U.S. DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

MASSPEC: A PC program to control and to process data from an automated mass-spectrometer

by

Robert C. Bigelow, Robert B. Vaughn, and Stanley E. Church[1]

Open-File Report 99-161

[1]Denver, Colorado

MASSPEC: A PC program to control and to process data from an automated mass-spectrometer

by Robert C. Bigelow, Robert B. Vaughn, and Stanley E. Church

I. INTRODUCTION:

The Geological Survey has used a single-filament, solid-source Mass Spectrometer (MS) for research on isotopic mass ratios in earth materials. The National Bureau of Standards (NBS) designed and built the spectrometer as a single-collector instrument that emphasizes measurement precision for scientific research rather than high-speed data acquisition. A custom-programmed personal computer controls the magnetic field of the spectrometer and acquires relative isotopic abundance data from the MS. A program, called MASSPEC, automates the collection of isotopic data and computes isotopic ratios subject to operator-specified statistical parameters.

The spectrometer came equipped with a Hewlett-Packard HP85 computer/controller that served adequately for many years, but is now obsolete. NBS supplied an HP85 program in a dialect of HP BASIC that one of us, (SC), modified to improve its ability to filter obviously erroneous data, but it was still necessary to import the data into a spreadsheet and reduce it manually to remove the effects of noise and sample instability. Since an MS-DOS Personal Computer (PC) was already employed to run the spreadsheet, we decided to replace the HP85 with a PC that was programmed to run the MS. The objectives of the MASSPEC program were to replace the functionality of the HP85 controller and to provide a higher degree of data reduction and more flexible control over measurement parameters than was previously possible. Some HP-85 BASIC program concepts were retained but most of the program could not be translated for the PC since it used HP-85 concepts and system features that are difficult to duplicate on a PC. This report describes the MASSPEC program.

II. MASS SPECTROMETER SYSTEM OVERVIEW:

A small amount of sample material is placed on a filament which is then mounted in the source of the MS. Heating the filament drives off sample ions which are accelerated through a fixed voltage and magnetically deflected to generate an ion current in a Faraday cup[1]. Adjusting the magnetic field of the MS tunes the machine to measure ions of a given atomic mass and charge, usually a singly ionized isotope. A programmable[2] Varian FR-41 Gaussmeter measures and controls the magnetic field. The ion collector current develops a measurable voltage across a known resistance (10^9 ,10^{10}, or 10^{11} ohms) at the input of an NBS-designed electrometer. An HP 3456A Digital Voltmeter (DVM) integrates the output voltage from the electrometer and converts it to digital form with 6 ½ digit resolution.

The entire system can be controlled and run manually; however, four devices, the Gaussmeter, the DVM, the electrometer[3] and a printer also can be controlled through GPIB (IEEE-488) interfaces. The PC communicates with those instruments via a National Instruments GPIB-PCIIA board, an IEEE-488 I/O interface. The block diagram (Fig 1) of the MS control system diagrams the relationships among these automated parts of the system. The computer sends GPIB commands to the Gaussmeter to change the magnetic field and then waits a predetermined amount of time to allow the magnetic field to stabilize before taking readings from the DVM. The electrometer converts the ion signal from the MS to a voltage in the range 0 to 10 Volts. The DVM determines the voltage from the electrometer by averaging over a period of either 10 or 100 power line cycles (60 Hz. in N. America, 50 Hz. in Europe) and broadcasts its reading over the GPIB interface. The advantage of this technique is that the timing accuracy for the readings is very precise; However, the computer must re-synchronize

with the DVM each time it accepts readings from it. After a full suite of readings is recorded by the PC, the results are stored on disk and transmitted to a printer.

.

III. MEASUREMENT DESIGN:

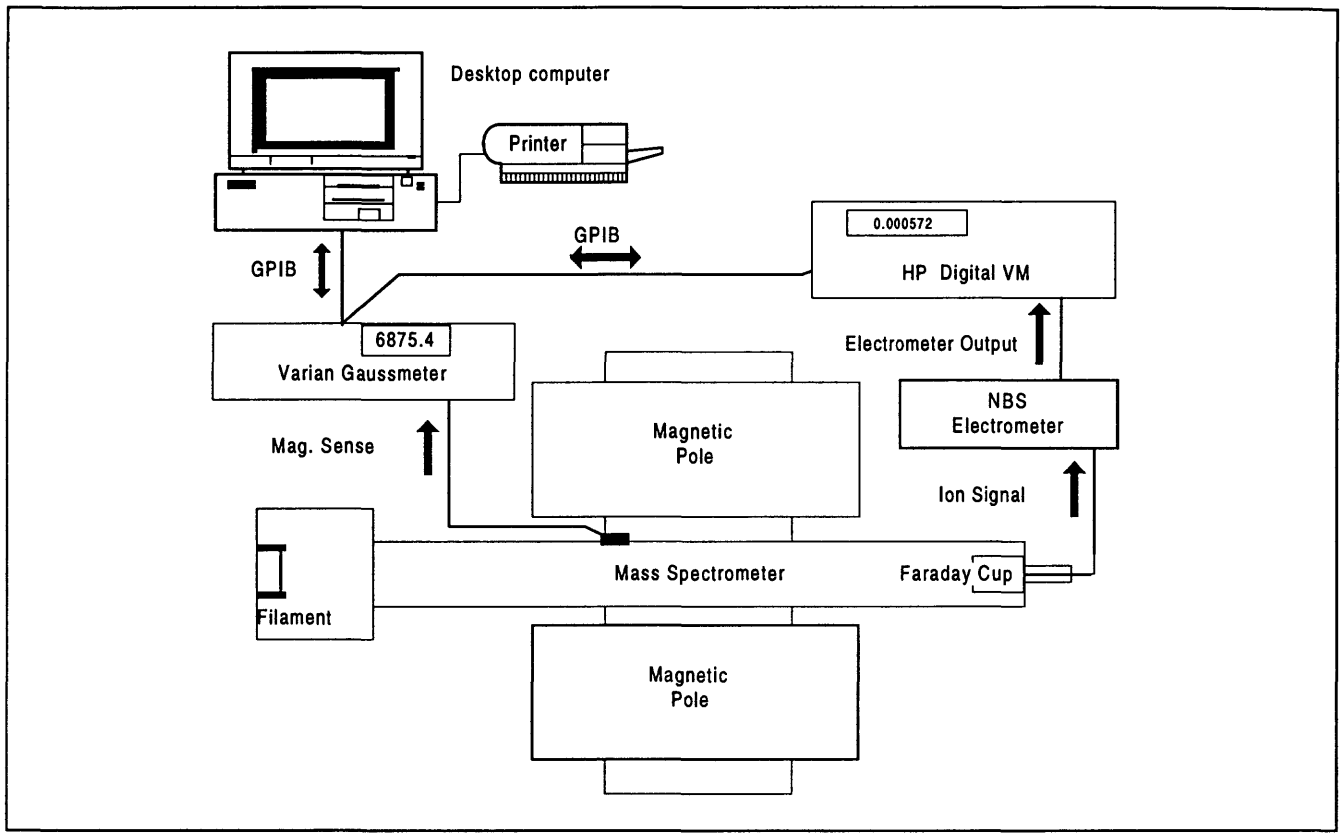

Figure1: Mass Spectrometer Layout

The objective is to determine the ratios of isotopes in the samples with as high a degree of precision as can be achieved. Large numbers of readings are taken and then strenuously filtered to remove questionable readings. The philosophy behind the MASSPEC program is that it is better to drop a good (if doubtful) reading than to risk including bad ones in the computations. However, all readings are preserved so that the investigator can override the program's judgement if he so chooses.

In an ideal MS, all isotopes would be measured simultaneously; however, that is not possible with our present instrument. Instead, the magnetic field is tuned to measure each isotope in turn, and that sequence of measurements is repeated for a predetermined number of times. The data must be interpolated to infer simultaneous data values. Systematic noise, emission irregularities and dropping signal levels are all limiting factors of accuracy in these measurements. The heated filament drives off ionized sample material and the rate of ion production is roughly proportional to the remaining amount of sample. Hence, the signal strength (ions per second) usually decreases gradually over time as the

sample depletes. Sometimes abrupt shifts in signal levels are observed that are attributed to sputtering, changes of sample distribution on the filament and other irregularities. Much of the MASSPEC program is specifically designed to reject spurious or questionable data caused by those irregularities.

IV. MEASUREMENT DETAILS:

As illustrated in Figure 2, peak signals from each isotopic species appear at well defined magnetic field settings and do not overlap. Both the present and previous programs determine background by averaging measurements at a fixed offset (points b, Figure 2) to each side of the isotope peaks. This technique works very well when isotope peaks are cleanly separated with no overlap or interference because the background changes very slowly with magnetic field. MASSPEC also permits background points to be set individually in Gauss units near each isotopic peak and can determine background from a single background point per isotope if desired.

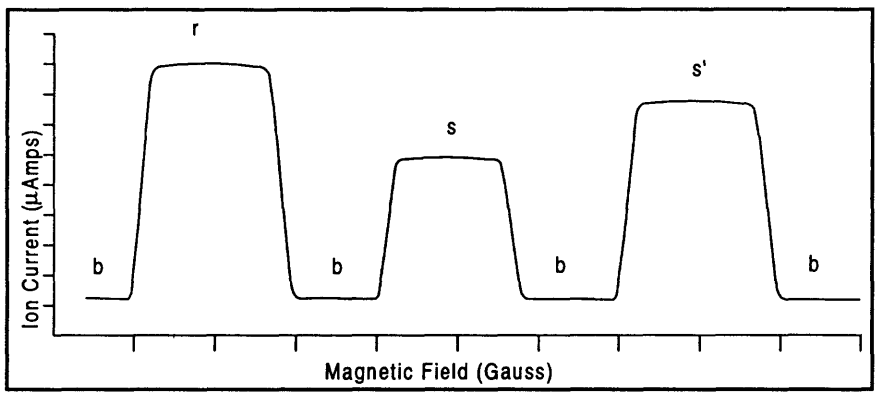

Figure 2: Ion current as a function of magnetic field. Points "r", "s" and "s' " are reference and two sample isotopic peaks, respectively. Background is measured near points "b" on at least one side of each isotopic peak.

A typical determination of isotopic ratios consists of a repetitive series of measurements on each isotope. The fundam-ntal building block of these measurements is the **set,** a fixed number (usually 10) of measure-ents over a period of several seconds at a single magnetic field setting (figure 3). A **good set** is one that has at least a predefined minimum number of valid data points as determined by filtering, (described below). Conversely, a **bad set**, is one that has too few valid data points after filtration. After a set is filtered, the mean and standard deviation of the set are stored, as are the number of valid data points remaining in the set. Individual data measurements comprising a set

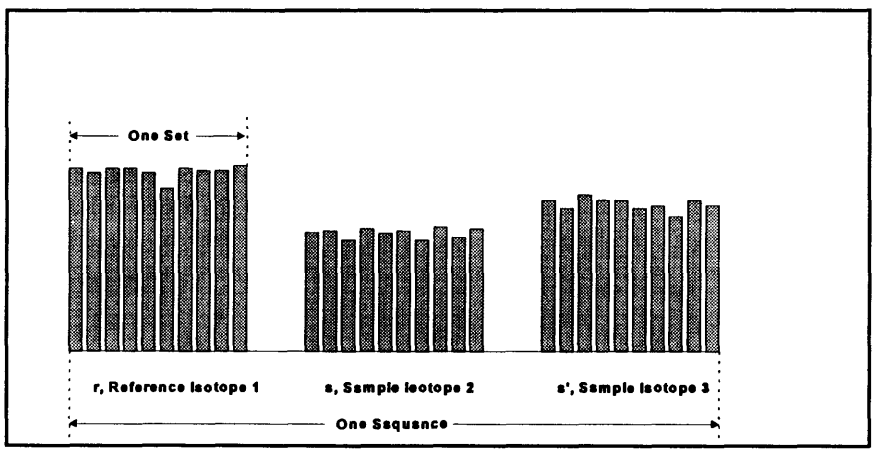

Figure 3: A set is a series of measurements of one isotope, a sequence is one set of each isotope measured seriatim, beginning with the reference isotope. Only the mean and standard deviation of each set is recorded, subject to filtering

are not retained. Measurements proceed by tuning the mass spectrometer to one isotope after another and making a set of measurements at each tuning. One isotopic species is designated as a reference (r) so the other isotopes (s, s' ...) may be ratioed against it. A **sequence** begins with a reference isotope set, followed by one set of each of the other isotopes (Figure 3). This procedure is repeated until a predetermined number of good sequences (ie containing no bad sets) have been measured. An additional constraint is that the good sequences must all be within "groups". As shown in Figure 4, a **group** consists of an uninterrupted number of good sequences plus a following reference set. The additional reference set is required to allow interpolation when computing ratios. A bad set (e.g.: set s_7 in Figure 4) forces the program to start a new group beginning with a new reference set. Since the number of groups is a measure of the number of bad sets, the program aborts if the group number gets too large.

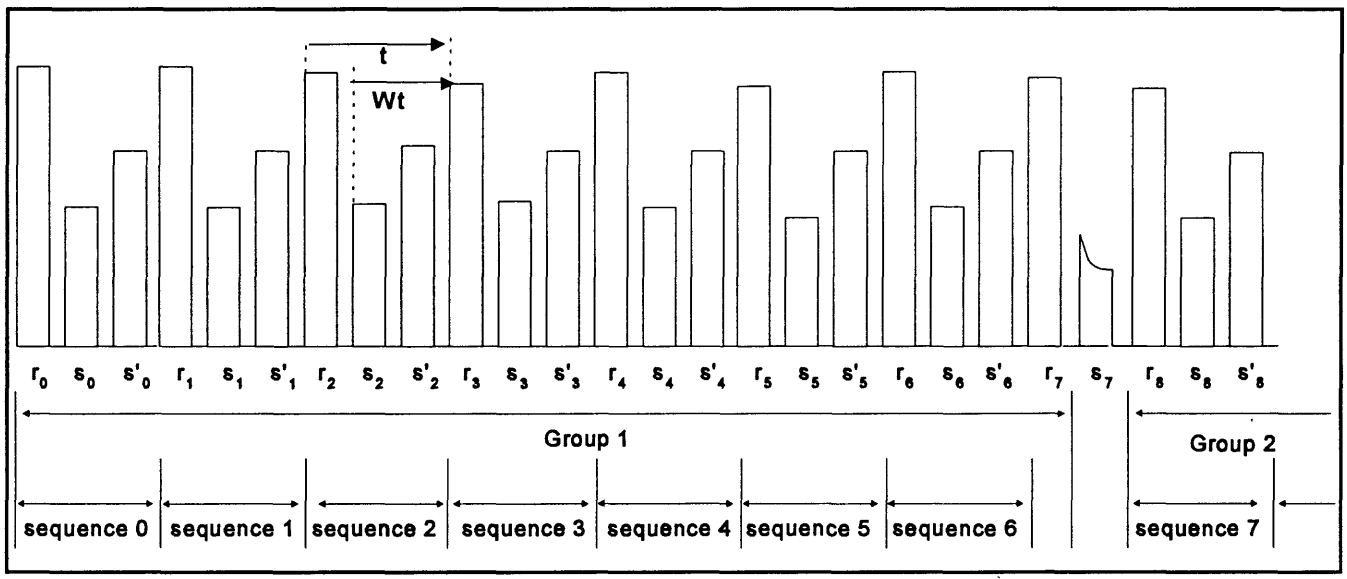

Figure 4: Plot illustrating the relationship between sequences and groups. Each bar represents one set. The weight, W is a fraction of the period, t, between the beginnings of successive sequences and is used to compute interpolations. The weight shown is for the first sample ion; the second (and succeeding) ions each have their own weights. Note that the bad set, s_7, does not belong to a sequence or group

Both MASSPEC and the HP85 program filter out defective measurements by comparing each measurement to the mean, and rejecting those that differ by more than a certain multiple of the Standard Deviation (SD). This is referred to as Rule 1 in Figure 5. Rule 1 is sufficient if most of the measurements are well behaved with only a few outliers (bad data points) and the signals are well above the noise. However, if the data are quite noisy, the standard deviation itself may be too large to be much help in determining what is or is not acceptable data To eliminate the effects of very noisy data, especially at small signal levels, MASSPEC also compares the measured Relative Standard Deviation (RSD) to a predefined limit. If the RSD is larger than the limiting value, the measured SD is replaced for filtering purposes by the limiting RSD multiplied by the measured mean value and the data comparison is made as before. The advantage to this scheme, called Rule 2, is that sets with very noisy or abruptly changing measurements are detected, filtered, and rejected if necessary. Figure 5 diagrams the effects of both data filtering rules in terms of the SD and RSD. After filtration, MASSPEC

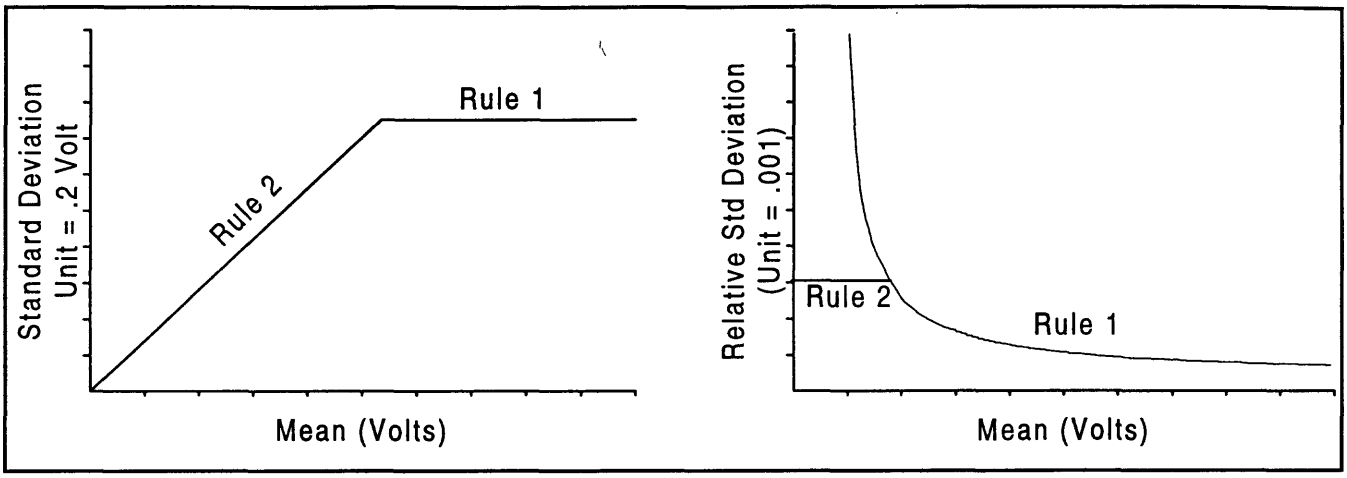

Figure 5: The effect of two rules limiting the allowable values of the variance of the means. Rule 1 limits the variance at large values of means to fixed multiples of SD. At small mean values, Rule 2 limits the allowable variance to no more than a fixed RSD

re-computes the mean average of the remaining data and reports the recalculated mean, the number of valid data points, and the standard deviation. The same filtering system is used at several levels in MASSPEC. It is applied to background sets, isotopic sets, and to ratios of sets. The desired output from MASSPEC is the computed ratio of each sample (ie: non-reference) isotope to the reference isotope along with the relevant statistics. Along with the rest of the data, MASSPEC records the time each set is measured . This makes it possible, when forming ratios, to correct for the fact that sample and reference isotopes are measured at slightly different times. We assume linear interpolation is sufficient to provide the correction. The usual procedure is to interpolate between reference isotope sets with time weighting (W, in figure 4) to derive a "reference" at the time the sample isotope set was measured. Optionally, time weighted interpolation can be applied to sample isotope sets and ratios formed with an intervening reference set; this is referred to in the program listing as an "inverse sense" ratio. For each sample isotope to reference isotope pair, the mean of all ratios is formed and each ratio is compared to it. Rules 1 and 2 are applied and outlier ratios are thrown out. The mean ratio is re-computed from the remaining ratios along with its standard deviation and reported.

It is possible to do many different experiments using MASSPEC without modifying the program itself. Experiments are defined for MASSPEC by entries in an auxiliary file called ISOPARAM.REC which contains the parameters for each experiment. The parameters include the gauss settings for each isotope peak; their associated background settings; and the filter parameters for each stage of measurement (See Appendix A). ISOPARAM.REC is an ASCII file; any parameter may be changed and new experiments may be added (up to a limit of 20) by editing the file with a text editor.

V. EXAMPLE:

As an example of how the parameters affect the program, consider the excerpt from a typical ISOPARAM.REC file listed in Appendix A. For convenience, Table 1 reproduces part of ISOPARAM.REC. Suppose we want to measure ^{206}Pb, ^{207}Pb and ^{208}Pb using ^{206}Pb as the reference.

The initial entries for those three lead isotopes define the gauss settings for each and where the background measurements are made. MASSPEC takes "NumReads" (10) readings for each set and computes the mean (M) and standard deviation (SD) for each isotope. The program invokes Rule 2 by comparing the SD to a "limiting" SD formed by multiplying IsoMaxDev by M and uses the lesser of those quantities when applying Rule 1. Any reading whose absolute difference from the mean exceeds IsoSigmaTol (1.5) times the SD violates Rule 1 and is rejected (again, see Figure 5).

```
206       ' Iso(1).Mass    INTEGER     Mass Units  Iso(1)=reference isotope
6703.0    ' Iso(1).Gauss   SINGLE      Gaussmeter setting for isotope peak
1.0       ' Iso(1).Range   SINGLE      Voltage range for Chart Recorder
... (similar entries for isotopes 2 and 3)...
10        ' NumReads INTEGER           Readings per set for each isotope
7         ' IsoGoodReads INTEGER       Min number of readings in a Good Isotope Set
.003      ' IsoMaxDev SINGLE           Maximum allowed Rel. Std. Deviation for an
                                         Isotope set (Rule 2)
1.5     ' IsoSigmaTol SINGLE           Tolerance: Multiplies Std. Deviation to form
                                         limit for outliers (Rule 1)
```

Table 1: A partial, somewhat reformatted list from Appendix A ISOPARAM.REC of parameters for one isotope.

Once outlier data points have been eliminated, MASSPEC checks to be sure that IsoGoodReads (7) good readings remain to form a valid measurement and if so recalculates the mean, M, with those data points. While it would be possible to continue to re-compute SDS, throw out additional points and compute a new means until they converged, our experience with this program and its predecessor indicates that one cycle of corrections is adequate to yield good statistics when the data are good. Most of the remaining parameters apply to Baselines and Ratios and are applied in the same way. as the Isotope parameters. The Delay parameter specifies the time for the magnetic field to settle whenever the MS is re-tuned; usually before beginning each set of measurements.

VI. PROGRAM OVERVIEW:

The MASSPEC program is written in Microsoft Quick BASIC version 4.5. Initially, we had hoped to translate the NBS HP85 BASIC program as a basis for MASSPEC, but the required changes were too great for that to be a productive approach. Nevertheless, many of the concepts of the NBS program were retained and incorporated in MASSPEC. The MASSPEC Program listing is in Appendix B (q.v.). The program is available on diskette.

The main part of the program is divided into 10 sections with 12 subprograms and functions.

A. The Declaration section has a $INCLUDE 'QBDECL.BAS' statement that invokes the library for National Instruments IEEE488 PC card. This statement is required so that the program can invoke functions supplied by National Instruments. In the setup phase, the program initializes variables and types. In the 'Open Files' section, the program enters a large loop at the 'SetupFiles:' label that terminates in a 'SELECT CASE' statement near the end of the main program. The program asks for a sample name which becomes the name for 3 files; the backup to the report file <Samplename>.Bnn, a log file

<Samplename>.LOG and a <Samplename>.Rnn file that is intended to be used with a spreadsheet, although this has not been tested. The LOG file is intended to record everything that happens in case the program aborts or for some other reason is unable to finish. The .Rnn file is written to a floppy disk so that it will be an automatic backup. If the same sample is re-measured, the logfile is appended and the other two files are created with incrementing extensions (Rnn, Bnn) so that data is not overwritten. This creates a lot of housekeeping but it means that, in general, data is not lost. After several optional questions about measurement parameters and documentation, the operator is given a chance to revise his entries and the program enters the "GET Parameters' section.

B. The ISOPARAMS.REC file is scanned twice. The first time, every experiment definition header is read in and served up in a menu. Once the operator chooses the desired definition the program re-reads ISOPARAMS.REC and imports the chosen parameters from it.

C. In the Change Parameters section, two menu options allow some of the parameters to be overridden. The RANGE question scales the output to the chart recorder but does not affect the data seen by the computer. The NUMBER OF CYCLES refers to the integration time of the DVM. 100 is the usual period and should give the highest accuracy but the measurements complete much faster at 10 cycle integration times. Sometimes the sample on the filament depletes quickly or the data are extremely regular and it is necessary or desirable to integrate for only 10 line cycles. The program now goes fully automatic and does not require further operator intervention until the end.

D. The Baseline is measured at one point or preferably two bracketing points near the isotopic peak. This determines systematic errors, particularly the positive offset (usually about 500 mV) that is deliberately introduced to avoid non-linearities in the MS system electronics. The program will terminate if there are more than BaseMaxBadSets, because that would indicate an extraordinarily unstable system. Good baseline information is summarized to the screen.

E. As the program moves into the measurement phase several variables are set up that keep their meanings throughout the program. The basic measurement is a set of readings that are averaged and checked for bad points as explained previously. The variable I indicates the isotope. The first isotope (I=1) is always the reference isotope. The variable J counts how many usable sets of measurement have been collected irrespective of which isotope. The variable N counts how many complete cycles through sequence of I values have taken place. N is the sequence number which indicates how much data will be available to compute ratios later. Sequences always start with a reference set. The group number, G, groups valid sequences together and indicates the number of restarts because of defective data as previously explained. If there are rules violations, the group number increments and if it becomes too large the program prints an error message and quits. Some debug statements, mostly PRINT USING, have been left in the program. They can be invoked by uncommenting them.

F. Once sufficient good data (as defined by N) has been collected the program enters the

computation stage, Analyze Ratios. First, reference sets are interpolated then ratios are formed between the sample and reference data. In the normal ratio procedure, bracketing reference data are interpolated to derive a value that represents the reference isotope signal at the time a sample isotope is measured. In the optional "reverse sense" ratioing method, bracketing sample data are interpolated to infer values at the time a reference signal is measured. Once all the ratios are formed, they are averaged and checked against their standard deviation in the same way that sets were. Ratios have their own limiting parameters in ISOPARAMS.REC which often are tighter than the parameters for the individual isotopes.

G. Once the computations are finished, the data are written out to the ASCII delimited <Samplename>.Rnn file for possible use in a spreadsheet, are printed out, and the <Samplename>.Bnn files are written out. Both files are closed. The Data are printed to the screen and to the printer in the Report section, using routines from the Display GoSubs section that follows the Loop section

H. In the Loop Logic section of the main program, the operator is given choices as to whether to repeat the measurement exactly, change sample IDs only and keep the parameters, change the parameters for the same sample, change everything or quit. Based on his decision, the program branches as indicated in the CASE statement. If the measurement is partly or fully repeated, some parameters are retained to simplify the operators task.

I. The Display Gosubs and Error Messages sections follow the main body of code. The Display GOSUBS generate formatted output to the screen, the printer, and the Log file.

III. SUBPROGRAMS AND FUNCTIONS

There are both logical and hardware-specific subprograms for the voltmeter and for the gaussmeter. This arrangement of subprograms has two major advantageous. First, it insures that a minimum of reprogramming will need to be done should a different IEEE PC interface board, Digital VoltMeter (DVM) or Gaussmeter be substituted for the present units in the future. Second, it allowed us to substitute dummy routines for the IEEE-specific routines and test the program on a separate PC, which avoided tying up the MS system. The Measure Subprogram performs statistical calculations on sets of measurements whereas the GetDVM subroutine has the IEEE-specific calls for reading the DVM. Likewise the Move Subprogram has the logic and the time delays for the Gaussmeter but the OutputGauss Subprogram has the IEEE specific calls. The Move Subprogram produces a rotating or whirling line display that is intended to reassure the operator that the system really is functioning while it moves to a new Gauss setting; however, it does not work well with some PCs. OutputGauss has an algorithm for converting 5 digit Gaussmeter settings to the binary coded decimal form required by the Gaussmeter along with the range data coded in for the electrometer. The forms of GetDVM and OutputGauss were based in large part on examples in National Instruments documentation. The conversion of data to Binary Coded Decimal plus Range control follows NBS documentation supplied with the electrometer and the NBS-modified Varian Gaussmeter. The Range data is sent to the Gaussmeter which then passes it to the electrometer.

The Menu subprogram provides a moderately flexible means of presenting up to 20 multiple choice menu items in a vertical format. Keystrokes on the grey vertical arrow keys (or on their numeric

keypad equivalents with the "Num Lock" key off) highlight the selections; hitting "Enter" selects the choice. The menu choices and their consequences are in arrays which are usually at the beginning of the program. The Interpol8 Subprogram defines the interpolation routines. The weighting function is DelT (for Delta-T), functionally the same as W, above. Ratio Statistics Subprogram actually computes the ratio statistics. It is possible to have near cancellation in the SigSq# variable and it can even compute to a negative. If it is less than 10^{-15} , it taken as zero. There are several very small Functions or Subprograms that are mostly self-explanatory. The Getxxx functions get different variable types from ISOPARAM.REC

VIII. ENDNOTES AND REFERENCES

1. The theory and operation of mass spectrometers is covered in, for example, Hamilton, E.I. and Aherns, L.H.,1965, Applied Geochemistry: London and New York, Academic Press, Solid sources are discussed on pp 17-23. Lead isotopes measurements are discussed in Chapter 9, pp 167-203

2. The Varian Gaussmeter and the electrometer were modified to be controlled over the HPIB bus at by Schiedler at NBS (unpublished report). The NBS report refers to the electrometer as a "parametric amplifier"(PA). Our present instrument has newer electronics but otherwise is similar to one described in Shields, W.R., ed., 1966, NBS Technical Note 277; Analytical Mass Spectrometry Section: Instrumentation and Procedures for Isotopic analysis, 25 July 1966: [Washington, D. C.], National Bureau of Standards.

3. The electrometer provides a fixed range output to drive the DVM and a range-switched output to drive a chart recorder. The range settings are derived from GPIB signals sent to the Gaussmeter. Since the output to the DVM is fixed, range settings should not affect the data seen by the PC. The NBS program indicates such an effect may exist; if so, it has been too small for us to observe and we have ignored it.

APPENDIX A: ISOPARAM.REC

Each ISOPARAM.REC file must start with a file date header as shown. MASSPEC checks the revision information in ISOPARAM.REC to insure that it has the correct format and sequences of parameters. Whenever the form of ISOPARAM.REC changes, a new header must be defined and code in MASSPEC changed. This insures that incompatible ISOPARAM.REC and MASSPEC versions cannot be used together. The parameters are listed in groups that define a measurement sequence in MASSPEC. Each parameter appears on its own line and is contained in the first 12 characters of the line; the remainder of the line is comment. All entries in a group of parameters must be defined. Each experiment definition starts with a header, such as "PB206-208", that identifies it as a menu choice in the program. Definitions must be separated from each other by at least one blank line.

The header is followed by the parameters for each isotope. For each isotope, the mass in AMU is followed by the Gauss setting for the isotopic peak and the Gauss settings at which the Baseline(s) near it will be measured. A negative BaseGauss2 reading indicates it is not taken. After each isotope is defined, definitions similar to Table 1 appear for the Isotopes, the Baselines, and the Ratios. The last two entries are the maximum allowable number of bad baseline sets and the time delay between sets to let the electrometer (PA) settle the magnetic field settle after it is retuned. If more than 3 (typically) baseline readings in a row are noisy or bad, the measurement should be aborted. The optimum time delay is determined by trial and error as a compromise between affecting the first readings in a set and slowing the measurements so much that drift becomes unacceptable

Sample ISOPARAM.REC file

```
              ' Last revised 11 Mar 93 for Rev 6.7
              ' The first non-blank line must be the revision line (11Mar93)
              ' add BaseGauss1 & 2; remove GaussOffset from records (11Mar93)
              ' This file holds parameters for the Masspec program
              ' up to 20 different parameter sets can be defined
              ' Sets must be separated by at least one line that is blank
              ' in the first 12 characters (these lines qualify).
              ' Pb 206, 207, 208 default sequence
PB 206-208    ' IsoName  STRING * 10     Name of Sequence of Isotopes
3             ' NumIsos  INTEGER         No. of Isotopes
206           ' Iso(1).Mass   INTEGER    Mass Units  Iso(1)=reference isotope
6703.0        ' Iso(1).Gauss  SINGLE     Gaussmeter setting for isotope peak
1.0           ' Iso(1).Range  SINGLE     Voltage range for Chart Recorder
6711.0        ' BaseGauss1  SINGLE       Baseline Gauss; 1 required
6695.0        ' BaseGauss2  SINGLE        2nd optional; ignore if 0 or negative.
207           ' Iso(2).Mass   INTEGER    Mass Units
6719.3        ' Iso(2).Gauss  SINGLE     Gauss
1.0           ' Iso(2).Range  SINGLE     Volts
6711.3        ' BaseGauss1  SINGLE       Baseline Gauss reading, one required
6727.3        ' BaseGauss2  SINGLE        2nd optional; ignore if 0 or negative.
208           ' Iso(3).Mass   INTEGER    Mass Unit
6736.0        ' Iso(3).Gauss  SINGLE     Gauss
3.0           ' Iso(3).Range  SINGLE     Volts
6728.0        ' BaseGauss1  SINGLE       Baseline Gauss reading, one required
6744.0        ' BaseGauss2  SINGLE        2nd optional; ignore if 0 or negative.
```

```
10              ' NumReads INTEGER          Isotope; No. of readings per set
7               ' IsoGoodReads INTEGER      Min No of reads in a Good Isotope Set
10              ' BaseNumReads INTEGER      No. of sets read for each isotope
7               ' BaseGoodReads INTEGER     Min no.reads in Good Baseline set
10              ' NumRatios INTEGER         Minimum number of valid ratios
.003            ' RatioMaxDev SINGLE        Max.Rel.Std.Dev for Ratios
1.5             ' RatioSigmaTol SINGLE      Tolerance * Ratio Std.Dev=Reject
.003            ' IsoMaxDev SINGLE          Max.Rel.Std.Dev for isotopes
1.5             ' IsoSigmaTol SINGLE        Tolerance * Std.Dev.= Reject level
.003            ' BaseMaxDev SINGLE         Max Rel.Std.Dev. for BaseLines
1.5             ' BaseSigmaTol SINGLE       Tolerance * Base.Std.Dev.=Rej.Level
3               ' BaseMaxBadSets INTEGER    Maximum allowed bad baseline sets
2               ' Delay INTEGER             Min. time to let PA settle
```

Mandatory: at least one blank line between definitions

```
                ' Pb 204, 206 default sequence
PB 204-206      ' IsoName  STRING * 10      Name of Sequence of Isotopes
2               ' NumIsos  INTEGER          No. of Isotopes
206             ' Iso(1).Mass   INTEGER     MU  Iso(1) is reference isotope
6703.0          ' Iso(1).Gauss  SINGLE      Gaussmeter reading for isotope
1               ' Iso(1).Range  SINGLE      Volts
6711.0          ' BaseGauss1  SINGLE        Baseline Gauss reading, one required
6695.0          ' BaseGauss2  SINGLE         2nd optional; ignore if 0 or negative.
204             ' Iso(2).Mass   INTEGER     Mass Units
6670.0          ' Iso(2).Gauss  SINGLE      Gauss
0.1             ' Iso(2).Range  SINGLE      Volts
6662.0          ' BaseGauss1  SINGLE        Baseline Gauss reading, one required
6678.0          ' BaseGauss2  SINGLE         2nd optional; ignore if 0 or negative.
10              ' NumReads INTEGER           readings per set per isotope
7               ' IsoGoodReads INTEGER       Min reads in a Good Isotope Set
10              ' BaseNumReads INTEGER       No. of sets read for each isotope
7               ' BaseGoodReads INTEGER      Min no.reads in Good Baseline set
10              ' NumRatios INTEGER          Minimum number of valid ratios
.003            ' RatioMaxDev SINGLE         Max.Rel.Std.Dev for Ratios
1.5             ' RatioSigmaTol SINGLE       Tolerance * Ratio Std.Dev=Reject
.003            ' IsoMaxDev SINGLE           Max.Rel.Std.Dev for isotopes
1.5             ' IsoSigmaTol SINGLE         Tolerance * Std.Dev.= Reject level
.003            ' BaseMaxDev SINGLE          Max Rel.Std.Dev. for BaseLines
1.5             ' BaseSigmaTol SINGLE        Tolerance * Base.Std.Dev.=Rej.Level
3               ' BaseMaxBadSets INTEGER     Maximum allowed bad baseline sets
2               ' Delay INTEGER              Min. time to let PA settle
```

up to 18 more definitions may be included; formats must conform to foregoing examples

APPENDIX B: PROGAM 'MASSPEC' LISTING

Note: a '\' at the end of a line indicates continuation onto the next line for listing purposes. Continuation lines are not supported by Quick Basic 4.5. To compile, use IEEEv9x5.BAS or edit this program to remove continuation lines and this heading. Required files and programs for compilation are: Microsoft Quick Basic 4.5, QBDECL.BAS from National Instruments Inc. ,IEEEv9x5, and ISOPARAM.REC. The PC must have a National Instruments IEEE interface installed that is supported by QBDECL.BAS. Possibly useful Debug statements have been left in place but are commented out by default.

```
' Program MASSPEC   R.C.Bigelow 23Mar99 Additional Comments
' Last major revision is IEEEv9x5.BAS 8Mar96
' Make Exe from the compiler menu will not recognize the need for V or X compiler
' options so compile will fail. Run compiler from command line with options:
'     C:\QB45> BC IEEEv9x5.BAS /E/O/V/C:512/X
DEFINT A-Z
' $INCLUDE: 'QBDECL.BAS'       'National Instruments IEEE488 software
' $DYNAMIC
DECLARE FUNCTION GetAns$ (Prompt$, Default$)
DECLARE FUNCTION GetStr12$ ()
DECLARE FUNCTION GetSingle! ()
DECLARE FUNCTION GetInt% ()
DECLARE FUNCTION Interpol8# (M() AS ANY, SampIndex%, RefIndex%, MeasIndex%)
DECLARE FUNCTION MaxOf# (A#, B#)
DECLARE FUNCTION YNans% (message$)
DECLARE SUB GetInfo (OpName$, FilAmps$, FilVolts$, SubmitName$, Comment$)
DECLARE SUB GetSampName (SampName$)
DECLARE SUB MoveTo (Gauss!, Range%, Delay)
DECLARE SUB Measure (SigmaTol!, MaxStDev!, Reads, Mean#, Sigma#, Time!, GoodReads)
DECLARE SUB RatioStats (Ratio() AS DOUBLE, Results() AS ANY, I%, J%, T%)
DECLARE SUB ReadDVM (VmRead#)
DECLARE SUB Statistics (VALUE AS DOUBLE, Sum AS DOUBLE, SqSum AS DOUBLE, Number%)
DECLARE SUB OutputGauss (Gauss!, Range%)
DECLARE SUB VertMenu (Header$, Menu$(), Size, RowNum, Choice)
'-----------------------------------------------------------------
'Comments
' Program IEEE.BAS gathers data from an NBS designed Mass spectrometer.
' Program controls a Varian Gaussmeter (Gauss), an NBS designed
' Parametric Amplifier (PA) attached to the Gaussmeter, and
' a Hewlett Packard High Model 3456A Digital Voltmeter (DVM) via a
' National Instruments IEEE-488 controller in an IBM-type 386 PC.
' DEFINITIONS
' Sequence: an ordered list of isotopes to be measured, beginning with the
'           Reference Isotope followed by one or more Sample Isotopes.
' Set:      a series of (usually 10) consecutive measurements of a single
'           isotope. Mean and standard deviation (Sigma) are reported;
'           individual measurements are not.
'Tolerance  Measurements are rejected if their difference from the mean is
'           greater than the Tolerance (usually 1.5 to 3) times Sigma
'MaxDev     Maximum value a base or isotope Sigma can have. Helps detect
'           bad sets( see below)
'For Isotope Measurements:
' Good Set  A Set with no less than 'IsoGoodReads' measurements (usually 7)
'           and a Sigma less than < 'IsoMaxDev'.  The Mean and Sigma for all
'           Good Sets are reported and used to compute Ratios if possible.
' Bad Set:  Violates Good Set rules and causes a break in measurements.
'           Increments the 'Group' number for subsequent measurements and
'           forces immediate re-measure of Ref. Isotope Set.  Bad Set
'           data is reported but is not used to compute ratios.
' Group:    A collection of good data sets, unbroken by a bad data set.
```

```
'           Begins with a Ref. Isotope set. Within a group, only sets between
'           the initial and final Ref. Isotope set are used to construct
'           Ratios. All sets in a Group have the same 'Group'number.
' Ratios:   For each Sample isotope, an inferred Reference Isotope value is
'           determined by interpolation betweeen bracketing Ref. Isotope
'           measurements. The interpolation is time-weighted to yield
'           an apparent value for the Ref. Isotope at the time the Sample
'           Isotope was measured. The ratio is the mean value of the Sample
'           Isotope set divided by the apparent Reference Isotope value.
'           Constructed for each sequence within a group for which there is
'           a Good Set of initial and final reference isotope measurements.
'For Baseline measurements:
' Good Set  A Set with no less than 'BaseGoodReads' measurements (usually 7)
'           and a Rel. Std. Deviation < 'BaseMaxDev'. The mean and standard
'           deviation for all Good Sets are reported and the average of
'           'BaseNumReads' (usually 10) mean values becomes the Iso.Baseline
'           parameter for each isotope
' Bad Set:  Violates Good Set rules and increments 'BadReadFlag'; which, if
'           it exceeds BaseMaxBadSets (usually 3), aborts the measurement.
'End Comments
'
'-----------------------------------------------------------------
'Setup


 CONST FALSE = 0, TRUE = NOT FALSE     'TRUE = -1  in Quick BASIC 4.0 & up
 TYPE IsoParams
      Mass     AS INTEGER          ' Isotopic Mass, from Parameters file
      Gauss    AS SINGLE           ' Gaussmeter reading for above
      Range    AS SINGLE           ' PA range from Params. File; modifiable
      BaseGauss1 AS SINGLE         ' Baseline Gaussmeter reading, one required
      BaseGauss2 AS SINGLE         ' 2nd optional; ignore if 0 or negative.
 END TYPE

 TYPE MeasRecord                  ' Measured data
      Mass     AS INTEGER          ' As in IsoParams; links the two arrays
      Avrg     AS DOUBLE           ' Av. Volt. from 1 set of (usually) 10 samples
      StDv     AS DOUBLE           ' Standard Deviation for Avrg
      Tick     AS SINGLE           ' Time (Sec.) set was measured
      Valid    AS INTEGER          ' Number of valid readings
 END TYPE

'  Subscripts are Sample #, Isotope #, and ratio type
 REDIM Ratio(30, 3, 2) AS DOUBLE

 TYPE ResultRecord
      SumRatio      AS DOUBLE       ' Sum of ratios
      SqSumRatio    AS DOUBLE       ' Squares of sums of ratios
      NumRatios     AS INTEGER      ' Counts all ratios of a given type
      RatioAv       AS DOUBLE       ' Average of Ratios
      RatioDev      AS DOUBLE       ' Std.Dev. of ratios
      GoodRatios    AS INTEGER      ' Number of Ratios within Tolerance
 END TYPE

' The Results Subscripts are Isotope # and Ratio Type where Ratio Type is:
'   0 to indicate combined results for both kind of ratios
'   1 for "normal" Ratio of Sample vs time-weighted, interpolated Reference
'   2 for "reverse sense" Ratio of weighted, interpolated Sample vs. Reference
 REDIM Results(3, 2) AS ResultRecord
```

```
'Entries in the IsoParams.Rec (Params) file are as follows:
 DIM IsoName         AS STRING * 10 ' Name of this Sequence of Isotopes
 DIM NumIsos         AS INTEGER     ' No. of Isotopes,
 REDIM Iso(3)        AS IsoParams   ' Holds params for each isotope; REDIMed
                                    ' later to actual size. Iso(1) is usual-
                                    ' ly the reference isotope
 DIM NumReads        AS INTEGER     ' Number of readings per set per isotope
 DIM IsoGoodReads    AS INTEGER     ' Min no of reads in a Good Isotope Set
 DIM BaseNumReads    AS INTEGER     ' Number of Base sets to read per isotope
 DIM BaseGoodReads   AS INTEGER     ' Min no of reads in a Good Baseline set
 DIM NumRatios       AS INTEGER     ' Minimum number of valid ratios
 DIM RatioMaxDev     AS SINGLE      ' Max. Relative Std.Dev for Ratios
 DIM RatioSigmaTol   AS SINGLE      ' Tolerance for Ratio Std.Dev
 DIM IsoMaxDev       AS SINGLE      ' Max. Permissible Std.Dev
 DIM IsoSigmaTol     AS SINGLE      ' Tolerance for Isotope Std. Dev.
 DIM BaseMaxDev      AS SINGLE      ' Max Rel. Stand. Dev. in BaseLine set
 DIM BaseSigmaTol    AS SINGLE      ' Tolerance for Base Std. Dev.
 DIM BaseMaxBadSets AS INTEGER      ' Maximum allowed bad baseline sets
 DIM Delay           AS INTEGER     ' Min. time to let PA settle

 SampName$ = "": FilAmps$ = "": FilVolts$ = ""
 OpName$ = "": SubmitName$ = ""


' Scratch variables for measurement calls and computing ratios
 DIM Mean AS DOUBLE, Sigma AS DOUBLE, Sum AS DOUBLE, SqSum AS DOUBLE
 DIM Time AS SINGLE

'RefIso is the index of the Reference Isotope (conventionally 1)
 RefIso = 1

 DIM RangeMenu(7) AS STRING
 RangeMenu(0) = ".003 Volts": RangeMenu(1) = "0.01 Volts"
 RangeMenu(2) = ".03  Volts": RangeMenu(3) = "0.1  Volts"
 RangeMenu(4) = "0.3  Volts": RangeMenu(5) = "1.0  Volts"
 RangeMenu(6) = "3.0  Volts": RangeMenu(7) = "10   Volts"

 DIM RevSenseMenu(0 TO 1) AS STRING
 RevSenseMenu(0) = "Omit Reverse Sense Ratios    "
 RevSenseMenu(1) = "Include Reverse Sense Ratios"

 DIM IntegrateMenu(0 TO 1) AS STRING
 IntegrateMenu(0) = "Integrate for 10 Line Cycles    "
 IntegrateMenu(1) = "Integrate for 100 Line Cycles   "

 RepeatHeader$ = "Choose one of the following:"
 DIM RepeatMenu(4) AS STRING
 RepeatMenu(0) = "New Sample, New parameters         "
 RepeatMenu(1) = "New Sample, Same parameters        "
 RepeatMenu(2) = "Same Sample, New Paramaters        "
 RepeatMenu(3) = "Redo this measurement (no changes)"
 RepeatMenu(4) = "Quit                               "

 REDIM RatioType$(3)
 RatioType$(0) = "Combined Data     "
 RatioType$(1) = "Samp/Interp-Ref   "
 RatioType$(2) = "Interp-Samp/Ref   "
 RatioType$(3) = "All types         "
 COLOR 7, 1
'End Setup
'-----------------------------------------------------------------
```

```
'-----------------------------------------------------------------
'Open Files
GetInitialData:
 CLS
 CALL GetSampName(SampName$)
 CALL GetInfo(OpName$, FilAmps$, FilVolts$, SubmitName$, Comment$)
 SameParams = FALSE
SetupFiles:
 COLOR 7, 1: CLS : TickMark$ = TIME$
 CLOSE
 OPEN "Cons:" FOR OUTPUT AS #1
 OPEN "LPT1:" FOR OUTPUT AS #2    'for parallel port printer
' OPEN "LPT2" FOR OUTPUT AS #2    'for HPIB printer; Note no colon

' Trick to make Rext$ display two digits (eg: .R00). start at 100 then use
' RIGHT$(STR$(..),2)) to lop off the 1; always leaves 2 digits in the string.
 FOR Rnn = 100 TO 199
      CLOSE #5
      Rext$ = ".R" + RIGHT$(STR$(Rnn), 2)  'part of trick.
      RecFile$ = "C:" + SampName$ + Rext$  'File #5 ASCII, Comma delimit file
      OPEN RecFile$ FOR APPEND AS #5
      IF LOF(5) = 0 THEN EXIT FOR  ' preserves Rnn
 NEXT Rnn
 IF Rnn > 199 THEN  'This means something is probably very wrong!
      PRINT "There are now 99 "; SampName$; ".Rnn files!  This program"
      PRINT "cannot make any more Record files with this Sample ID."
      INPUT "(hit 'ENTER'; Program will stop)", X$
      CLOSE : STOP
 END IF

OpenDataFile:
 PRINT " The Record file for this sample is: "; RecFile$: PRINT
 PRINT " Please be sure a formatted diskette with room for data is"
 INPUT ; " in Drive B: then hit <Enter>. ", X$: PRINT : PRINT
 CLOSE #4
 Bnn = Rnn
 Bext$ = ".B" + RIGHT$(STR$(Bnn), 2)
 Datafile$ = "B:\" + SampName$ + Bext$  'File #4 Backup for Record file
 ON ERROR GOTO BadOpenDataFile
 OPEN Datafile$ FOR APPEND AS #4
 ON ERROR GOTO 0
 IF LOF(4) > 0 THEN   'This means something is probably very wrong!
      CLOSE : CLS : KILL RecFile$: COLOR 14, 4
      PRINT "There is already a file named "; Datafile$; " but it does"
      PRINT "not correspond to the new Record file, "; RecFile$
      PRINT "Please rename "; Datafile$; ", move it to other storage "
      INPUT "and/or replace the Diskette.(hit 'ENTER'; Program will stop)", X$
      COLOR 7, 1
      STOP
 END IF

 PRINT " The Data file for this sample is: "; Datafile$: PRINT
 CLOSE #3
 Logfile$ = "C:" + SampName$ + ".LOG"      'File #3 records everything
 OPEN Logfile$ FOR APPEND AS #3
 PRINT #3, "---------Begin Log For "; SampName$; " ----------"
 PRINT " The Log file for this sample is: "; Logfile$
 IF LOF(3) > 0 THEN
      PRINT " That Logfile already exists. Your current session will be appended to it"
 END IF: PRINT
```

```
      PRINT
      PRINT "     Sample Name: "; SampName$; TAB(36);
      PRINT " Time: "; TickMark$; TAB(52);
      PRINT "Date: "; DATE$
      PRINT "  Filament Volts: "; FilVolts$; TAB(40);
      PRINT "   Filament Amps: "; FilAmps$
      PRINT "  Operators Name: "; OpName$; TAB(40);
      PRINT "Submitter's Name: "; SubmitName$
      PRINT "        Comments: "; Comment$

 LOCATE 24, 5
 IF NOT YNans("    Is this all correct?") THEN GOTO GetInitialData

'End Open files
'---------------------------------------------------------------------
'GET Parameters

 IF SameParams = TRUE THEN GOTO MeasureBase   'Keep same parameters

 ON ERROR GOTO ParamFileErr
 OPEN "IsoParam.Rec" FOR INPUT AS #6
 ON ERROR GOTO 0                  'turn off error routine
 REDIM NameList(20) AS STRING
 'look for string
 IsoParamRev$ = "Last revised 11 Mar 93 for Rev 6.7"

 DO 'Find first non-blank line
       LINE INPUT #6, Line1$
 LOOP UNTIL LEN(Line1$) > 0
 IF INSTR(Line1$, IsoParamRev$) = 0 THEN
       CLOSE : CLS :  COLOR 14, 4
       PRINT "Wrong revision of ISOPARAMS.REC. The revision is:"
       PRINT MID$(Line1$, 14)
       PRINT "it should be:"
       PRINT IsoParamRev$
       INPUT "(hit 'ENTER'; Program will stop)", X$
       COLOR 7, 1: STOP
 END IF

I = 0: Flag = FALSE
DO
      NameList(I) = GetStr12$
      IF (NameList(I) = "") THEN
            Flag = TRUE                    ' Blanks separate Param records
      ELSEIF (Flag = TRUE) THEN
            I = I + 1: Flag = FALSE
      ELSE
            Flag = FALSE
      END IF
LOOP UNTIL EOF(6)
 NameList(I) = ""                  ' Last one is invalid
 MaxI = I - 1                      ' Subscript for last Valid NameList entry
 CLS
 Choice = 0
 Header$ = "Select the desired Isotope Range"
 CALL VertMenu(Header$, NameList(), MaxI, 2, Choice)
 CLOSE #6

NewParam:
 OPEN "IsoParam.Rec" FOR INPUT AS #6  ' Again
 DO                                ' skip unwanted entries
```

```
 LOOP UNTIL GetStr12$ = NameList(Choice)
 IsoName = NameList(Choice)
 NumIsos = GetInt
 REDIM Iso(NumIsos) AS IsoParams
 FOR I = 1 TO NumIsos
 Iso(I).Mass = GetInt: Iso(I).Gauss = GetSingle: Iso(I).Range = GetSingle
 Iso(I).BaseGauss1 = GetSingle: Iso(I).BaseGauss2 = GetSingle:
 NEXT I
 NumReads = GetInt
 IsoGoodReads = GetInt
 BaseNumReads = GetInt
 BaseGoodReads = GetInt
 NumRatios = GetInt
 RatioMaxDev = GetSingle
 RatioSigmaTol = GetSingle
 IsoMaxDev = GetSingle
 IsoSigmaTol = GetSingle
 BaseMaxDev = GetSingle
 BaseSigmaTol = GetSingle
 BaseMaxBadSets = GetInt
 Delay = GetInt
'End Get Parameters
'-----------------------------------------------------------------------
'Change Parameters
ChangeParams:
 REDIM RangeNo(NumIsos) AS INTEGER
 FOR I = 1 TO NumIsos
 RangeNo(I) = 0
 ON ERROR GOTO fini
 DO UNTIL (ABS(VAL(RangeMenu(RangeNo(I))) - Iso(I).Range) < .00001)
       RangeNo(I) = RangeNo(I) + 1
       IF RangeNo(I) > 7 THEN ERROR 248
 LOOP
 ON ERROR GOTO 0
 RowNum = 2
 COLOR 7, 1: CLS
 Header$ = "Select Parametric Amplifier Range for Mass "
 Header$ = Header$ + STR$(Iso(I).Mass) + ":"
 CALL VertMenu(Header$, RangeMenu(), 7, RowNum, RangeNo(I))
 Iso(I).Range = VAL(RangeMenu(RangeNo(I)))
 NEXT I
 CALL VertMenu("", RevSenseMenu(), 1, 12, RevSense)
 Choice = 0
 CALL VertMenu("", IntegrateMenu(), 1, 16, Choice)
  TMax = 1 + RevSense          ' =1 for normal ratios, =2 to include RevSense
  IntegTime = 10 ^ (Choice + 1) 'returns 10 for 10 Hz and 100 for 100 Hz
 'End Change Parameters
'-----------------------------------------------------------------------
'Measure Baseline
MeasureBase:
 ErrFlag = FALSE 'Set for too many bad readings (BadReadFlag or G too large)
 P = 3
 GOSUB DisplayHeader

 COLOR 7, 1: CLS
 CALL ReadDVM(-100#)
' Debug ---  CALL ReadDVM(-10#)
 Time0! = TIMER

 REDIM Baseline(NumIsos, 2) AS MeasRecord
 REDIM NumBases(NumIsos) AS INTEGER
```

```
 PRINT
 PRINT "    Base Sigma Tol: "; BaseSigmaTol; "    Base Max Dev: "; BaseMaxDev
 PRINT "Base Gauss  Mean        Sigma       MaxRejLev.  RejLevel   Good Readings"
 FOR I = 1 TO NumIsos       'Isotope Index as usual
'  (Iso(I).BaseGauss2 > 0) Evaluates as TRUE (-1) or FALSE (0)
  NumBases(I) = 1 - (Iso(I).BaseGauss2 > 0)
  SqSum# = 0
  FOR L = 1 TO NumBases(I)
      SELECT CASE L
       CASE 1
        BaseGauss! = Iso(I).BaseGauss1      ' required baseline reading
       CASE 2
        BaseGauss! = Iso(I).BaseGauss2      ' optional baseline reading
      END SELECT

      CALL MoveTo(BaseGauss!, RangeNo(I), Delay)
      BadReadFlag = 0        ' keeps track of bad readings
      PRINT #3, "Base Gauss:"; BaseGauss!; "    Base Sigma Tol: "; BaseSigmaTol;
      PRINT #3, "    Base Max Dev: "; BaseMaxDev
      PRINT #3, "Mean        Sigma       MaxRejLev.  RejLevel   Good Readings"
      DO              ' get at least GoodReads good baseline readings without
                        ' more than BaseMaxBadSets in a row bad readings
        IF BadReadFlag >= BaseMaxBadSets THEN
          ErrFlag = TRUE
          PRINT BadReadFlag;" bad Baseline readings in a Row. Bailing out..."
          GOTO WriteRecordFiles
        END IF

        CALL Measure(BaseSigmaTol, BaseMaxDev, BaseNumReads, Mean#, Sigma#,T!, GoodReads)
        RejectLevel# = BaseSigmaTol! * Sigma#
        MaxRejLevel# = BaseMaxDev! * Mean#
        PRINT USING "####.##     "; BaseGauss!;
        PRINT USING "#.#######   "; Mean#; Sigma#; MaxRejLevel#; RejectLevel#;
        PRINT USING "####   "; GoodReads
        PRINT #3, USING "#.#######   ";Mean#;Sigma#;MaxRejLevel#;RejectLevel#;
        PRINT #3, USING "####   "; GoodReads
        BadReadFlag = BadReadFlag + 1
      LOOP UNTIL GoodReads >= BaseGoodReads
      PRINT
      PRINT #3,
      Baseline(I, L).Avrg = Mean#
      Baseline(I, L).StDv = Sigma#
      Baseline(I, L).Valid = GoodReads
      Baseline(I, 0).Avrg = Baseline(I, 0).Avrg + Mean# * GoodReads
      Baseline(I, 0).Valid = Baseline(I, 0).Valid + GoodReads
      SqSum# = SqSum# + Mean# ^ 2
  NEXT L
  Baseline(I, 0).Avrg = Baseline(I, 0).Avrg / Baseline(I, 0).Valid
 NEXT I

 FOR P = 1 TO 3 STEP 2
  GOSUB DisplayBaseData
 NEXT P
' Debug -- CLOSE : INPUT "halt", X$
'End Measure Baseline
'-----------------------------------------------------------------------
'Measure Samples
' Meas holds all data from one full set of ratio measurements
' change @ ver 7x5: now terminate after a full set; previously, only a
' single reference was done in the last set. Change accommodates
' reverse sense ratios
 REDIM Meas(2 * NumRatios, NumIsos) AS MeasRecord
```

```
REDIM Group(2 * NumRatios + 1)  'Allow plenty of room
G = 1       'Counts Groups of good data sets; bad data set forces new Group
N = 0       'Number of ratios completed so far
J = 0       'index to measurement array

CALL ReadDVM(-IntegTime * 1#)

'Debug--
' LPRINT : LPRINT "SetNum    Group     FullSets  ";
' FOR I = 1 TO Numisos
'   LPRINT USING "Mass ###     "; Iso(I).Mass;
' NEXT I: LPRINT
  CLS
     MeasFmt1$ = "###  ###     ###    #.######  #.#######   ##"
     MeasFmt2$ = "             ###    #.######  #.########  ##"
 PRINT "Measured Values:"
 PRINT : PRINT "Set  Group  Mass  Mean      Sigma       GoodReads"
 PRINT #3, "Measured Values:"
 PRINT #3, : PRINT #3, "Set  Group  Mass  Mean      Sigma       GoodReads"


 DO ' Loop on J and N values
.
' Debug-- LPRINT USING "###       "; J; G; N;

 FOR I = 1 TO NumIsos              ' Denotes isotope in sequence
     CALL MoveTo(Iso(I).Gauss, RangeNo(I), Delay)
     CALL Measure(IsoSigmaTol, IsoMaxDev, NumReads, Mean, Sigma, Time!,GoodReads)
     Mean = Mean - Baseline(I, 0).Avrg
' Keep all data in LOG file, even bad data! Note no CR (so error messages
' Can be printed on the same line )
  IF I = 1 THEN
       PRINT USING MeasFmt1$; J; G; Iso(I).Mass; Mean; Sigma; GoodReads;
       PRINT #3, USING MeasFmt1$; J; G; Iso(I).Mass; Mean; Sigma; GoodReads;
  ELSE
       PRINT USING MeasFmt2$; Iso(I).Mass; Mean; Sigma; GoodReads;
       PRINT #3, USING MeasFmt2$; Iso(I).Mass; Mean; Sigma; GoodReads;
  END IF

' Debug --
'  LPRINT USING "### "; I; J;
'  LPRINT USING "#.######    "; Mean; Sigma; GoodReads

     IF (ABS(Mean) < IsoSigmaTol * Sigma) THEN
             Mean = 0!        'forces bad read
             PRINT " ERR: |Mean| < "; IsoSigmaTol; "* Sigma"
             PRINT #3, " ERR: |Mean| < "; IsoSigmaTol; "* Sigma"
     ELSEIF (GoodReads < IsoGoodReads) THEN  'Not enough good reads this time
             Mean = 0!        'forces bad read
             PRINT " ERR: Too Few GoodReads"
             PRINT #3, " ERR: Too Few GoodReads"
     ELSEIF (IsoSigmaTol * Sigma > IsoMaxDev * Mean) THEN 'Sigma too big
             SigDivMean$ = STR$(CSNG(Sigma / Mean))
             Mean = 0!        'forces bad read
             PRINT " ERR: RSD"; SigDivMean$; " too big"
             PRINT #3, " ERR: RSD"; SigDivMean$; " too big"
     ELSE ' need CR at end of printed line for ok data
             PRINT
             PRINT #3,
     END IF
```

```
      IF (Mean = 0!) THEN
            G = G + 1                        ' start a new Group
            IF G > NumReads THEN
                  PRINT " Too many bad data sets.  Bailing out"
                  ErrFlag = TRUE
                  GOTO WriteRecordFiles
            END IF
            N = N - 1                        ' don't count partial sets of data
            EXIT FOR                         ' back to reference isotope
      END IF

      Meas(J, I).Mass = Iso(I).Mass
      Meas(J, I).Avrg = Mean
      Meas(J, I).StDv = Sigma
      Meas(J, I).Tick = Time! - Time0!
      Meas(J, I).Valid = GoodReads
      Group(J) = G
      IF (J = 2 * NumRatios) THEN GOTO ExceedsArray

 NEXT I:
  PRINT : PRINT #3,
' Debug --  LPRINT

'   If the Ref. Isotope is zero, there are no valid readings for the current
'   value of J, and we do not store a row of nulls
      IF Meas(J, 1).Avrg > 0! THEN J = J + 1

      LastSample = J        ' net total of J + 1 rows of information with
                            ' NumRatios good sets so far (keep updating)

      N = N + 1                           ' counts ratios
      IF (N = NumRatios) THEN EXIT DO  'Normal Exit
 LOOP    ' terminates on previous EXIT DO


      CALL MoveTo(Iso(1).Gauss, RangeNo(1), Delay)    'Move back to Ref

 P = 3
  GOSUB DisplayIsoMeas

'End Measure Samples
'-------------------------------------------------------------------
'Analyze Ratios
' Subscripts in Ratio are sample#, Isotope #, and ratio type
' Results record includes three kinds of ratios: "normal" and "reverse
' sense" interpolated ratios (Subscripts 1 and 2) and the combined results
' for both kinds of ratios (Subscript 0)
' RefIso is the index of the Reference Isotope (conventionally 1)

 REDIM Ratio(LastSample + 1, NumIsos, 2) AS DOUBLE
 REDIM Results(NumIsos, 2) AS ResultRecord
 FOR I = 1 TO NumIsos
      IF I <> RefIso THEN  ' Form Ratios for Sample Isotopes only
        FOR J = 0 TO LastSample - 1  ' Index to Meas() Array
' "Normal" ratio; interpolate between brackting Refs to time of Sample
            Test! = Meas(J, I).Avrg * Meas(J, RefIso).Avrg * Meas(J + 1, RefIso).Avrg
            IF (Test!) > 0 THEN     ' no zero elements in Test!
                  InterpRef# = Interpol8#(Meas(), I, RefIso, J)
                  Ratio(J, I, 1) = Meas(J, I).Avrg / InterpRef#
                  CALL RatioStats(Ratio(), Results(), I, J, 1)
            END IF 'Test!
```

```
' "Reverse Sense" ratio; interpolate between brackting Samps. to time of Ref
            IF RevSense THEN
             Test! = Meas(J, I).Avrg * Meas(J + 1, I).Avrg * Meas(J + 1, RefIso).Avrg
             IF (Test!) > 0 THEN      ' no zero elements in Test!
                  InterpAv# = Interpol8#(Meas(), RefIso, I, J + 1)
                  Ratio(J + 1, I, 2) = InterpAv# / Meas(J + 1, RefIso).Avrg
                  CALL RatioStats(Ratio(), Results(), I, J + 1, 2)
             END IF 'Test!
            END IF 'RevSense
        NEXT J
'     Now filter out of spec ratios
        FOR T = 1 TO TMax    'TMax = 2 => RevSense is TRUE. TMax = 1 otherwise
'     WHILE Results(I, T).NumRatios <> Results(I, T).GoodRatios
             Results(I, T).NumRatios = Results(I, T).GoodRatios 'save GoodRatios

'Activate this code to compare kinds of ratios separately
'       Mean = Results(I, T).RatioAv
'       Tol! = RatioSigmaTol * Results(I, T).RatioDev

' Activate this code to use the combined ratios for comparison  Also
' generates Normal Ratio data if Reverse Sense isn't used.
             Mean = Results(I, 0).RatioAv
             Tol! = RatioSigmaTol * Results(I, 0).RatioDev

             IF Tol! > RatioMaxDev * Mean THEN
                   Tol! = RatioMaxDev * Mean
                   BEEP: PRINT "ERR:  Ratio Deviation clamped to RatioMaxDev * Mean"
                   PRINT #3, "Ratio Tol! ="; Tol!; " Results("; I; ", 0).RatioDev = ";
                   PRINT #3, Results(I, 0).RatioDev
             END IF

             FOR J = 0 TO LastSample
                  IF Ratio(J, I, T) > 0 AND (ABS(Ratio(J, I, T) - Mean) > Tol!) THEN
                    Ratio(J, I, T) = -Ratio(J, I, T)
                    CALL RatioStats(Ratio(), Results(), I, J, T)
                  END IF
             NEXT J
'     WEND
        NEXT T
      END IF 'I<>RefIso
  NEXT I

 P = 3
      GOSUB DisplayRatios
      GOSUB DisplayAnalysis

'End Analyze Ratios
'-------------------------------------------------------------------
'Begin Write Recordfiles
' Write files as comma delimited ASCII.  These files are formatted and
' ordered for use with a spreadsheet.  The information is the order: header
' information, followed by results, calculated ratios, Isotope measurements
' and baseline measurements.  Nine or more significant figures are kept for
' double precision variables.

WriteRecordFiles:
 ON ERROR GOTO DiskWriteError
 FOR P = 4 TO 5
  WRITE #P, SampName$, DATE$, TickMark$, FilVolts$, FilAmps$, IsoName
  WRITE #P, STR$(IntegTime) + " Hz", SubmitName$, OpName$
  WRITE #P, Comment$
```

```
  FOR I = 1 TO NumIsos
       PRINT #P, USING " ###_, "; Iso(I).Mass;
  NEXT I: PRINT #P, USING " ###"; Iso(RefIso).Mass
  PRINT #P,
  Fs$ = CHR$(47)

  IF ErrFlag THEN GOTO RawData'  write computations only for complete runs
  FOR I = 1 TO NumIsos
      IF I <> RefIso THEN
       WRITE #P, Iso(I).Mass, Fs$, Iso(RefIso).Mass
       FOR T = 0 TO 2
        PRINT #P, USING "\            \"; RatioType$(T);
        PRINT #P, USING ", ###.##########"; Results(I, T).RatioAv;
        PRINT #P, USING ", #.############"; Results(I, T).RatioDev;
        PRINT #P, USING ", ###"; Results(I, T).GoodRatios
       NEXT T: WRITE #P,
      END IF
  NEXT I

  FOR I = 1 TO NumIsos
      IF I <> RefIso THEN
       WRITE #P, Iso(I).Mass, Fs$, Iso(RefIso).Mass
       FOR J = 0 TO LastSample
            IF Ratio(J, I, 1) <> 0 OR Ratio(J, I, 2) <> 0 THEN
                  PRINT #P, USING " ##_, ##"; J; Group(J);
              FOR T = 1 TO 2
                  IF Ratio(J, I, T) <> 0 THEN
                        PRINT #P, USING ", ###.##########"; Ratio(J, I, T);
                  ELSE
                        PRINT #P, ",    0.0         ";
                  END IF
              NEXT T: PRINT #P,
            END IF  'Ratios <> 0
       NEXT J: PRINT #P,
      END IF
  NEXT I
  PRINT #P,
RawData:
  FOR I = 1 TO NumIsos
       PRINT #P, USING "###_,  ####.#"; Iso(I).Mass; Iso(I).Gauss
       FOR J = 0 TO LastSample
            IF Meas(J, I).Avrg > 0 THEN
              PRINT #P, USING " ##_, ##"; J; Group(J);
              PRINT #P, USING ", ###.##########"; Meas(J, I).Avrg;
              PRINT #P, USING ",    #.##########"; Meas(J, I).StDv;
              PRINT #P, USING ", ###"; Meas(J, I).Valid;
              DelTime! = MaxOf(Meas(J, I).Tick - Meas(J + (J > 0), I).Tick, 0)
              IF (Meas(J + (J > 0), I).Tick = 0) THEN DelTime! = 0
              PRINT #P, USING ", ###.####"; DelTime!;
              PRINT #P,
            END IF
       NEXT J: PRINT #P,
  NEXT I

  FOR I = 1 TO NumIsos
       PRINT #P, USING " ###, "; Iso(I).Mass
       FOR L = 1 TO NumBases(I)
            IF L = 1 THEN PRINT #P, USING " ####.#, "; Iso(I).BaseGauss1;
            IF L = 2 THEN PRINT #P, USING " ####.#, "; Iso(I).BaseGauss2;
            PRINT #P, USING "#.########, "; Baseline(I, L).Avrg;
            PRINT #P, USING "#.########, "; Baseline(I, L).StDv;
```

```
            PRINT #P, USING "###"; Baseline(I, L).Valid;
            PRINT #P,
        NEXT L
        PRINT #P, USING "        , #.########,"; Baseline(I, 0).Avrg;
        PRINT #P, USING "            , ###"; Baseline(I, 0).Valid
        PRINT #P,
   NEXT I: PRINT #P, CHR$(12)
 NEXT P
ON ERROR GOTO 0
'End Write Recordfiles
'---------------------------------------------------------------------
'Report
CLS :

 FOR P = 1 TO 2

      GOSUB DisplayHeader

      GOSUB DisplayBaseData

      GOSUB DisplayIsoMeas

      IF ErrFlag = FALSE THEN 'Only if the measurement completed properly
            GOSUB DisplayRatios
            GOSUB DisplayAnalysis
      END IF
 NEXT P

 PRINT #2, CHR$(12); : PRINT #3, CHR$(12);
 PRINT #1, CHR$(13):

 LOCATE 25, 2: COLOR 15, 1
 INPUT ; "Hit 'ENTER' to continue ", X$: COLOR 7, 1

'End Report
'---------------------------------------------------------------------
'Begin LoopLogic
 CLS :
' size = 4 and RowNum = 2 (rows from top)
RepeatFlag = 0 'Default choice to be highlighted
 CALL VertMenu(RepeatHeader$, RepeatMenu(), 4, 2, RepeatFlag)
 COLOR 7, 1: CLS
 SameParams = FALSE
 SELECT CASE RepeatFlag
  CASE 0               ' New Sample, Change parameters
       SampName$ = ""
' Activate the following line to pre-clear all information
'     FilAmps$ = "": FilVolts$ = "": OpName$ = "": SubmitName$ = ""
       CALL GetSampName(SampName$)
       CALL GetInfo(OpName$, FilAmps$, FilVolts$, SubmitName$, Comment$)
  CASE 1               ' New Sample, Same parameters
       SampName$ = ""
       CALL GetSampName(SampName$)
       SameParams = TRUE
  CASE 2               ' Same Sample, Change Paramaters

' Activate the following line to pre-clear all information
'     FilAmps$ = "": FilVolts$ = "": OpName$ = "": SubmitName$ = ""

       CALL GetInfo(OpName$, FilAmps$, FilVolts$, SubmitName$, Comment$)
```

```
  CASE 3                ' Redo this measurement (no changes)
       SameParams = TRUE
  CASE ELSE             ' Quit
       STOP
 END SELECT

 GOTO SetupFiles

 PRINT " At 'Begin Error messages'. Should not ever get here!": GOTO fini

'End LoopLogic
'-----------------------------------------------------------------------
'Begin Display GoSubs
DisplayHeader:
      PRINT #P,
      PRINT #P, "     Sample Name: "; SampName$; TAB(36);
      PRINT #P, " Time: "; TickMark$; TAB(52);
      PRINT #P, "Date: "; DATE$
      PRINT #P, "  Filament Volts: "; FilVolts$; TAB(40);
      PRINT #P, "   Filament Amps: "; FilAmps$
      PRINT #P, "Integration Time: "; STR$(IntegTime); " Line Cycles"
      PRINT #P, "  Operators Name: "; OpName$; TAB(40);
      PRINT #P, "Submitter's Name: "; SubmitName$
      PRINT #P, "        Comments: "; Comment$
      PRINT #P,
RETURN

DisplayBaseData:
 PRINT #P,
 PRINT #P, "Baseline Measurements:"
 FOR I = 1 TO NumIsos
        PRINT #P, USING " Mass: ### "; Iso(I).Mass
        PRINT #P, "Base Gauss    ";
        PRINT #P, "Baseline     Sigma          Reads  ";
        PRINT #P,
      FOR L = 1 TO NumBases(I)
        IF L = 1 THEN PRINT #P, USING " ####.#       "; Iso(I).BaseGauss1;
        IF L = 2 THEN PRINT #P, USING " ####.#       "; Iso(I).BaseGauss2;
        PRINT #P, USING "#.########    "; Baseline(I, L).Avrg;
        PRINT #P, USING "#.########    "; Baseline(I, L).StDv;
        PRINT #P, USING "###"; Baseline(I, L).Valid;
        PRINT #P,
      NEXT L
        PRINT #P, USING "   Average:  #.########    "; Baseline(I, 0).Avrg;
        PRINT #P, USING "             ###"; Baseline(I, 0).Valid;
      PRINT #P, : PRINT #P,
 NEXT I
RETURN

DisplayIsoMeas:
      PRINT #P, "Isotope Measurements:"
      FOR I = 1 TO NumIsos
        PRINT #P, USING " Mass: ###   Gauss: ####.#"; Iso(I).Mass; Iso(I).Gauss
                  PRINT #P, " No. Group    ";
'               "##.#######    #.#######     #.#######     ###    ###.###"
     PRINT #P, " Mean         Sigma         Rel.Std.Dev   Reads  Time Int.";
        PRINT #P,
```

```
        FOR J = 0 TO LastSample
             IF Meas(J, I).Avrg > 0 THEN
                  PRINT #P, USING "### ###       "; J; Group(J);
                  PRINT #P, USING "##.#######     "; Meas(J, I).Avrg;
                  PRINT #P, USING "#.#######      "; Meas(J, I).StDv;
                  PRINT #P, USING "#.#######      "; Meas(J, I).StDv / Meas(J, I).Avrg;
                  PRINT #P, USING "###     "; Meas(J, I).Valid;
                  DelTime! = MaxOf(Meas(J, I).Tick - Meas(J + (J > 0), I).Tick, 0)
                  IF (Meas(J + (J > 0), I).Tick = 0) THEN DelTime! = 0
                  PRINT #P, USING "###.###"; DelTime!;
                  PRINT #P,
             END IF
        NEXT J: PRINT #P,
      NEXT I
RETURN

DisplayRatios:
      FOR I = 1 TO NumIsos
       IF I <> RefIso THEN
            PRINT #P, USING " Ratios: ###/###"; Iso(I).Mass; Iso(1).Mass
'               "### ###     ##.#######           ##.#######           "
            PRINT #P, " Set Group   Samp/Interp-Ref      ";
            IF RevSense THEN PRINT #P, "Interp-Samp/Ref ";
            PRINT #P,
            FOR J = 0 TO LastSample
'        print ratios only if one or more of them is <> 0
             IF Ratio(J, I, 1) <> 0 OR Ratio(J, I, 2) <> 0 THEN
                 PRINT #P, USING "### ###      "; J; Group(J);
                 FOR T = 1 TO TMax
                    IF Ratio(J, I, T) <> 0 THEN
                      PRINT #P, USING "######.#######       "; Ratio(J, I, T);
                    ELSE
                      PRINT #P, "     -----           ";
                    END IF
                  NEXT T: PRINT #P,
             END IF  'Ratios <> 0
            NEXT J: PRINT #P,
       END IF
      NEXT I
      PRINT #P,
RETURN

DisplayAnalysis:
      PRINT #P, "Analysis of Ratios  "
      FOR I = 2 TO NumIsos
        PRINT #P, USING "Mass Ratio: ###/###"; Iso(I).Mass; Iso(1).Mass;
                  PRINT #P, "   Mean       (Inverse)      ";
        PRINT #P, " Sigma      Rel.Std Dev   Valid"
        ' (N < 3) evaluates to -1 if it is TRUE  and to 0 if it is FALSE
        FOR TT = 0 TO 2 * (-(RevSense <> 0))' Either Samp/Interp-Ref only or
                                            ' all three
          T = (TT + 1) * -(TT < 2)    ' T sequence 1, 2, 0 or just 1
          PRINT #P, RatioType$(T);
          PRINT #P, USING "######.#######  "; Results(I, T).RatioAv;
          IF Results(I, T).RatioAv > 0 THEN
           PRINT #P, USING "(###.#######) "; 1 / (Results(I, T).RatioAv);
           PRINT #P, USING "###.#######  "; Results(I, T).RatioDev;
           PRINT #P, USING "#.#######    "; Results(I, T).RatioDev /Results(I, T).RatioAv;
```

```
          ELSE
              PRINT #P, "    -----        ";
              PRINT #P, "    -----    ";
              PRINT #P, "    -----      ";
            END IF
            PRINT #P, USING "###"; Results(I, T).GoodRatios
       NEXT TT: PRINT #P,
      NEXT I
RETURN

'End Display GoSubs
'---------------------------------------------------------------------
'Begin ERROR messages
ExceedsArray:
 BEEP
 PRINT " Can't store more than "; 2 * NumRatios; " measurements.  Bailing out..."
 RESUME fini:

ParamFileErr:
 BEEP: PRINT "Can't find disk file ISOPARAM.REC.    Bailing out..."
 RESUME fini


DiskWriteError:
 PRINT "Error on Writing to disk: "; ERR
 IF P = 4 THEN PRINT "Hard Disk Not responding or full"
 IF P = 5 THEN PRINT "Diskette Not ready or full"
 RESUME fini

fini:
' Debug--
COLOR 14, 4: INPUT ; "      Hit Return to Stop or ^C to Break", X$: COLOR 7, 1
 CLOSE : END

BadOpenDataFile:
PRINT "Open error = "; ERR
SELECT CASE ERR
  CASE 71
       PRINT "Diskette drive B: is not ready; either there is no disk ";
       PRINT "or it isn't properly inserted."
  CASE ELSE
       PRINT "There is some kind of error with Diskette Drive B:"
END SELECT
PRINT "Hit any key to continue": X$ = INPUT$(1)
RESUME OpenDataFile
```

SUBROUTINES AND FUNCTIONS:

```
REM $STATIC
FUNCTION GetAns$ (Prompt$, Default$)
' If there is no default, an answer is not required.  Prompt$ and Default$
' added in Version 6x4
'RCB 10Mar93
 COLOR 7, 1
 PRINT Prompt$;
 IF LEN(Default$) > 0 THEN 'answer is required or you get the default
      PRINT "(Default: "; Default$; ")   ";
      INPUT ; "", Entry$: PRINT
      IF LEN(Entry$) = 0 THEN
            Entry$ = Default$
      END IF
 ELSE  'Entry optional
      INPUT ; "", Entry$: PRINT
 END IF
 GetAns$ = Entry$
END FUNCTION

SUB GetInfo (OpName$, FilAmps$, FilVolts$, SubmitName$, Comment$)
  PRINT " Please Enter:"
       OpName$ = GetAns$("   Your initials or last name: ", OpName$)
      FilAmps$ = GetAns$("        The Filament Amperage: ", FilAmps$)
     FilVolts$ = GetAns$("         The Filament Voltage: ", FilVolts$)
 SubmitName$ = GetAns$("          The Submitters Name: ", SubmitName$)
      Comment$ = GetAns$("Comment: ", Comment$)
 PRINT :
END SUB

FUNCTION GetInt
GetInt = CINT(VAL(GetStr12$))
END FUNCTION

SUB GetSampName (SampName$)

GetSampName:
 SampMsg$ = " Please enter a unique Sample ID ( up to 8 Char): "
 DO
      SampName$ = GetAns$(SampMsg$, SampName$)
 LOOP UNTIL LEN(SampName$) > 0
 SampName$ = LEFT$(SampName$, 8)
 PRINT
END SUB

DEFSNG A-Z
FUNCTION GetSingle
GetSingle = VAL(GetStr12$)
END FUNCTION

DEFINT A-Z
FUNCTION GetStr12$
LINE INPUT #6, Lin$
X$ = LTRIM$(LEFT$(Lin$, 12))
GetStr12$ = X$
END FUNCTION
```

```
 FUNCTION Interpol8# (M() AS MeasRecord, SampIndex%, RefIndex%, MeasIndex%)
' M will be a 2 dimensional array (typically Meas(J,I) ). J is the index
' to the Meas array.  SampIndex and RefIndex are the sample and reference
' isotope indices (typically RefIndex = 1 and SampIndex = 2 or 3).
' The measurements are taken in the sequence Ref.. Samp.. Ref.. Samp....
' DelT is a weighting function based on the times of measurement for a
' Sample Isotope and its bracketing Reference Isotopes.  It determines
' Interpol8, the linearly interpolated value for the Ref. Isotope at the
' time the Sample Isotope was measured.  Note that calling this function with
' Sample and Refrence roles reversed finds an interpolated value of the
' sample at the time the reference was measured
'
 DIM DelT AS SINGLE
 SI = SampIndex%: R = RefIndex%:
 JS = MeasIndex%  ' Set index for the sample
 JR = MeasIndex%  ' Set index for the reference
' The "JR" subscript for the reference must be adjusted to
' bracket the sample in the time sequence. Time(JR) < Time(JS) < Time(JR+1)
 DO UNTIL M(JR, R).Tick < M(JS, SI).Tick
      JR = JR - 1
 LOOP
 DelT = (M(JS, SI).Tick - M(JR, R).Tick) / (M(JR + 1, R).Tick - M(JR, R).Tick)
 Interpol8# = (M(JR + 1, R).Avrg * DelT + M(JR, R).Avrg * (1 - DelT))
' LPRINT " Samp   Ref    JSamp  JRef    Times/Averages @ R Interp and R+1"
' LPRINT USING " ##    "; SI; R; JS; JR;
' LPRINT USING " #.#####      "; M(JR, R).Tick; M(JS, SI).Tick; M(JR + 1, R).Tick
' LPRINT USING " ##    "; SI; R; JS; JR;
' LPRINT USING " #.#####      "; M(JR, R).Avrg; (M(JR + 1, R).Avrg * DelT + M(JR, R).Avrg
* (1 - DelT)); M(JR + 1, R).Avrg
' LPRINT
 END FUNCTION

FUNCTION MaxOf# (A#, B#)
 IF A# >= B# THEN
      MaxOf = A#
 ELSE
      MaxOf = B#
 END IF
END FUNCTION

SUB Measure (SigmaTol!, MaxStDev!, Reads, Mean#, Sigma#, Time!, GoodReads)
' Calling routine supplies SigmaTol! MaxStdDev!, Reads
' Measure returns mean#, Sigma#, Time! and Good Readings
' RejLevel is a multiple of Sigma# but limited to a maximum fraction of Mean#
 DIM Sum AS DOUBLE, SumOfSqs AS DOUBLE
 REDIM Sample(Reads) AS DOUBLE

 CALL ReadDVM(Mean#)              'Dummy read to syncronize with DVM
 FOR J = 1 TO Reads
      CALL ReadDVM(Sample(J))
      Sum = Sum + Sample(J)
      SumOfSqs = SumOfSqs + (Sample(J)) ^ 2
 NEXT J
 Time! = TIMER
 Mean# = Sum / Reads
 Sigma# = SQR(SumOfSqs / Reads - (Mean# ^ 2))
 RejectLevel# = SigmaTol! * Sigma#     ' a multiple of Sigma
 MaxRejLevel# = MaxStDev! * Mean#      ' a fraction of Mean
```

```
 IF RejectLevel# > MaxRejLevel# THEN
'     PRINT "MEASURE: WARNING! Sigma exceeds ";SigmaTol!; * Max. Reject Level"
'     PRINT #3, "MEASURE: WARNING! Sigma exceeds ";SigmaTol!; * Max. Reject Level"
       RejectLevel# = MaxRejLevel#
 END IF
 GoodReads = Reads
 FOR J = 1 TO Reads
      IF (ABS(Sample(J) - Mean#) > RejectLevel#) THEN
            Sum = Sum - Sample(J)
            GoodReads = GoodReads - 1
      END IF
 NEXT J
 IF GoodReads > 0 THEN
       Mean# = Sum / GoodReads
 ELSE
       Mean# = 0:
 END IF
 EXIT SUB
 END SUB

SUB MoveTo (Gauss!, Range, Delay)
'Rev to return cursor to its col,row on input. all printing is on row 25
'no CRs allowed so screen will not roll up unless input row is 25, then
'it rolls up
Finish! = TIMER + Delay
CALL OutputGauss(Gauss!, 7)        ' Range 7 to avoid saturating the PA
C = CSRLIN
R = POS(0)
DO WHILE R > 24
 PRINT : R = POS(0)
LOOP
LOCATE 25, 1: COLOR 7, 1: PRINT SPACE$(79);
LOCATE 25, 1: PRINT "     Changing the Magnet    ";
P = POS(0) - 1: COLOR 1, 7
DO UNTIL (Finish! < TIMER)
      LOCATE 25, P: PRINT "|"; : LOCATE 25, P: PRINT "|";
      LOCATE 25, P: PRINT "/"; : LOCATE 25, P: PRINT "/";
      LOCATE 25, P: PRINT "-"; : LOCATE 25, P: PRINT "-";
      LOCATE 25, P: PRINT "\"; : LOCATE 25, P: PRINT "\";
LOOP
COLOR 7, 1
CALL OutputGauss(Gauss!, Range)    ' Changes range only
LOCATE 25, 1: COLOR 7, 1: PRINT SPACE$(79);
LOCATE C, R
END SUB

 SUB OutputGauss (Gauss!, Range)
' Contains IEEE-488 interface specific calls
' Transmits 4 bytes to GaussDevice; computation of array G() follows
' NBS Prgram.  Output is in BCD nibbles except for first byte in sequence:
' Braces indicate G() array values as 16 bit integers
' [0  Range], [1st BCD  2nd BCD], [3rd BCD 4th BCD], [5th BCD LOAD]
' Range is 1 to 8 (not 0 to 7 as in the rest of the program)
' LOAD is the load control byte, 14; 10 Kilogauss overrange is not used
' To use it increase LOAD to 15.
'
 STATIC InitFlag AS INTEGER, OldGauss AS SINGLE, OldRange AS INTEGER
 STATIC GaussDevice AS INTEGER
 IF Gauss! = OldGauss AND OldRange = Range THEN  ' we are already there
       EXIT SUB
 END IF
```

```
 OldGauss = Gauss!                    ' otherwise save the new ones
 OldRange = Range
 REDIM G(4) AS INTEGER

InitIt:
 IF InitFlag = 0 THEN
      GaussName$ = "GAUSSM"
'  GaussDevice = 8 is the usual IEEE-488 address of the Gaussmeter
      CALL IBFIND(GaussName$, GaussDevice)
      IF GaussDevice < 0 THEN
            PRINT "Can't Find Gaussmeter on IEEE-488 Buss. Bailing out..."
            ON ERROR GOTO fini
            ERROR 254
      END IF
      MASK = &H8000                  ' error condition
      CALL IBTRAP(MASK, 3)           ' sets up applications monitor for mode 3
                                                          ' error trapping
      CALL IBCLR(GaussDevice)        ' clear Gaussmeter
      InitFlag = 1                   ' should not need to do this again
 END IF                             ' now we are ready


 IF Gauss! < .1 OR Gauss! >= 9999.95 OR Range < 1 OR Range > 7 THEN
      ON ERROR GOTO fini
      PRINT "Error in OutputGauss: Gauss = "; Gauss!; " Range = "; Range
      ERROR 253
 END IF

' Form a string 0rggggg0 where r is the output range and n is a digit from
' the gauss readings, without decimal point. Convert string to 8 BCD digits
' (4 bytes). Then add hex 'E', Load Control trigger for the Gauss controller

 Gauss$ = STR$(FIX(Range * 100000! + 10 * Gauss!)) + "0"

 FOR I = 1 TO 7 STEP 2
  G(I \ 2) = &H10 * VAL(MID$(Gauss$, I, 1)) + VAL(MID$(Gauss$, I + 1, 1))
 NEXT I
 G(3) = G(3) + &HE  'adds trigger character
 Gauss$ = ""
 FOR I = 0 TO 3
      Gauss$ = Gauss$ + CHR$(G(I))
 NEXT I
 CALL IBWRT(GaussDevice, Gauss$)        ' Send the bytes to the Gaussmeter
 END SUB

 SUB RatioStats (Ratio() AS DOUBLE, Res() AS ResultRecord, I, J, T)
'
' All definitions are as in main program.  Logic is:
'      Sum = Sum + Value
'      SqSum = SqSum + Value ^ 2
'      Number = Number + 1
'      Average = Sum/Number
'      StdDev = SquareRoot(SqSum/Number - Average^2)
' Note that the combined stats (X=0) and the individual stats (X = T)
' are updated during each call
 IF Ratio(J, I, T) = 0 THEN
       LPRINT " DEBUG RatioStats: was called with a zero value for Ratio"
       LPRINT " Set Index "; J; " Isotope Index "; I; " and Ratio type "; T
       STOP  ' this shoudn't happen
 END IF
```

```
 FOR X = 0 TO T STEP T
      Res(I, X).SumRatio = Res(I, X).SumRatio + Ratio(J, I, T)
      IF Ratio(J, I, T) > 0 THEN
            Res(I, X).SqSumRatio = Res(I, X).SqSumRatio + Ratio(J, I, T) ^ 2
            Res(I, X).GoodRatios = Res(I, X).GoodRatios + 1
      ELSE
            Res(I, X).SqSumRatio = Res(I, X).SqSumRatio - Ratio(J, I, T) ^ 2
            Res(I, X).GoodRatios = Res(I, X).GoodRatios - 1
      END IF
      IF Res(I, X).GoodRatios = 0 THEN
            LPRINT "DEBUG RatioStats: Number of Ratios is zero for:"
            LPRINT " Set Index "; J; " Isotope Index "; I; " and Ratio type "; X
            STOP  ' this shoudn't happen
      END IF
      Res(I, X).RatioAv = Res(I, X).SumRatio / Res(I, X).GoodRatios
      SigSq# = Res(I, X).SqSumRatio / Res(I, X).GoodRatios
      SigSq# = SigSq# - Res(I, X).RatioAv ^ 2
  IF SigSq# < .000000000000001# THEN SigSq# = 0
      Res(I, X).RatioDev = SQR(SigSq#)
 NEXT X
'
 END SUB

SUB ReadDVM (VmRead#)
' Last change 3May93 RCB: use VmRead# as a flag on initialization so 10 or
' 100 Hz can be chosen on the fly. (10 is default). If the routine is
' called without previous initialization, it defaults
 STATIC InitFlag AS INTEGER
 STATIC DvmDevice AS INTEGER
 DIM DvmInput AS STRING * 14
 IF VmRead# < -1# THEN InitFlag = 0    ' Force reinitialization
InitDVM:
 IF InitFlag = 0 THEN
      DVMname$ = "HP-DVM"
'  DVMdevice = 22  '22 is the usual IEEE-488 address of the DVM
      CALL IBFIND(DVMname$, DvmDevice)
      IF DvmDevice < 0 THEN
            PRINT "Can't Find VoltMeter on IEEE-488 Buss. Bailing out..."
            ON ERROR GOTO fini
            ERROR 254
      END IF
      IF CINT(ABS(VmRead#)) = 100 THEN
            DVMcommand$ = "H6STG100STIF1R4Z0"  'Setup command from prev. program
      ELSE
            DVMcommand$ = "H6STGW10STIF1R4Z0"  'Setup command for testing
      END IF
      Stat = ILWRT(DvmDevice, DVMcommand$, 17) ' Send command to DVM

'  CONST EEND=    &H2000  ' EOI or EOS detected
'  CONST CMPL=    &H100   ' I/O completed
      IF (Stat <> CMPL) THEN
            PRINT "Bad Status ("; HEX$(Stat); ") on HP DVM.  Bailing out....."
            ON ERROR GOTO fini
            ERROR 251
      END IF
      InitFlag = 1                  ' should not need to do this again
 END IF                            ' now we are ready
 Stat = ILRD(DvmDevice, DvmInput, 14)
```

```
 IF (Stat <> EEND + CMPL) THEN
       PRINT "DVM Read error(s); Status is: "; HEX$(Stat); " Err code: "; IBERR;
       PRINT "  Bailing out....."
       ON ERROR GOTO fini
       ERROR 250
 ELSE
      VmRead# = VAL(DvmInput)
 END IF
END SUB

SUB VertMenu (Header$, Menu$(), Size, RowNum, Choice)
' VGA Colors
' Required Inputs: Header$, Menu$(), Size = index of largest valid entry
' in Menu$() and RowNum (Rows from the top of screen) Choice is default
' value to be highlighted on input.
' Menu subscripts start at 0.  Screen is not cleared.
' Menu choices must fit one to a line; up to 20 choices allowed (full screen)
' Returns to caller: Choice (Menu subscript from 0 to Size)
' If key is an up arrow or a down arrow Choice changes and Key$ is reassigned
' The Key$ variable is 2 bytes for arrow keys.   The value in Key$ is
' actually a null plus either "H" or "P".  Capslock is ignored.
' Numlock and Shift affect the numeric keypad as usual.

 Upkey$ = CHR$(0) + "H": Downkey$ = CHR$(0) + "P"   ' Actual values
 up$ = CHR$(24): dn$ = CHR$(25)   ' these print as up and down arrows
 CR$ = CHR$(13)
 COLOR 15, 2
 LOCATE 25, 5: COLOR 15, 6
 PRINT "Select your entry with the "; up$; dn$; " keys, then hit 'Enter'";
 COLOR 15, 2
 LOCATE RowNum, 5
 PRINT Header$: RowNum = RowNum + 2
 LOCATE RowNum, 5
 FOR I = 0 TO Size
      LOCATE , 5: PRINT Menu$(I)
 NEXT I
 LOCATE RowNum + Choice, 5
 ' Choice = 0
 GOSUB Hilight

WHILE (1)
       DO: Key$ = INKEY$: LOOP WHILE Key$ = ""
       IF (Key$ = Upkey$) THEN
             GOSUB Normal
             Choice = Choice - 1
             IF Choice = -1 THEN Choice = Size
             GOSUB Hilight
       ELSEIF (Key$ = Downkey$) THEN
             GOSUB Normal
             Choice = (Choice + 1) MOD (Size + 1)
             GOSUB Hilight
       ELSEIF Key$ = CR$ THEN
            COLOR 7, 1
            GOTO ArrowFini
       ELSE
            LOCATE 24, 5: COLOR 14, 4
            BEEP:
PRINT "Sorry, I don't understand that!  Hit 'Q' to quit; any other key to continue";
            X$ = INPUT$(1)
```

```
            IF UCASE$(X$) = "Q" THEN
             ON ERROR GOTO fini
             ERROR 245
            END IF
            LOCATE 24, 5: COLOR 15, 1: PRINT SPACE$(75); : LOCATE 24, 5: COLOR 15, 6
            PRINT "Select your entry with the "; up$; dn$; " keys, then hit 'Enter'";
            GOSUB Hilight
        END IF
WEND
Hilight:
      COLOR 15, 6: LOCATE RowNum + Choice, 5
      PRINT Menu$(Choice);
RETURN
Normal:
      COLOR 15, 2: LOCATE RowNum + Choice, 5
      PRINT Menu$(Choice);
RETURN
ArrowFini:
RowNum = RowNum + Size + 2
'COLOR 7, 1:  CLS
END SUB

FUNCTION YNans (message$)
' this routine returns -1 (true) for yes and 0 (false) for no. The
' <cr> option returns -1 RCB 28Feb92
 PRINT
 DO
......      PRINT message$; " ( y/n or <cr> ): ";
      COLOR 1, 7: PRINT "Y"; : COLOR 7, 1: LOCATE , POS(0) - 1
      A$ = INPUT$(1): PRINT A$;
      IF A$ = CHR$(13) THEN A$ = "Y"
      IF UCASE$(A$) = "Y" THEN
        Ans = -1
      ELSEIF UCASE$(A$) = "N" THEN
        Ans = 0
      ELSEIF UCASE$(A$) = "Q" THEN
             ON ERROR GOTO fini
             PRINT : ERROR 245
      ELSE
        Lp% = CSRLIN: GOSUB WipeIt: LOCATE 23, 5
        TRef! = TIMER + 4: COLOR 14, 4
        PRINT "Sorry, I don't understand that!   Please answer "; : LOCATE 24, 5
        PRINT "'y' or <cr> for 'yes'; 'n' for 'no' or 'q' to quit";
        DO: LOOP WHILE TIMER < TRef!
        GOSUB WipeIt: LOCATE Lp%, 1
        Ans = 1
      END IF
 LOOP UNTIL (Ans < 1)
 Lp% = CSRLIN
 YNans = Ans
 LOCATE 24, 5: PRINT SPACE$(50);
 LOCATE Lp% + 1, 1
 EXIT FUNCTION

WipeIt:
 COLOR 7, 1: FOR W = 23 TO 24
 LOCATE W, 1: PRINT SPACE$(80); : NEXT W
 RETURN

 END FUNCTION
```