*Application Note*

# *AFE7950-SP Single Event Effects Detection*

*Shaily Jain, Nikhil Jain, Srinivas Murthy, Aleena Paul*

**ABSTRACT**

The AFE7950-SP supports an internal mechanism to detect single event effects (SEEs) such as single event upsets and functional interrupts through monitoring techniques such as *Configuration Register (Bit) Monitoring*, *SNR Monitoring* and *RX Bias Monitoring*. The firmware helps to detect radiation-induced events by triggering an alarm, allowing an external host to recover from these events.

The Configuration Register (Bit) Monitoring can be triggered to periodically monitor the configuration register sets across the device and compare their values with the golden reference. Golden reference refers to parity of register sets computed after the device is configured for the desired use case. An alarm is raised if a bit flips indicating that a single event upset (SEU) event has occurred.

Single event functional interrupts (SEFI) can be detected through SNR and RX Bias Monitoring that is signaled on a different GPIO alarm dedicated for SEFI. SNR Monitoring provides and checks for SNR for all the channels over user-specified band. The monitoring band can be different for different channels. SNR is computed within the device via internal FFT engine for the specified band and compared against user programmable threshold to trigger a GPIO alarm. The RX Bias Monitoring periodically monitors bias voltages for receiver and feedback channels and raises an alarm if they go out of spec.

## Table of Contents



## Trademarks

# 1 Introduction

After the AFE7950-SP is configured for the desired use case, the user must prompt the firmware to compute the current parity sets of the registers, which serve as the golden reference. After the golden reference is computed, the user must enable the bit monitoring.

For SNR Monitoring, the user must specify the frequency band to monitor noise floor for the desired channels along with the threshold. Here, the user has the flexibility to select the monitoring band and the threshold for each channel independently. While specifying the frequency band, one must take care that the band to be monitored is not supposed to receive or transmit any signal to avoid any false alarms. After the band and threshold configuration, the user can start monitoring for desired channels and check for any alarms if raised on the GPIO or through Alarm readback CAPI.

Similarly for RX Bias monitoring, the user can enable the desired channels to monitor and start monitoring to check for any SEFI event if occurred.

To detect any SEU or SEFI event, there are two dedicated GPIOs to indicate them distinctly.

If SEFI event has occurred, following mechanism raises an alarm that can be looked out on GPIO pin. Note that an alarm is raised if either of the below mechanism fails.

1. *SnrMonitor:* Indicates that SNR has gone bad beyond the set threshold in the monitoring band.
2. *RxBiasMonitor*: Indicates RX Bias is out of spec.
3. *FwHungStatus*: Indicates that the watchdog timer has timed out, meaning the firmware has lost functionality.

In case of SEU event, an error can be signaled through the other GPIO pin, allowing an external host to take appropriate actions.

1. *BitMonitor*: Indicates a bit flip in the monitored register map.

The firmware cannot determine whether the SEU-affected register has a catastrophic impact on performance or is benign. Therefore, TI recommends that the host system take appropriate corrective actions.

The above monitoring techniques run fully in background and do not impact normal operations in the data path, latency or JESD. However, for TX SNR Monitoring, small drop (< 0.4dB) is expected in power as the signal is tapped off at the TX output to monitor it on feedback channels.

The SNR monitoring mechanism for different channels is demonstrated in Figure 1-1 and Figure 1-2.

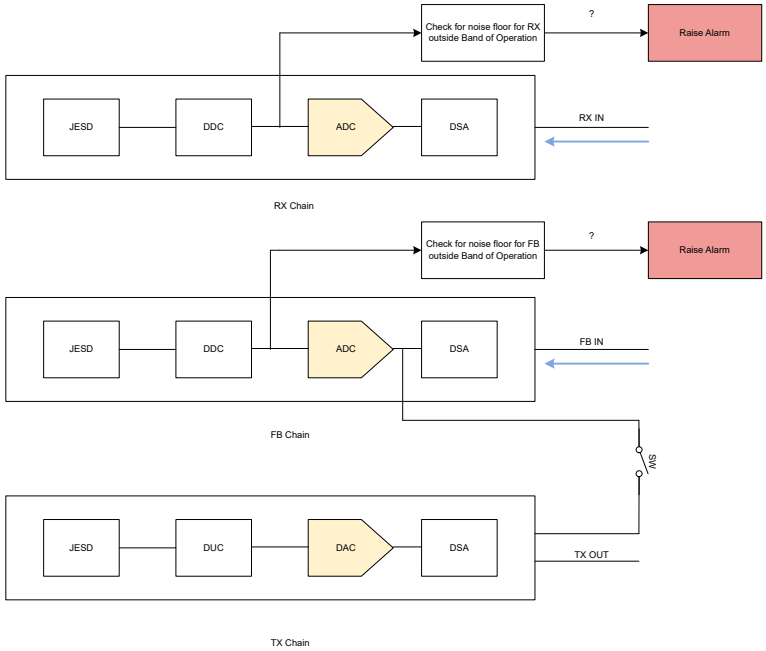


**Figure 1-1. Checking for SNR at RX and FB**

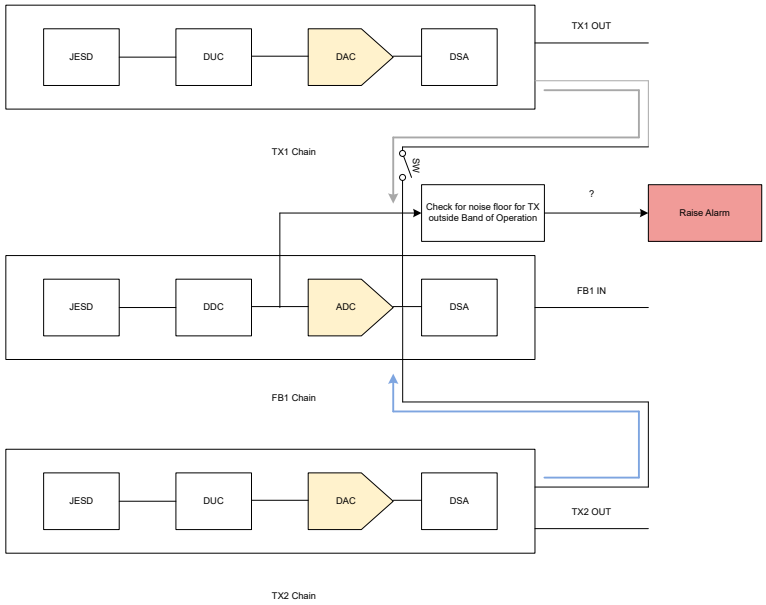


**Figure 1-2. Checking for SNR at TX through Loopback on FB**

## 2 Requirements

While using the monitoring features, the user must make sure the following conditions are met:

1. Feedback channels are used for TX SNR Monitoring- therefore, must not be used for data reception.
2. One must make sure that both the feedback channels have external 100Ω differential termination.
3. There is no such requirement for the RX channels.
4. SNR Monitoring band must be selected such that the band is *signal free* and *must not contain any external blockers*. Improper band selection or signal leakage into monitoring band can trigger false alarms.

# 3 Configuring AFE79xx Latte Python Script

The SEFI Detection Firmware is integrated starting from AFE79xx Latte V3.0. After Latte is installed, set the following parameters while configuring the device in desired mode:

1. 
```
sysParams.isAFE7950SP = 1
```

2. *sysParams.intPinsParams*: Multiple alarms from various blocks (e.g., JESD, PLL, Bit Monitoring, SNR Monitoring, RX Bias Monitoring and FW Watchdog Timeout) can be enabled and monitored. The AFE7950-SP provides two interrupt pins: ALARM1 and ALARM2. SEFI and SEU events can be triggered on AlARM1 and ALARM2 respectively.

   Format: *sysParams.intPinsParams[alarmNo][<parameter>] = True/False*

   Example: *sysParams.intPinsParams[0][‘SNRMonitor’]=True*

   True = Alarm is enabled for the pin

   False = Alarm is not enabled for the pin

**Table 3-1. Description of Different Alarms**

| Parameter | Default Value and Format | DESCRIPTION |
|---|---|---|
| BitMonitor | 1/True | SEU Event in configuration register set. |
| FwHungStatus | 1/True | Watchdog timer timeout. |
| JESD | 1/True | JESD Errors. |
| PLL | 1/True | PLL Lost of Lock. |
| SNRMonitor | 1/True | SNR goes bad below a certain threshold. |
| RXBiasMonitor | 1/True | RX Bias goes out of spec. |

   Refer to the example below to bring ‘SNRMonitor’, ‘RXBiasMonitor’ and 'FwHungStatus, on ALARM1, and ‘BitMonitor’ on ALARM2.

```
sysParams.intPinsParams[0]['BitMonitor']=0
sysParams.intPinsParams[0]['FwHungStatus']=1
sysParams.intPinsParams[0]['SNRMonitor']=1
sysParams.intPinsParams[0]['RxBiasMonitor']=1
sysParams.intPinsParams[1]['SNRMonitor']=0
sysParams.intPinsParams[1]['RxBiasMonitor']=0
sysParams.intPinsParams[1]['BitMonitor']=1
sysParams.intPinsParams[1]['FwHungStatus']=0
```

3. 
```
sysParams.gpioMapping = {'M15':'ALARM1','L14':'ALARM2'}
```

   After configuring interrupts, assign them to the GPIO M15 and L14. Refer to the example below where ALARM1 and ALARM2 interrupt is brought out on GPIO M15 and L14 respectively.

4. 
```
FbEnable = [True, True]
```

For monitoring TX, FB channels are used and therefore must be enabled in the bring up configuration. TX1 and TX2 are monitored using FB1 while TX3 and TX4 are monitored using FB2. Therefore, set the *FbEnable* accordingly. Other parameters related to Feedback channels can be set similar to RX.

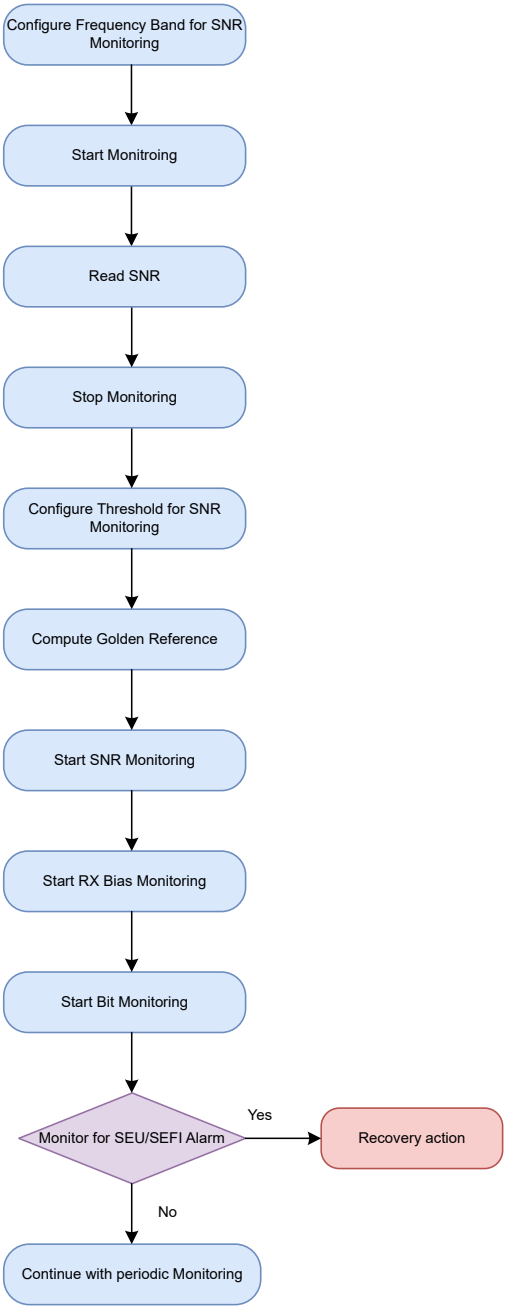


**Figure 3-1. Sequence After First Bring Up**

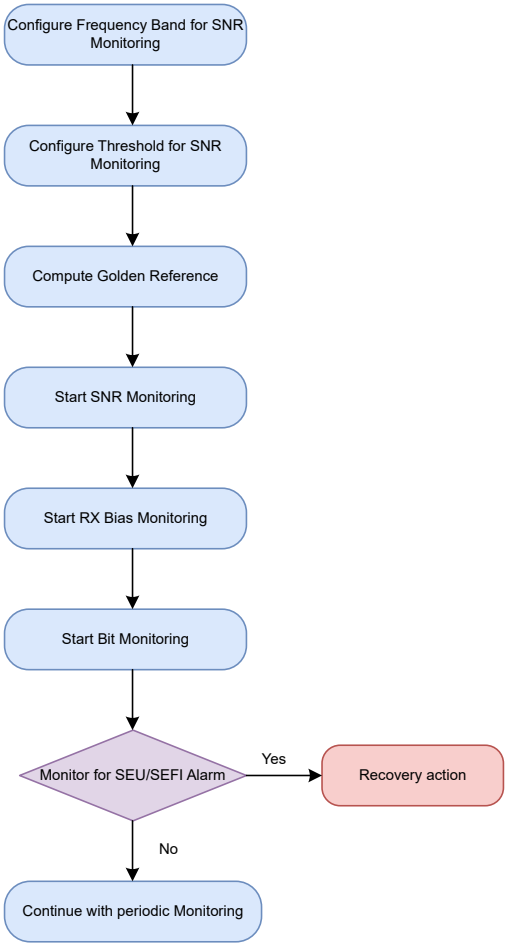


**Figure 3-2. Sequence After Subsequent Bring Ups**

# 4 Post Device Configuration

After the device has been configured by following the instructions in the previous section and bring up is completed, the following sequence needs to be followed:

1. Use CAPI *configSnrBand* to specify the SNR Monitoring Band for all the channels at once or use the channel select for RX, FB and TX to configure them separately.
2. Use CAPI *startSnrMonitoring* to start monitoring for selected channels.
3. Use CAPI *readRxSnr, readFbSnr, readTxSnr* to read the SNR for RX, FB and TX respective. Always make sure that *startSnrMonitoring* CAPI has been called before calling this CAPI. Use the reported values to set the threshold.
4. Run the CAPI *stopSnrMonitoring* to stop monitoring after reading the values.
5. Use CAPI *configSnrThreshold* to set the thresholds for selected channels using channel select option. For RX and FB channels set the threshold at least 10dB higher than the reported value. While, for TX set the threshold at least 5dB higher than the value that is reported.
6. Optionally, change the monitor period using *setSnrMonitorPeriod* for SNR, *setRxBiasMonitorPeriod* for RX Bias and *setBitMonitorPeriod* for Configuration Monitoring respectively. The default monitoring periodicity for each of them is set to 1 second.
7. Once the desired band, threshold and monitoring parameters are set, use CAPI *startSnrMonitoring* to start the periodic (default periodicity is approximately 1s) monitoring for selected channels.
8. Use CAPI *startRxBiasMonitoring* to start the periodic (default periodicity is approximately 1s) monitoring for selected channels.
9. Use CAPI *computeGolden* to compute the golden reference parity for configured register set.
10. Use CAPI *startMonitoring* to start the periodic (periodicity is approximately 1s) monitoring.
11. Monitor the GPIO Pin M15 dedicated for SEFI detection for any alarms. Additionally, one can use the CAPI *getSnrAlarm* to read alarm status for each channel in case the SNR goes bad or CAPI *getRxBiasAlarm* to get the alarm status for RX and FB channel in case the bias is out spec. Similarly, use CAPI *getBitAlarm* to get the alarm status in case any bit is flipped in the configuration register sets.
12. Use CAPI *stopSnrMonitoring* and *stopRxBiasMonitoring* to stop monitoring for SEFI and *stopMonitoring* to stop monitoring for SEU.

The above steps must be followed only when bringing up the device for the first time in a particular mode. For subsequent bringups, steps two, three, and four can be skipped and the threshold obtained from the first run can be set, provided the monitoring band is identical. See Figure 3-1 and Figure 3-2.

To call the CAPI sequence, use the following commands.

## 4.1 Configuring SNR Monitoring Parameters

1. Format: *CAFE.configSnrBand(afeInst, rxSelect, fbSelect, txSelect, startFreq, stopFreq)*
   a. `Example 1.1: CAFE.configSnrBand(0, 15, 3, 0, 9300000, 9400000)`
   b. `Example 1.2: CAFE.configSnrBand(0, 0, 0, 15, 9450000, 9550000)`

2. Format: *CAFE.startSnrMonitoring(afeInst, rxEnable,fbEnable,TxEnable)*
   a. `Example 2: CAFE.startSnrMonitoring(0,15,3,15)`

3. Format: *CAFE.readRxSnr(afeInst, chNo)*

   Format: *CAFE.readFbSnr(afeInst, chNo)*

   Format: *CAFE.readTxSnr(afeInst, chNo)*

   a. `Example 3.1: CAFE.readRxSnr(0,0)`

   Result: -75

   Similary for other RX Channels.

b. `Example 3.2 CAFE.readFbSnr(0,0)`

Result: -80

Similarly for another FB channel.

c. `Example 3.3: CAFE.readTxSnr(0,0)`

Result: -55

Similarly for other TX channels.

4. Format: *CAFE.stopSnrMonitoring(afeInst)*
   a. `Example 4: CAFE.stopSnrMonitoring(0)`

5. Format: *CAFE.configSnrThreshold(afeInst, rxSelect, fbSelect, txSelect, Threshold)*
   a. `Example 5.1: CAFE.configSnrThreshold(0, 15, 0, 0, -65)`
   b. `Example 5.2: CAFE.configSnrThreshold(0, 0, 3, 0, -70)`
   c. `Example 5.3: CAFE.configSnrThreshold(0, 0, 0, 15, -50)`

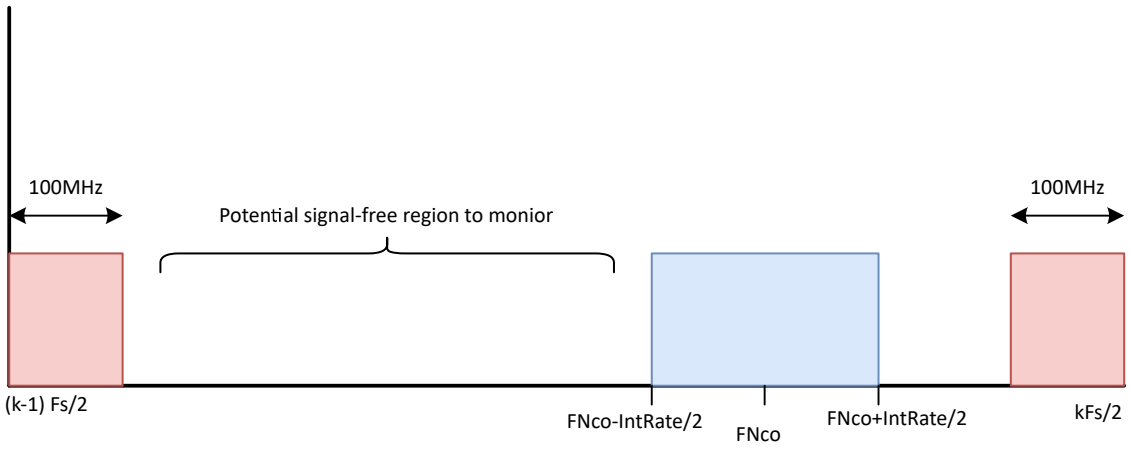


**Figure 4-1. The Monitoring Band of 100MHz Must Be Selected Outside the Blue and Red Region and Within the Same Nyquist in Which the Signal is Received**

### *4.1.1 Configuring Monitoring Parameters for RX*

Start and Stop Frequency must be selected such that:

1. The monitoring band must be at least 100MHz away from Nyquist frequency.
2. The band you monitor is not supposed to receive any signal and must be outside the DDC band. Any effect of external blockers or jammer must be eliminated.
3. The monitoring band must be selected in same Nyquist as the NCO Frequency.

For example, suppose the sampling frequency(Fs) for ADC is 3000MHz and you have your NCO at 9500MHz, and interface rate is 200MHz, then choose your start Frequency either 9300MHz or 9700MHz whichever satisfies the above conditions. Stop frequency must be 100MHz away from start Frequency making sure that the monitoring bandwidth is 100MHz. To set the threshold, use the CAPI *readRxSnr*. It is recommended that the threshold must be set at least **10dB** higher than the reported SNR. This makes sure that detection for 10dB or higher degradation in SNR. Refer to the example 3.1 above. Perform the sequence for all the RX Channels.

#### 4.1.2 Configuring Monitoring Parameters for FB

Set start Frequency (in kHz) and stop Frequency (in kHz) for both the FB Channels same as that of RX Channels. The monitoring bandwidth must be 100MHz. To set the threshold, use the CAPI *readFbSnr*. TI recommends that the threshold must be set at least **10dB** higher than the reported SNR. This makes sure that detection for 10dB or higher degradation in SNR. Refer to the example 3.2 above. Perform the sequence for all the FB Channels.

#### 4.1.3 Configuring Monitoring Parameters for TX

TX Channel Monitoring is done by looping back the transmit signal to an FB channel, via an internal loopback mechanism. Start and Stop Frequency must be selected such that:

1. The monitoring band must be at least 100MHz away from FB channel's Nyquist band edge.
2. The band you monitor is not supposed to transmit any signal or contain any other spur and must be outside theTX Signal band While selecting a TX band to monitor, do make sure that the band does not contain aliased copies of the TX DAC images that occur when the TX signal is observed via the FB ADC.
3. The monitoring band must be within the same Nyquist in which you transmit.

Here, the Nyquist frequency must be determined according to sampling rate of FB ADCs since the TX output is being monitored using FB channels. Any effect of **aliasing** must also be considered while selecting the band.

For example, suppose you have your TX NCO at 9250MHz, and interface rate is 200MHz, then you can choose your start Frequency either 9050MHz or 9450MHz whichever satisfies the above conditions. Stop frequency must be 100MHz away from start Frequency to make sure that the monitoring bandwidth is 100MHz. To set the threshold, use the CAPI *readTxSnr*. TI recommends that threshold must be set **5dB** higher than the reported SNR. Refer to the example 3.3 above. Perform the sequence for all the TX Channels.

### 4.2 Computing Golden Reference

Format: *CAFE.computeGolden(afeInst)*

```
Example 6: CAFE.computeGolden(0)
```

Golden reference refers to parity of register sets computed once the device is configured for the desired use case. Call the above command to compute the golden reference which shall serve as the reference for comparison of register sets against any corruption.

### 4.3 Start Monitoring

Format: *CAFE.startSnrMonitoring(afeInst, rxEnable, fbEnable txEnable)*

```
Example 7.1: CAFE.startSnrMonitoring(0,15,3,15)
```

Select the RX, TX and FB channels for SNR monitoring using the parameters *rxEnable*, *txEnable* and *fbEnable,* respectively.Each parameter is a bitwise representation of the channel indices, with the least significant bit corresponding to the first channel and so on.

Format: *CAFE.startRxBiasMonitoring(afeInst, rxEnable, fbEnable)*

```
Example 7.2: CAFE.startRxBiasMonitoring(0,15,3)
```

Select the RX and FB channels for monitoring using the parameters *rxEnable* and *fbEnable,* respectively.Each parameter is a bitwise representation of the channel indices, with the least significant bit corresponding to the first channel and so on.

Format: *CAFE.startMonitoring(afeInst)*

```
Example 7.3: CAFE.startMonitoring(0)
```

Call the above command to start Bit monitoring.

### 4.4 Reading the Alarms

After running the above sequence, once the monitoring has been enabled the user can look out for alarm on assigned GPIO. Additionally, one can call the alarm readback function in the following manner.

Format: *CAFE. getSnrAlarm(afeInst)*

```
Example 8.1: CAFE.getSnrAlarm(0)
```

Format:*CAFE. getRxBiasAlarm(afeInst))*

```
Example 8.2: CAFE.getRxBiasAlarm(0)
```

Format: *CAFE. getBitAlarm(afeInst))*

```
Example 8.3: CAFE.getBitAlarm(0)
```

CAPI *getSnrAlarm* reports three values where the first value corresponds to RX Alarms, second corresponds to FB Alarms while the third value corresponds to TX Alarms. RX Alarms can be interpreted bitwise where bit 0 corresponds to alarm for RX1 and bit 1 corresponds to alarm for RX2 and so on. Similarly, FB Alarms can be interpreted bitwise where bit 0 corresponds to alarm for FB1 and bit 1 corresponds to alarm for FB2. TX Alarms can be interpreted similar to RX Alarms.

Refer to the example below:

```
Output: [11,1,2]
Binary representation: [1011,01,0010]
```

This implies that SNR has crossed the set threshold for the channels: RX1, RX2, RX4, FB1 and TX2.

CAPI *getRxBiasAlarm* reports two values where the first value corresponds to RX Alarms and the second value corresponds to FB Alarms. RX Alarms can be interpreted bitwise where bit 0 corresponds to alarm for RX1 and bit 1 corresponds to alarm for RX2 and so on. Similarly, FB Alarms can be interpreted bitwise where bit 0 corresponds to alarm for FB1 and bit 1 corresponds to alarm for FB2.

See the following example:

```
Output: [3,1]
Binary Representation: [0011,01]
```

This implies that bias has gone out of spec for: RX1, RX2 and FB1.

CAPI *getBitAlarm* reports whether the configuration registers sets have been corrupted due to any radiation event. If the output is 1, this indicates a bit flip/flips in configuration register sets while 0 indicates the register sets are not impacted.

These alarms are **sticky** in nature. This means, if the alarm ever gets high, it holds that condition. You can use the clear alarm function to clear these alarms and call read alarms functions again to get the current state. Refer to the example below to clear alarms.

Format: *CAFE.clearSnrAlarm(afeInst)*

```
Example 10.1: CAFE.clearSnrAlarm(0)
```

Format: *CAFE.clearRxBiasAlarm(afeInst)*

```
Example 10.2: CAFE.clearRxBiasAlarm(0)
```

Format: *CAFE.clearBitAlarm(afeInst)*

```
Example 10.3: CAFE.clearBitAlarm(0)
```

## 4.5 Setting the Monitoring Periodicity

The mechanism also provides flexibility to choose the monitoring periodicity. Period must be set in milliseconds and must always be greater than 50 milliseconds. The monitoring periodicity must not be changed when the monitoring is running in the background. Therefore, while calling this CAPI, one must make sure that the monitoring is in stop state. Hence, call the stop monitoring API before changing the periodicity. To set monitoring periodicity as three seconds, refer to the example below.

Format: *CAFE.setBitMonitorPeriod(afeInst, period)*

```
Example 11.1: CAFE.setBitMonitorPeriod (0,3000)
```

Format: *CAFE.setSnrMonitorPeriod(afeInst, period)*

```
Example 11.2: CAFE.setSnrMonitorPeriod(0,3000)
```

Format: *CAFE.setRxBiasMonitorPeriod(afeInst, period)*

```
Example 11.3: CAFE.setRxBiasMonitorPeriod (0,3000)
```

## 4.6 Stop Monitoring

To stop monitoring, use the CAPI below:

Format: *CAFE.stopMonitoring(afeInst)*

```
Example 12.1: CAFE.stopMonitoring(0)
```

Format: *CAFE.stopSnrMonitoring(afeInst)*

```
Example 12.2: CAFE.stopSnrMonitoring(0)
```

Format: *CAFE.stopRxBiasMonitoring(afeInst)*

```
Example 12.3: CAFE.stopRxBiasMonitoring(0)
```

# 5 Dynamic Change Post Configuration

If dynamic change is required post device bring up, such as NCO or DSA update, CAPIs can change register sets which are being monitored. This results in a false alarm as the firmware can interpret this change as an SEU event. To avoid these false alarms, the following sequence is recommended:

1. Stop Monitoring using CAPI *stopMonitoring*.
2. Do the desired dynamic update (Example NCO Change).
3. Re-compute golden parity using CAPI *computeGolden*.
4. Re-start monitoring using CAPI *startMonitoring*.

While changing monitoring band, threshold, periodicity or any other monitoring related parameters needs when the background monitoring, directly calling the corresponding CAPIs can intervene with the background monitoring and firmware can get stuck.

To avoid such scenarios, the following sequence is recommended:

1. Stop monitoring using respective CAPI (as demonstrated in example 12).
2. Do the desired update using the corresponding CAPI.
3. Re-start Monitoring using start monitoring CAPI (as shown in example 7).

# 6 Results

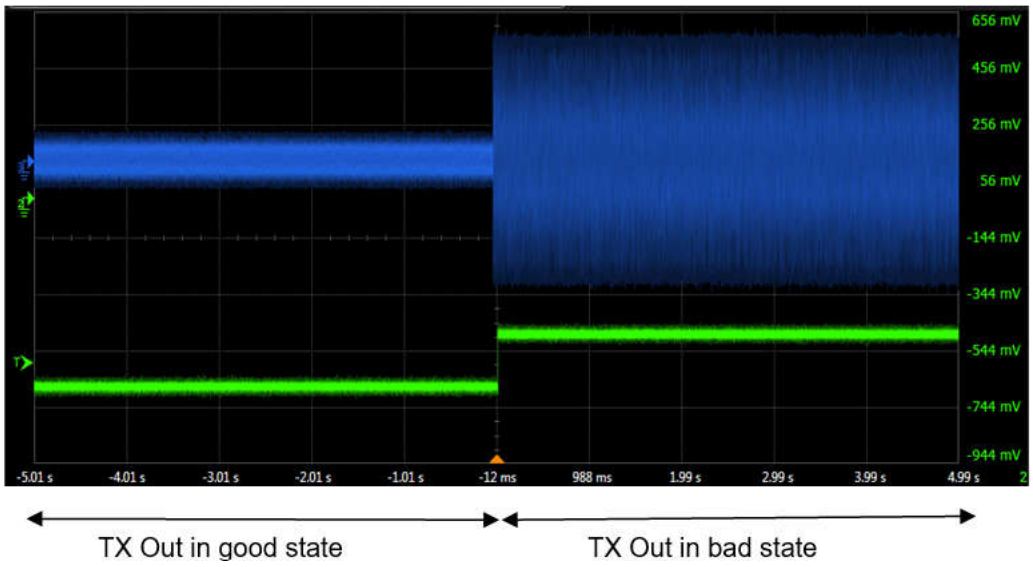


**Figure 6-1. The TX Output is Shown in Blue and GPIO Alarm is Represented in Green**

Once the monitoring is enabled, an alarm is triggered when the SNR exceeds the set threshold, and this alarm can be seen on GPIO M15, as shown in Figure 6-1.

# 7 Recovery Action

After an alarm is triggered, user must take appropriate action depending on the type of alarm. If the alarm is an SEU alarm, AFE hardware reset must mitigate the upset event allowing one to continue with the ongoing operation. After the reset is done, the user must reload configuration and re-bringup the device followed by the steps to get the diagnostic space firmware up.

While, in case of a SEFI alarm, power cycle is needed to recover the component in addition to the recovery steps mentioned above for SEU and as illustrated in Figure 7-1.

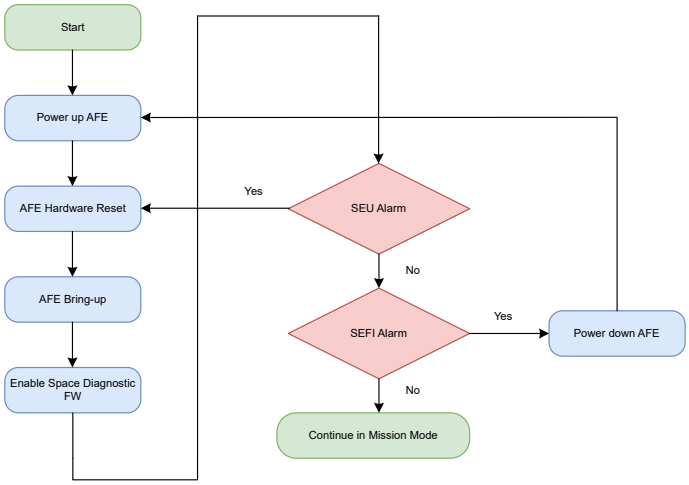


**Figure 7-1. Recovery Action for SEU and SEFI Alarms**

# 8 References

- Texas Instruments, *Texas Instruments, Radiation Handbook for Electronics*, e-book.
- Texas Instruments, *AFE7950-SP Single Event Effects Report*, radiation report.

# 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, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

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



Last updated 10/2025