SIS Mixer Measurement

Software Design Document

2000-06-06

Version 1.40

Revisions

Table 1: Document Revisions

Revision Number	Date	Who	Details
0.1	1998-09-03	jee	Initial
0.2	1998-09-25	jee	Revised to include new drawing format
0.3	1998-10-20	Jee	Added section on Excel data collection routines
0.31	1998-10-20	Jee	Updated Excel data collection routines to include CdataAcqForm class.
0.4	1998-11-18	Jee	Added hardware addresses section
0.5	1998-12-14	Jee	Further comments in database section
0.6	1999-01-04	Jee	Updated Visual Basic program file locations
0.7	1999-01-08	Jee	Updated Database Table Relationships with NumOfDependentVars
0.8	1999-01-21	Jee	Added section on generic parameter handling
0.9	1999-03-03	Jee	1) Updated hardware address table. 2) Updated Figure 1
0.91	1999-03-25	Jee	Added footnote to Hardware Addresses section about future requirement for PA3.
0.92	1999-04-27	Jee	Added noise inject to coupler and circulator for mixer 1 rack to hardware address table.
0.93	1999-05-05	Jee	Added further description of CSISDevice class.
0.94	1999-05-25	Jee	Updated data dictionary.
0.95	1999-06-22	Jee	Updated programming description of the IV Plotting Dialog Box.
0.96	1999-07-06	Jee	Restructured document into sub-documents.
0.97	1999-07-26	Jee	Added DIO-7 definition.
1.00	1999-10-28	Jee	Added addressing for IF plate: mixer bias supply control.
1.10	1999-11-08	Jee	Added Visio figures showing program flow.
1.20	2000-02-16	Jee	Added flowchart for I-V curves.
1.21	2000-03-07	Jee	Updated “Warm IF” address bit assignments.
1.22	2000-03-10	Jee	Added bWriteBias to CBias class.
1.23	2000-03-26	Jee	Added database management class diagram
1.24	2000-04-04	Jee	Updated warm IF address table to as-built values
1.30	2000-04-25	Jee	Additional restructuring of sections and updates to database section.
1.40	2000-06-06	Jee	Updated database sections.

Contents

List of Figures

List of Tables

1. Introduction

This document provides software requirements, design concepts, and implementation details for the SIS mixer measurement system.

2. Software Requirements

2.1 Measured Parameters

The following shall be measured and recorded in a database:
1. Bias voltage
2. Bias current
3. Magnet current
4. LO frequency
5. LO power
6. Standard deviation of bias current (used to determine when instabilities occur).
7. Ambient temperature
8. 50K stage Dewar physical temperature
9. 20K stage Dewar physical temperature
10. Dewar hot load physical temperature
11. Dewar cold load physical temperature
12. Mixer sideband rejection ratio
13. Noise power when mixer observing hot load
14. Noise power when mixer observing cold load
15. Noise power from IF amplifier switched to hot load
16. Noise power from IF amplifier switched to cold load
17. Noise power from IF amplifier switched to open circuit on input
18. Noise power from IF amplifier switched to short circuit on input
19. Noise power when three different noise levels are injected toward mixer output.
20. Noise power when same three noise levels are injected toward IF amplifier input.

2.2 Calculated Parameters

The following shall be calculated and available for data output
1. Receiver DSB noise temperature
2. Mixer DSB noise temperature
3. IF amplifier noise temperature
4. The mixer DSB noise temperature is calculated by plotting mixer noise temperature vs. receiver noise temperature using three different noise powers injected towards IF amp input. The y-intercept of this plot gives the mixer noise temperature, and the slope of this line gives the mixer loss. The R^2 value of the linear regression should be greater than 99.9.

2.3 Manual Operation Required

Hardware and software must be designed to permit the operator to manually change the parameters of all measurements.

2.4 Optimum Bias Point

The I-V routine shall search the following parameter space to find the volume containing the optimum noise temperature:

1. Bias voltage
2. Bias current
3. Magnet current
4. LO power
5. IF Frequency

The optimum volume in this multi-dimensional space is defined as that region with the lowest possible noise temperature consistent with a stable bias voltage and current, which is determined by the suppression of Josephson effects. A limit on the standard deviation of the bias current shall be used to determine the acceptable range for the optimum.

2.5 Reporting Requirements

2-D Plots shall consist of

1. Bias voltage vs. bias current with LO power as a parameter
2. Bias voltage vs. bias current with magnet current as a parameter
3. Mixer and receiver noise temperature as a function of frequency

3-D Plots shall consist of

1. Bias voltage vs. bias current with a range of LO powers as a parameter
2. Bias voltage vs. bias current with a range of magnet currents as a parameter
3. Receiver noise temperature as a function of frequency with bias voltage as a parameter

3. Software Design

Good design methodology dictates that the following code elements should be separated:

1. User interfaces
2. Database routines
3. Business logic

The software is partitioned into two separate functions, measurement software, which acquires the data and writes it to a database, and data analysis software, which retrieves the data from the database and provides plotting and other analysis tools. Figure 1 shows how the measurement and data analysis routines separately interact with the database. This software is written in Excel using Visual Basic for Applications, which allows significant code sharing with the measurement routines written in Visual Basic. For example, the database classes will be essentially the same for both programs.

3.1 Measurement Philosophy

The measurement system shall first determine the optimum bias point (for a particular LO frequency) and then measure the I-V curve corresponding to that bias point.

3.2 Architectural Schema

The software is designed using the general architecture outlined below and shown in Figure 1:

Low-level (*e.g.* instrument control) routines are coded in stand-alone Visual Basic.
Data are stored in an Access database file on shared NT server.
The data are retrieved from the database and plotted and analyzed using Excel *via* add-ins written in Visual Basic for Applications.

3.3 Stand-Alone Visual Basic Program Architecture

Stand-alone Visual Basic (VB) allows programs to be written as self-contained executables, which is the form used for the I-V data acquisition program. VB also provides for the creation of "Active-X" dynamic link libraries, and that code form is used for the instrument drivers for the noise temperature measurement.

3.4 Visual Basic for Applications Program Architecture

Visual Basic for Applications (VBA), included with Excel, allows for the construction of functions and subroutines that can be executed directly from Excel. VBA routines can be pre-compiled and stored as an "add-in", which is essentially a dynamic link library. One advantage of add-ins is that they can be loaded each time Excel is started, and the functions available from the VBA code are obtained from menus also written in VBA. A disadvantage is that add-ins must reside in the Excel library directory, which is on the users local drive, which complicates software updates.

VBA also provides the means to call routines stored in dynamic link libraries, and that capability is used to call instrument drivers, which are written in VB as an Active-X DLL.

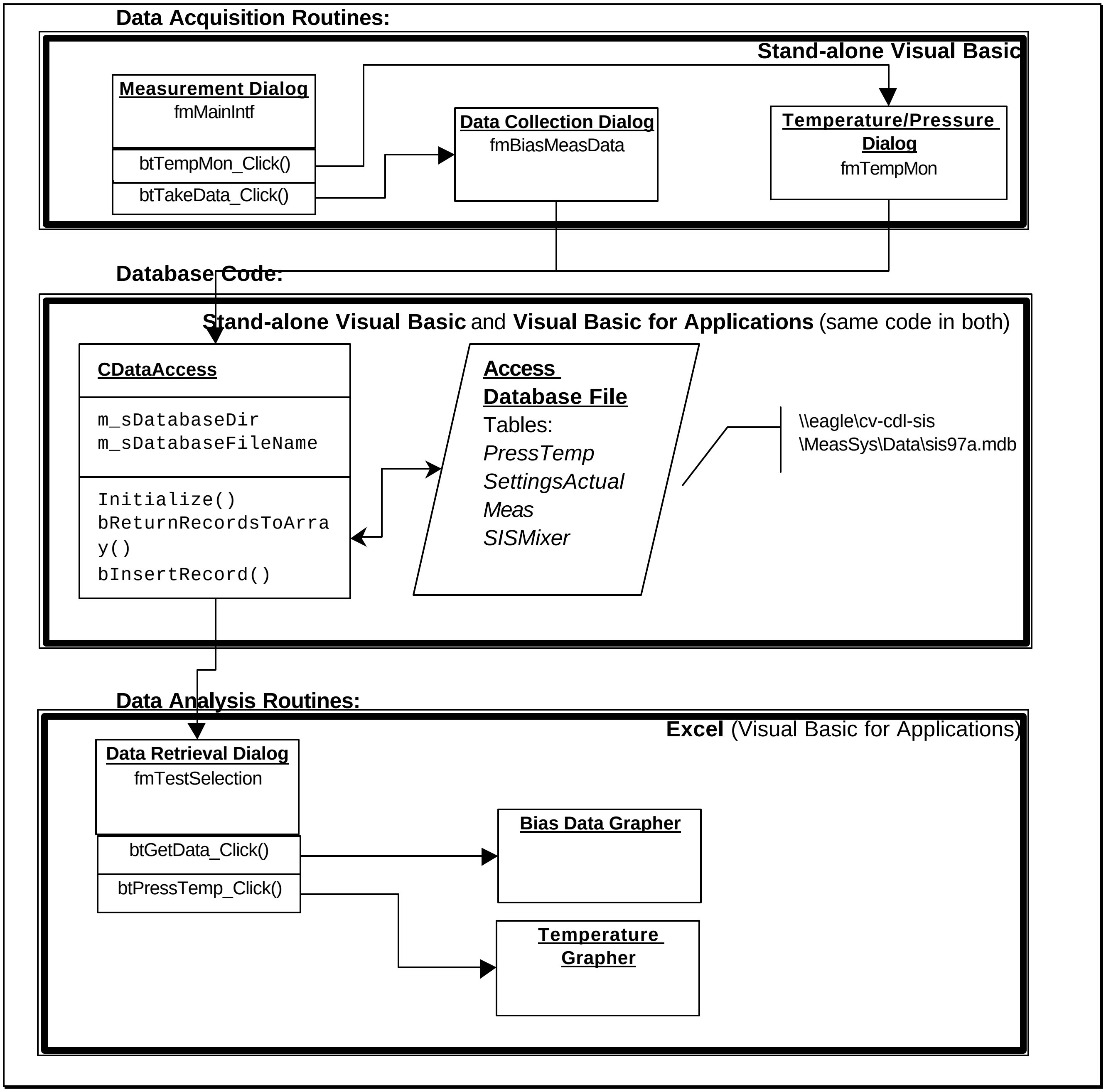


Figure 1: Overall Software Architecture

4. Measurement Subsystems

There are two major software components: The I-V plotter and the noise temperature measurement system as detailed in the following sections.

5. Overall Block Diagram

Figure 2 contains the overall block diagram for the noise power routines (incomplete).

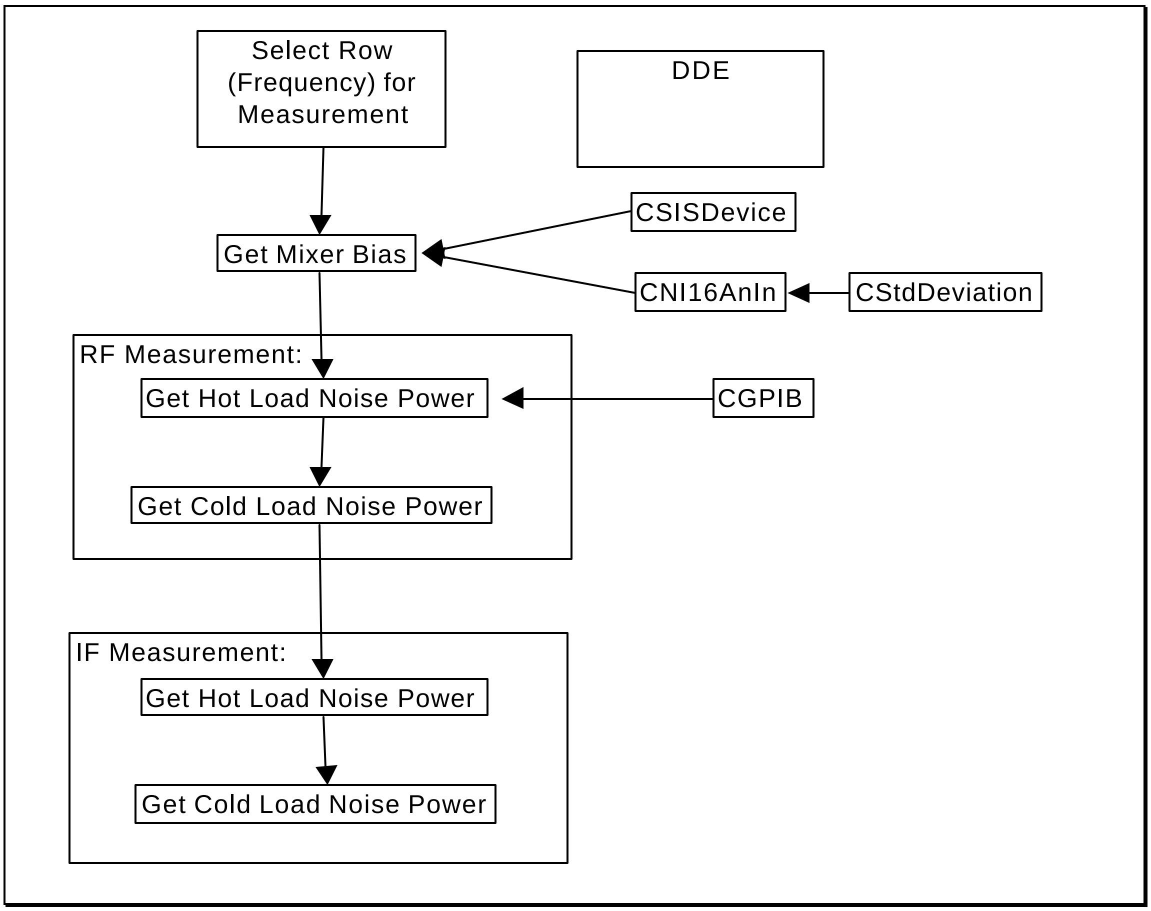


Figure 2: Overall Block Diagram for Noise Temperature Measurement Routine

6. Data Acquisition

7. Measurement Routines

Two steps are used for the measurement routines:

1. First, mixer and measurement description information is entered into a dialog box.
2. Next, a data acquisition dialog box actually acquires and provides a simple, run-time plot of the data.

The measurement description dialog box allows the user to review previous measurement descriptions in addition to entering information about new measurements. Data input for new measurements is simplified by using drop-down boxes for each data field, which allow the user to select an input from a list comprised of all previous entries for that field.

The MSHFlexGrid Control fgrdSweepParms displays the sweep parameters. This grid is control from the class CsweepGrid. The ADO data control was not used to control this grid because that control shows all the data. In addition, the database is not updated through the ADO control when changes are made to the flex grid.
Figure 3 shows the measurement description dialog box, and its inputs are defined in Table 2.

7.1.1.1.1 Measurement Parameters Control

This control is responsible for displaying and updating the measurement parameters, such as bias voltage range, LO power range, etc.

When the `Change` cell is clicked, another dialog box is presented that allows the user to select either to sweep the parameter or hold it constant during the measurement. If the parameter will be swept, than the change dialog box allows the user to change the range and step size for the sweep.

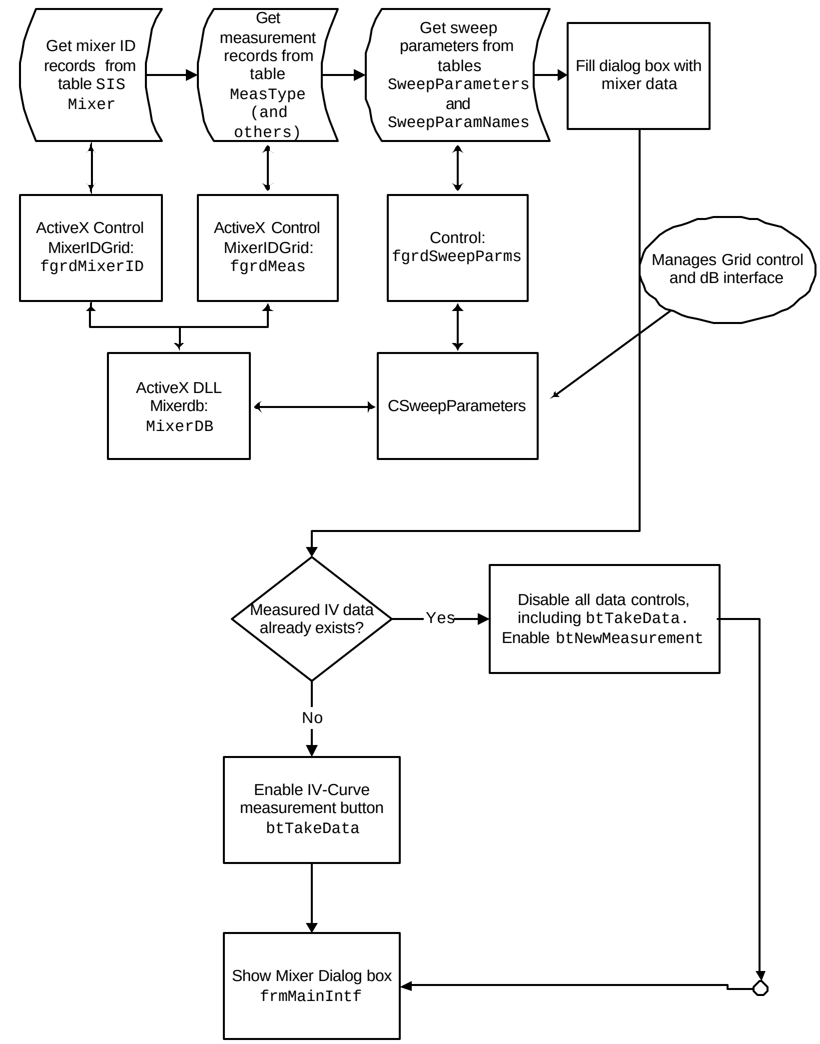
The parameter that is swept the most frequently is located at the top row of the control.

<u>Future Enhancements</u>:

1) Those parameters that are to be swept are moved to the top of the control.
2) The user can change the values in the individual cells without clicking on the `Change` cell.

7.1.1.1.2 Temperature Strip Chart

This routine records to the database temperatures, pressures, and flow rates and also plots this data. Figure 10 is the flowchart for the normalization routine, which maintains the data between preset limits.



SIS Mixer Software	Edit Date: 2000-6-6 14:19	
IV Measurement Program Flow		
Page 1 of 1	Rev: 0	Creator: jee
File: Design.doc		National Radio Astronomy Observatory

Table 2: Measurement Description Dialog Box Entities	
Entity	**Description**
Buttons:	
New Measurement	Updates the measurement date to the current time and clears the other input boxes
Start Data Acquisition	Updates the measurement record in the database, if required, with the information from this dialog box, and displays the measurement dialog box.
Previous, Next, Last, and Delete records	These buttons provide a means of navigating through previous measurements in the database
Get Database Engine Version	Returns version information about Microsoft's database engine, which is the program that actually accesses the database.
Quit	Returns all test equipment from remote to local control, and ends the program.
Text boxes:	
Measurement Date	The current date and time is automatically entered when the "New Measurement" button is pressed
Drop-down boxes:	
Device ID	Allows selection of a previously used device ID, after which the dialog box will display the information entered for this device ID. - or - Allows a new device ID to be entered
Measurement type	Allows selection of a previous measurement type. - or - Allows a new measurement type to be entered
Measurement by	Allows selection of the name or initials of the person who previously performed a measurement. - or - Allows a new measurement type to be entered
Notes	General notes of unlimited length can be entered into this box.

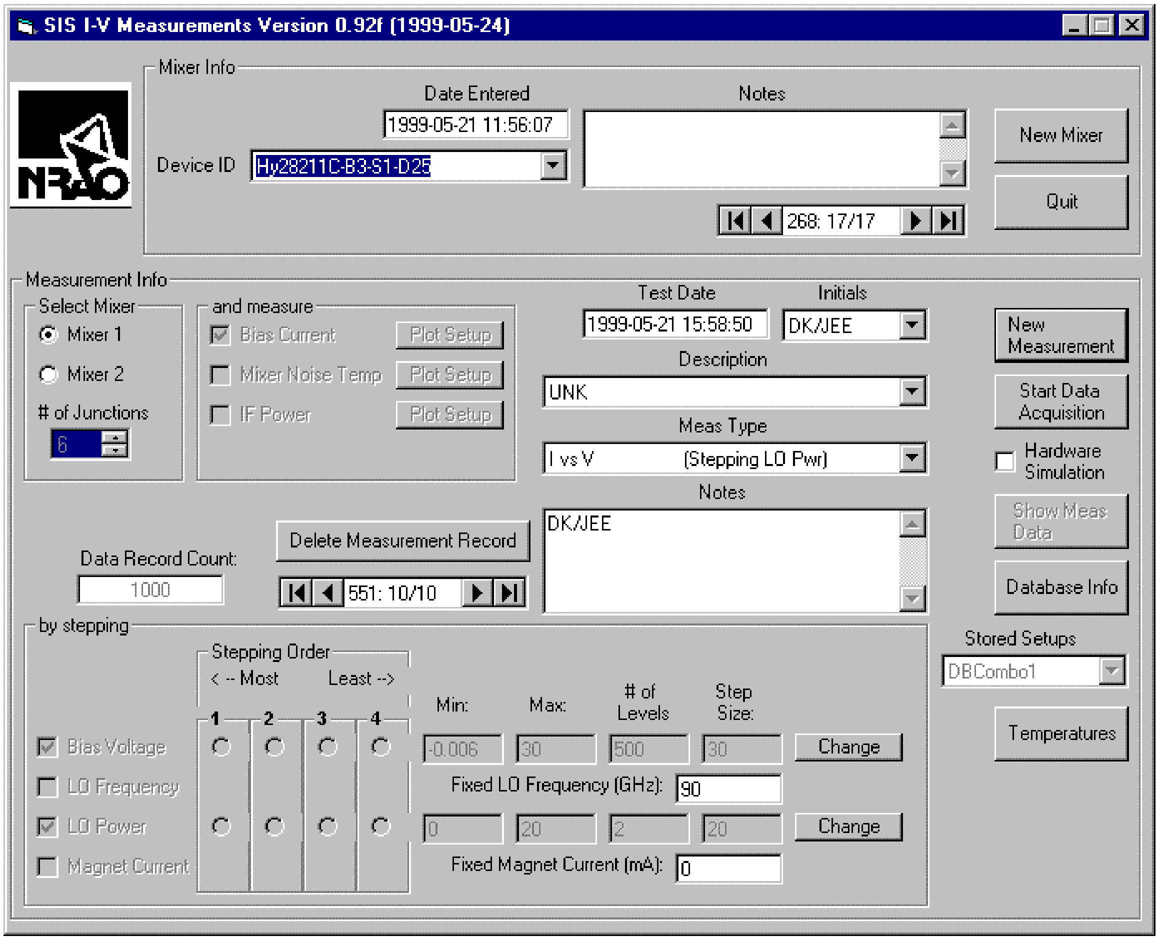


Figure 3: User Interface for I-V measurement input

After entries in the measurement description dialog box are completed, pressing the “Start Data Acquisition” button brings up the data acquisition dialog box. Figure 4 shows the data acquisition dialog box with inputs defined in **Table 3**.

Table 3: Data Acquisition Dialog Box Entities	
Entity	**Description**
Buttons:	
Init GPIB	Initializes the instruments
Take Data	Commences data acquisition
Reset Instrument	Returns the equipment to local mode
Print Graph	Prints the real-time graph
Print Dialog Box	Prints the entire dialog box
Quit	Returns control to the Measurement Description Dialog Box, Figure 3.
Text boxes:	
Number of Readings	Sets the number of power meter measurements to be taken during one acquisition phase
Check boxes:	
Store Data in Database	When checked, data are stored in the measurement database. If not checked, data are plotted but not stored, which allows a preliminary results check prior to measuring and storing final data.
Real Time Plot	When checked, results are plotted point-by-point. That option can significantly slow the rate of data acquisition. When unchecked, results are plotted after all data has been acquired.

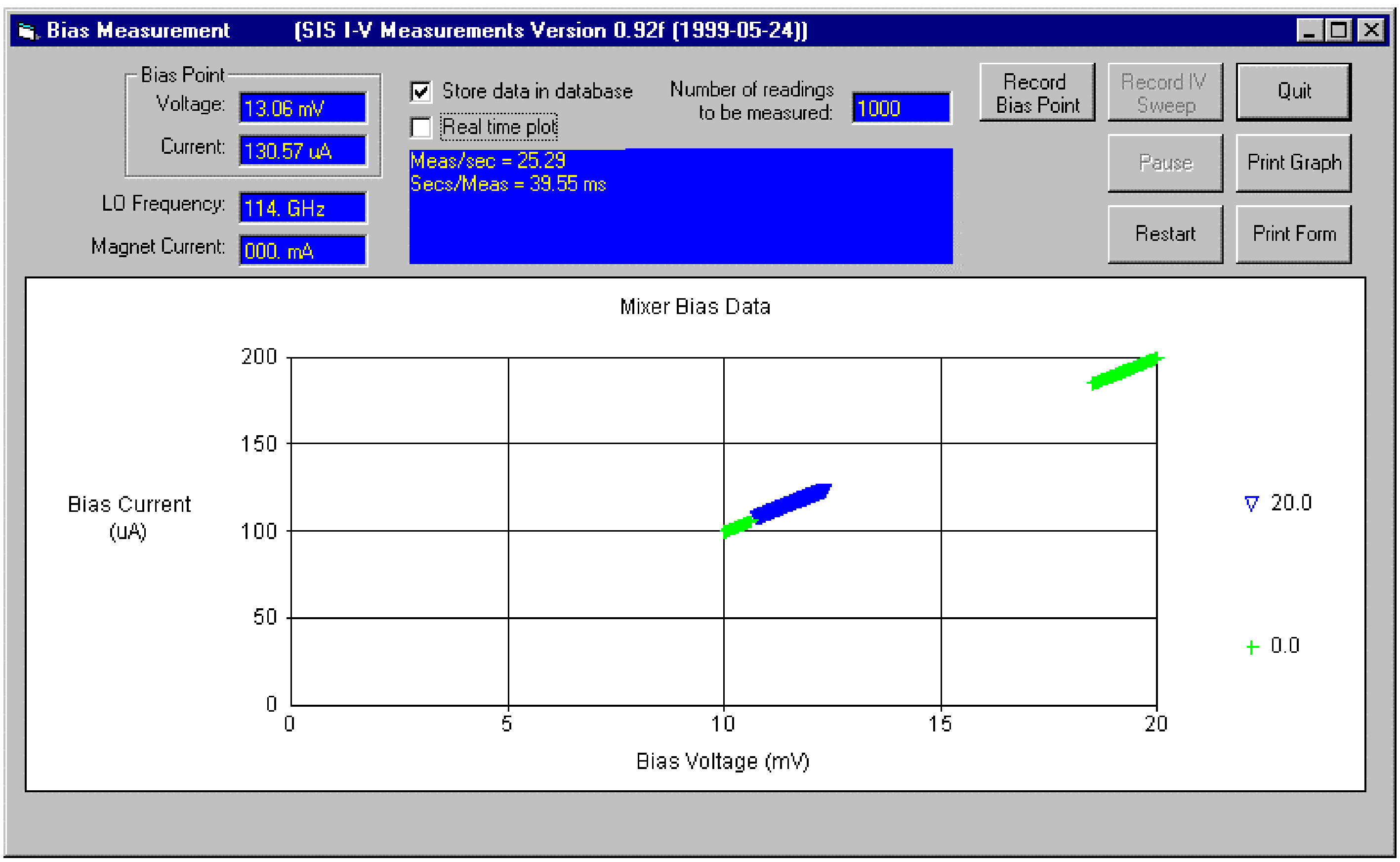


Figure 4: Dialog for Data Acquisition Routine

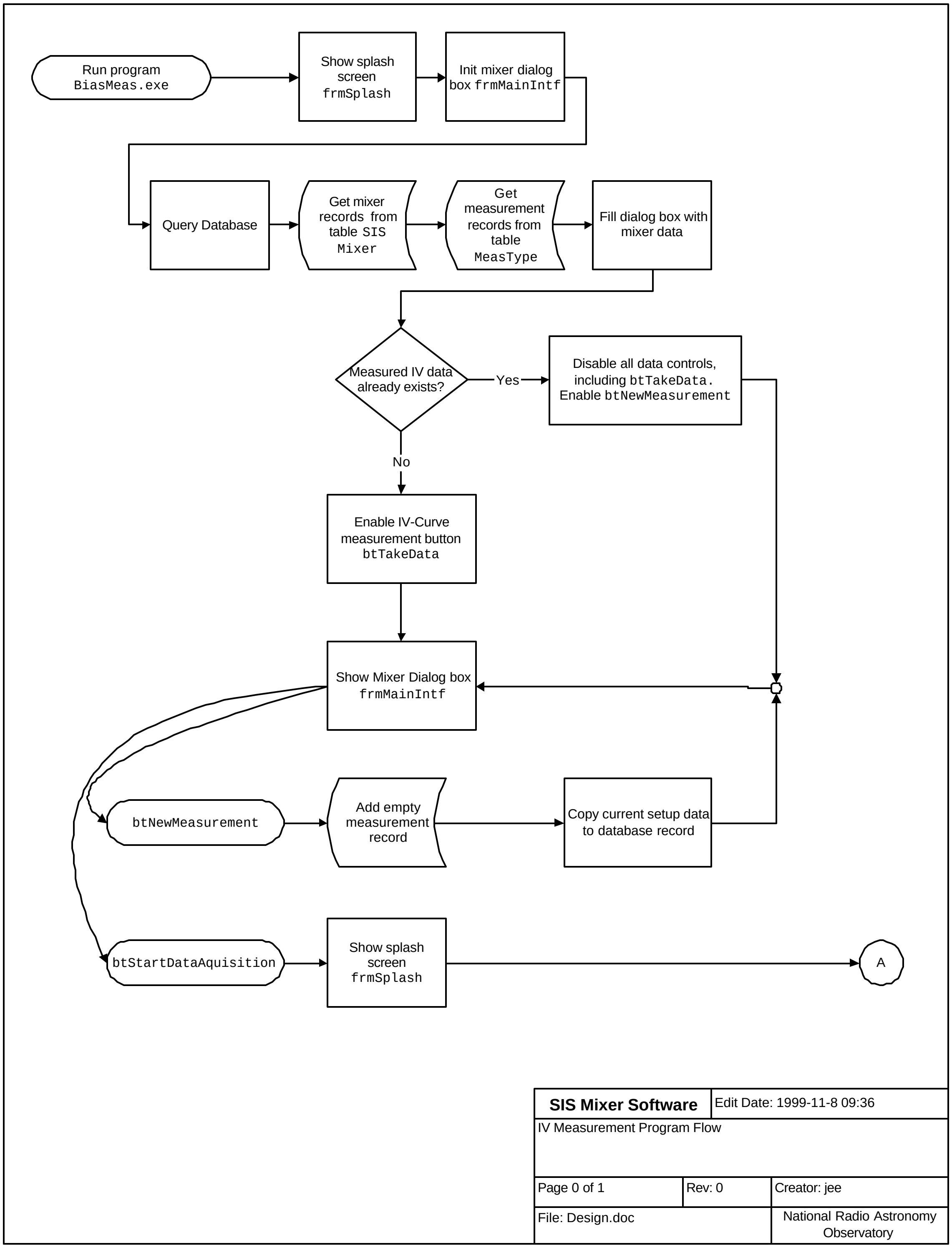


Figure 5: Measurement Program Flow

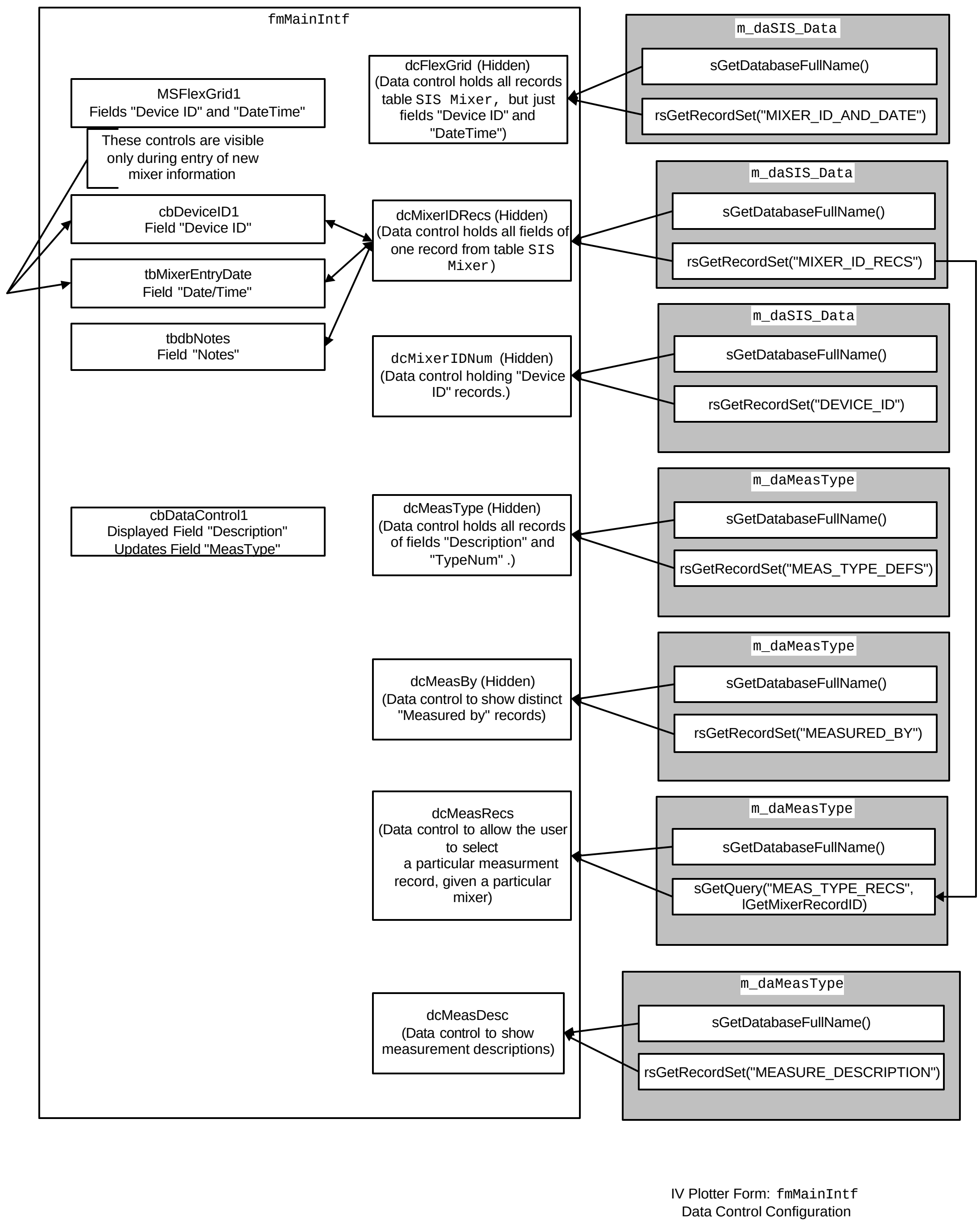


Figure 6: Main IV Measurement Program - Dialog Box Widget Description

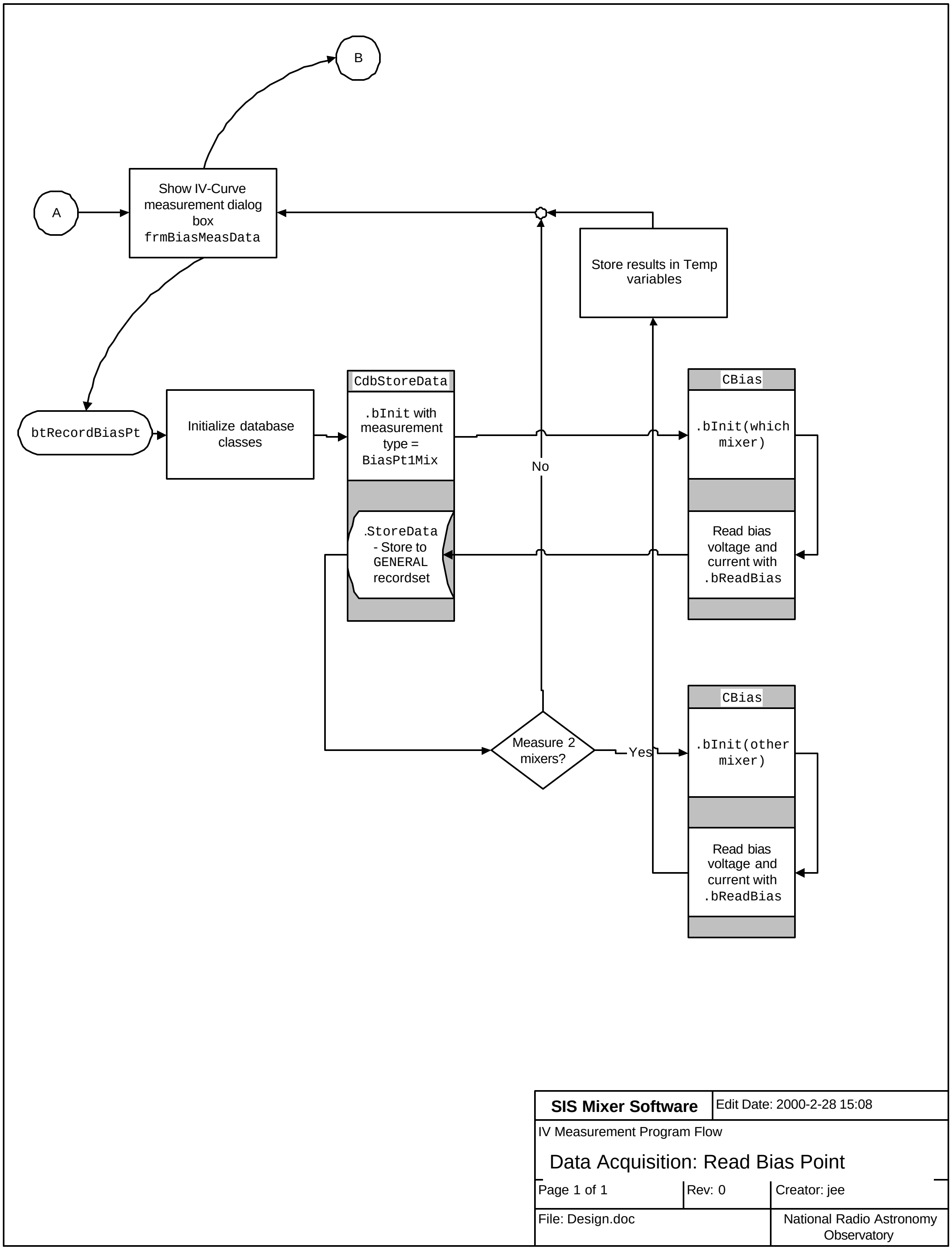


Figure 7: Program Flow for "Record Bias Point"

Program flow for the main data acquisition loop is shown in Figure 8. The flow diagram is entered on the upper left of the page with the "Take Data" button that has the name `btTakeData`. The shaded boxes represent important classes in the program that are described in Table 4. Although presenting the classes in this manner complicates the paths in the diagram, it helps to highlight the functions of each class.

Table 4 : Classes used during Main Data Acquisition Loop

Class Name	Description
`CBias`	This represents the mixer bias supply, and is responsible for mapping a specific mixer to an analog channel.
`CDataAccess`	This is the main generic database handler. Specific recordsets are retrieved by initializing the class with a set of constant parameters.
`CdbStoreData`	This "child" of `CdataAccss` is a specific database handler that maps field names to generic names for use by the program. The field mapping is a function of the type of measurement, which is passed to the class as a parameter of `bInit`. Note that two instances of this class are instantiated into objects: one is used to save the bias point data, the other to store the swept measurement data.
`CNI16AnOut`	This represents the analog card used for outputting the commanded mixer bias voltage. This class really should be encapsulated into `CBias`.
`CNoiseAnal`	Encapsulates the code required to calculate receiver and mixer noise temperatures.
`CSISDevice`	This class holds the characteristics of the SIS Mixer device, such as the commanded bias voltage, which is a function of the measurement loop index and the number junctions in the device.
`CSpecAnal`	Sets the IF frequency of the measurement by controlling the frequency of the spectrum analyzer with it's IF output connected to either the square law detector or the power meter.
`CTrgSqrLaw`	Responsible for controlling the chopper wheel and returning noise powers (as measured by the square law detector) when the receiver sees the hot and cold loads.

The main measurement loop is highlighted with bold lines. Some important functions are omitted from Figure 8 for clarity, such confirming that measurement records don't already exist for this measurement.

A typical program flow begins with entering the flowchart from circle B when the user presses the button named `btTakeData` on the form. Since no mixers have been measured, the methods `OpenRecordset` and `bRecordCopy` are run. `CdataAccess.bRecordCopy` copies the current measurement records to a new record, to simplify data entry, since users often change just a single entry on the measurement form. Next the chart is reset, then `CdbStoreData.bInit` maps specific field names in the database tables to generic field names used by the program. Any previously measured bias point data is then stored by `CdbStoreData.StoreData`, and the bias point is plotted by the graphing class. The measurement time is updated in the current record by the `CdataAccess` class. The required number of measurement steps is obtained from the routine `lGetTotalMeasSteps` in class `SISDevice`, which also returns the number of bias steps required per scan (A scan is a bias sweep with the parameter value constant, such as when the pump power remains at 1 uW). If the data are to be stored in a database, then `CdbStoreData.bInit` is called using another object to allow the swept bias data to be stored. Control is then passed to `CBias.bInit`, which maps channels of the analog card to a particular bias supply.

Now the main measurement loop is entered by incrementing the measurement counter, which is immediately checked to see if its limit has been reached. A check is then made to confirm that the pause button has not been pressed, then the bias voltage required by the bias supply is found in the routine `SISDevice.nVCmdToBiasBox`. A check is next made to determine if the parameter limit has been reached, which is true when the measurement count is a multiple of the number of steps per scan. Next, the desired bias voltage is sent to the bias supply (`CBias.bWriteBias`) and the resulting bias current is read by `CBias.bReadBias`). The class `CMeas` is responsible for reading the dependent parameters, such as noise temperature and the results are stored using the routine `CdbStoreData.StoreData`. Finally, the graph array is updated with the newly measured data. To maximize performance, the graph is not updated until a parameter limit has been reached.

The important module `MeasureResults`, highlighted in bold in Figure 8 near the title block, is responsible for selecting the test equipment required to perform the measurement and is responsible for filling the graph array with data. `MeasureResults` program flow is charted in Figure 9. The class `CTrgSqrLaw` is responsible for controlling the chopper and returning the hot load and cold noise powers (this module currently returns only the Y-factor, so additional coding is required to return the individual average noise powers). `CSpecAnal`, the spectrum analyzer instrument class, controls the measurement frequency. A new class, `CNoiseAnal`, calculates both receiver and mixer noise temperatures given the measured noise powers.

The mixer temperature calculation requires replacing the mixer's IF output with a hot and cold load and measuring the resulting noise powers. In the case of the integrated mixer/amplifier design, this will measure the noise temperature of the warm IF plate, referred to a reference point inside the Dewar. For mixers without integrated amplifiers, this measures the noise performance of the IF amplifier located in the Dewar.

It is unclear how frequently the IF amplifier noise performance should be measured, but initially it will be measured once at the beginning of a measurement scan, and whenever the IF frequency is changed. Measuring this too frequently can significantly slow the results, because hundreds of milliseconds are required to reliably switch the Radiall coax switch to each switch position. Since these measurements only determine the optimum operating point and not absolute noise temperatures, this approach seems like a reasonable time *vs*. accuracy tradeoff.

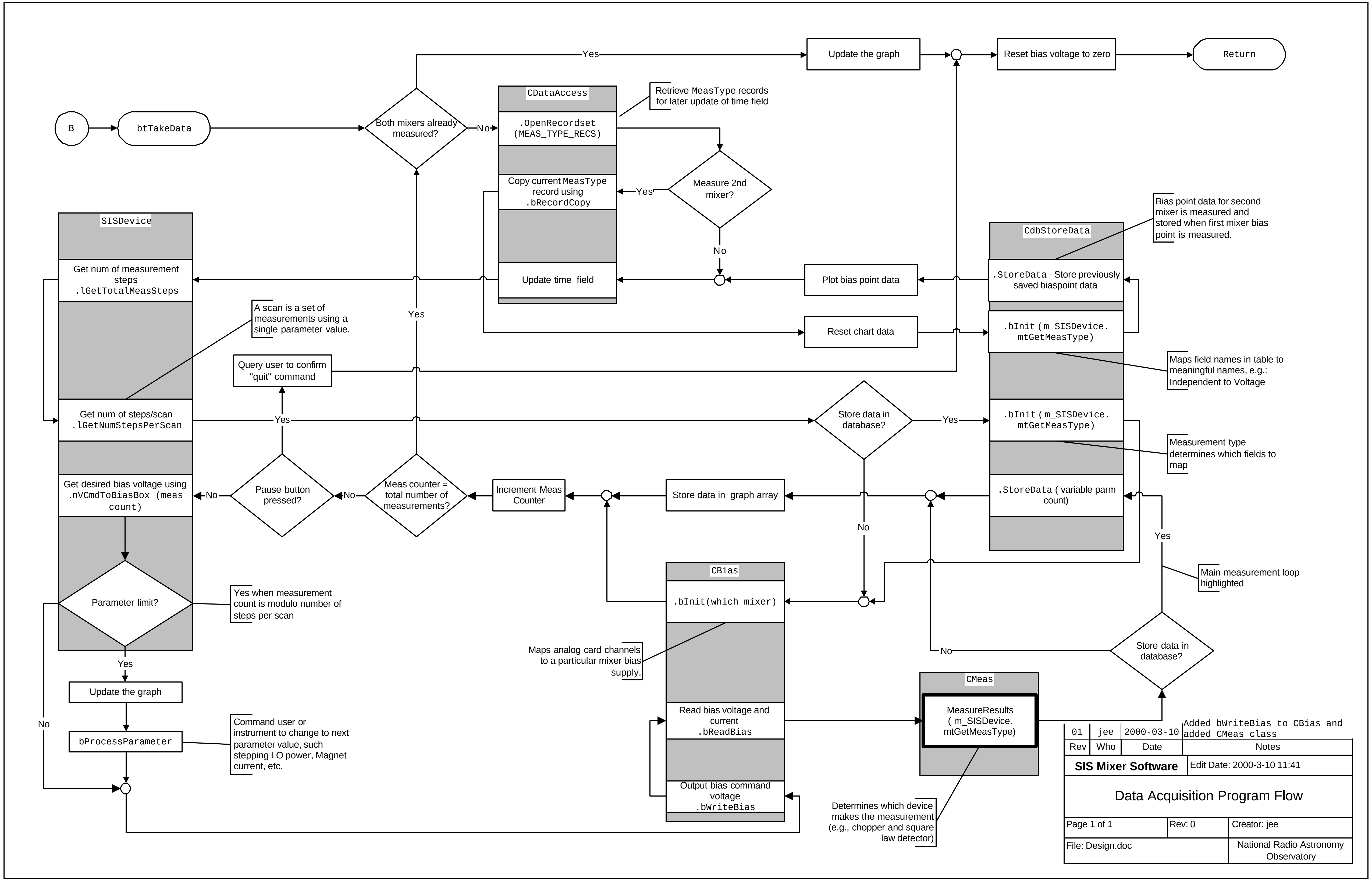
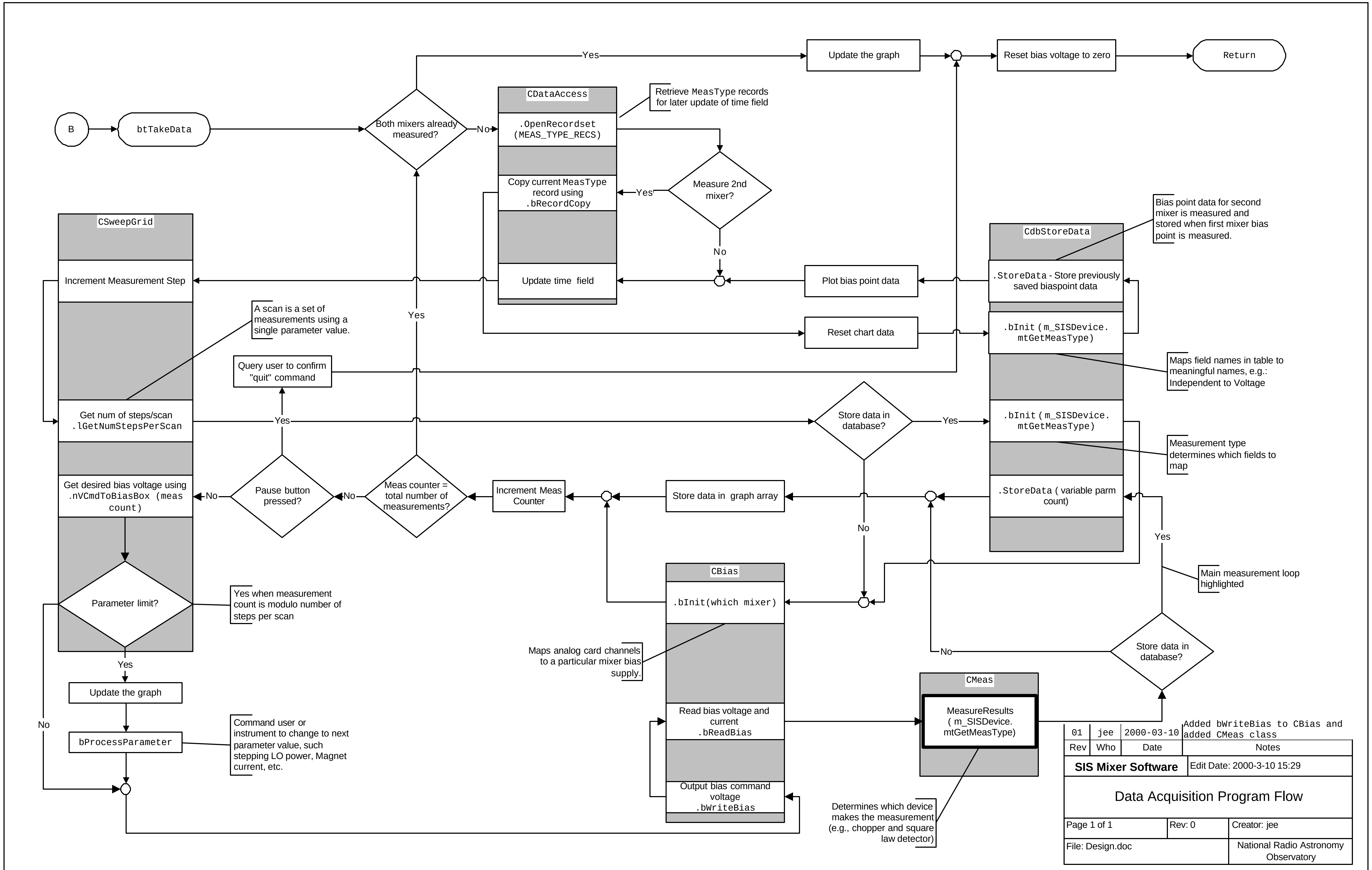
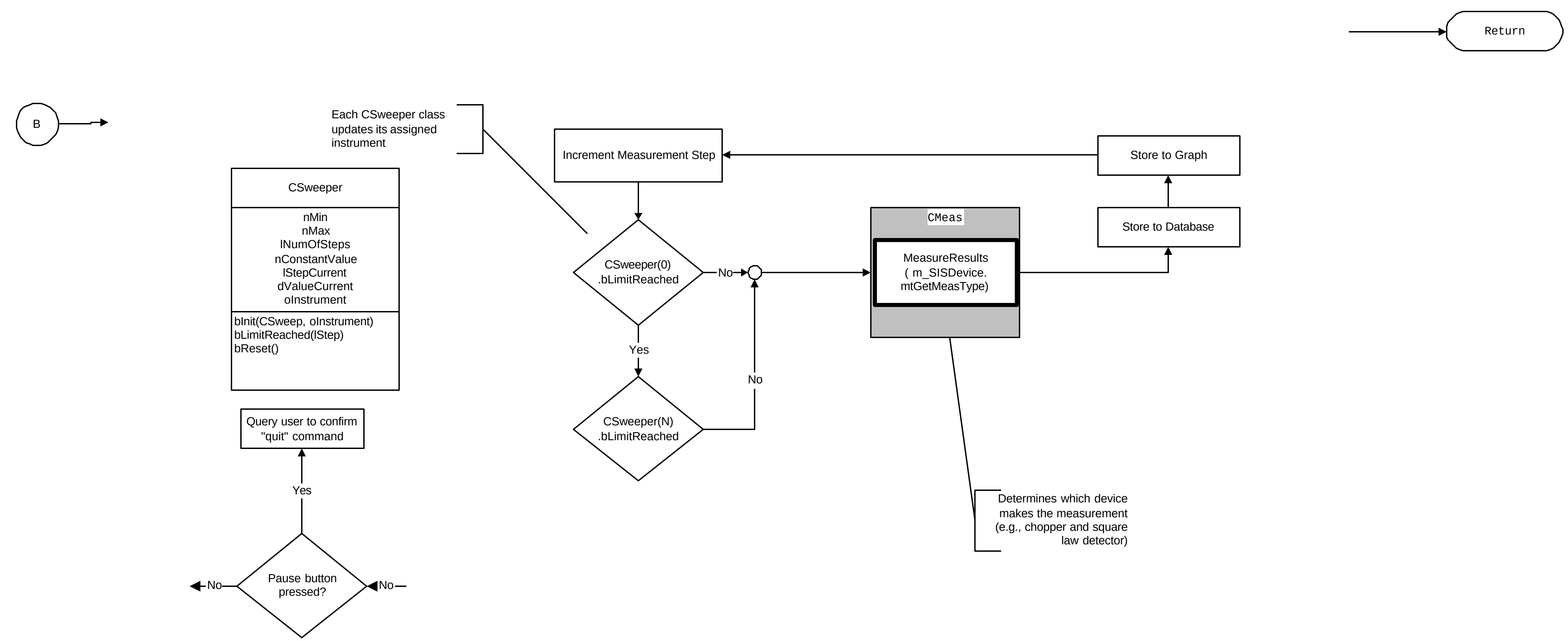


Figure 8: Program Flow for Main Measurement Loop



Rev	Who	Date	Notes
01	jee	2000-03-10	Added bWriteBias to CBias and added CMeas class

SIS Mixer Software	Edit Date: 2000-3-10 15:29
Data Acquisition Program Flow	
Page 1 of 1	Rev: 0
Creator: jee	
File: Design.doc	National Radio Astronomy Observatory



Rev	Who	Date	Notes
SIS Mixer Software		Edit Date: 2000-6-14 17:24	
CSweepGrid Structure			
Page 0 of 1	Rev: 0	Creator: jee	
File: Design.doc		National Radio Astronomy Observatory	

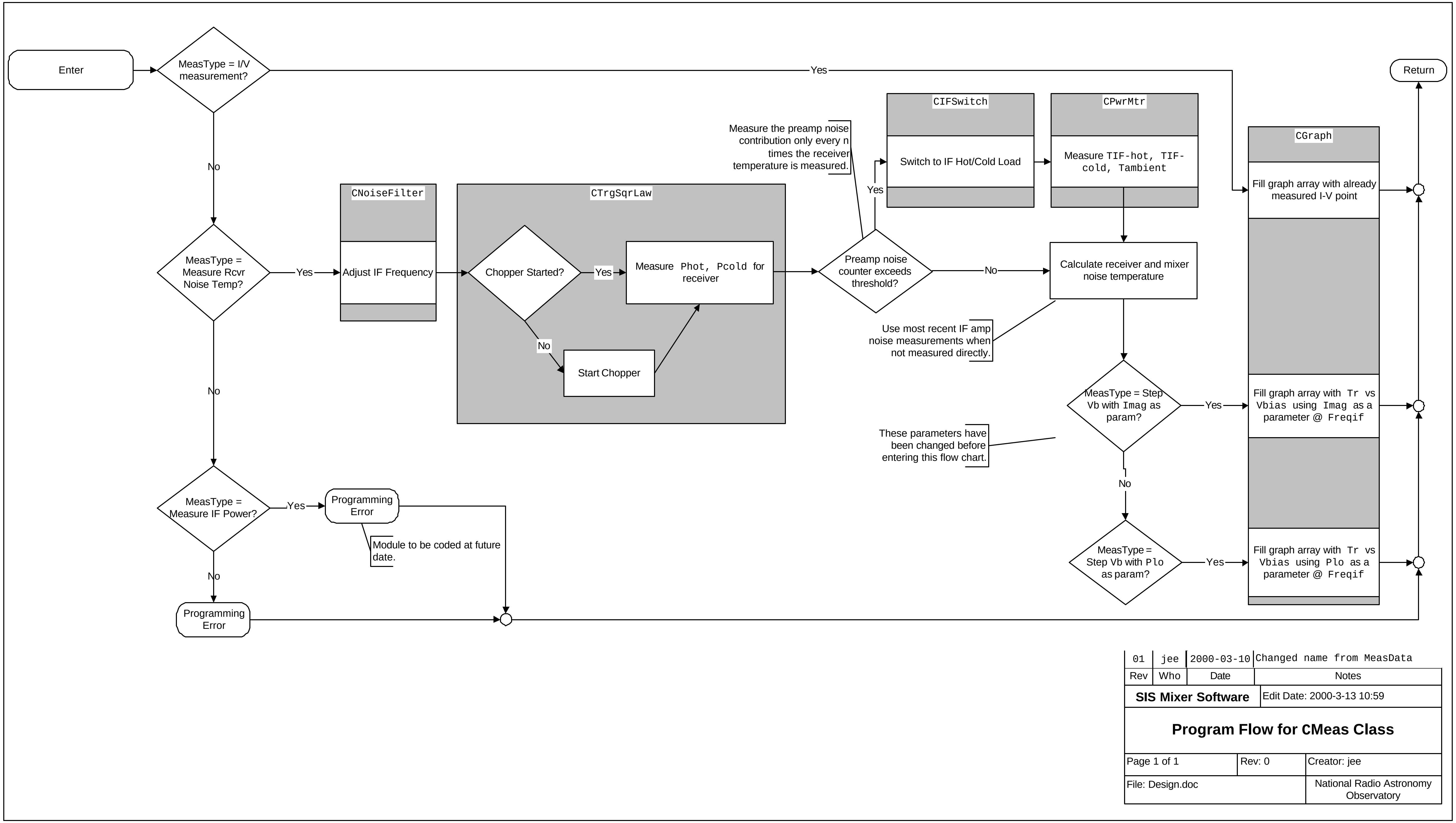


Figure 9: Program Flow for `Cmeas` Class

Figure 10 shows the program flow for scaling the temperature strip chart data. The temperature strip chart presents challenges for graphing because the temperature, pressure, and flow rate data can vary over a large range. A series of arrays are created for each measurement type. Then, the most-recently measured data point is compared to the current limits of the graph. If the data point is outside the limits of the graph, a scaling factor is calculated for this data point such that it will lie within the graph limits. The scaling factor is compared to the previous scaling factor: If they differ, the data in the array corresponding to the current measurement is rescaled with the most recent scaling factor.

The straight-forward plotting function is performed outside the flow chart.

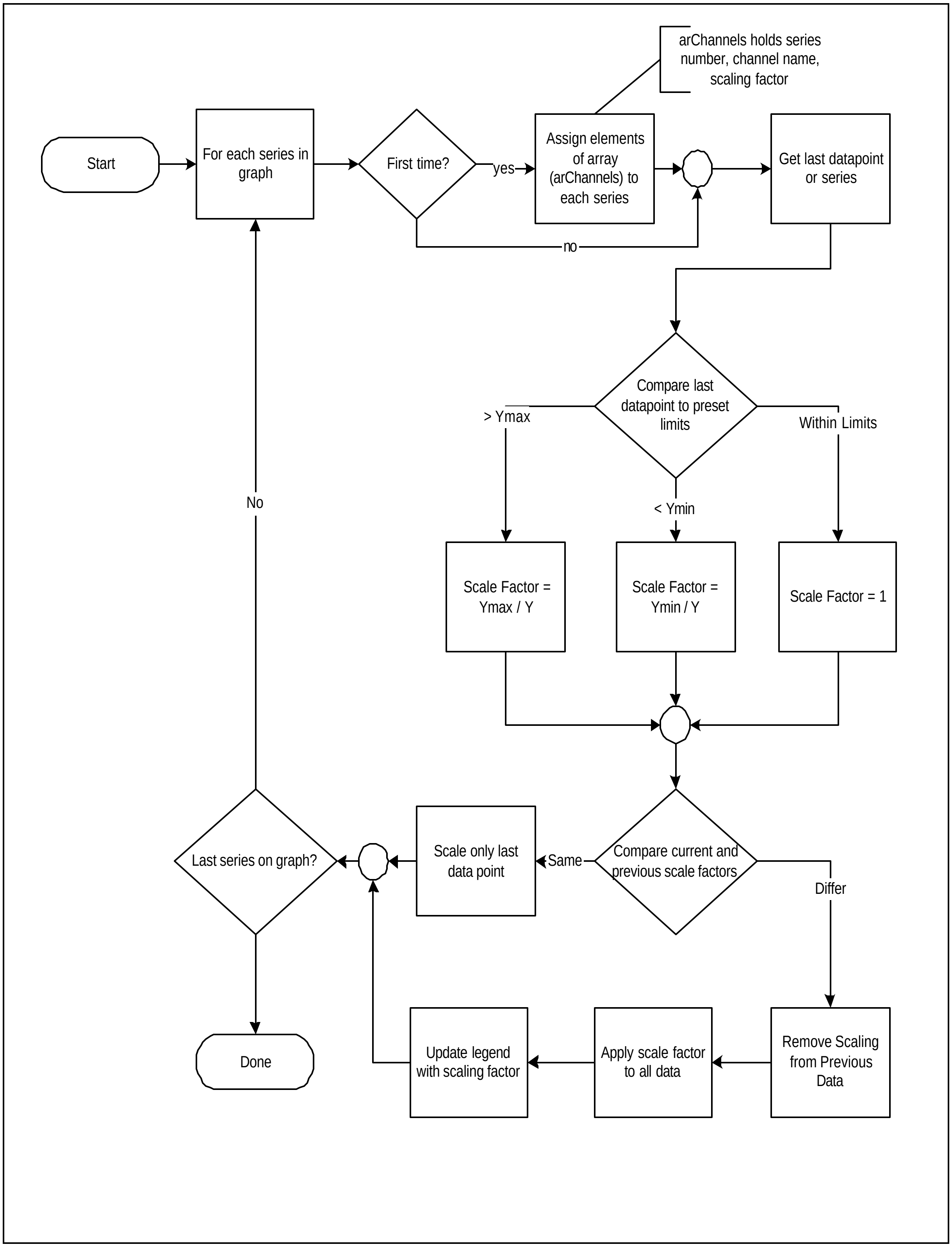


Figure 10: Data Scaling Routine for Temperature Graphing

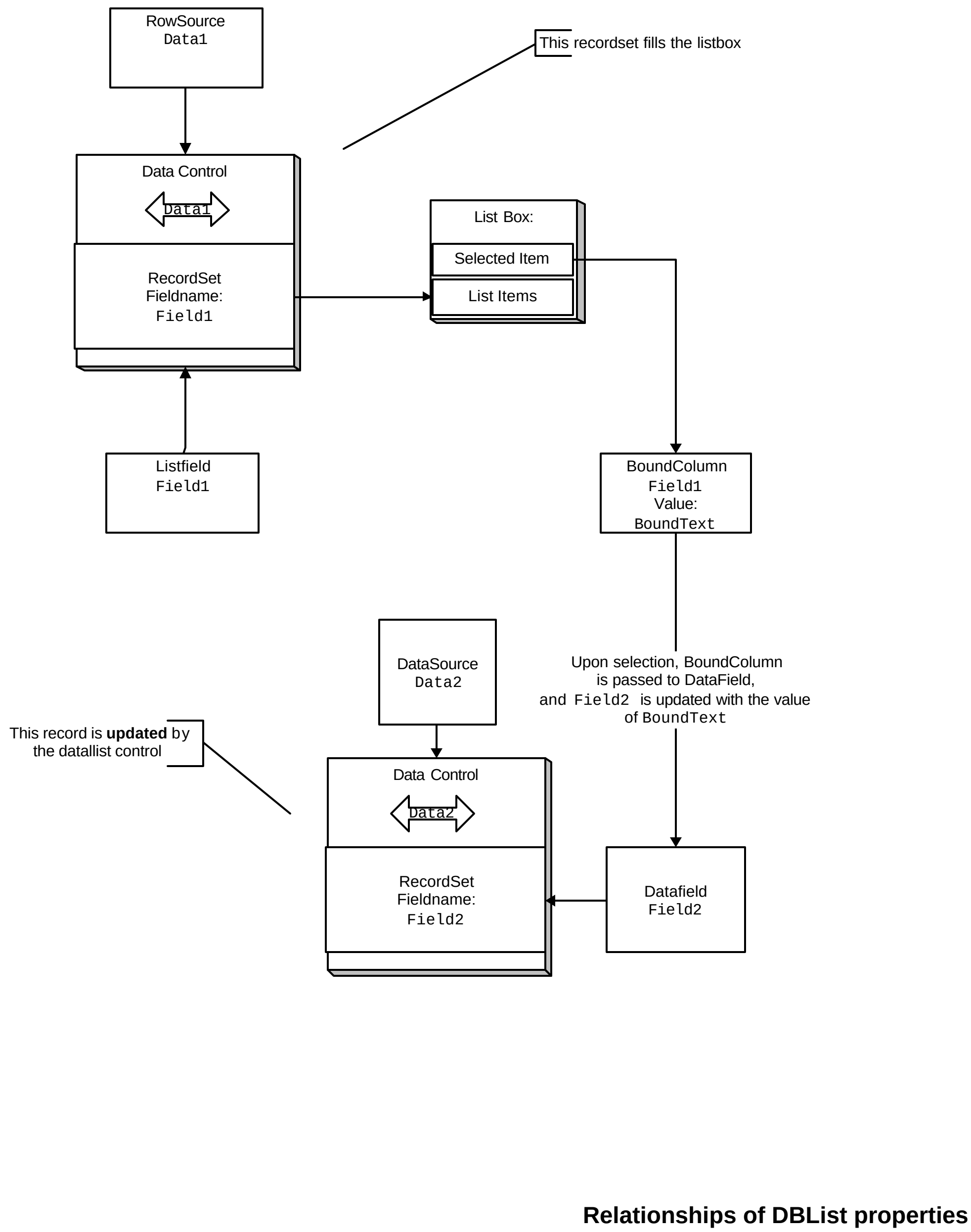


Figure 11: Relationships of dbList Properties

8. I-V Plotter

The data acquisition part of the I-V plotter sends a voltage command to the mixer bias control, reads the resulting voltage and current, and stores the results in a database. A graphing and data analysis routine retrieves the data from the database, stores it into a spreadsheet file, and graphs it. There are separate user interfaces for the measurement routines and data analysis routines.

9. Noise Measurement

10. 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 12 and Figure 14 as described in the following sections.

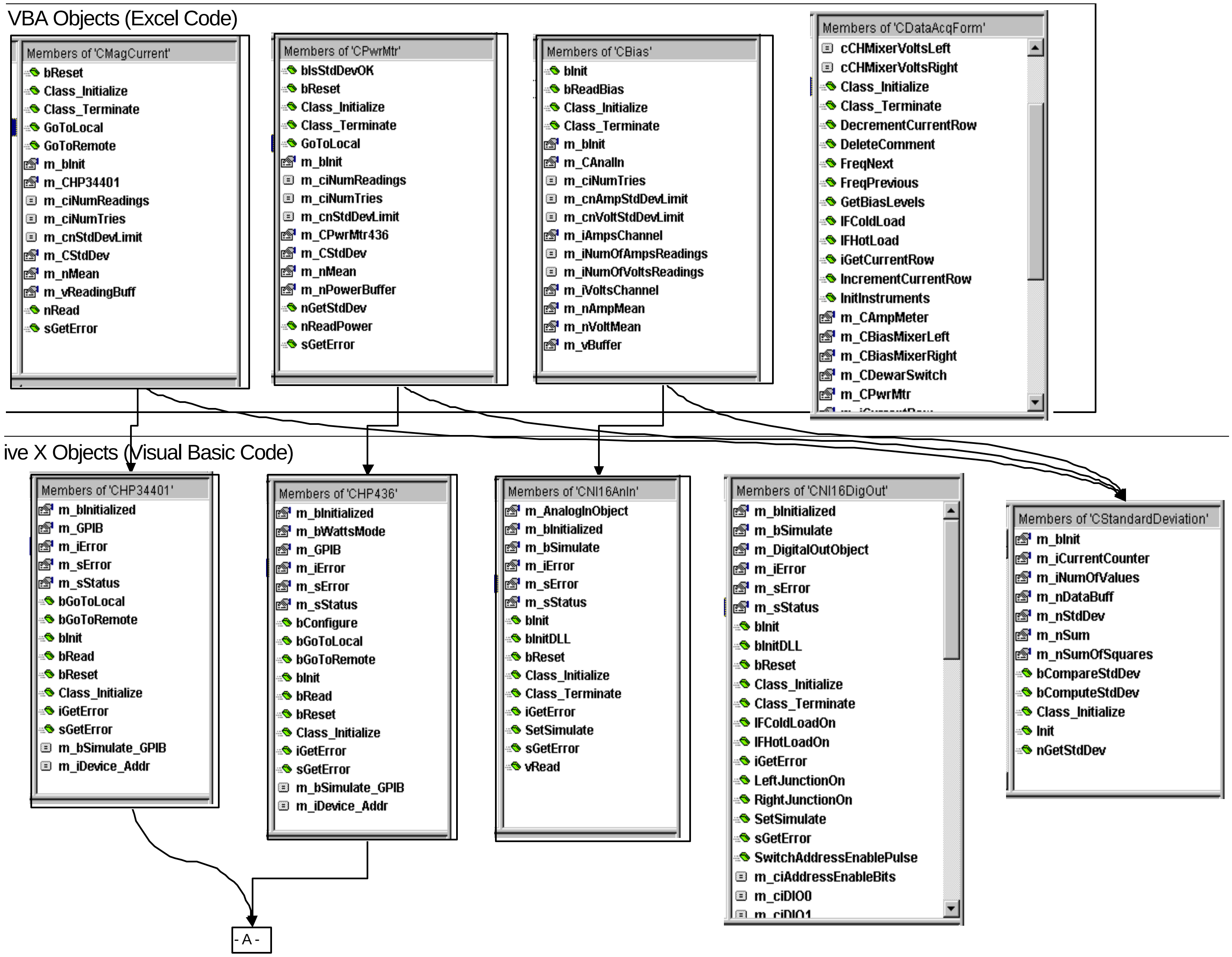


Figure 12: Class Diagram (1 of 2)

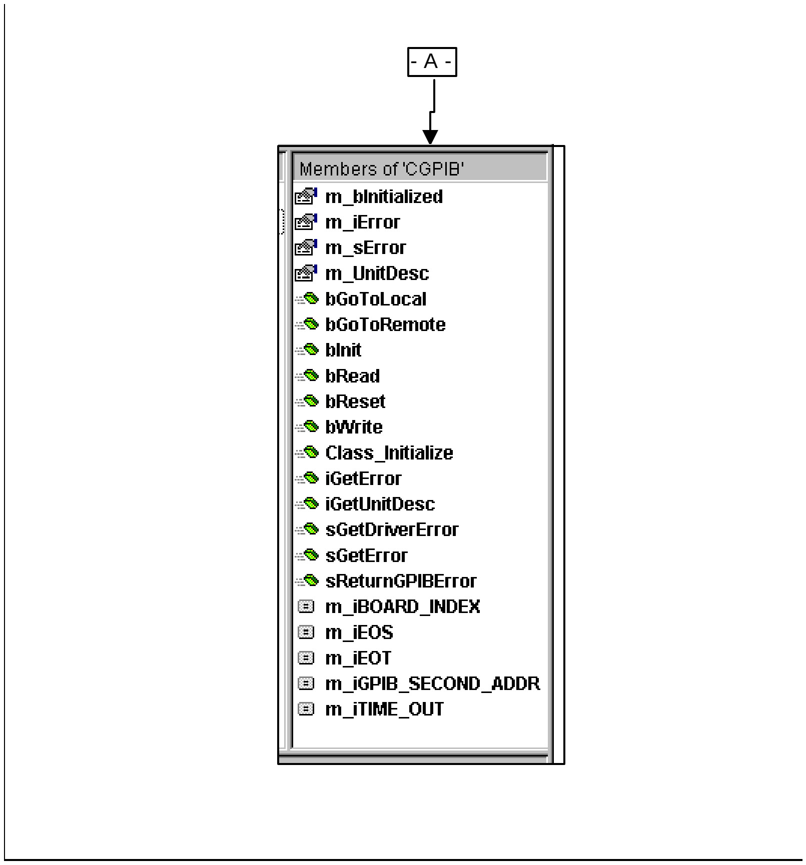


Figure 14: Class Diagram (2 of 2)

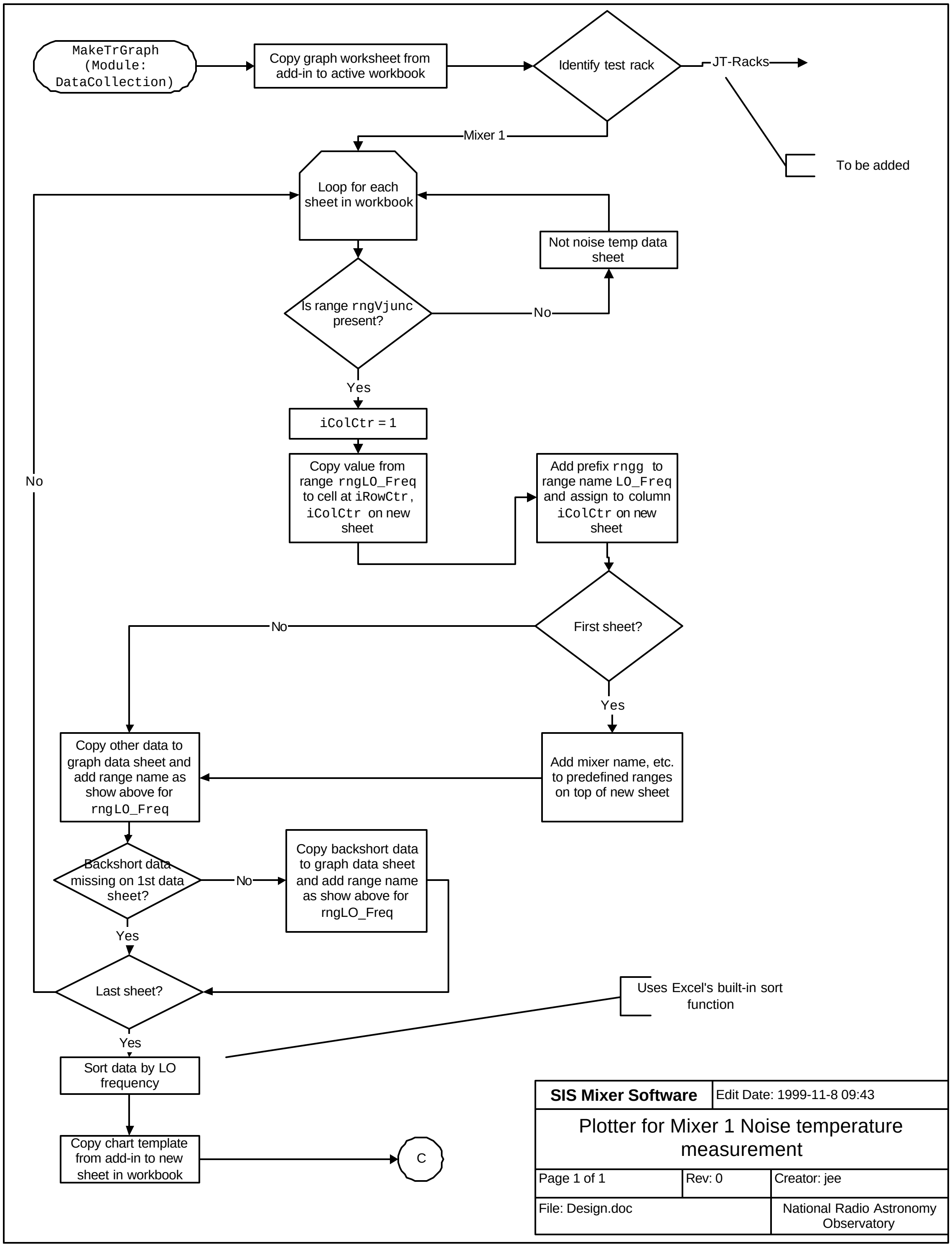


Figure 16: Flow Chart for Mixer 1 Noise Measurement Plotter

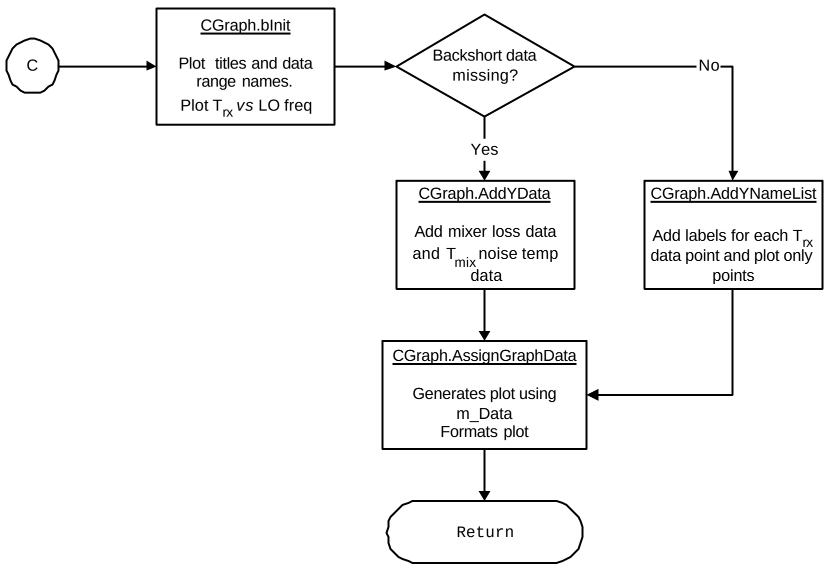


Figure 17: Flow Chart for Mixer 1 Noise Measurement Plotter (Continued)

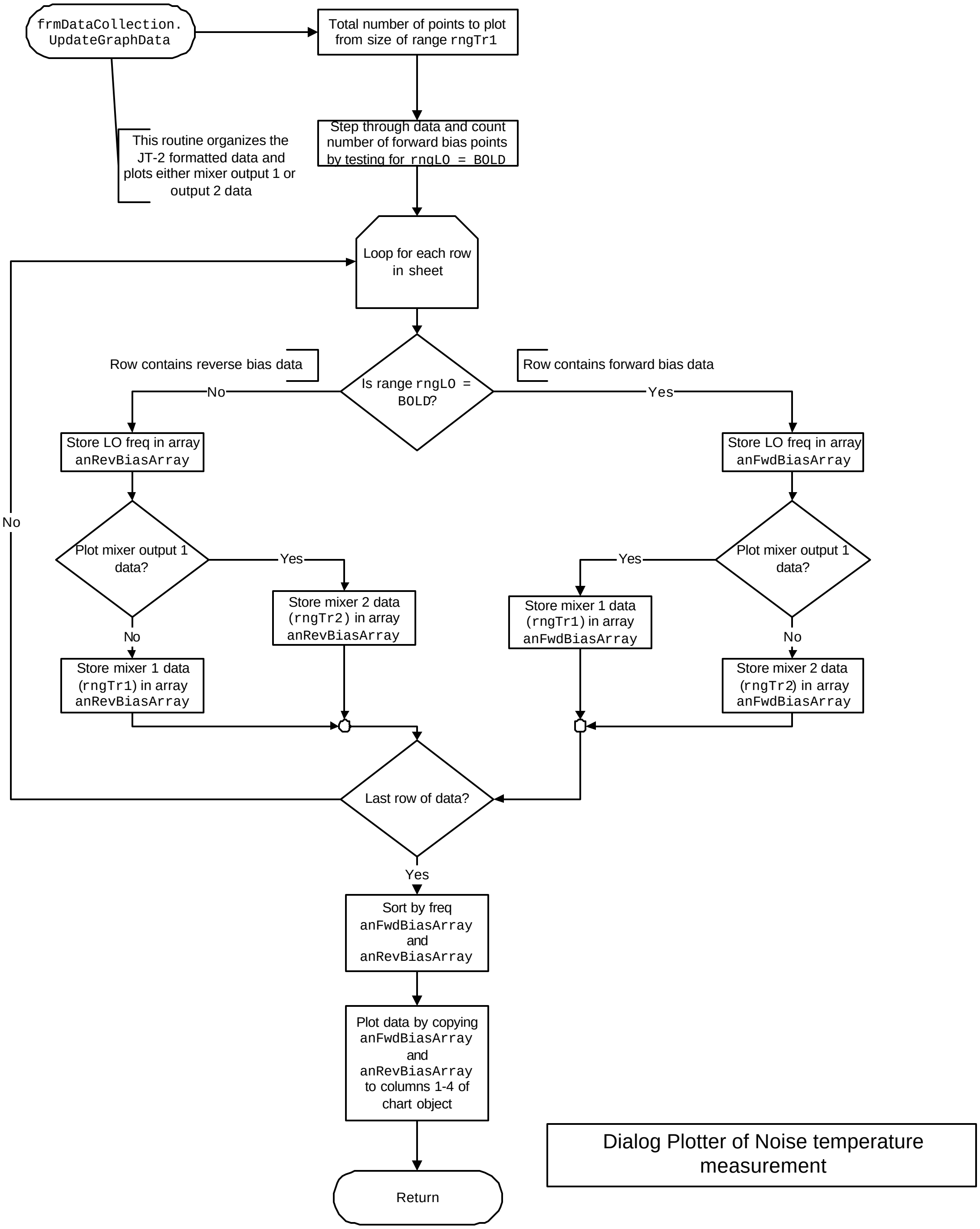


Figure 18: Flow Chart for Noise Measurement Dialog Plotter

10.1.1.1.1 Visual Basic Active X Routines

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 classes:

- `CNI16AnIn` – controls the analog ports on the interface board, which reads mixer bias voltage and current.
- `CNI16DigOut` – controls the digital output ports on the interface board, which set the Dewar switches. This class contains a number of data members, such as `m_ciDIO0`, which are used to define particular I/O lines using the same nomenclature as the schematics.

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.

10.1.1.1.2 GPIB

The `CGPIB` bass class interfaces with National Instrument's GPIB board software 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.
- `SGetError` returns a string containing a description of the GPIB error. The error routines tunnel down to the lowest level routine that returned an error and provide a series of error messages with the message from the lowest level routine at the bottom of the dialog box.

For example, the class `CHP438` contains methods required to access the HP 438 power meter as shown in Figure 19. The `nRead` method sends the string required by the HP 438 power meter to return power readings, and returns power readings as a single precision number.

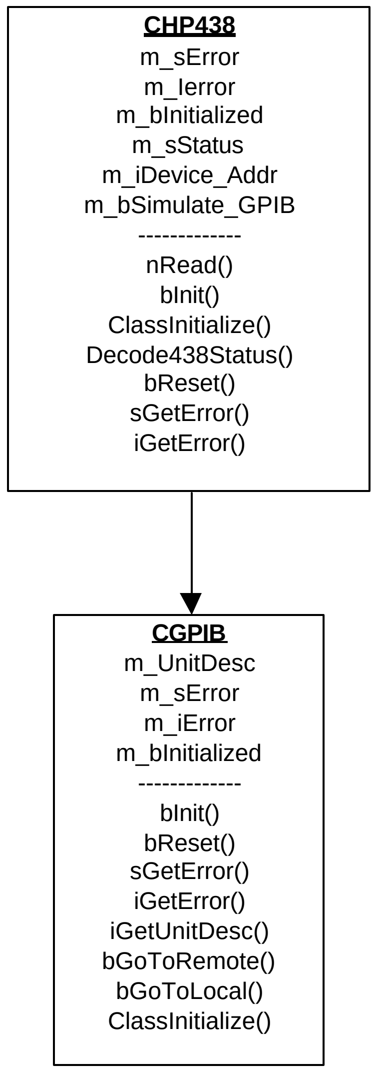


Figure 19: HP 438 interface to the GPIB Class

10.1.1.1.3 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.
- `CDataAcqForm` – this object contains routines to initiate reading each instrument and placing the data into the appropriate cell. Normally, this would be encapsulated in a `form` class, but there are two forms currently available to allow either maximum automation, or maximum flexibility in data acquisition.

Figure 21 is one of the dialog boxes 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.)

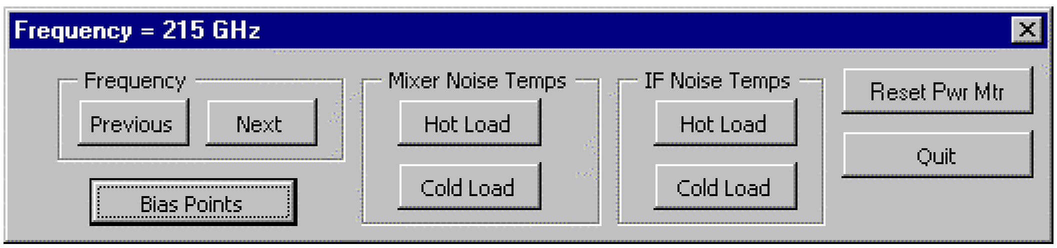


Figure 21: Excel Dialog Box for Data Acquisition

10.1.1.1.3.1 Computation of Standard Deviation

When each button in Figure 21 is pressed, the analog board is read 10 times and the power meter is read 5 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 for the analog board or 10 times for the power meter, after which the program continues by recording the value on the spreadsheet, and warning the user that the standard deviation exceeds limits. A note is also placed on the spreadsheet to warn that the standard deviation exceeded limits.

11. Chopper Wheel Data Acquisition

Hot/cold load power is measured by the square-law detector.

2) Output of square-law detector is routed to analog input on National Instruments analog card

3) A/D converters on the analog card are triggered by pulses from chopper wheel corresponding to hot and cold load openings in wheel.

4) Software polls analog card until n samples are obtained, then dumps data to memory buffer for averaging and real-time display processing.

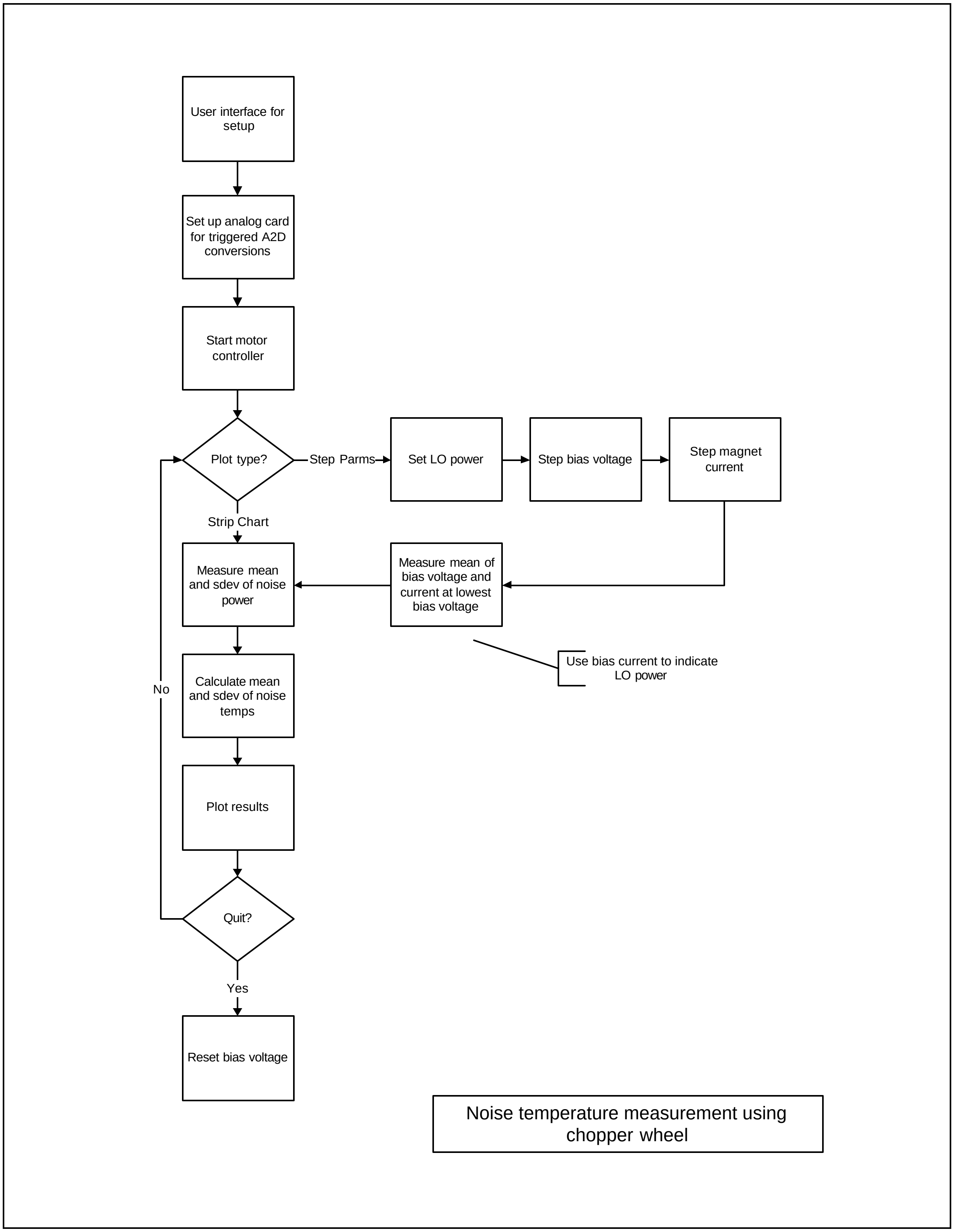


Figure 23: Noise Temperature Measurement Using Chopper Wheel

12. Database

Measurement data is stored in a Microsoft Access database file. The tables are designed to minimize data redundancy by using relational design concepts as defined in Section "13 Table Relationships".

13. Table Relationships

To minimize redundancy, data is spread across several tables as shown in the relationship diagrams in Figure 25 and summarized in the following paragraphs.

The top-level table (`SIS Mixer`) contains fields that describe the mixer undergoing testing, and related tables contain details about each measurement. The *type* of measurement and its associated data is stored in the table `MeasType`. The table `Meas` holds static results, that is, data that doesn't significantly change during a measurement. Data that changes for each measurement is stored in the child tables `Data`, `DependData`, `and` `Bias`. The table `Data` includes the following fields:

1. `Parameter`, which retains the value being stepped, such as magnet current,
2. `Independent`, which holds the independent quantity being changed, such as bias voltage.

The table `Data` is linked to the child table `DependData`, which allows for an unlimited number of dependent data terms.

`Bias` is another child table of `Meas`, and provides the structure required to support an unlimited number of mixer bias values. The table `Bias` is useful when either sweeping the bias voltage given the mixer state defined in `Meas`, or to record the bias point of four mixers in the sideband separating, balanced mixer design.

Instrument settings for a particular measurement (such as voltage limits and the number of steps) are stored in two tables. The table `SettingsDefault` contains standard setup values for each type of measurement. The information in this table is keyed to the field `TypeNum` in the measurement type definition table `MeasTypeDefs`. The settings actually used during a measurement are stored in the table `SettingsActual`, and that table is keyed to the `Data Key` field in the `MeasType` table, which allows these settings to be recalled for each measurement. The structure of the actual and default setting tables are identical.

Figure 20 shows how records are created and tables updated during a typical measurement. A new record is written to the table `SIS Mixer` whenever new mixer information is entered. Likewise, a new record is written into the table `MeasType` whenever a new measurement is defined for this mixer. When the measurement is actually started, parameters that change slowly, such as LO frequency and IF frequency, are recorded in the table `Meas` while parameters that change rapidly, such as receiver temperature and bias point, are entered into the generic table `Data`.

14. Temperature Measurement Table

The tables `TempMeters` and `TempSensors` provide data on the LakeShore temperature meters and sensor calibration factors. `TempSensors` contains fields for two types of sensor calibration: Single point and three-point calibrations. The field `Error` contains the temperature error (in K) for a single point calibration at 4.2K. The value of this field should be *added* to the temperature returned by the meter to obtain the corrected temperature. The fields `SensorUnits1` to `SensorUnits1` and `Temperature1 to Temperature1` respectively store information used by the three-point curve-fitting routine to calibrate the sensors. The data in these fields should be uploaded into the temperature meter.

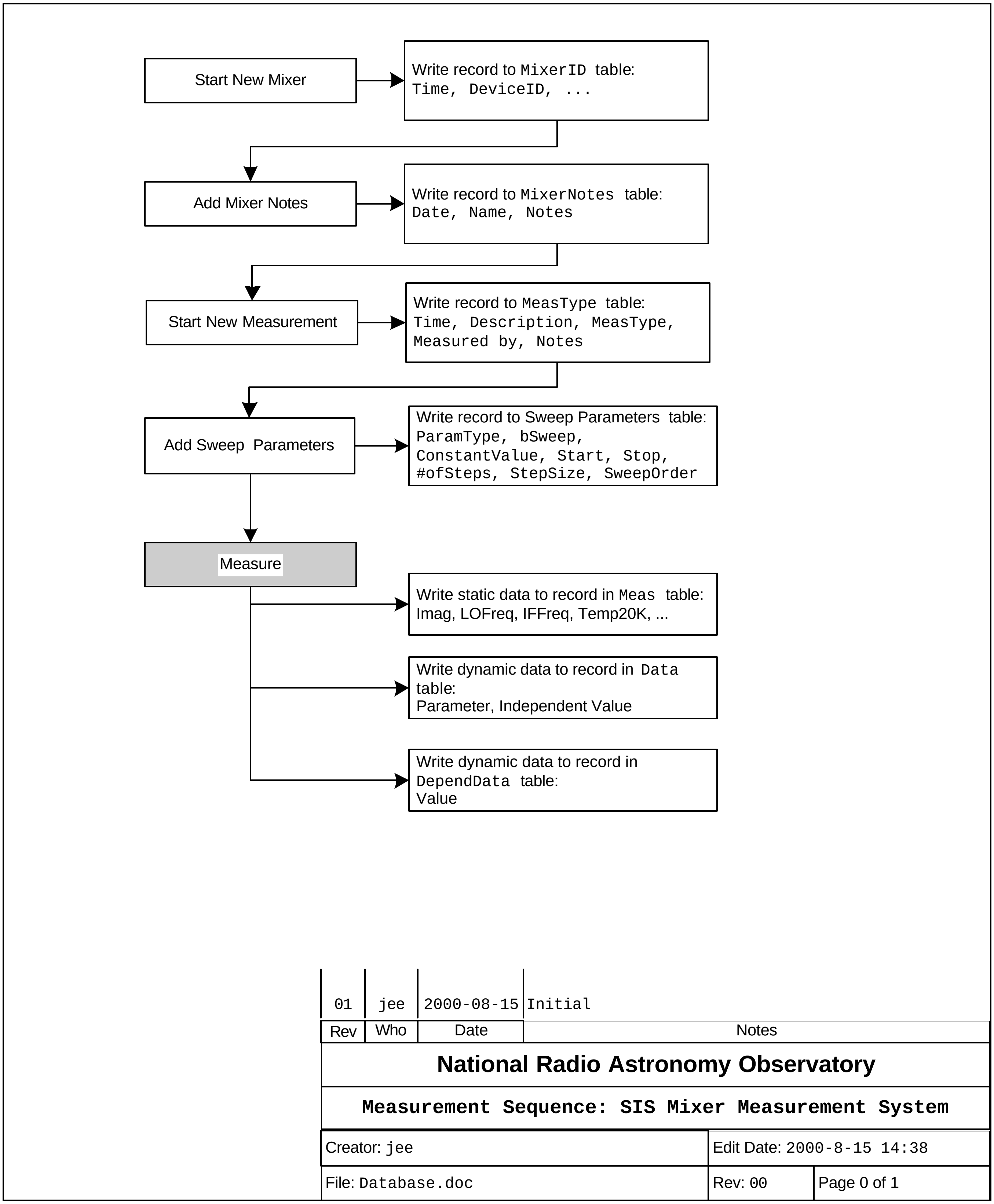
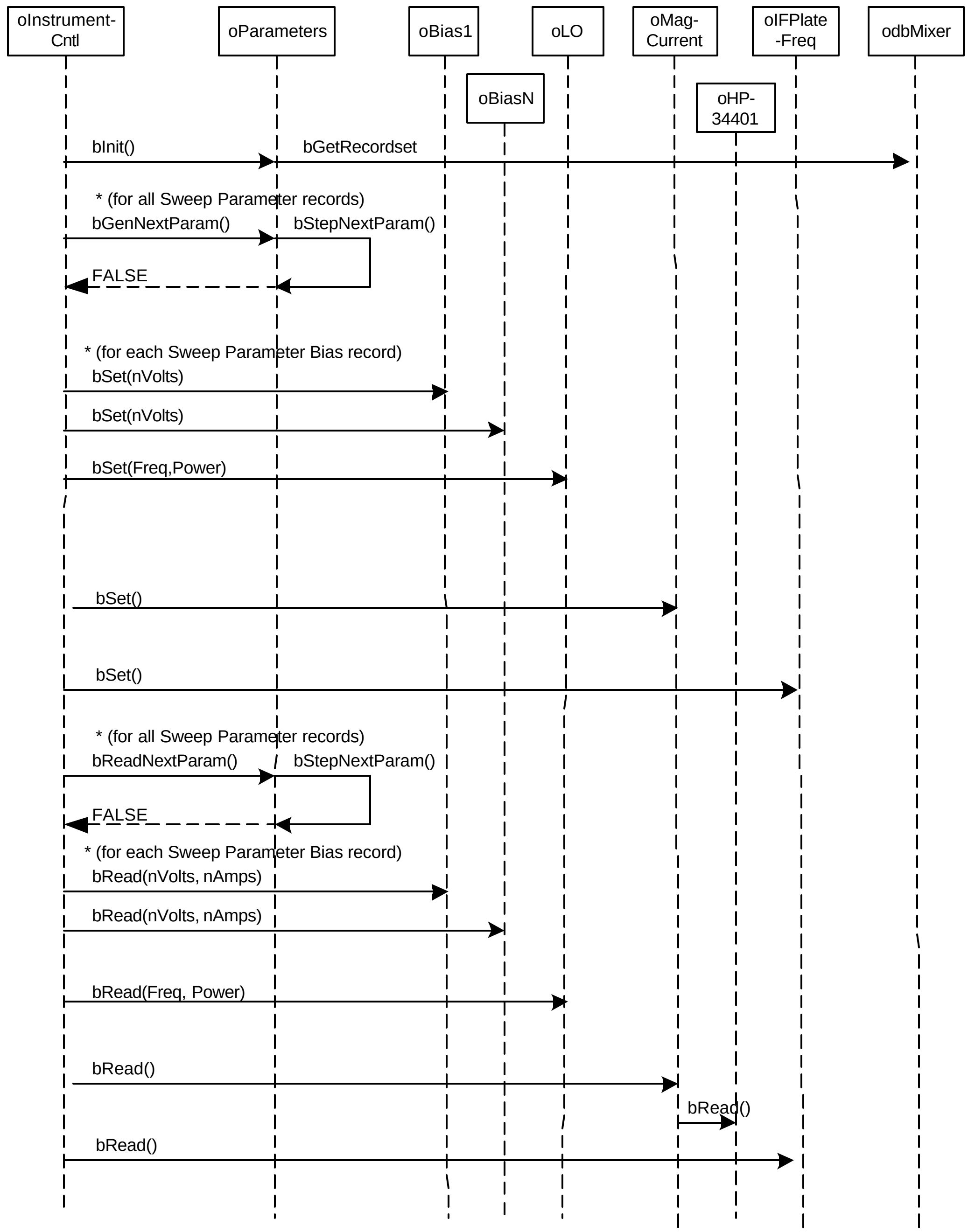


Figure 20: Database Records Created During a Measurement

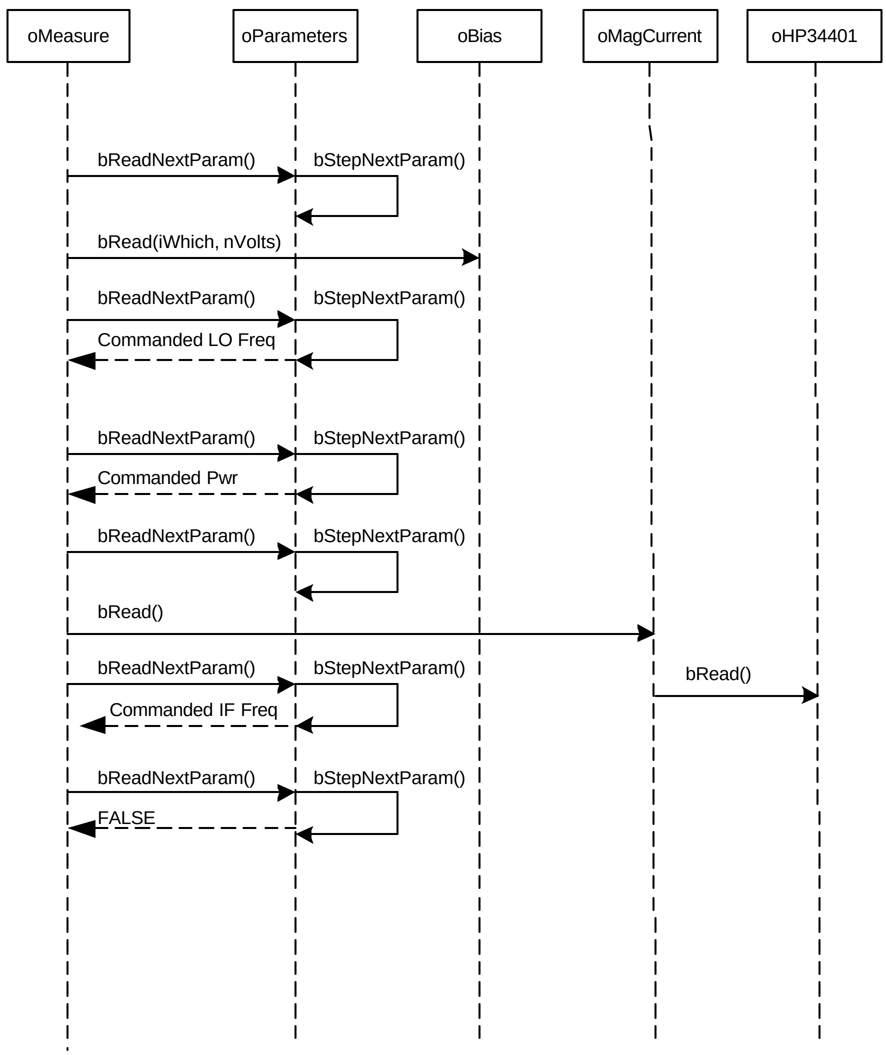


Rev	Who	Date	Notes
00	jee	2000-08-16	Initial

National Radio Astronomy Observatory

Instrument Setup Sequence: SIS Mixer Measurement System

Creator: jee	Edit Date: 2000-8-16 15:18	
File: Database.doc	Rev: 00	Page 1 of 1

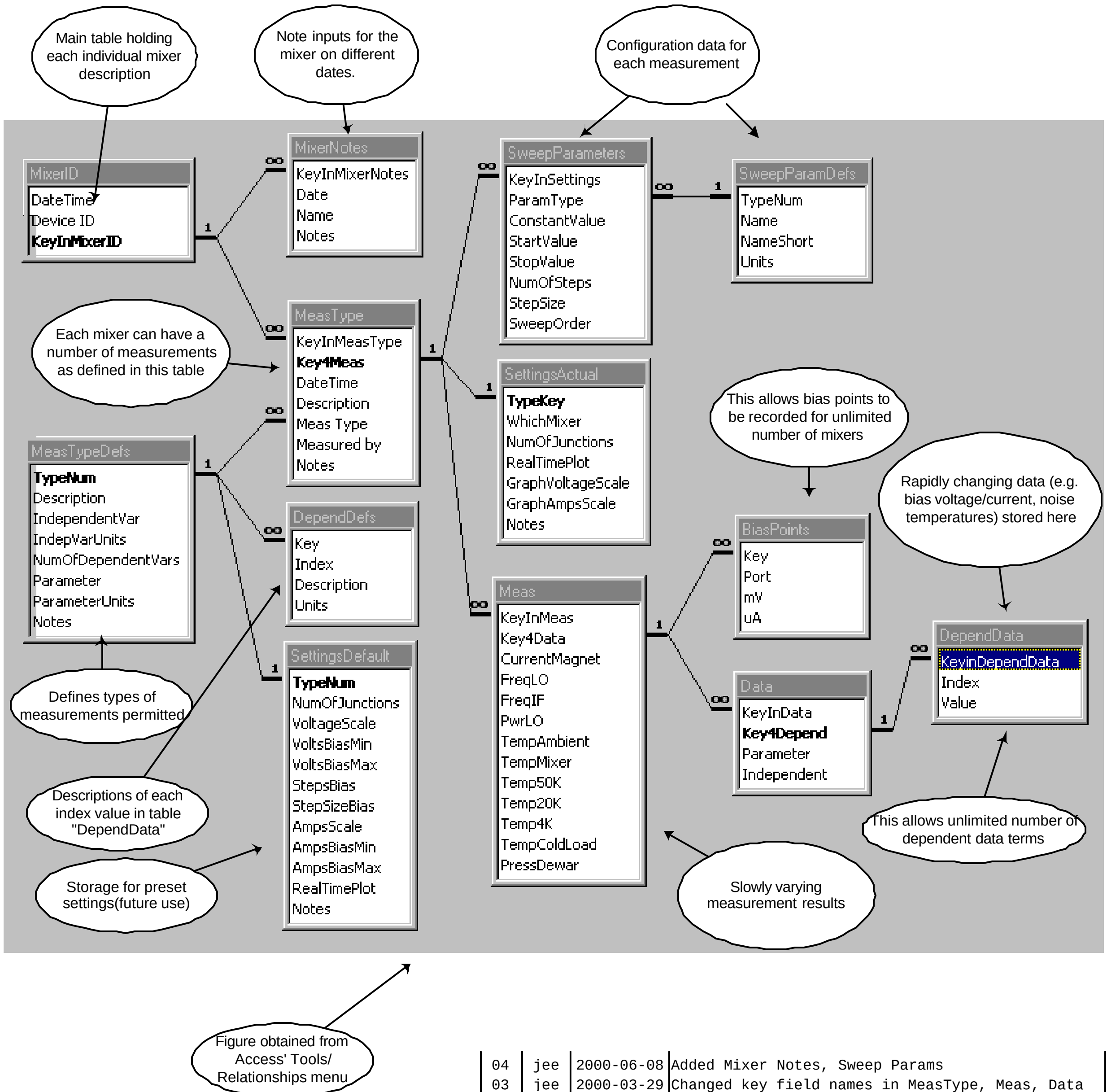


Rev	Who	Date	Notes
00	jee	2000-08-16	Initial

National Radio Astronomy Observatory

Measurement Sequence: SIS Mixer Measurement System

Creator: jee	Edit Date: 2000-8-16 14:05	
File: Database.doc	Rev: 00	Page 1 of 1



04	jee	2000-06-08	Added Mixer Notes, Sweep Params
03	jee	2000-03-29	Changed key field names in MeasType, Meas, Data
02	jee	2000-03-15	Added DependData, Bias, and DependDefs
01	jee	2000-03-03	Table Data linked to MeasType
Rev	Who	Date	Notes

National Radio Astronomy Observatory

SIS Mixer Measurement System: Database Structure

Creator: jee	Edit Date: 2000-6-6 15:09	
File: Design.doc	Rev: 04	Page 1 of 1

Figure 25: Database Table Relationships

15. Data Dictionary

```
Data Dictionary for Database: G:\MeasSys\Data\Database20000303\sis97a.mdb
Lising Routine Name         : "Get Data Dictionary"
Printed                     :  2000-15-03 13:19:42


*****************************
*****************************
Table:          Bias
Description:    I-V information
Last Updated:   2000-15-03 10:51:44
Record Count:   0
-----------------------------

Key              Long (4)    Keyed to DataKey in Meas
Port             Integer (2) Identifies Mixer
mV               Single (4)  Junction voltage, in mV
uA               Single (4)  Junction current, in uA


*****************************
*****************************
Table:          Bug Status
Last Updated:   2000-15-03 10:49:34
Record Count:   0
-----------------------------

KeyField         Long (4)    Field keyed to bug number in "Bug Report" table
Date             Date (8)    Date of status entry
Initials          Text (20)   Initials of person entering status report( i.e., person working on
the bug)
Notes            Memo (0)    Notes about this entry


*****************************
*****************************
Table:          BugReport
Last Updated:   2000-15-03 10:49:34
Record Count:   8
-----------------------------

IncidentNum      Long (4)    Number of bug report (used for key value)
Date             Date (8)    Date and time of report
Name of Reporter Text (25)   Name of person reporting bug
Class            Text (25)   Class of bug e.g. Meas Code, Anal Code,...
Details          Memo (0)    Description of Bug
Severity         Text (50)   Severity of Bug (Predefined choices)
Status           Text (50)   Status of bug (Predefined choices)


*****************************
*****************************
Table:          Data
Description:    Contains generic fields to record only changing data during a measurement
Last Updated:   2000-15-03 10:51:44
Record Count:   633571
-----------------------------
```

```
MeasRecKey        Long (4)    Maps to Record ID key in Meas Table
KeyDepend         Long (4)    Provides key for DependData table
Parameter         Single (4)  Parameter being stepped
Independent       Single (4)  Independent variable
Dependent1        Single (4)  Data field for first dependent variable
Dependent2        Single (4)  Data field for second dependent variable
Dependent3        Single (4)  Data field for third dependent variable


*****************************
*****************************
Table:          DependData
Description:    Holds dependency values
Last Updated:   2000-15-03 10:51:44
Record Count:   0
-----------------------------

Key               Long (4)    Keyed to DataKey in Meas
Index             Integer (2) Number of dependency as defined in MeasTypeDefs
Value             Single (4)  Value of dependency


*****************************
*****************************
Table:          DependDefs
Description:    Describes dependency data
Last Updated:   2000-15-03 10:51:44
Record Count:   0
-----------------------------

Key               Integer (2) Keyed to TypeNum in MeasTypeDefs
Index             Integer (2) Identifies type of dependency
Description       Text (20)   Descriptive text of dependancy


*****************************
*****************************
Table:          Meas
Description:    Includes fields for all variables of a measurement
Last Updated:   2000-15-03 10:51:44
Record Count:   597
-----------------------------

Record ID         Long (4)    Maps to appropriate record in SIS Run table
DataKey           Long (4)    Key for tables linked to this table
Voltage Bias 1    Single (4)  Bias voltage (device 1)  in volts
Current Bias 1    Single (4)  Bias current (device 1) in mA
Voltage Bias 2    Single (4)  Bias voltage  (device 2) in volts
Current Bias 2    Single (4)  Bias current (device 2)  in mA
Current Magnet    Single (4)  Magnetic field coil current in mA
Freq LO           Single (4)  Frequency of local oscillator in GHz
Freq IF           Single (4)  Intermediate frequency used in GHz
Pwr LO            Single (4)  Power of local oscillator in mW
Temp Ambient      Single (4)  Ambient temperature in degs C
Temp LN2          Single (4)  Hot load physical temperature in K
Temp Mixer        Single (4)  Mixer  temperature in K
Pwr Th RF         Single (4)  Hot-Load power at RF input to mixer (dBm)
Pwr Tc RF         Single (4)  Cold-Load power at RF input to mixer (dBm)
Pwr Th IF         Single (4)  Hot-Load power at Mixer IF (dBm)
Pwr Tc IF         Single (4)  Cold-Load power at mixer IF (dBm)
Pwr Noise Mix Out Single (4)  Noise power reflected from mixer output (mW)
```

```
****************************
****************************
Table:         MeasType
Description:   Defines the type of measurement
Last Updated:  2000-15-03 10:51:44
Record Count:  814
----------------------------

Record ID        Long (4)    Maps to appropriate record in SIS Mixer table
Data Key         Long (4)    Key for child tables linked to this table
DateTime         Date (8)    Date and time of individual measurement
Description      Text (128)  Description of this measurement
Meas Type        Integer (2) Type of Measurement
Measured by      Text (15)   Name of person taking the data
Notes            Memo (0)    General notes for each entry


****************************
****************************
Table:         MeasTypeDefs
Description:   For a particular measurement, defines the generic fields in the Data table
Last Updated:  2000-15-03 10:51:44
Record Count:  6
----------------------------

TypeNum          Integer (2) Measurement type number
SubTypeNum       Integer (2) Measurement sub type number
Description      Text (50)   Test description
IndependentVar   Text (20)   Description of independent variable
NumOfDependentVars Integer (2) Number of dependent variables
DependentVar1    Text (20)   Description of first dependent variable
DependentVar2    Text (20)   Description of second dependent variable
DependentVar3    Text (20)   Description of third dependent variable
Parameter        Text (20)   Description of parameter variable
Notes            Memo (0)


****************************
****************************
Table:         PressTemp
Description:   Pressure and temperature data
Last Updated:  2000-15-03 10:52:20
Record Count:  29129
----------------------------

MeasKey          Long (4)    Maps to key value in other tables
DateTime         Date (8)    Time of measurement
Temp77K          Single (4)  Temperature of 77K stage
Temp20K          Single (4)  Temperature of 20K stage
Temp4K           Single (4)  Temperature of 4K stage
TempHotLoad      Single (4)  Temperature of hot load
TempColdLoad     Single (4)  Temperature of ColdLoad
TempMixer        Single (4)  Temperature of mixer stage
PressVacPump     Single (4)  Pressure at vacuum pump
PressDewar       Single (4)  Pressure in dewar
FlowRateHe       Single (4)  Helium flow rate
TempAmbient      Single (4)  Ambient Temperature


****************************
****************************
```

```
Table:         SettingsActual
Description:   Contains actual equipment settings for a measurement
Last Updated:  2000-15-03 10:52:21
Record Count:  801
----------------------------

TypeKey          Long (4)    Maps to appropriate record in MeasType table
WhichMixer       Integer (2) Mixer 1 or Mixer 2
NumOfJunctions   Integer (2) Number of SIS junctions / mixer
MeasMixerAmps    Boolean (1) If yes, measure mixer current
MeasMixerNoise   Boolean (1) If yes, measure mixer noise temperature
MeasIfPower      Boolean (1) If yes, measure IF power level
BiasVoltsMin     Single (4)  Minimum bias voltage in mV
BiasVoltsMax     Single (4)  Maximum bias voltage in mV
BiasVoltsNumOfSteps Long (4)    Number of voltage steps for each bias sweep
BiasVoltsScale   Single (4)  Scale factor for graph
BiasAmpsMin      Single (4)  Minimum scale for bias plot in uA
BiasAmpsMax      Single (4)  Maximum scale for bias plot in uA
BiasAmpsScale    Single (4)  Scale factor for graph
LOPwrMax         Single (4)  Maximum or Fixed LO power in mW
LOPwrMin         Single (4)  Minimum LO power in mW
LOPwrNumOfPwrs   Integer (2) Number of power levels stepped during measurement
LOFreqMax        Single (4)  Maximum or Fixed LO frequency, in GHz
LOFreqMin        Single (4)  Minimum LO frequency in GHz
LOFreqNumOfFreqs Integer (2) Number of LO frequencies for each sweep
MagCurMax        Single (4)  Maximum or Fixed magnet current in mA
MagCurMin        Single (4)  Minimum magnet current in mA
MagCurNumOfCurs  Integer (2) Number of magnet current levels stepped during measurement
RealTimePlot     Boolean (1) Is real-time plot turned on for this measurement?
GraphVoltageScale Single (4)  Voltage scale for graph in mV/cm
GraphAmpsScale   Single (4)  Current scale for plot in uA/cm
Notes            Memo (0)


****************************
****************************
Table:         SettingsDefault
Description:   Contains default equipment settings for a measurement
Last Updated:  2000-15-03 10:52:21
Record Count:  0
----------------------------

TypeNum          Integer (2) Maps to appropriate record in MeasType table
NumOfJunctions   Integer (2) Number of SIS junctions / mixer
VoltageScale     Single (4)  Voltage scale for graph in mV/cm
VoltsBiasMin     Single (4)  Minimum bias voltage in mV
VoltsBiasMax     Single (4)  Maximum bias voltage in mV
StepsBias        Long (4)    Number of voltage steps for each bias sweep
StepSizeBias     Single (4)  Size of each step for bias sweep in mV
AmpsScale        Single (4)  Current scale for plot in uA/cm
AmpsBiasMin      Single (4)  Minimum scale for bias plot in uA
AmpsBiasMax      Single (4)  Maximum scale for bias plot in uA
RealTimePlot     Boolean (1) Is real-time plot turned on for this measurement?
Notes            Memo (0)


****************************
****************************
Table:         SIS Mixer
Description:   Contains mixer information, such as device ID, date/time, etc
Last Updated:  2000-15-03 10:52:21
Record Count:  41
```

```
-----------------------------

DateTime          Date (8)    Date and time information is entered
Device ID         Text (50)   Description of device
Notes             Memo (0)    Free-Field input for general annotation
Meas ID           Long (4)    Key for child tables linked to this table


*****************************
*****************************
Table:          Version Info
Description:    Documents changes to database.
Last Updated:   2000-15-03 10:49:08
Record Count:   7
-----------------------------

Item              Text (20)   Applicable item to track version information, such as table name,
code routine name...
Version Number    Text (10)   Version Number
Date              Date (8)    Date of new version
Initials          Text (20)   Name of person entering version info.
Notes             Memo (0)    Notes about this version
```

15.1.1 Database Access Classes

15.1.1.1 CdbMixer

The CdbMixer class contains the low-level routines for accessing the database. All SQL statements are contained in this class and are setup during class initialization. This class is packaged as an Active-X DLL so that it can be shared by both the VB and VBA routines.

CdbMixer uses the CQueries class to generate the parameterized queries. A call to OpenDataBase contains the …

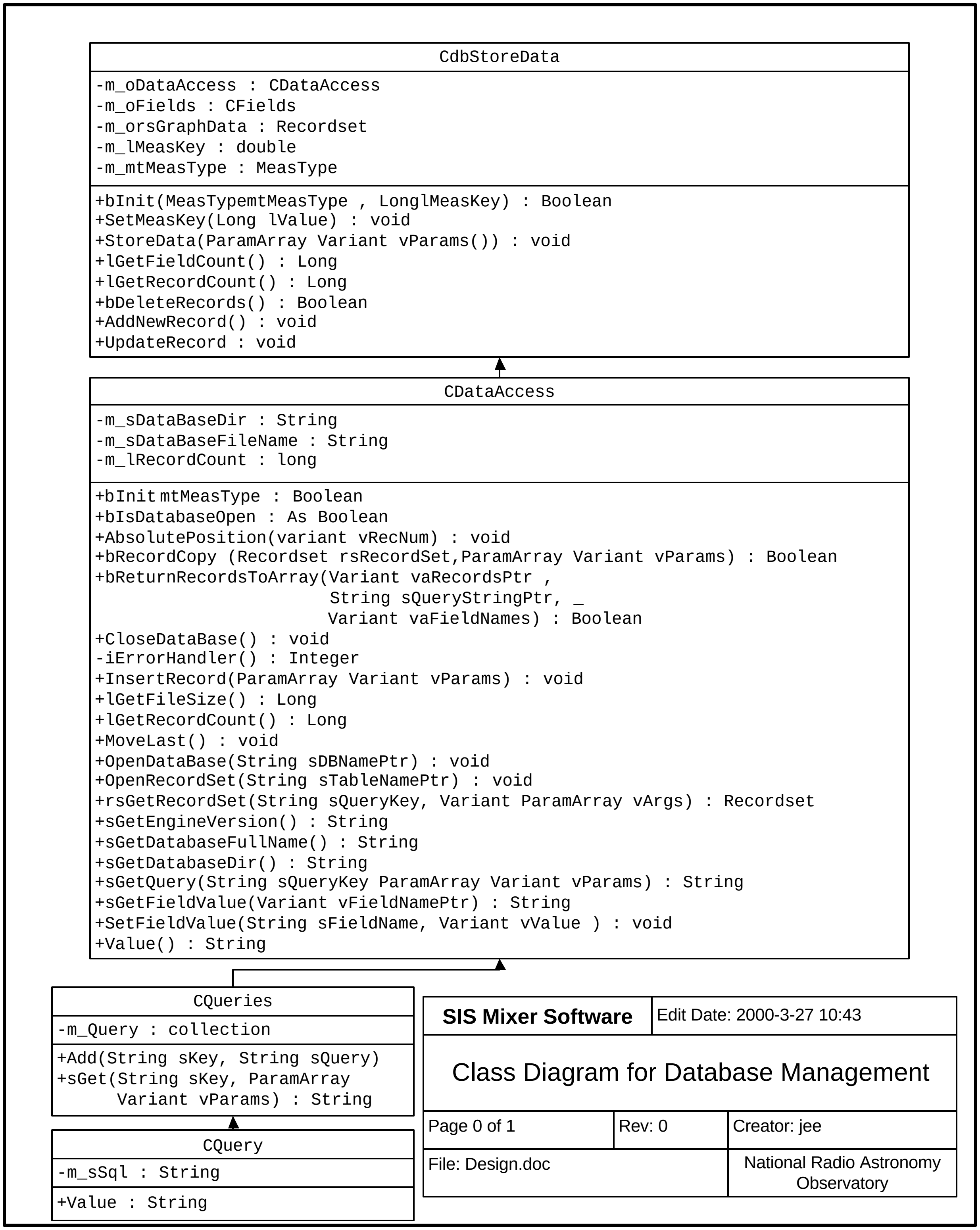


Figure 23: Class Diagram for Database Management

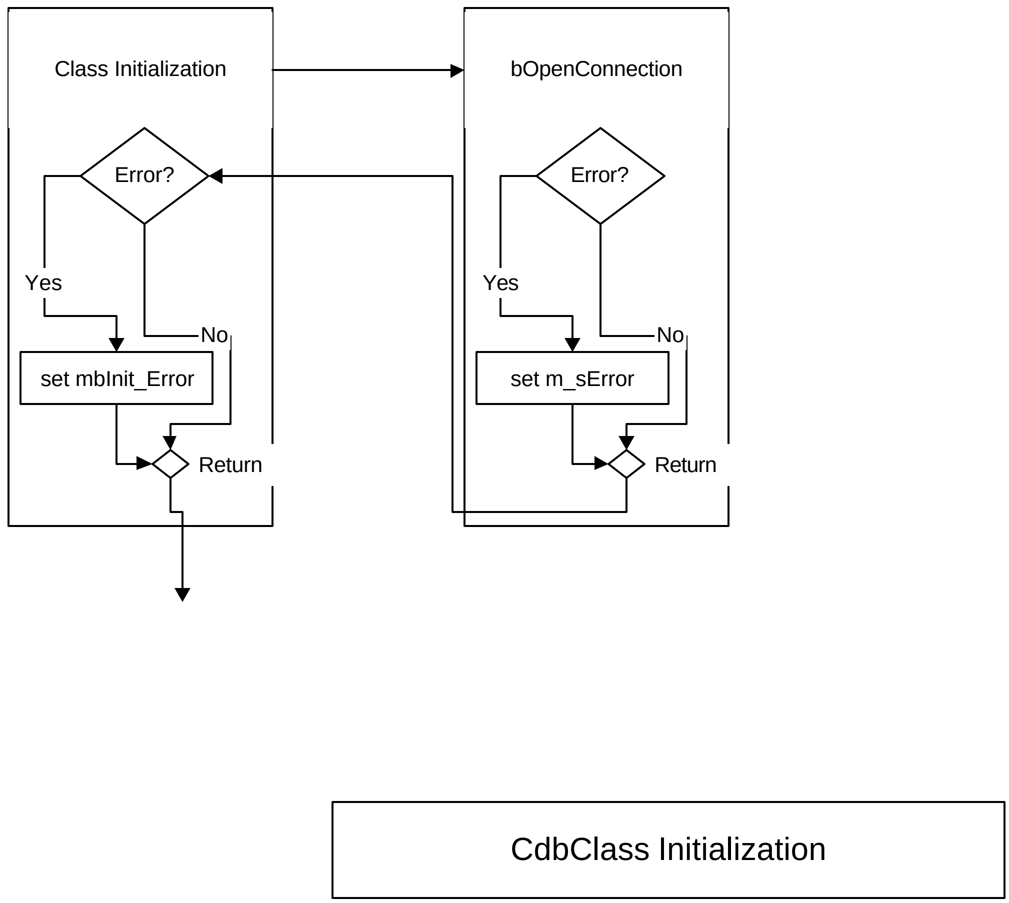


Figure 24: CdbClass Initialization

16. Graphing and Data Analysis

The plotting program for I-V data is written in Visual Basic for Applications (VBA) that is included with the spreadsheet program Excel.

17. Analysis Routines

As the data are acquired, it is immediately stored in the measurement database on a server accessible to all team members. Afterwards, the data is accessed and analyzed using the Excel spreadsheet program. Previously measured data can be analyzed at the same time that new data are acquired.

In the data analysis prototype, the user selects the desired measurement with a dialog box (see Figure 25) that displays device ID and measurement date in drop-down boxes. The dialog box then displays any notes associated with this measurement.

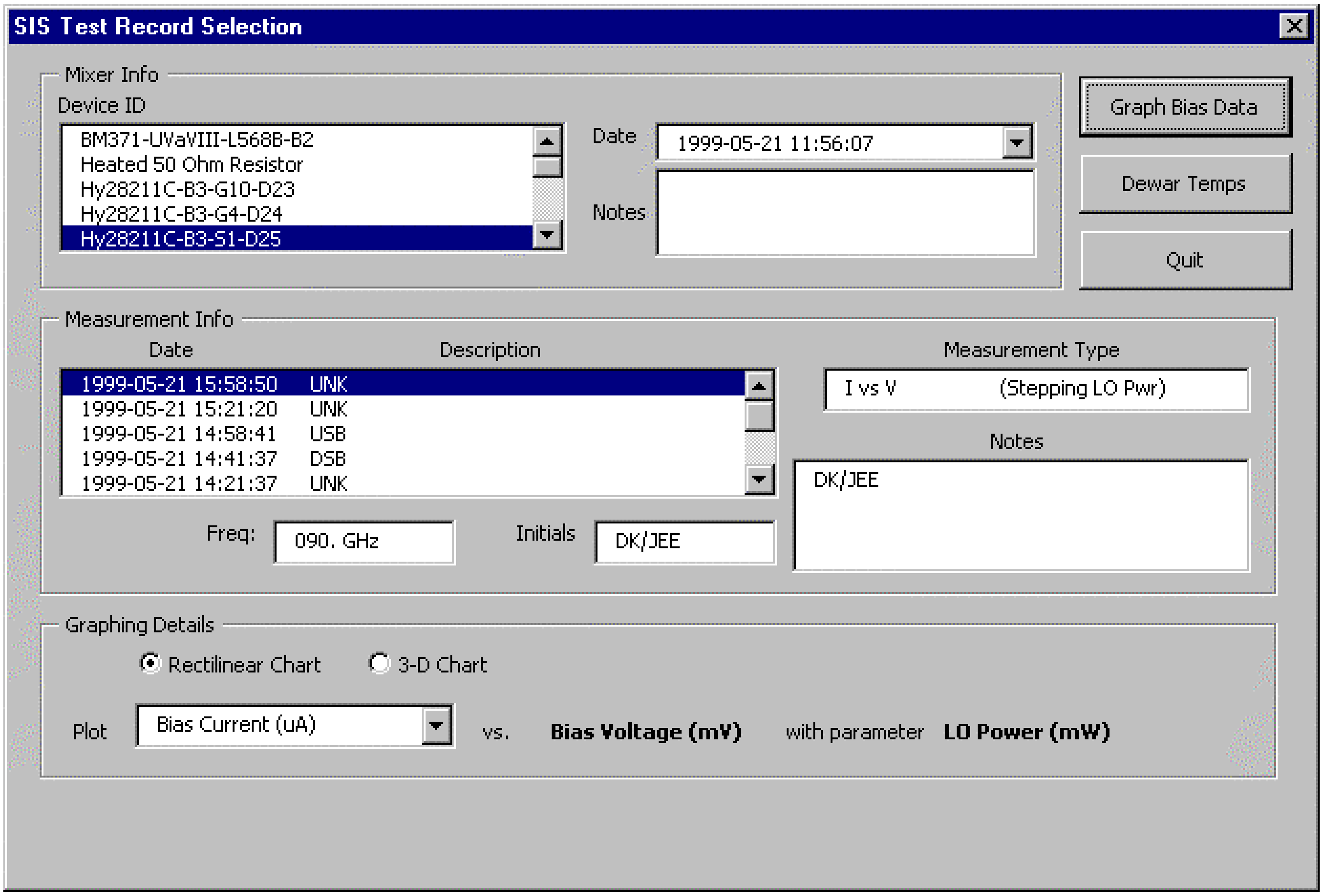


Figure 25: I-V Plotting Dialog Box

When the “Graph Bias Data” button is pressed, the database is queried to return the desired data to a spreadsheet, which is then automatically graphed in Excel. A portion of the results and graph is shown in Figure 26.

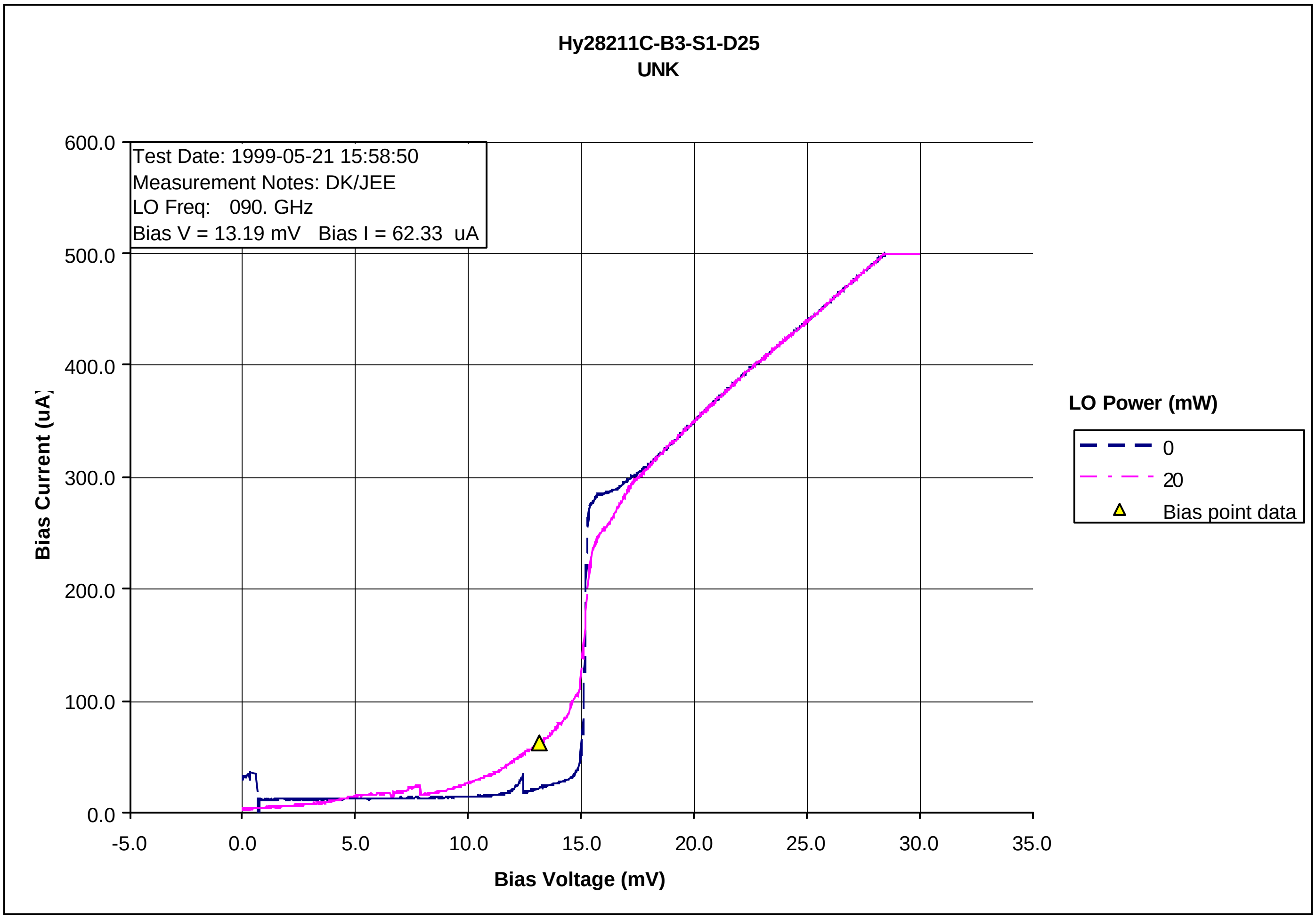


Figure 26: Returned Measurement Data and Graph

17.1.1 File Locations

18. General

Table 5: General File Locations	
The file containing this design document	`F:\Docs\Software\SISMeasSys\Design.doc`
Excel data analysis program	`\\eagle\cv-cdl-sis\MeasSys\Software\IVPlot\SISIVPlotV4.10.xls`
Visual Basic measurement program	`\\eagle\cv-cdl-sis\MeasSys\Software\BiasMeasV.xx\BiasMeas.vpb`
Access database file	`\\eagle\cv-cdl-sis\MeasSys\Data\SIS97a.mdb`

19. Excel Data Collection

Current file locations for these routines are given in Table 6. All file locations are relative to the root directory.

Table 6: File Locations for Excel Data Collection Routine

Description	Type	Location
Root Directory:		`\\jt2\pub\sismeas\ExcelIntf8`
The file containing this design document		`F:\Docs\Software\SISMeasSys\Design.doc`
Startup Module	Module	`Start.bas`
National Instrument Globals for Analog board	Module	`NI_Code\Niglobal.bas`
National Instrument Globals for GPIB	Module	`GPIB_Code\Vbib-32.bas`
Class for GPIB board	Class	`GPIB_Code\CGPIB.cls`
Class for Analog board	Class	`NI_Code\CNI16AnIn.cls`
Class for Standard Deviation	Class	`Common_Code\CStandardDeviation.cls`
Class for HP34401 multimeter	Class	`GPIB_Code\CHP34401.cls`
Class for HP436 Power Meter	Class	`GPIB_Code\CHP436.cls`
Form containing National Instrument's Component Works analog control	Form	`Control.frm`
Excel spreadsheet containing code	Spread-sheet	`Test.xls`

Figure 27 is a listing from the Visual Basic Project file (of type .vbp) that shows the relative directory locations for each module in the program as well as the plug-in components used. All directory locations are relative to the `.vbp` file whose directory was defined in Table 5.

```
Type=Exe
Reference=*\G{00025E01-0000-0000-C000-000000000046}#4.0#0#C:\Program
Files\Common Files\Microsoft Shared\DAO\DAO350.DLL#Microsoft DAO 3.0 Object
Library
Reference=*\G{00020813-0000-0000-C000-000000000046}#1.2#0#D:\Program
Files\Microsoft Office\Office\EXCEL8.OLB#Microsoft Excel 8.0 Object Library
Object={7A080CC8-26E2-101B-AEBD-04021C009402}#1.0#0; GAUGE32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.1#0; DBLIST32.OCX
Reference=*\G{0D452EE1-E08F-101A-852E-
02608C4D0BB4}#2.0#0#C:\WINNT\System32\FM20.DLL#Microsoft  Forms  2.0  Object
Library
Object={831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0; MSCOMCTL.OCX
Object={65E121D4-0C60-11D2-A9FC-0000F8754DA1}#2.0#0; MSCHRT20.OCX
Module=VBIB32; ..\Shared\gpib\Vbib-32.bas
Module=Globals; ..\Shared\Globals.bas
Class=CFields; CFields.cls
Class=CField; CField.cls
Class=CdbStoreData; CdbStoreData.cls
Class=CQueries; CQueries.cls
Class=CQuery; CQuery.cls
Class=CSISDevice; CSISDevice.cls
Form=BiasMeas.frm
Form=fmMeasData.Frm
Form=fmBiasSettings.frm
Class=CGPIB; ..\Shared\gpib\CGPIB.cls
Class=CNI16AnIn; ..\shared\NatInst\CNI16AnIn.cls
Class=CStandardDeviation; ..\shared\CStandardDeviation.cls
Class=CNI16AnOut; ..\shared\NatInst\CNI16AnOut.Cls
Module=NIGLOBAL; ..\shared\NatInst\Niglobal.bas
Form=MainIntf.frm
Form=fmLOPwrSettings.frm
Form=fmMagAmpSettings.frm
Class=CDataAccess; CDataAccess.cls
Designer=fmSQLError.frm
Class=CSettings; CSettings.cls
Class=CBias; CBias.cls
Class=CHP3631; ..\Shared\gpib\CHP3631.cls
Class=CSweep; CSweep.cls
Class=CLS218; ..\Shared\gpib\CLS218.cls
Form=fmTempMon.frm
Class=CTimer; ..\Shared\CTimer.cls
Module=Notes; Notes.bas
Module=NI_Constants; ..\shared\NatInst\nidaqcns.inc
Module=NIDAQ32; ..\shared\NatInst\nidaq32.bas
Module=NIDAQError; ..\shared\NatInst\nidaqerr.inc
Class=CChannels; ..\shared\NatInst\CChannels.cls
Module=MainRoutines; MainRoutines.bas
Form=..\Shared\Splash.frm
```

Figure 27: Partial Listing from Visual Basic Project (.vbp) File

20. Classes

21. Generic Parameter Handling

To increase program flexibility, the parameter to be measured will be displayed in generic "Parameter" section on the main interface dialog box. This also simplifies determining which parameters are to be plotted by the graphing routine. Figure 28 shows the current approach, which has hard-coded parameters in the main interface, and Figure 29 shows the concept of the new approach.

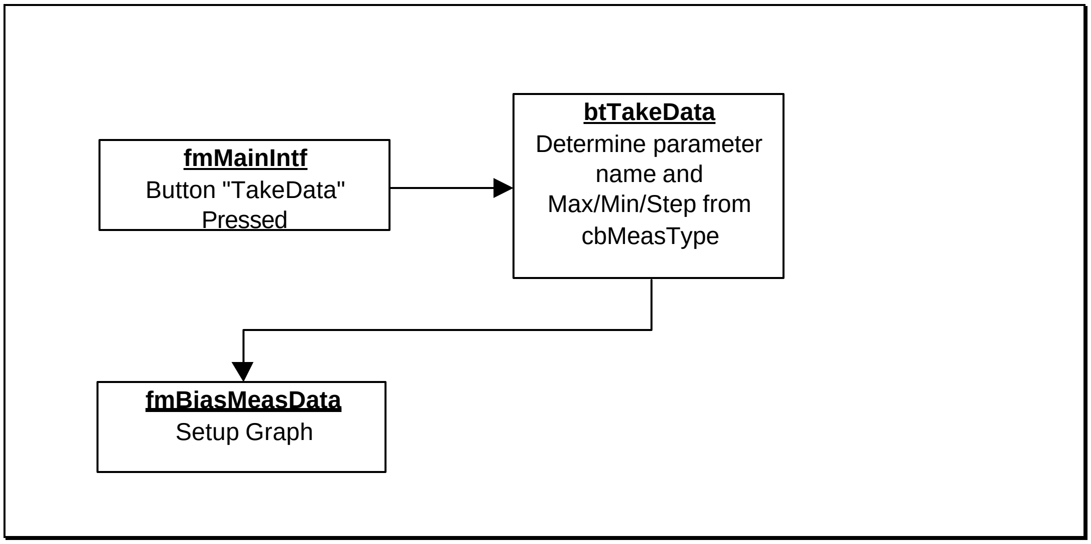


Figure 28: Determination of Parameter to be Measured (Current Approach)

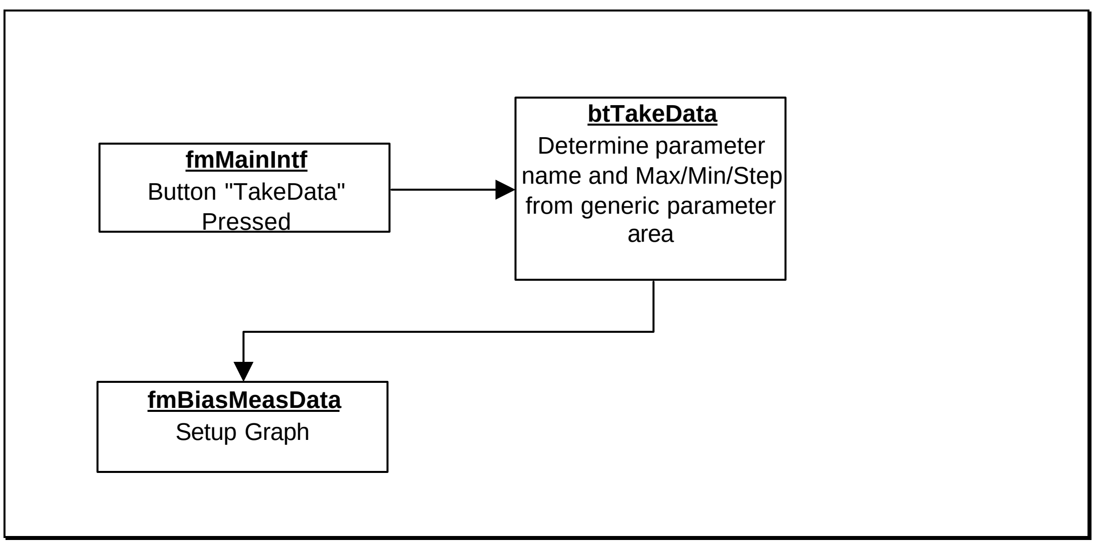


Figure 29: Determination of Parameter to be Measured (Future Approach)

22. Encapsulation

Database access has been encapsulated using a number of classes, as diagrammed in Figure 30, to isolate most of the software from inevitable database changes, such as adding field names and tables. The class `CdbGraphData` contains routines to access fields in the `Graph Data` table, and was built with the child classes, `CDataAccess` and `CQueries`[1]. `CDataAccess` provides routines to access the generic measurements database while `CdbGraphData` contains routines to access the specific database table `Graph Data`.

The `CFields` and `CField` child classes isolate database field names by using the `Add` and `StoreData` methods. `Add` initializes a mapping from the database field name (*e.g.*, `MeasKey`) to the name used to reference this field in the code (*e.g.*, `Key`). If the field name `MeasKey` is changed in the database, only the single routine containing all the `Add` methods requires changing. The implementation and use of this class is shown below:

Implementation:

```
m_cFields.Add "MeasKey", "Key"
```

Use:

```
Call dbGraphData.StoreData(lbDataSetNum, "Key")
```

The `Add` method maps the string `Key` to the field name `MeasKey`. The `StoreData` method is then used later to store the value in the variable `lbDataSetNum` in the `MeasKey` field by referring only to the key value `Key`.

The `CQueries` and `CQuery` child classes allow all database queries (SQL statements) to be located in a single routine (`CDataAccess.Class_Initialize()`). The `Add` method for this class maps a key value of string type (e.g., `Date_X_Y`) to a specific query, as exemplified below:

```
Call m_Query.Add("DATE_X_Y", "SELECT [Date],[DataX], [DataY] FROM [Graph Data] " &_
                  "WHERE [MeasKey] = ~")
```

The character ~ is used above as a delimiter so that a parameter can be inserted into the query at run time. Parameter insertion is completely general: An unlimited number of parameters can be inserted into the query at run time by including them in the `rsGetRecordSet` call, as shown below. Of course, during coding the same number of delimiters must be inserted in the appropriate locations in the query statement contained in the `Add` method.

Specific queries are referenced using a call to the `CDataAccess.rsGetRecordSet` method with the appropriate key values and parameters:

```
Set m_rsMeas_Date = fmMeasDataInput.GetDataAccessObjectPtr.rsGetRecordSet("DATE_X_Y", &_
                                                                           lRecordID)
```

This call inserts the contents of the variable `lRecordID` into the query defined in the `Add` method by the key `DATE_X_Y`. The location where the parameter is inserted is at the ~ as shown in the above `Add` statement.

[1] Class diagrams in this document show each class as a box with data members listed at the top and the methods with trailing parenthesis below the data member list. Class inheritance is indicated by convention with an arrow pointing towards the child class. Actually, Visual Basic doesn't allow class inheritance, but it was crudely simulated by calling child functions using the same method names in the parent classes.

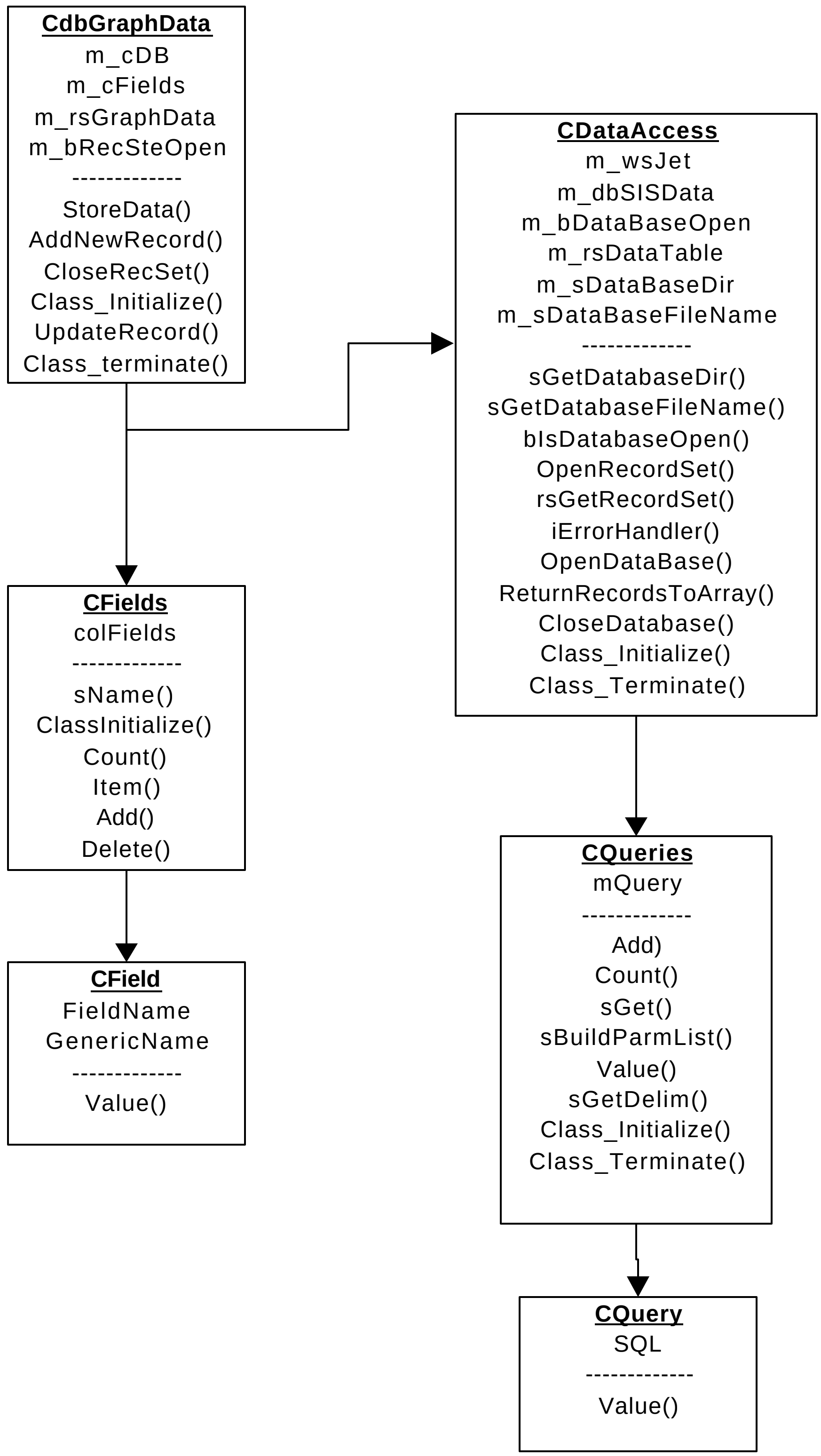


Figure 30: Database Classes

Empirical testing shows that overhead penalties incurred by these encapsulation functions are insignificant compared to other delays when measuring and storing data.

23. CdbStoreData

Stores measurement data in the database. Relevant database tables for a particular measurement are selected in the class method `StoreData` by examining the measurement type member, `m_iMeasType`, which is set during the initialization function, `bInit`. The key value (`Record ID`) that identifies the particular data record is also stored as a data member `m_iMeasType` in this class and is changed through the class method `SetMeasKey`.

CdbStoreData

```
bInit(mtMeasType, lMeasKey)
    Select Case mtMeasType
         Case IvsVwLOStep, IvsVwMagStep:
             m_oFields.Add "KeyInData", "KeyInData"
             m_oFields.Add "Independent", "Voltage"
             ...
             Set m_orsData = m_oDataAccess.rsGetRecordSet(rsType.IvsVwLOpwr, lMeasKey)
          ...
         End Case
    End Select
```

24. CSISDevice Class

This class contains information about:

1. Voltage command range
2. Current range
3. Voltage/Current scaling
4. Number of mixer devices
5. Measurement type
6. Number of times independent variable steps changes during measurement

Voltage output is calculated using the following constraints:

When StepNum = 1	$V_{out} = V_{min}$
When StepNum=NumSteps	$V_{out} = V_{max}$

And this defines V_{out}:

$$V_{out} = V_{\min} + \left[\frac{StepNum - 1}{NumSteps - 1}\right]\left[V_{\max} - V_{\min}\right]$$

or, rearranging

$$V_{out} = V_{\min} + \left[\frac{V_{\max} - V_{\min}}{NumSteps - 1}\right]\left[StepNum - 1\right]$$

where the "scale" is defined as:

$$Scale = \frac{V_{\max} - V_{\min}}{NumSteps - 1}$$

Now the standard definition of V_{min} is two gap voltage units below zero. For example, for a 4 junction device, the gap voltage is 4 times 0.2V = 0.8V, so V_{min} = -1.6V

24.1 IV-Curve Plotting Software Dialog Box Programming Details

The list box `lbMeasDescription` holds the measurement date and description for each mixer. It is populated in the routine `lbDeviceID_Click` and contains three fields – the measurement date, the description, and the measurement key. (The measurement key field is hidden from the user.)

25. Variable Naming Conventions

The naming conventions used in the software are given in Table 7.

Table 7 : Software Naming Conventions

(Recommended prefixes are given in bold)

Name	Variable Type	Notes
Variables		
iCounter	Integer	
bFault	Boolean	
lRows	Long	
ndB	Single	
dFreq	Double	
sName	String	
vDBArg	Variant	
oSheet	Object	
Functions		
Functions that return a data type should be named using the variable naming convention given above.		
Classes		
CMeter	Class	
m_Init	Data member of class	
Forms and Controls		
frmMain	Form	
cbMixers	Combo box on a form	
lbMeasurements	List box on a form	
tbNotes	Text box on a form	

dcMixers	Data control on a form	
optbLO	Option button control on a form	
chkbSimulate	Check box on a form	
btnQuit	Button on a form	
labText	Label on a form	

26. Hardware Addresses

Table 8 and Table 9 tabulate the hardware addresses when the DIO lines are used for the base address. Table 10 depicts hardware addresses when the PB5 to PB3 data output lines are used for the base address. There are three base address lines and three secondary address lines. The fourth base address line, DIO3, is used to latch the addresses.

In these tables, empty cells representing bit positions are generally unused.

Digital line **DIO-7** provides the trigger pulse for the data acquisition clock generated by the chopper wheel. This pulse gates the CONVERT pulses used by the National Instruments DAQ board to control the A/D converters. Each CONVERT pulse causes a single A/D conversion to occur.

Table 8 : Hardware Addresses: Computer I/O[2]

[2] Notes for Table 8: The lower three bits of Valve Open, Refrigerator On, and Dewar Heater On are latched in a D-latch, rather than through an address decoder. To activate these functions, set the desired lower bits PA2 to PA0, then set DIO2 to DIO0, and finally strobe DIO3. This also means that other sequences (e.g. 011) for this base address are not available for other functions.

Address							Function	Chassis	Comments
DIO3	DIO2	DIO1	DIO0	PA2	PA1	PA0			
Address enable. Set DIO0 to DIO2 and PA0 to PA2 to desired address, then strobe DIO3 to latch the address.	0	0	0	0	1	1	Dewar Load Heater On	Coax SW Control	Turns on the 4K hot load in the Dewar
	0	0	1	1	0	0	Noise Source On	Coax SW Control	Turns on the noise source
	0	0	1	1	0	1	Noise inject to coupler (Mixer 1 rack only)	Coax SW Control	Selects the coupler path to inject noise into the input of the IF amplifier (Mixer 1 rack only). Isolated via opto-isolator.
	0	0	1	1	1	0	Noise inject to circulator (Mixer 1 rack only)	Coax SW Control	Selects the circulator path to inject noise into the IF port of the mixer (Mixer 1 rack only). Isolated via opto-isolator.
	0	0	1	1	1	1	Spare	Coax SW Control	Spare control function available from power supply PCB. Isolated via opto-isolator.
	0	1	0	0	0	0	Analog Mux Select	Computer I/O	Selects 1 of eight differential channels that are input to the National Instruments board on lines Ain+7 and Ain-7 via connector "D" on the rear of the chassis.
	0	1	0		…		Analog Mux Select	Computer I/O	-- " --
	0	1	0	1	1	1	Analog Mux Select	Computer I/O	-- " --
	0	1	1	X	X	1	Valve Open	Computer I/O	Controls vacuum valve between vacuum pump and Dewar. This is fail-safe design: Setting this bit opens the valve.
	0	1	1	X	1	X	Refrigerator On	Computer I/O	Turns on the Dewar refrigerator
	0	1	1	1	X	X	Dewar Heater On	Computer I/O	Activates the Dewar heater for warm-up.
	1	0	0	0	0	1	Spare		
	1	0	0	0	1	0	Spare		
	1	0	0	0	1	1	Spare		
	1	0	0	1	0	0	Spare		
	1	0	0	1	0	1	Spare		
	1	0	0	1	1	0	Spare		
	1	0	1	0	0	0	SW1 Set	Coax SW Control	Controls the coaxial IF switches in the Dewar
	1	0	1	0	0	1	SW1 Reset	Coax SW Control	-- " --
	1	0	1	0	1	0	SW2 Set	Coax SW Control	-- " --
	1	0	1	0	1	1	SW2 Reset	Coax SW Control	-- " --
	1	0	1	1	0	0	SW3 Set	Coax SW Control	-- " --
	1	0	1	1	0	1	SW3 Reset	Coax SW Control	-- " --
	1	0	1	1	1	0	SW4 Set	Coax SW Control	-- " --
	1	0	1	1	1	1	SW4 Reset	Coax SW Control	-- " --

Table 9: Hardware Addresses: Warm IF Control

Address				Data												Function	Chassis	Comments
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0			
Address enable.	0	0	0										0	0	0	Spare	IF Controller	
	0	0	0										0	0	1	3-dB Pad In	IF Controller	
	0	0	0										0	1	0	3-dB Pad Out	IF Controller	
	0	0	0										0	1	1	Internal Load In	IF Controller	For zeroing power head
	0	0	0										1	0	0	Internal Load Out	IF Controller	For zeroing power head
	0	0	0										1	0	1	Output - Pwr Meter	IF Controller	Output switched to power meter
	0	0	0										1	1	0	Output - Sq Law	IF Controller	Output switched to square law detector.
	0	0	0										1	1	1	Spare	IF Controller	
	0	0	1							0	0	0	0	0	0	Attenuator Control	IF Controller	0 dB
	0	0	1							0	0	0	0	0	1	Attenuator Control	IF Controller	1 dB
	0	0	1							…						Attenuator Control	IF Controller	N dB
	0	0	1							1	1	1	1	1	1	Attenuator Control	IF Controller	MAX dB
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	YIG Filter Freq	IF Controller	1 GHz
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	YIG Filter Freq	IF Controller	1.001 GHz
	0	1	0	…												YIG Filter Freq	IF Controller	N GHz
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	YIG Filter Freq	IF Controller	12.0 GHz

Additional hardware addresses were required for the new Mixer Bias Supply, and these are identified in Table 10. The output-only data lines PB5, PB4, and PB3 are used as the base address. As in the DIO base addressing, there are three base address lines (PB5, PB4, and PB3) and three secondary address lines (PB2, PB1, AND PB0) . The fourth base address line, PB6, is used to latch the addresses. Line PB7 is a spare.

NO – THESE NOW OVERLAP WITH THE IF CONTROLLER.

Table 10: Hardware Addresses (PB5-PB3 Lines for base address)

Address		Function	Chassis	Comments
Primary (PB6,[3] PB5, PB4, PB3)	Secondary (PB2, PB1, PB0)			
000				Not used.
001	000	Open Loop	Mixer Bias Supply	
001	001	Closed Loop	Mixer Bias Supply	
001	010	Output State: Run	Mixer Bias Supply	
001	011	Output State: Zero	Mixer Bias Supply	
001	100	Output State: Ground	Mixer Bias Supply	
001	101	Bias source: Internal	Mixer Bias Supply	
001	110	Bias source: External	Mixer Bias Supply	
001	111	Ganged outputs: None	Mixer Bias Supply	
010	000	Ganged outputs: Paired	Mixer Bias Supply	
010	001	Ganged outputs: All	Mixer Bias Supply	

[3] Line PB6 is used for address enable. That is, bits PB0 to PB2 and PB3 to PB5 should be set to the desired address, then strobe PB6 to latch the address.