User's Guide
SBAU340–July 2019

AFE77xx Latte GUI

1 Trademarks

2 Introduction

The AFE77xx devices are a family of high-performance, multichannel transceivers that integrate four direct up-conversion transmitter chains, four direct down-conversion receiver chains, and two wideband RF sampling digitizing auxiliary chains (feedback paths).

Latte is a Python-based tool that can be used to generate the register commands to configure AFE77xx in a desired (and supported) mode. These register commands, saved in text format and referred to as configuration (or config) files in the rest of the document, can then be used by the host (FPGA or ASIC) in the application board to bring up the AFE77xx.

Latte can also be used to configure the AFE77xx evaluation module (EVM) when such an EVM is connected to the PC. Latte runs in simulation mode when no EVM is connected.

2.1 Hardware and Software Requirements

A PC with a minimum of 8GB RAM and a processor such as Intel i5 is required. The recommended RAM is 16GB or higher, as the execution time greatly depends on it. Latte is compatible with Windows 7 and 10.

2.2 Installation

Users can download the latest version of Latte from TI's mySecure website, or their Box account. The installer also includes the .INI files for the TSW14J5x board typically used to interface with TI EVMs.

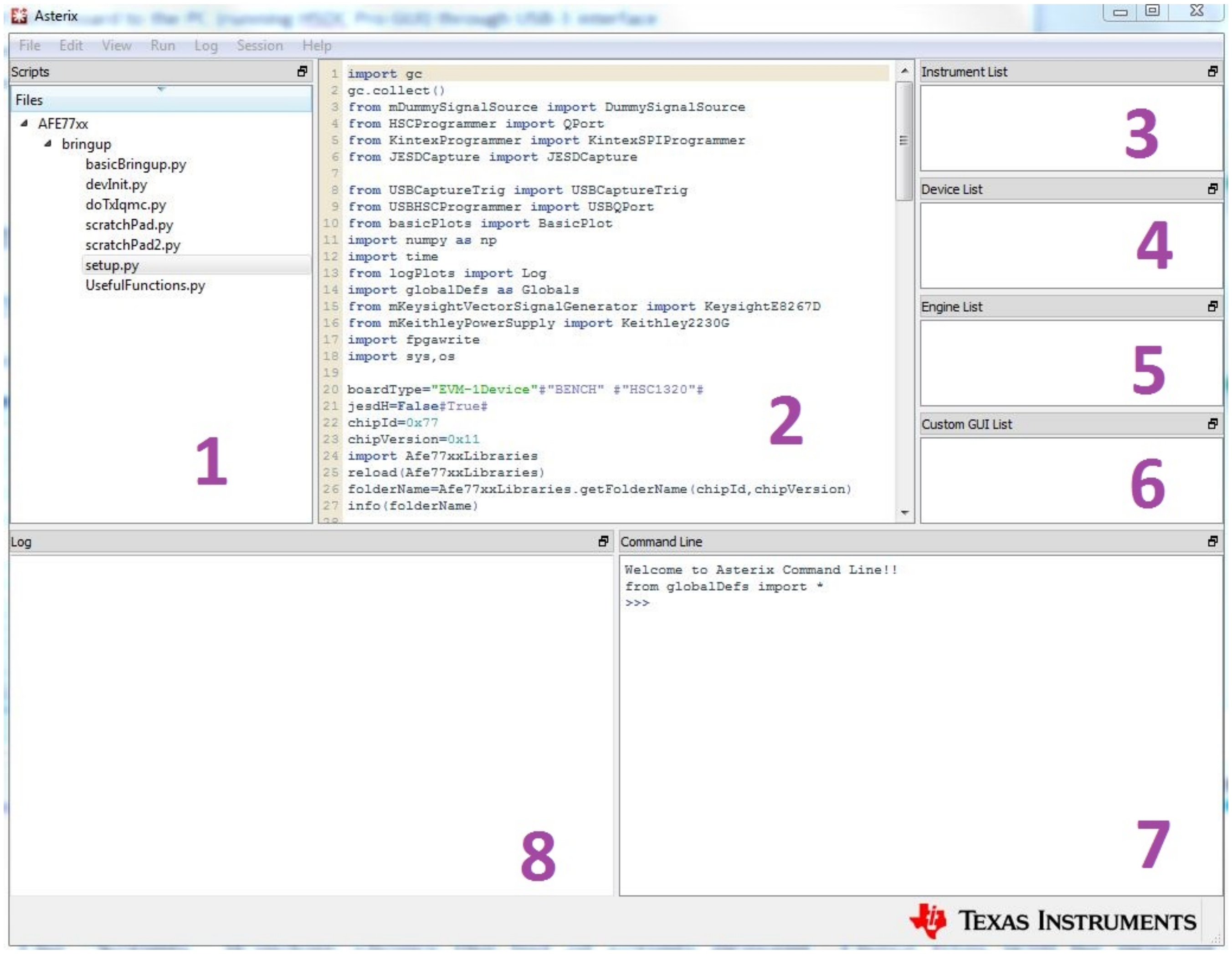
2.3 Outline of This Document

- Section 3 gives an overview of Latte, including a quick start guide to run it in a pre-defined mode.
- Section 4 lists the various parameters used in Latte to define the AFE77xx configuration mode.
- Section 5 describes the GPIO configurablity in AFE77xx.
- Section 6 shows the recommended flow to modify a script and run it in Latte to generate a config file.
- Section 7 describes the various built-in functions in Latte.
- Section 8 shows the configurablity through iGUI.

3 Latte Overview

Launch Latte GUI from the desktop shortcut or from All Programs->Texas Instruments. It will display as shown in Figure 1.

Figure 1. Latte User Interface



```
import gc
gc.collect()
from mDummySignalSource import DummySignalSource
from HSCProgrammer import QPort
from KintexProgrammer import KintexSPIProgrammer
from JESDCapture import JESDCapture

from USBCaptureTrig import USBCaptureTrig
from USBHSCProgrammer import USBQPort
from basicPlots import BasicPlot
import numpy as np
import time
from logPlots import Log
import globalDefs as Globals
from mKeysightVectorSignalGenerator import KeysightE8267D
from mKeithleyPowerSupply import Keithley2230G
import fpgawrite
import sys,os

boardType="EVM-1Device"#"BENCH" #"HSC1320"#
jesdH=False#True#
chipId=0x77
chipVersion=0x11
import Afe77xxLibraries
reload(Afe77xxLibraries)
folderName=Afe77xxLibraries.getFolderName(chipId,chipVersion)
info(folderName)
```

```
Welcome to Asterix Command Line!!
from globalDefs import *
>>>
```

The Latte GUI is split into eight windows (labeled 1 through 8) that have the following functionalities:

- Window 1: This window (also called *Scripts*) shows the list of Python scripts that can be executed. Users can find the pre-defined scripts to configure AFE77xx in this window.
- Window 2: This window (also called *Editor*) shows the code in the script currently selected, and can be used to edit and save the code.
- Window 3 to 6: These windows get updated as the scripts are run, and are for informational purposes.
- Window 7: This window (also called *Command Line*) is used to enter and run individual commands or functions, usually after the initial configuration of the device. Examples of such functionalities include changing LO frequency, DSA settings, and so forth.
- Window 8: This window (also called *Log*) displays messages during script execution to display the current status, errors, and so forth. It is useful for monitoring execution status.

3.1 *Useful Latte Short-Cuts:*

Users may find these short-cuts helpful in carrying out routine operations in Latte.

1. **Run Script file**: A script can be executed by selecting the file in the Scripts window (1) and then pressing F5 (or by selecting *Run* and *Buffer* in the menu bar).
2. **Run portion of a script**: Part of a script can be run by selecting the lines in the Editor window (2) and then pressing F7 (or by selecting *Run* and *Run Selection* in the menu bar).

3. **Stop Execution**: Current execution of a script can be stopped by pressing F10 (or by selecting *Run* and *Stop* in the menu bar).
4. **Clear Session**: A working session in can be cleared by pressing Ctrl+T (or by selecting *Session* and *Clear Session* in the menu bar). This is equivalent to a restart of the Latte GUI and is required after events such as disconnecting or power-cycling the EVM, which require re-establishing the USB connection.

3.2 Latte File Structure and Names

The installation files are stored in a folder named Latte in the \Documents\Texas Instruments\ directory of each user. The files are organized in two folders named "lib" and "projects".

The "lib" folder includes the python scripts that are compiled and run to configure the device. Files and scripts are organized with relevant names. AFE77xx specific files can be found in "...\Afe77xxLibraries\AFE77xxLibraryPG1P1". For example, a file named "mAfeParameters.py" contains the configurable parameters and their typical values.

The "projects" folder contains executable Python scripts that are displayed in the GUI (Window 1 in Figure 1). A few relevant scripts and their purposes are listed below:

- "setup.py": This script checks for connected hardware (such as the AFE77xx EVM) and establishes the USB connection if possible. When no hardware is connected, it enters simulation mode.
- "devInit.py": This script loads the device register map, libraries, and iGUI. iGUI is alternative way to set the device configuration mode through a graphical interface instead of the text-based scripts.
- "basicBringup_y.py": This script includes the parameters and the necessary function calls to configure the device. The "y" in the suffix indicates the configuration mode.
- "Useful_functions.py": This script includes a collection of Python functions useful for dynamic configuration, such as changing the TX DSA.

3.3 Running Latte

This section shows the steps to use Latte to get a desired output.

3.3.1 Establish Connection (setup.py)

In the Scripts window, select *setup.py* and press F5 to run the program. Review the Log window for any errors or messages. When no hardware is connected, a message about Latte being run in Simulation Mode is displayed.

When an EVM is powered and connected to the PC, this step should result in no errors or warnings.

3.3.2 Compile Libraries (devInit.py)

In the Scripts window, select *devInit.py* and press F5 to run the program. Check the Log window for status. No errors or warnings are expected. Close or minimize the iGUI window.

3.3.3 Device Configuration

This step actually configures the device or can be used to generate and save the register commands in text format. In the Scripts window, select *basicBringup_y.py* and press F5 to run the program. For example, running "basicBringup_Case-1.py" configures the device in the mode shown in Table 1.

Table 1. Example Use Case-1

Case	RX	TX	FB	CLK
1	• 245.76 Msps • 24410 • SerDes=9830.4 Mbps • PLL0 • LO=3500 M	• 491.52 Msps • 44210 • SerDes=9830.4 Mbps • PLL0 • LO=3500 M	• 491.52 Msps • 22210 • SerDes=9830.4 Mbps • NCO=3500 M	• FS=2949.12 M • REF=491.52 M

This completes the steps required to generate the config file and EVM configuration (when connected).

4 Latte Parameters

The basicBringup script executed to configure the device consists of various parameters to set the device configuration mode. These parameters, categorized according to the functionalities they address are described in this section.

4.1 System Parameters

System parameters that apply to the general operating conditions of the device are shown in Table 2

- Format: sysParams.<parameter> = "value"
- Example: sysParams.FRef = 491.52

Table 2. System Parameters

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
FRef	Reference frequency to the AFE PLLs. • Type, Unit: Floating number, MHz • Typical value: 491.52
Fs	Sampling rate of data converters in RX, TX,and FB. • Type, Unit: Floating number, MHz • Typical value: 2949.12
halfRateModeRx	Sets RX ADC sampling Rate to half of Fs. • Type: True/False • Typical value: False
halfRateModeFb	Sets FB ADC sampling Rate to half of Fs. • Type: True/False • Typical value: False
halfRateModeTx	Sets TX DAC sampling Rate to half of Fs. • Type: True/False • Typical value: False
enableFbCd	Enables FB2 when set to True • Typical value: True
mode2t2r	Configures mode for TDD enable pins. • 0: Common pin is used for TDD control. That is, one pin for 4TX, one pin for 4RX,and one pin for 2FB. • 1: AB and CD TDDs are enabled through different pins. • 2: AB: TDD, CD: FDD • 3: AB: FDD, CD: TDD Note that if this is non-zero when the pllMuxMode is in the following, some power saving mechanisms are not enabled, resulting in few 100 mW extra of power consumption. • 0: 4T4R Mode with PLL0 as Master • 1: 4T4R Mode with PLL2 as Master • 2: 4T4R FDD Mode • Typical value: 0
pllMuxModes	Sets PLL and LO Distribution mode. • 0: 4T4R Mode with PLL0 as Master • 1: 4T4R Mode with PLL2 as Master • 2: 4T4R FDD Mode (PLL0 for TX and PLL2 for RX) • 3: 2*2T2R FDD Mode: PLL0 TX 1 and 2, PLL2 TX 3 and 4, PLL3 RX 1 and 2, PLL4 RX 3 and 4 • 4: 2T2R FDD - TDD Mode: PLL0 TX 1 and 2, PLL3 RX 1 and 2, PLL2 RX/TX 3 and 4 • Typical value: 0

Table 2. System Parameters (continued)

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
pllLo	Sets carrier (LO) frequencies for relevant PLLs in the following order: [PLL0, PLL1, PLL2, PLL3, PLL4] • PLL1 must be set to the same value as Fs, as the data converter clock is sourced from PLL1. • Type, Unit: Floating number, MHz • Typical value: [3500, 2949.12, 4500.5, 2600.0, 2100.0]
usePllExternalLoClock	Sets External Clock feed mode for AFE77xx pins [1EXTLOIN, 2EXTLOIN]. • The external clock fed into 1EXTLOIN is routed to PLL2, and is referred to as TDD Clock. The external clock fed into 2EXTLOIN is routed to PLL0, and is referred to as FDD Clock. Note that two times the carrier (LO) frequency must be input to the pins. • True: Enables external clock feed. • False: Disables external clock feed. Internal PLL(s) used. • Typical value: [False,False]
JESD204B/C and SerDes Parameters JESD and SerDes blocks in AFE77xx are partioned into two cores where each core services a 2TX_2RX_1FB transceiver chain. The core servicing RX1 and 2, TX1 and 2,and FB1 is referred to as Core-0.The core servicing RX3 and 4, TX3 and 4,and FB2 is referred to as Core-1.	
LMFSHdRx	JESD mode for RX in the following order: [RX1 and 2, RX3 and 4] • Type, Unit: String • Typical value: ["24410","24410"]
LMFSHdFb	JESD mode for RX in the following order: [FB1, FB2] • Type, Unit: String • Typical value: ["22210","22210"]
LMFSHdTx	JESD mode for TX in the following order: [TX1 and 2, TX3 and 4] • Type, Unit: String • Typical value: ["44210","44210"]
systemMode	In TDD mode operation, it can be assumed that RX and FB channels are not enabled concurrently. Therefore, the SerDes lanes carrying the respective digital outputs can be shared between the two. This parameter allows configurablity of lane sharing for [FB1,FB2]. In FDD mode, it is assumed that the RX ADC output and FB ADC output are on dedicated (non-shared) SerDes lanes. • 1: FDD (non-shared or dedicated) • 2: TDD (shared) • Typical value: [1,1] For dedicated lanes in TDD mode, set this parameter to FDD (1), followed by True for the parameter "dedicatedLaneMode".
dedicatedLaneMode	This parameter along with "systemMode" configures SerDes lane sharing mode for [FB1,FB2] in TDD mode of operation. • True: Dedicated lanes for RX and FB if "systemMode" is set to FDD. • False: Shared lanes for RX and FB if "systemMode" is set to TDD. • Typical value: [True,True]
jesdProtocol	Sets JESD204 protocol. • 0: JESD204B • 2: JESD204C 64/66 encoding • 3: JESD204B 64/80 encoding • Typical value: 0
serdesFirmware	Setting it to True enables automatic adaptation for smooth SerDes bring up, SerDes PLL locking, and so forth. It is recommended to be set to True for customer board bring up. • Type: True/False • Typical value: True
jesdTxLaneMux	Sets mapping between JESD and SerDes Lanes on ADC side (AFE JESD TX). • Type: Integer array • Typical value: [0,1,2,3,4,5,6,7] See Section 4.6 for more details.

Table 2. System Parameters (continued)

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
jesdRxLaneMux	Sets mapping between JESD and SerDes Lanes on DAC side (AFE JESD RX). • Type: Integer array • Typical value: [0,1,2,3,4,5,6,7] See Section 4.6 for more details.
jesdRxRbd	Sets RBD value for [Core-0, Core-1]. • Type: Integer • Typical value: [15,15]
jesdScr	Enables JESD scrambler for [Core-0, Core-1].When set to True, the setting applies to both directions. • Type: True/False • Typical value: [True,True]
serdesTxLanePolarity	Sets polarity inversion for each SerDesTx lane in the following order: [STX1,STX2,STX3,STX4,STX5,STX6,STX7,STX8]. • True: Inverts polarity • False: No polarity inversion. • Typical value: [False,False,False,False,True,True,True,True] Polarity for STX5–STX8 is inverted to undo the inversion in the AFE77xx-EVM design.
serdesRxLanePolarity	Sets polarity inversion for each SerDesRx lane in the following order: [SRX1,SRX2,SRX3,SRX4,SRX5,SRX6,SRX7,SRX8] • True: Inverts polarity • False: No polarity inversion • Typical value: [False,False,False,False,True,True,True,True] Polarity for SRX5–SRX8 is inverted to undo the inversion in the AFE77xx-EVM design.
jesdK	Sets JESD parameter K for [Core-0, Core-1]. Setting applies to both directions. • Type: Integer • Typical value: [16,16]
syncLoopBack	Controls whether JESD SYNC is through hardware (pin) or software (SPI). • True: Hardware SYNC • False: Software SYNC • Typical value: True
jesdTxRxABSyncMux	Selects Sync pin for JESD TX Core-0 typically carrying digital outputs from RX1 and RX2. • 0- Pin V5 • 1- Pin U5 • 2- Pin C5 • 3- Pin D5 • Typical value: 0
jesdTxRxCDSyncMux	Selects Sync pin for JESD TX Core-1 typically carrying digital outputs from RX3 and RX4. • 0- Pin V5 • 1- Pin U5 • 2- Pin C5 • 3- Pin D5 • Typical value: 0 For a single JESD link to service all four RX, set the same value for parameters "jesdTxRxABSyncMux" and "jesdTxRxCDSyncMux".
jesdTxFBABSyncMux	Selects Sync pin for JESD TX Core-0 typically carrying digital outputs from FB1. • 0- Pin V5 • 1- Pin U5 • 2- Pin C5 • 3- Pin D5 • Typical value: 0

Table 2. System Parameters (continued)

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
jesdTxFBCDSyncMux	Selects Sync pin for JESD TX Core-1 typically carrying digital outputs from FB2. • 0- Pin V5 • 1- Pin U5 • 2- Pin C5 • 3- Pin D5 • Typical value: 0 For a single JESD link to service both FB, set the same value for parameters "jesdTxFBABSyncMux" & "jesdTxFBCDSyncMux".
jesdRxABSyncMux	Selects Sync pin for JESD RX Core-0 typically carrying digital inputs for TX1 and TX2. • 0- Pin Y5 • 1- Pin W5 • 2- Pin A5 • 3- Pin B5 • Typical value: 0
jesdRxCDSyncMux	Selects Sync pin for JESD RX Core-1 typically carrying digital inputs for TX3 and TX4. • 0- Pin Y5 • 1- Pin W5 • 2- Pin A5 • 3- Pin B5 • Typical value: 0 For a single JESD link to service all four TX, set the same value for parameters "jesdRxABSyncMux" and "jesdRxCDSyncMux".
jesdABLvdsSync	Controls SYNC signal type for Core-0 and applies to both directions (JESD TX and JESD RX). • True: LVDS sync • False: CMOS sync • Typical value: True When set to LVDS type, supported SyncMux values are 0 or 2, as the differential signal is carried by pins corresponding to 1 and 3.
jesdCDLvdsSync	Controls SYNC signal type for Core-1 and applies to both directions (JESD TX and JESD RX). • True: LVDS sync • False: CMOS sync • Typical value: True When set to LVDS type, supported SyncMux values are 0 or 2, as the differential signal is carried by pins corresponding to 1 and 3.
Digital Signal Chain Parameters	
ddcFactorRx	Decimation rates for [RX1 and 2, RX3 and 4] and is typically Fs/Output_data_rate, regardless of halfratemode. • Type: Floating number • Typical value: [12,12] The typical value results in a data rate of 245.76 Msps (2949.12 M/12) for all four RX.
ddcFactorFb	Decimation rates for [FB1, FB2] and is typically Fs/Output_data_rate, regardless of halfratemode. • Type: Floating number • Typical value: [6,6] The typical value results in a data rate of 491.52 Msps (2949.12 M/6) for both FB channels.
ducFactorTx	Interpolation rates for [TX1 and 2, TX3 and 4] and is typically Fs/Output_data_rate, regardless of halfratemode. • Type: Floating number • Typical value: [6,6] The typical value results in a data rate of 491.52 Msps (2949.12 M/6) for all four TX.
fbNco	[FB1,FB2] NCO frequency for Band 0. • Type, Unit: Floating number, MHz • Typical value: [1800,2100]
fbNcoBand1	[FB1,FB2] NCO frequency for Band 1. • Type, Unit: Floating number, MHz • Typical value: [1800,2100]

Table 2. System Parameters (continued)

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
lowIfNcoRx	Low IF NCO frequency for [RX1 and 2,RX3 and 4]. • Type, Unit: Floating number, MHz • Typical value: [0,0]
lowIfNcoFb	Low IF NCO frequency for [FB1,FB2]. • Type, Unit: Floating number, MHz • Typical value: [0,0]
lowIfNcoTx	Low IF NCO frequency for [TX1 and 2,TX3 and 4]. • Type, Unit: Floating number, MHz • Typical value: [0,0]
Calibration Parameters	
Parameter	**Description**
txIqMcCalibMode	Sets number of FB channles used to run TX QMC (Quadrature Mismatch Correction). • 0 - Single FB channel Mode (FB1). QMC in all four TX run using FB1 for loopback. • 1 - Single FB channel Mode (FB2) . QMC in all four TX run using FB1 for loopback. • 2 - Dual FB channel Mode (QMC in TX1 and 2 run using FB1 for loopback; QMC in TX3 and 4 run using FB2 for loopback) • Typical value: 0
rxDsaCalibMode	Configures RX DSA calibration sequence. A CW tone of specific frequency and power is required to be fed into RX to run the calibration and this parameter sets that sequence. • 0 - One channel at a time. Calibration is run on RX1 through RX4 in sequence. • 1 - Two channels at a time. RX 1 and 2 is run in parallel and then RX 3 and 4. • 2 - All 4 RX are run in parallel.
enableRxDsaFactoryCal	Enables RX DSA Factory Calibration when set to True. • Type: True/False • Typical value: False
enableTxDsaFactoryCal	Enables TX DSA Factory Calibration when set to True. • Type: True/False • Typical value: False
enableRxIqmcLolTrackingCorr	Enables RX QMC when set to True. This correction runs continuously without any further user intervention. • Type: True/False • Typical value: True
enableTxIqmcLolTrackingCorr	Enables TX QMC when set to True. This parameter merely sets up the device to be able to run the correction later as required. Correction algorithm occurs through user intervention, typically through a hardware (pin) trigger. Trigger can also be through software (SPI). • Type: True/False • Typical value: True
txIqmcExternalDelayValue	Delay in loopback from TX to FB for [TX1,TX2,TX3,TX4]. These values are typically arrived through characterization of loopback path. When set to 0, the delay within the device (from device characterization) is used. • Typical value: [0,0,0,0]
rxDsaPacketFilePath	Path to save RX DSA calibration data. • Type: String
txDsaPacketFilePath	Path to save TX DSA calibration data. • Type: String
txIqmcFullBandEstimation	Configures TX QMC correction bandwidth. BW is around the carrier (LO) frequency. • True: Correction applicable to 100% of TX BW. • False: Correction limited to 50% of TX BW. • Typical value: False.
jesdLoopbackEn	Enables internal loopback from JESD TX output to JESD RX input when set to True. • Type: True/False • Typical value: False.

Table 2. System Parameters (continued)

Parameter	Description, Type, and Typical Value
Analog Signal Chain Parameters	
useSpiSysref	Configures SYSREF mode. • True: Enables SYSREF through SPI (affects deterministic latency and phase synchronization) • False: Enables SYSREF through Pins (K19,L19) • Typical value: False
txDsaUpdateMode	Configures TX DSA update timing. • 0 - Normal SPI. Value sent through SPI is set instantly. • 1- Through Gain Smoothening. The analog DSA steps in 1 dB steps and digital DSA in 0.125 dB based on a set inter-step time. • 2- Update in TX TDD off state. DSA updated during TX Off state. • Typical value: 0
txDsaGainStepDelay	Controls step time for TX DSA update gain smoothening. Actual delay is 125 ns times the entered value. • Typical value: 5
gpioConfigMode	Configures GPIO mode. • 0: Internal AGC • 1: External AGC • 2: Custom Mode • Typical value: 0 In custom mode, the gpioMapping variable is taken.

4.2 AGC Parameters

Parameters related to AGC functionality are described in Table 3.

- Format: sysParams.agcParamsDict[channel No]['<parameter>'] = "value"
- Example: sysParams. agcRegConfigParams[0]['thresholdSa'] = -3.0

Table 3. AGC Parameters

Parameter	Description
General Parameters	
enableIa	Enables Internal AGC when set to True. • Type: True/False • Typical value: False
gainControls	Configures gain control mode in External AGC mode. • 0-Fast DSA • 1-Pin AGC 8-Pin • 2-Invalid • 3-SPI • 4-Fast DSA • 5-Pin AGC 4-Pin • Typical value: 3
phmOvrEn	Enables PHM and Fast Over Range (FOVR) when set to True. • Type: True/False • Typical value: False
Small Step Attack Digital Detector Parameters	
thresholdSa	Threshold for Small Step Attack. • Type, Unit: Floating number, dBFs • Typical value: -3.0
windowLenSa	Window of time to count number of crossings for Small Step Attack. • Type, Unit: Integer, 8/Fs seconds (Fs=ADC Sampling Rate) • Typical value: 128

Table 3. AGC Parameters (continued)

Parameter	Description
General Parameters	
stepSizeSa	Step Size for Small Step Attack. • Type, Unit: Integer, dB • Typical value: 1
numHitsSa	Number of Hits for Small Step Attack. • Type: Integer • Typical value: 8
enableSa	Enables Small Step Attack when set to True. • Type: True/False • Typical value: True
Big Step Attack Digital Detector Parameters	
thresholdBa	Threshold for Big Step Attack. • Type, Unit: Floating number, dBFs • Typical value: -1.0
windowLenBa	Window of time to count number of crossings for Big Step Attack. • Type, Unit: Integer, 8/Fs seconds (Fs=ADC Sampling Rate) • Typical value: 32
stepSizeBa	Step Size for Big Step Attack. • Type, Unit: Integer, dB • Typical value: 3
numHitsBa	Number of Hits for Big Step Attack. • Type: Integer • Typical value: 8
enableBa	Enables Big Step Attack when set to True. • Type: True/False • Typical value: False
Small Step Decay Digital Detector Parameters	
thresholdSd	Threshold for Small Step Decay. • Type, Unit: Floating number , dBFs • Typical value: -8.0
windowLenSd	Window of time to count number of crossings for Small Step Decay. • Type, Unit: Integer, 8/Fs seconds (Fs=ADC Sampling Rate) • Typical value: 131072
numHitsSd	Number of Hits for Small Step Decay. • Type: Integer • Typical value: 8
stepSizeSd	Step Size for Small Step Decay. • Type, Unit: Integer, dB • Typical value: 1
enableSd	Enables Small Step Decay when set to True. • Type: True/False • Typical value: True
Big Step DecayDigital Detector Parameters	
thresholdBd	Threshold for Big Step Decay. • Type, Unit: Floating number , dBFs • Typical value: -12.0
windowLenBd	Window of time to count number of crossings for Big Step Decay. • Type, Unit: Integer, 8/Fs seconds (Fs=ADC Sampling Rate) • Typical value: 32768
stepSizeBd	Step Size for Big Step Decay. • Type, Unit: Integer, dB • Typical value: 3

Table 3. AGC Parameters (continued)

Parameter	Description
General Parameters	
numHitsBd	Number of Hits for Big Step Decay. • Type: Integer • Typical value: 8
enableBd	Enables the Big Step Decay when set to True. • Type: True/False • Typical value: True
External LNA Control Parameters	
enableEl	Enables External LNA Control when set to True. • Typical value: False
lnabypassGain	Gain of External LNA. • Type, Unit: Floating number, dB • Typical value: 16
gainMargin	Margin to account for gain variation in External LNA. • Type, Unit: Floating number, dB • Typical value: 2
Digital Gain Control (DGC) Parameters	
dgcEnable	Enables DGC when set to True. • Type: True/False • Typical value: False
dgcMode	Configures how digital gain compensation is represented. • 0: IEEE Floating Point Mode after gain distribution • 1: reserved • 2: Coarse Gain indicated in every output sample and is replicated in both I and Q • 3: Coarse Gain distributed in (I,Q) sample together • 4: Coarse Gain sent over ALC pins • 5: Coarse Gain comes as input over ALC pins • 6: Dynamic Coarse Index 16-bit • 7: Dynamic Coarse Index 12-bit • Typical value: 3
coarseIndexBits	Allocation of bits to represent Coarse Index based on chosen dgcMode. • 0: 0 bits for Coarse Index. i.e., all bits are allocated for data. • 1: Invalid • 2: 2 bits for dgcMode (2/3/4/5) • 3: 3 bits for dgcMode (2/4/5) • 4: 4 bits for dgcMode (3) • Typical value: 3
coarseStep	• 0:"0dB" • 1:"1dB" • 2:"2dB" • 3:"3dB" • 4:"4dB" • 5:"5dB" • 6:"6dB" • Typical value: 6
floatingPointMode	• 0: No MSB sent if Exponent>0. • 1: MSB is always sent. • Typical value: 0
floatingPointFormat	• 0: 2-bit Exponent, 13-bit Mantissa, 1-Bit Exponent • 1: 3-bit Exponent, 12-bit Mantissa, 1-Bit Exponent • 2: 4-bit Exponent, 11-bit Mantissa, 1-Bit Exponent • Typical value: 0

4.3 Interrupt Pins Parameters

Built-in alarms in different blocks inside the AFE77xx can be enabled and monitored to route an interrupt signal to two interrupt pins (INT1 and INT2). Alarms that must trigger the interrupt on each pin can be configured through SPI. The list of available alarms is shown in the table named "Alarm List" in the device data sheet. Parameters to configure the interrupts are shown in Table 4. Alarms to be routed to the INT pins can be configured as follows:

- Format: sysParams.intPinsParams[pinNo]['<parameter>'] = "value"
- "pinNo" refers to the interrupt pins as follows: 0 = INT1 (C17), 1 = INT2 (E17)
- Example: sysParams.intPinsParams[0]['JESD'] = True
- Previous example configures JESD alarms to be routed to pin INT1 (C17).
- Multiple alarms can be routed to either or both INT pins. Similarly, the same alarm can be routed to one or both INT pins.

Table 4. Interrupt Pins Parameters

Type of Alarm	Description
JESD	All JESD-related errors, like: • Multiframe alignment error • Frame alignment error • Link configuration error • Elastic buffer overflow • Elastic buffer match error • Code synchronization error • 8b/10b not-in-table code error • 8b/10b disparity error • Frame sync error • Short pattern check • Invalid Header • CRC error • EoEMB error
SPI	Any error related to SPI transaction.
SRTXA	Error generated in PAP block for TX-A.
SRTXB	Error generated in PAP block for TX-B.
SRTXC	Error generated in PAP block for TX-C.
SRTXD	Error generated in PAP block for TX-D.
PLL0	Loss of lock in PLL0
PLL1	Loss of lock in PLL1
PLL2	Loss of lock in PLL2
PLL3	Loss of lock in PLL3
PLL4	Loss of lock in PLL4

4.4 PAP Parameters

In addition to generating interrupts, some of the alarms can be made to trigger the PA protection feature (PAP Triggers) to prevent damages to the power amplifier (PA) from un-conditioned signals or transients. Table 5 lists the parameters to enable and configure the PAP feature in the AFE77xx. PAP is configured for each TX channel.

- Format: sysParams.srConfigParams[chNo]['<parameter>'] = "value"
- chNo refers to the TX channel index (0 for TX1, 1 for TX2, 2 for TX3, 3 for TX4)
- Example: sysParams.srConfigParams[0]['enable'] = True
- Previous example enables PAP in TX1.

Table 5. PAP Parameters

Parameter	Description
GainStepSize	Ramp step size while gaining up. (actualSampleStep=GainStepSize×(last good sample)/1024) • Type, Unit: Integer • Typical value: 30
AttnStepSize	Ramp step size while Ramping down. (actualSampleStep=AttnStepSize×(last good sample)/1024) • Type, Unit: Integer • Typical value: 30
AmplUpdateCycles	Time of each step. • Type, Unit: Floating number, ns. • Typical value: 5.0
threshold	Threshold with respect to full scale. • Type, Unit: Floating number, %. • Typical value: 30.0
enable	Enables PAP functonality when set to True. • Type: True/False • Typical value: False

4.5 LMK Parameters

The AFE77xx EVM comes with a clock chip (LMK04828) to generate the reference and sysref clocks for the AFE77xx PLL and TSW14J5x capture card. Table 6 describes the parameters to configure LMK chip.

Format: LMKParams.pllEn = "value"

Example: LMKParams.pllEn = True

Table 6. LMK Parameters

Parameter	Description
LMKParams.pllEn	Enables LMK PLLs when set to True. • Type: True/False • Typical value: True
LMKParams.inputClk	Frequency of external clock frequency fed to LMK. In this case, the LMK operates in distribution/buffer mode with the PLLs turned off. • Type, Unit: Floating number, MHz
LMKParams.sysrefFreq	SYSREF frequency. • Type, Unit: Floating number, MHz • Typical value: 7.68
LMKParams.lmkFrefClk	Generates and outputs reference and SYSREF clocks when set to True. • Type: True/False • Typical value: True

4.6 JESD and SerDes Overview

This section gives an overview of the JESD and SerDes in the AFE77xx device.

- JESD TX refers to the JESD transmitter in the device (transporting ADC output for RX and FB). This includes the protocol, decoding, and lane mux (lane or lanes to data converter mapping).
- JESD RX refers to the JESD receiver in the device (transporting the input for TX DAC) and includes protocol, decoding, and lane mux.
- SERDES refers to the physical layer carrying the data.
- JESD and SerDes blocks in AFE77xx are partioned into two cores where each core services half of the transceiver chain of the device (2TX_2RX_1FB).

The core servicing RX1 and 2, TX1 and 2,and FB1 is referred to as Core-0. The core servicing RX3 and 4, TX3 and 4,and FB2 is referred to as Core-1. Core-0 includes STX1–STX4 and SRX1–SRX4. Core-1 includes STX5–STX8 and SRX5–SRX8.

Each core can be programmed without any interdependence. That is, SerDes lane rate and LMFS mode in one core can be set differently from the other.

The lane mux feature in AFE77xx allows the flexibility to route the signal chain data to any SerDes lane. This is realized with a configurable mux between the PHY layer (SerDes) and the internal data path (JESD). The device has eight bi-directional JESD lanes and eight bi-directional SerDes lanes. The mapping between the JESD and SerDes lanes can be set through the parameters named "jesdTxLaneMux" and "jesdRxLaneMux", respectively for ADC and DAC.

4.6.1 JESD TX Lane Mux

The ADC output (RX and FB) independent of the mode the device is configured in, is always routed to the JESD lanes in the following order:

- RX1
- RX2
- FB1
- RX3
- RX4
- FB2

The number of lanes each (signal chain) occupies is dependent on the LMFS mode. In the default mode, LMFS for 2_RX and 1_FB are 2441 and 2221 respectively. Therefore, I and Q data from each RX is carried on a single lane. For FB, data from I and Q each occupy a dedicated lane. The signal chain to JESD lane mapping for this case is shown in Table 7.

Table 7. Example JESD Mapping for RX and FB

Signal Chain	LMFS Mode	JESD Lane with Output
RX1&2	2441	• RX1 (I and Q) on 0 • RX2 (I and Q) on 1
FB1	2221	• FB1 (I) on 2 • FB1 (Q) on 3
RX3&4	2441	• RX3 (I and Q) on 4 • RX4 (I and Q) on 5
FB2	2221	• FB2 (I) on 6 • FB2 (Q) on 7

Lane mux allows the user to define the mapping between the JESD and SerDes lanes through a parameter "jesdTxLaneMux" on the ADC side. JESD lane for each SerDes output (STX) is specified in the following order:

- STX1
- STX2
- STX3
- STX4
- STX5
- STX6
- STX7
- STX8

That is, jesdTxLaneMux = [0,1,2,3,4,5,6,7] indicates a straightforward mapping between JESD and SerDes, as shown in Table 8. Note that JESD lanes are 0–7 and STX from 1–8.

Table 8. Example JESDTX Lane Mux Mapping

JESD Lane	SerDes Lane
0 (RX1 I and Q)	STX1
1 (RX2 I and Q)	STX2
2 (FB1 I)	STX3
3 (FB1 Q)	STX4
4 (RX3 I and Q)	STX5
5 (RX4 I and Q)	STX6
6 (FB2 I)	STX7
7 (FB2 Q)	STX8

RX and FB outputs in the example above are assumed to be on dedicated lanes, configured by setting (systemMode=FDD) or (systemMode=TDD and dedicatedLaneMode=True).

In case of shared lane mode (systemMode=TDD and dedicatedLaneMode=False), mapping between signal chain and JESD is altered such that FB1 output is available on JESD lanes which carried data for RX1 and 2. Similarly, FB2 output appears on JESD lanes which carried data for RX3 and 4. Therefore, the mapping shown in Table 7 is altered for shared lane mode, and is shown in Table 9.

Table 9. JESD Mapping for RX and FB in Shared Lane Mode

TDD Slot	Signal Chain	LMFS Mode	JESD Lane with Output
RX on, FB off	RX1 and 2	24410	• RX1 (I and Q) on 0 • RX2 (I and Q) on 1
	RX3 and 4	24410	• RX3 (I and Q) on 4 • RX4 (I and Q) on 5
RX off, FB on	FB1	22420	• FB1 (I) on 0 • FB1 (Q) on 1
	FB2	22420	• FB1 (I) on 4 • FB1 (Q) on 5

Variant: Staying with the same LMFS mode and setting jesdTxLaneMux = [0,1,4,5,2,3,6,7] with dedicated lane mode results in the mapping shown in Table 10. In this case, RX data is sent on lanes in Core-0 and FB on lanes in Core-1. This is necessary in cases where the SerDes lane rate for RX is different from that of FB.

Table 10. Mapping for RX/FB — Variant

Signal Chain	LMFS Mode	JESD Lane with Output	SerDes Lane with Output
RX1 and 2	2441	• RX1 (I and Q) on 0 • RX2 (I and Q) on 1	• RX1 on STX1 • RX2 on STX2
RX3 and 4	2441	• RX3 (I and Q) on 2 • RX4 (I and Q) on 3	• RX3 on STX3 • RX4 on STX4
FB1	2221	• FB1 (I) on 4 • FB1 (Q) on 5	• FB1 (I) on STX5 • FB1 (Q) on STX6
FB2	2221	• FB2 (I) on 6 • FB2 (Q) on 7	• FB2 (I) on STX7 • FB2 (Q) on STX8

4.6.2 JESD RX Lane Mux

The lane mux feature on the JESD RX (DAC side) is similar in principle to that in JESD TX (ADC), with the exception that there is no lane sharing in TX.

DAC input for the respective signal chains, independent of the mode the device is configured in, are routed from the JESD lanes in the following order:

- TX1
- TX2
- TX3
- TX4

The number of lanes each TX occupies depends on the LMFS mode. In the default mode, LMFS for 2_TX is 4421. Therefore, I and Q data for each TX is carried on a dedicated lane. The signal chain to JESD lane mapping for this case is shown in Table 11.

Table 11. Example JESD Mapping for TX (DAC)

Signal Chain	LMFS Mode	JESD Lane with Output
TX1 and 2	4421	• TX1 (I) on 0 • TX1 (Q) on 1 • TX2 (I) on 2 • TX2 (Q) on 3
TX3 and 4	4421	• TX2 (I) on 4 • TX2 (Q) on 5 • TX4 (I) on 6 • TX4 (Q) on 7

Mapping between JESD and SerDes lanes in the DAC side can be configured through the parameter "jesdRxLaneMux". JESD lane for each SerDes input (STR) is specified in the order:

- SRX1
- SRX2
- SRX3
- SRX4
- SRX5
- SRX6
- SRX7
- SRX8

That is, jesdRxLaneMux = [0,1,2,3,4,5,6,7] indicates a straightforward mapping between JESD and SerDes, as shown in Table 12. JESD lanes are 0–7 and STX from 1–8.

Table 12. Example JESDRX Lane Mux Mapping

JESD Lane	SerDes Lane
0 (TX1 I)	STX1
1 (TX1 Q)	STX2
2 (TX2 I)	STX3
3 (TX2 Q)	STX4
4 (TX3 I)	STX5
5 TX3 Q)	STX6
6 (TX4 I)	STX7
7 (TX1 Q)	STX8

5 GPIO Mux

The GPIO pins in AFE77xx can be configured in Latte. The list of supported GPIO Balls are shown below:

['C17', 'C13', 'F18', 'E17', 'E16', 'D15', 'F17', 'E18', 'D5', 'C9', 'E5', 'C10', 'F5', 'D10', 'C6', 'C11', 'D6', 'D11', 'T16', 'V14', 'U16', 'V15', 'Y18', 'W18', 'V16', 'V18', 'C12', 'D14', 'C8', 'B5', 'E13', 'C5', 'C7', 'D12', 'D7', 'A5', 'V6', 'V11', 'U7', 'U13', 'Y5', 'V10', 'U6', 'U11', 'W5', 'V5', 'U17', 'U15', 'R5', 'T5', 'T18', 'U18', 'R18', 'R17', 'V12', 'V9', 'V13', 'U10', 'U5', 'V7', 'U12', 'V8', 'U14', 'T17', 'T13', 'D13']

GPIO assigment in Latte is handled through the assignment of a function to the desired GPIO ball name.

- Format: sysParams.gpioMapping[<ballName>]=’functionName’
- Example: sysParams.gpioMapping['C17']=’rxab_dsa_gain_0’

Mapping of multiple GPIOs can be done as follows:

- sysParams.gpioMapping= {'U11': 'rxa_digpkdet_hth', 'V9':'rxb_digpkdet_hth',}

Supported input functions are shown in Table 13.

Table 13. List of Supported Input Functions

Name	Function Group
spib2_cs_n	SPIB2
spib2_clk	SPIB2
rxab_dsa_gain_0	EXTAGC_GPIOMode1
rxab_dsa_gain_1	EXTAGC_GPIOMode1
rxab_dsa_gain_2	EXTAGC_GPIOMode1
rxab_dsa_gain_3	EXTAGC_GPIOMode1
rxab_dsa_gain_4	EXTAGC_GPIOMode1
rxab_dsa_gain_5	EXTAGC_GPIOMode1
rxab_dsa_gainsel	EXTAGC_GPIOMode1
rxab_dsa_gainlen	EXTAGC_GPIOMode1
rxcd_dsa_gain_0	EXTAGC_GPIOMode1
rxcd_dsa_gain_1	EXTAGC_GPIOMode1
rxcd_dsa_gain_2	EXTAGC_GPIOMode1
rxcd_dsa_gain_3	EXTAGC_GPIOMode1
rxcd_dsa_gain_4	EXTAGC_GPIOMode1
rxcd_dsa_gain_5	EXTAGC_GPIOMode1
rxcd_dsa_gainsel	EXTAGC_GPIOMode1
rxcd_dsa_gainlen	EXTAGC_GPIOMode1
rxa_dsa_gain_0	EXTAGC_GPIOMode2
rxa_dsa_gain_1	EXTAGC_GPIOMode2
rxa_dsa_gain_2	EXTAGC_GPIOMode2
rxb_dsa_gain_0	EXTAGC_GPIOMode2
rxb_dsa_gain_1	EXTAGC_GPIOMode2
rxb_dsa_gain_2	EXTAGC_GPIOMode2
rxc_dsa_gain_0	EXTAGC_GPIOMode2
rxc_dsa_gain_1	EXTAGC_GPIOMode2
rxc_dsa_gain_2	EXTAGC_GPIOMode2
rxd_dsa_gain_0	EXTAGC_GPIOMode2
rxd_dsa_gain_1	EXTAGC_GPIOMode2
rxd_dsa_gain_2	EXTAGC_GPIOMode2
intbipi_spib1_sdi	SPIB1
intbipi_spib2_sdi	SPIB2
spib1_cs_n	SPIB1
spib1_clk	SPIB1

Table 13. List of Supported Input Functions (continued)

Name	Function Group
adc_sync_n_ab_0	
adc_sync_n_ab_1	
adc_sync_n_cd_0	
adc_sync_n_cd_1	
tdd_2t_en_ab	
tdd_2t_en_cd	
tdd_2r_en_ab	
tdd_2r_en_cd	
tdd_1f_en_ab	
tdd_1f_en_cd	
rxa_alc_input_0	INTAGC
rxa_alc_input_1	INTAGC
rxa_alc_input_2	INTAGC
rxb_alc_input_0	INTAGC
rxb_alc_input_1	INTAGC
rxb_alc_input_2	INTAGC
rxc_alc_input_0	INTAGC
rxc_alc_input_1	INTAGC
rxc_alc_input_2	INTAGC
rxd_alc_input_0	INTAGC
rxd_alc_input_1	INTAGC
rxd_alc_input_2	INTAGC
global_pdn	
tx_fb_loop_0	
tx_fb_loop_1	
tx_fb_loop_2	
tx_fb_loop_3	
txiqmc_coeff_update	
fb_nco_sw	
tx1dsa_gain_sw_rxtxlo	
tx2dsa_gain_sw_rxtxlo	
rxdsa_gain_sw	
rxa_alc_input_3	INTAGC
rxb_alc_input_3	INTAGC
rxc_alc_input_3	INTAGC
rxd_alc_input_3	INTAGC
rxa_agc_pin_freeze	INTAGC
rxb_agc_pin_freeze	INTAGC
rxc_agc_pin_freeze	INTAGC
rxd_agc_pin_freeze	INTAGC

Supported output functions are shown in Table 14.

Table 14. List of Output GPIO Functions

Name	Function Group
spib1_sdo	SPIB1
spib2_sdo	SPIB2

Table 14. List of Output GPIO Functions (continued)

Name	Function Group
intbipo_spib1_sdo	SPIB1
intbipo_spib2_sdo	SPIB2
rxa_alc_out_0	INTAGC
rxa_alc_out_1	INTAGC
rxa_alc_out_2	INTAGC
rxb_alc_out_0	INTAGC
rxb_alc_out_1	INTAGC
rxb_alc_out_2	INTAGC
rxc_alc_out_0	INTAGC
rxc_alc_out_1	INTAGC
rxc_alc_out_2	INTAGC
rxd_alc_out_0	INTAGC
rxd_alc_out_1	INTAGC
rxd_alc_out_2	INTAGC
rxa_digpkdet_hth	EXTAGC
rxb_digpkdet_hth	EXTAGC
rxc_digpkdet_hth	EXTAGC
rxd_digpkdet_hth	EXTAGC
rxa_ext_lnabypass	INTAGC
rxb_ext_lnabypass	INTAGC
rxc_ext_lnabypass	INTAGC
rxd_ext_lnabypass	INTAGC
alarm_1	
alarm_2	
dac_sync_n_ab_0	
dac_sync_n_ab_1	
rxa_alc_out_3	INTAGC
rxb_alc_out_3	INTAGC
rxc_alc_out_3	INTAGC
rxd_alc_out_3	INTAGC
dac_sync_n_cd_0	
dac_sync_n_cd_1	
rel_status_a_0	
rel_status_b_0	
rel_status_c_0	
rel_status_d_0	

6 AFE77xx Bringup Flow

The typical programming of an AFE77xx device consists of a set of register commands to bring up the device and static in nature, followed by set of register commands for some dynamic control. Configuring the different signal chains, PLLs, JESD blocks,and so forth falls under the static part, whereas the DSA or carrier (LO) frequency change falls under the dynamic part. The static part of the configuration is defined and known in advance. Therefore, recommend the user is recommended to generate these register commands using Latte instead of sequencing them manually. The dynamic part of the control is handled through C-functions provided separately. The use of validated tool and functions, instead of manual sequencing of register commands, minimizes errors.

The AFE77xx device supports a wide variety of modes, making it impractical to package scripts for all possible modes with the Latte Installer. A set of popular modes are included with the installer for a quick start. It is recommended that the user to modify one of the scripts to meet their application requirement. This section lays out those steps towards.

6.1 Finalize Configuration Parameters

Configuration using Latte entirely relies on user inputs through the various parameters described in Section 4, and it is important that those are set correctly. Use the check list shown in Table 15 to capture the mode the device is to be configured into. Refer to the device data sheet as required.

Table 15. Configuration Data Checklist

Clock Frequencies				
PLL for RX (Enter PLL Number)	RX1:	RX2:	RX3:	RX4:
PLL for TX (PLL Number)	TX1:	TX2:	TX3:	TX4:
LO Freq (MHz)	PLL0:	PLL2:	PLL3:	PLL4:
Data Converter Sampling Rate (Fs in MHz)			PLL1:	
AFE PLL Reference Frequency (MHz)			FRef:	
AFE SYSREF Frequency (MHz)			Fsysref:	
FB NCO Frequency Band 0 (MHz)			FB1:	FB2:
FB NCO Frequency Band 0 (MHz)			FB1:	FB2:
JESD204B/C Settings				
Settings Applicable to Core or Coress			Core-0	Core-1
SYNCb Input Type (CMOs or LVDS)				
JESD Scrambler (True/False)				
K (Frames/Multi-Frame)				
RBD				
systemMode (1 for FDD, 2 for TDD)				
dedicatedLaneMode (0 for shared, 1 otherwise)				
JESD204 Protocol (0 for B, 2 for C)				
LMFS for RX			RX1&2:	RX3&4:
Number of JESD links for Four RX (Enter 1 or 2)				
SYNCb Input Pin or Pins for RX1 and 2 (Enter Ball #)				
SYNCb Input Pin or Pins for RX3 and 4 (Enter Ball #)				
LMFS for FB			FB1:	FB2:
Number of JESD links for two FB (Enter 1 or 2)				
SYNCb Input Pin or Pins for FB1 (Enter Ball Number)				
SYNCb Input Pin or Pins for FB2 (Enter Ball Number)				
LMFS for TX			TX1&2:	TX3&4:
Number of JESD links for Four TX (Enter 1 or 2)				
SYNCb Output Pin or Pins for TX1 and 2 (Enter Ball Number)				
SYNCb Output Pin or Pins for TX3 and 4 (Enter Ball Number)				
SerDes Output Polarity (True for inversion, False otherwise)	STX1:	STX2:	STX3:	STX4:
	STX5:	STX6:	STX7:	STX8:
SerDes Input Polarity (True for inversion, False otherwise)	SRX1:	SRX2:	SRX3:	SRX4:
	SRX5:	SRX6:	SRX7:	SRX8:
SerDes Input (SRX) Lane Assignment for DAC	TX1:	TX2:	TX3:	TX4:

Table 15. Configuration Data Checklist (continued)

Clock Frequencies				
SerDes Input (SRX) Lane Assignment for RX ADC	RX1:	RX2:	RX3:	RX4:
SerDes Input (SRX) Lane Assignment for FB ADC			FB1:	FB2:
ADC Output Data Rates for RX (Msps)			RX1 and 2:	RX3 and 4:
ADC Output Data Rates for FB (Msps)			FB1:	FB2:
ADC Output Data Rates for TX (Msps)			TX1 and 2:	TX3 and 4:
Decimation & Interpolation Factors			Core-0	Core-1
RX Decimation Factor (Fs/rx-adc-output-rate)				
FB Decimation Factor (Fs/fb-adc-output-rate)				
TX Interpolation Factor (Fs/dac-input-rate)				
SerDes Lane Rates (Mbps)			Core-0	Core-1
SerDes Output (STX)				
SerDes Input (SRX)				
Calibration Status				
Enable RX DSA Calibration (True/False)				
# of RX DSA calibrated in parallel				
Enable TX DSA Calibration (True/False)				
FB used for TX DSA Calibration				
Enable RX QMC (True/False)				
Enable TX QMC (True/False)				
FB used for TX QMC				

6.2 *Finalize basicBringup*

Review the basicBringup scripts included with the installer to find the one closest to the desired requirements. Make a copy of it and modify the parameters to suit the use case. The function call *AFE.deviceBringup()* in the basicBringup script starts the device configuration. This function can be found towards the end of the script, and parameters must be set before this function call for them to take effect.

6.3 *Set File Format*

The parameters to set the file format to are shown in Table 16.

Table 16. File Format, Name and Location

Command	Description
logDumpInst.setFileName(ASTERIX_DIR+DEVICES_DIR+r"\config.txt")	• The file name can be modified using this parameter. The default file name is configCustom2.txt • The configuration or log file is saved in "\Documents\Texas Instruments\Latte\lib" by default.
logDumpInst.logFormat=0x04	Generates config file with register commands in the following format: • SPIWrite Addr,Data,lsb,msb • SPIRead Addr,lsb,msb See Section 6.5for detailed description of these keywords.
logDumpInst.rewriteFile	• 1: Overwrites the file • 0: Appends the file • Typical value: 1

6.4 Run basicBringup

The configuration file is generated once the user runs the script as described in Section 3.3. By default, the file is saved in “\Documents\Texas Instruments\Latte\lib” and named "configCustom2.txt". Refer to Section 6.3 to change the default location or name.

The configuration file generated contains the register command sequences, as explained in Section 6.5.

6.5 Commands in Configuration File

This section explains how to interpret the various keywords or comments seen in the configuration file.

6.5.1 SPIWrite

Format: SPIWrite<tab-space><16-bit address>,<8-bit data>,<LSB>,<MSB> //comment

The register writeing to the AFE77xx in some cases requires bit-wise modification rather than acting on the entire register. The LSB and MSB (both inclusive) in the SPIWrite indicate the range of bits that needs to be modified and the address has to be read, modified, and written back. LSB=0 and MSB=7 indicates the entire register, and in this case, data can be written directly without the need to read and modify. Comments added after the first register write may also apply to subsequent writes in cases where multiple registers are needed. All values are in hexadecimal.

Operation for read-modify-write: SPIWrite addr,data,LSB,MSB

1. If LSB = 0 and MSB=7, writeValue=data and go to step v. Else go to step 2.
2. Read the data from address(readValue): addr
3. mask=((1<<(MSB-LSB+1))-1)<<LSB
4. writeValue = (readValue AND (0xff XOR mask)) OR (data AND mask)
5. Write data into ‘addr’.

Example 1:

- SPIWrite 0000,30,7,7 //global_soft_reset=0

Example 2:

- SPIWrite 0107,9f,0,7 //FBDigMixerFreqWord=9fb8e38e
- SPIWrite 0106,b8,0,7
- SPIWrite 0105,e3,0,7
- SPIWrite 0104,8e,0,7

6.5.2 SPIRead

Format: SPIRead<tab-space><16-bit address>,<LSB>,<MSB> //comment

If the same property has multiple register reads, the comment is after the last read. The LSB and MSB corresponds to the property of interest in the read.

Example 1:

- SPIRead 01d5,01,0,0 // Read // pll_reg_spi_a_ack: 0x1(Meaning: access acknowledge)

Example 2:

- SPIRead 00f3,0,7
- SPIRead 00f2,0,7
- SPIRead 00f1,0,7
- SPIRead 00f0,0,7 // Read // cm4_wr_spi_rf0: 0x7

6.5.3 WAIT

Wait time before next register command:

- WAIT <time in seconds>

6.5.4 SPIPoll

When required, the address must be polled until the condition is met.

- SPIPoll <address>,<LSB>,<MSB>,<ExpectedValue>

Bits of the address from LSB to MSB (both inclusive) should be read till it equals ExpectedValue before continuing.

- Operation:ExpectedValue = (readValue>>lsb)&((1<<(msb-lsb+1))-1)
- Example: SPIPoll 00f0,0,0,1

6.5.5 START ... END

Description for a group of register commands are included in between these keywords.

6.5.6 EXTERNAL-ACTION

EXTERNAL-ACTION indicates an action required by the host. Typical scenarios are toggling the reset pin, feeding an external signal,and so forth.

6.6 Macro Calls in Configuration File

The configuration file generated includes Macro calls that simplify and reduce the register commands. These can be identified in the configuration file by searching for the Macro Opcodes and Operands. Opcodes are in addresses 0x190 through 0x193, and operands are in addresses 0xA0 through 0xEF. An example to understand such a macro call is as follows.

```
SPIWrite 00a3,00,0,7
SPIWrite 00a2,03,0,7
SPIWrite 00a1,1f,0,7
SPIWrite 00a0,1f,0,7
SPIWrite 0193,21,0,7
SPIWrite 0192,00,0,7
SPIWrite 0191,00,0,7
SPIWrite 0190,00,0,7
SPIRead 0193,0,7
SPIRead 0192,0,7
SPIRead 0191,0,7
SPIRead 0190,0,7  \\Read    macro_opcode_operand=0x0;     Address(0x190[31:0],) SPIRead 00f3,0,7
SPIRead 00f2,0,7
SPIRead 00f1,0,7
SPIRead 00f0,0,7  \\Read    cm4_wr_spi_rf0=0x0;     Address(0xf0[31:0],)
```

The operands are written first, and the number of operand registers written depend on the specifics of the Macro called. In the previous example, only four registers (0xa0–0xa3) are needed for this macro. Once the opcodes are loaded, the macro is executed by writing to the operand register, 0x193 in this case. The subsequent writes to 0x192–0x190 are not strictly required, as the macro execution is initiated with the write to 0x193. A successful macro execution results in the reset of operand registers, and that is verified by the next four register read back (SPIRead 0x193–0x190)

After a macro execution, the status (including errors) can be read from registers 0xf0–0xf3. Extended errors (0xf2, 0xf3) are for DSA calibration macros.

Table 17. Macro Status and Error Registers

Extended Error Status		Opcode	Error	Status
0xf3<7:0>	0xf2<7:0>	0xf1<7:0>	0xf0<7:4>	0xf0<3:0>

Table 18. Register 0xf0 - Status and Error Codes

0xf0<0>	MACRO_READY
0xf0<1>	MACRO_ACK
0xf0<2>	MACRO_DONE

Table 18. Register 0xf0 - Status and Error Codes (continued)

0xf0<3>	MACRO_ERROR
0xf0<4>	MACRO_ERROR_IN_OPCODE
0xf0<5>	RESERVED
0xf0<6>	MACRO_ERROR_IN_OPERAND
0xf0<7>	MACRO_ERROR_IN_EXECUTION

Table 19. Register 0xf2 - Extended Status and Error Codes

0xf2<7:0>	Code
0x0	Calibration OK
0x1	CMD_ERROR (wrong calibration command)
0x2	DSA_RELIABILITY_HIT
0x3	DSA_FREQ_WRONG (calibrtaion tone not in band)
0x4	DSA_PWR_ERROR (calibrtaion tone power level too low)
0x5	DSA_SNR_ERROR (Insufficent SNR to run calibration)
0x6	SYSCALIB_TIMEOUT (FFT done flag not set)
0x7	SIG_INVALID (invalid signal)
0x8	DSA_MULTITONE_ERROR (tone not in bin or spur level too high)
0x9	GAIN_STEP_ERROR
0xa	DVGA_GAIN_OVERFLOW (observed power outside DVGA correction range)
0xb	PKT_ERROR (corrupt DSA packet)
0xc	DSA_PHASE_ERROR (phase change per DSA step too large (>40°))

7 Access Latte Functions

Since Latte is a collection of Python scripts, functions in these scripts can be executed in the Latte Command Line (Window 7 shown in Figure 1) to accomplish different tasks. These functions are categorized according to the blocks they address, and are listed in this section.

7.1 General Functions

Table 20. General Functions in Latte

Function Name	Description
AFE.deviceBringup()	Starts AFE configuration based on the set parameter values. Therefore, any parameter to be set should precede this function call.
AFE.softReset()	Resets the device completely.
AFE.readReg(address)	Returns value in the register.
AFE.writeReg(address,value)	Writes value to the register specified.
AFE.serdesRead(address)	Returns value in the SerDes register.
AFE.serdesWrite(address,value)	Writes value to the SerDes register specified.
AFE.sysrefCheck()	Checks if SYSREF reached different blocks in the device.

7.2 PLL Functions

These functions can be called to modify the operational status of PLLs in the device. The PLLs are indexed from zero through five, and each PLL can be addressed by appropriately addressing it using the index "n" shown in Table 21.

Table 21. PLL Functions in Latte

Function Name	Description
AFE.PLL[n].configurePll()	Configures PLL based on inputs for systemParams.FRef and systemParams.pllLo[n].
AFE.PLL[n].powerUpPll(en)	• 0: Disables PLL[n] • 1: Enables PLL[n]
AFE.PLL[n].freeRunningVco(en)	en=1 disables PLL control loop resulting in free running VCO.
AFE.PLL[n].pllOpOntoPin(en)	en=1 routes PLL[n]'s output to pin.

7.3 JESD[TX] Functions

These functions address the JESD TX block in the device.

Table 22. JESD TX Functions in Latte

Function Name	Description
AFE.JESDTX[n].toggleSync()	Toggles SPI sync override.
AFE.JESDTX[n].clearDataAlarms()	Clears the data and alarms.
AFE.adcDacSync()	Issues sysref (within the device) and resyncs both ADC and DAC.
AFE.JESD.SUBCHIP.configJesdTxSyncMux()	Configures all JESD TX Syncs to CMOS, SYNC In 0.

7.4 JESD[RX] Functions

These functions address the JESD RX block in the device.

Table 23. JESD RX Functions in Latte

Function Name	Description
AFE.JESDRX[n].getJesdAlarms(clearAlarms)	Reads JESD alarms. If clearAlarms=1, alarms are cleared before reading. Else, alarms are read without clearing them.
AFE.adcDacSync()	Issues sysref (within the device) and resyncs both ADC and DAC.
AFE.JESDRX[n].clearDataAlarms()	Clears Data and Alarms
AFE.JESDRX[n].enableTestPattern(en, power)	• En=1 enables test Pattern. • Power= dbFs×32767
AFE.JESD.SUBCHIP.configJesdRxSyncMux()	Configures all JESD RX Syncs to CMOS, SYNC Out 0.

7.5 SerDes Functions

Table 24. SerDes Functions in Latte

Function Name	Description
AFE.SERDES[n].serdes1010Pattern(laneEna)	Outputs "1010" pattern for the lane addressed. Index "n" is used to address the specific SerDes core as described in Table 2. Lane in each SerDes core is enabled based on the binary value of "laneEna". That is, n=0 and laneEna=5 (0101) results in such pattern from lanes 0 and 2 in SerDes Core-0 (STX1 and STX3).
AFE.SERDES[n].serdesPrbsPattern(laneEna, PRBSPattern)	Outputs PRBS pattern for the lane addressed. PRBSPattern: • 0: =PRBS7 • 1: PRBS15 • 2: PRBS23 • 3: PRBS31
AFE.SERDES[n].serdesSendData(laneEna)	Sends Data on the selected lanes.

Table 24. SerDes Functions in Latte (continued)

Function Name	Description
AFE.SERDES[n].serdesPdn(pdn)	Powers down the selected SerDes core when pdn=1
AFE.JESD.SUBCHIP.serdesFirmwareLoad(filename)	Loads SERDES firmware from the path specified.
AFE.SERDES[n].enableRxToTxLoopback(en)	Loops back the RX data received onto TX lanes.
AFE.SERDES[n].getSerdesEye()	Gets and plots the SERDES eye.

7.6 Calibration Functions

Table 25. Calibration Functions in Latte

Function Name	Description
AFE.TOP.freezeTxIqmcEstim()	Freezes TX QMC Estimator.
AFE.TOP.freezeTxLoEstim()	Freezes TX LO Estimator.
AFE.TOP.resetTxIqmcLo()	Resets TX QMC and LO estimator and corrector.
AFE.TOP.TIMINGCTRL.txToFbSelectCh (overrideEn, channelSelect=0)	This function initiates TX QMC algorithm. • overrideEn=1: Overrides Pin status. • channelSelect (single FB mode): • 0: TX-1 looped back into FB-1 • 1: TX-2 looped back into FB-1 • 2: TX-3 looped back into FB-1 • 3: TX-4 looped back into FB-1 • 4: No loopback from TX to FB-1 channelSelect (dual FB mode): • 0: TX-1 looped back into FB-1 and TX-3 looped back into FB-2 • 1: TX-2 looped back into FB-1 and TX-4 looped back into FB-2 • 2,3: No loopback from TX to FB
AFE.RX.freezeRxIqmc(freeze)	freeze: • True: Freezes RX QMC • False: Un-freezes RX QMC
AFE.RX.freezeRxLol(freeze)	• True: Freezes RX LOLC • False: Un-freezes RX LOLC

7.7 Miscellaneous Functions

Table 26. Miscellaneous Functions in Latte

Function Name	Description
AFE.configPllMux()	Configures the PLL and LO distribution path based on inputs for systemParams.pllMuxModes.
AFE.setFbNcoWord(chNo,ncoFreq)	Sets NCO in FB. ncoFreq is in MHz. • chNo=0: FB-1 • chNo=1: FB-2
AFE.TOP.powerDownConfigFb(chNo)	Powers down FB channel based on binary value of chNo. • Bit<0> for FB-1 • Bit<1> for FB-2 A "1" for the bit powers down the corresponding channel.

Table 26. Miscellaneous Functions in Latte (continued)

Function Name	Description
AFE.TOP.powerDownConfigRx(chNo)	Powers down RX channel based on binary value of chNo. • Bit<0> for RX-1 • Bit<1> for RX-2 • Bit<2> for RX-3 • Bit<3> for RX-4 A "1" for the bit powers down the corresponding channel.
AFE.TOP.powerDownConfigTx(chNo)	Powers down TX channel based on binary value of chNo. • Bit<0> for RX-1 • Bit<1> for RX-2 • Bit<2> for RX-3 • Bit<3> for RX-4 A "1" for the bit powers down the corresponding channel.
AFE.TOP.overrideTdd(rx,fb,tx)	This function is used to simulate TDD On/Off control usually performed through pins. • "1" enables the corresponding chain • "0" disables it regardless of pin status That is, AFE.TOP.overrideTdd(1,0,0) enables RX and disables TX and FB.
AFE.TOP.disableOverrideTdd()	Switches control back to pins.
AFE.TOP.DSA[n] setTxDsa(chNo,dsaSetting)	Sets TX DSA for channel based on "n" and "chNo". n, chNo: • 0,0: TX-1 • 0,1: TX-2 • 1,0: TX-3 • 1,1: TX-4
AFE.TOP.DSA[n] setRxDsa(chNo,dsaSetting)	Sets RX DSA for channel based on "n" and "chNo" with same indexing scheme as that for TX.
AFE.TOP.DSA[n] setFbDsa(chNo,dsaSetting)	Sets FB DSA for channel based on "n" and "chNo" with same indexing scheme as that for TX.

7.8 Additional Latte Functions

A user can access additional functions that can be found in the scripts Latte library (...\Documents\Texas Instruments\Latte\lib). Such functions include adding an appropriate prefix. Table 27 shows the prefix needed to call such functions based on the files they are found in.

As an example, a function named "jesdSwing" defined in "mAfeLibrary.py" can be run to set the amplitude of (all) STX (SerDes) outputs. The desired amplitude is passed as an argument to the function. The proper syntax for such a function call is AFE.jesdSwing(3) where "3" is the amplitude setting.

Table 27. Prefix to Call Latte Functions

File Name	Prefix
mAfeBaseClass.py, mAfeLibrary.py, mHWAccess.py	AFE.
mDacDig.py	AFE.TX.DACDIG[topNo].
mDsaController.py	AFE.TOP.DSA[topNo].
mGpio.py	AFE.TOP.GPIO.
mJesd.py	AFE.JESD.
mJesdRx.py	AFE.JESD.JESDRX[topNo].
mJesdSerdes.py	AFE.JESD.SERDES[topNo].
mJesdSubchip.py	AFE.JESD.SUBCHIP.
mJesdTx.py	AFE.JESD.JESDTX[topNo].
mPll.py	AFE.TOP.PLL.
mRxAgc.py	AFE.RX.RXDIG[topNo].AGCDGC[chNo].

Table 27. Prefix to Call Latte Functions (continued)

File Name	Prefix
mRxDig.py	AFE.RX.RXDIG[topNo].
mRxLib.py	AFE.RX.
mTimingControl.py	AFE.TOP.TIMINGCTRL.
mTop.py	AFE.TOP.

8 iGUI

Latte includes an option to configure the device through a graphical interface called iGUI. This interface allows only a subset of the parameters shown in Section 4 to be configured. The recommended use of Latte is through the use of scripts, and use iGUI only if needed. The overview of iGUI is shown in Figure 2.

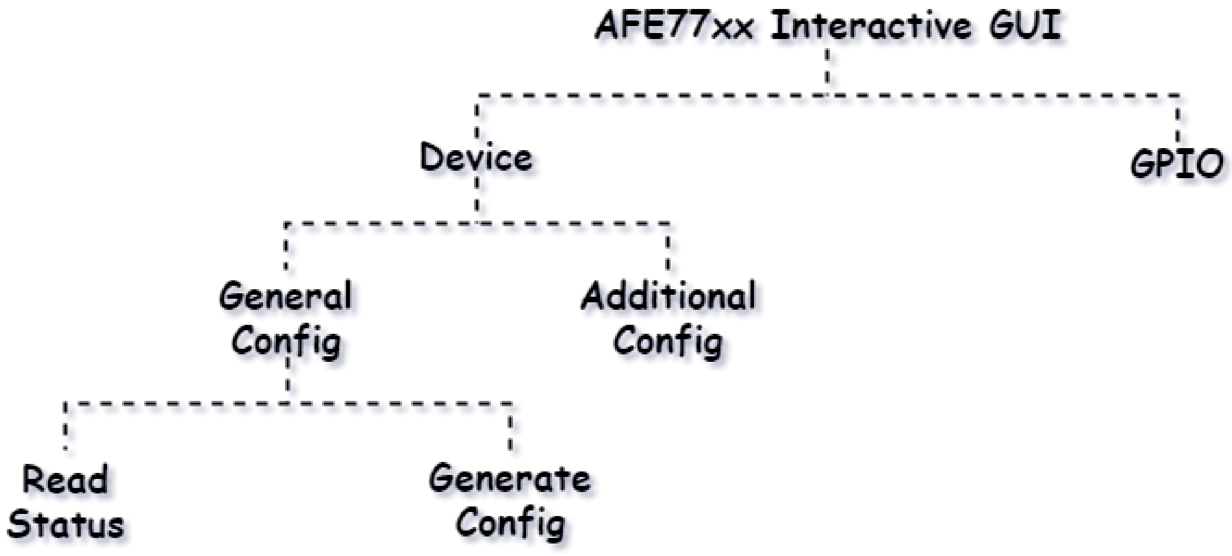


Figure 2. AFE77xx iGUI Overview

8.1 General Config

iGUI starts as shown in Figure 3 after a successful run of the *devInit.py* script.

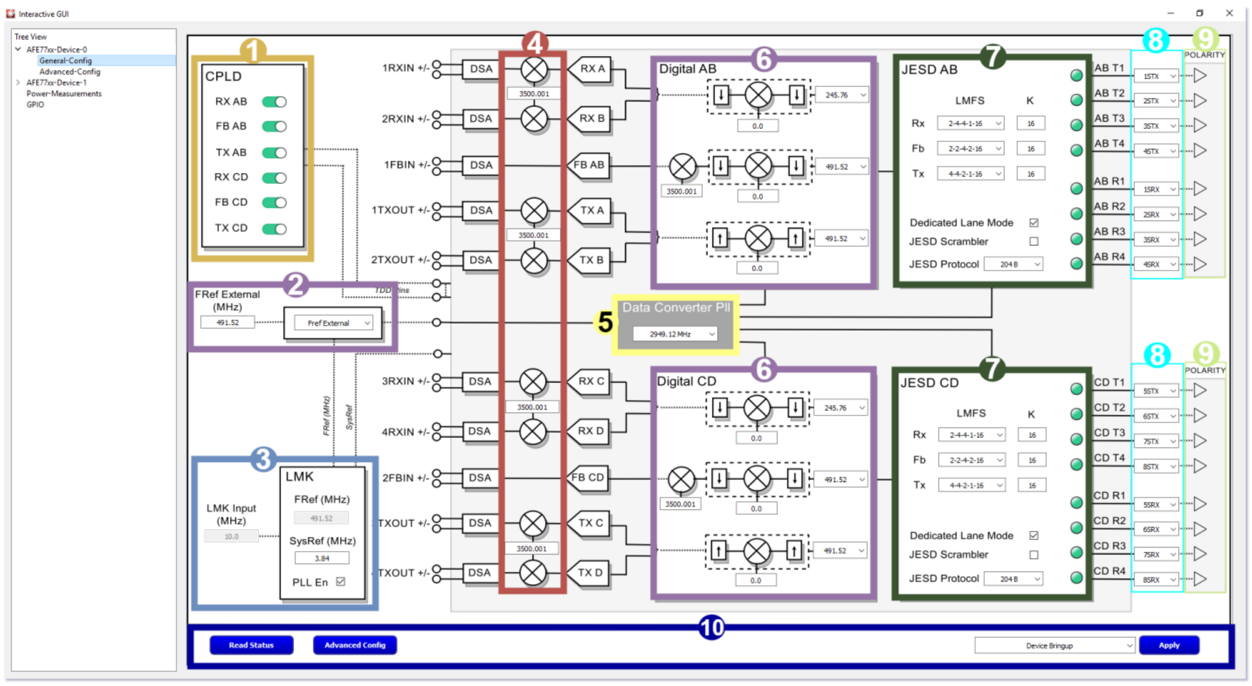


Figure 3. iGUI General Configuration Window

1. CPLD: This window includes the TDD enable pins for RX, TX, and FB.
2. Fref Selection: This window allows the user to configure FRef frequency and source (that is, derived from LMK or externally fed).
3. LMK: This window allows the user to configure LMK such as frequencies of FRef, SYSREF and so forth. The PLL En parameter allows control of LMK PLL. When this is checked, it uses the onboard VCXO as reference for LMK PLLs.
4. LO: The LO (Local Oscillator) for RX and TX can be configured in this window.
5. Data Converter PLL: Configures PLL1 for data converter clock.
6. Digital: This window allows the user to configure digital signal chain like FB NCO, Low IF NCO for TX/RX/FB and the data rates.
7. JESD: JESD parameters can be set in this window.
   - LMFS: A drop-down menu shows selectable options for RX, TX, and FB.
   - K: The K value can be programmed using the integer boxes available there in the GUI for all the channels.
   - Dedicated Lane Mode: The dedicated lane mode can be enabled to make Rx and Fb have independent lanes where the L in LMFS value of both Rx and Fb cannot be 3 or 4 because the available lanes for each channel is 2, so some choices are disabled from both the dropdowns. When the dedicated lane mode is disabled, Rx and Fb share the lanes where the F in LMFS value of both Rx and Fb mus be the same, so some choices are disabled from both the dropdowns.
   - JESD Scrambler: The JESD Scrambler is enabled to scramble the output data from the device.
   - JESD Protocol: The JESD protocol parameter allows the user to switch between JESD-204B and JESD-204C, which is by default JESD-204B.
   - Active lane indicators: Depending on the number of lanes (that is, L in LMFS value of Rx, Fb, and Tx) selected for each channel, the active lane indicators change to indicate the active lanes. When it is in dedicated lane mode 'AB T1' and 'AB T2 ' represents the Rx lane indicators, and 'AB T3' and 'AB T4' represents the Fb lane indicators. In non-dedicated lane mode, both Rx and Fb share 'AB T1'–'AB T4' lanes. On both dedicated and non-dedicated lane modes, 'AB-R1'–'AB-R4' are the Tx lane indicators.
8. Lane Mux: Allows configuration of Lane Mux as described in Section 4.6.1 and Section 4.6.2.

9. Lane Polarity: Sets polarity of SerDes inputs and outputs.
10. Functions: This block has two buttons named 'Read status' and 'Advanced Config', which redirect the user to the respective pages. The user can select a function from the available choices, such as 'Device Bringup', 'Configure NCOs' as so forth, and press the Apply button to run the selected function.
    - Generate Configuration: This directs the user to a new page where the configuration file can be generated.
    - Save and Load GUI State: The 'Save GUI state' function saves the current state of the GUI in a .state file. To load a state file, the 'Load GUI state' function is selected to load the .state files on to the GUI.

8.2 Read Status

Current status of the device configuration can be read in tab as shown in Figure 4.

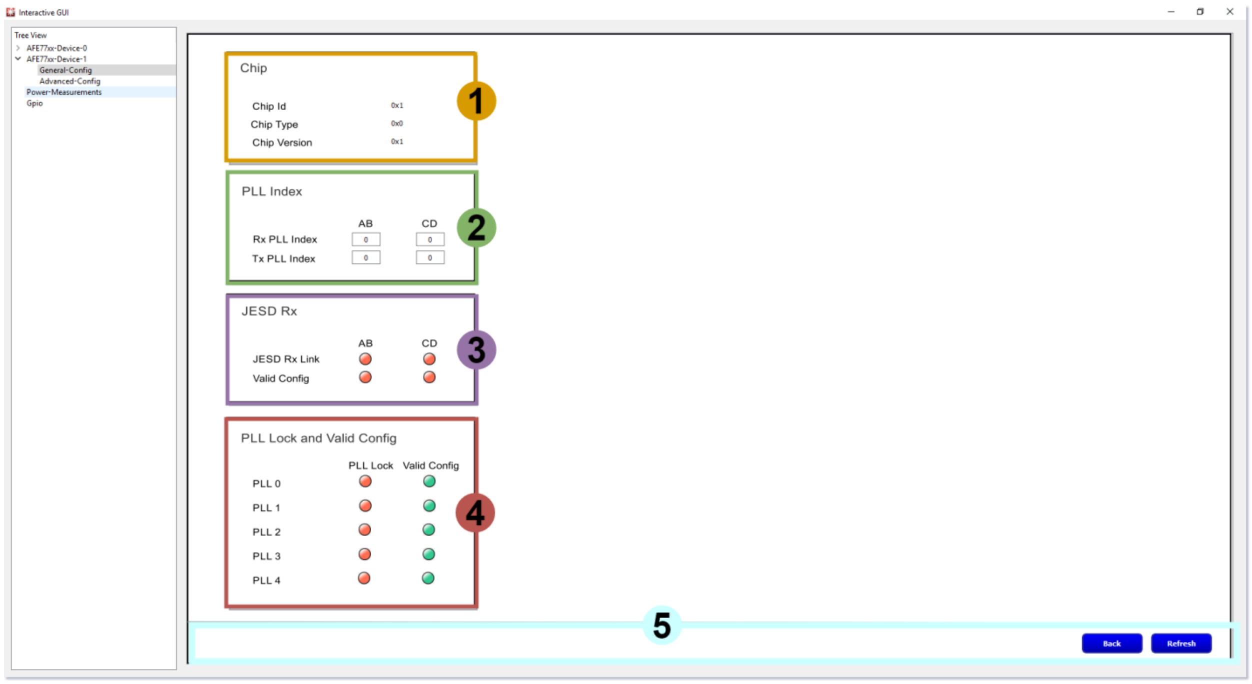
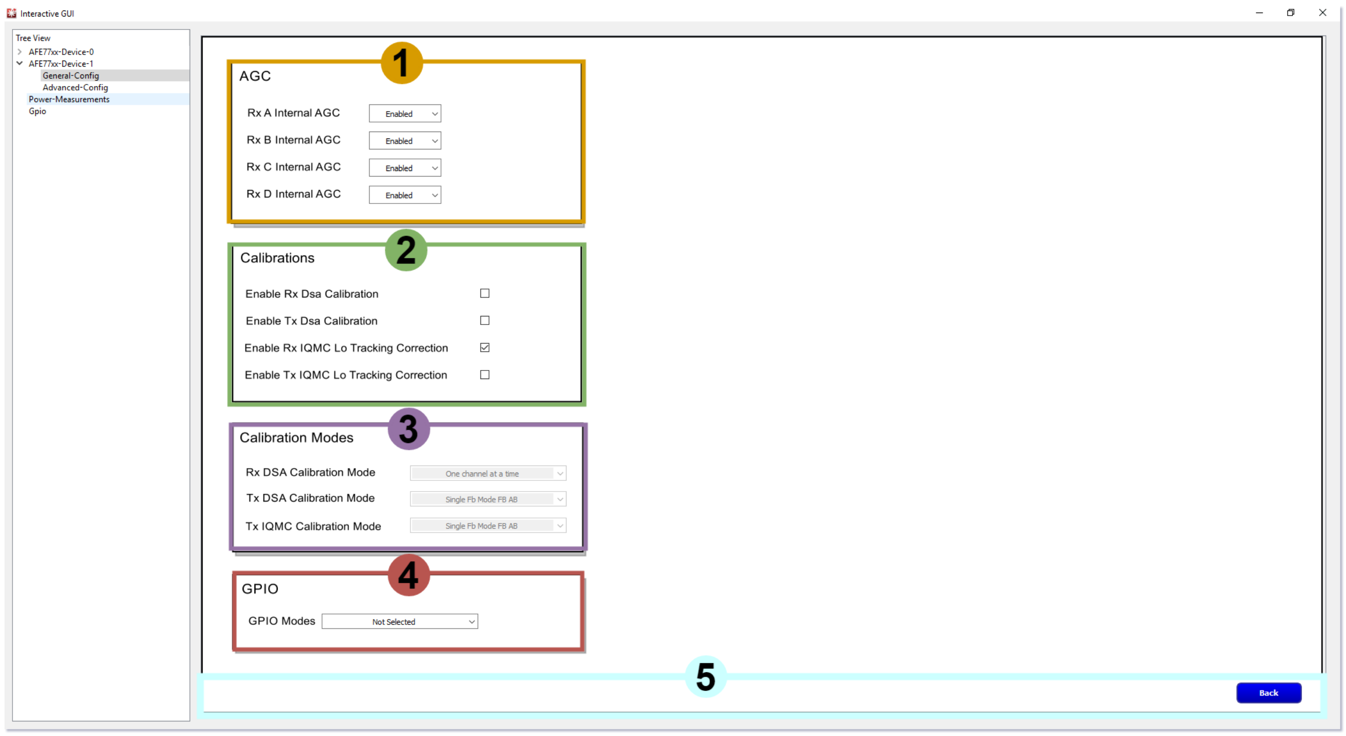


Figure 4. iGUI Read Status

1. Chip: Chip Id, type, and version are displayed.
2. PLL Index: Displays the LO distribution for RX and TX.
3. JESD RX: Displays the JESD link status.
4. PLL Lock Status.
5. To refresh or go back to previous tab.

8.3 Advanced Config

Additional parameters not covered in the general config tab can be configured through Figure 5. These parameters take effect only after a device bring up.



AGC

Rx A Internal AGC: Enabled
Rx B Internal AGC: Enabled
Rx C Internal AGC: Enabled
Rx D Internal AGC: Enabled

Calibrations

Enable Rx Dsa Calibration ☐
Enable Tx Dsa Calibration ☐
Enable Rx IQMC Lo Tracking Correction ☑
Enable Tx IQMC Lo Tracking Correction ☐

Calibration Modes

Rx DSA Calibration Mode: One channel at a time
Tx DSA Calibration Mode: Single Fb Mode FB AB
Tx IQMC Calibration Mode: Single Fb Mode FB AB

GPIO

GPIO Modes: Not Selected

Figure 5. iGUI Advanced Config

1. AGC: Enables internal AGC enables for Rx of the particular device.
2. Calibration: Select calibrations to be run.
3. Calibration Modes: Select modes for different calibrations.
4. GPIO: GPIO configuration mode.

8.4 Generate Config

A config file can be generated using the tab shown in Figure 6.

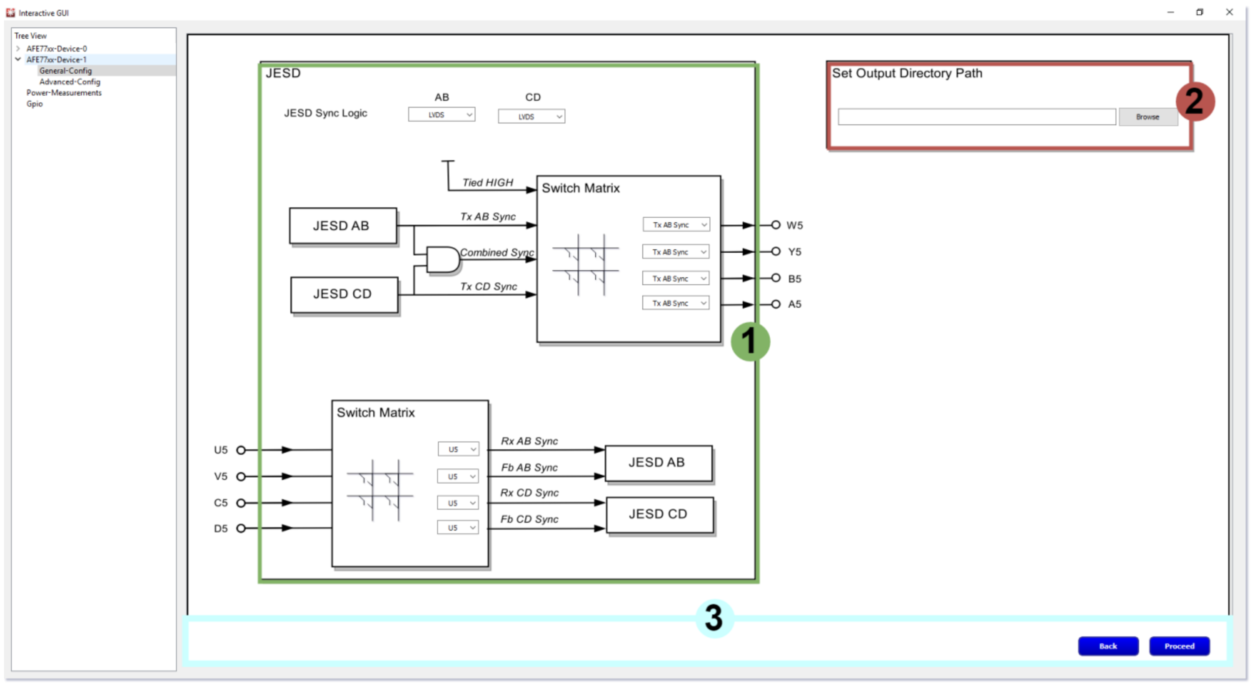


Figure 6. iGUI Generate Config File

1. JESD: Set SYNC logic and type.
2. Config Path: Set the path to save the generated config file.

9 Additional Notes

9.1 *Handling upgrades with Latte*

The Latte installer overwrites files that were installed by the prior version. Therefore, rename and back up the script or scripts that have been modified before installing the new version. Also, uninstalling the current version of Latte before installing the new one is recommended.

9.2 *Adding Scripts in Window 1*

Users can add a script in the scripts window (1) by right-clicking the folder "bringup" in this window. This adds a script with a default file name. Users can rename this file in Windows Explorer after closing the Latte GUI.