Mini-Circuits®

ISC-2425-25+ Digital Locked Loop Apps Note

AN-50-005

www.minicircuits.com

1. Introduction

The Digital Locked Loop (DLL) is an algorithm on the ISC-2425-25+ board's microcontroller. It tunes the RF signal towards the best matched frequency within a user-defined frequency range. The DLL is suitable for driving real time, applications with changing impedance conditions. The algorithm utilizes a feedback loop, based around the forward and reflected microwave power of the RF channel.

2. Detailed explanation

2.1 RF HEATING PROCESS AND RF ENERGY DELIVERY

During an RF heating process, the user wants to make sure that the available RF energy is used optimally to heat up the load (that is some type of material). In a typical application, this is not a trivial thing to do: The frequency where energy is effectively transferred into the load is called a "match" and needs to be determined. A good match means most of the RF energy is absorbed by the load, whereas a bad match means a large portion of the energy is reflected back into the RF generator system.

A match can be expressed in different ways depending on the units used to measure RF power. When measuring the power using dBm, the match is expressed as 'S11':

$$S11 = (PRFL - PFWD)$$

Here, S11 is expressed in dB, and the powers P_{FWD} and P_{RFL} are expressed in dBm.
The more negative the value of S11, the better. For example, a value of about -13 dB would indicatethat more than 95% of the forward RF power is used in the cavity.

When measuring the power using watts the match is expressed as "Reflection":

$$Reflection = (P_{RFL} / P_{FWD})$$

Here the reflection is a value typically between 0 and 1, Reflection is expressed as a percentage (%)and the powers P_{FWD} and P_{RFL} are expressed in watt. The closer the reflection is to 0%, the better.

2.2 A REAL PROCESS EXAMPLE

Figure 1 below depicts an example process, which would have its optimum energy delivery point (i.e., "best match") at 2450 MHz. The user would want to start the heating process at an operating frequency of 2450 MHz, with a match of about -18 dB. However, in real processes, this optimum frequency is not fixed and it changes depending on the impedance changes of the system which can be caused by the heating up of the load. The user needs to follow that frequency change – andthat's exactly where the DLL algorithm comes in to help.

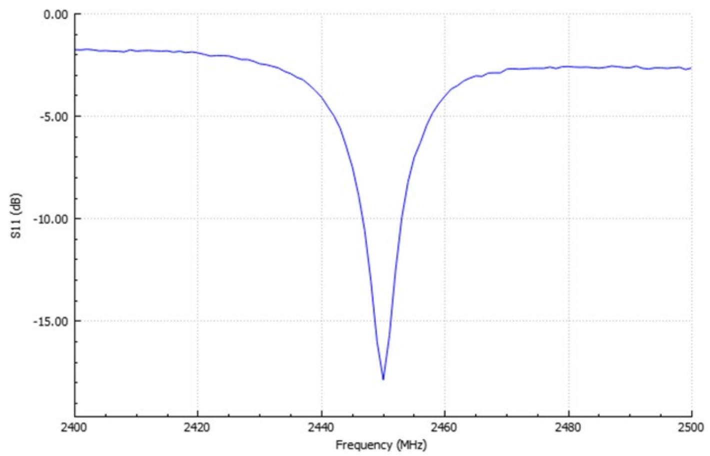

Figure 1: S11 Versus Frequency plot

2.3 EXPLANATION DLL OPERATION

The DLL operates in 2-stages:

First stage: Acquisition

During the acquisition stage, the DLL probes frequencies across the user-defined frequency range, seeking the first frequency which meets the user-defined return loss threshold (dB). (See **Figure 2**). The algorithm moves from high to low frequency beginning at the user-defined starting frequency and will loop around indefinitely if no suitable match is found. When a suitable frequency is found which meets the threshold (i.e., local minimum), DLL immediately proceeds to the second stage.

Second stage: Tracking & Fine tuning

During the tracking stage DLL's goal is two-fold:

1. Fine-tune the frequency towards the best-matched frequency in the local minimum.
2. Remain locked to a frequency which meets the threshold condition for return loss.

During this stage the current frequency will be referred to as the 'locked' frequency and thefrequencies directly adjacent to the center will be called its 'neighbor' frequencies.

The DLL algorithm probes the neighboring frequencies to check if any of them measure a better match than the center frequency. If this is the case, the DLL algorithm shifts the locked frequency towards the new 'sweet spot'. In this way DLL quickly crawls towards the best-matched frequencyof the current minimum.

It is common that certain RF loads may have changing impedance conditions which will cause the best-matched frequency of the RF system to shift along the frequency band. By continuously comparing the locked frequency to its neighbors, the DLL algorithm can DYNAMICALLY follow the changing impedance conditions of its load, while retaining the best energy delivery efficiency for the process.

If the locked frequency's match can no longer satisfy the return loss threshold, the DLL algorithm reverts to the Acquisition stage.

Remark: both DLL phases are run with the output power set prior to engaging the DLL. If no match is found, the DLL will remain in the acquisition phase at that power level. This may stress the RF system considerably and may not lead to the intended process result. To avoid this situation, it is advised to execute a sweep command at lower output power levels to characterize the overall system status and to find the proper DLL threshold limit. This way the DLL can start with the center frequency already below the DLL threshold value and it will go directly into the second Tracking &Fine Tuning phase.

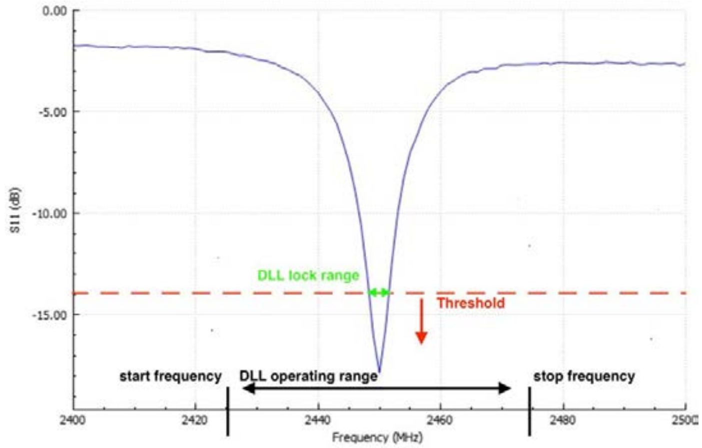

Figure 2: S11 Versus Frequency (with DLL operating and lock range highlighted)

TIP: Alternatively, the DLL can be configured with the threshold value set to 0. Then virtually any match will meet the DLL's threshold and the algorithm will immediately proceed to the 2nd stage. It will still shift towards the best match frequency it can find, but with the downside that it may lock onto non-ideal local minima or high reflection frequencies. This latter problem can be mitigated by performing periodic frequency sweeps (see $SWP command) to keep track of the overall processbehavior.

2.4 IN PRACTICE

It is advised to first characterize the cavity/load condition of the process with the help of the $SWP (S11 sweep) command ($SWP, channel, start_frequ,stop_frequ, frequ_step, power[W], mode). E.g., $SWP, 1,2400, 2500, 10, 50, 1. It returns either the complete dataset for S11 values against frequency (like in the figure above with “mode” = 0) or just the “best match” frequency with its Reflection value (in % for “mode” = 1). The user can then determine whether the Reflection value is good enough to **run the process or whether some changes need to be made**.

If the Reflection is good enough, the DLL can be configured, be enabled and RF power started as follows (this assumes that prior to this sequence the user has already set the intended output power for the system):

```
$DLCS,1,2400,2500,2440,1,0.0,0
$DLCS,1,OK
$DLES,1,1
$DLES,1,OK
$ECS,1,1
$ECS,1,OK
```

RF output power and the DLL are now enabled, and the DLL is tracking the optimum S11. The momentarily used “locked” frequency for the DLL can be viewed using the frequency get command:($FCG, 1).

It’s now up to the user to determine if the process conditions remain meaningful over time for their specific application.

Have fun developing your own application algorithm :)