Memorandum

To: Alex Grichener

cc: A. R. Kerr
John Webber
John Hibbard
S.-K. Pan
Dan Koller

From: John Effland

Date: 4 June 2003

Subject: Development of a System to Measure Noise Powers with a Chopper Wheel and Fast Power Meter Measurements

Purpose

The existing SIS mixer noise measurement system uses an NRAO-designed square law detector to measure noise powers when the receiver input is alternately connected using a chopper wheel to a hot and cold load. The chopper wheel rotates at 12 revolutions per second, and because there are two hot/cold load positions per rotation, the effective chopping rate is 24 measurements per second. The voltage output from the square law detector is read in synchronism with the computer by a National Instruments AT-MIO-16DE10 data acquisition board[1].

In the existing system, the chopper wheel and square law detector approach provides real-time Y-factors and noise temperatures to a PC-based chart recorder so the operator can manually optimize the mixer operating point. During actual receiver noise temperature measurements, a power meter is used to measure the noise power instead of the square law detector. The measurement system would be simplified if the square law detector and its complicated triggering mechanism could be replaced by the power meter for all power measurements.

The existing power meter, and HP 436A model, can generate power measurements at a maximum rate of two per second. This is insufficient for fast data acquisition using the chopper wheel. Consequently, the newer Agilent E4418 power meter, which can measure power at the rate of 200 measurements per second, will be used as the measurement instrument.

This document describes a system to use the power meter with the chopper wheel for all receiver noise temperature measurements. Hardware configuration, top-level software design, and a Statement of Work are included.

Hardware Configuration

Figure 1 shows the connections between the computer, power meter, and chopper wheel. Triggering of the power meter must occur indirectly, using GPIB commands, because the power meter has no inputs for hardware triggering. The chopper wheel generates trigger pulses using a trigger board and optical interrupters at

[1] "Coax Switch Controller, Refrigerator Controller, and Chopper Wheel Controller Design Document", Internal NRAO report from J. Effland dated 2000-09-21 and available at http://www.cv.nrao.edu/~jeffland/SwitchControlHardwareDesign.pdf

predetermined locations relative to its hot and cold load positions, which are installed on the chopper wheel assembly. The trigger board is designed so that the first trigger pulse always corresponds to the hot load position which provides a means for the software to synchronize to the hot and cold load positions of the chopper.

The software must wait for a trigger pulse from the chopper wheel, then send a GPIB command to the power meter to commence data acquisition. The power meter program will collect samples during most of the time the chopper wheel is in either the hot and cold load positions. The program pauses data acquisition only during guard times while the chopper changes states from hot to cold load. The software will return the average absolute power reading, and not just the relative power difference, for both the hot and cold load positions.

It would be ideal for the power meter, rather than the computer, to return the average of N to minimize traffic on the GPIB. However, the power meter can't calculate and return standard error values, and the standard error is essential for confirming that the data have settled. Consequently, each individual power measurement will be returned to the computer and the software will calculate the mean and standard error. The software continues reading power measurements until the standard error is below a predefined threshold (or a predetermined number of measurements has been made), after which the software returns the average power measurement and the calculated standard error.

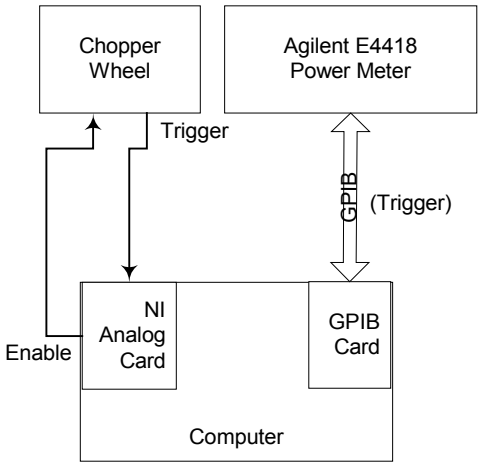

Figure 1: Power Meter Triggering and Control

Software Design

The software will consist of a number of classes designed according to the structure shown in Figure 2. For clarity, this class diagram shows only a subset of the class members and methods required. The base class, `CGPIB`, contains low-level calls to the GPIB card. The class `CHP4418` is built on `CGPIB` and provides an interface for a specific type of Agilent power meter.

`CPwrMtr` is a generic power meter class that provides a general interface for other modules requiring power meter functions. The class `CstdDev` is included as a member in class `CPwrMtr` to allow the power meter to repeatedly take measurements until the standard error, as calculated by `CstdDev`, is below a specific threshold. The method `Read`() returns the average power measurement only after the standard error is below the threshold, or after the number of measurements exceeds a predefined maximum, as specified by the class member `m_lMaxBufferSize`.

The top-level class, `CNoiseMeas`, returns average absolute power levels when the receiver beam is directed towards the hot and then towards the cold load by synchronizing power meter measurements with the chopper wheel state

using the class `CChopper`. The methods `GetPhot()` and `GetPcold()` return the average absolute powers in milliwatts for the hot and cold load states after the standard error is below a specified threshold.

The classes will be coded using Visual Basic .Net (VB.Net) compiler, but they will need to be called from legacy Visual Basic Version 6 (VB.6) code that exists both in stand-along Visual Basic programs and in Excel macros. That requires providing a software layer that encloses the .Net code in a COM wrapper so the Visual Basic 6 routines see these new classes as ordinary COM-compatible classes in an Active-X dynamic link library.

Maintaining VB6 compatibility means that many methods require two names for essentially the same function because VB6 can't use the exception-based `Try…Catch` error handling that's now available to VB.Net. Instead, calls intended for VB6 routines must test for errors by checking the state of the boolean value returned with the function. This is accomplished by using two names, such as `bRead()` and `Read()`. The method `bRead()` returns `TRUE` if the call was successful while `Read()` and is intended to be used by VB.Net's exception-based error handling construct:

```
Try

       Read(nPowerMean, dStdErr)

Catch ex as Exception

       M_sError = "Error in module…" & ex.message
       Throw New System.Exception(m_sError)

End Try
```

Statement of Work

The student should complete following tasks:

1. Port the existing `CHP4418` power meter class and associated sub classes to VB.Net and write routines to use the 200 measurements per second mode of the power meter.
2. Document the software design with UML class, activity, and sequence diagrams.
3. Configure the National Instruments AT-MIO-16DE10 analog card and associated classes to generate an enable signal and receive the trigger signal from the chopper wheel.
4. Design, document, code, and test the `CPwrMtr` and `CNoiseMeas` classes, including robust error handling for all classes, using the class diagrams shown in Figure 2.
5. Connect the hardware to the mixer measurement system and record noise powers using an actual SIS mixer.

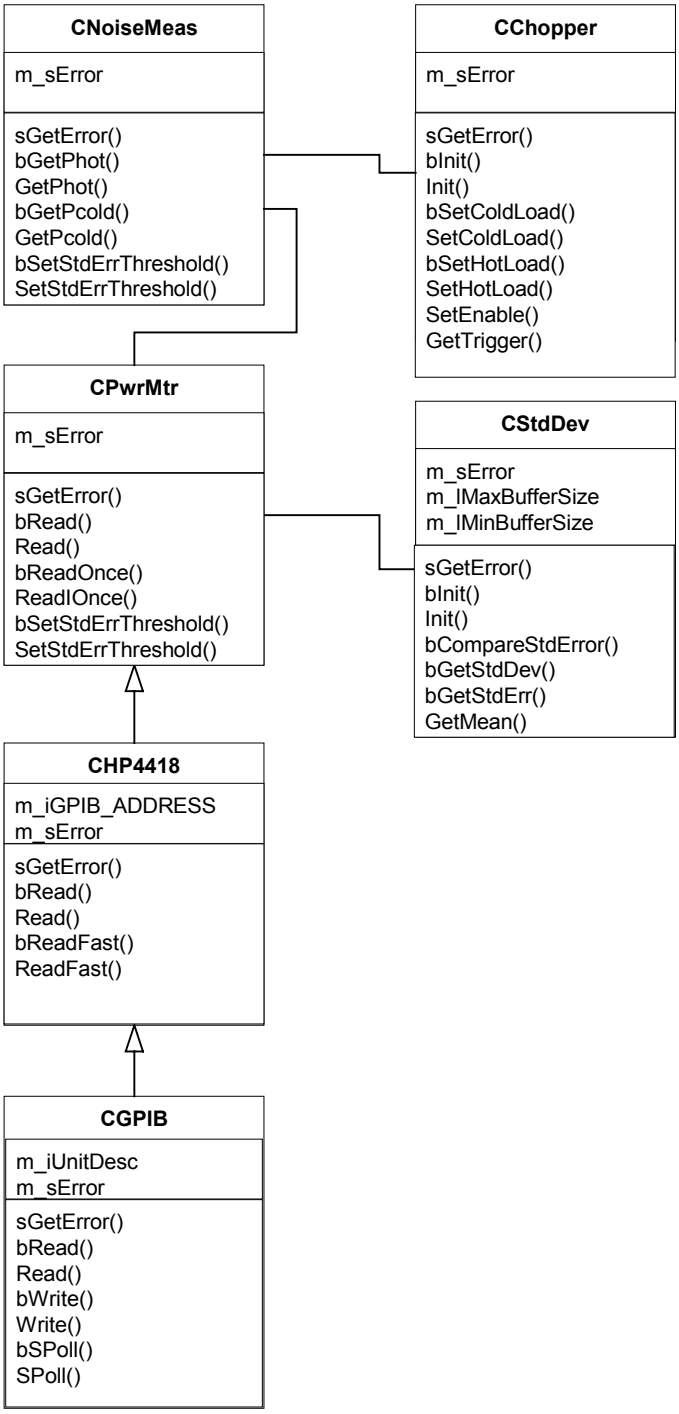

Figure 2: Class Diagram for Chopper Synchronized Noise Measurement Class