Application Report
SNAA323–November 2018

Achieving Low Frequency Switchover Time for Wireless Infrastructure Applications

Akshat Garg, Avinash Dinesh Shylaja

ABSTRACT

RADAR systems use a Phase-locked loop to generate the local oscillator frequency to demodulate the incoming signal. Some multiband systems require extremely fast switching of the local oscillator frequency to capture every communication frequency and not miss any information in the frequency spectrum. Conventional systems switch between the frequencies by using Direct Digital Synthesizer (DDS) ICs, which are extremely fast but suffer from poor phase noise performance and have limitations of maximum frequency achievable. This application note presents an alternate approach to this problem using TI's LMX25xx family of wideband, RF synthesizers with PLLATINUM™ integrated circuits, which results in an overall reduction in lock time, a low power consumption, and a superior phase noise performance. This approach can be used in many Wireless Infrastructure applications.

Contents

List of Figures

Trademarks

PLLATINUM is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Prior Art to Switch the Oscillator Frequency

The most common way to switch the oscillator frequency is by using 2 PLLs locked to frequencies F1 and F2, respectively. This is then followed by a RF Mux switch (2:1). The RF mux switches the output between the frequencies and thus enables a quick switchover time. However, this solution requires a larger footprint and consumes more power due to extra components. One major drawback of this solution is the input-to-input signal isolation. It mainly depends of the RF switch and may lead to unwanted spurs in the output signal.

A use of DDS IC instead of an Analog PLL is also employed to switch the frequencies. However, the main drawback of this solution is the poor phase noise performance due to the digital architecture and unavailability of DDS ICs that support wide frequencies ranges like the LMX2594 (15 Ghz maximum).

2 Introduction

A Phase-locked loop (PLL) is a feedback system that generates an output frequency that is 'locked' (matched in frequency and phase) to the input reference frequency. They can be used to translate one frequency to another, attenuate jitter, or both, simultaneously. In this application note, there is one solution that can replace the prior art mentioned earlier to switch the oscillator frequency. The solution presented in this app note uses only one synthesizer to change between multiple frequencies when required. This approach not only overcomes the drawbacks of the previous systems mentioned, but also provides a few advantages of its own:

1. Reduction in power consumed.
2. Provides excellent phase noise performance.
3. Eliminates the need of using extra RF synthesizers, thus eliminating signal- signal isolation and cost reduction.
4. Can be implemented for a wide range of frequencies.
5. Frequency is completely software-configurable, which means that the user can lock to any frequency in the entire output range without changing any hardware.
6. Makes the system compact by minimizing number of components.
7. Provides a system that is very cost-efficient

To change the output frequency, a slave MCU is used to communicate with the Synthesizer over SPI.

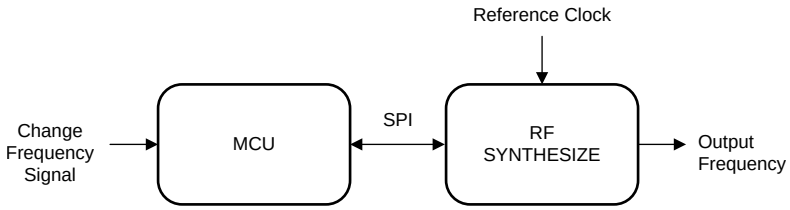


Figure 1. System Block Diagram

The LMX2572 is a 6.4-GHz, Low-Power, Wideband RF Synthesizer. It has a total of 126 registers (R0-R125) that govern the current state of the PLL. When the PLL is locked to a frequency, the values of some registers are unique to that frequency. This means that there are certain registers whose values are such that no two distinct frequencies can contain the same values.

3 Problem Statement

We now analyze the problem that occurs when a synthesizer plus MCU is used to transition between multiple frequencies.

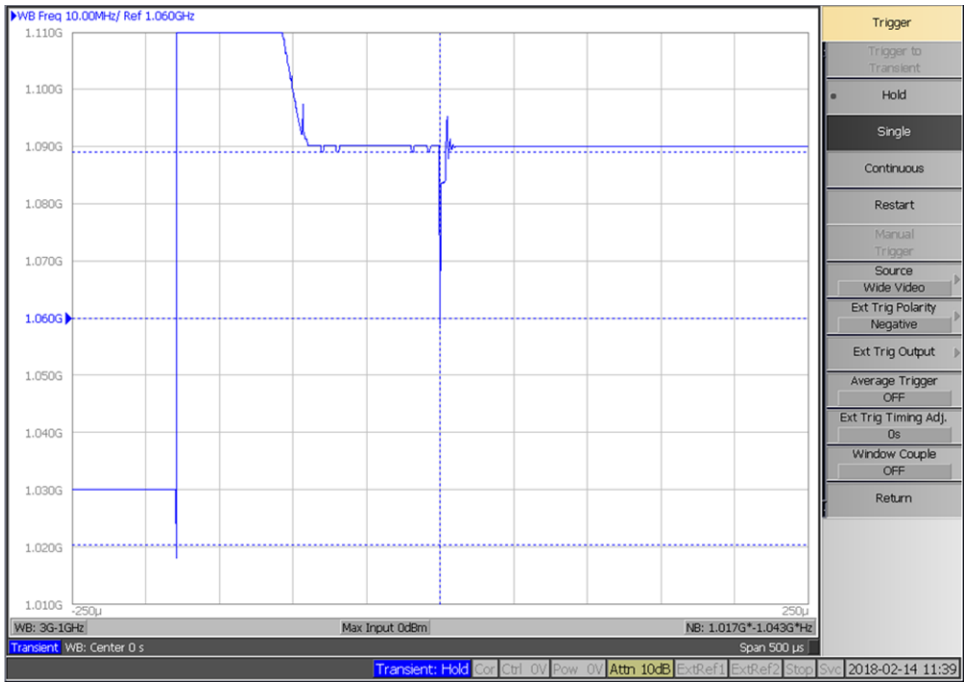


Figure 2. Total Lock Time of LMX2572 for Frequency Jump From 1.03 GHz to 1.09 GHz

Figure 2 shows the total frequency changeover time from 1.03 GHz to 1.09 GHz using the Auto calibration mode in the LMX2572, which is the typical mode employed when a Synthesizer is required to change frequencies. The total lock time to changeover from two frequencies f1 to f2 is influenced by the combined effect of three factors:

1. SPI Communication delay: This is the time taken by the controller to send across the frequency information over to the PLL. When there is a requirement to switch over from one frequency to another, the external controller sends the value of registers PLL_N (Register R36) and PLL_NUM (Register R43), which are the values of the frequency dividers, to the PLL. The following VCO calibration routine is initiated by programming the Register R0 (BIT[3]). This is limited by the SPI clock rate of the MCU. The LMX2572 can support SPI clocks of frequencies up to 75 MHz.
2. VCO Calibration delay: In the calibration routine, the PLL calculates:
   a. The correct VCO core (VCO_SEL),
   b. The best band within the core (VCO_CAPCTRL), and
   c. The best VCO amplitude setting (VCO_DACISET).

   Essentially, the PLL calculates the value of VCO_SEL (Register R20), VCO_CAPCTRL (Register R19), and VCO_DACISET (Register R16). This delay is the time taken by the PLL to calculate the above register values.
3. Analog PLL settling time: Following this calculation, the PLL's loop filter settles before the PLL locks to the new frequency. This way, the frequency transition is achieved. This is the time taken by the loop filter of the PLL to settle down. This is limited by the Loop filter bandwidth, which is determined by the external RC network.

The total time taken to transit from one frequency to another is the sum of the time delays due to the above three factors. This total time is around 190 µs (for the sample transition shown above), which is not acceptable in many applications.

4 Proposed Solution

As mentioned before, each frequency has a unique set of registers. This means that for a given frequency, the values of VCO_SEL (R20), VCO_CAPCTRL (R19), and VCO_DACISET (R16) registers are also unique (in addition to PLL_NUM, PLL_N, and PLL_DEN). Since our goal is to reduce the changeover time, the fully-assisted mode is employed, which involves recording these values beforehand during the production/initial programming of the product and storing them in a lookup table. When there is a need to switch to a particular frequency, the values of PLL_NUM (R43), PLL_N (R36), PLL_DEN (R39), VCO_SEL (R20), VCO_CAPCTRL (R19), and VCO_DACISET (R16) registers for that particular frequency are retrieved from the lookup table and are written over a high-frequency SPI Bus. Doing so allows us to bypass the time consumed by the PLL to calculate these values during the VCO calibration (in the auto calibration mode). This results in a dramatic reduction of the total lock time. Following this approach, the second factor influencing the total lock time (mentioned in the previous section) is eliminated. This leaves us with two delays: SPI delay and PLL Analog settling time delay.

To reduce the SPI delay, choose a high frequency SPI bus. The MSP430F2013 MCU that supports an SPI clock frequency of up to 16 MHz is set for this example. To reduce the PLL Analog settling time, the external RC network is designed to have a characteristic loop bandwidth of 1 MHz. As there is a clear tradeoff between the loop bandwidth and the phase noise performance (higher the bandwidth, poorer the phase noise), the user must design the loop filter such that they maximizes the bandwidth based on the system's phase noise spec.

5 Algorithm

As explained before, this method requires the prior values of PLL_NUM (R43), PLL_N (R36), PLL_DEN (R39), VCO_SEL (R20), VCO_CAPCTRL (R19), and VCO_DACISET (R16). The easiest way to obtain them is to run an Auto calibration routine on each device during the End equipment production or final Initialization and doing a read back of the values to store the result in the MCU flash.

This procedure is only required once for each device during production before the applications go in the field.

The algorithm written in the controller can be broken down as follows:

1. Wait for interrupt to change output frequency.
2. On the occasion of an interrupt, identify the desired output frequency.
3. Once the desired frequency has been identified, obtain the values of VCO_SEL, VCO_CAPCTRL, VCO_DACISET, PLL_N, and PLL_NUM for that frequency from the lookup table.
4. Send the register data along the SPI line:
   a. CPG –DISABLE (R14)
   b. VCO_SEL (R20)
   c. VCO_CAPCTRL (R19)
   d. VCO_DACISET (R16)
   e. PLL_N (R36)
   f. PLL_NUM (R43)
   g. PLL_DEN (R39)
   h. CPG –ENABLE (R14)
5. Once the output frequency has been locked, go back to step 1 and wait for next interrupt

It is very important to disable the charge pump (CPG) before any register writes.

The charge pump supplies current pulses to the loop filter. As the registers are written serially, at an intermediary instant in time when only a few registers have been written and a few are in progress, the PLL will try to lock to an intermediate frequency defined by the new values of the registers that have been written, along with the old values of the registers that have not been written. This may cause the PLL's output frequency to rapidly transition to one or more intermediate frequencies for brief instances, and then finally lock to the desired frequency. These sudden unwanted changes in frequencies may be unacceptable in certain applications. To mitigate this, the charge pump is disabled by setting the charge pump gain (CPG) to zero. Doing so minimizes these unwanted rapid transitions. After all registers are written, the charge pump is enabled.

Since the charge pump is enabled and then disabled, this comes at the cost of two additional register writes. This will marginally increase the SPI delay (and hence, the total lock time).

NOTE:

1. In some cases, the enable-disable of charge pump may not be necessary. TI advises to check the system performance with and without the CPG enable-disable register writes. Some frequency jumps can occur smoothly without interfering with the CPG values. In those cases, the SPI delay (and hence, the total lock time) can potentially be reduced by an amount (time period) equivalent to two register writes.
2. The above mentioned algorithm is generalized for two frequencies that differ in all the registers that are of concern in this example. This may not always be the case, however. For example, another demonstration shown further in this app note uses frequencies 1.0 GHz and 1.1 GHz. For a reference clock of 40 MHz, these two frequencies differ ONLY in PLL_N, VCO_CAPCTRL, and VCO_DACISET. This means that, in the register write required for a transition from 1.0 GHz to 1.1 GHz (or vice-versa), it is not necessary to send the values of VCO_SEL, PLL_NUM, and PLL_DEN because they are common for both the frequencies.

6 Frequencies Chosen

The two frequencies chosen are 1 GHz and 1.1 GHz. The register values recorded for these two frequencies (for a 40-MHz reference clock) are:

1 GHz:

VCO_CAPCTRL (R19): 0x132740

VCO_SEL (R20): 0x145448

DAC_ISET (R16): 0x10009E

PLL_N (R36): 0x240064

PLL_NUM (R43): 0x2B0000

PLL_DEN (R39): 0x2703E8

1.1 GHz:

VCO_CAPCTRL (R19): 0x132700

VCO_SEL (R20): 0x145448

DAC_ISET (R16): 0x10008F

PLL_N (R36): 0x24006E

PLL_NUM (R43): 0x2B0000

PLL_DEN (R39): 0x2703E8

As stated earlier, because the registers R20, R39, and R43 are common to 1 GHz and 1.1 GHz, it is not necessary to write these register values for a transition.

7 Results

The timing diagrams (Figure 3 and Figure 4) list the results of a register write.

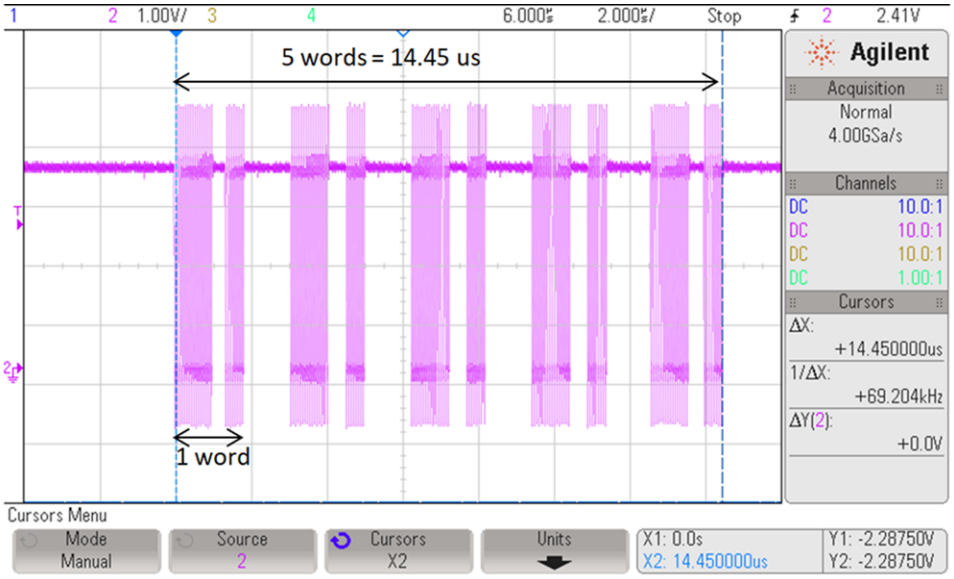


Figure 3. Timing Diagram: 5 Words

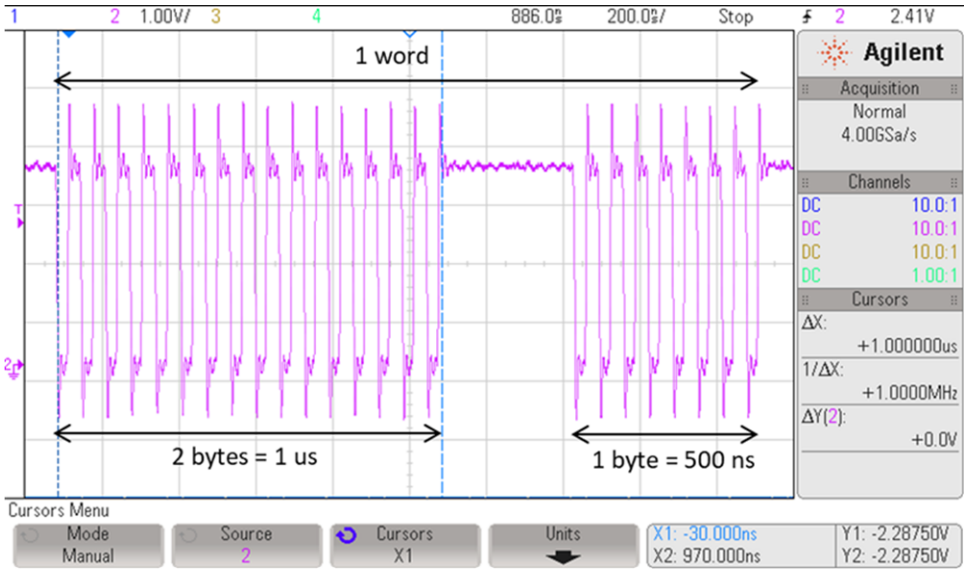


Figure 4. Timing Diagram: 1 Word: Close-Up

Each register is 24 bits long, but the transmit buffer is only 16 bits long. Hence, we split the register into 16 bits and 8 bits. As shown in Figure 3 and Figure 4, each register is split into lengths of 2 bytes and 1 byte and is written sequentially.

A simple MCU (MSP430) that has a maximum 16-MHz SPI clock was used in this example. The designer can also use an FPGA, however, to increase the SPI clock up to 75 MHz and reduce the SPI delay further by a factor of 5.

Auto calibration:

In this method, there is no assistance from the user for the calibration. The user just sends the value of PLL_N and triggers the VCO calibration.

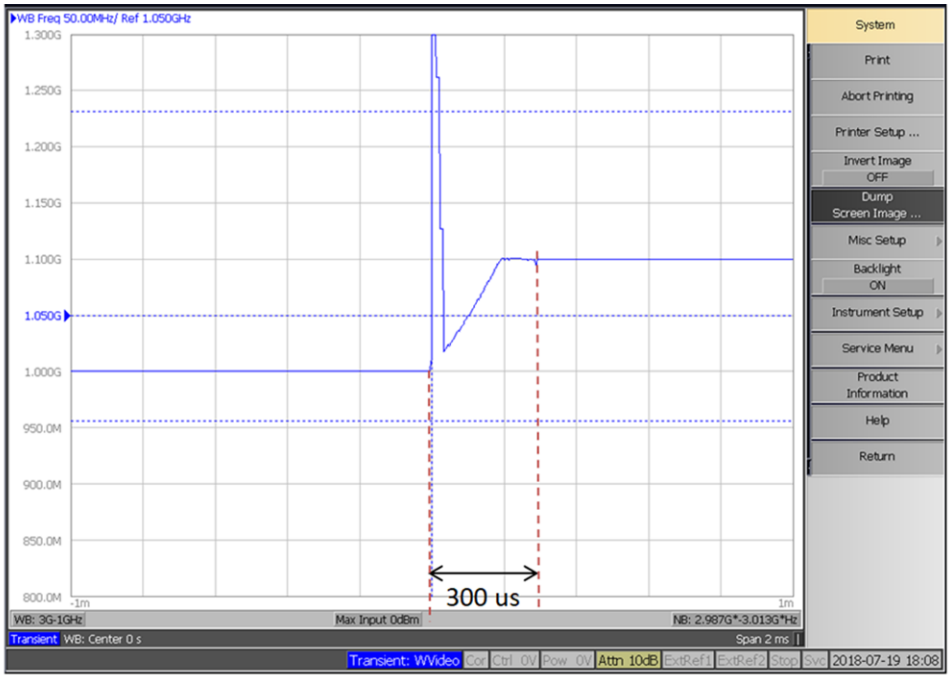
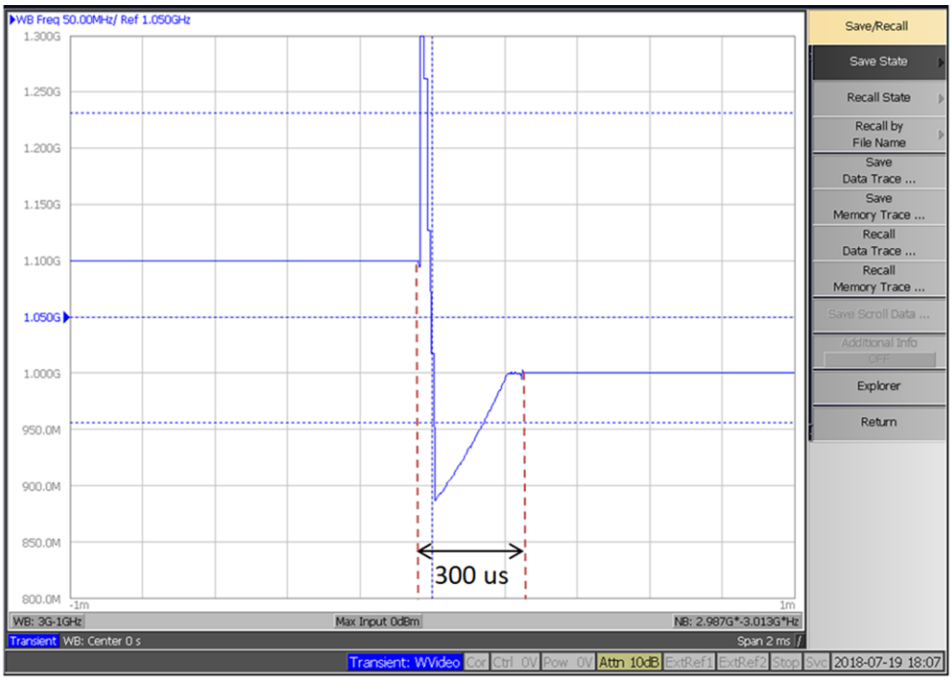


Figure 5. Auto Calibration- Frequency Transient Response for a Change From 1 GHz to 1.1 GHz



(1) Average lock time = 300 µs

Figure 6. Auto Calibration- Frequency Transient Response for a Change From 1.1 GHz to 1 GHz

Fully-Assisted Mode:

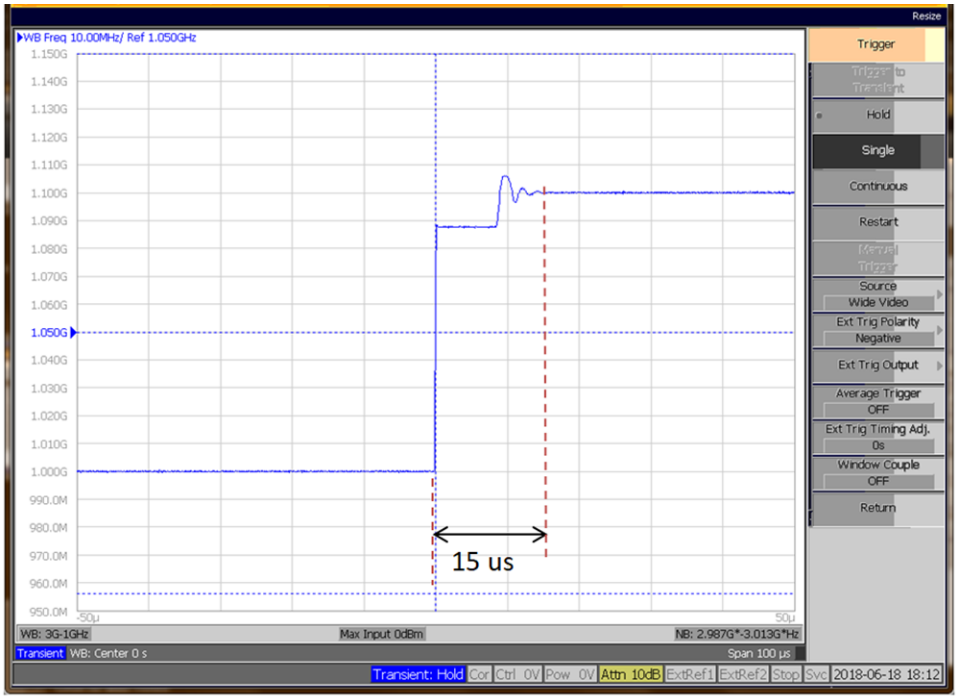
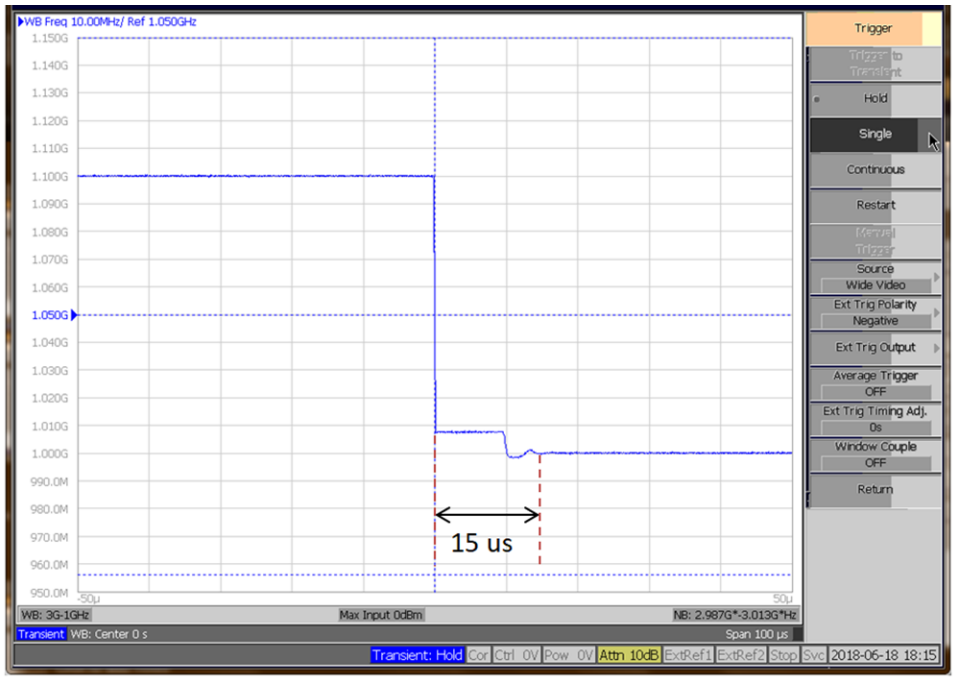


Figure 7. Fully-Assisted Mode: Frequency Transient Response for a Change From 1 GHz to 1.1 GHz



(1) Average lock time = 15 µs

Figure 8. Fully-Assisted Mode: Frequency Transient Response for a Change From 1.1 GHz to 1 GHz

Figure 7 and Figure 8 show the frequency transition plots between the two frequencies 1 GHz to 1.1 GHz, and vice-versa, with the fully-assisted mode. With this method, the total lock time was close to 15 µs, which mya be required in time-critical systems.

8 Conclusion

TI presents a cost-effective solution that employs a single synthesizer with numerous advantages, including a dramatically reduced lock time, minimal components, and ability to be employed for a large range of frequencies. The lock time for a transition between 1 GHz to 1.1 GHz was observed to be close to 15 µs with the MSP430. It can further reduce to around 5 µs to 8 µs by using an FPGA. This solution can very effectively replace the prior art that employs expensive DDS ICs and multiple synthesizers and multiplexers.

NOTE:

1. VCO FORCE – In the Fully-assisted mode, because the VCO_SEL, VCO_CAPCTRL, and VCO_DACISET are forced on to the chip, it is also necessary to enable the VCO_SEL_FORCE, VCO_CAPCTRL_FORCE, and VCO_DACISET_FORCE registers. Without doing this, the synthesizer will not take the register values sent over SPI and the PLL will not lock.
2. NO resistance in SPI lines- It is crucial to consider the resistance between the SPI (CS, SDA, SCL) lines from the controller to the synthesizer chip. These examples are operating at high frequencies (16 MHz), so any small resistance in the communication lines will form RC networks and distort the incoming signals. In this scenario, the synthesizer chip will not be able to accurately read the register values.
3. Debugging- If the PLL is not locking to the desired frequency, it is advisable to enable readback and read back the written values on to the TICS Pro GUI. This will enable the user to check what values are being read by the chip, and will also assist in debugging errors.

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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