10-19-98

Memorandum

To: A. R. Kerr
S. -K. Pan
K. Crady

cc: J. Webber

From: J. Effland

Date: 19 October 1998

Subject: Status Report for Automating SIS Mixer Measurements

1. Accomplishments this period

Software was written to programmatically record data in the appropriate cells on the spreadsheet that is currently used to store and analyze mixer performance. This memo contains sections from the software design document describing the design and another section providing details of tasks remaining to complete this effort. Finally, a prototype data acquisition dialog box is described.

2. Additional Tasks

The software can be used in its current form to collect mixer data on an Excel spreadsheet, but additional effort is required to minimize operator intervention. The following tasks will be accomplished in the next week:

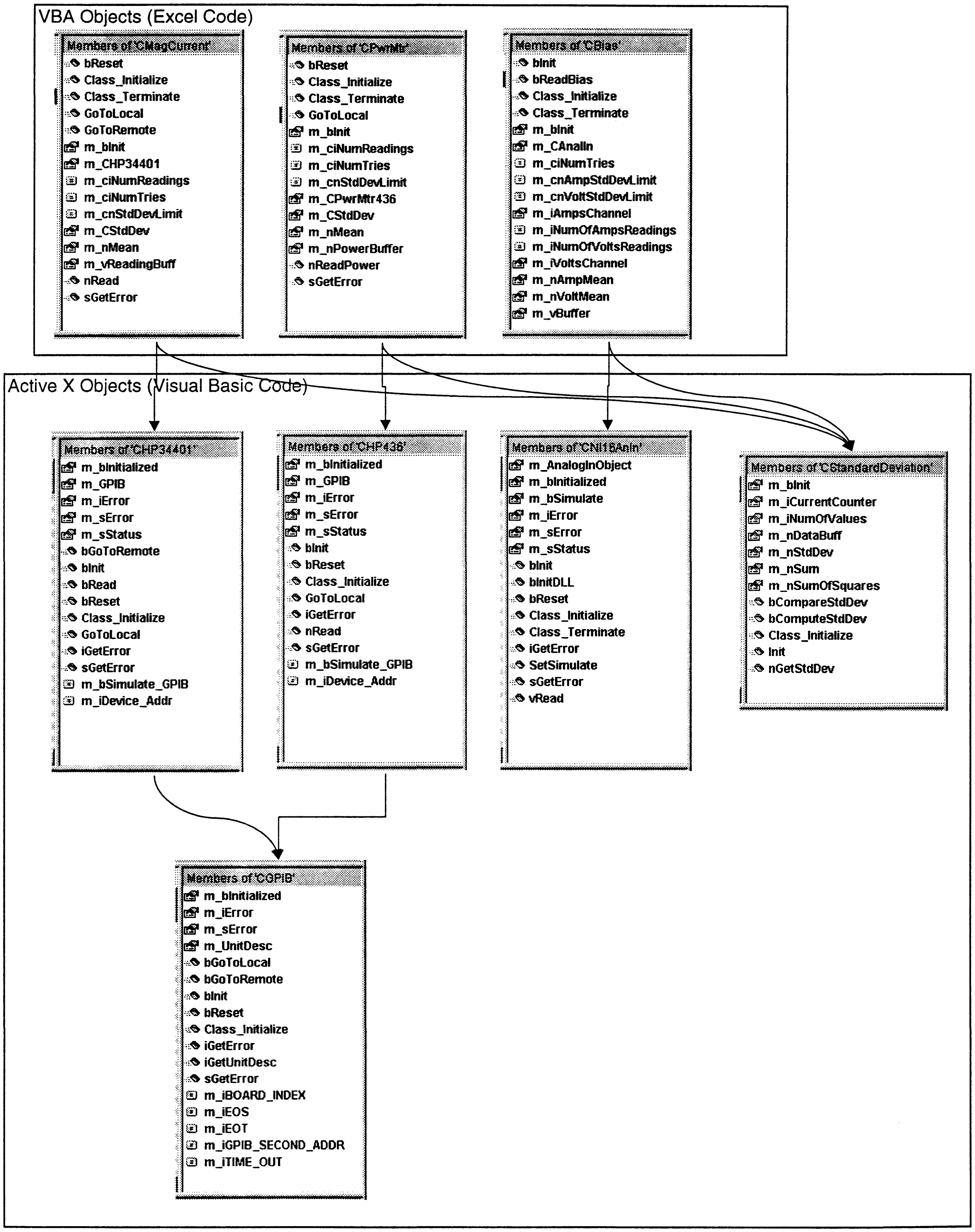
1. Create graphs in Excel for mixer noise temperature and loss. These graphs will duplicate those used in the current Quattro Pro spreadsheet.
2. Complete coding and testing of instrument error routines. For example, the software should alert the operator if the instrument doesn't respond because it is either turned off or the HPIB cable is unplugged.
3. Wire the National Instruments analog board to record bias voltage and current from both balanced mixers. This will eliminate manual switching that's currently required.
4. Test Kirk's hardware that allows software to switch between hot and cold IF loads, and write the software to effect the switching.

3. Overall Software Architecture

The software is partitioned into two program sets:

1. Routines that directly control the equipment are written as an "Active X" module in the stand-alone Visual Basic (VB) program.
2. Routines that average a number of measurements from each instrument and test for sufficiently small standard deviation are written in the Visual Basic for Applications (VBA), which is the version of Visual Basic included with Excel. The VBA code calls the Active X routines from Excel.

These routines are diagrammed in the class diagram shown in Figure 1 and described in the following sections.


VBA Objects (Excel Code)

Members of 'CMagCurrent'
- bReset
- Class_Initialize
- Class_Terminate
- GoToLocal
- GoToRemote
- m_bInit
- m_CHP34401
- m_ciNumReadings
- m_ciNumTries
- m_cnStdDevLimit
- m_CStdDev
- m_nMean
- m_vReadingBuff
- nRead
- sGetError

Members of 'CPwrMtr'
- bReset
- Class_Initialize
- Class_Terminate
- GoToLocal
- m_bInit
- m_ciNumReadings
- m_ciNumTries
- m_cnStdDevLimit
- m_CPwrMtr436
- m_CStdDev
- m_nMean
- m_nPowerBuffer
- nReadPower
- sGetError

Members of 'CBias'
- bInit
- bReadBias
- Class_Initialize
- Class_Terminate
- m_bInit
- m_CAnalIn
- m_ciNumTries
- m_cnAmpStdDevLimit
- m_cnVoltStdDevLimit
- m_iAmpsChannel
- m_iNumOfAmpsReadings
- m_iNumOfVoltsReadings
- m_iVoltsChannel
- m_nAmpMean
- m_nVoltMean
- m_vBuffer

Active X Objects (Visual Basic Code)

Members of 'CHP34401'
- m_bInitialized
- m_GPIB
- m_iError
- m_sError
- m_sStatus
- bGoToRemote
- bInit
- bRead
- bReset
- Class_Initialize
- GoToLocal
- iGetError
- sGetError
- m_bSimulate_GPIB
- m_iDevice_Addr

Members of 'CHP436'
- m_bInitialized
- m_GPIB
- m_iError
- m_sError
- m_sStatus
- bInit
- bReset
- Class_Initialize
- GoToLocal
- iGetError
- nRead
- sGetError
- m_bSimulate_GPIB
- m_iDevice_Addr

Members of 'CNI16AnIn'
- m_AnalogInObject
- m_bInitialized
- m_bSimulate
- m_iError
- m_sError
- m_sStatus
- bInit
- bInitDLL
- bReset
- Class_Initialize
- Class_Terminate
- iGetError
- SetSimulate
- sGetError
- vRead

Members of 'CStandardDeviation'
- m_bInit
- m_iCurrentCounter
- m_iNumOfValues
- m_nDataBuff
- m_nStdDev
- m_nSum
- m_nSumOfSquares
- bCompareStdDev
- bComputeStdDev
- Class_Initialize
- Init
- nGetStdDev

Members of 'CGPIB'
- m_bInitialized
- m_iError
- m_sError
- m_UnitDesc
- bGoToLocal
- bGoToRemote
- bInit
- bReset
- Class_Initialize
- iGetError
- iGetUnitDesc
- sGetError
- m_iBOARD_INDEX
- m_iEOS
- m_iEOT
- m_iGPIB_SECOND_ADDR
- m_iTIME_OUT

3.1 Visual Basic Active X Routines

The `CGPIB` bass class interfaces with National Instrument's GPIB board routines and includes functions such as:

- `bInit` – initializes the GPIB board for a particular instrument. This routine maps GPIB addresses to "unit descriptions", which identify all subsequent calls to the instrument.
- `bReset` – resets all instruments on the GPIB bus.

Parents of the `CGPIB` class contain specific functions for each instrument. These include:

- `CHP34401` – controls the HP 34401 multimeter that measures the magnet current.
- `CHP436` –controls the HP 436 power meter that measures noise powers of the mixer and IF system.

Interface to National Instrument's AT-MIO-16DE-10 analog interface board is provided through the class:

- `CNI16AnIn` – controls the, which reads mixer bias voltage and current.

An additional class, `CDtandardDeviation`, computes the standard deviation and contains two routines:

- `BComputeStdDev` – computes the standard deviation given a list of n data points
- `BCompareStdDev` – compares the standard deviation to a threshold and returns `True` when the standard deviation is less than the threshold. This object can be called after each measurement (rather than at the end of n measurements), and returns `False` until the specified number of data points that are required to compute the mean has been measured.

All of these routines are contained in an "Active X" dynamic link library, which retains the same programmatic interface as a normal windows dynamic link library, but includes additional operating system overhead (by using globally unique identifiers stored in the registry) to provide a form of automatic version control.

3.2 Routines Coded with Excel Visual Basic for Applications

Software written in VBA, like the VB software, also includes classes for each instrument, but the Excel VBA routines are written in a more general way to provide independence from the particular instrument used. Each instrument class also contains functions and data members to compute the standard deviation and compare it to a preset value. The VBA routines contain only *objects* for the standard deviation as well as the comparison code: The standard deviation *class* (`CStandardDeviation`) is defined in the Active X dynamic link library written in VB.

The following classes are defined in the VBA routines:

- `CMagCurrent` – reads the magnet current.
- `CPwrMtr` – reads the power meter.
- `CBias` – returns bias voltage and current.
- `frmDataCollection` – this is an object of type "form", which is essentially the same as a class, and it contains routines to initiate reading each instrument and placing the data into the appropriate cell.

Figure 2 is the dialog box available from a menu in Excel that is used for mixer data acquisition. The "Frequency" section increments or decrements the active spreadsheet row, because each row corresponds to a new frequency, and the current frequency is read from the leftmost cell in the spreadsheet row. The dialog box is designed so that each time the Enter key is pressed; the button with "focus" is advanced to the next relevant button. (A button that has "focus" activates its associated function when the Enter key is pressed). This simplifies data collection because the operator needs only to press either Enter key to read the appropriate instrument, store the data on the spreadsheet, and advance the focus to the next instrument. (At this point, switches must be manually configured for each measurement, but future enhancements will automate all switching.)

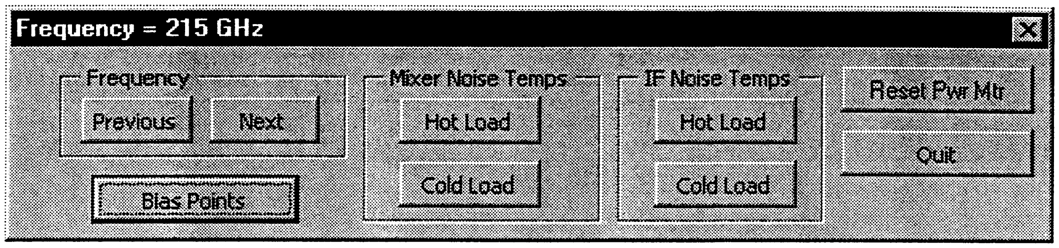


Figure 2: Excel Dialog Box for Data Acquisition

3.2.1 Computation of Standard Deviation

When each button in Figure 2 is pressed, the instrument is read 10 times, then the standard deviation is computed and compared to a threshold (currently 0.1 for all measurements). If the standard deviation exceeds the threshold, the instrument is reread, and the newest measurement replaces the oldest reading prior to again computing and comparing the standard deviation. The rereading process is repeated up to 100 times, after which the program continues by recording the value on the spreadsheet, and warning the user that the standard deviation exceeds limits.

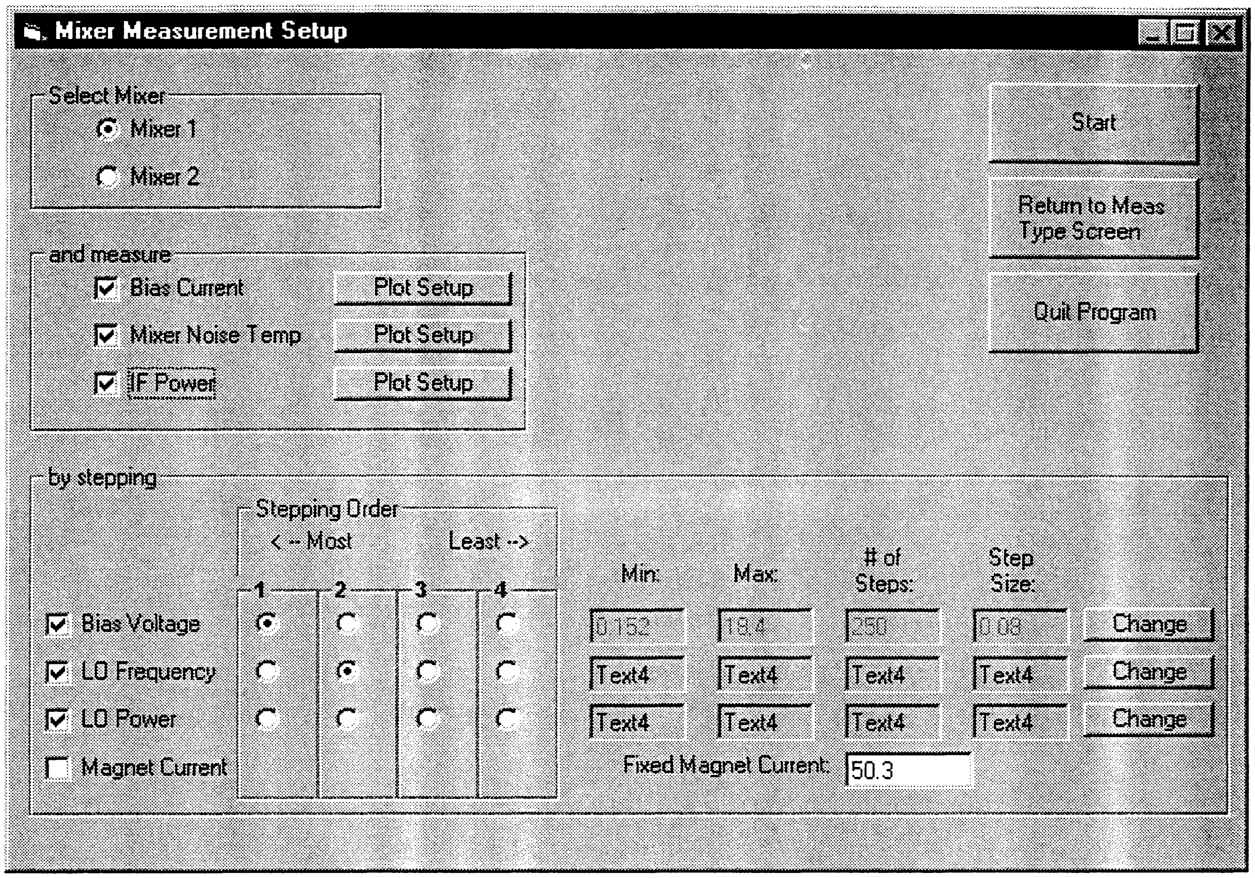
4. Mixer Measurement Dialog Box

Prior to working on the Excel spreadsheet data acquisition routines, a prototype user interface was defined for general mixer measurements (Figure 3) which permits measurement of the following data:

- I-V curves
- mixer noise temperatures, and
- IF powers

by changing a selectable number of parameters.

Check boxes allow selection of the values to be measured and the parameters to be changed. The lower section in the dialog box entitled "by stepping" provides the operator a way to define which parameters to change and how they should vary during data acquisition. When a parameter is unchecked, an edit box appears which allows the operator to enter a constant value. This is exemplified in the "Magnet Current" row.



Select Mixer
[✓ Mixer 1]
Mixer 2

and measure
[✓ Bias Current]
[✓ Mixer Noise Temp]
[✓ IF Power]

by stepping

Stepping Order: < -- Most ... Least -->

	1	2	3	4	Min:	Max:	# of Steps:	Step Size:
✓ Bias Voltage	●	○	○	○	0.152	18.4	250	0.08
✓ LO Frequency	○	●	○	○	Text4	Text4	Text4	Text4
✓ LO Power	○	○	○	○	Text4	Text4	Text4	Text4
Magnet Current								

Fixed Magnet Current: 50.3

Figure 3: Prototype User Interface for Complete Mixer Measurements