Application Report
SBAA443–July 2020

How to Achieve Frequency Hopping With the AFE79xx

Yusuf Agoro

ABSTRACT

This application note describes the NCO-based frequency hopping capability of the RF-sampling AFE79xx device. The AFE79xx is a family of high-performance, wide-bandwidth multi-channel transceivers, integrating four RF sampling transmitter chains, four RF sampling receiver chains, and up to two RF sampling digitizing auxiliary chains (feedback paths). Each receiver chain includes a 25-dB range DSA (Digital Step Attenuator), followed by a 3-GSPS analog-to-digital converter (ADC). Each transmitter chain includes a single or dual digital up converters (DUCs) supporting up to 1200-MHz combined signal bandwidth. The output of the DUCs drives a 12-GSPS digital-to-analog converter (DAC). The feedback path includes an 25-dB range DSA driving a 3-GSPS RF-sampling ADC, followed by a DDC with up to 1200-MHz bandwidth. The AFE79xx improvement in density and flexibility enables high-channel-count, multi-mission systems, and makes these devices a very attractive option for wideband, frequency-hopping applications.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

Frequency hopping describes a method in which communication systems rapidly change the operating frequency for a specific application. Applications such as radar, electronic warfare (EW), and communications use frequency hopping to avoid interference, avoid detection, or find signals that are attempting to remain undetected. The faster these systems can change frequencies, or frequency hop, the more agile these systems become, thus increasing the chance to avoid interference and detection. In a traditional frequency-hopping system, where an analog mixer and PLL or VCO is used as a local oscillator, changing frequencies can take quite a long time. As RF sampling has become more prevalent, frequency hopping is moving toward an NCO-based hopping technique.

2 Phase Coherency vs Phase Continuity

Phase coherency, or phase memory, defines the ability for a synthesizer to maintain phase so that when switching to another source, the original frequency source runs continuously in the background and maintains phase, even when not selected. Therefore, upon returning to the original frequency, the original phase is unaltered. Phase coherency is especially useful in systems where multiple frequency sources use a single reference clock. The overall system may switch sources to reflect the desired frequency source on the RF output, while all other synthesizers run continuously in the background while maintaining phases relative to the reference. Phase-coherent radar systems eliminate the need for recalibration when switching between multiple frequencies because the phase relationship relative to the reference is maintained. Figure 1 shows an example of phase-coherent frequency hopping between NCO0 and NCO1 (programmed to frequencies f1 and f2, respectively), and run continuously, even when not selected.

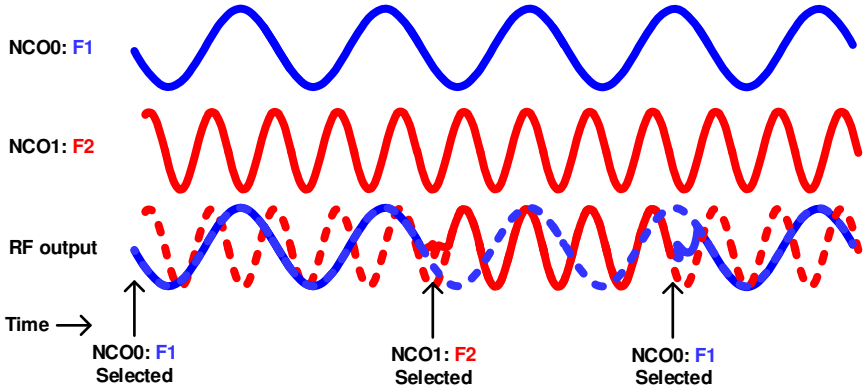

Figure 1. Example of Phase Coherent Frequency Hopping

Phase continuity; however, refers to a smooth and continuous transition from one frequency to another on the RF output when the selected source changes frequencies. Noncontinuous or abrupt transitions in the output frequency may lead to unwanted spurious content during fast Fourier transform (FFT) analysis. Figure 2 shows an example of continuous frequency hopping, where the selected source switches from NCO0 (programmed to F1) to NCO1 (programmed to F2). As Figure 2 illustrates, when frequency hopping from frequency 1 to frequency 2, there is a continuous transition between frequency 1 and frequency 2.

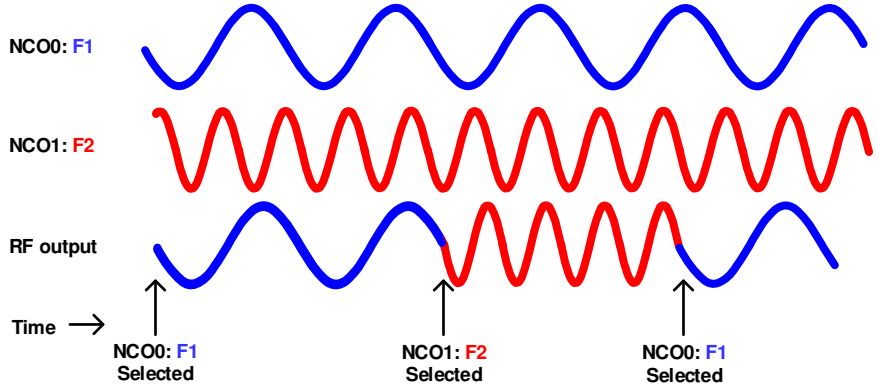

Figure 2. Example of Phase Continuous Frequency Hopping

3 AFE7920 Architecture

The digital mixer within the AFE7920 receivers and transmitters includes two NCOs, whose frequency can be set independently. The mixers can switch between two NCOs, each able to maintain the NCO phase at all times. The switch between the two NCOs can be controlled through the SPI or dedicated GPIO pin. The NCOs have two options for setting the frequency. First, they can have a 32-bit resolution with the frequency specified as the N × sample rate × 1 / 232. Optionally, the NCOs can provide an exact 1-kHz raster for an input reference clock frequency of N × 61.44 MHz, where N is a integer. The NCOs and mixer provide a SFDR of 100 dB.

3.1 AFE7920 Receivers: DDC

Figure 3 shows the AFE7921 and AFE7989 receiver block diagram with a single DDC.

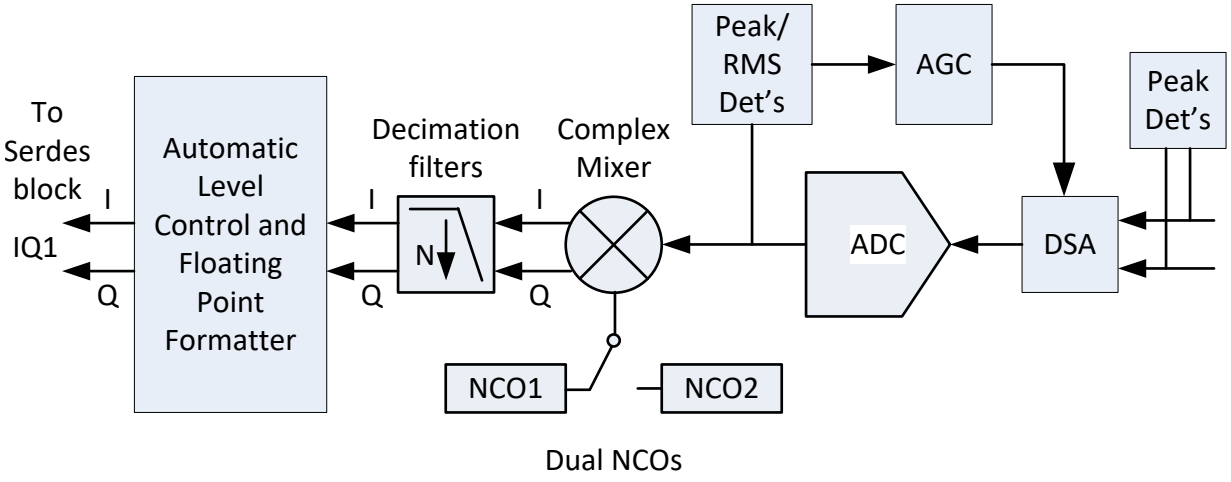

Figure 3. AFE7920 Receiver Single DDC Block Diagram

3.2 AFE7920 Transmitters: DUC

Figure 4 shows the TX chain block diagrams for a single DUC.

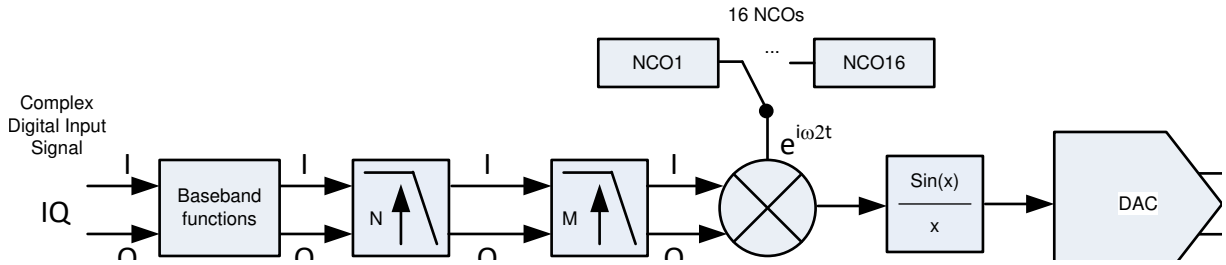

Figure 4. AFE7920 Transmitter Single DUC Block Diagram

4 Numerically Controlled Oscillator (NCO)

The complex digital mixers in the AFE7920 DDC and DUC include digital quadrature modulator (DQM) blocks with independent Numerically Controlled Oscillator or NCOs. The NCOs convert the complex input signal to a real output signal with flexible frequency placement between 0 and f_{DAC} / 2, where f_{DAC} is the DAC sampling clock frequency. The NCOs have a 32-bit frequency accumulator value that generates the sine and cosine terms for the complex mixing. Figure 5 shows the NCO block diagram.

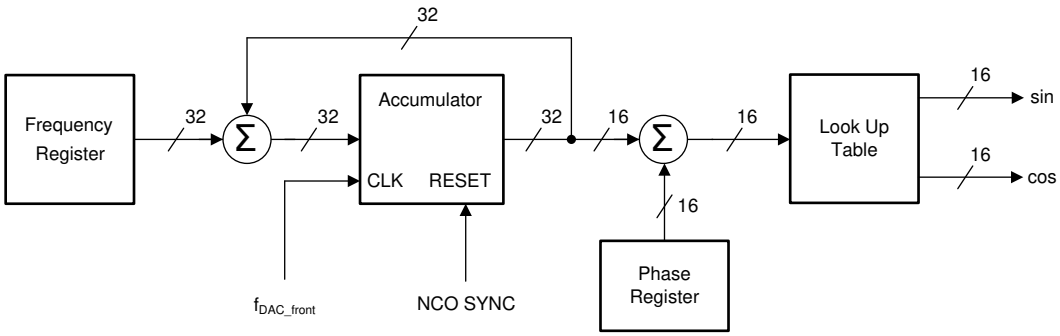

Figure 5. NCO Block Diagram

The NCOs in the AFE7920 receivers and transmitters run continuously, whether currently selected or not, maintaining phase coherency over time.

5 Configuring the AFE7920 for NCO Hopping via GPIO

5.1 Overview of Configurability

The AFE7920 transmitters and receivers come equipped with two NCOs per channel. Of the two NCOs, only one NCO can be selected to actively modulate the RF output at any given time. The GPIO functions RX/TX_NCOSEL_0/1/2/3 control which NCO is actively selected in each channel. The GPIO functions are mapped to any of the available AFE7920 device pins during AFE7920 bring up. This allows for flexibility in the GPIO pin to GPIO function assignment. The state of the GPIO pin assigned to the function TX_NCOSEL_X is what determines which NCO is selected.

The following Latte parameters must be configured properly to execute GPIO based NCO hopping.

NOTE: The associated parameter syntax between the receivers and transmitters are similar with the only difference being whether RX or TX is specified in the functions name.

- ncoTxMode/ncoRxMode
- broadcastTxNcoSel/broadcastRxNcoSel
- numTxNCO/numRxNCO
- TxNco0/RxNco0
- TxNco1/RxNco1
- TX_NCOSEL_[0-3] /RX_NCOSEL_[0-3]

The next section defines the function of each function for the transmitter.

NOTE: The Latte functions specific to the receivers perform the same functions specific to the AFE7920 receivers.

5.2 GPIO Hopping Related Latte Functions and Parameters

5.2.1 TX_NCOSEL_0/1/2/3

There are in total of 4 GPIO functions (TX_NCOSEL_0/1/2/3) for the AFE7920 receivers and transmitters. Each function controls whether NCO0 or NCO1 is selected for the associated TX channel. The NCO that is not selected will still run actively in the background therefore phase coherency is maintained when switching between NCOs. Each GPIO function is then assigned to a one of many available AFE7920 device pins as determined by the user. The device pin selected should be physically connected to a GPIO pin. The state of the GPIO pin connected to the AFE7920 device pin will ultimately determine which NCO is selected within the corresponding channel. During the default bring-up procedure, the AFE7920 is configured such the NCOs within band0 (single band mode) for all four transmitters are the only NCOs enabled for NCO hopping and they are controlled by the GPIO functions TX_NCOSEL_0 and TX_NCOSEL_1. Table 1 shows which channels are controlled by TX_NCOSEL_0 and TX_NCOSEL_1 in the default case.

Table 1. GPIO FUNCTION Control the Default Case

GPIO Function	Default Control
TX_NCOSEL_0	The state of this function determines the active NCO (NCO0 or NCO1) for Band0 in TXA and TXB
TX_NCOSEL_1	The state of this function determines the active NCO (NCO0 or NCO1) for Band0 in TXC and TXD

Figure 6 illustrates how the GPIO pin assigned to the GPIO function TX_NCOSEL_0 will select which NCO is active for channels A and B.

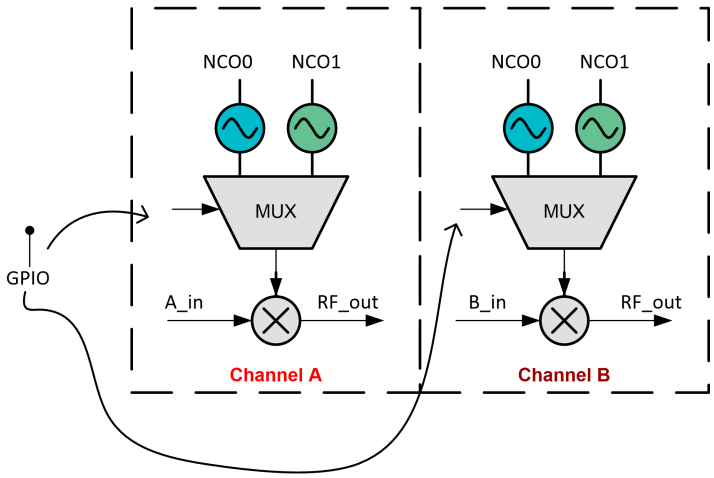

Figure 6. Displaying GPIO Pin Control for Channel A and B in the Default Case

The default case configuration is not mandatory as the mapping of GPIO functions to corresponding NCOs and TX channels is highly configurable with the AFE7920 device. For example, it is possible to configure the AFE7920 to control multiple channels with a single GPIO function. The GPIO function to channel mapping is variably configured depending on how the TX parameters **ncoTxMode** and **broadcastTxNcoSel** are set. The next sections provide details about **ncoTxMode** and **broadcastTxNcoSel** and how they dictate the possible NCO control configurations.

5.2.2 ncoTxMode and broadcastTxNcoSel

The NCOs within AFE7920 transmitters can be configured in various modes based on use case. NcoTxMode and broadcastTxNcoSel are the two parameters that together provide multiple options for how the NCO switching will function within the transmitters. **NcoTxMode** determines whether the corresponding channels are assigned their own unique GPIO function which determines whether NCO0 or NCO1 is selected. NCO is selected or whether the GPIO functions are shared between channels.

When **broadcastTxNcoSel** is set equal to 1, the same GPIO function assigned to Channels A and B is also assigned Channels C and D. Therefore, a single GPIO pin controls the selected NCO for all four channels. By default, broadcastTxNcoSel is set equal to 0.

Table 2 shows how NcoTxMode and broadcastTxNcoSel dictate how the GPIO functions are assigned to control the NCO switching within the associated TX channels.

Table 2. TX NCO Switching Modes to TX_NCOSEL_X Function Mapping

ncoTxMode	broadcastTxNcoSel	TXA GPIO Function	TXB GPIO Function	TXC GPIO Function	TXD GPIO Function
0	x	-	-	-	-
1	0	TX_NCOSEL_0	TX_NCOSEL_0	TX_NCOSEL_1	TX_NCOSEL_1
1	1	TX_NCOSEL_0	TX_NCOSEL_0	TX_NCOSEL_0	TX_NCOSEL_0
2	0	TX_NCOSEL_1: TX_NCOSEL_0	TX_NCOSEL_1: TX_NCOSEL_0	TX_NCOSEL_2: TX_NCOSEL_3	TX_NCOSEL_2: TX_NCOSEL_3
2	1	TX_NCOSEL_1: TX_NCOSEL_0	TX_NCOSEL_1: TX_NCOSEL_0	TX_NCOSEL_1: TX_NCOSEL_0	TX_NCOSEL_1: TX_NCOSEL_0

As Table 2 shows, when ncoTxMode is set equal to 0, NCO switching via GPIO is disabled for both bands. When ncoTxMode is set equal to 1, the state of the GPIO pin assigned to GPIO function TXNCOSEL_0, selects which NCO is active for TxA&B in Band0 and TXNCOSEL_1 selects which NCO is active for TxC&D in Band0. The Same Pins Controls which NCO is active in Band1 if Dual DUC mode is enabled. When ncoTxMode is set equal to 2, the active NCO within each TX channel is determined by state of its own respective GPIO function and GPIO pin. TXNCOSEL_0 for TXA, TXNCOSEL_1 for TXB, TXNCOSEL_2 for TXC, and so on.

Notice that whenever broadcastTxNcoSel is set equal to 1, the same GPIO function assigned to Channels A and B is also assigned Channels C and D, respectively.

The next sections describes the remaining parameters related to configuring the AFE7920 device for NCO hopping.

5.2.3 numTxNCO

numTxNco determines the number of active TX NCOs that are to be used. NumTxNco can be set to either 1 or 2. If no NCO switching is needed, set this to 1.

Default: numTxNCO = 1

5.2.4 TxNco0

TxNco0 effectively programs TX NCO0s operating frequency in MHz for all transmitter channels. NCO0 in both available TX bands are programmed through this function. The NCOs in band1 are only accessible when the AFE7920 transmitter is configured in Dual DUC mode.

Default: txNco0 = [[1800,2600],[1800,2600],[1800,2600],[1800,2600]]

[Band0, Band1] for TxA for NCO0; [Band0, Band1] for TxB for NCO0 [Band0, Band1] for TxC for NCO0 [Band0, Band1] for TxD for NCO0

5.2.5 TxNco1

TxNco1 effectively programs TX NCO1s operating frequency in MHz for all transmitter channels. NCO1 in both available TX bands are programmed through this function. The NCOs in band1 are only accessible when the AFE7920 transmitter is configured in Dual DUC mode.

Default: txNco1 =[[1900,2700],[1900,2700],[1900,2700],[190 0,2700]]

[Band0, Band1] for TxA for NCO1; [Band0, Band1] for TxB for NCO1 [Band0, Band1] for TxC for NCO1 [Band0, Band1] for TxD for NCO1.

5.2.6 TX Parameter Default Summary

Table 3 lists the TX parameter default summary.

Table 3. TX Parameter Default Summary

Parameter	Default Value and Format
ncoTxMode	[0,0]
numTxNCO	1
txNco0	[[1800,2600],[1800,2600],[1800,2600],[1800,2600]]
txNco1	[[1800,2600],[1800,2600],[1800,2600],[1800,2600]]
broadcastTxNcoSel	0

6 Measuring Hop Time

6.1 TX Hop Time via GPIO

6.1.1 Hardware Setup

- Test Equipment
  - Pulse/function generator
  - Oscilloscope
  - AFE7920 EVM

The function generator is configured to produce a low-frequency square wave. The square wave signal is fed into a signal splitter. One end of the splitter is fed to one channel of the oscilloscope while the other output of the splitter is connected to the appropriate GPIO pin on the AFE7920 EVM. The two SMA cables on the split end of the splitter should be length-matched. Channel A of the AFE7920 transmitter is then connected to a separate channel on the oscilloscope. Figure 7 displays the TX hop time via GPIO hardware setup.

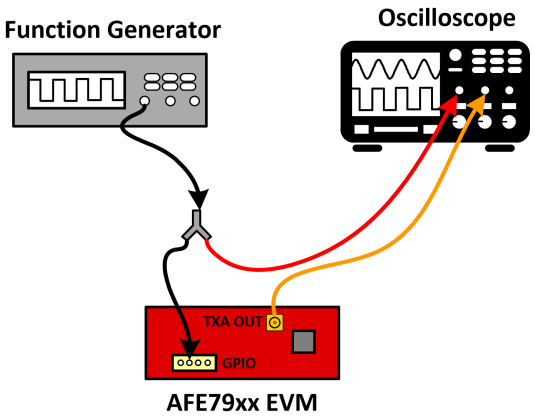

Figure 7. TX Hop Time via GPIO Hardware Setup

6.1.2 Software Configuration

Familiarity with the AFE7920 bring-up procedure is required to program the AFE7920 for frequency hopping. See the *AFE79xx Evaluation Module User's Guide*, if not previously accessed.

6.1.2.1 Setting up Latte for TX GPIO Hopping

To configure the AFE920 for GPIO hopping it is necessary first to set the appropriate parameters within Latte before initiating the bring-up scripts.

6.1.2.2 Programming the TX NCO Frequencies

Program NCO0 and NCO1 in Latte setting sysParams.txNco0. Figure 8 illustrates the following:

- FNCO0 is programmed to 2200 MHz.
- FNCO01 is programmed to 2800 MHz.

```
sysParams.txNco0                    =   [[2200,2200],     #Band0, Band1 for TxA for NCO0
                                        [2199.6,2600],     #Band0, Band1 for TxB for NCO0
                                        [2199.6,2600],     #Band0, Band1 for TxC for NCO0
                                        [2199.6,2600]]     #Band0, Band1 for TxD for NCO0

sysParams.txNco1                    =   [[2800,2800],    #Band0, Band1 for TxA for NCO1
                                        [2219.76,2620],    #Band0, Band1 for TxB for NCO1
                                        [2219.76,2620],    #Band0, Band1 for TxC for NCO1
                                        [2219.76,2620]]    #Band0, Band1 for TxD for NCO1

sysParams.ncoRxMode                 = [2,2]
sysParams.broadcastRxNcoSel         = 0
```

Figure 8. Programming NCO0 and NCO1 for TXA in Latte

6.1.3 Mapping GPIO Functions to AFE7920 Pins

The GPIO functions, TX_NCOSEL_0 and TX_NCOSEL_2 were assigned to the **'L14'** AFE7920 device pin. Figure 9 shows how this was done in Latte by writing the code **'L14': ['TX_NCOSEL_0','TX_NCOSEL_2']** within the Latte function sysParams.gpioMapping. On the AFE7920 device EVM, the "L14" device pin is routed to the "AUX2" GPIO pin located at the J11 pin header. As a result, the selected NCO within TX channel A(TX_NCOSEL_0) and TX channel C (TX_NCOSEL_2) are controlled by the state of the auxiliary 2 GPIO pin. As mentioned previously, the following experiments were performed using TX Channel A only.

```
## The following parameters sets up the GPIOs
sysParams.gpioMapping={
        'H8': 'ADC_SYNC0',
        'H7': 'ADC_SYNC1',
        'N8': 'ADC_SYNC2',
        'N7': 'ADC_SYNC3',
        'H9': 'DAC_SYNC0',
        'G9': 'DAC_SYNC1',
        'N9': 'DAC_SYNC2',
        'P9': 'DAC_SYNC3',
        'P14': 'GLOBAL_PDN',
        'K14': 'FBABTDD',
        'R6': 'FBCDTDD',
        'H15': ['TXATDD','TXBTDD'],
        'V5': ['TXCTDD','TXDTDD'],
        'E7': ['RXATDD','RXBTDD'],
        'R15': ['RXCTDD','RXDTDD'],
        'G16': ['RX_NCOSEL_0','RX_NCOSEL_1'],
        'L14': ['TX_NCOSEL_0','TX_NCOSEL_2'],
#       'H11': ['TX_NCOSEL_0','TX_NCOSEL_2'],
#       'H16': ['FB_NCOSEL_1','FB_NCOSEL_3'],
#       'G12': 'SPIB1_SDO',
#       'H11': 'INTBIPI_SPIB1_SDI',
#       'H16': 'SPIB1_CSN',
#       'G16': 'SPIB1_CLK',
        }
```

Figure 9. Mapping pin L14 to TX_NCOSEL_0 Using Latte Function sysParams.gpioMapping

Figure 10 shows the schematic symbol of the pin header housing the AUX2 GPIO pin. The AUX2 GPIO pin routed to the "L14" AFE7920 device pin.

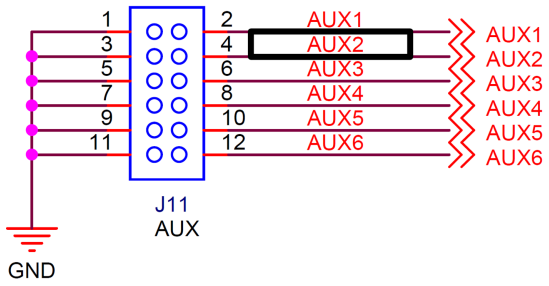

Figure 10. J11 pin Header Housing the AUX2 GPIO pin

6.1.4 Test Procedure and Results

The procedure to capture TX hop time is pretty straightforward. After setting the proper configuration settings for the AFE7920 EVM in Latte, run the bring-up procedure. Once the AFE7920 EVM is up and running, proceed with the following steps.

1. Enable constant TX output tone by issuing the following command in Latte.
   AFE.JESD.DACJESD[0].dacJesdConstantTestPatternValue(1,0,0,1500,1500)

```
200
201 AFE.JESD.DACJESD[0].dacJesdConstantTestPatternValue(1,0,0,1500,1500)
```

Figure 11. Latte Command for Enabling DAC Constant Output Tone

2. Set the function generator to output square pulses at a relatively slow output frequency. 1 Hz should work. Connect one output of the function generator to the NCO hopping GPIO pin, and connect the second output of the function generator the oscilloscope.
3. Connect the TXA output from the AFE7920 to the oscilloscope and program the oscilloscope to trigger on the square wave input. At this point a single pulse should change select the NCO while also triggering the oscilloscope. The change in frequency should reflect on the TXA output and the hop time should now be visible.

Figure 12 shows the oscilloscope shot displaying the captured hop time.

Figure 12. Scope Shot: TXA Hopping From 2200 MHz to 2800 MHz With Hop Time of 217.24 ns

6.2 RX Hop Time via GPIO

6.2.1 Hardware Setup

RX GPIO hop time is calculated using a TSW14J56EVM connected to an AFE7920EVM. The TSW14J56EVM is a data capture card with an input-trigger feature, where data capture begins as soon as the designated SMA input receives a 1.8-V logic high signal. A function generator connects to both the GPIO pin and the SMA trigger input of the TSW14J56EVM. The function generator outputs a single-pulse square wave at 1 Hz with an amplitude of 1.8 V. The signal from the function generator activates the TSW14J56 trigger and switches the selected NCO from NCO0 to RXNCO1. The TSW14J56EVM captures the change in frequency. Figure 13 shows the test setup for this example. A signal generator is used to provide a CW tone input signal to the channel A receiver of the AFE7920 device. The CW tone centered at 2720.40 MHz.

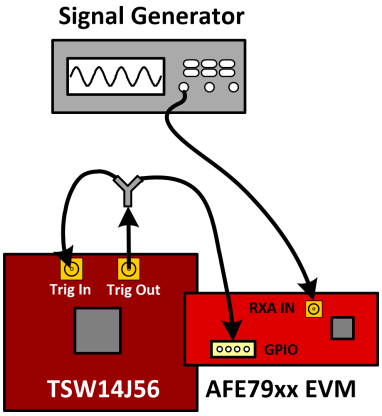

Figure 13. RX hop Time via GPIO Hardware Setup

6.2.2 Software Configuration

The RX GPIO hopping parameters programmed in Latte for RX GPIO hopping are found in Table 4.

Table 4. RX GPIO Hopping Parameters

Parameter	VALUE
ncoRxMode	[1,1]
numRxNCO	2
rxNco0	2600
rxNco1	2700
broadcastxNcoSel	0

6.3 Test Procedure and Results

Upon Latte bring up, RXNCO0 is initially programmed to 2600 MHz. Therefore, when RXNCO0 is active, the 2720.40-MHz input signal will be down converted to a baseband frequency of 120.40 MHz. RXNCO0 is programmed to 2700 MHz so that when the GPIO pin is triggered RXNCO1 becomes the active NCO, the down-converted signal frequency will hop from 120.40 MHz to 20.40 MHz. At the moment the GPIO pin is triggered, the time it takes for the baseband signal to change from 120.40 MHz to 20.40 MHz is considered the RX hop time.

The resultant waveform displays the captured baseband signal hopping from 120.40 MHz to 20.40 MHz.

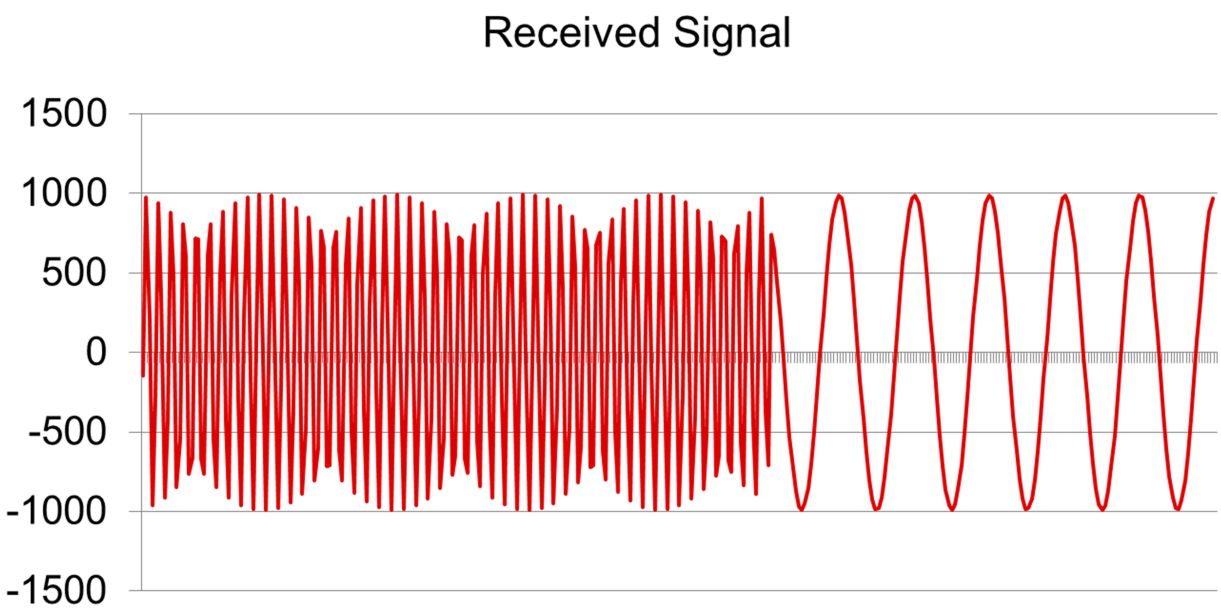

Figure 14. Baseband Signal Hopping From 120.40 MHz to 20.40 MHz

Use Equation 1 to calculate the frequency hop time.

$$\frac{Sample\ \#}{Data\ \ Rate} = \frac{207}{491.52E9} = 421ns \quad (1)$$

The change in frequency occurs at sample 207. The rate at which the TSW14J56 captures data is 491.52 MSPS. Therefore, the RX frequency hop time using GPIO is approximately 421 ns.

IMPORTANT NOTICE AND DISCLAIMER

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

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

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

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