Memorandum

To: File

From: J. Effland

Date: 15 April 2002

Subject: Software Design for Automated LO

Software responsible for setting the frequency and power level of the automated LO is documented in this memo. Test results using a similar YIG oscillator is also included.

The frequency of the automated LO plate is determined from a YIG oscillator, whose output is multiplied by a factor of 16 using a cascade of multipliers and amplifiers that was designed by Eric Bryerton. Software is required to set the YIG frequency, and to phase lock the frequency using the EIP 578 source-locking counter. The software driver to change frequency and power is included as the classes `CLODriver` and `CEIP578` in the Visual Basic instrument control library starting with build 5.2.42.

Test Setup

The frequency of the automated LO plate is determined by a YIG oscillator from Micro Lambda, model MLOB-1563MG, which tunes from 12 - 17 GHz. Because the wiring for that YIG hasn't been completed, software testing was performed using an Omniyig YOM1818-17DD that tunes from 8 - 18 GHz. The test setup is shown in Figure 1. The Omniyig oscillator is part of S.-K. Pan's millimeter wave source plate, but has similar characteristics to the Micro Lambda YIG.

Corrected open-loop YIG frequency Algorithm

The initial design of the LO driver simply set the YIG oscillator frequency to the desired value and commanded the EIP 578 counter to phase lock to this frequency. The problem with this approach is that the YIG's open loop frequency error from the desired frequency can be a large as 20 MHz, and that exceeds the EIP 578's ability to reliably acquire phase lock. To improve the reliability of phase locking the YIG to the EIP counter, the software was modified as shown in Figure 2 to measure the open-loop frequency of the YIG, calculate a corrected frequency, and command the YIG to that frequency. A final frequency check after phase locking is also included.

Figure 3 is the UML sequence diagram for setting the LO frequency with this implementation. The sequence diagram identifies the commands and their order as they are passed between objects, with the first command starting at the top of the figure. As first shown in Figure 3, each object is initialized, then the object `oLODriver` commands the YIG object `oYIG` to set the initial YIG frequency. Next `oLODriver` corrects the YIG frequency error by measuring the frequency with the counter and computing a corrected frequency from the difference between the desired and measured frequencies. This YIG frequency measurement is refined by repeatedly measuring the frequency until the standard error (using object `oStandardDeviation`) is below 0.01%. The mean YIG frequency is returned after the standard error is sufficiently low. To speed up the multiple reads of the counter, which are required to compute the standard error, counter resolution is reduced from 1 Hz to 100 kHz during this time. The YIG is then commanded to a corrected frequency, the counter resolution is reset to 1 Hz, and the counter is commanded to phase lock to the desired frequency.

After the GPIB status byte confirms that the counter is phase locked, the counter is read one last time and the phase lock frequency is compared to the desired frequency.

Figure 4 shows frequency errors after implementing the software that corrects the YIG frequency prior to closing the phase lock loop and reads the final frequency after phase lock. Fourteen out of 3000 (0.5%) frequency points were in error by as much as 350 MHz from the desired frequency. Further attempts to decrease the error by restarting the loop failed and the final frequency was always in error by the same amount. The failure rate remained unchanged for two different samples of the EIP 578 counter.

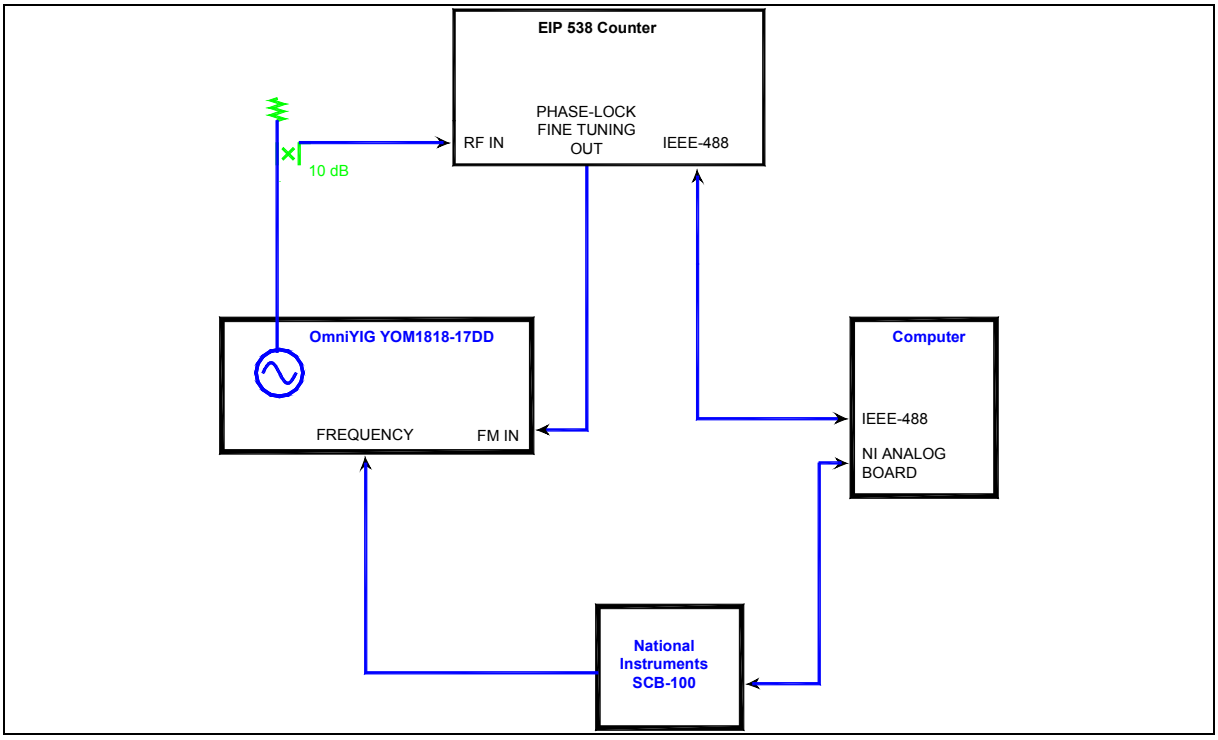

Figure 1: Test Setup for Software Evaluation

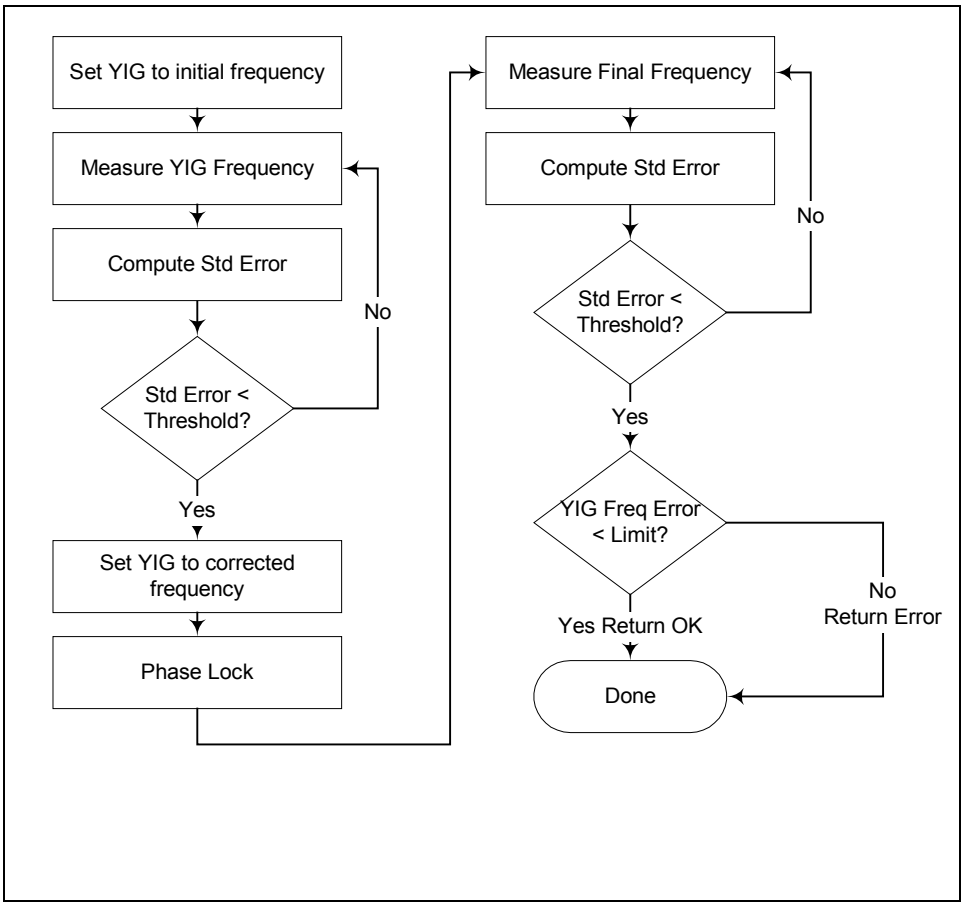

Figure 2: Flow Diagram

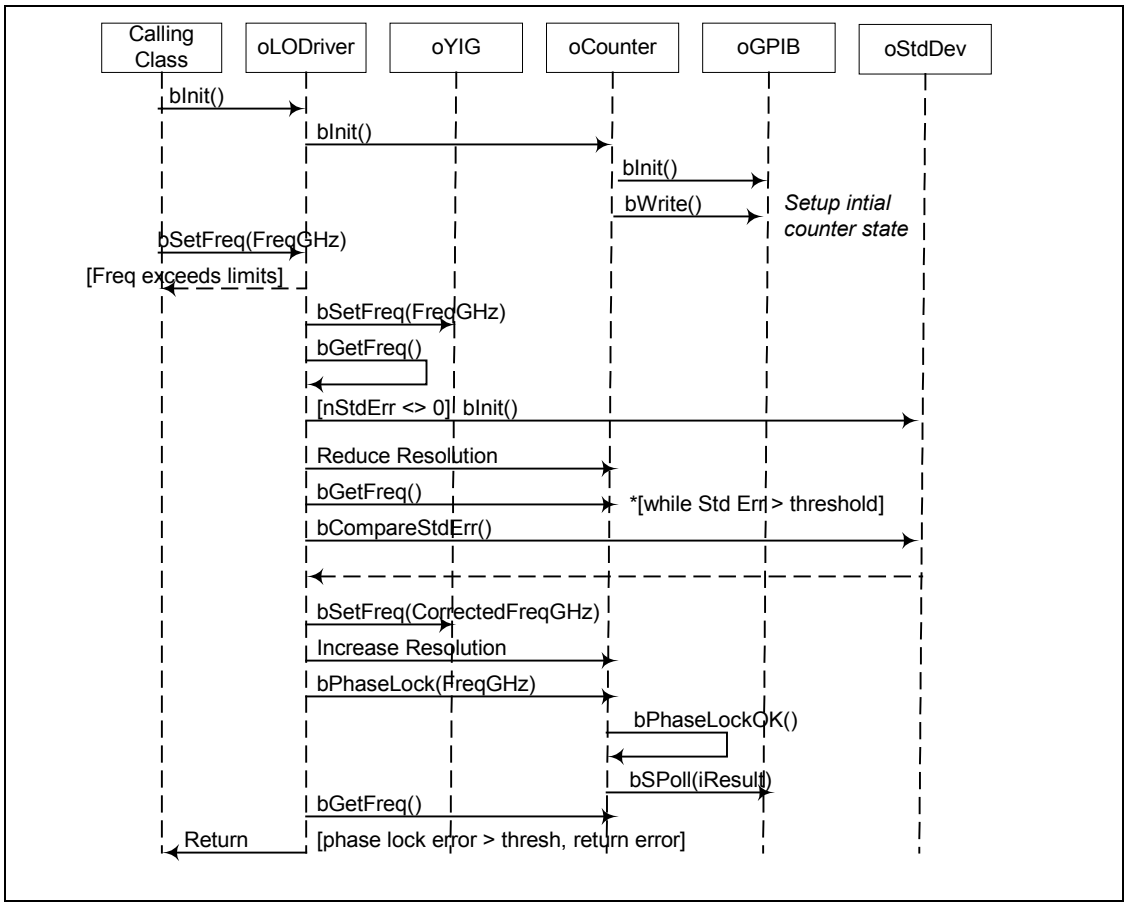

Figure 3: Sequence diagram for setting the LO frequency

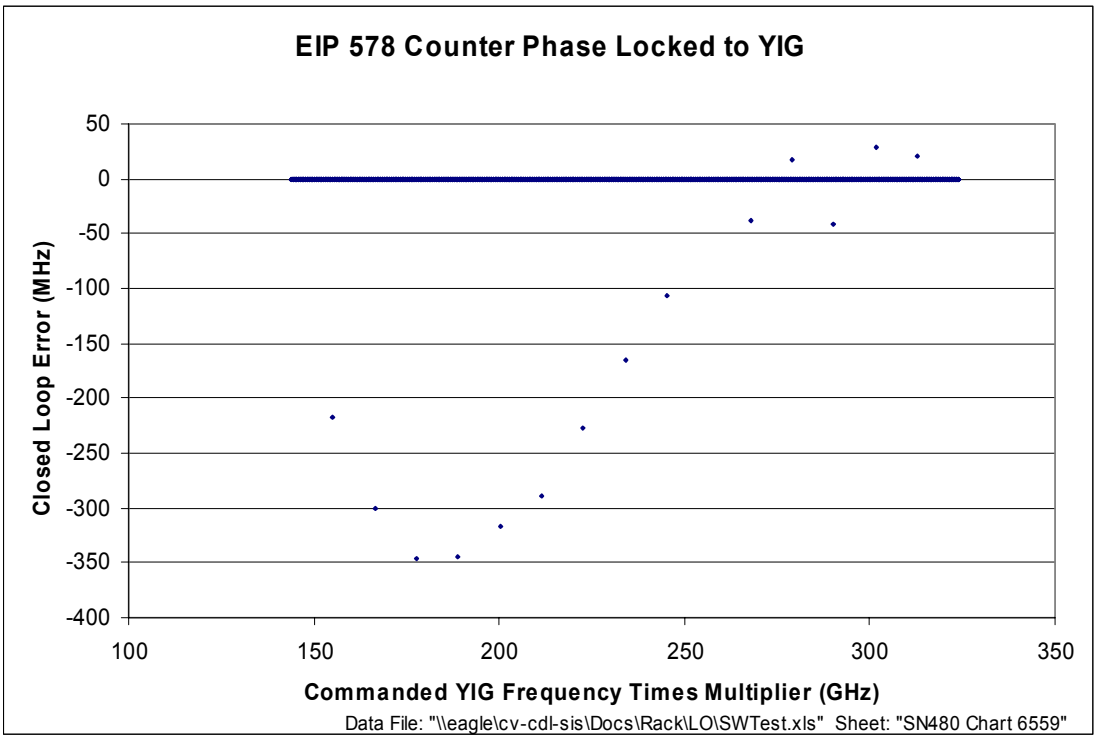

Figure 4: Frequency error after Loop Phase Lock