AN1436: SiWx917 QMS Crystal Calibration Application Note

This application note describes the crystal calibration procedure for the SiWx917 QMS package to arrive at the right value for carrier frequency offset and updating the optimized values to the flash of the chip. Customers are required to calibrate frequency offsets on their boards after the chip is assembled with the planned front-end circuit. This document covers the commands used to determine the frequency offset and the flow of measurement and correction.

KEY POINTS

- Prerequisites, Setup Requirements, and Setup Diagrams
- Frequency Offset Measurements
- Frequency Offset Corrections

Table of Contents

1. Introduction

The crystal calibration can be done with two approaches. The first approach uses a continuous wave (CW) DC tone unmodulated carrier for transmission and the second approach uses Burst Mode, which transmits modulated data. The first approach only requires a normal spectrum analyzer. Burst mode can only be used if test instrument supports modulation analysis where frequency error can be reported by instrument.

The sequence of steps includes the following:

- Start transmission in packet burst or CW mode that allows control of data rate, transmit power and frame length.
- Measure the carrier frequency offset.
- Update the calibrated parameters to chip.

2. Prerequisites and Setup Requirements

2.1 Hardware

- SiWx91x Wi-Fi Evaluation Kit. The SiWx91x supports multiple operating modes:
  - SoC Mode: BRD4325A, BRD4325B, BRD4325C, BRD4325G, BRD4338A
  - NCP Mode: BRD8036B
- Host MCU Silicon Labs WPK
- Windows PC
- Spectrum Analyzer
- RF Cable (or antenna) to connect the Radio/Exp Board and the spectrum Analyzer

2.2 Software

- Simplicity Studio IDE
  - Download the Simplicity Studio IDE.
  - Follow the Simplicity Studio User Guide to install the Simplicity Studio IDE.
  - Install a GSDK suite with the WiSeConnect SDK included.
  - Update the SiWx91x Connectivity Firmware.

Note: The WiSeConnect and Connectivity Firmware installation is described here in detail: Getting Started with the WiSeConnect SDK.

3. Setup Diagram

3.1 SoC Mode

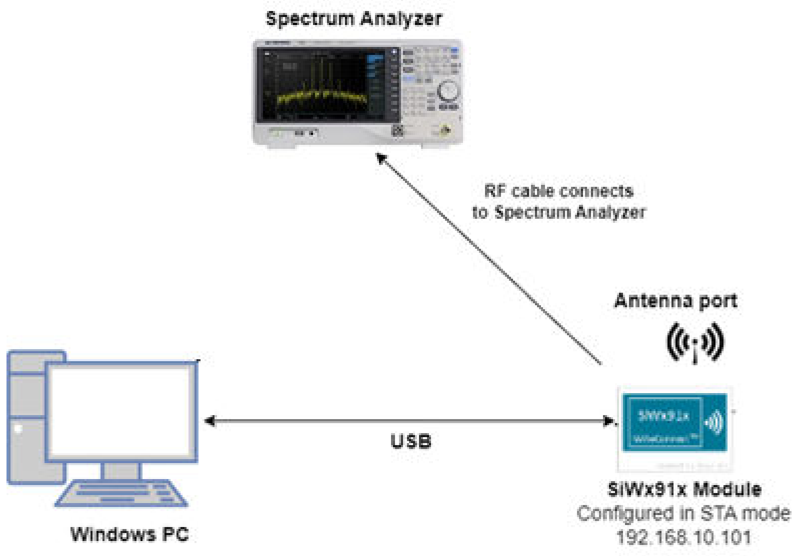


Figure 3.1. Calibration Test Setup for SoC Mode

3.2 NCP Mode

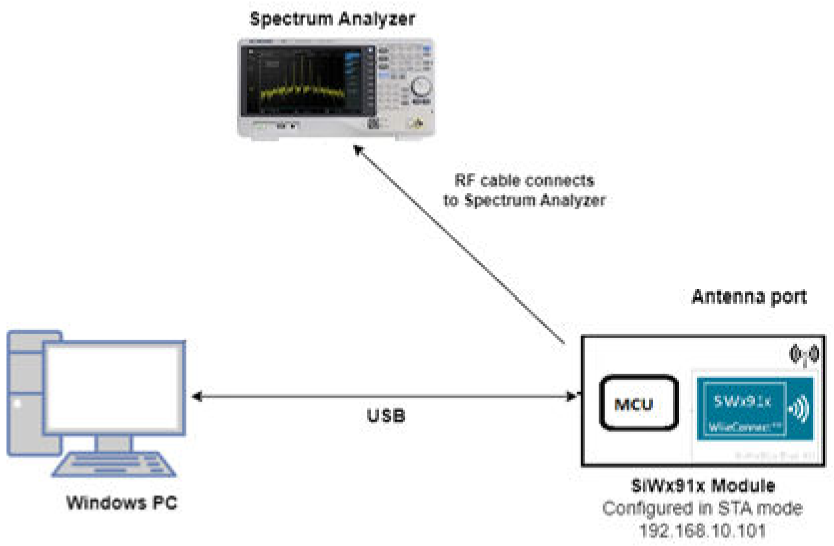


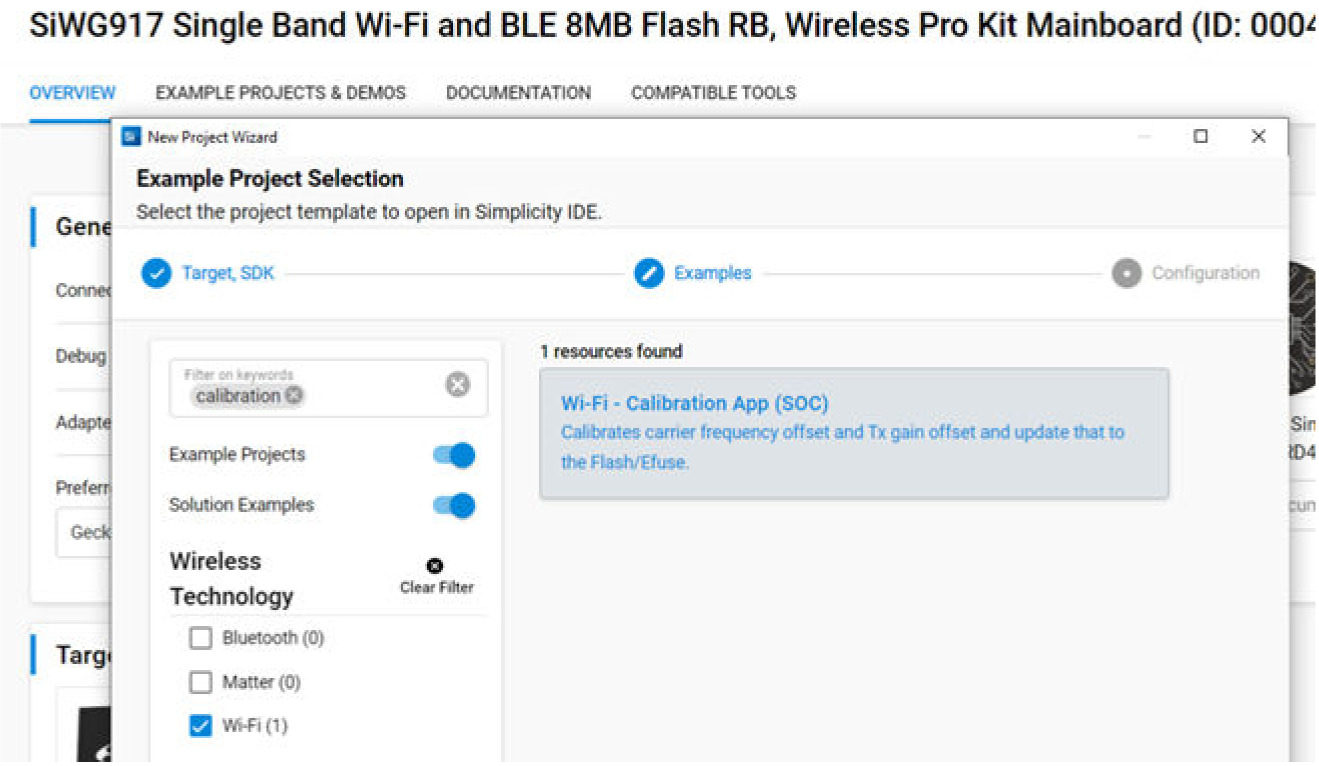
Figure 3.2. Calibration Test Setup for NCP Mode

4. Setting-up the Project Environment

4.1 Creating the Project

The calibration can be done easily with the Wi-Fi calibration application provided by the WiSeConnect SDK. Project configuration and running the calibration example is described here in more details: Project Environment.

To get started, create a Wi-Fi Calibration (SoC or NCP) application. Open the new project wizard in Simplicity studio and choose the correct application.



4.2 Configuring the Application

Once the application project is created, the application can be configured to suit the user requirements and the development environment. Read through the following sections and desired modifications.

Configure the following parameters in **app.c** to test Xtal Calibration app as per requirements:

```
#define TX_TEST_POWER        18 // Sets TX power in dBm. The valid values are from (2 to 18)dBm and 127.
#define TX_TEST_RATE        0 // Sets transmit data rate.
#define TX_TEST_LENGTH        30 // Configures length of the TX packet. Valid values are in the range of 24 to
1500 bytes in the burst mode.
#define TEST_CHANNEL        1 // Configures the channel number in 2.4 GHz Band
#define TX_TEST_MODE        BURST_MODE // Configures Burst mode, Continuous mode or CW mode
```

Note:

- Tx burst mode can only be used if test instrument supports Modulation analysis measurement, where the Freq error can be reported by the instrument.
- Tx CW mode is not implemented in this example. To access CW mode, modify the app.c file based as mentioned in the following section.

4.3 Implementing CW Mode

1. Define and set CW_MODE as a Tx test mode in the constants section.

```
#define BURST_MODE          0
#define CONTINUOUS_MODE     1
#define CW_MODE             2    // CW/DC unmodulated Carrier mode is defined here

#define TX_TEST_POWER         18
#define TX_TEST_RATE          0
#define TX_TEST_LENGTH         30
#define TX_TEST_MODE         CW_MODE    // Test mode is changed here
```

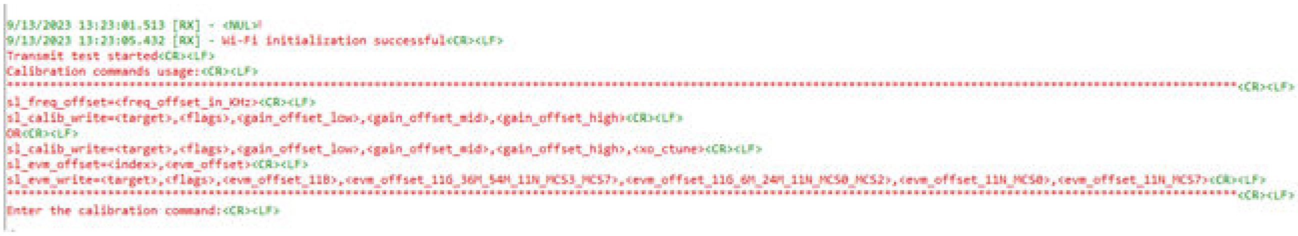
2. Start a burst generation, stop it and start the CW mode generation in the application_start() function. Add the next lines just before sl_si91x_transmit_test_start() is called.

```
status = sl_si91x_transmit_test_start(TX_TEST_POWER, TX_TEST_RATE, TX_TEST_LENGTH, 0, TEST_CHANNEL);
  if (status != SL_STATUS_OK) {
    printf("Transmit test start failed: 0x%lx\r\n", status);
  } else {
    printf("Transmit test started\r\n");
  }

  osDelay(1000);

  status = sl_si91x_transmit_test_stop();

  osDelay(1000);
```

3. Build the code.
4. After confirming the input parameters, build and flash the application.

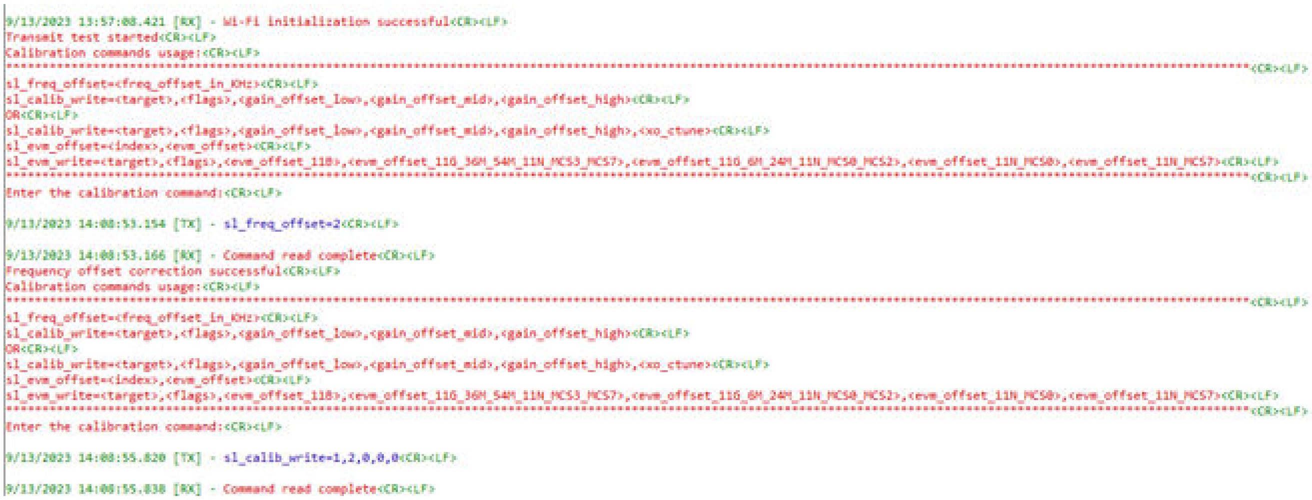
4.4 Application Execution Flow

1. Connect WPK Mainboard via USB to the computer and flash the compiled project to the board.
2. Lauch a console on the device to access the command line interface. In Simplicity Studio, choose Serial1 window on the console. After the program gets executed (device is reset or powered up), the SiWx91x device will start the transmit test with the given configuration. Application prints (over JLink port) would be as follows:



```
9/13/2023 13:23:01.513 [RX] - <NUL>
9/13/2023 13:23:05.432 [RX] - Wi-Fi initialization successful<CR><LF>
Transmit test started<CR><LF>
Calibration commands usage:<CR><LF>
****************************************************************************************************************<CR><LF>
sl_freq_offset=<freq_offset_in_KHz><CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high><CR><LF>
OR<CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high>,<xo_ctune><CR><LF>
sl_evm_offset=<index>,<evm_offset><CR><LF>
sl_evm_write=<target>,<flags>,<evm_offset_11B>,<evm_offset_11G_36M_54M_11N_MCS3_MCS7>,<evm_offset_11G_6M_24M_11N_MCS0_MCS2>,<evm_offset_11N_MCS0>,<evm_offset_11N_MCS7><CR><LF>
****************************************************************************************************************<CR><LF>
Enter the calibration command:<CR><LF>
```

3. The calibration commands (discussed in Frequency Offset Correction) can be entered in the serial terminal as input.



```
9/13/2023 13:57:08.421 [RX] - Wi-Fi initialization successful<CR><LF>
Transmit test started<CR><LF>
Calibration commands usage:<CR><LF>
****************************************************************************************************************<CR><LF>
sl_freq_offset=<freq_offset_in_KHz><CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high><CR><LF>
OR<CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high>,<xo_ctune><CR><LF>
sl_evm_offset=<index>,<evm_offset><CR><LF>
sl_evm_write=<target>,<flags>,<evm_offset_11B>,<evm_offset_11G_36M_54M_11N_MCS3_MCS7>,<evm_offset_11G_6M_24M_11N_MCS0_MCS2>,<evm_offset_11N_MCS0>,<evm_offset_11N_MCS7><CR><LF>
****************************************************************************************************************<CR><LF>
Enter the calibration command:<CR><LF>

9/13/2023 14:08:53.154 [TX] - sl_freq_offset=2<CR><LF>

9/13/2023 14:08:53.166 [RX] - Command read complete<CR><LF>
Frequency offset correction successful<CR><LF>
Calibration commands usage:<CR><LF>
****************************************************************************************************************<CR><LF>
sl_freq_offset=<freq_offset_in_KHz><CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high><CR><LF>
OR<CR><LF>
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high>,<xo_ctune><CR><LF>
sl_evm_offset=<index>,<evm_offset><CR><LF>
sl_evm_write=<target>,<flags>,<evm_offset_11B>,<evm_offset_11G_36M_54M_11N_MCS3_MCS7>,<evm_offset_11G_6M_24M_11N_MCS0_MCS2>,<evm_offset_11N_MCS0>,<evm_offset_11N_MCS7><CR><LF>
****************************************************************************************************************<CR><LF>
Enter the calibration command:<CR><LF>

9/13/2023 14:08:55.820 [TX] - sl_calib_write=1,2,0,0,0<CR><LF>

9/13/2023 14:08:55.838 [RX] - Command read complete<CR><LF>
```

5. Frequency Offset Measurement

5.1 Measure with Transmitted Continuous Wave (CW)

1. Start up the example application in CW mode and observe the frequency error difference between the calibrated spectrum analyzer and the SoC as indicated below (a marker could be useful to make accurate readings.)

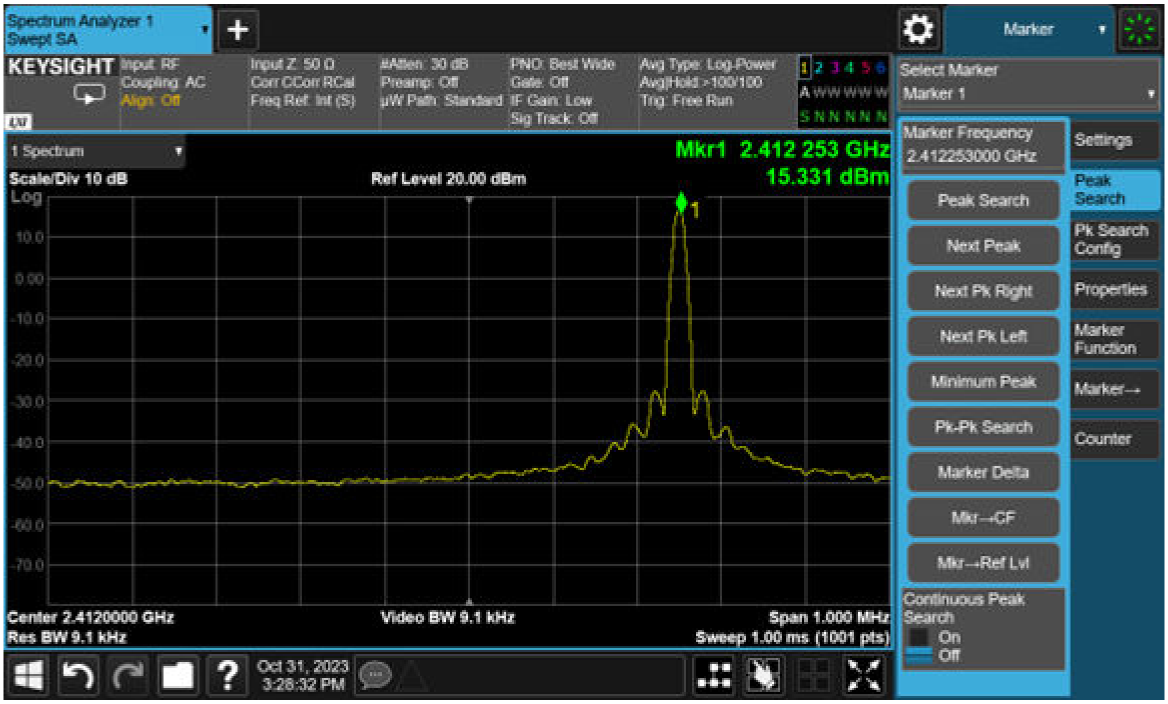


Figure 5.2. Frequency Offset Measured in CW Mode with a Traditional Spectrum Analyzer

2. Note the frequency offset, which will be fed back to the device in the following section.

5.2 Measure with Transmitted Burst

1. Start up the example application in burst mode and set up the spectrum analyzer in a way that the frequency offset can be measured. Here's an example of such measurement:

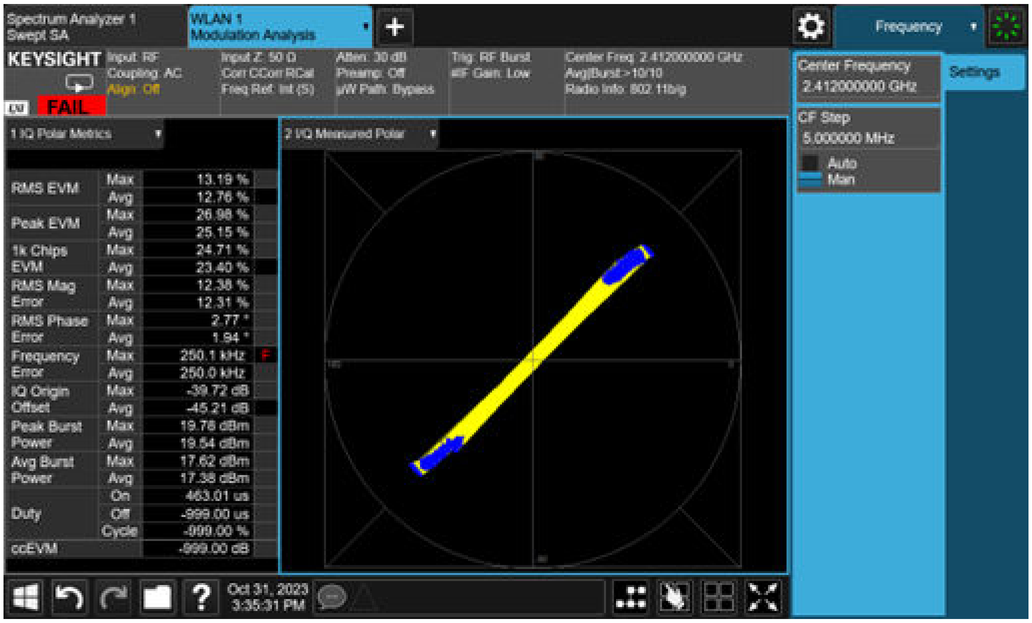


Figure 5.3. Frequency Offset Measured in Burst Mode with a Traditional Spectrum Analyzer

Note: In the above measurement example, for both CW and Burst Mode, a Frequency error of ~250 KHz is reported by Spectrum Analyzer.

6. Frequency Offset Correction

Frequency offset correction is done by using the `sl_freq_offset` command. This command is used during the RF calibration process and requires PER mode transmissions to be initiated prior. This command sets freq_offset (deviation) as observed on the signal analyzer against the expected channel frequency.

Frequency offset correction command syntax:

```
sl_freq_offset = freq_offset_in_kHz
          <CR><LF>
```

Examples:

If an error of 10 KHz is observed on the instrument, the command should be used as follows:

```
sl_freq_offset=10<CR><LF>
```

If an error of -10 KHz is observed on the instrument the command should be used as follows:

```
sl_freq_offset=-10<CR><LF>
```

With the above example measurements in CW and Burst Mode, a Frequency Error of ~250 KHz is observed:

```
sl_freq_offset=-250<CR><LF>
```

Note: The user can use the above command multiple times to correct the frequency error till it gets tuned to desired frequency offset.

The user can provide the input to correct the frequency error by sending the commands to the console. This should lead towards a correction in the frequency offset as observed earlier and repeated till the error is within the tolerance limits (+/- 2 KHz tolerance limits).

After the frequency error is corrected, the impact may be observed by a spectrum analyzer measurement. If the results still doesn't meet the requirement, a new round of calibration can be processed.

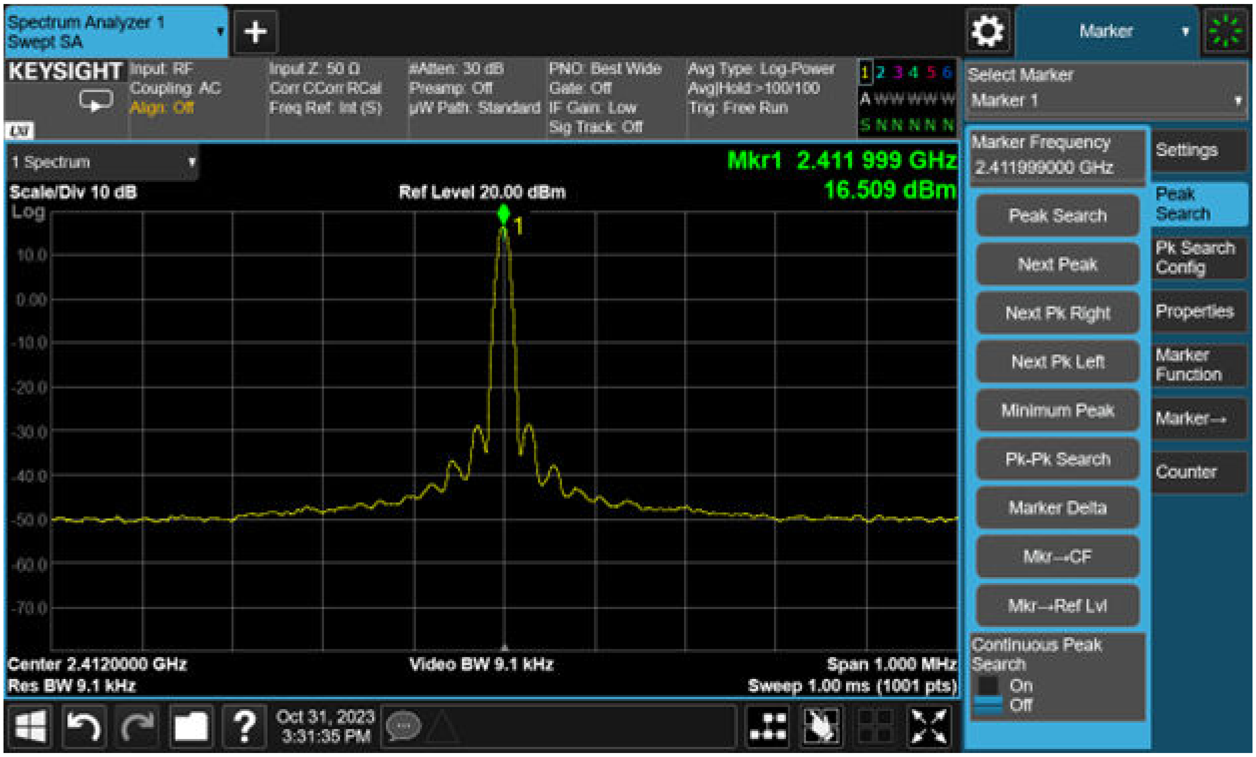


Figure 6.1. Corrected Frequency Offset Measured in CW Mode

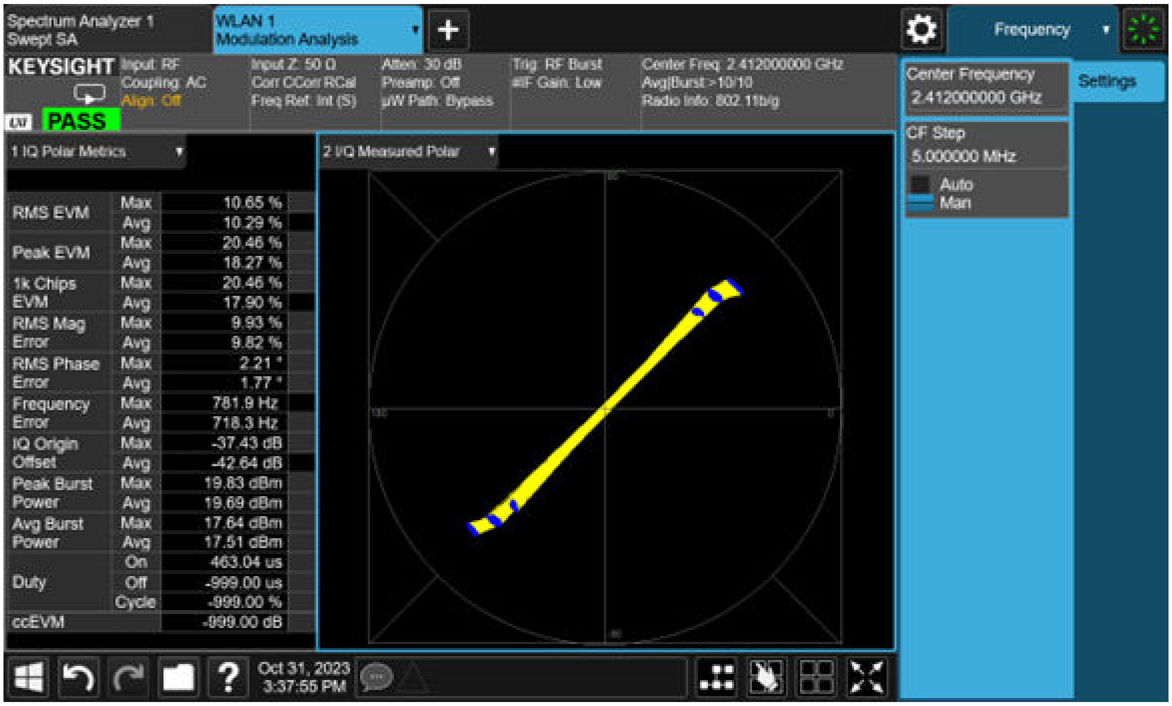


Figure 6.2. Corrected Frequency Offset Measured in Burst Mode

The corrected value of frequency offset is stored within a hardware register and for the correction to reflect upon reset, it is required that we copy the corrected value to flash. The value can be written to flash using the `sl_calib_write` command. The command should be used as follows: `sl_calib_write = 1,2,0,0,0`. Further details regarding the syntax of the `sl_calib_write` command are provided in the subsequent sections.

Using the `sl_calib_write` command, the calibrated XO Ctune and calculated gain offset can be updated to target memory (Flash/Efuse). Adjusting the gain offsets increases/reduces the average Tx power channel power in a particular channel. For further information about gain offset calibration, read through the gain offset calibration guide.

`sl_calib_write` command syntax:

```
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high>
```

OR

```
sl_calib_write=<target>,<flags>,<gain_offset_low>,<gain_offset_mid>,<gain_offset_high>,<xo_ctune>
```

6.1 Parameters

Parameter	Description		
target	**Value**	**Macro**	**Description**
	0	BURN_INTO_EFUSE	Burns calibration data to EFuse
	1	BURN_INTO_FLASH	Burns calibration data to Flash
flags	**BIT**	**Macro**	**Description**
	0		Reserved for future use
	1	BURN_FREQ_OFFSET	1 - Update XO Ctune to calibration data 0 - Skip XO Ctune update
	2	SW_XO_CTUNE_VALID	1- Use XO Ctune provided as argument to update calibration data 0 -Use XO Ctune value as read from hard-ware register
	3	N/A	
xo_ctune	This field allows the user to directly update xo_ctune value to calibration data bypassing the freq offset loop, valid only when BURN_FREQ_OFFSET & SW_XO_CTUNE_VALID of flags is set. The range of xo_ctune is [0, 255], and the typical value is 80.		

Precondition: sl_freq_offset command needs to be called before this command, when xo ctune value from hardware register is to be used.

7. Acronyms and Abbreviations

Acronym	Description
Tx	Transmit
Rx	Receive
RF	Radio Frequency
WLAN	Wireless Local Area Network
XO	Crystal Oscillator
Ctune	Captune

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com