown in Figure 127, which is composed of an analog front end (AFE) and a digital front end (DFE).

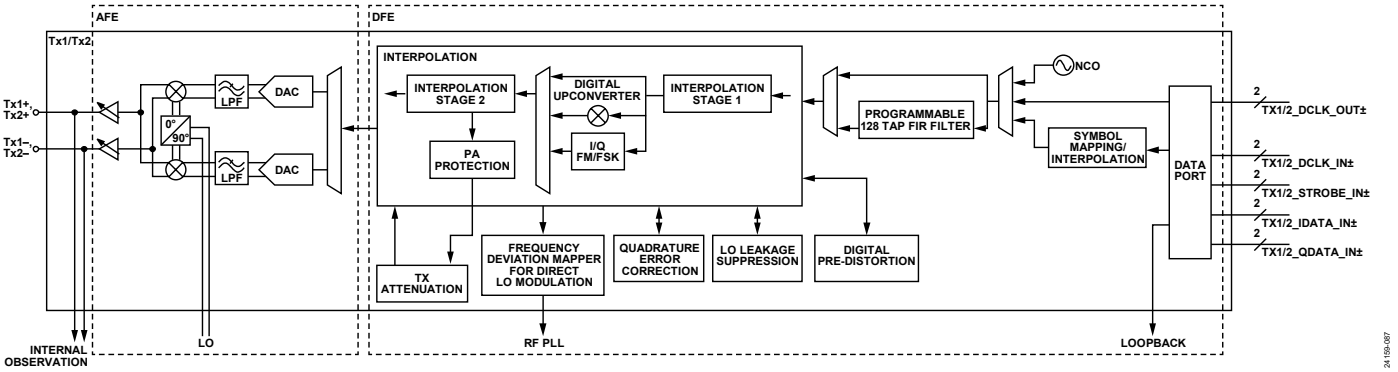


Figure 127. Tx Datapath Block Diagram

In the DFE subsystem, the SSI interface passes data to the transmitter preprocessing blocks, including a symbol mapping/interpolation and a 128-tap programmable FIR filter (PFIR). The symbol mapping/interpolation is used to perform interpolation and/or symbol mapping necessary for certain NB standards. Please note that if it is the configured as an interpolator, proper filtering of the interpolation images needs to be ensured in the PFIR. The PFIR is followed by 2 interpolation stages through a flexible combination of interpolation filters. The interpolation ratios and filters are controlled by the profiles. By design, the interpolation images are rejected by more than 110dB. Between the 2 interpolation stages, there is an optional FM/FSK modulator named IQ FM/FSK and a digital upconverter (DUC) which could both be bypassed. Finally, for IQ data, the signal is interpolated to the DAC sample rate.

As shown in Figure 127, the DFE subsystem also includes various signal conditioning algorithms, such as LO leakage (LOL) suppression and quadrature error correction (QEC). Besides those, it also provides power amplifier protection and transmitter attenuation control blocks.

The output of DFE will first go through the DAC in the AFE subsystem. The DAC standard clock rate can be programmed to be 184.32, 368.64, or 552.96 MHz, which is set by the profile. (Note other sample rates could also be supported when arbitrary sample rate is employed.) Then, the DAC output is filtered by the LPF and input to the up-conversion mixer.

As shown in Figure 127, the ADRV9001 device also supports another method of FM/FSK modulation named Direct FM/FSK modulation. In this mode, the DUC, IQ FM/FSK, the interpolation stage 2, power amplifier protection and transmitter attenuation blocks (digital part) are all bypassed. RFPLL is used to generate a constant envelope phase-modulated signal by modulating the Sigma-Delta Modulator (SDM) with the data stream. In Direct FM/FSK, both DAC and LPF can be powered down.

In the following subsections, major transmitter functionalities will be discussed.

DIGITAL FRONT END (DFE)

Programmable FIR Filter (PFIR)

The PFIR has 128 taps with 24-bit coefficients. There are 2 FIR filters, which are PFIR_I and PFIR_Q as shown in Figure 128. It can be configured to operate in parallel, one for I data and one for Q data in digital IQ modulation applications such as LTE. It can also be configured to use PFIR_I or PFIR_Q only or to operate both filters sequentially for FM/FSK applications. User could optionally use this PFIR for their applications by loading a set of PFIR coefficients.

Before this PFIR, use can also optionally perform interpolation and the interpolation factor supported in the current release is 1 and 2. In such a case, the PFIR can be designed to perform filtering after interpolation. The achieve unity gain, the PFIR coefficients scaling factor can be calculated as 2^23*Interpolation_factor/(Sum of the filter coefficients). If fewer than 128 coefficients are specified by the user, zeros will be appended.

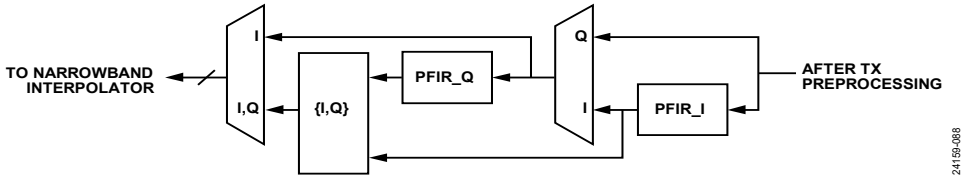


Figure 128. Programmable FIR Filters

Transmit Attenuation Control

Transmit Gain Table

The transmitter attenuation block controls the transmitter output power. A transmitter gain table with 960 entries is loaded into the ADRV9001's memory during initialization. (Currently, only the first 840 entries are used and the remaining 120 entries are reserved for future use.) Each entry equals a 0.05 dB gain step. Therefore, there is a total gain range of 42 dB. The total Tx attenuation is distributed into two portions, an analog attenuation portion and a digital attenuation portion. In the analog attenuation, there is a digitally controlled step attenuator (DSA) with 64 unit steps on a linear scale. The gaps between the analog gain steps are interpolated by a 12-bit digital multiplier to 0.05dB steps. The maximum analog attenuation is 36 dB and the maximum digital attenuation is 6 dB. Note in direct FM/FSK mode, the maximum transmitter attenuation is 12 dB with 0.5 dB step size.

Table 46 shows the first 5 rows of the transmitter gain table.

Table 46. Sample Rows from the Tx Gain Table

Tx Attenuation Index	Total Tx Attenuation (dB)	Analog Attenuation Control Word[5:0]	Analog Attenuation (dB)	Digital Attenuation (dB)	Digital Attenuation (Linear)	Digital Attenuation Control Word[11:0]
0	0	0	0.00	0.00	1.00	4095
1	0.05	0	0.00	0.05	0.9943	4072
2	0.10	0	0.00	0.10	0.9886	4049
3	0.15	1	0.14	0.01	1.00	4090
4	0.20	1	0.14	0.06	0.9928	4066
…	…	…	…	…	…	…

As shown in Table 46, the 1st column is the transmitter attenuation index. The 2nd column shows the total transmitter attenuation in dB for the corresponding index. Note the attenuation step size for adjacent index is 0.05dB. The 3rd column is the control word used to calculate the analog gain shown in the 4th column. The equation used for this calculation is:

$$Analog\ Gain\ (\text{dB}) = 20 \times \log_{10}(1 - Analog\ Attenuation\ Control\ Word/64)$$

The 5th and 6th column show the required digital attenuation in dB and linear domain respectively to achieve the total attenuation in the 2nd column. The last column stands for the digital attenuation control word, which is used to calculate the linear digital gain in the 6th column based on the following algorithm:

If Digital Attenuation Control Word = 4095

$$Digital\ Attenuation = 4096/2\text{^}12$$

else

$$Digital\ Attenuation = Digital\ Attenuation\ Control\ Word/2^{12}$$

Note only the 3rd column and the 7th column are actually stored in memory. Other columns shown in Table 46 are only for explanation purpose.

The user is allowed to read and write the table through the following API command, "adi_adrv9001_Tx_AttenuationTable_Read()" and "adi_adrv9001_Tx_AttenuationTable_Write()". Please refer to the doxygen document for more details about API functions.

Note the table content is defined by the data structure "adi_adrv9001_TxAttenTableRow_t" as the following:

```
typedef struct adi_adrv9001_TxAttenTableRow
{
    uint16_t txAttenMult;
    uint8_t  txAttenHp;
    uint8_t  Reserve;
} adi_adrv9001_TxAttenTableRow_t
```

where "txAttenMult" denotes the "Digital Attenuation Control Word" (in the range of 0 to 4095) and the "txAttenHp" denotes the "Analog Attenuation Control Word" (in the range of 0 to 63) as shown in Table 46.

Transmitter Attenuation Mode

There are 3 modes to control the transmitter attenuation block, which are bypass mode, SPI mode, and GPIO mode. The API command "adi_adrv9001_Tx_AttenuationMode_Set()" is provided to the user to set the transmitter attenuation mode.

Besides that, another API "adi_adrv9001_Tx_Attenuation_Configure()" is provided to the user to set more configurations for transmitter attenuation block, such as the transmitter attenuation step size.

Three transmitter attenuation modes are provided as defined by the enum "adi_adrv9001_TxAttenuationControlMode_e":

```
typedef enum adi_adrv9001_TxAttenuationControlMode
{
    ADI_ADRV9001_TX_ATTENUATION_CONTROL_MODE_BYPASS  = 0,
    ADI_ADRV9001_TX_ATTENUATION_CONTROL_MODE_SPI     = 1,
    ADI_ADRV9001_TX_ATTENUATION_CONTROL_MODE_PIN     = 3,
} adi_adrv9001_TxAttenuationControlMode_e
```

BYPASS MODE

Bypass mode is selected when the transmitter attenuation mode is set as "ADI_ADRV9001_TX_ATTENUATION_CONTROL_MODE_BYPASS". In this mode, the transmitter attenuation functionality is not used, which means 0dB total transmitter attenuation.

SPI MODE

SPI mode is selected when the transmitter attenuation mode is set as "ADI_ADRV9001_TX_ATTENUATION_CONTROL_MODE_SPI". In this mode, the user could set the transmitter attenuation value via the API command "adi_adrv9001_Tx_Attenuation_Set()".

SPI mode consists of two options, the TDD ramp mode and the constant-step size mode. The TDD ramp mode was designed for power ramping in TDD systems. Note it is not supported in the current release. The constant-step size mode allows to control an exact constant gain step size to reach the targeted attenuation level.

TDD Ramp Mode

The TDD ramp mode was designed for use in TDD systems. The user could program an "On power" and "Off power" for the next time slot. The ramp rate can be controlled via a step size (tdd_ramp_step_size) and wait duration (tdd_ramp_wait_duration) between steps. The ramp up or down could be initiated through API commands. Note those user interactions are currently not supported, but will be provided in the future. A typical TDD ramp is depicted in Figure 129.

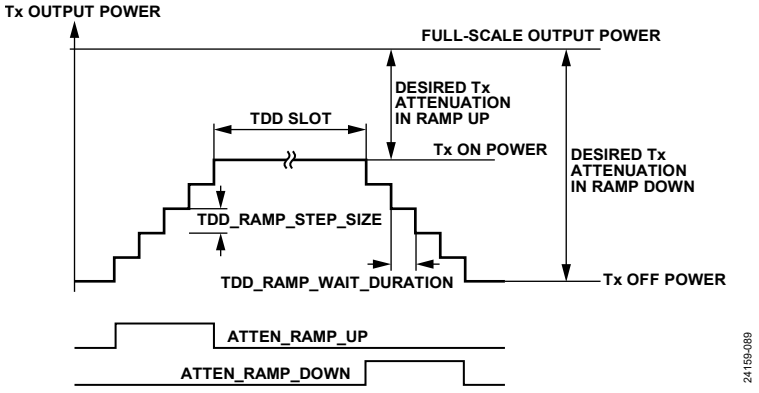


Figure 129. TDD Ramp Profile

Constant-Step Size Mode

In constant-step size mode, the transmitter attenuation controller ramps to the new transmitter attenuation value immediately after it is set through API command. Again the slope can be controlled via a step size (const_step_mode_step_size) and wait duration (const_step_mode_wait_duration) between steps. A typical output power transient for this mode is shown in Figure 130.

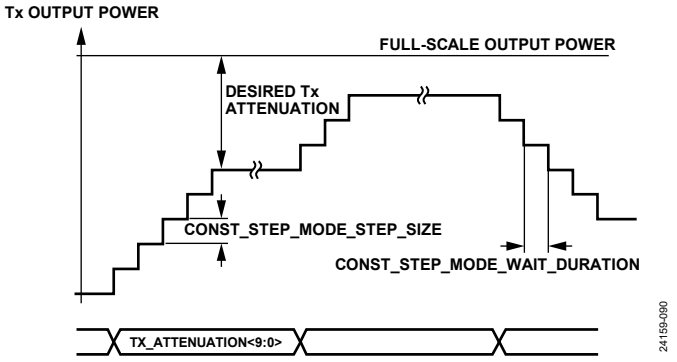


Figure 130. Constant Step Mode

GPIO MODE

Another method to control the transmitter attenuation block is through GPIO mode. In this mode, two GPIO pins are used to increment or decrement the current attenuation value. An API command "adi_adrv9001_Tx_Attenuation_PinControl_Configure()" is provided to the user to configure the GPIO pins and set the step size. A typical output power transient is shown in Figure 131.

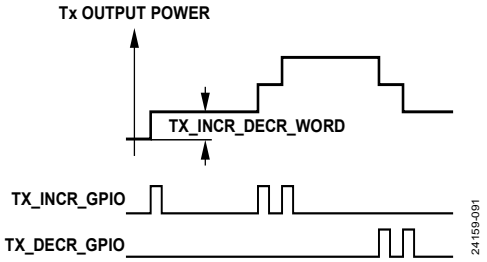


Figure 131. GPIO Incr/Decr Mode

Power Amplifier Protection

In the transmitter signal chain, two power amplifier protection mechanisms are provided to protect the power amplifier from excessive peak or average power excursions. Note these features are not fully supported in the current release.

Power Monitor

Power monitor is one of the power amplifier protection methods, and it uses the transmitter attenuation block to adjust the power by continuously monitoring the output power of the Tx datapath.

Through API commands, the user can enable power amplifier protection and set configuration parameters such as average and peak power thresholds. The average power is accumulated over a specified integration time, and an error flag is asserted if it exceeds the threshold. In addition, the instantaneous or peak power is detected and the error flag is asserted if a specified number of peaks is exceeded. The power amplifier error flag can be read via an API command. The power delivered to the power amplifier is automatically reduced if the error flag is asserted. In the scenario depicted in Figure 132, the error flag is asserted after two power peaks were detected. The power amplifier power is automatically ramped down to max attenuation. Note that the average power did also exceed its threshold, but not for long enough.

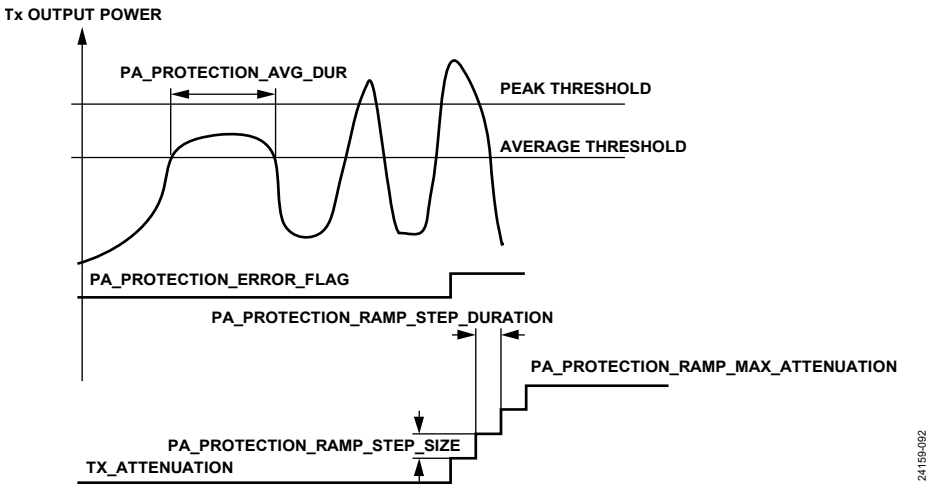


Figure 132. Power Amplifier Monitor

Slew Rate Limiter (SRL)

The slew rate limiter is the other method for power amplifier protection. It essentially limits the rate of change of a waveform by continuously monitoring the difference between the input and output of the block and limiting the amount the output that can change during one cycle. As a result, sudden changes in the input will be applied to the output over several cycles or symbols. Through API commands, the user can control this slew limit as a fraction of full-scale which can be varied from very strong slew limiting to no slew limiting at all. For example, if the slew limit is set to 10% of full-scale, a full-scale step input to the step limiter results in a ramp which spreads over the next 10 clock cycles. The basic block diagram of the implemented slew rate limiter is shown in Figure 133. As shown in this figure, the output sample is looped back to subtract from the input sample to decide the slew rate. Based on the slew limit selection, a proper scaling factor is applied to reduce the slew rate to the desired level.

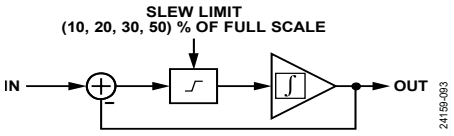


Figure 133. Basic Block Diagram of the Slew Rate Limiter

Transmitter QEC

In the analog circuitry of a direct-conversion transmitter, there are 3 major non-idealities which are gain variation between I and Q datapath, phase imbalance (non-90 degree between LO driving I and Q mixers) and differences in the LPF such as group-delay variations. Without properly correcting them, the output spectrum of the transmitter could be significantly degraded due to the undesired images.

Transmitter QEC is designed to estimate and correct those non-idealities through initial and tracking calibrations. The initial calibration is performed by generating a tone through the NCO and inserting it to the transmitter datapath. Note this tone is visible at the transmitter output therefore user must ensure that antenna is isolated from the transmitter (power amplifier is off) during transmit initial calibration. Internally in the device, the output from the transmit upconverter is looped to the observation receiver (ORx) through internal loopback (ILB) path. The ORx output and the transmitted tone are used to estimate the mismatches. Tables are generated to record the initial calibration results, which could be further refined through tracking calibrations on-the-fly. During signal transmission, the mismatch estimations are applied in the transmitter datapath so that the non-idealities could be compensated. For more detailed information, refer to the Transmitter/Receiver/Observation Receiver Signal Chain Calibrations section.

Transmitter LOL

In the transmitter, any coupling of the LO to the RF output or baseband DC offset could generate an undesired tone at the LO frequency. Without properly correcting it, it could cause a negative impact on the system performance.

Transmitter LOL is designed in the transmit signal chain to handle this problem. Similarly, it estimates the DC offset through initial and tracking calibrations and then apply the estimation in the baseband to cancel the undesired tone. For more detailed information, refer to the Transmitter/Receiver/Observation Receiver Signal Chain Calibrations section.

DPD

DPD is an optional feature available in the ADRV9001 device to enable users to achieve higher power amplifier efficiency while still meet Error Vector Magnitude (EVM) and adjacent channel leakage ratio (ACLR) requirements in their transmitter signal chain for compliance

with the standards requirements. DPD works on the principle of pre-distorting the transmitter data to cancel distortion caused by power amplifier compression. It uses the tracking calibration to capture the transmitted data samples and the data samples looped back through ORx to estimate the distortion parameters. By applying the estimations in the real time, the transmitted signal is pre-distorted to compensate for the power amplifier nonlinearity.

For more detailed information, please refer to the Digital Predistortion section in the User Guide.

Transmitter NCO Internal Signal Source

The ADRV9001 has an internal quadrature NCO. It serves 2 major purposes. First, it could be used to generate the calibration tones for the initial calibrations such as the transmitter QEC. Second, users can use this functionality to generate test tones through an API command to disable the data port interface and simplify the design for specific use cases. In both cases, as shown in Figure 86, the transmitter preprocessor takes the input data from NCO instead of data port interface.

Transmitter Frequency Offset Correction

The ADRV9001 provides user capability to correct small deviations in transmit LO frequency through an API command. Through this API, the user can provide the desired frequency offset in Hz and whether to change the frequency immediately or update it at the start of the next available frame.

FM/FSK Modulation

The ADRV9001 provides a FM/FSK modulation for standards which use constant-envelope frequency modulation scheme, such as DMR, Analog FM, P25 Phase 1 and Phase 2. It also provides an optional capability to perform symbol mapping and interpolation operation on the transmit data received from baseband processor for FM/FSK modulation. This capability provides user more flexibility when preparing transmit data for transmission. User has an option to send either pre mapped and interpolated transmit data by enabling this functionality or send post mapped and interpolated data by bypassing this functionality in ADRV9001. For example, for the Digital Mobile Radio (DMR) standard which uses 4.8 ksps symbol data. Baseband processor could send the symbol data directly to ADRV9001 and let ADRV9001 map the symbol data and then interpolate the data to generate frequency deviation data. Note this functionality is currently not enabled in the datapath and will be provided to user in the future.

Currently, to use the FM/FSK modulation capability of ADRV9001, user should perform symbol mapping, interpolating and pulse shaping filtering in baseband processor to generate frequency deviation data before sending to ADRV9001. Two different options of FM/FSK modulation are deployed in the ADRV9001 which are Direct FM/FSK and IQ FM/FSK as shown in Figure 86. They are briefly discussed in the following subsections.

Direct FM/FSK

Frequency modulation is implemented by modulating the transmitter RF PLL directly in Direct FM/FSK option. The transmitter datapath with Direct FM/FSK is shown in Figure 134.

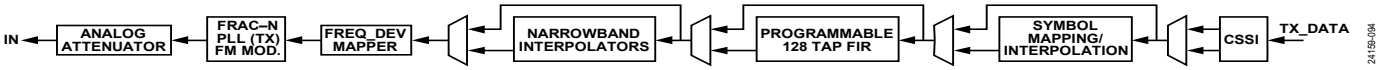


Figure 134. Direct FM/FSK Data Path Block Diagram

As shown in Figure 134, the baseband processor TX_DATA can optionally go through the symbol mapping/interpolation and programmable FIR, and after interpolation and frequency deviation mapping, the Frac-N PLL implements the FM/FSK modulation at the desired RF output frequency. Finally, the PLL output is attenuated before feeding to the transmitter RF interface. The programmable 128-tap FIR works as the pulse shaping filter in this scenario, customer could optionally load their filter coefficients according to the standard requirement through API commands. Note this feature is currently not available but will be provided in future releases. In Direct FM/FSK modulation, the DAC and LPF could both be powered down and some digital blocks such as the common interpolators, power amplifier protection and transmitter attenuation control could all be bypassed. Therefore, it could result in a significant power saving.

IQ FM/FSK

IQ FM/FSK modulation is implemented by modulating the digital NCO as shown in Figure 86. The modulated IQ data goes through interpolator, DAC, LPF, and then be upconverted to RF frequency by mixer. The previous processing stages before the digital FM/FSK modulator are similar to Direct FM/FSK option, which also contains optional symbol mapping/interpolation and pulse shaping functions. The selection between direct FM/FSK and IQ FM/FSK is determined by profile.

ANALOG FRONT END (AFE)

DAC

The ADRV9001 integrates a 16-bit DAC which can be operated at standard rate of 184.32, 368.64, or 552.96 MHz (Note when arbitrary sample rate is supported, DAC can operate at other different rates as well.). The sampling rate is set by the selected profile. The DAC is

auto-tuned to remove mismatches in the DAC units which improves the linearity of the DAC. The nominal full-scale current of the DAC can be boosted by 3dB through API command "adi_adrv9001_Tx_DacFullScaleBoost_Set()". The increased signal swing throughout the entire analog signal chain will result in better AM noise performance. By default, the 3dB boost is disabled.

LPF

The analog LPF is used to attenuate the sampling images of the DAC. The frequency response has 2nd-order Butterworth shape. The corner frequency is auto-tuned to compensate for process and temperature variation. The operating corner frequency is set by the API profile. ADRV9001 allows user to configure LPF at 3 different power consumption levels to help achieve system power saving target.

Up-conversion Mixer

The up-conversion mixer translates the baseband signal to RF. It is an IQ modulator which receives a quadrature baseband and LO signal. Due to the image rejection property of IQ modulators, it produces an output only on one side of the LO, i.e. the image is rejected. The LO leakage and quadrature errors of the mixer are calibrated at startup and continually tracked by transmitter LOL and transmitter QEC as discussed earlier.

RF Attenuator

Following the mixer is a digitally controlled step attenuator with 64 linear gain steps. This results in a total gain range of 42 dB. Please note that the analog gain steps are not linear-in-dB. However, as pointed out in the previous section the analog gain steps are interpolated by a digital multiplier to achieve 0.05 dB gain steps. Note in Direct FM/FSK mode, the total gain is 12 dB with 0.5 dB step size.

TRANSMIT DATA CHAIN API PROGRAMMING

A set of transmitter data chain APIs are provided for user interaction with the ADRV9001 device transmit datapath. Some of them have been discussed in the previous sections. The following table summarizes the list of API functions currently available with a brief description for each one. For more up-to-dated information and detailed descriptions, please refer to API doxygen document. Note more details about transmitter power amplifier ramp functionality can be found in this User Guide in the future.

Table 47. A List of Tx Data Chain APIs

Rx Gain API Function Name	Description
adi_adrv9001_Tx_Attenuation_Configure	Configures the Tx attenuation for the specified channel
adi_adrv9001_Tx_Attenuation_Inspect	Inspects the Tx attenuation for the specified channel
adi_adrv9001_Tx_AttenuationMode_Set	Sets the attenuation control mode
adi_adrv9001_Tx_AttenuationMode_Get	Gets the attenuation control mode
adi_adrv9001_Tx_Attenuation_Set	Sets the Tx attenuation for the specified channel
adi_adrv9001_Tx_Attenuation_Get	Gets the Tx attenuation for the specified channel
adi_adrv9001_Tx_OutputPowerBoost_Set	Enables or disables the Tx output power boost
adi_adrv9001_Tx_OutputPowerBoost_Get	Gets the current Tx output power boost enable status
adi_adrv9001_Tx_AttenuationTable_Write	Writes the attenuation table for the specified Tx channels
adi_adrv9001_Tx_AttenuationTable_Read	Reads the attenuation table for the specified Tx channels
adi_adrv9001_Tx_InternalToneGeneration_Configure	Sets the Tx NCO internal tone frequency for the specified Tx channel
adi_adrv9001_Tx_InternalToneGeneration_Inspect	Gets the Tx NCO internal tone frequency for the specified Tx channel
adi_adrv9001_Tx_PaProtection_Configure	Configures power amplifier Protection for the specified Tx channel
adi_adrv9001_Tx_PaProtection_Inspect	Inspects power amplifier Protection for the specified Tx channel
adi_adrv9001_Tx_SlewRateLimiter_Configure	Configures the slew rate limiter for the specified Tx channel
adi_adrv9001_Tx_SlewRateLimiter_Inspect	Inspects the slew rate limiter for the specified Tx channel
adi_adrv9001_Tx_PaRamp_Configure	Configures the power amplifier ramp for the specified Tx channel
adi_adrv9001_Tx_PaRamp_Inspect	Inspects the power amplifier ramp for the specified Tx channel
adi_adrv9001_Tx_Attenuation_PinControl_Configure	Configures the Tx attenuation PIN mode for the specified Tx channel.
adi_adrv9001_Tx_Attenuation_PinControl_Inspect	Inspects the Tx attenuation PIN mode for the specified Tx channel.
adi_adrv9001_Tx_FrequencyCorrection_Set	Sets the NCO frequency to correct for small deviations in Tx LO frequency.

RECEIVER/OBSERVATION RECEIVER SIGNAL CHAIN

The ADRV9001 offers dual receive channels. With a minimum number of external components, each receive channel could build a complete RF-to-bits signal chain which serves as RF front end for a wide range of applications. It supports both time division duplexing (TDD) and frequency division duplexing (FDD) modes and reception of both narrowband (NB) and wideband (WB) signals up to 40 MHz. NB applications include DMR, P25 and TETRA, while WB applications are geared towards LTE transmissions. For example, ADRV9001 supports standard sample rates of 24 kHz (typically for FM waveforms), 144 kHz and 288 kHz (typically for TETRA signals), and 1.92 MHz, 3.84 MHz, 7.68 MHz, 15.36 MHz, 23.04 MHz, 30.72 MHz, and 61.44 MHz (typically for LTE signals). Besides those standard rates, the ADRV9001 is also capable of supporting an almost continuous range of sample rates between 24 kHz and 61.44 MHz. Some sample rates could not be supported due to internal clocking constraints.

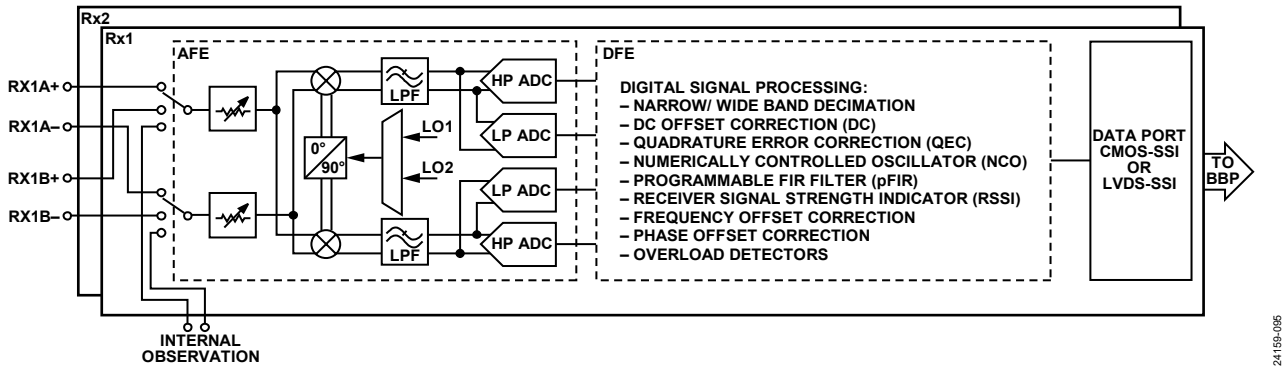


Figure 135. Top Level Structure of ADRV9001 Dual Receiver

Figure 135 describes the top-level structure of the ADRV9001 receivers. As shown in Figure 135, each receive path Rx1 or Rx2 contains 2 major subsystems, the Analog Front End (AFE) and the Digital Front End (DFE). The AFE subsystem contains 4 major components, which are programmable front end attenuator, matched I and Q mixer, low pass filter (LPF) and analog-to-digital converter (ADC). The attenuators are used to control the signal gain to avoid overloading the datapath when a strong interfering signal presents. It is followed by the mixers to down convert the received signals for digitization. The output current of the mixers is further converted to voltage and filtered by LPFs before passing to ADCs. The ADRV9001 provides two pairs of ADCs, a pair of high performance (HP) ADCs to achieve high linearity performance and a pair of low power (LP) ADCs with slightly less linearity performance but significant lower power consumption. This design allows for a flexible trade-off between power consumption and linearity performance.

The DFE subsystem contains a series of digital signal processing components such as sample rate decimation (DEC), dc offset correction (DC), quadrature error correction (QEC), digital down conversion (DDC) with numerically controlled oscillator (NCO), a programmable 128-tap FIR filter (PFIR), receiver signal strength indicator (RSSI), frequency offset correction (FOC), phase offset correction (POC) and overload detectors. DEC is used to decimate the ADC sample rate to the desired output sample rate. DC, QEC, PFIR, FOC and POC are used to condition the digital signals at different stages of the datapath for optimal performance. Overload detectors are used for gain control in the datapath. RSSI provides signal power measurement to control the bit-width of the output signal. In addition, it could be used to detect the presence of a signal in a desired frequency band. At the end of the signal chain, through CMOS-SSI or LVDS-SSI data port, the output signal is delivered to based band processor for further processing.

The ADRV9001 supports a RF local oscillator (LO) range from 30 MHz to 6 GHz. RF LOs can be generated via two internal phase lock loops (PLL) or applied externally to the part. The digital subsystem contains an optional digital mixer that is driven by a programmable NCO. Receiver LO can offset from the frequency of the desired channel and then make use of the digital mixer to down convert the signal to baseband before being processed by baseband processor. There are several advantages to offset the receiver LO from the frequency of the desired channel: Impairments that exist around the receiver LO, such as LO-leakage, can be avoided. The effect of flicker noise from baseband circuits can be mitigated since the received signal is offset from dc in the analog signal path. Also, image rejection can be improved if the receiver LO is offset enough from the desired channel, such that the image frequency lies in the attenuation region of the user's external RF filter. IF operation could work with both NB and WB applications. Typically, when the receiver is operating in NB mode, the sensitivity requirements for these applications demand very low noise performance, therefore, the intermediate frequency (IF) approach is preferred. The device is capable of receiving signals offset from the carrier, as well as an IF down conversion scheme. When the receiver is operating on a WB signal, it could use direct down conversion or zero IF (ZIF) (although IF approach is also available for WB signal). In this mode the DDC will be bypassed.

Figure 127 describes the simplified transmit and receive signal path between the antenna and the ADRV9001 device. The components between the antenna and the ADRV9001 device are external components. In the transmit path, typically, the output signal from the device goes through a variable gain amplifier (VGA), a low-pass filter (LPF) and a power amplifier before transmitting through antenna. In the receive path, typically, the RF signal receiving from antenna goes through a low noise amplifier (LNA) and a band-pass filter (BPF) before sending to the device. The duplexer is for supporting both FDD and TDD modes, which could stand for a frequency duplexer in FDD mode and a RF switch in TDD mode.

As shown in Figure 136, for each receiver, besides acting as a primary data channel for receiving RF signals, it could also serve as an observation channel, which receives loopback signals from transmitter. There are 3 loopback paths, which are internal loopback (ILB), external loopback type 1 (ELB1) and external loopback type 2 (ELB2) as shown in Figure 136. When users are in full control of the loopback channel for running their own algorithms, we rename the receiver as observation receiver (ORx). In such a case, user can use either ELB1 or ELB2 depending on the algorithm requirements. For example, if running an external DPD, user should use the ELB2 path.

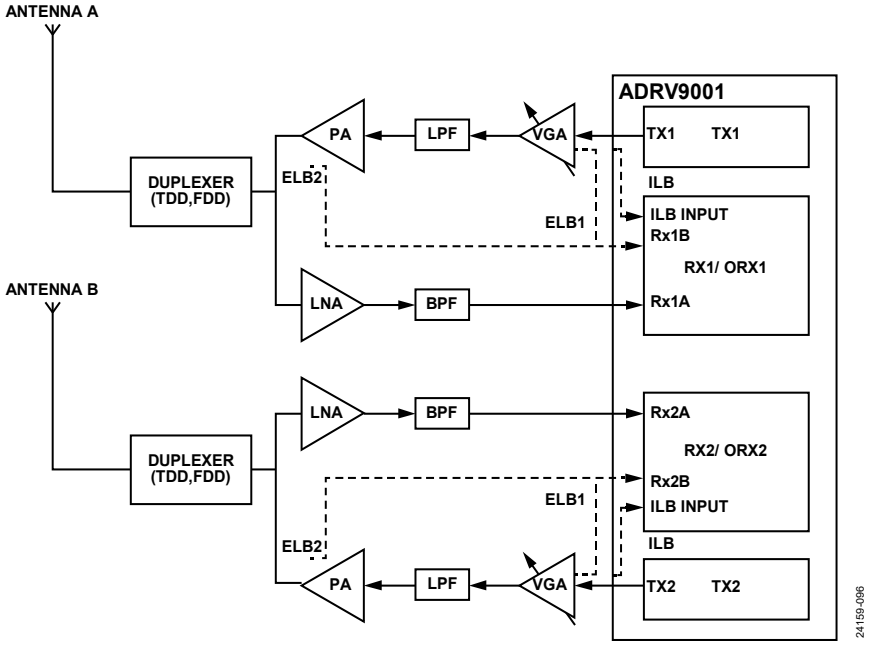


Figure 136. ADRV9001 Rx/ORx/Loopback Diagram

The 3 loopback paths could also be used internally by ADRV9001 for two major purposes: transmitter init calibrations and transmitter tracking calibrations, including the integrated digital pre-distortion (DPD) operation. Transmitter init calibrations is to configure the device properly based on system configurations during the initialization time. It can be done through either ILB, ELB1 or ELB2. The major advantage of using ELB1 comparing with ILB is to observe common mode voltage. In addition, during transmitter init calibrations procedure, test tones are generated and present at transmitter output. Therefore, users must ensure an appropriate level of isolation from ADRV9001 transmitter output to the antenna to ensure that test tones are not transmitted by the system. This isolation could be achieved by disabling power amplifier during transmitter init calibration in ELB1. For ELB2, the calibration signal might be transmitted out through the antenna. Although the power level of calibration signal is set as low as possible, the user should make sure that this will not cause any problem when using this option. See the Transmitter/Receiver/Observation Receiver Signal Chain Calibrations section for more information.

Transmitter tracking calibrations is to tweak the system on the fly during its normal operations for optimal performance. Similarly, it could use ILB, ELB1 or ELB2. ILB and ELB1 are used when DPD is not required, while ELB2 is used when DPD is required, in which the transmit signal is looped back to the receiver after power amplifier. Note ELB1 and ELB2 shares the same receiver input, so they can't be used simultaneously.

No matter used by the user or internally by the device, the observation channel shares the same datapath as the regular receiver, therefore the observation can only be performed when no regular reception is required at the same time. This is the case for transmitter init calibrations and the user should aware that receiver might not be idle since it works as observation channel during the time period of initialization internally by the device. Different from transmitter init calibrations, transmitter tracking calibrations are performed on the fly, so they have to time share with the regular receiver operations. For example, in a TDD system, when transmitter is transmitting receiver should not be receiving, therefore, it could be used for observation for transmitter tracking calibrations. For a system where receiver is fully occupied all the time for receiving RF signals, such as a 2Tx2Rx FDD system, it is not possible to perform transmitter

tracking calibrations including DPD operations in such a system. However, in a 1Tx1Rx FDD system, because one receiver is always idle, it can be used as an observation channel. For example, if Transmitter 1 is transmitting, then Receiver 1 can be used for observation by receiving loop back signals from Transmitter 1 and Receiver 2 can be used as the main receive path. Note it is required that the observation must be at the same side of the transmitter it observes so observation channel 1 is always for Transmitter 1 and observation channel 2 is always for Transmitter 2. When users are in control of the observation channel, they will be allowed to configure the ORx based on their requirements such as the ORx gain. When the ADRV9001 device is control of the observation channel, it is responsible to configure the observation channel properly without any user intervention.

As shown in Figure 136, each receiver has 3 inputs, one is the ILB input dedicated for receiving ILB signal. The others are Rx1A/Rx2A and Rx1B/Rx2B inputs, one could be configured to receive RF signals and the other one to receive ELB signals. ADRV9001 provides user an option to select main receive port (either port A or port B) during initialization. For example, if port B is selected as the main receive port, then port A is used to receive ELB signals. Furthermore, ADRV9001 provides dynamic port switching capability for user to switch receive ports (either port A or port B) on the fly. This is mainly designed to accommodate applications which have limitations on operating frequency range of the RF Balun, in which case, multiple Baluns may be used to cover the whole frequency range (30MHz to 6GHz) supported by ADRV9001. In order to support seamless switch between multiple Baluns that cover different frequency ranges, dynamic port switching feature allows the user to use both port A and B as main data receiver channels while the two ports can be switched at run time based on the carrier frequency range user defined for each port. Note once this feature is enabled, it applies to both receive channels, i.e. switching port A/B on Rx1 but using fixed port on Rx2 is not supported. In addition, ELB cannot be enabled in this case. The frequency range for port A and port B cannot overlap. Due to the support of a wide range of applications, user interaction with the receiver signal chain is mainly done through configuration profiles. Based on the channel profile, which includes key parameters such as bandwidth, sample rate and AGC settings, initial calibration is performed in the device to set up the receive chain properly. When DPD is performed internally in the device, the switch between receiver and ORx is fully determined without user interaction. Therefore, the device could support rapid switching between different RF channel profiles with different modulation schemes and bandwidths requirements. When DPD is performed externally by baseband processor, then baseband processor owns the entire ORx channel. It is the user's responsibility to make sure there is no conflict between the DPD operations and the transmitter tracking calibrations in the device.

In the ADRV9001, a specialized "Monitor mode" exists that allows the device to autonomously poll a region of the spectrum for the presence of a signal, while in a low power state. In this mode, the chip continuously cycles through sleep-detect-sleep states controlled by an internal state machine. Power savings are achieved by ensuring that the sleep duty cycle is greater than the "detect" duty cycle. In the "sleep" state, the chip is in a minimal power consumption configuration where few functions are enabled. After a pre-determined period, the chip enters the "detect" state. In this state, the chip enables a receiver and performs a signal detect over a bandwidth and at a receiver LO frequency determined by the user. If a signal is detected, the "Monitor Mode" state machine exits its cycle and normal signal reception will resume. If no signal is detected, the chip resumes its sleep-detect-sleep cycle. The sleep-detect duty cycle and durations, power measurement threshold, and receiver LO are user-programmable, and are set before enabling "Monitor Mode." Refer to the section in the User Guide for more details.

The ADRV9001 provides users with various levels of power control. Power scaling on individual analog signal path blocks can be performed to trade-off power and performance. In addition, enabling and disabling various blocks in TDD receive and transmit frames to reduce power could be customized, at the expense of receive/transmit or transmit/receive turnaround time. See the Rx Monitor Mode section for more information.

Receive Data Chain, AFE Components, Digital Front End Components, and Receive Data Chain API Programming

The following sections provide topical information regarding:

- Receive data chain: this section describes how the analog and digital components are used at the different stages of the receiver chain to convert RF signals to bits at the desired sample rate for further baseband processor processing.
- Analog front-end components: this section discusses each major AFE component and its functionality.
- Digital front-end components: this section discusses each major DFE component and its functionality.
- Receive data chain API programming: this section outlines the API programming capabilities of receiver data chain for user interactions.

RECEIVE DATA CHAIN

The ADRV9001 supports both NB and WB applications in a common design. Figure 137 describes the block diagram of the entire receiver data chain, which is composed of AFE and DFE. As mentioned previously, the AFE includes a front end attenuator which controls the received RF signal level, mixer for RF to baseband (or IF) down-conversion, low-pass filter and a pair of HP and LP ADCs. The LPF has a programmable bandwidth from about 5 MHz to 50 MHz depending on the profile. Its configuration and filter

characteristics are automatically tuned internally to achieve optimal performance for different applications. In principle, the AFE design is based on WB architecture with a very high dynamic range to absorb both desired signal and interference without distortion. Therefore, in such a design, very little channelization or blocker filtering is needed through LPF since the HP and LP ADC can simultaneously absorb weak signals and large blockers. Blocker suppression and channelization are then achieved efficiently in the digital signal path. After ADC, the digital output signal will be further processed through multiple stages in DFE.

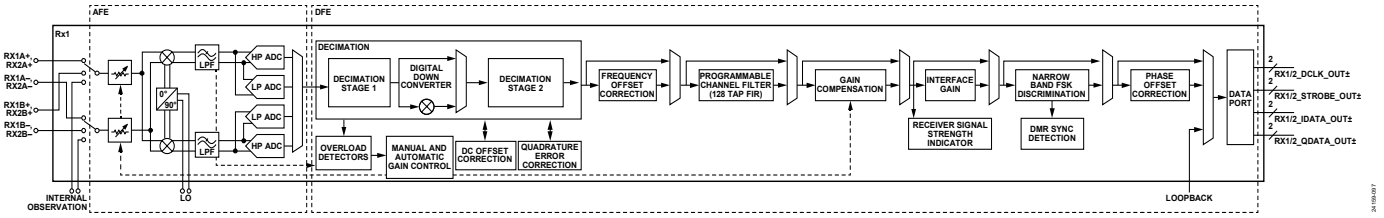


Figure 137. Rx Signal Chain Block Diagram

The ADRV9001 supports 3 standard ADC clocks, which are ADC-H clock 2211.84MHz, ADC-M clock 1474.56MHz and ADC-L clock 1105.92MHz for both HP ADC and LP ADC (Note ADC clock could vary with arbitrary sample rate.). In the DFE subsystem, the ADC output signal at 3 different sample rates will go through 2 decimation stages as shown in Figure 128 to convert to the desired sample rate by using a flexible combination of decimators. Between the 2 decimation stages, there is an optional DDC which is employed in the applications which adopts IF reception scheme.

At different decimation stages, several signal conditioning algorithms are performed, which are overload detection for gain control, DC offset correction (DC) and quadrature error correction (QEC) as shown in Figure 137. The overload detection result is used by automatic gain control (AGC) or manual gain control (MGC) algorithms to properly control both analog and digital gain through a receiver gain table. The analog gain is applied at the front end attenuator to avoid overload/underload situations. The digital gain is applied at the gain compensation block in the receiver datapath and it has 2 major functionalities: one is to correct the small step size inaccuracy of the front end analog gain and the other is to compensate for the front end gain change completely so that it is transparent to users. Different receiver gain tables are loaded for either correction or compensation based on user's configuration during the device initialization. In ADRV9001, a sophisticated gain control mechanism (AGC/MGC) is provided, see the Rx Gain Control section for more details. DC and QEC are used to correct the DC offset and quadrature error so that the signal distortion could be minimized to achieve an optimal performance before sending data to baseband processor. To achieve best performance for different applications, QEC algorithm is designed differently for WB and NB applications.

After decimation stage 2, the ADRV9001 provides an option to correct small carrier frequency offset through API commands, followed by a 128-tap programmable PFIR as a channel selection filter. In the future, API commands will be provided for PFIR for more user interactions.

After PFIR, besides applying the digital gain as discussed earlier, an interface gain could be optionally applied by utilizing the signal strength measurement from RSSI. The interface gain is applied through a "Slicer" by properly shifting the signal. When the signal is large, it could be used to avoid saturate the data port due to a limited bit-width, and when the signal is small, it could be used to avoid lose sensitivity. An API command is provided to the user to read the signal strength measurement. The interface gain can be applied automatically in the device or manually by the user through API commands. This is beneficial when saturation is observed in baseband processor. See the Rx Gain Control section for more details.

The RSSI could also be used as the signal detector in Rx Monitor Mode. In NB applications, at the end of the datapath, the device provides an optional capability to discriminate the FSK frequency shift and in addition, detect the DMR sync patterns, which is critical for receiver Monitor Mode. Note phase offset correction capability is also provided at the end of the receiver datapath to ensure the signal fidelity. In the future, API command will be provided to allow user interaction. Finally, the output signal is sent through CMOS-SSI/LVDS-SSI data port to baseband processor for further processing.

ANALOG FRONT-END COMPONENTS

Analog Front Attenuator

The analog front attenuator is a PI resistive network that in conjunction with the passive mixer provides a constant 100 Ohm differential input impedance. It is controlled by the gain control functionality in the receive datapath to adjust the signal gain to avoid overload the datapath through overload detectors. When a strong interferer presents, the gain will be decreased and when the interferer disappears, the gain will be increased so that the desired signal level could be adjusted back to the proper level.

The attenuator has 256 gain settings providing an receiver attenuation range from 0 to 20*log(1/256) = -48dB. Typically, only a subset of this range will be used. In ADRV9001, the current range of the attenuation is from 0 to -30dB with a 0.5dB resolution. The gain of the attenuator is calculated by the following equation:

$$ATTEN_{dB} = 20 \times \log10 \left[\frac{256 - fe\_gain\_cw[7:0]}{256} \right]$$

The fe_gain_cw[7:0] is a 8 bit control word defined in the receiver gain table. Based on the information from the signal detectors, the gain control algorithm will find the index of this gain table so that the corresponding gain control word at this index could be used to calculate the gain at the front end attenuator. See the Rx Gain Control section for more details.

Mixer

Following the RF attenuator is the passive down converting mixer. It is used to down convert the RF signal to IF or baseband. The passive mixer uses non-overlapping ¼ duty cycle local oscillator generated by the 4 phase 50% duty cycle LO. The non-overlapping time is controlled by the duty cycle distortion (DCD) circuit. The DCD is implemented by delaying the rising edge of the 50% duty cycle LO.

LPF

In the receiver data chain, the LPF sits between the mixer and the ADC as a receiver baseband filter, supporting a baseband bandwidth of 5-50MHz. It also converts the baseband signal current to voltage. The capacitor arrays are implemented to program the various cut-off frequencies based on the system requirements. In addition, along with other AFE components, it provides a static gain of about 20dB which is independent of the gain control functionality through the receiver data chain.

The LPF could be configured in transimpedance amplifier (TIA) mode with single pole or in bi-quad (BIQ) mode with 2 complex poles in the transfer function. While the in-band performance of both modes is similar, the BIQ mode offers additional advantages comparing with the TIA mode, such as improving anti-alias filtering which might be necessary while using LP ADC. However, the use of the BIQ mode consumes about twice the power than the TIA mode.

The LPF is calibrated during device initialization to ensure a consistent frequency corner across all devices. The 3 dB bandwidth is set within the device data structure and is profile dependent. The user could optionally tune the 1dB/3dB cutoff frequency of the LPF based on their application. ADRV9001 also allows user to configure LPF at 3 different power consumption levels to help achieve system power saving target.

ADC

As mentioned previously, the ADRV9001 provides a pair of HP ADCs and a pair of LP ADCs to achieve a flexible trade-off between power consumption and linearity performance. The HP ADC is based on Continuous Time Delta Sigma (CTDS) architecture and is 5-bits wide. The LP ADC is based on voltage-controlled oscillator (VCO) architecture and is 16-bits wide. Each type of ADC is capable of accepting the same input voltage, but the output bus width is different due to the different modulator orders and presence of linearity correction in the LP ADC.

HP and LP ADCs provide a similar level of noise and dynamic range (full scale to thermal noise) performance. Therefore, the noise figure (NF) performance is similar at the input. (Even with slight NF difference at the device input, the difference at antenna input would be smaller as a result of the LNA gain in the front end.) The major difference between HP and LP ADC is the linearity performance and power consumption. The intermodulation distortion (IMD) performance of HP ADC is slightly better than LP ADC, at the expense of higher power consumption. Please refer to the data sheet for detailed information.

Given the high dynamic range of both the HP and LP ADC, very little channelization or blocker filtering occur in the analog signal chain since the HP ADC can simultaneously absorb weak signals and large blockers. Blocker suppression and channelization are then achieved efficiently in the digital signal path.

Therefore, HP ADC provides the maximum interferer tolerance, performance and LP ADC provides the best power consumption performance under slightly relaxed interferer condition. Based on the application, the user is allowed to select between HP and LP ADC for linearity and power consumption performance trade-off. In addition, user is allowed to dynamically switch HP ADC and LP ADC on the fly through API commands adi_adrv9001_Rx_AdcSwitchEnable_Set( ) and adi_adrv9001_Rx_AdcSwitch_Configure( ). The first API function is used to enable the ADC switching feature, and it should be called at STANDBY state before initial calibrations. When dynamic ADC switch is enabled, both HP ADC and LP ADC initial calibrations will be performed. The second API configures the ADC switching functionality for a specified receiver channel to operate in different modes. It should be called at CALIBRATED state after performing initial calibrations.

When receiver Monitor Mode (not supported currently) is enabled, the device might switch between the HP ADC and LP ADC to reduce power consumption. Additional algorithms are employed in ADRV9001 to compensate for the gain and delay differences while operating with different type of ADCs so any internal switch is transparent to users.

DIGITAL FRONT END COMPONENTS

DEC

In receiver data chain, a series of decimators (organized into 2 different decimation stages) are employed to convert the ADC sample rate to a desired sample rate in both NB and WB modes. The following diagram shows how the standard sampling rates for different standards are achieved through a flexible combination of decimators in the data chain. For simplicity, any other non-DEC blocks are skipped in the diagram.

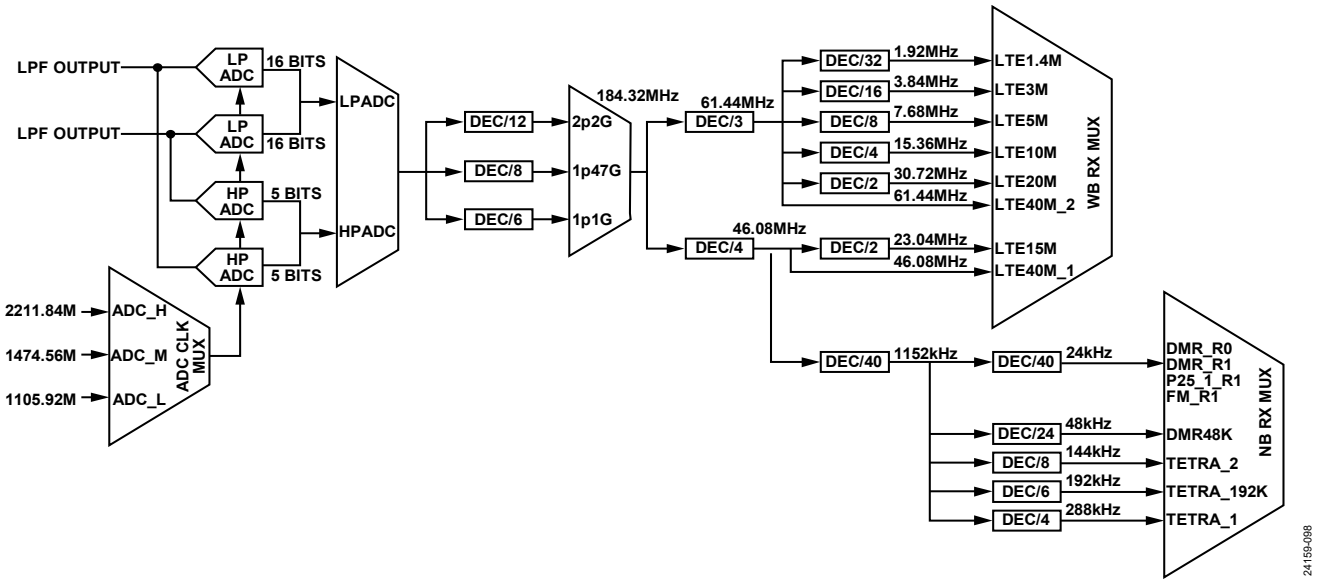


Figure 138. Decimation Schemes in Receiver Data Chain to Support Various Standards

As shown in Figure 138, in NB and WB mode, 3 different ADC output sample rates are first decimated to a fix rate of 184.32 MHz. Then, it is further converted to 2 different rates, one is 61.44 for WB mode only and the other is 46.08 MHz for both NB and WB modes. All LTE standard modes are considered WB and the desired sample rate is further generated from both 61.44 MHz and 46.08 MHz through a decimation rate of 2 to 32. DMR, FM, P25 and Tetra are NB modes and the desired sample rate is further generated from 46.08 MHz with a decimation rate of 160-1920.

For each decimator show in Figure 138, it could consist a combination of lower rate decimation filters. For example, DEC/40 could be implemented as a cascade of DEC/10 and 2 DEC/2 decimators. In addition, the different decimation rates are achieved by strategically enabling and disabling some lower rate decimators. For example, in WB mode, with an initial sample rate of 61.44 MHz, if all lower rate decimators are used, it can achieve a decimation rate of 32. If two of the DEC/2 are disabled, a decimation rate of 8 can be achieved. All the decimation filters are carefully designed to satisfy the system performance requirements.

With arbitrary sample rate, the user could get an almost continuous range of sample rates from 24 kHz to 61.44 MHz except for some "dead zones" due to internal clocking constraints. This is achieved through adjusting the internal CLK PLL frequency as well as a flexible arrangement of decimators.

DC OFFSET

The ADRV9001 receiver supports both IF down conversion and ZIF down conversion. The source of the DC offset is mainly from the receiver LO leakage caused by the finite isolation between the LO and RF ports of a mixer, which is typical for silicon-based ICs. It could generate a high DC component at the center of the desired signal band especially for ZIF operation. Through the datapath, the induced dc offset is amplified and could reduce the ADC dynamic range significantly. In addition to receiver LO leakage, the device mismatch in LPF and ADC also contributes to the DC offset problem. Without properly correcting the DC offset, it could cause a negative impact on the system performance.

In ADRV9001, a two-step approach is taken to estimate and correct the DC offset. The first step comprises of an DC estimation step in the digital domain and a correction procedure in the analog domain, which is named as RFDC. The second step is an all-digital DC offset estimation and correction technique that estimates and corrects for any residual DC offset after the first step, which is named as BBDC. BBDC is basically a notch filter and user is allowed to control the width of the notch. The default value is 1/2048, but the user can change it if they want a wider or narrower notch.

QEC

In an ideal analog mixer, the in-phase (I) and quadrature-phase (Q) sinusoidal signals are orthogonal. In addition, the I and Q path of LPF and ADC should have identical frequency responses. However, in reality, IQ imbalance always exists in the mixer, LPF and ADC, resulting in quadrature errors. Without properly handling it, it seriously degrades the reception performance. For IF reception, the respective image mixes partially onto the desired signal during the IF down conversion. In direct conversion reception, IQ-imbalance leads to a distortion of the IQ-signals themselves within the respective desired baseband channel.

In general, quadrature error can be classified as frequency independent error (FIE) and frequency dependent error (FDE). FIE is mainly caused by the mixer I/Q sinusoid mismatch in both gain and phase, while FDE is mainly caused by the inconsistent filter responses.

Because ADRV9001 supports both NB and WB modes, NBQEC and WBQEC algorithms are developed accordingly to handle quadrature error in these 2 modes effectively. NBQEC employs a time-domain adaptive algorithm to estimate both gain and phase mismatch. Then, the estimations are applied to correct the distorted input signal in real-time before passing to DDC. WBQEC designs a correction filter to cancel the effect of the mismatch filter by modeling the quadrature error generation as a mismatch filtering process. The correction filter parameters are obtained through the initial calibration by injecting RF tones into the mixer at selected frequencies and then on-the-fly adjustment by processing the Rx data in real-time.

DDC

DDC is only used when IF reception is employed. By using a programmable NCO configurable from 45kHz to 21MHz, it further converts the IF signal to the baseband.

FREQUENCY OFFSET CORRECTION

In a communication system, a desired signal is transmitted by the transmitter at RF over the air. Since the clock reference at the transmitter or the receiver are independent to each other, this may result in the RF carrier frequency offset between the transmitter and the receiver. This frequency difference is named as the carrier frequency offset (CFO). In the receiver data chain, a frequency offset correction block is provided as an option to further correct small carrier frequency offset in both NB and WB modes through an API command. The correction value must be estimated and provided by the BBIC. The correction may occur immediately or relative to receiver frame boundary. Another programmable NCO is employed with a configurable frequency -12kHz to +12kHz and FTW 32 bits wide.

The API command adi_adrv9001_Rx_FrequencyCorrection_Set( ) is used to correct small deviations in receiver LO frequency. The user must provide the frequency deviation value in Hz and specify if the correction should take place immediately or at the start of next available frame. Note the device employs the digital NCO in the datapath to correct the frequency deviation instead of RF PLL retuning.

PFIR

PFIR is an optional 128-tap programmable FIR used in both NB and WB modes. 4 sets of customized FIR profiles can be stored at the initialization phase. One of the 4 stored FIR profiles can be switched to be loaded on the fly under the control of the baseband processor.

The PFIR can be loaded a customized low-pass filter profile to stop the adjacent channel interference, which is helpful to achieve better channel selectivity. Please refer to Rx Demodulator section for more details.

RSSI

RSSI measures the receive signal power over a period of time, which could be employed to calculate the interface gain to avoid saturate the data port. In addition, in Monitor Mode, it performs signal detection in WB applications and works together with FSK Discrimination block to detect NB signals in NB applications.

The measured signal level could be retrieved by the user through an API command adi_adrv9001_Rx_Rssi_Read( ). The API function reads back the RSSI status for the given receiver channel and may be called any time after the device is fully initialized. The following data structure is used to retrieve the power measurement in both milli-dBFS and linear format:

```
typedef struct adi_adrv9001_RxRssiStatus
{
    uint16_t power_mdB;              /* Power in milli dB */
    /* Linear power is calculated by this formula: linear power = (mantissa * 2^-15) * 2^-
exponent */
    uint16_t linearPower_mantissa;   /* Mantissa of Linear Power */
    uint16_t linearPower_exponent;   /* Exponent of Linear Power */
} adi_adrv9001_RxRssiStatus_t
```

Interface Gain

Due to the bit-width limitation of the data port, an interface gain is applied by shifting the signal properly so as not to clip the output upon saturation. It could also increase the signal level for small signals to avoid losing sensitivity when a quantizer is utilized to limit the receiver output data bit-width. The interface gain could be automatically adjusted internally inside the device by utilizing the RSSI measurement or by user through API commands. The user could also optionally retrieve the signal level measured by RSSI through an API command to control the interface gain. See the Receiver Gain Control section or more details.

Phase Offset Correction

In both NB and WB modes, a phase offset correction block is provided as an option to adjust the sampling phase offset on IQ data or frequency deviation data. It re-samples the incoming received signal, by reconstructing intermediate samples between every 2 inputs samples according to the phase parameter configured by user through an API command. Currently, it is only programmable by the device. More user interaction will be provided in the future.

NB FSK Discrimination

In NB applications, the ADRV9001 device provides the capability of demodulating and detecting FSK/FM signals. This block has 2 operation modes, one is detecting mode and the other is detected mode. The detecting mode is only used when Monitor Mode is enabled. It is employed to detect the FSK/FM signals. As mentioned previously, the signal detection could be accomplished by RSSI only. However, this block could be further used in NB mode to achieve more accurate signal detection. After FSK/FM signal is detected, this block will operate in the detected mode. Some components in the datapath will be reconfigured to operate differently from the detecting mode. In case no FSK/FM signal is detected, transmitter/receiver will move to sleep mode.

It is well known that DMR and FM radio has an about 90% idle time, during which, both RF front end and baseband processor are put to sleep to save power. As a traditional solution, both baseband processor and transmitter/receiver have to power up to do the carrier detection and transmitter/receiver only passes through the data. With the equipped capability of the ADRV9001, it detects the DMR and FM signal independent of the baseband processor during its idle state, so that the baseband processor could sleep at the whole idle state to extent the battery life. Please refer to Rx Demodulator section for more details.

RECEIVE DATA CHAIN API PROGRAMMING

A set of receiver data chain APIs are provided for user interaction with the ADRV9001 device receive datapath. Some of them have been briefly discussed in the previous sections. This set of APIs could be classified into 3 categories: Receiver Gain APIs, Interface Gain APIs and Miscellaneous APIs as shown in Table 48, Table 49 and Table 50, respectively. Each table summarizes the list of API functions with a brief description for each one. More APIs will be provided to user in the future to allow more programmability of the receiver datapath. Please refer to the ADRV9001 Device API doxygen document for more details.

Table 48. A List of Rx Gain APIs

Rx Gain API Function Name	Description
adi_adrv9001_Rx_GainTable_Write	Programs the gain table settings for Rx channels.
adi_adrv9001_Rx_GainTable_Read	Reads the gain table entries for Rx channels requested.
adi_adrv9001_Rx_Gain_Set	Sets the Manual Gain Index for the given Rx channel.
adi_adrv9001_Rx_Gain_Get	Reads the Rx Gain Index for the requested Rx channel.
adi_adrv9001_Rx_GainIndex_Gpio_Configure	Configure GPIO pins to route the ADRV9001 Rx1 and Rx2 gain indices

Table 49. A List of Interface Gain APIs

Rx Gain API Function Name	Description
adi_adrv9001_Rx_InterfaceGain_Configure	Sets the Rx interface gain control configuration parameters for the given Rx channel.
adi_adrv9001_Rx_InterfaceGain_Set	Sets the Rx interface gain for the given Rx channel.
adi_adrv9001_Rx_InterfaceGain_Inspect	Gets the Rx interface gain control configuration parameters for the given Rx channel.
adi_adrv9001_Rx_InterfaceGain_Get	Gets the Rx interface gain for the given Rx channel.
adi_adrv9001_Rx_DecimatedPower_Get	Gets the decimated power at configurable locations for the specified channel.

Table 50. A List of Rx Miscellaneous APIs

Rx Miscellaneous API Function Name	Description
adi_adrv9001_Rx_Rssi_Read	Reads the received signal power measurement in both linear and dB format.
adi_adrv9001_Rx_FrequencyCorrection_Set	Corrects for small deviations in Rx LO frequency offset.
adi_adrv9001_Rx_AdcSwitchEnable_Set	Sets the readiness of dynamic switch between Low Power and High Performance ADCs.

Rx Miscellaneous API Function Name	Description
adi_adrv9001_Rx_AdcSwitchEnable_Get	Gets the readiness of dynamic switch between Low Power and High Performance ADCs.
adi_adrv9001_Rx_AdcSwitch_Configure	Configures ADC dynamic switch settings for the specified channel.
adi_adrv9001_Rx_AdcSwitch_Inspect	Inspects the current ADC dynamic switch settings for the specified channel.
adi_adrv9001_Rx_AdcType_Get	Gets the current ADC type for the specified channel.

TRANSMITTER/RECEIVER/OBSERVATION RECEIVER SIGNAL CHAIN CALIBRATIONS

In ADRV9001, to achieve optimal performance, an ARM performs calibrations which can be classified into two categories: initial calibrations performed at the initialization time before the device is operational; and tracking calibrations performed regularly while the device is operational.

Initial calibrations are considered as a part of the device initialization, which moves the device from "STANDBY" state to "CALIBRATED" state to prepare for transmit/receive operations. Tracking calibrations are performed regularly on-the-fly while the device is operational to track the changes such as attenuation, temperature and so on. As discussed in the Rx Signal Chain section of this User Guide, ADRV9001 includes 2 transmitters and 2 receivers. for each receiver, besides acting as a primary data channel for receiving RF signals, it could also serve as an observation channel, which receives transmit signals through loopback paths. The observation channel could be controlled fully by the user or internally controlled by the device for some transmitter initial and tracking calibrations. Note for some systems such as FDD 2T2R, the transmitter tracking calibrations requiring loopback paths could not be performed since the observation channel is not available. Refer to ADRV9001 Example Use Cases section for more details.

Most initial calibrations use internally generated tones or wideband signals for calibration, which need user to satisfy external system requirements. This topic will be discussed in more details in later sections. Different from initial calibrations, tracking calibrations usually use the real-time traffic data for calibration. Therefore, tracking calibrations are transparent to users which do not require any user intervention. Both initial and tracking calibrations are scheduled and performed by the ADRV9001 ARM.

INITIAL CALIBRATIONS

There are three types of initial calibrations, which are:

- System (non-channel related) initial calibrations
  - Initial calibrations for RF PLLs to calibrate the RF PLL for very fast frequency hopping mode (currently not available),
  - Aux PLL initial calibration (currently not available).
- Tx initial calibrations
  - Quadrature Error Correction (QEC),
  - Local Oscillator (LO) Leakage,
  - Loop Back Path Delay (LB PD),
  - Duty Cycle Correction (DCC),
  - Baseband Analog Filter (BBAF),
  - Baseband Analog Filter-Group Delay (BBAF GD),
  - Attenuation Delay (ATTEN DELAY),
  - Digital to Analog Converter (DAC),
  - Path Delay.
- Rx initial calibrations
  - High Power ADC Resistor/Capacitor (HP ADC RC),
  - High Power ADC Flash offset (HP ADC Flash),
  - High Power ADC DAC (HP ADC DAC) (currently not available),
  - Duty Cycle Correction (DCC),
  - Low Power ADC (LP ADC),
  - TIA Cutoff Frequency (TIA Cutoff),
  - Transimpedance Amplifier Group Delay,
  - Wideband Quadrature Error,
  - Frequency Independent Quadrature Error,
  - Internal Loop Back LOD (ILB LOD) (currently not available),
  - DC Offset (RF DC),
  - Gain Path Delay,
  - DMR Path Delay.

Note receiver initial calibrations are also required to be performed on loopback paths to prepare for transmitter initial and tracking calibrations.

To successfully perform all the initial calibrations, the ADRV9001 device should be configured properly. This is fully controlled by the ADRV9001 ARM therefore no user interaction is required. However, besides the internal configurations, there are also requirements for

the external system. For example, during some transmitter initial calibrations, tones are generated and present at transmitter output. Therefore, user should ensure appropriate level of isolation from ADRV9001 transmitter output to antenna to make sure that test tones are not transmitted by the system. This isolation could be achieved by disabling power amplifier during transmitter initial calibration.

Initial Calibrations API Programming

The ADRV9001 ARM in the device is tasked with scheduling/performing initial calibrations to optimize the performance of the device prior to device operation. Initial calibrations is performed using the top-level API function adi_adrv9001_cals_InitCals_Run( ).

The initial calibration performed is based on the initial calibration configuration defined by the following data structure:

```
typedef struct adi_adrv9001_InitCals
{
    uint32_t sysInitCalMask;
    uint32_t chanInitCalMask[ADI_ADRV9001_MAX_RX_ONLY];
    adi_adrv9001_InitCalMode_e calMode;
    bool force;
} adi_adrv9001_InitCals_t
```

In this structure, sysInitCalMask is the initial calibration mask for system calibrations, chanInitCalMask[] is an array containing calibration bit mask for channel related initial calibrations (chanInitCalMask[0] is the mask for Rx1/Tx1 channels and chanInitCalMask[1] is the mask for Rx2/Tx2 channels), calMode specifies the mode to run the desired initial calibration algorithms and force is a flag which will force all enabled calibrations to rerun when it is set to be true.

The following enumerator type defines all the initial calibrations:

```
typedef enum adi_adrv9001_InitCalibrations
{
    ADI_ADRV9001_INIT_CAL_TX_QEC               = 0x00000001,
    ADI_ADRV9001_INIT_CAL_TX_LO_LEAKAGE        = 0x00000002,
    ADI_ADRV9001_INIT_CAL_TX_LB_PD             = 0x00000004,
    ADI_ADRV9001_INIT_CAL_TX_DCC               = 0x00000008,
    ADI_ADRV9001_INIT_CAL_TX_BBAF              = 0x00000010,
    ADI_ADRV9001_INIT_CAL_TX_BBAF_GD           = 0x00000020,
    ADI_ADRV9001_INIT_CAL_TX_ATTEN_DELAY       = 0x00000040,
    ADI_ADRV9001_INIT_CAL_TX_DAC               = 0x00000080,
    ADI_ADRV9001_INIT_CAL_TX_PATH_DELAY        = 0x00000100,
    ADI_ADRV9001_INIT_CAL_RX_HPADC_RC          = 0x00000200,
    ADI_ADRV9001_INIT_CAL_RX_HPADC_FLASH       = 0x00000400,
    ADI_ADRV9001_INIT_CAL_RX_HPADC_DAC         = 0x00000800,
    ADI_ADRV9001_INIT_CAL_RX_DCC               = 0x00001000,
    ADI_ADRV9001_INIT_CAL_RX_LPADC             = 0x00002000,
    ADI_ADRV9001_INIT_CAL_RX_TIA_CUTOFF        = 0x00004000,
    ADI_ADRV9001_INIT_CAL_RX_GROUP_DELAY       = 0x00008000,
    ADI_ADRV9001_INIT_CAL_RX_QEC_TCAL          = 0x00010000,
    ADI_ADRV9001_INIT_CAL_RX_QEC_FIC           = 0x00020000,
    ADI_ADRV9001_INIT_CAL_RX_QEC_ILB_LO_DELAY  = 0x00040000,
    ADI_ADRV9001_INIT_CAL_RX_RF_DC_OFFSET      = 0x00080000,
    ADI_ADRV9001_INIT_LO_RETUNE                = 0x000B902B,
    ADI_ADRV9001_INIT_CAL_RX_GAIN_PATH_DELAY   = 0x00100000,
    ADI_ADRV9001_INIT_CAL_RX_DMR_PATH_DELAY    = 0x00200000,
    ADI_ADRV9001_INIT_CAL_PLL                  = 0x00400000,
    ADI_ADRV9001_INIT_CAL_AUX_PLL              = 0x00800000,
    ADI_ADRV9001_INIT_CAL_TX_ALL               = 0x000001FF,
```

```
    ADI_ADRV9001_INIT_CAL_RX_ALL                = 0x001FFE00,
    ADI_ADRV9001_INIT_CAL_RX_TX_ALL             = 0x001FFFFF,
    ADI_ADRV9001_INIT_CAL_SYSTEM_ALL            = 0x00C00000,
} adi_adrv9001_InitCalibrations_e
```

The following enumerator type defines the operating modes for initial calibrations:

```
typedef enum adi_adrv9001_InitCalMode
{
    ADI_ADRV9001_INIT_CAL_MODE_ALL,
    ADI_ADRV9001_INIT_CAL_MODE_SYSTEM_AND_RX,
    ADI_ADRV9001_INIT_CAL_MODE_LOOPBACK_AND_TX,
    ADI_ADRV9001_INIT_CAL_MODE_ELB_ONLY
}adi_adrv9001_InitCalMode_e;
```

in which ADI_ADRV9001_INIT_CAL_MODE_ALL is for running all the selected initial calibrations, including both receiver (non-loopback and loopback paths) and transmitter initial calibrations. ADI_ADRV9001_INIT_CAL_MODE_SYSTEM_AND_RX is for running the selected receiver initial calibrations (non-loopback paths) and ADI_ADRV9001_INIT_CAL_MODE_LOOPBACK_AND_TX is for running the selected receiver calibrations on loopback paths (both internal and external loopback paths) and the selected transmitter initial calibrations. When using external LO for both receiver and transmitter and when receiver LO and transmitter LO are at different frequencies, it takes time for user to change LO frequency, therefore, instead of running all the initial calibrations (select mode 0), the user could first set receiver LO and run receiver initial calibrations (non-loopback path) (select mode 1) and then change to Tx LO and run the receiver initial calibrations (loopback path) and transmitter initial calibrations (select mode 2). ADI_ADRV9001_INIT_CAL_MODE_ELB_ONLY is for running all the initial calibrations on external loopback paths only. Usually user should not explicitly use this mode. It is used when the user calls the adi_adrv9001_cals_ExternalPathDelay_Run() API command to get the external loopback path delay, which can be used as an input to adi_adrv9001_cals_ExternalPathDelay_Set() for characterization.

Table 51 describes the mask bit assignment for initial calibrations in adi_adrv9001_InitCalibrations_e. It also explains the functionality of each initial calibration. Note it is possible to select a different mask for Channel 1 (Tx1/Rx1) and Channel 2 (Tx2/Rx2).

Table 51. Initial Calibration Mask Bit Assignments

Bits	Corresponding Enum	Calibration	Description
D0	ADI_ADRV9001_INIT_CAL_TX_QEC	Tx QEC Initial Calibration	This performs an initial QEC calibration for frequency independent errors for the Tx path. It estimates the gain and phase mismatch and apply the gain mismatch in the digital domain. Currently it uses the Tx path and an ILB path. If transmitted data is quadrature modulated, this initial calibration is performed, but it is not used if the data modulation is direct modulation (DM).
D1	ADI_ADRV9001_INIT_CAL_TX_LO_LEAKAGE	Tx LOL Initial Calibration	This performs an initial LOL calibration. It estimates the LOL and applies the cancellation in the digital domain. Currently it uses the Tx path and an ILB path. If transmitted data is quadrature modulated, this initial calibration is performed, but it is not used if the data modulation is direct modulation (DM).
D2	ADI_ADRV9001_INIT_CAL_TX_LB_PD	Tx Loop Back Path Delay Calibration	This is used to calibrate the Tx Loop Back Path Delay (could be for either ILB or ELB). This information is required for QEC and LOL calibration. Currently it uses the Tx path and an ILB path.
D3	ADI_ADRV9001_INIT_CAL_TX_DCC	Tx DCC Initial Calibration	This corrects the 50% duty cycle for external LO when the divisor is 2.
D4	ADI_ADRV9001_INIT_CAL_TX_BBAF	Tx BBAF Initial Calibration	This is used to tune the low-pass corner frequency and the pass-band flatness of the Tx baseband analog filter.
D5	ADI_ADRV9001_INIT_CAL_TX_BBAF_GD	Tx BBAF-GD Initial Calibration	This is used to estimate and correct the filter group delay to remove frequency dependent quadrature error between the I and Q channels in each transmitter.
D6	ADI_ADRV9001_INIT_CAL_TX_ATTEN_DELAY	Tx ATTD Initial Calibration	This is used to estimate the delay between the Tx analog attenuation and digital attenuation. The delay will be

Bits	Corresponding Enum	Calibration	Description
			the same for all dynamic datapath profiles, gain indices and frequency regions and so on The calibration need only be performed on a single channel.
D7	ADI_ADRV9001_INIT_CAL_TX_DAC	Tx DAC Initial Calibration	This is used to calibrate the DAC for the required profile bandwidth.
D8	ADI_ADRV9001_INIT_CAL_TX_PATH_DELAY	Tx Path Delay Initial Calibration	This is used to estimate the delay between the Tx input and the Tx output.
D9	ADI_ADRV9001_INIT_CAL_RX_HPADC_RC	Rx HP ADC RC Initial Calibration	This is used to determine how much the unit R and C vary from ideal and then tune the HP ADC's programmable Rs and Cs to obtain their desired values. Without this calibration, the HP ADC's noise performance will be negatively impacted. The HP ADC could also become unstable. It is not used when only LP ADC is used.
D10	ADI_ADRV9001_INIT_CAL_RX_HPADC_FLASH	Rx HP ADC Flash Offset Initial Calibration	This is used to optimize the HP ADC output noise by correcting comparators offsets in the backend flash. It is not used when only LP ADC is used.
D11	ADI_ADRV9001_INIT_CAL_RX_HPADC_DAC	Rx HP ADC DAC Initial Calibration	It corrects for element mismatch HP ADC current. It is not used when only LP ADC is used. **This is disabled by default – uncalibrated performance sufficient.**
D12	ADI_ADRV9001_INIT_CAL_RX_DCC	Rx HP ADC Stability Initial Calibration	This corrects the 50% duty cycle for external LO when the divisor is 2.
D13	ADI_ADRV9001_INIT_CAL_RX_LPADC	Rx LP ADC Initial Calibration	This is used to calibrate the LP ADC. The major purpose of using the LP ADC instead of the HP ADC is to reduce power consumption. It is not used when only HP ADC is used.
D14	ADI_ADRV9001_INIT_CAL_RX_TIA_CUTOFF	Rx TIA Cutoff Initial Calibration	This is used to tune the 3dB cut-off frequency of the TIA filter.
D15	ADI_ADRV9001_INIT_CAL_RX_GROUP_DELAY	Rx TIA Fine Initial Calibration	This is used to compensate the mismatch in 3dB cutoff frequency between I and Q path. It helps to correct quadrature error in analog domain, which simplifies the correction in the digital domain.
D16	ADI_ADRV9001_INIT_CAL_RX_QEC_TCAL	Rx Tone Initial Calibration	This performs an initial QEC calibration in wideband systems for frequency dependent quadrature errors.
D17	ADI_ADRV9001_INIT_CAL_RX_QEC_FIC	Rx Frequency Independent Error Initial Calibration	This performs an initial QEC calibration for frequency independent errors for both narrowband and wideband systems.
D18	ADI_ADRV9001_INIT_CAL_RX_QEC_ILB_LO_DELAY	Rx ILB LO Delay Initial Calibration	This is used to adjust the analog delay between the inphase and quadrature LO components at the mixer on the internal loopback path. This is not enabled currently.
D19	ADI_ADRV9001_INIT_CAL_RX_RF_DC_OFFSET	Rx RFDC Offset Initial Calibration	This is used to mitigate the RFDC offset added due to LO self-mixing.
D20	ADI_ADRV9001_INIT_CAL_RX_GAIN_PATH_DELAY	Rx Gain Path Delay Initial Calibration	This is used to calculate the path delay between the Rx analog and digital attenuation blocks. This delay is then used to offset the onset of Rx analog and digital attenuations relative to each other to compensate for the path delay between these blocks. It is independent of gain index and frequency region. The calibration needs only be performed on a single channel.
D21	ADI_ADRV9001_INIT_CAL_RX_DMR_PATH_DELAY	Rx DMR Path Delay Initial Calibration	This is used to measure the delay from internal FIFO to sync detection time needed in DMR and P25 monitor mode.
D22	ADI_ADRV9001_INIT_CAL_PLL	PLL Initial Calibration	This is used to perform VCO frequency calibration, VCO real-time temperature/aging calibration and charge

Bits	Corresponding Enum	Calibration	Description
			pump calibration to make RF PLL ready for operation. It is only used for very fast frequency hopping mode. **This is not enabled currently.**
D23	ADI_ADRV9001_INIT_CAL_AUX_PLL	AUX PLL Initial Calibration	This is used to perform VCO frequency calibration, VCO real-time temperature/aging calibration and charge pump calibration to make aux PLL ready for operation. **This is not enabled currently.**

The ADRV9001 ARM proceeds through the calibrations in the required sequential order. The system initial calibrations are performed first, followed by receiver, initial calibrations and then transmitter initial calibrations. The receiver, initial calibration order and the transmitter initial calibration order are shown in Table 52 and Table 53, respectively.

Table 52. Rx Initial Calibration Order

Order	Rx Initial Calibrations
1	RX_HPADC_RC
2	RX_HPADC_FLASH
3	RX_HPADC_DAC
4	RX_LPADC
5	RX_TIA_CUTOFF
6	RX_DCC
7	RX_GROUP_DELAY
8	RX_RF_DC_OFFSET
9	RX_GAIN_PATH_DELAY
10	RX_QEC_ILB_LO_DELAY
11	RX_QEC_TCAL
12	RX_QEC_FIC
13	RX_DMR_PATH_DELAY

Table 53. Tx Initial Calibration Order

Order	Tx Initial Calibrations
1	TX_DCC
2	TX_BBAF
3	TX_DAC
4	TX_LB_PD
5	TX_LO_LEAKAGE
6	TX_QEC
7	TX_BBAF_GD
8	TX_ATTEN_DELAY
9	TX_PATH_DELAY

The calibration order is mostly determined by the algorithm dependency. It is important that the users wait for these calibrations to complete successfully before continuing with other steps of initialization for the device.

NOTE: Table 51 provides a full list of initialization calibrations for the device. Not all of these calibrations have been implemented at this time and are expected with future software updates.

System Considerations for Transmitter Initial Calibrations

In this section, high level block diagrams are used to show the device configurations and external system requirements for some transmitter initial calibrations. In all the diagrams, grayed-out lines and blocks are not active in the calibration. It should be noted that as the ADRV9001 ARM performs each of the calibrations, it is tasked with configuring the ADRV9001 device as per the diagrams below, with respect to enabling/disabling paths, and so on No user input is required in this regard. However, it is important that the user ensures that external system conditions are met, such as having the power amplifier off for all calibrations except for some initialization calibration utilizing ELB2.

Among 9 transmitter initial calibrations, except for TX_DAC, all other 8 calibrations require to insert tone/wideband signal into the transmitter datapath from an internal signal generator. Therefore, the internal microprocessor will disable the data port to avoid

interference. Some calibration algorithms, such as TX_LB_PD, TX_QEC, TX_LOL, TX_DCC and TX_ATTEN_DELAY further require the use of observation datapath through ILB or ELB to receive the transmitted signal so that a joint analysis can be performed by observing the relationship between the transmitted signal and received signal. As aforementioned, currently, only ILB feedback path is supported for initial calibrations.

Transmitter Initial Calibrations Utilizing Internal Signal Generation Without Loopback

Figure 139 shows a high level block diagram of system configurations for transmitter initial calibrations requiring inserting signals into the transmitter datapath without using loopback path.

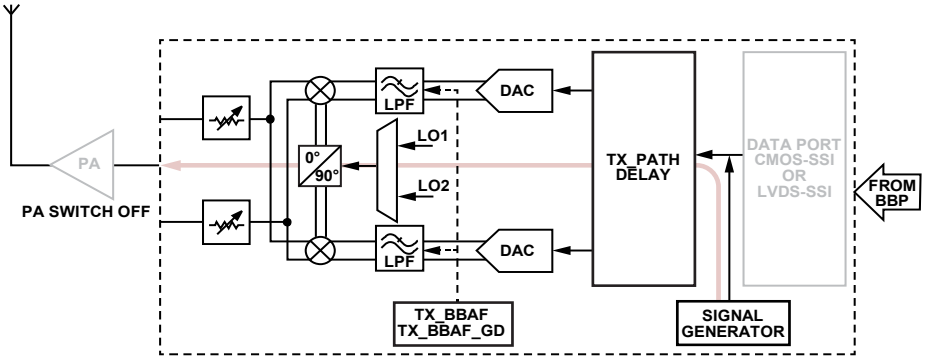


Figure 139. Transmitter Initial Calibration System Configuration with Signal Generation Without Loop Back

As shown in Figure 139, an internal signal generator inserts calibration signals (red) into the transmitter datapath. The data port is disabled during initial calibrations to avoid producing interference. TX_PATH_DELAY is performed in the digital domain, whereas TX_BBAF and TX_BBAF_GD are performed in the analog domain. All of them use the signal generator to insert tones for calibration. During all these calibrations, test signals inserted into the transmitter datapath can appear at the transmitter output, so it is important that the power amplifier connected to the device output be switched off, which also prevents signals from the antenna reaching the transmitter during calibrations. 50 Ω termination is also needed to prevent tone signals bouncing back from power amplifier input and reaching the device transmitter output, thereby confusing internal calibrations. The following paragraph summarizes the external system requirement.

External system requirement: for transmitter initial calibrations utilizing internal signal generation without loopback, the power amplifier in the transmitter path should be powered off during these calibrations. When the power amplifier is disabled, the load seen at the transmitter output should be 50 Ω.

Transmitter Initial Calibrations Utilizing Internal Signal Generation and ILB

Figure 140 shows a high level block diagram of system configurations for transmitter initial calibrations utilizing internal signal generation and ILB path.

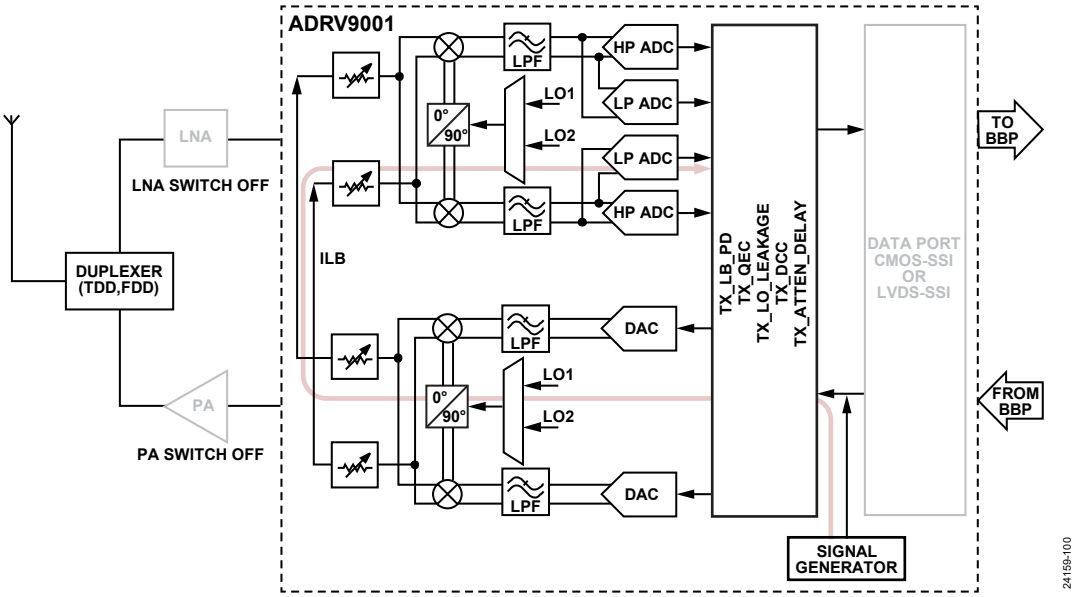


Figure 140. Tx Initial Calibration System Configuration with Signal Generation and Internal Loop Back

As shown in Figure 140, TX_LB_PD, TX_QEC, TX_LO_LEAKAGE, TX_DCC, and TX_ATTEN_DELAY use the ILB for calibrations. TX_LO_LEAKAGE and TX_QEC calculate the initial correction parameters. TX_LB_PD provides a measurement of the loop back path delay for TX_LO_LEAKAGE and TX_QEC algorithms. Both TX_LO_LEAKAGE and TX_QEC calibrations sweep through a series of attenuation values, creating a table of initial calibration values. Then during operation and upon application of a new transmitter attenuation setting, the corresponding QEC and LO_LEAKAGE correction values are applied to the transmitter channel by the ADRV9001 ARM. TX_DCC estimates the duty cycle error in the digital domain but applies the correction in the analog domain. TX_ATTEN_DELAY measures the delay between the transmitter digital attenuation block and transmitter analog attenuation block and it uses the ILB for delay observation and estimation.

During all these calibrations, similarly, the power amplifier connected to the device output should be switched off and 50 Ω termination is needed. In addition, it is important to switch off the LNA (or RF switch if no LNA presented externally) external to the receiver datapath to avoid the interference from the RF port into Rx input used for data traffic. The following paragraph summarizes the external system requirement.

External system requirement: for transmitter initial calibrations using ILB, the power amplifier in the transmitter path should be powered off during these calibrations. When the power amplifier is disabled, the load seen at the transmitter output should be 50 Ω. The LNA (or RF switch if no LNA presented externally) for the loopback path should also be switched off to avoid receiving signals from RF port.

Initial Transmitter Calibration Utilizing Internal Signal Generation and ELB

Although not currently supported, it is also possible to perform some transmitter initial calibrations using ELB. As mentioned in the Receiver/Observation Receiver Signal Chain section, using ELB1 for initial calibrations provides the advantage of observing common mode voltage. When ELB1 is used, it has the same external system requirements as using ILB. Figure 141 shows the high level block diagram of initial transmit calibrations using internal signal generation and ELB1.

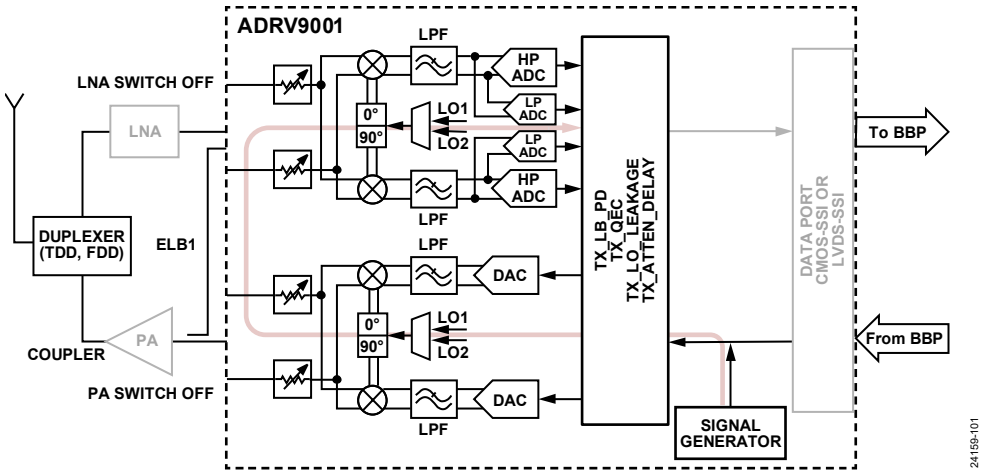


Figure 141. Transmitter Initial Calibration System Configuration with Signal Generation and External Loop Back Type 1

For TX_LO_LEAKAGE, another option is to use ELB2. Figure 142 shows the high level block diagram of system configurations for TX_LO_LEAKAGE initial calibrations using ELB2 (Note TX_LB_PD initial calibration using ELB2 is required for TX_LO_LEAKAGE).

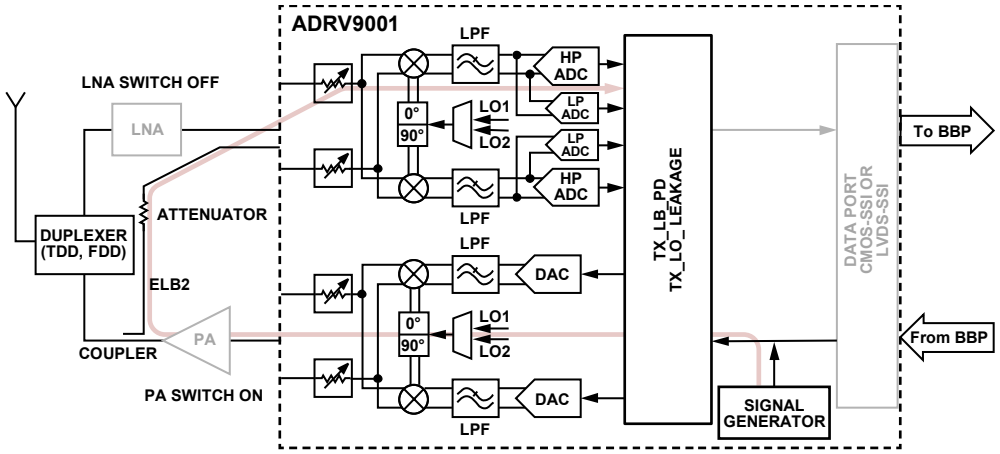


Figure 142. Tx Initial Calibration System Configuration with Signal Generation and External Loop Back Type 2

When ELB2 feedback path is used, it requires that the power amplifier be enabled such that a full external loop is made between the transmitter outputs and the observation channel inputs. The advantage of this calibration is to obtain a good estimate (gain/phase) of the external loop channel conditions prior to operation. The device configuration is shown in Figure 133. Note in this case, the calibration signal might be transmitted out through the antenna. Although the power level of calibration signal is set as low as possible, user should make sure that this will not cause any problem when using this option.

It is important that a suitable attenuator be chosen between the power amplifier output and the observation channel input. This is to prevent transmit data from saturating the observation channel input. The following paragraph summarizes the external system requirement.

External system requirement: a suitable attenuator must be chosen between the power amplifier output and the observation channel input to prevent transmit data from saturating the observation channel input. The LNA (or RF switch if no LNA presented externally) for the loopback path should be switched off to avoid receiving signals from RF port.

System Considerations for Receiver Initial Calibrations

In this section, similarly, high level block diagrams are used to show the device configurations and external system requirements for receiver initial calibrations. In all the diagrams, grayed-out lines and blocks are not active in the calibration. Blue blocks are related calibrations. It should be noted that the ADRV9001 ARM performs each of the calibrations. It is tasked with configuring the ADRV9001 device as per the diagrams below, with respect to enabling/disabling paths, and so on. No user input is required in this regard. However, it is important that the user ensures that external system conditions are met, such as having the receiver input properly terminated for Rx initialization calibrations.

Among 13 different receiver initial calibrations, RX_HPADC_RC, RX_HPADC_FLASH, RX_HPADC_DAC (not enabled) and RX_LPADC calibration are performed in the analog domain and the corrections are applied to HP ADC or LP ADC (based on which one is used), while all other receiver initial calibrations are performed in the digital domain. For RX_QEC_FIC, RX_QEC_TCAL, RX_GAIN_PATH_DELAY and RX_DMA_PATH_DELAY, the calibration results are applied in digital domain for correction. For RX_DCC, RX_RF_DC_OFFSET, RX_TIA_CUTOFF, RX_GROUP_DELAY and RX_QEC_ILB_LO_DELAY the calibration results are applied in the analog domain for correction. Figure 134 shows the high level block diagram of system configurations for receiver initial calibrations. Note different calibration performs at different locations in the receiver datapath which is simplified in Figure 143.

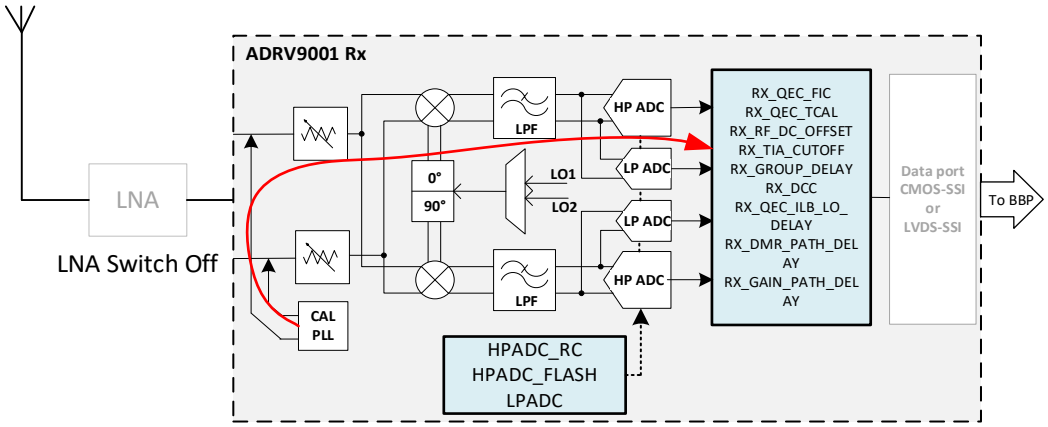


Figure 143. Receiver Initial Calibration System Configuration

During receive initial calibration, as shown in Figure 143, the data port is disabled to avoid sending data to baseband processor. This is controlled by ADRV9001 ARM which requires no user interaction. Except for RX_RF_DC_OFFSET calibration, all other digital domain calibration algorithms require injecting calibration tones generated by calibration PLL and injected internally at the receiver input. For example, the RX_QEC_TCAL calibration routine sweeps a number of internally generated test tones across the desired frequency band and then measures quadrature performance and calculates correction coefficients. Therefore, during receive calibration, it is required to not receive any incoming signals from RF port which could interfere with the calibration tones. To ensure that, it is important to isolate the device receiver input port from incoming signals by disabling LNA (or by switching off the external RF switch if no LNA is presented externally). This also prevents the calibrations tones from reaching antenna through RF coupling. 50Ω termination is needed to prevent tone signals bouncing back from external LNA output and reaching receiver input confusing internal calibrations. The following paragraph summarizes the external system requirement.

External system requirement: for optimal performance, and lower calibration duration, during receiver initial calibrations, the device receiver input port should be isolated from incoming signals. For many receiver calibrations, the calibration tones will appear on the receiver pins, therefore, must be prevented from reaching the antenna through the receiver port being properly terminated. This also

prevents the calibrations tones from reaching antenna through RF coupling. 50Ω termination is needed to prevent tone signals bouncing back from external LNA output and reaching receiver input confusing internal calibrations.

Configure the Initial Calibrations Through TES

To achieve optimal performance, all initial calibrations should be enabled. However, the TES provides the capability to allow user to disable some initial calibrations, mainly for debugging purpose. Table 54 summarizes a comparison for all initial calibrations in terms of "User Override Capability", "Run at Boot", "Signal Used by Calibration", "External Termination Needed" and "Dependency". A minimum set of initial calibrations must be rerun after "LO changes equal to or more than a certain range (for example, 100MHz) or divide by 2 boundary change" and this set of initial calibrations is defined by the bit mask "ADI_ADRV9001_INIT_LO_RETUNE = 0x000B902B" in "adi_adrv9001_InitCalibrations_e" enumerator type as suggested by Table 54. Note use can also rerun the full suite of initial calibrations to achieve the optimal performance. Figure 144 and Figure 145 demonstrate the Tx LO leakage performance and Tx image rejection performance under different initial calibrations. In this experiment, the LO is swept from 100MHz to 2.9GHz in 100MHz step size. The blue line stands for the performance if the full suite of initial calibrations are rerun at each LO change. The red line stands for the performance of running the full suite of initial calibrations and then the minimum set of initial calibrations alternatively (full initial calibrations for LO=100MHz, 300MHz, 500MHz,… and minimum initial calibrations for LO=200MHz, 400MHz, 600MHz,….). Grey line stands for the performance of running the full suite of initial calibrations and then no initial calibrations alternatively (full initial calibrations for LO=100MHz, 300MHz, 500MHz,… and no initial calibrations for LO=200MHz, 400MHz, 600MHz,….). It can be seen clearly that without rerunning any initial calibrations after LO change = 100MHz will cause a significant performance penalty. However, the optimal performance (achieved by rerunning the full suite of initial calibrations) can be largely kept if rerunning the minimum set of initial calibrations.

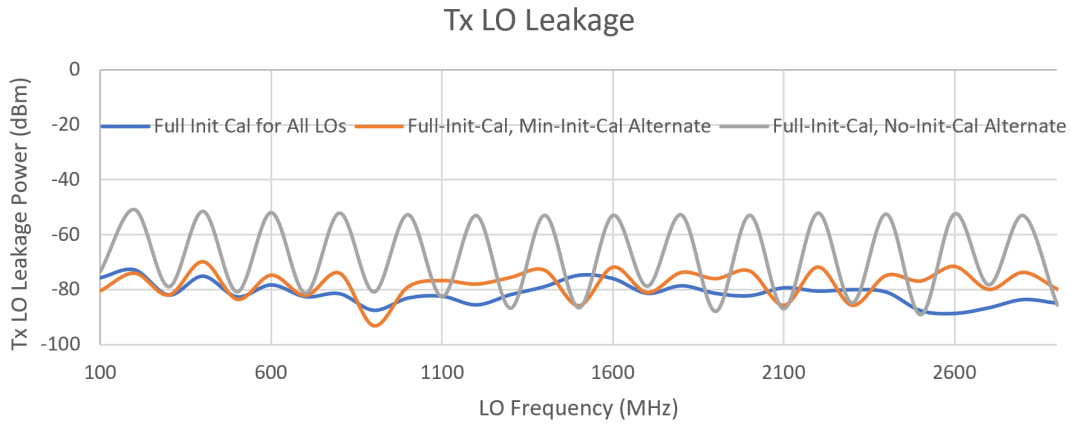


Figure 144. Tx LO Leakage Performance when LO Change = 100MHz with Full Init Cals, Min Init Cals and No Init Cals

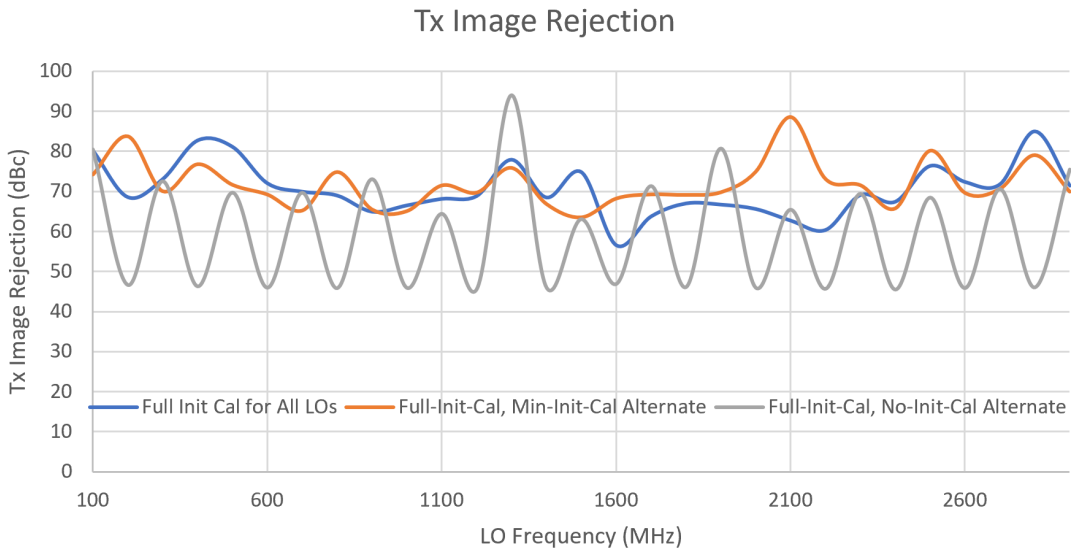


Figure 145. Tx Image Rejection Performance when LO Change = 100MHz with Full Init Cals, Min Init Cals and No Init Cals

Table 54. Initial Calibration Comparison Summary

Bits	Enum	User Override Capability	Run at Boot	Re-Run after LO Change >100 MHz or Run After ÷2 Boundary Change	Signal Used by Calibration (Tones, Wide-band, None)	External Termination Needed	Dependent on Which Init Cals to be Run First
D0	TX_QEC	Yes	Yes	Yes	Tone	Yes	TX_DAC, TX_BBAF, all RX Init Cals on ILB, TX_LB_PD
D1	TX_LO_ LEAKAGE	Yes	Yes	Yes	Wideband	Yes	TX_DAC, TX_BBAF, all RX Init Cals on ILB and ELB, TX_LB_PD
D2	TX_LB_PD	Yes	Yes	No	Wideband	Yes	TX_DAC, TX_BBAF, RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC, RX_TIA_CUTOFF, RX_RF_DC_OFFSET
D3	TX_DDC	No	Yes	Yes	Tone	Yes	None
D4	TX_BBAF	No	Yes	No	Tone	Yes	None
D5	TX_BBAF_GD	No	Yes	Yes	Tone	Yes	TX_BBAF, TX_QEC
D6	TX_ATTEN_ DELAY	No	Yes	No	Tone	Yes	TX_DAC, RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC
D7	TX_DAC	No	Yes	No	None	No	None
D8	TX_PATH_DELAY	Yes	Yes	No	Tone	Yes	TX_ATTEN_DELAY
D9	RX_HPADC_RC	No	Yes	No	None	No	None
D10	RX_HPADC_ FLASH	No	Yes	No	None	No	None
D11	RX_HPADC_DAC	Not enabled	Not enabled	Not enabled	Not enabled	Not enabled	Not enabled
D12	RX_DDC	No	Yes	Yes	Tone	Yes	RX_TIA_CUTOFF
D13	RX_LPADC	No	Yes	No	None	No	None
D14	RX_TIA_CUTOFF	No	Yes	No	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC
D15	RX_GROUP_DELAY	No	Yes	Yes	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC, RX_TIA_CUTOFF
D16	RX_QEC_TCAL	No	Yes	Yes	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC, RX_RF_DC_OFFSET, RX_TIA_CUTOFF, RX_GROUP_DELAY
D17	RX_QEC_FIC	Yes	Yes	Yes	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC, RX_RF_DC_OFFSET, RX_TIA_CUTOFF, RX_GROUP_DELAY
D18	RX_QEC_ILB_LO_DELAY	No	Yes	No	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC
D19	RX_RF_DC_ OFFSET	Yes	Yes	Yes	None	No	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC
D20	RX_GAIN_PATH_ DELAY	No	Yes	No	Tone	Yes	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC
D21	RX_DMR_PATH_DELAY	No	Yes	No	None	No	RX_HPADC_RC, RX_HPADC_FLASH, RX_LPADC

Bits	Enum	User Override Capability	Run at Boot	Re-Run after LO Change >100 MHz or Run After ÷2 Boundary Change	Signal Used by Calibration (Tones, Wide-band, None)	External Termination Needed	Dependent on Which Init Cals to be Run First
D22	PLL	Not enabled	Not enabled	No	Not enabled	Not enabled	Not enabled
D23	AUXPLL	Not enabled	Not Enabled	No	Not enabled	Not enabled	Not enabled

For the optional initial calibrations, the TES provides the option to enable or disable those calibrations as shown in Figure 135. Note in the current release, the configurable transmitter initial calibrations are LO Leakage (TX_LO_LEAKAGE), Loop Path Delay (TX_LB_PD), QEC (TX_QEC) and Duty Cycle Correction (external LO only) (TX_DDC). When TX_LO_LEAKAGE or TX_QEC is enabled, TX_LB_PD must be enabled too. When "Tx Direct FM/FSK" mode is selected in DMR or AnalogFM profiles, TX_LO_LEAKAGE, TX_QEC and TX_LB_PD calibrations are not used. Those options become unconfigurable in TES. TX_DDC is only applicable when external LO is used for transmitter. The configurable receiver initial calibrations are QEC frequency independent (RX_QEC_FIC), RFDC (RX_RF_DC_OFFSET) and Duty Cycle Correction (external LO only) (RX_DDC). Similarly, RX_DDC is only applicable when external LO is used for Rx.

The initial calibrations are performed when user clicks **Program** button in TES. It takes some time to complete all the calibrations (The calibration time is still under optimization.). When it is successful, the TES will show a status as "Programmed". If not, it will show "Programming Failed". When "Programming Failed" happens, the user could try the "Program" again. If it continues to fail after multiple attempts, as the next step, the user could enable/disable the optional initial calibrations to see if the problem could be related to some calibrations. Similarly, when performance issues are observed during the test, the user could play with the optional initial calibrations as a preliminary debug method. In the future TES releases, more configurable calibration options will be provided.

Figure 146. Initial Tx/Rx Calibration Configuration through TES

TRACKING CALIBRATIONS

There are 14 different types of tracking calibrations, which can be classified into transmitter tracking calibrations and receiver/observation receiver tracking calibrations:

Transmitter Tracking Calibrations

- Quadrature Error Correction tracking calibration (QEC)
- Local Oscillator Leakage tracking calibration (LOL)
- Loopback path delay tracking calibration (LB PD) (currently not available)
- Power Amplifier Correction tracking calibration (PAC) (currently not available)
- Digital Pre-distortion tracking calibration (DPD)
- Close Loop Gain Control (CLGC)

Receiver/Observation Receiver Tracking Calibrations

- Harmonic Distortion (2nd order) tracking calibration (HD2)
- Receiver Quadrature Error Correction Wideband Poly tracking calibration (Rx QEC WBPLOY)
- Observation Receiver Quadrature Error Correction Wideband Poly tracking calibration (ORx QEC WBPLOY
- Baseband DC offset tracking calibration (BBDC)
- RF DC tracking calibration (RFDC)
- Quadrature Error Correction FIC tracking calibration (QEC FIC)
- Automatic Gain Control tracking calibration (AGC)
- RSSI tracking calibration

All the tracking calibrations are fully controlled by the ADRV9001ARM therefore no user interaction is required.

Tracking Calibrations API Programming

The ADRV9001 ARM in the device is tasked with scheduling/performing tracking calibrations to optimize the performance of the device during its operation. Tracking calibrations are performed using the top-level API function adi_adrv9001_cals_Tracking_Set( ).

The tracking calibrations performed is based on the tracking calibration configuration defined by the following data structure:

```
typedef struct adi_adrv9001_TrackingCals
{
    adi_adrv9001_TrackingCalibrations_e chanTrackingCalMask[ADI_ADRV9001_MAX_RX_ONLY];
} adi_adrv9001_TrackingCals_t
```

In this structure, chanTrackingCalMask[] is an array containing calibration bit mask for channel related tracking calibrations (chanTrackingCalMask[0] is the mask for Rx1/Tx1 channels and chanTrackingCalMask[1] is the mask for Rx2/Tx2 channels).

The following enumerator type defines all the initial calibrations:

```
 typedef enum adi_adrv9001_TrackingCalibrations
{
    ADI_ADRV9001_TRACKING_CAL_TX_QEC            = 0x00000001,
    ADI_ADRV9001_TRACKING_CAL_TX_LO_LEAKAGE     = 0x00000002,
    ADI_ADRV9001_TRACKING_CAL_TX_LB_PD          = 0x00000004,
    ADI_ADRV9001_TRACKING_CAL_TX_PAC            = 0x00000008,
    ADI_ADRV9001_TRACKING_CAL_TX_DPD            = 0x00000010,
    ADI_ADRV9001_TRACKING_CAL_TX_CLGC           = 0x00000020,
    /* Bit 6-7: Not used (Reserved for future purpose) */
    ADI_ADRV9001_TRACKING_CAL_RX_HD2            = 0x00000100,
    ADI_ADRV9001_TRACKING_CAL_RX_QEC_WBPOLY     = 0x00000200,
    /* Bit 10-11: Not used (Reserved for future purpose) */
    ADI_ADRV9001_TRACKING_CAL_ORX_QEC_WBPOLY    = 0x00001000,
    /* Bit 13-18:  Not used (Reserved for future purpose) */
    ADI_ADRV9001_TRACKING_CAL_RX_BBDC           = 0x00080000,
    ADI_ADRV9001_TRACKING_CAL_RX_RFDC           = 0x00100000,
    ADI_ADRV9001_TRACKING_CAL_RX_QEC_FIC       = 0x00200000,
    ADI_ADRV9001_TRACKING_CAL_RX_AGC            = 0x00400000,
    ADI_ADRV9001_TRACKING_CAL_RX_RSSI           = 0x00800000
    /* Bit 24-31: Not used */
} adi_adrv9001_TrackingCalibrations_e
```

Table 55 describes the mask bit assignment for tracking calibrations in the "adi_adrv9001_TrackingCalibrations_e". It also explains the functionality of each tracking calibration. Note it is possible to select different mask for Channel 1 (Tx1/Rx1) and Channel 2 (Tx2/Rx2).

Table 55. Tracking Calibration Mask Bit Assignments

Bits	Corresponding Enum	Calibration	Description
D0	ADI_ADRV9001_TRACKING_CAL_TX_QEC	Tx QEC Tracking Calibration	This performs tracking QEC calibration for frequency independent errors for the Tx path. It estimates the gain and phase mismatch on-the-fly using the real-time traffic data and apply the gain mismatch in the digital domain. Similar as the initial calibration, currently it uses the Tx path and an ILB path. If transmitted data is quadrature modulated, this tracking calibration is performed, but it is not used if the data modulation is direct modulation (DM).

Bits	Corresponding Enum	Calibration	Description
D1	ADI_ADRV9001_TRACKING_CAL_TX_LO_LEAKAGE	Tx LOL Tracking Calibration	This performs tracking LOL calibration. It estimates the LOL on-the-fly and applies the cancellation in the digital domain. It uses the Tx path and an loopback path (external loopback path preferred if available). If transmitted data is quadrature modulated, this calibration is performed, but it is not used if the data modulation is direct modulation (DM).
D2	ADI_ADRV9001_TRACKING_CAL_TX_LB_PD	Tx Loop Back Path Delay Tracking Calibration	This is used to track the Tx Loop Back Path Delay (could be for either ILB or ELB) on-the-fly. This information is required for QEC, LOL and DPD tracking calibrations. Currently this tracking calibration is not available. QEC, LOL and DPD tracking calibration use the delay measurement obtained from Tx loopback path delay initial calibration.
D3	ADI_ADRV9001_TRACKING_CAL_TX_PAC	Tx PAC Tracking Calibration	This is used to perform power amplifier correction. Currently this tracking calibration in not available.
D4	ADI_ADRV9001_TRACKING_CAL_TX_DPD	Tx DPD Tracking Calibration	This is used to pre-distort the transmit signal in real-time to compensate for the power amplifier nonlinearity for achieving higher power efficiency. Please refer to the Digital Predistortion section in the User Guide for more details.
D5	ADI_ADRV9001_TRACKING_CAL_TX_CLGC	Tx CLGC Tracking Calibration	This is used to compensate for the gain variation in power amplifier.
D6	Reserved		
D7	Reserved		
D8	ADI_ADRV9001_TRACKING_CAL_RX_HD2	Rx HD2 Tracking Calibration	This is used to correct the Rx 2nd order harmonic distortion.
D9	ADI_ADRV9001_TRACKING_CAL_RX_QEC_WBPOLY	Rx QEC WB PLOY Tracking Calibration	This is used to correct Rx frequency dependent quadrature error for WB applications.
D10	Reserved		
D11	Reserved		
D12	ADI_ADRV9001_TRACKING_CAL_ORX_QEC_WBPOLY	ORx QEC WB Tracking Calibration	This is used to correct ORx frequency dependent quadrature error for WB applications.
D13	Reserved		
D14	Reserved		
D15	Reserved		
D16	Reserved		
D17	Reserved		
D18	Reserved		
D19	ADI_ADRV9001_TRACKING_CAL_RX_BBDC	Rx BBDC Tracking Calibration	This is used to mitigate the Rx DC offset at baseband on-the-fly.
D20	ADI_ADRV9001_TRACKING_CAL_RX_RFDC	Rx RFDC Tracking Calibration	This is used to mitigate the Rx DC offset at RF (analog) on-the-fly.
D21	ADI_ADRV9001_TRACKING_CAL_RX_QEC_FIC	Rx QEC FIC Tracking Calibration	This is used to correct Rx frequency independent quadrature error for both NB and WB applications.
D22	ADI_ADRV9001_TRACKING_CAL_RX_AGC	Rx AGC Tracking Calibration	This is used to enable/disable automatic Rx gain control operation on-the-fly. By disabling it, AGC stops responding to changes in the input signal and will instead hold its last gain index.

Bits	Corresponding Enum	Calibration	Description
D23	ADI_ADRV9001_TRACKING_CAL_RX_RSSI	Rx RSSI Tracking Calibration	This is used to enable/disable Rx signal strength measurement on-the-fly.

External System Requirements for Tracking Calibrations

Different from initial calibrations, tracking calibrations are performed on-the-fly with real-time traffic data. Therefore, it is mostly transparent to the users and fully controlled by the internal micro-processor. The external system requirements for users are as the following:

- Make sure external paths are available when running some tracking calibrations on external loopback paths.
- When external loopback path after power amplifier is used (ELB2), a suitable attenuator must be chosen between the power amplifier output and the observation channel input to prevent transmitter output data from saturating the observation channel input.
- When external DPD is employed in the system, it should time share with other transmitter tracking calibrations to avoid conflicts.

For tracking calibrations, the TES provides the option to enable or disable those calibrations as shown in Figure 147. Note in the current release, the configurable transmitter tracking calibrations are digital pre-distortion (TX_DPD), LO Leakage (TX_LO_LEAKAGE) and QEC (TX_QEC). When "Tx Direct FM/FSK" mode is selected in DMR or AnalogFM profiles, TX_LO_LEAKAGE and TX_QEC calibrations are not applicable. Those options become unconfigurable in TES. The configurable receiver tracking calibrations are automatic gain control (RX_AGC), baseband DC offset (RX_BBDC), (2nd order) harmonic distortion (RX_HD2), frequency independent QEC (RX_QEC_FIC) and frequency dependent QEC for WB (RX_QEC_WBPOLY).

The tracking calibrations can be enabled or disabled on-the-fly when the device is operational.

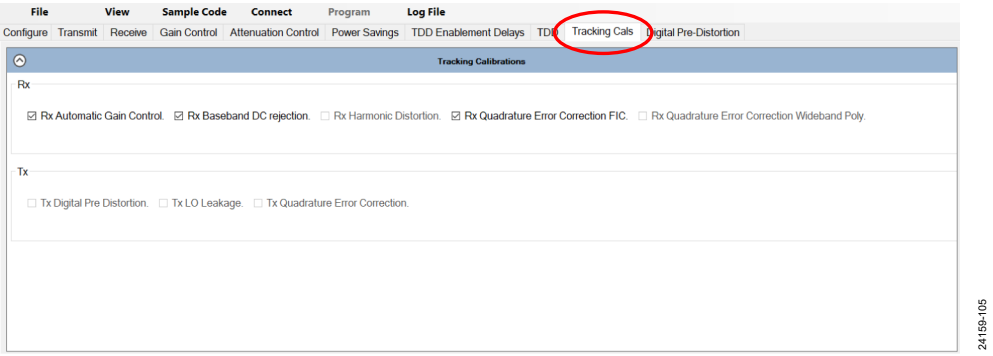


Figure 147. Tracking Tx/Rx Calibration Configuration Through TES

RECEIVER GAIN CONTROL

The ADRV9001 receivers feature automatic and manual gain control modes for flexible gain control in a wide array of applications. It controls the gain at various stages of the receiver datapath to avoid overloading during the onset of a strong interferer. In addition, it could ensure that the receiver digital output data is representative of the RMS power of the receiver input signal so that any internal front-end gain changes to avoid overloading are transparent to the baseband processor.

In ADRV9001, the two gain control modes are named Automatic Gain Control (AGC) and Manual Gain Control (MGC). AGC allows for receivers to autonomously adjust the receiver gain depending on variations of the input signal. It controls the gain of the device based on the information from a number of signal detectors named peak detector and power detector. The receivers are also capable of operating in MGC mode where changes in gain are initiated by the baseband processor through API commands or Digital GPIO (DGPIO) pins. In the MGC mode, by enabling the signal detectors, baseband processor could optionally use the information provided from signal detectors through Digital GPIO (DGPIO) pins to properly control the gain.

The gain control is highly flexible and can be configured differently in various scenarios. For example: for BTS receivers the received signal is a multi-carrier signal in most cases. A gain change should be performed only under large over range or under range conditions and gain changes should not occur very often for typical 3G/4G operations. In such a case, it might be sufficient to use peak detectors. Nevertheless, if an asynchronous blocker does appear, a "fast attack" mode exists which could reduce the gain at a fast rate. As another example, to support GSM blockers and radar pulses which have fast rise and rapid fall times, a "fast attack and fast recovery" mode can be employed. This mode is capable of fast recovery in addition to the fast attack as mentioned earlier.

Section Topics

The list of topics reviewed in detail are found in the following sections:

- Receiver Data Path: This section outlines the gain control and signal observation elements of the receiver chain, followed by a description of the receiver gain table concept.
- Gain Control Modes: This section advises how to select between AGC and MGC mode, followed by a detailed description of how to operate in each mode. In AGC mode, peak detect mode and peak/power detect mode will be further discussed and compared.
- Gain Control Detectors: This section outlines the operation and configuration of various gain control detectors in the device.
- AGC Clock and Gain Block Timing: This section describes the speed of the AGC clock and the various gain event and delay timers.
- Analog Gain Control API Programming: This section outlines how to configure the analog gain control using API commands, explaining each parameter of the API structures. It also provides a summary of all API functions currently supported.
- Digital Gain Control and Interface Gain (Slicer): This section outlines the various forms of digital gain control available in the ADRV9001.
- Digital Gain Control and Interface Gain API Programming: This section outlines how to configure the digital gain control and interface gain using API commands, explaining each parameter of the API structures. It also provides a summary of all API functions currently supported.
- Usage Recommendations: This section provides a list of recommended gain control configurations to achieve optimal performance.
- TES Configuration and Debug Information: This section advises user how to configure receiver gain control functionality through TES and perform simple debugging when some gain control performance problems are encountered.

Important Terminology

A list of important terms used in following sections are summarized below:

Manual Gain Control (MGC)
This term is used to refer to a use case when the user is in control of the currently applied gain settings in the receiver chain.

Automatic Gain Control (AGC)
This term is used to refer to the device's own internal AGC, where the device is in control of the receiver gain settings. If the user does not use the internal AGC, then it is expected that an AGC would run in the baseband processor. However, in this document such a case would be referred to as MGC because the gain of the receive path is under user's control.

Gain Attack
This term is used to indicate the reduction of the receiver gain due to an overloaded signal path.

Gain Recovery
This term is used the indicate the increase of the receiver gain due to a reduction in the power of the signal being received.

Gain Compensation
The process of compensating for the analog attenuation in the device (prior to the ADC) with a corresponding amount of digital gain before the digital signal is sent to the user.

High Threshold
This threshold is used to trigger gain attack event. Some detectors could have multiple high thresholds.

Low Threshold
This threshold is used to trigger gain recovery event. Some detectors could have multiple low thresholds.

Threshold Overload
When a threshold is exceeded in a signal detector, this is referred to as an overload.

Threshold Underload
When a threshold is not exceeded in a signal detector, this is referred to as an underload.

Overrange Condition
An overrange condition exists when the AGC is required to reduce the gain. This can either be a peak condition, where a programmable number of individual overloads of a high threshold have occurred within a defined period of time, or a power condition, where the measured power exceeds a high power threshold.

Underrange Condition
An underrange condition exists when the AGC is required to increase the gain. This can either be a peak condition, where a lower threshold is not exceeded a programmable number of times within a defined period of time, or a power condition, where the measured power does not exceed a low power threshold.

RECEIVER DATAPATH

Figure 148 shows the simplified receiver datapath and gain control blocks. The receivers have front end attenuators prior to the mixer stage that are used to attenuate the signal in the analog domain to ensure the signal does not overload the receiver chain. Note ADRV9001 provides about 20dB gain so the front end gain attenuator further attenuates signal from that level. In the digital domain, there is also digital gain control capability.

As shown in this figure, the receiver chain has a number of observation elements that can monitor the incoming signal. These can be used in either MGC or AGC mode. Firstly, an Analog Peak Detector (APD) exists prior to ADC. Being in the analog baseband, this peak detector will see signals first, and will also have visibility of interfering signals which can overload the ADC but could be filtered as they progress through the digital chain. The second peak detector is called the HB Peak Detector since it monitors the data at the output of the Half Band (HB) Filtering block in the receiver chain.

A power measurement detection block is also provided at the same output of HB Filtering block which takes the RMS power of the received signal over a configurable period of time.

Besides the front end gain control, this device could also control an external gain element through analog GPIO (AGPIO) pins. In the digital domain, this device can further control the digital gain in both wideband (WB) and narrowband (NB) modes. To avoid saturate the output signal due to the limitation of the bit width of data port, an optional interface gain (slicer) is applied at the end of the datapath by properly shifting the data. The interface gain could be automatically controlled internally inside the device by utilizing the information provided from the Receiver Signal Strength Indicator (RSSI) block or manually controlled by users through API command.

As shown in Figure 148, the gain control block has multiple inputs, which come from 2 peak detectors the 1 power detector. By utilizing the information from those detectors, the gain control block controls the gain of the signal chain using a predefined gain table. Note the default gain table is loaded into the device during initialization. An API function adi_adrv9001_Rx_GainTable_Write() can be called by the user to load a custom gain table or reconfigure the gain table. Note this operation should be done before performing initial calibrations.

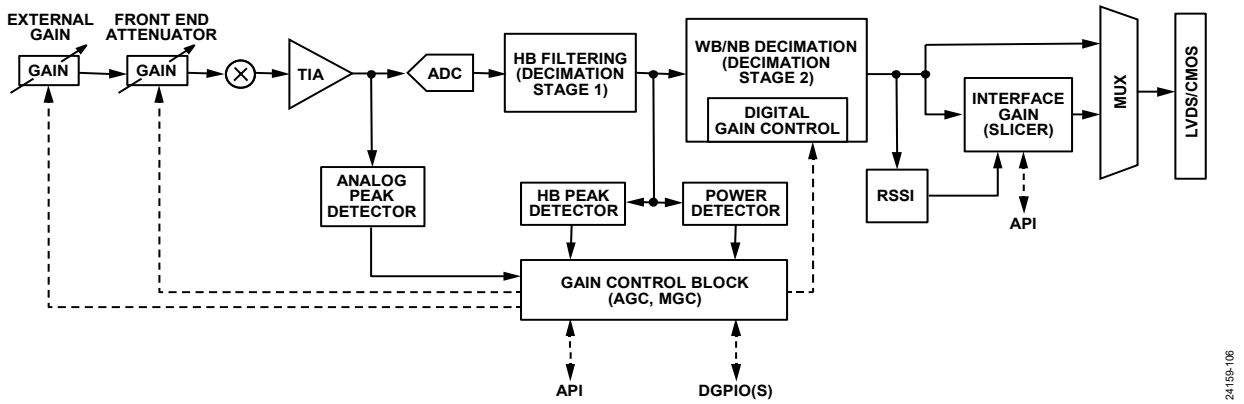


Figure 148. Rx Data Path and Gain Control Blocks

In this gain table, each row provides a unique combination of 6 fields including Front-end Attenuator, TIA Control, ADC Control, External Gain Control, Phase Offset and Digital Gain/Attenuator. Among them, the TIA Control which sets the TIA gain, the ADC Control which sets the ADC gain and the Phase Offset which compensates for the phase discontinuity during gain change are reserved for future use.

Based on the row of this table selected, either by the user in MGC mode, or automatically by the device in AGC mode, the gain control block updates the variable gain elements depicted by the dash lines. In the MGC mode, the user can control the gain control block using the API commands and DGPIOs.

Table 56 shows the first three and last three rows in a sample gain table.

Table 56. Sample Rows from the Default Rx Gain Table

Gain Table Index	Front-End Attenuator Control Word [7:0]	TIA Control	ADC Control	External Gain Control [1:0]	Phase Offset	Digital Gain/Attenuator Control Word [10:0]
187	248	0	0	0	0	-2
188	247	0	0	0	0	-17
189	250	0	0	0	0	–4
…	…	…	…	…	…	…
253	28	0	0	0	0	–2
254	14	0	0	0	0	–1
255	0	0	0	0	0	0

The gain table index is the reference for each unique combination of gain settings in the programmable gain table. The current possible range of the gain table is 187 to 255. The gain index region is user configurable. An API function adi_adrv9001_Rx_MinMaxGainIndex_Set() could be called by the user right after loading the gain table to load multiple gain table regions and switch between multiple gain table regions during runtime.

The 2 fields which are used in the default gain table are the Front-end Attenuator and the Digital Gain/Attenuator. The Front-end Attenuator is an 8-bit control word. The amount of attenuation applied depends on the value set in this column of the selected gain table index. The following equation provides an approximate relationship between the internal attenuator and the front-end attenuation value programmed in the gain table, N:

$$Attenuation\ (dB) = 20\log_{10}\left(\frac{256 - N}{256}\right)$$

It can be seen that index 255 denotes a 0dB Front-end attenuation and the step size between adjacent gain index is approximately 0.5dB. Note when the index is below 195, the actual step size become less accurate.

The Digital Gain/Attenuator column is used to apply gain or attenuation digitally. The 11-bit signed word defines the digital gain applied, which equals to the control word times 0.05 in dB. As shown in Table 56, for gain index 253, the digital gain can be calculated as -2*0.05 = -0.1dB.

2 Types of receiver gain tables are provided. One is for gain correction in which the digital gain is for correcting the small step size inaccuracy in the Front-end Attenuator. The other is for gain compensation which compensates the entire front-end attenuation. The

example shown in Table 56 stands for a receiver gain correction table. In the Receiver/Observation Receiver Signal Chain section of this User Guide, it mentions that receiver can also be used as ORx for signal observation. Note another gain table exists for ORx gain control. However, for ORx gain control, there is no AGC mechanism but only MGC. In addition, the digital gain is only for gain correction.

Besides these 2 fields, the External Gain Control sets the external gain such as the gain of an LNA. User configures the LNA gain by configuring the data structure adi_adrv9001_RxLnaConfig_t and then calling the API adi_adrv9001_Rx_ExternalLna_Configure(). Note user should not modify the default Rx gain table. Based on user configuration, ADRV9001 creates a new Rx gain table internally with extended gain indices to accommodate the additional LNA attenuation. In such a case, the minimum possible gain index becomes 137, which can provide a maximum total of 59dB attenuation including the maximum LNA attenuation of 29dB. Note with the extended gain table, the Front-end attenuation included is only 30dB to avoid gain step size inaccuracy from gain index 194 to 187.

As indicated in the default gain table, the External Gain Control uses a 2 bit control word via 2 AGPIO pins which could yield 4 different gain step size for each receive channel. The 4 step sizes are based on attenuation relative to the max gain of LNA and defined as the following:

step 0 (control word 0) = 0 dB

step 1 (control word 1) = -N dB

step 2 (control word 2) = -N - M dB (optional)

step 3 (control word 3) = -N - M - L dB (optional)

Gain steps N, M and L should be multiple integers of 0.5 dB steps. In addition, N+M+L should not exceed 29dB. Note N and M are optional. The gain table maintains a gain step of 0.5 dB between adjacent gain indices and it assumes that the LNA step sizes are accurate.

The new Rx gain table is created by first assuming the max LNA Gain (0dB) until the ADRV9001 front-end attenuator "runs out of" attenuation. Then new gain indices are produced by assuming LNA gain of -N dB. To achieve the desired total attenuation by maintaining the 0.5dB step size, the front end gain needs to be recalculated and set properly in the new rows. Once the front-end attenuator "runs out of" attenuation again with LNA gain of -N dB, the new gain indices are further produced by assuming LNA gain of -N-M dB if user configures LNA step 2. The same method applies to LNA step 3 if it is configured.

As an example, if user configures the LNA with step 1 of 10 dB attenuation only, the Rx gain table uses LNA with 0 dB attenuation for gain indices from 255 to 195 and set the External Gain control word to be 0. For gain entries below 195 it switches LNA to 10 dB of attenuation. Since gain index 195 has an attenuation of 30dB, to maintain 0.5dB step size, the next gain index 194 represents 30.5 dB of total attenuation with a 10 dB external LNA. Therefore the front end attenuation should be 30.5-10 =20.5dB, which should reuse the setting associated with the index 255 – 20.5*2 = 214. So the entries for new gain index 194 should be copied from index 214, plus the external control should indicate LNA is enabled with 10 dB attenuation step by using control word 1. Each lower gain entry is simply copied from the next lower gain from table entry 213, 212, until it reaches 195 which exhaust the maximum front end attenuation.

The generated new Rx gain table is shown in Table 57. It has new entries from 194 to 175 to accommodate the 10dB LNA attenuation and they are copied from 214 to 195 with external gain control set to be 1.

Table 57. New Rx Gain Table Created from the Default Rx Gain Correction Table

Gain Table Index	Front-End Attenuator Control Word [7:0]	TIA Control	ADC Control	External Gain Control [1:0]	Phase Offset	Digital Gain/Attenuator Control Word [10:0]
175	248 (copied from 195)	0	0	1	0	-2
…	…	…	…	…	…	…
193	233 (copied from 213)	0	0	1	0	-20
194	232 (copied from 214)	0	0	1	0	-17
195	248	0	0	0	0	-2
196	247	0	0	0	0	-17
197	247	0	0	0	0	–7
…	…	…	…	…	…	…
253	28	0	0	0	0	–2
254	14	0	0	0	0	–1
255	0	0	0	0	0	0

The above example uses Rx gain correction table. The similar algorithm applies to generate the new Rx gain compensation table with only one difference: the digital gain should also compensate the LNA attenuation besides the front end attenuation therefore based on the External Gain control word setting, the digital gain needs to be further adjusted.

Note LNA must be powered down for initial calibrations. Once it is configured or bypassed during radio on operation, it cannot be dynamically configured or bypassed.

To accurately apply the attenuations for different components in the data path, user needs to measure the external path delay associated with LNA and set the delay through API command adi_adrv9001_Rx_ExternalLna_DigitalGainDelay_Set(). More details about how to measure the external LNA delay will be provided in the future.

As mentioned, two AGPIO pins are used for each receiver to perform external gain control. Depending on the hardware register setting, the AGPIO pins for Receiver 1 and Receiver 2 can be selected from AGPIO[3:0], AGPIO[7:4] and AGPIO[11:8]. Table 58 shows an example of Receiver 1 and Receiver 2 external gain element control when AGPIO[0:3] is selected (Note it is also possible to use AGPIO[1:0] for Receiver 2 and AGPIO[3:2] for Receiver 1. Please refer to GPIO section in the User Guide for more information.).

Table 58. An Example of Analog GPIOs for External Gain Element Control

Receiver	AGPIO Pins to Control External Gain Element
Rx1	AGPIO[1:0]
Rx2	AGPIO[3:2]

These AGPIOs must be enabled as outputs and set for external gain functionality. The 2-bit value programmed is directly related to the status of these AGPIO pins. For example: if the external gain word of the Receiver 1 gain table is programmed to 3 in selected gain index, then AGPIO[0] and AGPIO[1] will be high if AGPIO[1:0] is used to control external gain element as the example show in Figure 149.

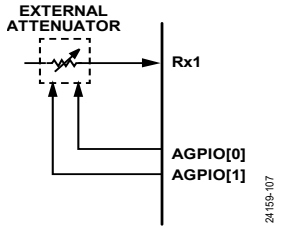


Figure 149. AGPIO Control of an External Gain Element to Rx1

GAIN CONTROL MODES

The gain control mode is selected through the API function adi_adrv9001_Rx_GainControl_Mode_Set() for a specified channel. Please refer to API doxygen document for more details.

adi_adrv9001_RxGainControlMode_e is an enum for selecting the gain mode. The possible options are shown in Table 59.

Table 59. Definition of adi_adrv9001_RxGainControlMode_e

ENUM	Gain Mode
ADI_ADRV9001_RX_GAIN_CONTROL_MODE_SPI	Manual Gain Control SPI Mode
ADI_ADRV9001_RX_GAIN_CONTROL_MODE_PIN	Manual Gain Control PIN Mode
ADI_ADRV9001_RX_GAIN_CONTROL_MODE_AUTO	Automatic Gain Control Mode

adi_common_ChannelNumber_e is an enum which indicates which Rx channel is used:

Table 60. Definition of adi_common_ChannelNumber_e

ENUM	Rx channel
ADI_CHANNEL_1	Rx1
ADI_CHANNEL_2	Rx2

Automatic Gain Control (AGC)

In Automatic Gain Control (AGC) mode, a built-in state machine automatically controls the gain based on user defined configuration. The AGC can be configured to one of two modes:

- Peak Detect mode, where only the peak detectors are used to make gain changes.

- Peak/Power Detect mode, where information from both the power detector and the peak detectors are used jointly to make gain changes.

Peak Detect Mode

In this mode, the peak detectors alone are used to inform the AGC to make gain changes. This section explains the basic premise of the operation, while more explicit details of configuring the peak detectors is covered in subsequent sections.

The APD and HB detector both have a high threshold and a low threshold, apdHighTresh, apdLowThresh, hbHighTresh and hbUnderRangeHighThresh, respectively. These levels are user programmable, as well as the number of times a threshold must be exceeded for an over range condition to be flagged.

The high thresholds are used as limits on the incoming signal level and will principally be set based on the maximum input of the ADC. When an over range condition occurs, the AGC will reduce the gain (gain attack). The low thresholds are used as lower limits on signal level. When an under range condition occurs, the AGC will increase the gain (gain recovery). The AGC stable state (where it will not adjust gain) occurs when neither an under range nor over range condition is occurring.

Each overrange/underrange condition has its own attack and recovery gain step as shown in Table 61.

Table 61. Peak Detector Gain Steps

Over Range/Under Range	Gain Step
apdHighThresh over range	Reduce gain by apdGainStepAttack
apdLowThresh under range	Increase gain by apdGainStepRecovery
hbHighThresh over range	Reduce gain by hbGainStepAttack
hbUnderRangeHighThresh under range	Increase gain by hbGainStepHighRecovery

An overrange condition occurs when the high thresholds have been exceeded a configurable number of times within a configurable period. An underrange condition occurs when the low thresholds have not been exceeded a configurable number of times within the same configurable period. These counters make the AGC less sensitive to occasional peaks in the input signal, ensuring that a single peak exceeding a threshold does not necessarily cause the AGC to react. Table 62 outlines the counter parameters for the individual overload/under range conditions.

Table 62. Peak Detector Counter Values

Over Range/Under Range	Counter
apdHighThresh over range	apdUpperThreshPeakExceededCnt
apdLowThresh under range	apdLowerThreshPeakExceededCnt
hbHighThresh over range	hbUpperThreshPeakExceededCnt
hbUnderRangeHighThresh under range	hbUnderRangeHighThreshExceededCnt

The AGC uses a gain update counter to time gain changes, with gain changes made when the counter expires. The counter value, and therefore the time spacing between possible gain changes, is user programmable through the gainUpdateCounter parameter. The user specifies the period, in AGC clock cycles, that gain changes can be made. Typically, this might be set to frame or sub-frame boundary periods.

Once the gain update counter expires, all the peak threshold counters are reset. The gain update period is therefore a decision period. The overload thresholds and counters are therefore set based on the number of overloads considered acceptable for the application within the gain update period.

Figure 150 shows an example of the AGC response to a signal versus the APD or HB peak detector threshold levels. APD and HB detector works in the same fashion in this mode. For ease of explanation, only APD is mentioned in the following discussions. The green line is representative of the peaks of the signal. Initially the peaks of the signal are within the apdHighThresh and apdLowThresh. No gain changes are made. An interferer suddenly appears whose peaks now exceed apdHighThresh. On the next expiry of the gain update counter (assuming a sufficient number of peaks occurred to exceed the counter), the AGC decrements the gain index (reduces the gain) by apdGainStepAttack. This isn't sufficient to get the signal peaks within the threshold levels, and hence the gain is decremented once more, with the peaks now between the two thresholds. The gain is stable in this current gain level until the interferer is removed, and the peaks of the desired signal are now below the apdLowThresh which results in an under range condition. Hence the AGC increases gain by the apdGainStepRecovery at the next expiry of the gain update counter, continuing to do so until the peaks of the signal are within the two thresholds once more.

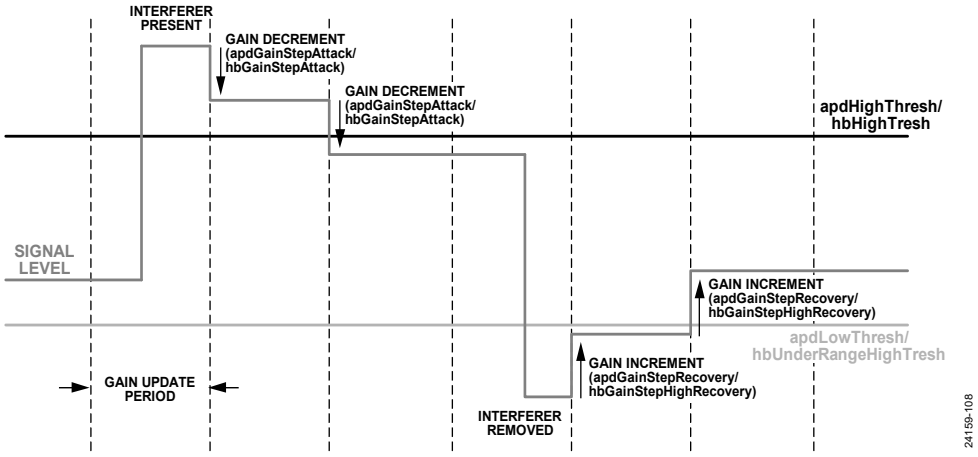


Figure 150. APD/HB Thresholds and Gain Changes Associated with Underrange and Overrange Conditions

It is possible to enable a fast attack mode whereby the AGC is instructed to reduce gain immediately when an over range condition occurs, instead of waiting until the next expiry of the gain update counter using changeGainIfThreshHigh. This parameter has independent controls for the APD and HB detectors. Values from 0 to 3 are valid as shown in Table 63.

Table 63. changeGainIfThreshHigh Settings

changeGainIfThreshHigh[1:0]	Gain Change following APD Over range	Gain Change following HB Overrange
00	After expiry of gainUpdateCounter	After expiry of gainUpdateCounter
01	After expiry of gainUpdateCounter	Immediately
10	Immediately	After expiry of gainUpdateCounter
11	Immediately	Immediately

Figure 151 shows how the AGC reacts when the changeGainIfThreshHigh is set for APD or HB detector. In this case when the interferer appears, the gain is updated as soon as the number of peaks exceed the peak counter. It does not wait for the next expiry of the gain update counter. Hence a number of gain changes can be made in quick succession providing a much faster attack than the default operation. The assumption here is that if the ADC is overloaded then it is best to decrease the gain quickly rather than wait for a suitable moment in the received signal in order to change the gain. This mode is referred as "fast attack" mode.

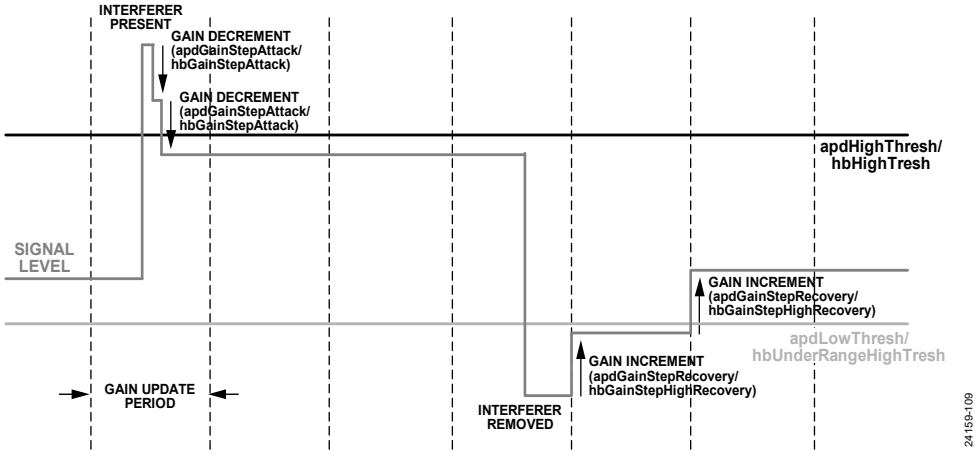


Figure 151. APD/HB Gain Changes with Fast Attack Enabled

Besides the "fast attack" mode, it is also possible to enable a "fast recovery" mode. This functionality is enabled with the enableFastRecoveryLoop parameter.

This "fast recovery" mode only works with the HB detector. The operation is illustrated in Figure 136. In this mode, the "fast recovery" is achieved by utilizing multiple low thresholds and step sizes as well as the update periods. When the signal level falls below hbUnderRangeLowThresh, the lowest threshold, the gain is incremented the most by hbGainStepLowRecovery following the expiry of a gain update period. Note that in the "fast recovery" mode the agcUnderRangeLowInterval is used instead of the gain update counter to set the gain update period (This also applies to the APD.). After sufficient gain increases to bring the signal level above hbUnderRangeLowThresh, the gain is incremented by hbGainStepMidRecovery after the expiry of agcUnderRangeMidInterval, which is

a multiple of agcUnderRangeLowInterval. Finally, when the signal level is increased above hbUnderRangeMidThresh, the gain is incremented by hbGainStepHighRecovery following the expiry of agcUnderRangeHighInterval, which is a multiple of agcUnderRangeMidInterval.

The multiple thresholds and interval parameters allow for faster gain recovery. Typically, agcUnderRangeHighInterval could be set to gain update counter as shown in Figure 152. Therefore, when the signal level is below the mid and low thresholds, the recovery could happen multiple times within a single gain update counter, which makes the recovery much faster. Note in "fast recovery" mode, gain recovery might not always happen at the expiry of the gain update counter, which is different from the mode without "fast recovery". If the gain update counter is set to align with the frame or subframe boundary, it is possible that a fast recovery could happen in the middle of a frame or subframe. Therefore, it is recommended to not use "fast recovery" mode when there is a stringent requirement for keeping a constant gain for an entire frame or subframe.

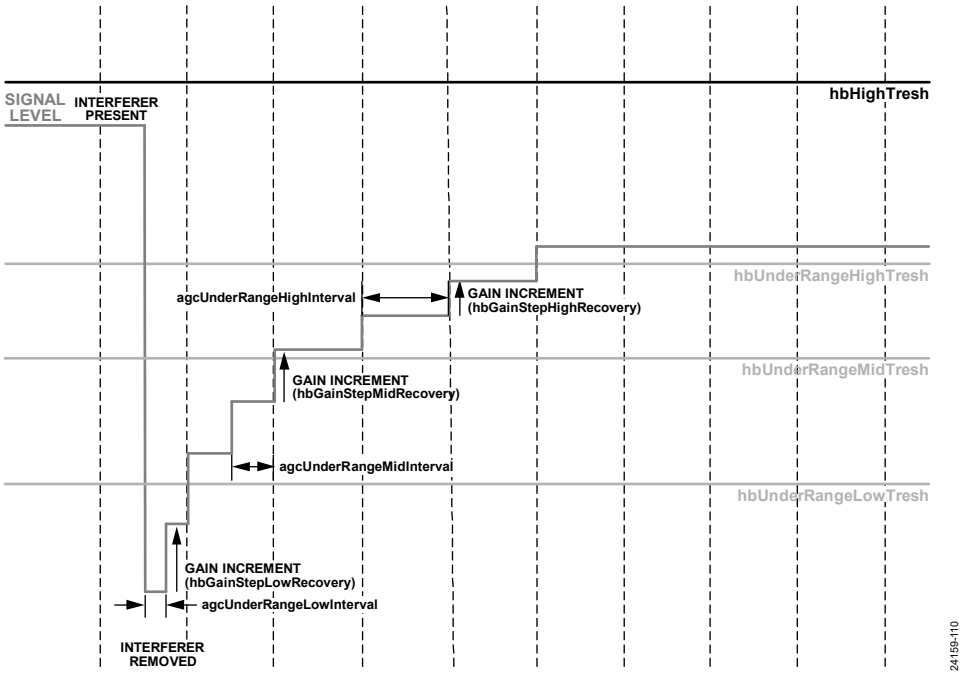


Figure 152. AGC operation with HB Detector in Fast Recovery Mode

It is highly recommended that the apdHighThresh and hbHighThresh are set to an equivalent dBFS value. Likewise, it is highly recommended that the apdLowThresh and the hbUnderRangeHighThresh are set to equivalent values. This equivalence will be approximate, as these thresholds have unique threshold settings and will not be exactly equal. This section discusses the relevant priorities between the detectors and how the AGC reacts when multiple threshold detectors have been exceeded. Table 64 shows the priorities between the detectors when multiple overranges occur.

Table 64. Priorities of Attack Gain Steps

apdHighThresh Over Range	hbHighThresh Over Range	Gain Change
No	No	No Gain Change
No	Yes	Gain Change by hbGainStepAttack
Yes	No	Gain Change by apdGainStepAttack
Yes	Yes	Gain Change by apdGainStepAttack

For recovery, the number of thresholds is dependent on whether fast recovery is enabled or not. Considering the fast recovery scenario, the priority of the thresholds is:

1. hbUnderRangeLowThresh Underrange Condition
2. hbUnderRangeMidThresh Underrange Condition
3. hbUnderRangeHighThresh Underrange Condition
4. apdLowThresh Underrange Condition

Upon one underrange condition, the AGC changes the gain by the corresponding gain step size of this condition. However, if multiple conditions occur simultaneously, then the AGC prioritizes based on the priorities indicated; that is, if hbUnderRangeLowThresh is reporting an under range condition then the AGC will adjust the gain by hbGainStepLowRecovery with two exceptions.

The apdLowThresh has priority in terms of preventing recovery. If apdLowThresh reports an over range condition (sufficient signal peaks have exceeded its threshold in a gain update counter period), then no further recovery is allowed. apdLowThresh and hbUnderRangeHighThresh should be configured to be as close to the same value of dBFS, but assuming some small difference between the thresholds, then as soon as apdLowThresh is exceeded, recovery will no longer occur. The reverse is not true, hbUnderRangeHighThresh will not prevent the gain recovery towards the apdLowThresh. Given the strong recommendation that apdLowThresh and hbUnderRangeHighThresh being set equally, then a condition whereby apdLowThresh was at a lower dBFS level to hbUnderRangeLowThresh or hbUnderRangeMidThresh should not occur.

Another exception is if the recovery step size for a detector is set to zero. If so, the AGC makes the gain change of the highest priority detector with a non-zero recovery step. Figure 153 provides a flow diagram of the decisions of the AGC when recovering the gain in peak detect mode.

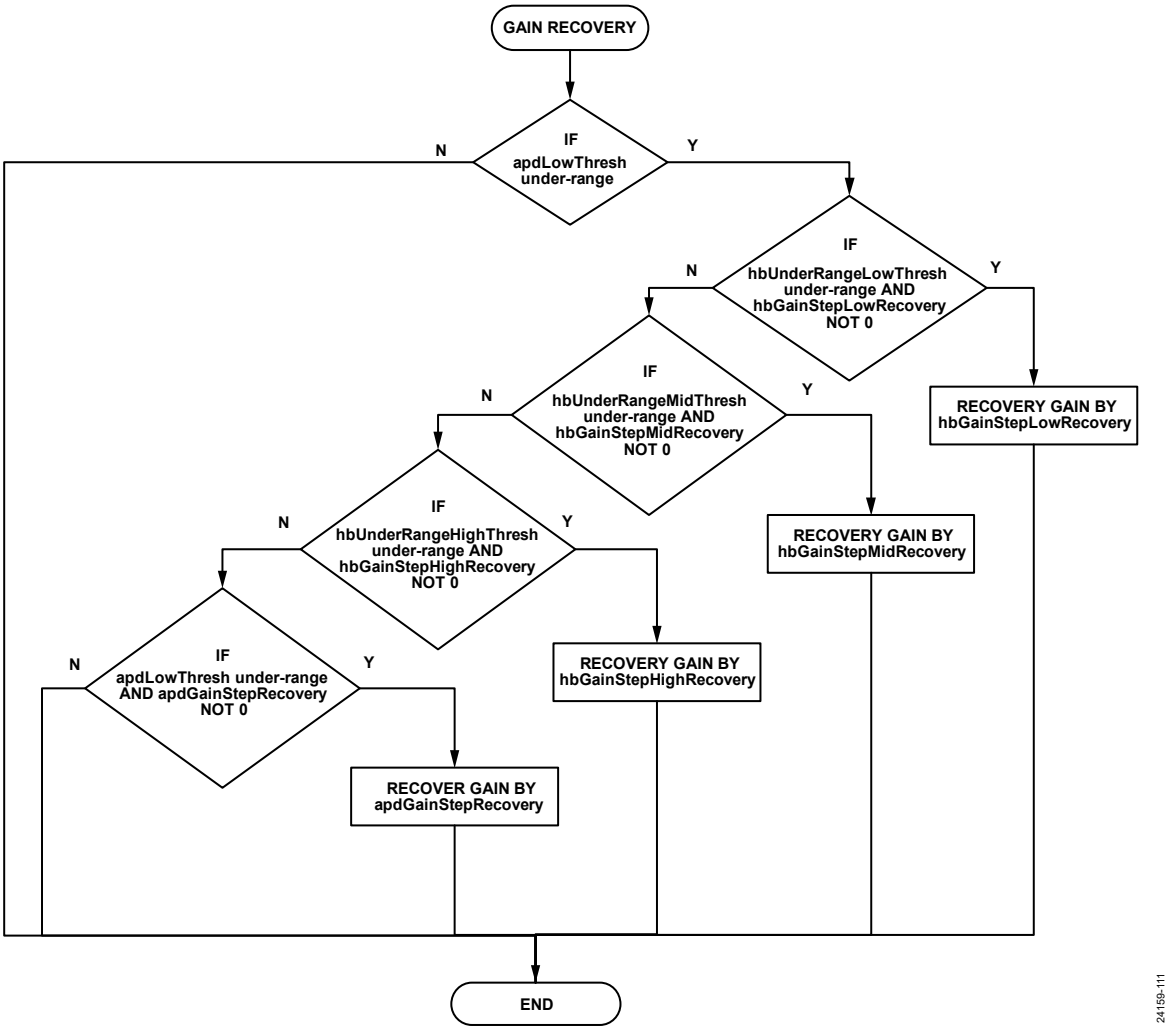


Figure 153. Flow Diagram for AGC Recovery in Peak Detect AGC Mode

Peak/Power Detect Mode

In this mode, the peak and power detect work jointly to control the gain of the receiver chain. In the event of an over-range condition, then both the peak and the power detect can instantiate a gain decrement. In the event of an under-range, only the power detect can increment the gain. The power detector will change gain solely at the expiry of the gain update counter. As previously mentioned, the peak detect can be set in one of two modes (depending on the setting of gainChangeIfThreshHigh) whereby the AGC: 1) waits for the gain update counter to expire before initiating a gain change; or 2) immediately updates the gain as soon as the overrange condition occurs (see Figure 150 and Figure 151). Therefore, in the peak/power detect mode, if the gain attack is instantiated by peak detectors, it is possible to perform fast attack.

The power detect provides the RMS power measurement of the receiver data at the output of HB Filtering block. In power detect mode, the AGC compares the measured signal level to programmable thresholds which provide a 2nd order control loop, whereby gain can be changed by larger amounts when the signal level is farther from the target level while make smaller gain changes when the signal is closer to the target level. This could allow the gain change faster when the level is farther away from the targeted range.

Figure 154 shows the operation of the AGC when using the power detect. Considering the power detect in isolation from the peak detectors, the AGC will not modify the gain when the signal level is between overRangeLowPowerThresh and underRangeHighPowerThresh. This range is the target range for the power measurement. The associated thresholds are also called inner thresholds.

When the signal level goes below underRangeLowPowerThresh, the AGC waits for the next gain update counter expiry and then increments the gain by underRangeLowPowerGainStepRecovery. When the signal level is greater than underRangeLowPowerThresh but below underRangeHighPowerThresh, the AGC will increment the gain by underRangeHighPowerGainStepRecovery. Likewise, when the signal level goes above overRangeHighPowerThresh, the AGC decreases the gain by overRangeHighPowerGainStepAttack, and when the signal level is between overRangeHighPowerThresh and overRangeLowPowerThresh, the AGC will decrease the gain by overRangeLowPowerGainStepAttack. underRangeLowPowerThresh and overRangeHighPowerThresh are also called outer thresholds.

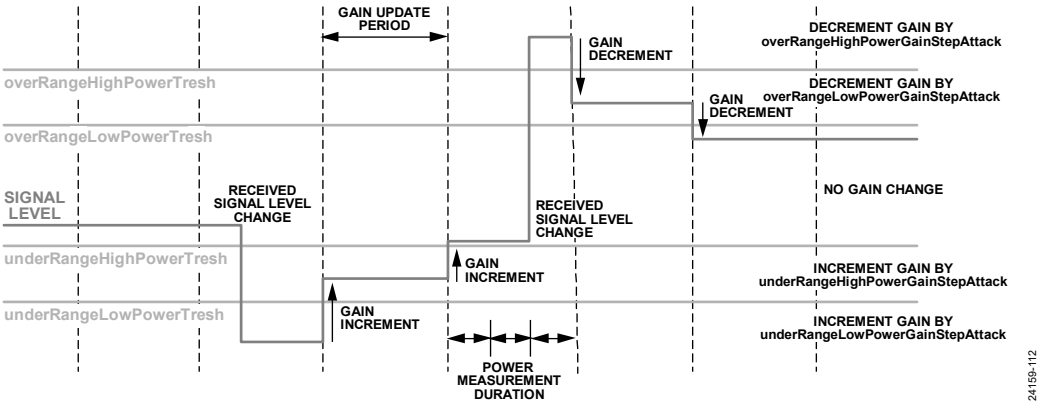


Figure 154. PMD Thresholds and Gain Changes for Under-range and Over-range Conditions

It is possible for the AGC to get contrasting requests from the power and peak detectors. An example would be an interferer that is visible to the analog peak detector but is significantly attenuated at the power detector. In this case the APD could be requesting a gain decrement, while the power detector could be requesting a gain increment. The AGC has the following priority scheme in peak/power detect mode:

1. APD Overrange
2. HB Overrange
3. APD lower level peak exceeded
4. HB lower level peak exceeded
5. Power Measurement

In this example, the gain would be decremented because the APD over-range has a higher priority than the power measurement. However, APD and HB lower level overloads act differently in peak detect and peak/power detect mode. In peak detect mode, the lower level thresholds for these detectors are used to indicate an under-range condition which caused the AGC to increase the gain. In peak/power detect mode, these detectors are not used for gain recovery, but used to control gain recovery by setting the API parameter, lowThreshPreventGainInc. If this parameter is set, and if the signal level is exceeding a lower level threshold, the AGC is prevented from increasing the gain regardless of the power measurement.

When a signal has higher than expected Peak to Average Power Ratio (PAR), the power detector could indicate a gain increase while the peak detector low threshold could still be exceeded. In such a case, gain increase will be prevented to avoid an overloading possibility. In addition, this could prevent an oscillation condition that could otherwise occur to an interferer visible to APD but filtered before the power detect. In such a case, the peak detect could cause the AGC to decrease gain. It would do this until the interferer is no longer exceeding the defined threshold. At this point, the power detect could request an increase in gain and would do so until the detector's peak threshold is exceeded. This might cause an oscillation condition. By using these lower level thresholds of peak detect, the AGC is prevented from increasing gain as the signal level approaches an overload condition, providing a stable gain level for the receiver chain under such a condition.

Peak Detect and Peak/Power Detect Mode Comparison

Among the two detect modes, peak detect offers the quickest response time to overload signals by employing "fast attack" mode. It allows the AGC to respond within hundreds of nanoseconds in overload scenarios. In addition, the peak detect also provides "fast recovery" option to increase the gain of the desired signal quickly when an interferer disappears. It can also avoid the possible gain index oscillation issue of peak/power detect when the signal has higher than expected PAR.

With power detect, the gain change can only happen at the expiration of the gain update counter, which is typically set at the order of hundreds of microseconds or milliseconds. However, the power detector is usually more stable and will not likely cause frequent gain changes. In addition, it can provide a tighter control of signal level by utilizing a set of inner and outer thresholds comparing with peak detect.

It is highly recommended to use peak detect especially when fast gain control is desired.

Manual Gain Control (MGC)

The gain control block applies the settings from the selected gain index in the gain table. In MGC mode, the baseband processor is in control of selecting the gain index. There are two options: 1) API commands (ADI_ADRV9001_RX_GAIN_CONTROL_MODE_SPI); and 2) pin control (ADI_ADRV9001_RX_GAIN_CONTROL_MODE_PIN). By default, if MGC is chosen the part is configured for API commands.

In API command mode, the user selects a gain index in the gain table through the API function adi_adrv9001_Rx_Gain_Set(). The gain index selected for a channel can be read back through the API function adi_adrv9001_Rx_Gain_Get().

The pin control MGC mode is useful when real time control of gain is required. API command adi_adrv9001_Rx_GainControl_PinMode_Configure() can be used to properly configure this mode. In this mode, out of 16 digital DGPIO pins, 2 pins per receiver are used, one increasing and the other decreasing the gain table index. The user specifies both the max and min gain index as well as the increment and decrement step size (in the range of 0 to 7 gain table indices). A pulse is applied to the relevant DGPIO pin to trigger an increment or decrement in gain as shown in Figure 155. This pulse must be held high for at least 2 AGC clock cycles for a reliable detection of the rising edge to trigger the gain change (see AGC clock section for details). The configuration data structure for this mode is defined as the following:

```
typedef struct adi_adrv9001_RxGainControlPinCfg
{
    uint8_t minGainIndex;  //Minimum gain index. Must be >= gainTableMinGainIndex and <
maxGainIndex
    uint8_t maxGainIndex;  //Maximum gain index. Must be > minGainIndex and <=
gainTableMaxGainIndex
    uint8_t incrementStepSize;  //Number of indices to increase gain on rising edge on
incrementPin (Range: 0 to 7)
    uint8_t decrementStepSize;  //Number of indices to decrease gain on rising edge on
decrementPin (Range: 0 to 7)
     adi_adrv9001_GpioPin_e incrementPin;// A rising edge on this pin will increment gain by
incrementStepSize.
    adi_adrv9001_GpioPin_e decrementPin; //A rising edge on this pin will decrement gain by
decrementStepSize.
} adi_adrv9001_RxGainControlPinCfg_t
```

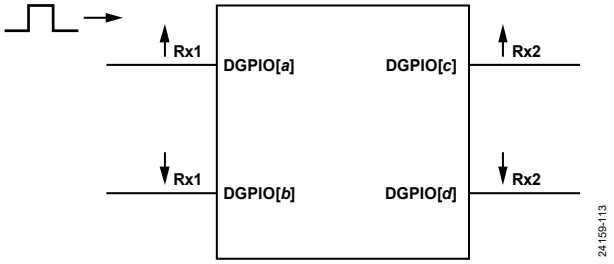


Figure 155. MGC PIN Mode: DGPIO (a to d) Represent Any of DGPIO[0:15]

In the MGC mode, to properly control the gain, user could optionally retrieve the status of peak detectors and power detector in the device through a set of DGPIO pins (this could also be done in the AGC mode for observation). In order to make sure that the status information from the signal detectors are meaningful, it is important that the user should first enable and configure the signal detectors

properly, which can be done through API command. The feedback information can be configured into 2 modes, the peak detect mode and peak and power detect mode. In peak detect mode, the over-range and under-range conditions of both APD and HB detectors will be provided through DGPIO pins to user. In peak and power detect mode, over-range and under-range conditions of power detector, over-range and under-range condition of APD will be provided through DGPIO pins to user.

Table 65 describes feedback configuration and the bitfield definition and position. For example, when it is configured in peak mode, the user could connect to a set of DGPIO pins to retrieve all the APD and HB detector status. Note the DGPIO pins could be selected from Pin 0 to Pin 15 and 2 consecutive DGPIO pins should always be configured as a pair to retrieve 2 consecutive bitfields (Bit 0 and Bit 1 or Bit 2 and Bit 3 in both modes). DGPIO pin selection is defined by the following enum type:

```
typedef enum adi_adrv9001_GpioPinCrumbSel
{
    ADI_ADRV9001_GPIO_PIN_CRUMB_UNASSIGNED,
    ADI_ADRV9001_GPIO_PIN_CRUMB_01_00,
    ADI_ADRV9001_GPIO_PIN_CRUMB_03_02,
    ADI_ADRV9001_GPIO_PIN_CRUMB_05_04,
    ADI_ADRV9001_GPIO_PIN_CRUMB_07_06,
    ADI_ADRV9001_GPIO_PIN_CRUMB_09_08,
    ADI_ADRV9001_GPIO_PIN_CRUMB_11_10,
    ADI_ADRV9001_GPIO_PIN_CRUMB_13_12,
    ADI_ADRV9001_GPIO_PIN_CRUMB_15_14,
} adi_adrv9001_GpioPinCrumbSel_e
```

In both peak mode and peak and power mode, a pair of bits (Bit 0 and Bit 1 or Bit 2 and Bit 3) can choose any one pair of GPIO pins defined in "adi_adrv9001_GpioPinCrumbSel_e". If "ADI_ADRV9001_GPIO_PIN_CRUMB_UNASSIGNED" is selected, it means no GPIO pins assigned so the corresponding bitfields cannot be observed by user. The DGPIO pins can be associated with either one of the receivers.

Table 65. DGPIO Configuration for Retrieving Signal Detector Information

Mode	Bit Field Definition	Feedback Mask Bit Position
Peak Mode	hb_low_threshold_counter_exceeded (low threshold has been exceeded counter times, no under load condition)	0
	apd_low_threshold_counter_exceeded (low threshold has been exceeded counter times, no under load condition)	1
	hb_high_threshold_counter_exceeded (high threshold has been exceeded counter times, over load condition)	2
	apd_high_threshold_counter_exceeded (high threshold has been exceeded counter times, over load condition)	3
Peak and Power Mode	power_inner_low_threshold_exceeded (inner low threshold exceeded, no under load condition)	0
	power_inner_high_threshold_exceeded (inner high threshold exceeded, over load condition)	1
	apd_low_threshold_counter_exceeded (low threshold has been exceeded counter times, no under load condition)	2
	apd_high_threshold_counter_exceeded (high threshold has been exceeded counter times, over load condition)	3

GAIN CONTROL DETECTORS

In this section, three gain control detectors will be discussed in more details.

Analog Peak Detector (APD)

The analog peak detector is located in the analog domain following the TIA filter and prior to the ADC input. It functions by comparing the signal level to programmable thresholds. When a threshold has been exceeded a programmable number of times in a gain update period, then the detector flags that the threshold has been overloaded.

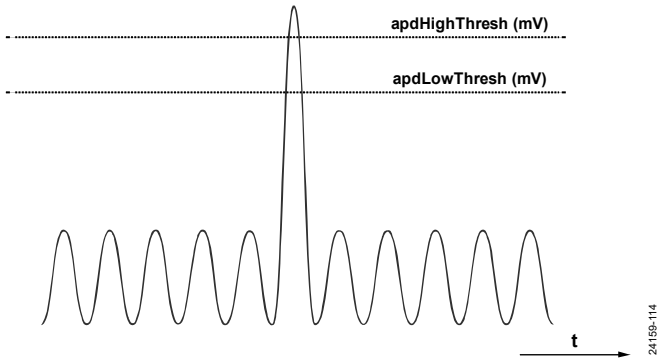


Figure 156. Analog Peak Detector Thresholds

There are two APD thresholds as shown in Figure 156. These thresholds are contained in the agcPeak API structure, apdHighThresh and apdLowThresh, respectively. The thresholds are typically considered relative to full scale voltage of the ADC, which is 850mVpeak. The mV setting of the APD thresholds can be determined using the following equations:

$$apdHighThresh\ (mV) = \frac{apdHighThresh\ \times 15\ (mV)}{0.91}$$

$$apdLowThresh\ (mV) = \frac{apdLowThresh\ \times 15\ (mV)}{0.91}$$

To determine the setting of the APD thresholds in terms of the closest possible setting in terms of dBFS of the ADC assuming apdHighdBFS and apdLowdBFS for apdHighThresh and apdLowThresh, respectively, the following equations can be used:

$$apdHighThresh = round\left(\frac{\left(10^{\left(\frac{apdHighdBFS}{20}\right)} \times 850\right) \times 0.91}{15}\right)$$

$$apdLowThresh = round\left(\frac{\left(10^{\left(\frac{apdLowdBFS}{20}\right)} \times 850\right) \times 0.91}{15}\right)$$

The APD threshold must be exceeded a programmable number of times within a gain update counter period before an over range condition occurs. Both the upper and lower thresholds have a programmable counter in the API structure, as indicated in Table 66.

Table 66. APD Programmable Threshold Counters

Threshold	Counter
Upper Threshold (apdHighThresh)	apdUpperThreshPeakExceededCnt
Lower Threshold (apdLowThresh)	apdLowerThreshPeakExceededCnt

As described in the earlier section on AGC control, the APD is used for both gain attack and gain recovery in peak detect mode. In peak/power detect mode, the APD could be used for gain attack, and is used to prevent overloading during gain recovery. For more details, refer to the relevant sections.

In AGC mode, the APD has programmable gain attack and gain recovery step sizes as shown in Table 67.

Table 67. APD Attack and Recovery Step Sizes

Gain Change	Step Size
Gain Attack	apdGainStepAttack
Gain Recovery	apdGainStepRecovery

Step size refers to the number of indices of the gain table the gain is changed. As explained earlier, the gain table is programmed with the largest gain in the Max Gain Index (typically index 255), with ever decreasing gain for decreasing gain index. Thus, if the APD gain attack step size was programmed to 6, then this means that the gain index is reduced by 6 when the apdHighThresh has been exceeded more than apdUpperThreshPeakExceededCnt times. For example: if the gain index had been 255 before this over range condition, then the gain index would be reduced to 249. The amount of gain reduction this equates to is dependent on the gain table in use. The default table has 0.5dB steps which in this example would equate to a 3dB gain reduction upon an APD over range condition.

The APD is held in reset for a configurable amount of time following a gain change to ensure that the receiver path is settled at the new gain setting.

Half-Band Peak Detector

The HB peak detector is located in the digital domain at the output of the HB Filtering block. It can therefore also be referred to as the Decimated Data Overload Detector because it works on decimated data. Like the APD detector, it functions by comparing the signal level to programmable thresholds. It monitors the signal level by observing individual samples (I2 + Q2 or peak I/peak Q) over a period of time and compares these samples to the threshold. If a sufficient number of samples exceed the threshold in the period of time, then the threshold is noted as exceeded by the detector. The duration of the HB measurement is controlled by hbOverloadDurationCnt, while the number of samples that should exceed the threshold in that period is controlled by hbOverloadThreshCnt.

Once the required number of samples exceed the threshold in the duration required, then the detector records that the threshold was exceeded. Like the APD detector, the HB detector requires a programmable number of times for the threshold to be exceeded in a gain update period before it will flag an over-range condition.

Figure 157 shows the two-level approach which is different from APD. It shows the gain update counter period, with the time being broken into subsets of time based on the setting of hbOverloadDurationCount. Each of these periods of time is considered separately, and hbOverloadThreshCount individual samples must exceed the threshold within hbOverloadDurationCount for an overload to be declared. These individual samples greater than the threshold are shown in grey. Two examples are shown, one where the number of samples exceeding the threshold is sufficient for the HB peak detector to declare an overload, and a second example where the number of samples exceeding the threshold is not sufficient to declare an overload. The number of overloads is counted, and if the number of overloads of the hbHighThresh exceed hbUpperThreshPeakExceededCount in a gain update counter period, then an over-range condition is called. Likewise, if the number of overloads of the hbUnderRangeHighThresh does not exceed hbUnderRangeHigh-ThreshExceededCount, then an under-range condition is called. Note that if hbOverloadDurationCount is set to equal to the time duration of 1 sample and hbOverloadThreshCount is set to 1, the HB two-level approach becomes similar to the APD algorithm.

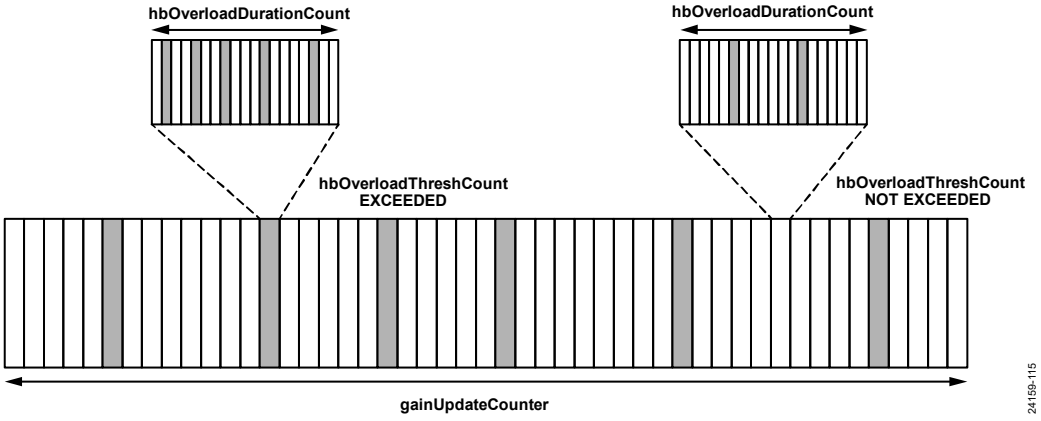


Figure 157. HB Detector, Two-Level Approach for an Overload Condition

The HB detector has a number of programmable thresholds. Some of these thresholds are only used in the fast recovery mode of the peak detect AGC configuration, as summarized in Table 68.

Table 68. HB Overload Thresholds

HB Threshold	Usage
hbHighThresh	Used for gain attack in both peak and peak/power detect AGC modes.
hbUnderRangeHighThresh	Used for gain recovery in peak detect AGC mode. In peak/power detect AGC mode it is used to prevent overloads during gain recovery.
hbUnderRangeMidThresh	Used only when the fast recovery option of the peak detect AGC mode is being used.
hbUnderRangeLowThresh	Used only when the fast recovery option of the peak detect AGC mode is being used.

For more details of how these thresholds are used by the AGC, refer to the relevant sections in this document.

The thresholds are related to an ADC dBFS value using the following equations:

$$hbHighThresh = 32768 \times 10^{\left(\frac{hbHigh\ dBFS}{20}\right)} - 1¶$$

$$hbUnderRangeHighThresh = 32768 \times 10^{\left(\frac{hbUnderRangeHigh\ dBFS}{20}\right)} - 1¶$$

$$hbUnderRangeMidThresh = 32768 \times 10^{\left(\frac{hbUnderRangeMid\ dBFS}{20}\right)} - 1¶$$

$$hbUnderRangeLowThresh = 32768 \times 10^{\left(\frac{hbUnderRangeLow\ dBFS}{20}\right)} - 1¶$$

Each threshold has an associated counter such that an over-range condition is not flagged until the threshold has been exceeded this amount of times in a gain update period.

Table 69. Counters for HB Overrange and Underrange Conditions

HB Threshold	Counter
hbHighThresh	hbUpperThreshPeakExceededCount
hbUnderRangeHighThresh	hbUnderRangeHighThreshExceededCount
hbUnderRangeMidThresh	hbUnderRangeMidThreshExceededCount
hbUnderRangeLowThresh	hbUnderRangeLowThreshExceededCount

In AGC mode, the HB peak detector has programmable gain attack and gain recovery step sizes.

Table 70. HB Attack and Recovery Step Sizes

Gain Change	Step Size
Gain Attack	hbGainStepAttack
Gain Recovery (hbUnderRangeHighThresh)	hbGainStepHighRecovery
Gain Recovery (hbUnderRangeMidThresh)	hbGainStepMidRecovery
Gain Recovery (hbUnderRangeLowThresh)	hbGainStepLowRecovery

The HB peak detector is held in reset for a configurable amount of time following a gain change to ensure that the receiver path is settled at the new gain setting.

Power Detector

The power measurement block measures the RMS power of the incoming signal at the output of HB Filtering block. The number of samples that are used in the power measurement calculation is configurable using the powerMeasurementDuration API parameter:

$$Power\ Meas\ Duration\ (Rx\ Sample\ Clocks) = 8 \times 2^{powerMeasurementDuration}$$

where *Rx Sample Clocks* is the number of clocks at the power measurement location.

It is important that this duration not exceed the gain update counter. The gain update counter resets the power measurement block and therefore a valid power measurement must be available before this event. In the case of multiple power measurements occurring in a gain update period, the AGC will use the last fully completed power measurement, any partial measurements being discarded.

Note currently, only inner thresholds of power detector are utilized. The inner thresholds defined in API are related to an ADC dBFS value using the following equations:

underRangeHighPowerThresh = -(underRangeHighPowerThresh _dBFS + 6)

overRangeLowPowerThresh = -(overRangeLowPowerThresh_dBFS + 6)

The power measurement block has a dynamic range of 60dB. Signals lower than −60 dBFS may not be measured accurately. The power masurement could be read through the API function adi_adrv9001_Rx_DecimatedPower_Get().

Detector Overload and QEC

If the device is operating in AGC mode, when an overload is detected, the Rx QEC calibration will be frozen. AGC will be able to adjust the Rx Gain Index to attempt to bring the received signal to a level that is under the overload threshold so that input signal no longer triggers the overload detector. Then Rx QEC calibration will resume.

If the device is operating in MGC mode, when an overload is detected, the Rx QEC calibration will be frozen as well. However since the device is operating in MGC mode, if input signal remains a high level above the overload threshold, then Rx QEC will remain frozen and will not recover until the input signal is below the overload threshold.

AGC CLOCK AND GAIN BLOCK TIMING

The AGC clock is the clock which drives the AGC state machine. In ADRV9001 device, the default AGC clock (to support a set of standard sample rates) is at 184.32 MHz. When arbitrary sample rate is adopted in Rx, the AGC clock could vary.

The AGC state machine contains 3 states: Gain Update Counter, followed by the Slow Loop Settling (SLS) Delay, and 5 AGC clock cycles delay. The total time between gain updates (gain update period) is a combination of slowLoopSettlingDelay and 5 AGC clock cycles. (Note the first slowLoopSettlingDelay in grey is a part of Gain Update Counter.)

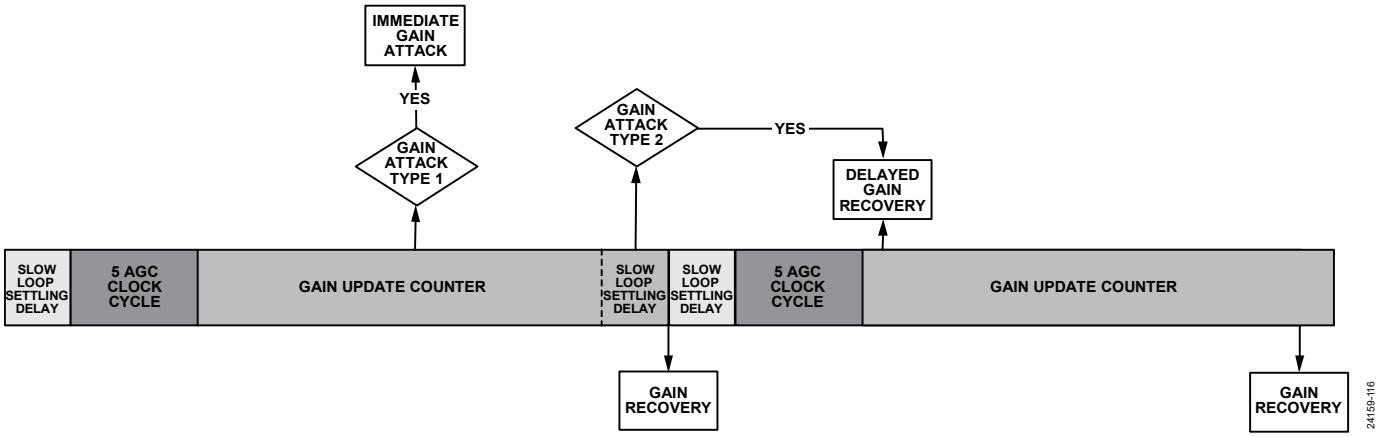


Figure 158. Delayed Gain Attack for Nondelayed Gain Recovery

Figure 158 outlines the operation of the AGC state machine. The diagram outlines possible gain change scenarios rather than a practical example of AGC operation. The possible gain change scenarios are described below:

- AGC Gain Attack within gain update counter, but more than an SLS delay before the gain update counter expiry – Because slow loop settling (SLS) is typically several orders of magnitude smaller than gain update counter, this is the most common gain decrement scenario. This type of AGC Gain Attack is named as Gain Attack Type 1 as shown in Figure 158.
- AGC Gain Attack within gain update counter, but within a SLS delay before the gain update counter expiry – This is a special case, which will rarely occur in applications per the reasoning in 1). This type of AGC Gain Attack is named as Gain Attack Type 2 as shown in Figure 158.
- AGC Gain Recovery at the end of the gain update counter – Note that when fast recovery is enabled, the gain update counter is substituted with the low under range interval. A gain attack may occur within the gain update counter when fast attack is enabled. A gain recovery event may only occur at the end of gain update counter (or low under range interval in “fast recovery” mode) as previously discussed. This is mainly for aligning the gain recovery (for desired signal) with the frame or subframe boundary. After a gain attack, a gain change counter with a value equal to the SLS delay is started. No further gain attacks are allowed while this counter is running. This allows the minimum time to be set between gain changes.

However the gain change counter also prevents the AGC from moving from the gain update counter state to the slow loop settling delay state since it must wait until the expiry of the SLS delay. Therefore if a gain attack occurred very close to end of the gain update counter state, the gain change counter would delay the start of the SLS state and shift the gain recovery event as shown in Figure 158. Whereas in Figure 159, gain recovery event is always aligned with the vertical dash lines.

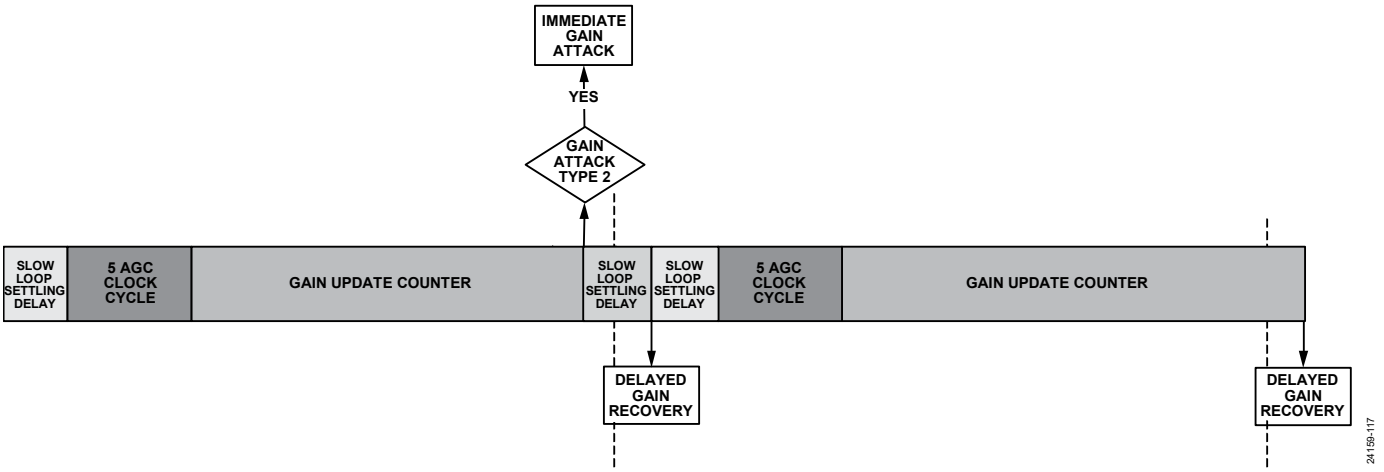


Figure 159. Immediate Gain Attack Causing Delayed Gain Recovery

To prevent this happening and maintain a perfectly periodic gain recovery event, gain attacks are prevented from happening towards the end of gain update counter state as shown in Figure 159. If a gain attack would happen in this period, it is delayed until the start of the next gain update counter state. This can cause gain attacks to be held off for up to 2× SLS + 5 delay, therefore it is recommended to keep SLS delay as short as possible to minimize the gain attack delay. Note that it is possible to disable this blocking feature, thus allowing gain attacks to occur anywhere within the gain update counter state, however the periodicity of the gain recovery event is no longer guaranteed as gain attacks towards the end of the gain update counter state will cause the gain recovery event to be delayed as shown in Figure 159.

At the expiry of the gain update counter (or low under range interval in "fast recovery" mode), all measurement blocks are reset and any peak detector counts will be reset back to zero. When the Rx is enabled, the counter begins. This might mean that its expiry is at an arbitrary phase to the slot boundaries of the signal. The expiry of the counter can be aligned to the slot boundaries by setting the parameter enableSyncPulseForGainCounter. While this bit is set, the AGC will monitor a DGPIO pin to find a synchronization pulse. This pulse will cause the reset of the counter at this point of time, hence if the user supplies a DGPIO pulse time aligned to these slot boundaries then the expiry of the counter will be aligned to slot boundaries. Any of DGPIO pin 0-15 can be used for this purpose.

For example, considering 100us gain update period and a 184.32 MHz AGC clock, a total of 18,432 AGC clocks will exist in the gain update period:

Gain Update Period (*AGC Clocks*) = 184.32 MHz × 100 µs = 18,432

As noted, the full gain update period is the sum of the gainUpdateCounter, the slowLoopSettlingDelay and a number of AGC clock cycles. If the slowLoopSettlingDelay is set to 4, the gain update counter must be set to 18,423 from the following calculation:

Gain Update Period (*AGC Clocks*) = *gainUpdateCounter* + *slowLoopSettlingDelay* +5

Gain Update Period (*AGC Clocks*) = 18,423 + 4 + 5 = 18,432

When Rx is enabled, the AGC can be kept inactive for a number of AGC clock cycles by using attackDelay_us. This means the user can specify one delay for AGC reaction when entering receive mode, and another for after a gain change occurs (slowLoopSettlingDelay).

ANALOG GAIN CONTROL API PROGRAMMING

As mentioned previously, the Rx gain control mode could be configured as MGC or AGC mode. In both modes, the API function adi_adrv9001_Rx_GainControl_Configure() is used to configure the gain control blocks, such as the peak detectors and the power detector for a specific channel. Those detectors are used not only in the AGC mode but also could be used in MGC mode to feed user important information. This API function also configures the DGPIO pins for retrieving the signal detectors information. Note although signal detectors information is critical for MGC mode, it can also be obtained in AGC mode for observation and debugging purpose.

The composition of the gain control configuration structure adi_adrv9001_GainControlCfg_t will be discussed in details in the next section. Once it is configured, the desired gain control mode can be enabled by using adi_adrv9001_Rx_GainControl_Mode_Set()API function.

If MGC mode is selected, as previously discussed, the gain could be manually controlled through API commands or DGPIO pins. In API command mode, the user selects a gain index in the gain table through the API function adi_adrv9001_Rx_Gain_Set(). The gain index selected for a channel can be read back through the API function adi_adrv9001_Rx_Gain_Get().

Figure 160 describes a high level flow chart of Rx gain control programming. Note the final step is to configure any GPIOs as necessary such as GPIO inputs which allow the AGC gain update counter to be synchronized to a slot boundary, or DGPIOs to directly control the gain index. Note the configure of the DGPIO pins for retrieving signal detectors information is included in the API command adi_adrv9001_Rx_GainControl_Configure(). The operation of these has been described earlier.

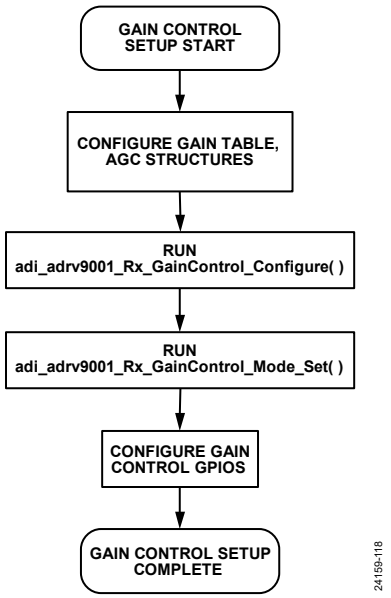


Figure 160. Gain Control Programming Flowchart

Gain Control Data Structures

Figure 161 shows the member structure of adi_adrv9001_GainControlCfg_t, and its substructures, adi_adrv9001_PeakDetector_t, adi_adrv9001_PowerDetector_t and adi_adrv9001_ExtLna_t. Each of the parameters are briefly explained in Table 71 to Table 74 the wider context of these parameter settings being outlined in the previous relevant sections.

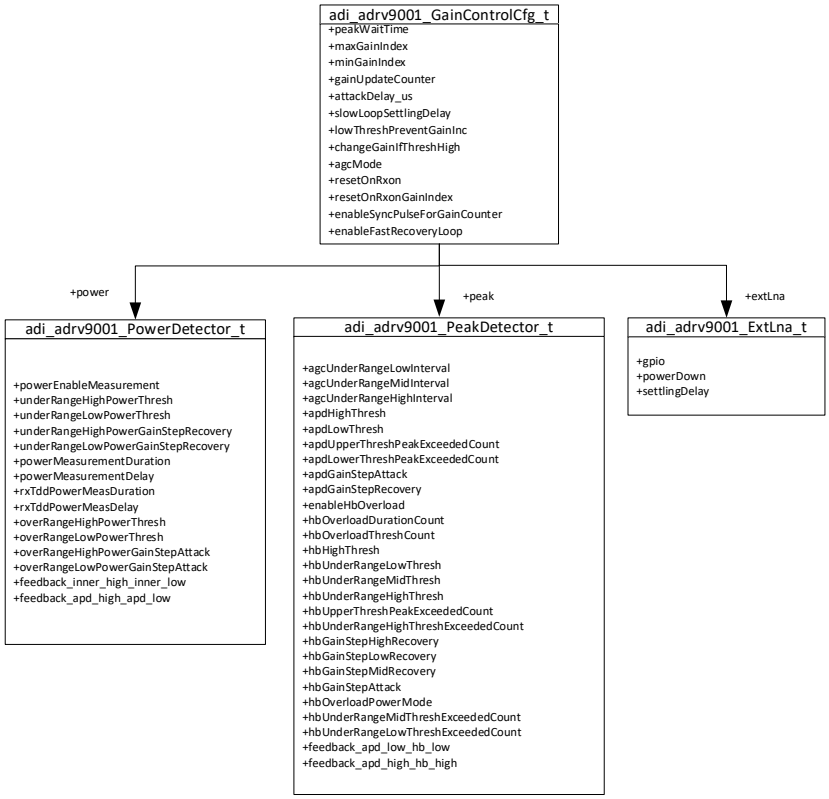


Figure 161. Member Listing of adi_adrv9001_GainControlCfg_t Data Structure

Table 71. adi_adrv9001_GainControlCfg_t Structure Definition

Parameter	Description	Min Value	Max Value	Default Value
peakWaitTime	Number of gain control clock cycles to wait before enabling peak detectors after a gain change.	0	31	4
maxGainIndex	Maximum gain index allowed. Must be greater than minGainIndex and be a valid gain index.	195	255	255
minGainIndex	Minimum gain index allowed. Must be less than maxGainIndex and be a valid gain index.	195	255	195
gainUpdateCounter	Is used as a decision period, with the detectors reset on this period. Gain changes in AGC mode can also be synchronized to this period (the expiry of this counter). The full period is a combination of the gainUpdateCounter and slowLoopSettlingDelay and a number of AGC cycles.	Depends on Overload Detector Settings	4194303 AGC_CLK Cycles	11520
attackDelay_us	The duration the AGC should be held in reset when the Rx path is enabled.	0	63	10
slowLoopSettlingDelay	Number of AGC clock cycles to wait after a gain change before the AGC will change gain again.	0	127	16
lowThreshPreventGainInc	Only relevant in Peak and Power Detect AGC operation. 1: If AGC is in Peak and Power Detect Mode, then gain increments requested by the power detector are prevented if there are sufficient peaks (APD/HB Low Threshold Exceeded Count) above the apdLowThresh or hbUnderRangeHighThresh. 0: apdLowThresh and hbUnderRangeHighThresh are don't cares for gain recovery.	0	1	0
changeGainIfThreshHigh	Applicable in both peak and peak and power detect modes. 0: Gain changes will wait for the expiry of the gain update counter if a high threshold count has been exceeded on either the APD or HB detector. 1: Gain changes will occur immediately when initiated by HB. Gain changes initiated by the APD will wait for the gain update to expire. 2: Gain changes will occur immediately when initiated by APD. Gain changes initiated by HB will wait for the gain update to expire. 3: Gain changes will occur immediately when initiated by APD or HB detectors.	0	3	3
agcMode	1: AGC in Peak AGC mode, power-based gain changes are disabled. 0: AGC in Peak and Power AGC mode where both Peak Detectors and Power Detectors are used.	0	1	1
resetOnRxon	1: AGC state machine is reset when Rx is disabled. The AGC gain setting will use the "resetOnRxonGainIndex" after resuming the operation. 0: AGC state machine maintains its state when Rx is disabled and the last AGC gain index will be used after resuming the operation.	0	1	0
resetOnRxonGainIndex	The AGC index to start with when "resetOnRxon" is set as 1.	195	255	255
enableSyncPulseForGainCounter	1: Allows synchronization of AGC Gain Update Counter to the time-slot boundary. GPIO setup required. 0: AGC Gain Update Counter free runs.	0	1	0
enableFastRecoveryLoop	1: Enables the fast recovery AGC functionality using the HB overload detector. Only applicable in Peak Detect Mode. 0: AGC fast recovery is not enabled.	0	1	0
power	Structure containing all the power detector settings.	N/A	N/A	N/A
peak	Structure containing all the peak detector settings.	N/A	N/A	N/A
extLna	Structure containing all external LNA settings	N/A	N/A	N/A

Table 72. adi_adrv9001_PowerDetector_t Structure Definition

Parameter	Description	Min Value	Max Value	Default Value
powerEnableMeasurement	1: Power Measurement block enabled. 0: Power Measurement block disabled.	0	1	1
underRangeHighPowerThresh	Threshold (negative sign assumed) which defines the lower boundary on the stable region of the power detect gain control mode.	0	127	10
underRangeLowPowerThresh	Offset (negative sign assumed) from underRangeHighPowerThresh which defines the outer boundary of the power based AGC convergence. Typically, recovery would be set to be larger steps than when the power measurement is less than this threshold.	0	15	4
underRangeHighPowerGainStepRecovery	The number of indices that the gain index pointer should be increased (gain increase) in the event of the power measurement being less than underRangeHighPowerThresh but greater than underRangeLowPowerThresh.	0	31	2
underRangeLowPowerGainStepRecovery	The number of indices that the gain index pointer should be increased (gain increase) in the event of the power measurement being less than underRangeLowPowerThresh.	0	31	4
powerMeasurementDuration	Number of IQ samples on which to perform the power measurement. The number of samples corresponding to the 4-bit word is 8 × 2^(pmdMeasDuration[3:0]). This value must be less than AGC Gain Update Counter.	0	31	10
powerMeasurementDelay	Measurement delay to detect power for specific slice of gain update counter	0	255 AGC clock cycles	2
rxTddPowerMeasDuration	Following an Rx Enable, the power measurement block can be requested to perform a power measurement for a specific period of a frame. This is applicable in TDD modes. This parameter sets the duration of this power measurement. A value of 0 causes the power measurement to run until the next gain update counter expiry.	0	65535 AGC clock cycles	0
rxTddPowerMeasDelay	Following an Rx Enable, the power measurement block can be requested to perform a power measurement for a specific period of a frame. This is applicable in TDD modes. This parameter sets the delay between the Rx Enable and the power measurement starting on Rx.	0	65535 AGC clock cycles	0
overRangeHighPowerThresh	Offset (positive sign assumed) from threshold overRangeLowPowerThresh which defines the outer boundary on the stable region of the power detect gain control mode. Typically attack would be set to be larger steps than when the power measurement is greater than this threshold.	0	15	0
overRangeLowPowerThresh	Threshold (negative sign assumed) which defines the upper boundary on the stable region of the power detect gain control mode.	0	127	7
overRangeHighPowerGainStepAttack	The number of indices that the gain index pointer should be decreased (gain reduction) in the event of the power measurement being greater than overRangeHighPowerThresh.	0	31	4
overRangeLowPowerGainStepAttack	The number of indices that the gain index pointer should be decreased (gain decrease) in the event of the power measurement being less than OverRangeHighPowerThresh but greater than OverRangeLowPowerThresh.	0	31	4
feedback_inner_high_inner_low	A pair of DGPIO pins to retrieve power detector inner low threshold exceeded status and inner high threshold exceeded status	0 (not assigned)	9 (Select DGPIO pins 14 and 15)	0 (not assigned)

Parameter	Description	Min Value	Max Value	Default Value
feedback_apd_high_apd_low	A pair of DGPIO pins to retrieve the apd detector low threshold counter exceeded status and apd high threshold counter exceeded status	0 (not assigned)	9 (Select DGPIO pins 14 and 15)	0 (not assigned)

Table 73. adi_adrv9001_PeakDetector_t Structure Definition

Parameter	Description	Min Value	Max Value	Default Value (TBD)
agcUnderRangeLowInterval	This sets the time constant (in AGC clock cycles) that the AGC will recover when the signal peaks are less than hbUnderRangeLowThresh. Only applicable when the fast recovery option is enabled in Peak Detect AGC mode.	Depends on hb detector settings	65535	50
agcUnderRangeMidInterval	This sets the time constant (in AGC clock cycles) that the AGC will recover when the signal peaks are less than hbUnderRangeMidThresh. Calculated as (underRangeMidInterval+1) × underRangeLowInterval. Only applicable when the fast recovery option is enabled in Peak Detect AGC mode.	0	63	2
agcUnderRangeHighInterval	This sets the time constant (in AGC clock cycles) that the AGC will recover when the signal peaks are less than hbUnderRangeHighThresh. Calculated as (underRangeHighInterval+1) * underRangeMidInterval. Only applicable when the fast recovery option is enabled in Peak Detect AGC mode.	0	63	4
apdHighThresh	This sets the upper threshold of the analog peak detector. When the input signal exceeds this threshold a programmable number of times (set by its corresponding overload counter) within a gain update period, the overload detector flags. In AGC modes, the gain will be reduced when this overload occurs.	apdLow Thresh	63	21
apdLowThresh	This sets the lower threshold of the analog peak detector. When the input signal exceeds this threshold a programmable number of times (set by its corresponding overload counter) within a gain update period, the overload detector flags. In Peak AGC mode, the gain is increased when this overload is not occurring. In Power AGC mode, this threshold can be used to prevent further gain increases if the lowThreshPreventGainInc bit is set.	0	apdHigh Thresh	12
apdUpperThreshPeakExceededCount	Sets number of peaks to detect above apdHighThresh to cause an APD High Over Range Event. In AGC modes, this will result in a gain decrement set by apdGainStepAttack.	0	255	6
apdLowerThreshPeakExceededCount	Sets number of peaks to detect above apdLowThresh to cause an APD Low Overload Event. In Peak Detect AGC mode, if an APD Low Overload Event is not occurring then this will result in a gain increment set by apdGainStepRecovery.	0	255	3
apdGainStepAttack	The number of indices that the gain index pointer should be decreased in the event of an APD High Over Range in AGC modes. The step size in dB depends on the gain step resolution of the gain table (default 0.5dB per index step).	0	31	2

Parameter	Description	Min Value	Max Value	Default Value (TBD)
apdGainStepRecovery	The number of indices that the gain index pointer should be increased in the event of an APD Under range event occurring in Peak Detect AGC mode. The step size in dB depends on the gain step resolution of the gain table (default 0.5dB per index step).	0	31	0
enableHbOverload	1: HB Overload Detector enabled 0: HB Overload Detector disabled	0	1	1
hbOverloadDurationCount	The number of clock cycles (at the HB output rate) within which hbOverloadThreshCnt must be exceeded for an overload to occur. A HB overload flag is only raised when the number of these overloads exceeds hbUpperThreshPeakExceededCnt or hbLowerThreshPeakExceededCnt within a gain update period.	0	7	1
hbOverloadThreshCount	Sets the number of individual samples exceeding hbHighThresh or hbLowThresh necessary within hbOverloadDurationCnt for an overload to occur. The HB overload flag will only be raised when the number of these overloads exceeds hbUpperThreshPeakExceededCnt or hbLowerThreshPeakExceededCnt within a gain update period.	1	15	1
hbHighThresh	This sets the upper threshold of the HB detector.	0	16383	13044
hbUnderRangeLowThresh	This sets the lower threshold of the HB under range threshold detectors. Used only when the fast recovery option of the peak detect AGC mode is being used.	0	16383	5826
hbUnderRangeMidThresh	This sets the middle threshold of the HB under range threshold detectors. Used only when the fast recovery option of the peak detect AGC mode is being used.	0	16383	8230
hbUnderRangeHighThresh	Peak Detect Mode: Threshold used for gain recovery. Peak Detect with Fast Recovery Mode: This sets the highest threshold of the HB under range threshold detectors. Power Detect Mode: Threshold used to prevent further gain increases if lowThreshPreventGainInc is set.	0	16383	7335
hbUpperThreshPeakExceededCount	Sets number of individual overloads above hbHighThresh (number of times hbOverloadThreshCount was exceeded in hbOverloadDuractionCount) to cause an HB High Over Range event. In AGC modes, this will result in a gain decrement set by hbGainStepAttack.	0	255	6
hbUnderRangeHighThreshExceededCount	Sets number of individual overloads above hbUnderRangeHighThresh (number of times hbOverloadThreshCount was exceeded in hbOverloadDurationCount) to cause an HB Under Range High Threshold Overload Event. In Peak Detect AGC mode, not having sufficient peaks to cause the overload is flagged as an under-range event and the gain is recovered by hbGainStepHighRecovery.	0	255	3
hbGainStepHighRecovery	The number of indices that the gain index pointer should be increased in the event of an HB Under Range High Threshold Under Range Event.	0	31	2

Parameter	Description	Min Value	Max Value	Default Value (TBD)
hbGainStepLowRecovery	Only applicable in fast recovery mode of peak detect AGC. This sets the number of indices that the gain index pointer should be increased in the event of an HB Under Range Low Threshold Under Range Event.	0	31	6
hbGainStepMidRecovery	Only applicable in fast recovery mode of peak detect AGC. This sets the number of indices that the gain index pointer should be increased in the event of an HB Under Range Mid Threshold Under Range Event.	0	31	4
hbGainStepAttack	The number of indices that the gain index pointer should be decreased in the event of an HB High Threshold Over Range event in AGC modes. The step size in dB depends on the gain step resolution of the gain table (default 0.5dB per index step).	0	31	2
hbOverloadPowerMode	Sets the measurement mode of the HB detector. HB uses I^2+Q^2 when set to 1. Otherwise the HB uses max(I, Q) per sample.	0	1	0
hbUnderRangeMidThresh ExceededCount	Only applicable in fast recovery mode of peak detect AGC. Sets number of individual overloads above hbUnderRangeMidThresh (number of times hbOverloadThreshCount was exceeded in hbOverloadDurationCount) to cause an HB Under Range Mid Threshold Overload Event. In Peak Detect AGC mode, not having sufficient peaks to cause the overload is flagged as an under-range event and the gain is recovered by hbGainStepMidRecovery.	0	255	3
hbUnderRangeLowThresh ExceededCount	Only applicable in fast recovery mode of peak detect AGC. Sets number of individual overloads above hbUnderRangeLowThresh (number of times hbOverloadThreshCount was exceeded in hbOverloadDurationCount) to cause an HB Under Range Low Threshold Overload Event. In Peak Detect AGC mode, not having sufficient peaks to cause the overload is flagged as an under-range event and the gain is recovered by hbGainStepLowRecovery.	0	255	3
feedback_ apd_low_hb_low	A pair of DGPIO pins to retrieve the hb low threshold counter exceeded status and apd low threshold counter exceeded status	0 (not assigned)	9 (Select DGPIO pins 14 and 15)	0 (not assigned)
feedback_ apd_high_hb_high	A pair of DGPIO pins to retrieve the hb high threshold counter exceeded status and apd high threshold counter exceeded status	0 (not assigned)	9 (Select DGPIO pins 14 and 15)	0 (not assigned)

Table 74. adi_adrv9001_ExtLna_t Structure Definition

Parameter	Description	Min Value	Max Value	Default Value
gpio	TBD	TBD	TBD	0
powerDown	TBD	TBD	TBD	0
settlingDelay	External LNA Settling Delay	TBD	TBD	0

A set of receiver gain control APIs are provided for user interaction with the ADRV9001 device. Some of them have been mentioned in the previous sections. The following table summarizes the list of API functions currently available with a brief description for each one. For more up-to-dated information and detailed descriptions, please refer to API doxygen document.

Table 75. A List of Rx Gain Control APIs

Rx Gain API Function Name	Description
adi_adrv9001_Rx_GainControl_Mode_Set	Configures the Rx gain control mode for a specific channel
adi_adrv9001_Rx_GainControl_Mode_Get	Retrieves the currently configured Rx gain control mode
adi_adrv9001_Rx_Gain_Get	Reads the Rx Gain Index for the requested Rx channel
adi_adrv9001_Rx_Gain_Set	Sets the current AGC Gain Index for the requested Rx channel
adi_adrv9001_Rx_GainTable_Write	Programs the gain table settings for Rx channels
adi_adrv9001_Rx_GainTable_Read	Reads the gain table entries for Rx channels requested
adi_adrv9001_Rx_DecimatedPower_Get	Gets the decimated power for the specified channel
adi_adrv9001_Rx_GainControl_Configure	Sets up the device Rx Gain Control for a specified channel
adi_adrv9001_Rx_GainControl_Inspect	Inspects the device Rx Gain Control for a specified channel
adi_adrv9001_Rx_GainControl_MinMaxGainIndex_Set	Sets the min/max gain indexes for gain control operation for the specified channel
adi_adrv9001_Rx_GainControl_MinMaxGainIndex_Get	Gets the min/max gain indexes for gain control for the specified channel
adi_adrv9001_Rx_GainControl_Reset	Resets all state machines within the gain control block
adi_adrv9001_Rx_GainControl_PinMode_Configure	Configures gain control for MGC PIN mode
adi_adrv9001_Rx_GainControl_PinMode_Inspect	Inspects gain control configurations for MGC PIN mode

DIGITAL GAIN CONTROL AND INTERFACE GAIN (SLICER)

The digital gain control has two major purposes, one for gain correction which is to correct the small step size inaccuracy in analog front-end attenuation, and the other for gain compensation which is to compensate for the entire analog front-end attenuation. In the gain compensation mode, for example, if 5 dB analog attenuation is applied at the front end of the device then 5dB of digital gain will be applied. This ensures that the digital data is representative of the RMS power of the signal at the receiver input port (plus the nominal receiver analog gain) so that any internal front-end attenuation changes in device for preventing ADC overloading are transparent to the baseband processor. In this way, the device's AGC can be used to react quickly to incoming blockers without the need for the baseband processor to track the current gain index for the level of the received signal at the input to the device for signal strength measurements.

The digital gain block is controlled by the receiver gain table as mentioned earlier. Note different digital gain will be applied when configured in gain correction or gain compensation mode. The receiver gain table has a unique front-end attenuator setting, with a corresponding amount of digital gain, programmed at each index of the table, as shown in Table 56.

For the gain compensation mode, it can be used in either AGC or MGC mode. The digital gain allows for compensation of both the internal analog attenuator and an external gain component (such as a DSA or LNA). After the digital gain compensation, the signal power should only depend on the input signal power.

Around the end of the receiver datapath, receiver interface gain could be further applied by using a "Slicer" block for 2 major purposes. One is to avoid digital saturation due to the bit-width limitation of the data port in gain compensation mode. The other one is to ensure the overall SNR is limited only by analog noise and unaffected by quantization noise. When gain compensation mode is used, any analog attenuation is compensated by a corresponding digital gain, such that the sum of the analog and digital gain is always equal to the nominal receiver analog gain of 20 dB. At the ADC input, the full scale input signal is approximately 8.6 dBm. This value translates to 0 dBFS in the digital datapath for either the I or Q channel. As an example, assuming a 5dBm signal is applied at receiver input port, at the receiver output, the signal power will be 5 + 20 = 25 dBm or 25 − 8.6 = 16.4 dBFS. This will cause clipping in 16-bit output signal. Therefore, interface gain (less than 0 in this case) could be applied to attenuate the signal to avoid clipping. On the other hand, for a very low signal level, at receiver input, within the RF bandwidth of interest, it must be assured that the analog noise dominates the quantization noise. In the receiver datapath, different modes of receiver data interface are provided, which can be classified into 2 major categories, one using a final 15-bit or 16-bit quantizer to round the 22-bit receiver data to 15-bit or 16-bit, and the other passing the 22-bit receiver data entirely without using any quantizer. In the first mode, the quantizer becomes the dominant noise source as a result of the final interface quantization. This quantization noise as a result of the final 15-bit or 16-bit quantizer will be spread over a bandwidth equivalent to its output sampling frequency. For NB applications where the output sampling frequency is low, the total quantization noise per Hz could be larger than the analog noise per Hz. By applying interface gain (greater than 0 in this case), prior to the final quantizer, the signal level and analog noise level are both increased. Therefore, the analog noise dominates over the quantization noise so that SNR is dominated by analog front-end noise in the RF bandwidth of interest. For WB applications, since the sampling frequency is higher, the total quantization noise becomes much smaller. In such a case, the analog noise could be way above the quantization noise, therefore, interface

gain is not required. In the second mode, it passes the full 22-bit I/Q data from the receiver data path to the interface without rounding. Therefore, it lowers the quantization noise without the need for additional interface gain. It uses 32 bit I data and 32 bit Q data on the interface for CMOS 1-lane (64-bit) and LVDS 2-lane (32-bit I data and 32-bit Q data). Besides including the 22-bit I/Q data, the 32-bit data also includes some extra fields, which are 1 bit of slicer gain or AGC gain change indicator and one of the followings: zeros, interface gain or AGC gain index. Please refer to Data Interface section for more details.

Figure 162 is a block diagram of the digital gain control portion of the Rx chain, showing the locations of the various blocks in the simplified datapath.

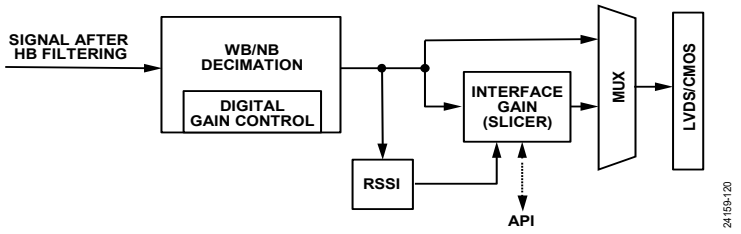


Figure 162. Gain Control and Slicer Section of the Receiver Datapath

It can be seen from Figure 162that digital gain control is performed in the WB/NB Decimation block. In NB and WB applications, the digital gain control is actually performed at different stages of the receiver data chain to achieve optimal performance, which is simplified in Figure 162. The slicer must be dependent on the desired signal power alone and must be done only when all the interfering signals have been filtered out, for example, close to the end of the datapath. The Slicer operation can either be controlled automatically by the device internally or by user externally through API commands. When controlled internally the RSSI block is used to determine the amount of interface gain.

The following sections describe four different digital gain control modes in the device.

Mode 1: No Digital Gain Compensation with Internal Interface Gain Control

In this mode the digital gain block is used for gain correction. It applies a small amount of digital gain/attenuation to provide consistent gain steps in a gain table. The premise is that because the analog attenuator does not have consistent steps in dB across its range then the digital gain block can be used to even out the steps for consistency (the default table uses the digital gain block to provide consistent 0.5 dB steps).

With internal control, the device automatically applies the interface gain determined by RSSI, which measures the input signal power right before the slicer. Note in the gain correction mode, interface gain less than 0 is not needed since the Rx output level should not exceed 0 dBFS through either AGC or MGC. When in NB applications, the interface gain range could be from 0 dB to 18 dB in 6 dB step size (0, 6, 12, 18) for improving the sensitivity when a quantizer is used. In WB applications, as discussed earlier, the sensitivity is already satisfied by the high sampling rate so the interface gain is always 0.

After applying the interface gain, the signal is provided to the data port. The baseband processor could retrieve the interface gain through API commands to scale the power of the received signal to determine the power at the input to the device (or at the input to an external gain element if considered part of the digital gain compensation).

Mode 2: No Digital Gain Compensation with External Interface Gain Control

This mode is similar to mode 1 except that user controls the interface gain manually. Similarly, when in NB applications, the interface gain range could be selected from 0 dB to 18 dB in 6 dB step size when a quantizer is used while in WB applications the interface gain is fixed at 0 dB.

Mode 3: Digital Gain Compensation with Internal Interface Gain Control

In this mode gain compensation is used and the interface gain is determined internally. The device should be loaded with gain tables that compensate for the analog front-end attenuation applied. Thus, as the analog front-end attenuation is increased, and equal amount of digital gain is applied. The interface gain is determined by RSSI. If the power level is too high, the Slicer will shift the signal properly before sending to the data port to avoid saturation.

Slicer example: Considering 3 different input signal power levels. The Power Level 1 fits a data length of 16 bit-width. The Power Level 2 is 0 dB to 6 dB higher than Power Level 1which increases the bit-width by 1. The Power Level 3 is 6 dB to 12 dB higher than Power Level 1 which further increases bit-width by 1. Figure 163 outlines this effect, with gray boxes indicating the valid (used) bits in each case.

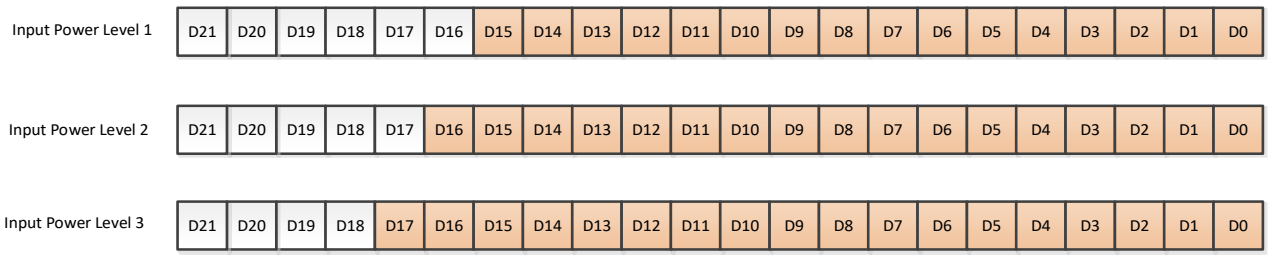


Figure 163. Bit Width of Input Signal with Increasing Power Levels

The slicer is used to attenuate the data such that it can fit into the resolution of the data port. Since the output is a shifted version of the input, the slicer can only handle gains that are in ±6 dB steps.

Figure 164 explains the slicer operation. For Power Level 1, the slicer shift value is calculated as 0 so the 16-bit output data is taken from D15 – D0. As the power level increases, the bit-width of the signal has increased. For Power Level 2, now the bit-width is 17. The slicer shift value becomes 1 so the 16-bit output data is taken from D16 – D1. This is equivalent to apply 6 dB of attenuation by slicer which ensures that the bit-width of the signal is 16 once more; that is, the 16 MSBs have been selected (sliced) with the LSB dropped. When the power level further increases as Power Level 2, the signal bit-width becomes 18-bit. The slicer shift value becomes 2 so the 16-bit output data is taken from D17 – D2, which is equivalent to apply 12 dB of attenuation by slicer or slice the 16 MSBs dropping the 2 LSBs.

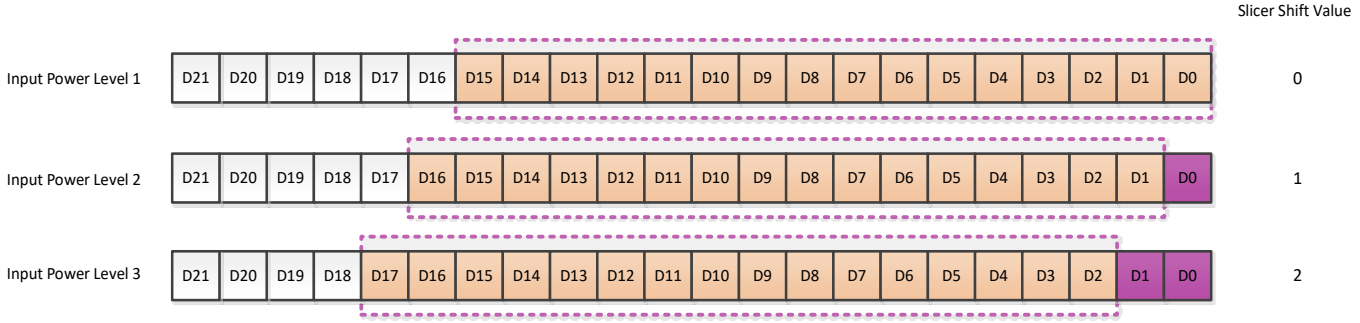


Figure 164. Slicer Bit Selection with Different Input Power Levels

The slicer algorithm assumes a max PAR of 15dB and it adjusts the interface gain such that the measured signal power + 15 dB is less than 0 dBFS. For NB applications, the interface gain is from −36 dB to +18 dB and for WB applications, the interface gain is from −36 dB to 0 dB in 6 dB step size.

Similarly, the baseband processor could retrieve the interface gain through API commands to scale the power of the received signal to determine the power at the input to the device (or at the input to an external gain element if considered part of the digital gain compensation).

Mode 4: Digital Gain Compensation with External Interface Gain Control

This mode is similar to mode 3 except that user controls interface gain by selecting a proper value. The baseband processor could measure the input signal power or use the power measurement done by RSSI in the device to determine the interface gain. Then through API commands and the Slicer will operate in the same way as mentioned in mode 3. For NB applications, the interface gain is from −36 dB to +18 dB and for WB applications, the interface gain is from −36 dB to 0 dB in 6 dB step size. This mode could be used especially when baseband processor input signal clipping is observed by the user.

DIGITAL GAIN CONTROL AND INTERFACE GAIN API PROGRAMMING

The API function adi_adrv9001_Rx_InterfaceGain_Configure() is provided to configure the interface gain. The configuration structure adrv9001_RxInterfaceGainCtrl_t is defined as the following:

```
typedef struct adi_adrv9001_RxInterfaceGainCtrl
{
    adi_adrv9001_RxInterfaceGainUpdateTiming_e updateInstance;  /* Time at which Rx interface
gain control must be updated. 0: To be updated at start of next frame 1: To be updated
immediately */
adi_adrv9001_RxInterfaceGainCtrlMode_e   controlMode; /* 0: Uses internal Rx interface gain
value 1: Uses external Rx interface gain value. Gain value must be provided in this case. */
```

```
adi_adrv9001_RxGainTableType_e    gainTableType;   /* 0: Gain Correction table 1: Gain
Compensation table
adi_adrv9001_RxInterfaceGain_e            gain;            /* a value between 0 and 9 (0x0 =
18dB, 0x1 = 12dB, 0x2 = 6dB, 0x3 = 0dB, 0x4 = -6dB, 0x5 = -12dB, 0x6 = -18dB, 0x7 = -24dB, 0x8 =
-30dB, 0x9 = -36dB). */
} adrv9001_RxInterfaceGainCtrl_t
```

It is clear from the above that there are 2 interface gain control modes, which are internal (automatic) control and external control. In addition, there are 2 options to apply the interface gain. The first option is to apply it at the start of the next frame and the second option is to apply it immediately. The interface gain could be selected from −36 dB to +18 dB in 6 dB step size, a total of 10 options, which is defined by "adi_adrv9001_RxInterfaceGain_e" as the following:

```
typedef enum adi_adrv9001_RxInterfaceGain
{
    ADI_ADRV9001_RX_INTERFACE_GAIN_18_DB = 0,       /*  18 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_12_DB,           /*  12 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_6_DB,            /*   6 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_0_DB,            /*   0 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_6_DB,   /*  -6 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_12_DB,  /* -12 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_18_DB,  /* -18 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_24_DB,  /* -24 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_30_DB,  /* -30 dB */
    ADI_ADRV9001_RX_INTERFACE_GAIN_NEGATIVE_36_DB,  /* -36 dB */
} adi_adrv9001_RxInterfaceGain_e
```

Note as discussed before, depending on the gain table type and the bandwidth of the application, the interface gain could be limited to a subset of the 10 options. This API must be called in the "CALIBRATED" state.

To change the interface gain on the fly while the channels are operational, the API function adi_adrv9001_Rx_InterfaceGain_Set() could be used. The gain should be selected from one of the 10 options.

The following table summarizes the list of API functions currently available for digital gain control and interface gain. For more up-to-date information and detailed descriptions, refer to the API doxygen document.

Table 76. A List of Rx Interface Gain Control APIs

Rx Gain API Function Name	Description
adi_adrv9001_Rx_Rssi_Read	Reads back the RSS status for the given channel
adi_adrv9001_Rx_InterfaceGain_Configure	Sets the Rx interface gain control configuration parameters for the given Rx channel
adi_adrv9001_Rx_InterfaceGain_Set	Sets the Rx interface gain for the given Rx channel
adi_adrv9001_Rx_InterfaceGain_Inspect	Gets the Rx interface gain control configuration parameters for the given Rx channel
adi_adrv9001_Rx_InterfaceGain_Get	Gets the Rx interface gain for the given Rx channel

USAGE RECOMMENDATIONS

In this section, a list of recommendations is summarized for achieving optimal gain control performance:

It is recommended to start with AGC and default configurations.

- When changing the default configurations is needed, it is better to change parameters one by one.
- The high thresholds are used as limits on the incoming signal level and should be set based on the maximum input of the ADC. The high thresholds should be set at least 3 dB to 6 dB lower than the full input scale of the ADC.
- The apdHighThresh and hbHighThresh are set to an equivalent dBFS value. Likewise, the apdLowThresh and the hbUnderRangeHighThresh are set to equivalent values.
- The apdUpperThreshPeakExceededCount and hbUpperThreshPeakExceededCount should be set properly to achieve a desired level of AGC sensitivity to input signal peaks.
- The gain change period should typically be aligned to the frame or subframe boundary periods.

- Peak detect can achieve faster response time comparing with peak/power detect. For applications requiring “fast attack” and “fast recovery”, peak detect provides better performance.
- “Fast recovery” mode should not be used when it is required to increase gain only at the frame or subframe boundary.
- For applications requiring “fast attack” and “fast recovery”, SLS delay should be set as short as possible to minimize the delay while allowing sufficient time to set the gain changes.

TES CONFIGURATION AND DEBUG INFORMATION

User could interact with the receiver gain control functionality through API commands as discussed in previous sections or through ADRV9001 Transceiver Evaluation Software (TES). While using TES, user should first configure the receiver gain control operation mode and the signal detector operation mode as shown in Figure 165. (Note TES design could continuously change, see the ADRV9001 Evaluation System section for updated information.)

This configuration page is under **GPIO Configurations** tab in TES. Gain Control Mode provides user three options which are **Automatic**, **Manual Pin,** and **Manual SPI**, which are corresponding to the AGC, MGC with pin control and MGC with SPI control. By default, it is set as **Manual SPI**. When **Manual Pin** is selected, user should further select the GPIO pins for gain increment and decrement. After selecting the gain control mode, user can further configure the **Detection Mode** which has two options “**Peak Only**” and **Peak and Power**. By default, it is set as **Peak Only** mode. The **Detection Mode** indicates which AGC mode is configured when the **Gain Control Mode** is selected as **Automatic**. When the **Gain Control Mode** is selected as **Manual Pin** or **Manual SPI**, it further enables the ADRV9001 internal signal detectors in either **Peak Only** or **Peak and Power** mode. By configuring the GPIO pins, user is allowed to retrieve the signal detector status which could be used to control receiver gain in **Manual** mode. Note the signal detector status could also be retrieved in AGC mode.

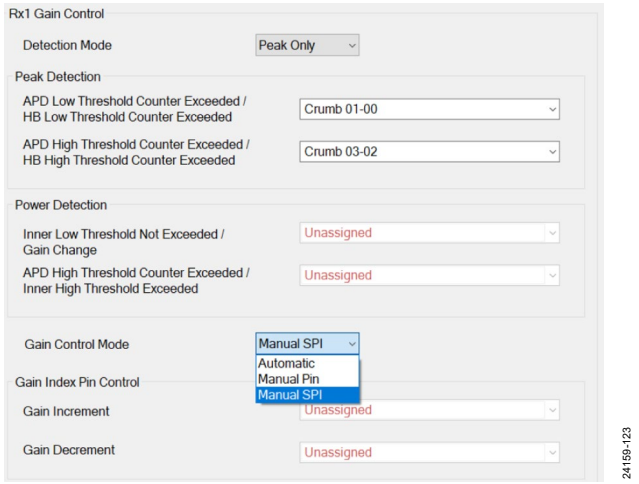


Figure 165. TES Configuration for Rx Gain Control Mode and Signal Detector Operation Mode

After configuring **Gain Control Mode** and **Detection Mode**, user could further configure signal detector parameters, interface gain and manual control parameters under the **Gain Control** tab in TES as shown in Figure 166 and Figure 167. In Figure 166, **AGC** mode is selected and in Figure 167, **MGC Pin** mode is selected. As shown in both figures, TES provides three sections for user configurations. The first section is **Rx Interface Gain**. User can choose **Manual** or **Automatic** receiver interface gain control. When **Manual** is selected, user can further enter the desired interface gain value and when the gain update should take effective (note interface gain configuration relates to the selection of gain table. User should first select Rx gain table type at the bottom of this configuration page in TES.) The second section displays the current gain control mode. If AGC is configured as shown in Figure 166 users are not allowed to enter the other parameters in this section. If MGC is configured as shown in Figure 167, user should further configure other parameters. With **MGC Pin** mode, user should configure the targeted **Manual Gain Index** as well as the step sizes. The third section displays the detection mode user selected and it allows user to configure the signal detector related operating parameters, such as **Peak Overload threshold** and **Peak Underload Threshold** (they apply to both APD and HB detectors), **Gain Refresh Period**, **Max Gain Index**, and **Min Gain Index** when **Peak Only** mode is selected. Note some selections are greyed out either because they are not configurable due to current software limitation or not applicable in the selected modes. TES provides sanity checks for the parameter user enters so when the value is out of range, the user input is not allowed.

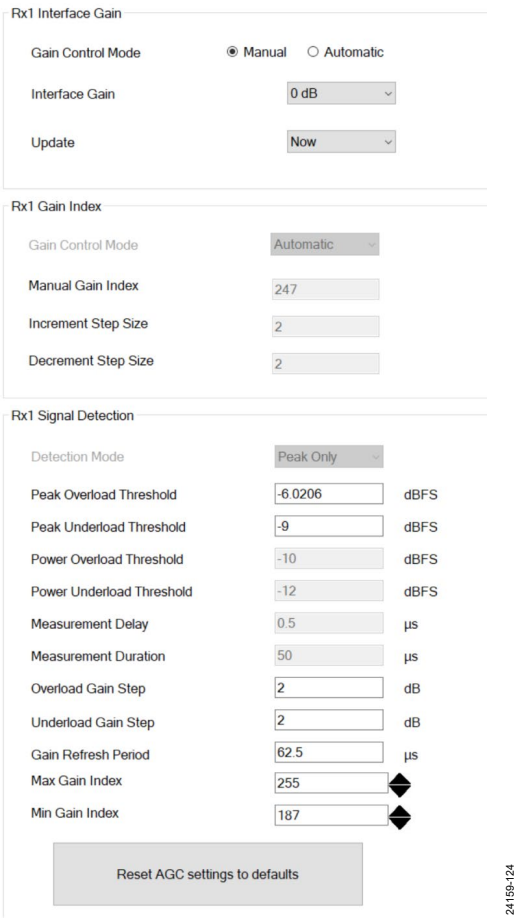


Figure 166. TES Configuration for Rx Interface Gain, Signal Detection Parameters and Manual Control Mode Parameters (when AGC is Configured)

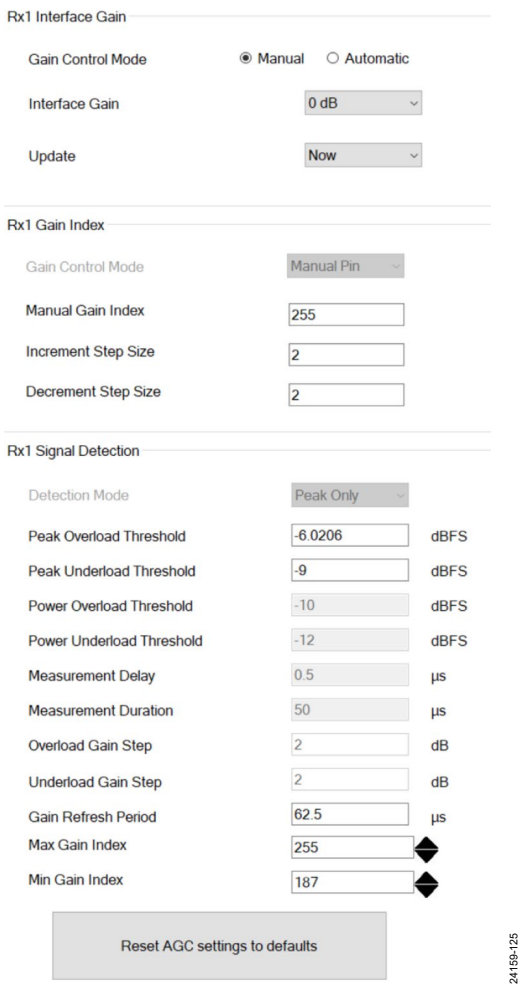


Figure 167. TES Configuration for Rx Interface Gain, Signal Detection Parameters and Manual Control Mode Parameters (when MGC Pin is Configured)

After finishing all the configurations, user could start the receive operations and observe the receiver gain changes. It is recommended to start from the default settings and change the parameters one by one as needed. "Reset AGC settings to defaults" can be used to reset all the parameters to their default values.

When the receiver gain control is not working as expected, the user could perform the following simple self-debugging:

- Check if the gain control mode is set as AGC or MGC.
- Check if the MAX and MIN index is set properly. When set improperly, the gain control capability could be significantly impacted.
- If detectors are not working, check if the gain step is set as 0, which will disable gain attack or gain recovery.
- When signal saturation is observed, adjust the slicer/interface gain could help.

Rx DEMODULATOR

Rx NARROW-BAND DEMODULATOR SUBSYSTEM

ADRV9001 Rx narrow-band demodulator subsystem, denoted by rxnbdem, is the digital baseband backend partition of ADRV9001 Rx channel. Note that narrow band, commonly for a wireless communication system, if the channel spacing, also known as channel bandwidth, is no more than 1 MHz, we call it "Narrowband System". Otherwise, we call "Wideband System". Figure 168 illustrates the rxnbdem subsystem, incorporating signal buffering, carrier frequency offset correction, programmable channel filtering, frequency discrimination, narrowband programmable pulse shaping, and resampling function. The input of rxnbdem, driven by the RX decimation filters, is the ZIF digital baseband IQ signal. Programmability exists to bypass each block in the rxnbdem subsystem. The output of rxnbdem, is directly sent to the RX SSI interface. Depending on the programmed functionality, the output can be Frequency Deviation(I)-only or IQ.

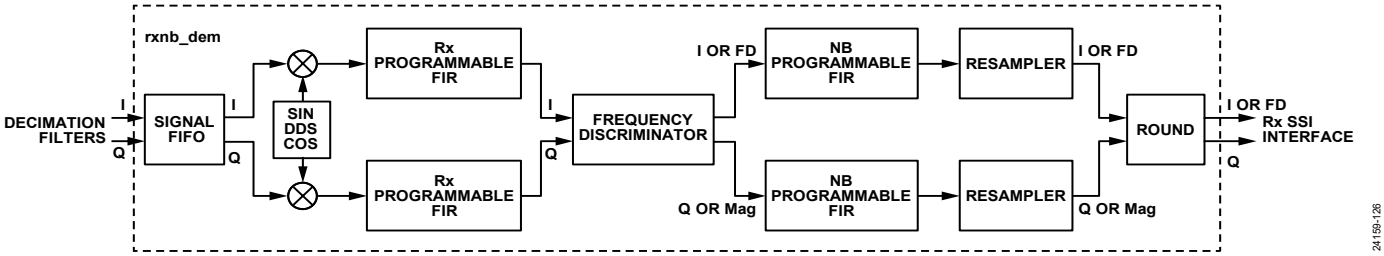


Figure 168. Functional Diagram of rxnbdem

Signal FIFO

Signal FIFO is to buffer the input IQ data stream, and it is applicable only in the CMOS SSI operation mode. The Signal FIFO depth is 2048. As a result, it can store for more than 85 ms at the sampling rate of 24 kHz

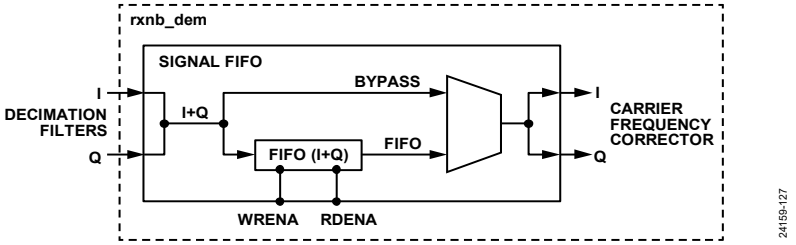


Figure 169. Functional Diagram of Signal FIFO

The Signal FIFO can be disabled or enabled based on the users' requirement. As the Signal FIFO is disabled, this block is bypassed and cannot be written or read. As the Signal FIFO is enabled, the writing control and reading control of the Signal FIFO can be manipulated separately. The FIFO reading clock is configurable, can be 1×, 2×, 4×, or 8× of the FIFO writing clock. For wideband modes, only 1x and 2× are supported and the reading clock rate cannot be above 61.44 MHz.

In the Signal FIFO, as shown in Figure 169 there is an output mux, which has 2 inputs: one input, denoted as "Bypass", is driven by the input IQ stream to the FIFO; the other, denoted as "FIFO", is driven by the output IQ stream from the FIFO. The mux inputs can be switched on demand to drive the following modules in rxnbdem.

Below is an example explaining how to use the Signal FIFO.

During the signal capturing phase before the wireless data link is established, the mux should be kept at "Bypass", and the FIFO writing control is enabled, and the FIFO reading control is disabled. Through the Rx SSI port, the BBIC can keep on detecting the received signal. Meanwhile, the Signal FIFO keeps on buffering the IQ stream. The FIFO writing overflow may or may not happen. If happen, the FIFO always stores the latest 2048 IQ data.

As the BBIC detects the wanted Rx frame from the input data stream and estimates the right starting point of the wanted Rx frame, further the synchronous parameters, the BBIC may switch the mux from "Bypass" to "FIFO", then enable the reading control of the FIFO, to process the stored data stream and the following data stream seamlessly.

Carrier Frequency Corrector (CFC)

Carrier Frequency Corrector in rxnbdem is to remove the carrier frequency offset. This module can be bypassed.

In a communication system, a desired signal is transmitted by the transmitter at RF over the air. Since the clock reference at the transmitter and the receiver are independent, this may result in the RF carrier frequency offset between the transmitter and the receiver. This frequency difference is named by the carrier frequency offset (CFO). The CFC in rxnbdem enables the BBIC to remove the CFO before the channel selection filtering (Rx PFIR) at the receiver side. The correction value applied to the CFC, must be estimated and further input by the BBIC. The change of the correction value may occur immediately or relative to RX frame boundary.

The CFC is implemented as a Digital Down Converter (DDC), which consists of an NCO and a complex multiplier in the datapath. As the correction value, the NCO frequency word should be in the range of min (±20k,20% of sample rate).

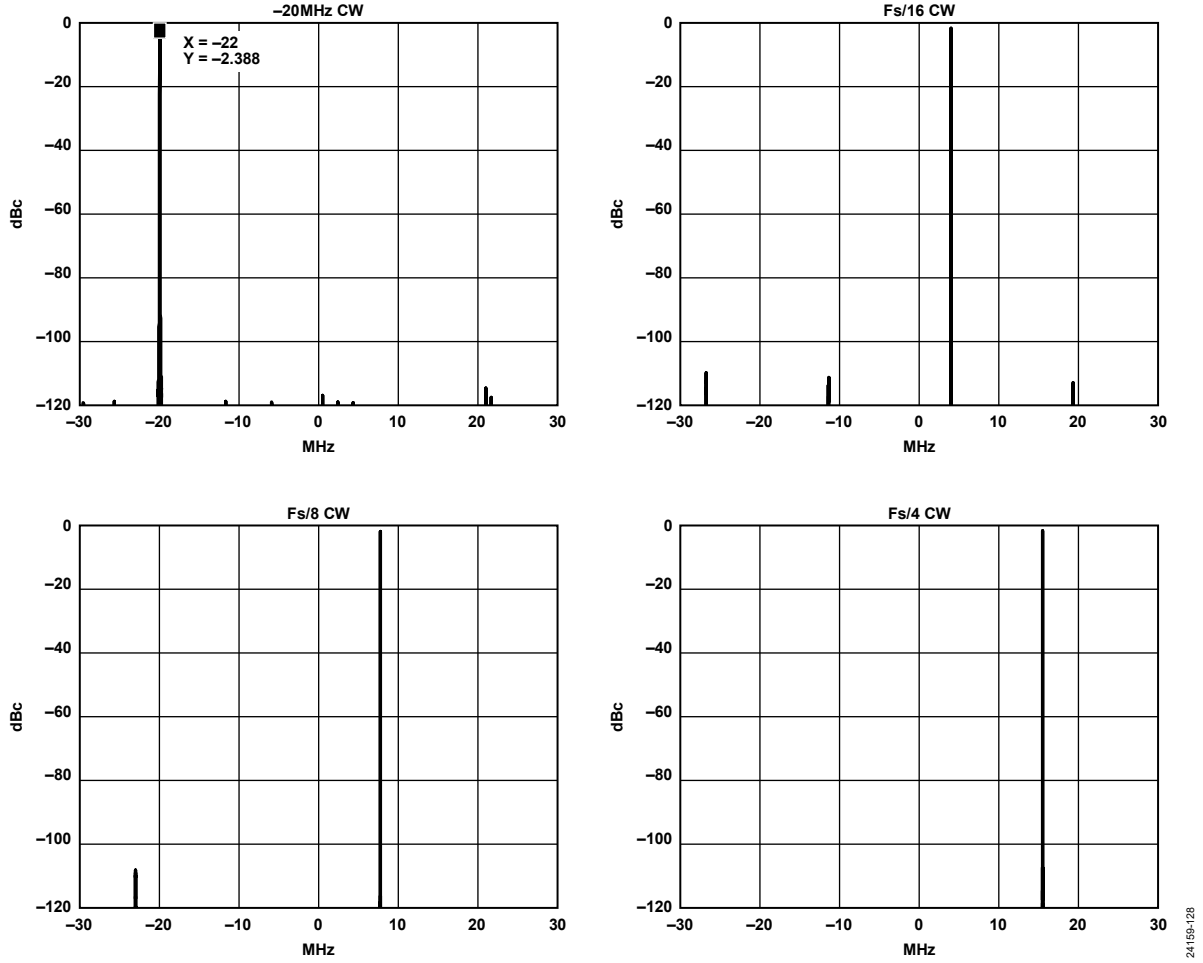


Figure 170. Output Spectrum of the CFC/NCO as f_S = 61.44 MHz

Figure 170 presents the spectrum of desired tone and the generated NCO spurs levels relative to desired tone for the CFC NCO at 61.44 MHz sampling frequency. The outlined plots show a typical case (-20Mhz CW) and some worst cases. As shown in Figure 170, the NCO output spurs are −100 dBc below desired tone across the range of [$-f_S/2$, $f_S/2$]:

- −20 MHz CW,
- $fs/16$ CW,
- $fs/8$ CW, and
- $fs/4$ CW.

Please note these four tones' frequency is not within the NCO range defined above, just for the NCO spurs level demonstration.

Rx Programmable FIR Filter

Rx Programmable FIR Filter in rxnbdem is multi-functional and customizable. This module can be bypassed.

The Rx Programmable FIR supports up to 128 taps. Each tap is 24 bits width with the signed bit included. 4 sets of customized FIR profiles can be stored at the initialization phase. One of the 4 stored FIR profiles can be switched to be loaded on the fly under the control of the BBIC.

The Rx programmable FIR can be loaded a customized lowpass filter profile to stop the adjacent channel interference, which is helpful to achieve better channel selectivity. For example: as shown in Figure 171, before the CFO is corrected, the BBIC may program a loose filter profile onto the Rx Programmable FIR to perform common filtering. However, after the CFO is corrected via Carrier Frequency Corrector block, a tight filter profile can be loaded to perform the deep channel selection filtering. The change in the filter profile can be initiated by the BBIC on demand in the RF_Enabled State.

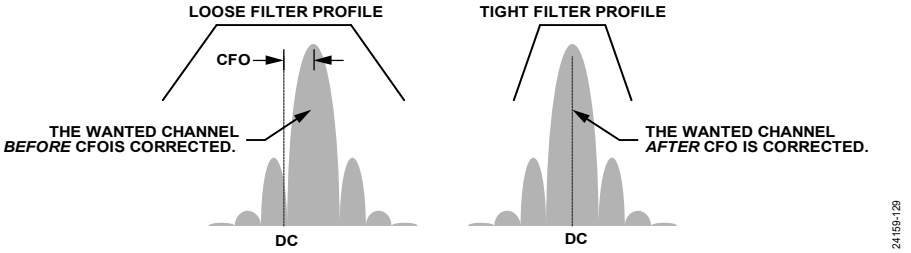


Figure 171. Loose Filter Profile vs. Tight Filter Profile

Frequency Discriminator

Frequency Discriminator in rxnbdem is to translate the IQ signal into Frequency Deviation (FD) signal, performing the frequency demodulation in the digital domain. This module can be bypassed.

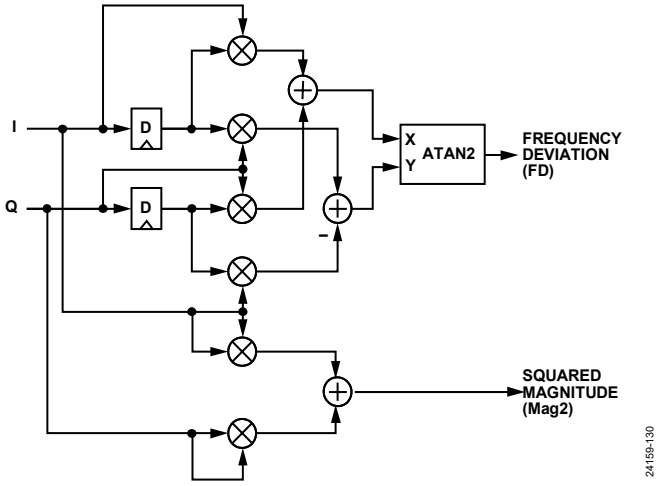


Figure 172. Functional Diagram of Frequency Discriminator

Illustrated by Figure 172, the Frequency Discriminator outputs the transient frequency deviation (FD) and the transient squared magnitude (Mag2) sample by sample. The output FD and the output Mag2 are defined as the following:

$$FD(n) = \frac{1}{\pi} atan2[I(n-D)Q(n) - I(n)Q(n-D), I(n)I(n-D) + Q(n)Q(n-D)]$$

$$Mag2(n) = I(n)^2 + Q(n)^2$$

where $atan2$ is same as the function in Octave, and D is the programmable delay. Typically, D is chosen as '1', which means 1 sampling clock delay.

Assuming the input IQ signal is the complex single tone, given by

$$A \cdot e^{\frac{j2\pi f_t n}{f_s}}, n = 0,1,2,\ldots,$$

where A is the signal magnitude, f_t is the tone frequency, f_S is the sampling frequency. The output of the frequency discriminator (FD) is $D \times 2f_t/f_S$ while the output Mag2 is A^2.

NB Programmable FIR

NB Programmable FIR in rxnbdem is to perform the pulse shaping filtering or the low pass filtering at the output of the Frequency Discriminator. This module can be bypassed.

The NB Programmable FIR supports up to 128 taps. Each tap is 24 bits width including the sign-bit, and only 1 set of customized FIR profiles can be loaded.

Resampler

Resampler in rxnbdem adjusts the sampling phase of the IQ signal or that of the FD signal. This module can be bypassed.

Figure 173 illustrates the functional diagram of the Resampler, the Resampler resamples the incoming received signal, by reconstructing intermediate samples between every 2 inputs samples according to the re-sample phase parameter, μ, where $|\mu| \leq 0.5$.

In the frequency domain, the ideal digital Resampler has the transfer function as below:

$$H_{ideal}(j\omega, \mu) = \begin{cases} T_s \cdot e^{j\omega\mu T_s}, & |\omega/2\pi| \leq B_x \\ don't\ care, & |\omega/2\pi| > B_x \end{cases}$$

where B_x is the maximum bandwidth of the input signal. Filtered by the ideal resampling filter, the input signal, $x(kT_S)$, is converted to the output signal $y(kT_S) = x((k + \mu) \times T_S)$

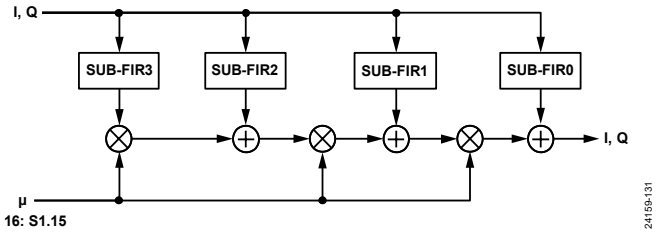


Figure 173. Functional diagram of the Resampler

Scanning the sampling phase μ from 0 to 0.5, the maximum magnitude difference of the frequency transfer function between the Resampler and the ideal digital resampler is collected and plotted at the upside of Figure 174. The maximum phase error is collected and plotted at the downside of Figure 174.

The Resampler can be used in both the wideband and narrowband modes.

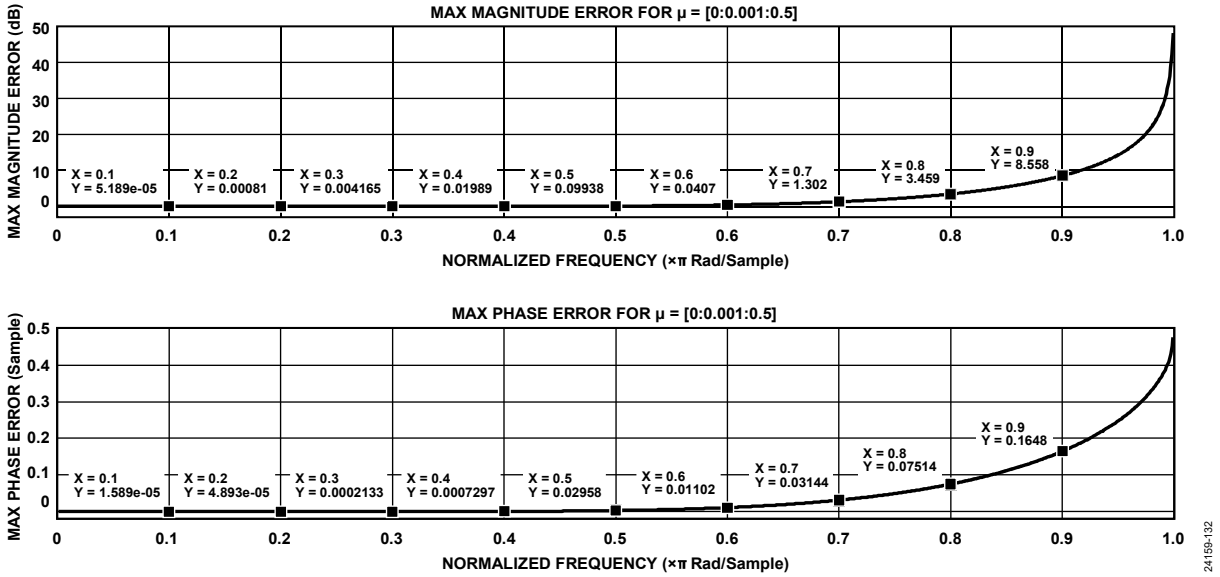


Figure 174. Maximum Magnitude/Phase Error of the Resampler

Note: The resampler configuration is not supported by current ADRV9001 software release yet.

Round Module

Round in rxnbdem is to map the ADRV9001 internal datapath bit-width to the Rx SSI output. This module can be bypassed if users choose IQ-22bit mode.

The Round module can output 16-bit I/Q or 15-bit I/Q data to the Rx SSI output. If required, the Round module can output 16-bit I data to the Rx SSI output with the 16-bit output Q data set as '0'.

NORMAL IQ OUTPUT MODE

The Normal IQ Output mode is applicable for both wideband and narrow band as the frequency discriminator bypassed. Except the Rounding, all other modules can be bypassed. See Figure 175 ADRV9001 Rx narrow-band demodulator can be the common output stage of Rx channel.

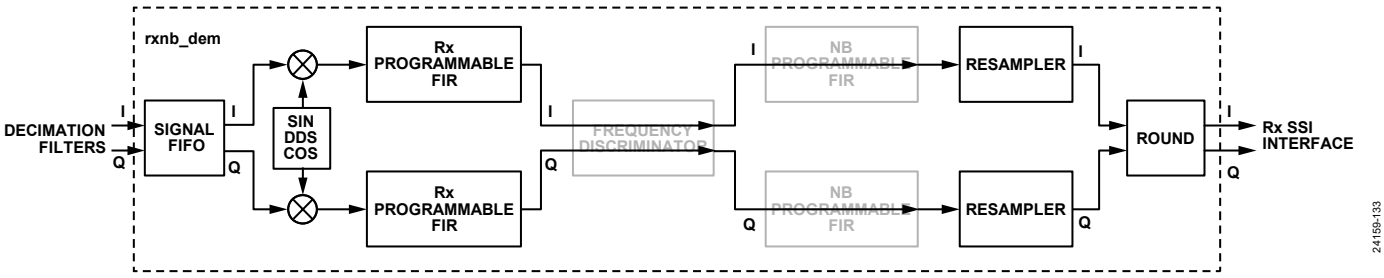


Figure 175. Rxnbdem in Normal IQ output mode

In the Normal IQ output mode, the rxnbdem can provide the signal processing resources as followed:

- a spur-free carrier frequency offset correction,
- a profile-switchable channel filter, and
- a precise IQ resampler (not supported by software yet).

All above resources can be manually enabled and controlled by the BBIC by API (will provide the support in the later software releases).

FREQUENCY DEVIATION OUTPUT MODE

The Frequency Deviation Output mode is only applicable for the narrowband modes. Except the Round and the Frequency Discriminator, all other modules can be bypassed. See Figure 176. ADRV9001 Rx narrowband demodulator contains a frequency discriminator hardware block. Cooperating with other hardware blocks, for example, the CFC/DDC, and programmable FIR filters, and so on, ADRV9001 Rx narrowband demodulator can perform FSK and FM demodulation under the control of the BBIC.

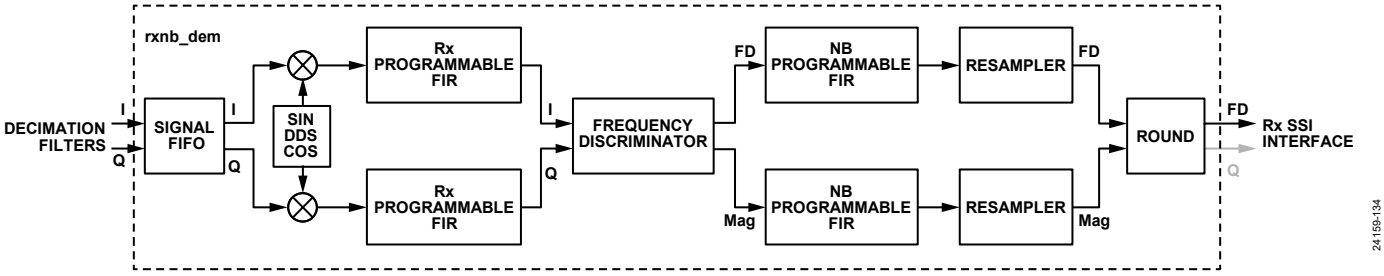


Figure 176. Rxnbdem in Frequency Deviation Output mode

The FSK/FM demodulation can cover the standards listed as followed:

- Analog FM with 12.5 kHz channel bandwidth
- Analog FM with 20 kHz channel bandwidth
- Analog FM with 25 kHz channel bandwidth
- P25 Phase I with 12.5 kHz channel bandwidth
- DMR with 12.5 kHz channel bandwidth

Regarding to each standard at above, Table 77 lists the usage suggestion of each modules in the Frequency Deviation Output mode.

Table 77. Suggestion of rxnbdem Usages in the Frequency Deviation Output Mode

	FM, 12.5kHz	FM, 20kHz	FM, 25kHz	P25 I, 12.5kHz	DMR, 12.5kHz
Input Sampling Clock	24 kHz or 48 kHz		48 kHz	24 kHz or 48 kHz	
Signal FIFO	Optional				
CFC/DDC	Enable for manual control				
Rx Prog. FIR	Enabled for channel filtering with 2 sets of FIR profiles				

Freq. Discriminator	Enabled with D = 1		
NB Prog. FIR	Low-pass filtering for smoothing the output FD signal	Pulse Shaping	
Resampler	Disable		Enable for manual control

Contact an Analog Devices Application Engineer for the specific filter profiles of Rx Programmable FIR and NB Programmable FIR.

In summary, in the frequency deviation output mode, the rxnbdem can provide more important resources as followed:

- A 2k-depth FIFO,
- A spur-free carrier frequency offset correction,
- A profile-switchable channel filter,
- An accurate digital frequency discriminator,
- A low-pass filter or a pulse shaping filter for the frequency deviation, and
- A precise frequency deviation resampler.

All above resources can be manually enabled and controlled by the BBIC.

API PROGRAMMING

Configuration for blocks in rxnbdem subsystem is profile related on so far, all relative blocks are enabled or bypassed in profile by selecting Rx "IQ" mode or "Frequency Deviation" mode, more user configurable capability will be added in later software release.

Carrier Frequency Corrector API Programming

The API function adi_adrv9001_Rx_FrequencyCorrection_Set() is provided to set CFC frequency word, as mentioned earlier, the frequency correction word should be in the range of min (±20k,20% of sample rate) (will change the limitation in later software). And the frequency correct operation can take effect immediately or at the start of next available frame by setting the parameter "immediate" to "Ture" or "False".

Rx Programmable FIR Filter API Programming

Profile predefined Rx PFIR coefficients or customized Rx PFIR coefficients are atomically loaded during chip initialization, there is no need for baseband processor to call any PFIR coefficients loading API function.

The configuration structure adi_adrv9001_PfirWbNbBuffer_t is defined as the following for the programming FIR filter coefficients.

```
typedef struct adi_adrv9001_PfirWbNbBuffer
{
    uint8_t         numCoeff;                 /* number of coefficients */
    adi_adrv9001_PfirSymmetric_e symmetricSel; /* symmetric selection */
    adi_adrv9001_PfirNumTaps_e   tapsSel;     /* taps selection */
    adi_adrv9001_PfirGain_e      gainSel;     /* gain selection */
    int32_t         coefficients[ADI_ADRV9001_WB_NB_PFIR_COEFS_MAX_SIZE]; /* coefficients */
} adi_adrv9001_PfirWbNbBuffer_t;
```

Baseband processor can prepare new FPIR coefficients in one or more adi_adrv9001_PfirWbNbBuffer_t instances and call the API function adi_adrv9001_arm_NextPfir_Set() in "PRIMED" or "RF_ENABLED" state to load each required instance into ADRV9001 hardware. Multiple PFIRs using the same coefficients can be loaded in a single call. However, note that the old coefficients remain in effect until adi_adrv9001_arm_Profile_Switch() is called.

The API function adi_adrv9001_arm_NextRxChannelFilter_Set() calls adi_adrv9001_arm_NextPfir_Set() once or twice as needed to update channel filter coefficients for Rx1, Rx2, or both. Either PFIR pointer may be NULL to prevent modifying the corresponding PFIR but it is an error if both PFIR pointers are NULL.

The ADRV9001 performs the PFIR coefficients switch for all channels that have new coefficients prepared and waiting when the API command adi_adrv9001_arm_Profile_Switch() is called. If ADRV9001 is in PRIMED state, the new coefficients will take effect on the next transition to RF_ENABLED. If it is in RF_ENABLED, they take effect immediately.

An example python code for the RX PFIR coefficients switch is in below:

```
pfir_dmr_12p5k_coeff = [1,4,10,14,10,-8,-36,-56,-43,18,110,176,140,-36,-296,\
-477,-385,65,717,1164,945,-116,-1630,-2655,-2161,224,3612,5917,4823,-660,-8930,\
-15835,-16492,-8267,6681,21178,26054,15428,-8503,-34452,-46572,-33192,4645,50326,\
77802,65235,9575,-66663,-122526,-118715,-41686,81623,189288,211654,109809,-93600,\
```

```
-309489,-411032,-287232,101328,691337,1327974,1817574,2001178,1817574,1327974,\
691337,101328,-287232,-411032,-309489,-93600,109809,211654,189288,81623,-41686,\
-118715,-122526,-66663,9575,65235,77802,50326,4645,-33192,-46572,-34452,-8503,\
15428,26054,21178,6681,-8267,-16492,-15835,-8930,-660,4823,5917,3612,224,-2161,\
-2655,-1630,-116,945,1164,717,65,-385,-477,-296,-36,140,176,110,18,-43,-56,-36,\
-8,10,14,10,4,1,0]
pfir_dmr_12p5k = adi_adrv9001_PfirWbNbBuffer_t()
pfir_dmr_12p5k.numCoeff = 128
pfir_dmr_12p5k.symmetricSel = adi_adrv9001_PfirSymmetric_e.ADI_ADRV9001_PFIR_COEF_NON_SYMMETRIC
pfir_dmr_12p5k.tapsSel = adi_adrv9001_PfirNumTaps_e.ADI_ADRV9001_PFIR_128_TAPS  # PFIR_128_TAPS
pfir_dmr_12p5k.gainSel = adi_adrv9001_PfirGain_e.ADI_ADRV9001_PFIR_GAIN_ZERO_DB  # PFIR_GAIN_0DB
for i in range(pfir_dmr_12p5k.numCoeff):
pfir_dmr_12p5k.coefficients[i] = pfir_dmr_12p5k_coeff[i]

Adrv9001.arm.NextPfir_Set(1, pfir_fm_12p5k) # put in the right filter object
Adrv9001.arm.Profile_Switch()
```

NB Programmable FIR API Programming

Same with Rx PFIR, a profile predefined set of NB PFIR coefficients (customized NB PFIR coefficients will be supported in the later software release) are automatically loaded during chip initialization, there is no need for baseband processor to call any PFIR coefficients loading API function.

POWER SAVING AND MONITOR MODE

ADRV9001 is a high-performance integrated transceiver with low power considerations. To accommodate different user cases, ADRV9001 provides flexibility for users to trade-off between power consumption and performance with some static configuration options, such as:

- Clock PLL option of high performance and low power;
- Clock PLL power option of high, medium and low;
- ADC option of high performance or low power;
- ADC clock rate option of high, medium and low;
- RF PLL LOGEN optimization option of best phase noise and best power consumption;
- RF PLL power option of high, medium and low;
- ARM clock rate option of divided by 1, 2, 4

These static options are chosen and configured in chip initialization stage and are not allowed to dynamically change except the ADC option, high performance ADC, and low power ADC can be dynamically switched after chip has been initialized. Users can refer the relative sections for above options detail in the User Guide.

For TDD applications, ADRV9001 defines different power saving modes to meet the power saving requirement in various user cases. Some standards like DMR (Digital Mobile Radio) require the radio enter periodical sleep and carrier detection cycles in order to save power (Monitor Mode) when radio is not in use. ADRV9001 has dedicated hardware to meet this Monitor Mode requirement, and ADRV9001 software adds additionally static and dynamic power saving schemes in order to extend the power saving feature to a broader market beyond DMR.

ADRV9001 defines five extra power down modes that provides from low to high power saving but short to long recovery time, details will be introduced in the following section.

Three power saving schemes are designed for different power saving applications.

- Temporarily powering up/down the unused Tx/Rx channel in Calibrated state
- Dynamic interframe power saving is running automatically during all regular TDD TX/RX operations. DGPIO pins could be configured to support additional power savings. All configurations can be set by API via fast messages on the fly. Power saving software will smartly handle powering up/down HW components based on PLL mapping and selected power saving mode. There are two power saving choices in interframe operations:
  - Channel power saving. This is to power down a channel (both TX and RX) based on power down mode 0-2.
  - System power saving. This is to power down the whole chip by power down mode 3-5
- Monitor Mode. This can allow baseband processor move into sleep state after it configures and moves ADRV9001 into Monitor mode, ADRV9001 software will control the dedicated hardware and timers for periodical sleeping and detecting. Only power down modes 3-5 are allowed in Monitor Mode.

Users can choose proper power down modes and power saving schemes according to their application scenarios. The following sections explain the detail power saving schemes.

POWER-DOWN MODES

Power down modes are defined to dynamically power down and up different level of ADRV9001 components. Five extra power down modes are defined from low power saving but short recover time to high power saving but long recover time as shown in Table 78. Each higher power down mode would power down additional components than the lower mode, power down mode 3 is the exception.

Table 78. Power-Down Modes and Related Power-Down Components

		Power Down Modes	**0 (default)**	**1**	**2**	**3**	**4**	**5**
Components	TX	Analog and Digital Data Path	x	x	x	x	x	x
		TX Internal PLLs		x	x	x	x	x
		TX LDOs			x		x	x
	RX	Analog and Digital Data Path	x	x	x	x	x	x
		RX Internal PLLs		x	x	x	x	x
		RX LDOs			x		x	x
	System	CLK PLL				x	x	x
		Converter and CLKPLL LDOs					x	x
		ARM (+ memories)						x
Approximate Power-Up Time (µs) [1,3]		DEV_CLK = 30Mhz	4.5	350	500	250	650	3200

		Power Down Modes	0 (default)	1	2	3	4	5
		DEV_CLK = 50Mhz		180	380			
		DEV_CLK = 100Mhz		170	370			
Approximate Power-Up Time (µs) [2,3]		DEV_CLK = 30Mhz		100	300			
		DEV_CLK = 50Mhz		60	260			
		DEV_CLK = 100Mhz		40	240			

[1] RF PLL is in Normal Calibration mode, power up time varies with DEV_CLK frequency

[2] RF PLL is in Fast Calibration mode, power up time varies with DEV_CLK frequency

[3] At 184.32Mhz processor clock

- Power down mode 0 is the default power saving if power saving API is not called to set other values. In this mode, TX/RX enable pin will automatically trigger powering up/down the TX/RX analog and digital datapath components such as Mixer, A/D, filters, and so on The transition time is very short around 4.5 us.
- Power down mode 1 would power down internal RF PLLs in addition to mode 0 power down. After powering up, PLL requires re-calibration, so it takes more time to power up.
- Power down mode 2 would power down some LDOs related to channels and RF PLLs in addition to mode 1 power down.
- Power down mode 3 powers down the TX/RX channels and PLLs (clock PLL, RF PLLs) only. No LDOs are powered down.
- Power down mode 4 powers down Clock PLL and system LDOs related to TX/RX channels in addition to mode 3 power down.
- Power down mode 5 powers down almost the whole ADRV9001 chip including ARM and memory except some wake up circuits.

POWER-DOWN/POWER-UP CHANNEL IN CALIBRATED STATE

User could power down/up individual channel (TX1/TX2/RX1/RX2) dynamically in Calibrated State if these channels are statically enabled in device profile. adi_adrv9001_Radio_Channel_PowerDown() can be called to power down the specified channel, it will power down the channel related LDOs and PLL for the channel in addition to datapath power down. adi_adrv9001_Radio_Channel_PowerUp() is used for power up the specified channel. User should notice that these two APIs can only be called in Calibrated state.

Figure 177 shows a DMR radio switch from TX only frames into TX/RX alternate frames, ADRV9001 is initialized with Tx and Rx enabled, at the beginning of TX only frames, baseband processor can bring the RX channel into Calibrated State and power it down. Then before the transition of TX/RX alternate frames, baseband processor can power up RX and move RX into Primed state. The power saving of TX channel in the grey area would be addressed by power down modes in the following sections.

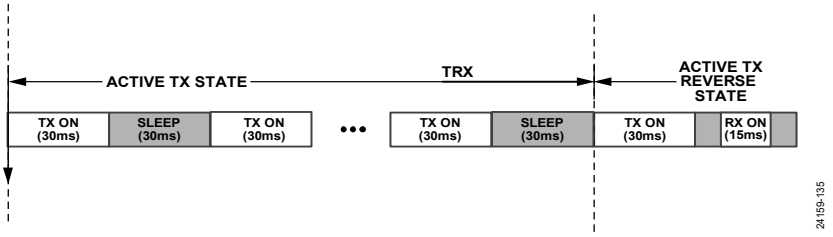


Figure 177. DMR Typical State Transition

Another use case example, if 4 channels ((TX1/TX2/RX1/RX2) are enabled in the profile, user can power down the channels not used temporarily after moving those channels to Calibrated state.

DYNAMIC INTERFRAME POWER SAVING

Dynamic inter-frame power saving is running automatically during all regular TDD TX/RX operations, higher level power down mode can be configured to get more power saving if the application has longer TX/RX transition time. DGPIO pins could be configured to support additional power savings.

There are two power saving choices that can be applied for various TDD interframe scenarios, one is Channel Power Saving and another one is System Power Saving, users can configure either or both of these two options according to their system specification.

Channel Power Saving (CPS)

Channel power saving is to save power on channel granularity for dynamic TDD inter-frame operations. There are two kinds of power saving events triggered by either TX/RX Enable pins or DGPIO pins respectively. The configuration selects power saving modes for both kinds of events. Only power down mode 0-2 can be configured for CPS.

TX_ENABLE/RX_ENABLE Pin Triggers Power Saving

Power saving triggered by TX_ENABLE/RX_ENABLE pin powers up/down based on TX_ENABLE/RX_ENABLE rising or falling edges. TX_ENABLE/RX_ENABLE rising edge powers up the components based on power down mode and falling edge powers down them.

Figure 178 shows TX/RX Enable pin powers up/down channels. If Tx/Rx Enable Pin power down mode is set to mode 1, TX1/RX1 Enable falling edge powers down TX1/RX1 PLL and TX1/RX1 datapath, rising edge powers them up. As mentioned previously, the higher power down mode, the longer recovery time, users should make sure their system has enough transition time between the power down and power up of the same component if users set a high power down mode. For example: if TX1 and RX1 uses the same internal PLL and there is very short transition time between TX enable falling edge and RX enable rising edge, then mode 1 and 2 should not be selected because the same PLL and LDOs are always used.

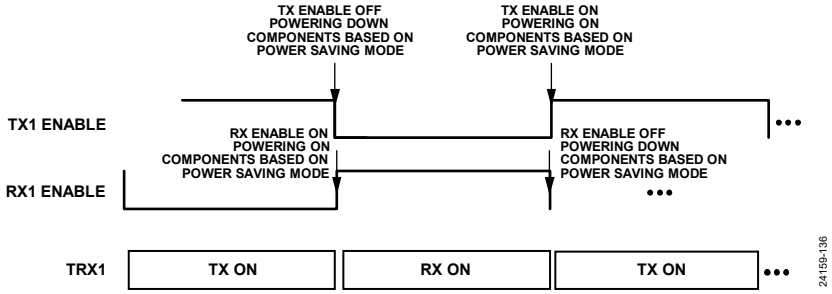


Figure 178. TX/RX Enable Pin triggers Power Saving

DGPIO Triggers Power Saving

DGPIO pin triggered Channel Power Saving can provide additional power saving than the TX/RX Enable pin when it is enabled, therefore if enabled, the power down mode triggered by DGPIO should be larger than TX/RX enable pin triggered power down mode. Both TX and RX channel would be powered down at the DGPIO rising edge and powered up at the DGPIO falling edge, this is because only one DGPIO is assigned for TX and RX channel. Users should be noticed, the DGPIO can only be allowed to pull up when both TX Enable and Rx Enable is low.

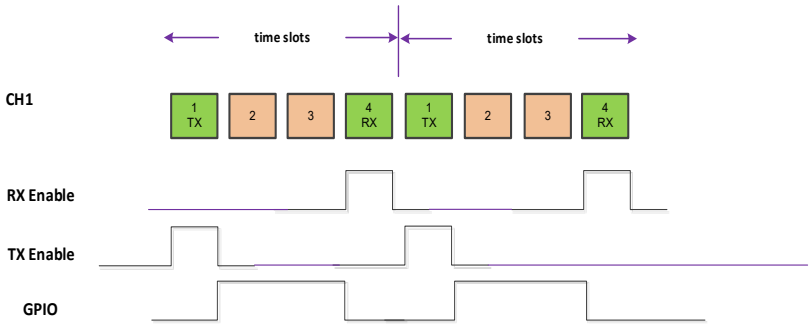


Figure 179. TX/RX Pin Triggers Power Saving and DGPIO Triggers Power-Down Saving

Figure 179 shows an example that both TX/RX enable and DGPIO pin trigger power saving is enabled. The grey time slots are the ones TX/RX must be active. If TX and RX transition time is not long enough to allow power down mode 1 or 2, then users have to select TX/RX Enable pin power down mode to 0. DGPIO power saving can be engaged during time slots 2 and 3 by selecting power down mode 2 to power down both TX/RX LDOs and PLLs in slot 2 and slot 3 areas which neither TX nor RX is active.

The API command adi_adrv9001_powerSavingAndMonitorMode_ChannelPowerSaving_Configure () is used to configure Channel Power Saving modes for a specified channel. It can be called in Calibrated, Primed or RF Enabled state. The new setting would not take effect immediately after mailbox acknowledgment but start at the power down pin edge (Enable falling edge and DGPIO rising edge). Baseband processor should leave enough time to send this command and receive acknowledge before the next power down event.

The channel power saving trigger modes are defined in following data structure:

```
typedef struct adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerSavingCfg
{
    adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerDownMode_e channelDisabledPowerDownMode;
    adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerDownMode_e gpioPinPowerDownMode;
} adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerSavingCfg_t;
```

The enumerator adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerDownMode defines three power down modes that has been described in .

```
typedef enum adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerDownMode
{
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_CHANNEL_MODE_DISABLED = 0, /*!< Default radio
operation, no extra power down */
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_CHANNEL_MODE_RFPLL = 1,    /*!< RF PLL power down */
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_CHANNEL_MODE_LDO = 2,      /*!< Channel LDO power
down */
} adi_adrv9001_PowerSavingAndMonitorMode_ChannelPowerDownMode_e;
```

adi_adrv9001_powerSavingAndMonitorMode_ChannelPowerSaving_Inspect() is used to insect the channel power saving settings for the specified channel.

System Power Saving (SPS)

More power saving can be achieved by System Power Saving but longer transition time. System Power Saving mode will use additional DGPIO pin to trigger the whole ADRV9001 chip into sleep in power saving mode 3-5.

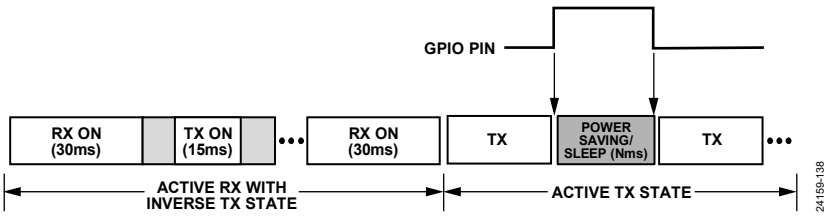


Figure 180. Combined CPS and SPS for Power Saving

Figure 180 shows an example how CPS and SPS combined to achieve the best power savings. User can select channel power saving to TX/RX Enable pin power down mode 2, so TX/RX enable falling edge powers down channel LDOs and TX/RX PLL and rising edge can power them up. After switching to TX only state, although RX channel can be powered down by command adi_adrv9001_Radio_Channel_PowerDown() in Calibrated state, the dark gray area will only have TX LDOs and PLL powered down by TX enable falling edge. Another option is user can power down more by using System Power Saving if the dark gray area is very long. User can set power down mode 3 to 5 to power down most of ADRV9001 components to save power and wake them up by DGPIO falling edge early enough before TX enable rising edge.

Similar with the DGPIO usage in Chanel Power Saving mode, the DPGIO in System Power Mode can only be pulled up when both Tx Enable and Rx Enable is low.

adi_adrv9001_powerSavingAndMonitorMode_SystemPowerSavingMode_Set () is used to set the System Power Saving modes. The enumerator adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerDownMode_e defines three power down modes that described in Table 78:

```
typedef enum adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerDownMode
{
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_SYSTEM_MODE_CLKPLL = 3, /*!< CLK PLL power down */
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_SYSTEM_MODE_LDO = 4,    /*!< LDO power down */
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_SYSTEM_MODE_ARM = 5     /*!< ARM power down */
} adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerDownMode_e;
```

MONITOR MODE

ADRV9001 Monitor mode is designed to do detection and sleep autonomously in idle state which can allow baseband processor to sleep during the whole Idle cycle to get the highest system level power saving. The detection process checks the assigned channel to ascertain if there is a valid signal on the channel of interest to commence communication with other Radios. ADRV9001 provides multiple modes of detection processes.

Figure 181 shows a typical Monitor Mode operation, baseband processor fully controls Monitor mode operation before it enables ADRV9001 into Monitor Mode. First, baseband processor sets the Monitor Mode configuration through an API command. then, baseband processor asserts the Monitor Enable pin (specified by a DGPIO) to move ADRV9001 into Monitor Mode and baseband processor itself can go into sleep state until it's waked up by ADRV9001 or other system interrupt. During the Monitor mode, ADRV9001 fully controls itself to perform the Sleep-Detection cycling, the Sleep/Detection cycle of the ADRV9001 is continuous unless a valid signal

is detected via the configured detection method or it is terminated by baseband processor pulling down the Monitor Enable pin. ADRV9001 will trigger the wake-up pin to wake up baseband processor once the carrier is detected in any detection cycle.

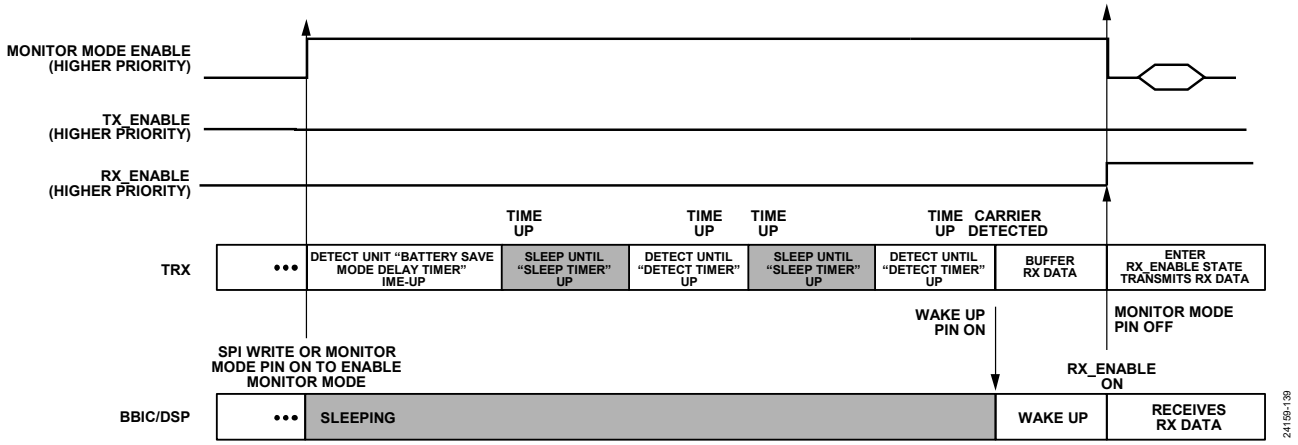


Figure 181. Monitor Mode: Baseband Processor in Sleep and ADRV9001 in Alternate Sleep and Carrier Detection

API command adi_adrv9001_powerSavingAndMonitorMode_SystemPowerSavingAndMonitorMode_Configure () is used to configure the monitor mode, baseband processor should leave enough time to send this command and receive acknowledge before enable Monitor Enable pin. Data structure adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerSavingAndMonitorModeCfg defines the Monitor Mode configuration and is shown as following:

```
typedef struct adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerSavingAndMonitorModeCfg
{
    adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerDownMode_e powerDownMode;
    uint32_t initialBatterySaverDelay_us;
    uint32_t detectionTime_us;
    uint32_t sleepTime_us;
    uint8_t detectionFirst;
    adi_adrv9001_PowerSavingAndMonitorMode_MonitorDetectionMode_e detectionMode;
    bool bbicWakeupLevelEnable;
    bool externalPllEnable;
} adi_adrv9001_PowerSavingAndMonitorMode_SystemPowerSavingAndMonitorModeCfg_t;
```

Monitor modes support there type of system power down modes which are same with the System Power Saving mode. The initial, sleep and detection durations are user configurable, and users can decide detection first or sleep first when ADRV9001 is moved into monitor mode.

The enumerator adi_adrv9001_PowerSavingAndMonitorMode_MonitorDetectionMode_e defines five detection modes which can be used in different radio standards. Some modes are standard dependent, such as the modes with suffix "SYNC" are only available for the DMR standard, and the modes with suffix "FFT" are only available for the standards that use FSK modulation scheme. But the "RSSI" can be used for any radio standards. Current software version(SDK18.0) only supports "RSSI" detection mode and " SYNC" mode, other detection modes will be added in software support in future.

```
typedef enum adi_adrv9001_PowerSavingAndMonitorMode_MonitorDetectionMode
{
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_MONITOR_DETECTION_MODE_RSSI,
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_MONITOR_DETECTION_MODE_SYNC,
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_MONITOR_DETECTION_MODE_FFT,
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_MONITOR_DETECTION_MODE_RSSI_SYNC,
    ADI_ADRV9001_POWERSAVINGANDMONITORMODE_MONITOR_DETECTION_MODE_RSSI_FFT,
} adi_adrv9001_PowerSavingAndMonitorMode_MonitorDetectionMode_e;
```

"RSSI" detection monitor mode is radio standard independent, the configurable "RSSI" threshold and measurement duration can be set by API adi_adrv9001_powerSavingAndMonitorMode_MonitorMode_Rssi_Configure(), a "DETECTED" state will be asserted once the input signal level is beyond the threshold in "DETECTING" state, then the following "Wake Up" procedure will be started.

"SYNC" detection monitor mode is only for DMR standard, the Rx Datapath should be configured in " Frequency Deviation" mode to enable the relative functions of RX Demodulator. The correlator will parallelly detect the SYNC code from the incoming data steam, and up to 14 different SYNC code detection can be supported simultaneously. Similarly, once a SYNC code is detected, ADRV9001 will start the wake up procedure.

ADRV9001 has the option to buffer the latest incoming data in Monitor Mode "Detecting" cycle, once a valid incoming signal is detected and the baseband processor has been waked up by ADRV9001, ADRV9001 can send out the buffered Rx data to baseband processor. This procedure can make sure the baseband processor won't miss the valid incoming signal when it's in the sleep state.

Monitor Mode uses same power down modes with System Power Saving, but additional detection function than System Power Saving, and they can use the same DGPIO as the power saving trigger interface. Users can use either System Power Saving or Monitor Mode, and these two modes can also be dynamically switched for different time slots, a System Power Saving or Monitor Mode API command should be sent during each time switching between System Power Saving.

DIGITAL PREDISTORTION

BACKGROUND

It is well known that one of the main criteria of a power amplifier (PA) operation is its ability to maintain linearity, i.e. the gain is constant regardless of the input amplitude. However, in practice, a PA can only maintain linearity up to a certain input level beyond which the gain starts to lower and the PA enters into a nonlinear or compression region as shown in Figure 182. For most low-power linear amplifiers, they operate in the linear region as shown in the "LINEAR REGION" circle. Unfortunately, a PA that operates mostly in the linear region has lower efficiency. PA efficiency is defined as the ratio of output RF power to the DC supply power. Therefore, it is desirable to operate PA at high efficiency to save DC power and reduce heat dissipation.

To achieve higher PA efficiency, the highest input signal peak is usually set at around 1dB (P1dB) compression region as shown in the "1dB COMPRESSION REGION" circle in Figure 182. However, compression of the peak signals produces harmonics and hence intermodulations. Some of the intermodulations fall back right into or adjacent to the carrier spectrum, therefore not only distorting the transmit signal but also widening the spectrum of the transmit signal, so called spectral regrowth. If left untreated, the error vector magnitude (EVM) performance of the transmit signal would be degraded and the spectral regrowth would interfere adjacent channels, resulting in worse than required adjacent channel power ratio (ACPR) performance. Digital Pre-Distortion (DPD) is designed to mitigate this problem.

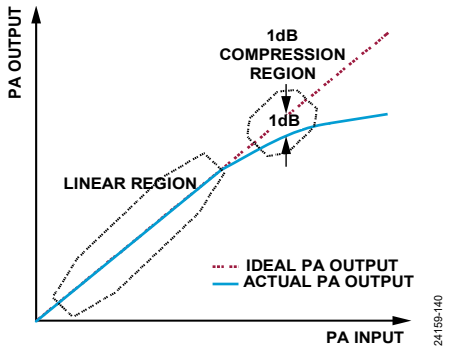


Figure 182. Ideal Power Amplifier Output vs. Actual Power Amplifier Output

ADRV9001 DPD FUNCTION

The ADRV9001 device provides a fully integrated DPD function that supports both narrow-band (NB) and wide-band (WB) applications. It is a hardware/software combined solution which performs linearization of the PA by pre-distorting the digital transmit signal with the inverse of the PA's nonlinear characteristics. After amplifying by the PA, the pre-distortion compensates PA's nonlinearity so the amplified RF transmit signal becomes linear. Therefore, the integrated DPD solution allows PA to operate at very high efficiency while achieving a satisfactory EVM and ACPR performance.

Figure 183 depicts a high level block diagram of the DPD algorithm. As shown in this figure, before the PA, a "Predistortor" block is added in the transmit datapath which distorts the transmit signal d(t) with the inverse of the PA's nonlinear characteristics, as shown by the first Input/Output figure curve. Spectral regrowth is introduced after the pre-distortion. However, after the "pre-distorted" transmit signal x(t) being amplified by the PA, the PA nonlinear characteristics, as shown by the second Input/Output figure curve cancels out the pre-distortion. Therefore, the output of the PA y(t) becomes linear, as shown by the third Input/Output figure curve. In addition, the spectral regrowth after pre-distortion is also corrected. The "DPD Coefficients Computation" block is used to compute the pre-distortion parameters by utilizing "Predistortor" output signal x(t) as well as the power amplifier output signal y(t) through a feedback path. It models the behavior of the PA in the reverse direction, i.e. from output to input, therefore, it characterizes the inverse of the PA nonlinearity and then feeds the parameters to the "Predistortor".

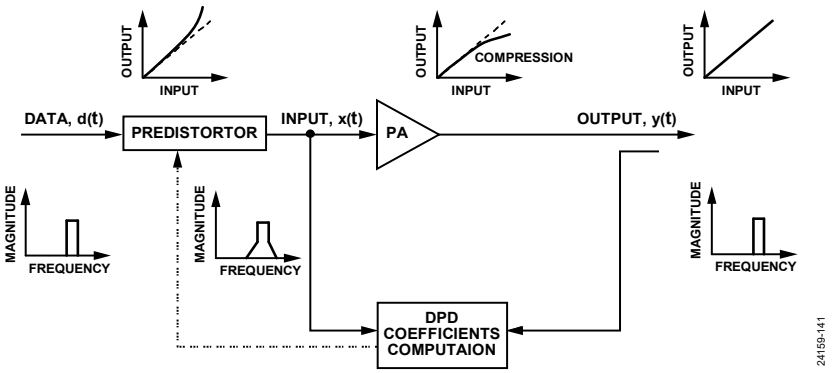


Figure 183. High Level Block Diagram of DPD Algorithm

In the ADRV9001 device, DPD is considered as one of the transmitter tracking calibrations. It is a real-time signal processing with iterative updates to account for hardware variations such as temperature and power level changes. Similar to some other transmitter tracking calibrations, it requires a loopback path from the transmitter to the observation receiver (ORx) to perform the calibration. In this case, an external loop back path (ELB) type 2 is required (please refer to the Receiver/Observation Receiver Signal Chain section for more details about the loopback paths), in which, the transmitter output signal after PA is looped back to the ORx as shown in Figure 184. The user must make sure this path is established before enabling the integrated DPD. In FDD applications where only one Rx is used or in the TDD applications during transmit time slots, unused receiver path can be used to perform DPD calibration as well as some other transmitter tracking calibrations. Please refer to ADRV9001 Example Use Cases section for more details.

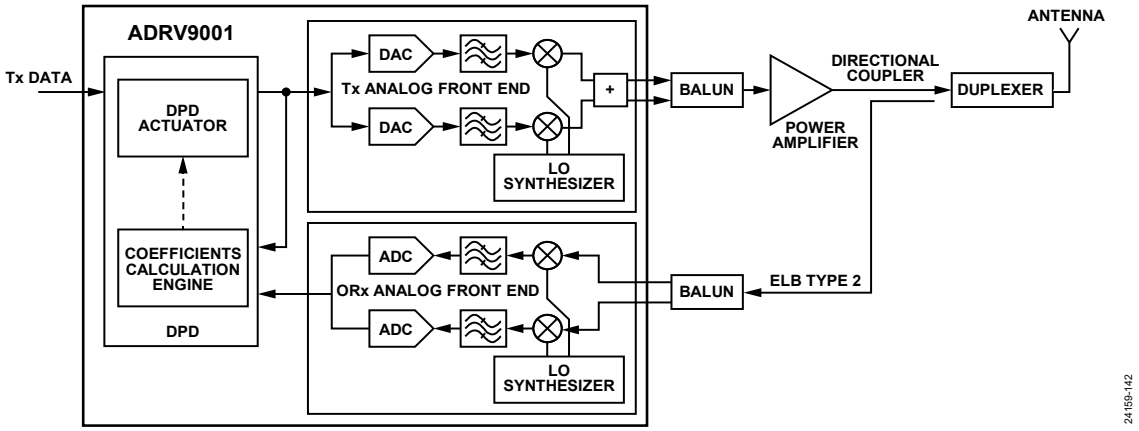


Figure 184. High level Block Diagram of ADRV9001 DPD Implementation

Similar to what is shown in Figure 183, ADRV9001 DPD includes 2 major components, a "DPD Actuator" and a "Coefficients Calculation Engine". The "Coefficients Calculation Engine" computes the DPD coefficients periodically and then updates the "DPD Actuator" for real-time pre-distortion of the transmit signal. The pre-distortion coefficients are associated with polynomial terms defined by the PA model. In order to meet the real-time processing requirement, polynomial terms that are associated with a common time-delay input data are pre-computed and stored into Look-up Tables (LUT) in the "DPD Actuator". In the device, without frequency hopping, 2 LUTs are used for all waveforms, one is currently being active for performing pre-distortion while the other one is being updated at the background to track the changes and replace the current LUT when ready, resulting in seamless transmit operation. "DPD Actuator" also includes a functionality to perform the calculation of the amplitude of the input signal, which is used to search the LUT. The outputs of the LUT are then multiplied with different time delayed input data according to the configured DPD model and combined to form the final pre-distorted transmit data.

ADRV9001 DPD SUPPORTED WAVEFORMS

The integrated DPD supports NB waveforms such as TETRA. Note some NB standard waveforms such as Direct Modulation types with constant envelope do not require DPD. The different modes of operation for TETRA are listed in Table 79. The integrated DPD supports all TETRA 1 and 2 modes.

Table 79. Supported NB Standards and Associated Operational Parameters

Standard	Bandwidth (kHz)	Modulation	Number of Carriers	PAR (dB) before CFR
TETRA1	25	DQPSK	1	3.1
TETRA2	25	4 QAM	8	9.6
TETRA2	25	16 QAM	8	9.5
TETRA2	25	64 QAM	8	10.3
TETRA2	50	4 QAM	16	10.2
TETRA2	50	16 QAM	16	10.3
TETRA2	50	64 QAM	16	10
TETRA2	100	64 QAM	32	11.2
TETRA2	150	64 QAM	48	10.8

Besides that, the integrated DPD also supports some WB LTE and LTE-like waveforms with their associated operational parameters. Other WB waveforms can be supported if the power amplifier behavior fits the designed hardwired amplifier model, as well as the sampling rates and transceiver bandwidth.

The WB LTE standards supported and their associated operation parameters are summarized in Table 80.

Table 80. Supported WB Standards and Associated Operational Parameters

LTE Bandwidth (MHz)	Number of Carriers	PAR (dB) Before CFR
1.4	Multicarrier	~11
3	Multicarrier	~11
5	Multicarrier	~11
10	Multicarrier	~11
15	Multicarrier	~11
20	Multicarrier	~11

In theory, DPD can support any profile with a RF signal bandwidth less than 1/5 of the ADRV9001 system frequency. For example, if the ADRV9001 has a system frequency of 184.32MHz, DPD can support a signal bandwidth if it is less than 36.864MHz.

As shown in Table 79 and Table 80, multicarrier TETRA2 has a Peak-to-Average Ratio (PAR) between 9.6 dB to 11.2 dB and multi-carrier LTE signal typically has a PAR of about 11dB. To achieve higher PA efficiency and DPD algorithm stability, a waveform with a large PAR is expected to have crest factor reduction (CFR) performed in baseband processor before DPD operation. It is important that CFR is applied to a multicarrier signal before transmission since it keeps the average power higher while maintaining the same peak back off in the digital transmit data. Also, CFR suppresses large peaks to below a preset threshold therefore eliminating occasional large peaks that could make DPD unstable if the peak is beyond the compression threshold limit of the PA. For example, an LTE signal has a PAR of about 11 dB and the PAR can be reduced by CFR to be about 7 dB by trading off EVM. Note it is the responsibility of baseband processor to perform CFR with an appropriate tradeoff between PAR reduction and EVM degradation before sending the transmit data to ADRV9001. It should be also noted that additional EVM degradation caused by the integrated DPD is negligible compared to the degradation caused by CFR.

DPD WITH FREQUENCY HOPPING (FH)

ADRV9001 also supports DPD operation during frequency hopping (FH). In this case, user could identify multiple frequency regions and for any LO frequency in one region, the same DPD solution is applied. ADRV9001 allows a maximum of 8 frequency regions. User could define each region by specifying the lower bound and upper bound LO frequencies ([lower bound, upper bound)) for up to 7 regions and the rest LO frequencies are in the 8th region.

ADRV9001 DPD PERFORMANCE

The integrated DPD algorithm has been tested using both MOS type and GaN type of PAs. As an example, Figure 185 and Figure 186 shows the AM-AM and AM-PM performance of the raw transmit signal input versus the MOS type of PA output before and after DPD without FH. The test waveform is TETRA1. From Figure 186, it can be seen clearly that the PA nonlinearity is successfully corrected by the integrated DPD.

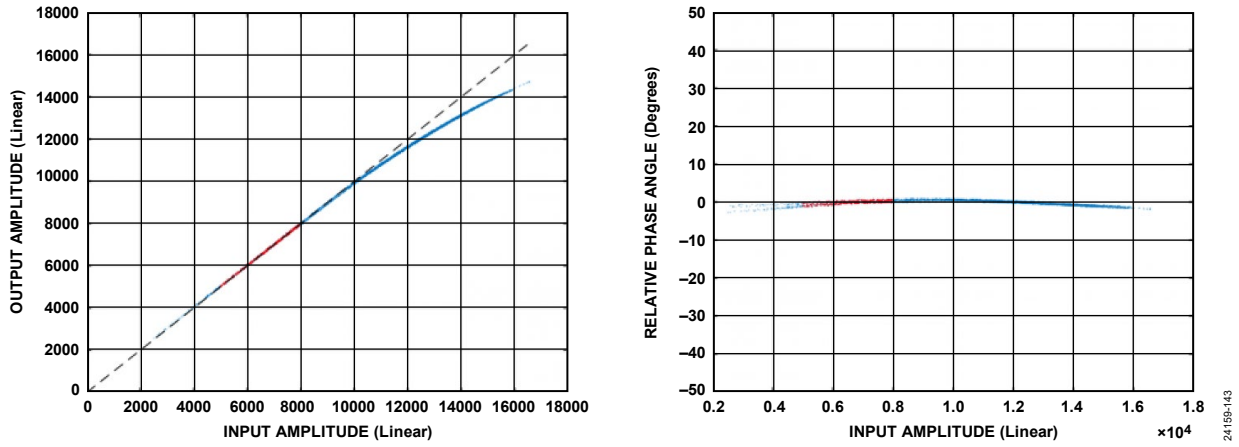


Figure 185. Raw Transmit Signal Input vs. Nonlinearized Power Amplifier Output

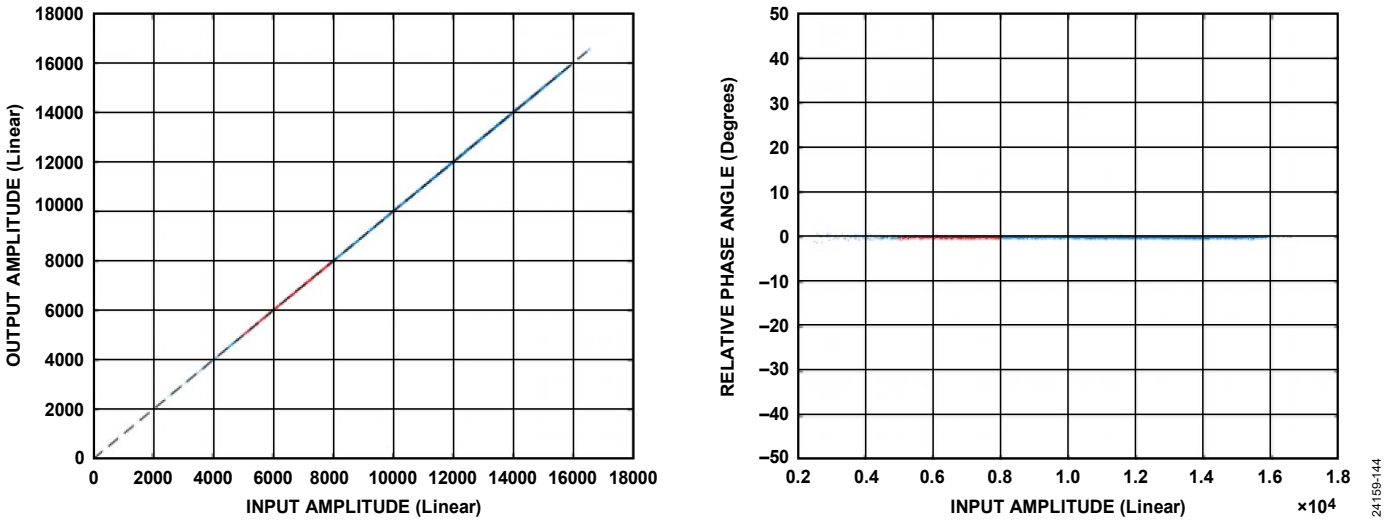


Figure 186. Raw Transmit Signal Input vs. Linearized Power Amplifier Output

Figure 187 shows an example ACPR performance before and after DPD. The blue curve represents the ACPR performance before DPD, from which, spectral regrowth could be observed. The black curve represents the ACPR performance after DPD. It is obvious that the ACPR performance is significantly improved.

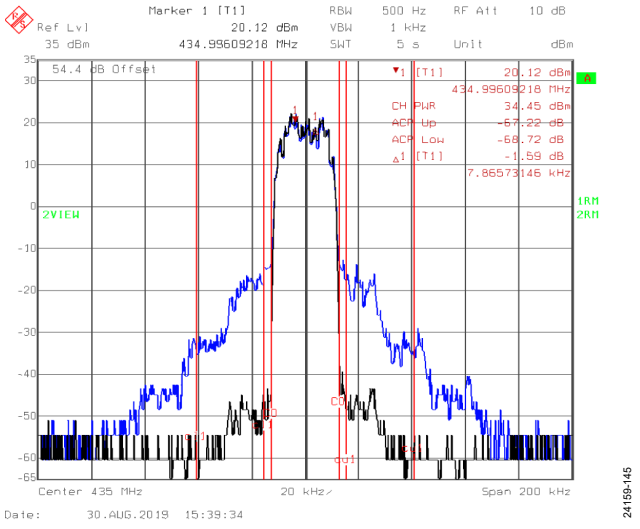


Figure 187. ACPR Performance Before and After DPD

Satisfactory DPD performance also depends on successful completion some operations inside the device such as ADC and DAC calibrations. Most of those operations are guaranteed internally in the device but the following operations are user configurable. It is important to perform all those optional operations to achieve the optimal DPD performance.

- Time alignment between transmit x(t) and loopback y(t) for data capture
- Transmit closed loop gain control (CLGC) tracking calibration
- Transmit LOL calibration
- Transmit and Receive QEC calibration

CLOSED LOOP GAIN CONTROL (CLGC)

The ADRV9001 provides CLGC as one of the transmit tracking calibration algorithms. The sole purpose of CLGC is to maintain a fixed gain between the PA output and transmit input digital amplitude. The main source of gain variation comes from the changes in the gain of the PA, which could vary due to transistor temperature change in response to power output in the short term, RF carrier frequency change within the bandwidth of the amplifier, and slow hardware degradation in the long term. Maintaining a fixed gain in the transmit path is important to maintain a precise calibrated transmit power at the antenna port.

CLGC helps to achieve a fixed gain target by adjusting the transmit attenuation automatically through monitoring the gain variation of PA. It can work with or without the DPD algorithm. When DPD is not activated, CLGC is an optional feature. When DPD is activated, CLGC should always be activated. Note ADRV9001 allows to enable DPD without CLGC but it is not recommended. As discussed earlier, DPD compensates the instantaneous compression of peak signal by expanding the input signal so that the peak signals are linearized. When linearization is achieved, the gain in the compression region is adjusted (increased) to match the gain of the lower linear region, so that the overall gain is independent of the input or output average power level. This has the advantage of having CLGC to focus on compensating only the PA gain variation mainly due to temperature change. By setting a proper gain target, CLGC could also help to monitor and limit the transmit power level to keep the amplifier output power from rising beyond the linearization capable power limit of the PA. Therefore, it is crucial to always enable CLGC while DPD is active.

As the first step of CLGC, user should set up a target transmit gain. This could be measured through ADRV9001 by using the "CLGC Loop Open" method. The detailed steps of measuring target gain will be discussed later. After the measurement, ADRV9001 provides user both an unfiltered and a filtered transmit gain value. Based on those, user could further adjust the value and set a proper gain target, then close the loop and start the ADRV9001 CLGC algorithm to continuously track the gain variation based on the determined gain target.

When DPD is active, the PA gain is defined as the gain in the linear region of the AM-AM curve as shown in Figure 185. To estimate this reference gain, data samples are usually selected in the upper linear region and below the compression region, as indicated in red in Figure 185. Note this same gain plus relative phase is used by DPD to scale the y(t) loopback data to match the pre-distorted transmit x(t) data. The PA gain derived from data between these bounds represents the real gain in the linear region of the amplifier, while excluding the distorted gain at the upper end due to compression. However, after DPD has converged, the gain in the compression region will increase to match the gain in the lower region. When DPD is off, compression at the top region will not be corrected by DPD, hence it is necessary to include all samples for integration to estimate the total power, including the compressed region, to define the transmit gain. User could define the region for calculating the gain through API configurations.

Similar as the DPD algorithm, CLGC algorithm requires the time alignment between transmit x(t) and loopback y(t) for data capture. User should measure the delay and provide it to ADRV9001 which is especially important for WB profiles. When DPD is enabled, the same delay measurement serves both DPD and CLGC algorithm.

Note that ADRV9001 CLGC algorithm has a limit to track gain variations not exceeding ±3dB (this should be able to accommodate most types of PA) and for each CLGC iteration, the maximum gain adjustment is limited to ±0.5 dB to prevent DPD algorithm becoming unstable.

DPD/CLGC CONFIGURATION

To use the integrated DPD/CLGC properly and ensure optimal performance, user must configure DPD/CLGC parameters properly. This could be done through ADRV9001 Transceiver Evaluation Software (TES) or Software Development Kit (SDK). The configuration consists of 2 sets of DPD/CLGC parameters. The first set of DPD/CLGC parameters is "pre initial calibration" parameters since they should be configured before performing initial calibration when the device is at the "STANDBY" state. The second set of DPD/CLGC parameters is "post initial calibration" parameters since they should be configured after performing initial calibration when the device is at the "CALIBRATED" state. These DPD/CLGC parameters will be explained in details in the next two subsections.

DPD/CLGC Pre Initial Calibration Parameters Configuration

In order to properly set the pre initial calibration parameters of DPD/CLGC, the user should have a general understanding of the DPD model used in the device. The DPD model is described by the following equations:

$$x(n) = \sum_{t=0}^{T-1} \psi_t(|d(n - l_t)|)d(n - k_t)$$

$$\psi_t(|d(n - l_t)|) = \sum_{i=0}^{7} b_{t,l_t,i}\, a_{t,l_t,i}\, |\, d(n - l_t)\,|^i$$

where:

T is the total number of taps in the DPD model.

$\psi_t(|d(n - l_t)|)$ is the function implemented by the LUT for tap, t. "t"

l_t and k_t are part of the hardware model, representing the amplitude and data delay, respectively. The user can optionally include/exclude each individual power term in $\psi_t(|d(n - l_t)|)$ by controlling the corresponding $b_{t,l_t,i}$ setting it to either 0 for excluding or 1 for including), to better model their power amplifier.

$a_{t,l_t,i}$ are coefficients that are estimated by the coefficients calculation engine and used to generate the LUTs by the DPD actuator. For $b_{t,l_t,i}$ and $a_{t,l_t,i}$, the subscripted t represents the index for the tap, l_t represents the amplitude delay, and i represents the order of the power term. ADRV9001 only supports 0th to 7th order power term in the function $\psi_t(|d(n - l_t)|)$.

As aforementioned, this set of DPD/CLGC parameters must be configured before initial calibration. It is defined by the following API data structure:

```
typedef struct adi_adrv9001_DpdInitCfg
{
    bool enable;
    adi_adrv9001_DpdAmplifier_e amplifierType;
    adi_adrv9001_DpdLutSize_e lutSize;
    adi_adrv9001_DpdModel_e model;
    bool changeModelTapOrders;
    uint32_t modelOrdersForEachTap[4];
    uint8_t preLutScale;
    uint8_t clgcEnable;
} adi_adrv9001_DpdInitCfg_t;
```

Table 81 briefly summarizes all the DPD/CLGC pre initial calibration parameters described in the above data structure.

Table 81. DPD Pre Initial Calibration Parameters

Parameter	Type	Description	Default	Note
enable	bool	Sets "TRUE" to place the "DPD Actuator" in the datapath on the specified channel to prepare for DPD operation.	FALSE	Set "TRUE" does not start the DPD operation. DPD starts when the corresponding tracking calibration bit is set.
amplifierType	enum	Selects the type of amplifier ADI_ADRV9001_DPD_AMPLIFIER_NONE = 0, ADI_ADRV9001_DPD_AMPLIFIER_DEFAULT =1, ADI_ADRV9001_DPD_AMPLIFIER_GAN =2	1	"1" is the only allowed power amplifier type currently for both MOS type and GaN type of PA.
lutSize	enum	Determines the LUT size ADI_ADRV9001_DPD_COMPANDER_SIZE_256 = 0, ADI_ADRV9001_DPD_LUT_SIZE_512 = 1	1	Only 2 LUT sizes are supported currently.
model	enum	Selects the DPD model. ADI_ADRV9001_DPD_MODEL_0 = 0, ADI_ADRV9001_DPD_MODEL_1 = 1, ADI_ADRV9001_DPD_MODEL_3 = 3, ADI_ADRV9001_DPD_MODEL_4 = 4, Model 4 is the ADRV9001 Model.	4	"4" is the only allowed DPD model currently. User should always choose "4".

Parameter	Type	Description	Default	Note
changeModelTapOrders	bool	Sets "TRUE" to use the model tap orders defined by "modelOrdersForEachTap". Set "FALSE" to ignore "modelOrdersForEachTap" and use the default order.	FALSE	The default model tap order for DPD Model 4 is: [0] = 0x001F, [1] = 0x007F, [2] = 0x001F, [3] = 0x001E
modelOrdersForEachTap	array	A bit map for each of the taps in a model to indicate which power terms are included in the model. Tap 0 and 2 should have the same order.	[0] = 0x001F, [1] = 0x007F, [2] = 0x001F, [3] = 0x001E	The default bit map represents the default tap order for Model 4.
preLutScale	uint8_t (U2.2)	Describes the prescaling factor for the LUT.	2.0	Min=1, Max=3.75
clgcEnable	bool	Enable CLGC functionality.	False	Set "TRUE" does not start the CLGC operation. CLGC starts when the corresponding tracking calibration bit is set.

Each of these parameters are described in more details as the following:

enable, clgcEnable

The "enable" parameter is used to place the "DPD Actuator" on the datapath of the specified channel to prepare for DPD operation. The "clgcEnable" parameter is used to enable CLGC functionality. This could be done through TES under the "Advanced Features" tab as shown in Figure 188. Note when DPD is enabled, CLGC should also be enabled.

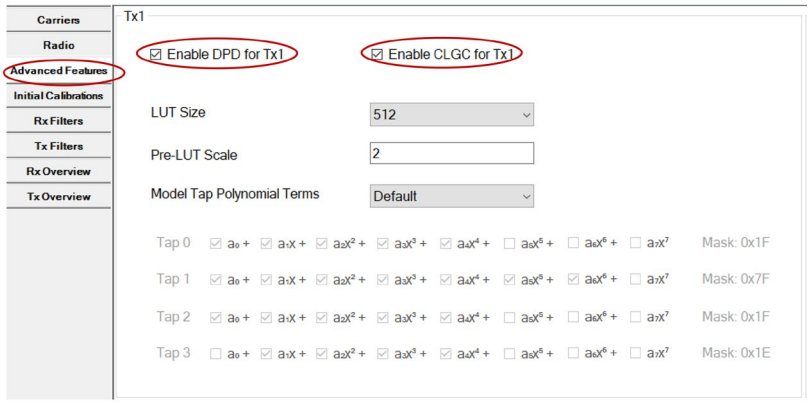


Figure 188. TES Configuration for Placing the DPD Actuator and Enable CLGC Functionality in the Data Path

Note that setting it to be "TRUE" is necessary but not sufficient to start DPD/CLGC operation. The DPD/CLGC starts when the corresponding tracking calibration bit is also set, as shown in Figure 189.

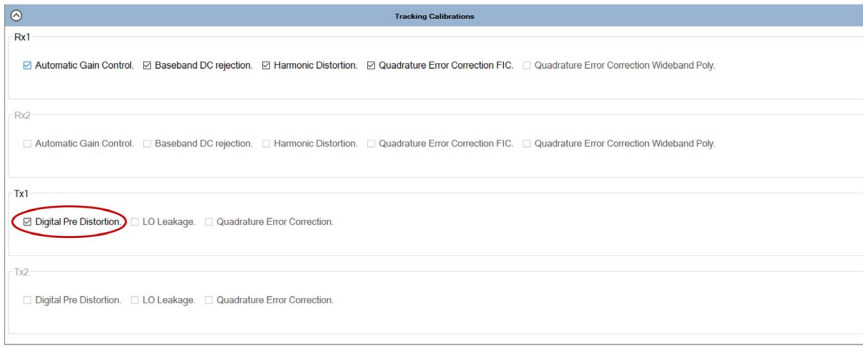


Figure 189. TES Configuration for Enabling DPD/CLGC Tracking Calibration

For DPD/CLGC to work, the profile must indicate that there exists an external loopback connection and an external PA for this channel. This can be done by setting "Board Configurations" in TES properly, as shown in Figure 190.

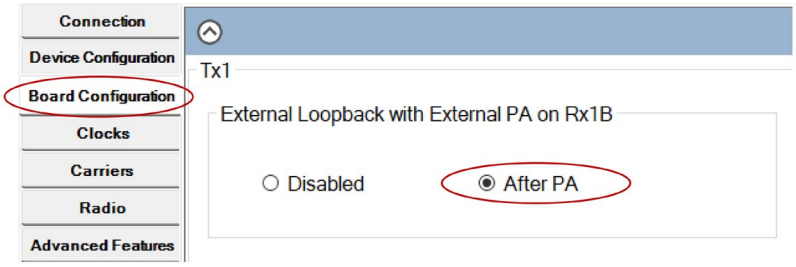


Figure 190. TES Configuration for External Loopback with External Power Amplifier

amplifierType

Currently, the PA type should always be set to ADI_ADRV9001_DPD_AMPLIFIER_DEFAULT if DPD is enabled. The default PA type is referring to both MOS and GaN type of PA. This field will be cleaned up in future releases.

lutSize

Currently, the supported LUT sizes are 256 and 512. This size determines the number of entries in the DPD LUT. A larger number of entries provides better LUT granularity.

model

Currently, the model should be set to ADI_ADRV9001_DPD_MODEL_4 only. The other models exist for backwards compatibility with other transceivers and should not be used at this time. Model 4 consists of four taps (T=4) which can be described by the following equation:

$$x(n) = \sum_{t=0}^{3} \psi_t(|d(n-l_t)|)d(n-k_t)$$

Delays for each tap are described in Table 82.

Table 82. Delays of DPD Model 4

Tap	Delay of Data (k)	Delay of Magnitude (l)
0	0	0
1	1	1
2	2	2
3	1	2

Based on Table 82, the equation could be rewritten as:

$$x(n) = \psi_0(|d(n)|)d(n) + \psi_1(|d(n-1)|)d(n-1) + \psi_2(|d(n-2)|)d(n-2) + \psi_3(|d(n-2)|)d(n-1)$$

where $\psi_0(|d(n)|)$, $\psi_1(|d(n\text{-}1)|)$, $\psi_2(|d(n\text{-}2)|)$ and $\psi_3(|d(n\text{-}2)|)$ represent four taps, generated by the LUT.

DPD Model 4 tap configuration used to generate the final pre-distorted data x(t) is shown in Figure 191:

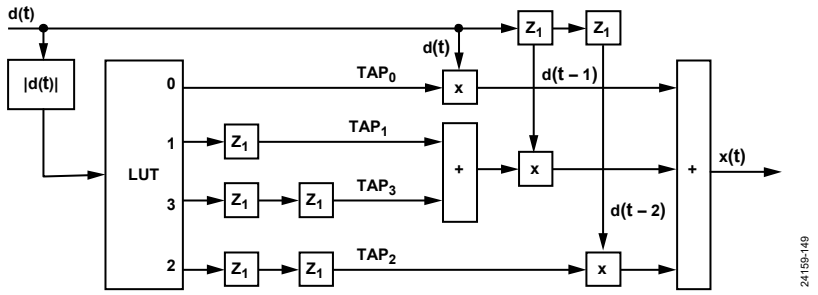


Figure 191. ADRV9001 DPD Model 4 LUT Configuration

As shown in Figure 191, d(t) is the raw complex transmit signal before predistortion. Its amplitude is the basis that the DPD actuator uses to predistort the d(t) via its L UT. The LUT consists of four taps, which are calculated with precomputed DPD coefficients α, as the following:

$$TAP_0 = a_{0,0,0} + a_{0,0,1}|d(t)| + a_{0,0,2}|d(t)|^2 + a_{0,0,3}|d(t)|^3 + a_{0,0,4}|d(t)|^4$$

$$TAP_1 = a_{1,1,0} + a_{1,1,1}|d(t-1)| + a_{1,1,2}|d(t-1)|^2 + a_{1,1,3}|d(t-1)|^3 + a_{1,1,4}|d(t-1)|^4 + a_{1,1,5}|d(t-1)|^5 + a_{1,1,6}|d(t-1)|^6$$

$$TAP_2 = a_{2,2,0} + a_{2,2,1}|d(t-2)| + a_{2,2,2}|d(t-2)|^2 + a_{2,2,3}|d(t-2)|^3 + a_{2,2,4}|d(t-2)|^4$$

$$TAP_3 = a_{3,2,1}|d(t-2)| + a_{3,2,2}|d(t-2)|^2 + a_{3,2,3}|d(t-2)|^3 + a_{3,2,4}|d(t-2)|^4$$

Note the TAPx equations represent the default power term setting for each tap in Model 4, from which, $b_{t,l_t,i}$ can be easily derived as the following depending if a power term is included or excluded:

Tap 0: $b_{0,0,0} = 1$, $b_{0,0,1} = 1$, $b_{0,0,2} = 1$, $b_{0,0,3} = 1$, $b_{0,0,4} = 1$, $b_{0,0,5} = 0$, $b_{0,0,6} = 0$, $b_{0,0,7} = 0$

Tap 1: $b_{1,1,0} = 1$, $b_{1,1,1} = 1$, $b_{1,1,2} = 1$, $b_{1,1,3} = 1$, $b_{1,1,4} = 1$, $b_{1,1,5} = 1$, $b_{1,1,6} = 1$, $b_{1,1,7} = 0$

Tap 2: $b_{2,2,0} = 1$, $b_{2,2,1} = 1$, $b_{2,2,2} = 1$, $b_{2,2,3} = 1$, $b_{2,2,4} = 1$, $b_{2,2,5} = 0$, $b_{2,2,6} = 0$, $b_{2,2,7} = 0$

Tap 3: $b_{3,2,0} = 0$, $b_{3,2,1} = 1$, $b_{3,2,2} = 1$, $b_{3,2,3} = 1$, $b_{3,2,4} = 1$, $b_{3,2,5} = 0$, $b_{3,2,6} = 0$, $b_{3,2,7} = 0$

If using an array B for 4 taps and for each tap using a byte to represent the above setting (the least significant bit represents the 0th power term), it is clear that the default setting is equivalent to B[0] = 0x1F, B[1] = 0x7F, B[2] = 0x1F and B[3] = 0x1E.

The connections from the 4 outputs are combined to produce the final output, x(t) as the following:

$$x(t) = TAP_0[|d(t)|] \times d(t) + \{TAP_1[|d(t-1)|] + TAP_3[|d(t-2)|]\} \times d(t-1) + TAP_2[|d(t-2)|] \times d(t-2)$$

changeModelTapOrders

This flag is used to provide user an option to select the default model tap orders or choose a customized model tap orders. If this flag is set to be "TRUE", the next field in the data structure "modelOrdersForEachTap", should be used to set the model tap orders for the specified channel. If it is "FALSE", then "modelOrdersForEachTap" will be ignored and it will use the default tap orders as discussed (B[0] = 0x1F, B[1] = 0x7F, B[2] = 0x1F and B[3] = 0x1E).

modelOrdersForEachTap

This is an array of bitmaps $b_{t,l_t,i}(i = 0\ to\ 7)$ for each tap t (t=0 to 3), formulated in the same way as discussed above for the default setting. It provides user an option to customize the order so that a power term could be included or excluded in the polynomial to better model the PA. Table 83 shows recommendations for setting this field. The user could try those suggestions and find out the best model through tests. The method of selecting the best model tap orders is discussed in the DPD Tuning and Testing section as a part of DPD tuning recommendations.

Table 83. Suggested Model Orders for Narrow-Band Waveforms

Taps	Model Orders for Each Tap
Tap 1	B[1] = 0x1F, 0x3F, 0x7F, 0xFF
Tap 0 and Tap 2	B[0] = B[2] = 0x03, 0x07, 0x0F, 0x1F, (Tap 0 and Tap 2 should be the same.)
Tap 3	B[3] = 0x0, 0x02, 0x06, 0x0E, 0x1E, 0x3E

The user could configure the changeModelTapOrders and modelOrdersForEachTap through TES, as shown in Figure 192 and Figure 193 . Figure 192 shows the default model tap configuration and Figure 193 shows a customized model tap configuration which is equivalent to

B[0] = 0x07, B[1] = 0x7F, B[2] = 0x07 and B[3] = 0x06. (Note Tap 0 and Tap 2 should always be the same. For simplicity GUI uses X to represent $|d(n - l_t)|$)

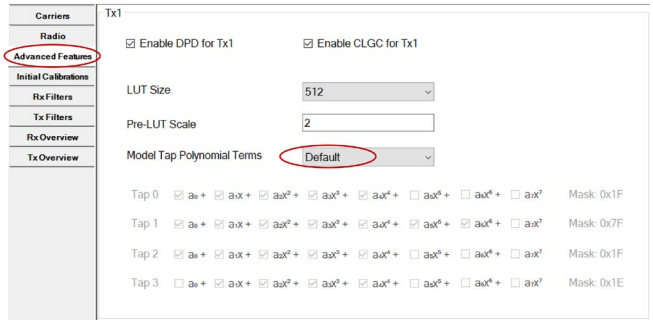


Figure 192. Configuring Default Model Tap Order Through TES

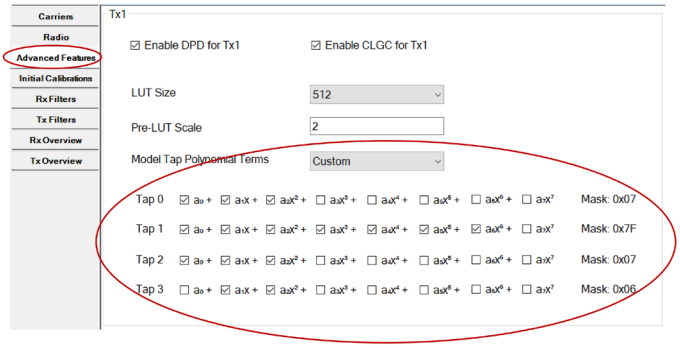


Figure 193. Configuring Customized Model Tap Order Through TES

preLutScale

This value, given as a fixed point U2.2 number, sets the scaling factor before searching the LUT. The scaling factor can be set as 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3, 3.25, 3.5 and 3.75. This allows the user to scale the input signal magnitude in order to cover close to the full range of the LUT for better DPD performance. If the signal input to the compander is too small, then only part of the LUT is used. When the input signal is small, user could try different scaling factors to increase the signal level which might improve the DPD performance. The scaling factor can be determined according to the dBFS of the input data peak. As an example, if the signal peak power is less than −4 dBFS, the scaling factor 3.5 could be applied. As shown in Figure 192 and Figure 193, the default value of "pre-LUT Scale" is 2, which could be further changed by user.

DPD/CLGC Post Initial Calibration Parameters Configuration

The second set of DPD/CLGC parameters should be configured after initial calibration. It is defined by the following data structure:

```
typedef struct adi_adrv9001_DpdCfg
{
    uint32_t numberOfSamples;
    uint32_t additionalPowerScale;
    uint32_t rxTxNormalizationLowerThreshold;
    uint32_t rxTxNormalizationUpperThreshold;
    uint32_t detectionPowerThreshold;
    uint32_t detectionPeakThreshold;
    uint16_t countsLessThanPowerThreshold;
    uint16_t countsGreaterThanPeakThreshold;
    bool immediateLutSwitching;
    bool useSpecialFrame;
    bool resetLuts;
    uint32_t timeFilterCoefficient;
```

```
    uint32_t dpdSamplingRate_Hz;
    uint8_t clgcLoopOpen;
    int32_t clgcGainTarget_HundredthdB;
    uint32_t clgcFilterAlpha;
    int32_t clgcLastGain_HundredthdB;
    int32_t clgcFilteredGain_HundredthdB;
} adi_adrv9001_DpdCfg_t
```

Table 84 briefly summarizes all the DPD/CLGC post initial calibration parameters described in the data structure.

Table 84. DPD/CLGC Post Initial Calibration Parameters

Parameter	Type	Description	Min	Max	Default	Note
numberOfSamples	uint32_t	Specifies the number of samples to use for each iteration of DPD computation.	1024	4096	4096	The maximum value is preferred.
additionalPowerScale	uint32_t	Provides an estimate of the standard deviation of the modem input data magnitude to scale the data for internal DPD computation.	0	2^{32-1}	4	
rxTxNormalizationLowerThreshold	uint32_t (U2.30)	Signal power for the lower threshold for the normalization of the magnitude and phase of the RX and TX data	0	1.0	0.0031622776602 (−25 dBFS)	
rxTxNormalizationUpperThreshold	uint32_t (U2.30)	Signal power for the upper threshold for the normalization of the magnitude and phase of the RX and TX data	0	1.0	0.031622776602 (-15 dBFS)	
detectionPowerThreshold	uint32_t (U1.31)	Power threshold used for invalid capture detection				
detectionPeakThreshold	uint32_t (U1.31)	Peak threshold used for invalid capture detection				
countsLessThanPowerThreshold		If the number of samples below the detectionPowerThreshold exceeds this number, the capture is discarded.				To disable the detection, set it to 4096
countsGreaterThanPeakThreshold		If the number of samples above the detectionPeakThreshold is less than this number, the capture is discarded.				To disable the detection, set it to 0
immediateLutSwitching	bool	Determines whether the LUT switches immediately or at the end of Tx data frame.			TRUE	FALSE not currently supported.
useSpecialFrame	bool	DPD only runs on a user indicated special frame.			FALSE	Currently not supported.
resetLuts	bool	Reset LUTs so that no pre-distortion is applied.			FALSE	User should reset LUTs at the start of DPD operation.
timeFilterCoefficient	uint32_t	Coefficient of a time filter to remove spectral spikes from LUT swiching.	0	1.0	0	It helps if there are spectral spikes from LUT

Parameter	Type	Description	Min	Max	Default	Note
						switching, but could hurt convergence. A coefficient of zero means no filtering.
dpdSamplingRate_Hz	uint32_t	Sampling rate in Hz for the DPD actuator and capture.				Read only. No effect on DPD configuration.
clgcLoopOpen	uint8_t	Open or close the gain loop.			0	If true, the loop is open and the TX attenuators are not updated. Used to measure a target gain.
clgcGainTarget_HundredthdB	int32_t	Set by user as the gain target.				
clgcFilterAlpha	uint32_t	Gain filter coefficient.	0	1	0.75	When it is 0, the filter is disabled.
clgcLastGain_HundredthdB	int32_t	Unfiltered gain measured when loop is open				Only valid for user to retrieve.
clgcFilteredGain_HundredthdB	int32_t	Filtered gain measured when loop is open				Only valid for user to retrieve.

Each of these parameters are described in more details as the following:

numberOfSamples

It specifies the number of samples used per DPD/CLGC data capture with a limit of 4096. In general, the DPD/CLGC performance could be improved if more samples are used. For TETRA1, currently, the Linearization Channel (LCH) is not supported so the numberOfSamples should be set to 4096. When LCH is supported, the numberOfSamples could be changed to a different value. More information will be provided in the future.

For LTE, the Number of Sample should be set to 4096 without frequency hopping.

additionalPowerScale

This parameter is used to scale the higher power terms during the calculation of the auto-correlation matrix using transmit data d(n). It is used to keep the nominal magnitude of each of the power terms about the same to avoid ill condition of the correlation matrix. The scaling factor α could be defined as $\alpha \approx 2 \times std(d(n))$, where "std" stands for standard deviation. User could measure α and then pass the information through this parameter.

rxTxNormalizationLowerThreshold/rxTxNormalizationUpperThreshold

These are required parameters for the normalization of the magnitude and phase of the receive and transmit data. These thresholds are used to pick a linear region for normalizing the data. The region chosen should be below the compression point but above the noise. In the AM-AM and AM-PM plots shown in Figure 185, a possible choice of the linear region is highlighted in red. In general, rxTxNormalizationUpperThreshold should be set to 0.5 of the peak signal amplitude and rxTxNormalizationLowerThreshold should be set to 0.3 of the peak signal amplitude. Once they are set, the threshold values should be fixed and not vary from capture to capture. Therefore, by knowing the peak transmit signal in dBFS, rxTxNormalizationUpperThreshold should be set to "peak Tx dBFS – 6 dB" and rxTxNormalizationLowerThreshold should be set to "peak Tx dBFS – 10.5 dB". Peak transmit signal is usually set at P1dB by adjusting the Tx attenuation setting.

The user could enter those thresholds in dBFS through TES. If using API, the linear numbers should be used which can be calculated as $10^{(threshold\_dBFS/10)}$.

detectionPowerThreshold

It is used to detect an invalid data capture for DPD/CLGC operation if a specified number of samples (countsLessThanPowerThreshold) are below the defined power threshold.

detectionPeakThreshold

It is used to detect a valid data capture for DPD/CLGC operation if a specified number of samples (countsGreaterThanPeakThreshold) are greater than the defined peak threshold. (DPD/CLGC operation needs a good set of large signal samples to model the PA compression behavior properly.)

countsLessThanPowerThreshold

It defines the number of samples below the detectionPowerThreshold to determine an invalid capture. To disable it, user can set it to be the DPD/CLGC maximum number of capture samples 4096.

countsGreaterThanPeakThreshold

It defines the number of samples greater than the detectionPeakThreshold to determine an valid capture. To disable it, user can set it to be 0.

immediateLutSwitching

When a new DPD solution is formed, the new solution is loaded into a spare LUT, which can then be switched with the active LUT. There are two options regarding the LUT switching. When immediateLutSwitching is set to be "TRUE", the new LUT swaps out the active LUT immediately when ready. When immediateLutSwitching is set to be "FALSE", after the updated LUT is available, LUT swapping is triggered after the next data frame is completed. Note this only applies to TDD operations. FDD systems should always use immediate LUT switching. Currently, it should always set to be "TRUE".

useSpecialFrame

In order to achieve optimal performance, DPD must capture the peaks of the signal. Some standards allow for transmission of special data that can be used for DPD estimation. In other cases, the baseband processor may know in advance that a frame will contain good data for DPD estimation. In these cases, the user may choose to control which frames are used for DPD. The flag indicates that the user will control the frames that DPD can capture. This is currently not supported so it should be disabled.

resetLuts

To start DPD operation from a known state, user should reset LUTs. By setting resetLuts to be 1, it sets most polynomial terms to be 0 to remove the pre-distortion at the beginning of DPD operation.

timeFilterCoefficient

This parameter defines the coefficient of a single pole filter which can help to mitigate the spectral spikes caused by LUT switching, especially in FDD and NB use cases. In TDD, user could avoid spectral spikes by not performing immediate LUT switch. The range of "timeFilterCoefficient" is between 0 and 1. If setting it to be 0, it is equivalent to disable this functionality. User can experiment this parameter and pick the optimal value to reduce the spikes. User should also note that enabling this feature might hurt the convergence time. The bigger this parameter is, the slower the convergence might be. Therefore, this method should be considered as an available tool when all other methods are exhausted and be employed with caution.

dpdSamplingRate_Hz

This parameter is for showing the sampling rate in Hz for the DPD actuator and capture. It is read only and does not have any effect on DPD configurations.

clgcLoopOpen

This parameter is used to open or close the gain control loop. When it is set to 1, the transmit attenuators are not updated. This is used to measure the gain to help determine the target gain. Once target gain is set, user should set it to 0 to close the loop to start CLGC operation.

clgcGainTarget_HundredthdB

This parameter should be configured by user to notify ADRV9001 about the gain target for CLGC in an accuracy of hundredth of a dB.

clgcFilterAlpha

This parameter stands for the coefficient of a single pole filter to smooth the gain measurement. The min value is 0 which is equivalent to disable this filter by using the instantaneous gain measurement result. The max value is 1 and the default value is 0.75. User could set bigger value to achieve smoother gain measurement results.

clgcLastGain_HundredthdB

This parameters represents the unfiltered gain measurement results when loop is open. User could use API to retrieve this value which is in a accuracy of hundredth of a dB. Note this parameter is only valid for retrieving.

clgcFilteredGain_HundredthdB

This parameters represents the filtered gain measurement results when loop is open (based on the filter coefficient user configured). User could use API to retrieve this value which is in a accuracy of hundredth of a dB. Note this parameter is only valid for retrieving.

User could configure the DPD/CLGC post initial calibration parameters through TES as shown in Figure 194. Note all data capture related configurations such as "Number of Samples", "Rx/Tx Normalization" and "Activation" are also utilized by CLGC algorithm.

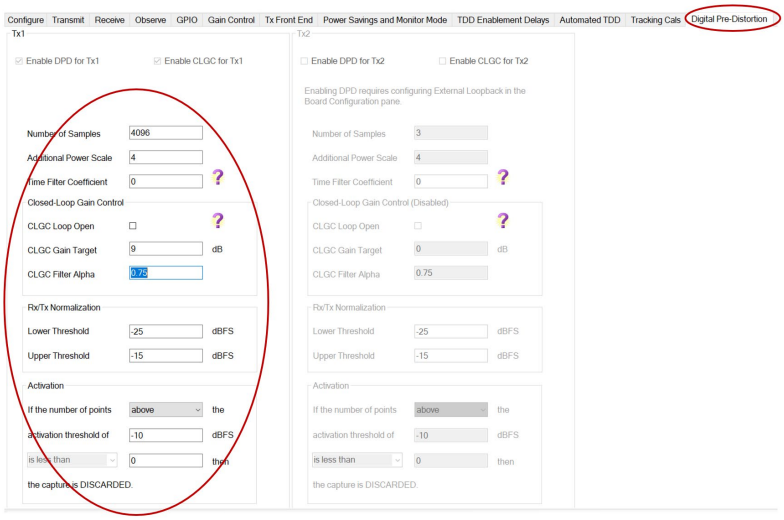


Figure 194. Configuring DPD/CLGC Post Initial Calibration Parameters Through TES

BOARD CONFIGURATION

Besides configuring the DPD/CLGC pre initial calibration and post initial calibration parameters, user should also configure 2 other parameters related to the board configuration. These 2 parameters are externalLoopbackPeakPower and externalLoopbackPathDelay, which should be provided to ADRV9001 before performing initial calibrations.

externalLoopbackPeakPower

It indicates the peak power of ORx input signal loop backed from the Tx output. For DPD/CLGC to achieve an optimal performance, the ideal externalLoopbackPeakPower should be set about -18dBm with a tolerance of ±5dBm. User could adjust the peak power by utilizing an external step attenuator after power amplifier.

externalLoopbackPathDelay

DPD/CLGC requires the alignment of the transmit signal capture x(t) with the external loopback capture y(t). The externalLoopbackPathDelay parameter provides user the capability to compensate for additional delays on the external loopback path from the ADRV9001 transmit output to ORx input. User should measure this delay and provide it to ADRV9001 before initial calibration. The measured delay is then used to compensate the delay between x(t) and y(t). This parameter is critical especially for WB applications due to high sample rate. In NB applications, it is less critical so user could simply set it to be zero unless there is a larger than usual delay in the external loopback path.

User could set these 2 configurations through TES as shown in Figure 195.

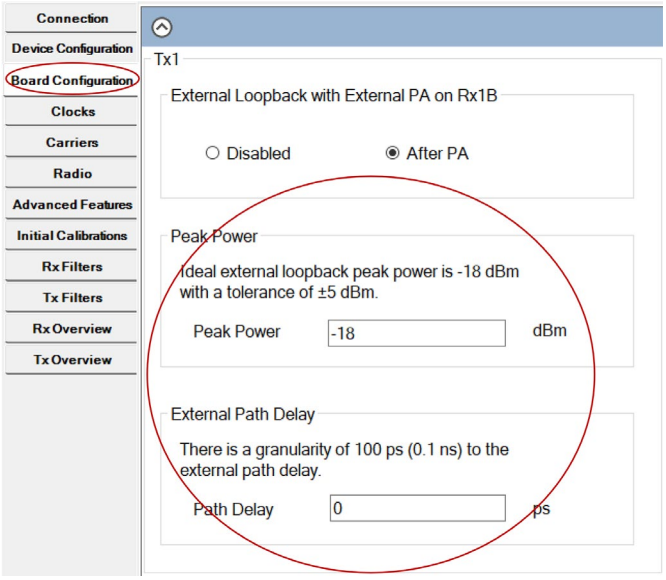


Figure 195. Configuring Board Configuration Related Parameters Through TES

SAVE AND LOAD DPD COEFFICIENTS FROM LAST TRANSMISSION

The ADRV9001 DPD also provides user an option to save and load DPD coefficients from last transmission. Therefore, DPD could either start from scratch (unity coefficients) or a set of known coefficients. This is a very useful option if user wants to reach convergence quickly under a similar transmit operation condition. User could utilize this feature as shown below in the TES.

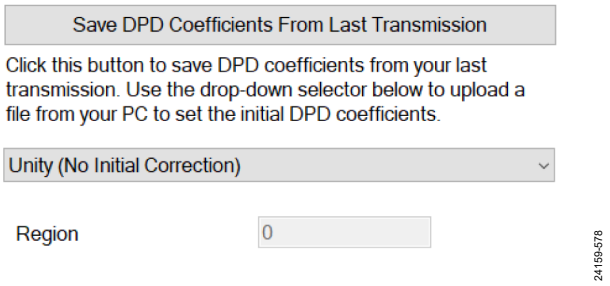


Figure 196. Save and Load DPD Coefficients from Last Transmission Through TES

DEFINE THE FREQUENCY REGION WHEN PERFORMING DPD WITH FH

As aforementioned, DPD operation can be performed during FH. User can define the 7 frequency regions as shown in TES below:

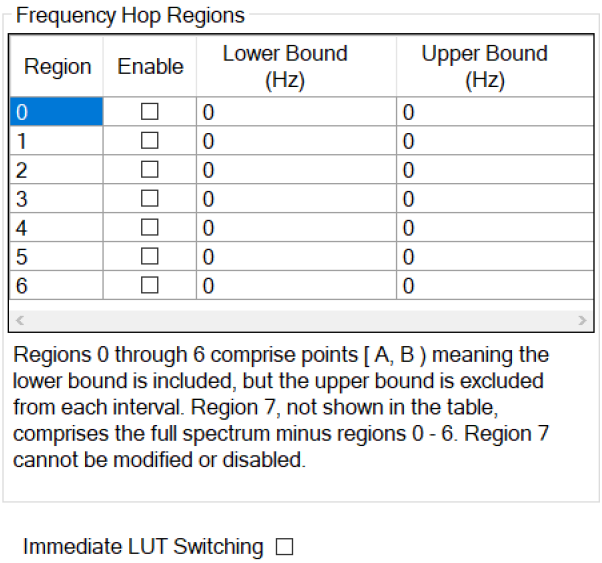


Figure 197. Define Frequency Hop Regions Through TES

Note in this case, immediate LUT switching should be disabled since the LUT switch only happens at the beginning of each hopping frame with the correct LUT for that LO frequency. In addition, the length of each hopping frame should be sufficient to allow capture of a specified number of samples at the DPD sampling rate plus the additional time it takes for the system to setup the DPD tracking calibration. If this condition is not satisfied, DPD could not be performed successfully with FH. For example, if the number of samples is set as 4096, for LTE20, the DPD sampling rate is at 184.32MSPS (more information about the DPD sampling rate will be provided in future releases), the hopping frame needs to be longer than 4096/184.32=22.22us.

DPD/CLGC API PROGRAMMING

A set of API commands are provided to set and inspect the DPD/CLGC parameters, which is summarized in Table 85. Board configuration parameters should be set through ADRV9001 initialization structure. Please refer to the doxygen document for more details.

Table 85. DPD APIs

DPD Rx Function Name	Description
adi_adrv9001_dpd_Initial_Configure	Configures the pre initial calibration DPD parameters. Called by adi_adrv9001_Utilities_InitRadio_Load() as part of device initialization.
adi_adrv9001_dpd_Initial_Inspect	Inspects the pre initial calibration DPD parameters.
adi_adrv9001_dpd_Configure	Configures the post initial calibration DPD parameters.
adi_adrv9001_dpd_Inspect	Inspects the post initial calibration DPD parameters.
adi_adrv9001_dpd_coefficients_Set	Sets DPD coefficients to be used at the next start of DPD.
adi_adrv9001_dpd_coefficients_Get	Gets DPD coefficients for the last solution.
adi_adrv9001_dpd_CaptureData_Read	Reads DPD captured data.
di_adrv9001_dpd_fh_regions_Configure	Configures DPD FH frequency regions.
adi_adrv9001_dpd_fh_regions_Inspect	Inspects DPD FH frequency regions.

DPD TUNING AND TESTING

Figure 198 describes an example setup for testing the integrated DPD with the ADRV9001 evaluation board in NB applications. (In NB applications such as TETRA, PA input should be connected to the TX1 output and PA output should be connected to RX1B.) As shown in Figure 198 , an LPF is required at the Tx1 output port to filter out the Tx harmonics before feeding the signal to PA driver (If the PA

driver has an internal LPF, then the external LPF is not needed). Since the device uses square wave mixer, it produces strong odd-order harmonics. Without filtering those harmonics, the DPD performance could be impacted. The step attenuators external to the ADRV9001 evaluation board are optional. Note it is important to set up the external loopback path before operating the integrated DPD. To achieve optimal DPD performance for TETRA waveforms, it is recommended to use an external LO source for transmitter due to possible better phase noise performance, while the receiver LO remains internal because the RF receive signal is downconverted to an IF instead of directly to baseband. For WB applications, the setup is similar but both transmitter and receiver LOs could be set to be internal because a WB signal is less sensitive to phase noise. A spectrum analyzer can be set up to observe the ACPR performance during a DPD operation.

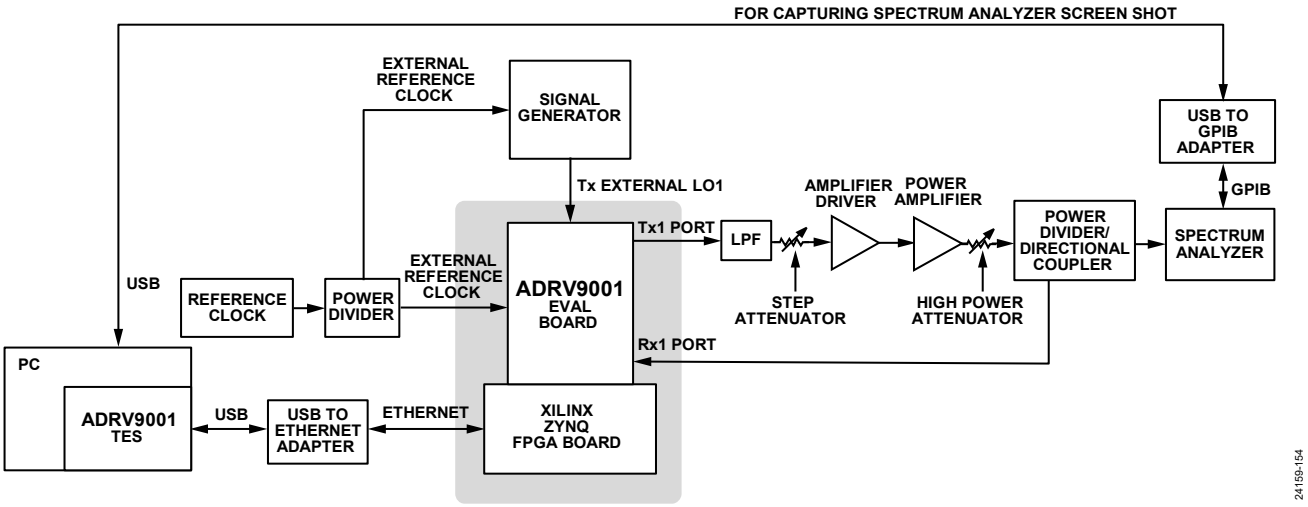


Figure 198. An Example Setup for Testing the Integrated DPD in Narrowband Applications

Once the setup is ready, user should further configure the TES and available external components properly which includes the following major steps:

- Select desired profile.
- Perform board configuration to indicate external loopback path with external PA is available.
- Enter the peak power of the loopback signal (ideally, it should be adjusted to be -18dBm±5dB. This could be achieved by tuning the external step attenuator).
- Measure the external loopback delay and provide it through TES. This could be done through API commands which will be discussed at the end of this section.
- Configure other initialization parameters such as RF frequency, LO source, and so on as desired. Also, enabling DPD for transmitter and configure the model tap polynomial terms. It is recommended to start with the default model tap. (The method of tuning the model tap order will be discussed in the next section.)
- Turn on DPD tracking calibration and all the other available tracking calibrations and start with the default DPD post calibration parameter settings provided in TES.
- After programming, load, and play the provided sample transmit input file.
- Properly tune the transmitter attenuation and/or the step attenuator to make sure that the ACPR performance at the device transmitter output is satisfactory before passing to PA. In addition, make sure that the transmit peak signal is around P1dB compression region for optimal DPD performance.

The user could compare the ACPR performance through spectrum analyzer with and without using the integrated DPD. Significant ACPR performance improvement with the integrated DPD should be observed even with internal LO sources. For TETRA waveforms, the ACPR after the second iteration of DPD is between −70 dB and −60 dB at an amplifier compression of P1dB. For LTE waveforms, the ACPR after the second iteration of DPD is between −55 dB and −50 dB at an amplifier compression of P1dB.

Tuning the Model Tap Order

DPD can be considered as an adaptive filter which is modelled according to the behavior of the PA. As mentioned previously, the ADRV9001 default model (Model 4) consists of four taps. Each tap consists of a series of polynomial terms to fit the nonlinear behavior due to compression at higher output power. The order of polynomial terms is determined by intermodulation falling closer to the carrier spectrum. In DPD, the orders of intermodulations that must be considered are usually third, fifth, and seventh orders, with decreasing magnitude, respectively. An nth-order intermodulation expands the signal bandwidth by n times. By inspecting the bandwidth expansion factor on a spectrum analyzer, the user can estimate how many orders of intermodulations that must be included in the polynomial terms,

in order to suppress the spectral regrowth down to the required ACPR. It is important not to include higher order power terms than needed, which might cause the DPD unstable.

DPD Model 4 consists of four taps as shown in the example tap arrangement diagram in Figure 199. The four taps can be classified into three categories:

The main tap – The main tap of the DPD adaptive filter that suppresses most of the spectral regrowth due to intermodulation; hence it has the greatest number of polynomial terms. It is labelled as TAP_1.

The side taps – There are two side taps on each side of the main taps. They are memory terms that compensate for frequency-dependent distortion in the frequency domain, and time misalignment between the transmit and receive captured data. The side taps have the same number of polynomial terms; and each side tap has about half of the number of polynomial terms of the main tap. They are labelled as TAP_0 and TAP_2.

The cross-term tap – The cross term is designed for further suppressing the residual spectral regrowth left over by the other three taps. The number of polynomial terms is usually equal to or less than that of each side tap. It is labelled as TAP_3.

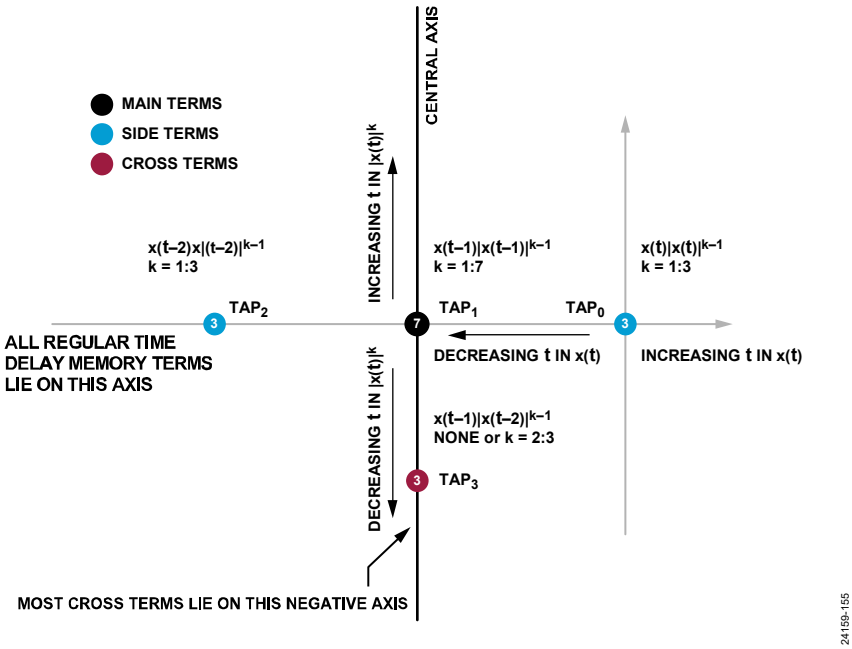


Figure 199. Example Polynomial Constellation Configuration for Model 4

Note that in Figure 199, k represents the order. To handle 7th harmonics, the main tap must include the power terms up to k = 7.

To find the best model tap order for a specific PA design model, the user can take the following recommended procedures:

1. Set the amplifier output power to have a compression ratio of 1 dB or slightly less, as shown in Figure 182, i.e. the maximum peak of the output signal is 1 dB below ideal linearity.
2. Determine the initial highest polynomial order of the main tap, TAP1, by measuring the spectral regrowth bandwidth to carrier bandwidth ratio. All lower order polynomial terms must be included. For example, if the bandwidth ratio is 5, set the initial highest polynomial to k = 5,that is, $x(t-1)|x(t-1)|^4$
3. Set the other taps to zeros, that is, turn off the other taps.
4. Use only TAP1 to execute DPD with order 5, then 6, and 7 (i.e. up to two orders above the initial estimate). Measure the ACPR for each case. Select the one that yields a better ACPR. If the difference is small, select a lower order one, say 5.
5. While keeping the main tap determined above, set the side taps, TAP0 and TAP2, to about half the order of the main tap. In the above example, the main tap has an order of 5, select the initial side tap order to be 2. Execute DPD with the main tap of order 5, and the side tap order of 2, then 3, and 4. Select the side tap order that yields the lowest ACPR, say 3.
6. While keeping the main tap and side tap orders determined above, set the initial cross term tap order, TAP3, to be 2. Execute DPD with the cross-term tap order of 2 and 3. Select none, 2 and 3, which yields the best ACPR. If the difference is small, select the lower order one, including "none" taps. Some power amplifiers do not need a cross term.
7. Iterate the above procedure as necessary with different combinations until you are confident with the selections. Keep all tap order selections to be minimal that satisfies your ACPR requirement with a 5 dB margin, which helps to keep DPD more stable. For example: if your ACPR requirement is −60 dB, set your ACPR target to be −65 dB.

Measuring the External Path Delay

User could call the following API commands to measure and check the external path delay:

1: adi_adrv9001_cals_ExternalPathDelay_Calibrate(). This API should be called when the channel state is CALIBRATED. It internally calls the following functions (user does not need to call those 2 lower level APIs).

- ExternalPathDelay_Run(), which runs external path delay calibrations.
- ExternalMinusInternalPathDelay_Measure(), which measures and gets the result of the difference in path delays between ILB and ELB and calculate the delay.

2: adi_adrv9001_cals_ExternalPathDelay_Set(). This API sets the external path delay value measured by "adi_adrv9001_cals_ExternalPathDelay_Calibrate()". This API should be call when the Channel state in STANDBY and CALIBRATED only.

3: adi_adrv9001_cals_ExternalPathDelay_Get(). It gets the current external path delay value. User could use this API to check the delay.

CLGC TARGET GAIN MEASURMENT

The user is responsible to set the gain target for their choice of PA. The gain target can be determined in lab by using a PA with a typical gain at room temperature. The following guidelines provide optimal operation of ADRV9001. The optimal gain target and adjustable gains in transmit and external loopback receive should be fixed during operation, except for the transmit attenuation adjustment by CLGC.

1. Set the transmit frequency to the middle of the band of interest.
2. Adjust the peak digital transmit signal amplitude to be between -6 dBFS at the DAC.
    - margin 3 dB
    - Maximum DPD expansion 3 dB
3. Adjust the transmit attenuator so that the PA input ACPR (before PA) is about -70 dBc.
4. Connect the amplifier drivers and PA to the RF output
5. Increase the transmit attenuation so that the peak compression is about 1 dB
6. If a peak compression of 1 dB cannot be reached, increase the gain of an amplifier driver.
7. Adjust the gain of the external loopback path to make the feedback signal peak amplitude to be around -18dBm with a tolerance of ±5dB .
8. These settings should not be changed during normal operation. Otherwise, a new gain target should be determined.
9. Use TES to measure the transmit gain to help determine the gain target. To do this, user should enable "CLGC Loop Open" option under "Digital Pre-Distortion" tab in TES as shown in the plot below. Then user should play a Tx modulated input signal and observe the "CLGC last gain" and "CLGC filtered gain" at the "Transmit" tab. Note the a single pole filter can be applied to smooth the gain measurement and user could set the filter coefficient ($0<\alpha<1$) as shown in Figure 200. The mesurment is smoother when bigger α is used. By default, it is set as 0.75.

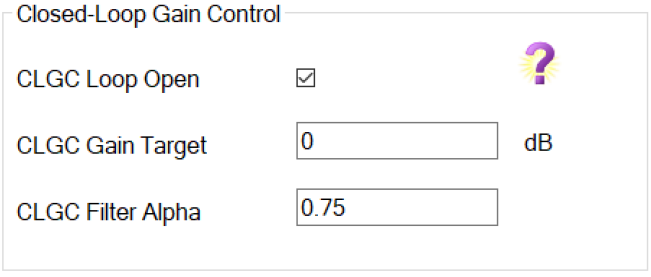


Figure 200. CLGC Configuration Parameters

10. By determining the final gain target based on the "CLGC last gain" and "CLGC filtered gain" provided by ADRV9001, user should configure "CLGC Gain Target" as shown in the above figure. After that user should disable "CLGC Loop Open" to close the loop to allow CLGC to achieve the gain target. Note if the transmit attenuation is changed by anything else rather than the CLGC, the change must be factored in the gain target. To achieve the most accurate gain control, the gain variation of the external components and ORx data path components should also be compensated by adjusting the gain target.

DYNAMIC PROFILE SWITCHING

OVERVIEW

Dynamic Profile Switching (DPS) is a feature supported by ADRV9001 to allow user to switch among several predefined profiles with different signal bandwidth and sampling rate on the fly. With a switching time of about 50us, DPS enables a very fast change to a different profile without the need to reinitialize the chip. However, the signal bandwidth and sampling rate of different profiles must satisfy a relationship of being a multiple integer of 2 between each other. In addition to that, all profiles must be wideband profiles with a signal bandwidth no less than 1MHz. For example, the set of LTE profiles including the sampling rate of 61.44MSPS, 30.72MSPS, 15.36MSPS, 7.68MSPS, 3.84MSPS and 1.92MSPS satisfy the requirements, therefore, users can perform DPS on those profiles. Note in the current release, this is the only set of profiles supported by ADRV9001 for DPS. In future releases, other profiles might be added with a limitation of the maximum total number of profiles to be 6.

ADRV9001 supports DPS for both TDD and FDD operations. When performing switching, BBIC should first switch all channels from "RF enabled" state to "Primed" state. In the current release, the new profile is applied on all the configured Tx and Rx channels simultaneously so DPS could not operate on channels individually. Figure 201 depicts a high-level diagram showing the DPS operation in a TDD system and Figure 202 depicts a high-level diagram showing the DPS operation in an FDD system, respectively.

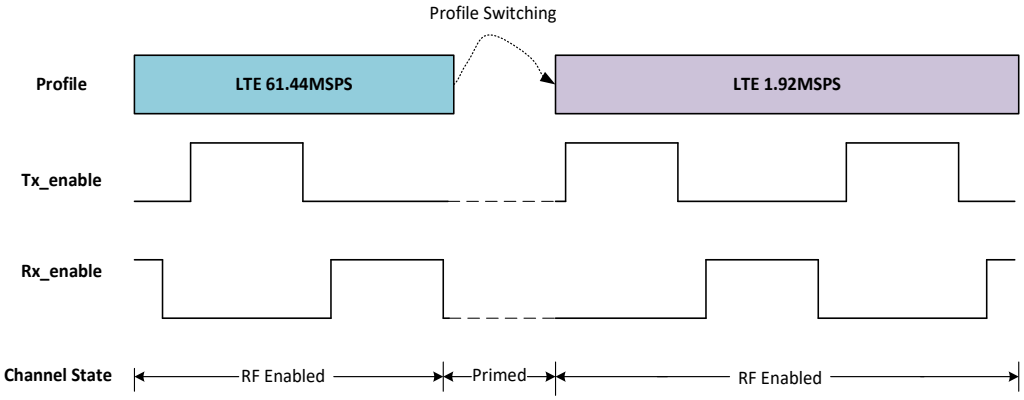


Figure 201: DPS Operation in TDD system

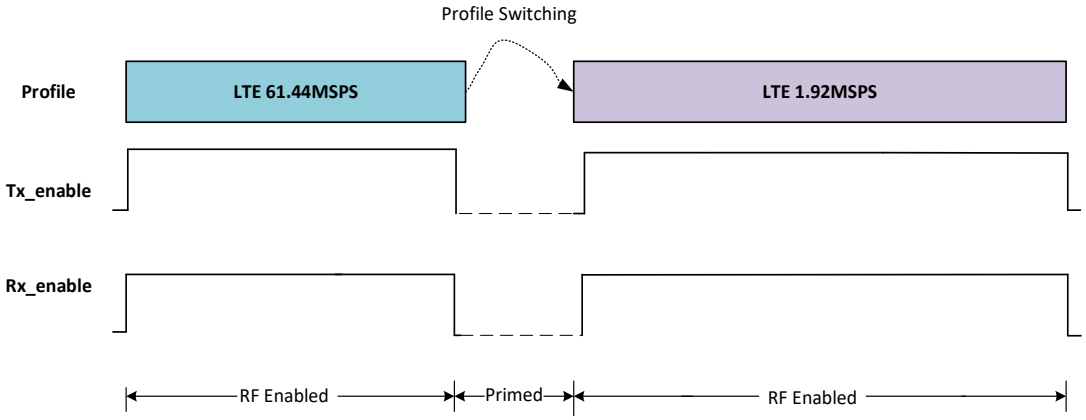


Figure 202: DPS Operation in FDD system

INITIAL CALIBRATION WITH DPS

From ADRV9001 point of view, a dynamic profile change is considered as a change in signal bandwidth and sampling rate relative to a main profile through changing decimation/interpolation settings in the data path and Rx PFIR coefficients. For the set of LTE profiles currently supported, LTE 61.44MSPS with the highest sampling rate is considered as the main profile. During the initialization, user enables DPS by configuring more than 1 profile. Note in the current implementation, user is only allowed to configure all 6 LTE profiles for DPS. In the future, if user sets the number of profiles to be N which could be between 2 to 6, then DPS is enabled and the N profiles from the highest sampling rates to lower sampling rates in order are configured for DPS. For example, if user sets the number of profiles

to be 3 for the set of LTE profiles, then LTE 61.44MSPS, 30.72MSPS and 15.36MSPS profiles are enabled for DPS. Each profile has a profile index associated with it. Index 0 denotes the profile with the lowest bandwidth and sampling rate and Index 5 denotes the profile with the highest bandwidth and sampling rate. During initialization, user could enable Rx PFIR for each profile by using either the default PFIR coefficients or providing a set of custom PFIR coefficients.

ADRV9001 is first calibrated with the main profile similarly as in the regular operation mode without DPS. Then it is further calibrated for all the dynamic profiles using the API function adi_adrv9001_cals_Dynamic_profiles_calibrate(). When calibration is completed, the main profile is set as the initial profile to operate. Note the SSI rate is configured based on the main profile and it does not change during the entire profile switching operation.

Figure 203 described the initial calibration procedure when DPS is enabled.

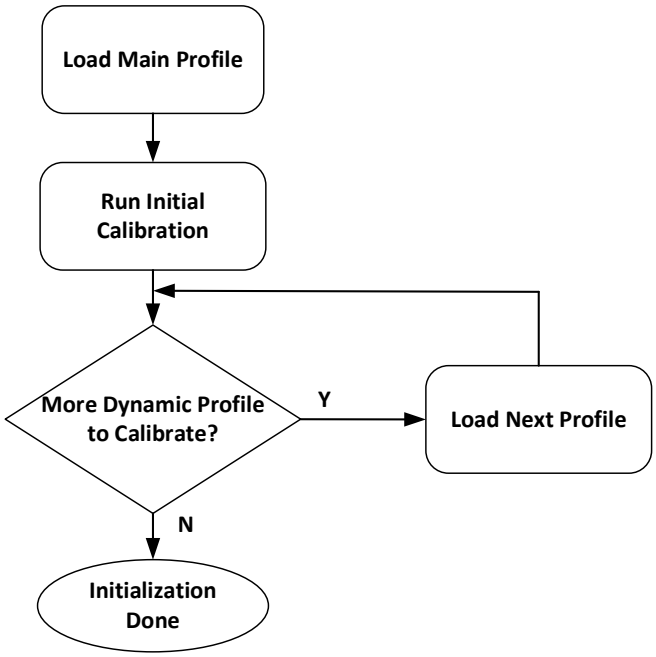


Figure 203: Initial Calibration with DPS

PERFORM DPS ON THE FLY

As mentioned earlier, after initialization, ADRV9001 operates on the main profile with the fixed SSI rate which does not change during the entire profile switching operation. To prepare for operating with the next profile, BBIC should properly configure the ratio between the interface rate and the new sampling rate before requesting profile switching. It should also notify ADRV9001 about the next profile by calling the API command adi_adrv9001_arm_NextDynamicProfile_Set(). Furthermore, as an option, BBIC could also set the Rx and Tx PFIR coefficients associated with the next profile on the fly by calling the API command adi_adrv9001_arm_NextPfir_Set(). Note these 2 APIs could be called in different channel states including "Standby", "Calibrated", "Primed" and "RF_enabled". When BBIC is ready to perform profile switching, it should first move all Tx and Rx channels from "RF_enabled" state to "Primed" state and call the API command adi_adrv9001_arm_Profile_Switch () to request ADRV9001 to switch to the new profile. Once receiving this command from BBIC, ADRV9001 starts to perform switching by applying the new profile and PFIR coefficients BBIC set earlier and it will not respond to any signals on Tx_enable and Rx_enable pins. ADRV9001 takes about 50us to complete the switch. After that, BBIC can move the channels from "Primed" state back to "RF_enabled" state and continue the transmit and receive operations with the new profile.

Figure 204 shows the procedure of performing DPS and the communication between BBIC and ADRV9001.

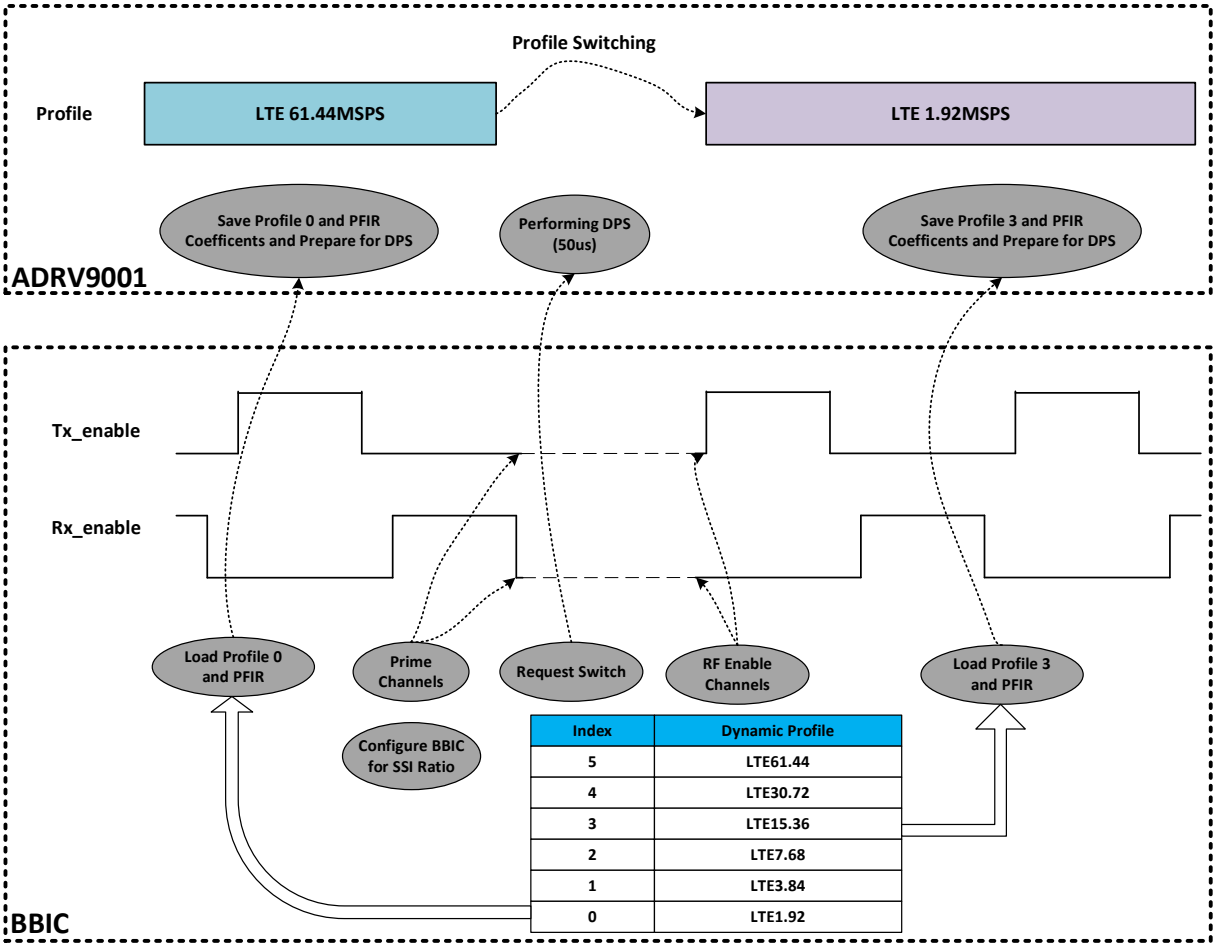


Figure 204: BBIC and ADRV9001 Interaction to Perform DPS

DPS API PROGRAMMING

Table 86 summarizes the set of ADRV9001 API commands provided for DPS. Please refer to the API doxygen document for more details.

Table 86. DPS APIs

DPD Rx Function Name	Description
adi_adrv9001_cals_Dynamic_profiles_calibrate	Runs the initial calibrations for dynamic profiles.
adi_adrv9001_arm_NextDynamicProfile_Set	Sends the next dynamic profile to ADRV9001 and waits it to process when profile switching is performed.
adi_adrv9001_arm_NextPfir_Set	Sends a bank of PFIR coefficients to ADRV9001 and waits it to process when profile switching is performed.
adi_adrv9001_arm_Profile_Switch	Requests ADRV9001 to perform dynamic profile switching.

SUMMARY OF DPS LIMITATIONS

DPS allows user to switch between different profiles very fast on the fly. In order to operate it properly, it is important to understand the limitations in the current implementations. The following list provides a summary:

- DPS is limited to LTE 1.92MSPS, LTE 3.84MSPS, LTE 7.68MSPS, LTE 15.36MSPS, LTE 30.72MSPS, and LTE 61.44MSPS profiles with the 16-bit interface.
- The maximum number of profiles user can configure for DPS is 6 and currently only 6 can be selected.
- DPS operates simultaneously on all configured channels and cannot operate on any channels individually.

- Only the LVDS interface is supported for all profiles and no switching between LVDS and CMOS is allowed.
- Switching between different TDD channel modes, or between different FDD channel modes, or between TDD and FDD channel modes is not permitted with a profile change.
- The LO frequency of any channel cannot be modified with a profile change.
- The BBPLL frequency cannot be modified with a profile change.
- Tx/Rx TIA bandwidth and DAC/ADC sample rate could not change with a profile switch.
- The physical user interface (e.g. CMOS versus LVDS, or interleaved data versus non-interleaved data) cannot be modified with a profile change.
- SSI rate will be the highest rate of all dynamic profiles.
- ADRV9001 will either operate in Frequency Hopping mode or DPS mode, never both.

Note those limitations might be removed or relaxed in future releases.

DPS OPERATIONS IN TES

TES provides a user interface for experimenting DPS. In the current implementation, LTE 61.44MSPS profile must be configured under the "Device Configuration" page in either TDD or FDD mode. To enable DPS, under "Rx Filters" tab, choose the "Number of Dynamic Profiles" to be 6 and then define the Rx PFIR coefficients for all 6 enabled dynamic profiles. User has the option of disabling PFIR, using the default PFIR or uploading a set of custom PFIR coefficients, as shown in Figure 205.

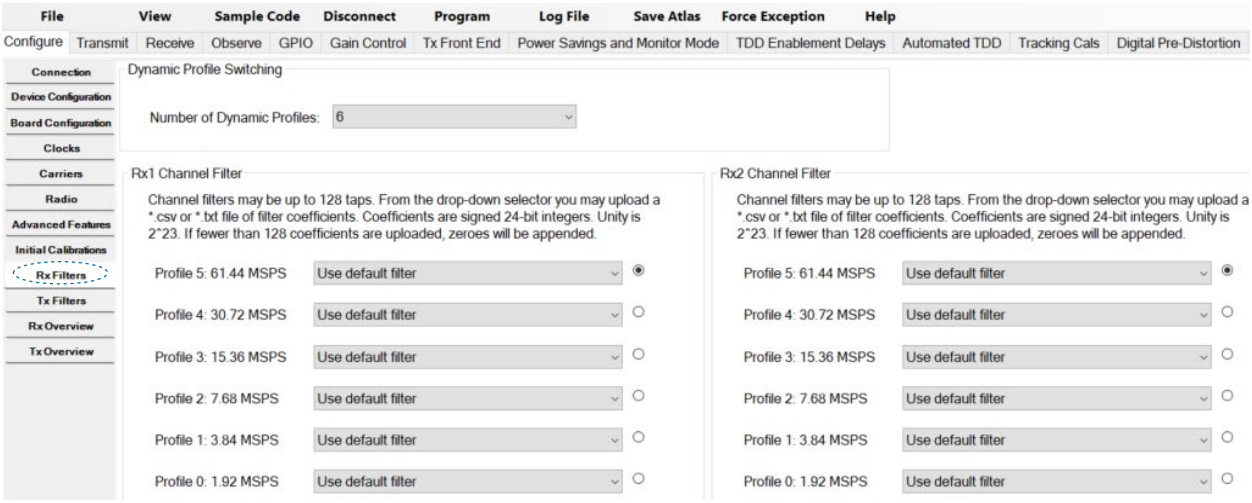


Figure 205: DPS Initialization in TES

The frequency response of the PFIR for a certain dynamic profile can be displayed in TES by selecting this profile.

After programming, main profile is used as the initial profile to operate. To perform DPS, as shown in Figure 206, user should first move all channels (all Tx and Rx channels) to "Primed" state, then pick a new profile to request a profile switch from the "Sample Rate" drop down menu under either "Transmit" or "Receive" tab. After that, play Tx and/or Rx to observe the profile change and verify that all channels work as expected.

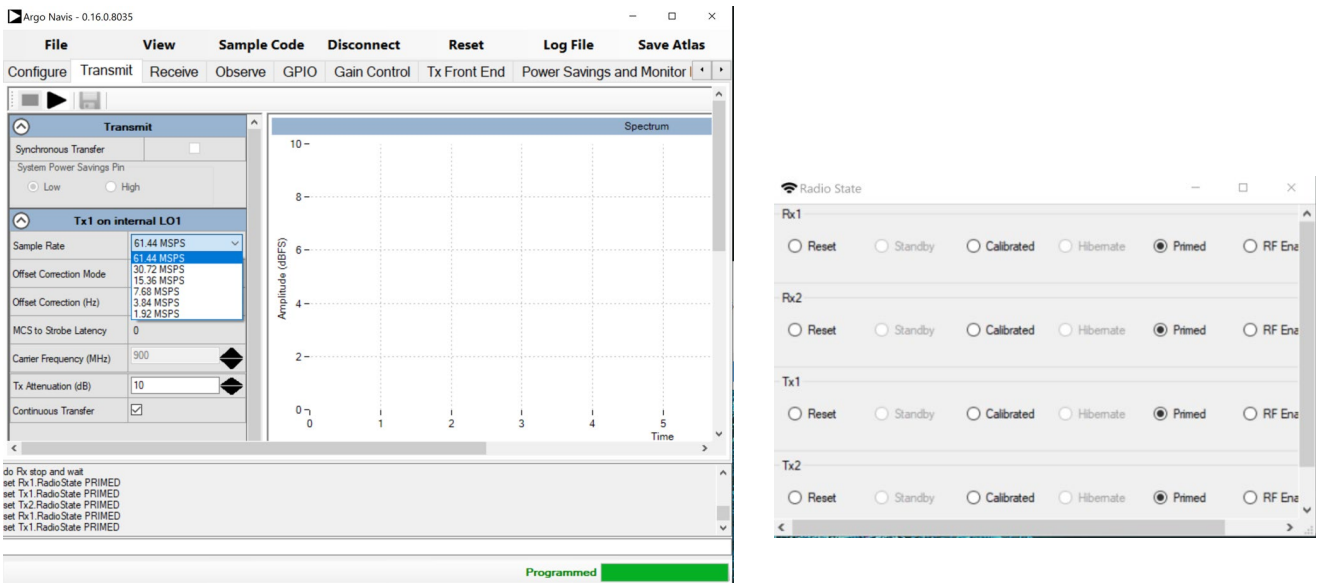


Figure 206: Performing DPS in TES

GENERAL-PURPOSE INPUT/OUTPUT AND INTERRUPT CONFIGURATION

ADRV9001 provides user with number of software configurable General-Purpose Input/Output (GPIO) pins. By utilizing API functions, user can configure GPIO pins to operate with a variety of control or monitoring functions. ADRV9001 has two types of GPIO:

- 16 digital GPIO pins
  - referenced to VIOCTRL_1P8 supply
  - designated DGPIO_0 through DGPIO_15
- 12 analog GPIO pins
  - referenced to VAGPIO_1P8 supply
  - designated AGPIO_0 through AGPIO_11

The Digital and Analog GPIO pins can be used as real-time status signals that provide device status information from ADRV9001 to the baseband processor when the GPIO pins are configured as outputs, with respect to ADRV9001. When set as inputs, the GPIO pins can be used as real-time control signals that can alter the device state. The API functions related to GPIO configuration give the user the ability to configure pins as inputs or outputs and assign functionality to specific pins.

Figure 207, Figure 208 illustrates the different functionalities that can be enabled in the device and then controlled using the Digital GPIO pins and Analog GPIO pins, not all functionalities can be enabled at the same time.

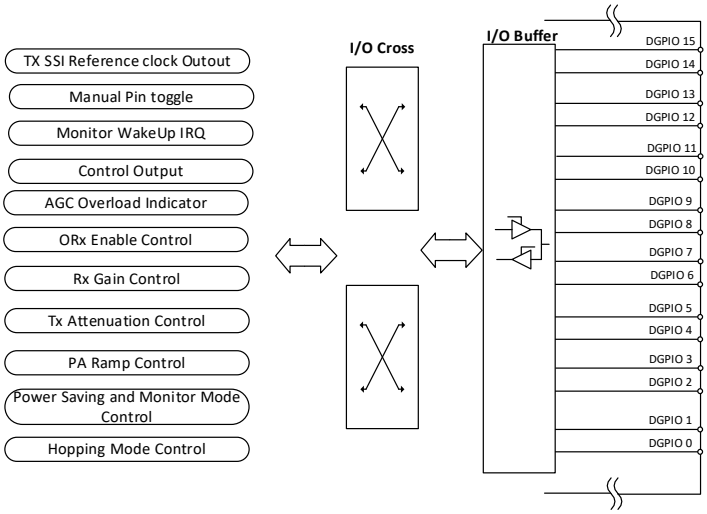


Figure 207. Digital GPIO Features Overview

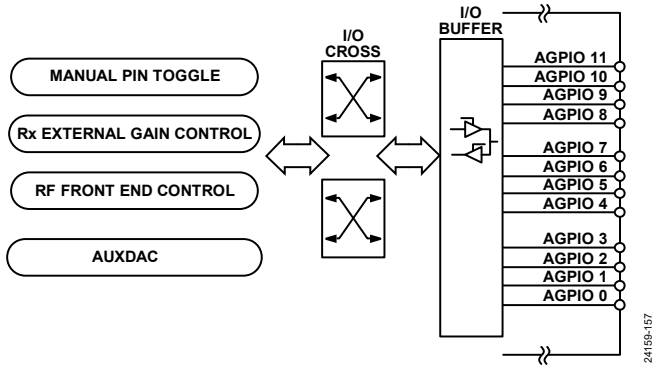


Figure 208. Analog GPIO Feature Overview

In configuring the GPIO, the two major factors to consider are the GPIO output enable control and the GPIO source control.

The output enable determines the direction of the pin, if a pin is set as output, then the GPIO I/O buffer is configured as an output.

The GPIO source control determines the functionality of the pin. The Digital GPIO source control is assigned in groups of 2, this means that DGPIO_0 to DGPIO_1 share a single source control, DGPIO_2 to DGPIO_3 share a single source control, and so on The Analog

GPIO source control is assigned in groups of 4, that means that AGPIO_0 to AGPIO_3 share a single source control, AGPIO_4 to DGPIO_7 share a single source control, and so on.

The API commands adi_adrv9001_gpio_ControlInit_Configure() and adi_adrv9001_gpio_Configure() are used to configure the digital or analog GPIO work modes, and some of the GPIO work modes are configured with the specified functions, for example the Pin based Tx Attenuation, etc. The following subsections will explain the operation details of digital GPIO and analog GPIO.

DIGITAL GPIO OPERATION

Each Digital GPIO pin can be set to either input or output mode, the input mode allows the baseband processor to drive pins on the ADRV9001 to execute specific tasks, the output mode allows the ADRV9001 to output various control or status signals to baseband processor.

Note that conflicts regrading GPIO usage may occur when using combinations of certain features, users should ensure that multiple functions are not assigned to the same GPIO pins.

Digital GPIO Input Features

The following table provides a list of digital GPIO input features available that interact with datapath control elements on ADRV9001, for the Digital GPIO features with the Table 86, the APIs automatically set the I/O direction of the GPIO pins assigned for the feature.

Table 87. Summary of Input Digital GPIO Features

Feature	Description	GPIO Pins Available for Feature
ORx Enable	Configure specific Digital GPIO pins to enable/disable Rx observation channel.	DGPIO_0 through DGPIO_11: ORx Enable control pin select.
Pin Based Tx attenuation Increment and Decrement	Configures specific Digital GPIO pins to increase or decrease attenuation on any Tx channel after a rising edge on the assigned pin.	DGPIO_0 through DGPIO_15: Tx attenuation increment pin select. DGPIO0 through DGPIO_15: Tx attenuation decrement pin select.
Pin Based Rx Gain Index Increment and Decrement	Configures specific Digital GPIO pins to increase or decrease gain index on any Rx or ORx channel after a rising edge on the assigned pin.	DGPIO_0 through DGPIO_15: Rx/ORx gain index increment pin select. DGPIO0 through DGPIO_11: Rx/ORx gain index decrement pin select.
Tx Power Amplifier Ramp Control	Configure specific Digital GPIO pins to ramp up the power amplifier controlling on any Tx channel after a rising edge on the assigned pin and ramp down the power amplifier controlling on any Tx channel after a falling edge on the assigned pin	DGPIO_0 through DGPIO_15: power amplifier Ramp controlling pin select.
Power Saving and Monitor Mode control	Configure specific Digital GPIO pins to enable or disable Channel Power Saving or System Power Saving or Monitor Mode	DGPIO_0 through DGPIO_11: Mon_enable pin select
Hopping Mode control	Configure specific Digital GPIO pins to control the Hopping mode, including the hopping enable, update gain value, frequency index, and so on	DGPIO_0 through DGPIO_11: Hopping Event pin select DGPIO_0 through DGPIO_15: Hopping gain value pin select DGPIO_0 through DGPIO_15: Hopping frequency index pin select

ORx Enable Control

ADRV9001 Receiver can be reused as observation channel through either port A or port B in TDD system, a DGPIO pin should be assigned as ORx Enable signal once the ORx channel is configured. BBIC can toggle the DGPIO in Tx RF Enabled state to enable/disable the ORx channel. Users should be noted that ORx Enable timing should only be the subset of Tx Enable timing.

Enum ADI_ADRV9001_GPIO_SIGNAL_ORX_ENABLE_1/ ADI_ADRV9001_GPIO_SIGNAL_ORX_ENABLE_2 is defined for DPGIO as ORx Enable signal, the DPGIO assignment can be set by API adi_adrv9001_gpio_Configure().

Pin-Based Tx Attenuation Control

A complete description of Tx attenuation control is provided in the Transmitter Signal Chain section in this User Guide.

Pin-based Tx attenuation control provides an interface to make attenuation adjustments with precise timing control. The pin-based control offers lower latency than SPI based attenuation change operations. In pin-based attenuation control, certain digital GPIO pins are assigned "increment attenuation" or "decrement attenuation" functionality. By applying a high pulse on the assigned GPIO pin, the attenuation for a specific channel is either increased or decreased, depending on the assigned functionality. The pulse width requirement is at least 2 system clock cycles (184.32 MHz in standard profiles) in the logic high state. Increment and decrement functionality can be assigned to any digital GPIO from DGPIO_0 to DGPIO_15. Pin-based Tx attenuation control allows multiple increments or decrements of Tx attenuation.

Set Tx attenuation control mode to "MODE_PIN" by the API function adi_adrv9001_Tx_AttenuationMode_Set(), and select the properly GPIOs for each channel by the API function adi_adrv9001_Tx_Attenuation_PinControl_Configure(), baseband processor can send the pulses to ADRV9001 via the specific digital GPIO pins to increase or decrease the Tx attenuation.

Pin-Based Rx Gain Control

A complete description of Rx Gain control is provided in the Receiver Gain Control section of this User Guide.

Pin-based Rx gain control is relevant for applications which require Manual Gain Control (MGC) and precise timing for gain change events. The pin-based control offers lower latency than SPI based gain change operations. In pin-based gain control, certain digital GPIO pins are assigned "increment gain index" or "decrement gain index" functionality for a particular receiver channel. By applying a high pulse on the assigned GPIO pin, the gain index for a specific channel is either increased or decreased, depending on the assigned functionality. The pulse width requirement is at least 2 system clock cycles (184.32 MHz in standard profiles) in the logic high state. Increment and decrement functionality can be assigned to any digital GPIO from DGPIO_0 to DGPIO_15.

Note that if the user has programmed a gain table that operates in a subset of the full gain table range (i.e. using index 195 to 255), once the gain Index has reached Min/Max Gain index subsequent the pin-based Rx gain control rising edge will not change the gain index.

Set Rx Gain control mode to "MODE_PIN" by the API adi_adrv9001_Rx_GainControl_Mode_Set(), configures the properly digital GPIO pins for gain increase and decrease control and other control parameters by API adi_adrv9001_Rx_GainControl_PinMode_Configure(), then baseband processor can send the pulses to ADRV9001 via the specific digital GPIO pins to increase or decrease the Rx Gain index.

Power Amplifier Ramp Control

A complete description of power amplifier ramp control will be provided in the user guide in the future.

When the power amplifier ramp control function is used in ADRV9001, an optional digital GPIO pin can be assigned as the "power amplifier ramp control enable" functionality driven by the baseband processor, the rising edge of power amplifier ramp control enable with programable delay will be act as the ramp up trigger signal and the falling edge of power amplifier ramp enable with optional programable delay acts as the ramp down trigger signal.

The DPGIO assignment for PA ramp control can be set by API adi_adrv9001_Tx_PaRamp_Configure().

Power Saving and Monitor Mode control

The DPGIO can be used as Channel Power saving, System Power saving and Monitor Mode Enable signal, see the Power Saving and Monitor Mode section for the details.

Enum ADI_ADRV9001_GPIO_SIGNAL_MON_ENABLE_SPS and ADI_ADRV9001_GPIO_SIGNAL_POWER_SAVING_CHANNEL1/ ADI_ADRV9001_GPIO_SIGNAL_POWER_SAVING_CHANNEL2 are for the DPGIO as System Power saving/Monitor Mode and Channel power saving enable receptive.

When the CPS and/or SPS/Monitor Mode is enable, BBIC can call the API adi_adrv9001_gpio_Configure() to set the power saving and monitor mode control enable signals on DGPIOs.

Hopping Mode Control

A DPGIO should be assigned as frequency hopping hop control signal, and users can also use the DGPIOs to choose the hop table index, Rx Gain table index, Tx Attenuation index. Please refer the Frequency Hopping section for more detail.

The hopping mode control GPIO functions can be set through API adi_adrv9001_fh_Configure().

Digital GPIO Output Features

Table 87 summarizes the available digital GPIO output features, the relative APIs will automatically set the GPIO I/O directions.

Table 88. Summary of Digital GPIO Output Features

Feature	Description	GPIO Pins Available for Feature
Control out Mux	Allows a choice of Main/RX/TX control signals to output from ADRV9001 to monitor the status of the device	DGPIO0 through DGPIO_11
Manual Pin Toggle	Manual control the GPIO output level, API functions set output pin levels and read the input pin levels	DGPIO0 through DGPIO_11
Monitor WakeUp Baseband Processor/DSP	Interrupt signal to wake up baseband processor/DSP when baseband processor/DSP is in sleep state	DGPIO0 through DGPIO_11
Rx AGC overload indicator	Allows output the AGC overload signals	DGPIO0 through DGPIO_11
TX DCLK OUT	Allows output the SSI reference clock for baseband processor to generate the TX SSI clock, data and strobe to ADRV9001	DGPIO_12 through DGPIO_13 TX Channel 1 SSI reference clock out pin select, DGPIO_14 through DGPIO_15 TX Channel 2 SSI reference clock out pin select

Control Out Mux

Control Out Mux (sometimes referred as "Monitor out") allows status signals within the ADRV9001 to be output to digital GPIOs, such as, AGC mode the gain change flag, gain index can be mapped to DGPIO for BBIC observation by API adi_adrv9001_Rx_GainIndex_Gpio_Configure().

ADRV9001 internal stream status can be mapped to DGPIO for the accurate Tx/Rx enable control affective timing measurement, adi_adrv9001_Stream_Gpio_Debug_Set() can be called to enable this feature, DGPIO0~3 is configured to represent the Tx/Rx Enable control effective timing. Please refer Pin control mode timing measurement for the detail.

Rx AGC Overload Indicator

The status of peak detectors and power detector in the Rx channel can be retrieved to baseband processor through a set of DGPIO pins. One DGPIO configuration is for using the peak detect mode, in which the overrange and under-range conditions of both APD and HB detectors are provided to user. The other DGPIO configuration is for using the peak/power detect mode, in which the overrange and underrange conditions of APD and power detector are provided to user.

The DGPIO pins could be associated with either one of the receivers, Rx1 or Rx2. However, when the similar information is required for both receivers, they could be selectively muxed and provided to user simultaneously.

Data structure of adi_adrv9001_GainControlCfg_t, and of its substructures, adi_adrv9001_PeakDetector_t, adi_adrv9001_PowerDetector_t initialize the necessary Gain control parameters as well as the digital GPIO pins assignment for the overload indicator, API command adi_adrv9001_Rx_GainControl_Configure() is provided to set the parameters. (See the Receiver Gain Control section for details.)

Manual Pin Toggle

This feature allows control of the logic level of individual digital GPIO pins, adi_adrv9001_gpio_ManualOutput_Configure() configures the relative GPIO to manual control mode, the adi_adrv9001_gpio_OutputPinLevel_Set() command is used to set the output level of GPIO pins. adi_adrv9001_gpio_OutputPinLevel_Get() command is used to read the GPIO pins output levels.

Additionally, adi_adrv9001_gpio_InputPinLevel_Get() command can be used to read the input GPIO level if the relative GPIO is configured as input by adi_adrv9001_gpio_ManualInput_Configure() .

Monitor Wake-Up Baseband Processor/DSP

Certain digital GPIO pin can be assigned as "wake up baseband processor/DSP" to output the interrupt signal to wake up the baseband processor/DSP when ADRV9001 works in monitor mode and specific detection conditions are met.

Enum "ADI_ADRV9001_GPIO_SIGNAL_MON_BBIC_WAKEUP" is used to as DPGIO for monitor wake up interrupt signal, API adi_adrv9001_gpio_Configure() is called to enable this function.

TX DCLK OUT

This mode allows to configure the DGPIO pins to a pair of differential or a single-ended reference clock for baseband processor if the TX SSI and RX SSI runs at different lane rate, the users could use this reference clock to generate the TX LSSI clock, data and strobe when the RX SSI and TX SSI run at different clock rate. TX1_DCLK_OUT± functionality can be assigned DGPIO_12 and DGPIO_13 when it is in LVDS mode, or either of DGPIO_12 or DGPIO_13 can be used as the Tx1 DCLK out if it is in CMOS mode. Similarly, TX2_DCLK_OUT±

functionality can be assigned DGPIO_14 and DGPIO_15 when it is in LVDS mode, or either of DGPIO_14 or DGPIO_15 can be used as the Tx1 DCLK out if it is in CMOS mode.

Noted, when Tx DCLK OUT function is disabled, the corresponding DGPIOs (DGPIO12/13 or DGPIO 14/15) can only be reused as input function.

ANALOG GPIO OPERATION

The main purpose of the Analog GPIO pins is to serve as control pins for the external control elements, such as a Digital Step Attenuator (DSA), Low Noise Amplifier (LNA), external LO/VCO components, T/R switch of TDD system, and so on. An alternative function of some Analog GPIO pins are to provide the auxiliary DAC output.

A high level overview of the analog GPIO features are provided in Table 88.

Table 89. Summary of Analog GPIO Features

Feature	Description	GPIO Pins Available for Feature
RX Gain Table External Control Word	The RX gain table can include a column for 2-bit control of an external gain element (LNA), each Rx channel has 2 Analog GPIO pins associated with it.	Any Analog GPIO, but Rx1/Rx2 external gain word must be in one AGPIO nibble
RF Front-End Control	Allow AGPIO timing to be associated with Tx/Rx_Enable to control the RF Front End	AGPIO_0 for Tx1 AGPIO_1 for Rx1 AGPIO_8 for Tx2 AGPIO_9 for Rx2
Manual Pin Toggle	Manual control the GPIO output level, API functions sets output pin levels and reads the input pin levels	Any Analog GPIO
Auxiliary DAC Output	Allow the auxiliary DAC output on analog GPIO pins	AGPIO_0: AuxDAC0 output pin select AGPIO_1: AuxDAC1 output pin select AGPIO_2: AuxDAC2 output pin select AGPIO_3: AuxDAC3 output pin select

RX Gain Table External Control Word

A complete description of RX Gain Table external control is provided in the Receiver Gain Control section in this User Guide.

External LNA gain can be controlled by ADRV9001 AGPIO output, each channel has 2 AGPIO control signals and achieve up to 4 external LNA gain steps control. The external LNA gain control can be enabled and configured by adi_adrv9001_Rx_ExternalLna_Configure().

The AGPIOs for channel1 and channel 2 must be in one AGPIO nibble, which means the 4 AGPIO for external gain control has to be AGPIO[3:0] or AGPIO[7:4] or AGPIO[11:8]. For example, AGPIO_7/AGPIO_6 for Rx1 external gain control, AGPIO_5/AGPIO_4 for Rx2.

RF Front-End Control

To save the baseband processor control pins, ADRV9001 provides the function to output control signals via analog GPIO pins to power up/down the external RF front end components (i.e. LNA, TX Gain blocks, Ext PLL) or switch the T/R switch of a TDD system. For example, a TX_On, RX_ON output signal through the analog GPIOs and associated with the ADRV9001 Tx/Rx Enable timing and state can be used to enable/disable the power amplifier and LNA respectively, or do the antenna switch.

To get the best timing control performance, the dedicated AGPIOs are assigned for Tx/Rx front end control, AGPIO_0, AGPIO_1, AGPIO_8, AGPIO_9 are associated with Tx1_Enable, Rx1_Enable, Tx2_Enable, Rx2_Enable respectively.

The AGPIO for external RF front end control is initialized in adi_adrv9001_gpio_ControlInit_Configure(), and the relative AGPIOs are configured by API adi_adrv9001_gpio_Configure().

Noted, once the AGPIO external RF front end control is enabled, the Rx Gain table external control can only use AGPIO[7:4].

Manual Pin Toggle

Similar with the manual pin toggle for digital GPIOs, this feature allows control of the logic level of individual analog GPIO pins, the adi_adrv9001_gpio_ManualAnalogOutput_Configure() and adi_adrv9001_gpio_ManualAnalogInput_Configure () is used to configure the Analog GPIO to manually output and input mode respectively. adi_adrv9001_gpio_OutputPinLevel_Set() and adi_adrv9001_gpio_InputPinLevel_Get() can be used to set and read relative analog GPIO level respectively.
Users can use the manual AGPIO level toggle to control external RF components, like power up/down the PA, etc.

Auxiliary DAC Output

Auxiliary DAC can supply bias voltages, analog control voltages, or other system functionality, refer the Auxiliary Converters and Temperature Sensor section for the detail. Analog GPIO 0 through 3 provide the alternative function for the Aux DAC 0 through 3 output respectively.

INTERRUPT

The ADRV9001 features the general purpose interrupt output pin (GP_INT), the GP_INT pin can alert the baseband processor that an important event or error regarding the device operation has occurred. These events include of unlocking of PLLs, Stream Processors errors or ARM exception, and so on.

A description of the interrupt sources and their bit positions is provided in Table 88. An Interrupt source can be masked so that it won't be transmitted to the BBIC on GP_INT pin or in status registers. An interrupt is masked when the corresponding mask bit is set to '1'. The GP_INT pin represents a logical OR of the enabled GP_INT mask sources. It is not necessary to enable all of the interrupt sources.

Table 90. GP_INT Bitmask Description

Bit Position	Description	Component
0	ARM Error.	ARM
1	Force set an interrupt	ARM
2	ARM system error	ARM
3	ARM calibration error	ARM
4	Monitor interrupt	ARM
5	Tx1 power amplifier Protection Error	Transmitter
6	Tx2 power amplifier Protection Error	Transmitter
7	Low Power Clock PLL Lock indicator	Lower Power Clock PLL
8	RF PLL 1 Lock indicator	RF PLL1
9	RF PLL 2 Lock indicator	RF PLL2
10	Aux PLL Lock indicator	Aux PLL
11	Clock PLL Lock indicator	Clock PLL
12	Main clock 1105 MCS	Clock Distribution
13	Main clock 1105 Second MCS	Clock Distribution
14	RX1 LSSI MCS	RX SSI
15	RX2 LSSI MCS	RX SSI
16	Main Stream Processor Error	Stream Processor
17	Stream Processor 0 Error	Stream Processor
18	Stream Processor 1 Error	Stream Processor
19	Stream Processor 2 Error	Stream Processor
20	Stream Processor 3 Error	Stream Processor
21	Not used	
22	Not used	
23	Not used	
24	Not used	

Full control (via public API functions) is given to the user to set/get mask, sticky mask and status registers (although the status register is a read only). Hence the user can custom tailor solutions (recovery actions) to handle the different events/interrupts.

adi_adrv9001_gpio_GpIntMask_Set() can be used to mask the corresponding interrupt events after device initialization. When a rising edge is detected on the GP_INT pin, the baseband processor should call the API command adi_adrv9001_gpio_GpIntStatus_Get() to find out which interrupt sources trigger the interrupt signal.

AUXILIARY CONVERTERS AND TEMPERATURE SENSOR

The ADRV9001 device features auxiliary data converters including four 12-bit auxiliary Digital-to-Analog converters (AuxDAC) and four 10-bit auxiliary analog-to-digital converter (AUXADC_x). An integrated diode-based temperature sensor is available to readback the approximate die temperature of the device.

These features are included to simplify control tasks and reduce pin count requirements on the baseband processor by offloading these tasks to the ADRV9001. Example usage of the auxiliary converters include static voltage measurements performed by the AuxADC and flexible voltage control performed by the AuxDAC. This section outlines the operation of these features along with API command for configuration and control.

AUXILIARY DAC (AUXDAC)

There are four, independent, 12-bit AuxDACs integrated in the ADRV9001. The auxiliary DACs have an output voltage of approximately 0.05 V to VAGPIO_1P8 – 0.05V. The AuxDACs use the enumeration adi_adrv9001_AuxDac_e when referenced in the API. The pins used for the AuxDAC features are listed in Table 91.

Table 91. AuxDAC Pin Mapping and adi_adrv9001_AuxDac_e Enum Description

Aux DAC Number	Pin Name	Pin Number	Enum Name
AUXDAC[0]	AGPIO_0	F12	ADI_ADRV9001_AUXDAC0
AUXDAC[1]	AGPIO_1	F10	ADI_ADRV9001_AUXDAC1
AUXDAC[2]	AGPIO_2	F3	ADI_ADRV9001_AUXDAC2
AUXDAC[3]	AGPIO_3	F5	ADI_ADRV9001 _AUXDAC3

The capacitive load of the AuxDAC pins should not exceed more than 100 pF otherwise stability issues may occur.

The AuxDAC uses the AGPIO pins on the device. Conflicts between AGPIO and AuxDAC functionality may occur. In case of these conflicts, the AuxDAC takes precedence over all other AGPIO functionality when AuxDAC is enabled for a specific pin. When the AuxDAC is disabled, the configured AGPIO functionality is applied. The AuxDAC can be enabled one pin at a time to allow flexibility between AuxDAC and AGPIO functionality.

The AuxDAC is typically used in applications requiring analog control signals. The data interface used to set the output level of the AuxDAC is SPI (API) or internal LUT (power amplifier RAMP function enabled) based. There is no CMOS/LVDS data interface to provide input data to the AuxDAC.

The (ideal) output voltage expressed on the AuxDAC is based on the following equation (in volts):

$$V_{AUXDAC} = 0.9 + \frac{AuxDacValue - 2048}{4096} \times 1.7$$

where *AuxDacValue* is the 12-bit digital code applied to the AuxDAC.

The AuxDAC is not a precision converter, it is best used in feedback systems. Above AuxDAC output equation is to be characterized, AuxDAC output voltage versus input codes for a full range code sweep of the AuxDAC will be added in the future after necessary characterization.

AuxDAC API Programming

A set of API commands are provided to set and inspect the AuxDAC, which is summarized in Table 92.

Table 92. AuxDAC API list

AuxDAC Function Name	Description
adi_adrv9001_AuxDac_Configure	Sets the configuration for AuxDACs, enable/disable the selected AuxDAC
adi_adrv9001_AuxDac_Inspect	Gets the configuration of selected AuxDAC
adi_adrv9001_AuxDac_Code_Set	Sets 12 bit DAC code of selected AuxDAC
adi_adrv9001_AuxDac_Code_Get	Reads the DAC word of selected AuxDAC

AUXILIARY ADC (AUXADC)

ADRV9001 contains four dedicated AuxADCs denoted as: AUXADC_0, AUXADC_1, AUXADC_2, and AUXADC_3. The AuxADC is a 10-bit output delta-sigma converter useful for measuring DC and near-DC signals (<8 kHz). The input voltage range of the AuxADC is 150 mV to 800 mV. Readback of the AuxADC input voltage is performed using API commands.

The AuxADCs use the enumeration adi_adrv9001_AuxAdc_e when referenced in the API. The pins used for the AuxADC features are listed in Table 93.

Table 93. AuxADC Pin Mapping and adi_adrv9001_AuxAdc_e Enum Description

Aux DAC Number	Pin Name	Pin Number	Enum Name
AUXADC[0]	AUXADC_0	H11	ADI_ADRV9001_AUXADC0
AUXADC[1]	AUXADC _1	B8	ADI_ADRV9001_AUXADC1
AUXADC[2]	AUXADC _2	B7	ADI_ADRV9001_AUXADC2
AUXADC[3]	AUXADC _3	H4	ADI_ADRV9001_AUXADC3

The AuxADC clock rate is set to 30.72 MHz (or close when ADRV9001 ARM system clock is changed) to get the best ADC performance. There are no on chip calibrations executed for the AuxADC, the ADC accuracy is limited to the accuracy of the supply reference. A simplified procedure for measuring and accounting for the AuxADC gain and offset error is performed, those AuxADC gain and offset errors are used to compensate the AuxADCs measure results.

AuxADC API Programming

AuxDAC relative API commands are summarized in Table 94, users can find the detail in API help file.

Table 94. AuxADC API list

AuxDAC Function Name	Description
adi_adrv9001_AuxAdc_Configure	Sets to enable/disable the selected AuxADC
adi_adrv9001_AuxAdc_Inspect	Gets the configuration of selected AuxADC
adi_adrv9001_AuxAdc_Voltage_Get	Sets 12 bit DAC code of selected AuxADC
adi_adrv9001_AuxDac_Code_Get	Reads the ADC code of selected AuxADC and converts to mV

TEMPERATURE SENSOR

The device features a temperature sensor that measures the temperature on the die. The temperature sensor uses an ADC similar to the AuxADC, however it is a separate instantiation and has no connections to a device pin.

The initiation of a temperature measurement is performed without user intervention by the ARM processor. The user can retrieve this measurement results in centigrade through an API command adi_adrv9001_Temperature_Get().

RF PORT INTERFACE INFORMATION

The RF ports of the ADRV9001 consisting of the transmit (TX1±, TX2±) and the receive ports (RX1A±, RX1B±, RX2A±, and RX2B±) support a operational frequency range from 30 MHz to 6 GHz. This wide frequency range fulfils the requirements of many application space. However, for optimized performance within a narrowband with minimal amplitude roll off and optimized linearity and noise performance, the RF ports will must be impedance-matched.

This User Guide provides some example impedance matching networks for a selection of frequency bands. Locating a balun/transformer to cover the entire frequency range of the ADRV9001 with minimal phase and amplitude imbalance proves to be a challenging task given limited selection of commercially available baluns. For this reason, the example RF matching networks were chosen based on the available baluns/transformers and RF trace implementations on evaluation PCB.

The matching networks are divided into two categories, wideband and narrowband. The wideband matching networks cover a range of almost 3 GHz and possess more amplitude roll off as compared to the narrowband match. This roll off is often dominated by the characteristic performance of the balun/transformer. For more optimized performance within a narrowband, the frequency specific narrowband matches are recommended.

TRANSMIT PORTS: TX1± AND TX2±

The ADRV9001 uses a direct conversion transmitter architecture consisting of two identical and independently controlled TX channels. The differential output impedance of transmitter outputs is matched to 50 Ω as shown in Figure 209. Additionally, the TX outputs must be biased to a low noise 1.8 V supply.

RECEIVE PORTS: RX1A±, RX1B±, RX2A±, AND RX2B±

The ADRV9001 has two RF inputs for each receiver to accommodate different matching for each RF bands of interest. The mixer architecture is very linear and inherently wideband which facilitates wideband impedance matching. The differential input impedance of the RX inputs are 100Ω as shown in Figure 210 and Figure 211.

When selecting a balun/transformer for the receive paths a 2:1 impedance transformation is required to accommodate the 50 Ω single-ended impedance to 100 Ω differential impedance as required by the ADRV9001 RX inputs.

The receiver input pins are self-biased internally to 650mV and therefore will require AC coupling/DC blocking capacitors at its inputs.

EXTERNAL LO PORTS: LO1± AND LO2±

Two external LO inputs (LO1 and LO2) can be applied to ADRV9001 and each external LO signal can be used for any of two receivers or two transmitters instead of internally generated LO signal. AC-coupling interface is needed for both positive and negative sides of external LO input pins which are internally biased. Similar to RX RF interface, a balun with 2:1 impedance transformation is necessary to accommodate the 50 Ω single-ended impedance to 100 Ω differential impedance as required by the ADRV9001 Ext LO inputs.

DEVICE CLOCK PORT: DEV_CLK1±

There are two low-frequency (below 100MHz) clock interface modes and a LVDS type clock interface mode that can support clock signal running as fast as 1GHz. For the high frequency clock interface, off-chip 100 ohm resistive termination will be required along with ac-coupling caps. More information is available on the subsequent section named connection for external device clock.

RF RX/TX PORTS IMPEDANCE DATA

This section provides the port impedance data for all transmitters and receivers in the ADRV9001 integrated transceiver. Please note the following:

- The reference plane for this data is the ADRV9001 ball pads.
- Single-ended mode port impedance data is not available. However, a rough assessment is possible by taking the differential mode port impedance data and dividing both the real and imaginary components by 2.

Contact Analog Devices Applications Engineering for the impedance data in Touchstone format.

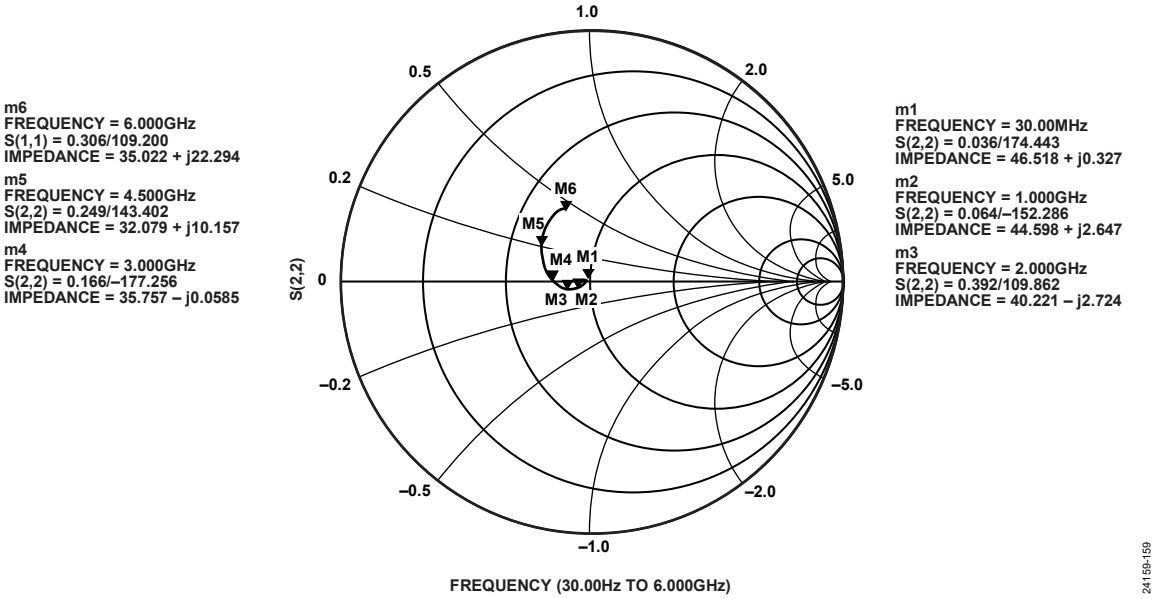


Figure 209. ADRV9001 TX port Series Equivalent Differential Impedance

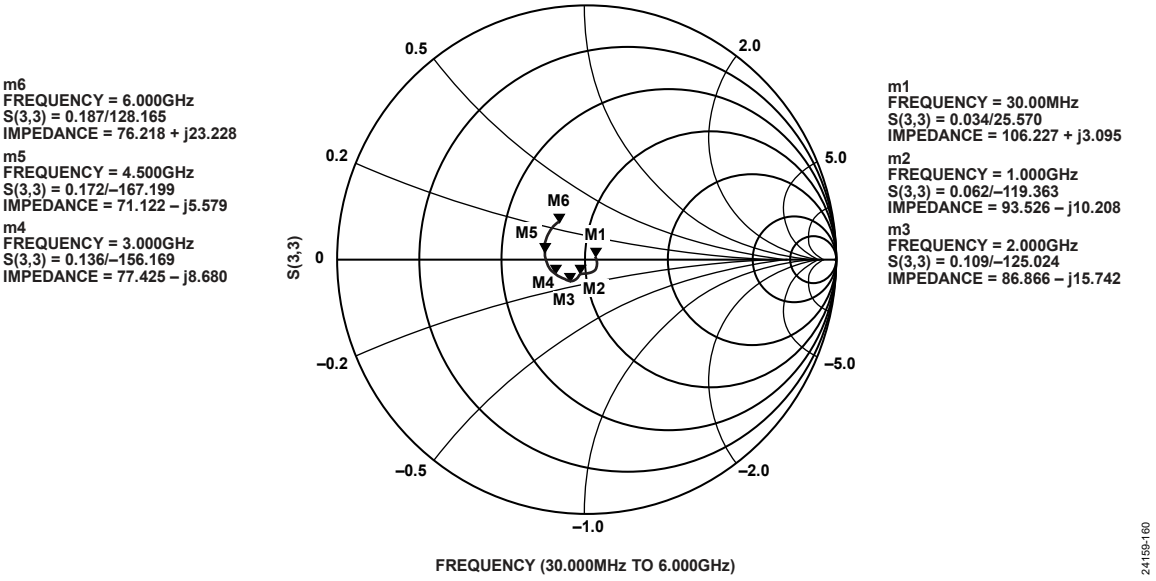


Figure 210. ADRV9001 RX A Port Series Equivalent Differential Impedance

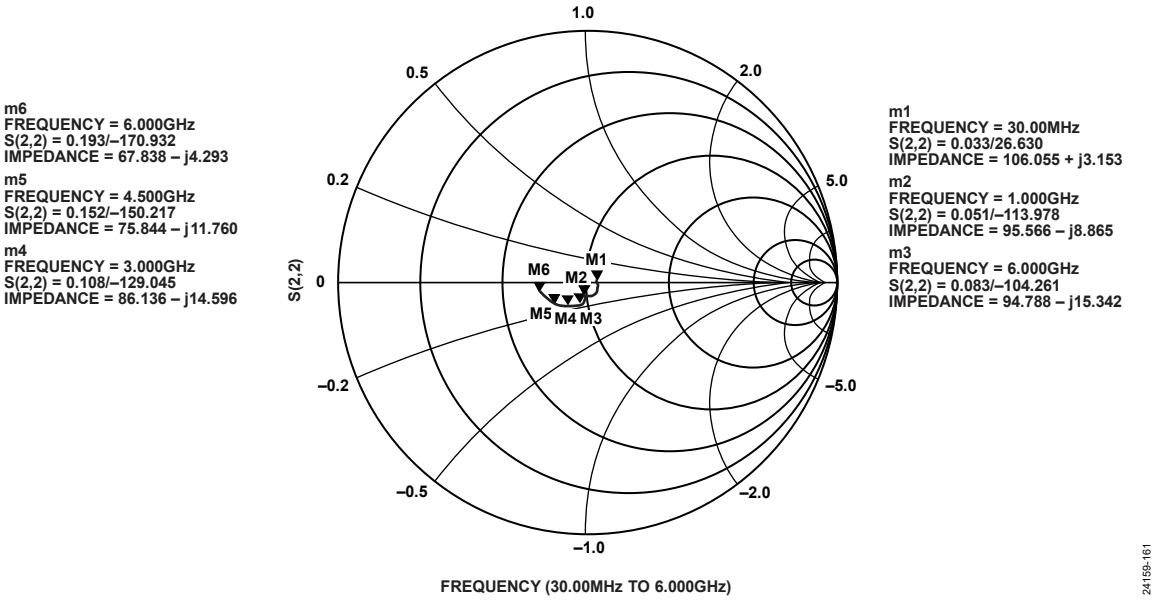


Figure 211. ADRV9001 RX B Port Series Equivalent Differential Impedance

ADS Setup Using DAC and SEDZ File

ADI supplies the port impedance as an *.s1p Series Equivalent Differential Z(impedance) file. This format allows simple interface to ADS by using the Data Access Component (DAC). In the below diagram Term1 is the single ended input or output and Term2 represents the differential input or output RF port on ADRV9001. The Pi on the single ended side and the differential Pi configuration on the differential side allows maximum flexibility in designing matching circuits, and is suggested for all design layouts as it can step the impedance up or down as needed with appropriate component population.

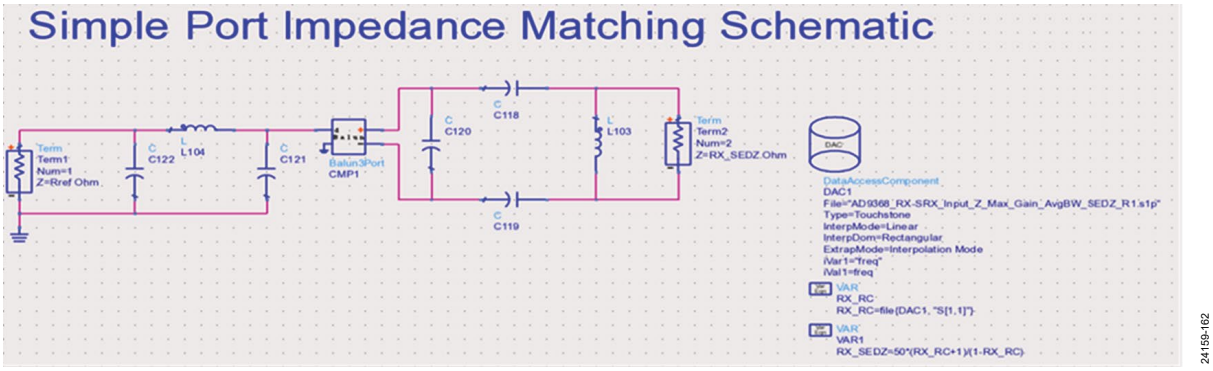


Figure 212. Simulation Setup in ADS with SEDZ S1P Files and DAC Component

Operation is as follows:

1. The DAC block reads the **rf port*.s1p** file. This is the device RF port reflection coefficient.
2. The two equations convert the RF port reflection coefficient to a complex impedance. The end result is the RX_SEDZ variable.
3. The RF port calculated complex impedance (RX_SEDZ) is used to define the Term2 impedance.
   a. Term2 is used in differential mode and Term1 is used in single-ended mode.

Setting up the simulation this way allows one to measure the S11, S22, and S21 of the 3-port system without complex math operations within the display page.

Note for highest accuracy, EM modelling result of the PCB artwork and S-parameters of the matching components and balun should be used in the simulations.

The first differential shunt reactive component such as L103 in Figure 212, is inserted to tune out parallel parasitic reactance of input impedance of the device. If ac-coupling cap is necessary, C118 and C119 can be used used for this purpose.

For a wideband match application, because of well controlled input/output impedance characteristics of ADRV9001 for entire range of its RX/TX operational frequency band, minimal matching network can be implemented to control undesirable impedance deviation typically associated with the high side of frequency range a balun operates. Additionally, by selecting a balun with the same differential

side termination impedance as the impedance of RX inputs and of TX outputs, a broad-band match can be accomplished by avoiding unnecessary impedance transformation network which is inherently band-limiting.

One can also consider adding additional differential series capacitive component on the balanced side of balun to facilitate ac-coupling and Pi match on both sides of the balun as shown below.

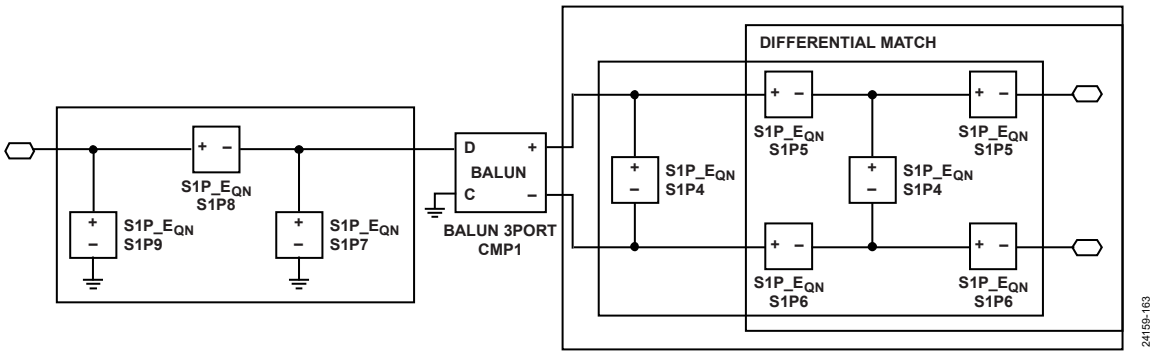


Figure 213. RF Matching Network with Additional Series AC-Coupling Capacitors

For a narrowband impedance match application to filter out signals outside of frequency band of interest, one can use Pi match technique for desired bandwidth of impedance match with a selected balun's terminal impedance. Pi match can be considered as two L match networks back to back and would allow independent control of Q and impedance ratio obtainable from a matching network. Narrowband matching network tuned for frequency bands of RX and TX can further improve out of band rejection of a transceiver for frequency duplexed systems.

GENERAL RECEIVER PORT INTERFACE

ADRV9001 has two independent receive input channels(Rx1 and Rx2). Both Rx channels can support up to 40MHz bandwidth and use differential signalling interface. The differential input signals would be applied to an integrated mixer. The mixer input pins are internally biased to 0.65 Volt and would must be AC coupled depending on the common mode voltage level of the external circuit

Important considerations for the receiver RF port interface are as follows:

1. Device to be interfaced: filter, balun, T/R switch, external LNA, and so on Does this device represent a short to ground at DC?
2. Rx1 and Rx2 maximum safe input power is 18 dBm (peak).
3. Rx1 and Rx2 optimum DC bias voltage is 0.65 V bias to ground.
4. Board Design: reference planes, transmission lines, impedance matching, and so on. Figure 214 shows possible differential receiver port interface circuits. The options in Figure 214 and Figure 215 are valid for all receiver inputs operating in differential mode, though only the Rx1 signal names are indicated. Impedance matching may be necessary to obtain datasheet performance levels.

Receiver Port A/B Switching

ADRV9001 supports wide range of RF frequencies from 30 MHz to 6 GHz, however, typical RF balun will not support all frequencies but only cover a smaller range. There is a special feature to support switching Rx A and B ports, which allows user to use both of them as receiver channels and they can be switched at run time depending on the carrier frequency. Port A and B will be connected to different RF baluns and support different RF frequencies. The frequencies will be configurable.

Differential Receiver Input Interface Circuits

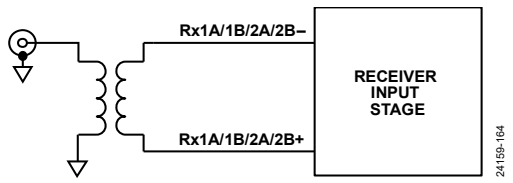


Figure 214. Differential Receiver Interface Using A Transformer

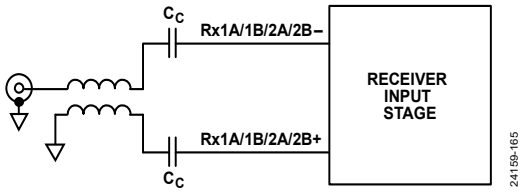


Figure 215. Differential Receiver Interface Using a Transmission Line Balun

Given wide RF bandwidth applications, SMD balun devices function well offering acceptable differential balance and insertion loss in a relatively small (0603, 0805) package.

Example of RX1 A Port Frequency Match Simulation

Reasonable approximation of return loss of a frequency matching network can be obtained with a simple S parameter simulation available in ADS without PCB artwork. Figure 216 illustrates a wide-band frequency match simulation setup in ADS for ADRV9001 RX1(2) A input pins in ADS for evaluating a possible configuration for a desired match to 3 GHz.

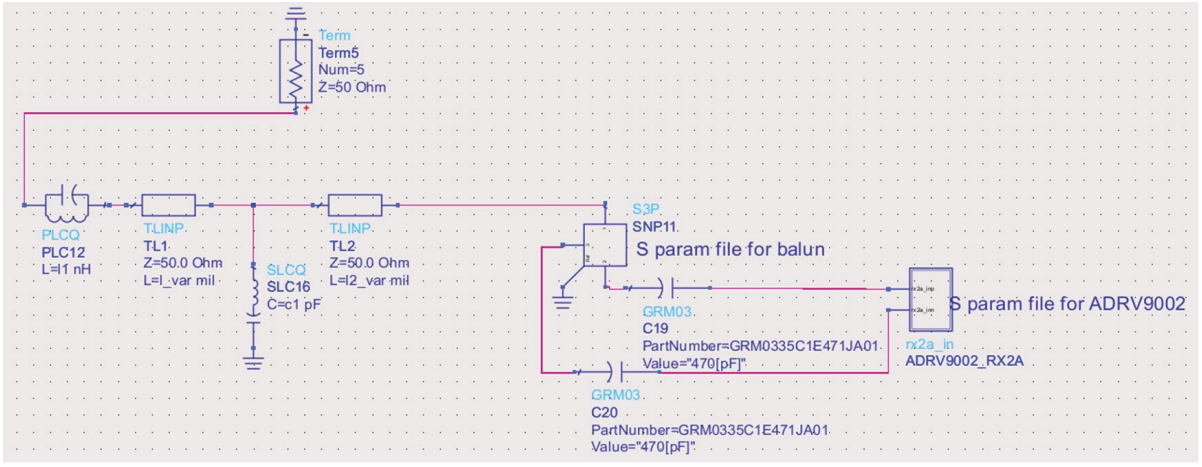


Figure 216. ADS Simulation Example Setup with Simple Physical Board Trace Models

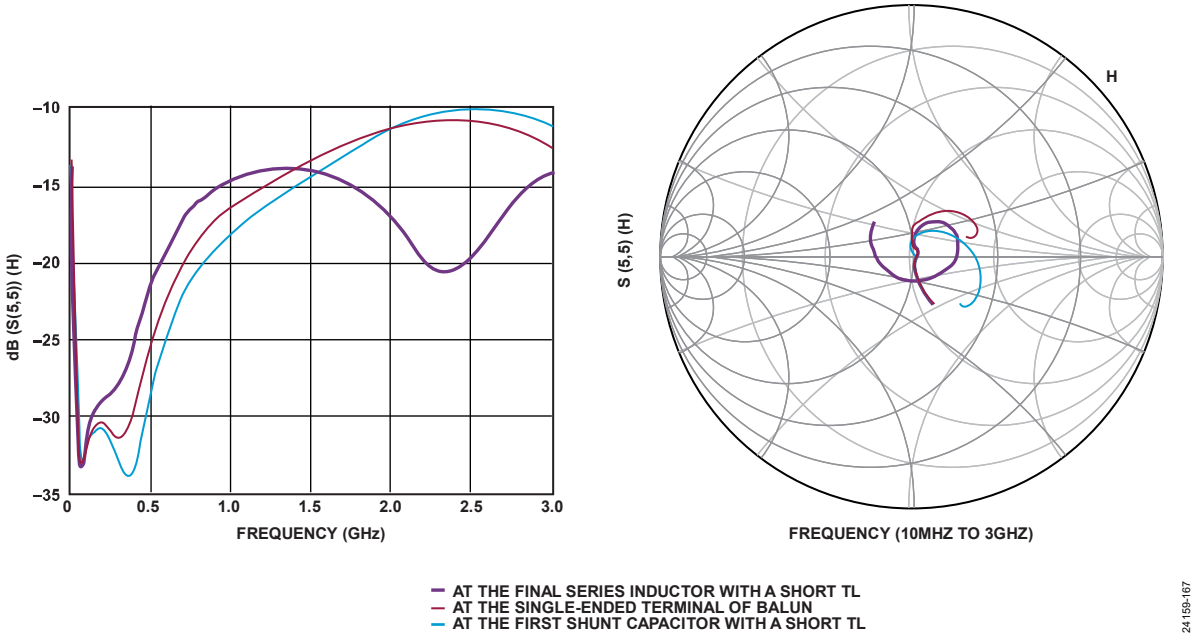


Figure 217. ADS Simulation Results of Return Loss Curve

S parameters for a selected balun and ac-coupling SMD type caps and ADRV9001 RX input impedance can be used to represent balun's balanced side interface to the device. Shunt and series matching component can be added with short TLs to represent possible PCB traces associated with these matching components on the single side of balun. Peaking of return loss at 2.5 GHz looking into the single-ended interface of balun has been reduced by a clockwise rotation of high frequency portion of S11 curve on the Smith chart by adding a shunt capacitor followed by series inductor after short transmission lines away from the balun's single-ended terminal.

GENERAL TRANSMITTER BIAS AND PORT INTERFACE

This section considers the dc biasing of the ADRV9001 transmitter (Tx) outputs and how to interface to each Tx port. ADRV9001 transmitters operate over a range of frequencies. At full output power, each differential output side draws approximately 100mA of DC bias current. The Tx outputs are DC biased to a 1.8 V supply voltage using either RF chokes (wire-wound inductors) or a transformer center tap connection.

Careful design of the dc bias network is required to ensure optimal RF performance levels. When designing the dc bias network, select components with low dc resistance (R_{DCR}) to minimize the voltage drop across the series parasitic resistance element with either of the suggested dc bias schemes suggested in Figure 218. The R_{DCR} resistors indicate the parasitic elements. As the impedance of the parasitics increase, the voltage drop (ΔV) across the parasitic element increases causing the transmitter RF performance (that is, PO 1 dB PO MAX, and so forth) to degrade. The choke inductance (LC) should be selected high enough relative to the load impedance such that it does not degrade the output power.

The recommended dc bias network is shown in Figure 219. This network has fewer parasitic and fewer total components.

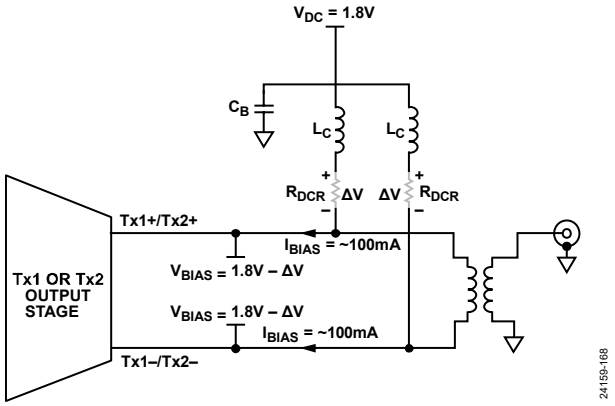


Figure 218. ADRV9001 RF DC Bias Configurations Depicting Parasitic Losses Due to Wire Wound Chokes

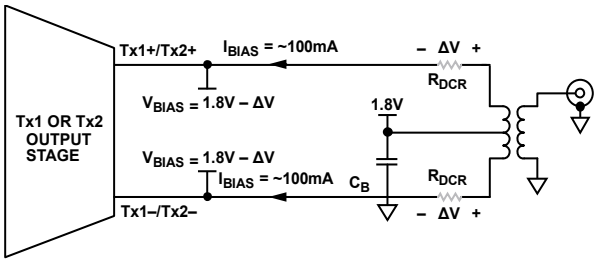


Figure 219. ADRV9001 RF DC Bias Configurations Depicting Parasitic Losses Due to Center Tapped Transformers

Figure 220 to Figure 223 identify four basic differential transmitter output configurations. Impedance matching networks (balun single-ended port) are most likely to be required to achieve optimum device performance from ADRV9001. Also, the transmitter outputs must be ac-coupled in most applications due to the dc bias voltage applied to the differential output lines of the transmitter.

The recommended RF transmitter interface is shown in Figure 220 featuring a center tapped balun. This configuration offers the lowest component count of the options presented.

Brief descriptions of the transmitter port interface schemes are provided as follows.

- Center tapped transformer passes the bias voltage directly to the transmitter outputs
- RF chokes are used to bias the differential transmitter output lines. Additional coupling capacitors (CC) are added in the creation of a transmission line balun
- RF chokes are used to bias the differential transmitter output lines and connect into a transformer
- RF chokes are used to bias the differential output lines that are ac-coupled into the input of a driver amplifier

Transmitter Interface Configurations

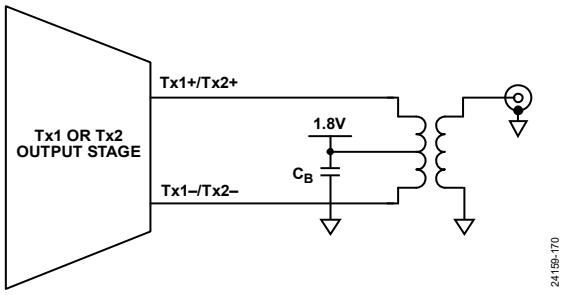


Figure 220. ADRV9001 RF Transmitter Interface Configuration A

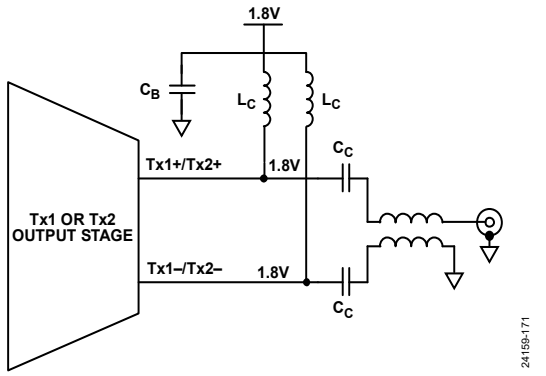


Figure 221. ADRV9001 RF Transmitter Interface Configuration B

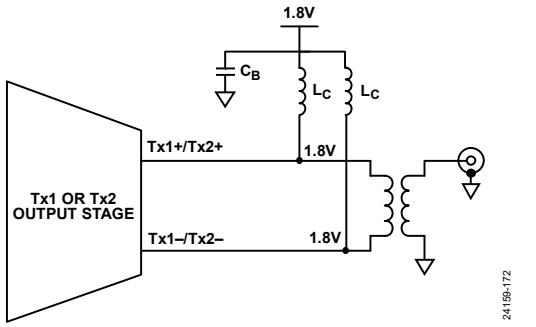


Figure 222. ADRV9001 RF Transmitter Interface Configuration C

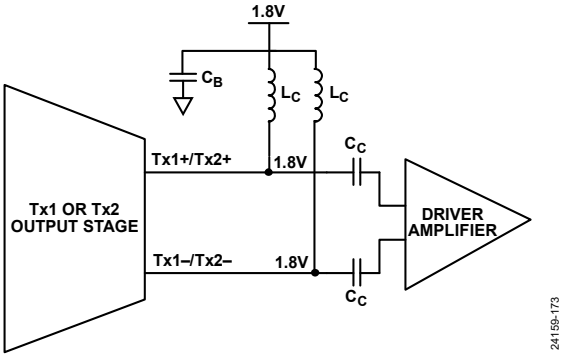


Figure 223. ADRV9001 RF Transmitter Interface Configuration D

If a Tx balun is selected that requires a set of external dc bias chokes, careful planning is required. It is necessary to find the optimum compromise between the choke physical size, choke dc resistance (RDCR) and the balun low frequency insertion loss. In commercially available dc bias chokes, resistance decreases as size increases. However, as choke inductance increases, resistance increases. Therefore, it is undesirable to use physically small chokes with high inductance as they exhibit the greatest resistance. For example: the voltage drop of a 500 nH, 0603 choke at 100 mA is roughly 50 mV.

Table 95. Sample Wire-Wound DC Bias Choke Resistance vs. Size vs. Inductance

Inductance (nH)	Resistance (Size: 0603)	Resistance (Size: 1206)
100	0.10	0.08
200	0.15	0.10
300	0.16	0.12
400	0.28	0.14
500	0.45	0.15
600	0.52	0.20

When selecting a dc bias choke inductor, shunting impedance of the choke inductor would must be high for TX frequency band in order to minimize its loading to outputs . Therefore, the self resonant frequency of the selected choke inductor must be higher than intended TX frequency.

Additionally, ADRV9001 provides built-in TX power ramp-up pattern generator to bring transmit power level in a pre-determined way to protect internal devices from sudden voltage spikes which may happen due to in-rush current passing through an external DC bias choke inductor. The supply side of choke inductors should also be tied to a capacitor with its self-resonant frequency higher that TX frequency. When both TX channel are active, each TX outputs should be tied to its own supply plane via a bias chock inductor or ferrite bead to reduce coupling between two TXs through the same supply feedline.

IMPEDANCE MATCHING NETWORK EXAMPLES

Impedance matching networks are required to achieve performance levels noted on the datasheet. This section provides example topologies and components used on the evaluation Board. The impedance matching networks provided in this section have not been evaluated in terms of Mean Time to Failure (MTTF) in high volume production. Please consult with component vendors for long-term reliability concerns. Additionally, please consult with balun vendors to determine appropriate conditions for DC biasing.

The schematics in Figure 201, Figure 202, and Figure 203 show two or three circuit elements in parallel marked DNI (Do Not Include). This was done on the evaluation board schematic to accommodate different component configurations for different frequency ranges. Only one set of SMD component pads are placed on the board to provide a physical location that can be used for the selected parallel circuit element. For example: R216, L216, and C216 components only have one set of SMD pads for one SMD component. The schematic shows that in a generic port impedance matching network, the series elements may be either a resistor, inductor or a capacitor whereas the shunt elements may be either an inductor or a capacitor. Only one component of each parallel combination is placed in a practical application. Note that in some matching circuits, some shunt elements may not be required. All components for a given physical location remain DNI in those particular applications.

RECEIVER RF PORT IMPEDANCE MATCHING NETWORK

RX1A± and RX2A± Impedance Matching Network

The ADRV9001 evaluation board uses both the top and bottom layers of the PCB evaluation platform to accommodate two balun footprints. The 0805 footprint accommodates the high frequency narrowband baluns while the backside accommodates the larger DB1627 case style transformer.

The PCB traces of the evaluation board were included in the simulation when designing the impedance match. Table 96 provides impedance matching networks specific to the ADRV9001 evaluation board. The component values apply to RX1A± and RX2A±.

RX1B± and RX2B± Impedance Matching Network

Both the RXA and RXB paths share the same input S-parameters. However, given the ball locations of the RXB paths being in an inner row and column and the layout of both paths are slightly different, the RXB path will have its own distinct impedance matching network that will be different from the RXA path. On the RXB path the low frequency DB1627 case style transformer is located on the top side of evaluation platform and the high frequency 0805 footprint transformer is located on the bottom of the board. This configuration is opposite of the RXA path. This was done to minimize coupling of the receive paths.

The PCB traces of the evaluation board were included in the simulation when designing the impedance match. Table 97 provides impedance matching networks specific to the ADRV9001 evaluation board. The component values applies to RX1B± and RX2B±.

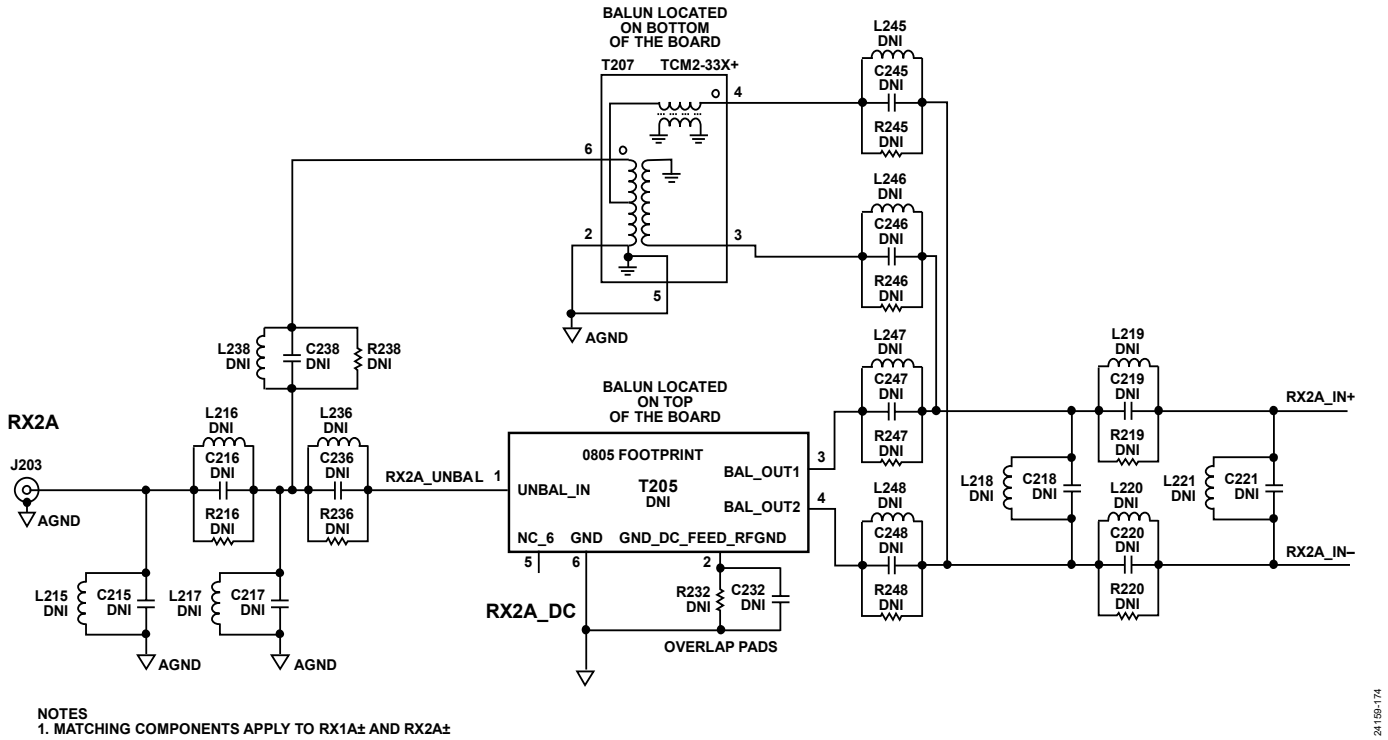


Figure 224. RX1A and RX2A Impedance Matching Network

Table 96. RX1A± and RX2A± Impedance Matching Network

Frequency	Balun	L/C/R 215	L/C/R 216	L/C 217	L/C/R 238	L/C/R 236	L/C/R 245 L/C/R 246	L/C/R 247 L/C/R 248	L/C 218	L/C/R 219 L/C/R 220	L/C 221	C232 R232
30 MHz – 3 GHz	MiniCircuits TCM2-33WX+	L215: DNI C215: DNI	L216: 1.2 nH AVX Ind (L0201 Series) C216: DNI R216: DNI	L217: DNI C217: DNI	L238: DNI C238: DNI R238: 0 Ω	L236: DNI C236: DNI R236: DNI	L245/246: DNI C245/246: DNI R245/246: 0 Ω	L247/248: DNI C247/248: DNI R247/248: DNI	L218: DNI C218: DNI	L219/220: DNI C219/220: 470 pF Murata (GRM03 Series) R219/220: DNI	L221: DNI C221: DNI	C232:DNI R232:DNI
3 GHz – 6 GHz	Johanson 4400	L215: DNI C215: DNI	L216: 0.82 nH AVX Ind (L0201 series) C216: DNI R216: DNI	L217: DNI C217: DNI	L238: DNI C238: DNI R238: DNI	L236: DNI C236: DNI R236: 0 Ω	L245/246: DNI C245/246: DNI R245/246: DNI	L247/248: 1.0 nH AVX Ind (L0201 series) C247/248: DNI R247/248: DNI	L218: DNI C218: 0.1 pF Murata (GRM03 series)	L219/220: DNI C219/220: DNI R219/220: 0 Ω	L221:DNI C221: DNI	C232: 1.5pF Murata (GJM03 Series) R232:DNI
30 MHz – 1 GHz	MiniCircuits TCM2-33WX+	L215: DNI C215: DNI	L216: 3.3 nH AVX Ind (L0201 series) C216: DNI R216: DNI	L217: DNI C217: 1.5pF Murata (GJM03 Series)	L238: DNI C238: DNI R238: 0 Ω	L236: DNI C236: DNI R236: DNI	L245/246: DNI C245/246: DNI R245/246: 0 Ω	L247/248: DNI C247/248: DNI R247/248: DNI	L218: DNI C218: DNI	L219/220: DNI C219/220: 470 pF Murata (GRM03 Series) R219/220: DNI	L221: DNI C221: DNI	C232:DNI R232:DNI
625 MHz – 2.8 GHz	Johanson 1720BL15B0100	L215: DNI C215: DNI	L216: 0.82 nH AVX Ind (L0201 series) C216: DNI R216: DNI	L217: DNI C217: DNI	L238: DNI C238: DNI R238: DNI	L236: DNI C236: DNI R236: 0 Ω	L245/246: DNI C245/246: DNI R245/246: DNI	L247/248: DNI C247/248: 10pF Murata (GRM03 Series) R247/248: DNI	L218: DNI C218: DNI	L219/220: DNI C219/220: DNI R219/220: 0 Ω	L221: DNI C221: DNI	C232:DNI R232:DNI

Frequency	Balun	L/C/R 215	L/C/R 216	L/C 217	L/C/R 238	L/C/R 236	L/C/R 245 L/C/R 246	L/C/R 247 L/C/R 248	L/C 218	L/C/R 219 L/C/R 220	L/C 221	C232 R232
2.8 GHz – 5 GHz	Anaren BD3150L50100AAF	L215: DNI C215: 0.3 pF Murata (GJM03 Series)	L216: 1.2 nH AVX Ind (L0201 series) C216: DNI R216: DNI	L217: DNI C217: 0.3 pF Murata (GJM03 Series)	L238: DNI C238: DNI R238: DNI	L236: DNI C236: DNI R236: 0 Ω	L245/246: DNI C245/246: DNI R245/246: DNI	L247/248: DNI C247/248: 10pF Murata (GJM03 Series) R247/248: DNI	L218: DNI C218: DNI	L219/220: 1.2 nH AVX Ind (L0201 series) C219/220: DNI R219/220: DNI	L221: DNI C221: 0.1pF Murata (GRM03 Series)	C232:DNI R232:DNI
4.5 GHz– 6 GHz	Johanson 5400BL15B100	L215: DNI C215: 0.2 pF Murata (GJM03 Series)	L216: 1.0 nH AVX Ind (L0201 series) C216: DNI R216: DNI	L217: DNI C217: 0.3 pF Murata (GJM03 Series)	L238: DNI C238: DNI R238: DNI	L236: DNI C236: DNI R236: 0 Ω	L245/246: DNI C245/246: DNI R245/246: DNI	L247/248: DNI C247/248: 10pF Murata (GJM03 Series) R247/248: DNI	L218: DNI C218: DNI	L219/220: 1.2 nH AVX Ind (L0201 series) C219/220: DNI R219/220: DNI	L221: DNI C221: DNI	C232:DNI R232:DNI

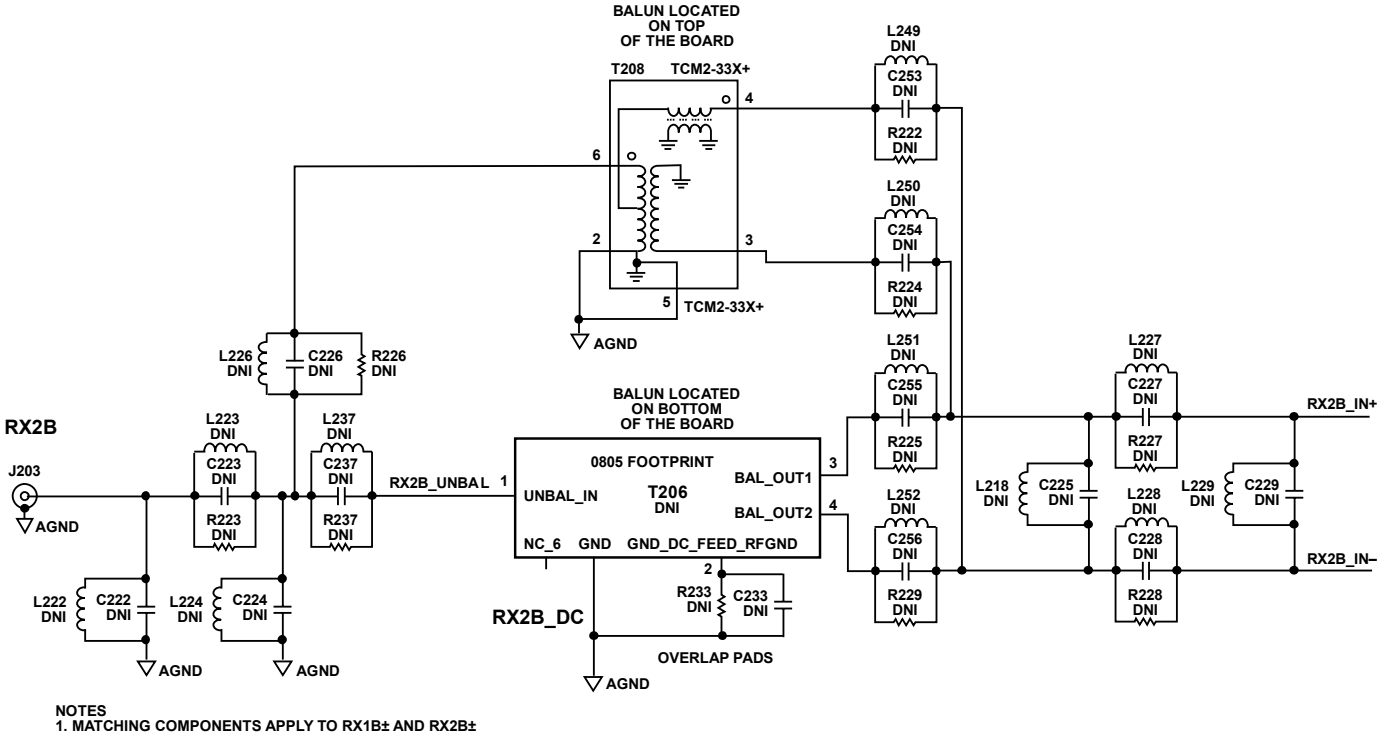


Figure 225. RX1B and RX2B Impedance Matching Networks

Table 97. RX1B± and RX2B± Impedance Matching Network

Frequency	Balun	L/C 222	L/C/R 223	L/C 224	L/C/R 226	L/C/R 237	L/C/R 249 L/C/R 250	L/C/R 251 L/C/R 252	L/C 225	L/C/R 227 L/C/R 228	L/C 229	C233 R233
30 MHz – 3 GHz	MiniCircuits TCM2-33WX+	L222: DNI C222: 0.5pF Murata (GJM03 Series)	L223: 2.2 nH AVX Ind (L0201 series) C223: DNI R223: DNI	L224: DNI C224: DNI	L226: DNI C226: DNI R226: 0 Ω	L237: DNI C237: DNI R237: DNI	L249/250: DNI C253/254: DNI R222/224: 0 Ω	L251/252: DNI C251/252: DNI R225/229: DNI	L225: DNI C225: DNI	L227/228: DNI C227/228: 470 pF Murata (GRM03 Series) R227/228: DNI	L229: DNI C229: DNI	C233: DNI R233: DNI
3 GHz – 6 GHz	Johanson 4400	L222: DNI C222: DNI Note: C257: 0.2pF Murata (GJM03 Series)	L223: 1.2 nH AVX Ind (L0201 series) C223: DNI R223: DNI	L224: DNI C224: DNI	L226: DNI C226: DNI R226: DNI	L237: DNI C237: DNI R237: 0 Ω	L249/250: DNI C249/250: DNI R222/224: DNI	L251/252: 0.33 nH AVX Ind (L0201 series) C251/252: DNI R251/252: DNI	L225: CNI C225: 0.1pF Murata (GRM03 Series)	L227/228: 1.0 nH AVX Ind (L0201 series) C227/228: DNI R227/228: DNI	L229: DNI C229: 0.1pF Murata (GRM03 Series)	C233: 1.4pF Murata (GJM03 Series) R233: DNI
30 MHz – 1 GHz	MiniCircuits TCM2-33WX+	L222: DNI C222: DNI	L223: DNI C223: DNI R223: 0 Ω	L224: DNI C224: DNI	L226: DNI C226: DNI R226: 0 Ω	L237: DNI C237: DNI R237: DNI	L249/250: DNI C249/250: DNI R222/224: 0 Ω	L251/252: DNI C251/252: DNI R225/229: DNI	L225: DNI C225: DNI	L227/228: DNI C227/228: 470 pF Murata (GRM03 Series) R227/228: DNI	L229: DNI C229: DNI	C233: DNI R233: DNI
625 MHz – 2.8 GHz	Johanson 1720BL15B0100	L222: DNI C222: 0.7pF Murata (GJM03 Series)	L223: 2.7 nH AVX Ind (L0201 series) C223: DNI R223: DNI	L224: DNI C224: 0.4pF Murata (GJM03 Series)	L226: DNI C226: DNI R226: DNI	L237: DNI C237: DNI R237: 0 Ω	L249/250: DNI C249/250: DNI R222/224: DNI	L251/252: DNI C251/252: DNI R225/229: 0 Ω	L225: DNI C225: DNI	L227/228: DNI C227/228: 470pF Murata (GRM03 Series) R227/228: DNI	L229: DNI C229: DNI	C233: DNI R233: DNI
2.8 GHz – 5 GHz	Anaren BD3150L50100AAF	L222: DNI C222: 0.5pF Murata (GJM03 Series)	L223: 1.5 nH AVX Ind (L0201 series) C223: DNI R223: DNI	L224: DNI C224: 0.5pF Murata (GJM03 Series)	L229: DNI C229: DNI R229: DNI	L237: DNI C237: DNI R237: 0 Ω	L249/250: DNI C251/252: DNI C249/250: DNI R222/224: DNI	L251/252: DNI C251/252: 100pF Murata (GRM03 Series) R251/252: DNI	L225: DNI C225: DNI	L227/228: 0.82 nH AVX Ind (L0201 series) C227/228: DNI R227/228: DNI	L229: DNI C229: DNI	C233: DNI R233: DNI
4.5 GHz– 6 GHz	Johanson 5400BL15B100	L222: DNI C222: 0.2pF Murata (GJM03 Series	L223: 1.2 nH AVX Ind (L0201 series) C223: DNI R223: DNI	L224: DNI C224: 0.2pF Murata (GJM03 Series	L226: DNI C226: DNI Series R226: DNI	L237: DNI C237: DNI R237: 0 Ω	L249/250: DNI C249/250: DNI R222/224: DNI	L251/252: DNI C251/252: 30pF Murata (GRM03 Series) R251/252: DNI	L225: DNI C225: DNI	L227/228: 0.68 nH AVX Ind (L0201 series) C227/228: DNI R227/228: DNI	L229: DNI C229: DNI	C233: DNI R233: DNI

RECEIVER RF PORT IMPEDANCE MATCH MEASUREMENT DATA

Receiver RF Port Impedance Match Measurement Data for 30 MHz to 3 GHz Band Match

Return loss was measured on RX1(2)A and RX1(2)B RF ports of eval boards and plotted below; blue and pink curves represent four different return loss measurements and black dotted line represents simulated return loss curve on Figure 226 and Figure 227. Simulated insertion loss curve including balun loss is plotted on Figure 228.

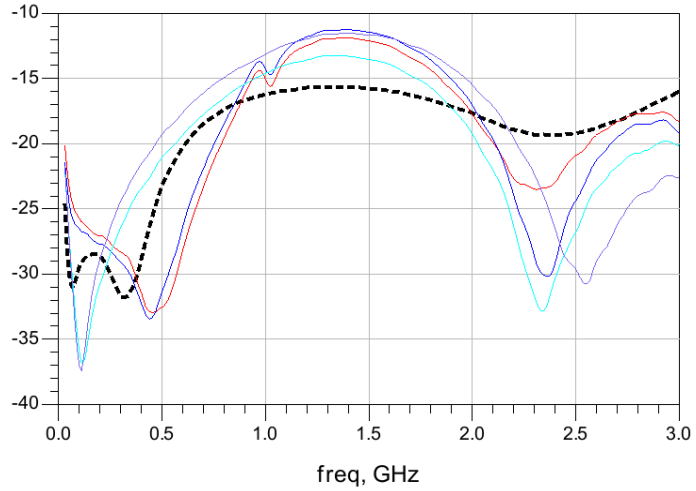


Figure 226. Return Loss of RX1(2)A Port

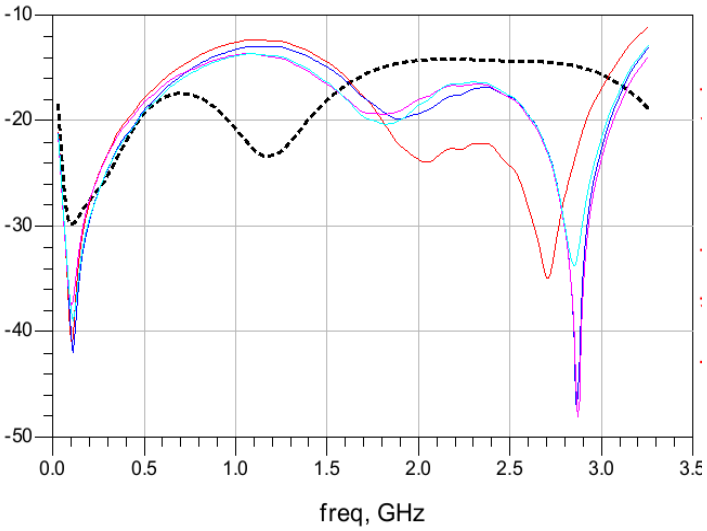


Figure 227. Return Loss of RX1(2)B Port

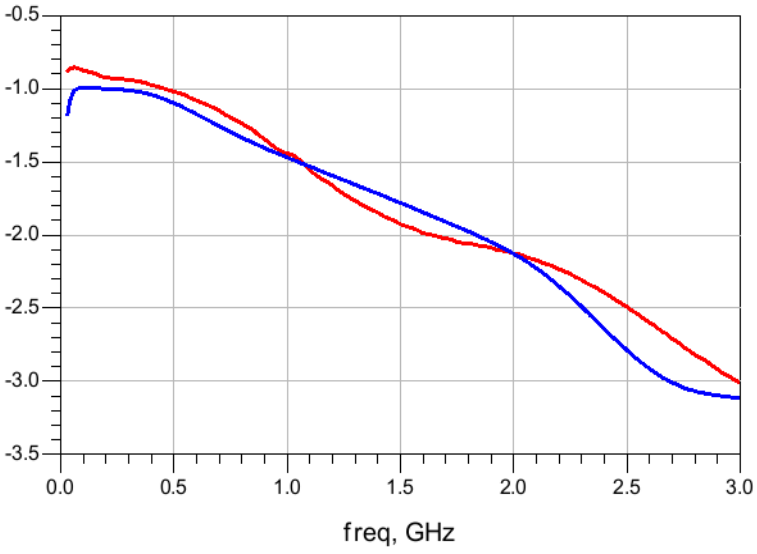


Figure 228. Insertion Loss – Simulated RX1(2)A Port – Red Curve RX1(2)B Port – Blue Curve

TRANSMITTER RF PORT IMPEDANCE MATCHING NETWORK

TX1± and TX2± Impedance Matching Network

For the TX path, the ADRV9001 evaluation board uses both the top and bottom layers of the PCB evaluation platform to accommodate two balun footprints. The 0805 footprint accommodates the high frequency narrowband baluns while the backside accommodates the larger AT224-1A case style transformer.

The ADRV9001 evaluation board provides two options in providing the DC common mode bias for the TX outputs. For transformers that provide a DC feed pin, this can be used to bias the TX output. For transformers that do not provide a DC feed pin, the TX outputs are biased to 1.8 V through pull up inductors. Only one bias option should be chosen, and provisions should be made to disable the unused path.

The PCB traces of the evaluation board were included in the simulation when designing the impedance match. Figure 229 and Table 98 provides impedance matching networks specific to the ADRV9001 evaluation board. The component values apply to TX1± and TX2±. Placement of C335 should be as close to dc feed pin of balun T302 as its purpose is to eliminate TX spectrum spurs and dampen the transients. Ground terminal of C335 should be tied to a ground plane and the cap should be oriented in the same direction of ground plane surrounding TX input trace so that the return current forms as small a loop as possible with the ground plane.

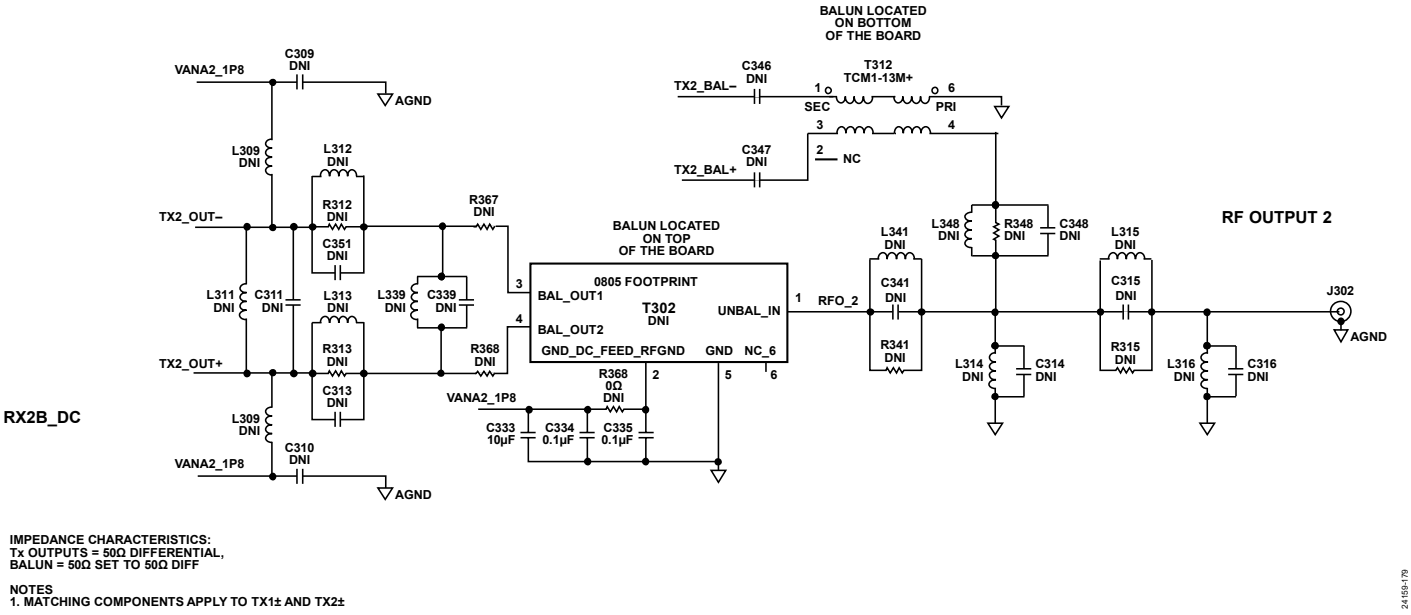


Figure 229. TX1 and TX2 Impedance Matching Network

Table 98. TX1± and TX2± Impedance Matching Network

Frequency	Balun	L/C 311	L309 L310	C309 C310	L/C/R 312 L/C/R 313	L339 C339	C346/347 R367/368	R361	C333 C334 C335	L/C/R 341	L/C/R 348	L/C 314	L/C/R 315	L/C 316
30 MHz – 3 GHz	MiniCircuits TC-1-13M+	L311: DNI C311: 0.3 pF (Murata GJM03)	L309/310: 220 nH (CoilCraft LQW18AN)	C309/C310 : 10 nF (Murata GRM03)	L312/313: 1.2 nH AVX Ind L0201 Series C312/313: DNI R312/313: DNI	L339: DNI C339: 0.8 pF (Murata GJM03)	C346/347 : 330 pF (Murata GRM03) R367, R368: DNI	R361: DNI	C333: DNI C334: DNI C335: DNI	L341: DNI C341: DNI R341: DNI	L348: 2.2 nH AVX Ind (L0201 Series C348: DNI R348: DNI	L314: DNI C314: 1pF (Murata GJM03)	L315: 2.2 nH AVX Ind (L0201 Series) C315: DNI R315: DNI	L316: DNI C316: DNI
3 GHz – 6 GHz	Johanson 4400	L311: DNI C311: 0.2 pF (Murata GJM03)	L309/310: DNI	C309/C310 : DNI	L312/313: 0.68 nH AVX Ind (L0201 Series) C312/313: DNI R312/313: DNI	L339: DNI C339: 0.3 pF (Murata GJM03)	C346/347: DNI R367/368 : 0 Ω	R361: 27 nH Murata Ind (LQW18)	C333: 10uF C334: 0.1uF C335: 2.7pF (Murata GJM03)	L341: 0.33nH AVX Ind (L0201 Series) C341: DNI R341: DNI	L348: DNI C348: DNI R348: DNI	L314: DNI C314: 0.3 pF (Murata GJM03)	L315: 0.82 nH AVX Ind (L0201 Series) C315: DNI R315: DNI	L316: DNI C316: DNI

Frequency	Balun	L/C 311	L309 L310	C309 C310	L/C/R 312 L/C/R 313	L339 C339	C346/347 R367/368	R361	C333 C334 C335	L/C/R 341	L/C/R 348	L/C 314	L/C/R 315	L/C 316
30 MHz – 1 GHz	MiniCircuits TC-1-13M+	L311: DNI C311: 1.4 pF (Murata GJM03)	L309/310: 220 nH (CoilCraft LQW18AN)	C309/C310 : 10 nF (Murata GRM03)	L312/313: DNI C312/313: DNI R312/313: 0 Ω	L339: DNI C339: DNI	C346/347: 330 pF (Murata GRM03) R367/368 : DNI	R361: DNI	C333: DNI C334: DNI C335: DNI	L341: DNI C341: : DNI R341: DNI	L348: DNI C348: R348: 0 Ω	L314: DNI C314: 1 pF (Murata GJM03)	L315: 3.9 nH (CoilCraft 0201 DS C315: DNI R315: DNI	L316: DNI C316: DNI
625 MHz – 2.8 GHz	Johanson 1720BL15B0050	L311: DNI C311: DNI	L309/310: DNI	C309/310: DNI	L312/313: DNI C312/313: DNI R312/313: 0 Ω	L339: DNI C339: 0.2 pF (Murata GJM03)	C346/347: DNI R367/368 : 0 Ω	R361: 120nH Murata Ind LQW15ANR12J00	C333: 10uF C334: 0.1uF C335: 47pF (Murata GRM03)	L341: DNI C341: DNI R341: 0 Ω	L348: DNI C348: DNI R348: DNI	L314: DNI C314: DNI	L315: 0.33 nH AVX Ind (L0201 Series) C315: DNI R315: DNI	L316: DNI C316: 0.1 pF (Murata GRM03)
2.8 GHz – 5 GHz	Anaren BD3150L50100AAF	L311: DNI C311: DNI	L309/310: 220 nH (Murata Ind LQW18AN)	C309/C310 : 10 nF (Murata GRM03)	L312/313: 1.2 nH AVX Ind (L0201 Series) C312/313: DNI R312/313: DNI	L339: DNI C339: DNI	C346/347: DNI R367/368: 20 pF Cap (Murata GJM03)	R361: 0 Ω	C333: DNI C334: DNI C335: DNI	L341: DNI C341: DNI R341: 0 Ω	L348: DNI C348: DNI R348: DNI	L314: DNI C314: 0.2 pF (Murata GJM03)	L315: 0.68 nH AVX Ind (L0201 Series) C315: DNI R315: DNI	L316: DNI C316: 0.2 pF (Murata GJM03)
4.5 GHz– 6 GHz	Johanson 5400BL15B0050	L311: DNI C311: 0.1 pF (Murata GRM03)	L309/310: 220 nH (Murata Ind LQW18AN)	C309/C310 : 10 nF (Murata GRM03)	L312/313: 0.82 nH (Murata LPQ03HQ) C312/313: DNI R312/313: DNI	L339: DNI C339: 0.1 pF (Murata GRM03)	C346/347: DNI R367/368 : 20 pF Cap (Murata GJM03)	R361: 0 Ω	C333: DNI C334: DNI C335: DNI	L341: DNI C341: DNI R341: 0 Ω	L348: DNI C348: DNI R348: DNI	L314: DNI C314: 0.3 pF (Murata GJM03)	L315: 0.82 nH AVX Ind (L0201 Series) C315: DNI R315: DNI	L316: DNI C316: 0.1 pF (Murata GRM03)

TRANSMITTER RF PORT IMPEDANCE MATCH MEASUREMENT DATA

Data for Transmitter RF Ports for 30 MHz to 3 GHz Band Match

Return loss was measured on TX RF ports of eval boards and plotted on Figure 230; blue and pink curves represent four different return loss measurements and black dotted line represents simulated return loss curve. Simulated Insertion loss including balun loss is plotted on Figure 231.

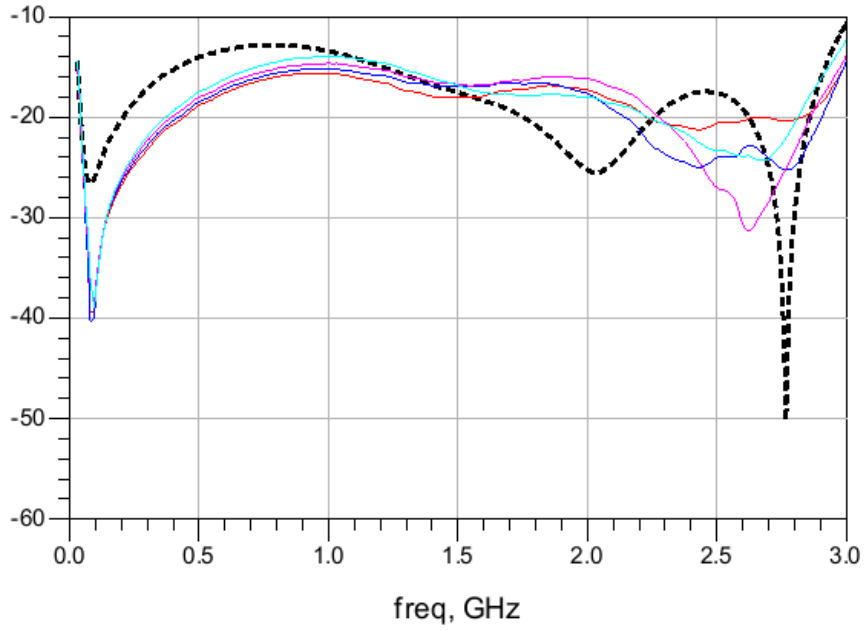


Figure 230. Tx1/Tx2 Return Loss

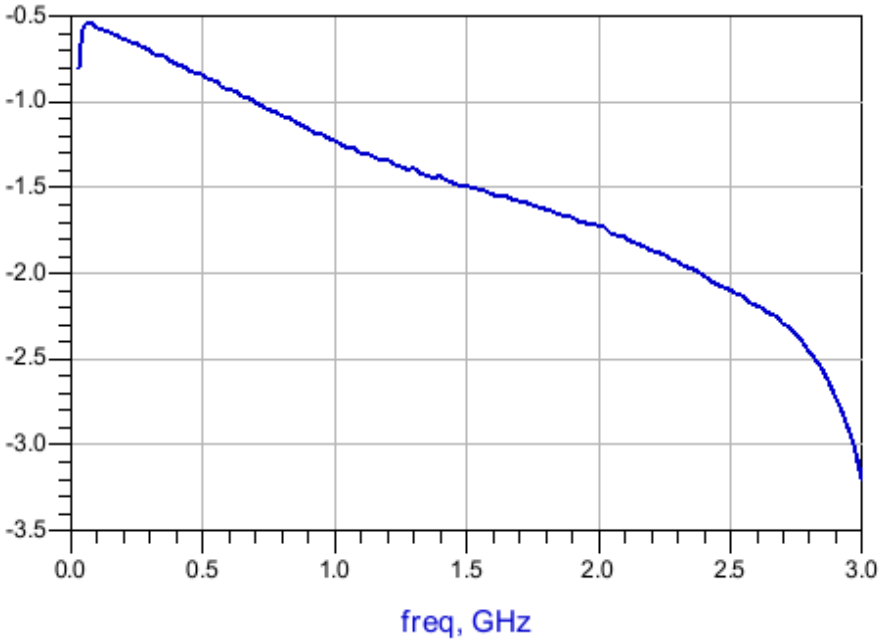


Figure 231. Tx1/Tx2 Insertion Loss, Simulated

EXTERNAL LO PORT IMPEDANCE MATCHING NETWORK

External LO1 and LO2 PORT can be used for injecting LO signal with very high spectral purity for internal receivers and transmitters. RF matching network for these ports would be implemented on single-ended and differential sides of balun to reduce insertion loss due to reflections at the desired LO frequency. Method of obtaining matching network is similar to RX and TX port matching. Depending on the selected divide ratio of ADRV9001 external LO input frequency divider SPI register setting, a band of frequency in which external LO matching network need to operate should be correctly derived by the division ratio chosen.

EXT LO Inputs

Unlike the internal synthesizers that always operate from 6 – 12 GHz regardless of the RF tune frequency, when an external LO pins are used the frequency applied must be a multiple of 2 times(i.e. 2x,4x,8x, and so on) of the desired RF signal channel frequency. The LO input signal is internally divided by a series of dividers to generate the required LO quadrature relationship at desired RF frequency of

upconvertion or downconversion for internal transmitters and receivers. Table 99 describes specification for EXT LO input pins when input pins are driven by a differential signal by use of balun.

Alternatively single-ended external LO signal source can be used for positive side of EXT LO pin with negative side of input pin terminated with a capacitor to provide ac ground. The frequency of external LO source should be set to 4x of desired RX or TX frequency with EXT LO divider configured to division of 4 for the best in-phase and quadrature generation of LO necessary for internal receivers and transmitters. Table 100 describes specification for single-ended EXT LO input source.

In general, higher power level of external LO signal applied gives better phase noise to some extent. The minimum input power level that satisfy RX/TX phase noise requirements with some margin should be used. Refer to Table 100 for power level recommendation.

Table 99. Specifications for ADRV9001 RF EXT LO Differential Input Pins

Parameter	Note	Min	Typical	Max	Unit
External LO frequency	FEXTLO	60		12000	MHz
RF Channel frequency	FCHANNEL	30		6000	MHz
External LO power	100 Ω matching Signal amplitude depends on FEXTLO frequency. Typical = 0dBm for FEXTLO <= 2GHz. From 2 GHz, add 0.5dB/GHz. For example, Typical = +3dBm for FEXTLO = 8GHz.	−6	−3 ~ +3	+6	dBm
Input Impedance	nominal, small signal input. Note below.		100		Ω
Differential Phase Error	Combined Differential Phase Error, Differential Amplitude Error, Duty Cycle Error, and Even Order Harmonic Content			±5	degrees
Differential Amplitude Error				1	dB
Duty Cycle Error				2	%
Even Order Harmonic Content				−50	dBc
EXT LO Source Modulus	Must match internal modulus on ADRV9001	8386560			

Table 100. Specifications for ADRV9001 RF EXT LO Single-ended Input Pin

Parameter	Note	Min	Typical	Max	Unit
External LO frequency	FEXTLO	60		2000	MHz
RF Channel frequency	FCHANNEL	30		1000	MHz
External LO power	50 Ω matching	0	+3	+6	dBm
Input Impedance	nominal, small signal input. Note below.		50		Ω
EXT LO Source Modulus	Must match internal modulus on ADRV9001	8386560			

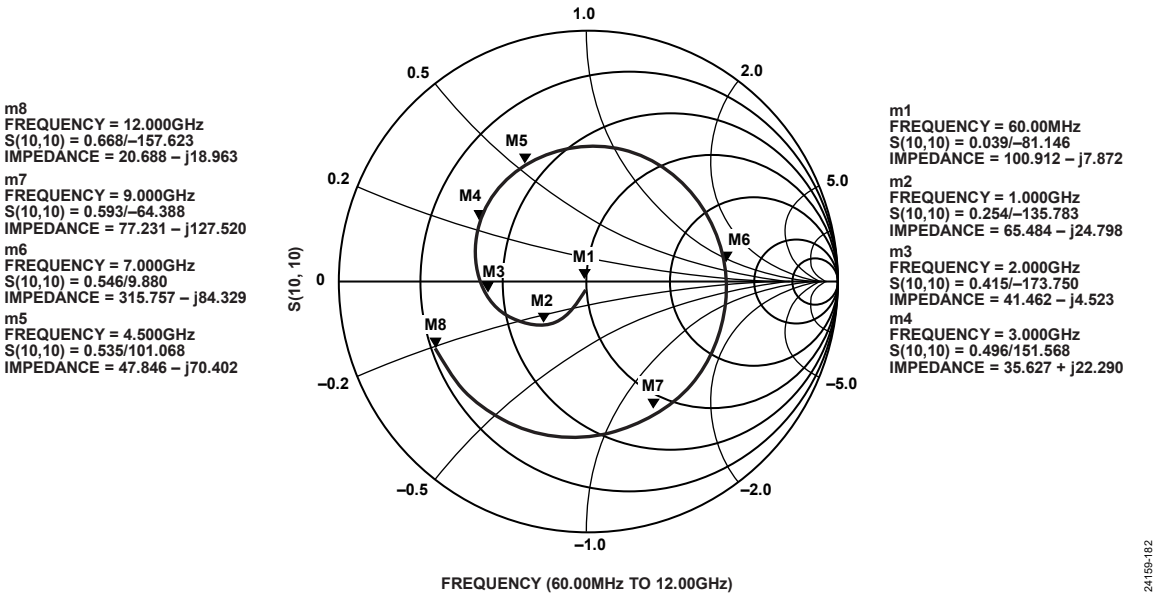


Figure 232. External LO Series Equivalent Differential Input Impedance

Care should be taken when selecting an on-board balun for this application. Combination of amplitude and phase balance performance of the balun can affect quadrature error performance. Additionally, duty cycle and differential second order harmonic distortion impacts the ability of to correct quadrature error. The recommended minimum requirement for Ext LO input pins is a combination of no more than 5 degree differential phase error, 1dB differential amplitude error, 2% duty cycle error, and less than −50 dBc even order harmonics(primarily 2nd order).

The ADRV9001 provides special mode of operation for external LO in range from 500 MHz to 1000 MHz. In that region it is possible to inject external LO that will produce RF Channel frequency with x1 multiplier.

For example:

For FEXTLO = 500 MHz the FCHANNEL = 500 MHz

For FEXTLO = 1000 MHz, the FCHANNEL = 1000 MHz

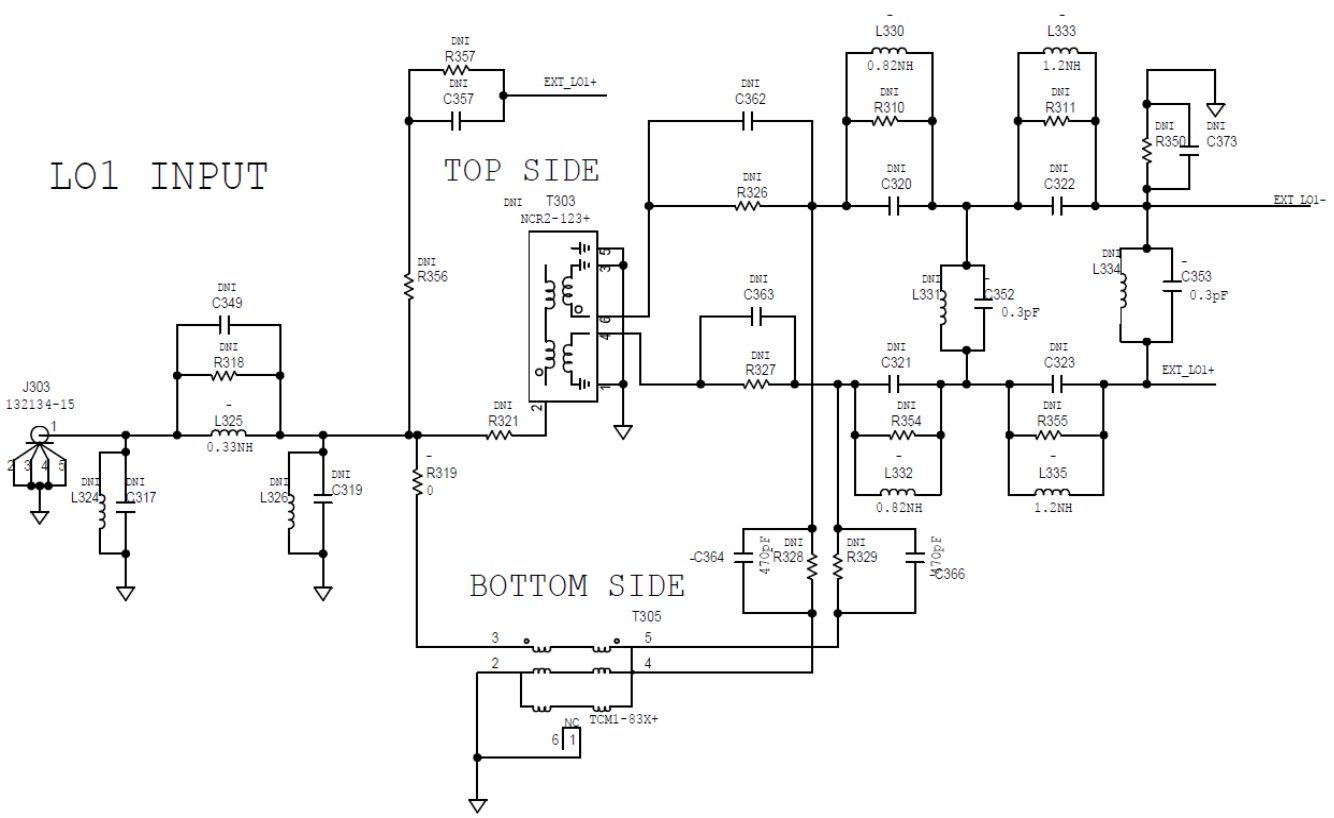


Figure 233. External LO Impedance Matching Network

Table 101. EXTLO1± and EXTLO2± Impedance Matching Network

Frequency	Balun	L324 C317	C349 L325 R318	L326 C319	R356 R321 R319	C357 R357	C362/363 R326/327	C364/366 R328/329	L330/332 C320/321 R310/354	L331 C352	L333/335 C322/323 R311/355	L334 C353	R350 C373
60 MHz – 6 GHz	MiniCircuits TCM1-83X+	L324: DNI C317: DNI	L325: 0.33nH (AVX L0201) C349: DNI R318: DNI	L326: DNI C319: DNI	R356: DNI R321: DNI R319: 0 Ω	C357: DNI R357: DNI	C362/363: DNI R326/327: DNI	R328/329: DNI C364/366: 470 pF (Murata GRM03)	L330/332: 0.82 nH (AVX L0201) C320/321: DNI R310/354: DNI	L331: DNI C352: 0.3 pF (Murata GJM03)	L333/335: 1.2 nH (AVX L0201) C322/323: DNI R311/355: DNI	L334: DNI C353: 0.3 pF (Murata GJM03)	R350: DNI C373: DNI
6 GHz – 12 GHz	MiniCircuits NCR2-123+	L324: DNI C317: DNI	L325: DNI C349: DNI R318: 0 Ω	L326: DNI C319: DNI	R356: DNI R321: 0 Ω R319: DNI	C357: DNI R357: DNI	C362/363: : 470 pF (Murata GRM03) R326/327: DNI	R328/329: DNI C364/366DNI	L330/332: DNI C320/321: DNI R310/354: 0 Ω	L331: DNI C352: DNI	L333/335: DNI C322/323: DNI R311/355: 0 Ω	L334: DNI C353: DNI	R350: DNI C373: DNI

A single-ended external LO signal can be applied by bypassing balun interface and installing appropriate impedance matching network comprised of L324/C317, C349/L325/R318, and L326/C319 and AC-coupling cap of C357. Additionally, a large cap for C373 should be installed to provide an ac-ground for the negative side of input pins to internal buffer circuitry.

EXTERNAL LO IMPEDANCE MATCH MEASUREMENT DATA

External RF Port Impedance Match Measurement Data for 60 MHz to 6 GHz Band Match

Return loss was measured on EXT LO RF ports of eval boards and plotted on Figure 234; blue and pink curves represent three different return loss measurements and black dotted line represents simulated return loss curve. Simulated Insertion loss including balun loss is plotted on Figure 235.

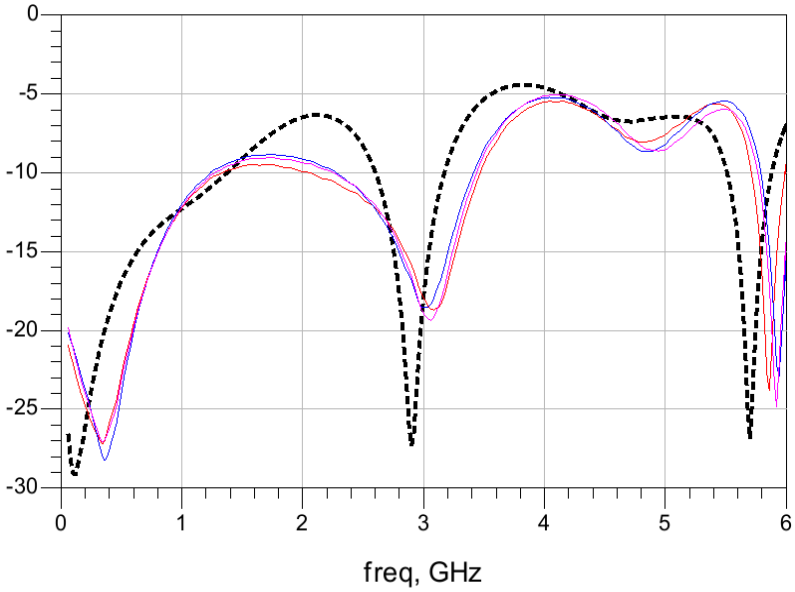


Figure 234. External LO1/ External LO2 Return Loss of Ext LO Port

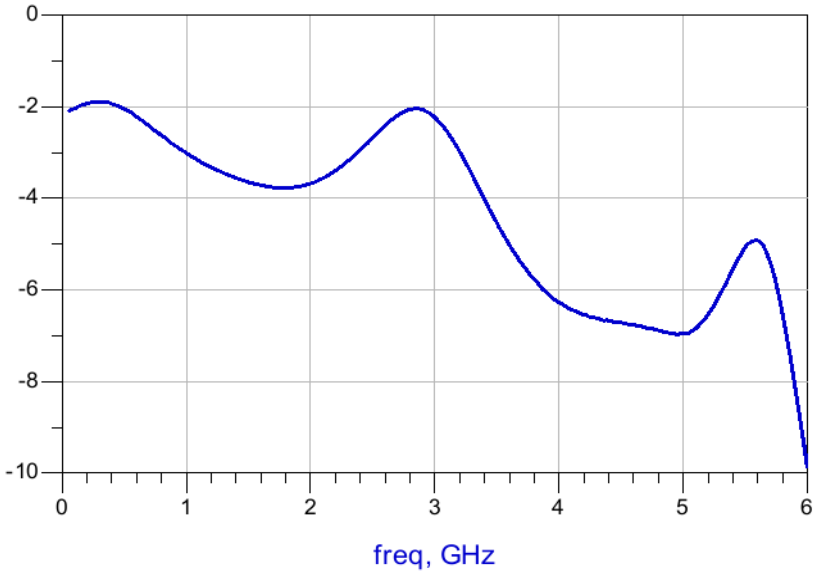


Figure 235. External LO1/ External LO2 Insertion Loss, Simulated

CONNECTION FOR EXTERNAL DEVICE CLOCK (DEV_CLK_IN)

ADRV9001 can accommodate 3 different types of external clock signals applied at device clock input pins. A differential low voltage differential signaling (LVDS) clock signal or a single-ended clipped sinewave clock signal from a TCXO can be applied to the device input pins. Furthermore, a crystal can be connected to device clock input pins to configure it as a crystal oscillator/driver by applying DC voltage into MODEA pin as shown below.

Table 102. Device Clock Input Interface Modes Description

Voltage Applied at MODEA Pin	Device Clock Input Electrical Interface	DEV_CLK_OUT Divider Value Applied to DEV_CLK_IN Signal	Note
0 V (grounded)	LVDS	/16	Up to 1GHz clock
0.45 V	CMOS or XTAL	/2	CM0S(10MHz to 80MHz) /XTAL(20 MHz to 80 MHz) with Nominal Gm multiplier = x8
0.9 V	CMOS or XTAL	/2	CM0S(10MHz to 80MHz) /XTAL(20 MHz to 80 MHz) with Nominal Gm multiplier = x6
1.35 V	CMOS or XTAL	/2	CM0S(10MHz to 80MHz) /XTAL(20 MHz to 80 MHz) with Nominal Gm multiplier = x2
1.8 V	CMOS or XTAL	/2	CM0S(10MHz to 80MHz) /XTAL(20 MHz to 80 MHz) with Nominal Gm multiplier = x4

By applying 1.8 V to MODEA pin A for CMOS interface mode, a clipped sinewave clock signal from a TCXO can be applied to pin named DEV_CLK_IN+(E7) via a AC coupling capacitor and pin DEV_CLK_IN-(E8) should be left unconnected as shown in Figure 237.

A Xtal should be connected to both DEV_CLK_IN+ and DEV_CLK_IN- pins with a DC voltage between 0.45 and 1.8 V applied to MODEA pin as shown in Figure 238.

When LVDS mode input clock interface is selected with MODEA pin grounded, an external clock is used as the reference clock for the RFPLL and the Clocking PLL on the device and thus must be a very clean clock source. Connect the external clock inputs to the DEV_CLK_IN+ (E7) and DEV_CLK_IN- (E8) balls via AC coupling capacitors and should be terminated with 100 Ω as shown in Figure 236. The inputs are biased on the device to a 200 mV voltage level. The input impedance plot over operating frequency is shown on Figure 216. The operational frequency range of the DEV_CLK signal is between 10 MHz and 1000 MHz. Ensure that the external clock peak-to-peak amplitude does not exceed 800mV (Note that either positive and negative side of differential input pins should not exceed 400mV peak-to-peak.). For best synthesizer performance, a high slew rate signal is best with fast rise and fall times.

Device Clock Interface Modes

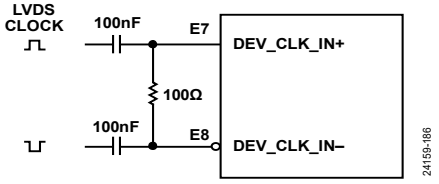


Figure 236. LVDS Interface Mode

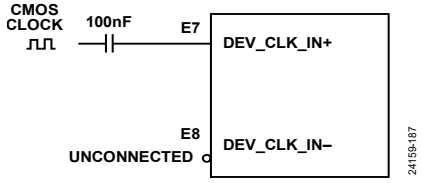


Figure 237. CMOS Interface Mode

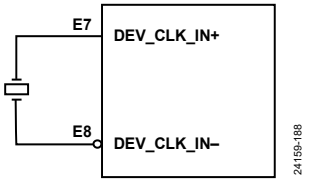


Figure 238. Crystal (XTAL) Interface Mode

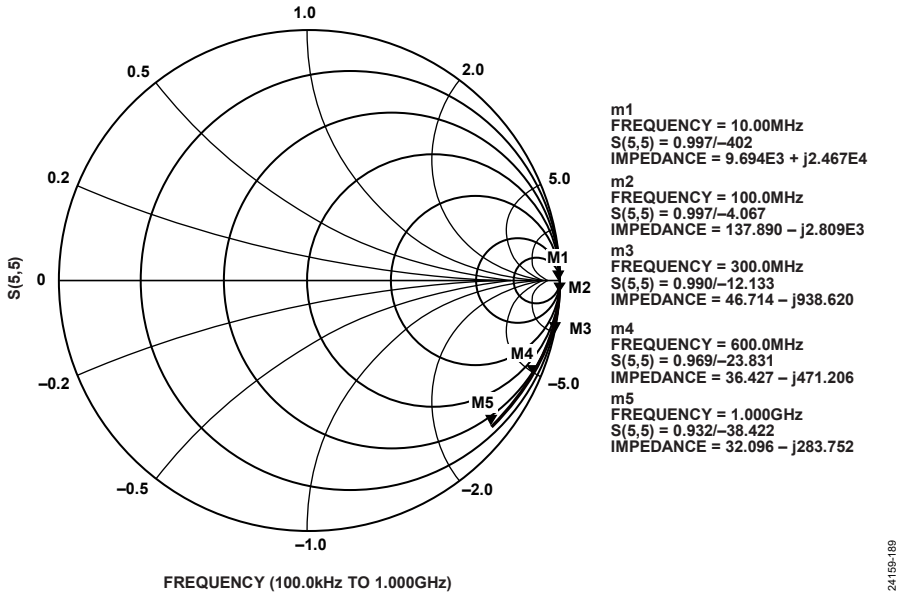


Figure 239. Device Clock Input Series Equivalent Differential Impedance

Device clock input board traces connected to device clock inputs balls should be implemented with stripline transmission lines using inner copper layers in PCB stackup. The frequency of device clock input signal can go as high as 1GHz and stripline transmission line approach will provide better signal integrity of clock signal especially at higher frequency as well as superior shielding of RF emission of device clock signal.

The DEV_CLK_IN signal is available on the DEV_CLK_OUT pin. Table 102 describes default division applied to DEV_CLK_IN signal after power up. Use can change this divider later on using API command. It should be noted that DEV_CLK_OUT pin is a CMOS type pin with 80MHz of its maximum frequency of operation. It is intended to be used to provide clock to BBIC or on-board microcontroller or audio CODEC type devices. It is not intended to be used by another RF sensitive IC.

For application where customer want to utilize internal RF LOs it is recommend to use 40MHz or above as a DEV_CLK frequency to get the best in band phase noise.

There is a known issue with DEV_CLK input working in CMOS mode. When applied DEV_CLK input signal is in range between 10MHz to 30MHz an internal RF LOs exhibit in band phase noise degradation of around 10dB .

If:

- this phase noise degradation is not acceptable and
- in customer end application it is mandatory to use DEV_CLK below 30MHz

then user should

- connect MODEA pin to GND which enables differential mode of operation for DEV_CLK input circuitry
- apply DEV_CLK as single ended to DEV_CLK_IN+ (E7 ball) and leave DEV_CLK_IN- (E8 ball) unconnected. Basically, the same hardware configuration as in CMOS mode, outlined in Figure 237.
- ensure that amplitude of applied signal does note exceed 1V peak-to-peak. Signal in range of 400mW peak-to-peak is recommended. The DEV_CLK_IN+ (E7 ball) inputs when operating in LVDS mode is biased on the device to around 200 mV voltage level. Maximum of 400mV peak-to-peak amplitude ensures that the external clock stays compliant with electrical specification of DEV_CLK_IN+ pin.

DEV_CLK_IN PHASE NOISE REQUIREMENTS

To prevent performance degradation, the DEV_CLK reference must be a very clean signal. Best performance from the synthesizer would result if the applied reference were ideal, however that is unrealistic. Table 103 lists the required phase noise of the DEV_CLK signal for a 1dB system PN degradation compared to an ideal DEVICE CLOCK. For different DEV_CLK frequencies, the table can be scaled appropriately. Clock source with phase noise performance outlined in Table 104 (or better) allows ADRV9001 to deliver datasheet performance. It should be noted that Table 103 provide reference information for ADRV9001 operating with LTE type standards. Each standard will determine its own DEV_CLK phase noise requirements. As an example, Table 104 provides recommendation for DEV_CLK

when ADRV9001 is intended to operate with LMR type standards. Ideally DEV_CLK phase noise requirement should be derived from customer specific application and its requirements set for adjacent channel rejection.

In general, using a higher phase noise source can degrade performance delivered by ADRV9001 transceiver.

Table 103. DEV_CLK_IN Phase Noise Requirements for 1dB system PN degradation compared to an ideal DEVICE CLOCK

Frequency Offset From Carrier	Narrow PLL Loop Bandwidth (Approximately 50 kHz) (Default, Typically <3 GHz)			Wide PLL Loop Bandwidth (Approximately 300 kHz) (User Configured, Typically >3 GHz)		
	122.88 MHz (dBc/Hz)	153.6 MHz (dBc/Hz)	245.76 MHz (dBc/Hz)	122.88 MHz (dBc/Hz)	153.6 MHz (dBc/Hz)	245.76 MHz (dBc/Hz)
100 Hz	−113.02	−111.08	−107.00	−114.02	−112.08	−108.00
1000 Hz	−125.02	−123.08	−119.00	−127.02	−125.08	−121.00
10 kHz	−133.02	−131.08	−127.00	−138.02	−136.08	−132.00
100 kHz	−137.02	−135.08	−131.00	−146.02	−144.08	−140.00
1 MHz	−133.02	−131.08	−127.00	−147.02	−145.08	−141.00
10 MHz	−104.02	−102.08	−98.00	−118.02	−116.08	−112.00

Table 104. DEV_CLK_IN Phase Noise Requirements for LMR Type Applications

Frequency Offset From Carrier	PLL Loop Bandwidth Optimized for LMR Type Applications, 38.4 MHz (dBc/Hz)
100 Hz	−106
1000 Hz	−151
10 kHz	−151
100 kHz	−151
10 MHz	−151

CONNECTION FOR MULTICHIP SYNCHRONIZATION (MCS) INPUT

A LVDS type MCS signal applied between MCS+(D7) and MCS-(D8) pins is used to provide time alignment synchronization for the both RF and datalink systems. Similar to device clock input signal, a clock source with fast rise and fall times should be used as MCS input signal. PCB traces for routing MCS signals should be implemented following guidelines that are similar to LVDS mode device clock input trace.

PRINTED CIRCUIT BOARD LAYOUT RECOMMENDATIONS

The ADRV9001 is a highly integrated RF agile transceiver with significant signal conditioning integrated onto one chip. Due to the integration complexity of the ADRV9001 and its high pin count, careful printed circuit board (PCB) layout is important to optimize performance. This section provides a checklist of issues to look for and guidelines on how to optimize the PCB to mitigate performance issues. The goal of this document is to help achieve the best possible performance from the ADRV9001 while reducing board layout effort. It is assumed that the reader is an experienced analog/RF engineer who understands RF PCB layout and has an understanding of RF transmission lines as well as low-noise analog design techniques. The ADRV9001 evaluation card is used as the reference for this information, but all guidelines are best practices that can be applied to other reference designs. This document provides guidelines for system designers and discusses the following issues relative to layout and power management.

- PCB material and stack up selection
- Fan-out and layout guidelines relative to trace widths and spacing
- Component placement and routing guidelines
- RF and Data Port transmission line layout
- Isolation techniques used on the ADRV9001 customer evaluation board

PCB MATERIAL AND STACK UP SELECTION

Figure 240 shows the PCB stackup used for the ADRV9001 customer evaluation boards. These boards employ 12 layers to achieve proper routing and isolation to best demonstrate all device functionality. The dielectric material used is I-SPEED with a thickness of 7 mil on outer layers. The board design uses the I-SPEED laminate for its low loss tangent at high frequencies. The ground planes under the I-SPEED laminate (layers 2 and 11) are the reference planes for the transmission lines routed on the outer surfaces. These layers are solid copper planes under the RF traces with no discontinuities. Layers 2 and 11 are crucial to maintaining the RF signal integrity.

RF traces on the outer layers must be a controlled impedance to get the best performance. These outer layers use 0.5 ounce copper. 0.5 and 1-ounce copper thickness are used for all the inner layers in this board. All ground planes on this board are full copper floods with no splits except for vias, through-hole components and isolation structures (more on this in later sections).

Layers 3, 5, 7, 9 are mainly used to route power supply domains. The Date Port interface lines are routed on layers 1, 7 and 12. Those layers have impedance control set to 100Ω differential for the differential LVDS pairs. The remaining digital signals are routed on inner layers 3, 5, 9 and 10. Table 100 describes details of the trace impedance controls used on different layers.

There are no buried in or blind vias used in this PCB design. All vias used in the PCB design are thru hole type. For vias carrying high frequency or RF sensitive signals, back drilling technique is applied.

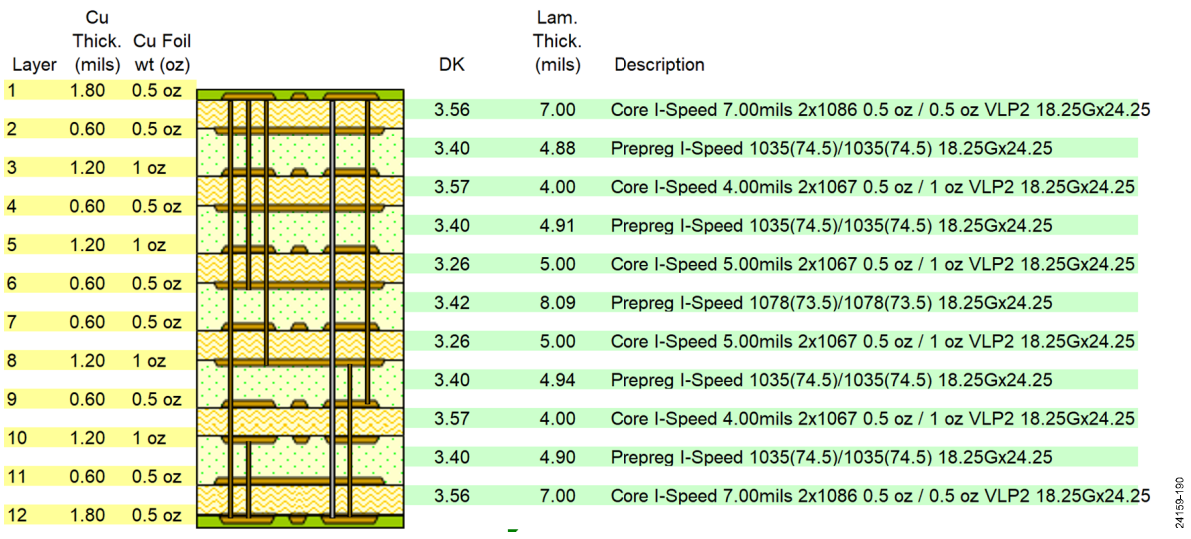


Figure 240. ADRV9001 Customer Evaluation Card Stackup

Table 105. Impedance Table

Layer	Structure Type	Coated Microstrip[1]	Target Impedance (Ω)	Impedance Tolerance (Ω)	Target Linewidth (mils)	Edge Coupled Pitch (mils)	Reference Layers	Modelled Linewidth (mils)	Modelled Impedance (Ω)	Coplaner Space (mils)
1	Single ended	N/A	50.00	±5	12.00	0.00	(2)	13.00	50.87	9.50
1	Single ended	Yes	50.00	±5	13.50	0.00	(2)	12.00	50.42	10.75
1	Edge Coupled Differential	Yes	100.00	±10	8.25	15.25	(2)	8.00	100.43	9.15
1	Edge Coupled Differential	N/A	100.00	±10	7.50	14.50	(2)	9.00	100.55	9.25
3	Single ended	N/A	50.00	±5	4.00	0.00	(2, 4)	4.25	49.53	17.88
3	Edge Coupled Differential	N/A	100.00	±10	3.75	10.75	(2, 4)	3.75	100.86	12.02
7	Edge Coupled Differential	N/A	100.00	±10	6.00	14.25	(6, 8)	6.00	99.75	12.02
9	Edge Coupled Differential	N/A	100.00	±10	6.25	15.00	(8, 11)	6.00	100.68	12.14
10	Edge Coupled Differential	N/A	100.00	±10	4.25	9.50	(11, 8)	4.50	100.23	11.89
12	Edge Coupled Differential	Yes	100.00	±10	8.00	15.25	(11)	8.00	100.80	10.00
12	Single ended	Yes	50.00	±5	12.00	0.00	(11)	12.00	50.31	10.00
12	Edge Coupled Differential	N/A	100.00	±10	7.50	14.50	(11)	9.00	100.55	9.25
12	Edge Coupled Differential	N/A	100.00	±10	8.25	15.50	(11)	8.25	99.64	10.02

[1] N/A means not applicable.

FAN-OUT AND TRACE SPACE GUIDELINES

The ADRV9001 device family uses a 196-pin BGA 12 × 12 mm package. The pitch between the pins is 0.8 mm. This small pitch makes it impractical to route all signals on a single layer. RF pins have been placed on the outer edges of the ADRV9001 package. This helps in routing the critical signals without a fan-out via. Each digital signal is routed from the BGA pad using a 4.5 mil trace. The trace is connected to the BGA using via-in-the-pad structure. The signals are buried in the inner layers of the board for routing to other parts of the system.

Extra care must be taken to ensure that DEV_CLK signal is shielded from any potential source of noise. Recommended approach is to use differential signaling for DEV_CLK clock. The data port interface signals when used in LVDS-SSI mode must be routed as 100 Ω differential pairs. Figure 241 shows the fan out scheme of the ADRV9001 evaluation card. There are no traces being routed between BGA pads on the top layer. As mentioned before ADRV9001 evaluation card uses via-in-the-pad technique. This routing approach can be used for ADRV9001 if there are no issues with manufacturing capabilities.

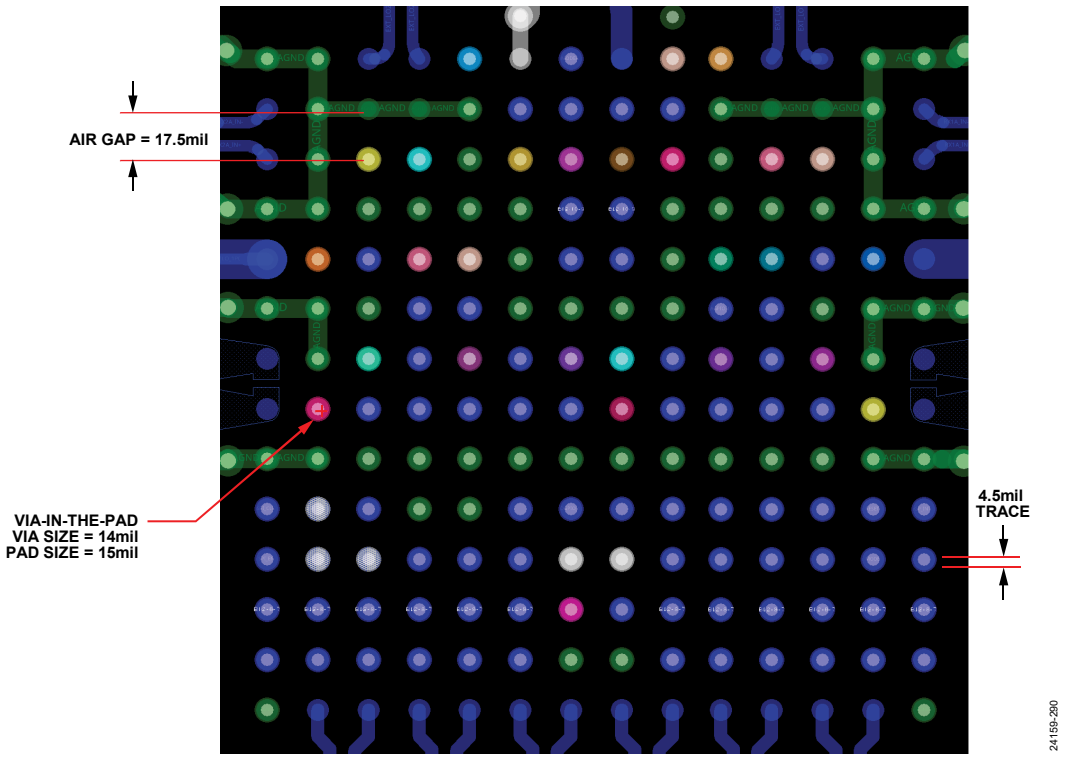


Figure 241. Trace Fan-Out Scheme on ADRV9001 Evaluation Card (PCB layer TOP and Layer 8 Enabled)

COMPONENT PLACEMENT AND ROUTING PRIORITIES

The ADRV9001 transceiver requires few external components to function, but those that are needed require careful placement and routing to optimize performance. This section provides a priority order and checklist for properly placing and routing critical signals and components as well as those whose location and isolation are not as critical.

Board layout design involves compromise. The recommendations within this User Guide are intended for wide RF bandwidth applications. For narrow RF bandwidth applications, the board line impedance parameters within this document may not be optimal.

The following list provides general suggestions for board design:

- Match the customer board design as close as possible to the ADRV9001 board design.
- Be attentive to power distribution and power ground return methodology.
- Do not run high speed digital lines in close proximity to dc power distribution routes or RF line routes.

Signals with Highest Routing Priority

RF lines and DEV_CLK clock are the signals that are most critical and should be routed with highest priority. Figure 242 shows the general directions in which each of the signals should be routed so that they can be properly isolated from noisy signals.

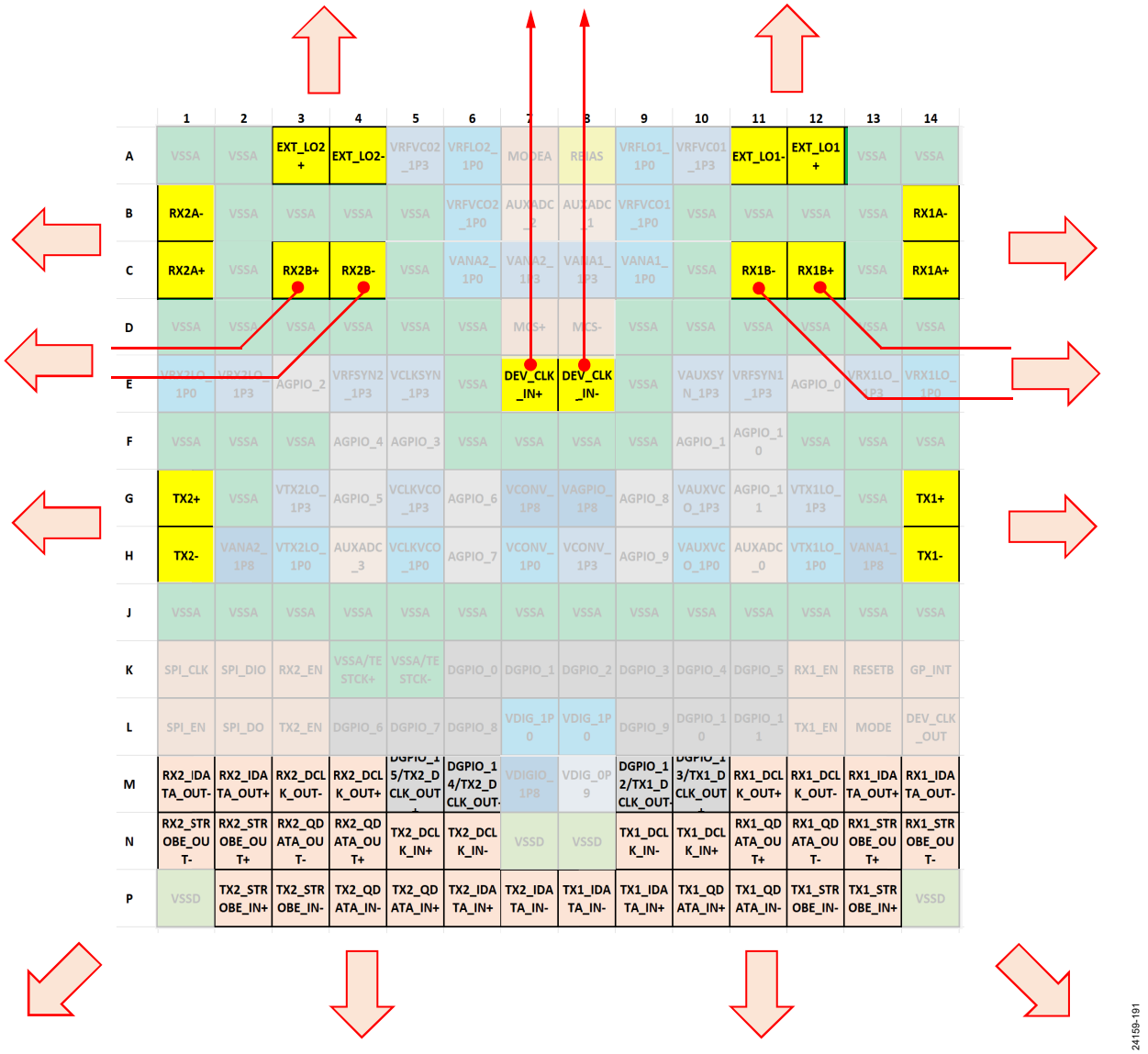


Figure 242. RF I/O, DEV_CLK, and Data Port Signal Routing Guidelines

- RF baluns are typically used to interface single-ended signals to the differential receiver and transmitter ports. These baluns and their associated matching circuits affect overall RF performance. Every effort should be made to optimize the component selection and placement to avoid performance degradation. The RF Port Interface Information section describes proper matching circuit placement and routing in more detail. Please refer to that paragraph for more information.
- Use microstrip or coplanar waveguides (CPWG) for transmission lines. These structures do not require via structures that cause additional impedance discontinuities that vary across frequency. For Rx1B and Rx2B, receiver ports, which do not have balls on the perimeter of the BGA, a via structure such as stripline may be necessary.
- Design the RF line systems between the device ball pad reference plane and the balun/filter reference plane for a differential impedance (ZDIFF) of 100 Ω for the receivers and 50 Ω for the transmitters. This is a compromise impedance with respect to frequency and a good starting point for design. The ZDIFF can be optimized to fit a narrower frequency range. It is desirable to design the lines for reasonable coupling (−10 dB to −15 dB) to promote adequate EMI suppression performance.
- In most cases, the required board artwork stack-up is going to be different than the ADRV9001 evaluation board stack-up. Optimization of RF transmission lines specific to the desired board environment is essential to the design and layout process.
- The ADRV9001 evaluation board uses microstrip lines for Rx and Tx RF traces. Some data port signal are routed using a combination of microstrip lines on the bottom of the PCB and stripline traces on internal layers due to board complexity. In general, RF traces should not use vias unless a direct line route is not possible.

- Differential lines from the balun to the Rx and Tx pins must be as short as possible. The length of the single ended transmission line should also be short to minimize the effects of parasitic coupling.
- The system designer can optimize the RF performance with the proper selection of balun, matching components, and ac coupling capacitors. The external LO traces and the DEV_CLK_IN traces may require matching components as well to ensure optimal performance. Matching network design is explained in greater detail in the RF Port Interface Information section of this document.
- RF signal path isolation is critical to achieving the level of isolation specified in the ADRV9001 datasheet. More details on proper isolation are provided in the Isolation Techniques Used on the ADRV9001 Evaluation Card section.
- For each RF Tx output, install a 10µF capacitor near the balun power supply pin connected to the VANA1_1P8, VANA2_1P8 supplies. If baluns with no dc supply connection are used, power will must be supplied to the Tx outputs using RF chokes. Connect chokes between the VANA1_1P8 and Tx1 output and VANA2_1P8 and Tx2 output respectively. In both cases, the 10µF capacitor acts as a reservoir for Tx supply current. The TX Balun DC Supply Options section describes the Tx output power supply configuration in more detail.
- Connect the external clock inputs to the DEV_CLK_IN+ (E7) and DEV_CLK_IN– (E8) pins using ac coupling capacitors. Use a 100 Ω termination at the input to the device. Figure 243 illustrates the recommended placement for termination resistor near the DEV_CLK_IN pins. Traces should be shielded by surrounding ground with vias staggered along the edge of the differential trace pair. This arrangement creates a shielded channel that prevents the reference clock from any interference from other signals. Refer to the ADRV9001 evaluation card layout for exact details.

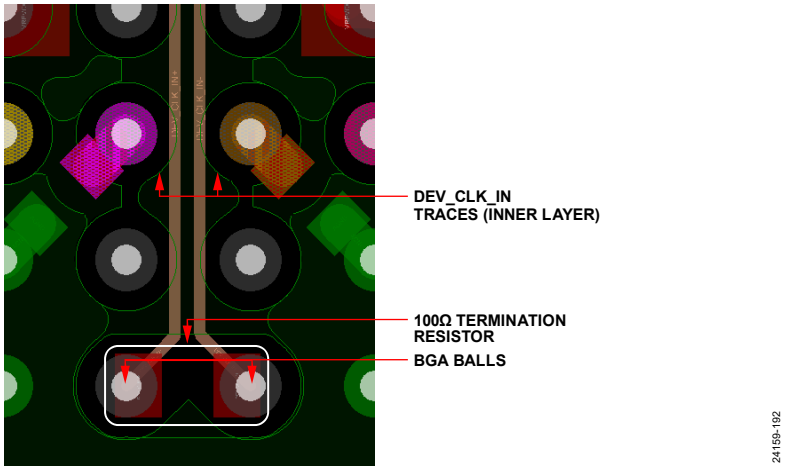


Figure 243. DEV_CLK_IN Signal Routing Recommendations

- The EXT_LO1+ (A12), EXT_LO1- (A11), EXT_LO2+ (A3), EXT_LO2- (A4) pins are internally dc biased. If an external LO is used, connect it via ac coupling capacitors.
- The data port interface should be routed at the beginning of the PCB design and with the same priority as RF signals. This is especially important if data port runs in LVDS configuration. Attention should be paid to provide appropriate isolation between data port differential pairs.

Signals with Secondary Routing Priority

Power supply quality has direct impact on overall system performance. To achieve optimal performance, users should follow recommendations regarding power supply routing. The following recommendations outline how different power domains should be routed and which supplies can be tied to the same supply but separated by a ferrite bead.

A general recommendation for power supply routing is to follow the star methodology in which each power domain is deliver by a separate trace from the source supply. Care should be taken to make sure that each power trace is surrounded by ground. Figure 244 shows an example of such traces routed on the evaluation card on layer 3. Each trace is separated from any other signal by ground plane fill and vias. This approach is essential to providing necessary isolation between power domains.

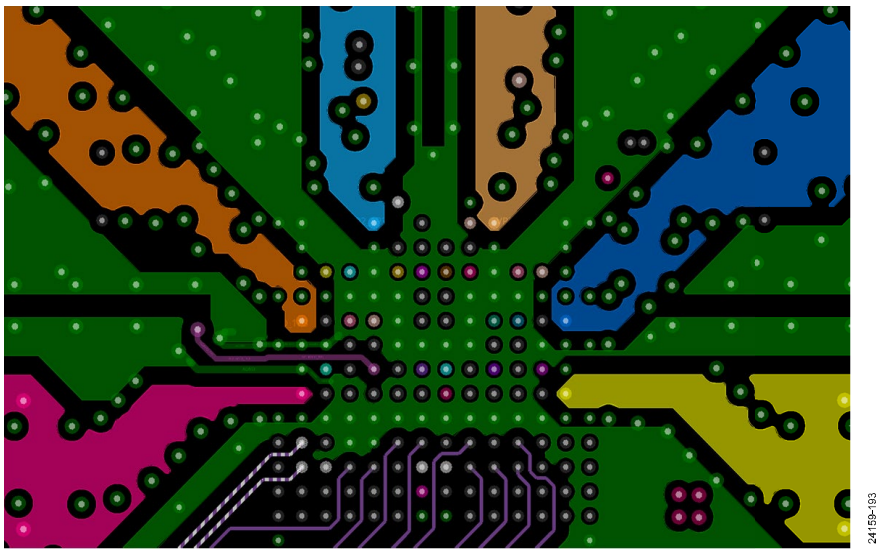


Figure 244. Layout Example of Power Supply Connections Routed with Ground Shielding (Layer 3)

Figure 245 shows an example of how the ferrite beads, reservoir capacitors and decoupling capacitors should be placed. Recommendation is to connect a ferrite bead between a power plane and ADRV9001 at a distance away from ADRV9001. The ferrite bead should supply a trace with a reservoir capacitor connected to it. That trace should then be shielded with ground and provide power to ADRV9001 Power pin. A 1 µF capacitor should be placed near the power supply pin with the ground side of the bypass capacitor placed so that ground currents flow away from other power pins and their bypass capacitors.

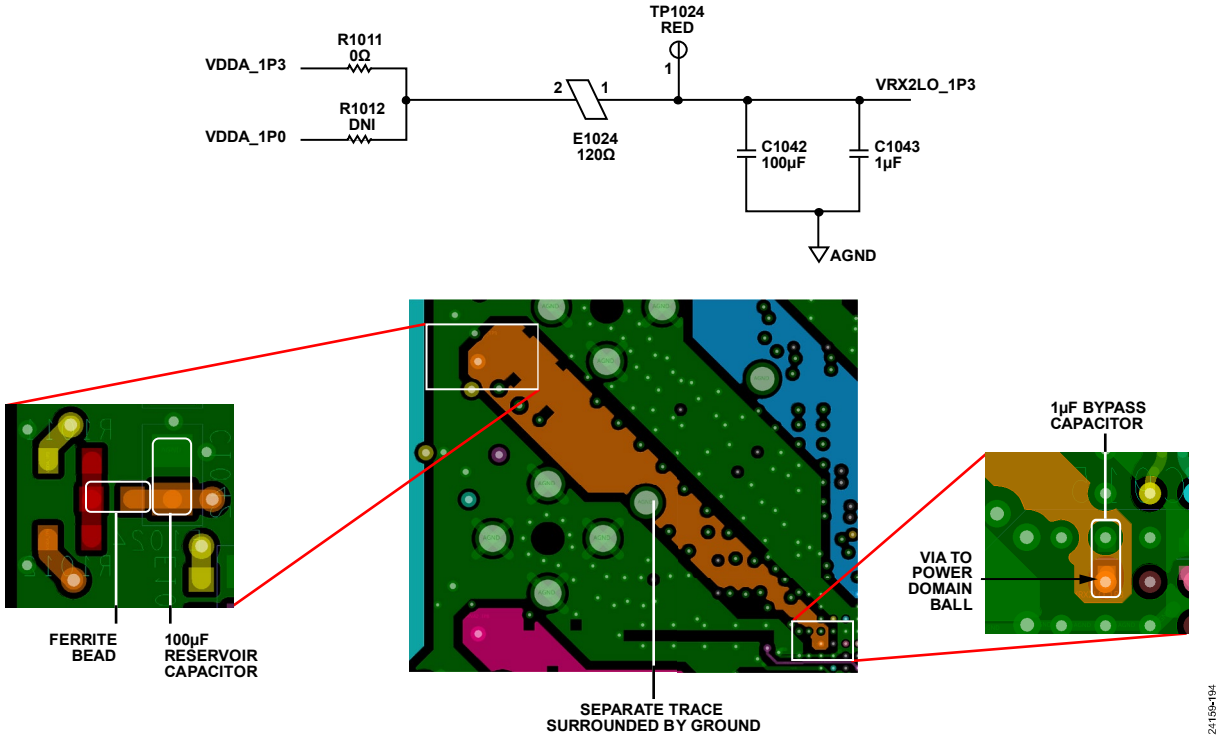


Figure 245. Placement Example of Ferrite Beads, Reservoir and Bypass Capacitors on ADRV9001 Customer Card (Layers: TOP, 3-Power and BOTTOM)

There are two possible power supply architectures for ADRV9001 transceivers, as follows:

- High performance, low risk, four power domains
  - 1.8 V digital
  - 1.8 V analog
  - 1.3 V analog
  - 1.0 V digital

This approach uses the ADRV9001 internal LDOs to generate 1.0 V for all internal blocks. Figure 246 outline power supply routing recommendations for this architecture.

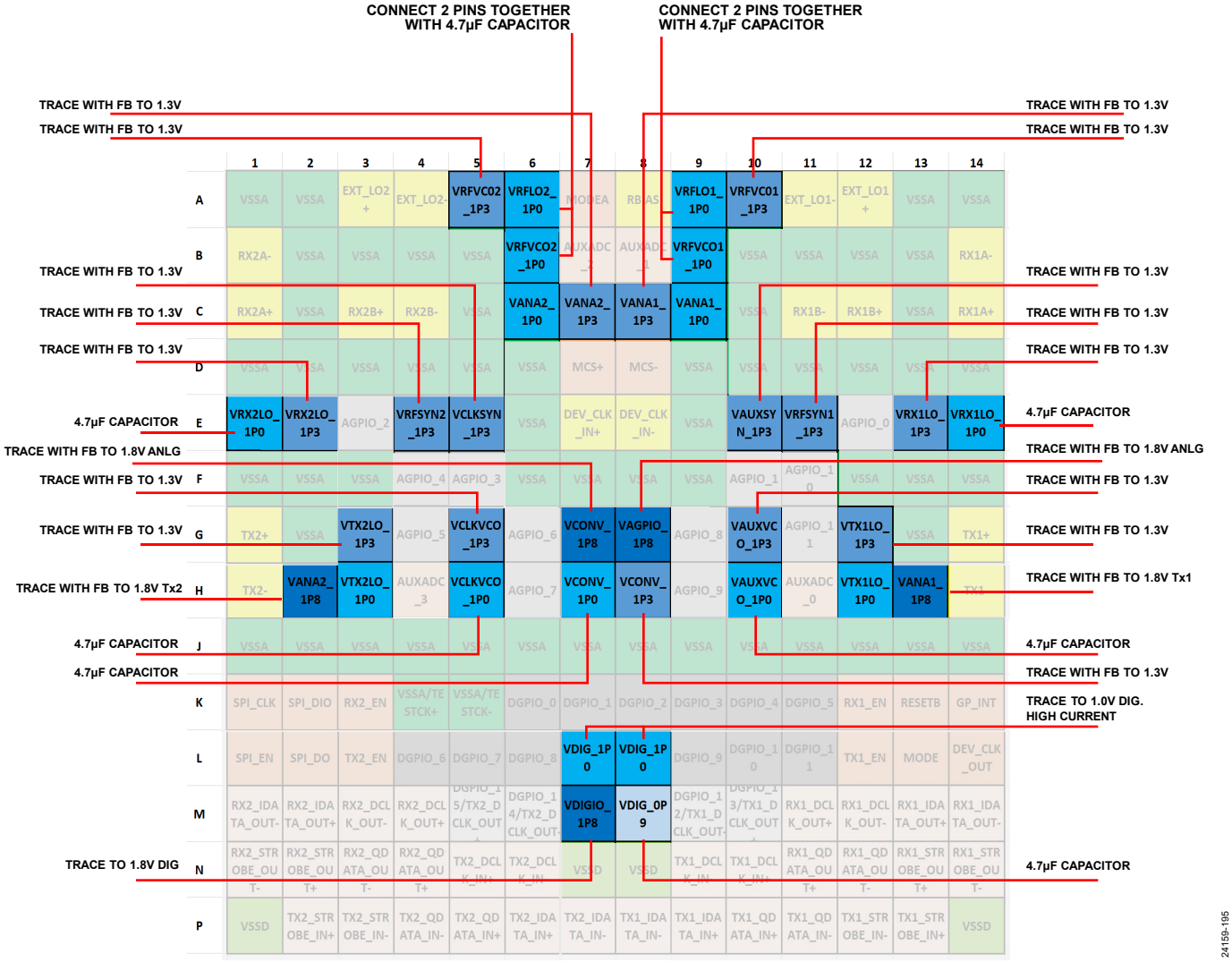


Figure 246. ADRV9001 Power Supply Domains with Connection Guidelines, All Internal LDOs in Use

- Power supply optimization, higher risk (use noise sensitive 1.0V analog), 5 power domains:
  - 1.8 V digital,
  - 1.8 V analog,
  - 1.3 V analog,
  - 1.0 V digital,
  - 1.0 V analog,

This approach that uses some of ADRV9001 internal LDOs to generate 1.0 V for internal blocks. For remining blocks it expect the 1.0 V to be delivered from external power source. Figure 247 outline power supply routing recommendations for this architecture.

- For domains shown in Figure 247 that should be powered through a ferrite bead (FB), care should be taken to place the ferrite beads near the ADRV9001 supply pins. The ferrite beads should also be spaced to ensure their electric fields do not influence each other. The ferrite bead should supply a trace with a reservoir capacitor connected to it. That trace should then be shielded with ground and provide power to input power pin.

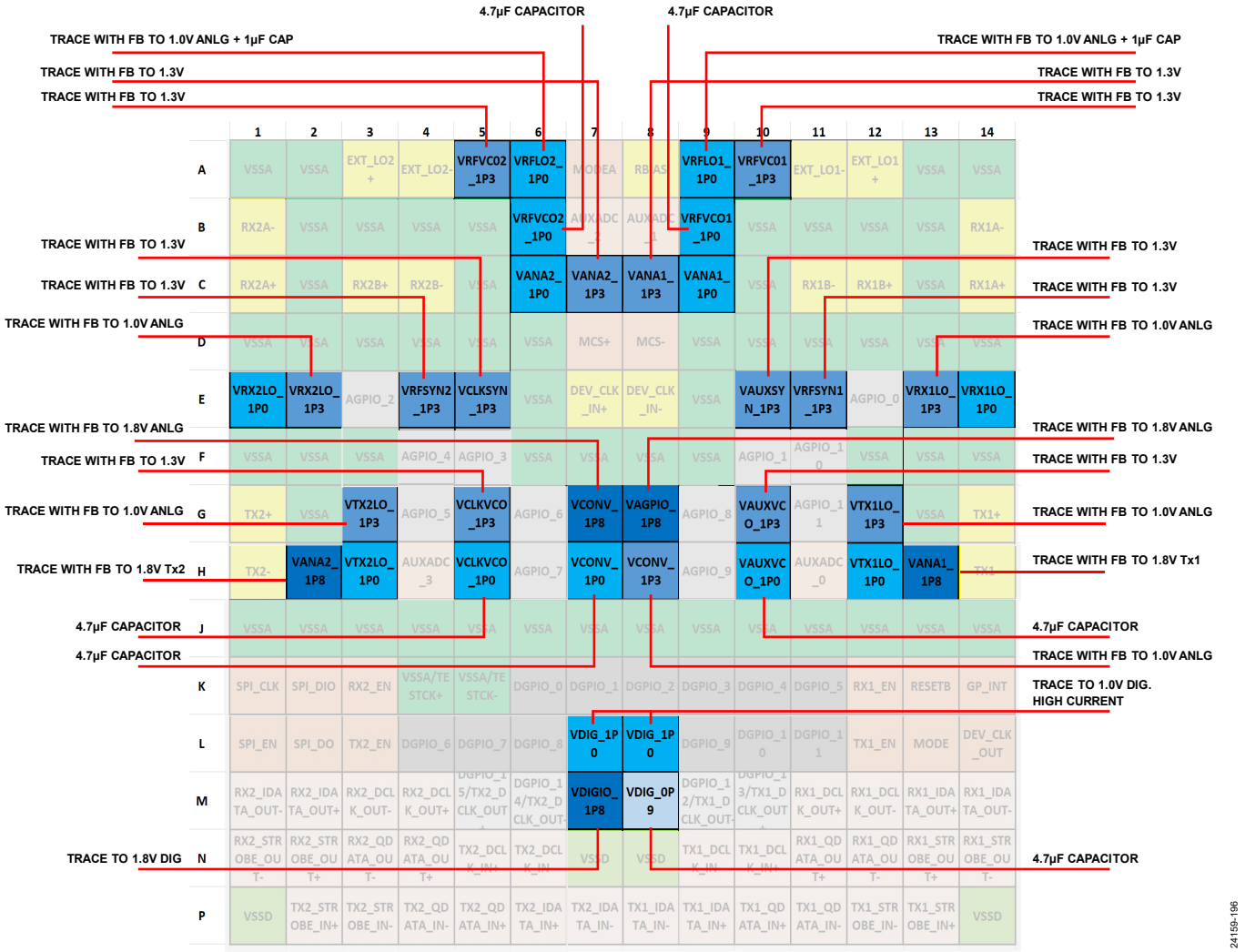


Figure 247. ADRV9001 Power Supply Domains with Connection Guidelines, Some Internal LDOs bypassed, 1.0 V Analog Domain Required

Ceramic 4.7 µF bypass capacitors must be placed at the VRFVCO2_1P0, VRFVCO1_1P0, VRX2LO_1P0, VRX1LO_1P0, VCLKVCO_1P0, VAUXVCO_1P0, VCONV_1P0 and VDIG_0P9 pins. Place these capacitors as close as possible to the device with the ground side of the bypass capacitor placed so that ground currents flow away from other power pins and their bypass capacitors if at all possible.

In scenario, when power supply follows recommendation outlined in Figure 247 (some internal LDOs bypassed, external 1.0V analog domain in use), 4.7 µF capacitors at VRX2LO_1P0, VRX1LO_1P0 pins are not necessary. 1.0 V domains connected to VRFLO1_1P0 and VRFLO2_1P0 require 1 µF capacitors.

Signals with Lowest Routing Priority

The following guidelines govern those signals that are the lowest signal routing priority. These can be routed after all critical signal routes have been completed so they don't interfere with the critical component placement and routing. The signals shown in Figure 225 can be routed with the lowest priority.

- Connect a 4.99 kΩ resistor to RBIAS pin (C14). This resistor must have a 1% tolerance or better.
- The device has support for JTAG boundary scan, and the MODE pin is used to access the function. Connect the MODE pin (L13) to ground for normal operation. Refer to the datasheet for JTAG boundary scan information.
- Connect the RESETB pin (K13) to VIOCTRL_1P8 with a 10 kΩ resistor for normal operation. The device can be reset by driving this pin low.
- When routing digital signals from rows K and below, it is important to route them away from the analog section (rows A through H). Digital signal routing should not pass above the red dotted line highlighted in Figure 248.
- The AGPIO_N signals can be routed using inner PCB layers. Those signals are intended to control analog blocks such as power amplifiers or low noise amplifiers. The AGPIO_0 thru AGPIO_3 can also be used as general purpose analog outputs when muxed to

the internal AUXDAC outputs. To prevent noise coupling into those signals, the user should route them away from digital region (above the red dotted line highlighted in Figure 248.

- The AuxADC_N signals can be routed using inner PCB layers. Those signals are intended to sense analog voltage levels such as temperature sensors. To prevent noise coupling into those signals the user should route them away from digital region (above red dotted line highlighted in Figure 248).
- MODEA signal is intended to setup operation of DEV_CLK_IN± pins (LVDS differential, CMOS single-ended, XTAL with different bias voltage). User should follow recommendation outlined in RF Port Interface Information section when controlling this pin.
- MCS± signals should be treated as differential. If multi-chip synchronization feature is intended to be used in end application, those signals should be routed with traces matching length of DEV_CLK_IN± traces.

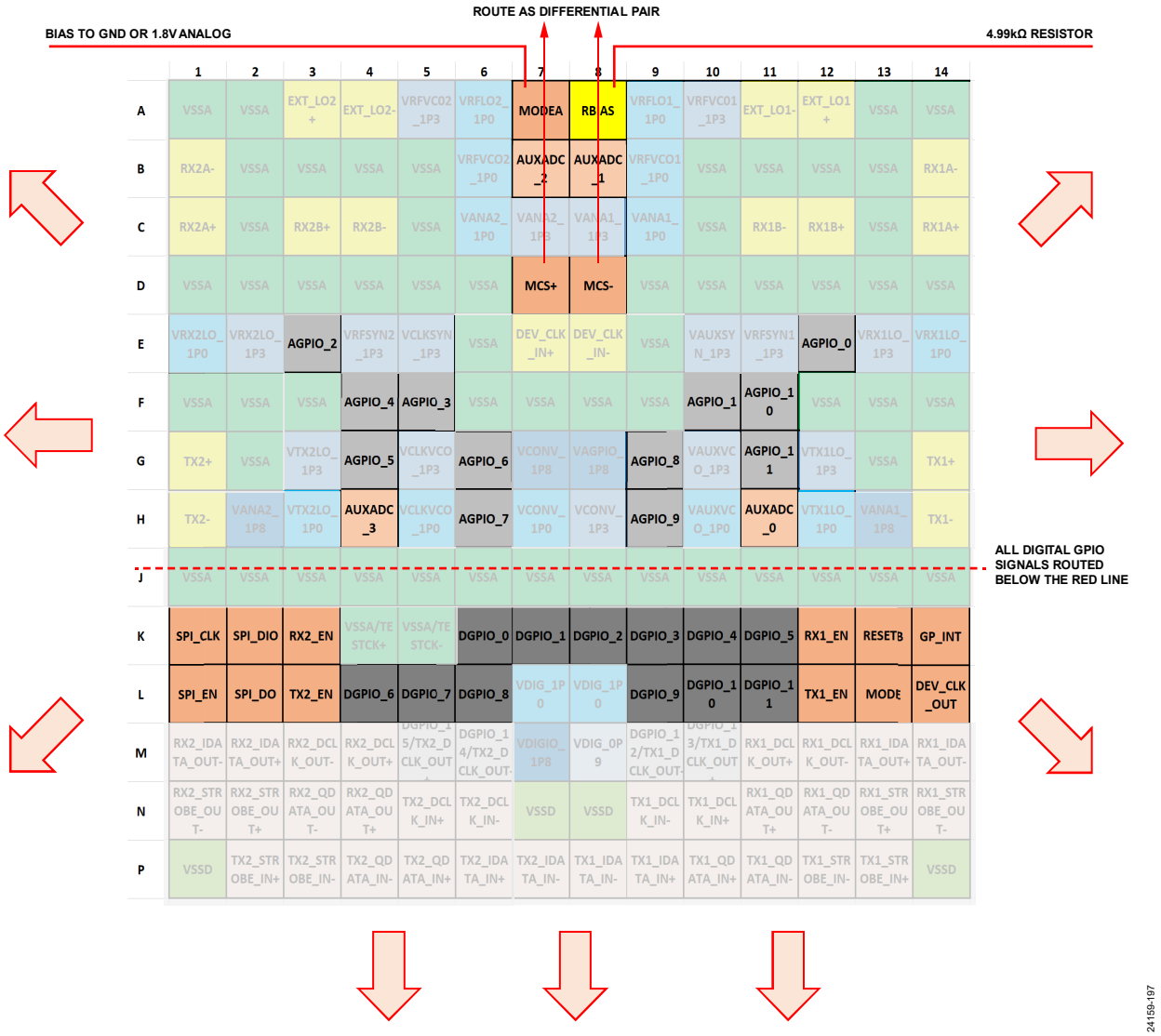


Figure 248. ADRV9001 AuxADC, SPI, Analog GPIO/AuxDAC, MCS±, and Digital GPIO Signal Routing Guidelines

RF AND DATA PORT TRANSMISSION LINE LAYOUT

RF Line Design Summary

The RF line design is a compromise between many variables. Line impedance, line to line coupling, and physical size represent the parameters subject to compromise. Smallest physical size is in direct opposition to the ZCM of the line, which is directly opposed to the line EMI performance. In addition, the interface between the RF line width and the device ball pad diameter on the PCB represents a potential discontinuity. As the RF line width approaches the ball pad diameter, the risk associated with potential interface discontinuity reduces.

Balanced lines for differential mode signalling used between the device and the RF balun should be as short as possible. The length of the single ended transmissions lines for RF signals should also be as short as possible. Keeping signal paths as short as possible reduce susceptibility to undesired signal coupling and reduce the effects of parasitic capacitance, inductance, and loss on the transfer function of the transmission line and impedance matching network system. The routing of these signal paths is the most critical factor in optimizing performance and, therefore, should be routed prior to any other signals and maintain the highest priority in the PCB layout process.

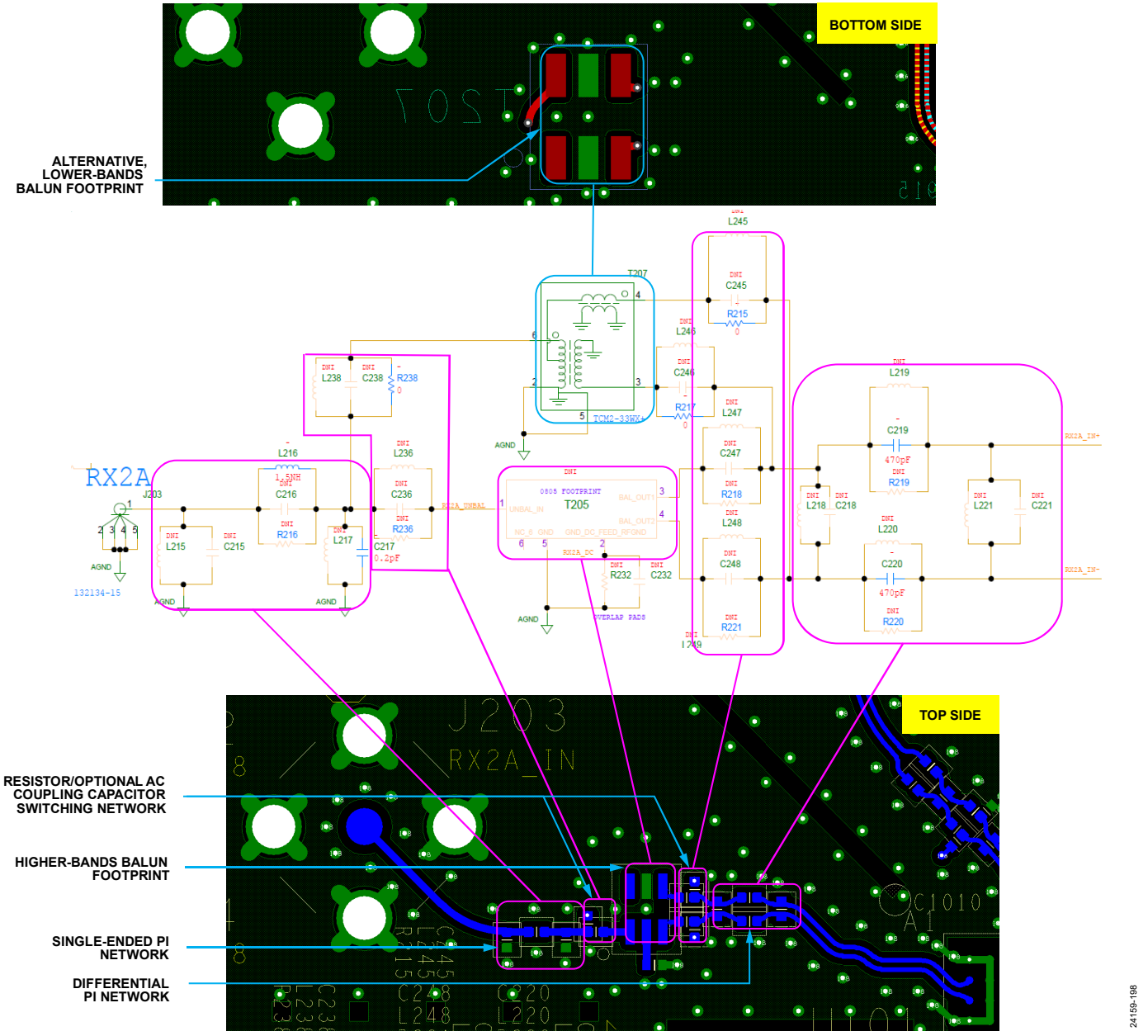


Figure 249. Receiver Matching Network on ADRV9001 Evaluation Board

The circuit in Figure 249 shows the layout topology for the chosen receiver matching network. Note the location and orientation of each component – placement is critical to achieve expected performance. Similarly, the circuit in Figure 250 shows the layout topology used for the transmitter matching network. (see the RF Port Interface Information section for circuit details). More details concerning the dc supply to the transmitter section are provided in the next section.

All the RF signals must have a solid ground reference under each path to maintain the desired impedance. None of the critical traces should run over a discontinuity in the ground reference.

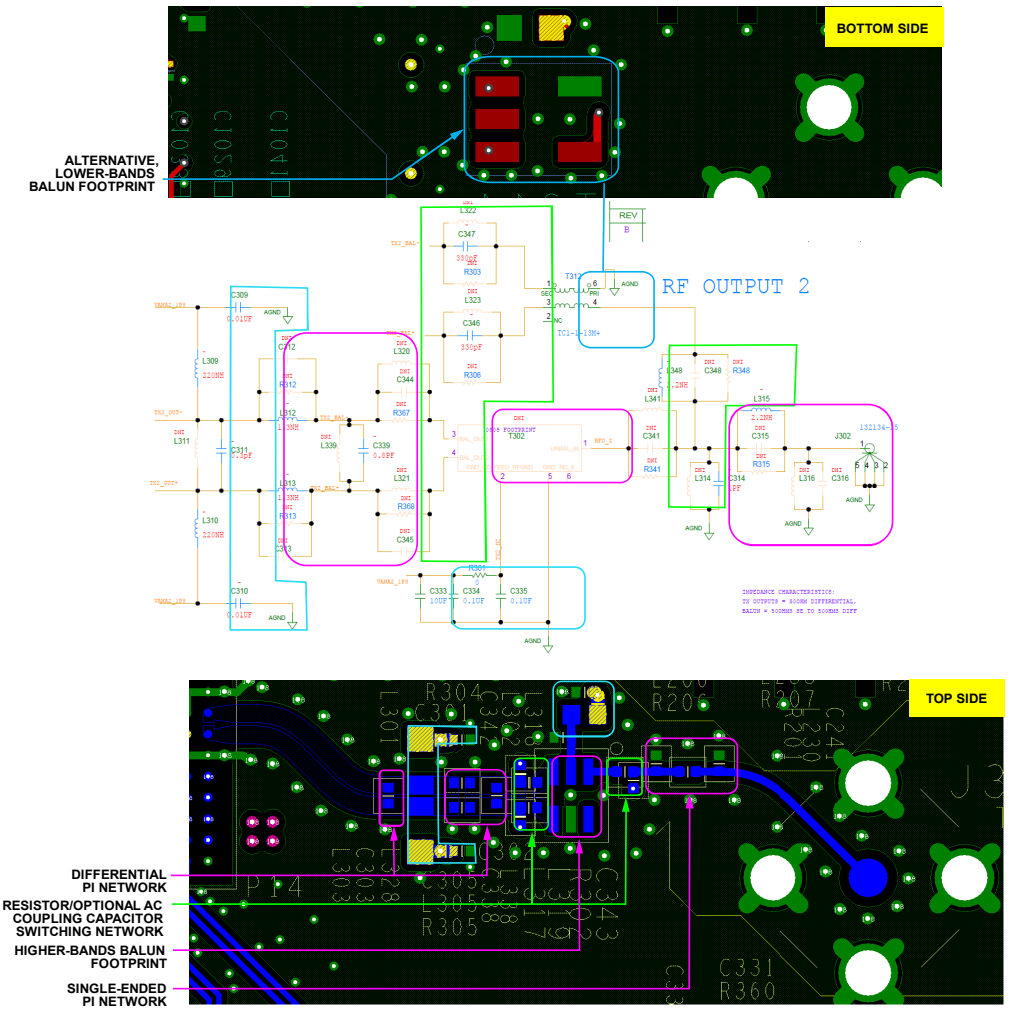


Figure 250. Transmitter Matching Network on ADRV9001 Evaluation Board

Transmitter Bias and Port Interface

This section considers the dc biasing of the ADRV9001 transmitter (Tx) outputs and how to interface to each Tx port. At full output power, each differential output side draws approximately 100mA of DC bias current. The Tx outputs are dc biased to a 1.8 V supply voltage using either RF chokes (wire-wound inductors) or a transformer (balun) center tap connection.

Careful design of the DC bias network is required to ensure optimal RF performance levels. When designing the dc bias network, select components with low dc resistance (RDCR) to minimize the voltage drop across the series parasitic resistance element with either of the dc bias schemes suggested in Figure 241 and Figure 252. The red resistors (R_DCR) indicate the parasitic elements. As the impedance of the parasitic increase, the voltage drop (ΔV) across the parasitic element increases which causes the transmitter RF performance (i.e PO,1dB, PO,MAX, etc…) to degrade. The choke inductance (L_c) should be selected high enough relative to the load impedance such that it does not degrade the output power. If chokes are used they should be very well matched (including PCB traces). Uneven matching of chokes design can cause unwanted emission of spikes at the Tx output. This emission can affect components connected to the Tx output.

The recommended dc bias network is the one using the center tap balun is shown in Figure 252. This network has fewer parasitic and fewer total components.

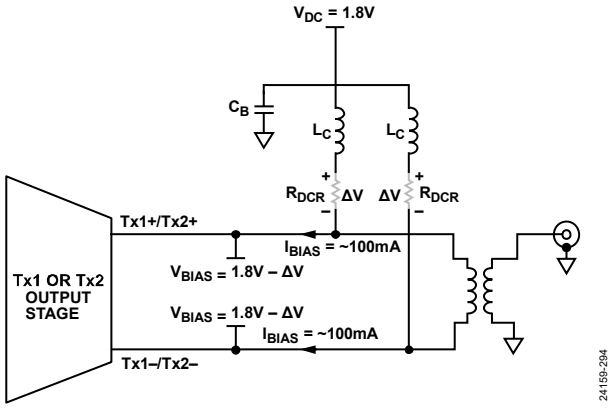


Figure 251. ADRV9001 DC Bias Configuration for the Transmitter Output Using Wire-Wound Chokes

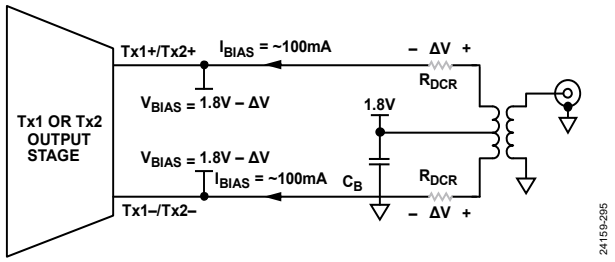


Figure 252. ADRV9001 DC Bias Configuration for the Transmitter Output Using a Center-Tapped Transformer

The ADRV9001 evaluation board provides flexibility to configure each Tx output to work with either a center tapped transformer (balun) or a set of two closely matched wire wounded chokes. The center tapped transformer passes the bias voltage directly to the transmitter outputs through each differential input. This configuration offers the lowest component count.

In some cases, the desired balun does not provide a dc connection to the transmitter output lines. To support this situation, the ADRV9001 evaluation board provides the placeholders for RF chokes tied to the VANA1_1P8 (for Tx1 output) and VANA2_1P8 (for Tx2 output) supply. It also provides the placeholders for ac coupling capacitors to prevent creating a dc short through the balun to ground.

Impedance matching networks on the balun single-ended port are usually required to achieve optimum performance. In addition, ac coupling is often required on the single-ended side if the balun contains a dc path from one of the transmitter's differential outputs to the single-ended port.

Careful planning is required for the Tx balun selection. If a Tx balun is selected that requires a set of external DC bias chokes, it is necessary to find the optimum compromise between the choke physical size, choke dc resistance (RDCR) and the balun passband insertion loss. Users should refer to the RF Port Interface Information section of this document for more information on Tx output balun and RF choke selection as well as matching circuit recommendations.

TX Balun DC Supply Options

Each transmitter requires approximately 200 mA supplied through an external connection. The PCB layout of the ADRV9001 board allows use of external chokes to provide 1.8 V power domain to the ADRV9001 outputs to allow users to try different baluns that may not have a dc center tap pin to supply the bias voltage to the transmitter outputs.

To reduce switching transients when attenuation settings change, the balun dc feed should be powered directly by the 1.8 V plane. The geometry of the 1.8 V plane should be designed so that each balun or each pair of chokes is associated with its Tx output. The VANA1_1P8 should be used to power Tx1 output and VANA2_1P8 should be used to power Tx2 output.

If careful layout and isolation of the dc supply is not followed, it can adversely affect Tx-Tx isolation. Figure 253 shows the power supply layout configuration used on the ADRV9001 board to achieve the desired Tx-Tx isolation performance. This image illustrates star connection from common 1.8 V analog power plane.

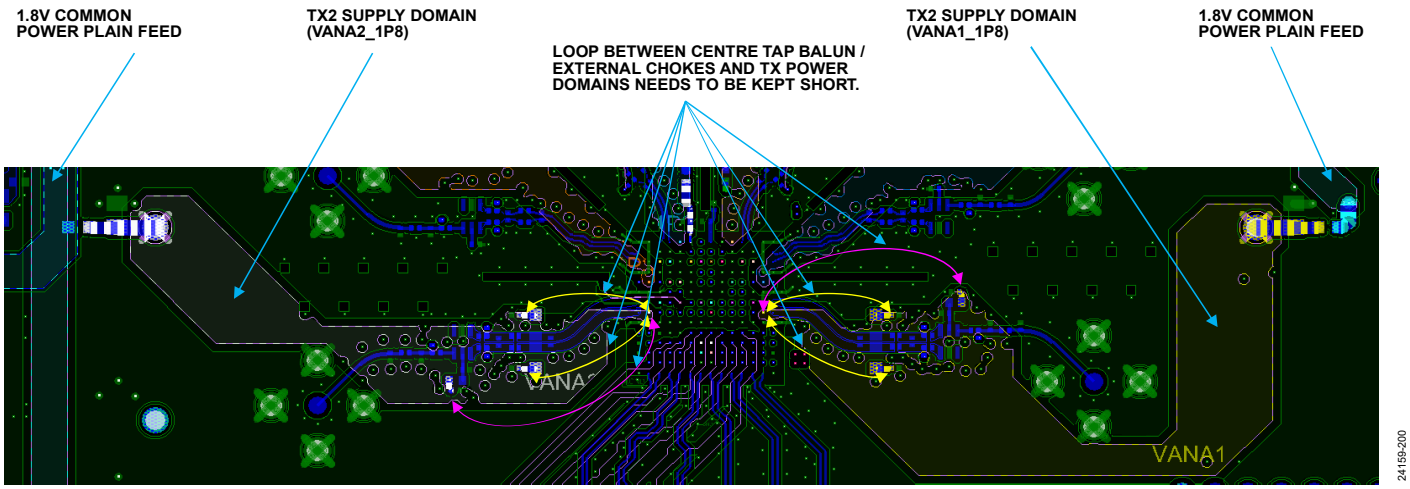


Figure 253. 1.8 V Transmitter Power Supply Routing on the ADRV9001 Evaluation Board

An example of the balun feed supply designed to achieve the isolation managed in the evaluation board is shown in Figure 254 and Figure 246.

DC Balun

When a Tx balun that is able to conduct dc is used then the system shown in Figure 254 should be used. The decoupling cap near the Tx balun should be placed as close as possible to the balun's DC feed pin. Its orientation should be perpendicular to the ADRV9001 device so the return current avoids a ground loop with the ground pins surrounding the Rx input. The customer card provides an option to install an RF isolation inductor which can provide extra isolation between the Tx1 and Tx2 balun supply feeds. A 10µF capacitor and a 0.1 µF capacitor are helpful on the dc feed pin to eliminate Tx spectrum spurs and dampen the transients. Note that when this supply approach is used the series matching components must be dc shorts. It is recommended to use 0 Ω if an inductor is not needed to match the balun impedance to the Tx output impedance.

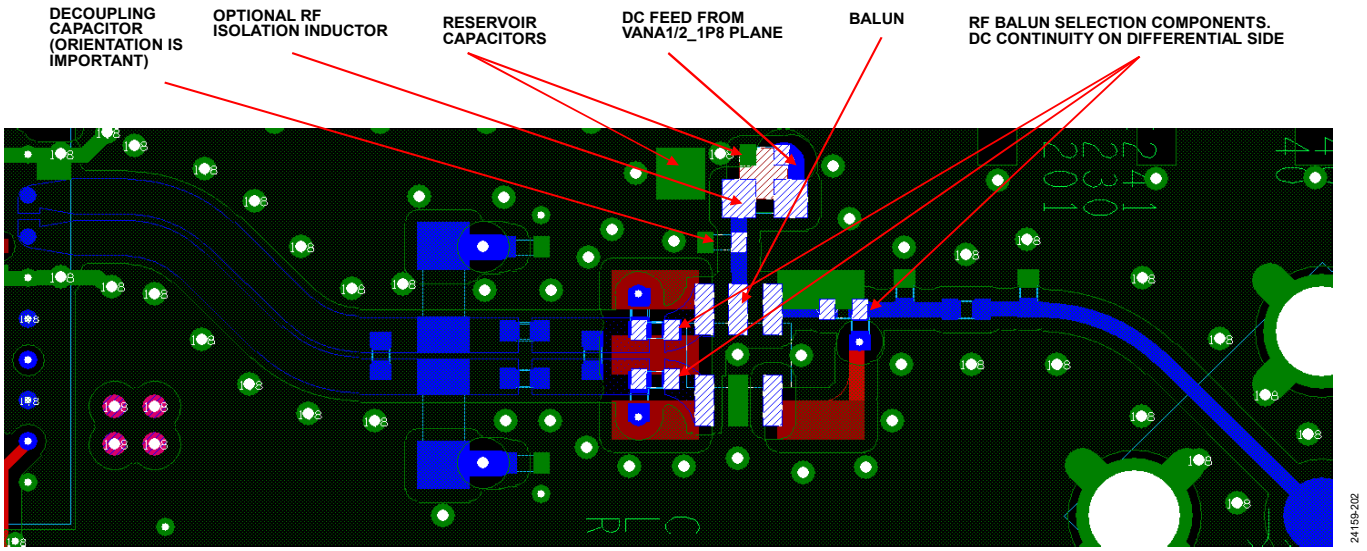


Figure 254. Transmitter Power Supply for a Balun with a Center Tap

Chokes

The ADRV9001 evaluation board provides flexibility to be configured to use a Tx balun that is not capable of conducting dc current. In such a scenario, the user should install dc chokes as well as their decoupling capacitors as highlighted in Figure 255. Care should be taken to match both chokes to avoid potential current spikes. Difference in parameters between both chokes can cause unwanted emission at Tx outputs. Note that if the differential input to the balun can create a dc short to ground (through the balun), the series matching components must be capacitors. If a short can form on the single-ended side, the single-end series blocking element must be a capacitor.

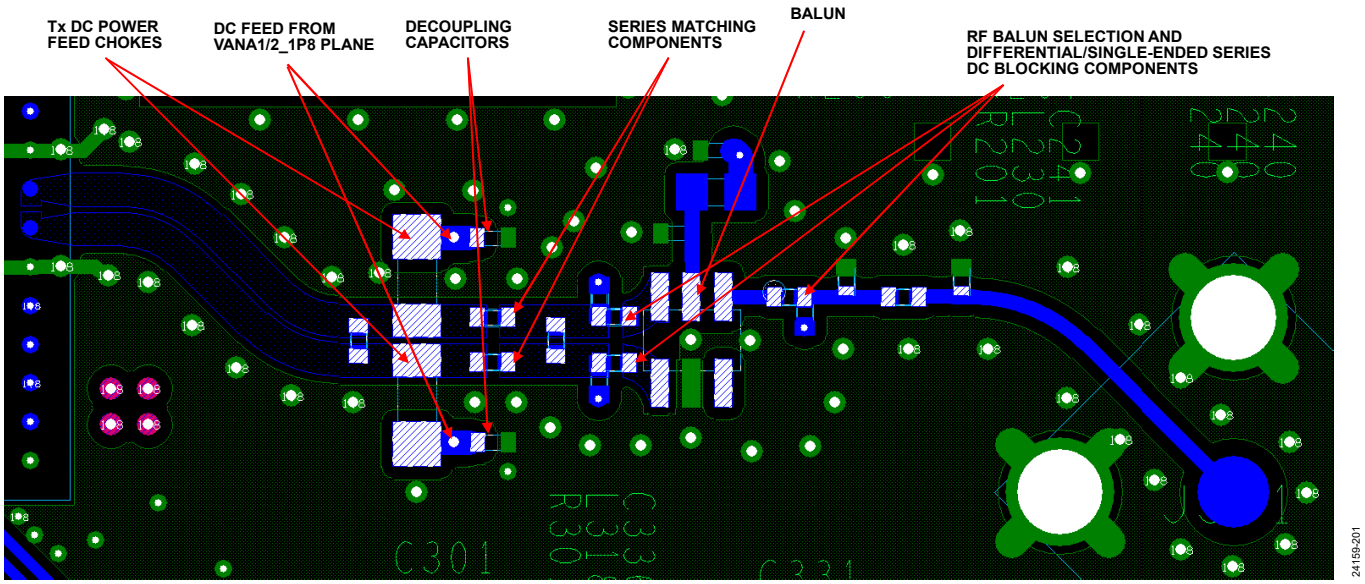


Figure 255. Transmitter Power Supply Using RF Chokes

SSI Data Port Trace Routing Recommendations

The Data Port interface transfer I/Q data between BBIC/FPGA and ADRV9001 Tx and Rx datapaths. There are two possible mode of operation for SSI data port:

- CMOS-SSI mode – single ended - with clock rate for data transfer up to 80 MHz
- LVDS-SSI mode – differential - with clock rate for data transfer up to 500 MHz DDR (1000 MHz data rate)

Correct layout practice should be followed while routing SSI interface signals.

If CMOS-SSI mode is selected, single-ended signal lines between ADRV9001 and BBIC/FPGA should be as short as possible. Trace capacitance should also be minimized to minimize the current needed by ADRV9001 to drive the line. Refer to the ADRV9001 datasheet document for details regarding pin drive capabilities.

If LVDS-SSI mode is selected, the user should route all LVDS signals as 100 Ω differential pairs.

- When routing the PCB layout for LVDS-SSI data lines, the designer must decide to route the signals using stripline or microstrip traces. There are positives and negatives for each that should be carefully considered.
- Stripline has less loss and emits less EMI than microstrip lines, but stripline traces require the use of vias that can add complexity to the task of controlling the impedance by adding line inductance.

Microstrip is easier to implement if the component placement and density allow for routing on the top layer, simplifying the task of controlling the impedance.

If using the top layer of the PCB is problematic or the advantages of stripline are desirable, then follow these recommendations:

- Minimize the number of vias.
- Use blind vias wherever possible to eliminate via stub effects, and use micro-vias to minimize via inductance.
- If using standard vias, use maximum via length to minimize the stub size. For example, on an 8-layer board, use layer 7 for the stripline pair.
- For each via pair, a pair of ground vias should be placed in close proximity to them to minimize the impedance discontinuity.

In LVDS-SSI mode

- for Tx data port inputs, termination of 100Ω is implemented inside ADRV9001
- for Rx data port outputs, it is expected that 100 Ω termination is implemented at the receiver end.

Evaluation Board FMC Connector Signals Mapping

The ADRV9001 evaluation board use FMC standard connector as an interface to carrier boards. Table 106 outlines signal mapping used on FMC connector implemented on ADRV9001 evaluation board. Second column refers to FMC standard pinout names. For more information refer to ADRV9001 EVB schematic.

Table 106. FMC Pinout Mapping Used by ADRV9001 Evaluation Board

Schematic Net Name	FMC Connector Mappings
FPGA_REF_CLK+	G02-FMC_CLK1_M2C_P
FPGA_REF_CLK-	G03-FMC_CLK1_M2C_N
DEV_CLK_OUT	H04-FMC_CLK0_M2C_P
SM_FAN_TACH	H05-FMC_CLK0_M2C_N
RX1_DCLK_OUT+	G06-FMC_LA00_CC_P
RX1_DCLK_OUT-	G07-FMC_LA00_CC_N
RX1_STROBE_OUT+	H07-FMC_LA02_P
RX1_STROBE_OUT-	H08-FMC_LA02_N
RX1_IDATA_OUT+	G09-FMC_LA03_P
RX1_IDATA_OUT-	G10-FMC_LA03_N
RX1_QDATA_OUT+	H10-FMC_LA04_P
RX1_QDATA_OUT-	H11-FMC_LA04_N
DGPIO_13_TX1_DCLK_OUT+	D08-FMC_LA01_CC_P
DGPIO_12_TX1_DCLK_OUT-	D09-FMC_LA01_CC_N
TX1_DCLK_IN+	H13-FMC_LA07_P
TX1_DCLK_IN-	H14-FMC_LA07_N
TX1_STROBE_IN+	C10-FMC_LA06_P
TX1_STROBE_IN-	C11-FMC_LA06_N
TX1_IDATA_IN+	G12-FMC_LA08_P
TX1_IDATA_IN-	G13-FMC_LA08_N
TX1_QDATA_IN+	D11-FMC_LA05_P
TX1_QDATA_IN-	D12-FMC_LA05_N
RX2_DCLK_OUT+	D20-FMC_LA17_CC_P
RX2_DCLK_OUT-	D21-FMC_LA17_CC_N
RX2_STROBE_OUT+	H25-FMC_LA21_P
RX2_STROBE_OUT-	H26-FMC_LA21_N
RX2_IDATA_OUT+	G21-FMC_LA20_P
RX2_IDATA_OUT-	G22-FMC_LA20_N
RX2_QDATA_OUT+	H22-FMC_LA19_P
RX2_QDATA_OUT-	H23-FMC_LA19_N
DGPIO_15_TX2_DCLK_OUT+	C22-FMC_LA18_CC_P
DGPIO_14_TX2_DCLK_OUT-	C23-FMC_LA18_CC_N
TX2_DCLK_IN+	G24-FMC_LA22_P
TX2_DCLK_IN-	G25-FMC_LA22_N
TX2_STROBE_IN+	H28-FMC_LA24_P
TX2_STROBE_IN-	H29-FMC_LA24_N
TX2_IDATA_IN+	D23-FMC_LA23_P
TX2_IDATA_IN-	D24-FMC_LA23_N
TX2_QDATA_IN+	G27-FMC_LA25_P
TX2_QDATA_IN-	G28-FMC_LA25_N
RX1_ENABLE	C14-FMC_LA10_P
RX2_ENABLE	D27-FMC_LA26_N
TX1_ENABLE	D14-FMC_LA09_P
TX2_ENABLE	G30-FMC_LA29_P
SPI_EN	H19-FMC_LA15_P
SPI_CLK	G15-FMC_LA12_P
SPI_DIO	G31-FMC_LA29_N

Schematic Net Name	FMC Connector Mappings
SPI_DO	G16-FMC_LA12_N
MODE	D17-FMC_LA13_P
RESET_TRX	D18-FMC_LA13_N
DEV_MCS_FPGA_IN+	C18-FMC_LA14_P
DEV_MCS_FPGA_IN-	C19-FMC_LA14_N
DGPIO_0	G18-FMC_LA16_P
DGPIO_1	G19-FMC_LA16_N
DGPIO_2	H20-FMC_LA15_N
DGPIO_3	H17-FMC_LA11_N
DGPIO_4	D15-FMC_LA09_N
DGPIO_5	C15-FMC_LA10_N
DGPIO_6	C26-FMC_LA27_P
DGPIO_7	D26-FMC_LA26_P
DGPIO_8	H31-FMC_LA28_P
DGPIO_9	H32-FMC_LA28_N
DGPIO_10	H16-FMC_LA11_P
DGPIO_11	C27-FMC_LA27_N
GP_INT	H34-FMC_LA30_P
VADJ_TEST_1 (VADJ_ERR)	G33-FMC_LA31_P
VADJ_TEST_2 (PLATFORM_STATUS)	G34-FMC_LA31_N
FPGA_MCS_IN+	H37-FMC_LA32_P
FPGA_MCS_IN-	H38-FMC_LA32_N

ISOLATION TECHNIQUES USED ON THE ADRV9001 EVALUATION CARD

Given the density of sensitive and critical signals, significant isolation challenges are faced when designing a PCB for the ADRV9001. Isolation requirements listed below were followed to accurately evaluate the ADRV9001 device performance. Analytically determining aggressor-to-victim isolation in a system is very complex and involves considering vector combinations of aggressor signals and coupling mechanisms.

Isolation Goals

Table 107 lists the isolation targets for each RF channel-to-channel combination type. To meet these goals with significant margin, isolation structures were designed into the ADRV9001 evaluation board.

Table 107. Port to Port Isolation Goals

	30 MHz to 1 GHz	1 GHz to 6 GHz
Tx1 to Tx2	75 dB	70 dB
Tx1 to Rx1A/Rx1B	75 dB	70 dB
Tx1 to Rx2A/Rx2B	75 dB	70 dB
Rx1A/Rx1B to Rx2A/Rx2B	70 dB	65 dB
Rx1A to Rx1B	70 dB	65 dB

Isolation Between RF IO Ports

These are the primary coupling mechanisms between RF IO paths on the evaluation board:

- Magnetic field coupling
- Surface propagation
- Cross domain coupling via ground

To reduce the impact of these coupling mechanisms on the ADRV9001 customer evaluation board, several strategies were used. Large slots are opened in the ground plane between RF IO paths. These discontinuities prevent surface propagation. These structures consist of a combination of slots and square apertures. Both structures are present on every copper layer of the PCB stack. The advantage of using square apertures is that signals can be routed between the openings without disturbing the isolation benefits that the array of apertures provides. A careful designer will notice various bends in the routing of differential paths. These routes were developed and tuned through

iterative electromagnetic simulation to minimize magnetic field coupling between differential paths. These techniques are illustrated in Figure 256.

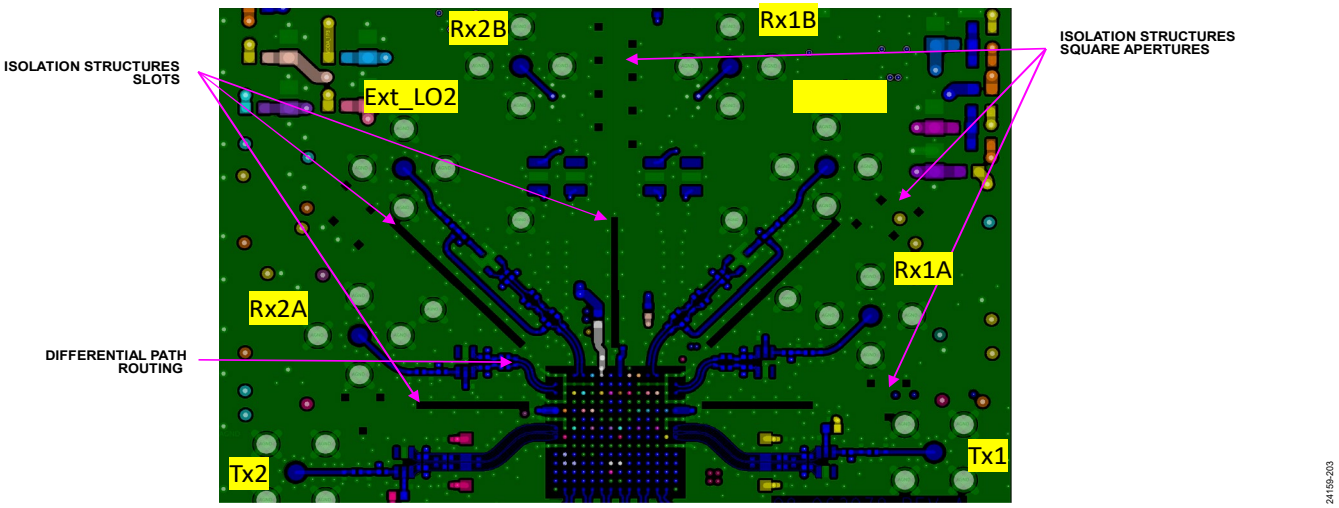


Figure 256. RF I/O Isolation Structures

When utilizing the proposed isolating structures, it is important to place ground vias around the slots and apertures. Figure 257 illustrates the methodology used on the ADRV9001 evaluation card. When slots are used, ground vias should be placed at each end of the slots and along each side. When square apertures are used, at least one single ground via should be placed adjacent to each square. These vias should be through-hole vias connecting the top to the bottom layer and all layers in between. The function of these vias is to steer return current to the ground planes near the apertures.

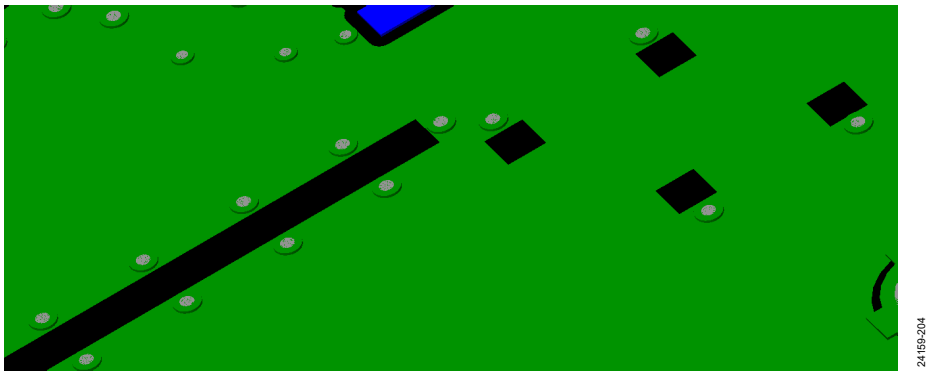


Figure 257. Current Steering Vias Placed Near Isolation Slots and Square Apertures

For accurate slot spacing and square apertures layout, simulation software should be used when designing a PCB for an ADRV9001 based transceiver. As a general rule, spacing between square apertures should be no more than 1/10 of the shortest wavelength supported. The wavelength can be calculated using Equation 1

$$Wave\ length[m] = \frac{300}{Frequency[MHz] \times \sqrt{\varepsilon_r}} \quad (1)$$

where:

ε_r is the dielectric constant of the isolator material. For ISOLA I-speed material, ε_r = 3.56 and for FR4-408 HR material, ε_r = 3.77.

Example: given a maximum RF signal frequency of 6 GHz, for ISOLA I-speed material, using microstrip structures, and ε_r = 3.56, the minimum wavelength is approximately 26.4 mm To fulfil the 1/10 of a wavelength rule, square aperture spacing should be at a distance of 2.64 mm or closer.

Additional shielding is provided by using connecting VSSA balls under the device to form a shield around RF IO ball pairs. This ground provides a termination for stray electric fields. Figure 250 shows how this is done for Tx1. The same is done for each set of sensitive RF I/O ports. Ground vias are used along single ended RF IO traces. Optimal via spacing is 1/10 of a wavelength, but that spacing can vary somewhat due to practical layout considerations.

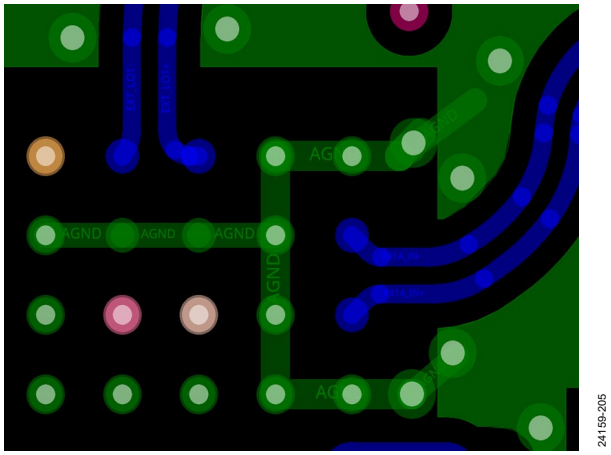


Figure 258. Shielding of Rx Launches

RF IO baluns are spaced and aligned to reduce magnetic coupling from the structures in the balun package. Care must also be taken to reduce cross talk over shared grounds between baluns. Another precaution taken involved placing and orienting SMA connectors to minimize connector to connector coupling between ports.

POWER SUPPLY RECOMMENDATIONS

Aim of this chapter is to provide reader with an overview of ADRV9001 power supply solution. Power supply solution for ADRV9001 will change based on user desired mode of operation (FDD, TDD, DPD, Tx tracking, number of active Rx inputs, number of active Tx outputs, internal LOs vs external LOs, 1.0V power domain) therefore it is important to understand available configurations and optimize PCB layout based on selected mode. Power supply solution implements on an EVB will be used as a reference in this chapter.

POWER MANAGEMENT CONSIDERATIONS

The ADRV9001 family of devices requires four or five different power supply domains:

(1) 1.0V Digital: this supply is connected to the device through the two VDIG_1P0 pins. This is the supply that feeds all digital blocks. Care should be taken to properly isolate this supply from all analog signals on the PCB to avoid noise corruption. This supply input can have a tolerance of ±5%, but note that the total tolerance must include the tolerance of the supply device added to the voltage drop of the PCB. In some modes of operation, this supply can draw high current. It is critical that the input traces for these two inputs be balance (same impedance for inputs) and as thick as possible to minimize the I×R drop.

(2) 1.0V Analog: these supplies are collectively referred to in the datasheet as the VDDA_1P0 supply. This power domain is optional and suggested to be used only in scenario where minimum power consumption is to be achieved. It will require low noise 1.0V power domain available in the end system. In mode of operation where VDDA_1P0 is not used, this 1.0V power domain is created internally inside the ADRV9001 by its own LDOs. This power domain, supplies voltage to noise sensitive blocks of the ADRV9001. If user intend to provide external 1.0V, care should be taken to ensure very low noise level on this power domain. This supply input has a tolerance of ±2.5%.

(3) 1.3V Analog: these supplies connect to all functional blocks in the device through number of different input pins. They are collectively referred to in the datasheet as the VDDA_1P3 supply. Each input should be treated as a noise-susceptible input, meaning proper decoupling and isolation techniques should be followed to avoid crosstalk between channels. The tolerance on these supply inputs is ±2.5%. If VDDA_1P0 is utilized, some of VDDA_1P3 supply pins will change its intended voltage input level from 1.3V to 1.0V. This chapters provides detail overview of those modifications.

(4) 1.8V Analog: these supplies are primarily used to supply the transmitter outputs. They also supply current for multiple transmitter, receiver, converter, and auxiliary converter blocks. They are collectively referred to in the datasheet as the VDDA_1P8 supply. This supply has a tolerance of ±5%.

(5) 1.8V Digital: This is an interface supply. The VDIGIO_1P8 supply is a separate power domain shared with the BBP interface. The nominal input voltage on this supply is 1.8V with a tolerance of ±5%. This input serves as the voltage reference for the digital interface (SPI and SSI), DGPIO, and digital control inputs.

IMPORTANT:

During operation, supply currents can vary significantly, especially if operating in TDD mode. The supply needs to have adequate capacity to provide the necessary current (as indicated on the datasheet) so that performance criteria over all process and temperature variations are maintained. ADI recommends adding at least 15% margin to all supply maximums to ensure proper operation under all conditions.

POWER SUPPLY SEQUENCE

The ADRV9001 requires a specific power-up sequence to avoid undesired power-up currents. The optimal power-on sequence requires VDD_1P0 to power up first. The VDDA_1P3, VDDA_1P8 and VDD_1P8 supplies must then power up after the VDD_1P0 supply. If VDDA_1P0 is utilized, it must be powered up after VDDA_1P3 and VDDA_1P8 are enabled. The user must toggle the RESET signal after power has stabilized prior to configuration.

The power-down sequence recommendation is similar to power-up. All supplies should be disabled in any order (or all together) before VDIG_1P0 is disabled. If such a sequence is not possible, then all supplies should have their sources disabled simultaneously to ensure there is no back feeding circuits that have not been powered down.

POWER SUPPLY DOMAIN CONNECTIONS

One key aspect to ensuring good performance is careful low-noise power management design. Table 108 lists the pin number, the pin type and name, expected voltage on that pin and a brief description of routing technique together with the block it powers on the chip. Power supply to the ADRV9001 should be delivered following star configuration where, a separate trace from a common power plane is used to power each power supply pins. Refer to the user guide section on PCB layout and an evaluation board CAD layout file for details.

Table 108. Power Supply Pins and Functions

Pin No.	Type	Pin Name	Voltage [V]	Description
A1, A2, A13, A14, B2 to B5, B10 to B13, C2, C5, C10, C13, D1 to D6, D9 to D14, E6, E9, F1 to F3, F6 to F9, F12 to F14, G2, G13, J1 to J14,	ANALOG	VSSA	0	Analog Supply voltage (V_{SS}).
A5	ANALOG	VRFVCO2_1P3	1.3	1.3 V Internal LDO Input Supply for RF LO2 VCO and LO generation circuitry. This pin is sensitive to supply noise.
A6	ANALOG	VRFLO2_1P0	1.0	1.0 V internal supply node for RF LO2 LO generation circuitry. Connect this pin together with B6 and bypass with a 4.7µF capacitor, when internal LDO operated from A5 input is in use. Provide 1.0V supply to this pin when internal LDO operated from A5 is not in use
A9	ANALOG	VRFLO1_1P0	1.0	1.0 V internal supply node for RF LO1 LO generation circuitry. Connect this pin together with B9 and bypass with a 4.7µF capacitor, when internal LDO operated from A10 input is in use. Provide 1.0V supply to this pin when internal LDO operated from A10 is not in use
A10	ANALOG	VRFVCO1_1P3	1.3	1.3 V Internal LDO Input Supply for RF LO1 VCO and LO generation circuitry. This pin is sensitive to supply noise.
B6	ANALOG	VRFVCO2_1P0	1.0	1.0 V internal supply node for RF LO2 VCO circuitry. Connect this pin together with A6 and bypass with a 4.7µF capacitor, when internal LDO operated from A5 input is in use.
B9	ANALOG	VRFVCO1_1P0	1.0	1.0 V internal supply node for RF LO1 VCO circuitry. Connect this pin together with A9 and bypass with a 4.7µF capacitor, when internal LDO operated from A10 input is in use.
C6	ANALOG	VANA2_1P0	1.0	1.0 V internal supply node for Tx2/Rx2 Baseband Circuits, Transimpedance Amplifier (TIA), Tx Trans-conductance (GM), Baseband Filters, and Auxiliary DACs/ADCs. For normal operation leave this pin unconnected.
C7	ANALOG	VANA2_1P3	1.3	1.3 V Internal LDO Input Supply for Tx2/Rx2 Baseband Circuits, Transimpedance Amplifier (TIA), Tx Trans-conductance (GM), Baseband Filters, and Auxiliary DACs/ADCs. This pin is sensitive to supply noise.
C8	ANALOG	VANA1_1P3	1.3	1.3 V Internal LDO Input Supply for Tx1/Rx1 Baseband Circuits, Transimpedance Amplifier (TIA), Tx Trans-conductance (GM) and Baseband Filters. This pin is sensitive to supply noise.
C9	ANALOG	VANA1_1P0	1.0	1.0 V internal supply node for Tx1/Rx1 Baseband Circuits, Transimpedance Amplifier (TIA), Tx Trans-conductance (GM) and Baseband Filters. For normal operation leave this pin unconnected.
E1	ANALOG	VRX2LO_1P0	1.0	1.0 V internal supply node for Rx2 LO buffers and mixers. This pin is sensitive to supply noise. Bypass this pin with a 4.7µF capacitor.
E2	ANALOG	VRX2LO_1P3	1.3/1.0	1.3 V Internal LDO Input Supply for Rx2 LO buffers and mixers. Provide 1.0V supply to this pin when internal LDO is not used. This pin is sensitive to supply noise.
E4	ANALOG	VRFSYN2_1P3	1.3	1.3 V Supply for RF LO2 Synthesizer. This pin is sensitive to supply noise.
E5	ANALOG	VCLKSYN_1P3	1.3	1.3 V Supply for Clock Synthesizer. This pin is sensitive to supply noise.

Pin No.	Type	Pin Name	Voltage [V]	Description
E10	ANALOG	VAUXSYN_1P3	1.3	1.3 V Supply for Auxiliary Synthesizer. This pin is sensitive to supply noise.
E11	ANALOG	VRFSYN1_1P3	1.3	1.3 V Supply for RF LO1 Synthesizer. This pin is sensitive to supply noise.
E13	ANALOG	VRX1LO_1P3	1.3/1.0	1.3 V Internal LDO Input Supply for Rx1 LO buffers and mixers. Provide 1.0V supply to this pin when internal LDO is not used. This pin is sensitive to supply noise
E14	ANALOG	VRX1LO_1P0	1.0	1.0 V internal supply node for Rx1 LO buffers and mixers. This pin is sensitive to supply noise. Bypass this pin with a 4.7µF capacitor.
G3	ANALOG	VTX2LO_1P3	1.3/1.0	1.3 V Supply for Tx2 LO buffers, upconverter and LO delay. Provide 1.0V supply to this pin when internal LDO is not used. This pin is sensitive to supply noise.
G5	ANALOG	VCLKVCO_1P3	1.3	1.3 V Internal LDO Input Supply for Clock LO VCO and LO generation circuitry. This pin is sensitive to supply noise.
G7	ANALOG	VCONV_1P8	1.8	1.8 V Supply for Tx1/Tx2 DAC and Rx1/Rx2 ADC
G8	ANALOG	VAGPIO_1P8	1.8	1.8 V Supply for AuxDACs, AuxADCx and AGPIO signals.
G10	ANALOG	VAUXVCO_1P3	1.3	1.3 V Internal LDO Input Supply for Auxiliary LO VCO and LO generation circuitry. This pin is sensitive to supply noise.
G12	ANALOG	VTX1LO_1P3	1.3/1.0	1.3 V Internal LDO Input Supply for Tx1 LO buffers, upconverter and LO delay. Provide 1.0V supply to this pin when internal LDO is not used. This pin is sensitive to supply noise.
H2	ANALOG	VANA2_1P8	1.8	1.8 V Supply for Rx2 Mixer, Rx2 Transimpedance Amplifier (TIA), Tx2 Low Pass Filter (LPF) and Internal References.
H3	ANALOG	VTX2LO_1P0	1.0	1.0 V internal supply node for Tx2 LO buffers, upconverter and LO delay. For normal operation leave this pin unconnected.
H5	ANALOG	VCLKVCO_1P0	1.0	1.0 V internal supply node for Clock LO VCO and LO generation circuitry. Bypass this pin with a 4.7µF capacitor.
H7	ANALOG	VCONV_1P0	1.0	1.0 V internal supply node for Rx ADCs and Tx DACs. Bypass this pin with a 4.7µF capacitor.
H8	ANALOG	VCONV_1P3	1.3/1.0	1.3 V Internal LDO Input Supply for Rx ADCs and Tx DACs. Provide 1.0V supply to this pin when internal LDO is not used. This pin is sensitive to supply noise.
H10	ANALOG	VAUXVCO_1P0	1.0	1.0 V internal supply node for Auxiliary LO VCO and LO generation circuitry. Bypass this pin with a 4.7µF capacitor.
H12	ANALOG	VTX1LO_1P0	1.0	1.0 V internal supply node for Tx1 LO buffers, upconverter and LO delay. For normal operation leave this pin unconnected.
H13	ANALOG	VANA1_1P8	1.8	1.8 V Supply for Rx1 Mixer, Rx1 Transimpedance Amplifier (TIA), Tx1 Low Pass Filter (LPF), Xtal oscillator, DEV_CLK circuitry and Internal References
K4	ANALOG	VSSA/TESTCK+	0	Connect to VSSA for normal operation.
K5	ANALOG	VSSA/TESTCK-	0	Connect to VSSA for normal operation.
L7, L8	DIGITAL	VDIG_1P0	1.0	1.0 V Digital Core. Connect Pin L7 and Pin L8 together. Use a wide trace to connect to a separate power supply domain. Provide reservoir capacitance close to the chip.
M7	DIGITAL	VDIGIO_1P8	1.8	1.8 V supply input for Data Port Interface (CMOS-SSI/LVDS-SSI), SPI Signals, Control Input/Output Signals and DGPIO Interface.
M8	DIGITAL	VDIG_0P9	0.9	0.9 V internal supply node for digital circuitry. Bypass this pin with a 4.7µF capacitor.
N7, N8, P1, P14	DIGITAL	VSSD	0	Digital Supply voltage (V_{SS})

POWER SUPPLY ARCHITECTURE

The diagram in Figure 259 outlines the power supply configuration used on the ADRV9001 evaluation board. This configuration follows the recommendations outlined in Table 108. This diagram includes the use of C/FB/C/FB cascaded filters and ferrite beads for additional RF isolation. The use of ferrite beads and 0Ω resistors in EVB power supply solution accomplishes 3 goals:

- Serve as place holders for ferrite beads or other filter devices that may be needed when users encounter RF noise problems in their application and additional isolation is required. For more details regarding ferrite bead selection, refer to RF and Clock Synthesizer Supplies section of this document.
- Ensures following of power routing recommendations outlined in the Printed Circuit Board Layout Recommendations section. Ferrite beads and resistor placeholders in series force the use of separate traces to deliver different power domains to the ADRV9001 device.
- Provide a place in the circuit where the current can be monitored and measured for debugging purposes. For this case, the 0Ω components or ferrite bead can be replaced by very low-impedance shunt resistors and the voltage can be measured to determine current to the specified input ball.

For more details on exact power supply implementation, refer to the ADRV9001 Customer Evaluation Board Schematic that is supplied with an evaluation kit.

The ADRV9001 evaluation board also provides and on-board current and voltage sensors (ADM1293) which monitors and reports back to TES power consumed by TRx in selected mode of operation. Idea here is to provide end user with live feedback from an evaluation system regarding active power consumption. Current readbacks from those sensors are accurate within 2.5% tolerance. If those readback numbers are used to estimate overall power for power supply design, user should add an extra power margin to accommodate dynamic conditions. First paragraph in this section provides more suggestions.

EVB Power Supply overview

The diagram in Figure 259 outlines the Power Supply configuration used on ADRV9001 evaluation board. This supply architecture follows conservative approach to the power supply design. Switch Mode regulator (ADP5056) is used to achieve power efficiency while generating domains that supply ADRV9001. Remote sensing configuration is utilized to take in account voltage drop in filters and ensure power domains accuracy at ADRV9001 input pins.

The ADP5056 contains three switch-mode step down regulators. Each of those regulators produces a different power domain that supplies power to the ADRV9001. They operates as follow:

- Channel 1 – generates 1.3V which supplies voltage for ADRV9001 1.3V power domains. Sensing is done after C/FB/C/FB cascaded filters and current sensing shunt resistor ensuring that 1.3V analog power domain voltage stays within datasheet specification.
    - The ADRV9001 EVB supports also an optional 1.0V power domain. This domain could be generated by using an on-board LDO (ADP1762) That LDO utilize its own remote sensing scheme to ensure that 1.0V analog power domain voltage stays within datasheet specification.
- Channel 2 - generates 1.0V which supplies voltage for ADRV9001 digital domain. Sensing is done after C/FB/C/FB cascaded filters and current sensing shunt resistor to ensure that 1.0V digital power domain voltage stays within datasheet specification.
- Channel 3 - generates 1.8V which supplies voltage for ADRV9001 digital and analog power domains. The C/FB/C/FB cascaded filters are used to isolate analog power domain from digital power domain. Since more than one power domain is produced from single source, remote sensing is done before C/FB/C/FB cascaded filters.

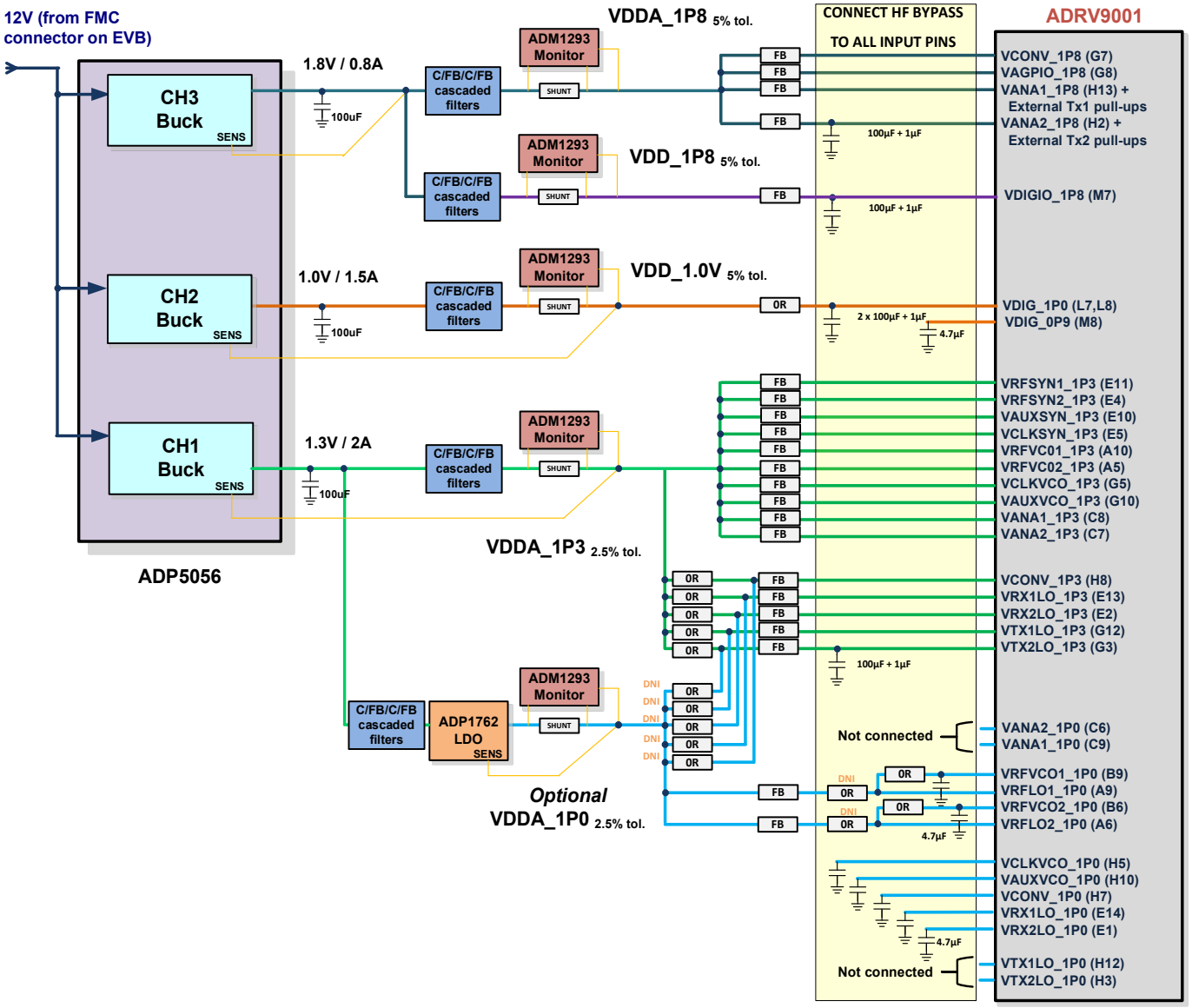


Figure 259. Power Supply Connection Diagram

Power domain filtering

Power signals to the ADRV9001 are further isolates from each other using C/FB/C/FB cascaded filters followed by high current ferrite beads. Figure 260 outlines at filtered approach implemented on an EVB.

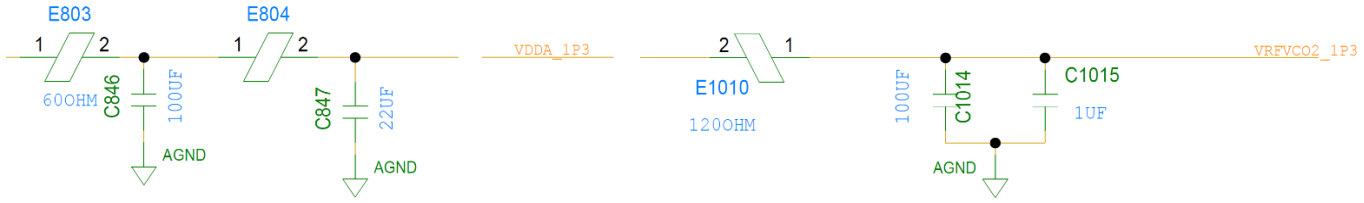


Figure 260. An example of power domain filtering implemented on ADRV9001 EVB (1.3V -> VRFVCO2_1P3)

Figure 261 provides simulation schematic with power supply filter components used on EVB power rails in ADS environment. It basically attempts to simulate frequency response of filter outlined in Figure 260.

- The E1, C1, E2, and C2 elements be located right at each rail input pin.
- The E3, and C4 elements be located right at trace that feeds particular pin.
- The C4 element is the RF capacitor that should be placed right at each ADRV9001 pin using the rail. For more information about placing of those capacitors refer to PCB layout section and EVB PCB file itself.
- The combined DC resistance of the three ferrites is 0.106Ω.

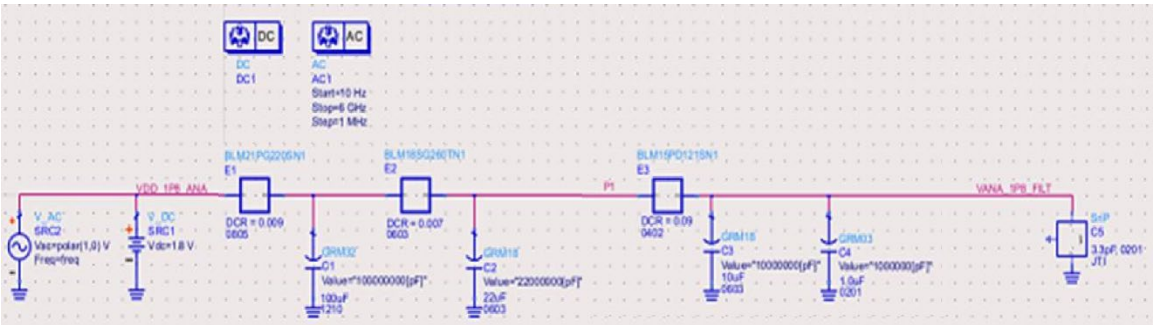


Figure 261. ADS model of C/FB/C/FB cascaded filter utilized on ADRV9001 EVBs

Finally Figure 262 provides results of simulation that is described in Figure 261. The overall frequency response is better than typical bank of capacitor. The region >1 GHz outlines performance of implemented cascaded C/FB/C/FB filtering approach.

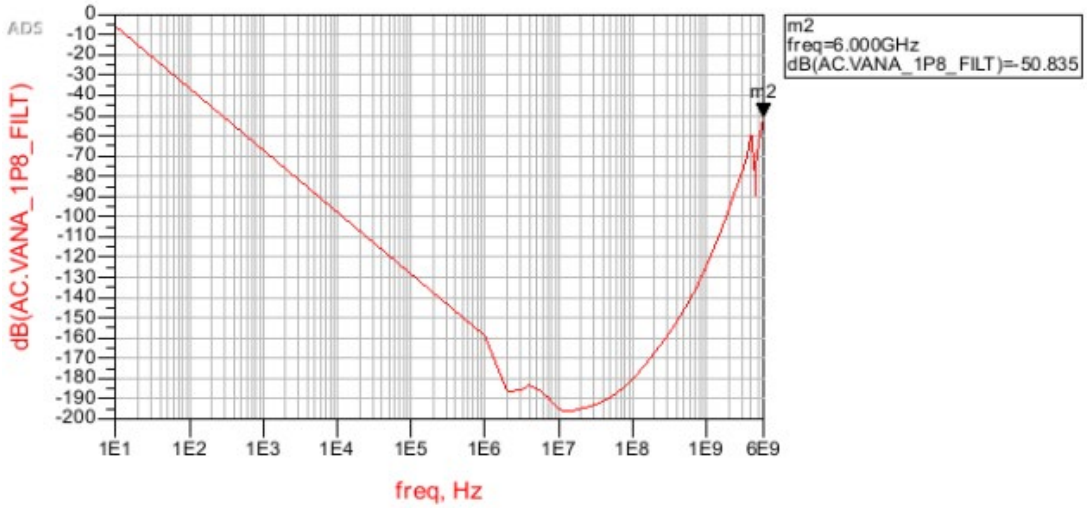


Figure 262. Response of C/FB/C/FB cascaded filter utilized on ADRV9001 EVBs

Care must be taken when introducing multiple inductors and ferrite beads in series creates issue with potential IxR voltage drop on them that could violate recommended power domain voltage accuracy. Power supply solution on ADRV9001 EVB uses sense line to monitor the voltage output after the ferrite bead (where possible). This approach ensures that the voltage drop resulting from the cascaded filters and FB resistance is taken into account and the voltage level delivered to the ADRV9001 is in line with expected accuracy. In scenarios where single power domain powers multiple power pins, current tends to be distributed over multiple pins and this helps with minimizing IxR voltage drop on FB components.

RF AND CLOCK SYNTHESIZER SUPPLIES

The noise performance of the power domain used to power the RF blocks will directly affect the overall TRx phase noise. Power domains should be powered using separate traces with extra isolation using a low DCR ferrite bead such as the Murata BLM18KG121TN1D or similar device:

The power supply noise rejection on the synthesizer power input pins is very low. This means that any noise ripple on these pins will affect the synthesizer performance. The RF synthesizer requires more critical supply decoupling because any noise or variation in voltage that occurs during operation is directly imposed on the RF channel. Refer to Figure 259 for an example of how the power supply connections are made on the ADRV9001 evaluation card.

The synthesizers are more susceptible to low frequency noise than other supplies because they have programmable loop filters. The loop filter bandwidth directly affects the supply noise rejection on the synthesizers. For example, if the loop filter bandwidth is 50 kHz, then any noise on the supply below 50 kHz will not be filtered. The roll-off of the loop filter provides the noise rejection above 50 kHz.

For each power domain, a ferrite bead with high isolation at the frequency of operation is recommended to help isolate the pin from the supply source for best performance. This is especially important when operating in TDD mode. Such high isolation ferrite beads tend to also have high dc resistance. This tradeoff is acceptable for the synthesizer power inputs because their low current draws will result in relatively small voltage drops that are well within the supply tolerance range.

For domains that consume higher amount of current care should be taken to ensure that voltage drop on in-series Ferrite Bead does not exceed power domain voltage tolerances. Good candidates for such Ferrite Beads with low DCR are PANASONIC EXC-ML20A390U 39OHM or MURATA, BLM15AX300SN1D or BLM18KG121TN1D.

For power domains which does not consume higher amount of current and at the same time higher level of isolation is desire a Ferrite Beads with higher DCR are recommended. An example of recommended high DCR Ferrite Bead would be Taiyo Yuden, BK1005LL470-T

POWER SUPPLY CONFIGURATIONS

From power supply implementation point of view, the ADRV9001 can work in multiple configurations. This section outlines them in details. Depends on final application of ADRV9001 in end system user have a freedom to implement different ways to power the IC. Final solution will depend on:

1. if external 1.0V power domain is utilized or not
2. type of application: FDD, TDD
    a. In FDD case if DPD and/or Tx tracking calibrations are utilized or not
3. number of active Rx inputs and number of active Tx outputs,
    a. Note should be taken that in scenario where Rx2/Tx2 are used and Rx1/Tx1 are not used (not recommended scenario from an optimal power savings point of view) the power supply to VANA1_1P3 (C8) needs to be present.
4. LO scheme:
    a. Internal LOs with internal PLL+VCO+LO_GEN powered by internal LDO
    b. External LOs with internal LO_GEN powered by
        i. internal LDO
        ii. external LDO

    Note should be taken that even in scenarios when external LO are utilized the VRFVCO1_1P0 (B9), VRFVC01_1P3 (A10) and VRFSYN1_1P3 (E11) needs to be powered up. The RF PLL1 is used to generate test signals during initialization calibration stage therefore power supply to those blocks needs to be provided. After initial calibrations are performed those blocks are powered down internally.

This section will outline how decisions based on descriptions above could impact final power supply interconnectivity.

Figure 263 outlines recommended power supply interconnective in scenarios where user might want to utilize external 1.0V analog power domain. In such mode number of power supply input pins that are assigned to 1.3V analog power domain will have to be re-connected physically on PCB to new 1.0V analog power domain. In case where external RF LO is used, more power savings can be achieved by physically disconnecting/grounding LO supplies responsible for powering up internal LO generation blocks.

Figure 265 and Figure 264 provides user with recommendations how to interconnect power supply in case where not all of available RF IOs (Tx and Rx) are utilized in end application. It should be noted that in order to perform Tx1 tracking calibration or DPD on Tx1, Rx1 data path must to be available to observe Tx output. The same relationship exists between Tx2 and Rx2. It is not possible to perform Tx1 tracking calibration or DPD operation on Tx1 output using Rx2 data path (and vice versa, Tx2 using Rx1).
Figure 265 provides suggestions for 1T1R configuration. In TDD applications, in order to achieve lowest possible current consumption in deep sleep state user should utilize Rx1 and Tx1 data paths and disable Rx2 and Tx2 data paths.

Once any of configurations outlined in Figure 263, Figure 265 or Figure 264 is implemented, user needs to initialize ADRV9001 device with correct initialization settings in described in data structure TBD.

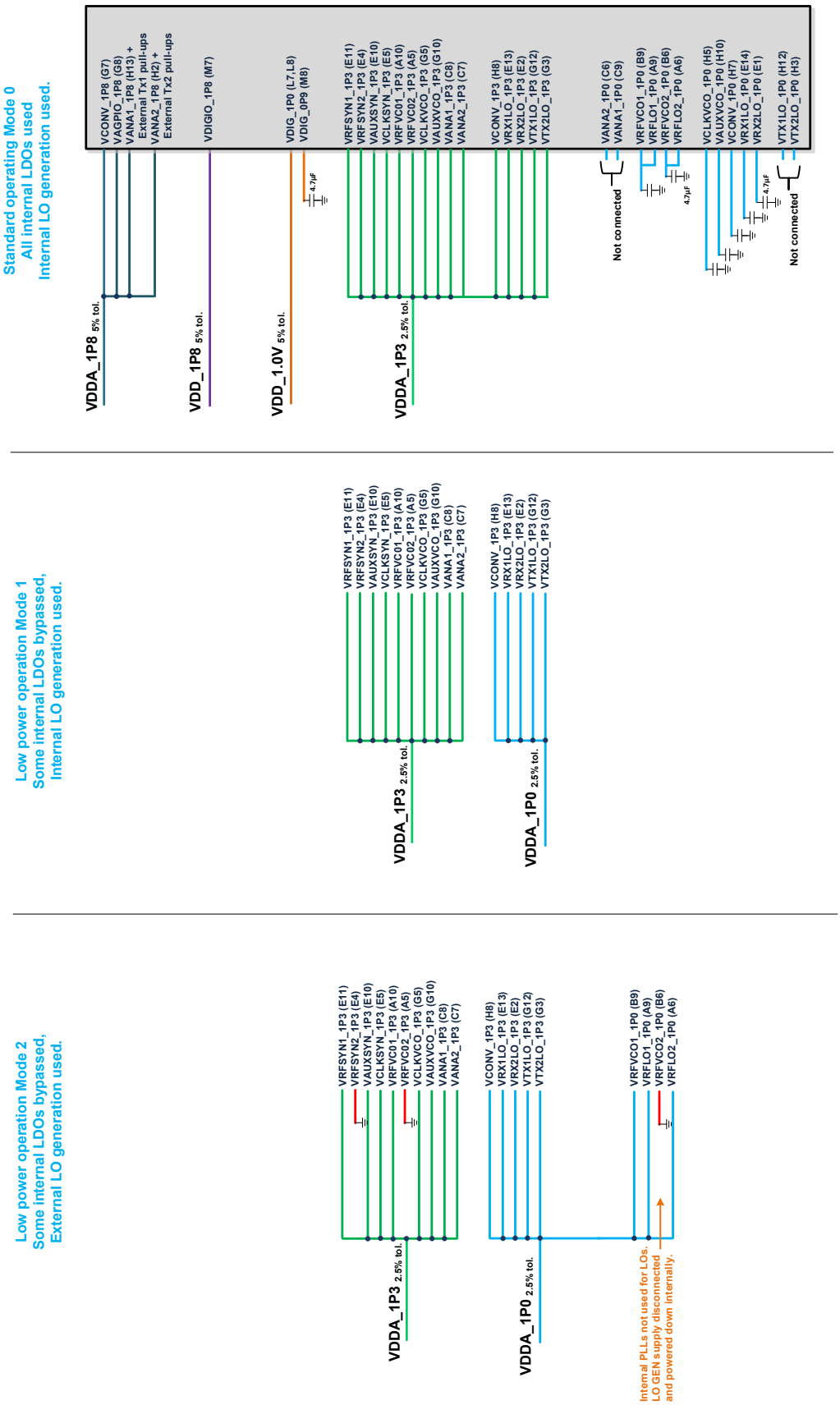


Figure 263. Available modes for external 1.0V power domain and LO power supply configuration

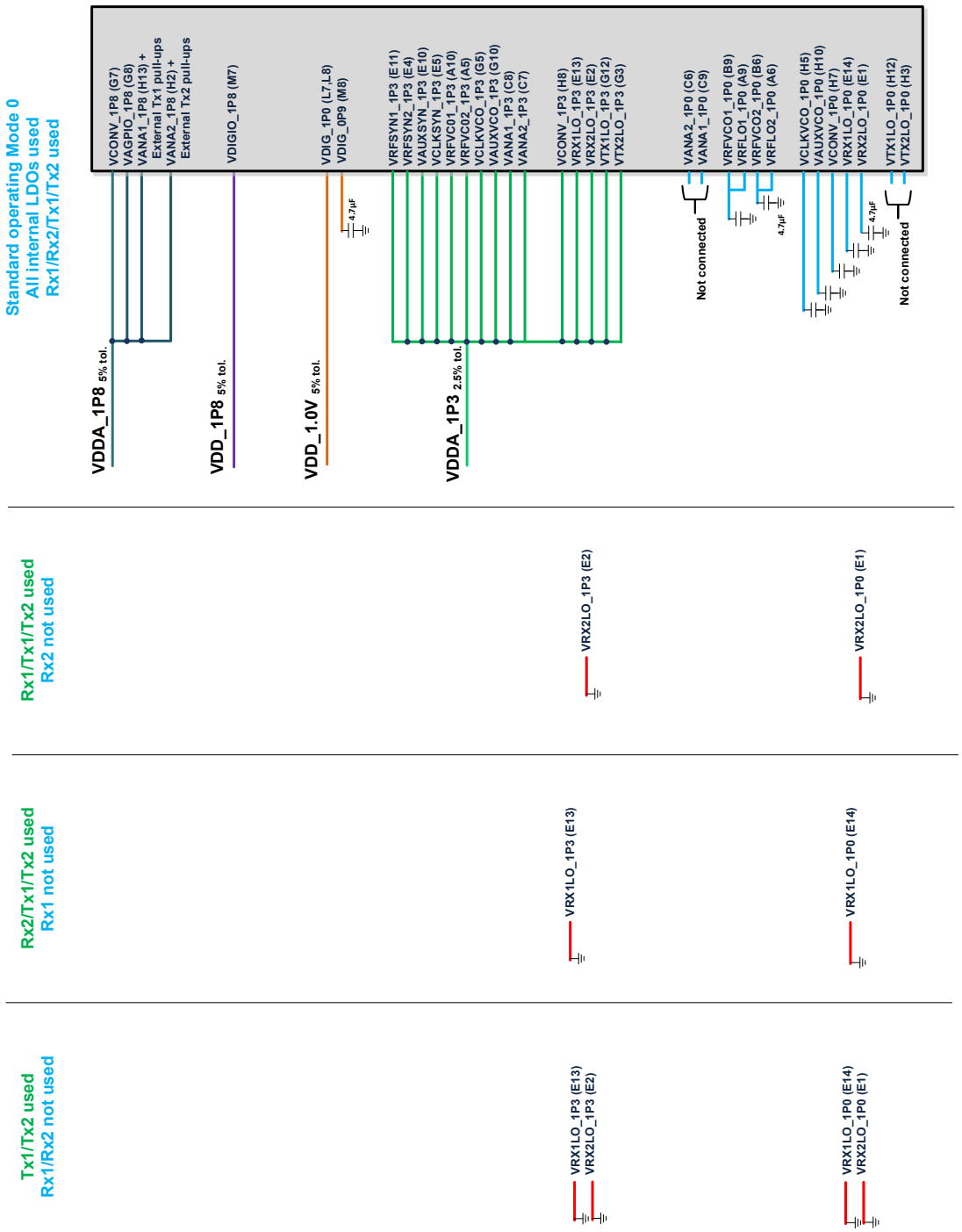


Figure 264. Power supply modes for different number of active Tx RF IOs

Figure 265. Power supply modes for different number of active Rx and Tx RF IOs

In case where not all RF IOs or other interface pins are utilized in end application user should follow Table 109 for recommendation what to do with unused pins.

Table 109. Instruction Explaining How to Handle ADRV9001 Unused Pins

Pin No.	Type	Mnemonic	Unused instructions
A1, A2, A13, A14, B2 to B5, B10 to B13, C2, C5, C10, C13, D1 to D6, D9 to D14, E6, E9, F1 to F3, F6 to F9, F12 to F14, G2, G13, J1 to J14,	Input	VSSA	Not applicable.
A3, A4	Input	EXT_LO2+, EXT_LO2−	Connect to VSSA.
A5	Input	VRFVCO2_1P3	Connect to VSSA when unused
A6	Input/Output	VRFLO2_1P0	Not applicable.
A7	Input	MODEA	Not applicable.
A8	Input	RBIAS	Not applicable.
A9	Input/Output	VRFLO1_1P0	Not applicable.
A10	Input	VRFVCO1_1P3	Connect to VSSA when unused
A11, A12	Input	EXT_LO1-, EXT_LO1+	Connect to VSSA.
B1, C1	Input	RX2A-, RX2A+	When accidentally enabled, bias voltage could be present on those inputs. Connect to VSSA thru capacitor.
B6	Output	VRFVCO2_1P0	Connect to VSSA when unused
B7	Input	AUXADC_2	Do not connect.
B8	Input	AUXADC_1	Do not connect.
B9	Output	VRFVCO1_1P0	Connect to VSSA when unused
B14, C14	Input	RX1A-, RX1A+	When accidentally enabled, bias voltage could be present on those inputs. Connect to VSSA thru capacitor.
C3, C4	Input	RX2B+, RX2B-	When accidentally enabled, bias voltage could be present on those inputs. Connect to VSSA thru capacitor
C6	Input/Output	VANA2_1P0	Connect to VSSA when unused
C7	Input	VANA2_1P3	Connect to VSSA when unused
C8	Input	VANA1_1P3	Not applicable.
C9	Input/Output	VANA1_1P0	Not applicable.
C11, C12	Input	RX1B-, RX1B+	When accidentally enabled, bias voltage could be present on those inputs. Connect to VSSA thru capacitor.
D7, D8	Input	MCS+, MCS-	Connect to VSSA.
E1	Output	VRX2LO_1P0	Connect to VSSA when unused
E2	Input	VRX2LO_1P3	Connect to VSSA when unused
E3, E12, F4, F5, F10, F11, G4, G6, G9, G11, H6, H9	Input/Output	AGPIO_xx	Do not connect.
E4	Input	VRFSYN2_1P3	Connect to VSSA when unused
E5	Input	VCLKSYN_1P3	Not applicable.
E7, E8	Input	DEV_CLK_IN+, DEV_CLK_IN-	Not applicable.
E10	Input	VAUXSYN_1P3	Not applicable.
E11	Input	VRFSYN1_1P3	Connect to VSSA when unused
E13	Input	VRX1LO_1P3	Not applicable.
E14	Output	VRX1LO_1P0	Not applicable.
G1, H1	Output	TX2+, TX2-	Do not connect.
G3	Input	VTX2LO_1P3	Connect to VSSA when unused
G5	Input	VCLKVCO_1P3	Not applicable.

Pin No.	Type	Mnemonic	Unused instructions
G7	Input	VCONV_1P8	Not applicable.
G8	Input	VAGPIO_1P8	Not applicable.
G10	Input	VAUXVCO_1P3	Not applicable.
G12	Input	VTX1LO_1P3	Not applicable.
G14, H14	Output	TX1+, TX1-	Do not connect.
H2	Input	VANA2_1P8	Not applicable.
H3	Output	VTX2LO_1P0	Connect to VSSA when unused
H4	Input	AUXADC_3	Do not connect.
H5	Output	VCLKVCO_1P0	Not applicable.
H7	Output	VCONV_1P0	Not applicable.
H8	Input	VCONV_1P3	Not applicable.
H10	Output	VAUXVCO_1P0	Not applicable.
H11	Input	AUXADC_0	Do not connect.
H12	Output	VTX1LO_1P0	Not applicable.
H13	Input	VANA1_1P8	Not applicable.
K1	Input	SPI_CLK	Not applicable.
K2	Input/Output	SPI_DIO	Not applicable.
K3	Input	RX2_EN	Do not connect.
K4	Input	VSSA/TESTCK+	Not applicable.
K5	Input	VSSA/TESTCK-	Not applicable.
K6 to K11, L4 to L6, L9 to L11	Input/Output	DGPIO_xx	Do not connect.
K12	Input	RX1_EN	Do not connect.
K13	Input	RESETB	Not applicable.
K14	Output	GP_INT	Do not connect.
L1	Input	SPI_EN	Not applicable.
L2	Output	SPI_DO	In SPI 3-wire mode, do not connect.
L3	Input	TX2_EN	Do not connect.
L7, L8	Input	VDIG_1P0	Not applicable.
L12	Input	TX1_EN	Do not connect.
L13	Input	MODE	Connect to VSSA.
L14	Output	DEV_CLK_OUT	Do not connect.
M1	Output	RX2_IDATA_OUT-	Do not connect
M2	Output	RX2_IDATA_OUT+	Do not connect
M3	Output	RX2_DCLK_OUT-	Do not connect
M4	Output	RX2_DCLK_OUT+	Do not connect
M5	Input/Output	DGPIO_15/TX2_DCLK_OUT+	Do not connect.
M6	Input/Output	DGPIO_14/TX2_DCLK_OUT-	Do not connect.
M7	Input	VDIGIO_1P8	Not applicable.
M8	Output	VDIG_0P9	Not applicable.
M9	Input/Output	DGPIO_12/TX1_DCLK_OUT-	Do not connect.
M10	Input/Output	DGPIO_13/TX1_DCLK_OUT+	Do not connect.
M11	Output	RX1_DCLK_OUT+	Do not connect
M12	Output	RX1_DCLK_OUT-	Do not connect
M13	Output	RX1_IDATA_OUT+	Do not connect
M14	Output	RX1_IDATA_OUT-	Do not connect
N1	Output	RX2_STROBE_OUT-	Do not connect
N2	Output	RX2_STROBE_OUT+	Do not connect
N3	Output	RX2_QDATA_OUT-	Do not connect
N4	Output	RX2_QDATA_OUT+	Do not connect
N5	Input	TX2_DCLK_IN+	Do not connect.
N6	Input	TX2_DCLK_IN-	Do not connect

Pin No.	Type	Mnemonic	Unused instructions
N7, N8, P1, P14	Input	VSSD	Not applicable.
N9	Input	TX1_DCLK_IN-	Do not connect
N10	Input	TX1_DCLK_IN+	Do not connect.
N11	Output	RX1_QDATA_OUT+	Do not connect
N12	Output	RX1_QDATA_OUT-	Do not connect
N13	Output	RX1_STROBE_OUT+	Do not connect
N14	Output	RX1_STROBE_OUT-	Do not connect
P2	Input	TX2_STROBE_IN+	Do not connect
P3	Input/Output	TX2_STROBE_IN-	Do not connect
P4	Input	TX2_QDATA_IN-	Do not connect
P5	Input	TX2_QDATA_IN+	Do not connect
P6	Input	TX2_IDATA_IN+	Do not connect
P7	Input	TX2_IDATA_IN-	Do not connect
P8	Input	TX1_IDATA_IN-	Do not connect
P9	Input	TX1_IDATA_IN+	Do not connect
P10	Input	TX1_QDATA_IN+	Do not connect
P11	Input	TX1_QDATA_IN-	Do not connect
P12	Input/Output	TX1_STROBE_IN-	Do not connect
P13	Input	TX1_STROBE_IN+	Do not connect

SUMMARY

Circuit board layout and power supply decoupling recommendations covered in this document are the recommendations of the Analog Devices applications engineering team. These recommendations are based on circuit analysis, simulation, best practice layout techniques, and experience with this and other transceiver devices. Designers are strongly encouraged to follow these guidelines carefully when developing their own PCB applications. Deviation from these recommendations could result in sub-standard system performance that is very difficult to isolate after the system has been committed to a PCB design. Contact Analog Devices for demo board layout files and schematics that utilize many of the recommended techniques.

LDO CONFIGURATIONS

Note: Consider this section carefully. Failure to implement the LDO modes correctly may result in an incorrect voltage being applied and potentially damage RF circuits.

VDDA_1p0 is a power domain that powers all the Tx and Rx LO circuits. This domain can be powered using internal LDOs that generate the 1V needed from the 1.3V supply. Alternatively it can be powered externally, bypassing some of internal LDOs. This can be used in cases where the customer wants power savings or already has a 1v supply in the system. Some power savings can be achieved by turning off the LDOs and applying a higher efficiency external power source to the 1V rails. The new 1V external power domain will need to be a low noise supply with a tolerance of ±2.5%.

The LDO configurations on the ADRV9001 can affect the overall power consumption. On the ADRV9001 evaluation board all the internal LDOs are used and they generate the 1v needed for all the rails. This allows for different RF channels to be used while evaluating. In a end system the user will know what RF channels they are using and can tailor the RF channels to the system. Three examples of this can be seen in Figure 266.

To implement an ideal LDO configuration for the end use case, the user will need to:

- Use the application to determine how many Tx, Rx and LO channels are needed
- Design a power supply to implement the determined use case
- Choose what LDO configuration is needed and set the LDO modes appropriately

APPLICATION POWER NEEDS

Deciding what LDOs to use/bypass will depend on the Tx, Rx and LO combinations that are going to be used. For example, in 1T/2R FDD use case the user will not use one of the transmit channels and should then also power down the internal LDOs for this channel. Details on how to wire the powers domains can be found in the Power Supply Configuration section above.

The following board power supply configurations are supported:

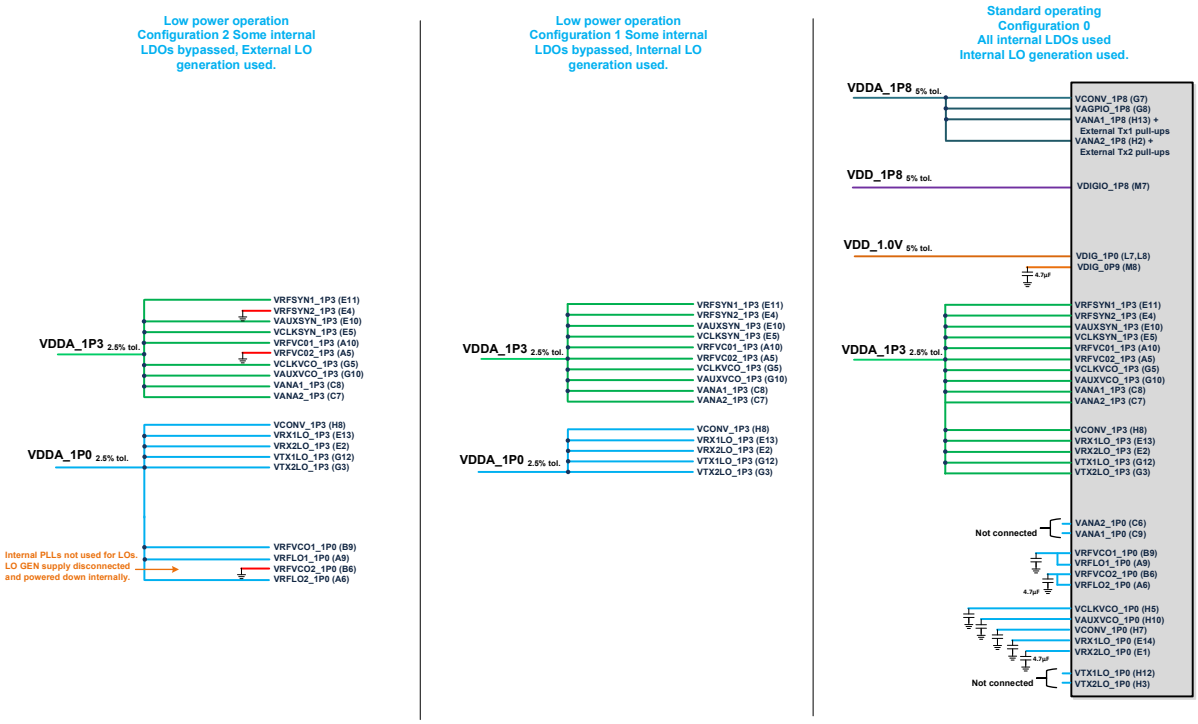


Figure 266. Power Saving LDO Configurations

CHOOSING LDO CONFIGURATION

Before choosing the proper LDO configurations, it is important to understand the power domains used in the system. The LDO configurations will vary depending on the use of RF channels, internal/external LO or if an external VDDA_1p0 is supplied. Care is needed when setting the LDO modes to avoid any potential of incorrect voltages getting supplied to internal blocks.

Table 110 shows the LDO operating modes. The enumerators for each of the modes are used when setting up the adi_adrv9001_powermanagement_Configure function.

Table 110. LDO mode for power saving configurations

	Enumerator	
1	ADI_ADRV9001_LDO_POWER_SAVING_MODE_1	Normal LDO operation.
2	ADI_ADRV9001_LDO_POWER_SAVING_MODE_2	LDO always off. Input and output of the LDO connected to ground. Used when a power domain is not required.
3	ADI_ADRV9001_LDO_POWER_SAVING_MODE_3	Not used.
4	ADI_ADRV9001_LDO_POWER_SAVING_MODE_4	Not used.
5	ADI_ADRV9001_LDO_POWER_SAVING_MODE_5	LDO bypassed. Supply an external source at the level expected at the LDO output (1v, Low noise, 2.5% tolerance)

LDO mode 1 (Normal operation): The LDO is in a normal mode of operation. It is used to generate an on-chip voltage.

LDO mode 2 (LDO always off via PCB wiring): An LDO is not required and is permanently powered down via shorting its Input pin, and its Output pin (if available) to ground on the PCB. Generally, this mode is used if the entire power domain is not required. No software control of the LDO is available.

LDO mode 5 (LDO in bypass mode): An LDO is not required and is placed into bypass mode via software control. However, the power domain is still required, and the customer applies a high efficiency power source at the Input pin. The purpose of this mode is to allow the user to use higher efficiency power sources to supply the ADRV9001, as opposed to having the power overhead of associated with an LDO.

In Table 111 Each of the 19 LDOs is listed and constraints provided to explain when each can be powered on, off or bypassed.

Table 111. LDO Constraints

Index	LDO	Constraints	Input Pin	Output Pin
0	GP_LDO_1	Do not power down or bypass.	VANA1_1P3	VANA1_1P0
1	DEV_CLK_LDO	Do not power down or bypass.	VANA2_1P3	VANA2_1P0
2	CONVERTER_LDO	Power down only if all ADCs and DACs need to be powered down.	VCONV_1P3	VCONV_1P0
3	RX_1_LO_LDO	Power down if RX1 is not needed.	VRX1LO_1P3	VRX1LO_1P0
4	TX_1_LO_LDO	Power down if TX1 is not needed.	VTX1LO_1P3	VTX1LO_1P0
5	GP_LDO_2	Do not bypass.	VANA2_1P3	VANA2_1P0
6	RX_2_LO_LDO	Power down if RX2 is not needed.	VRX2LO_1P3	VRX2LO_1P0
7	TX_2_LO_LDO	Power down if TX2 is not needed.	VTX2LO_1P3	VTX2LO_1P0
8	CLK_PLL_SYNTH_LDO	Power down if the CLK_PLL_LP is in use	VCLKSYN_1P3	N/A
9	CLK_PLL_VCO_LDO	Power down if the CLK_PLL_LP is in use	VCLKVCO_1P3	VCLKVCO_1P0
10	CLK_PLL_LP_SYNTH_LDO	Power down if the CLK_PLL is in use	VCLKSYN_1P3	N/A
11	CLK_PLL_LP_VCO_LDO	Power down if the CLK_PLL is in use	VCLKVCO_1P3	VCLKVCO_1P0
12	LO1_PLL_SYNTH_LDO	Power down if LO1 is being externally supplied	VRFSYN1_1P3	N/A
13	LO1_PLL_VCO_LDO	Power down if LO1 is being externally supplied	VRFVCO1_1P3	VRFVCO1_1P0
14	LO2_PLL_SYNTH_LDO	Power down if LO2 is being externally supplied	VRFSYN2_1P3	N/A
15	LO2_PLL_VCO_LDO	Power down if LO2 is being externally supplied	VRFVCO2_1P3	VRFVCO2_1P0
16	AUX_PLL_SYNTH_LDO	Power down if the AUX_PLL is not required.	VAXUSYN_1P3	N/A
17	AUX_PLL_VCO_LDO	Power down if the AUX_PLL is not required.	VAUXVCO_1P3	VAUXVCO_1P0
18	SRAM_LDO	Do not power down or bypass.	VDIG_1P0	VDIG_0P9

The three different configurations, as shown in Figure 266, can be set up in this function by setting the LDO modes as per Table 111 and Table 112.

Table 112. Power saving configuration for each LDO

Index	LDO	Configuration 2	Configuration 1	Configuration 0
0	GP_LDO_1	1	1	1
1	DEV_CLK_LDO	1	1	1
2	CONVERTER_LDO	5	5	1
3	RX_1_LO_LDO	5	5	1
4	TX_1_LO_LDO	5	5	1
5	GP_LDO_2	1	1	1
6	RX_2_LO_LDO	5	5	1
7	TX_2_LO_LDO	5	5	1

8	CLK_PLL_SYNTH_LDO	1	1	1
9	CLK_PLL_VCO_LDO	1	1	1
10	CLK_PLL_LP_SYNTH_LDO	1	1	1
11	CLK_PLL_LP_VCO_LDO	1	1	1
12	LO1_PLL_SYNTH_LDO	1	1	1
13	LO1_PLL_VCO_LDO	2	1	1
14	LO2_PLL_SYNTH_LDO	2	1	1
15	LO2_PLL_VCO_LDO	2	1	1
16	AUX_PLL_SYNTH_LDO	1	1	1
17	AUX_PLL_VCO_LDO	1	1	1
18	SRAM_LDO	1	1	1

A device driver interface is implemented through the API function, adi_adrv9001_powermanagement_Configure, allowing the user to set the ldoPowerSavingsModes.

The user should configure the ldoPowerSavingModes struct in adi_adrv9001_PowerManagementSettings (GUI generated code sets all modes = 1) to achieve the different power saving configurations shown in Figure 266.

The default setup of the LDO array in the power management settings function is configuration 0, where all the LDOs are set to normal operation (mode 1) as seen here:

```
adi_adrv9001_PowerManagementSettings_t initialize_powerManagementSettings_35 = {

            .ldoPowerSavingModes = { ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1  }
```

For example in Configuration 2 of Figure 266 where some internal LDOs are bypassed and an External LO is supplied the struct would be set up as follows:

```
adi_adrv9001_PowerManagementSettings_t initialize_powerManagementSettings_35 = {

            .ldoPowerSavingModes = { ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_5,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_5, ADI_ADRV9001_LDO_POWER_SAVING_MODE_5,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_5,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_5, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_2,
```

ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1 }

In this case it is still assumed that 2T/2R channels are still being used. If it's the case that the user application only uses Tx1 and Rx1 then the LDOs for those channels can be set to a bypass or power down mode along with hardware modifications. There is a relationship between the hardware power layout and the LDO mode so attention is needed to correctly set the part to reflect the physical implementation. Attention is needed when bypassing LDOs to make sure that internal circuits are not supplied with incorrect voltage levels.

EXAMPLE:

TDD operation using Tx1, Rx1 and external LOs

The hardware power setup will look like Figure 267 where the pins for the Tx2 and Rx2 1v and 1.3v LO domains are tied to ground. The LDOs powering these RF blocks, VANA2_1p0 and VANA2_1p3, are also tied to ground because they are not needed.

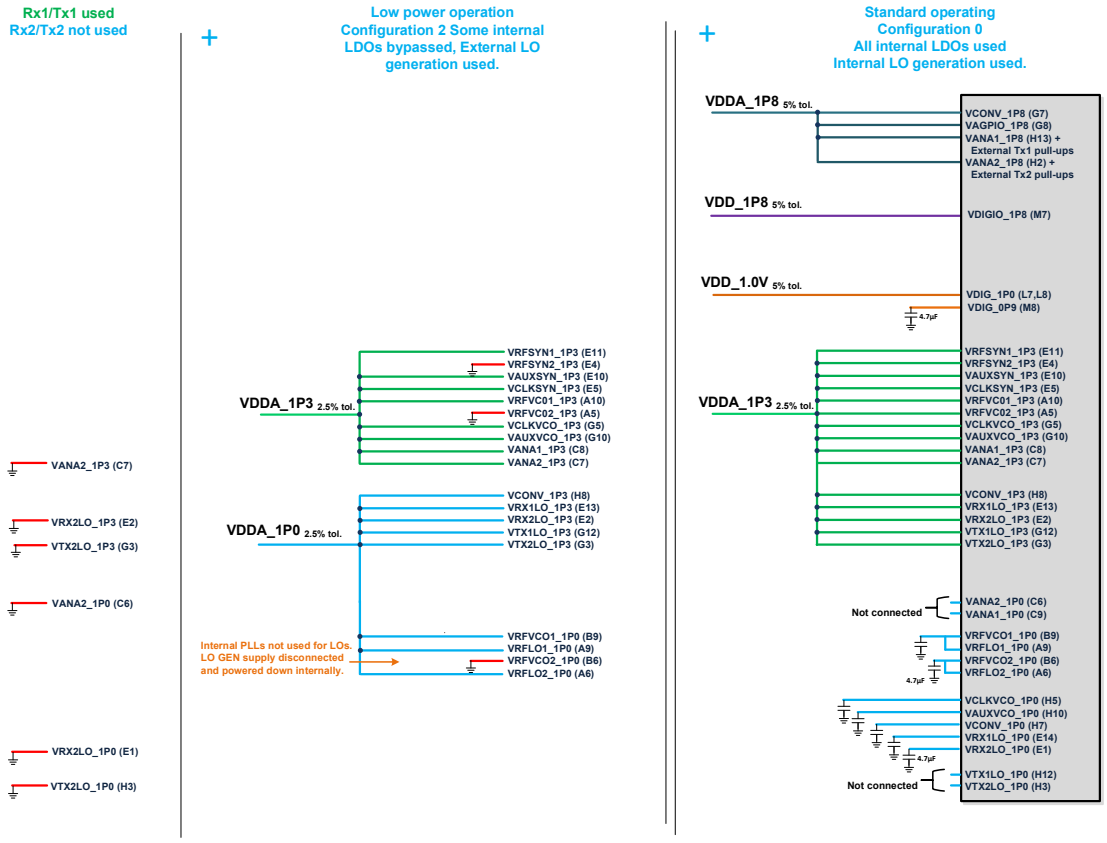


Figure 267. Standard Operating Config 0 & External LO Config & Rx1/Tx1 Power Supply Config

When the external LO config and Rx1/Tx1 config are super-positioned onto the standard operating configuration the power supplies are connected as per Figure 268.

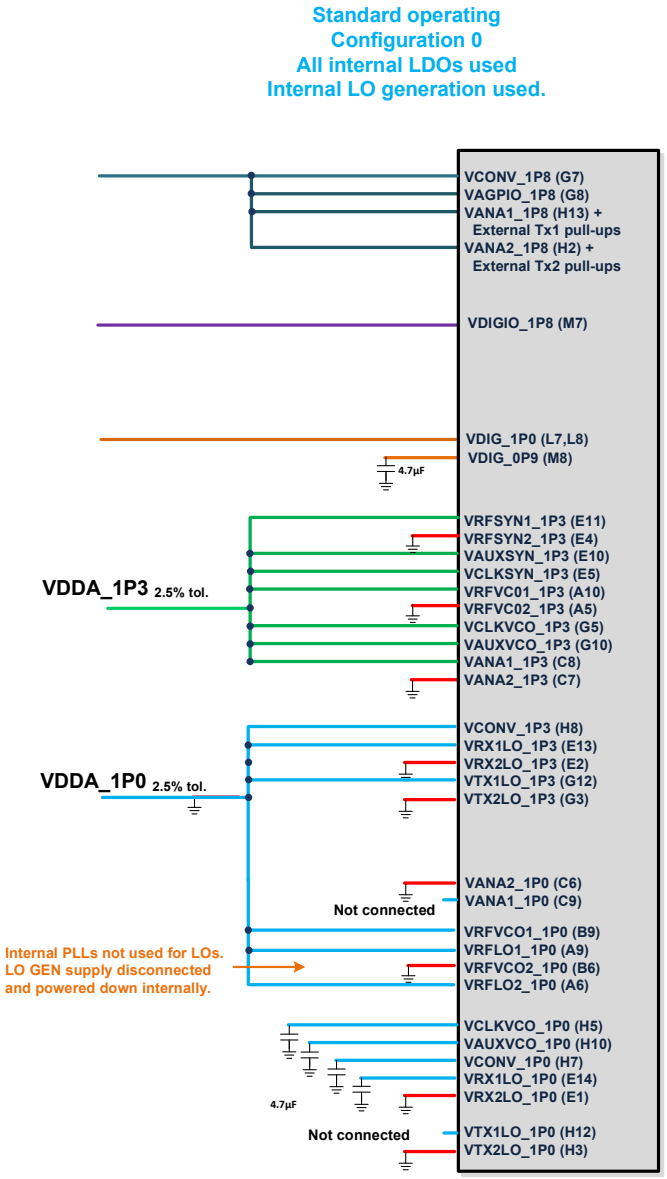


Figure 268. Tx1/Rx1 External LO Power Solution

Now that the hardware configuration is known the LDO configuration can be determined and the LDOs each set to the correct mode. In this case the following table outlines the mode needed for each of the LDOs to prevent overvoltage failures or other unwanted errors.

Table 113. Example LDO Modes and Pin Mapping

Index	LDO	LDO Mode	Output Pin	Output Pin Number	Comments
0	GP_LDO_1	1	VANA1_1P0	C9	Must be left on
1	DEV_CLK_LDO	1	N/A	N/A	Must be left on
2	CONVERTER_LDO	5	VCONV_1P0	H7	Bypassed due to external 1.0v
3	RX_1_LO_LDO	5	VRX1LO_1P0	E14	Bypassed due to external 1.0v
4	TX_1_LO_LDO	5	VTX1LO_1P0	H12	Bypassed due to external 1.0v
5	GP_LDO_2	2	VANA2_1P0	C6	Powered off, ch2 not required
6	RX_2_LO_LDO	2	VRX2LO_1P0	E1	Powered off, ch2 not required

7	TX_2_LO_LDO	2	VTX2LO_1P0	H3	Powered off, ch2 not required
8	CLK_PLL_SYNTH_LDO	1	N/A	N/A	Left on
9	CLK_PLL_VCO_LDO	1	VCLKVCO_1P0	H5	Left on
10	CLK_PLL_LP_SYNTH_LDO	1	N/A	N/A	Left on
11	CLK_PLL_LP_VCO_LDO	1	VCLKVCO_1P0	H5	Left on
12	LO1_PLL_SYNTH_LDO	2	N/A	N/A	Powered off, using external LO
13	LO1_PLL_VCO_LDO	2	VRFVCO1_1P0	B9	Powered off, using external LO
14	LO2_PLL_SYNTH_LDO	2	N/A	N/A	Powered off, LO2 not required
15	LO2_PLL_VCO_LDO	2	VRFVCO2_1P0	B6	Powered off, LO2 not required
16	AUX_PLL_SYNTH_LDO	1	N/A	N/A	Left on
17	AUX_PLL_VCO_LDO	1	VAUXVCO_1P0	H10	Left on
18	SRAM_LDO	1	VDIG_0P9	M8	Left on

The C# implementation of this configuration will be the following:

```
adi_adrv9001_PowerManagementSettings_t initialize_powerManagementSettings_35 = {

                .ldoPowerSavingModes = { ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_5,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_5, ADI_ADRV9001_LDO_POWER_SAVING_MODE_5,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_2,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_2,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_2,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_2, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1,
ADI_ADRV9001_LDO_POWER_SAVING_MODE_1, ADI_ADRV9001_LDO_POWER_SAVING_MODE_1  }
```

ADRV9001 EVALUATION SYSTEM

The ADRV9001 family demonstration system enables customers to evaluate the device without having to develop custom software or hardware. The system is comprised of a radio daughtercard, an Xilinx ZYNQ ZC706 motherboard* or a Xilinx ZCU102 motherboard, an SD card with operating system, a 12 V power supply for the ZYNQ ZC706 or ZCU102 that connects to a wall outlet, and a C#-based evaluation software application. The evaluation system uses an Ethernet interface to communicate with the PC.

INITIAL SETUP

The ADRV9001 transceiver evaluation software (TES) is the graphical user interface (GUI) to communicate with the evaluation platform. It can run with or without evaluation hardware connected. When TES runs without the hardware connected, it can be fully configured for a particular operating mode. If the evaluation hardware is connected, set up the desired operating parameters with TES and then the software can program the evaluation hardware. After the device is configured, the evaluation software can be used to transmit waveforms using custom waveform files as well as observe signals received on one of the receiver input ports. An initialization sequence in form of an IronPython script can be generated and executed using TES.

HARDWARE KIT

The ADRV9001 demonstration system kit contains:

- The customer evaluation (CE) board in form of a daughter card with FMC connector
- One (1) SD card containing image of Linux operating system with required evaluation software
- SD card type is 16 GB size, type 10

Requirements

The hardware and software require the following:

- The ADRV9001 demonstration system kit
- A Xilinx ZC706 ZYNQ (EK-Z7-ZC706) or Xilinx ZCU102 evaluation platform (Xilinx platforms not included in the ADRV9001 demonstration kit)*
- One (1) 12 V power supply for powering the Xilinx Platform.
- The operating system on the controlling PC must be Windows® 7 (×86 and ×64) or Windows 10 (×86 and ×64)
- The PC must have a free Ethernet port with the following constraints:
  - If the Ethernet port is occupied by another LAN connection, use a USB-to-Ethernet adapter
  - The PC should be able to access over this dedicated Ethernet connection the following ports:
    - 22—SSH protocol
    - 55557—access to the evaluation software on the Xilinx Platform
    - TES—contact your ADI representative to obtain access to this software
  - The user must have administrative privileges

SD Card Imaging

To image the SD card properly for use in either the Xilinx Platform you will need to download the ADRV9001 Disk Imaging Utility and the dotNet Disk Imager. These can be found on the Engineer Zone support forum for the TES GUI & Software Support. Follow the instructions and make sure to check if there is any encryption when writing to an SD card that will prevent the FPGA from reading the card.

Hardware Setup ZYNQ ZC706

Xilinx ZYNQ ZC706 platform setup requires the following steps:

1. All jumpers are in the positions shown in Figure 269.
2. SW11 is in position as shown in Figure 269(1, 2, 5 = A position)
3. The SD card included with the evaluation kit is placed in the J3 slot of the ZYNQ platform

The evaluation hardware setup is shown in Figure 270.

**All variants of the ZC706 eval kit should be suitable to demonstrate performance of the ADRV9001. Analog Devices does not guarantee that all future versions and derivatives of the ZC706 will work with the ADRV9001 daughter card and ADRV9001 TES to demonstrate ADRV9001 performance. However, as of the date of the current release of this User Guide, there are no known incompatibilities with any versions and derivatives of the ZC706. For example, the following versions have been successfully used for evaluation purposes: EK-Z7-ZC706-G, EVAL-TPG-ZYNQ3*

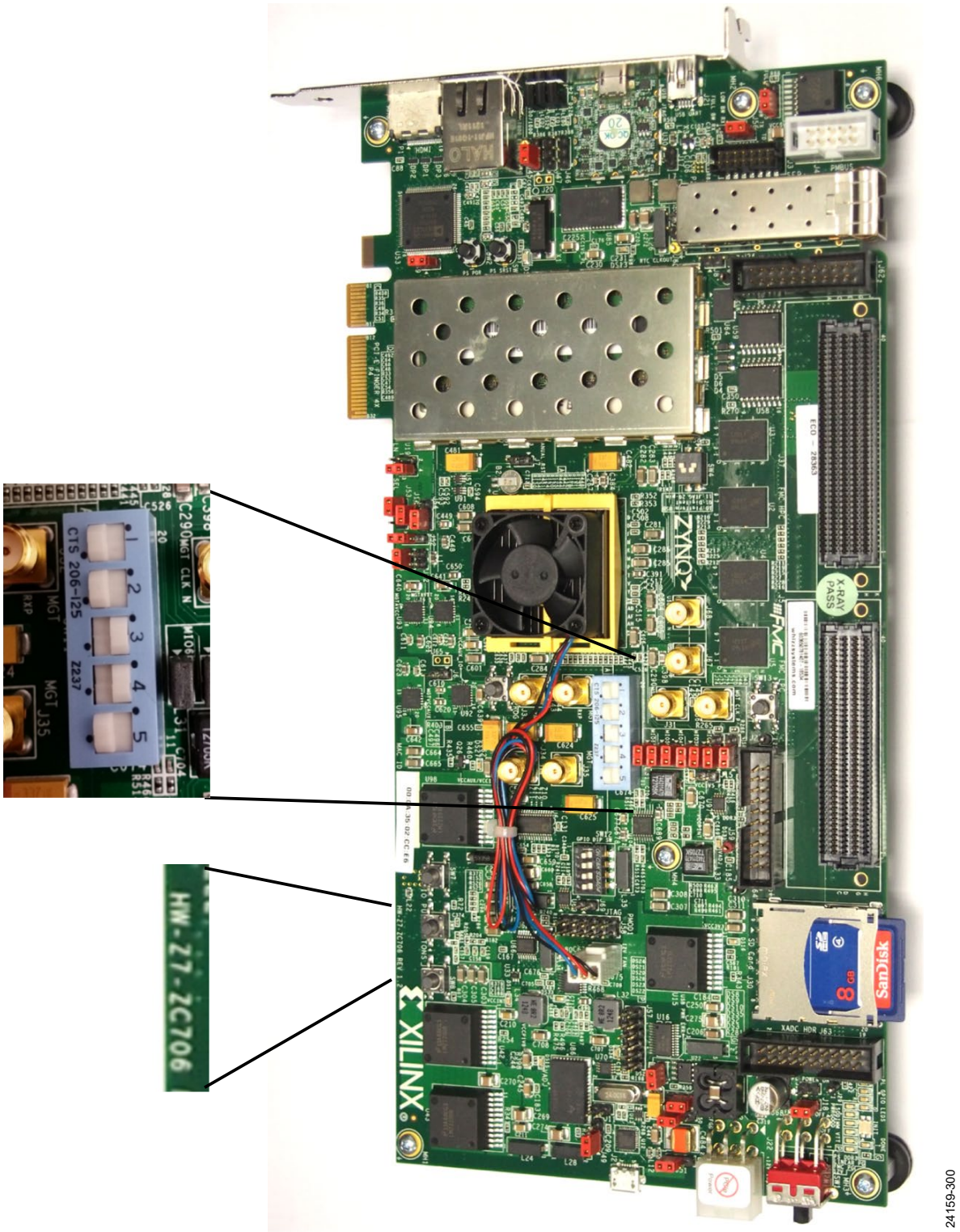


Figure 269. Xilinx Evaluation Board with Jumper Settings and Switch Position Configured to Work with ADRV9001 Evaluation Platform

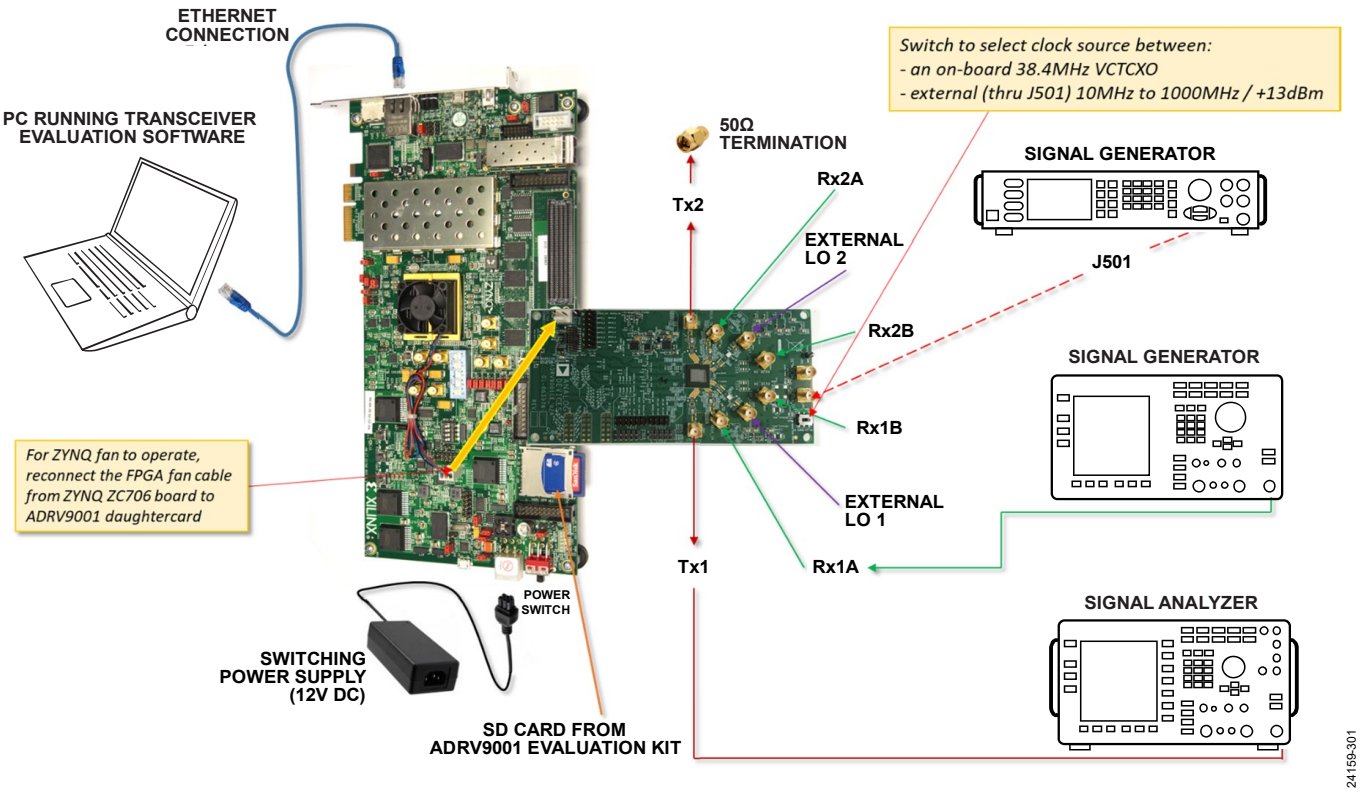


Figure 270. ADRV9001 Evaluation Card and ZYNQ ZC706 Evaluation Platform with Connections Required for Testing

To set up the evaluation board for testing, follow steps listed below:

1. Connect the ADRV9001 evaluation card and the ZYNQ ZC706 evaluation platform together as shown in Figure 270. Use the LPC FMC connector (J5). Take care to be sure the connectors are properly aligned.
2. Make sure that all jumpers on the ZYNQ ZC706 evaluation platform as well as the SW11 position (1, 2, 5 = "A" position) match settings shown in Figure 269.
3. Insert the SD card that came with the ADRV9001 evaluation kit into ZYNQ ZC706 evaluation platform SD card slot (J30).
4. On the ADRV9001 evaluation card, provide a device clock (frequency must match the setting selected in the TES), at a +13dBm power level to J501 connector. (This signal drives the reference clock into the ADCLK944 clock distribution chip on the board – the Q1/Q1_N pins of ADCLK944 generates the DEV_CLK for the ADRV9001 and REF_CLK for the Xilinx FPGA on the ZYNQ platform).
    a. It should be noted that quality of clock source used to generate DEV_CLK will directly impact overall system performance. User must ensure that high quality, stable and low phase noise clock source is used here.
5. Connect a 12V, 5A power supply to the ZYNQ evaluation platform at the J22 header.
6. Connect the ZYNQ evaluation platform to the PC with an Ethernet cable (connect to P3). There is no driver installation required.
    a. In the case when the Ethernet port is already occupied by another connection, use an USB-to-Ethernet adapter.
    b. On an Ethernet connection dedicated to the ZYNQ platform, the user must manually set the following:
        i. IPv4 Address to: 192.168.1.2
        ii. IPv4 Subnet Mask to: 255.255.255.0

Refer to Figure 271 for more details. The user should make sure that ports listed below are not blocked by firewall software on their PC:

- 22—SSH protocol
- 55557—access to the evaluation software on ZYNQ platform

Note that the ZYNQ ZC706 evaluation platform IP address is set by default to: 192.168.1.10.

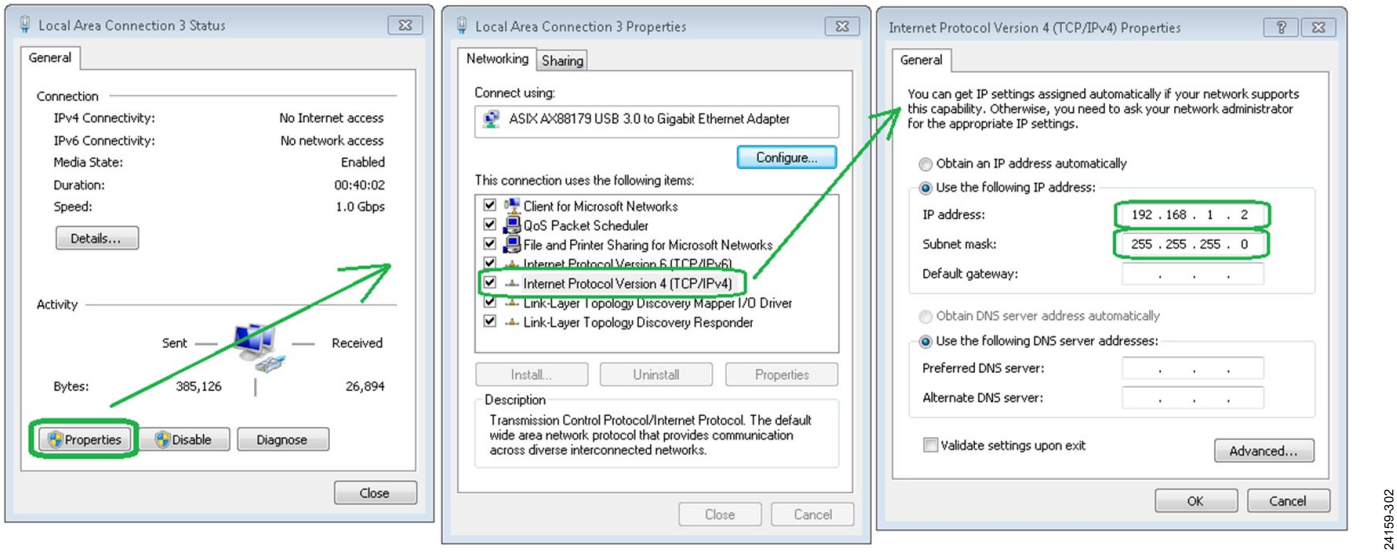


Figure 271. IP Settings for Ethernet Port Dedicated for ZYNQ ZC706 Evaluation Platform

Hardware Setup ZCU102

Xilinx ZCU102 platform setup requires the following steps:

1. All jumpers are in the positions shown in Figure 272.
2. SW6 is in position as shown in Figure 269 (2, 3, 4 = A position)
3. The SD card included with the evaluation kit is placed in the J100 slot of the ZYNQ platform

The evaluation hardware setup is shown in Figure 273.

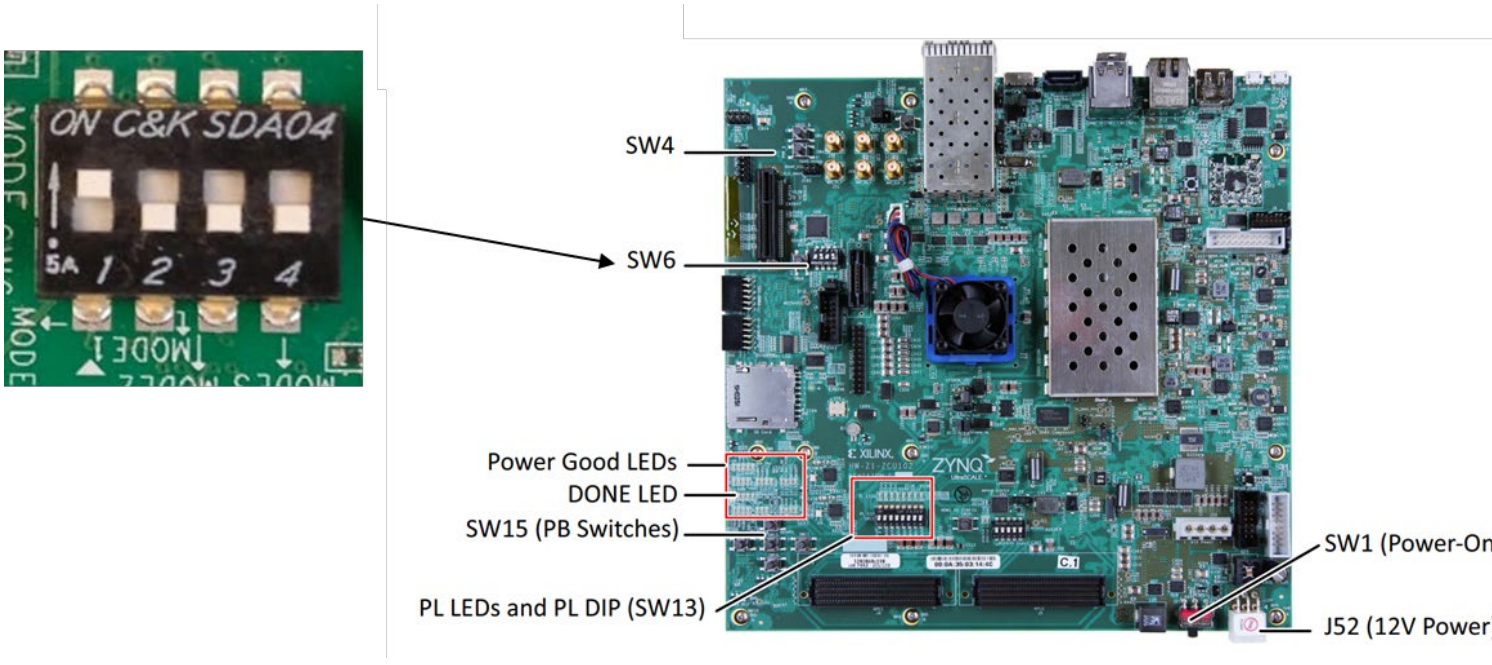


Figure 272. Xilinx ZCU102 Evaluation Board with Jumper Settings and Switch Position Configured to Work with ADRV9001 Evaluation Platform

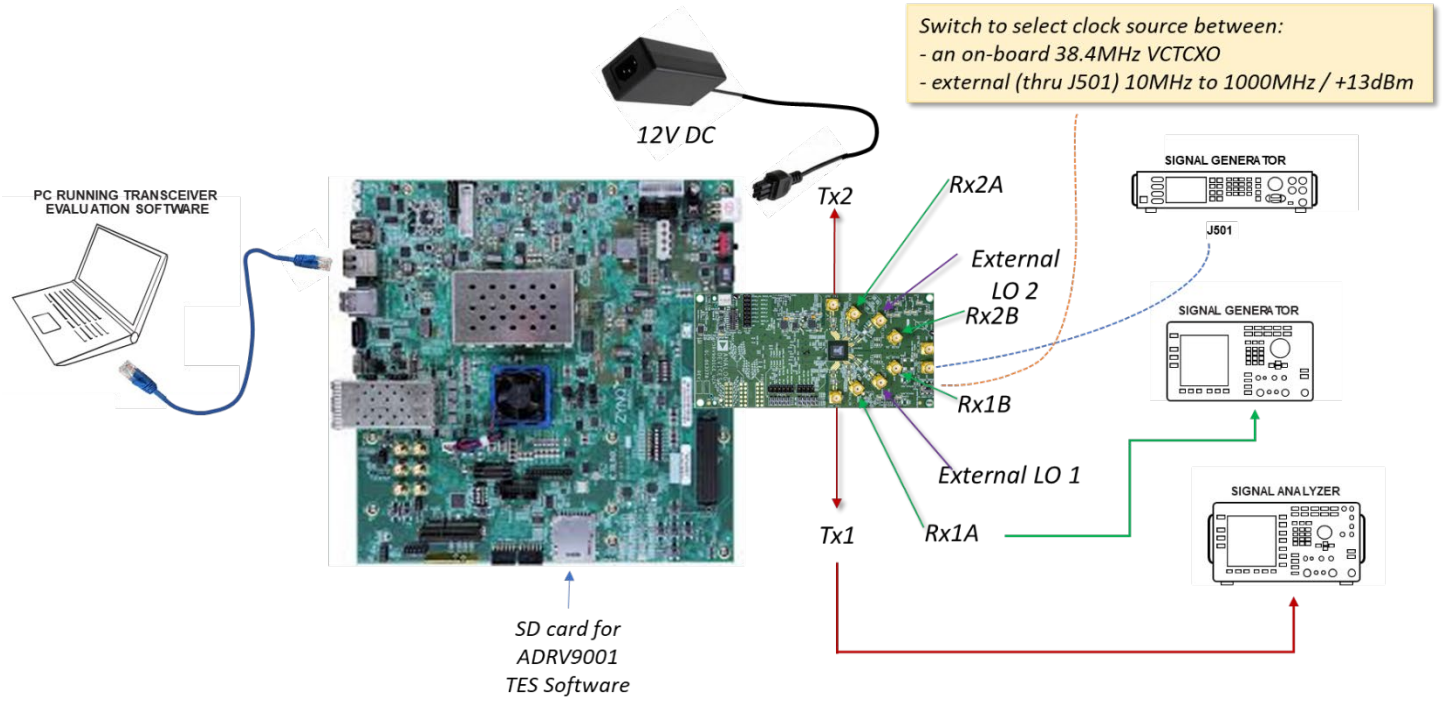


Figure 273. ADRV9001 Evaluation Card and ZCU102 Evaluation Platform with Connections Required for Testing

To set up the evaluation board for testing, follow steps listed below:

1. Connect the ADRV9001 evaluation card and the ZCU102 evaluation platform together as shown in Figure 273. Use the FMC connector (J5). Take care to be sure the connectors are properly aligned.
2. Make sure that all jumpers on the ZCU102 evaluation platform as well as the SW6 position (2, 3, 4 = "A" position) match settings shown in Figure 272.
3. Insert the SD card that came with the ADRV9001 evaluation kit into ZCU102 evaluation platform SD card slot (J100).
4. On the ADRV9001 evaluation card, provide a device clock (frequency must match the setting selected in the TES), at a +13dBm power level to J501 connector. (This signal drives the reference clock into the ADCLK944 clock distribution chip on the board – the Q1/Q1_N pins of ADCLK944 generates the DEV_CLK for the ADRV9001 and REF_CLK for the Xilinx FPGA).
    a. It should be noted that quality of clock source used to generate DEV_CLK will directly impact overall system performance. User must ensure that high quality, stable and low phase noise clock source is used here.
5. Connect a 12V, 5A power supply to the Xilinx Platform at the J52 header.
6. Connect the Xilinx Platform to the PC with an Ethernet cable (connect to P12). There is no driver installation required.
    a. In the case when the Ethernet port is already occupied by another connection, use an USB-to-Ethernet adapter.
    b. On an Ethernet connection dedicated to the Xilinx Platform, the user must manually set the following:
        i. IPv4 Address to: 192.168.1.2
        ii. IPv4 Subnet Mask to: 255.255.255.0

Refer to Figure 274 for more details. The user should make sure that ports listed below are not blocked by firewall software on their PC:

- 22—SSH protocol
- 55557—access to the evaluation software on Xilinx Platform

Note that the ZYNQ ZCU102 evaluation platform IP address is set by default to: 192.168.1.10.

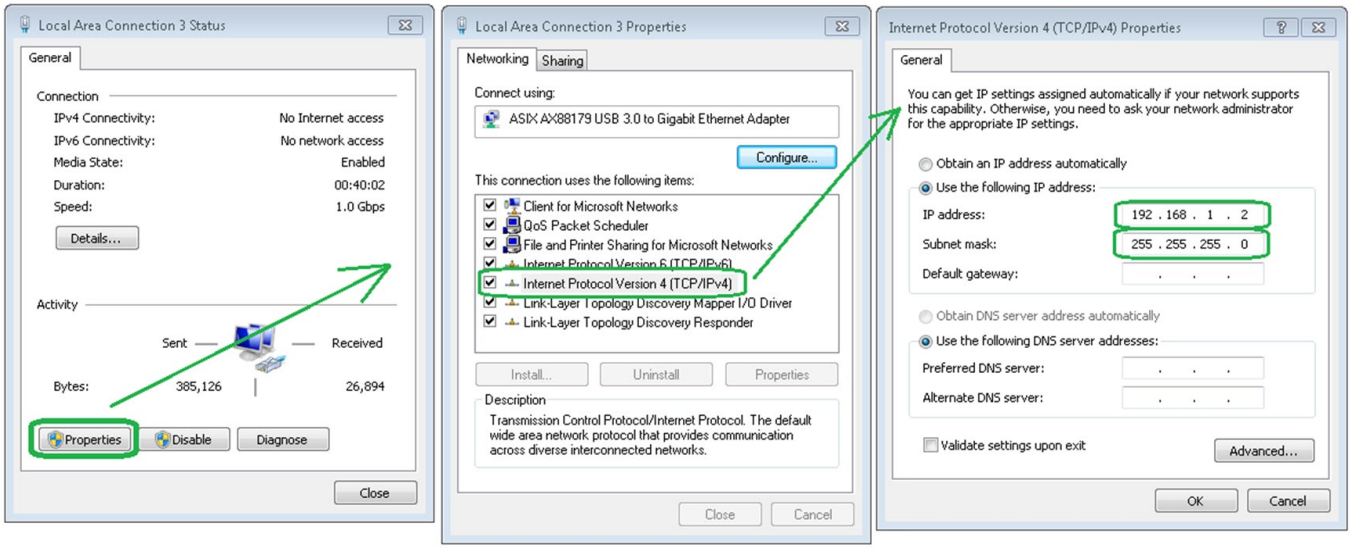


Figure 274. IP Settings for Ethernet Port Dedicated for ZCU102 Evaluation Platform

HARDWARE OPERATION

ZYNC ZC706

1. Turn on the evaluation system by switching the ZYNQ ZC706 evaluation platform power switch (SW1) to the on position. If hardware is connected correctly, two green LEDs (D801 and DS901) on the ADRV9001 evaluation card should be on.
2. The ZYNQ ZC706 evaluation platform uses a Linux operating system. It takes approximately 30 seconds before the system is ready for operation and can accept commands from PC software. Boot status can be observed on ZYNQ ZC706 evaluation platform GPIO LEDs (L, C, R, O). The correct sequence should follow the description below:
    a. After SW1 is turned on, all 4 LEDs should start flashing (moving single OFF light).
    b. On the ADRV9001 Evaluation board the System OK LED should also begin to flash.
    c. These LEDs will continue to flash in the same sequence through FPGA boot up, TES connection and TES program.
3. The reference clock signal (in range from 10 MHz to 1000MHz, CW tone, +13dBm max) should be connected to J501.
    a. It should be noted that quality of clock source used to generate DEV_CLK will directly impact overall system performance. User must ensure that high quality, stable and low phase noise clock source is used here.
4. For receiver testing on the ADRV9001 evaluation card, use a clean signal generator with low phase noise to provide an input signal to the selected Rx RF input. Use a shielded RG-58, 50Ω coaxial cable (1m or shorter) to connect the signal generator.
5. To set the input level near the Rx receiver's full scale, the generator level (for a single tone signal) should be set to approximately -15dBm. This level depends on the input frequency and the gain settings through the path.
    a. Note that there should be no input signal applied to the Rx input when performing an init calibration.
6. For transmitter testing, connect a spectrum analyser to either Tx output on the ADRV9001 evaluation card. Use a shielded RG-58, 50Ω coaxial cable (1m or shorter) to connect the spectrum analyser.
    a. Both Tx outputs should be terminated, either into spectrum analysers or into 50Ω if unused. This is because the initial calibrations will run on both channels and can take a long time to complete if a Tx channel is not correctly terminated.
7. Power off must be executed using TES software or the user must power down ZYNQ ZC706 evaluation platform using SW9 push button before the user powers off the evaluation system by switching SW1 to off position.

Important

The ADRV9001 evaluation system uses a Linux operating system. Linux requires time to boot up as well as soft shut down before hardware power off. The user is expected to use the software power off feature or press the SW9 button on the ZYNQ ZC706 evaluation platform before physically switching power off using SW1. If this advice is not followed, the file system on the SD card can get corrupted and the ADRV9001 evaluation system might stop operating.

Correct shutdown should be performed by executing one of these options:

- Selecting *"File" -> "Shutdown Zynq Platform"* in the TES.
- Pressing the SW9 push button on the ZYNQ platform

After a few seconds, when all 4 GPIO LEDs on the ZYNQ platform blink together, the user can safely power off the system using SW1 on the ZYNQ platform.

ZYNC ZCU102

1. Turn on the evaluation system by switching the ZCU102 evaluation platform power switch (SW1) to the on position. If hardware is connected correctly, two green LEDs (D801 and DS901) on the ADRV9001 evaluation card should be on.

8. The ZCU102 evaluation platform uses a Linux operating system. It takes approximately 30 seconds before the system is ready for operation and can accept commands from PC software. Boot status can be observed on ZCU102 evaluation platform GPIO LEDs (DS37 – DS40). The correct sequence should follow the description below:
    a. After SW1 is turned on, all 4 LEDs should start flashing (moving single OFF light).
    b. On the ADRV9001 Evaluation board the System OK LED should also begin to flash.
    c. These LEDs will continue to flash in the same sequence through FPGA boot up, TES connection and TES program.
    d. When boot is done 'Init' LED is green and 'Init Done' led is on in the Power and Status LEDs.

2. The ADRV9001 reference clock signal (in range from 10 MHz to 1000MHz, CW tone, +13dBm max) should be connected to J501.
    a. It should be noted that quality of clock source used to generate DEV_CLK will directly impact overall system performance. User must ensure that high quality, stable and low phase noise clock source is used here.

3. For receiver testing on the ADRV9001 evaluation card, use a clean signal generator with low phase noise to provide an input signal to the selected Rx RF input. Use a shielded RG-58, 50Ω coaxial cable (1m or shorter) to connect the signal generator.
4. To set the input level near the Rx receiver's full scale, the generator level (for a single tone signal) should be set to approximately -15dBm. This level depends on the input frequency and the gain settings through the path.
    a. Note that there should be no input signal applied to the Rx input when performing an init calibration.
5. For transmitter testing, connect a spectrum analyser to either Tx output on the ADRV9001 evaluation card. Use a shielded RG-58, 50Ω coaxial cable (1m or shorter) to connect the spectrum analyser.
    a. Both Tx outputs should be terminated, either into spectrum analysers or into 50Ω if unused. This is because the initial calibrations will run on both channels and can take a long time to complete if a Tx channel is not correctly terminated.
6. Power off must be executed using TES software before the user powers off the evaluation system by switching SW1 to off position.

Important

The ADRV9001 evaluation system uses a Linux operating system. Linux requires time to boot up as well as soft shut down before hardware power off. The user is expected to use the software power off feature from the TES before physically switching power off using SW1. If this advice is not followed, the file system on the SD card can get corrupted and the ADRV9001 evaluation system might stop operating.

TRANSCEIVER EVALUATION SOFTWARE (TES)

Installation and Configuration

Customers should contact an ADI representative to obtain access to TES. After the initial software download, copy the software to the target system and unzip the files (if not already unzipped). The downloaded zip container should have an executable file that installs the SDK and then the evaluation software.

Administrator privileges are not demanded by TES installer by default. However, if user intends to install TES to a folder that requires administrative privileges, then installation process must run with administrator privileges.

After running an executable file, a standard installation process follows. Figure 275 shows the recommended configuration. Microsoft .NET Framework 4.5 or newer is necessary for TES to operate. During installation process TES will look for Microsoft .NET Framework and if not available on PC it will try to download it from Microsoft server. When Microsoft .NET Framework installation is selected, installer will check and inform the operator if newer version is already installed. If so, it will skip .NET Framework installation.

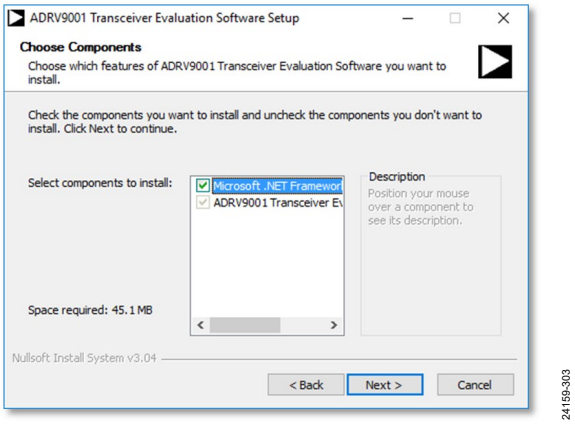


Figure 275. Software Installation Components

During installation process TES will ask user for Destination Folder where files should be installed, Figure 276. It is recommended to stay with the default location **C:\Program Files (x86)\Analog Devices\ADRV9002 Transceiver Evaluation Software** If this is not possible, TES can be installed into any other location that user have write access to it. The last step of the installation process is to select shortcut configuration.

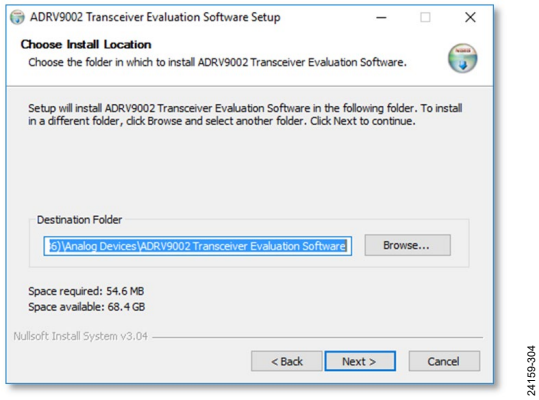


Figure 276. Software Installation Directory

Starting the Transceiver Evaluation Software

User can start the TES by clicking on Start -> ADRV9001 Transceiver Evaluation Software. Figure 277 shows the opening page of the TES after it is activated.

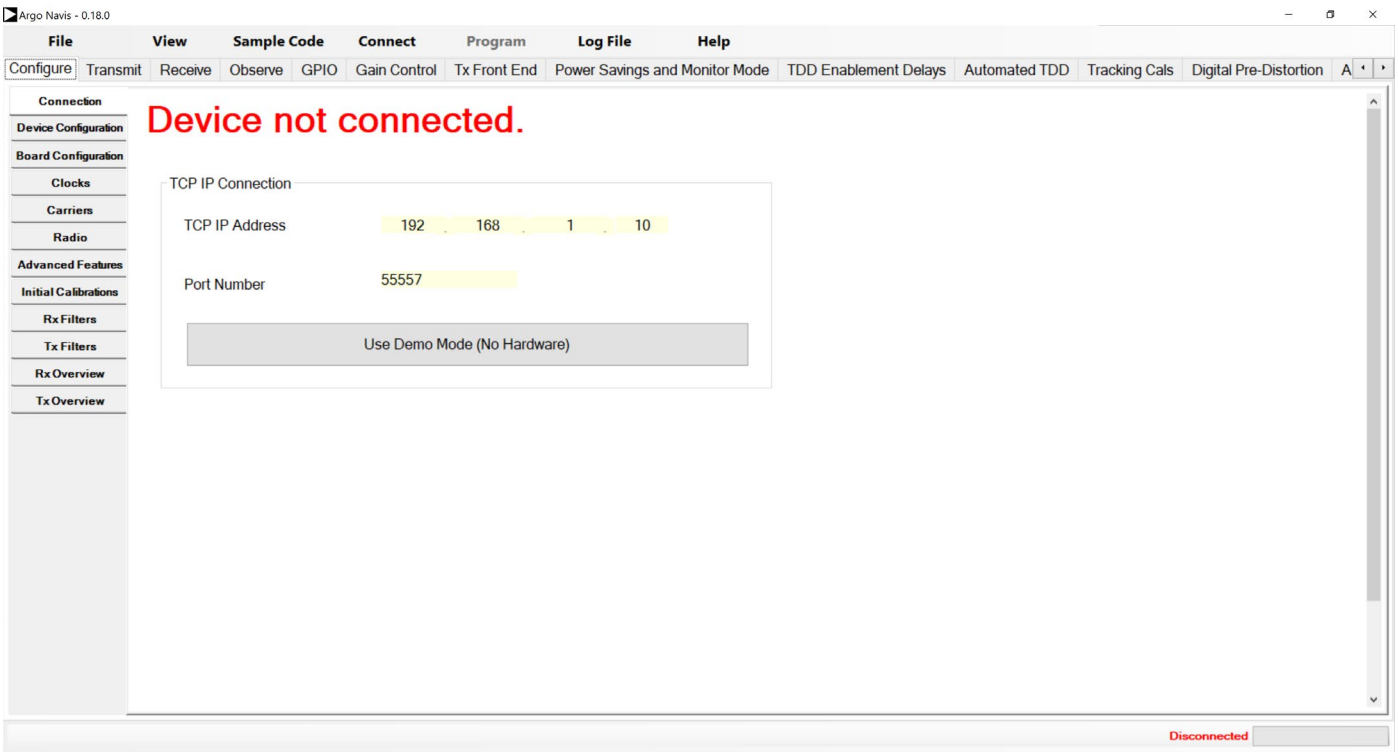


Figure 277. Main Interface of Transceiver Evaluation Software Bridge

When evaluation hardware is connected to a PC and the user wants to start using the complete evaluation system, TES will establish a connection with the Xilinx Platform system via Ethernet connection after clicking the Connect button. When proper connection is established, the user can configure an evaluation hardware. After selecting Connection tab, top part of that window shows the TCP IP address (default 192.168.1.10) and Port Number (default 55557), where bottom part of the window displays information about connected hardware and revisions of different software setup blocks. DHCP is enabled by default in Firmware version 0.14.5.5. When connecting the evaluation platform directly to a PC the default IP address mentioned above will work. If the platform is connected via router the IP address that is assigned will need to be identified and populated into the TES before connecting. Please contact the ADI Applications Engineering team if the ADRV9001 Evaluation System must operate over a remote connection and a different IP address for the Xilinx Platform is desired.

Figure 278 shows an example of correct connection between a PC and a Xilinx Platform with an ADRV9001 daughter card connected to it. In this window user can check used hardware version as well as all software components versions used by the system in current TES revision.

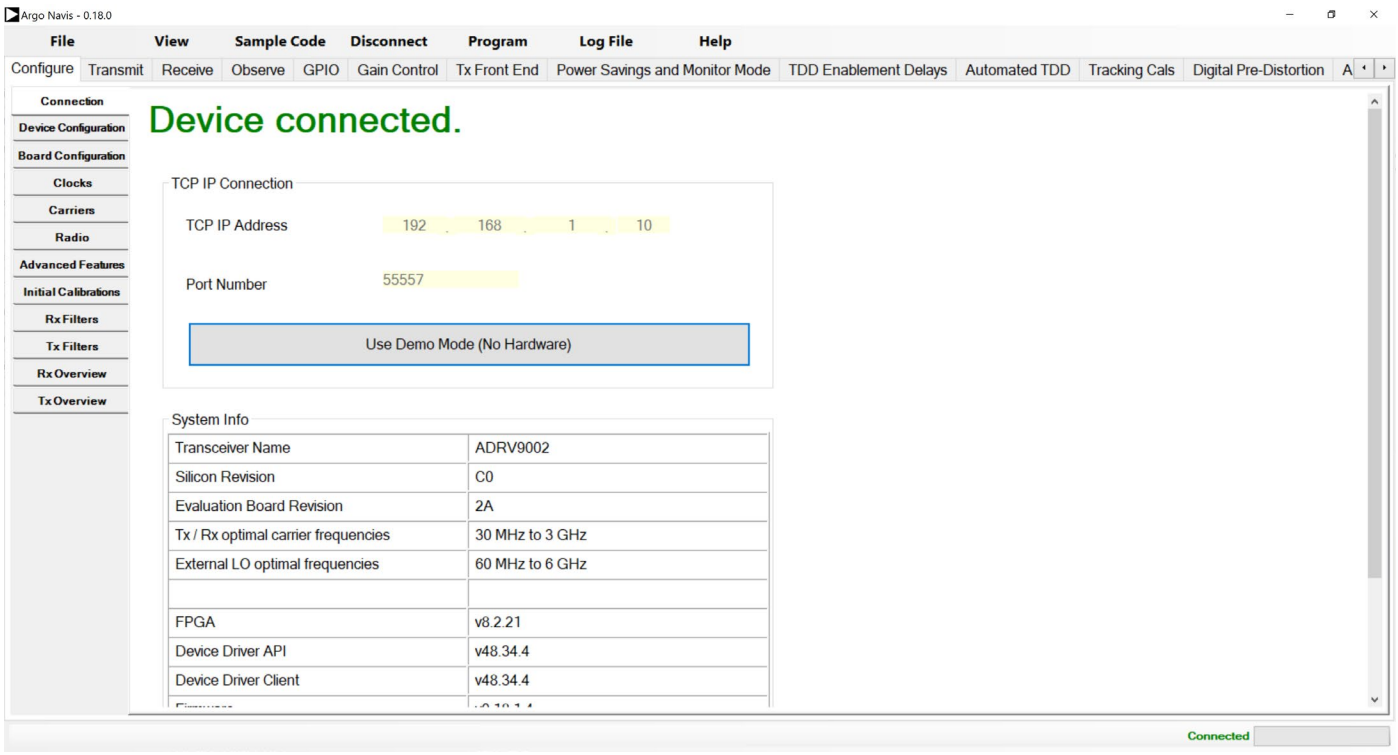


Figure 278. Setup Revision Information

Configuring the Device

Contained within the **Device Configuration** tab are setup options for the device. In this page the user can select the following:

- Product:
  - Supports ADRV9002, ADRV9003 and ADRV9004
- System
- TDD, FDD, TDM_FDD are supported
  - Under TDD
    - DMR setup is supported
    - Analog FM setup is supported
    - LTE setup is supported
    - Configuration 1, 3 and 4 setup is supported
    - Custom
    - Custom Extended (allows the System Clock to run as slow as 150 MHz)
  - Under FDD
    - Analog FM setup is supported
    - LTE setup is supported: This mode allows users to configure different channel configurations such as Rx2/Tx1.
    - Configuration 2 setup is supported
    - Custom
    - Custom Extended (allows the System Clock to run as slow as 150 MHz)
  - Under TDM_FDD
    - Tetra is supported
    - Custom configuration is supported
    - Custom Extended (allows the System Clock to run as slow as 150 MHz)
- SSI can be set to CMOS or LVDS
  - There is 1 or 4 lane options for CMOS
  - 2 lanes for LVDS
  - SSI Strobe can be set to long or short
- Signal type, this depends on the selected system and setup
  - RX supports I/Q and frequency deviation types and bit lengths

  - TX supports I/Q, I/Q FM/FSK, Direct FM/FSK types
- Frequency Deviation
  - This option is available only for TX FM type setups.
- ORx1 and ORx2 can be enabled for IQ input
- Interface Rate allows the user to select the rate for the interface, this can be used to over or under sample from the sample rate. The user may need to provide their own PFIR to account for this.

There is a traffic light indicator that checks the settings selected by the user and indicates if the settings are acceptable or not.

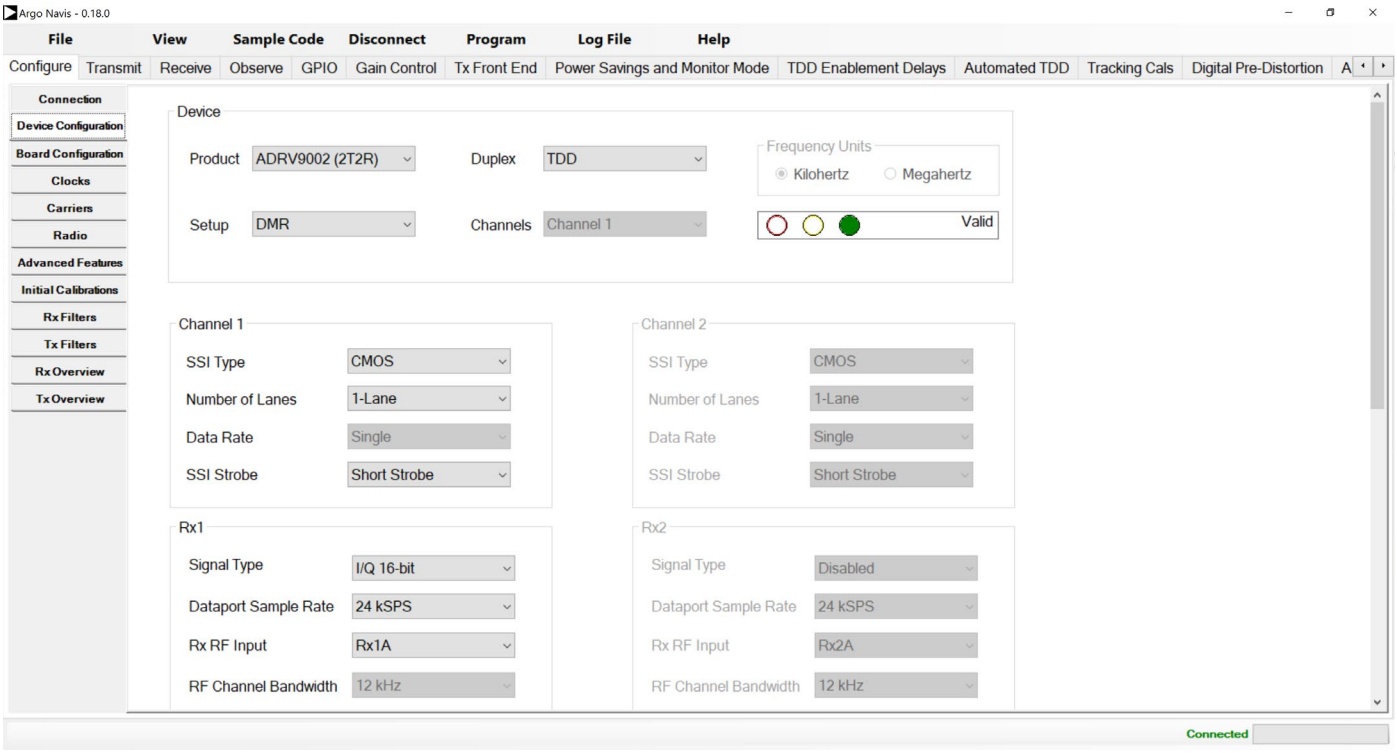


Figure 279. Device Configuration Tab

In the **Board Configuration** tab, there are settings for transmitter **External Loopback**. This is typically used for DPD type applications. The user can either enable and disable the external loopback after power amplifier. If it is enabled, the user should enter the expected loopback peak power in the **Peak Power** entry. Default peak power is −18 dBm. These are associated with RX1/2B ports.

The external loopback path delay can be measured using API, sending a low level wideband signal in the datapath for delay measurement. This action disrupts transmit signal in the air. User should do this in a test environment and before the power amplifier is transmitting real data. The user can use the ExternalPathDelay_Calibrate() and ExternalPathDelay_Get() to retrieve the external loopback path delay in ns. There is an IronPython example of this available in the SDK, for details see IronPython Scripting section below. Note that for this measurement there is a limitation with external delay measurement used by DPD.

1. Measurement with LTE10 profiles is recommended to obtain the highest possible measurement accuracy.
2. Customer should only make the measurement upon Navassa entering CALIBRATED state for the first time.

SSI Ref Clock can be obtained from the Tx channel or the Rx channel. When using the clock from the Tx channel this will be pushed to two GPIO pins. Using the Rx SSI clock releases these two pins for other uses.

External LNA is used to control the analog output from GPIOs to control the gain of the system LNA. The pin settings currently are locked to one nibble. Filling in the LNA gain steps will allow the API functions to calculate extra gain table settings. This can be seen in the Gain Control Tab after you have programmed the part.

The Gain Table will now be extended with gain settings that include the user defined LNA configuration. The Extended Gain Control column will specify the control word for the LNA and this is shown with colored backgrounds. The Duplicates column shows the row

number from the table that is copied and appended with the External Gain Control word to get the desired gain. Shown in Figure 281 the user can see that the top row is a copy of row 196 with just the external gain control changed.

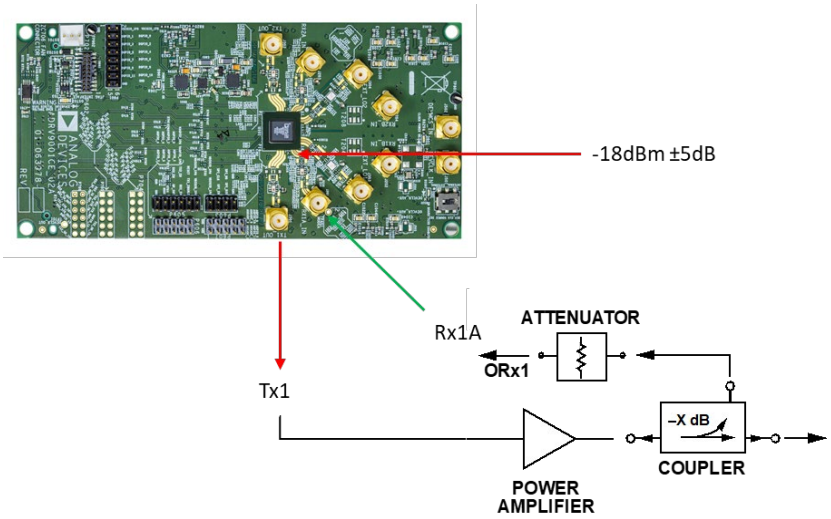


Figure 280. Receiver/Observation Receiver Loopback Diagram

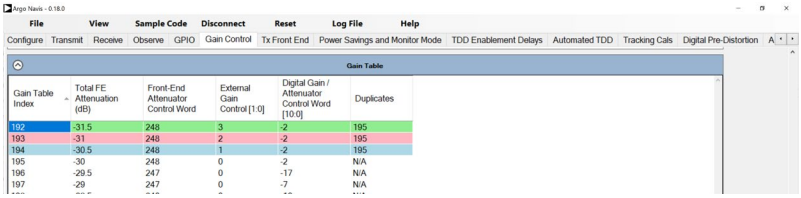


Figure 281. Extended Gain Table Example

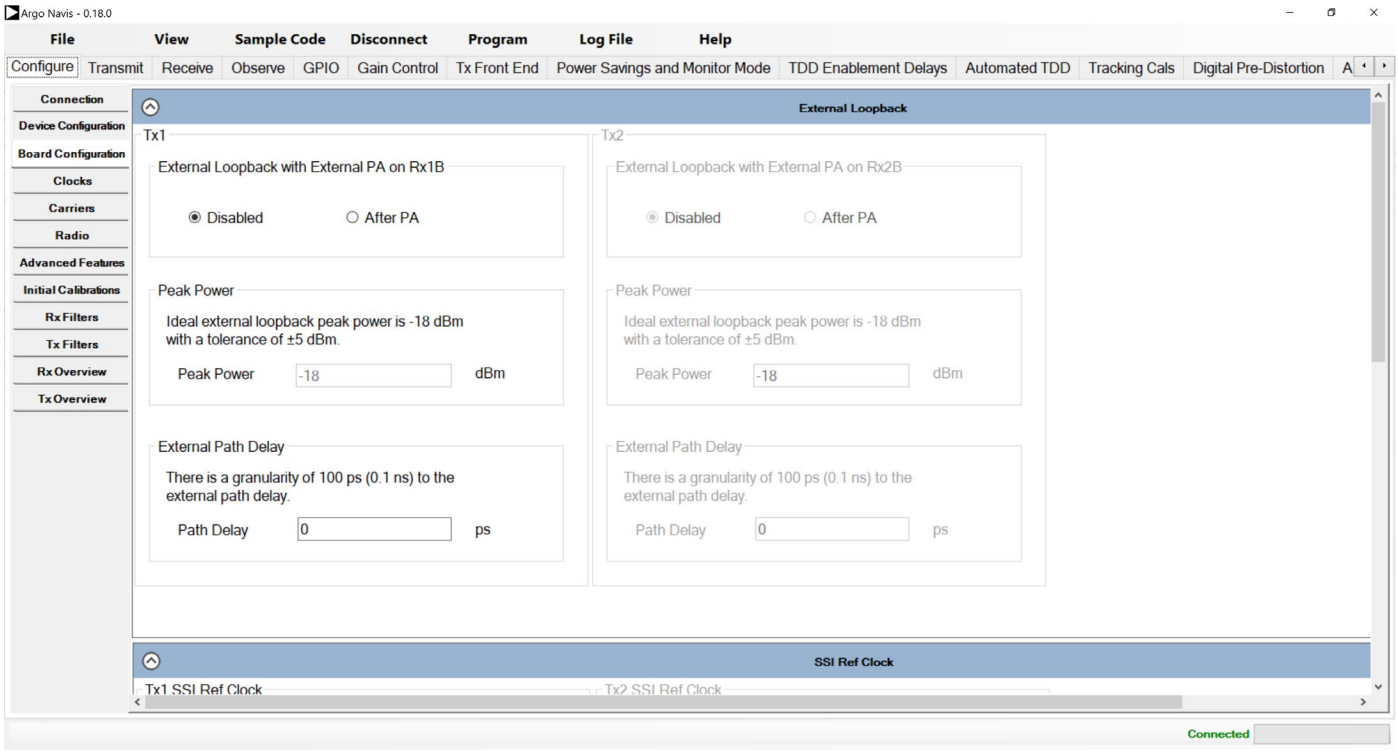


Figure 282. Board Configuration Tab

Clocks

The **Clocks** tab (Figure 283) provides access to the settings that determine device clock configuration. This page allows the user to:

- Set the device clock.
  - Set the device clock frequency.
  - Set the divisor value applied to the frequency at DEV_CLK_OUT.
  - Enable/disable the DEV_CLK_OUT signal.
  - Select the clock PLL type to be either high performance or low power (Note that LP PLL supports only certain sampling rates, see Clock Generation section above for limitations).
  - Select the Processor Clock Divisor value from 1, 2, and 4. Lower clock rate saves power. Changing the Processor Clock Divisor value will have effects on the whole system from changing the power up time to tracking calibration times.

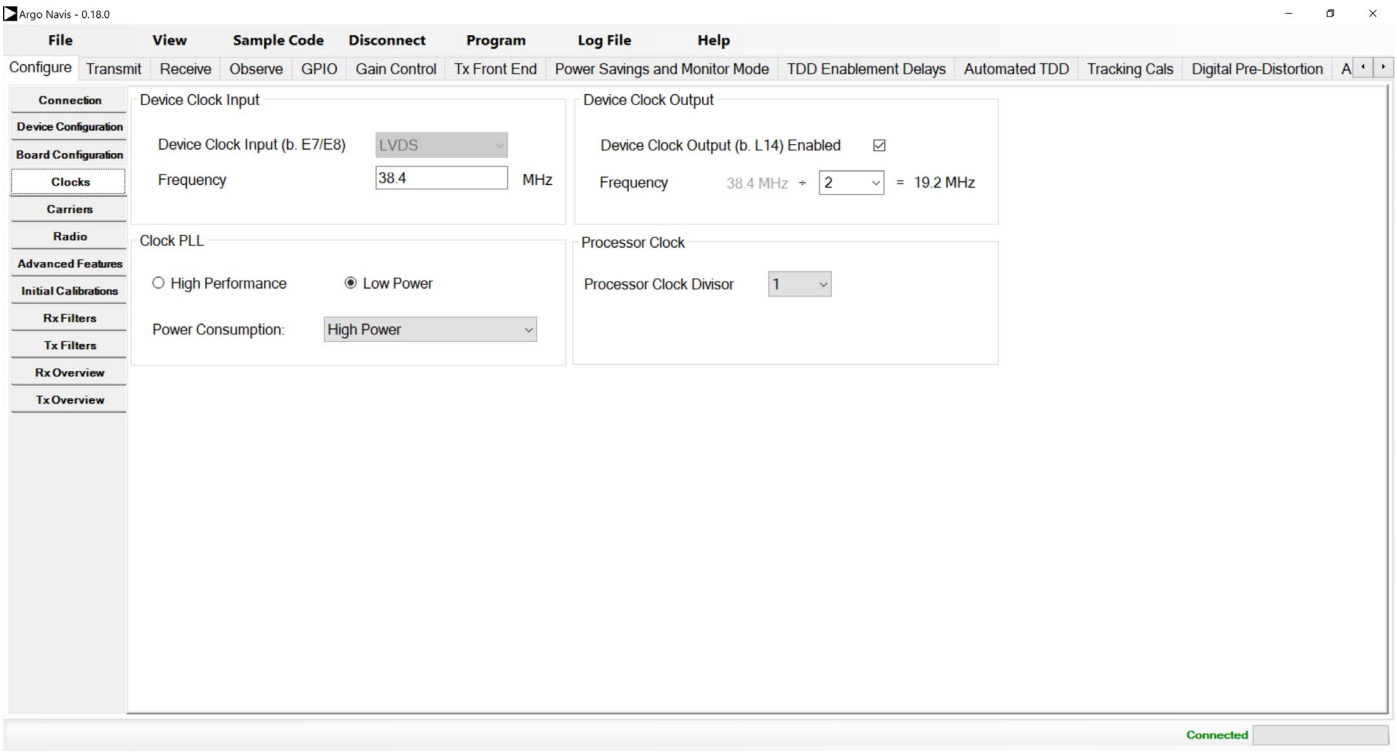


Figure 283. Clocks Configuration Tab

Carriers

The **Carriers** tab (see Figure 284) provides access to the settings that determine device LO configuration. This page allows the user to:

1. **RX1 Port Switching** allows the user to set up port switching depending on the Rx LO frequency. In this section the user can set the max and min values to set up the switching for both the Rx channels. Note that this prevents the use of ORx channels
2. **Define Carrier Frequencies** (default selection in the Carrier Configuration Mode drop down)

- Configure the LO
  - Set **PLL Retuning** to allow or disallow PLL retuning when switching between Tx and Rx
    - When Tx and Rx are using the same LO, but different frequency in the case of Low IF mode for example, when switching between Tx and Rx, PLL must be retuned to lock. If Tx and Rx are using different LOs, there is no need to do PLL retuning.
    - In some configuration modes, such as TDD LTE, an Antenna Diversity checkbox appears here. This checkbox can be used to change the routing of the LOs from Rx1/Tx1 and Rx/Tx2 to all on the same LO.
  - Set carrier frequency
  - Intermediate frequency is supported for RX. Enabled by ticking the NCO box. Recommended range from 490kHz to 20MHz.
  - Set Rx1/Rx2/Tx1/Tx2 carrier source (internal or external, options vary depending of selected setup).
  - If external LO is used then
    - Set the divisor value
    - TES informs the user about the external LO frequency that must be provided to the ADRV9001 transceiver at the External LO input.
  - If Internal LO is used, user has the option to select **Best Phase Noise** and **Best Power Saving** for their application. Note only Sub-1 GHz Tx frequencies are supported for **Best Phase Noise** option. This option changes some of the analog DC biases to reduce the VCO voltage swing and as a result the power consumption. The PLL calibration can be set to normal or fast mode and the PLL bandwidth can also be set from 50kHz to 1200kHz.

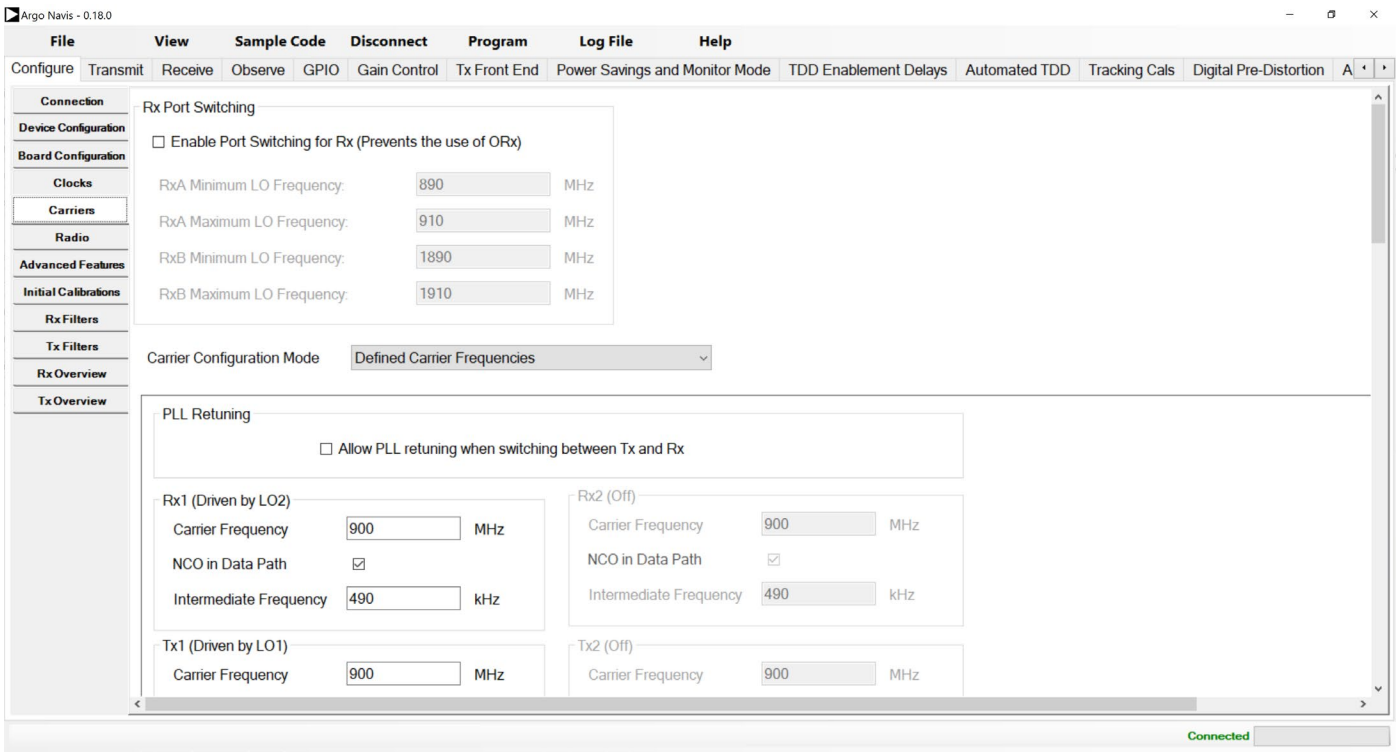


Figure 284. Carriers Configuration Tab

3. **Frequency Hopping** (selected from Carrier Configuration Mode drop down menu)
   - For details on using the Frequency Hopping settings in the TES see the **Frequency Hopping** section above.

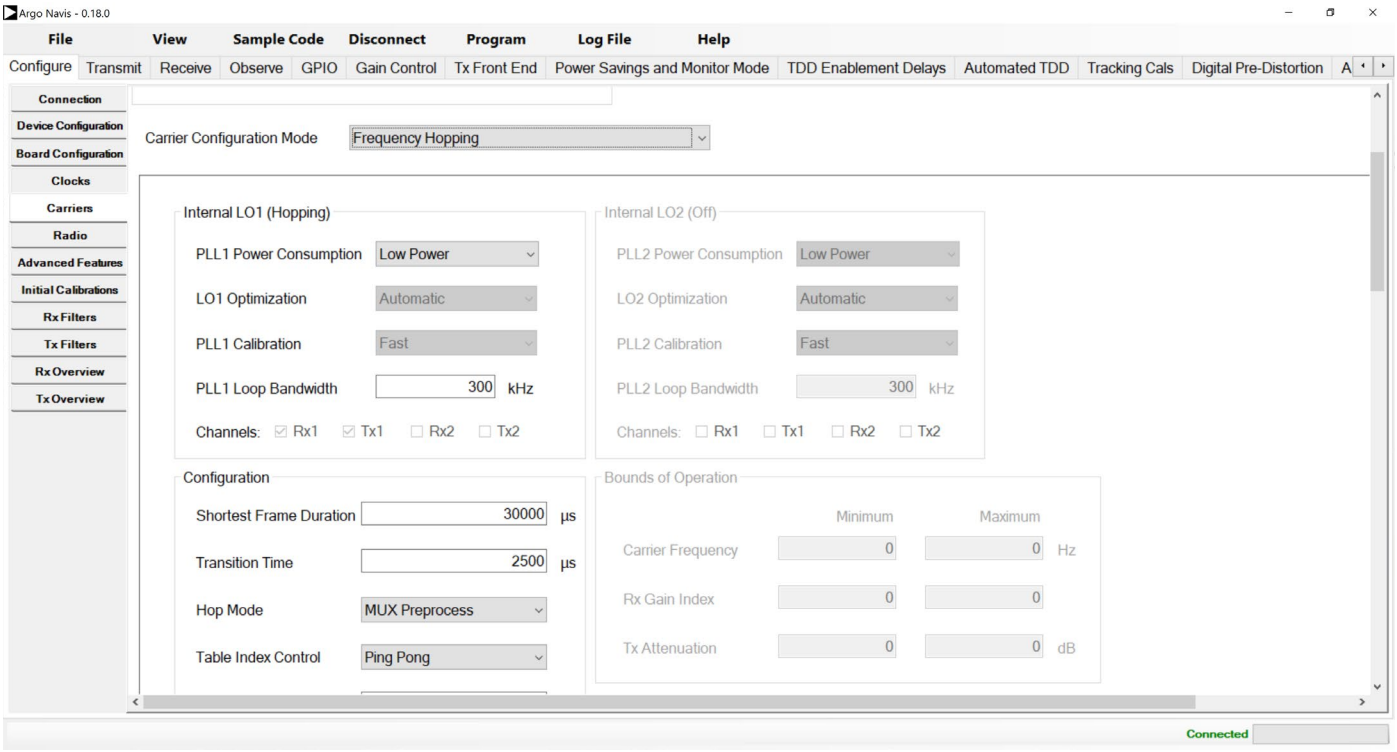


Figure 285. Carriers Configuration Tab (Frequency Hopping)

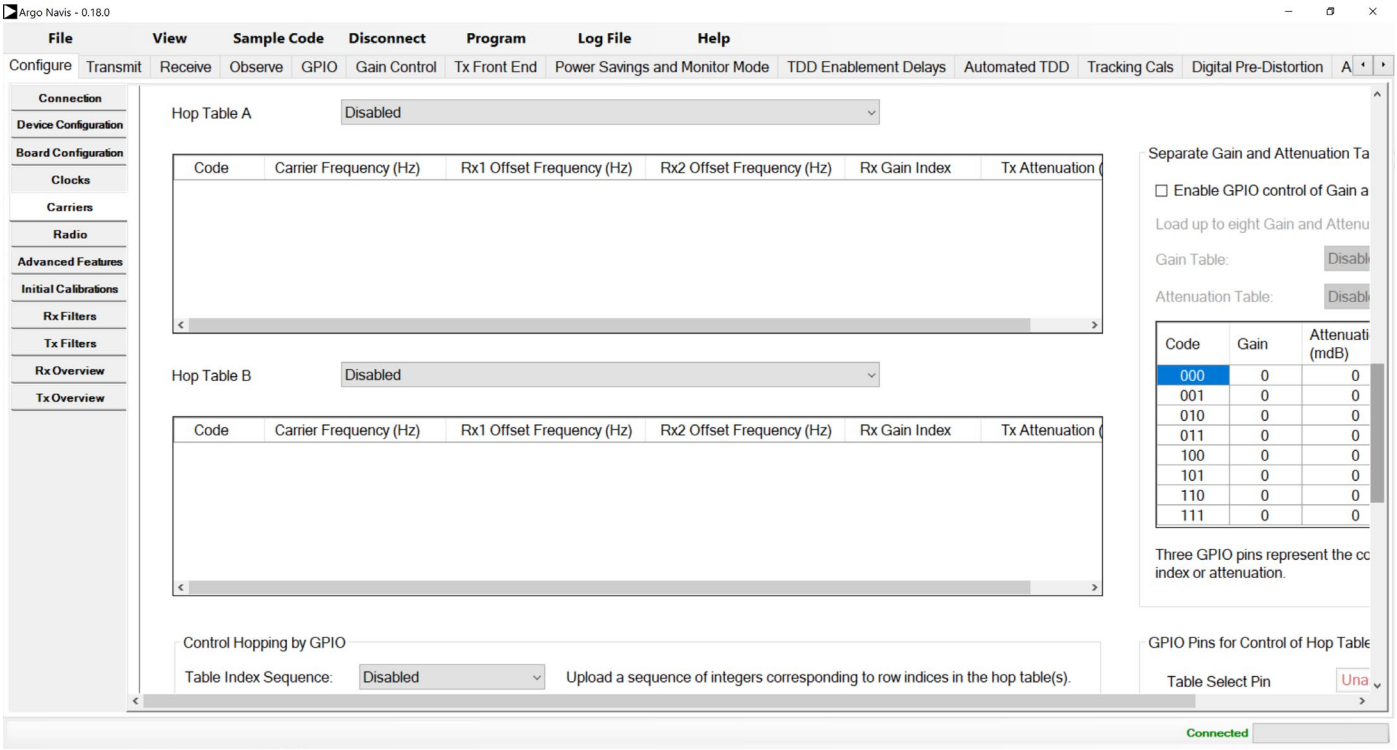


Figure 286. Carriers Configuration Tab (Frequency Hopping Tables)

Radio

The **Radio** tab (see Figure 287) lets the user configure the channel enablement, Tx and Rx characteristics.

- Select channel control mode (hardware enable signals or API command).
- Select **HIGH**, **MED**, or **LOW** receiver ADC rate.
- Select active Rx ADC from high performance or low power types
- Determine the Analog low-pass filter frequency response
- Select Rx frequency offset correction
- Select DAC, 3 dB boost mode
- Select Tx frequency offset correction
- Select LPF Power consumption for the Tx and Rx.
- Transmit Data Source can be used to send data from the FPGA or from the internal NCO internal signal source.

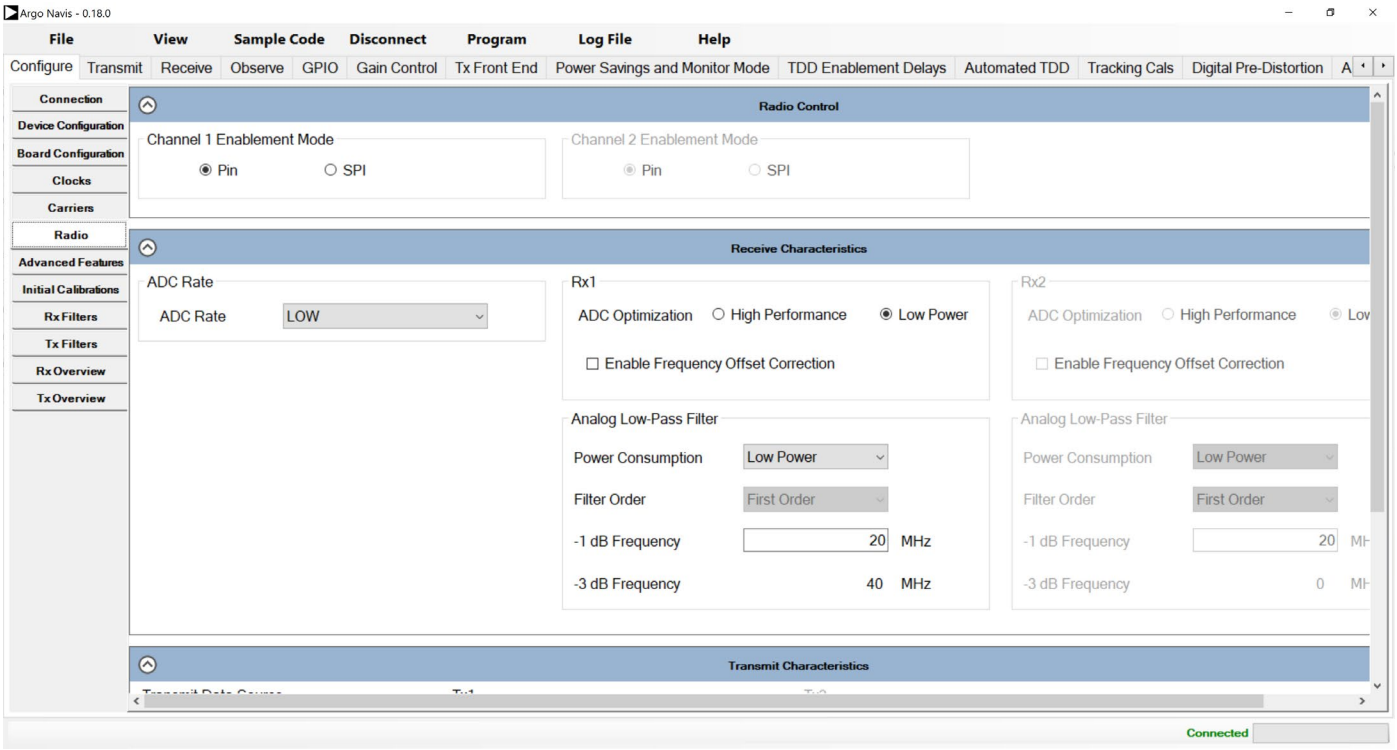


Figure 287. Radio Configuration Tab

Advanced Features

The **Advanced Features** tab (see Figure 288) provides access to the settings that determine device DPD configuration. This page allows the user to:

- Multi-Chip Sync
  - Select from Enabled and Enabled with RF PLL Phase Sync options
  - Sample and Read delays for each of the channels can be input here. For details on this refer to the Multichip Synchronization section.
- SSI Power Down
  - Allows the user to power down either of the two SSI channels
- Loopback
  - Internal loopback from the Tx SSI or Datapath to the Rx SSI or Datapath.
- Enable or disable DPD
  - Select DPD tap polynomial terms
    - A default configuration is provided
    - The user has the freedom to configure individual tabs
      - Enable/disable Rx and Tx initialization calibrations
- Enable or Disable CLGC (Closed Loop Gain Control)
  - This enable the CLGC setting in the Digital Pre-Distortion Tab
- Monitor Mode RSSI Configuration
  - Set the measurement parameters for the Monitor Mode including:
    - Number of measurements to average
    - Measurement duration
    - Start period
    - Detection threshold
  - Sync Fast Buffer Read:
    - In monitor mode when the device has detected a signal it waits for the user to bring Rx_EN pin high and it will increase the interface rate X4 to push the data out of the buffer until the input matches the output and it will revert to normal rate.
    - Device configuration needs to be in DMR and have Frequency Deviation enabled and the ADCs in High Performance operation.
- Stream Status Output over GPIO
  - This can enable the stream status to be seen on the GPIOs to be able to measure the rise-to-analog-on time

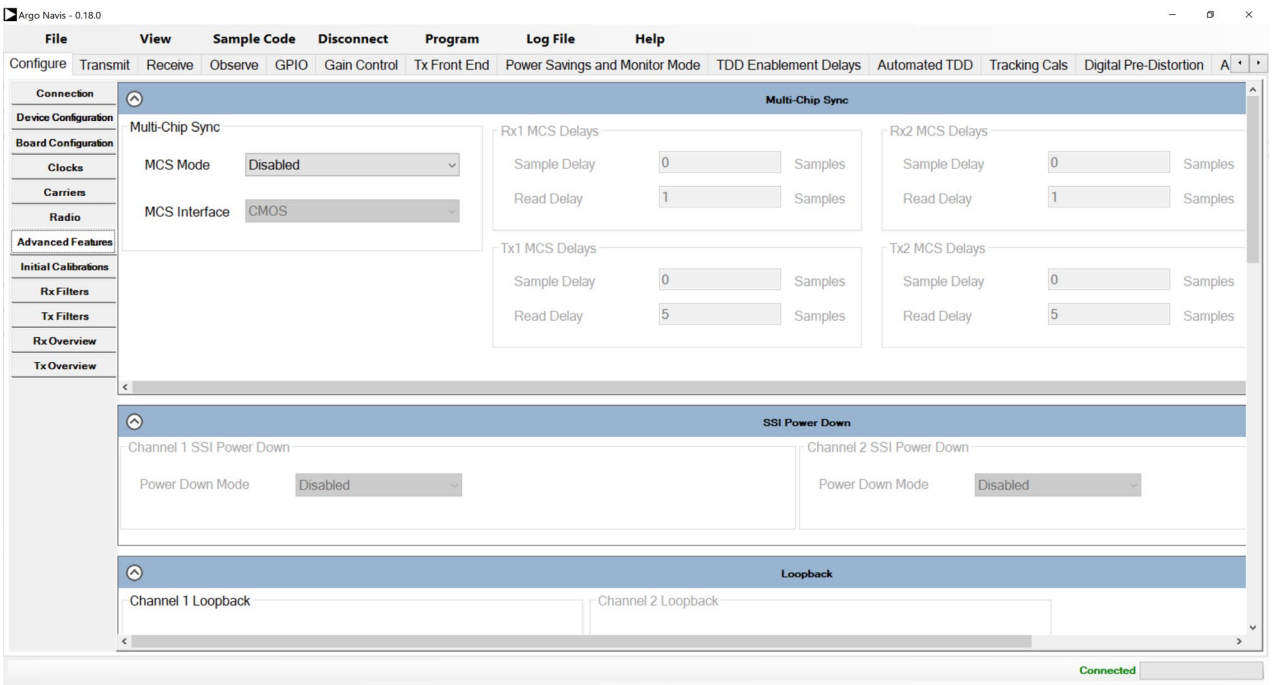


Figure 288. Advanced Features Tab

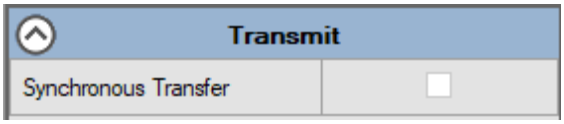


Figure 289 Synchronous Transfer option in the Transmit Tab

Note the evaluation software has synchronous transfer option provided for the user. This option allows the user to begin transfer of data (through the FPGA DMA) on both channels (Tx1&Tx2 or Rx1&Rx2) at the same point in time. Its purpose is for testing MCS to ensure that both channels have the same sample at the same point in time. The trigger source for DMA on both channels is set to adi_fpga9001_DmaTrigger_e.ADI_FPGA9001_DMA_TRIGGER_SYNC. FPGA generates an pulse (using the FPGA generated MCS signal) which triggers simultaneous transfer on both ports.

Initial Calibrations

The "Initial Calibrations" (Figure 290) shows a list of the calibrations that will be run during the initialization of the part. The user can unselect some of the calibrations for measurement comparisons, however it is recommended to use all the calibrations in a real use case.

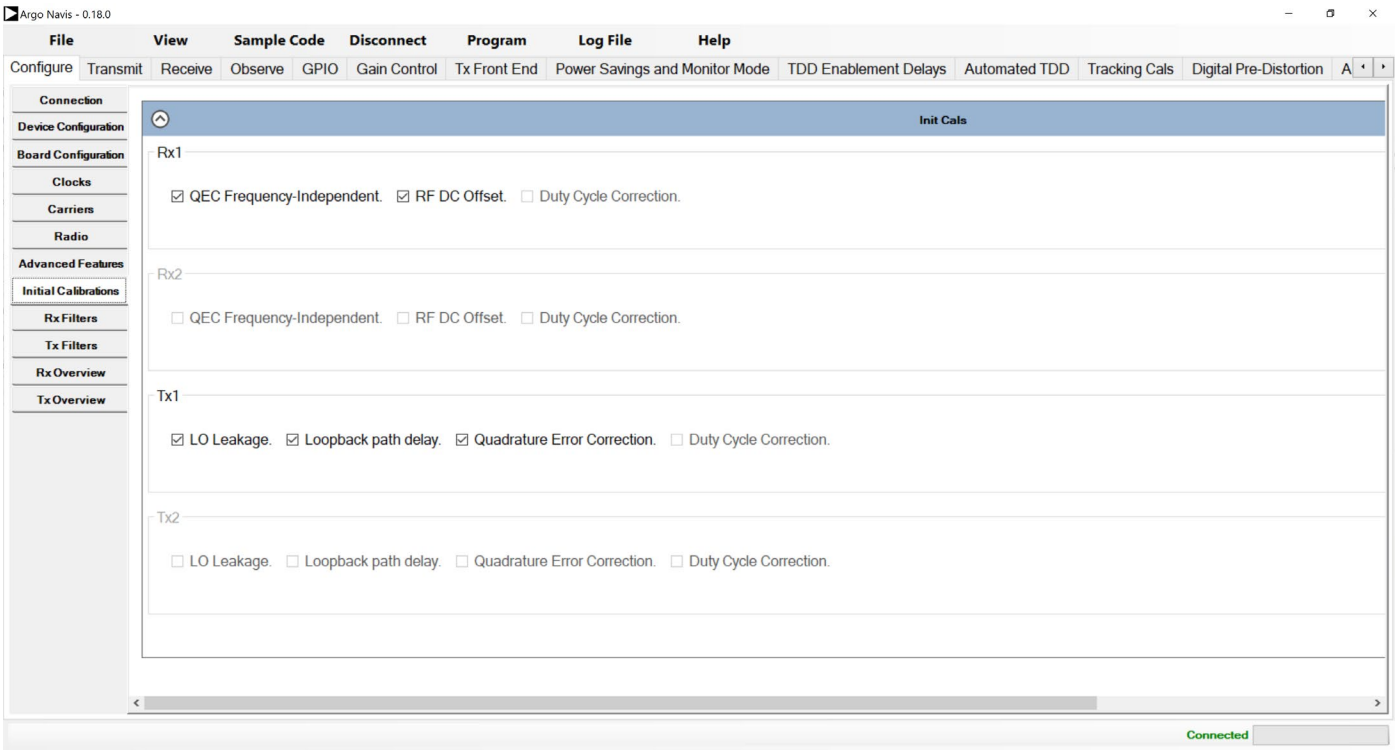


Figure 290. Initial Calibrations

Rx and Tx Filters

The "Rx Filters" (Figure 291) and "Tx Filters" (Figure 292 and Figure 294) tabs allow the user to input programmable FIR filters on the chose channels. In the Tx Filters tab the user can set the interpolations factor.

The RX Filer tab also shows controls for the Dynamic Profile Switching (currently only available on LTE profiles). For more details on this see Dynamic Profile Switching section above.

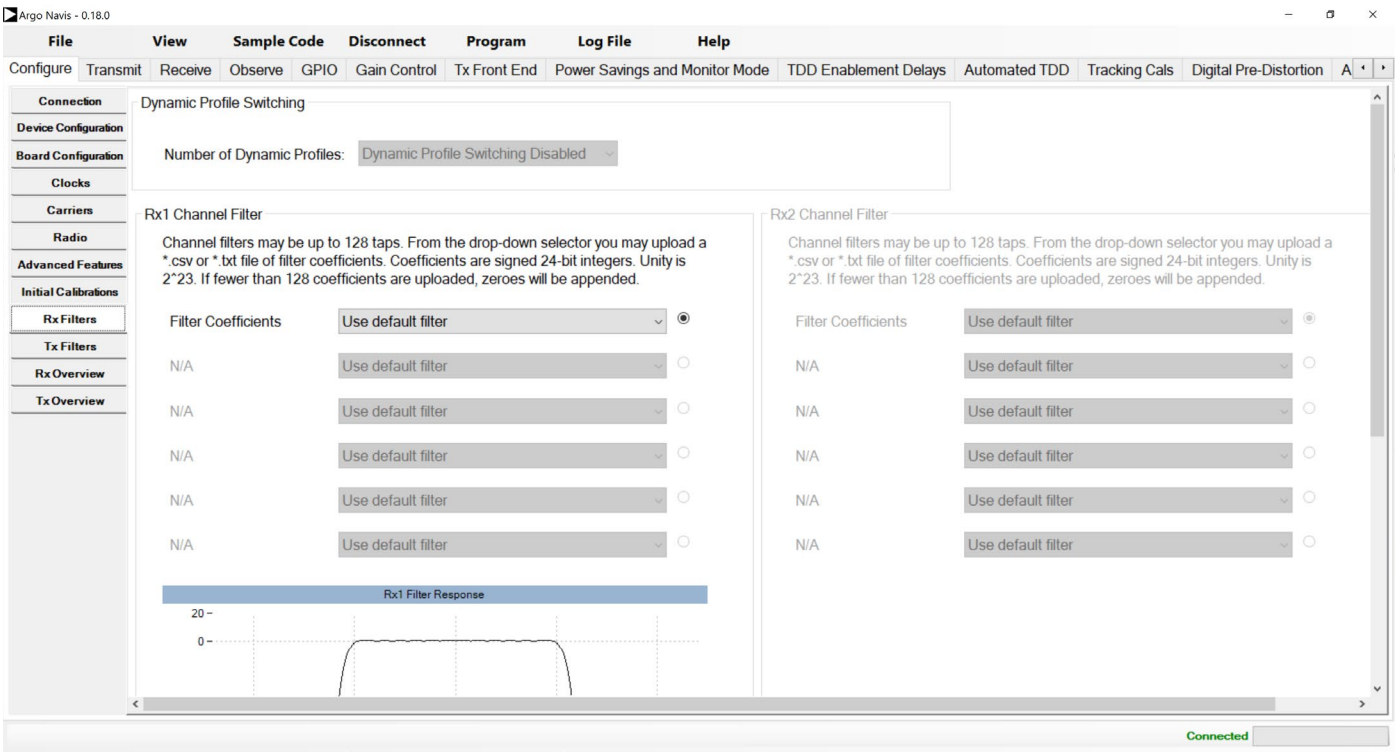


Figure 291. Rx Filters

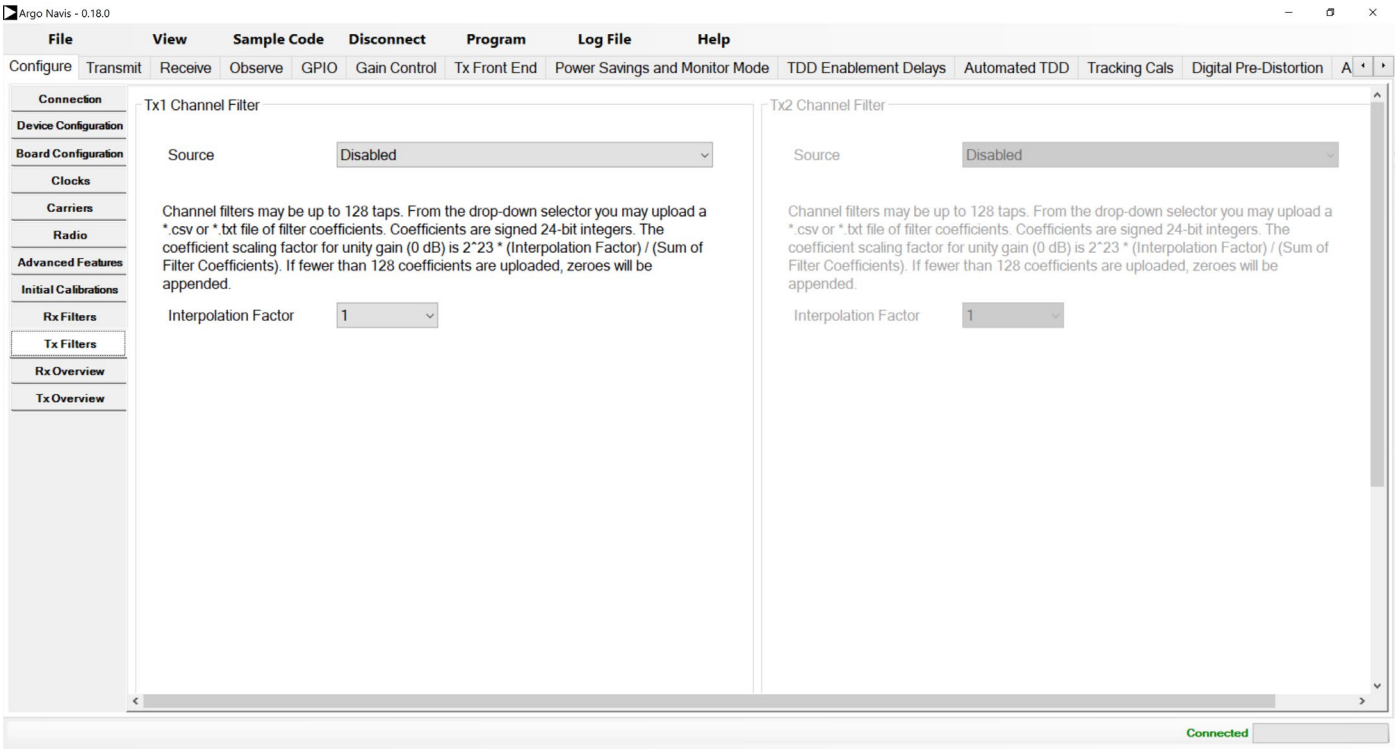


Figure 292. Tx Filters

Rx and Tx Overview

The “Rx Overview” (Figure 293) and “Tx Overview” (Figure 294) tabs aim to provide more detail on ADRV9001 selected mode of operation using “Device Configuration” tab (Figure 279). The Rx and Tx datapath overview diagrams are provided in each tab. These tabs provide user with read back of ADC/DAC sampling frequencies, analog filtering configuration, datapath sampling rate, data port format, mode of operation and sampling rate.

In “Rx Overview” tab user can also read back IF frequency and observe pFIR channel filtering characteristics and their passband flatness. Quick zooming capability allows zooming of the passband response using the mouse cursor as well as restoring to the full-scale plot. The TES also provides capability to export the data plotted on the graphs to an external file. This is done by right-clicking on the graph area and selecting option “Export Data to File”. Data can then be saved to file for later analyses.

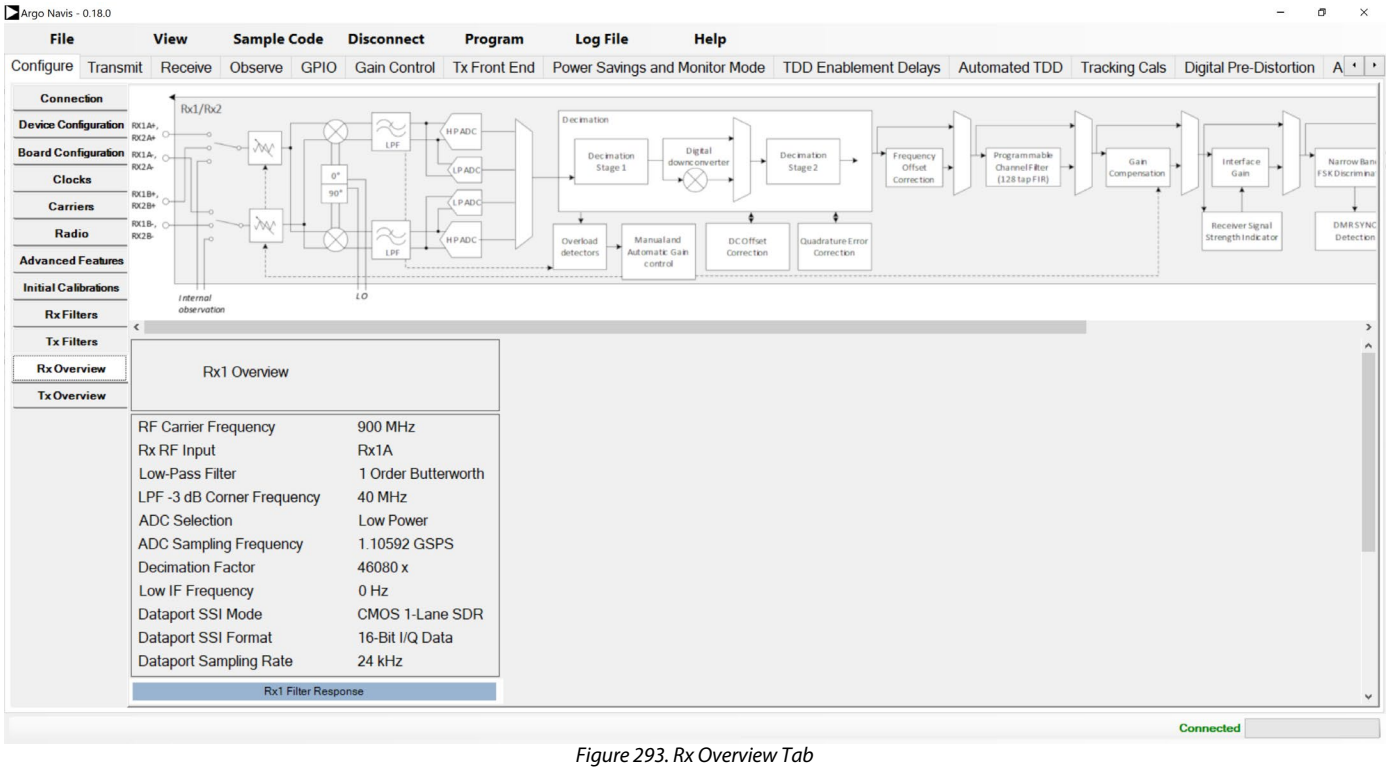


Figure 293. Rx Overview Tab

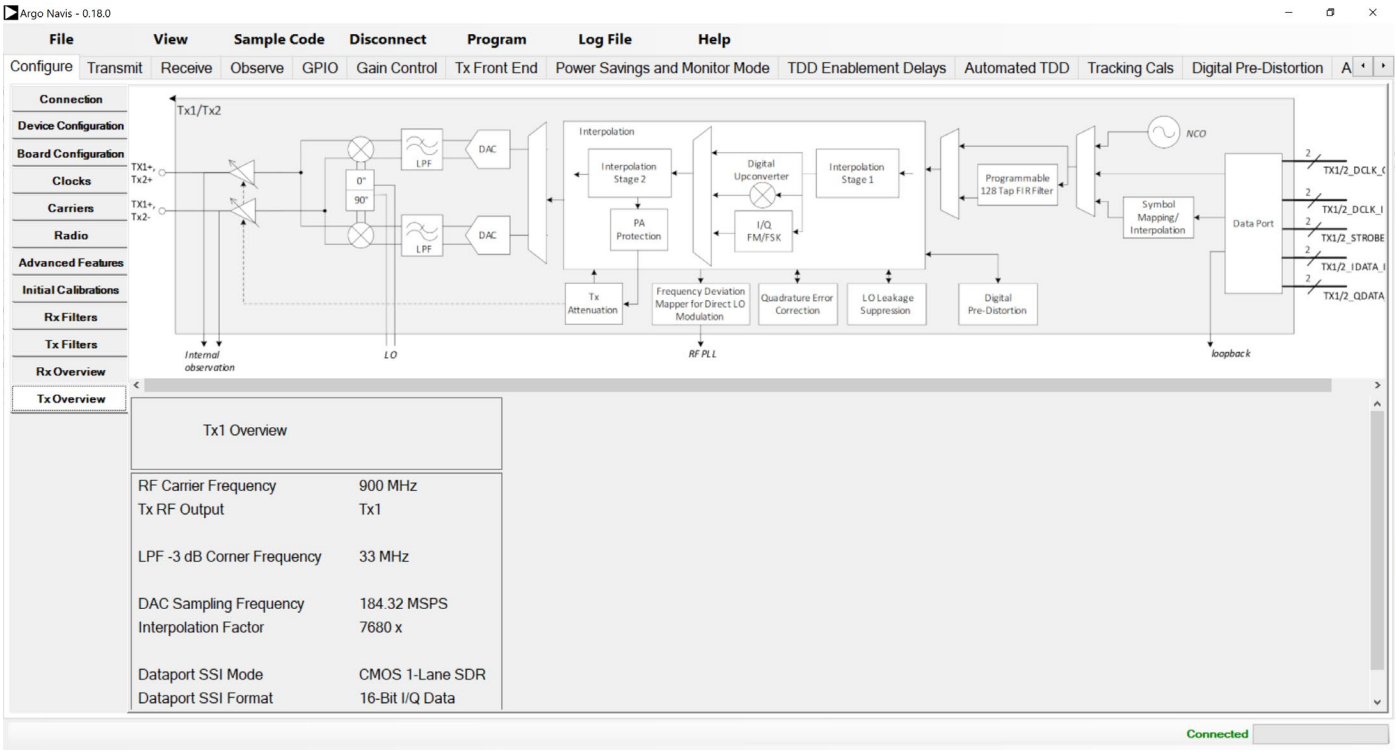


Figure 294. Tx Overview Tab

Receiver Gain Control

The receiver **Gain Control** tab (Figure 295) allows user to configure per channel, receiver gain control mode. Configuration selected in that tab is then applied to the ADRV9001 during initialization. During runtime user can change interface gain as well as if manual mode is enabled Rx gain.

Interface Gain provides ability to select MSBs or LSBs 16 bits from the data bus. This operation can be interpreted as a signal gain. In TDD operation, the user has the option to update the interface gain **Now** or in the **Next Frame**. For more details, see the Receiver Gain Control section.

By selecting the **Manual** radio button in **Gain Control Mode**, the user can select initial gain value. Receiver gain can be changed dynamically during receiver capture operation.

By selecting the **Automatic** radio button in **Gain Control Mode**, the user can configure basic ADRV9001 internal AGC parameters. The AGC becomes operational and automatically adjusts the receiver gain level when the ADRV9001 starts to receive data in the **Receive** tab. See the Receiver Gain Control section for more information about AGC operation.

The user also has the ability to select **Correction** or **Compensation** for **Gain Compensation** operation.

- Compensation: the process of compensating for the analog attenuation in the device (prior to the ADC) with a corresponding amount of digital gain before the digital signal is sent to the user. Gain compensation uses the digital gain to effectively undo analog gain so from receiver data recipient signal stays constant. The digital gain is effectively compensating for the analog attenuation.
- Correction: the process of correction uses digital gain to make the gain steps more accurate. This is to ensure receiver gain steps are accurate.

Filters

ADRV9001 evaluation software allows users to specify their own custom programmable filter for the receiver. This filter is up to 128 taps. The custom filter must be in the format of csv or txt file and coefficients must be 24-bit signed integers with no carriage returns. An example filter .txt file is provided in the SDK . There is also the option to bypass this filter entirely.

An important thing to note when programming the part is the calibrations will use the carrier BW set by the user. If the BW of the loaded filter is smaller than the BW of the carrier then the calibration will fail. This is due to the signal being attenuated by the filter. Setting a small filter is best done after the part has been initialized and the calibrations have been run.

GPIO Configuration

Receiver Gain Control

DGPIOs on the evaluation board from DGPIO_0 to DGPIO_11 can be used for feedback signals as well as setting gain index for receiver gain control (see Figure 295).

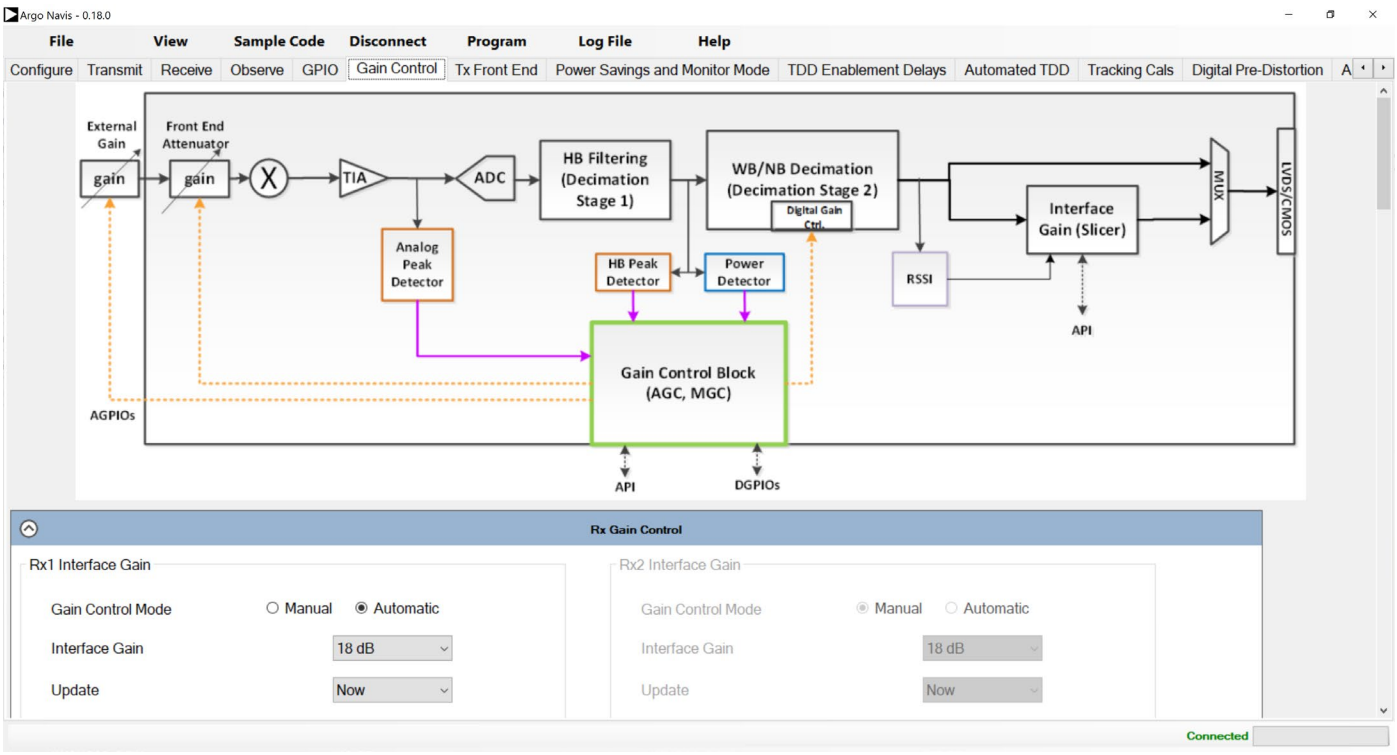


Figure 295. Rx Gain Control Tab

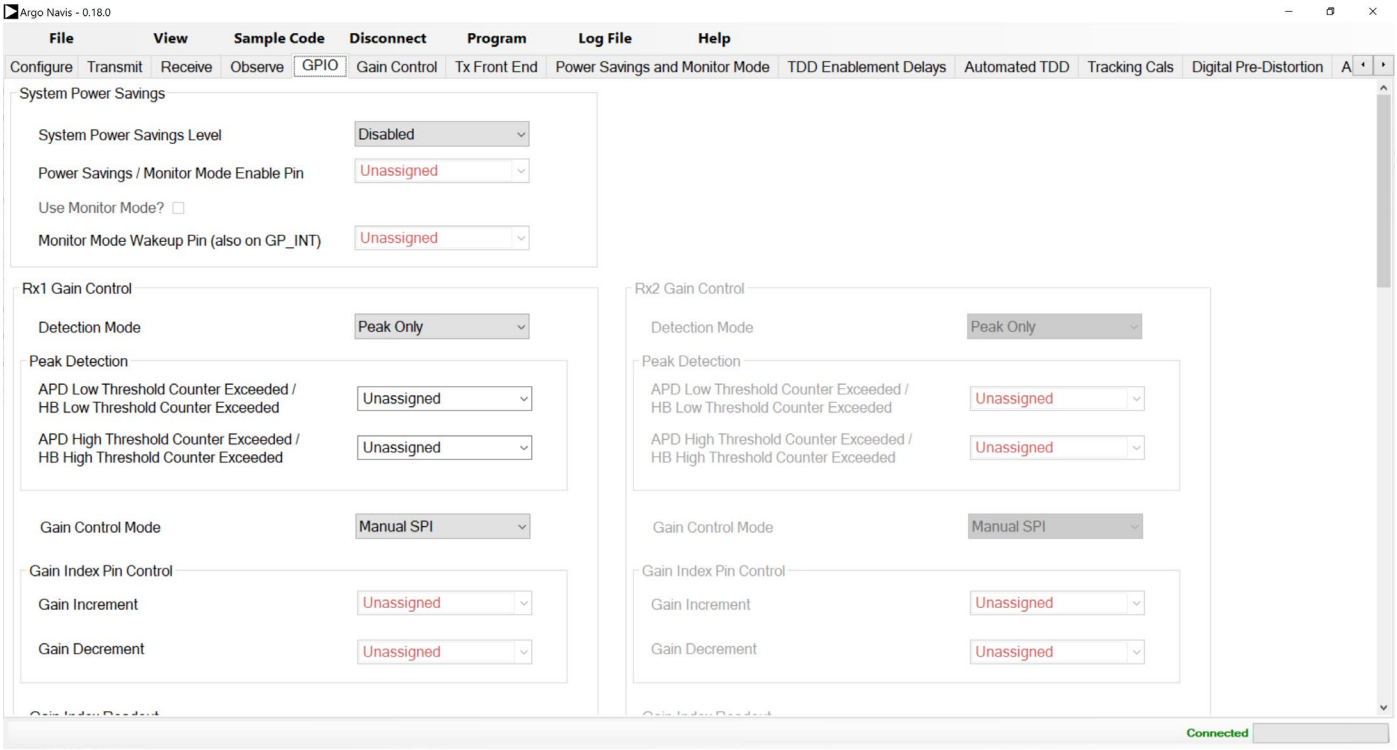


Figure 296. GPIO Configuration

For more detailed information refer to Rx Gain Control section of this document. The GPIO tab also shares a section with the frequency hopping as seen in Figure 285.

Tx Front End

Tx Attenuation

- User can use DGPIO pins for TX attenuation control. User can assign DGPIO pins to attenuation increment and decrement. The step size can be specified in the “Attenuation Control” tab. Default step size is set to 0.05 dB.

Tx PA Ramp

- The user has the option of using one of the Aux DACs to output a ramp to control an external PA.

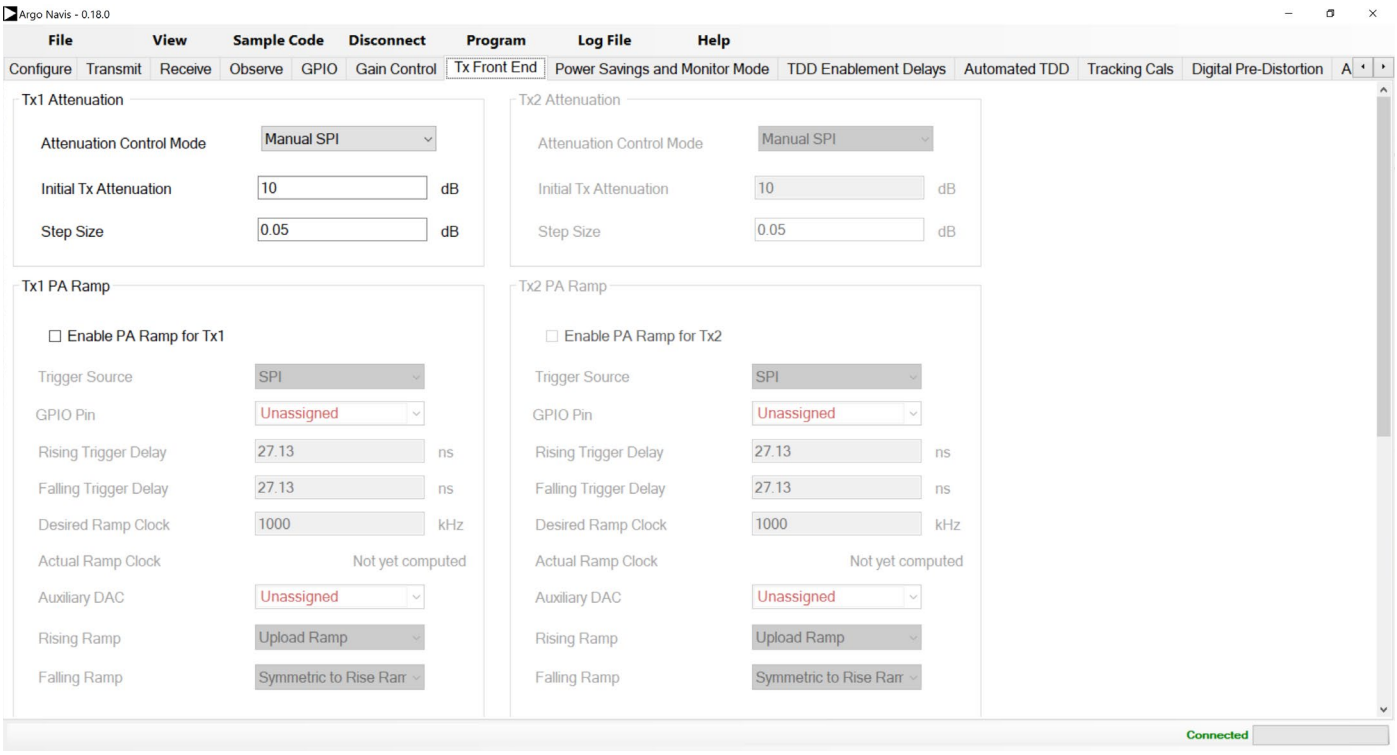


Figure 297. Tx Front End

Once the pins have been assigned, user can go to Transmit tab and start normal playback. User can then adjust Tx attenuation level using the up-down arrows and this will adjust the Tx attenuation value by the step size specified before.

Power Savings and Monitor Mode

User can specify certain power saving mode in this tab (Figure 299). We divide power saving modes to two categories. System Power Savings and Channel Power Savings. System Power Savings include CLKPLL, LDO and ARM power down. These can be controlled via DGPIO pins. Channel Power Savings include RF PLL and LDO power down. These can be controlled via DGPIO pins, as well as using Tx/Rx enables.

Monitor mode can be enabled if Monitor Mode Wakeup Pin is set. Monitor mode window can be brought up from View->Power Savings. If Monitor Mode Wakeup Pin is unassigned, then monitor mode is not enabled. Also note that this window and all pop-up windows can be docked to the main GUI by clicking the ‘Dock’ button in the menu bar of the pop-up.

By selecting the Use Monitor Mode checkbox this allows the user to set up timing for the monitor mode pin to go high giving the BBIC more time to see the pin change. Without this checkbox ticked the pin will just toggle.

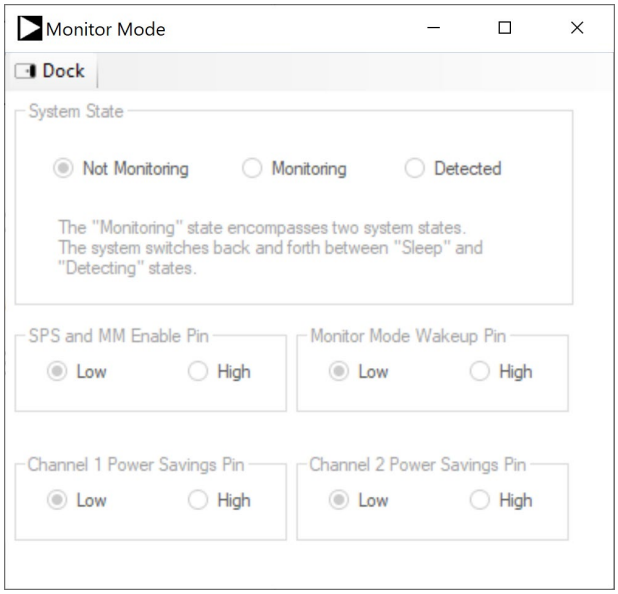


Figure 298. Monitor Mode Window

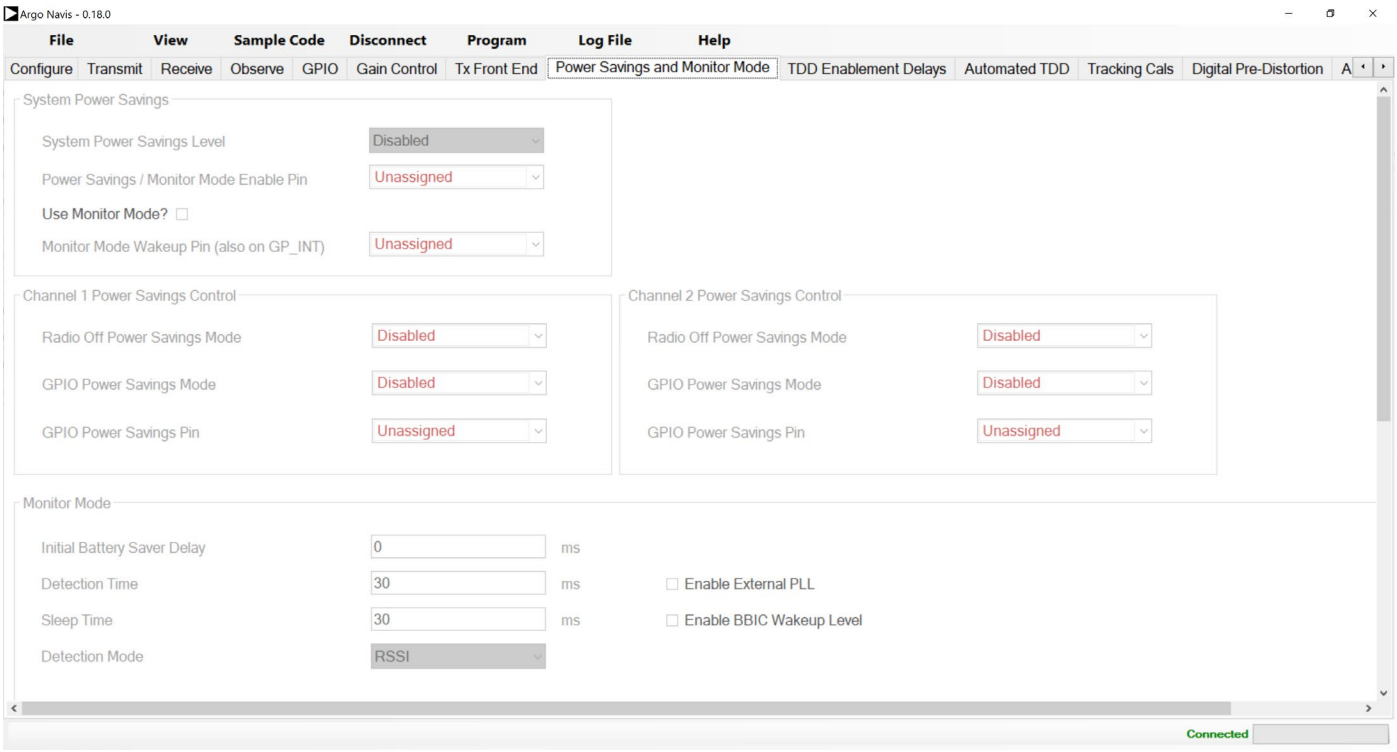


Figure 299. Power Savings and Monitor Mode

Transmitter Operation

Selecting the ***Transmit*** tab opens a page as shown in Figure 300. The upper plot displays the FFT of the digital data and the lower plot shows its time domain waveform. When multiple Tx outputs are enabled, the user can select desired data to be displayed in the Spectrum plot using the checkboxes below the plot.

Once the **Transmit** tab is open, the user can:

- Check the RF Tx Carrier frequency in MHz,
- Change Tx attenuation level in 0.05dB steps
- Continuous Transfer is selected by default and will repeat the chose Tx signal on loop. To transmit just once this can be unselected.

- Transmit content of selected file. Some example files are supplied with TES. Assuming default TES installation process, example files are located in **C:\Program Files (x86)\Analog Devices\ADRV9002 Transceiver Evaluation Software\Example** directory.
- Transmit a single tone, two tones and zeros. User has the ability to adjust the digital power of the single/dual tone signal as well as their frequency offsets
- Have a frequency offset correction option. This allows user to change frequency on the air without re-programming the chip.

Pressing the play symbol moves the ADRV9001 to the transmit state and starts a process where selected Data Files for the "Tx1" and "Tx2" are sent to the ADRV9001. The data is then stored on the Xilinx platform motherboard RAM and the RAM pointer loops through the data continuously until the stop button is pressed.

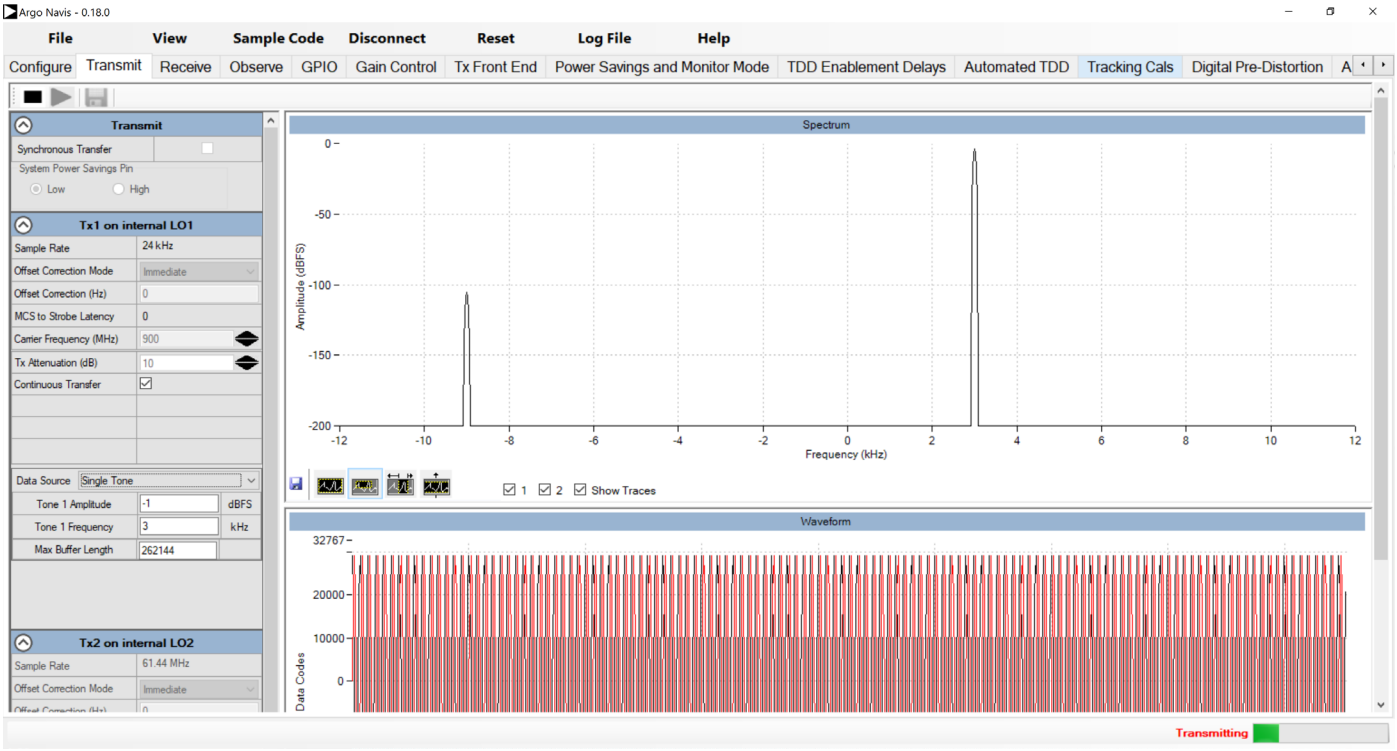


Figure 300. Transmit Data Tab

Transmit Data File Format

Transmit data should be saved in file with extension of txt or csv. The data samples should be either complex (real and imaginary) or real only. Data samples should be Q1.15 fixed point integers (Note in TX Direct Modulation mode, the data sample should be Q4.12 fixed point integers). Data samples should follow the following format:

If the data is only real, imaginary column should be removed, with only one column of real samples.

For example in the case of DMR FM/FSK Direct Modulation, only real data samples are used, in which case data will have only one column.

The length of TX transmit data should be multiple of 64. data file will be played continuously, therefore the data should be phase continuous.

```
Real       | Imaginary
-----------|-----------
I1         | Q1
I2         | Q2
I3         | Q3
I4         | Q4
.          | .
.          | .
```

Receiver Operation

The ***Receive*** tab opens a window as shown in Figure 301. The upper plot displays the FFT of the received input data and the lower plot shows its time domain waveform. When multiple Rx inputs are enabled, the user can select the desired data to be displayed in the Spectrum plot using the checkboxes below the plot.

In TDD operation, Rx data is displayed only when Rx enabled is high. It will not display data gap between TDD time slots.

Once the **Receive** tab is open, the user can:

- Check the RF Rx Carrier frequency in MHz,
- Change capture length in number of samples,
- Change Rx gain level (gain table index),
- Change Rx interface gain (in 4 steps),
- Enable/disable Baseband DC Rejection tracking calibration
- Change frequency offset in Hz
- Read back main parameters measured in received signal such as fundamental frequency, its amplitude and DC offset,
- Plot and save received data by clicking on floppy disk icon in bottom left corner.
- Save Rx captured data (specified in Capture Length window) in form of *.csv or .tsv file.
    - There is an interleaving option pop-up widow displayed when saving as a .tsv file.

Pressing the play symbols enables the selected receivers and displays received data continuously until the **Stop** button is pressed.

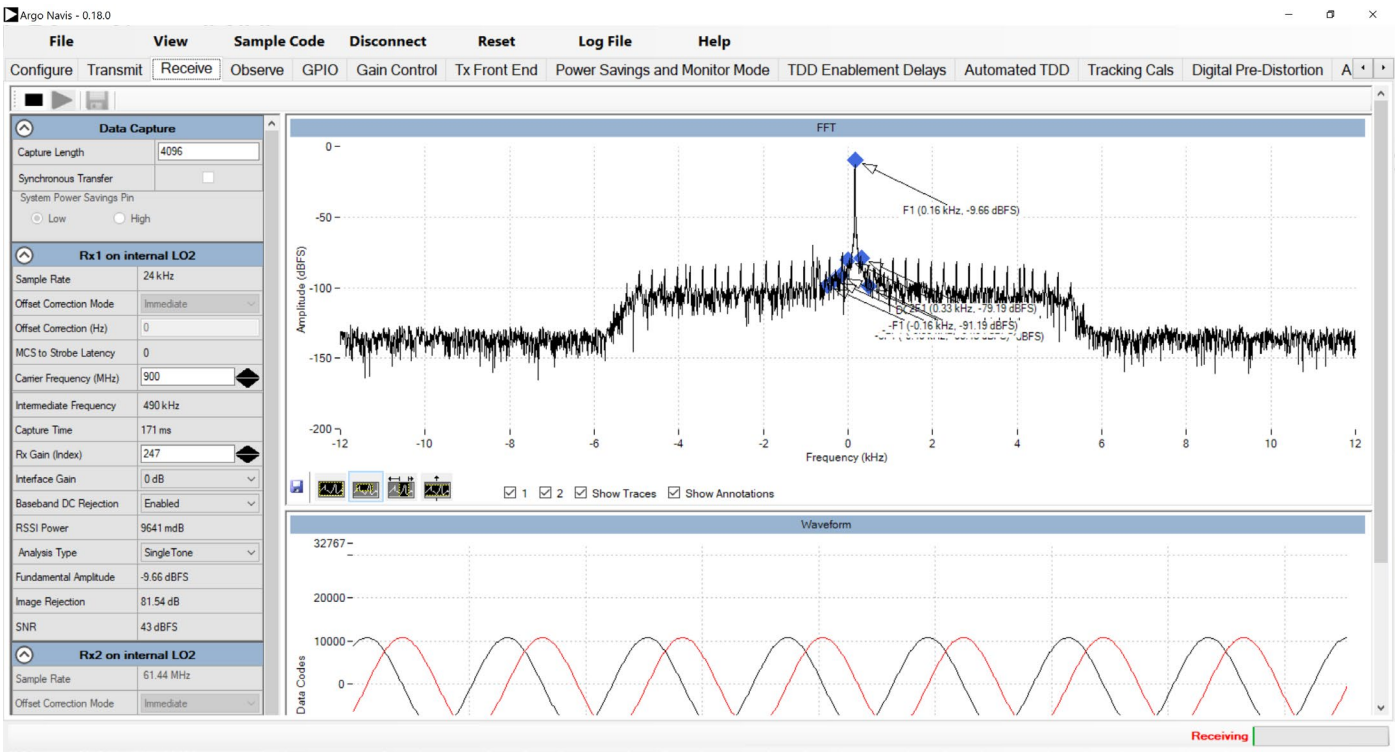


Figure 301. Receive Data Tab

Frequency Deviation

If Rx frequency deviation is enabled in the configuration step, Rx input signal will be demodulated. For example if a continuous wave of 900.003 MHz is sent to the RX port with LO of 900 MHz, which means in baseband there is a tone of 3 kHz offset from LO, on the RX tab it is expected to see a constant of 3000 in the time domain plot.

Captured Data Format

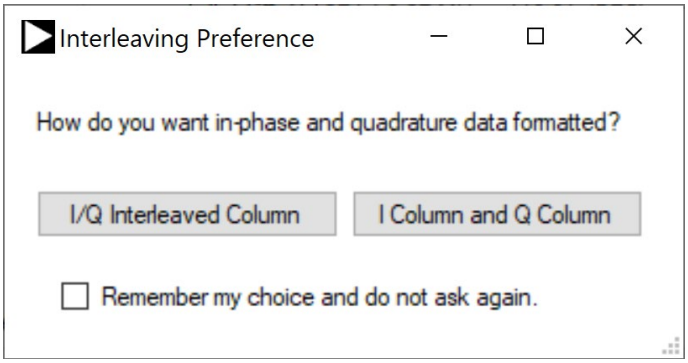


Figure 302. Interleaving Option

RX captured data can be saved using the save button next to the play button. The data is saved in either tsv or csv format. Each column corresponds to one channel. Data samples follow 1Q15 fixed point format, when the user selects the interleaved option. Shown as follows:

```
Channel 1 | Channel 2
----------|-----------
I1        | I1
Q1        | Q1
I2        | I2
Q2        | Q2
.         | .
.         | .
.         | .
```

When the I and Q column option is selected the data is formatted as follows:

```
Channel 1I | Channel 1Q
----------|-----------
I1        | Q1
I2        | Q2
I3        | Q3
I4        | Q4
.         | .
.         | .
.         | .
```

In the case of RX frequency deviation, only I samples are shown, all Q samples are 0.

Observer Operation

The ***Observe*** tab opens a window as shown in Figure 303. The upper plot displays the FFT of the received input data and the lower plot shows its time domain waveform. This tab operates the very same as the ***Receive*** tab above. It is used to observe the Transmitted signals.

Enable this tab by selecting the I/Q Signal Type in the Device Configuration tab when setting up the device parameters. The user needs to start transmitting from the Transmit tab before being able to press the 'play' button in the observe tab.

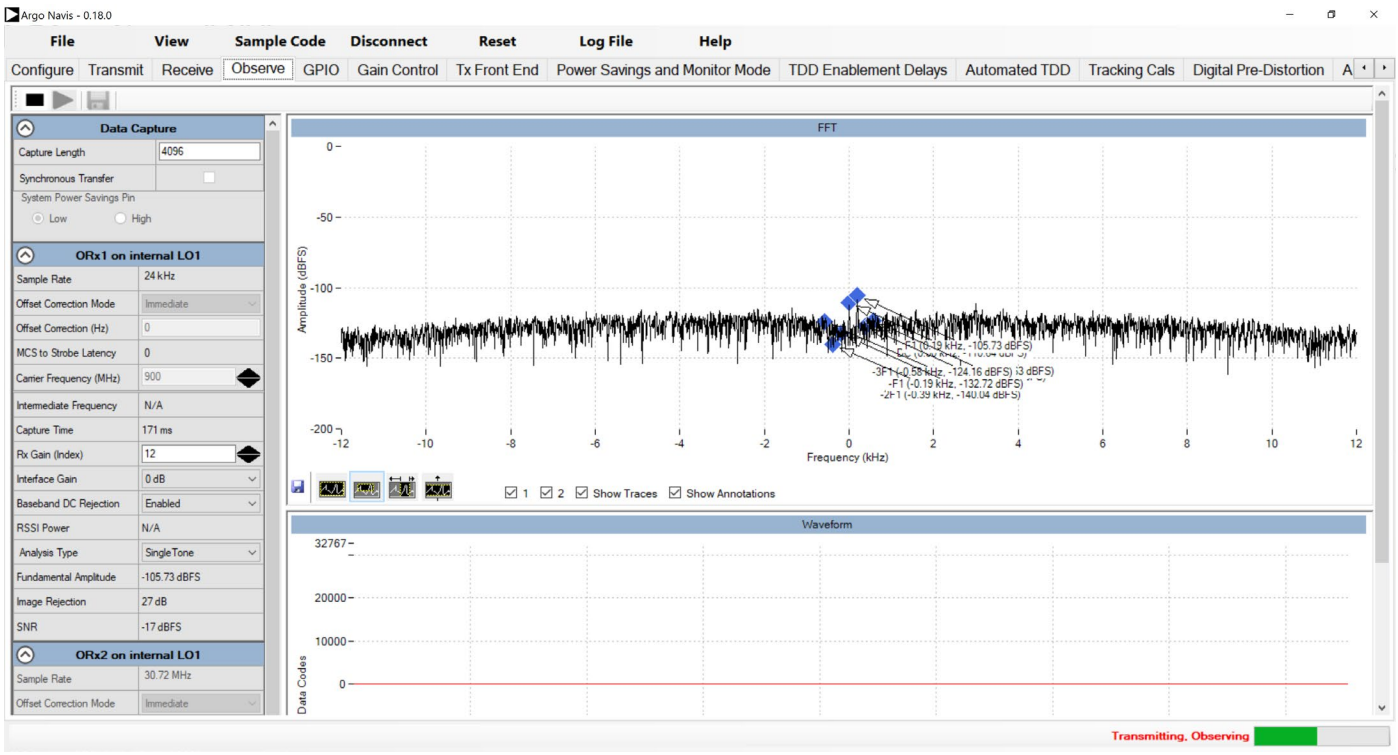


Figure 303. Observe Tab

Automated Time Division Duplexing (TDD)

ADRV9001 supports automatic TDD operation. User can send and receive TDD framed data by configuring this tab (see Figure 304). This of course depends on how system and setup is selected described in the previous sections. ADRV9001 comes with predefined timing configurations by default. However user can configure the timing as needed.

In the Automated TDD tab, user can configure the parameters using TDD configuration files:

- Frame and Sequences
  - User can specify the duration of a frame
  - User can select from sequencing of frames
  - User can specify the number of frames in a sequence.

When the Enable Automated TDD check box is selected the used will be prompted to display a timing diagram. This diagram will update every time one of the parameters in the TDD table is changed. This timing diagram shown in Figure 305 is based off the TDD table in Figure 304.

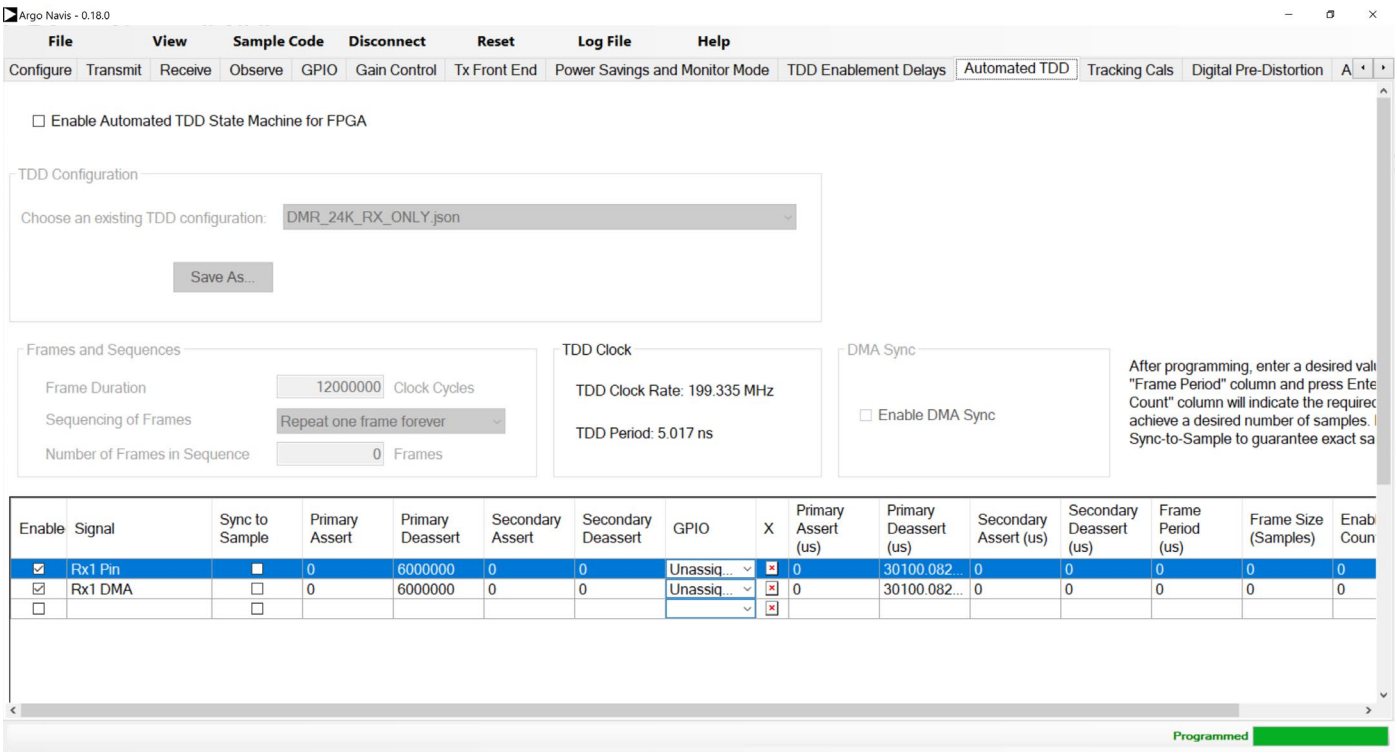


Figure 304. Automated TDD Configuration Tab

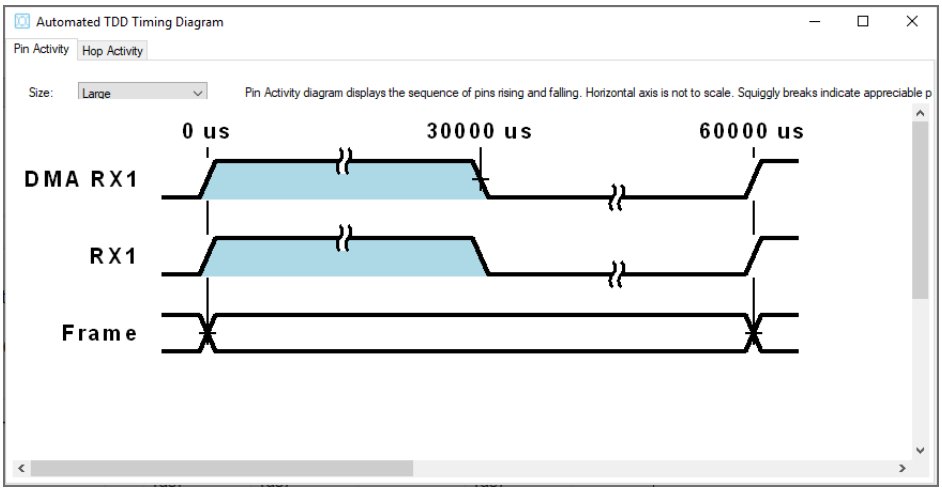


Figure 305. Automated TDD Timing Diagram

TDD Parameter Table

The table s auto populated by the TES based on the configuration file chosen

- Enable Column
  - User can enable/disable receiver/transmitter channel.
- Signal Column
  - This displays the signal name attributed to that row.
- Frame Timing Columns
  - A predefined timing is generated by the TES based on the profile selected
  - User can modify the timing by entering **Primary Assert/Deassert**, **Secondary Assert/Deassert**

- Assert/Deassert entries are frame locations, they are not durations, for example if RX1 primary assert is 0 and primary deassert is 10000 µs, this means within the specified frame the RX1 enable is on from 0 to 10000 µs and off for the rest of the frame.
- In Figure 306, it shows visually what primary/secondary assert/deassert mean. Black and grey indicates TX and RX subframe data.

- Signal Input
  - User can change the values of a signal already in the table or add a new signal to the table. This is done by selecting the row to be edited in the table, filling in the parameters in the row underneath the table and clicking Apply.
  - To add a new row, select the blank row at the bottom of the table and click Apply with the parameters for the new row.

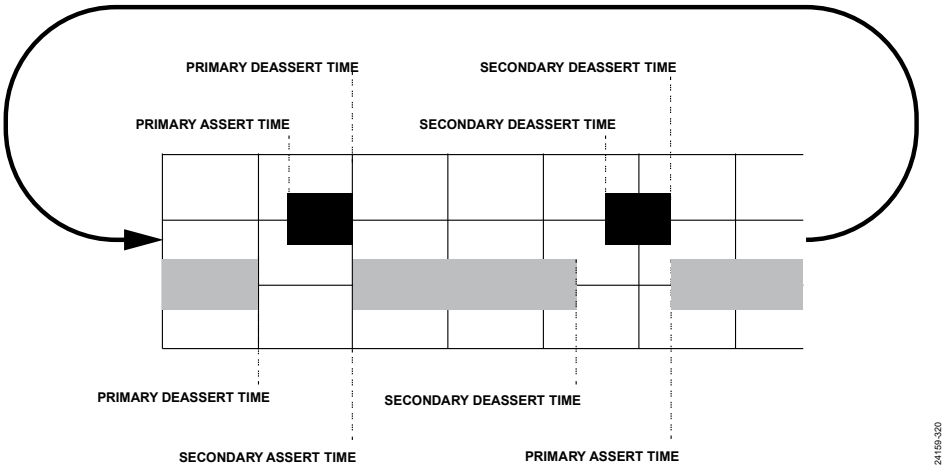


Figure 306. TDD Frame Timing Illustration

Table 114. TDD Signals

Signal	Description
RX1 Pin TX1 Pin RX2 Pin TX2 Pin	These signals are hardwired to the Rx/Tx ENABLE pins (used as SETUP signals in FH)
ORX1 Pin ORX2 Pin	This can be used as ORX enable signal when routed to the GPIO assigned as ORX control
RX1 DMA RX2 DMA TX1 DMA TX2 DMA ORX1 DMA ORX2 DMA	The DMA enables that gate data transfer for each of the channels
RX1 DMA Trigger RX2 DMA Trigger TX1 DMA Trigger TX2 DMA Trigger ORX1 DMA Trigger ORX2 DMA Trigger	These signals can be configured as triggers for the DMAs to signify that data transfer on that channel should only occur when the trigger signal has pulsed high followed by the DMA signal for that channel going high.
SMA1 Trigger	Hardwired to dedicated SMA (J67) on the ZC706 board, this signal can be used to trigger external equipment. No option on ZCU102 currently.
SMA2 Trigger	Hardwired to dedicated SMA (J68) on the ZC706 board, this signal can be used to trigger external equipment. No option on ZCU102 currently.
General Purpose 1/ Hop Pin	General Purpose signal that can be routed to GPIO pins. This is also used as the Hop Pin in FH mode.
General Purpose 2 - 6	General purpose signals that can be routed to GPIO pins as needed

In Figure 304 a sample JSON file with DMR settings has been loaded. The parameters appear in the table for the Rx enable (Rx1 Pin) and DMA signals. In this example the frame length is 12000000 clock cycles (60ms). Both signals primary assert happens at the beginning of the frame and then drops on the 6000000 clock cycle (30ms). The frame then repeats itself as the Repeat one frame forever has been selected.

Enabling Tx1 DMA sends data from FPGA to SSI interface. Disabling Tx1 DMA stops sending data from FPGA to SSI interface. It works together with Tx_interface enabling/disabling (accepting data from SSI at ADRV9001) so it provides more flexibility for user to control what data to transmit. For example, the user can have 4 different scenarios:

- DMA disabled, Tx_interface disabled: nothing is transmitted
- DMA disabled, Tx_interface enabled: 0s are transmitted
- DMA enabled, Tx_interface disabled: Data in DMA is not transmitted and is lost
- DMA enabled, Tx_interface enabled: Data in DMA is transmitted

The Tx1 DMA trigger is defined by the following enum:

```
typedef enum adi_fpga9001_DmaTrigger
{
   ADI_FPGA9001_DMA_TRIGGER_SMA_1     = 0,
   ADI_FPGA9001_DMA_TRIGGER_SMA_2     = 1,
   ADI_FPGA9001_DMA_TRIGGER_MCS       = 2,
   ADI_FPGA9001_DMA_TRIGGER_GPIO      = 3,
   ADI_FPGA9001_DMA_TRIGGER_TDD_ENABLE = 4,
   ADI_FPGA9001_DMA_TRIGGER_IMMEDIATE  = 5
} adi_fpga9001_DmaTrigger_e;
```

- If using ADI_FPGA9001_DMA_TRIGGER_IMMEDIATE then the DMA will transfer data whenever the DMA_Enable goes high (and has valid data from the SSI).
- If using ADI_FPGA9001_DMA_TRIGGER_TDD_ENABLE then the adi_fpga9001_TddConfig_t->ADI_FPGA9001_TDDSELECT_RX1_DMA_TRIG signal will be the trigger. When this signal goes high (programmed in the TDD config) then the DMA is triggered and will transfer data whenever the DMA_Enable goes high (and has valid data from the SSI)
- If using ADI_FPGA9001_DMA_TRIGGER_GPIO/MCS/SMA_1/2 then the GPIO/MCS/SMA signal will be the trigger. When this signal goes high the DMA is triggered and will transfer data whenever the DMA_Enable goes high (and has valid data from the SSI)

ADI_FPGA9001_DMA_TRIGGER_IMMEDIATE is used in the current release.

Tracking Calibrations

Tracking calibration algorithms can be enabled/disabled in the ***tracking cals*** tab. Certain algorithms can only be enabled in certain profiles. For example RX harmonic distortion can only be enabled if it is configured in DMR, Analog FM, and Tetra profiles. It grays out and is disabled in LTE and custom profiles.

The user can enable and disable an individual tracking algorithm to observe its effect while transmitting and receiving a signal.

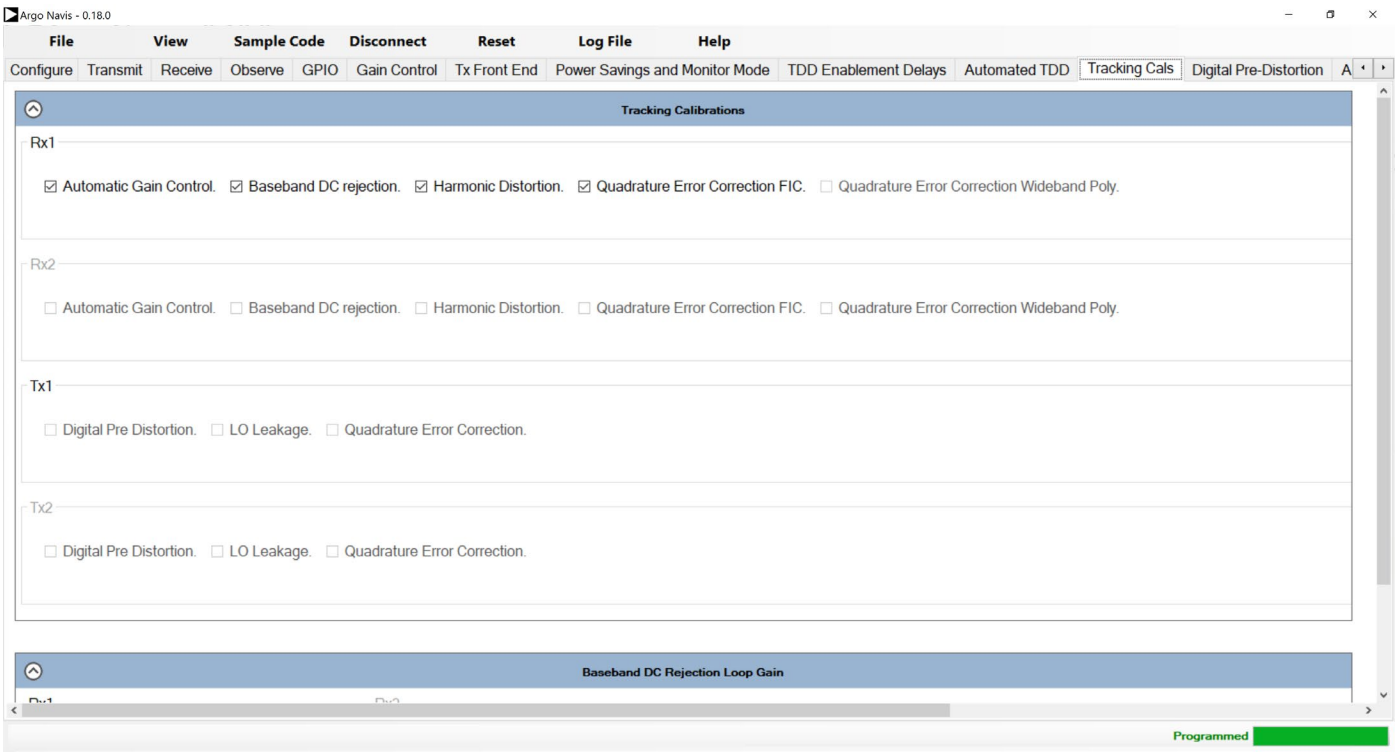


Figure 307. Tracking Calibration Tab

Digital Predistortion

For more detailed information on DPD, see the Digital Predistortion section for more details. This tab also contains the Closed-Loop Gain Control (CLGC) settings. This uses the same processing engine as the DPD and is used to keep the gain from the output of the PA at a constant level.

To use the CLGC use it in open-loop configuration at first and read the 'CLGC Last Gain' and 'CLGC Filtered Gain' from the Transmit tab. The user then inputs these readings into the CLGC Gain Target and CLGC Filter Alpha fields in the Digital Pre-Distortion tab and closes the loop by unchecking the open-loop tick box. The Tx output power can be controlled by the CLGC Target Gain (dB) field in the transmit tab.

To use either of these functions the DPD tracking calibration needs to be turned on, the two functions use the same tracking engine.

TDD Enablement Delays

For more detailed information on TDD enablement delays, see the Timing Parameters Control section for more details.

Auxiliary DAC/ADC

ADRV9001 evaluation software allows user for setting Auxiliary ADC/DAC for different control or monitoring purposes. User can go to Auxiliary tab and enable Aux DAC/ADC here. For Aux DACs, user must specify a DAC code, valued from 0 ~ 4095. This effectively sets the voltage level for that Aux DAC pin. For Aux ADCs, user can press **Capture Again** to observe the one time voltage value on the Aux DAC pin.

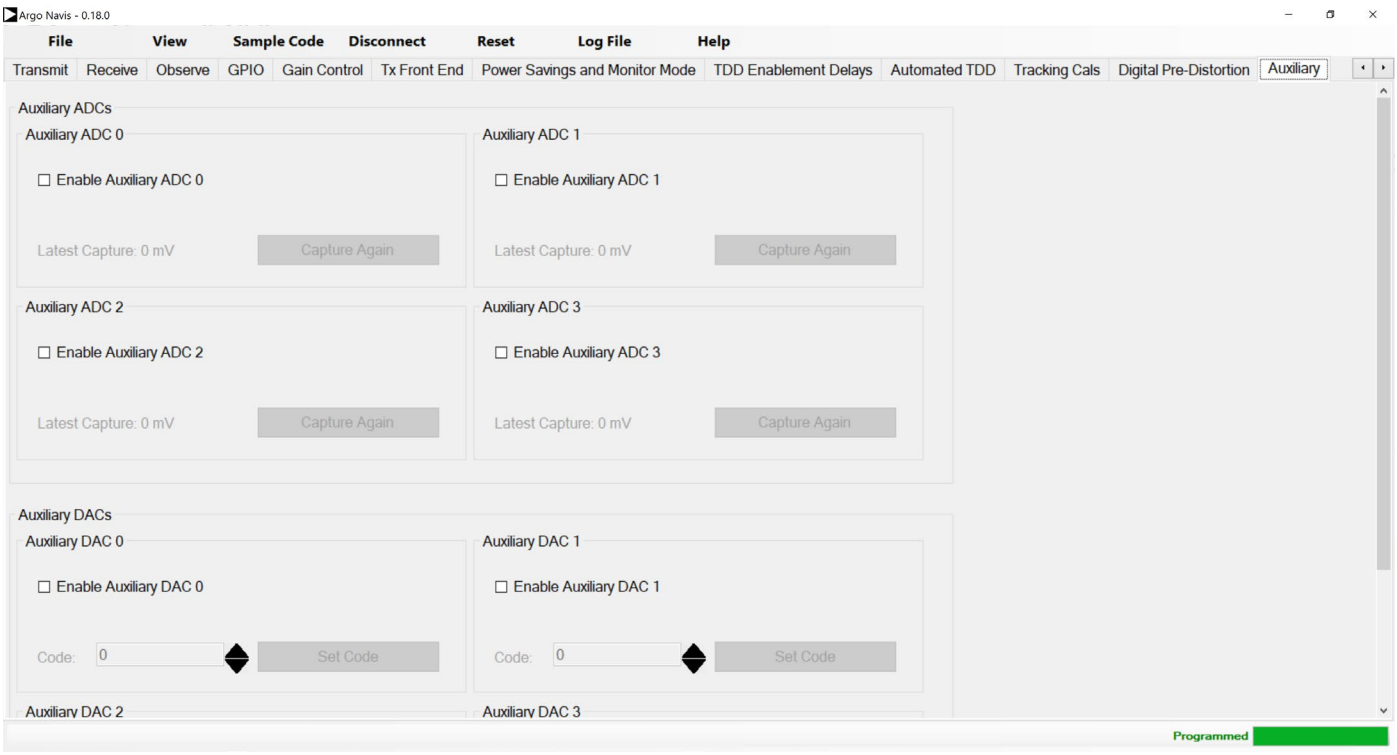


Figure 308. Auxiliary DACs and ADCs

FREQUENCY HOPPING TES EXAMPLES

Here we show two examples of utilizing TES to achieve frequency hopping for ADRV9001. For details on the frequency hopping operation go to the Frequency Hopping section of this document.

Example 1: Manual Frequency Hopping

In this mode the user does not need to specify timing. In most cases, this is a mode to help users understand how frequency hopping operates in ADRV9001. Here we use DMR profile as an example. Other profiles are largely the same.

In this example we will show frequency hopping for Tx only Rx only and TRx.

3. Connect.
4. Go to **Carriers** tab on the left pane.
5. Select Frequency Hopping under Carrier Configuration Mode.

The first three steps are the same for Automated TDD Frequency Hopping as well.

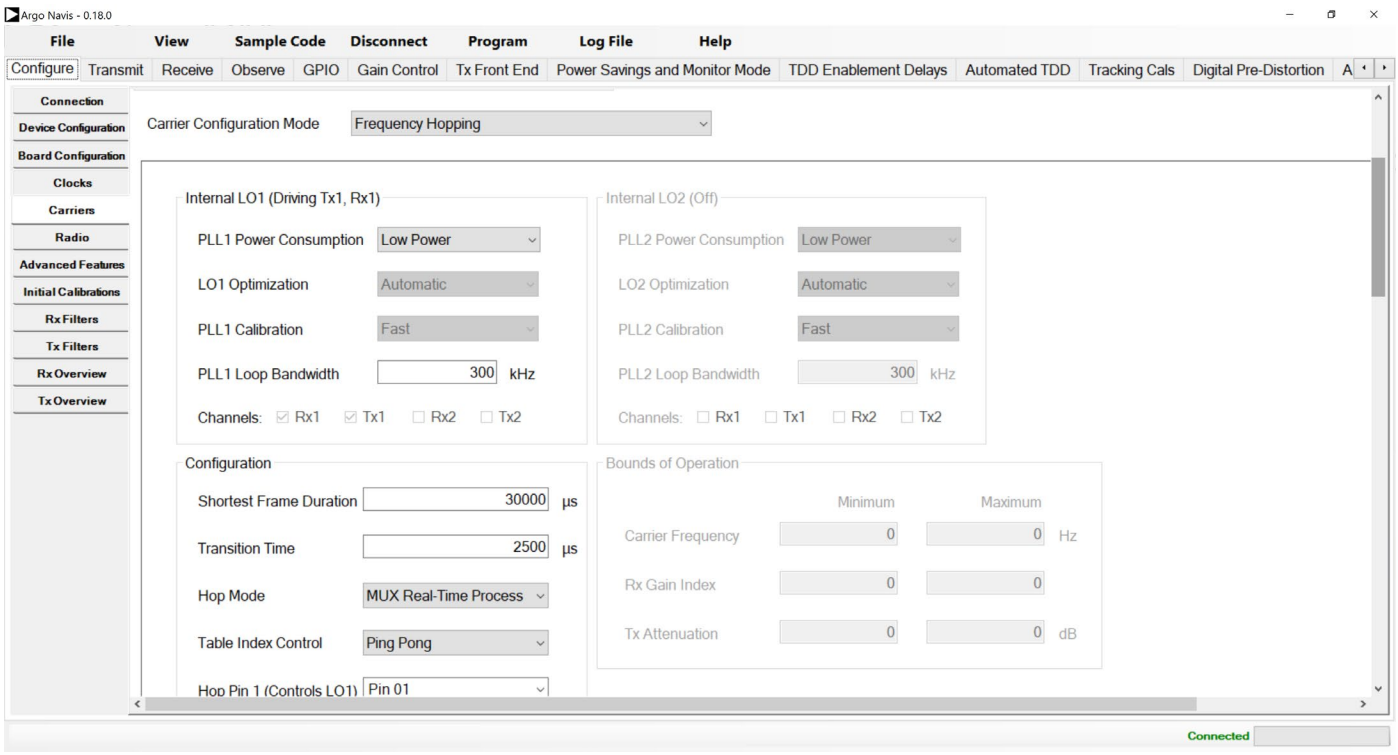


Figure 309. Manual Frequency Hopping Example

4. Specify the Hop Pin as Pin 01. By default it should be set to Pin 01.
5. Specify Hop Mode. Different profiles may have certain modes enabled/disabled
    a. Mux Preprocess indicates two LOs are in use for frequency hopping, and the tables are preprocessed prior to hopping
    b. Mux Real-Time Process indicates two LOs are in use for frequency hopping, and tables are processed at the hopping stage.
    c. LO Retuning (Real-Time) indicates only one LO is used for frequency hopping and the tables are processed at the hopping stage. This will mean that the immediate next frame will be updated and not the frame-after-next which is expected when using two LOs.

    In this example we will use default Mux Preprocess.
6. Specify Operation.
    a. Automatic Loop - Automatically increment through a frequency hopping table and wrap around once end has been reached.
    b. Automatic Ping Pong - Ping pong operation between ADI_ADRV9001_FHHOPTABLE_A and ADI_ADRV9001_FHHOPTABLE_B. Automatically increment through one frequency hopping table and switch to the other once end has been reached.
    c. GPIO - Use DGPIO pins to index entries within a frequency hopping table.

    In this example, we will use default Ping Pong mode.
7. Load frequency hopping tables.
    a. Here user can load two tables (table A and table B) or load only one table (table A or table B). We provide the frequency hopping table examples in the GUI. Under /Example.
    b. At this point user can edit the frequency tables in the files that will be loaded. User cannot edit values directly on the GUI. Once values in the table are changed, the user should load the table again.

    In this example, we load two tables.

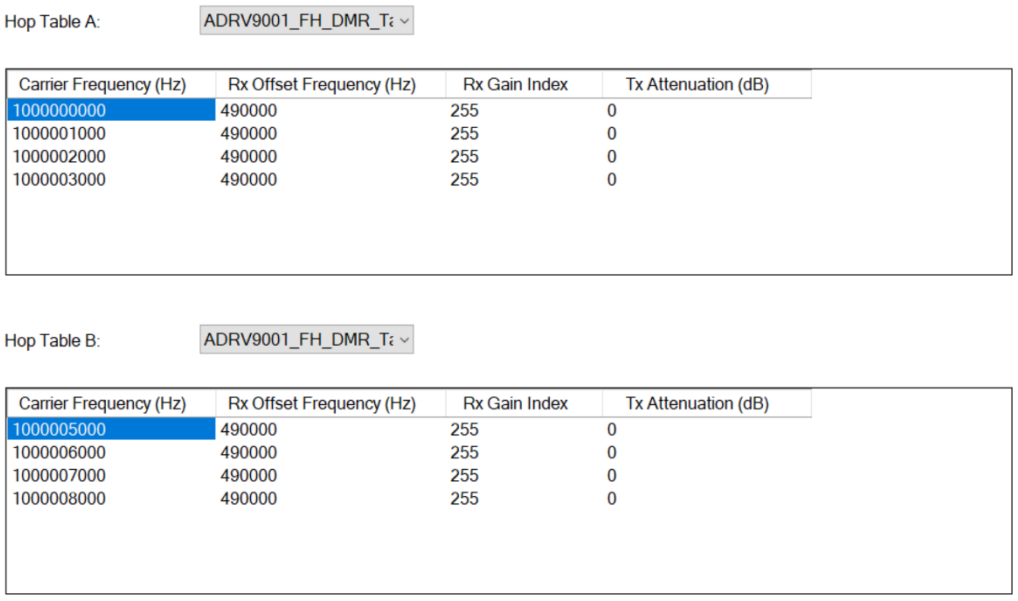


Figure 310. Frequency Hopping Tables

8. The Executed Sequence is displayed in the table at the bottom of the tab. Any changes made to the frequency hopping setting will automatically populate this table to show the sequence that will be programmed.

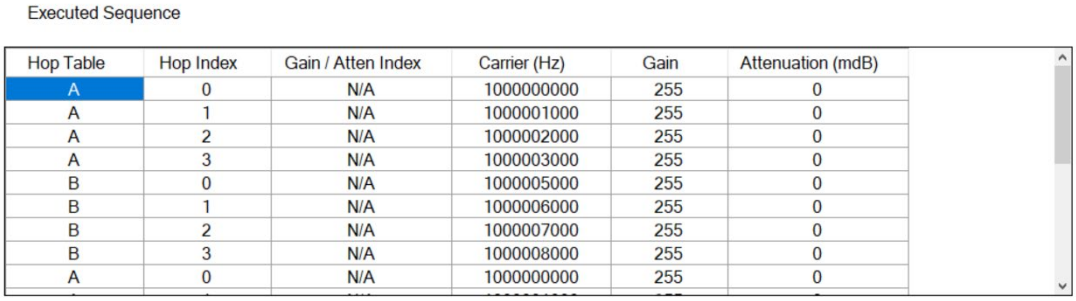


Figure 311. Frequency Hopping Executed Sequence

9. Click Program.
10. Upon successful programming, go to Transmit tab and click play, a window should pop up and indicates to the user frequency hopping is working in manual mode.
    a. Commit Frame-After-Next to Rx/Tx and Perform Hop button will "commit" the "Frame-after-Next" to Rx/Tx. Notice this is not the same as next frame, it's the one after.
    b. Scroll to Current Frame button will locate the current frame to show to the user. This is very useful if the user has a large set of frequencies in the frequency table.
    c. Each rectangle box under it shows a frequency entry in a frequency table. This includes:
        i. Table being used (table A or table B)
        ii. Carrier frequency
        iii. Rx Gain index
        iv. Rx offset frequency
        v. Tx attenuation

    Frequency, gain and attenuation values should match those in the tables loaded.

Tx Only

11. Change the Tx Data Source Single Tone, in the main TES window, and put 0 Hz under Tone 1 Frequency.
    a. You should see figure

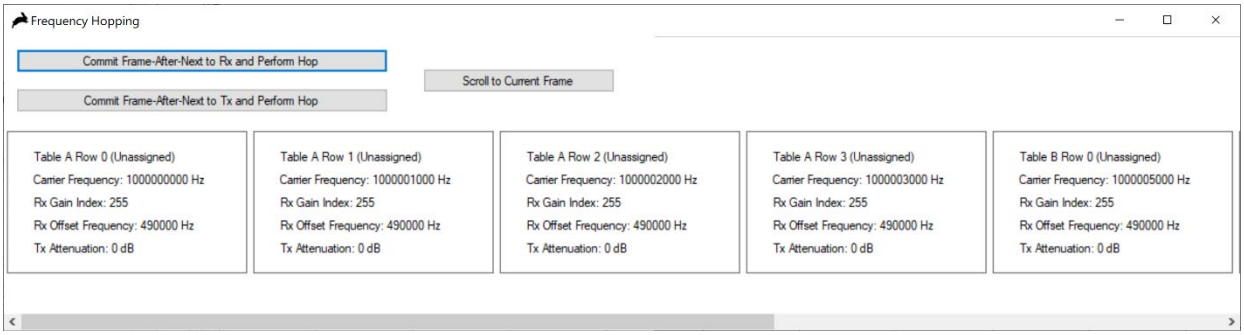


Figure 312. Manual Frequency Hopping Using TES

    b. This indicates
        vi. All entries of frequencies are shown, both in table A and table B.
        vii. All entries should display Unassigned.
        viii. The upcoming frame is not assigned.
        ix. There should not be any signal coming out of Tx.
12. Click on "Commit Frame-After-Next to Tx and Perform Hop."

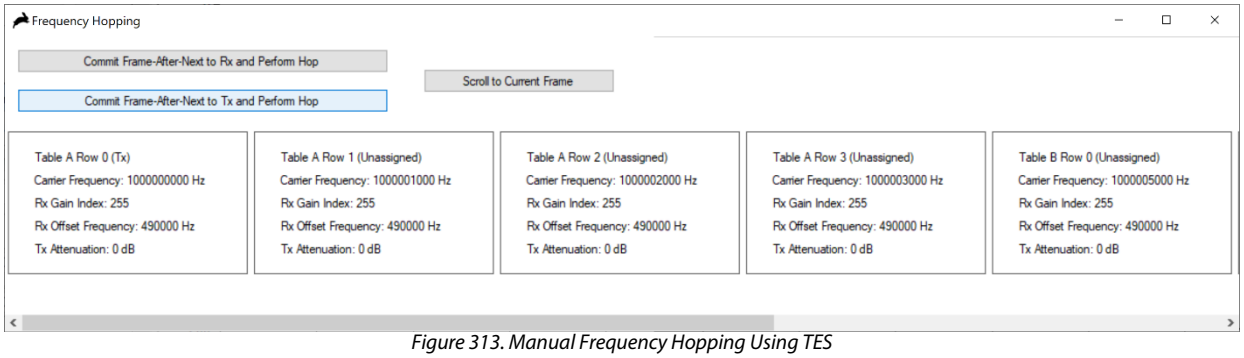


Figure 313. Manual Frequency Hopping Using TES

    c. The upcoming frame is assigned to Tx.(first box).
    d. There should not be any signal coming out of Tx.
13. Click on "Commit Frame-After-Next to Tx and Perform Hop" again.

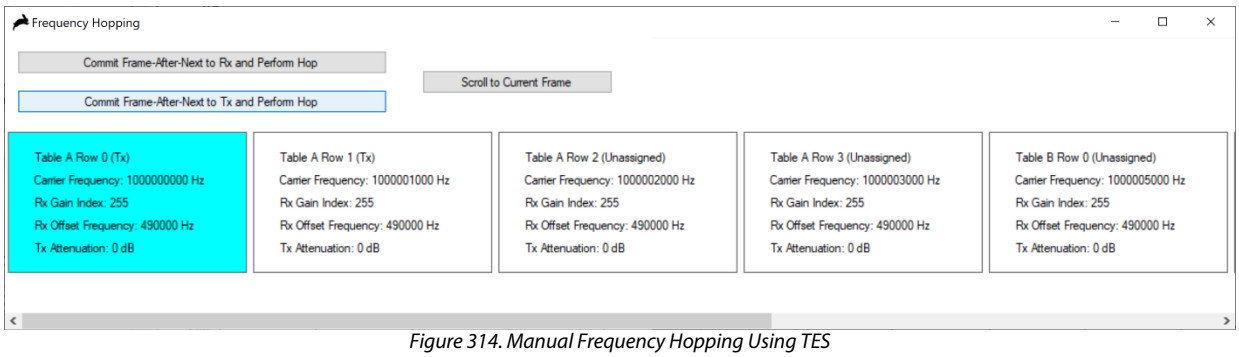


Figure 314. Manual Frequency Hopping Using TES

e. The current frame is being played for frequency 1000 MHz, highlighted as blue.
f. Next frame is assigned as Tx as well.
g. Signal should appear at Tx output as 1000.001 MHz single tone.

14. Repeat the last step.

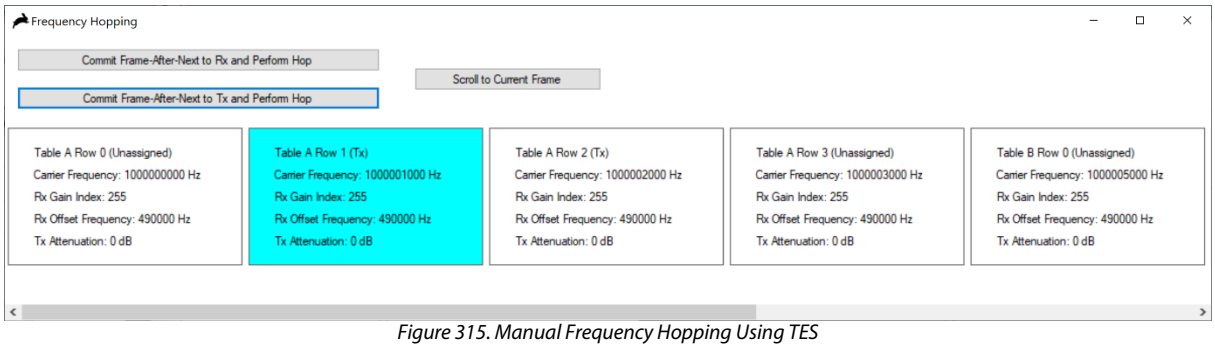


Figure 315. Manual Frequency Hopping Using TES

h. The current frame is advanced to the next 1000.001 MHz.
i. The previous entry returned to the state Unassigned.
j. The next frame is set to 1000.002 MHz.
k. Signal should appear at Tx output as 1000.001 MHz single tone.

Rx Only

Rx only steps are the same for Tx Only shown in Tx Only. The only difference is user instead of click play on Transmit tab, must click play on Receive tab.

TRx

TRx steps are also very similar. Except user must click play on both Transmit and Receive tabs before operating frequency hopping.

Note, upon reset, all frequencies entries should be assigned back to Unassigned.

Example 2: GPIO Controlled Frequency

Following the setup of ***Example 1*** through to step 8, the user can now look to implement GPIO controls with the following instructions.

1. The user can enable separate gain and attenuation tables that correspond with a GPIO codes that can be called for individual hops. This is done by loading tables into the Gain Table and Attenuation Table in the Frequency Hopping Tab.

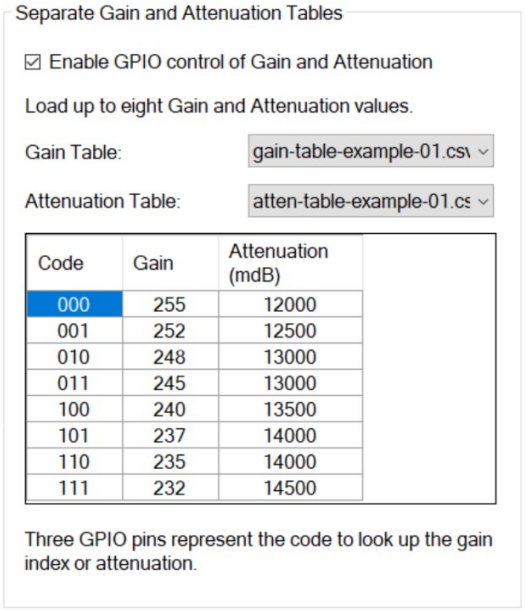


Figure 316. Separate Gain and Attenuation Tables

2. When the tables are loaded the user can enable an upload sequence in the Control Hopping by GPIO section shown in Figure 317. This CSV file contains a row of integers from 0 to 7 in the sequence that the user wants. The integer 0 will correspond to the gain and attenuation value used in the table shown in Figure 316. This allows the user to reuse the same gain and attenuation settings for multiple hops without the need to load the setting multiple times.

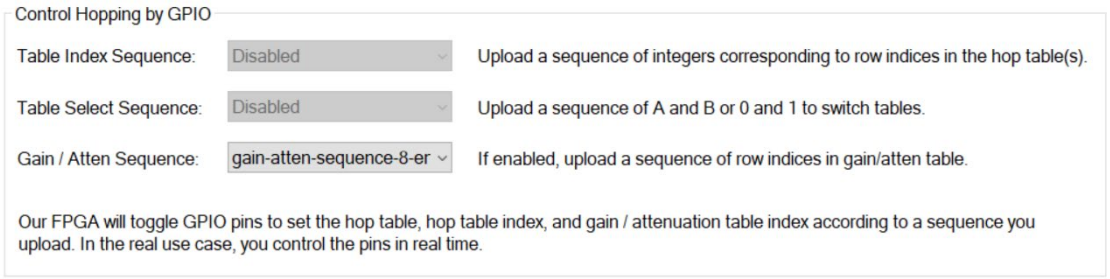


Figure 317. Control Hopping by GPIO

The FPGA board will toggle the GPIO pins to set the hop table index according to the sequence loaded. The user can select GPIO pins used in the GPIO Pins for Control of Hop Table section.

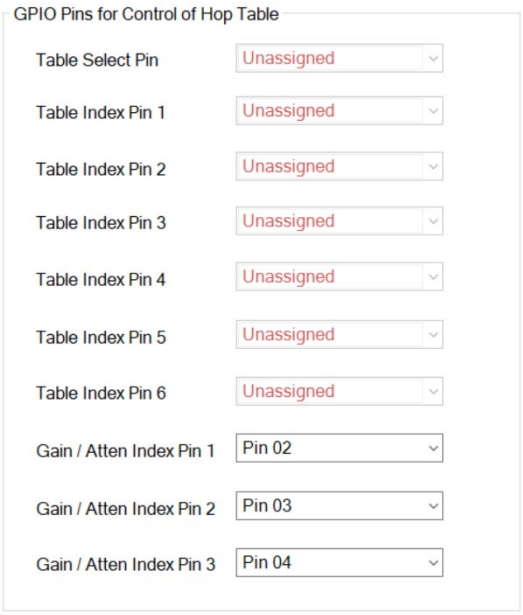


Figure 318. GPIO Pins for Control of Hop Table

3. The Executed Sequence table gets automatically updated with the gain and attenuation settings as indicated by the sequence table loaded in step 2.

Executed Sequence

Hop Table	Hop Index	Gain / Atten Index	Carrier (Hz)	Gain	Attenuation (mdB)
A	0	000	1000000000	255	12000
A	1	001	1000001000	252	12500
A	2	010	1000002000	248	13000
A	3	011	1000003000	245	13000
B	0	100	1000005000	240	13500
B	1	101	1000006000	237	14000
B	2	110	1000007000	235	14000
B	3	111	1000008000	232	14500
A	0	000	1000000000	255	12000

Figure 319. Executed Sequence Table (GPIO settings)

4. As per ***Example 1*** when the user programs the part and clicks play in the ***Receive tab*** they will be shown the Frequency Hopping window and can commit frames to Rx or Tx. In this example the user will notice that the signal is now being attenuated at the rate described in the gain and attenuation tables.

Example 3: Automated TDD with Frequency Hopping

Unlike the previous mode, in this mode user will have to specify TDD timing. User also will not have the frequency hopping window to advance frames manually, instead the frames will be played automatically.

To achieve this,

1. Follow the steps above before programming
2. In Automated TDD tab, click Enable Automated TDD State Machine for FPGA
3. ADRV9001 TES includes several pre-defined examples in the /Examples folder.

Rx Only

1. Select the pre-defined json file DMR_24K_RX_ONLY_FH.json
2. Hop Pin should be set automatically.
3. Go to Receive tab, set the capture length to longer value. Here we set it to 65536.
4. Click play

You should see the signal being played for multiple frequencies.

Note in the time domain, TES only shows actual data frames, that is without gap between data frames.

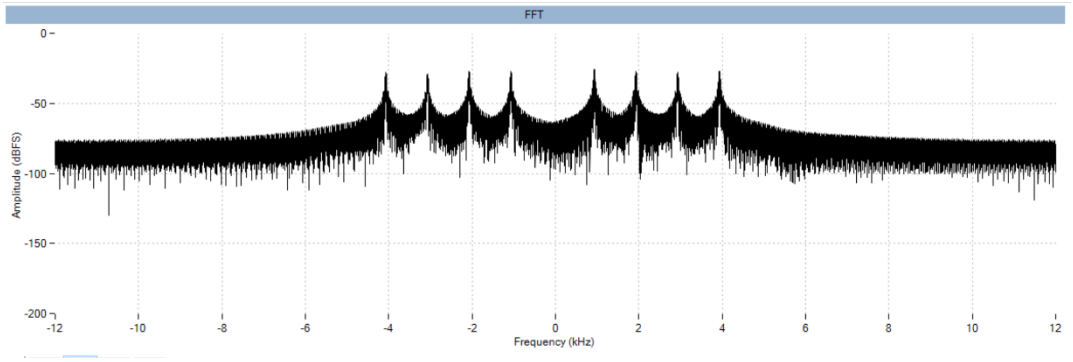


Figure 320. Rx TDD DMR with Frequency Hopping

Tx Only

Steps are the same as Rx Only, except the pre-defined json file must be DMR_24K_TX_ONLY_FH.json

We use 0 Hz tone as an example, here we see the 6 frequencies, the plot is done with max hold.

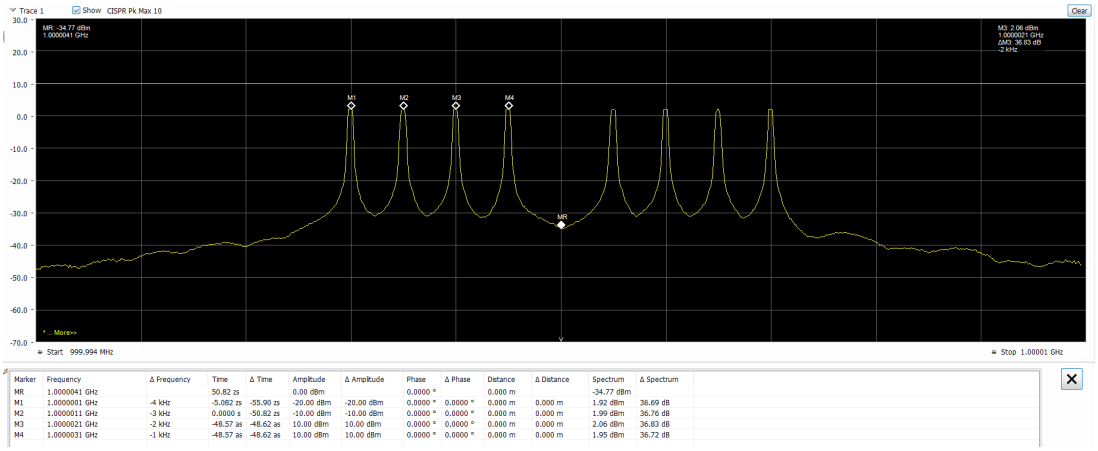


Figure 321. Tx TDD DMR with Frequency Hopping

TRx

Steps are the same as Tx Only or Rx Only, except the pre-defined json file must be DMR_24K_TX_RX_FH.json

In this mode, the frames are interleaved between Tx and Rx. Therefore, from time domain, the Tx spectrum analyzer should show gaps between frames.

OTHER FUNCTIONALITIES

Under **File** menu there are:

- **Save Session** and **Load Session** options which allow save and restore TES configuration parameters,
- **Generate Profile File** provides option to create JSON type profile configuration file.
- **Force Update Platform** provides option to forcibly clean up the resources in the SD card in case of error
- **Shut Down Platform** which allows to safely power down Xilinx platform,
- **Exit** which exits TES software.

In case of some erroneous operation, TES is capable to capture its state. This functionality is provided by means of *Log File* functionality that allows to capture steps that lead to error operation. File created using *Log File* function can be sent back to the ADI support team for further debug.

Programming the Evaluation System

After all tabs are configured, the user must press the ***Program*** button. This starts programming and initialization of an evaluation hardware. The TES sends a series of API commands that are executed by a dedicated Linux application that runs on the Xilinx platform. The user will see a progress bar at the bottom of the window. When programming has completed the system is ready to operate.

IronPython Scripting

IronPython is an implementation of the Python programming language targeting the .NET Framework. The **IronPython** editor is in the **View** menu and allows the user to use IronPython to write a unique sequence of events and then execute them using the ADRV9001 evaluation system.

For the **IronPython** scripting tab to operate, the user must download the Iron Python 2.7 environment. The latest version can be downloaded from the Iron Python website (https://ironpython.net/). After Iron Python is installed, the user must tell TES its installation library path. To set this, open the IronPython editor**,** then select **File** and select **Set IronPython path**. For the default Iron Python installation, this path is set to **C:\Program Files (x86)\IronPython 2.7\Lib.**

Figure 322 shows the **IronPython** editor after executing the **File** > **New** function in the **IronPython Script** tab. The top portion of the window contains IronPython script commands whereas the bottom portion of the window displays the script output.

To use this tab, take the following steps:

1. Scroll to the bottom of the file where there is text that states **#### YOUR CODE GOES HERE ####**
2. This editor will bring up suggestions of all the API functions available when you type "Adrv9001."
3. Information on parameters for different API calls can be found in the doxygen file (*ADRV9001_API.chm*) in the SDK.
4. Go to **IronPython**, select **Build** and then select **Run**. This function executes Iron Python script open in currently active script tab using ADRV9001 evaluation hardware. Script output is displayed in bottom side of the Iron Python script tab.

For this example, the Tx attenuation for the selected channel changes.

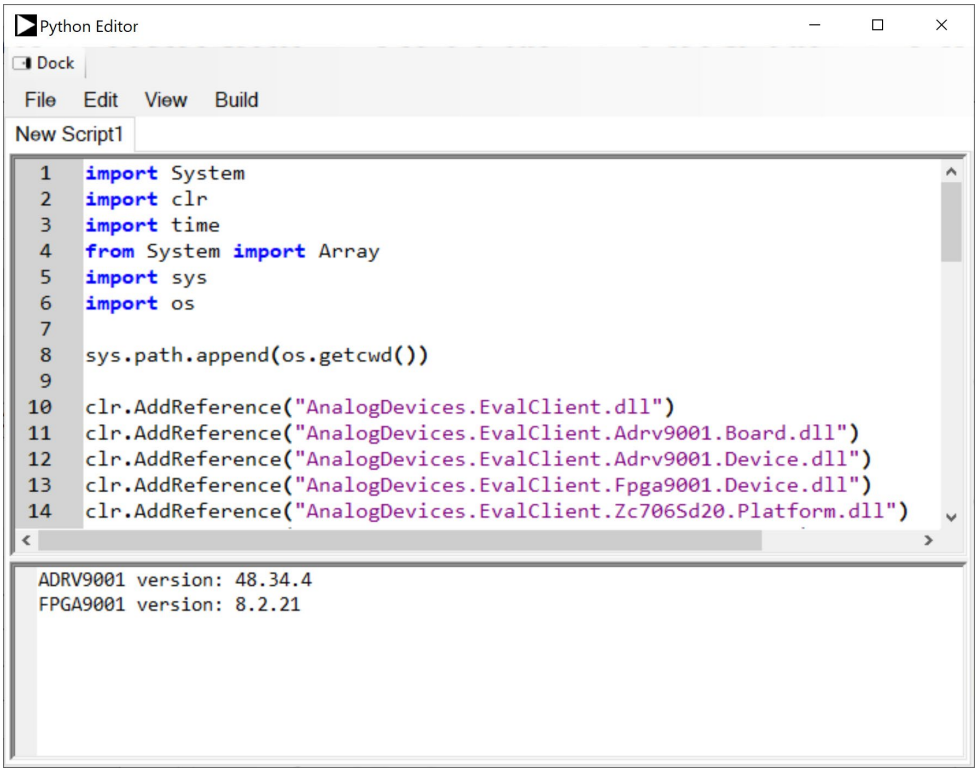


Figure 322. IronPython Scripting Window

There are example python scripts included in the SDK to run API functions that do not appear in the GUI. These files can be found in the *ADRV9001 Transceiver Evaluation Software\IronPython* folder and can be loaded via the *File* menu and *Load*.

External Path Delay Measurement (for DPD)

Connect the Tx1 output, through the chosen PA, to Rx1B input with a cable (an optional step attenuator can be added to the loop). Configure the TES to indicate that the external path after PA exists in the *Board Configuration* tab. The DPD should be enabled from the *Advanced Features* tab. Run the script immediately after programming to make sure the device is in the "Calibrated" state. The state of the part can be checked and changed from the View menu under Radio State.

This script runs the *ExternalPathDelay_Clibrate* cal to get the path delay. It then sets the path delay value and checks it to make sure it was written correctly. The Delay will be displayed in the Output window in ps. This script can be run without the PA using just a cable or cable and attenuator to test the script. For proper use case results the PA will be needed.

Setting the PLL Filter Bandwidth (for Phase Noise Optimization)

To test this script first begin by running a phase noise measurement using the default PLL settings. Stop transmitting on the Tx to load and run the python script. Inside the script, you should only change PLL loop fitter bandwidth. Other PLL parameters should be kept the same. (You can read them, change the bandwidth and then write it back). The change takes effect when the carrier is configured so you need to read the carrier status and then configure it. This can only be performed when device is at calibrated state as shown in the script. Start transmitting on the Tx and measure the phase noise again to observe the difference.

Radio State

Once board is connected, user can view/set radio state. This is under **View->Radio State**. Certain operations in the GUI can set radio to certain states. User should be aware in what state the radio is operating on and control the GUI accordingly. User can also set the radio to certain state from this window.

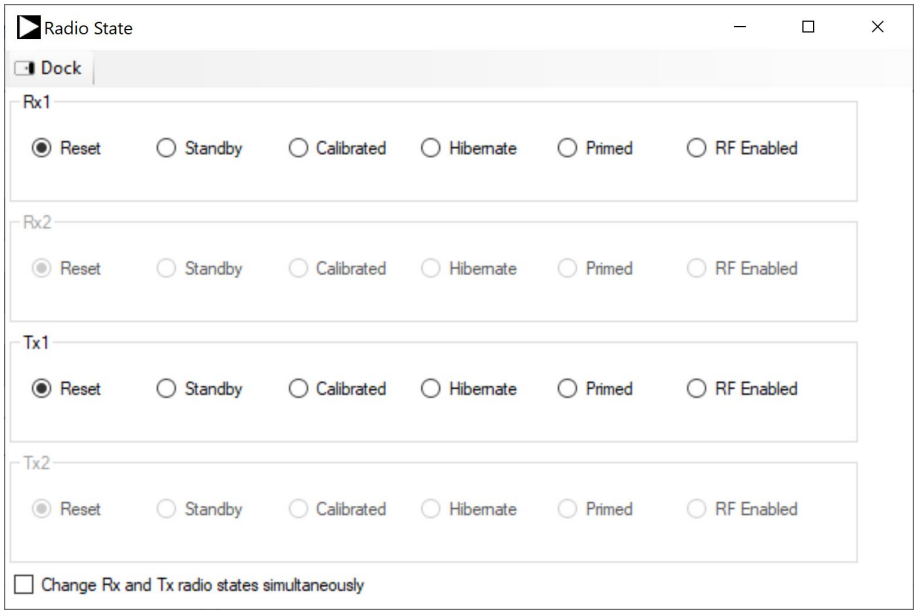


Figure 323. Radio State Window

Power/Temperature Monitoring

The ADRV9001 evaluation software allows user to monitor power usage of the system. On top there is a button **Power / Temp Monitoring**, which shows detailed voltage, current and power status of each power domain. It also shows the temperature from an internal temp sensor. Below is a screenshot of the power monitor window.

Note the VDDA_1P0 power domain is currently not supported by ADRV9001 and its readback should be ignored.

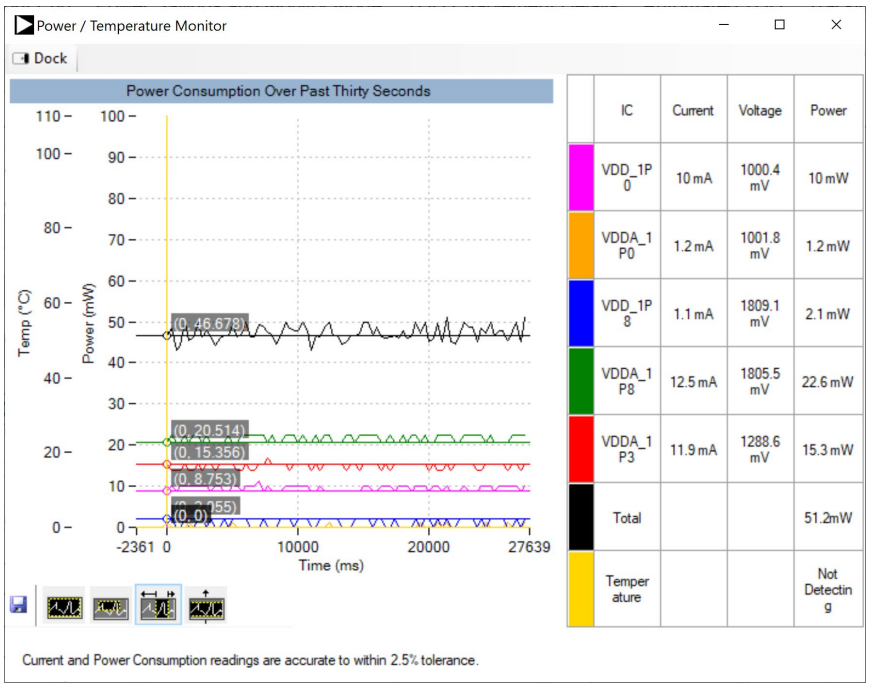


Figure 324. Power / Temperature Monitoring Window

Driver Debugger

A driver debugger is available from the View menu. This is a live window that captures all the driver calls being used by the TES. This can be used for debugging issues or understanding driver calls needed.

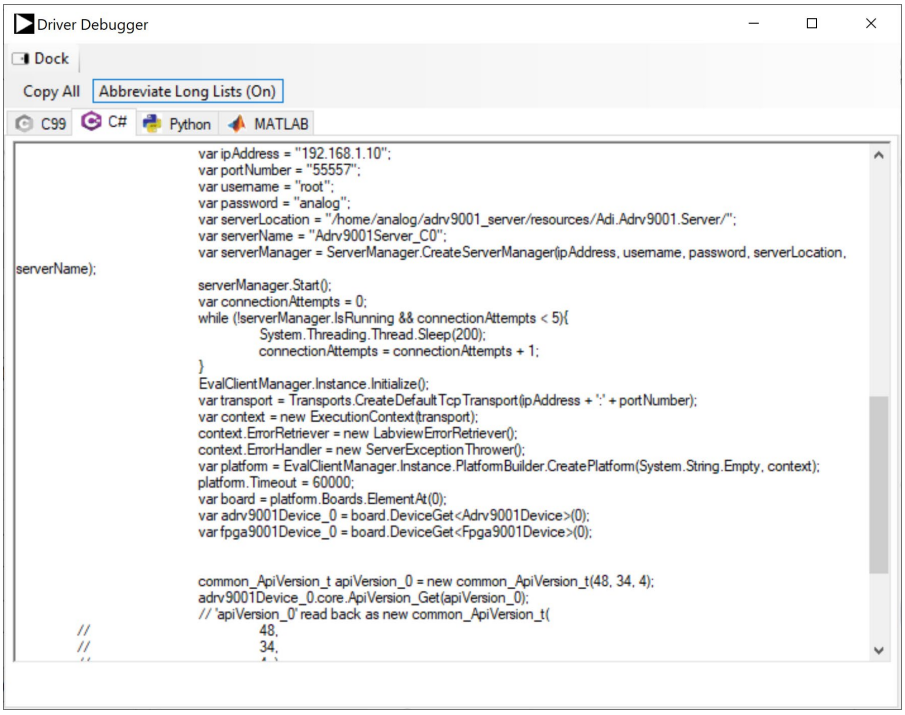


Figure 325. Driver Debugger Window

Log File

On top of the GUI, there is a button **Log File**, which shows logging information of the system. If the PC is connected to the evaluation platform, log file will show the version numbers for different component of the system, including firmware, FPGA, API and so on If errors occur, for example programming the chip fails, log file will provide certain debugging information on what is failing.

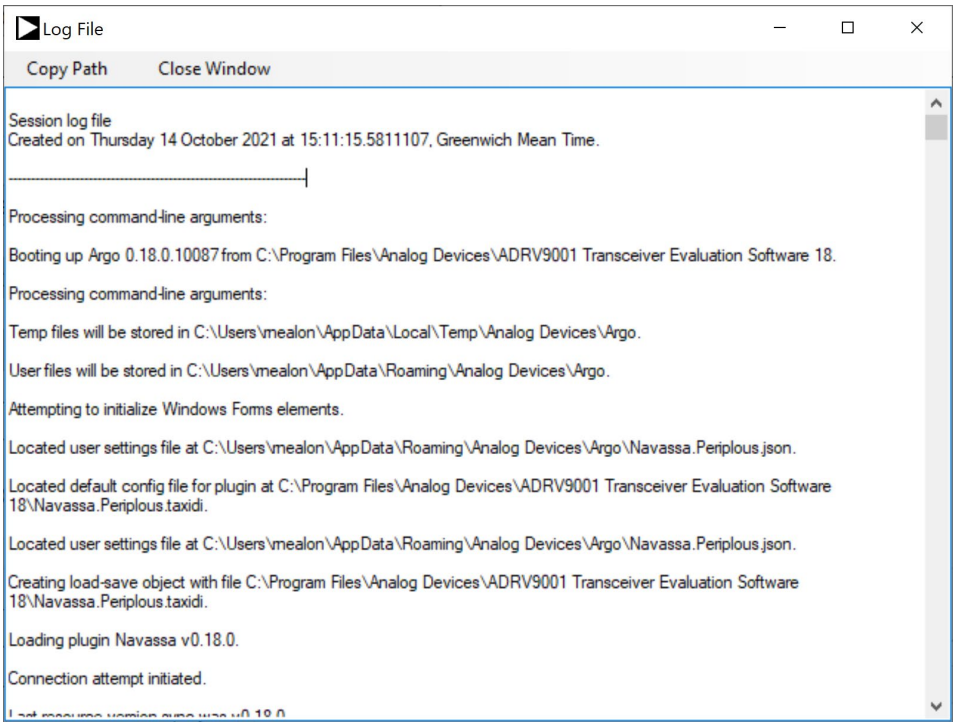


Figure 326. Log File Window

Note that all of these View menu pop-ups can be docked to the main GUI using the 'Dock' button in the menu bar of the pop-up. Users can dock multiple pop-ups at once on the GUI and they will appear in a tabbed format on the right side of the TES. Shown in Figure 327.

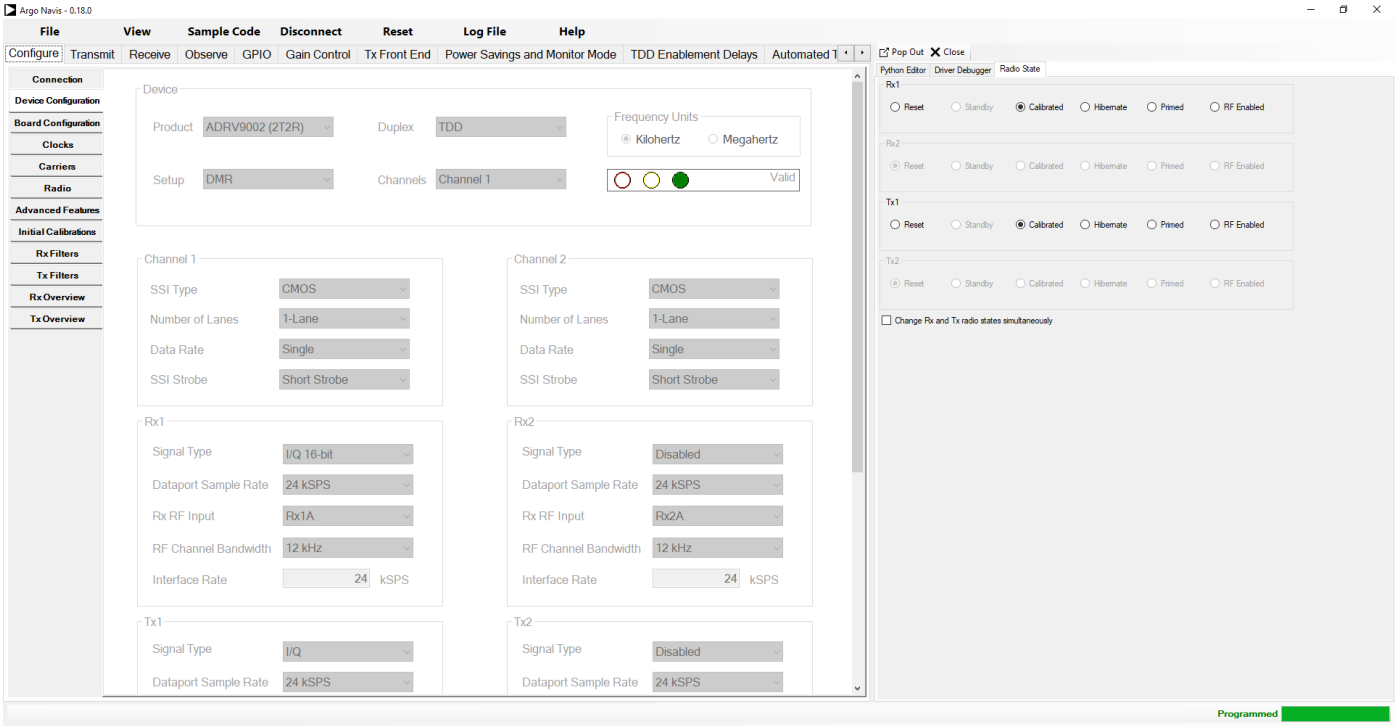


Figure 327. Pop-up options docked in TES

AUTOMATICALLY GENERATE INITIALISATION CODE

User can use Matlab and Python to initialize their system. First connect PC to the part by clicking on **Connect** button. Then configure the system to the desired state. Then program. After program is successful, in the GUI click **Sample Code-> Matlab / Python/ C99/ C#** to generate Matlab, Python, and C initialization code. User can then execute the generate Matlab Python or C code to bring the part to the same desired state as GUI did. Note before executing the generated code from their platform, the user should **Disconnect** the board in the TES GUI.

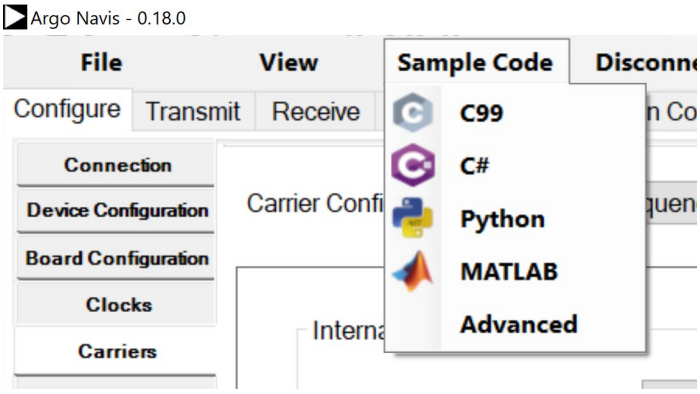


Figure 328. Auto Generated Code Options

EVALUATION SYSTEM TROUBLESHOOTING

The following is a quick help guide describing what to do if the system is not operational. This guide assumes that the user followed instructions and assembled his setup according to hardware configuration described in this document.

TES Connection Issues

1. Ethernet connection (firewall on IP address, port blocked, etc.)
    a. Make sure your firewall settings allow communication to the Xilinx platform.
    b. Check that the IP address for the ethernet is set correctly. With a direct connection to the PC the IP address in the TES should be 192.168.1.10 port 55557 and the LAN connection should be set to 192.168.1.2. With an indirect connection to the PC the IP address should be set to IP address that the router has dynamically assigned to the FPGA.
2. FPGA platform incorrectly configured (refer to user guide for jumper settings)
    a. Check that the switch on the Xilinx platform is set properly to boot from the SD card. You can find these settings here: Hardware Operation
3. SD card not compatible with FPGA platform
    a. There is a different SD card image required for the different Xilinx platforms that are supported. ADI provides SD card imaging software. To find the settings needed revisit SD Card Imaging
4. SD card not compatible with TES version
    a. There is an early image required for all SDK version previous to 0.13.0. For SDK releases 0.13.0 or later a new SD card image is required. This is all taken care of with the ADI SD card imaging SW but should be checked if older versions have been used before.

No LED Activity (ZYNQ ZC706)

1. Check if the board is properly powered. There should be 12V present at the J22 input, and after powering the Xilinx platform on (SW1 turned on) the following should be true:
    a. Fan on the ZYNQ platform is activated. Ensure that fan cable is reconnected to ADRV9001 evaluation platform fan header P702.
    b. A number of green LEDs on the ZYNQ platform near SW1 are ON with no red LEDs active on the ZYNQ platform
    c. ZYNQ GPIO LEDs follow the sequence described in the Hardware Operation section.
    d. Two green LEDs (D801 and D901) on the ADRV9001 evaluation card should be ON.
    2. If the LED sequence does not follow the described one, check jumper settings and SW11 positions on the ZYNQ platform. If these are correct, check if the SD card is correct and properly inserted in the J30 socket. The user should use the SD card supplied with the evaluation kit.
3. If there is still a problem and the user is certain that the ZYNQ platform is operational, contact an ADI representative for help.

LED Active but TES Reports That Hardware is Not Connected

1. Check if the Ethernet cable is properly connected between the PC used to run TES and the Xilinx platform. LEDs on the Xilinx platform next to the Ethernet socket should flash when connection is active.
2. If the cable is properly connected, then check if Windows OS is able to communicate over the Ethernet port with the Xilinx platform. Check if the IP number and open ports for the Ethernet connection used to communicate with the Xilinx platform follow advice described in the Hardware Operation section.
    a. Run cmd.exe and then type: ping 192.168.1.10. The user should be able to see a reply from the Xilinx platform. If no reply is received, connection with the Xilinx platform must be re-examined.
    b. If connection with the Xilinx platform is established but TTES still reports that hardware is not available, ensure that ports number 22 (SSH) and 55557 (Evaluation Software) are not blocked by firewall software on the Ethernet connection used to communicate with the Xilinx platform. Both ports are required to be open for normal operation.
3. Check for physical damage to the EVB.
    a. Look for ferrite bead E803 located on the TOP side of PCB, next to mounting hole. There is a possibility that during transit or when in use nut used to keep PCB stand in place damaged E803. Ensure that E803 is in place with good connection. If E803 got broken, replace it with BLM41PG600SN1L from Murata or similar.

Red LED Constantly On

1. The Xilinx platform generated power domain for IOs that control ADRV9001 over FMC interface. That power domain is called VADJ. For proper operation voltage on that power domain should not exceed 1.89V. The SD card provided with the evaluation card ensures that VADJ is properly set.
   On an Evaluation Card, there is a red LED installed close to the FMC connector. The role of this LED is to indicate if VADJ voltage exceeded 2.0V level. If that was the case this LED will be ON
    a. When the FPGA is powered on the LED will be on for a few seconds as the SD card image readjusts the FPGA VADJ voltage to 1.8 V.
    b. If this LED does not turn off then there is an issue and while the part might still operate this is exceeding the recommended level for VADJ. This will decrease the lifetime of the part and can lead to permanent damage of the IC in the worst case. Possible causes for this LED not turning off are connected to the SD card image.
2. Chip might still operate correctly after this issue but users should understand that VADJ exceeded recommended level. The only way to remove uncertainty here is to change ADRV9001 on an evaluation board to the new one.

Init Calibration Fail

User may experience program failure due to init calibration. This is usually caused by Rx input is connected to the signal generator and RF output is ON. This causes ADRV9001 to interfere with its own internal Rx calibration. User should turn off RF signal during programming.

DLL Bug Fix

If more than one installation of the SDK exists on the computer the TES software may call an incorrect set of DLLs, causing unexpected bugs. Knowing which set of DLLs are called is not feasible, however there is one legacy bug that was fixed in Day 11 which flags this problem. Generate some C99 code in TES and inspect the #include's. In the generated main.c file, the includes should look like this (as of Day 11):

```
#include "initialize.h"
#include "calibrate.h"
#include "configure.h"
#include "prime.h"
#include "beginReceiving.h"
#include "dataCapture.h"
#include "stopReceiving.h"
#include "beginTransmitting.h"
#include "stopTransmitting.h"
#include <stdlib.h>
#include "adi_adrv9001ee01.h"
```

Deviations from this when using the version 11.0 SDK may indicate errors. For example, consider the below examples of cases where the includes have errors:

```
#include "initialize.h"
#include "calibrate.h"
#include "configure.h"
#include "prime.h"
#include "beginReceiving.h"
#include "dataCapture.h"
#include "stopReceiving.h"
#include "beginTransmitting.h"
#include "stopTransmitting.h"
#include <stdlib.h>#include "adi_adrv9001ee01.h"
```

Note the lack of a new line between the last two includes. This may or may not flag an error. Or you may see this:

```
#include "initialize.h"
#include "calibrate.h"
#include "configure.h"
#include "prime.h"
#include "beginReceiving.h"
#include "dataCapture.h"
#include "stopReceiving.h"
#include "beginTransmitting.h"
#include "stopTransmitting.h"
#include <stdlib.h>
#include "adrv9001ee01.h"
```

Now note the lack of "adi_" at the beginning of the final include. This is an old include used in previous revisions of the SDK, however as of version 11.0 this include is no longer valid and attempting it will cause a fatal error during compilation. You may also see these errors in combination with each other or in combination with other innocuous errors that make little sense for generated code, such as this:

```
main.c: In function âmainâ:
main.c:43:2: error: âadrv9001Ce01_Board_device_pointerâ undeclared (first use in this function)
  adrv9001Ce01_Board_device_pointer = (adrv9001Ce01_Board_t*) calloc(1, sizeof(adrv9001Ce01_Board_t));
  ^
main.c:43:2: note: each undeclared identifier is reported only once for each function it appears in
main.c:43:39: error: âadrv9001Ce01_Board_tâ undeclared (first use in this function)
  adrv9001Ce01_Board_device_pointer = (adrv9001Ce01_Board_t*) calloc(1, sizeof(adrv9001Ce01_Board_t));
                                       ^
main.c:43:60: error: expected expression before â)â token
  adrv9001Ce01_Board_device_pointer = (adrv9001Ce01_Board_t*) calloc(1, sizeof(adrv9001Ce01_Board_t));
                                                              ^
main.c:45:2: error: âadrv9001Ee01_Board_device_pointerâ undeclared (first use in this function)
  adrv9001Ee01_Board_device_pointer = (adrv9001Ee01_Board_t*) adrv9001Ce01_Board_device_pointer;
  ^
main.c:45:39: error: âadrv9001Ee01_Board_tâ undeclared (first use in this function)
  adrv9001Ee01_Board_device_pointer = (adrv9001Ee01_Board_t*) adrv9001Ce01_Board_device_pointer;
                                       ^
main.c:45:60: error: expected expression before â)â token
  adrv9001Ee01_Board_device_pointer = (adrv9001Ee01_Board_t*) adrv9001Ce01_Board_device_pointer;
                                                              ^
main.c:47:2: warning: implicit declaration of function âadrv9001Ce01_Createâ [-Wimplicit-function-declara
tion]
  adrv9001Ce01_Create(adrv9001Ce01_Board_device_pointer);
  ^
main.c:49:2: error: âadrv9001_device_pointerâ undeclared (first use in this function)
  adrv9001_device_pointer = adrv9001Ee01_Board_device_pointer->adrv9001Device;
  ^
main.c:51:2: error: âfpga9001_device_pointerâ undeclared (first use in this function)
  fpga9001_device_pointer = adrv9001Ee01_Board_device_pointer->fpga9001Device;
  ^
<builtin>: recipe for target 'main.o' failed
make: *** [main.o] Error 1
root@analog:/home/analog/src/example/test1#
```

Errors and bugs pertaining to implicit declarations, undeclared pointers and missing brackets should be taken as indications of DLL errors in your SDK installation.

These provided examples only discuss the behavior seen in the main.c file should there be confusion between different DLL versions, however odd combinations of include errors (such as above) can be seen in most of the generated c files if this problem exists.

The most effective solution for this is to remove all SDK installations from the drive and reinstall only one valid release of the software.

NOTES

ESD Caution

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

UG24159-10/21(PrC)

ANALOG DEVICES

www.analog.com