



3 4456 0377659 0

# oml

ORNL/TM-12197

**OAK RIDGE  
NATIONAL  
LABORATORY**

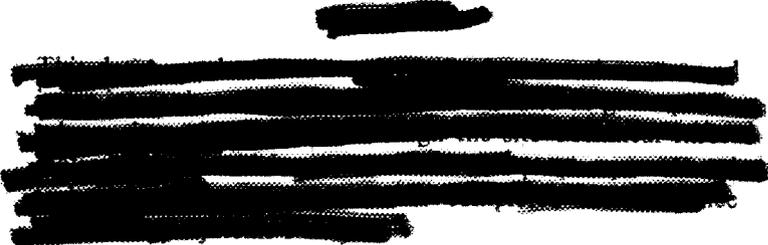
**MARTIN MARIETTA**

## **New FORTRAN Computer Programs To Acquire and Process Isotopic Mass Spectrometric Data**

### **Technical Manual**

OAK RIDGE NATIONAL LABORATORY  
CENTRAL RESEARCH LIBRARY  
CIRCULATION SECTION  
4500N ROOM 1F5  
**LIBRARY LOAN COPY**  
DO NOT TRANSFER TO ANOTHER PERSON  
If you wish someone else to see this  
report, send in name with report and  
the library will arrange a loan.  
RCN7269 (1-6-73)

D. H. Smith  
H. S. McKown



MANAGED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-9401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or approval by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

NEW FORTRAN COMPUTER PROGRAMS  
TO ACQUIRE AND PROCESS  
ISOTOPIC MASS SPECTROMETRIC DATA

Technical Manual

D. H. Smith  
H. S. McKown

*Analytical Chemistry Division*

Date Published: September 1993

OAK RIDGE NATIONAL LABORATORY  
Oak Ridge, Tennessee 37831  
managed by  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract DE-AC05-84OR21400





## CONDENSED TABLE OF CONTENTS

	<u>Page</u>
1. INTRODUCTION .....	1
2. DISK ORGANIZATION AND THE USER INTERFACE .....	7
3. FILES .....	9
4. PROGRAM FILE .....	17
5. PROGRAM MSR .....	37
6. PROGRAM TEST6 .....	49
7. PROGRAMS TRAN AND REC .....	57
8. PROGRAM MSC .....	59
9. PROGRAM REP .....	107
10. MISCELLANEOUS PROGRAMS .....	115
11. DHS_LIB .....	121
12. SCREEN_LIB LIBRARY .....	133
13. DISK_IO LIBRARY .....	141



## TABLE OF CONTENTS

	<u>Page</u>
1. INTRODUCTION	
1.1 Instrument Description .....	1
1.2 Program Overview .....	5
2. DISK ORGANIZATION AND THE USER INTERFACE	
2.1 Disk Organization .....	7
2.2 The User Interface .....	7
3. FILES	
3.1 SCAN File .....	9
3.2 Samp File .....	10
3.3 Chdata File .....	10
3.4 Sumcts File .....	10
3.5 RATIO File .....	12
3.6 SPIKE File .....	12
3.7 STAND File .....	13
3.8 Massab File .....	13
3.9 DTBias File .....	14
3.10 OUT File .....	15
4. PROGRAM FILE	
4.1 Linking File .....	17
4.2 Menu .....	18
4.3 Option 0: Exit .....	20
4.4 Option 1: DTBias .....	20
4.5 Option 2: SCAN .....	21
4.6 Option 3: RATIO .....	22
4.6.1 Ratio_Edit .....	22
4.6.2 New_Ratio .....	22
4.6.3 Edit_Ratio .....	23
4.6.4 List-Ratio .....	23
4.6.5 Move_Ratio .....	23
4.6.6 Delete_Ratio .....	23
4.7 Option 4: MASSAB .....	23
4.7.1 Subroutine Find_El .....	24
4.7.2 Subroutine New_Entry .....	24
4.7.3 Subroutine Display_El .....	24
4.7.4 Executing Massab .....	24
4.7.4.1 Option 0: Stop .....	24
4.7.4.2 Option 1: Edit or Inspect .....	25
4.7.4.3 Option 2: List .....	25
4.7.4.4 Option 3: Delete .....	25
4.7.4.5 Option 4: New Entry .....	25

## Table of Contents - Continued

	<u>Page</u>
4.8 Option 5: SPIKE .....	25
4.8.1 Option 0: Return .....	26
4.8.2 Option 1: New Entry .....	26
4.8.3 Option 2: Edit or Inspect .....	26
4.8.4 Option 3: List .....	26
4.8.5 Option 4: Delete .....	26
4.8.6 Option 5: Update Pu-947 .....	26
4.9 Option 6: STAND .....	26
4.9.1 Option 0: Return .....	27
4.9.2 Option 1: New Entry .....	27
4.9.3 Option 2: Edit or Inspect .....	28
4.9.4 Option 3: List Labels .....	28
4.9.5 Option 4: List Entire File .....	28
4.9.6 Option 5: Edit Last Record Used .....	28
4.10 Option 7: SAMP .....	28
4.10.1 Samp_Edit Menu .....	30
4.10.2 Option 0: Return .....	30
4.10.3 Option 1: List All Variables .....	30
4.10.4 Option 2: List One Subset .....	31
4.10.5 Option 3: Change One Variable .....	31
4.10.6 Option 4: Bookkeeping Record .....	31
4.11 Option 8: Raw Data .....	32
4.12 Option 9: OUT .....	32
4.12.1 Option 0: Return .....	33
4.12.2 Option 1: Edit or Inspect .....	33
4.12.3 Option 2: Enter Companion Information .....	33
4.12.4 Option 3: Permanent Companions .....	34
4.12.5 Option 4: List Permanent Companions .....	34
4.12.6 Option 5: Correct NBS-947 .....	34
4.12.7 Option 6: Change Last Reord Used .....	34
4.12.8 Option 7: Shift Data .....	34
4.12.9 Option 8: Access BKOUT .....	35
5. PROGRAM MSR	
5.1 Details of MSR .....	37
5.2 Setup .....	39
5.3 Msop .....	40
5.3.1 Analysis Sequence .....	43
5.3.2 Execution of Msop .....	44
6. PROGRAM TEST6	
6.1 Menu .....	49
6.2 Option 1: Control Panel Switches .....	49
6.3 Option 2: Read Ion Current Register .....	50

## Table of Contents - Continued

	<u>Page</u>
6.4 Option 3: Test Digital Read-outs on Control Panel and Sweep Output . . .	50
6.5 Option 4: Test Oscilloscope Display . . . . .	51
6.6 Option 5: Test Output of Sweep Amplifier . . . . .	51
6.7 Option 6: Test the Interrupt . . . . .	51
6.8 Using the External Source . . . . .	51
6.9 Option 7: Test Ion Current Input Circuit and Timer . . . . .	54
6.10 Option 8: Test Ion Current Input Circuit, Time, Delay, and Interrupt . . . . .	54
6.11 Option 9: Full Test of Input Scaler . . . . .	54
6.12 Subroutine Test5 . . . . .	56
7. PROGRAMS TRAN AND REC	
7.1 Batch File Tran . . . . .	57
7.1.1 Batch File Startnet . . . . .	57
7.1.2 Program Tran2 . . . . .	57
7.2 Program Rec . . . . .	58
7.3 In Case of Trouble . . . . .	58
8. PROGRAM MSC	
8.1 Organization . . . . .	59
8.2 Overall Logic . . . . .	61
8.3 Module Msroot . . . . .	63
8.4 Subroutine C1 . . . . .	63
8.4.1 Subroutines in C1 . . . . .	64
8.4.2 Function Get_Option . . . . .	64
8.4.2.1 Option 0 . . . . .	65
8.4.2.2 Option 1 . . . . .	65
8.4.2.3 Option 2 . . . . .	65
8.4.2.4 Option 4 . . . . .	65
8.4.2.5 Options 3, 5, 6, 7 . . . . .	65
8.4.2.6 Option 9 . . . . .	65
8.4.2.7 Option 10 . . . . .	65
8.4.3 Subroutine List_Contents . . . . .	66
8.4.4 Subroutine Get_Samps . . . . .	66
8.4.5 Subroutine Check_Sequence . . . . .	67
8.4.6 Function Get_Printer . . . . .	67
8.5 Subroutine C2 . . . . .	67
8.5.1 Subroutine Init_C2 . . . . .	68
8.5.2 Subroutine Check_Validity . . . . .	68
8.5.3 Subroutine Set_Mix_Calcs . . . . .	68
8.5.3.1 Variable mix . . . . .	69
8.5.3.2 Variable recalc . . . . .	69

## Table of Contents - Continued

	<u>Page</u>
8.5.3.3 Variable pass_nr . . . . .	69
8.5.3.4 Variable imix . . . . .	69
8.5.4 Subroutine Set_Bias_Tau . . . . .	70
8.5.5 Library C2_Lib . . . . .	71
8.6 Subroutine C3a . . . . .	72
8.6.1 Subroutine Init_C3 . . . . .	72
8.6.2 Subroutine Write_Header . . . . .	73
8.6.3 Subroutine Set_Spike_Calcs . . . . .	73
8.6.4 Set_Int_Cal_Calcs . . . . .	73
8.6.5 Subroutine Set_IC_Masses . . . . .	74
8.6.6 Subroutine Set_IC_Ratios . . . . .	75
8.6.7 Subroutine Correct_238_Pu . . . . .	75
8.7 Subroutine C3b . . . . .	75
8.7.1 Subroutine Init_C3b . . . . .	77
8.7.2 Subroutine Read_Chdata . . . . .	77
8.7.3 Subroutine Set_Major_Peaks . . . . .	77
8.7.4 Subroutine Find_Peak_Width . . . . .	78
8.7.5 Calc_Raw_Counts . . . . .	78
8.7.6 Subroutine Apply_Dead_Time_Corr . . . . .	78
8.7.7 Apply_Bias_Corr . . . . .	79
8.7.8 Apply_Int_Cal . . . . .	79
8.7.9 Subroutine Calc_IC_Bias_Factors . . . . .	80
8.7.10 Subroutine Calc_Cts_Per_Sg . . . . .	80
8.7.11 Subroutine List_Counts . . . . .	81
8.8 Subroutine C4 . . . . .	81
8.8.1 C4 Subroutines . . . . .	82
8.8.1.1 Set_Nr_Ratios_Runs . . . . .	82
8.8.1.2 Subroutine Set_Corr_Counts . . . . .	82
8.8.1.3 Subroutine Find_Sgs_For_Masses . . . . .	82
8.8.1.4 Subroutine Apply_Bg_Corr . . . . .	82
8.8.1.5 Subroutine Corr_Pu_For_U . . . . .	83
8.8.1.6 Subroutine Corr_U_For_Pu . . . . .	83
8.8.1.7 Subroutine Calc_Total_Cts_Per_Sg . . . . .	83
8.8.1.8 Subroutine Calc_Diffs . . . . .	84
8.8.1.9 Subroutine Calc_Sums . . . . .	84
8.8.1.10 Subroutine Calc_Ratios . . . . .	84
8.8.2 Library C4_Iso_Diln_Lib . . . . .	85
8.8.2.1 Subroutine Iso_Diln_Calcs . . . . .	85
8.8.2.2 Subroutine Set_ID_Params . . . . .	85
8.8.2.3 Calc_ID_Ratio . . . . .	86
8.8.3 Library C4_Contam_Lib . . . . .	86
8.8.3.1 Subroutine Contam_El_Corr . . . . .	86
8.8.3.2 Set_Contam_Index . . . . .	87
8.8.3.3 Subroutine Find_Contam_Sgs . . . . .	88
8.8.3.4 Subroutine Calc_Theor_Ratios . . . . .	88

## Table of Contents - Continued

	<u>Page</u>
8.8.3.5 Subroutine Apply_Contam_Corr .....	88
8.8.3.6 Subroutine Set_Pct_Array .....	88
8.9 Subroutine C5 .....	88
8.9.1 NBS_500_Calcs .....	89
8.9.1.1 Subroutine Set_NBS_Params .....	90
8.9.1.2 Subroutine Calc_Bias .....	90
8.9.1.3 Subroutine Calc_Tau .....	91
8.9.1.4 Calc_NBS_Counts .....	91
8.9.2 Subroutine Calc_Avgs .....	91
8.9.3 Subroutine Outlier_Test .....	92
8.10 Subroutine C6a .....	92
8.10.1 Subroutine Collect_At_Pcts .....	92
8.10.2 Subroutine Set_Iso_Diln_Ratio .....	92
8.10.3 Subroutine GpG_Quick_List .....	93
8.10.4 Subroutine Print_Header .....	93
8.10.5 Subroutine Print_Results .....	94
8.10.6 Subroutine Calc_Pu_238 .....	94
8.10.7 Subroutine Print_At_Pcts .....	94
8.10.8 Subroutine O_Corr .....	95
8.10.9 Subroutine Print_Corr_Pu .....	95
8.10.10 Subroutine Print_Wgt_Pcts .....	95
8.10.11 Subroutine Find_Exact_Masses .....	95
8.10.12 Subroutine Print_Std .....	96
8.11 Subroutine C6b .....	96
8.11.1 Subroutine Print_Unspiked_Info .....	96
8.11.2 Subroutine Calc_Int_Cal_Orig_Iso .....	96
8.11.3 Subroutine Print_Conc .....	97
8.11.4 Subroutine List_Mist .....	97
8.11.5 Subroutine Print_Spike .....	97
8.11.6 Subroutine Get_Nis .....	97
8.11.7 Library C6b_Blank .....	98
8.11.7.1 Subroutine Apply_Blank_Corr .....	98
8.11.7.2 Subroutine Blank_Iso_Diln_Calc .....	98
8.11.7.3 Subroutine Print_Blank_Info .....	99
8.11.8 Library C6b_NO_Comp .....	99
8.11.8.1 Subroutine NO_Comp_ID_Calc .....	99
8.11.8.2 Subroutine Set_NO_Params .....	100
8.11.8.3 Subroutine Calc_Corr_At_Pcts .....	100
8.11.8.4 Subroutine Print_NO_ID .....	100
8.12 Subroutine C7a .....	100
8.12.1 Subroutine Print_Cal_Std .....	100
8.12.2 Subroutine Fix_Rejected_Runs .....	100
8.12.3 Subroutine Update_Samp .....	101

## Table of Contents - Continued

	<u>Page</u>
8.13 Subroutine C7b .....	101
8.13.1 Subroutine A126 .....	101
8.13.2 Subroutine Fractionation .....	102
8.13.3 Library C7b_Burnup .....	102
8.13.3.1 Subroutine Burnup .....	103
8.13.3.2 Subroutine Correct_Nd_for_Ce .....	103
8.13.3.3 Subroutine Burnup_Calcs .....	103
8.13.4 Subroutine Bayne .....	104
8.13.5 Subroutine RIMS .....	104
8.14 Internal Calibration Calculations .....	104
8.14.1 Description of the Procedure .....	104
8.14.2 Calculation of Bias .....	105
8.14.2.1 Successive Approximation .....	105
8.14.2.2 A Word on Nomenclature .....	105
8.14.2.3 Linear Solution .....	106
8.14.2.4 Quadratic Equation .....	106
9. PROGRAM REP	
9.1 Out_Edit .....	107
9.2 Report1 .....	108
9.2.1 Subroutine Get_Label .....	108
9.2.2 Subroutine Get_Rec_Nrs .....	108
9.2.3 Subroutine Find_Rec_Nrs .....	108
9.2.4 Subroutine Set_Rec .....	109
9.2.5 Function Get_List_Format .....	109
9.2.6 Subroutine Put_Info .....	109
9.2.7 Subroutine Get_Data .....	109
9.3 Subroutine Report2 .....	109
9.3.1 Subroutine Write_Header .....	110
9.3.2 Subroutine Set_At-Pct_Ratios .....	110
9.3.3 Subroutine List_Out .....	110
9.3.4 Subroutine List_Ratios .....	110
9.3.5 Subroutine List_At_Pcts .....	111
9.3.6 Subroutine List_One_Line .....	111
9.3.7 Subroutine Set_Avg_Array .....	111
9.3.8 Subroutine Correct_Pu .....	111
9.3.9 Subroutine List_Names .....	111
9.3.10 Subroutine List_Mist .....	111
9.3.11 Subroutine List_StdS .....	111
9.3.12 Subroutine Calc_Avgs .....	111
9.4 Executing Rep .....	112
9.4.1 Option 0: Exit .....	112
9.4.2 Option 1: Edit Out_dat .....	112
9.4.3 Option 2: Average specified samples .....	112

## Table of Contents - Continued

	<u>Page</u>
9.4.4 Option 3: List Specified Records .....	112
9.4.5 Option 4: List Permanent Companions .....	113
9.4.6 Option 5: Special Standard List .....	113
9.4.7 Option 6: List by Batch Code .....	113
9.4.8 Option 7: List by Element .....	113
9.4.9 Option 8: List Sample Identification Codes .....	113
9.4.10 Option 9: List All Between Limits .....	113
9.4.11 Option 10: Correct Pu .....	113
9.5 Automatic Calculation of Averages .....	113
10. MISCELLANEOUS PROGRAMS	
10.1 Program Atwtpc .....	115
10.2 Program IDA .....	115
10.2.1 Isotope Dilution .....	115
10.2.1.1 The Isotope Dilution Equation .....	115
10.2.1.2 Reverse Isotope Dilution .....	116
10.2.2 Subroutine Read_File_Info .....	118
10.2.3 Subroutine Set_IDA_Params .....	118
10.2.4 Subroutine Read_Manual_Input .....	118
10.2.5 Subroutine IDA_Calc .....	119
10.2.6 Subroutine Print_Results .....	119
10.2.7 Subroutine Read_Mix_Ratios .....	119
10.2.8 Subroutine List_Instructions .....	119
10.3 Program Putz .....	119
11. DHS_LIB LIBRARY	
11.1 Subroutine Add_Blanks (str, size, n) .....	122
11.2 Subroutine Afapct (ix, niso, ab) .....	122
11.3 Subroutine AMax (npt, array, ib, rmax) .....	122
11.4 Subroutine Arrsum (n, arr, sum) .....	122
11.5 Subroutine Atpct (nixo, r, apct, moms, mass) .....	123
11.6 Subroutine Aver (ni, avg, sd, sdm, civ) .....	123
11.7 Subroutine Blank_String (str) .....	123
11.8 Subroutine Check (nd, name, ich) .....	123
11.9 Subroutine Ddat (idat) .....	124
11.10 Subroutine Dec (int, nr) .....	124
11.11 Subroutine Devin (idev) .....	124
11.12 Subroutine Elemental_Input() .....	124
11.13 Subroutine Getich (ich) .....	124
11.14 Subroutine Iaver (ni, ir, avg, sd) .....	124
11.15 Subroutine Inc (int, nr) .....	125
11.16 Subroutine Llsqf (nptk, x, y, slope, yint, sd) .....	125
11.17 Subroutine Look (iel, nis, xmas, mass, ab) .....	125
11.18 Subroutine Max (isz, nar, ib, mx) .....	125

## Table of Contents - Continued

	<u>Page</u>
11.19 Subroutine MaxReal (size, array, locn, max) .....	126
11.20 Subroutine Min (isz, nar, ib, mx) .....	126
11.21 Subroutine MinReal (size, array, locn, min) .....	126
11.22 Function Nday (jdat, idat) .....	126
11.23 Subroutine Niscal (mrat, msum, nis, nrat) .....	127
11.24 Function Non-Zero_Els (array) .....	127
11.25 Function Odd (nr) .....	127
11.26 Subroutine Outlie (npt, rato, avg, sd, sdm, cir, kstar) .....	127
11.27 Subroutine Outloo (samp_id, out_rec, indx) .....	128
11.28 Subroutine Qsort (inx, n, a, index) .....	128
11.29 Subroutine Stdev (ni, r, avg, sd, sdm, cir) .....	129
11.30 Subroutine Swap_Integer (a,b) .....	129
11.31 Subroutine Swap_Real (a,b) .....	130
11.32 Subroutine Swap_String (a,b) .....	130
11.33 Subroutine Tzero (iel, niso, ab, jdat, idat, atw, iday, ato) .....	130
11.34 Function Ucase (string) .....	130
11.35 Subroutine Upmtsm (indx, nr, mch) .....	131
11.36 Subroutine Upout (lsam, nr) .....	131
11.37 Subroutine Wgpct (nix, apct, emas, awsam, wpct) .....	131
11.38 Subroutine Xmass (nix, niso, mass, xmas, exmas) .....	132
<b>12. SCREEN_LIB LIBRARY</b>	
12.1 INTEGER*4 vs. INTEGER*2 .....	134
12.2 Subroutine AnyKey (pbid) .....	134
12.3 Subroutine Create_Pasteboard (pbid, row2, col2) .....	134
12.4 Subroutine Create_Virtual_Display (vdid, size_row2 size_col2,attributes2,title) .....	135
12.5 Subroutine Create_Virtual_Keyboard (kbid) .....	135
12.6 Subroutine Create_Window (pbid, vdid, size_row, size_col, cor_row cor_col, attributes, title) .....	135
12.7 Subroutine Delete_Virtual_Display (vdid) .....	136
12.8 Subroutine Delete_Virtual_Keyboard (kbid) .....	136
12.9 Subroutine Erase_Display (vdid, row_begin2, col_begin2, row_end2, col_end2) .....	136
12.10 Subroutine Erase_Line (vdid, row2, col2) .....	136
12.11 Subroutine Erase_Pasteboard (pbid, vdid) .....	137
12.12 Subroutine Erase_Screen .....	137
12.13 Subroutine Erase_Total_Display (vdid) .....	137
12.14 Function Get_Index (pbid, row, col) .....	137
12.15 Subroutine Get_Input (pbid, vdid, nr_char2 row2, col2, input, text_size2) .....	137
12.16 Function Get_Int (pbid, vdid, row, col) .....	138

## Table of Contents - Continued

	<u>Page</u>
12.17 Function Get_Integer (pbid, title, msg, row, col) .....	138
12.18 Function Get_Real (pbid, vdid, row, col) .....	138
12.19 Function Get_Record (pbid, row, col) .....	138
12.20 Subroutine Mask (pbid, vdid, size_row, size_col, cor_row, cor_col, attributes, title, first_line, nr_prompt, prompt_array) .....	139
12.21 Subroutine Paste_Virtual_Display (pbid, vdid, cor_row2, cor_col2) .....	139
12.22 Subroutine Put_Int (vdid, row, col, int, digits) .....	139
12.23 Function Read_EI_Sym (pbid, vdid, row, col) .....	140
12.24 Subroutine Trim (out_string, in_string, size2) .....	140
12.25 Subroutine Unpaste_Virtual_Display (pbid, vdid) .....	140
12.26 Subroutine Write_Msg (vdid, msg, row2, col2, attributes2) .....	140
 13. DISK_IO LIBRARY	
13.1 IO Subroutines .....	141
13.2 LAST_Subroutines .....	142



## LIST OF TABLES

		<u>Page</u>
3.1	File Description .....	9
3.2	SAMP Structure .....	11
3.3	RATIO Structure .....	12
3.4	SPIKE Structure .....	13
3.5	STAND Structure .....	13
3.6	MASSAB Structure .....	14
3.7	Mass Spectrometer Numbers .....	14
3.8	DTBIAS Structure .....	14
3.9	OUT Structure .....	15
4.1	Batch File File. COM .....	18
4.2	FILE Menu .....	19
4.3	Subroutines in File_ .....	19
4.4	Bias per Mass and Mass Ranges .....	20
4.5	SCAN Menu .....	21
4.6	Subroutines in Ratio_Edit .....	22
4.7	RATIO Menu .....	22
4.8	Subroutines in Massab_Edit .....	23
4.9	MASSAB Menu .....	24
4.10	SPIKE Menu .....	25
4.11	Subroutines in Stand_Edit .....	27
4.12	STAND Menu .....	27
4.13	Windows for Editing Samp .....	29
4.14	Short Options Menu .....	29
4.15	Subroutines in Samp_Edit .....	30
4.16	SAMP Menu .....	30
4.17	Subroutines in Out_Edit .....	32
4.18	OUT Menu .....	33
4.19	Companion Information .....	34
5.1	Input Screen for MSR .....	39
5.2	Input Screen for MSR with Information Entered .....	41
5.3	Assembly Language Routines .....	42
5.4	MSR Screen at Completion of Data-Taking .....	44
5.5	Next Sample Options .....	44
5.6	Switch Position Codes .....	46
5.7	OUT1 .....	46
5.8	Subgroup Banks .....	47
5.9	OUT3 .....	47
5.10	OUT4 .....	47
5.11	OUT2 .....	48
6.1	Options in Test6 .....	49
6.2	Bit Assignments for Switches .....	50
6.3	Entry Points in Test5 .....	56

## List of Tables - Continued

	<u>Page</u>
7.1	Batch File Tran . . . . . 57
8.1	MSC Subroutine Summary . . . . . 60
8.2	Batch File to Link MSC . . . . . 61
8.3	Subroutines in C1 . . . . . 64
8.4	Menu for MSC in C1 . . . . . 64
8.5	Subroutines in C2 . . . . . 68
8.6	Values of Variables Used in Calculation of Uranium-Plutonium Pairs . . . . . 70
8.7	Subroutines in C3a . . . . . 72
8.8	Nomenclature of Variables Used in Internal Calibration Calculations . . . . . 74
8.9	Subroutines in C3B . . . . . 76
8.10	Subroutines in C4 . . . . . 81
8.11	Subroutines in C4_Contam_Lib . . . . . 85
8.12	Subroutines in C4_Contam_Lib . . . . . 86
8.13	Values of Contam_index . . . . . 87
8.14	Subroutines in C5 . . . . . 89
8.15	Subroutines in NBS_500_Calcs . . . . . 90
8.16	Subroutines in C6a . . . . . 93
8.17	Subroutines in C6b . . . . . 96
8.18	Contents of C6b_Blank . . . . . 98
8.19	Contents of C6b_NO-Comp . . . . . 99
8.20	Subroutines in C7a . . . . . 100
8.21	Special Cases in C7b . . . . . 101
8.22	Subroutines in C7b_Burnup . . . . . 103
8.23	Nd_iso_inx Array . . . . . 103
8.24	Nomenclature Summary . . . . . 105
9.1	Batch File Rep.COM . . . . . 107
9.2	Subroutines in Rep . . . . . 107
9.3	Subroutines in Report1 . . . . . 108
9.4	Definition of List_format . . . . . 109
9.5	Subroutines in Report2 . . . . . 110
9.6	Rep Menu . . . . . 112
10.1	Miscellaneous Programs . . . . . 115
10.2	Subroutines in IDA . . . . . 117
10.3	Structure for IDA . . . . . 118
11.1	DHS-Lib Subroutines . . . . . 121
11.2	Values of indx in Upmtsm . . . . . 131
12.1	Contents of Screen_Lib . . . . . 133
12.2	Values of Attributes . . . . . 135
13.1	Contents of Disk_IO . . . . . 141

## LIST OF FIGURES

1.1	Computer Network .....	2
1.2	Electronic Schematic of Mass Spectrometer Control and Data Acquisition System .....	3
5.1	Control Panel Status Word .....	38
6.1	Video Display Tests .....	52
6.2	Sweep Amplifier Test .....	53
6.3	Input Scaler Tests .....	55



## ABSTRACT

New versions of computer programs have been written to acquire and process isotopic mass spectrometric data. Data acquisition is performed by DEC 316 Workstations (IBM PC clones). There is one workstation for each of our three mass spectrometers. The workstations are linked to a DEC MicroVax via ethernet. Data processing is performed on the host computer.

This manual describes the data system from the programmer's point of view. A companion manual addresses the needs of the analyst.<sup>3</sup> The programs are written in Fortran, using VAX or Microsoft versions as appropriate. This manual holds short descriptions of all subroutines and functions used in data acquisition and processing. When complex enough to warrant it, program logic is explained. Various tables define the values different variables can assume.



## ACKNOWLEDGMENTS

The programs described in this manual would not be in their present condition without input from a number of individuals. First and foremost, Ray Walker and his analytical problems dictated the presence of many of the options in *MSC*. He also checked the validity of many of the calculations. Edd Miller made numerous suggestions helpful in improving the "user-friendliness" of the programs. Charlie Pritchard, John Sites, and Ray Eby all contributed helpful requests that made the programs more versatile. W. H. Christie, who wrote the Section's first versions of the program, was always ready with advice and answers. Last, but not least, Doris Smith typed the manuscript, deciphering my copper-plate handwriting and coping patiently with the multitude of underscores and mysterious variable names. My thanks to all of them.



## PREFACE

This is the third description of ORNL's isotope ratio computer programs to have received treatment as a Technical Memo. The first appeared in 1979, the second in 1982.<sup>1,2</sup> The programs have been extensively rewritten since that time, primarily due to the acquisition of an entirely new data system. A DEC MicroVAX computer was obtained to replace our old PDP 11/23, and our old DEC MINCs have been replaced with DEC 316 microcomputers; these latter are IBM PC clones.

This is one of a pair of new manuals that describe the new programs. This manual was written with for the programmer and contains considerable detail about the code. The companion manual (ORNL/TM-12196) is directed to the analysts who use the programs in their work but who do not make changes in them.

Note that there are two tables of contents. One is short and lists only chapter headings. The second is quite long and includes the heading for each individual section. It is hoped that the latter will serve as an index. The assumption is that the user will almost certainly know which chapter contains the information sought; scanning the section headings for one chapter should not be too onerous a task and should lead one to the right page number.

Variable, program, and file names are in italics. Program and file names are capitalized; variable names are not unless they begin a sentence.



## 1. INTRODUCTION

The history of these programs and their relationship to ORNL's mass spectrometers is covered in this manual's companion.<sup>3</sup> The primary use of the programs that are the subject of the two manuals is to support the three ORNL-built spectrometers that are specifically designed to measure isotopic ratios. These are pulse-counting instruments, which means that the data arrives at the computer as counts--i.e., it is inherently digital, rather than analogue, in nature. The organization of the programs, however, makes it a straightforward job to accommodate analogue data as well; if our VG-354 mass spectrometer is ever interfaced to the DEC MicroVAX, this flexibility will need to be exploited.

An important part of our duties is to provide support for ORNL's stable isotope enrichment program; another important aspect of our work is to support reactor activities. These two combine to require that we be able to analyze any solid element, both naturally occurring and man-made. So far as the authors are aware, we are the only facility in the world performing mass spectrometry on many of the transplutonium elements. From the programmer's point of view, this means building in as much versatility as possible and not limiting what is analytically feasible through shortcomings in the software. This in turn means that a disk-based system is essential and leads, for example, to the creation of a file that holds abundance and mass information for all elements.

There are presently three mass spectrometers linked via Ethernet to our DEC MicroVAX computer. A DEC Station 316+ system is used at each instrument for controlling data acquisition; these computers are IBM PC clones made by Tandy to DEC specifications. Figure 1.1 is a drawing showing how the Ethernet system is configured. The IBM PC is used for program development and is not associated with an instrument.

### 1.1 Instrument Description

A schematic drawing of the controls of a mass spectrometer is given in Figure 1.2. It is important for the programmer to understand how the computer and instrument interact. The major electronic components are shown in the center. Power supplies with the legend "Manual" associated with them have their outputs controlled by the operator.

A mass spectrometer, as the name implies, analyzes a sample by separating it into its constituents on the basis of their atomic masses. There are several ways of doing this; in our instruments it is accomplished by accelerating charged particles (ions) through a magnetic field. The ions are bent at radii that are functions of their mass-to-charge ratios, their energies, and the strength of the magnetic field. Our instruments are designed to operate at constant magnetic field; scanning is accomplished by altering the accelerating potential (i. e. energy) to which the ions are subjected. We work only with singly-charged positive ions; thermal ionization does not provide sufficient energy to remove more than one electron from an atom. Thus the atomic mass being monitored is numerically equal to its mass-to-charge ratio.

To alter mass position while taking data, the high voltage (often about 8 kV) is adjusted in accordance with instructions sent to it from the computer. The accelerating voltage the ions experience is the algebraic sum of the output of the ion source high voltage

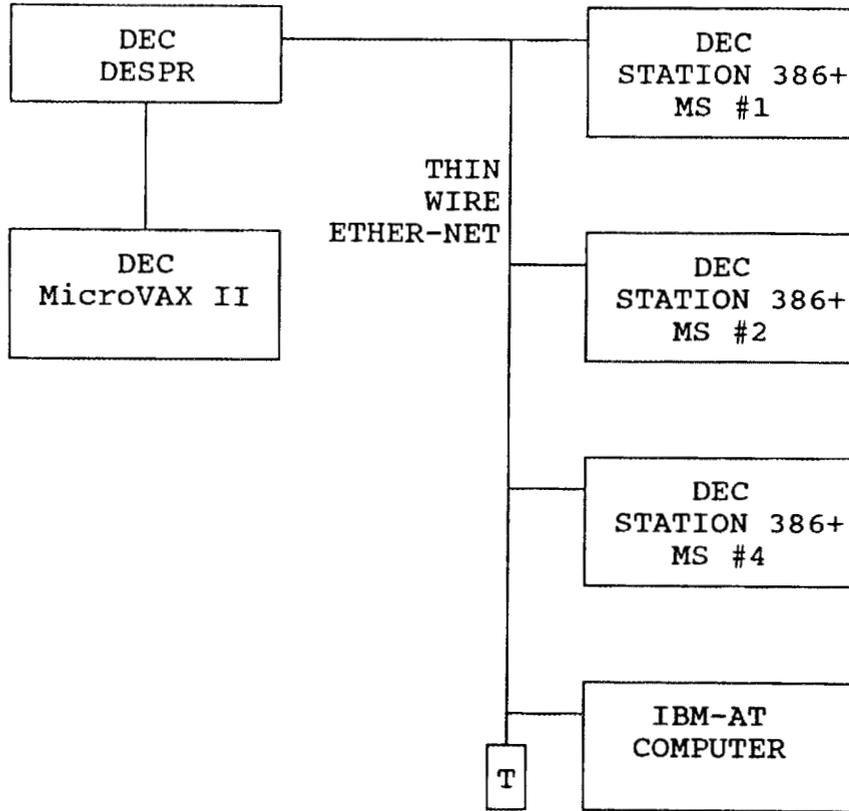
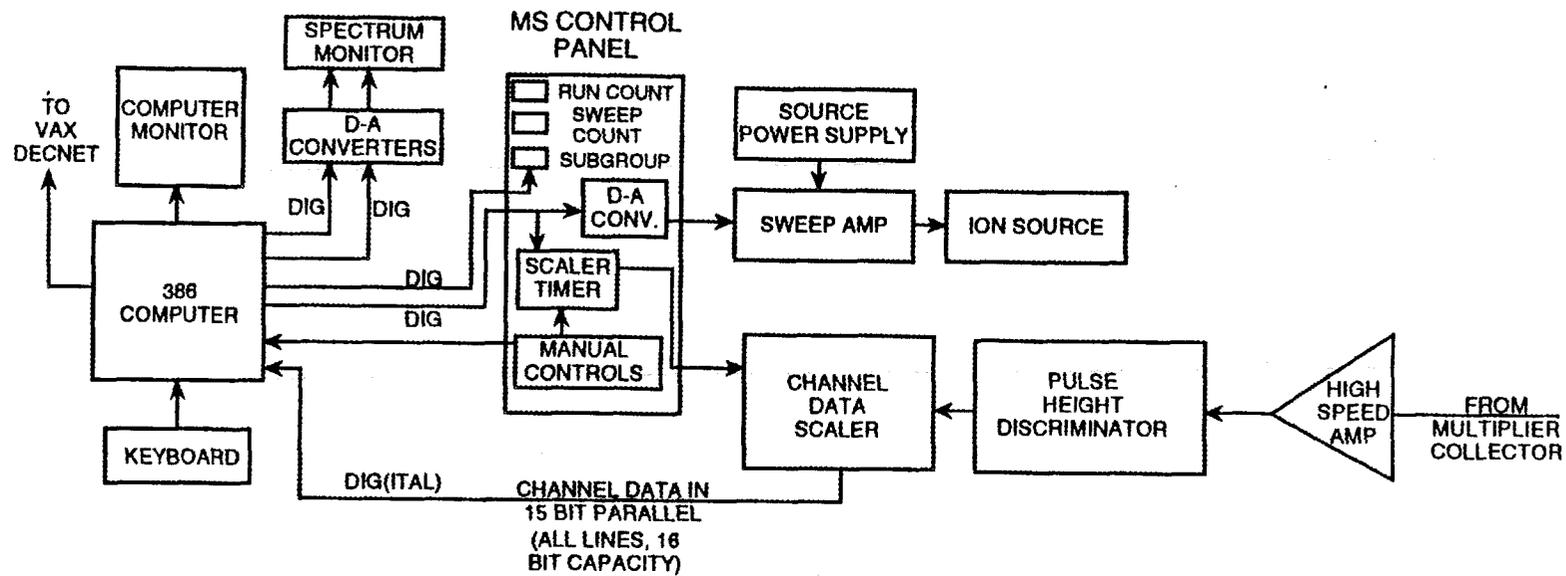


Fig. 1.1 Computer Network



3

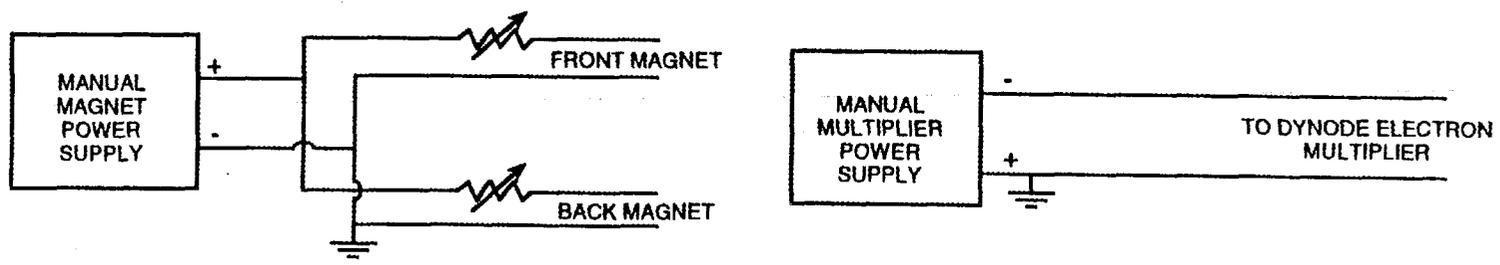


Fig. 1.2 Electronic Schematic of Mass Spectrometer Control and Data Acquisition System

power supply and that of the sweep amplifier. Because we want to sweep up in mass, the output of the sweep amplifier is negative and becomes increasingly more so as a scan toward increasing mass progresses. The masses to be scanned are chosen by the operator by selection of the proper voltage windows with the potentiometers on the lower part of the Sweep Control Panel; this is presently a manual operation.

We need to define some terms. An analysis is usually comprised of 10 runs; each run is comprised of a number of cycles across the mass spectrum. The number of cycles is constant for a given analysis but will vary from one kind of analysis to another. It is usually chosen to make the total time per run about 90 seconds. Each cycle is in turn comprised of scanning the tops of a number of mass positions chosen by the analyst in advance. Sixteen mass positions is the maximum that can be monitored in a single analysis. Any mass position can be swept more than once per cycle; this provides the ability to improve the statistics on the measurement of a minor isotope relative to those on a major.

The computer controls most of the other parameters affecting the scan; it pauses at each mass position to take 32 data points for the top of each peak, the voltage being incremented in small steps across it. It is the operator's responsibility to have the gain of the sweep amplifier and the locations of the potentiometers properly set. The counts for each point are stored in a data array; since each sweep exactly replicates those preceding it, the sum that builds up for each point is the total number of counts (ions) registered in it. The sum of the 32 channels for each mass position is thus the total number of counts collected for the isotope in question, which, after suitable doctoring, can be used to calculate isotope ratios--the goal of the analysis. These ion counts are measured by a pulse-counting system; each ion is registered as a count rather than being integrated into an overall current (as a Faraday cup does, for example). The ion impinging on the first dynode of the electron multiplier detector generates several secondary electrons. These electrons are directed toward the second dynode where they produce more secondary electrons, which are accelerated toward the next dynode, and so on. The entire dynode chain serves to amplify the signal due to the original single ion by a factor of  $10^6$  -  $10^8$ , generating a single pulse in the process. This pulse is passed to a preamplifier that has a gain of about 30; it then proceeds to a discriminator. This device serves to remove background noise from the signal through use of a discriminator setting. Only pulses higher than this setting are passed on to the scaler. Counts are accumulated in the scaler for a length of time controlled by the computer. At the expiration of this time, the counts stored in the scaler are transferred to the computer and added to the number of counts already in that channel; this updates the data array that is the mass spectrum. The scaler is reset to zero and the accelerating voltage stepped to its next value.

An analogue rate meter displays the number of ions being registered at any given time. It is not involved directly with data acquisition and is used by operators to help optimize focus. The data array of counts is continuously displayed on a spectrum monitor. This is a real-time display of the analysis and is invaluable to the operator monitoring its progress.

The Sweep Control Panel is the heart of the data system. Through it the operator can change the mode of operation (basically one for set-up and focusing and another for data taking); the sweep rate can be varied; and, as mentioned above, the 16 voltage windows (called subgroups) can be set to their desired locations. Several digital displays keep the operator apprised of the status of the current analysis. A special computer program is used in trouble-shooting this panel; it is called *Test6* and is described in Chapter 6.

## 1.2 Program Overview

There are several programs associated with operation of the mass spectrometers and treating the data they generate; data-taking operations are confined to the PC clones interfaced to the mass spectrometers, and data-reduction occurs on the MicroVAX. One program, called *MSR* (for Mass Spec Run) is used to interact with the instrument and to take data; it resides on the PC's. A second, *MSC* (for Mass Spec Calculate), processes the data and prints out a report of the results. Program *Rep* (for Report) is used to access the file containing results and print out a condensed version of them. Both *MSC* and *Rep* reside only on the VAX; there is no PC implementation of either. Another program, called *File*, is used to access the various files for entry of new information or revision of old; there are versions both on the VAX and on the PC's. There are programs to transfer data from the work stations to the VAX (*Tran* and *Rec*) and to test the Sweep Control Panel (*Test6*); there are several other programs of lesser importance whose descriptions are included in Chapter 10. Several modules, including libraries, containing generally useful routines, will also be described.

Program *MSR* and other PC-resident programs are written primarily in Microsoft Fortran; a few subroutines and functions that interact with the instrument are in assembly language. An attempt was made to minimize assembly language programming. The other programs are written in DEC's VAX Fortran-77. Considerable use has been made of DEC extensions of the language, which of course reduces their portability. This matter is unimportant now, but may become so if and when the MicroVAX is replaced. Structures (records) are used extensively, for example.

A few general comments need to be made. File and subroutine names in the text are capitalized and italicized; variable names are in italics but not capitalized. So far as possible, variable names in the programs on the VAX have been chosen to reflect unambiguously what is being represented. For example, the weight of the spike added to a sample in an isotope dilution analysis is called *spike\_wgt*. The underscore is liberally used to set off words in a multiple word variable name--as in *spike\_wgt*, *first\_run*, etc. This often leads to variables and subroutines whose names are of cumbersome length, but reading and understanding the program are vastly easier (even for the author). *Spike\_wgt* means a lot more to the unfamiliar reader than something like *sw*. Single-character variables are always DO loop index counters and of only local importance.

A logical extension of using meaningful variable names was to apply the name of the file to the structure that holds data from it. Such a structure usually defines the contents of one record in the file; multiple records are defined as arrays of the fundamental structure. A record in the *Spike* file, for example, would have the structure variable name *spike* on the

VAX. In this way, variables to be associated with a similar parameter can readily be distinguished. As an example, let us consider "mass." Since these programs deal with mass spectrometry, there are many times when one wants to use "mass" as a variable name. In an isotope dilution experiment, there are masses associated with the spike, the unspiked sample, and the mixture whose data are being processed. Use of structures allows these various masses to be identified in a straightforward way without ambiguity. For example, *samp.mass* and *spike.mass* refer to the masses associated with the sample at hand (the mixture in isotope dilution cases) and the spike, respectively.

It will be noticed that some of the variable names hark back to archaic terminology. The first versions of these programs were written to process data taken by multi-channel analyzers, and some residue of that technology is still to be found in the programs. For example, we still refer to each data point as a channel; we still use the term "subgroups" when something like "mass position" would be more appropriate today; and so on.

Structured programming principles are adhered to in general, but not with fanatic devotion--GO TO statements are used when convenient. On the other hand, every effort was made to avoid the infamous "spaghetti" code, where a complex sequence of GO TO statements can confuse the most experienced programmer. All variable names are typed--the IMPLICIT feature of Fortran is not used. All integers are typed INTEGER\*2 except those that the system demands be INTEGER\*4; these are mostly variables associated with pasteboards and virtual displays (See Chapter 12). Real numbers are typed REAL\*4. A conscientious effort was made to be generous with comment statements in the listings; one of the features of comments, however, is that there are never enough of them. The modular concept of Modula-2 can be observed in the programs, especially in the code written toward the end of the project when the author was more at ease with the practice; this was adopted because of the inherent ease of understanding the logic of the program. Fortran is not a language designed graciously to comply with the modular concept (not being able, for example, to import structures defined in another entity of code), but use of INCLUDE statements helps reduce redundant code in the listings.

Finally, a disclaimer. Neither of the authors is a professional programmer. We program as the job demands, and each of us has many other demands on his time. We do not claim these programs are exemplars of the art of programming; there are without doubt ways to make them more elegant or faster or more efficient in other ways. We only claim that they work; they have been in use for 15 years now with no major changes of approach, which implies we have the big picture right.

## 2. DISK ORGANIZATION AND THE USER INTERFACE

### 2.1 Disk Organization

All programs located on the VAX and executed by the analysts running the instruments are located in the [MSCALC] directory of the large hard drive on the VAX. All data files accessed by the programs are also located there. Source code, object files, and versions of the programs compiled and linked for debugging are stored in various subdirectories of either the HSM or the DHS directories. HSM holds *Tran* and *Rec*. DHS holds *MSC* modules (in the [DHS.CALC.NEW] subdirectory); *File* and *Rep* modules (in the [DHS.FILE] subdirectory); and various others (in the [DHS.MISC] directory). To edit any of these modules, it is most convenient to move to the subdirectory in question before invoking the editor.

### 2.2 The User Interface

While no single user interface was written for these programs, a serious attempt was made to give all the programs a consistent look to the user. This applies to all VAX programs and most PC programs. The same approach was used here as the one developed for the DEC PDP-11/23.<sup>4</sup> The interface will be described in general terms here; the companion manual is liberally sprinkled with specific examples.<sup>3</sup>

The main idea underlying the approach is to present information in a display that is easy to read and to give easy access to whatever datum needs to be entered or changed. This requires the ability to move the cursor to the relevant area of the screen, which in turn requires use of DEC's pasteboards and virtual displays. These are roughly equivalent to the screens and windows, respectively, of other computer systems--leave it to DEC to be different! The pasteboard is the framework upon which a specific display is hung; it is the display that holds the data being shown.

Using such an approach is cumbersome. Only string information can be read from, or listed to, the display, necessitating conversion of numerical information to and from that form. A library of routines, *Screen\_Lib*, has been written to facilitate use of pasteboards and displays; it is described in Chapter 12.

Editing a file is usually driven by one of two operations: making a new entry or correcting an old one. When making a new entry, the necessary input is automatically called for in the sequence displayed on the screen; the analyst need only enter the requested information. When editing an existing record, a means of identifying which datum is to be changed is needed. Since we do not have a mouse, some other means of controlling cursor position on the display is required. The method chosen is to associate an index number with each variable that can be accessed. This number is listed on the display just before the text describing the associated variable and is almost always set off by a period. For example, a line on the screen might read (with no quotation marks) "3. Sample id U1555A." The analyst first enters a value for the index; the value entered is used both to locate the cursor on the screen and to identify which variable is being altered. Each variable has a specific position on the screen associated with it, and it is to this position that the cursor is moved. In the example above, if the analyst wished to change the sample identification code, he would enter a 3 (the

index) in the special window used for this purpose. The index window is then erased, and the cursor moved to the first character of the value associated with an index value of 3, in this case the U of U1555A. He would then enter the new value for the sample identification code, terminating input by hitting the RETURN key. The index window reappears and prompts him for a new value for the index. Variables may be changed indefinitely in a single session: A new value for the index is requested until a zero is entered. A zero value indicates that no more editing is desired, and control reverts to the level above the one in which the editing is done.

While this may not be the most elegant interface around, it is simple, easy to use, and easy for analysts to grasp. And it works.

### 3. FILES

Most of the programs access the same data base. This data base is in the form of discrete files that hold the information necessary for correct acquisition and processing of data. The names and a short description of the contents of these files are given in Table 3.1.

---

Table 3.1: File Description

<u>File</u>	<u>Size*</u>	<u>Contents</u>
Scan	100, 30	Scanning schemes for each instrument. (PC only).
Samp	401, 128	Basic information for each sample.
Chdata	1141, 256	Raw channel data.
Sumcts	5701, 20	Condensed raw data.
Ratio	300, 75	Controls calculation of ratios.
Spike	100, 35	Spike information.
Stand	51, 80	Standard information.
Massab	120, 21	Elemental information.
DTBias	5, 11	Dead time and bias information.
Out	3001, 160	Results.

---

\* Size is in terms of maxrec, recordsize in the Fortran OPEN statement used on the VAX; note that recordsize is type longword.

It is important that the programmer understand the purpose of each file and how its contents are used by the programs. Most of these files contain a bookkeeping record as the last one in the file; this is the reason the number of records in many files ends in "1." This record contains the number of the last record used; when a new entry is made, the value of the last record used is incremented by one, and the new information stored in the indicated record.

Access to these files is provided by program *File* (Chapter 4), which allows entry of a new record and editing an old one. Disk operations for most files are in a library, *Disk\_IO* (Chapter 13 ). Short summaries of each file follow.

#### 3.1 SCAN File

The *Scan* file holds the parameters that control data acquisition. It resides only on the three DEC 316's; no purpose would be served to have it on the VAX since that computer plays no role in data collection. The contents and their meaning are described in Section 3.4 of the Operator's Manual.

The *Scan* file is configured to hold 100 records of 120 bytes each.

### 3.2 SAMP File

This file contains information about samples that have been analyzed. It is described in detail in Section 3.9 of the Operator's Manual. The data stored here might be categorized as control information; the raw data of intensities are stored in the *Chdata* file (Section 3.3). Information contained in *Samp* includes the element symbol, the sample identification code, the number of runs, and so on.

A structure is used in *MSC* and program *File* to hold this information. It is called *samp*; in general, the structures containing file information take the same name as the file. Thus, the masses scanned in data taking are referenced by *samp.mass*, as discussed in Chapter 1. The meaning of those parameters controlling programming branches is discussed in Chapter 8, where *MSC* is described. Table 3.2 shows the structure.

*Samp* is configured to hold data for 400 samples with a recordsize of 128 longwords or 512 bytes. The counter in the bookkeeping record (last record used) is reset to 1 when it is incremented past 400.

### 3.3 CHDATA File

This file contains the raw channel-by-channel count data acquired by program *MSR*. It can hold 1141 records with a recordsize of 256 (1024 bytes). The bookkeeping record in this file is set to 1 when 1140 will be exceeded by the data for the sample under consideration. The counts for each step of the data-taking process are stored sequentially as array elements. They are stored as real numbers. Each record can thus contain data for one 256-channel run; this amounts to a maximum of eight mass positions. When more than eight mass positions are swept (maximum is 16), two records in *Chdata* are used to hold the data. Subroutine *Read\_Chdata* in routine *C3b* is the code that reads the data from the file for data processing. They are written to the *Chdata* file in the transfer program (Chapter 7).

Because the contents of *Chdata* are only arrays of real numbers, no structure is associated with it.

### 3.4 SUMCTS File

This file holds condensed raw data. No data processing has been done beyond summing the appropriate number of channels (data points). This vastly reduces the amount of storage required--there is one number per subgroup (mass position) rather than 32.

The file is configured to hold 5701 records of 20 longwords each. The bookkeeping record is updated automatically and reset to 1 when the maximum number of records (5700) will be exceeded. Data for a single sample will not "bend around the corner": i.e., it will not start near the end of the file, come to the end, and start writing in the first record. A test has been inserted in the code to preclude this.

There is no structure associated with this file.

Table 3.2 SAMP Structure

STRUCTURE		/samp_rec/
	INTEGER*2	calc_index
	CHARACTER*2	el_sym
	CHARACTER*6	samp_id,
1		bead_comp_id,
2		unspiked_comp_id
	CHARACTER*20	title
	REAL*4	sweep_rate
	INTEGER*2	high_subgroup,
1		sweep_factor(16),
2		mass(16),
3		last_run,
4		nr_cycles
	REAL*4	spike_wgt
	CHARACTER*20	comments
	INTEGER*2	anal_date(3),
1		cal_std_index,
2		spike_rec,
3		ratio_rec,
4		sumcts_rec,
5		mix_index
	REAL*4	samp_wgt,
1		diln_wgt,
2		aliq_wgt,
3		density
	INTEGER*2	nr_oxygens,
1		iso_dil_ratio(2),
2		chdata_rec,
3		blank_index
	CHARACTER*2	batch_code
	INTEGER*2	cal_std_rec,
1		reverse_index,
2		recalc,
3		dummy,
4		Pu_date(3)
	CHARACTER*6	blank_comp_id
	CHARACTER*2	contam_el_sym(3)
	INTEGER*2	mch_nr,
1		pass_nr,
2		first_run,
3		cycle_vg,
4		nr_run_vg,
5		posn_nr,
6		coll_type,

Table 3.2 - Continued

7	ion_mode,
8	loading_type
REAL*4	amp_gain(5)
END STRUCTURE	

### 3.5 RATIO File

This file dictates a great deal about exactly what calculations will be made; it is fully described in Section 3.5 of the Operator's Manual. The contents of one record are defined as a structure in programs *MSC* and *File*.

It is configured for 300 ratios of 75 longwords each. Its structure is shown in Table 3.3

Table 3.3 RATIO Structure

STRUCTURE	/ratio_rec/
CHARACTER*2	el_sym
INTEGER*2	base_mass,
1	ratio_mass(22,2),
2	sum_mass(3,16),
3	diff_mass(3,16),
4	bgd_mass,
6	norm_mass
END STRUCTURE	

### 3.6 SPIKE File

This file holds information about the spikes used in isotope dilution analysis. This includes the masses, isotopic composition, and the ratios to be used for isotope dilution and internal calibration calculations.

It is configured for 100 spikes of 35 longwords each. Its structure is given in Table 3.4.

---

 Table 3.4 SPIKE Structure

```

STRUCTURE                                /spike_rec/
  CHARACTER*2                             el_sym
  INTEGER*2                               iso_dil_ratio(2),
  1                                       int_cal_ratio(2),
  2                                       mass(10)
  REAL*4                                  at_pct(10),
  1                                       at_wgt
  CHARACTER*40                             label
  INTEGER*2                               int_cal_index
END STRUCTURE

```

---

### 3.7 STAND File

This file holds the isotopic compositions of the standards used for quality control. The most important of these is NIST (formerly NBS) U-500, which is used for calibration purposes.

The file is configured for 50 standards of 80 longwords each. Its structure is shown in Table 3.5.

---

Table 3.5 STAND Structure

```

STRUCTURE                                /stand_rec/
  CHARACTER*2                             el_sym
  CHARACTER*20                             label
  INTEGER*2                               dead_time_ratio(2),
  1                                       bias_ratio(2),
  2                                       ratio_mass(9,2),
  3                                       at_pct_mass(10),
  REAL*4                                  ratio(9),
  1                                       at_pct(10)
END STRUCTURE

```

---

### 3.8 MASSAB File

This file holds the masses of all isotopes and their abundances for naturally occurring elements. It also contains the masses for many man-made isotopes. It is accessed by element symbol rather than record number.

There is room for 120 elements of 21 longwords each. *Massab*'s structure is given in Table 3.6. Note that *exact\_mass* and *abun* are dimensioned (10) to accommodate tin, the element with the most naturally occurring isotopes.

---

Table 3.6 MASSAB Structure

STRUCTURE CHARACTER*2 INTEGER*2 REAL*4 1 END STRUCTURE	/massab_rec/ el_sym nr_isotopes exact_mass(10), abun(10)
---	--

---

### 3.9 DTBIAS File

This file contains the dead time and bias for each mass spectrometer. The instruments have historically had a number assigned to them, and this number is used as the record number in the file that holds the information relevant to them. Table 3.7 defines the instruments.

---

Table 3.7 Mass Spectrometer Numbers

<u>No.</u>	<u>Instrument</u>
1	Three-stage mass spectrometer.
2	Two-stage mass spectrometer for low level work.
3	Inactive; was used for single stage now in 4500S.
4	Two-stage (TRU) mass spectrometer.
5	Inactive; was used briefly for the VG-354.

---

There is room for 5 records of 11 longwords each. The structure is shown in Table 3.8. The use of the four values bias is described in Section 4.3 of this manual and 3.32 of the Operator's Manual. Table 4.3 lists the mass ranges to which each bias is applied. The bias used is a function of the mass of the element undergoing analysis.

---

Table 3.8 DTBIAS Structure

STRUCTURE REAL*4 1 END STRUCTURE	/dtbias_rec/ tau, bias(4)
---	---------------------------------

---

## 3.10 OUT File

This file holds the results of the calculations. Included are sample information, values for ratios and atom percents, and statistical information (standard deviations).

This file will hold results for 3000 samples, with 160 longwords for each sample. When the bookkeeping record exceeds 3000, it is reset to 51 rather than 1; the first 50 records have been designated to hold "permanent companion" information. This area is for samples that are repeatedly analyzed when it is not necessary to make an independent isotopic analysis but to use the known values. Natural uranium is an example.

The structure for OUT is shown in Table 3.9.

---

Table 3.9 OUT Structure

STRUCTURE	<i>/out_rec/</i>
CHARACTER*2	el_sym
CHARACTER*6	samp_id,
1	bead_comp_id,
2	unspiked_comp_id
INTEGER*2	ratio_rec,
1	ratio_mass(22,2),
2	mass(16)
REAL*4	ratio_avg(22),
1	sd_ratio(22)
INTEGER*2	sum_mass(16)
REAL*4	at_pct(10)
1	sd_at_pct(10),
2	at_wgt
CHARACTER*20	title
REAL*4	samp_wgt,
1	diln_wgt,
2	aliq_wgt,
3	density,
4	grams_per_gram,
5	grams_per_liter,
6	spike_wgt
INTEGER*2	mch_nr,
1	int_cal_index
REAL*4	total_cts,
1	total_cts_per_mass(16)
INTEGER*2	nr_runs,
1	anal_date(3)
REAL*4	corr_at_pct(8),
1	bias_per_mass
END STRUCTURE	

---



## 4. PROGRAM FILE

*File* is the name of the program that provides access to the files described briefly in Chapter 3. Through this one program, any of the files may be inspected; new entries and revisions to old ones may also be made. Table 3.1 contains a listing of the files, their sizes, and brief descriptions of their contents. Structures associated with some of the files are given in Chapter 3 and won't be repeated here. The various tasks are accomplished largely through interactions with the user interface described in Chapter 2.

Note that there are two different versions of *File*, one for the VAX and one for the PC's. The PC's have only three files resident: *Scan*, *Samp*, and *Chdata*. Of these, *Samp* and *Chdata* are also on the VAX (much expanded in size); only *Scan* is unique to the PC's. The editing routines on the PC's for *Samp* and *Chdata* are primitive in comparison to those on the VAX. They were written before the days of a standard user interface and do not conform to its conventions. Because they are seldom used (any changes can always be made on the VAX) and because of the effort involved in developing updated versions, no attempt has been made to modernize them. The editing module for *Scan*, on the other hand, has been modernized.

Discussion below is based on the VAX versions (except, of course, for *Scan*).

### 4.1 Linking File

Linking program File is most conveniently accomplished by executing a batch file, *File.COM*. This automatically assigns all the modules required by the linker. A listing is given in Table 4.1.

---

 Table 4.1 Batch File File.COM

```

!      FILE.COM
!      To link program FILE.
!      This program consists of a number of stand-alone modules that provide
!      editing and listing functions for data files.
!      The editing program has an 'edit' suffix affixed to a root that is supposed
!      to identify the .DAT file being accessed.
!link/debug/executable-file file-
link/executable+file file-
+Bias_Tau_Edit-
!      +Scan_Edit-
+Ratio_Edit-
+Massab_Edit-
+Stand_Edit-
+Samp_Edit-
+Raw_Data_Edit-
+Out_Edit-
+[dhs.lib]outloo-
+[dhs.lib]Disk_IO/lib-
+[dhs.lib]DHS_Lib/lib
!      See p. 1-34 in Prog. in VAX Fortran.
$assign/user_mode sys$command sys$input
$r file

```

---

As always on the VAX, a batch file is invoked by preceding the name of the file with @. To execute File.COM, enter @File from the keyboard. You must be in directory [DHS.FILE] when you do this.

#### 4.2 Menu

The menu for File is given in Table 4.2.

---

 Table 4.2 FILE Menu

0	to exit
1	for DTBIAS--dead time and bias
2	for SCAN (PC's only)
3	for RATIO
4	for MASSAB--mass and abundance file
5	for SPIKE
6	for STAND--standard file
7	for SAMP
8	for SUMCTS, CHDATA--raw data files
9	for OUT

---

Each option will be described in turn.

Because of the use of a standard user interface, execution of all these routines is similar. In most cases, a record number is input and its contents displayed on the screen. Each variable whose value can be changed has associated with it an index number that is often set off by a period. Index numbers run consecutively from 1 to however many variables can be edited. To change the value of a desired variable, enter its index number. The cursor will move to the screen location associated with the variable. Enter the new value for the variable, finishing entry by hitting 'RETURN.' A new index number will be called for. The cycle will repeat itself until an index of zero is entered, at which time a new value for a record number is requested. Entering zero for the record number will take you back to the most recent menu. A succession of zero entries will ultimately lead to leaving *File* and return of control to the monitor.

Each file has its own editing subroutine; the subroutine called is determined by the value of the index entered from the menu. The subroutines are listed in Table 4.3. Their names are usually made by appending *\_Edit* to the name of the file.

---

Table 4.3 Subroutines in File

<u>Option</u>	<u>File</u>	<u>Editing Routine</u>
1	DTBias	Bias_Tau_Edit
2	Scan	Scanch
3	Ratio	Ratio_Edit
4	Massab	Massab_Edit
5	Spike	Spike_Edit
6	Stand	Stand_Edit
7	Samp	Samp_Edit
8	Chdata, Sumcts	Raw_Data_Edit
9	Out	Out_Edit

---

All routines draw extensively on the Screen Library (Chapter 12).

#### 4.3 Option 0: Exit

Entering zero for the option means you are done with program *File*. The pasteboard is erased and Call Exit invoked. Control returns to the monitor.

#### 4.4 Option 1: DTBIAS

*Dtbias* is the file that holds dead time and bias values for as many as five mass spectrometers. Instruments are defined by identification numbers as described in Table 3.7. The structure associated with *Dtbias* is listed in Table 3.8.

Records are identified by entering the identification number of the mass spectrometer, cleverly made to equal the record number in the file. This information is given in Table 3.7.

Each instrument has four bias corrections per mass associated with it as defined in Table 4.4.

---

Table 4.4 Bias per Mass and Mass Ranges

<u>Bias</u>	<u>Mass range</u>
1	≤ 100 (or extra bias for resin bead Pu)
2	100-150
3	150-200
4	> 200

---

The four biases were created in an attempt to reduce the bookkeeping required for instrument operation. Even though bias correction per mass can be assumed to be linear over limited mass ranges, it has long been known that it is not linear over larger ranges. Keeping four values of bias allows the analyst to have current values for, say, uranium, neodymium, and strontium at the same time.

This attempt, however, has not met with unqualified success. The first problem arose when it was discovered that Pu run from a resin bead required a significantly different bias than Pu run as an aqueous loading. This was such a common occurrence for many years that an ugly patch was grafted onto the program by using *bias (1)* as the amount of bias to be added to *bias (4)* for Pu loaded on resin beads. To analyze Pu as an aqueous loading, batch code AQ has to be used. This is still the case today.

More recent experience has shown that, at least for some elements, chemical interactions on the filament profoundly affect the bias. Ni, Fe, Cu, and Ti all require radically different biases, making it necessary to access *Dtbias* and change *bias (1)* each time a different element is analyzed. With the 20-20 vision of hindsight, it is clear that a more convenient

means of reading in the bias could have been found. Incorporating an element-specific value in the *Massab* file suggests itself as a better choice.

The logic in *Bias\_Tau\_Edit*, the editing subroutine, is straightforward and should be no challenge to unravel. An example of the display is given in Table 3.4 in the Operator's Manual.

#### 4.5 Option 2: SCAN

*Scanch*, the routine to access the *Scan* file, resides only on the PC's because there is no *Scan* file on the VAX. It is a modern revision of the original, using the PC version of the standard user interface.

The nature of our business leads to many entries in the *Scan* file. The contents of the file will be different for each instrument. It is obvious that each element analyzed requires at least one *Scan* file record. Some commonly analyzed elements, such as uranium, require many records. This multiplicity of records originates from the need to vary sweep factors for various mass positions depending on their abundances and the demands of the analysis. Stable Isotopes, for example, can (and has!) submitted several samples of the same element, with each having a different isotope enriched. Each of these samples requires either a separate record in the *Scan* file or modification of an existing one. By the same token, it is desirable to change the ratios listed during the analysis to reflect the changing major isotope. There is a default option for these ratios, using the major peak as the denominator and the next most intense peaks as the numerators, unless otherwise specified.

Editing *Scan* is a straightforward application of computer logic. The *Scan* menu is given in Table 4.5.

---

Table 4.5 SCAN Menu

0	to return
1	to edit existing record
2	to create a new record
3	to list

---

These options should be largely self-explanatory. When an existing record is being edited, its current contents are listed on the screen and variables modified in the usual manner through use of the edit index. If a new record is to be created, record numbers of empty records are identified and listed to the screen. The analyst chooses a number that appeals to him (it is desirable to keep entries for the same element in contiguous records). The analyst is prompted for a value for each variable in turn. The number of cycles per run (at 2 msec/channel) to make it take 90 seconds is suggested to the analyst. If this value is not the one wanted, the analyst need only change it. At the end of input, the analyst is given the opportunity to correct erroneous entries. An example of a *Scan* record is given in Table 3.5 of the Operator's Manual.

Option 3 allows listing of the file on the printer to produce a hard copy of the contents. There should be a reasonably current copy available at each instrument.

#### 4.6 Option 3: RATIO

The *Ratio* file is accessed by calling the subroutine *Ratio\_Edit*. *Ratio* is the file in which are defined which ratios will be calculated. *Ratio\_Edit* is divided into several subroutines, which are listed in Table 4.6. The names should be self-explanatory.

---

Table 4.6 Subroutines in Ratio\_Edit

New\_Ratio  
 Edit\_Ratio  
 List\_Ratio  
 Move\_Ratio  
 Delete\_Ratio

---

The Ratio structure is listed in Table 3.3.

##### 4.6.1 Ratio\_Edit

*Ratio\_Edit* is a calling routine that functions according to the value for the ratio index entered in response to the menu listed in Table 4.7.

---

Table 4.7 RATIO Menu

0	to return to calling program
1	for a new entry
2	to edit or inspect an existing record
3	to list contents
4	to move a ratio to a different record
5	to delete a ratio

---

These descriptions should be largely self-explanatory. The subroutines involved are described below.

##### 4.6.2 New\_Ratio

This routine is accessed when a new entry is being made. Empty records are listed, and the analyst chooses which to use. The analyst is prompted for a value of each variable in turn. Entry ceases when a value of zero is entered for the numerator of a ratio, a value otherwise meaningless. As always, provision is made for correcting erroneous entries.

Entering a zero for the numerator of the first ratio leads to calculation of default ratios. These ratios are defined (in *MSC*) by having the most intense peak in the denominator and all other isotopes involved in *sum(I)* in the numerator. The background position is also used as a numerator.

#### 4.6.3 Edit\_Ratio

This routine allows inspecting and editing an existing record. The record number is input, and its present contents listed on the screen through use of Subroutine *Put\_Ratio*. A correction index is entered, and a new value for the affected parameter read in using Subroutine *Get\_Ratio\_Params*. An example of the *Ratio* edit screen appears in Table 3.6 of the Operator's Manual.

#### 4.6.4 List\_Ratio

This routine lists the contents of the *Ratio* file on either the screen or the printer at the analyst's option. Either the entire file can be listed or only those records associated with a selected element.

#### 4.6.5 Move\_Ratio

This simple routine moves the contents of one record to another. Values for both record numbers are input by the analyst. The original record remains intact; if it is no longer wanted, it must be specifically deleted (see next Section).

#### 4.6.6 Delete\_Ratio

The record number to be deleted is entered by the analyst. The element symbol is set to blank; all other parameters (integers) are set to 0, and the "empty" record written to the disk.

#### 4.7 Option 4: MASSAB

*Massab* is short for MASS and ABundance. This file contains the exact masses and abundances of all naturally occurring isotopes. Entries are also there for man-made elements we analyze: Np, Pu, Am, Cm, Cf, etc. There are masses entered for these elements but no abundances, which are all 0.00.

The *Massab* file is described in Section 3.8 and its structure listed in Table 3.6. There are three subroutines in *Massab\_Edit*; they are listed in Table 4.8.

---

Table 4.8 Subroutines in Massab\_Edit

Find_EI	Locates entry in the file
New_EI	Accepts input for a new entry
Display_EI	Display the contents of one record

---

#### 4.7.1 Subroutine Find\_El

This routine is called at various places in the program. Its function is to associate a record number in the *Massab* file with the element symbol entered. Access to *Massab* is provided by entering the element symbol rather than a record number. Element symbols in the file are compared sequentially with the one entered. If a match is found, the record number is returned; if no match is found, a zero is returned. In the latter case, the analyst is warned; no attempt is made to access record 0--an exercise that infallibly causes the program to crash.

#### 4.7.2 Subroutine New\_Entry

This routine is accessed only when a new entry is to be made to the file. A recent example of this came about when  $\text{GdO}^+$  interfered with  $\text{Lu}^+$  at masses 175 and 176. A new entry of the GdO spectrum (symbol GO) was made to allow correction of Lu data in program *MSC*.

An example of the input window is given in Table 3.6 in the Operator's Manual.

#### 4.7.3 Subroutine Display\_El

This routine displays the contents of one record of *Massab* on the terminal screen. See Figure 3.6 in the Operator's Manual for an example.

#### 4.7.4 Executing *Massab*

Invoking *Massab\_Edit* brings a menu to the terminal screen. It is listed in Table 4.9.

---

Table 4.9 MASSAB Menu

- 0 to stop
  - 1 to edit or inspect an element
  - 2 to list entire file on printer
  - 3 to delete an entry
  - 4 for a new entry
- 

These options are described below. The entire *Massab* file is read into memory to facilitate access to it.

##### 4.7.4.1 Option 0: Stop

As usual in this program, entering zero as a menu option signifies that you are done with the present file. Control reverts back to the main *File* menu (Table 4.1).

#### 4.7.4.2 Option 1: Edit or Inspect

Selecting a 1 from the *Massab* menu allows you to inspect or edit the entry of your record identification choice. A window is opened, and you are prompted to enter the element symbol. Subroutine *Find\_El* is used to locate the appropriate record in the file; remember that element symbols are used as input and not record numbers. Subroutine *Display\_El* lists its contents to the terminal screen. Each parameter displayed has an index number associated with it, and changing a value is accomplished in the usual way.

#### 4.7.4.3 Option 2: List

This option results in an entire file being listed on the printer. In the program code, this is accomplished within the DO loop that reads the file into memory (array *msab*).

#### 4.7.4.4 Option 3: Delete

Option 3 allows an entry to be removed from the file. This should occur only when an erroneous entry has been made by an analyst (presumably) unfamiliar with the program. The element symbol is called for, its location in the file identified, and the user asked if he or she is sure it is to be deleted. Only a "Y" is accepted as permission to proceed. The element symbol is set to blanks and numerical parameters to zero before the record is rewritten to the disk.

#### 4.7.4.5 Option 4: New Entry

Calling for a new entry causes subroutine *New\_El* to read in values for all necessary parameters. A chance to edit input is provided. The new entry is written into the first available empty record in the file.

### 4.8 Option 5: SPIKE

The Spike file is designed to hold information about enriched isotopic spikes used in isotope dilution analysis. The file is described in Section 3.6 and its structure listed in Table 3.4.

Upon entering *Spike\_Edit*, a screen opens and the menu in Table 4.10 appears.

---

Table 4.10 SPIKE Menu

- 0 to return to calling program
  - 1 for a new entry
  - 2 to edit or inspect an existing record
  - 3 to list contents of file on printer
  - 4 to delete a spike
  - 5 to update Pu-947
-

The result of choosing each of these options is described below.

#### 4.8.1 Option 0: Return

Selecting 0 returns control to the main *File* menu (Table 4.1).

#### 4.8.2 Option 1: New Entry

This option steps the analyst through all the entries required for a new entry. This operation is described in Section 3.7.1 of the Operator's Manual. Note that the atomic weight is calculated automatically; no independent entry is permitted.

Upon completion of input, provision is made for correction of any errors.

#### 4.8.3 Option 2: Edit or Inspect

Choosing this option allows one to inspect and edit an existing record. One of the most common reasons for editing is to change the isotope dilution ratio, but any indexed parameter can be altered. An example is given in Table 3.10 of the Operator's Manual.

#### 4.8.4 Option 3: List

Selecting this option results in listing the file on the printer, thus providing a hard copy.

#### 4.8.5 Option 4: Delete

This option allows deletion of specific records in the *Spike* file. The element symbol is set to blanks and all other parameters to zero.

#### 4.8.6 Option 5: Update Pu-947

NIST Pu-947 is in the *Spike* file because of its use in calibrating Pu spikes. The half-lives of some of its isotopes are short enough to change significantly in a short time (a few months). This option corrects the isotopic composition of Pu-947 to the present day (or any other).

#### 4.9 Option 6: STAND

*Stand* is a file containing the isotopic compositions of certified isotopic standards. A number of subroutines support the main effort; they are listed in Table 4.11.

---

Table 4.11 Subroutines in Stand Edit

Get\_Ratio\_Masses  
Get\_At\_Pcts  
Calc\_Ratios  
New\_Entry  
List\_Stand  
Edit\_Stand  
Print\_Stand  
Last\_Stand

---

The functions of these routines is pretty well indicated by their names, and the logic involved seems straightforward enough not to require further description.

*Stand* is described in Section 3.7 and its structure given in Table 3.5.

The menu for *Stand\_Edit* is given in Table 4.12.

---

Table 4.12 STAND Menu

0 to return to calling program  
1 to insert new entry  
2 to edit or inspect a record  
3 to list labels only  
4 to list entire file  
5 to edit last record used

---

These options are described below.

#### 4.9.1 Option 0: Return

Entering zero for your option causes the program to return to the main *File* program menu (Table 4.1).

#### 4.9.2 Option 1: New Entry

A new entry is automatically stored in the next available record. The value of the last record used is kept in *Stand's* bookkeeping record (record 51). Its value is incremented and assigned to the new entry.

The analyst is prompted for the various parameters. Subroutines *Get\_Input* (Section 11.15), *Get\_Ratio\_Masses*, and *Get\_At\_Pcts* are used to read in the data.

#### 4.9.3 Option 2: Edit or Inspect

This option provides the ability to inspect or edit a record. The record number is entered and its contents displayed on the screen. Editing is accomplished in the usual way through use of the index. An example of a record is given in Table 3.12 of the Operator's Manual.

#### 4.9.4 Option 3: List Labels

This option will list only the labels of the entries in the *Stand* file. It is sometimes helpful to find out if a given standard is present and, if so, in which record its information is stored.

The list can be to any legal output device (normally printer or terminal screen) at the analyst's option.

Subroutine *Print\_Stand* is used to implement this option.

#### 4.9.5 Option 4: List Entire File

This option is similar to Option 3; the entire contents of each record, rather than just the labels, are listed on the output device of choice.

#### 4.9.6 Option 5: Edit Last Record Used

*Stand* is one of the files with a bookkeeping record. This option allows its value to be changed; the only time it's likely to be required is if major mistakes have been made, and it's easier simply to change the last record used rather than change entries item by item through Option 2.

#### 4.10 Option 7 SAMP

*Samp* is the file whose entries are most often changed. It contains most of the information that dictates the course taken by the calculations in *MSC*. When it doesn't contain the information directly, it does so in the form of pointers to file records that do. An example of this is the *Ratio* file record number, whose contents determine what ratios are calculated, among other things.

Some of the most common variables changed are the *Ratio* record number, the first and last runs to be included in the calculations, and the calculation index.

*Samp* is described in Section 3.2 and its structure listed in Table 3.2.

Each record in *Samp* holds more information than can conveniently be displayed on

a single screen. The variables have been grouped into six subsets in what seemed to the programmer a logical way. The data for each subset occupies its own screen. The various subsets are described in Section 3.9.1 of the Operator's Manual and listed below in Table 4.13 (Identical to Table 3.16 of the Operator's Manual).

---

Table 4.13 Windows for Editing Samp

<u>Window</u>	<u>Description</u>
Identification Info	Sample codes, etc.
Calculation Parameters	Calculation index, etc.
File Record Info	File record numbers
Run Info	First and last runs, etc.
Spike Info	Ratio, weights, etc.
Subgroup Info	Masses, sweep factors

---

Selecting a subset of variables allows the analyst quick access to the desired parameter. It can be a bit tedious to have to step through all six windows when only one variable is of concern. Examples showing the contents of each of these windows are given in Tables 3.16-3.22 of the Operator's Manual.

Another feature provided for the convenience of the user is the ability to access only one variable. While this feature is of most utility during debugging, it is also helpful when an analyst has to change the same variable for a number of samples. Only the most commonly changed variables can be accessed this way; this is accomplished through the short options menu, listed in Table 4.14, (identical to Table 3.17 in the Operator's Manual).

---

Table 4.14 Short Options Menu

- Select one parameter from the following:
- 0 to end alterations
  - 1 to change calculation index (ICALC)
  - 2 to reinitialize
  - 3 to change Batch Code
  - 4 to change masses in isotope dilution ratio
  - 5 to change SPIKE file record no.
  - 6 to change RATIO file record no.
  - 7 to change 1st run
  - 8 to change last run
  - 9 to interchange spike and sample by new spike companion code and new SPIKE file record
  - 10 to change resin bead companion
  - 11 to change original Pu date
-

---

There are a number of subroutines that handle the necessary chores. Many of them simply display the contents associated with one subset of variables; editing itself is performed from within *Samp\_Edit* itself. The subroutines are listed in Table 4.15. Note that those beginning *Write\_* control the various window displays.

---

Table 4.15 Subroutines in Samp\_Edit

<u>Subroutine</u>	<u>Description</u>
Book_Samp_Edit	Access bookkeeping record
Screen_Setup	Initializes the windows
Write_ID_Data	Identification information window
Write_Calc_Data	Calculation parameter window
Write_File_Data	File record window
Write_Run_Data	Run information window
Write_Spike_Data	Spike information window
Write_SG_Data	Subgroup information window

---

Examples of each window with the variables displayed on each are given in the Operator's Manual in Tables 3.16-3.22. The meanings of the various variables are given there, too, beginning with Section 3.9.2.

#### 4.10.1 Samp\_Edit Menu

The menu for *Samp\_Edit* is shown in Table 4.16.

---

Table 4.16 SAMP Menu

- 0 to return to calling program
  - 1 to list all variables
  - 2 to list only 1 subset of variables
  - 3 to change 1 variable only (no list)
  - 4 to access bookkeeping record
- 

These options are described below.

#### 4.10.2 Option 0: Return

As in other editing routines, entering a zero for the *Samp\_Edit* option returns you to the main menu for program *File*.

#### 4.10.3 Option 1: List All Variables

This option provides access to any variable in a selected record. Once the record number is read in, its value is displayed in its own window at the top of the screen, and the first window of information (Identification Information window) appears. Any of the indexed variables can be edited in the usual manner. Entering zero for the edit index brings the second window of information to the screen. This process is repeated for each window in turn. Successive entries of zero has the effect of scrolling through the six windows. When the last window is cleared from the screen, you are asked for a new record number. Entering a zero here takes you back to the *Samp\_Edit* menu.

#### 4.10.4 Option 2: List One Subset

This option displays only one of the six windows for editing or inspection. The six choices are listed in Table 4.13. Entering zero for the edit index clears the window, and you are asked for a new record number.

#### 4.10.5 Option 3: Change One Variable

When this option is selected, the user is asked to choose from among the variables listed in Table 4.14. Once this choice is made and a record number entered, the window holding the variable in question is displayed in its entirety. The cursor is located at the variable in question. Once a new value for the variable is entered, the window is erased and a new record number called for.

#### 4.10.6 Option 4: Bookkeeping Record

Bookkeeping in *Samp* is far more complex than it is for most other files. It was discovered (empirically, of course) that searching all 400 records for calculation indices of 0 took quite a long time--over 45 seconds. To address this undesirable situation, a list of the record numbers of those samples awaiting calculation was appended to the bookkeeping record, after the value of the last record used. Subroutine *Upmtsm* (Section 11.33) was written to handle operations involving this array. It is called from both *MSC* and *File*, adding or removing record numbers as appropriate. Record numbers for samples being calculated for the first time are removed from the array in subroutine *C2* (Section 8.5). The contents of this array are displayed on the screen when this option is chosen; it can be edited, although there should be little reason to do so.

*Upmtsm* is called whenever the value for a sample's calculation index is changed. If the new value is zero, the record number is added to the array. A new value of 1 will cause the record number to be removed. There is room for 150 entries in this array, a number unlikely to be approached.

The other entry in *Samp*'s bookkeeping record is the value of the last record used. Each time a new sample is inserted in the file, its record number is determined by incrementing by one the last record used. The new value of the last record used is then written to the bookkeeping record. There is room for 400 samples in the file. When the last record used is incremented to 401, it is reset to 1. Thus the *Samp* file continuously cycles through its 400 records. It is unlikely at this late date that anything serious will go wrong, but means of changing the value of the last record used is provided in case it does.

#### 4.11 Option 8: Raw Data

*Raw\_Data\_Edit* accesses two files, *Sumcts* and *Chdata*. *Sumcts* is a condensed form of *Chdata*, in which 32 real numbers per subgroup have been reduced to one. These editing routines are useful when a problem in the calculations makes it desirable to inspect the raw data. Each file has a bookkeeping record whose contents can be altered. The only reason to change the raw data in either file is to generate a test case for program debugging.

There are two subroutines, *Sumcts\_Edit* and *Chdata\_Edit* called as appropriate from *Raw\_Data\_Edit*.

#### 4.12 Option 9: OUT

The *Out* file contains results for the last 3000 samples analyzed. It is sometimes desirable to inspect a record but never appropriate to change a measured value. *Out\_Edit* operates through a number of subroutines, which are listed in Table 4.17.

---

Table 4.17 Subroutines in Out\_Edit

File_IO	Disk file Operations
Sam_Edit	To edit the contents of a record
List_Record	To list the contents of a record to the screen
Set_Ratio_Strings	Converts ratio values to strings
Comp_Info	To handle input of companion sample information
Comp_Input	To read in companion sample information
List_Perm_Comp	To list permanent companions on the printer
Correct_NBS-947	Corrects NBS-947 to specified data
Change_Last_Rec	Accesses bookkeeping record
Shift_Output	Copies one record to another
Bkout_Edit	Accesses Out bookkeeping array

---

The most common reason for accessing *Out* is to enter information for companion samples. Isotope dilution calculations require knowledge of the isotopic composition of the unspiked sample, and *MSC* assumes it will be stored in *Out*.

Choosing to access *Out* results in a menu appearing on the screen. It is shown in Table 4.18.

---

Table 4.18 OUT Menu

- 0 to return
  - 1 to edit or inspect a record
  - 2 to enter companion info
  - 3 to enter permanent companion info
  - 4 to list permanent companions
  - 5 to correct NBS-947 to any date
  - 6 to change last record used
  - 7 to shift data to a new record
  - 8 to access BKOUT (bookkeeping for Out)
- 

Note that *Out\_Edit* is called from program *Rep* (Chapter 9) as well as program *File*. This is merely a convenience to allow analysts to access the file from either program.

The various menu options are described below.

#### 4.12.1 Option 0: Return

Choosing zero causes the program to leave *Out\_Edit* and return to the main menu of *File* or *Rep*.

#### 4.12.2 Option 1: Edit or Inspect

There is a lot of information in an *Out* file record, but the need to make alterations arises so seldom that it was not deemed worth the time required to develop a multiple window display as was done for *Samp* (Section 4.10). The display is not the easiest thing in the world to read, but all the information is on the output sheets, so no serious problems should result.

Only a few of the many variables can be changed. These are the sample identification code, the two companion sample codes, the descriptive title, the batch code, and the date of analysis. Only these variables have index numbers (1-5) associated with them. No variable that directly affects the calculations (such as masses) and no variables calculated in the program can be changed.

#### 4.12.3 Option 2: Enter Companion Information

This option provides a convenient method for an analyst to enter information for a companion sample. Companion samples are used in a number of different kinds of analyses, isotope dilution and U-Pu pairs analyzed from resin beads being the most common. The minimum amount of information necessary is entered. The information requested is listed in Table 4.19.

---

 Table 4.19 Companion Information
 

---

<u>Variable</u>	<u>Description</u>
Samp_id	Sample identification code (6 chars)
Title	Descriptive title
Sum_mass	Isotopic masses (from Massab)
At_pct	Atom percents

---

The date of entry is assigned to the record by the program. Isotopic masses are read from *Massab*, and the analyst may accept or reject them. If they are rejected, the analyst must enter them manually. Atom percents are entered; weight percents and atomic weight are calculated using Subroutine *Wgpct* (Section 11.37). The information is written into the record next in line to be used in normal operation.

#### 4.12.4 Option 3: Permanent Companions

This option is identical to number 3 except that the information is written into one of the first 50 records in the *Out* file. These records are designated to hold "permanent" companion samples. Such samples are those that are needed on a regular basis in our work; examples are NIST U-950 (natural), NIST Pu-947, and normal lutetium. The *Out* file has room for 3000 samples. When the last record is used, the next sample is stored in record 51, leaving the information in the first 50 records intact.

#### 4.12.5 Option 4: List Permanent Companions

This option results in a listing of the contents of permanent companion records on the printer.

#### 4.12.6 Option 5: Correct NBS-947

This option works identically to that described in Section 4.8.6. It updates the isotopic composition of NIST (NBS) Pu-947 to the current date.

#### 4.12.7 Option 6: Change Last Record Used

This option provides the means to alter the last record used. It is occasionally useful to do this to prevent loss of information for samples that are about to be written over and with which you are not yet done.

#### 4.12.8 Option 7: Shift Data

This option provides another means of protecting data from being overwritten. The contents of one record are moved to another, as specified by the analyst.

#### 4.12.9 Option 8: Access BKOUT

*Bkout* is a file separate from *Out* that is used to help locate samples in it. The contents of *Bkout* are the 3000 six-character sample identification codes of the samples whose information is presently stored in *Out*. The reason for this cumbersome device is that it takes a long time to search each record (read from the disk) one by one and compare its sample code with the target code. Searching an array through means of a DO loop is far faster.

This routine is more complex than necessary. When first implemented on our DEC PDP 11/23, there was not enough memory to hold 3000 six-character names. The contents of *Bkout* were divided into three records of 1000 names each. *Bkout\_Edit* reads each set of 1000 until the search is successful and returns the value of the record in *Out* occupied by the target sample.

All these shenanigans should be transparent both to the analyst and to the programmer. They have been working for a long time without problems.

This option provides a means of correcting the identification code of a sample should it ever become necessary. You should know what you're doing before you try it.



## 5. PROGRAM MSR

*MSR* is the program that collects data, interacting with various mass spectrometric components to do so. It resides only on the PC's; the VAX is not designed to interface to laboratory equipment, so there is no reason to have it there. All three PC's have identical versions of *MSR*; differences between them arise only in the contents of the data files (*Scan*, *Chdata*, *Samp*). Operation of the mass spectrometer itself was described in an earlier technical memorandum.<sup>3</sup>

*MSR* is the root module of the data acquisition program. It calls two main subroutines, *Setup* and *Msop* (for Mass Spec Operate). *Setup*'s function is to accept as input the information necessary both for data accumulation and for data processing. *Msop* is the routine that actually acquires the data and stores it in disk files. The *MSR* module itself does little beyond coordinating these two routines. When first entered, it checks to be sure the Sweep Control Panel is in the proper condition and informs the operator of what the problem is if there is trouble. After an analysis is completed, it ascertains from the operator whether he wants to analyze another sample or leave the program.

### 5.1 Details of *MSR*

*MSR* is a short program written in Microsoft Fortran which does very little beyond calling various subroutines. Immediately upon entering the program, a function, *Msdsw*, is accessed to check the status of the Sweep Control Panel. This function returns the control panel status word; the condition of the bits indicates the status of the switches on the Sweep Control Panel. Figure 5.1 summarizes the bit pattern. *MSR* will not allow the operator to proceed unless the Sweep Mode switch is in the Wait position.

Once the Sweep Control Panel is in the proper mode, *MSR* calls *Setup*. This subroutine, written entirely in Microsoft Fortran, displays an input screen (Table 5.1) and reads in values for variables entered by the operator. Certain parameters are essential for successful execution of either *MSR* or *MSC*; most are optional. The operator is prompted for input of each variable in sequence; the exact path taken depends on what he enters. Once all input has been entered, control is shifted to Subroutine *Msop*, which controls data acquisition.

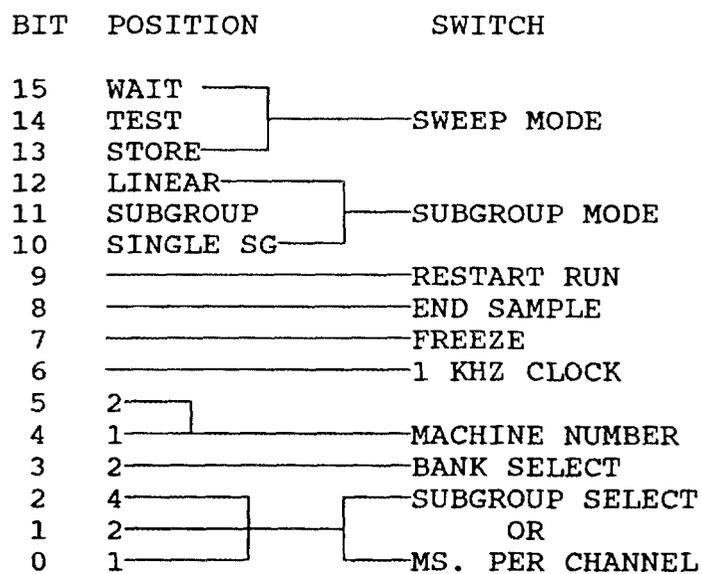


Fig. 5.1 Control Panel Status Word

---

Table 5.1 Input Screen for MSR

Input Parameters

1. SAMP file record no.
2. SCAN file record no.
3. Sample id
4. Descriptive title
5. Batch code
6. RATIO file record no.
7. Resin bead companion id
8. SPIKE file record no.
9. Unspiked companion id
10. Isotope dilution ratio
11. Spike weight
12. Sample weight
13. Dilution weight
14. Aliquot weight
15. Density
16. Reverse iso diln index 17. Containant el symbols 18.  
Blank companion id

Index no. to correct; 0 to proceed:

---

## 5.2 Setup

*Setup* accepts input of the various parameters required for execution of both *MSR* and *MSC*. The goal here was to make processing of the data by *MSC* proceed automatically at the conclusion of data-taking, with no further information required from the operator beyond what is input in *MSR*.

*Setup* first determines whether or not all previous data have been sent to the VAX. The first entry for each record in the *Samp* file is a variable called *icalc* which is used as a flag for this situation. (Note that it serves a different function on the VAX.) *Icalc* = 1 indicates that the data for that sample have been transferred; a value of 0 indicates that they have not. If data for one or more samples have not been transferred, a warning is written to the screen to alert the operator of this fact; there is no real danger here unless all records have new data and none has been transferred. Since there is room for 100 samples in the *Samp* file on the PC's, it is unlikely that this will ever happen.

As described in Chapter 3, record 101 of the *Samp* file is used for bookkeeping, with the value of the last record written to kept in it. This value is read and incremented each time *Setup* is entered. It is reset to 1 when 100 would be exceeded.

After initializing a few variables, a window opens that accepts input from the operator for all the variables that control acquisition and processing of data. An example is shown in Table 5.1. The meaning of all variables is given in Section 3.4 of the Operator's Manual. Input is accepted in reverse video to make it easy for the eye to find the proper line on the screen. The length of the bar spans the maximum number of characters allowed for that entry. Hitting return terminates entry for that variable, and the highlighted input bar moves down to the next line. It is not necessary for the operator to enter values for every variable listed on the screen. Some, however, are essential for proper execution of either *MSR* or *MSC*, and the operator is forced to make entries for these. These are the *Scan* file record number (*nscan*), which is the record number in the *Scan* file that will control data-taking in *MSR*--it is not possible to operate the Sweep Control Panel without this information; the sample identification code (*Lsam*), which is used to locate it in the *Out* file on the VAX; and the *Ratio* file record number (*ifrat*), which identifies the record number in the *Ratio* file to be used in processing the data in *MSC*. Note that the record to be used in the *Samp* file is assigned by the software and not entered by the operator. Spiked samples require a lot more information, and some of the variables, such as the batch code, have special meanings in *MSC* that are covered in the chapter on that program (Chapter 8). Table 5.2 shows an input window after entry of all the information necessary to run an unspiked uranium sample.

As described in the section on the user interface (Chapter 2), the operator is given the opportunity to correct any input errors before the program proceeds. By entering the index value associated with the variable in question, the value for any variable may be altered. The program cycles through this sequence until a 0 is entered, at which time control returns to *MSR*; *Msop* will be called at that point.

### 5.3 Msop

*Msop* controls data acquisition. Parameters input in *Setup* are passed to *Msop* for use in this process. It is in *Msop* that certain assembly language routines are called that interact with the instrument at the most fundamental level in situations either impossible to address with Fortran or too slow of execution in that language. These programs communicate with the outside world as shown in Table 5.3. The names on the cables correspond with variable names in the software.

When data-taking is complete, the data is transferred to the MicroVAX via Ethernet using batch file *Tran* (Section 7.1).

---

Table 5.2 Input Screen for MSR with Information Entered

Input Parameters

1. SAMP file record no.	12		
2. SCAN file record no.	1	E1	U
3. Sample id		U10039	
4. Descriptive title		U233 SPIKE	
5. Batch code			
6. RATIO file record no.		1	
7. Resin bead companion id			
8. SPIKE file record no.		0	
9. Uspiked companion id			
10. Isotope dilution ratio	0	0	
11. Spike weight	1.0		
12. Sample weight		1.0	
13. Dilution weight		1.0	
14. Aliquot weight		1.0	
15. Density		1.0	
16. Reverse iso diln index			17. Contaminant el symbols 18.
Blank companion id	0		

Index no. to correct; 0 to proceed.

---

---

Table 5.3 Assembly Language Routines

File: MSRUN.ASM - contains the following entry points

MSDSW: Function - initializes interface registers. Reads MSSTAT which indicates the position of various switches on the MS Control Panel.

I = MSDSW()

FIN1: Function - returns the value of the MSSTAT as above.

I = FIN1()

INDATA: Function - returns the current value in the ion current counting scaler.

I = INDATA()

CLEARSC: Subroutine - clears screen. Puts cursor at top left corner.

CALL CLEARSC(0)

KEYREAD: Function - returns keyboard shift status value.

I = KEYREAD()

ROUT1: Function - returns the value of the variable OUT1.

I = OUT1()

SOUT1: Subroutine - send value of argument to I/O board as addressed by AOUT1.

CALL SOUT1(I)

SOUT2: Subroutine - send value of argument to I/O board as addressed by AOUT2.

CALL SOUT2(I)

SOUT34: Function - send value of argument (INTEGER\*4) to the I/O board as addressed by AOUT3 and AOUT4.

I = SOUT34(MSDATA(k))

MSRUN: Function - passes names and addresses of variables used by the control system and data collection to MSRUN. Calls SETUP to initialize an interrupt, etc.

I = MSRUN(IOUT1, LINSG, NSF(ISG+1), FLAG1, MSDATA(IISG\*32+1))

MSDATA() is formatted INTEGER\*4.

SETUP: Internal subroutine. Called by MSRUN above.

CALL SETUP

Shutdown: Subroutine - returns computer to original configuration prior to calling SETUP.

CALL SHUTDOWN

MES1: Internal subroutine. Used for troubleshooting to write a message on the screen.

CALL MES1

---

### 5.3.1 Analysis Sequence

The sequence of operations performed during a typical analysis is described below; the programmer needs to be familiar enough with each operation so that he understands how it affects the programs. We assume that all necessary power switches have been turned on (and that the slit valve is open!).

1. The wheel upon which sample filaments are mounted is rotated to bring a new sample to the analytical position.
  2. The Sweep Mode switch on the Sweep Control Panel is set to Wait.
  3. Program *MSR* is activated on the DEC 316+. It in turn begins execution of subroutine *Setup*; if a sample has already been run and *MSR* was not left, entering the appropriate reply in response to the prompt will return control to *Setup*.
  4. The variables necessary for the analysis are entered; these are described in detail in Chapter 4 of the Operator's Manual. Once input is complete the program automatically continues to Subroutine *Msop*.
  5. The filament temperature is raised and the ion beam located and focused. The Sweep Mode switch on the Sweep Control Panel is in either Wait or Test for this operation.
  6. When ready to start the analysis, the operator sets the Sweep Mode switch to Store to initiate accumulation of data. A real-time display of the data array appears on the spectrum monitor. Values for relevant variables are listed at the top of the computer monitor screen, and the values for up to three ratios are listed for each run as the analysis progresses. An example of this screen is given in Table 5.4.
- Data-taking can be interrupted at any time by the operator to check the ion-beam focus and adjust the filament temperature. This is accomplished by moving the Sweep Mode switch from Store to Test or Wait.
7. Data-taking is terminated by the operator pressing the End Sample switch. This can be done at any time during an analysis; if it is done during Run 1, no meaningful data have been accumulated and no data are kept of the analysis in question.
  8. The operator is prompted to define the next analysis or to exit *MSR*. Table 5.5 defines the options.
  9. When *MSR* is terminated, the operator can initiate transfer of data to the VAX by executing batch file *Tran* (Section 7.1). The relevant data from file *Samp* and file *Chdata* are transferred to the VAX, *MSC* is called by the resident batch file on the VAX, the calculations performed, and a report of the results output on the printer. When *MSC* is done, the batch file returns to a wait condition and is ready to accept the next transmission of data. All steps proceed automatically once *Tran* is executed on the work station.

---

Table 5.4 MSR Screen at Completion of Data-Taking

Sample U10039 No. SG 8 Cycles 80 SAMP File 12 CHDATA File 653

Subgroups	1	2	3	4	5	6	7	8
Masses	315	233	234	235	236	365	238	239
Sweep Factors	4	1	4	4	4	0	4	1
Run	CR 233	235/238		238/233		234/235		
1	5.54E+05	.014344		.000212		.714286		
2	3.09E+05	.009597		.000203		.800000		
3	3.17E+05	.017730		.000214		.550000		
4	2.88E+05	.019608		.000202		.631579		
5	3.34E+05	.018467		.000195		.850000		
6	2.65E+05	.012373		.000202		.818182		
7	3.37E+05	.013274		.000202		.533333		
8	2.49E+05	.025434		.000209		.136364		
9	3.49E+05	.016393		.000210		.700000		
10	3.04E+05	.013013		.000197		1.076923		

End of sample after 10 runs

Temp and comments: 1730-1800

Starting run: 1

Ending run: 10

---

The full meaning of these options is given in Section 10.2.1.4 of the Operator's Manual, although the short descriptions in Table 5.5 should suffice for most needs.

---

Table 5.5 Next Sample Options

<u>Value</u>	<u>Result</u>
0	Leave <i>MSR</i>
1	Next sample has same parameters as present one.
2	New weights for isotope dilution sample; all else the same.
3	Completely new parameters will be entered.

---

### 5.3.2 Execution of Msop

One of the features of *Msop* is the display of information helpful to the operator during analysis. An example of this display is shown in Table 5.4. The header contains general information and allows the analyst to verify that the *Scan* file called will actually acquire data as desired. Information concerning the subgroups is listed on three lines, and

below that is listed information diagnostic of the present analysis. With each run number is listed the count rate for the most intense isotope measured and the values of up to three ratios. The mass position for which the count rate is to be calculated is not specified in the *Scan* file; after the first run, the program identifies which subgroup contains the most counts and calculates the count rate for it. Identification of this mass position is done only after the first run to avoid the potential confusion of having the count rate calculated for different isotopes during an analysis. The masses of the ratios to be listed are contained in the *Scan* file; the program goes through a routine to correlate the ratios (entered as masses to make it simple for the analyst) with the subgroup numbers (to make it simple for the programmer). If no ratios are specified in the *Scan* file, the program assigns the most intense peak to the denominator and calculates the three biggest ratios for display.

Upon entering *Msop*, this display is written to the screen; it has no values for the ratios, and the isotope for which the count rate will be calculated is as yet unknown. This leads to an entry for the header over the count rate column of 'CR 0' or some other meaningless mass. This is perfectly normal and is not indicative of a problem anywhere (despite what the analyst may think!).

*Msop* retrieves the instrument identification number (also called machine number; the variable name is *mch* in *MSR*) and sweep rate code (*iswr*) from the Sweep Control Panel device status word (See Figure 5.1). The instrument number is held in bits 4 and 5; the sweep rate index in bits 0-2. Note that, if the instrument number is 0 (*mch. EQ. 0*), it is automatically set to 4--the transuranium mass spectrometer. Table 5.6 decodes the sweep rate bits to the actual sweep rate as chosen by the Sweep Rate switch on the Sweep Control Panel.

*Msop* next sets up the arrays that control the display of ratios on the computer monitor during an analysis. These ratios are calculated using the central 25 channels of each subgroup. Using the masses entered by the operator when setting up the *Scan* file (or the default selections), starting channels for the sum of each subgroup used in the raw count array are determined. Each mass has associated with it a sweep factor (number of times this mass position is swept for each traversal of the mass spectrum); these, too, are set into an array for use in calculating the display ratios. The reason for default values in the display ratios is that it is sometimes convenient for the operator not to have to enter display ratios in the *Scan* file. This might happen, for example, for a group of samples from Stable Isotopes that are all the same element but each with a different enriched isotope. In this situation, the analyst can enter 0 for the numerator of the first ratio to be listed (in program *File* while setting up the scan). Upon detecting this zero in *Msop*, a subroutine, *SetListRatios*, is called that identifies the mass position with the most counts in it. This mass position is the default denominator for all ratios to be displayed. The three next most intense peaks are the numerators. If these ratios are not the ones desired, explicit entries into the *Scan* file must be made.

Table 5.6 Switch Position Codes  
(Bits 0-2 of Control Panel Status Word)

<u>Switch Position*</u>	<u>Sweep Rate Code</u>	<u>Sweep Rate</u> (ms/channel)
8	7	0.0002
7	6	0.0005
6	5	0.001
5	4	0.002
4	3	0.005
3	2	0.010
2	1	0.020
1	0	0.050

\*Sweep rate switch, when subgroup mode switch is in the linear or subgroup position, otherwise single SG. switch.

The run counter is set to 0. The 16-bit connection, *Out1*, to the control panel is also set to 0. The various bits of *Out1* control different parts of the control panel; these are shown in detail in Table 5.7. By setting *Out1* to 0, the control panel is initialized and set to an inactive configuration.

Table 5.7 *OUT1*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			S.G. Number			S.G. Sweep Count					Channel Number					
			S.G. Number			S.G. Sweep Count					Channel Number					

Values in binary format

The number of mass positions to be swept is determined by the contents of the *Scan* file record. Any number of masses from 1 to 16 can be swept in an analysis, although sweeping one mass would not lead to a very informative mass spectrum. Because most analyses can be accommodated by eight masses or fewer, the 16 potentiometers have been divided into two banks of eight each. A variable, *ibank*, takes on a value to identify which subgroups are to be used, as listed in Table 5.8.

Table 5.8 Subgroup Banks

<u>IBANK</u>	Number of <u>Subgroups</u>	Number of <u>Channels</u>	<u>Used</u> Subgroups
-1	16	512	1-16
0	8	256	1-8
1	8	256	9-16

The position of the switches on the Sweep Control Panel is determined by the function *FinI()* or *Msdsw()*. The bit pattern is described in Figure 5.2. Bit 6 of the device status word is set to 0 at the start of each run; it represents ticks of the clock and is used for reference purposes during the run.

The counts read in from the counting system are stored in an array called *Mldata*. It can hold 512 32-bit integers. It is initialized to zero at the start of each run; data accumulate in each array element as the analysis proceeds. The indices of the array elements correspond to the channel numbers of the data as it is displayed on the spectrum monitor. To allow this display to be made as rapidly as possible, the channel number is encoded in the top nine bits of the *Mldata* words. The lower 22 bits contain the sums of the ion counts for that channel. Figure 5.6 shows cable connections between the computer and the spectrum monitor. Note that the lower 16 bits of an *Mldata* word are transmitted in cable *Out4* and the upper 16 bits in *Out3* as shown in Tables 5.9 and 5.10.

Table 5.9 OUT3

Channel Number									Channel Data*						
Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

\*Most significant bits, binary format

Table 5.10 OUT4

Channel Data*															
Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

\*Least significant bits, binary format

At the start of each run, the run count (*nrun*) is incremented and the sweep count (*ncyc*) set to 1. To make the signal compatible with the seven-segment LED displays on the control panel, these values are encoded in combined word (*Out2*) in binary coded format. The bit assignments of *Out2* are given in Table 5.11.

---

Table 5.11 OUT2

Bit	Run Count				Sweep Count												
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Values in binary coded decimal format

---

## 6. PROGRAM TEST6

*Test6* is a PC program that tests proper operation of the Sweep Control Panel and scaler. The oscilloscope (spectrum monitor) is used to display information diagnostic of the instrument's performance. This program is very useful for troubleshooting and should be used on a regular basis to check for dropped bits. Most of the instructions in this chapter appear on the terminal screen during program execution.

### 6.1 Menu

Execution of the program is initiated by entering *Test6* at the DOS prompt. A menu appears on the screen. The options are summarized in Table 6.1. When first called from DOS, the program is automatically set to option 1.

---

Table 6.1 Options in Test 6

<u>Option</u>	<u>Description</u>
0	Leave Test6
1	Read control panel switches
2	Read ion current data register
3	Test digital readouts on control panel and sweep output
4	Test scope display output
5	Test sweep output
6	Test interrupt
7	Test ion current input circuit and timer
8	Same as No. 7 with delay and interrupt circuits added
9	Test transmission of all bits from ion current circuit

---

Option 9 tests all circuits; everything must be working to pass this test. The various tests are described in more detail below.

### 6.2 Option 1: Control Panel Switches

This option provides the means to verify that the computer is correctly reading the switches on the Sweep Control Panel. Each switch is assigned one or more bits in the hexadecimal word; these assignments are summarized in Table 6.2.

---

 Table 6.2 Bit Assignments for Switches
 

---

<u>Bits</u>	<u>Switch</u>
15-13	Sweep mode
12-10	Subgroup mode
9	Restart run
8	End sample
7	Power on
6	1 kHz clock
5-4	Machine number
3	Bank select
2-0	Subgroup select or ms/ch selected by subgroup mode

---

The program calls subroutine *Msdsw* to read the position of the switches and *Bdisp* to list the binary number on the screen.

When a given switch is activated, a 1 is returned for its bit; otherwise the bit is zero. By using the locations summarized in Table 6.2 (and listed on the screen when running the program), you can turn the various switches to different settings and note whether the value of the binary number accurately reflects the changes.

This routine is not often needed, but is indispensable when it is.

### 6.3 Option 2: Read Ion Current Register

This option causes the ion current data register in the scaler to be read. This is input line 1.

The bits are defined:

15: Clear counts if Sweep Mode switch is in Test or Reset switch on scaler is down.

14-0: Ion current

Subroutine *Bdisp* is used to write the binary number to the screen.

### 6.4 Option 3: Test Digital Read-outs on Control Panel and Sweep Output

This test uses the horizontal deflection of the spectrum monitor to display the settings of the subgroup potentiometers. The Subgroup Mode switch must be set to subgroup and the Sweep Mode switch to test. At the same time the red LED displays on the front panel can be checked to be sure they are functioning correctly. Pressing the <ctrl> key will increment the displays; pressing the <alt> key allows selection of another option. This test checks output lines 1 and 2.

This option is useful during debugging; malfunction of either a display element or potentiometer should be obvious to the operator running the instrument.

#### 6.5 Option 4: Test Oscilloscope Display

This option checks the video display circuit using the spectrum monitor; this circuit uses output lines 3 and 4. The Sweep Mode switch must be set to wait or store. The Counts Full Scale selector is set to 127, 32K, and 2M in turn. This routine checks for malfunctions in the display circuit only. Examples of the spectrum monitor displays are shown in Figure 6.1. Problems elsewhere are identified with different options (See Section 6.11).

#### 6.6 Option 5: Test Output of Sweep Amplifier

This option is similar to Option 3 but faster and more fully automatic. The computer sets the sweep rate to 2 msec/channel; there is no delay at the start of the sweep across a new subgroup. The Sweep Mode switch must be in test; the Subgroup Mode switch can be in any position. The horizontal display of the spectrum monitor reflects the settings of the subgroup potentiometers; the width of each subgroup, determined by the chassis-mounted potentiometer inside the Sweep Control Panel, is also reflected in the display. The output signal at P13 can be monitored by a test oscilloscope for linearity and ripple. An example of the display is given in Figure 6.2.

#### 6.7 Option 6: Test the Interrupt

This option tests only the interrupt. An interrupt signal is a pulse train created in the control panel to indicate to the computer that a new data word is ready for acceptance into the data array. The interval between interrupt pulses is determined by the position of the Sweep Rate switch. The controls must be set to test and linear while the Sweep Rate switch is set to different positions. Each time an interrupt is received, "INTERRUPT" appears on the screen. It does this 11 times and exits.

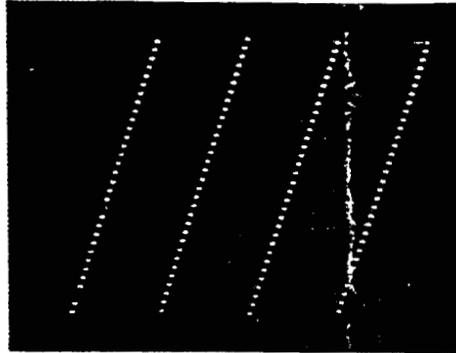
#### 6.8 Using the External Source

Provision was made in the design of the Sweep Control Panel and associated electronics to make proper operation easy to check. It is essential to be able to verify that the counting circuitry is working properly. This circuitry includes the scaler, the ion current input circuit, and the various delays and interrupts built into the system. Testing this circuitry requires a signal of known frequency. Incorporated into the Sweep Control Panel is a signal generator that puts out two frequencies: 100 kHz and 500 Hz.

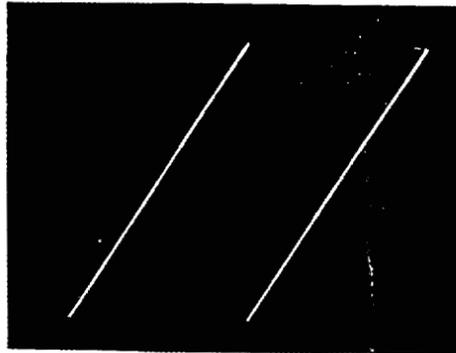
To make use of this feature, a portable signal conditioning box is connected to the frequency of choice. The signal cable from the mass spectrometer is disconnected from the input to the scaler and replaced by the coaxial cable output from the conditioning box.

This apparatus is used for tests 7, 8, and 9, described below.

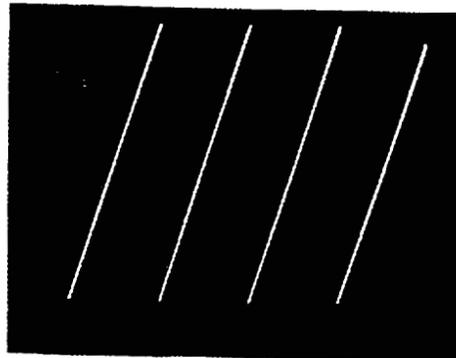
Test 6  
Option 4



Counts  
Full Scale  
127



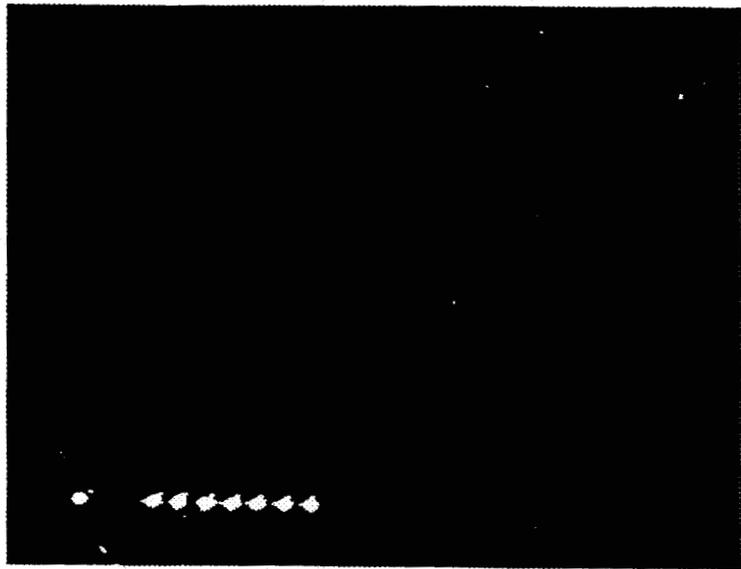
32K



2M

Fig. 6.1 Video Display Tests

Test 6  
Option 5



IN  
SG  
Mode

Fig. 6.2 Sweep Amplifier Test

### 6.9 Option 7: Test Ion Current Input Circuit and Timer

This option tests the ion current input circuit by feeding a constant signal (100 kHz) to the scaler from an external source. With the controls set to test, linear, 2 msec/channel, and 500 counts full scale, a straight line should appear on the spectrum monitor at the level of 200 counts. The delay before each subgroup and the interrupt are not tested in this option. To do so, option 8 or 9 should be chosen.

### 6.10 Option 8: Test Ion Current Input Circuit, Time, Delay, and Interrupt

This option is similar to Option 7 but has tests for the interrupt and the 5 msec delay included. The sweep rate is not included in this test; it is tested in Option 7. Set the controls to test, linear, and 500 counts full scale. Use the 100 kHz external signal generator as input. As in Option 7, a straight line at 200 counts should appear on the spectrum monitor. It will be traced at a slower rate than in option 7 but is otherwise identical.

### 6.11 Option 9: Full Test of Input Scaler

All systems are tested using this option. If the tests are successful, the entire scaler-sweep control system is performing as desired.

There is a switch on the back of the scaler that must be set to the down position. Setting it thus prevents the scaler from being set to zero after each channel. The data for each new channel is thus added to that already present, generating a straight line for the spectrum display.

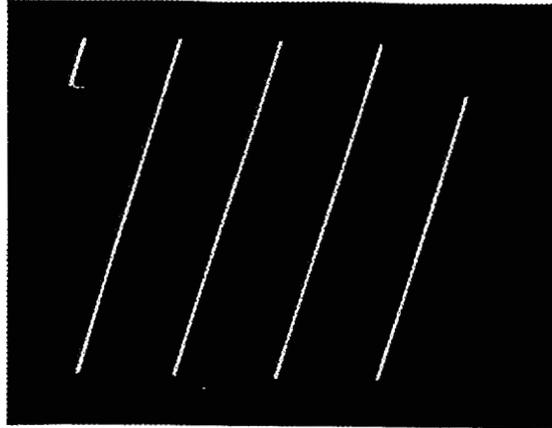
Note that the switch on the back of the scaler is in the down position **ONLY** for this test. Be sure to return it to the up position when you are done. Failing to do so will preclude taking mass spectral data--you get a weird display!

Examples of a properly functioning system are given in Figure 6.3. This routine is extremely helpful in determining whether or not dropped bits are plaguing the system. A dropped bit will be obvious on one of the displays in Figures 6.3a and 6.3b; which one is determined by which bit is being dropped. A dropped bit can be extremely difficult to detect in other ways. When the most sensitive bit or two is dropped, it is usually obvious on the spectrum monitor. Background noise doesn't look right, and the analyst becomes suspicious. Dropping the least sensitive bits is even more obvious: The analyst loses a factor of two in intensity. It's the intermediate bits that are most difficult to detect. They often appear only as a subtle bias on some ratio measurements. It is therefore a good practice to use this option of *Test6* on a regular basis, quarterly or semi-annually.

For the first part of the test, set the controls to Linear, Store, 2 msec channel, and 127 counts full scale. After one sweep, a four-cycle ramp is displayed on the spectrum monitor as shown in Figure 6.3a. Using 511 counts full scale gives a one-cycle display.

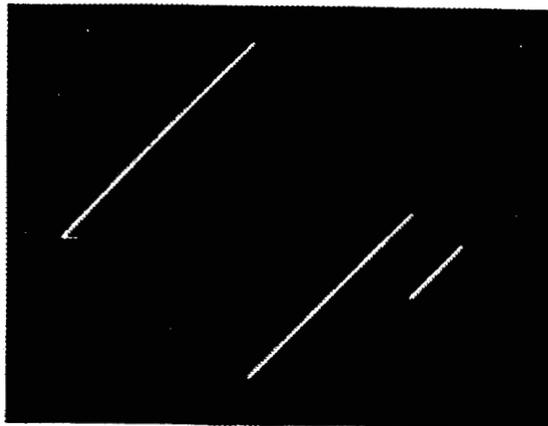
Change the controls to 200 msec/channel and 32K full scale. One sweep results in the display of approximately 1.2 cycles on the spectrum monitor--see Figure 6.3b. Repeating

Test6  
Option 9



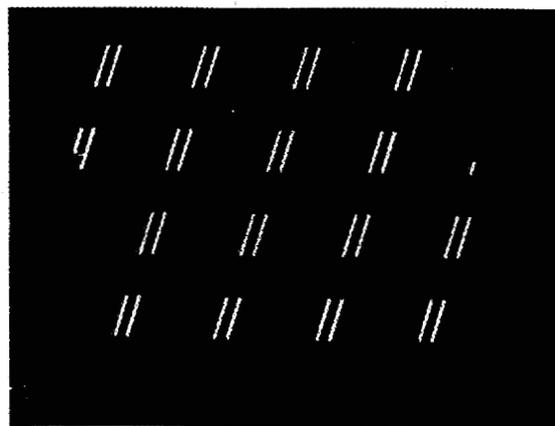
2. MS/CH  
127 Cts F.S.

(6.3a)



200 MS  
32K Cts F.S.

(6.3b)



2. MS/CH  
127 Cts F.S.

Pin #2  
IC-9, Ref 202  
Disconnected

Fig. 6.3. Input Scaler Tests

offsets from the desired ramps indicates that one or more bits are stuck on or off. To get out of the test, hit <alt> or <ctrl>.

Don't forget to return the switch to the up position; you will get strange data if you forget.

Using this option is the quickest way to see if the entire system is working. If all is well here, none of the other tests need to be made. If a problem is revealed, there should be some clue as to its nature; using the other options will help pinpoint it.

## 6.12 Subroutine Test5

This is an assembly language routine with several entry points. These are summarized in Table 6.3.

---

Table 6.3 Entry Points in Test5

Entry	Description
MSDSW	Initializes registers; reads MSSTAT, a variable indicating switch positions on the control panel I = MSDSW()
SOUT1	Send the value of J1 to control panel. Subgroup number, Subgroup sweep, Channel number.
SOUT2	Sends value of J1 to control panel. Run count, Sweep count.
SOUT34	Sends x and y data to spectrum monitor. I = SOUT34()
INTER3	Checks for the presence of interrupts.
S6, S7, S8	Control and accept data for options 6, 7, and 8.

---

## 7. PROGRAMS TRAN AND REC

Programs *Tran* (for TRANSfer) and *Rec* (RECEive) comprise a complementary pair of programs; together they control transfer of data from the PC's to the VAX. *Tran* resides on the PC's, *Rec* on the VAX. Note that these programs are specific to mass spectral data; other files must be transferred using DEC's Ethernet software.

### 7.1 Batch File Tran

Strictly speaking, *Tran* is not a program but a batch file. It resides only on the PC's and is invoked by entering *Tran* on the terminal. It first brings in Ethernet (*Startnet*) and then *Tran2*, the program that effects data transfer. A listing of the *Tran* batch file is given in Table 7.1.

---

Table 7.1 Batch File Tran

```
! tran.bat
Call STARTNET
print /d:1pt2
c:
cd\run
tran2
```

---

#### 7.1.1 Batch File Startnet

*Startnet* is a batch file that activates Ethernet. For it to execute properly, the currently active drive must be the one from which the PC was booted. It is unfortunate that keeping Ethernet active at all times interferes with interrupt levels, making it impossible to take data. (This was, of course, discovered experimentally.) This is what makes it necessary to call *Tran* separately and prevents transferring data directly from *MSR*.

The bulk of the *Startnet* batch file follows DEC guidelines and, as a consequence, should cause no problems.

#### 7.1.2 Program Tran2

*Tran2* is a Fortran program that identifies which information is to be transferred and sends it out on Ethernet to program *Rec* on the VAX. Samples to be transferred are identified by having a value of 0 (zero) for the variable *icalc*; the corresponding count data are located by the variable *ncd*, which is the record number in the *Chdata* file where run data for the sample start.

Data for all samples are first written into two files on the PC, *SampleN.DAT* and *ChdatN.DAT* where *N* in the file names represents the identification number for the mass

spectrometer involved. Information for one sample is appended to that for the sample preceding it to condense all information into the two files. These gyrations are necessary because Ethernet is a file transfer routine and cannot address parts of files. Once all samples have been converted, the two files are sent out over Ethernet. Program *Rec* on the VAX translates the compacted data back into its original form, as described in the next section.

## 7.2 Program Rec

*Rec* is a Fortran program that resides on the VAX. Its function is to receive the data sent from the PC's via *Tran* and to store it properly in the *Samp* and *Chdata* files in VAX directory [MSCALC]. It accomplishes this by first storing the incoming data in files in the [SYSO.DECNET] directory and subsequently transferring it to *Samp* and *Chdata* in the [MSCALC] directory.

It is our policy to keep *Rec* available on the VAX at all times so that analysts can transfer data whenever they choose. It is usually activated by entering @RECMSC at the \$ prompt on the printer terminal dedicated to Ethernet operations. This batch file activates *Rec* and, when data transfer is complete, automatically calls program *MSC* to process the data before invoking *Rec* again.

If *Rec* were to remain active at all times, it would appreciably slow other operations. It is therefore kept in a quiescent state, interrogating Ethernet every ten seconds or so to see if new data have been transmitted. If none have, it sleeps for another ten seconds and tries again. When data have been received, it goes through the operations described above.

## 7.3 In Case of Trouble

Note that, under normal conditions, entering *Tran* at a PC terminal is all that is necessary to effect data transfer and processing. Problems can arise, however, that require more action. It sometimes happens that *MSC*, when invoked from the batch file *RecMSC*, crashes and leaves the Ethernet terminal out in left field. Entering *Tran* under these conditions yields no results because *Rec* is not active on the VAX. It is essential to activate it. To do so, return to the monitor on the PC and get out of *Tran*. Do the same on the VAX printing terminal associated with Ethernet. (Hit CTRL/C to accomplish both of these.) When the prompt appears on the printing terminal, enter @ RECMSC; this should bring *Rec* into action. With this accomplished, enter *Tran* on the PC. Data should transfer successfully. If they do not, consultation with an expert will be necessary.

It should be noted that the crashing of *MSC* referred to above is as yet unexplained. It occurs only occasionally and, when @RECMSC is re-entered, it has always performed flawlessly. An intermittent bug like this is always very difficult to find, and in this case not worth the effort.

## 8. PROGRAM MSC

*MSC* is the program that performs all isotope ratio calculations; it is the report sheet generated by this program that is our *raison d'être*. Because of the numerous demands placed on us by the analytical requirements of our samples, there are many branches in this program that only occasionally are followed, and the logic can sometimes become rather complex.

The organization of this program dates back to antiquity; there are many vestiges of the original version in today's code. A large part of the original organization was dictated by the limited amount of memory available (64 K) on the PDP 11/34 that served as our host computer. It was impossible for the entire program to fit into core at once--it wasn't even close, in fact. This necessitated breaking the program into a calling program and a series of overlays. These overlays were originally named *Calc1*, *Calc2*, ..., *Calc7*. There were no batch files in those early days, necessitating manual entry of all commands to the Linker each time a change was made in the program. This was a tedious process! The names were therefore shortened to *C1*, *C2*, ..., *C7*. Several of the routines grew too large for the core (128K) of our PDP 11/23 (successor to the 11/34), forcing their division into two parts. Thus we have *C3a* and *C3b*, *C6a* and *C6b*, and *C7a* and *C7b*. The subroutines still bear these names, although the divisions between them have become increasingly arbitrary.

There are two versions of *MSC* residing on the VAX. The version called *MSC* resides in both the [MSCALC] and the [DHS.CALC.NEW] directories. The latter also contains a version called *DHS*, which is useful while debugging. It is identical to *MSC* except that it is compiled and linked with all subroutines activated for debugging (FOR/DEBUG/NOOPT) and with all output directed to the terminal as the default device. See Chapter 17 in DEC's manual, "Programming in VAX FORTRAN" for instructions on how to use the debugger.

This chapter is arranged such that a short description of the organization is given first, followed by an overview of the logic of the program. Detailed descriptions of each subroutine then follow.

### 8.1 Organization

*MSC* consists of a calling program, called *Msroot*, and the series of subroutines mentioned above. A brief summary of the routines is given in Table 8.1.

---

Table 8.1 MSC Subroutine Summary

<u>Name</u>	<u>Description</u>
Msroot	Main line calling program.
C1	Presents menu; determines number of samples to be calculated; entered once per execution of <i>MSC</i> .
C2	Reads <i>Samp</i> file; initializes various parameters; entered for each sample.
C3a	Initializes parameters necessary for isotope dilution and internal calibration calculations.
C3b	Reads <i>Chdata</i> or <i>Sumcts</i> ; sums counts in each subgroup; writes to <i>Sumcts</i> ; applies dead time and bias corrections; performs internal calibration calculations.
C4	Applies background correction; calculates ratios for each run; performs isotope dilution calculations; applies contaminant correction.
C5	Calculates averages and performs statistical calculations for ratios calculated in <i>C4</i> ; performs NBS-500 calculations.
C6a	Prints most of the output sheet.
C6b	Prints rest of output sheet; performs blank correction; performs isotope dilution calculations when no unspiked companion is available; writes results to <i>Out</i> file.
C7a	Updates <i>Samp</i> file; initializes various parameters in preparation for next sample.
C7b	Performs specialized calculations that are activated by special values of the batch code. The most important of these is Nd burn-up calculations.

---

Linking of *MSC* is performed by executing a batch file, *MSC.COM*. It is listed in Table 8.2 and is executed by entering @*MSC*.

---

 Table 8.2 Batch File to Link MSC

```

! MSC.COM
! To link program MSC in normal form.
! The executable program is called MSC.EXE.
! Note that the only difference between this and DHS.COM. is the link
! statement and that dsroot is the root.
! rather than msroot. MSC has no debugging associated
! with it; DHS has full debugging.
  link/executable=MSC Msroot-
! link/executable=DHS Dsroot- (For debugging.)
C1-
C2, C2_Lib-
C3a-
C3b-
C4, C4_Contam_Lib, C4_Iso_Dil_Lib-
C5-
C6a-
C6b, C6b_Blank, C6b_NO_Comp-
C7a, C7a_Update_Cal-
C7b, C7b_A126, C7b_Fractionation, C7b_Burnup, C7b_Bayne, C7b_RIMS
[dhs.lib]outloo-
[dhs.lib]Disk_IO/lib-
[dhs.lib]Screen_Lib/lib-
[dhs.lib]dhs_lib/lib
assign/user_mode sys$command sys$input
r MSC
  
```

---

Each of the principal subroutines is divided into smaller subroutines to avoid unbroken code extending over 20 or 30 pages. Each of these smaller subroutines performs a limited number of operations; the goal was to make the logic easy to follow and to provide easy readability. These routines may be in the same module as the principal subroutine or in a special library module. The latter was chosen when the length of the module would have been awkwardly long (more than 30 pages or so) without creation of a subsidiary library. Note that these are not libraries in the technical Fortran sense. Fortran libraries are specified at the end of the link command in Table 8.2; they are Disk\_IO, Screen\_Lib, and dhs\_lib. The modules with subroutine libraries are C2, C4, C6b, C7a, and C7b; the names of the libraries occur on the same linker command line as the calling module. All subroutines will be described in the relevant sections of this chapter.

## 8.2 Overall Logic

This section is a short summary of the logic followed by the main program. Details of the various routines are given in later sections.

Note that INCLUDE files are used to hold the structures associated with the various files. This is a great convenience to the programmer because any change in a structure need be made only in the one place (followed, of course, by the great pleasure of recompiling all affected modules!). An additional benefit is that the same structure does not appear in the listing of the program over and over again.

The first task upon entering *MSC* is to determine what the analyst requires. In *C1*, there is a menu (Table 8.4) from which the selection is made. The two most common choices are to run samples that have not yet been calculated and to recalculate one or more that have. The samples to be calculated are set up in a queue (order is important for mixed resin bead calculations), and their data are processed in turn. *C1* is accessed only once per execution of *MSC*; its sole function is to determine which samples are to be calculated.

In *C2*, the *Samp* file is read for the sample being calculated. This information includes the record numbers to be read in the *Ratio* file, the *Spike* file (if any), and the *Dtbias* file. The various structures used in the calculations are initialized, and the bias corrections for each subgroup calculated.

In *C3a*, the header for the first output sheet is written to the printer and various parameters are initialized (if required) for isotope dilution and internal calibration calculations.

In *C3b*, the *Chdata* file is read if the data haven't been previously processed; various tests are done to check for statistically invalid values; and the sum of raw counts in each subgroup is calculated. These values are written to the *Sumcts* file. If the sample has already been calculated, the *Sumcts* file is read directly and the above steps omitted. A corrected counts array is created from the original raw counts array by correcting for dead time and bias; if internal calibration calculations are called for, these are executed here. A list of raw counts is printed.

In *C4*, isotope ratios are calculated based on information derived from the *Ratio* file record specified by the analyst prior to analysis. Background correction is applied, and sums and differences calculated. Isotope dilution calculations are performed (if requested), as are contaminant element corrections.

In *C5*, the averages of the ratios are calculated, as are various statistical parameters such as standard deviations and 95% confidence intervals. An outlier test is performed, and averages and statistics redetermined if a run is rejected. NIST-500 (instrument calibration) calculations are performed in this routine only because memory in our PDP 11/23 was exhausted in Subroutine *C4*, which is where they logically belong.

Nearly all calculations have now been performed, and it only remains to print out the results and store them in the *Out* file. *C6a* assembles the atom percents, which have thus far been carried as part of the ratio arrays. It prints the header for the second output sheet and most of the results; the biggest exception is that isotope dilution results are printed in *C6b*. All the ratios, their averages, and associated statistics are printed in this routine.

*C6b* prints the remainder of the second output sheet, primarily isotope dilution calculation results. It also performs a few calculations on those samples that need them; if no unspiked companion is available for isotope dilution calculations, those calculations are performed here. This feature basically applies only to uranium and thorium because we use  $^{233}\text{U}$  and  $^{230}\text{Th}$  for spikes, and these isotopes are not usually found in our samples; this allows both quantitative and isotopic information to be garnered from a single analysis. The other calculations performed in *C6b* are blank corrections. Both of these calculations are performed on averages for the whole analysis rather than on individual runs. At the end of *C6b*, the results are written to the *Out* file.

*C7a* updates the *Samp* file; this primarily involves setting the parameter indicating whether or not the data have been processed to 1; a value of 1 indicates that the data have been processed and removes the record number from the array of samples yet to be calculated.

*C7b* performs specialized calculations; most of these were used extensively for one project and are no longer needed on a routine basis. The only exception is that neodymium burn-up calculations are done here; this is a very important capability for us.

Leaving *C7b* returns control to *Msroot*. If all samples have been calculated, the program is exited; if another sample is in the calculational queue, *C2* is called and the entire calculational process repeated.

### 8.3 Module *Msroot*

*Msroot* is the root program that organizes *MSC* into a series of subroutine calls. The only things done in *Msroot* beyond calling subroutines are a few IF statements to test for completion of execution and setting the output device to 6 (printer) or 7 (screen) as desired. The screen is used as output when *DHS* is linked because it is more convenient for the programmer to have all output readily available there while he is debugging.

To determine whether or not all samples have been calculated, the variable *nr\_times* (a variable incremented with each sample) is compared to *nr\_samples* (the total number of samples to be calculated). If *nr\_times* .GT. *nr\_samples*, it is time to exit; control otherwise passes to *C2* (in the GO TO 100 statement).

### 8.4 Subroutine *C1*

*C1* can be thought of as the bookkeeping module for *MSC*. Its sole functions are to determine which option has been selected from the menu and which samples are to be calculated and in what order. It is entered once per execution of *MSC*; once the queue of samples to be calculated is set up, its job is done. The main subroutine itself has very little logic in it; the code is basically a series of call statements to the subroutines described below.

#### 8.4.1 Subroutines in C1

*C1* has been subdivided into several subroutines. The names of these and a short description of their functions are give in Table 8.3; the names have been chosen to give a reasonable idea of the modules' roles in the program.

---

Table 8.3 Subroutines in C1

<u>Name</u>	<u>Function</u>
Get_Option	Reads option from terminal.
List_Contents	Lists contents of Samp.dat file.
Get_Samps	Determines which samples are to be calculated.
Check_Sequence	Checks sequence samples are to be calculated in and alters as necessary.
Get_Printer	Reads which output device is to be used.

---

#### 8.4.2 Function Get\_Option

*Get\_Option* is a function returning the integer value of the option chosen by the analyst. This value is chosen from a menu that appears in its own window on the terminal screen. The menu is shown in Table 8.4. It is the only menu in *MSC*; all calculations follow from the selection made here.

---

Table 8.4 Menu for MSC in C1

Program branch options

- 0 for normal calculations
- 1 to recalculate
- 2 to reject runs
- 4 for individual channel list
- Sum for any combination:
  - 3 for recalc & reject
  - 5 for recalc & list
  - 6 for reject & list
  - 7 for all three
- 9 to list Samp.DAT contents
- 10 to select output device

---

The meanings of these options are described in the next few sections.

#### 8.4.2.1 Option 0

Option 0 is the one most commonly used. It is used when data are being processed for the first time, and, since there is no reason to collect data without processing it, virtually all samples go through this operation. All samples whose value of *samp.calc\_index* is 0 are included in the array to be calculated.

#### 8.4.2.2 Option 1

This option is similar to option 0 except the records of samples to be calculated are entered manually. The value of *samp.calc\_index* is irrelevant; the program will process the data for whatever records are entered.

#### 8.4.2.3 Option 2

It is occasionally desirable to prevent one or more runs from being calculated. This can happen, for example, when an arc occurs in a run and, even though the results are bad, the statistical routines do not reject all ratios associated with it. Selecting this option allows the runs to be rejected to be entered manually. This feature is not often used; rejecting runs at the start or the end of an analysis (far and away the likeliest times to reject runs) is more easily effected by altering either the first or the last run in the *Samp* file record through use of program *File* (Section 4.9).

#### 8.4.2.4 Option 4

This option is used only for diagnostic purposes. There are times when a problem (usually electronic in nature) is subtle enough that a list of the number of counts in each channel is desired. Choosing this option is one way to get such a list. Going through program *File* to list the contents of *Chdata* is more convenient in most cases; see Section 4.10.

#### 8.4.2.5 Options 3, 5, 6, 7

These options are merely combinations of options 1, 2, and 4. The combination desired is determined by adding up the individual options, as indicated in Table 8.4.

#### 8.4.2.6 Option 9

This option will list the contents of *Samp.dat* on the screen or printer. It is most often used by the programmer, rather than the analyst, to help locate a sample for use as a test case for program modification. The record numbers, sample identification codes, and titles are listed between the bounds chosen by operator.

#### 8.4.2.7 Option 10

This option allows the operator to choose among three output devices. It is especially convenient when samples are being recalculated. See Section 8.4.6 below for a description of the options.

### 8.4.3 Subroutine List\_Contents

This subroutine lists the contents of the *Samp.dat* file to the output device selected from within *List\_Contents* itself. It was found by experience that operators normally forget to select the device through use of option 10 when they want a list to the screen; they received instead a list to the VAX printer. The operator can enter the first and last records to be listed if so desired; this is to avoid the tedium of watching the first 300 entries glide by when you know your area of interest is above that number. Entering zero for either entry causes the program to default to the extrema (1 for the first record and 400 for the last).

The information listed includes the file record number, the instrument number, element symbol, sample identification code, calculation index, and the record numbers for the first run in the *Chdata* and *Sumctis* files.

This option is more often used by the programmer than by the analyst. Its most common use is to locate a sample suitable for a test case in program debugging.

### 8.4.4 Subroutine Get\_Samps

This subroutine identifies which record numbers in the *Samp* file hold data that is to be processed. This is done in one of two ways, depending on the value of the option chosen by the analyst (See the menu in Table 8.4). If the option is 0, identification of the samples is automatic and no further input is required. If the option is 1, the record numbers must be entered manually.

Samples whose data have not yet been processed have a value of 0 for the calculation index (*samp.calc\_index* = 0). It was experimentally determined (unplanned, of course) that it takes quite a long time (over 45 seconds) to read all 400 records in the *Samp* file searching for those in which *samp.calc\_index* (*icalc* on the PC's) is 0. An array that is stored in the bookkeeping record of *Samp* (record number 401) was established to hold the record numbers of samples whose data have not been processed. This array is updated automatically in program *Rec*, where the record number for each new sample is added to the array, and *MSC*, where the record number for the sample whose data have just been processed is removed. It is also updated when the *Samp* file is edited in program *File*: changing *samp.calc\_index* will trigger the appropriate adjustment in this array.

The name of this array is *samp\_rec\_to\_be\_calc* when it is read from the disk file; it is called *samp\_to\_be\_calc* for use in *MSC*. There is room for 150 samples, which is gross overkill; it is inconceivable that as many as 150 samples would need to be calculated for the first time. On the other hand, it isn't clear what a safe maximum number would be; 150 was selected so we wouldn't have to worry about it.

Most of the array will be filled with 0's, of course. If there are no entries in the array, the number of samples to be calculated will be 0. In this case, *MSC* gives a warning message and returns control to the monitor.

#### 8.4.5 Subroutine Check\_Sequence

This subroutine is called only when the option chosen is 0 and the number of samples to be processed exceeds 1. It is necessary only because there are certain samples that are calculated as related pairs; one member of each pair must immediately follow the other in the calculational queue. This situation arises when plutonium and uranium are run from the same filament; such samples are almost always loaded on resin beads. Each element has an isotope at mass 238, and each element's spectrum must be corrected for the contribution of the other. The calculational queue must have such analyses arranged as Pu, U pairs, in that order, and there can be no other samples between the members of a pair.

Altering the calculational sequence is necessary only if, during data acquisition, the two elements of the pair were stored in non-contiguous records in the *Samp* file on the local data system. This can happen occasionally if Pu is stored in the last record and U in the first. It can also occur if the analyst decides to run all Pu's on a wheel first and then run the U's. Program *Tran* sends file records to the VAX in the order it encounters them in the *Samp* file. When *Check\_Sequence* encounters such a situation, the U member of the pair is inserted in the queue immediately after the Pu member.

#### 8.4.6 Function Get\_Printer

This function serves a purpose similar to that of *Devin* (Section 11.11). It asks the operator to select which output device he wants the program to use and returns an integer value denoting the selection. It is especially useful when the programmer is debugging and it is convenient to have all output listed on the screen. All samples run on the two "cold" instruments have their output listed on the printer associated with the VAX; samples run on the TRU instrument have their output directed to the printer in the TRU lab for the first time and to the VAX printer on subsequent ones. There are thus three output options, assigned as follows:

- 6: VAX printer
- 7: terminal screen
- 8: TRU printer

These numbers are the logical unit numbers by which they are addressed in Fortran; any number outside the above range is assigned a value of 6.

#### 8.5 Subroutine C2

*C2* initializes many parameters for subsequent calculations. It is entered each time a new sample is calculated (in contrast to *C1*, which is entered only once for each execution of *MSC*). *C2* reads the *Samp* file for each sample in turn, storing the information in a structure called *samp*. Like most of the other principal subroutines, *C2* itself consists mostly of calls to smaller subroutines and functions. It has in addition a library associated with it containing routines that initialize most of the structures used in *MSC*; this is not a library in the formal Fortran sense but is a module holding routines called by the program. Disk files are read as necessary.

Subroutines associated with *C2* are listed in Table 8.5, together with a short description of their functions.

---

Table 8.5 Subroutines in C2

<u>Name</u>	<u>Function</u>
Init_C2	Initializes various parameters
Check_Validity	Verifies the validity of values for several crucial parameters.
Set_Mix_Calcs	Sets calculational queue for mixed resin bead samples.
Set_Bias_Tau	Sets the dead time and bias corrections for each sample.

---

All operations in *C2* are performed automatically and use the value of the option read in *C1* and information read from disk files to determine which branches to take.

#### 8.5.1 Subroutine Init\_C2

This routine initializes various parameters. The output device is set equal to 8 for samples analyzed on the transuranium instrument. The variable *imix* is set to 5; see Section 8.5.4 for a full description of this important variable. The value of the first run is set to 1 if it is 0 because a value of 0 for a run number has no meaning. Various variables dealing with rejected runs are also set.

#### 8.5.2 Subroutine Check\_Validity

This routine verifies that certain crucial parameters have valid values. It verifies that at least one subgroup has a sweep factor greater than 0; nonsense will result if all sweep factors are 0 as this would mean no data were taken. The value of the record in the *Ratio* file must be greater than 0 and less than 300; trying to read record 0 of a file gives Fortran a tremendous case of heartburn--execution of the program is terminated forthwith. The maximum valid value for the *Ratio* record is 300. The number of runs is checked to be sure it is greater than zero--zero has no meaning.

#### 8.5.3 Subroutine Set \_Mix\_Calcs

This routine addresses the situation that arises when plutonium and uranium are run sequentially from the same resin bead. The analysis consists of running plutonium first and then uranium; plutonium ionizes at a lower temperature than uranium, and a crude separation of the two elements is achieved. Both elements have isotopes at atomic mass number 238. The resolution of our mass spectrometers is insufficient to separate them. It is therefore necessary to apply a correction to this mass position in the spectrum of each element. More details of the scientific issues of this operation are given in Section 11.6 of the Operator's Manual.

From the programmer's point of view, it is necessary to process the data for each sample of the plutonium-uranium pair twice. The first pass calculates isotopic ratios on the basis of the counts measured when the element in question was analyzed. The second pass applies the appropriate correction to the 238 mass position. It is the results of the second pass that form the basis of the final report. This two-pass calculational algorithm is necessary because the isotopic composition of neither element is known in advance of the analysis. To be mathematically rigorous, it would be necessary to employ an iterative algorithm repeating the calculations until the value of some critical isotopic ratio converged. In the real world of science, however, this is unnecessary: If the second pass through the calculations doesn't give decent results, more passes won't help. The chemist would rightly have no faith in the analysis.

For the calculations to be correctly performed, certain conditions must be met. These are that both members of the pair must be in the calculational queue and that the plutonium member of the pair must be calculated immediately before the uranium. The calculations fail if these conditions are not met.

All this leads to a certain complexity in the logic. There are several variables whose values dictate the path the calculations will take. These will be discussed below.

#### 8.5.3.1 Variable *mix*

*Mix* takes on only two values: It is set to 0 for most samples and to 1 when the sample in question is one of a mixed pair. When *mix* = 1, if other critical parameters have valid values, the two-pass correction algorithm outlined above is invoked.

#### 8.5.3.2 Variable *recalc*

*Recalc* is a variable that tells the program whether or not the data for the sample in question are being calculated for the first time or RECALCulated. It has a value of 0 for the first processing and 1 for subsequent ones.

#### 8.5.3.3 Variable *pass\_nr*

This variable is associated with the *samp* structure and is thus indicated by *samp.pass\_nr*. It is 0 during the first calculation pass through the data, 1 for the second, and 2 thereafter. The value of *pass\_nr* is tested when assigning a value to *recalc*.

#### 8.5.3.4 Variable *imix*

*Imix* is the variable that controls branching for each pass through the calculations. It is assigned a value during execution of *MSC* and is not part of the *samp* structure. It has a value of 5 for most samples and takes on values of 1 through 4 when a uranium-plutonium pair is being calculated. These values, along with those of other parameters, are listed in Table 8.6.

---

Table 8.6 Variables Used in Calculation of Uranium-Plutonium Pairs

---

<u>Element</u>	<u>calc_index</u>	<u>pass_nr</u>	<u>mix</u>	<u>imix</u>
Pu	0	0	1	1
U	0	0	1	2
Pu	0	1	1	3
U	0	1	1	4

---

When data from a mixed pair are being reprocessed, *pass\_nr* is initially 2, but is set in accordance with the above table during data-processing. The value of *recalc* is set to 1 to indicate to the program that data are being reprocessed.

Note that, to start all calculations from scratch, both the plutonium and the uranium sample must be reinitialized using program *File* (See Section 4.4).

#### 8.5.4 Subroutine Set\_Bias\_Tau

This routine assigns values for the dead time (the algebraic symbol for dead time has traditionally been the Greek letter tau) and a bias to be associated with each mass position. The necessary information is read from the *DTBias* file. The record number in this file is numerically equal to the instrument identification number and is contained in the *Samp* file (*samp.mch\_nr*). The information from the file is contained in structure *dtbias*. See Section 3.9 for the organization of this structure.

A dead time and four bias corrections per mass are associated with each instrument. Dead times are stored in units of microseconds and are used unaltered in *MSC*. Which value for the bias correction is used depends on the mass range covered by the analysis; see Table 4.4. The most commonly used value is *dtbias.bias(4)* because it is assigned when the isotopic masses are over 200. Masses between 150 and 200 are assigned *dtbias.bias(3)*; those between 100 and 150 are assigned *dtbias.bias(2)*; those below 100 are assigned *dtbias.bias(1)*. The first value (*dtbias.bias(1)*) is also used to hold the extra bias required for plutonium when it has been loaded on a resin bead. This is a substantial fraction of the total bias and is added to the value held in *bias(4)* before bias corrections are calculated for each mass. If plutonium is being analyzed as a solution loading and this additional bias is not required, a batch code of AQ should be used.

The reason for this different bias is thought to be due to the fact that data are taken on different parts of the fractionation curve. A lower count rate for plutonium is used when mixed beads are being analyzed to reduce so far as possible contamination from <sup>238</sup>U. Lower temperatures are required, and a different part of the fractionation curve obtains. It is not clear whether or not this is the complete explanation--chemistry occurring on the filament may also play a role.

To determine the bias correction to be applied to each mass position, the value of the bias correction per mass is multiplied by the difference in mass between the base mass in the ratio file (*ratio.base\_mass*) and the mass associated with the position in question. The bias for whatever mass is specified for *ratio.base\_mass* is thus defined as 1.0000.

The value calculated in this multiplication is subtracted from 1.000 to arrive at the bias to be applied to the mass position in question.

Algebraically:

$$\text{bias} = 1.0 - (\text{ratio.base\_mass} - \text{mass}(i)) * \text{bcpm}$$

where *mass(i)* is the mass associated with the mass position in question and *bcpm* is the bias correction per mass. As an example, consider a typical uranium analysis. *ratio.base\_mass* = 238; *mass(i)* = 235; *bcpm* = 0.3% per mass. The bias would then be  $1.0 - (238 - 235) * 0.003 = 0.9910$ .

Some situations require special consideration when bias is being calculated. The background position is used to monitor electronic noise in the counting system; bias is obviously not relevant to it and its value for this mass position should be assigned as 1.0. The mass number of a background mass position is assigned an arbitrary value that must differ from that of the base mass by more than 20. A value of 315 is often used for uranium, for example, because a location near mass 231.5 is often used for background. It should be noted, however, that the number assigned to the background position need have nothing whatever to do with its location in the spectrum. There are only two criteria that must be met: the value assigned must differ by more than 20 from the base mass, and the location of the position must not be that of an actual peak. (Steer clear of 232, for example--Th<sup>+</sup> occurs there.)

A second special case dates back to the days when our vacuum systems were substantially poorer than they are today. 238-U tailed into the 236-U mass position due to gas phase collisions and required correction. Mass 365 (for 236.5) was monitored; years of experience showed that the multiplicative factor needed before subtraction from 236 was 0.6. This has been left implemented in the program but is not in current use; it is invoked whenever a mass of 365 is called for, so this mass assignment should not be used otherwise.

A third situation arises when a plutonium-uranium pair is being analyzed. 235-U is monitored for correcting the 238 mass position for uranium. Clearly the uranium bias correction needs to be applied in this case rather than that for plutonium. This has been implemented in the program.

#### 8.5.5 Library C2\_Lib

*C2\_Lib* is a module holding routines that initialize some of the structures used in *MSC*. The *\_Lib* is not meant to imply that this is a library in the technical Fortran sense that requires that the linker be informed of it. *C2\_Lib* is basically a calling subroutine named *Set\_Structures* that consists of a series of calls to the initialization routines, each of which addresses a single structure.

The structures initialized in *Set\_Structures* are: *spike*, *bead\_comp*, *blank\_comp*, *el\_info*, and *out*. Each of these is initialized in its own subroutine whose name is formed by using *Set\_* as a prefix and adding to it the name of the structure. Thus, for example, the routine that initializes the *bead\_comp* structure is named *Set\_Bead\_Comp*.

The contents of each routine are basically a series of assign statements; no particular logic is involved.

Note that the *Ratio* file is read in *Set\_Structures*. It is used in *MSC* without change, so there is no initialization routine for it.

## 8.6 Subroutine C3a

*C3a* is the first part of the subroutine originally named *C3*; the second is, surprisingly enough, named *C3b*. The function of these routines is to reduce the 32 raw data points for each mass position to one value, corrected for dead time and bias.

Subroutine *C3a* itself is now no more than a series of call statements to the routines that do the actual work. These routines are primarily for initialization of various parameters; most of the mathematical operations attendant on data processing are carried out in *C3b*.

The subroutines associated with *C3a* and which are compiled with it are listed in Table 8.7

---

Table 8.7 Subroutines in C3a

<i>Init_C3</i>	Initializes variables.
<i>Write_Header</i>	Prints header to output sheet.
<i>Set_Spike_Calcs</i>	Initialization routine for isotope dilution calculations.
<i>Set_Int_Cal_Calcs</i>	Initialization routine for internal calibration calculations.
<i>Set_IC_Masses</i>	Identifies masses used in internal calibration.
<i>Correct_238_Pu</i>	Calculates Pu ratio for future correction of mass 238 position for plutonium in internal calibration calculations.

---

Each of these routines is described in more detail below.

### 8.6.1 Subroutine *Init\_C3*

This short routine initializes variables for use later on. One called *orig\_nr\_sg* is set to *samp.high\_subgroup* because, during internal calibration calculations, what might be called synthetic subgroups (mass positions) are added to hold values for mass positions to which corrections have been applied. A LOGICAL variable called *read\_sumcts* is defined to indicate whether or not the *Sumcts* file is to be read or not; it is not read when the data for the sample are being processed for the first time and is read for subsequent processing. A variable used during internal calibrations calculations, *instd*, is set to 0; its function is

described in Section 8.6.4. The sweep rate is converted to units of milliseconds for convenience.

#### 8.6.2 Subroutine Write\_Header

This routine prints header information for the first output sheet.

#### 8.6.3 Subroutine Set\_Spike\_Calcs

This routine is entered when isotope dilution calculations are called for. One of the critical pieces of information necessary for such calculations is the identity of the isotope dilution ratio (in terms of mass units--e. g., 238/233). There are two places where it might get this information. It is one of the parameters stored in the *Spike* file; it is also read during execution of *MSR* in *Setup* (see Section 5.2) and is thus stored in the *Samp* file. If the value in the *Samp* file is zero, the information in the *Spike* file is used; if *Samp* has non-zero values for the numerator and denominator of the ratio, this value is used.

The location of the mass position to be used for the background correction is identified for use in internal calibration calculations. If internal calibration calculations are indicated, the routine *Set\_Int\_Cal\_Calcs* (next section) is called.

#### 8.6.4 Set\_Int\_Cal\_Calcs

Internal calibration (sometimes called internal standard) is a means to improve the precision of an isotope ratio measurement. The spike has two isotopes, usually of approximately equal abundance, whose ratio is very well known. During the calculations, the measured value of this ratio is compared to the known value, and a bias correction is calculated from the difference on a run-by-run basis. Improvements in precision of five or more are usually realized by this technique in comparison to those obtained using an average bias correction. The logic required for the algorithm becomes somewhat complex, but the underlying principle is simple enough. It is beyond the scope of this manual to describe the algebra in detail; the reader is referred to a previous publication if more information is desired.<sup>6</sup>

Internal calibration requires knowledge of at least two ratios. One of these is always the ratio used as the internal calibrant. The other is the ratio for which an accurate value is desired. For concentration measurements, this latter ratio is that used for isotope dilution. Internal calibration can be used to obtain a more accurate and precise value for any ratio, however, and is not confined to isotope dilution applications.

*Set\_Int\_Cal\_Calcs* initializes *instd* to 1; this variable is used to control branching in the calculations. It is set to 0 for normal isotope dilution calculations and to 1 when internal calibration is being used. *Set\_IC\_Masses* (next section, 8.6.5) is called before *Set\_Int\_Cal\_Calcs* is left. *Set\_IC\_Masses* calls *Set\_IC\_Ratios* (Section 8.6.6).

The subroutine *Set\_IC\_Ratios* (next section) is called before control is returned to the calling module.

#### 8.6.6 Subroutine Set\_IC\_Ratios

This routine identifies the ratios to be used in internal calibration calculations and calculates those that are invariant during the analysis--those having to do with the spike and the unspiked sample. Only the ratios from the mixture of spike and sample vary during analysis--it is this mixture that is being analyzed and which provides the information desired.

More nomenclature needs to be described. That used in the program is taken directly from the algebraic notation used in developing the algorithm. Values for ratios start with the letter *r*; *a*, *b*, and *c* have definitions given in Table 8.8. The sample, spike, and mixture are identified by *s*, *t* (for tracer), and *m*, respectively. Together these make up a series of four-letter variable names that can be bewildering if the notation is not kept clearly in mind. As an example, the variable *ract* holds the value of the ratio of isotope *a* to isotope *c* in the tracer (the word "tracer" is used to distinguish between sample and spike, which both begin with *s*). For the case of uranium, *ract* would usually be the 233/238 ratio in the spike. As another example, *rbc* is the ratio of isotope *b* to isotope *c* in the unspiked sample; for uranium this would usually be the 236/238 ratio.

The program allows for more than one additional ratio to be calculated; this means, for example, that both the 235/238 and the 234/238 ratios for uranium could be refined. There are several variables associated with this capability: *nr\_corr\_masses* is the number of mass positions to be corrected in addition to those fundamental to internal calibration, and *extra\_mass\_sg* is an array to hold the masses in question. Values for the associated ratios in the spike are held in an array called *rdat*, consistent with the nomenclature described above.

The reference value of the internal calibration ratio is also calculated in this routine (the variable is *theor\_ratio*).

The values of all these ratios are used in calculations performed in *C3b*.

#### 8.6.7 Subroutine Correct\_238\_Pu

This routine calculates the 238/239 ratio for plutonium on a mixed bead. It also identifies which subgroup (mass position) mass 238 occupies for spike, unspiked sample, and mixture. It is necessary to correct the 238 mass position for calculation of bias during internal calibration calculations for these mixed samples.

#### 8.7 Subroutine C3b

This routine carries out the mathematical operations associated with reducing the number of data points per mass position from 32 to one; this one number represents the total number of counts collected for the mass position in question for the given run. Depending

on the situation, this process can be quite simple or quite complex. The main C3b routine itself does little beyond calling various subroutines, each of which performs some specialized function. These routines are listed in Table 8.9.

---

Table 8.9 Subroutines in C3b

Init_C3b	Initializes a few variables and opens disk files.
Read_Chdata	Reads the Chdata file.
Set_Major_Peaks	Identifies which peaks are "major" and flags them. Identifies spuriously high or low channels and corrects them. Finds minimum peak width to serve all subgroups.
Find_Peak_Width	Finds the peak width for each mass position.
Calc_Raw_Counts	Calculates the sum of raw counts in each mass position. Writes the values in the Sumcts file.
Apply_Dead_Time	
_Corr	Applies dead time correction.
Apply_Bias_Corr	Applies bias correction.
Apply_Int_Cal	Implements internal calibration calculations.
Calc_IC_Bias	
_Factors	Calculates bias correction on the basis of internal calibration calculations for each mass position.
Calc_Cts_Per_Sg	Calculates the total number of raw counts in each subgroup for the whole analysis.
List_Counts	Lists raw counts on the first output sheet.

---

Much of the work of this routine is executed inside a DO loop (DO 555). This loop's upper bound is the number of runs taken for the sample whose data are being processed, and the data for each run are treated identically. The first job executed is to read in the raw count data from the appropriate disk file. This will be *Chdata.dat* when the data are being processed for the first time and *Sumcts.dat* for subsequent executions of *MSC*. The logic involved when the latter file is read is much simpler than that required for raw channel-by-channel data read from *Chdata*. Although the data in *Sumcts* represent raw data in the sense that dead time and bias corrections have not been applied, there are certain operations carried out before *Chdata* data are condensed and stored in *Sumcts*. In *Chdata*, there are 32 channels for each mass position, but they do not necessarily all lie on the flat-topped portion of the peak. The number of channels included in the sum, determined in *C3b*, is one of the items stored for each run in *Sumcts*. This is essential information because one of the factors in applying dead time correction is the amount of time required to accumulate the given number of counts. (See Section 8.7.6.) Other corrections occur only rarely and date back to the time when multi-channel analyzers were used to accumulate data rather than computers. These analyzers, as they aged, developed a penchant for storing incorrect values of counts in individual channels. These were called "dropped channels" or "noise spikes" when the number of counts was too low or too high, respectively. This problem has not occurred

since computers replaced the analyzers, but it costs nothing to check, and the routines to correct it have been left active.

These are the only modifications made to the raw data before a sum is calculated (over the specified number of channels) for storage in *Sumcts*.

The *Sumcts* file is read in *C3b* when indicated by the value of the logical variable *read\_sumcts*. No manipulation of the raw data is done in this case until dead time and bias corrections are called for. For first-time calculations, several subroutines are called to perform the operations described above (determination of peak width and correction of erroneous channels).

Calculations for instrumental calibration standards (usually NIST U-500) require no application of dead time and bias correction since it is the goal of such calculations to determine those very parameters. All other samples have a dead time correction applied, followed by correction for bias. This latter correction can either be the normal linear correction based on an average bias correction per mass or an individual run-by-run correction when an internal calibration analysis is being performed.

The final operation of *C3b* is to call the subroutine that lists raw counts on the first output sheet.

#### 8.7.1 Subroutine Init\_C3b

This subroutine opens the raw data files (*Chdata* and *Sumcts*) and initializes a few parameters needed in the calculations.

#### 8.7.2 Subroutine Read\_Chdata

As one might suspect from its name, this routine reads the *Chdata* file. It reads 256 data points (channels) when the highest subgroup used is 8 or less and 512 points when it is larger than 8. Two read statements are required in the latter case, and all data are collected into a single array.

#### 8.7.3 Subroutine Set\_Major\_Peaks

A major peak is defined as one that has more than 500 counts per channel. This is an arbitrary definition. Peaks too small are statistically unreliable when it comes to determining peak widths; peaks with at least 500 counts per channel are quite reliable. The distinction between major and minor is used only in the determination of the peak width to be used for all peaks in the run and has no effect on ratios calculated beyond that.

The average of the eight central channels is used to make the distinction between major and minor. These channels were chosen because they should always be on the flat-topped portion of the peak; if they are not, the analysis will almost certainly be useless. If the peak is major, *Find\_Peak\_Width* (Section 8.7.4) is called; it is not called for minor peaks. The last statements in the routine determine the minimum peak width found for all major peaks for the run in question; this is the width that will be used in future calculations and is

the number of channels whose counts will be included in the sum for all mass positions for that run.

#### 8.7.4 Subroutine Find\_Peak\_Width

This subroutine determines the number channels in the peak in question that meet the definition of a flat top. It does this by comparing the number counts in each channel with a test value. This test value is defined as the number of counts three standard deviations below the preliminary average calculated in *Set\_Major\_Peaks*. The standard deviation of any number of counts is its square root. The edge of a peak is defined as three successive channels whose counts are less than the test value. This situation can occur at either the low or the high mass side of the peak, or at both. There is special logic to handle the situation when the counts in less than three channels on either side of the peak fall below the test value. The first channel to be included in the sum is passed on for later use, as is the width of the peak.

#### 8.7.5 Calc\_Raw\_Counts

The total number of raw counts in each mass position is calculated in this subroutine. These sums are corrected only for data-taking anomalies and are not corrected for bias or dead time. They thus represent the raw data. The sum is accumulated for the number of channels defined in *Find\_Peak\_Width*. Sums are accumulated beginning with the starting channel (variable *low\_edge*) and include the number of channels in the flat-topped portion of the peak. Noise spikes are identified and corrected. Dropped channels are also corrected if present; these represent flaws in the electronic storage equipment (multi-channel analyzers) and seem to have become a thing of the past with computer-based data systems.

The last action of the subroutine is to write the sums of counts into the *Sumcts* file and to update that file's bookkeeping record.

#### 8.7.6 Subroutine Apply\_Dead\_Time\_Corr

This subroutine applies the dead time correction to the sums of counts generated in *Calc\_Raw\_Counts*. Dead time arises in a counting system because there is always a period after registering a count during which it is unable to register a second count. Dead times for our systems are usually in the range of 10-20 nsec. The equation for applying this correction is:

$$cc = rc / (1.0 - rc * dt / t)$$

where:   cc = corrected counts  
           rc = raw counts  
           dt = dead time  
           t = time of measurement.

Time in this case is calculated as:

$$t = sr * cy * ch * sf$$

where: sr = sweep rate  
 cy = number of cycles per run  
 ch = number of channels in the peak  
 sf = sweep factor.

All times are converted to seconds.

This correction is applied to all peaks, regardless of size.

#### 8.7.7 Apply\_Bias\_Corr

Linear bias correction based on an average determined through analysis of certified isotopic standards is applied in this subroutine. It is applied after the dead time correction described in the previous section, and is applied to all samples except those used for instrument calibration (usually NBS [NIST] U-500) and those using internal calibration to determine bias.

The *Dtbias* file has stored for each mass spectrometer its own dead time and bias. There are in fact four bias corrections stored for each instrument; the rationale for this was described in Section 8.5.4; see Table 4.4. The bias correction to be applied to each mass position is calculated in Subroutine *Set\_Bias\_Tau* (Section 8.5.4).

The bias correction is a factor by which the raw counts, corrected for dead time, are multiplied. One mass position is defined as having a bias correction of 1.00, as discussed in Section 8.5.4. Bias corrections are normally less than 1.0, but this is only by convention and isn't required.

A standard deviation is calculated for each number of counts calculated in this routine. This is not presently used in calculations; in the past, when extremely small samples were analyzed, it was used as a weighting factor in calculating a weighted average. In the analysis of a rapidly decaying sample, the runs with the most counts usually give more reliable estimates of the ratios than those with fewer.

#### 8.7.8 Apply\_Int\_Cal

This subroutine calculates and applies bias when data for an internal calibration sample is being processed. The idea behind internal calibration (discussed more fully in Section 8.14) is to add to the sample a spike containing two abundant isotopes whose ratio is precisely known. Comparing the measured value of the ratio to the known allows calculation of the bias to be applied to the run in question. This technique improves precision and accuracy by a factor of five or more and has been described in detail in Ref. 6. A general solution to the algebraic equations is used, but only two elements have been analyzed using this technique. These are uranium, where the spike was comprised of the 233 and 236 isotopes, and plutonium, where 242 and 244 were used.

Three different methods of calculating the biases are implemented, more with a view to comparing results than from any necessity. The first is an iterative technique; the other two are direct algebraic solutions to the simultaneous equations describing the system.

The iterative technique uses some original estimate of the bias correction (the value stored in the *DtBias* file) and calculates the reference ratio for the measurement in question. It compares this value to the expected one and calculates a new value for the bias correction. It loops through these steps until the measured ratio agrees with the reference value to within some specified limit (usually 1.0 E-6). The convergence criterion is loosened every ten iterations; calculations are halted after 100 iterations on the assumption that convergence will not occur or, if it does, the data are too poor to yield reliable results. Convergence usually occurs in five or six iterations.

The first of the algebraic solutions is one derived by yours truly.<sup>6</sup> It requires solution of a quadratic equation. The second algebraic solution is a linear one developed by Stein Deron of the International Atomic Energy Agency. Solutions from the three methods should of course yield the same results.

All three solutions require calculating the number of counts due to the spike alone that are contributed by each of the two isotopes that make up the reference ratio. Only in this way can a valid measured ratio be calculated for comparison to the reference value. These calculations involve use of various ratios, as described in Section 8.14. If the sample is part of a uranium-plutonium pair, the contribution of the unwanted member to the 238 mass position must be subtracted away; it is added back in near the end of the subroutine to allow subsequent calculations to be valid.

*Apply\_Int\_Cal* calculates a different bias correction for each run, as opposed to the single value used for an entire analysis in the normal case. These individual bias values are returned as arguments and used in the appropriate place to correct the raw counts array. Internal calibration is seldom used despite the advantages to be gained from it. There is not much call for high precision isotopic measurements of small samples any more, which explains its relatively infrequent use at ORNL.

#### 8.7.9 Subroutine Calc\_IC\_Bias Factors

This routine calculates the average of the bias correction factors calculated for individual runs in *Apply\_Int\_Cal*. It then calculates the bias that would be applied to each mass were this average to be used for that purpose. This is done for comparative purposes only; no ratios are calculated using the bias factors derived from this average.

#### 8.7.10 Subroutine Calc\_Cts\_Per\_Sg

This routine calculates the number of raw counts accumulated during an analysis for each mass position. Each sum is the sum of the sums for each mass position generated in *Calc\_Raw\_Counts*. It gives, for example, the total number of raw counts registered for 238-U during an analysis.

### 8.7.11 Subroutine List\_Counts

Raw counts are listed on the printer in this subroutine. This forms the first output sheet of the analysis. Header and file record information is also listed (Section 8.6.2).

The numbers of counts listed here are those that are stored in the *Sumcts* file and allow hand-calculation of any of the ratios that appear on the second page of output. This allows fairly rapid verification of calculational accuracy should it come into question.

Information for only eight mass positions can be accommodated on one line; if more than eight mass positions were scanned, a second section of the report is generated containing the information for mass positions above eight.

### 8.8 Subroutine C4

This routine, like the corresponding ones for *C2* and *C3*, serves primarily as a calling routine for those that do the actual work. The primary function of the set of subroutines under *C4* is to calculate the ratios called for in the *Ratio* file. This, of course, is exactly the goal of the entire analysis; so this routine is in that sense the keystone of all of *MSC*. Two libraries have been established to keep the listing of *C4* to manageable size: *C4\_Contam\_Lib*, which contains the routines necessary to apply correction for contaminant elements; and *C4\_Iso\_Diln\_Lib*, which contains the routines that perform calculations for isotope dilution analysis. Neither of these is a library in the formal Fortran sense; they are modules to hold the routines and thus reduce the size of the main file.

The routines applied to all samples being processed are described in Section 8.8.1. They are directed at calculating the isotopic ratios defined in the *Ratio* file record specified by the analyst at run time; the record number is part of the contents of the *Samp* file record.

The subroutines called by *C4* and short descriptions of their role are listed in Table 8.10.

---

Table 8.10 Subroutines in C4

Set_Nr_Ratios_Runs	Determines number of ratios and runs.
Set_Corr_Counts	Eliminates rejected runs from corrected counts array.
Find_Sgs_For_Masses	Finds the subgroup corresponding to each mass.
Apply_Bg_Corr	Applies background correction to each mass position.
Corr_Pu_For_U	Corrects Pu of mixed pair for U.
Corr_U_For_Pu	Corrects U of mixed pair for Pu.
Calc_Total_Cts_Per_Sg	Generates sum of corrected counts per mass for each position.
Calc_Diffs	Calculates differences.
Calc_Sums	Calculates sums specified in <i>Ratio</i> file.
Calc_Ratios	Calculates ratios and atom percents.

---

There are additional routines called when a contaminant element is to be corrected for and when isotope dilution calculations are to be performed; these are described in Sections 8.8.2 and 8.8.3, respectively.

### 8.8.1 C4 Subroutines

#### 8.8.1.1 Set\_Nr\_Ratios\_Runs

This subroutine determines the number of ratios to be calculated by inspecting the record specified in the *Ratio* file. Default values for ratios have been established as a convenience for the analyst. When no ratios are called for (i.e., *ratio.ratio\_mass*(1,1) = 0), default ratios are calculated. Default ratios have as their denominator the mass position that had the most counts collected during the analysis. Numerators are those masses appearing in *Sum(1)* (except for the mass of the denominator). The background mass position is also included.

This subroutine also adds to the list of ratios to be calculated those required for calculation of atom percents. The mass positions to be included in atom percent calculations (when the normalizing mass is not zero) are specified in *Sum(1)* in the *Ratio* file; this matter is discussed further in Section 8.8.1.9 and 3.5.2.5 in the Operator's Manual.

It also determines how many runs are to be rejected; and, if the number of runs is greater than 19, it is set to 19 to avoid exceeding the specified dimension of many arrays.

#### 8.8.1.2 Subroutine Set\_Corr\_Counts

The purpose of this subroutine is to adjust the *orig\_counts* array for rejected runs, eliminating their values from the array. It then creates a new array variable, *corr\_counts*, that is initialized to *orig\_counts*. Any subsequent adjustment to the corrected counts array (for contaminant element correction, for example) is performed on *corr\_counts*, leaving *orig\_counts* to hold the array corrected only for bias and dead time.

#### 8.8.1.3 Subroutine Find\_Sgs\_For\_Masses

This subroutine associates a mass position with the masses specified in various variables and arrays in the *Ratio* record. These include the background position, the normalizing mass used in the calculation of atom percents, and the arrays used to calculate sums and differences. It also determines the number of masses in each sum and difference. Finally, it associates with each ratio the mass positions involved, using the mass numbers in the *Ratio* record to identify the correct subgroup. The numbers of the mass positions (e. g., 1-8) serve as subscripts for the *corr\_counts* array, and are used, for example, to identify numerators and denominators for ratios to be calculated.

#### 8.8.1.4 Subroutine Apply\_Bg\_Corr

The purpose of this routine is to subtract the number of counts in the background subgroup from the counts in each other mass position. There is always at least some

electronic noise in the counting system, and this is an effort to correct for it. It is particularly important for the TRU instrument because radioactive isotopes accumulate on the first dynode of the multiplier and decay there; each time an atom decays, it registers as a count. These rates depend on how many samples of what elements the multiplier has seen; they can become dismayingly large and necessitate a change of multiplier.

Before the advent of all-metal vacuum systems and ion pumps in the mid-1960's, the pressure in the instrument was poor enough that it was necessary to correct the 236-U mass position for tailing from the 238-U. Scatter of the beam due to gas-phase collisions at the pressures in those days caused 238 to contribute a few ppm to the 236 position; the magnitude of the correction varied both with pressure and the cleanliness of the sample. Long experience showed that this contribution was about 0.6 times the number of counts measured at mass position 236.5. The program assumes that a subgroup assigned this mass is to be used for this purpose and makes the described correction. Warning: Do not use mass 365 unless you want the correction! The correction is implemented in this subroutine.

#### 8.8.1.5 Subroutine *Corr\_Pu\_For\_U*

When mixed resin bead samples are analyzed, it is necessary to correct the 238 mass position for the element whose data are being processed for the contribution from other element. This operation is described more fully in Section 11.6 of the Operator's Manual.

*Corr\_Pu\_For\_U* corrects 238-Pu for 238-U using information already read from the *Out* file for the resin bead companion sample (U in this case). The correction is based on the number of 235-U counts observed in the run in question; the 235-U/238-U ratio is calculated from resin bead companion data and applied to the sample now being processed. This operation is performed on the second pass through the Pu data. The correction thus applied is often rather imprecise because there are frequently only a few counts of 235-U on which to base the calculation. It is the best that can be done in the circumstances, however. 238-Pu is usually only a minor component of Pu, so the lack of precision has little effect.

#### 8.8.1.6 Subroutine *Corr\_U\_For\_Pu*

This routine is a companion to the one described in the previous section. It corrects the 238 mass position in the uranium member of a uranium-plutonium pair for contributions from the 238-Pu. 239-Pu is monitored for this purpose, and the 238-Pu/239-Pu ratio is known from information for the Pu analysis in the resin bead companion file. This correction is usually quite small, since 238-Pu is normally a minor constituent of Pu; the situation is further aided by the fact that 238-U is usually the major isotope in the uranium sample.

#### 8.8.1.7 Subroutine *Calc\_Total\_Cts\_Per\_Sg*

The sole function of this subroutine is to calculate the number of corrected counts for the entire analysis for each mass position. These totals are thus the sums of the *corr\_counts* array for each run.

### 8.8.1.8 Subroutine Calc\_Diffs

This routine calculates the differences specified in the *Ratio* record using the *corr\_counts* array. Differences are identified for ratio calculations by adding 18 to the difference number; i.e., differences 1, 2, and 3 are identified by 20, 21, and 22, respectively. See the next section (8.8.1.9) for an explanation of this apparently arbitrary assignment.

### 8.8.1.9 Subroutine Calc\_Sums

There are several different sums used in *MSC*. These are specified in the *Ratio* file record for use in ratio calculations. The most commonly used sum is *Sum(1)*, which defines the mass positions to be used in calculation of atom percents. If a sum is wanted and atom percents are not, the normalizing mass can be set to zero or one of the other sums used (*Sum(2)* or *Sum(3)*). Any of the three sums can be used independently of the others.

*Sum(2)* has a special meaning only in isotope dilution calculations where there is no isotopic information for the unspiked companion sample in the *Out* file. In these cases, identified by setting the unspiked companion code to 'NO' and the *Spike* file record number greater than zero, *Sum(2)* must contain the isotopes of the unspiked sample. This is a common source of confusion. For uranium, for example,

$$\begin{aligned} \text{Sum}(1) &= 233 + 234 + 235 + 236 + 238; \\ \text{Sum}(2) &= 234 + 235 + 236 + 238. \end{aligned}$$

Note the absence of 233 in *Sum(2)*. Another example is thorium; in this case,

$$\begin{aligned} \text{Sum}(1) &= 230 + 232; \\ \text{Sum}(2) &= 232. \end{aligned}$$

Keep in mind that a reverse isotope dilution would require suitable adjustment of *Sum(2)*; for thorium, for example, *Sum(2)* = 230.

Sums are identified for ratio calculations by adding 16 to the sum number; i. e., they will be 17, 18, and 19 for *Sums* 1, 2, and 3, respectively. This apparently arbitrary convention arises from the fact that 16 is the maximum number of mass positions that can be scanned; identifiers 1-16 are thus reserved for them. Sums and differences have simply been added on top so that sums are identified by 17-19 and differences by 20-22. This simplifies programming because all relevant variables can be included very simply in the same DO loops.

### 8.8.1.10 Subroutine Calc\_Ratios

This routine calculates the isotopic ratios that are the *raison d'être* of the whole analysis. It uses the appropriate values from the *corr\_counts* array, identified in *Find\_Sgs\_For\_Masses* (Section 8.8.1.3). There is a test to avoid division by 0. Standard deviations are also calculated, although they are not used. They would be used if weighted averages were ever desired. This has happened historically when extremely small samples,

which produced rapidly decaying signals, were analyzed. The averages for these samples were calculated in such a way that the runs with the most counts were given more weight than those with fewer counts. Standard deviations are thus available if needed.

### 8.8.2 Library C4\_Iso\_Diln\_lib

This module is a library that holds the subroutines that perform isotope dilution calculations. They are summarized in Table 8.11.

---

Table 8.11: Subroutines in C4\_Iso\_Diln\_Lib

Iso_Diln_Calcs	Calling routine; swaps arrays for a reverse isotope dilution calculation.
Set_ID_Params	Initializes isotope dilution parameters.
Calc_ID_Ratio	Calculates isotope dilution ratio.

---

#### 8.8.2.1 Subroutine Iso\_Diln\_Calcs

This subroutine is the one called by *C4* when isotope dilution calculations are desired. It is essential to have a positive value for the *Spike* file record number for this to happen. An unspiked companion with the sample identification code of 'NO' is a special case when the isotopic composition of the unspiked sample is not available; isotope dilution calculations are performed in subroutine *C6b* in this case. (See Section 8.11.8).

*Iso\_Diln\_Calcs* itself swaps various parameters if the calculation is identified as a reverse isotope dilution. In this case, the analyst is calibrating a spike solution with some standard reference and wants to measure its concentration for use in other isotope dilution analyses. The potential spike's isotopic composition is often already in the *Spike* file, and that of the reference material is often in the permanent companion section of the *Out* file; an example of the latter is natural uranium (NBS U-950), which is normally used to calibrate uranium spikes and whose isotopic composition is in the *Out* file. It is a convenience for the analyst to be able to use information already stored on disk. Identifying the analysis as reverse isotope dilution causes the program to swap spike and unspiked parameters in this subroutine.

This routine calls the other two listed in Table 8.11 and which are described in the next sections.

#### 8.8.2.2 Subroutine Set\_ID\_Params

This routine performs for isotope dilution calculations a function similar to that performed by *Find\_Sgs\_For\_Masses* (Section 8.8.1.3) for other cases. It associates the correct mass positions with the masses specified in the isotope dilution ratio and identifies the appropriate ratio in the unspiked sample and the spike. If the isotope dilution ratio is, for example, 238/233 for a uranium sample, the calculations need the value of that ratio in the

unspiked sample and the spike, and the location of this ratio must be identified. The routine will also take care of things if the reciprocal of the desired ratio is all that's available. It avoids division by zero, a cause of great indigestion in computers.

### 8.8.2.3 Calc\_ID\_Ratio

This routine calculates the isotope dilution ratio using Hintenberger's form of the equation. This ratio is not a simple ratio of isotopic abundances; its algebra allows use of any isotope for determining concentrations regardless of whether or not the spike isotope is present in the sample. The equation is completely general and is valid for any isotope of any element. The equation and a discussion of it are presented in Section 11.1 of the Operator's Manual.

The standard deviation of this ratio is also calculated and returned to the calling program.

### 8.8.3 Library C4\_Contam\_Lib

This is a library (but not in the formal Fortran sense) that contains the routines that correct a spectrum for contamination elements. The maximum number of contamination elements it will handle is three. The subroutines in this library are listed in Table 8.12

---

Table 8.12 Subroutines in C4\_Contam\_Lib

Contam_El_Corr	Calling routine; it is called by C4.
Set_Contam_Index	Assigns an index to each mass position that describes its contributors.
Find_Contam_Sgs	Locates interference-free subgroups for each contaminant element.
Calc_Theor_Ratios	Calculates the expected value of the ratios for which there are interferences.
Apply_Contam_Corr	Applies correction for contamination elements to each affected mass position.
Set_Pct_Array	Converts two-dimensional arrays into one-dimensional to allow subroutine <i>Look</i> to be used.

---

#### 8.8.3.1 Subroutine Contam\_El\_Corr

This is the subroutine called by *C4* and is invoked when the first two characters of the array *contam\_el\_sym* are not blank. A maximum of three contaminant elements is allowed; two is the maximum encountered analytically so far (Mo and Ru in Tc). If contaminant element symbol (1) is blank, you will be unable to make corrections for contaminant 2 or 3; fill the symbol array from the beginning. It is important to remember that the isotopic compositions of the contaminants indicated in this array must be in the *Massab* file. The natural abundances of all elements are included in this file; if the contaminant has an isotopic composition different from the natural element, it must be entered in the file. Do not use the normal element symbol for this purpose! The computer will be confused as to whether

you want the natural element or the altered one you are entering. Create your own symbol--UU, for example, for an altered uranium contaminant.

For the corrections to work, each contaminant element must have at least one isotope monitored during the analysis that is unique to it. As pointed out above, the isotopic composition of the contaminants must be known and stored in the *Massab* file.

*Contam\_El\_Corr* serves as a calling program, accessing in turn the routines that do the work.

### 8.8.3.2 Set\_Contam\_Index

Contamination element corrections are based on the assignment of an index to each mass position. Table 8.13 summarizes how the values are assigned, which is the function of this subroutine.

---

Table 8.13 Values of *contam\_index*

<u>Value</u>	<u>Definition</u>
1	background.
2	sample isotope only.
3	contaminant 1 only.
4	contaminant 2 only.
5	sample + contaminant 1.
6	sample + contaminant 2.
7	sample + contaminant 1 + contaminant 2.
8	contaminant 1 + contaminant 2.
9	contaminant 3 only.
10	sample + contaminant 3.
11	sample + contaminant 1 + contaminant 3.
12	sample + contaminant 2 + contaminant 3.
13	sample + contaminant 1 + contaminant 2 + contaminant 3.
14	contaminant 1 + contaminant 3.
15	contaminant 2 + contaminant 3.
16	contaminant 1 + contaminant 2 + contaminant 3.

---

Thus, for the simplest and most common case of one contaminant element, one subgroup will have *contam\_index* = 2 (sample isotope), one will have *contam\_index* = 3 (contaminant 1 isotope only), and one will have *contam\_index* = 5 (sample + contaminant 1). More than one mass position can have these values, but at least one must have each value for the calculations to be valid.

This subroutine determines which index to assign to each subgroup on the basis of which masses in the sample and contaminants match the mass assigned to it.

### 8.8.3.3 Subroutine Find\_Contam\_Sgs

This routine identifies which subgroups have monitored interference-free contaminant element isotopes. The counts in these subgroups are used later to correct sample peaks for which contaminant peaks interfere.

### 8.8.3.4 Subroutine Calc\_Theor\_Ratios

This routine calculates the expected ratio of two contaminant isotopes when more than one interference-free isotope was monitored for one element. This is an uncommon occurrence (usually only one free isotope per contaminant element is monitored), but there have been occasions when it was desirable. The measured value of the ratio is compared to the expected; if they differ by more than 5%, no contaminant corrections are made for that element for the run in question.

### 8.8.3.5 Subroutine Apply\_Contam\_Corr

This is the routine that applies corrections for contaminant elements. Corrections are made on the basis of the number of counts collected in interference-free mass positions for the contaminant elements. These values allow calculation of the number of counts the contaminant element contributes to the mass position of interest, which will also have a contribution from a sample isotope. The counts thus calculated are subtracted from the total number of counts for that subgroup; the difference is the number of counts due to the sample isotope. In algebraic form:

$$cc = uc - R * cts$$

where: cc = counts corrected for contamination.

uc = uncorrected counts.

R = ratio of counts in mass position being corrected to those in interference-free position.

cts = counts in interference-free mass position.

### 8.8.3.6 Subroutine Set\_Pct\_Array

The purpose of this subroutine is to insert the one-dimensional arrays returned by Subroutine *Look* (Section 11.17) into a two-dimensional one. It is a great programming convenience to have the compositions of the contaminant elements in a 3 X 10 array.

## 8.9 Subroutine C5

The main function of this subroutine is to calculate averages and standard deviations for the individual ratios calculated in Subroutine *C4*. Another important function performed here is the calculations involved when an instrument calibration standard has been analyzed. *C5* itself is a calling routine, accessing as needed the routines that perform the necessary tasks. These routines are summarized in Table 8.14.

---

Table 8.14 Subroutines in C5

NBS_500_Calcs	Calculates instrumental bias and dead time when a calibration standard has been analyzed.
Calc_Avgs	Calculates averages and standard deviations for all ratios.
Outlier_Test	Checks for consistency in atom percent calculations.

---

### 8.9.1 NBS\_500\_Calcs

This is the routine called from *C5* to perform instrumental calibration calculations. The most common time these are required is when a new multiplier has been installed, but they may be necessary at other times as well. The analysis requires a standard with certain characteristics; NBS (NIST) U-500 is the only one we have ever used for this purpose, but there is no reason why another couldn't be. The software is completely general and, as long as the *Stand* file (Section 4.8) is set up properly, there should be no problem. In particular, the two ratios used for dead time and bias correction calculation must be identified correctly and must also meet certain criteria. The ratio used for estimation of bias should be close enough to 1.00 that dead time corrections can be neglected (i.e., the count loss will be the same for both peaks). The ratio used to estimate dead time should be one-sided enough that count loss differences between the two peaks are significant. For the case of uranium and NBS U-500, the bias ratio is 235-U/238-U (certified value 0.9997), and the dead time ratio is 234-U/235-U (certified value 0.01043).

Bias corrections in the ORNL instruments are divided into two components: dead time and everything else. Dead time is a measurable characteristic of the counting system; other bias is due to so many sources, none of which is readily accessible, that it is practicable only to lump them together. Some of these sources of bias are isotopic fractionation during the evaporation process; isotopic discrimination in extracting the ions from the ionization region; mass-dependent transmission efficiencies through the mass spectrometer (should be small); and discrimination during the conversion process at the first dynode of the multiplier, where each ion is converted into a pulse of electrons. Of these, the effect of sweeping the high voltage, and thus changing the extraction field experienced by the ions, is almost certainly the largest. Instruments in which mass position is changed by scanning the magnet do not exhibit mass discrimination as large as those where the voltage is scanned. The best case of all, of course, is not to scan at all but to collect all ions simultaneously. Our VG-354 can do this for up to five isotopes.

Dead times are usually in the range of 10-20 nsec, and bias corrections for uranium usually fall in the range 0.2-0.4% per mass. Bias usually increases as a multiplier ages, and values greater than 0.4% are acceptable as long as results of analyses of reference standards (e. g., NIST-U-010) are within acceptable limits. Bias is also a function of mass, increasing as the relative difference from the reference mass increases. Thus, a bias correction determined for uranium is not valid for the rare earth region. There also seems to be a chemical component to bias; the bias for iron is quite different from that for nickel, for

example. The point here is that you need a reference material for the specific element under analysis to determine the bias correction for that element.

NIST (NBS) U-500 is used for instrument calibration because its isotopic composition is ideal for this purpose. The ratio of 235-U/238-U is certified as 0.9997; this is close enough to one to assume that no correction needs to be applied for count loss (dead time). This does not mean that there has been no loss of counts due to dead time, but that the losses will be equal for each isotope. The measured value of this ratio is used to calculate the correction needed for other sources of bias. The ratio of 234-U/235-U is then used to calculate the dead time (see Section 8.9.1.3 for more information).

*NBS\_500\_Calcs* calls several subroutines, which are summarized in Table 8.15.

---

Table 8.15 Subroutines in NBS\_500\_Calcs

Set_NBS_Params	Initializes various parameters.
Calc_Bias	Calculates the bias correction per mass.
Calc_Tau	Calculates dead time.
Calc_NBS_Counts	Calculates the number of corrected counts collected during an analysis for each subgroup.

---

#### 8.9.1.1 Subroutine Set\_NBS\_Params

This routine initializes several parameters to their correct values preparatory to performing calibration calculations. It identifies which subgroups contain the members of the two calibration ratios; the nomenclature of the variable names assumes uranium is the calibration element, but that need not be the case. 235-U appears in both ratios; it is assigned two variable names to cater to the possibility that sometime in the future a calibration standard will use four different isotopes for the two ratios. Once the subgroups associated with the four isotopes have been identified, the values of the two ratios are calculated from the certified isotopic composition. This information is returned to the calling program.

#### 8.9.1.2 Subroutine Calc\_Bias

It is in this subroutine that the bias correction is calculated for each run. For NIST U-500, the 235-U/238-U ratios are used, as explained above. The bias correction per mass is calculated using the following equation:

$$bcm = (1.0 - Rt/Rm) / (m2 - m1)$$

where: bcm = bias correction per mass.

Rt = certified value of the ratio (= 0.9997 for NIST U-500).

Rm = measured value of the ratio.

m2 - m1 = difference in mass between the two isotopes  
(= 3 for NIST U-500).

After calculation of the bias, a call to the routine that calculates the dead time is made; this routine is described in the next section. A set of bias correction factors, one for each mass position, is calculated before execution returns to the calling program.

### 8.9.1.3 Subroutine Calc\_Tau

This is the routine that calculates the dead time. Its name derives from the fact that the Greek letter tau has traditionally been used to represent this parameter.

Calculation of dead time requires knowledge of the count rates, which are the first things calculated in the subroutine. Dead time is then calculated from the equation:

$$dt = \frac{(Rd - cr5 / cr4 * b4 / b5)}{cr4 * b4 * (Rd - 1.0)}$$

where: dt = dead time.

Rd = certified value of dead time ratio (= 0.01043 for NIST U-500).

cr4 = count rate of 234-U (for NBS U-500 only).

cr5 = count rate of 235-U (for NBS U-500 only).

b4 = bias for 234-U

b5 = bias for 235-U

A dead time is calculated for each run.

### 8.9.1.4 Calc\_NBS\_Counts

This routine calculates the number of corrected counts per subgroup for calibration standard analyses. The counts are corrected for dead time and bias, calculated as described in the previous sections. Count rates corrected for bias are also calculated.

### 8.9.2 Subroutine Calc\_Avgs

This routine calculates the unweighted averages and standard deviations for all samples; except for instrument calibration standards, the ratios involved will have been calculated in Subroutine C4. These are the averages listed on the second output sheet. Statistical parameters are also calculated. These include the standard deviation, standard error of the mean, and the 95% confidence interval. Standard error of the mean is the standard deviation divided by the square root of the number runs. This means it will be numerically smaller than the standard deviation, making it the preferred unit of imprecision for many commercial vendors. Our experience has always been that standard deviation approximates the ill-defined term reproducibility far better than standard error of the mean. By this is meant that, over replicate analyses, a new analysis will usually fall within the limits

defined by the standard deviation; standard error of the mean is too tight. The 95% confidence interval, determined by referring to a table of comparative values, is the unit preferred by the professional statistician (C. K. Bayne) consulted on this matter. It almost always has a value quite close to the standard deviation, so the distinction is not very important in a practical sense. Note that, to quote more than one, or at most two, significant figures for relative standard deviation makes no sense. To quote a relative standard deviation of, say, 0.343% provides no more information than to quote 0.3%. This is because the significant figures in the value of the ratio become meaningless beyond the influence of the first significant figure of the relative standard deviation.

Statistical outliers are identified in by calling subroutine *Outlie* (Section 11.24) and eliminated from the averages.

### 8.9.3 Subroutine *Outlier\_Test*

One of the little thorns in the analyst's side is that customers want the sum of the atom percents to add up to 100.000000000. They don't understand the concept of 'statistical variation' or 'rounding error' to explain why the sum is, say, 100.001 or 99.999. A major contributor to this problem is the elimination of outliers from inclusion in the averages of the atom fractions. If a ratio for one run is rejected, then the sum accumulated for calculation of the average for that ratio is inconsistent with the sums for the other ratios. This often leads to the sum of the atom percents differing from 100.00.

*Outlier\_Test* was devised to address this problem. If an outlier for one atom fraction ratio has been eliminated, all the atom fractions are normalized to add to 1.000.

### 8.10 Subroutine *C6a*

Subroutine *C6*, like *C3*, became too large for the memory of our old DEC PDP-11/23 computer and was broken into two pieces, *C6a* and *C6b*. Together the two comprise the main output routine for *C6*, although some calculations are performed for special samples. Most of the output is taken care of in *C6a*, and most of the calculations fall in *C6b*. *C6a* itself is little more than a calling routine, accessing the various subroutines as necessary. These subroutines are summarized in Table 8.16.

#### 8.10.1 Subroutine *Collect\_At\_Pcts*

This routine assigns to the appropriate variable in the *Out* structure (*out.at\_pct*) the averages of the atom fractions calculated in *C5* and converted to atom percents.

#### 8.10.2 Subroutine *Set\_Iso\_Diln\_Ratio*

This routine assigns to the correct *Out* structure variables the parameters associated with isotope dilution calculations.

### 8.10.3 Subroutine GpG\_Quick\_List

GpG stands for grams per gram; concentrations for isotope dilution analyses are calculated in this subroutine. The concentration in grams per liter is also calculated; it will be numerically identical to the concentration in grams per gram unless a value for the solution density has been entered.

---

Table 8.16 Subroutines in C6a

Collect_At_Pcts	Collects the ratios used in calculating atom percents into a special array.
Set_Iso_Diln_Ratio	Assigns various variables to appropriate names.
GpG_Quick_List	Calculates concentration; lists values of ratios on first output sheet.
Print_Header	Lists header information to the printer.
Print_Results	Lists values for individual ratios and their averages and statistics.
Calc_Pu_238	Calculates the number of counts of 238-Pu on the basis of total counts in each subgroup.
Print_At_Pcts	Prints atom percents and their statistics.
O_Corr	Corrects isotopic composition if the analysis was performed on oxide peaks.
Print_Corr_Pu	Calculates and prints isotopic composition of Pu corrected back to a specified time.
Print_Wgt_Pcts	Calculates and prints weight percents.
Find_Exact_Masses	Determines exact masses to be used for the analysis in question.
Print_Std	Prints isotopic composition for a certified reference standard.

---

The other function of this subroutine is to print a one-line, unlabelled list of the values for the ratio averages. This is to accommodate impatient staff members who resent having to wait for the end of the second output sheet for this information.

### 8.10.4 Subroutine Print\_Header

Header information for the second output sheet is printed in this subroutine. By this is meant most of the information associated with the sample that doesn't require calculation. On the first line are the identification codes for the sample and the two companion samples (if any); title; the element symbol and date; the *Samp* file input and *Out* file record numbers; the batch code; and the mass spectrometer identification number. The second line contains the dead time and bias correction per mass; the number of cycles and the sweep rate; the normalizing mass for atom percent calculations; and comments, which often include the filament temperature. The isotope dilution ratio appears on this line if appropriate. The next lines list sums and differences, one sum and one difference per line. The next lines contain in turn the subgroup numbers and masses, sweep factors, bias corrections, and corrected counts associated with them. Note that, except for the background position, the corrected counts are the sums for each subgroup with all corrections applied; the counts listed for the background position are those actually collected at this position. This number is

subtracted from all other mass positions, and the corrected counts listed for them have had this correction made. Applying it to the background position itself, however, would infallibly give 0.000 and hence no information. It is for this reason that the described choice was made.

#### 8.10.5 Subroutine Print\_Results

This routine prints the values of all ratios for each run, their averages, and the statistics associated with them. This is normally a one-page output, but if the number of ratios called for exceeds 11, there will be a second page containing the twelfth ratio and above. Individual ratios identified as statistical outliers are flagged with an asterisk (\*) after the value. These values are not included in the calculation of the average for that ratio. The run number, number of channels included in the sum (headed SCh), and the total counts for each run appear in the left-most three columns. Each column has a label over it to identify what information it contains. The isotope dilution ratio is arbitrarily defined as 999/999. The numerators and denominators of ratios are integer arrays, and the isotope dilution ratio (Section 10.2 and Section 11.1 of the Operator's Manual) is not a simple isotopic ratio. Some means of identification was needed, and 999/999 was chosen on the assumption it would always be nonsensical if taken literally. We can only hope.

#### 8.10.6 Subroutine Calc\_Pu\_238

The 238 mass position is difficult to measure accurately when U-Pu pairs are being analyzed because each element has a 238 isotope. Although the algebraic correction of this peak is exact in the mathematical sense, in practice it often leaves something to be desired because of the disparity in abundance of this isotope in the two elements (small in Pu, large in U). It is particularly troublesome in plutonium analyses when for some reason that element doesn't run as well as could be wished. This often causes the filament temperature to be higher than desired, and consequently the contribution of uranium to the 238 peak increases. The contribution of 238-U is subtracted on the basis of the number of counts of 235-U collected; 235-U is usually a small fraction of 238-U (usually a few percent or less), and thus we have a large correction being made on the basis of the measurement of a small peak.

This subroutine calculates the amount of 238-Pu present on the basis of total counts collected in the analysis for the relevant mass positions. It is, unfortunately, of doubtful value because these counts often go negative due to the problems outlined in the preceding paragraph. Negative counts have no physical meaning, of course.

#### 8.10.7 Subroutine Print\_At\_Pcts

This routine prints the isotopic composition of the sample in atom percent. The values are carried through the ratio calculations as atom fractions; the isotopes to be included are identified as those appearing in *Sum(I)*.

Isotopic masses, atom percents, standard deviations, standard errors of the mean, and 95% confidence intervals are printed on successive lines.

#### 8.10.8 Subroutine O\_Corr

Back in the dark ages, John Sites asked that provision be made in the programs to allow analyses to be made using oxide peaks. When oxide peaks are monitored, it is necessary to correct for the isotopic composition of oxygen to them; oxygen has isotopes at 17 and 18 as well as the major one at 16.

This routine addresses that situation. The only situations catered to are those when either the monoxide or dioxide peaks are measured ( $\text{MO}^+$  and  $\text{MO}_2^+$ ). Corrections are applied only to average atom percents; ratios are not corrected.

#### 8.10.9 Subroutine Print\_Corr\_Pu

Plutonium is a radioactive element, and its isotopes decay at various rates. Some of the half-lives are quite short and decay fast enough to be significant on a human time scale. The shortest-lived isotope usually measured is 241-Pu with a half life of about 14.4 years. This is short enough that its decay has significant effect on its abundance in a time span of a few months. An important instance of this is in the case of NBS (NIST) Pu-947, an isotopic standard certified as of October 13, 1971, to have 4.540% 241-Pu. As of February 21, 1992, this value had declined to 1.750%.

Provision is made in the program for entry of a date to which a plutonium composition is to be corrected. This becomes quite important in interlaboratory comparison experiments, where analyses in the different laboratories may be made months apart; each laboratory corrects its values to an agreed-upon date to allow valid comparison.

The corrections are made for all isotopes, even though the changes are trivial for most of them. It is accomplished by calling subroutine *Tzero* (Section 11.31).

#### 8.10.10 Subroutine Print\_Wgt\_Pcts

This little routine calculates and prints the composition of sample in weight percent using the known atom percent composition. Subroutine *Wgpct* (Section 11.36) calculates weight percents. Masses and weight percents are printed on the output sheet. The precise masses from the *Massab* file (Section 4.6) are used in the calculations.

#### 8.10.11 Subroutine Find\_Exact\_Masses

This routine makes proper assignment of isotopic masses. It is necessary to adjust this array if 233-U, included in the isotope list in *Massab* for convenience, is not monitored during the analysis.

"Exact mass" is perhaps a loose term, although it is in common use. A better term might be "precise mass." What is meant here is the atomic masses of the isotopes are carried out to a number of significant figures past the decimal point rather than using the nominal (integer) mass. For example, the nominal mass for 238-U is 238; its "exact" mass is 238.0508.

## 8.10.12 Subroutine Print\_Std

As a convenience when a certified isotopic standard is being analyzed, this routine prints its isotopic composition to allow easy comparison to the measured values. The program gets this information from the *Stand* file; it is thus essential to have entered the composition of the standard in question. This routine is called only when the batch code is 'ST'.

## 8.11 Subroutine C6b

This routine, the second part of the original C6, prints information not taken care of in C6a. This is most frequently the results of isotope dilution calculations. The routine also performs some calculations; chief among these are the application of a blank correction and isotope dilution calculations when no unspiked companion is available. Disk file *Out* is also updated by writing the results of the current analysis into it.

C6b, like its counterparts, is primarily a calling routine and performs very few calculations on its own. The subroutines it calls are listed in Table 8.17.

---

Table 8.17 Subroutines in C6b

Print_Unspiked_Info	Prints the isotopic composition of the unspiked sample.
Calc_Int_Cal_Orig_Iso	Corrects isotopic composition on the basis of the bias calculated in internal calibration analyses.
Print_Conc	Prints concentration for isotope dilution analysis.
List_Mist	Special output for Mist samples.
Print_Spike	Prints masses and isotopic composition of the spike.
Get_Nis	Determines the number of isotopes.
Apply_Blank_Corr	Applies blank correction when called for.
NO_Comp_ID_Calc	Performs isotope dilution calculations when no unspiked companion information is available.

---

The last two routines, *Apply\_Blank\_Corr* and *NO\_Comp\_ID\_Calc*, are located in libraries (*C6b\_Blank* and *C6b\_NO\_Comp*, respectively). These are described in Sections 8.11.7 and 8.11.8.

## 8.11.1 Subroutine Print\_Unspiked\_Info

This routine prints the isotopic composition of the unspiked sample in an isotope dilution analysis. This information must be in the *Out* file and is read in subroutine C2 (Section 8.5).

## 8.11.2 Subroutine Calc\_Int\_Cal\_Orig\_Iso

This routine estimates a new isotopic composition for the unspiked sample when internal calibration calculations are being performed. A new average bias correction per mass

is estimated from the value of the largest ratio not used for internal calibration purposes; it is then applied to all isotopes.

This procedure has limited applicability. Of the two uranium internal calibration spikes, only the one made with the purer individual components yields reliable results when applied to refining an isotope ratio measurement. Both spikes work well in isotope dilution calculations. For uranium, the largest ratio not involved in internal calibration is usually  $^{235}\text{U}/^{238}\text{U}$ . When a more accurate value for this ratio is desired, the high purity spike should be used; the corrections made thereby are quite reliable. With the low purity spike, which has nearly 10%  $^{235}\text{U}$ , the contribution from the spike to the 235 mass position is almost always considerably greater than the sample. The large correction required renders the resulting  $^{235}\text{U}/^{238}\text{U}$  ratio so uncertain that it should be carefully evaluated before being accepted.

#### 8.11.3 Subroutine Print\_Conc

This routine prints the concentration calculated during isotope dilution calculations. It also prints other information relevant to these calculations: spike, sample, dilution, and aliquot weights; density; dilution factor; concentration in grams per gram and grams per liter; the value of the isotope dilution ratio and its 95% confidence interval. These appear with appropriate labels that are rather terse to conserve space.

#### 8.11.4 Subroutine List\_Mist

This routine is unlikely to be used again. It was written to address the specific case of MIST samples, which originated at the K-25 and Paducah Gaseous Diffusion Plants. The goal of these analyses was to provide the best possible measurement of the minor uranium isotopes (234 and 236) as a diagnostic aid for the enrichment process. The  $^{235}\text{U}/^{238}\text{U}$  ratio was measured at K-25 using their gas mass spectrometers, whose precision and accuracy for this measurement are much better than ours. The value for this ratio was input through *MSR*, and the analysis monitored only 234, 235, and 236; 238 was far too intense to monitor as the 235 intensity was several hundred thousand counts per second. Abundances for 236 as low as about 20 ppb were measured to good precision.

This routine handles the special output needed for these samples. Such samples are identified by using M& as the batch code.

#### 8.11.5 Subroutine Print\_Spike

This routine prints the isotopic masses and composition of a spike. It is accessed at several different places in C6b.

#### 8.11.6 Subroutine Get\_Nis

This routine determines the number of isotopes in a sample by counting the number of non-zero entries in *Sum(I)*.

8.11.7 Library *C6b\_Blank*

This is a module holding the subroutines involved in applying a blank correction. It is not a library in the formal Fortran sense. The subroutines are listed in Table 11.18.

---

Table 8.18 Contents of *C6b\_Blank*

<i>Apply_Blank_Corr</i>	Called by <i>C6b</i> ; performs many of the basic calculations.
<i>Blank_Iso_Diln_Calc</i>	Performs isotope dilution calculations for the blank and sample.
<i>Print_Blank_Info</i>	Prints results of a blank correction.

---

A blank correction is required for isotope dilution calculations when the contribution from reagents or other undesirable sources is significant with respect to the concentration being measured for the sample. By being very careful in chemical preparation, it is usually possible to make such a correction unnecessary, but there are instances when it is unavoidable. Vegetation samples, for example, often have a uranium content on the order of 20 ppb; blanks often run about 1% or so of this value. For these and other similar cases, it is desirable to subtract the blank contribution from the concentration calculated for a normal isotope dilution analysis. The set of procedures in this library accomplishes this.

For the correction to be effected, the results of a blank analysis must be in the *Out* file, just as information for an unspiked sample must be there for a normal isotope dilution analysis. This blank is identified by its unique six-character code, which is entered by the analyst at the time of analysis (in *MSR*). Essential information includes its isotopic composition and its concentration. The ratio file must also contain the isotopes of the sample in *Sum(2)*; *Sum(1)* is for the sample plus spike. These calculations are set up on the assumption that the spike isotope is not present in either the unspiked sample or the blank; this is a good approximation for uranium. If a situation should arise where this assumption was not valid, modifications to the software would be required.

8.11.7.1 Subroutine *Apply\_Blank\_Corr*

This routine gets the necessary arrays set up for blank calculations. It calls *Blank\_Iso\_Diln* (see next section) and *Print\_Blank\_Info* (Section 8.11.7.3).

8.11.7.2 Subroutine *Blank\_Iso\_Diln\_Calc*

This routine calculates the elemental concentration of the sample corrected for the contribution of the blank. The Hintenberger isotope dilution equation (Section 10.2 of this manual and Section 11.1 of the Operator's Manual) was revised to accommodate contributions from more than one source. The revised equation is perfectly general and is applicable to any element. The equation can be written:

$$C = W_t/W_s (A_s/A_p) (a_t/a_s) (R_m - R_t)/(R_s - R_m) + \\ W_b (A_s/A_b) (a_b/a_s) (R_m - R_b)/(R_s - R_m)$$

where: W = weight;  
 A = atomic weight;  
 a = abundance;  
 R = ratio;  
 C = concentration.

The subscripts are defined:

s = sample;  
 t = tracer (spike);  
 m = mixture;  
 b = blank.

R is defined as the ratio of the abundance of the sample isotope to the spike isotope; this would usually be  $^{238}\text{U}/^{233}\text{U}$  for uranium, although the equation works fine if the inverse of this ratio is used.

#### 8.11.7.3 Subroutine Print\_Blank\_Info

This routine prints information relevant to the blank correction.

#### 8.11.8 Library C6b\_NO\_Comp

This library contains the routines used when no unspiked companion is available for isotope dilution calculations. Such calculations are valid only if the spike isotope is not present in the sample; if this is not the case, normal isotope dilution calculations should be employed, which require that the isotopic composition of the unspiked sample be in the *Out* file. The two most common cases for which this shortcut is valid are uranium and thorium. The usual uranium spike,  $^{233}\text{U}$ , is very seldom present in samples (it does not occur in nature); similarly, for thorium, the  $^{230}\text{Th}$  spike very rarely is present in samples.

The subroutines contained in *C6b\_NO\_Comp* are listed in Table 8.19.

---

Table 8.19 Contents of C6b\_NO\_Comp

NO_Comp_ID_Calc	Calling routine, itself called by C6b.
Set_NO_Params	Initializes various parameters.
Calc_Corr_At_Pct	Calculates corrected atom percent.
Print_NO_ID	Calculates concentration and prints results.

---

##### 8.11.8.1 Subroutine NO\_Comp\_ID\_Calc

This is the routine called by *C6b*. It is in turn a calling routine that accesses the other routines as needed. Note that the ID in its name refers to isotope dilution.

### 8.11.8.2 Subroutine Set\_NO\_Params

Various parameters are initialized in this routine. This mostly involves identifying the locations of the variables needed in the isotope dilution equation (Section 10.2 of this manual and Section 11.1 of the Operator's Manual).

### 8.11.8.3 Subroutine Calc\_Corr\_At\_Pcts

This routine calculates the isotopic composition of the sample with the contribution from the spike subtracted. It must be emphasized that this procedure is valid only if the sample does not contain the spike isotope.

### 8.11.8.4 Subroutine Print\_NO\_ID

This routine uses the information generated in the two subroutines just described to calculate the concentration of the sample; that is, Hintenberger's isotope dilution equation is applied (Section 10.2). The results of this calculation and the corrected isotopic composition of the sample are printed, along with other relevant information.

## 8.12 Subroutine C7a

Subroutines *C7a* and *C7b* were created when *C7* became too large for the memory of our old PDP-11/23. *C7* itself was originally incorporated in *C6*, but that subroutine became too large early in the history of *MSC*.

*C7a* is quite short; its most important function is to update the *Samp* file by inserting a value of 1 as the flag to indicate that the data for the current sample have been processed. The subroutines used in *C7a* are listed in Table 8.20.

---

Table 8.20 Subroutines in C7a

Print_Cal_Std	Prints result of a calibration standard (NBS U-500) analysis.
Fix_Rejected_Runs	Adjusts rejected run arrays to allow reprocessing the data.
Update_Samp	Updates the entry in the Samp file.

---

### 8.12.1 Subroutine Print\_Cal\_Std

This routine prints the results when an instrumental calibration standard has been analyzed. This has always been an NIST U-500 so far, but it is certainly conceivable that other standards will emerge to meet changing requirements.

### 8.12.2 Subroutine Fix\_Rejected\_Runs

This routine adjusts the number of runs to allow smooth reprocessing when some have been rejected. The number of runs is changed back to its original value in this routine and

applies only to runs within the body of the analysis; runs rejected by changing the first or last run numbers from their original values will not be affected. If you want to include runs at the start or end of the analysis, you should alter the first or last run accordingly using program *File*.

This routine also sets up arrays to allow runs rejected on the first pass through the data of plutonium-uranium mixed pairs to be rejected on the second as well. One effect of this is to necessitate reinitializing both members of the pair if both need to be recalculated.

### 8.12.3 Subroutine Update\_Samp

This routine updates the entry in the *Samp* file of the sample whose data are being processed. It sets the variable *samp.calc\_index* to 1; a value of 1 indicates that the data for the sample have been processed; a 0 means they haven't. The value of the pass number used in mixed resin bead analyses is also set before the disk file is updated.

### 8.13 Subroutine C7b

This routine, originally part of *C7*, attends to special calculations required only by small numbers of samples. Many of the routines were written to meet a special need that was temporary and no longer exists; their descriptions are given here for the sake of completeness and in case something like them is ever needed again. The only one of these special cases in current use is the one that does reactor burn-up calculations; it is potentially a very important capability. The special cases are summarized in Table 8.21.

---

Table 8.21 Special Cases in C7b

<u>Batch Code</u>	<u>Routine</u>	<u>Description</u>
A1	A126	Saves data in special files to prevent it being overwritten.
RC	Fractionation	Makes calculations to quantify isotopic fractionation.
BU	Burnup	Performs reactor burn-up calculations using Nd.
[]	Bayne	Special output for C. K. Bayne's statistical experiment.
R.	RIMS	Performs calculation and prints output for RIMS experiments.

---

The only function performed by *C7b* itself is to call the appropriate routine, which is identified by a specific batch code.

#### 8.13.1 Subroutine A126

This routine was written to address the challenges presented by ISPO Task A126 (hence the name). That experiment was a multi-laboratory effort to assess the viability of resin bead sample methodology. It required output in a specific format, which this routine

takes care of. In addition, the routine stores the results in a separate output file (*A126.DAT*) to prevent them from being written over (as are those in *Out.DAT*).

This routine is accessed by setting the batch code to A1. It is unlikely to be needed in its present form; it may some day be necessary to store data for some project in a special file, and parts of the code may be useful then.

### 8.13.2 Subroutine Fractionation

This routine was to help fractionation studies being performed in conjunction with development of using an overcoat of rhenium powder to enhance ionization efficiency.<sup>9</sup> The  $U^+$  produced with this technique was so stable that it made such studies meaningful. Normally fractionation is obscured by other effects.

This routine accessed by setting the batch code to RC. It is unlikely to be needed again.

### 8.13.3 Library C7b\_Burnup

This is the only one of these special routines still in use. Its function is to calculate reactor burn-up using neodymium isotopic information and follows ASTM Procedure E321.<sup>10</sup> The complete burn-up calculation requires six analyses: isotopic composition and concentration for each of Nd, Pu, and U. It is accessed by using a batch code of BU.

Both the analysis and the calculations are complex. The calculations use neodymium because it is the fission product generated in reactor operation chosen by the engineers for the purpose of calculating burn-up. Burn-up is calculated two different ways, one using 148-Nd, the other 145-Nd + 146-Nd. The quantity of neodymium is one of the factors needed by the equations, so an isotope dilution analysis must be performed, which requires analysis of the unspiked sample; no assumptions can be made about the isotopic composition of neodymium since it will vary with burn-up. The analysis is complicated by the fact that cerium interferes at masses 142 and 144, with the former being of most concern. The correction to 144 is at present disabled because it caused more trouble than it was worth.

Ray Walker devised a method to separate samarium from neodymium by extracting from an alcohol-acid medium, but he was never able to separate cerium. This fact adds considerable complexity to the programs. Neodymium ionizes at a lower temperature than cerium, so it is analyzed first. Cerium is then analyzed, and the neodymium results are corrected for its contributions before burn-up is calculated. Nd-Ce pairs are identified in the same way the Pu-U resin bead pairs are; there is thus what is called a 'resin bead companion' whose six-character identification code is entered by the analyst before the analysis begins.

There are several individual subroutines to perform the various tasks involved; they are summarized in Table 8.22.

---

 Table 8.22 Subroutines in C7b Burnup

Burnup	Calling routine.
Correct_Nd_for_Ce	As indicated in title.
Burnup_Calcs	Performs burn-up calculations and prints results.

---

### 8.13.3.1 Subroutine Burnup

This is the subroutine called by *C7b*. It serves as a calling routine, guiding the program through the proper steps.

### 8.13.3.2 Subroutine Correct\_Nd\_for\_Ce

This routine does what it says in its name. It identifies which mass positions in each component (Nd spiked sample, Nd unspiked sample, and Ce sample) were used for the 142 and 144 isotopes. It then corrects the Nd spectra on the basis of the number of counts of 140-Ce collected.

### 8.13.3.3 Subroutine Burnup\_Calcs

Burn-up calculations are performed in this subroutine. The results of the calculations are printed.

A two-dimensional array is set up to facilitate the calculations. It holds the locations with regard to mass position of the isotopes used in the calculation: 142, 148, 150 (150 is the spike). This array is named *Nd\_iso\_inx*; it is defined in Table 8.23.

---

 Table 8.23 *Nd\_iso\_inx* Array

<i>Nd_iso_inx</i> (1,x)	Positions for unspiked sample.
<i>Nd_iso_inx</i> (2,x)	Positions for natural element.
<i>Nd_iso_inx</i> (3,x)	Positions for spike.
<i>Nd_iso_inx</i> (4,x)	Positions for Ce companion sample.
<i>Nd_iso_inx</i> (x,1)	142
<i>Nd_iso_inx</i> (x,2)	148
<i>Nd_iso_inx</i> (x,3)	150

---

Once this array is set up, the values for the various ratios needed by the equations are calculated. The variable names chosen are supposed to identify what they are; for example, *rat\_150\_148\_unspk* is the ratio of 150-Nd/148-Nd in the unspiked sample. For other variables, nomenclature follows that used in the ASTM procedure.

Results are printed at the end of the subroutine.

#### 8.13.4 Subroutine Bayne

The name of this program bestows undying fame to that eminent statistician, Charles K. (Chuck) Bayne. He came up with an original way of evaluating pulse-counting data and needed special output to facilitate his analysis of the results. This work resulted in a publication,<sup>11</sup> but neither of us has had time to get back to the follow-up study we planned.

This routine is accessed through batch code []. It will almost certainly never be used again unless Chuck and I find time to pursue the project.

#### 8.13.5 Subroutine RIMS

This routine was written to accommodate some of the early resonant ionization mass spectrometry (RIMS) experiments. There is little chance it will ever be used again. R. is the batch code used to access this routine.

### 8.14 Internal Calibration Calculations

Internal calibration is complex enough that I thought it worthwhile to give it separate treatment. The program applies internal calibration in Subroutine *C3b* (Section 8.7.8); initialization routines are in *C3a* (Sections 8.6.4 and 8.6.7). It is also described in Section 11.2 of the Operator's Manual.

The idea behind internal calibration is to calculate a bias applicable to the sample in question as it is being analyzed. Normally an average bias, determined by independent analysis of NIST certified isotopic standards, is applied. Like any average, its applicability to any specific analysis is problematic. Improvement of a factor of five or more can be realized using internal calibration.

#### 8.14.1 Description of the Procedure

In use, a spike is prepared that has high abundances of two isotopes. For uranium, these isotopes have been 233 and 236, and, for plutonium, 242 and 244. Other isotopes could have been used. The ratio between these two isotopes is measured as accurately as possible. For uranium, we used NIST-500 as an internal calibrant; for Pu, we used NIST-947. The double spike is added to the sample and the ratio of the two critical isotopes measured. The contribution of the sample to the critical mass positions must, of course, be subtracted before the ratio due to the internal calibration spike is calculated. The measured value of the ratio is compared to the known, and the bias for that run calculated. This bias is then applied to all masses for the run in question. A similar calculation is done for each run, generating an individual bias correction for each. Improvement in precision and accuracy is realized because bias is calculated for the specific filament and conditions under which the analysis was made.

The calibration ratio isotopes can be in either the spike or the sample. A variable, *instd*, is used to indicate which. If *instd* = 0, the isotopes are in the spike; if *instd* = 1, they are in the sample.

### 8.14.2 Calculation of Bias

Bias is calculated in three ways. Details are given in reference 6. These three ways are successive approximation, solution of a linear equation, and solution of a quadratic equation. These will be described in turn.

#### 8.14.2.1 Successive Approximation

The isotope dilution ratio ( $238/233$  or  $238/236$  for U) is used to test convergence. An initial value is calculated using the value for the bias stored in the *DTBias* file. The test ratio is calculated, and the ratio of the expected value to the measured is calculated; 1.000 is subtracted from it and the difference compared to the value of *convergence\_test*, a variable that defines how close to the expected value the measured must be corrected. *Convergence\_test* is initialized to  $1 \times 10^{-6}$  and is doubled every ten iterations to encourage convergence. If convergence is not attained, the process is repeated, using the value calculated in the previous iteration for the bias. Each iteration calculates the amount of bias necessary to add (algebraically) to that already used to bring the measured value of the ratio into agreement with the expected. Most samples require 5 or 6 iterations for convergence.

Calculating the bias this way is as reliable as using either of the two analytical solutions described below.

#### 8.14.2.2 A Word on Nomenclature

Internal calibration can be confusing, and I thought a short description of the nomenclature used in the programs would be helpful. It is necessary to keep track of three isotopes and three contributors. Isotopes have the subscripts a, b, and c; contributors have subscripts s, t, and m (for sample, tracer, and mixture). Lower case r is used for ratio. These are summarized in Table 8.24.

Table 8.24 Nomenclature Summary

<u>Subscript</u>	<u>Definition</u>	<u>U Example</u>
a	Isotope a in spike	233
b	Isotope b in spike	236
c	Isotope C in sample	238
s	Sample	
t	Tracer (spike)	
m	Mixture	
r	Ratio	
m	Mass	

Either isotope a or b will be used with c in the isotope dilution ratio; it makes no difference which is used; concentrations will be exactly equal in either case. Thus,  $r_{ab}$  is the ratio of

the two major isotopes in the internal calibrant; this would be 233/236 for uranium. Rbcm or racm would be the isotope dilution ratio of the mixture. For uranium, ma = 233, mb = 236, and mc = 238.

#### 8.14.2.3 Linear Solution

The linear equation solved for the bias correction was derived by Stein Deron of the International Atomic Energy Agency. Using the nomenclature described in Section 8.14.2.2, it can be written:

$$\text{bcm} = \frac{((\text{rbcm} - \text{rbc}) * (\text{ract} - \text{racm}))}{((\text{racm} - \text{racs}) * (\text{rbct} - \text{rbc}))} - \frac{(\text{racm} * (\text{ma} - \text{mc}) * (\text{rbct} - \text{rbc}) - (\text{rbcm} * (\text{mb} - \text{mc}) * (\text{ract} - \text{racs})))}{(\text{rbcm} * (\text{mb} - \text{mc}) * (\text{ract} - \text{racs}))}$$

where bcm is the bias correction per mass. It appears on lines 288-291 of the June 27, 1991, listing of *Apply\_Int\_Cal* (Section 8.7.8).

#### 8.14.2.4 Quadratic Equation

Remember the formula for solving the quadratic equation? Here's a refresher, free of charge. For the equation

$$ax^2 + bx + c = 0,$$

$$x = \frac{-b \pm (b^2 - 4ac)^{1/2}}{2a}$$

For the equation here,

$$\begin{aligned} a &= (\text{mb} - \text{ma}) * (\text{mc} - \text{mb}) * \text{rbcm} * \text{rabm} * (\text{rbct} - \text{rbc}) \\ b &= (\text{mb} - \text{ma}) * \text{rabm} * \text{rbcm} * (\text{rbc} - \text{rbct}) + \\ &\quad (\text{mc} - \text{mb}) * \text{rbcm} * (\text{rbc} * (\text{rabm} - \text{rabs}) + \\ &\quad \quad \text{rbct} * (\text{rabt} - \text{rabm})) \\ c &= \text{rbcm} * \text{rbct} * (\text{rabm} - \text{rabt}) + \text{rbcm} * \text{rbc} * \\ &\quad (\text{rabs} - \text{rabm}) + \text{rbc} * \text{rbct} * (\text{rabt} - \text{rabs}) \end{aligned}$$

These are lines 268-273 of June 27, 1991, listing of *Apply\_Int\_Cal* (Section 8.7.8).

## 9. PROGRAM REP

*Rep* is short for REPort. It is designed to produce a report sheet for samples that are somehow associated. Results from the *Out* file are listed in one of several formats. One of the uses of the batch code is to help identify samples that form a group. Batch code ST, for example, is used for QA standards. Entering *Rep* and telling it to list all samples with batch code ST will produce an output sheet containing results of all standards.

Like *MSC* and *File*, *Rep* is most conveniently linked by execution of a batch file. It is named *Rep.COM* and is invoked by entering @Rep from the keyboard in directory [DHS.FILE]. A listing of the batch file is found in Table 9.1.

---

Table 9.1 Batch File Rep.COM

```
!           To link new Report program.
!           6-26-90
! Link/debug rep-
Link rep-
+report1-
+report2-
+Out_Edit-
+[dhs.lib]Outloo-
+[dhs.lib]Disk_IO/lib-
+dhs.lib]Screen_Lib/lib-
+[dhs.lib]dhs_lib/lib
```

---

Another use of *Rep* is to obtain averages of replicate analyses. The contents of file records as input by the analyst can be averaged; averages will include all ratios and atom percents. *Rep* consists of a main line calling program and three principal subroutines, which are listed in Table 9.2.

---

Table 9.2 Subroutines in Rep

Out_Edit	Edit contents of Out file
Report1	Initialization routine
Report2	Listing routine

---

These subroutines are described briefly below.

### 9.1 Out\_Edit

This routine is called by program *File* as well as by *Rep*. It is described in Section 4.11.

## 9.2 Report1

This routine accepts input and gets the program cocked to make the lists in *Report2* (Section 9.3) A number of subroutines, listed in Table 9.3, are used to facilitate these chores.

---

Table 9.3 Subroutines in Report1

Get_Label	Init_Format_Msg
Get_Rec_Nrs	Put_Info
Find_Rec_Nrs	Get_Date
Get_List_Format	

---

These routines are described below.

### 9.2.1 Subroutine Get\_Label

This routine reads in a descriptive label for the list; a special window is opened to facilitate this task.

### 9.2.2 Subroutine Get\_Rec\_Nrs

This subroutine determines which record numbers in *Out* are to be read to produce the desired list. Input may be either record numbers directly or six-character sample identification codes, as the analyst prefers. If sample code entry is chosen, the locations of the samples in the *Out* file are found by calling subroutine *Outloo* (Section 11.25).

### 9.2.3 Subroutine Find\_Rec\_Nrs

Many of the listing options provide the ability to confine the search between two record numbers. For example, the analyst may only be interested in producing a list of standards for a given period. The extremes (in record numbers) are entered in cases such as this.

Other limits are set; permanent companions are restricted to the first 50 records, so those limits are imposed when appropriate. The batch code is set to ST when only standard results are desired. A batch code or element symbol is read if the corresponding search option was chosen by the analyst.

For all these options *Find\_Rec\_Nrs* then identifies which record numbers within the specified range hold data of interest. Subroutine *Set\_Rec* is used to generate an array (*rec\_nr*) to hold these record numbers for future use.

### 9.2.4 Subroutine Set\_Rec

This routine adds the latest *Out* file record number whose contents are to be listed to the array *rec\_nr*. Variable *nsam* (for number of samples) is a counter used to keep track of the number of non-zero elements in the array.

### 9.2.5 Function Get\_List\_Format

This function establishes the value for a variable called *list\_format*, which can take on values from 1 through 7. This variable controls the output format for the list being generated by the program. It is defined as shown in Table 9.4.

---

Table 9.4 Definition of List Format

<u>Value</u>	<u>Description</u>
1	List all ratios and percents
2	List ratios only
3	List atom percents only
4	List weight percents only
5	List atom and weight percents only
6	List atom percents and ratios with only one line per sample
7	List ratios with one line per sample

---

Of these, a value of 6 is most often used. Nearly all the required information is presented in an easily readable format.

### 9.2.6 Subroutine Put\_Info

This little routine is used mostly to provide the analyst (and programmer) with assurance that the program is locating the correct samples. The index for the sample (runs from 1 to *nsam*) and the associated sample identification codes are listed to the screen as each sample is found.

### 9.2.7 Subroutine Get\_Date

This is a routine to enter the date to which the isotopic composition of a plutonium sample is to be corrected.

## 9.3 Subroutine Report2

This routine generates the output list that is the goal of *Rep*. It also calculates any averages desired. A number of subroutines were written to facilitate these tasks and are listed in Table 9.5.

---

Table 9.5 Subroutines in Report2

Write_Header	Set_Avg_Array
Set_At_Pct_Ratios	Correct_Pu
List_Out	List_Names
List_Ratios	List-Mist
List_Percents	List_Std
List_One_Line	Calc_Avgs

---

*Report2* is basically a calling program that invokes subroutines as needed. There is essentially no logic in it beyond some IF tests. The subroutines are described below.

### 9.3.1 Subroutine Write\_Header

This short routine writes labelling information (the header) on each page of output. It also increments the page number as needed and sets the number of entries on the page to zero.

### 9.3.2 Subroutine Set\_At\_Pct\_Ratios

This routine identifies which ratios should be used to calculate atom percents; it also returns values for those ratios. Such ratios must meet two criteria: (1) the mass of the numerator must be included in *out.sum\_mass*, which was set equal to *ratio.sum\_mass(1,..)* in program *MSC*; and (2) the denominator must also be in *out.sum\_mass* and must be the same for all ratios.

When corrected atom percents have been calculated (corrected for blank or contaminant contributions, for example), appropriate ratios are calculated; these are not normally calculated in *MSC* and are often what the analyst wants to see. These ratios are assembled into an array called *at\_pct\_ratio*. When no corrected atom percents have been calculated (the normal case), *at\_pct\_ratio* is assigned values from *out.ratio\_avg*.

### 9.3.3 Subroutine List\_Out

This routine is called from *Report2* and directs the appropriate output to the printer by calling various subroutines described in the following sections. It also calculates weight percents.

### 9.3.4 Subroutine List\_Ratios

This routine lists ratios in the *Out* file to the printer.

### 9.3.5 Subroutine List\_At\_Pcts

This routine lists atom percents in the *Out* file to the printer. If the sample represents an isotope dilution analysis, it lists information relevant to concentration calculations.

### 9.3.6 Subroutine List\_One\_Line

This routine presents information in the *Out* file in a condensed format. Despite the title, two lines of output are required to hold values for ratios, atom percents, and their standard deviations. A third line is necessary if weight percents are listed.

### 9.3.7 Subroutine Set\_Avg\_Array

This routine assigns values to elements of an array of structures, *avg*. This structure is used as a convenience to facilitate calculation of averages. It holds values for ratios, atom percents, and concentration, among others.

### 9.3.8 Subroutine Correct\_Pu

This routine corrects the isotopic compositions of plutonium samples to any specified date. It is similar to other routines (See Section 4.7.6, for example). Subroutine *Tzero* (Section 11.31) is called to accomplish this task.

### 9.3.9 Subroutine List\_Names

This routine causes only identifying information to be listed on the printer. This includes the record number in the *Out* file, element symbol, sample identification code, and so on.

### 9.3.10 Subroutine List\_Mist

This is a special listing routine for Mist samples. These were isotope enrichment samples from K-25 and Paducah. Mist was a large program--there were well over 200 samples in one of the batches. It is unlikely that it will be needed again, but it seems a shame to throw away a working routine. It remains available if needed.

### 9.3.11 Subroutine List\_Std

This routine accesses the *Stand* file and lists the certified composition of the relevant standard. It then lists the results of standards from the *Out* file.

### 9.3.12 Subroutine Calc\_Avgs

This routine calculates any averages called for and their associated statistics and lists them on the printer.

## 8.6.5 Subroutine Set\_IC\_Masses

The *IC* in the name of this routine stands for Internal Calibration. The routine is called from *Set\_Int\_Cal\_Calcs*. As described in the previous section, at least two ratios must be identified for internal calibration calculations to be valid. This in turn requires identification of at least three masses: the two spike isotopes used in the internal calibration ratio and the isotope representative of the sample. There may, of course, be more than one isotope in the sample affected by the calculations. An attempt was made to develop a consistent, readily understood nomenclature for assigning the large number of variable names required for internal calibration calculations. The underscore is used to set off what in algebraic notation would be subscripts. Table 8.8 summarizes the conventions used and gives examples for uranium.

---

Table 8.8 Nomenclature of Variables Used in  
Internal Calibration Calculations

---

Suffix	Definition	Example
<i>_a</i>	One mass of internal calibration ratio.	233-U
<i>_b</i>	Other mass of internal calibration ratio.	236-U
<i>_c</i>	Sample isotope.	238-U
<i>_d</i>	Second sample isotope.	235-U
<i>_mix</i>	Refers to the mixture.	
<i>_spk</i>	Refers to the spike.	
<i>_unsp</i>	Refers to the unspiked sample.	

---

Thus, *mass\_a*, *mass\_b*, and *mass\_c* are identified from the masses in the internal calibration and isotope dilution ratios. These are then associated with their subgroups in the analysis at hand (called *sg\_mix\_a*, etc.); their relative positions in the spike and unspiked sample isotopic compositions (*pos\_spk\_a*, etc. and *pos\_unsp\_a*, etc., respectively) are also identified.

Each mass in the internal calibration ratio is assigned to a subgroup higher than the highest actually used during the analysis. For example, if eight subgroups were used, the numerator of the internal calibration ratio would be assigned to subgroup 9 and the denominator to subgroup 10. A new ratio is defined that holds the run-by-run values for the corrected internal calibration ratio. This complexity is not truly necessary once one is satisfied that the algorithm for the calculations is correct; on the other hand, it is reassuring to look at the new ratio and see that it has the same (correct) value for each run. This is the known value of the internal calibration ratio, and obtaining it for each run is an indication that the calculations are proceeding as desired.

## 9.4 Executing Rep

The menu for *Rep* requests the analyst to choose which type of listing he or she wants. Some options require explicit selection of a listing format; others assign a format automatically. The options should be largely self-explanatory and are described briefly below. They are listed in Table 9.6.

---

Table 9.6 Rep Menu

- 0 to EXIT program
- 1 to edit Out.dat
- 2 to average specified samples
- 3 to list specified samples
- 4 to list permanent companions
- 5 to list analyses of standards
- 6 to list by batch code
- 7 to list by element
- 8 to list sample id codes only
- 9 to list all samples between limits
- 10 to correct Pu to a given date

---

### 9.4.1 Option 0: Exit

Entering a zero for the menu option tells the program you are done with *Rep*. Control returns to the monitor.

### 9.4.2 Option 1: Edit Out.dat

*Out\_Edit* is called by both *File* and *Rep*. Its operation is described in Section 4.11.

### 9.4.3 Option 2: Average specified samples

This option requests the analyst to identify which samples are to be averaged. Either *Out* file record numbers or sample identification codes can be input. Note that no checking is done to verify the validity of input. The program will blindly average ratios and atom percents for all entries, even if some are incompatible with others. To take an extreme example, it would try to average results from different elements, resulting in averages without meaning. To be valid, all records must be for the same element calculated using the same *Ratio* file record.

### 9.4.4 Option 3: List Specified Records

This option results in a listing of those samples identified by the analyst. No averages are calculated.

#### 9.4.5 Option 4: List Permanent Companions

This option lists the contents of the first 50 records in the *Out* file. It is similar to the option accessed via *Out\_Edit* (Section 4.11.5).

#### 9.4.6 Option 5: Special Standard List

This option lists results for standards (batch code ST) on the printer.

#### 9.4.7 Option 6: List by Batch Code

This routine lists samples with a chosen batch code. This use of the batch code is very helpful in identifying samples from the same customer or project.

#### 9.4.8 Option 7: List by Element

This option results in listing the results for a given element that are held between specified record limits in the *Out* file.

#### 9.4.9 Option 8: List Sample Identification Codes

This option can be extremely helpful in case problems arise. It lists only the sample identification codes and their associated records. An example of when this has been helpful is when two (or more) samples have been given identical identification codes. In seeking information in the *Out* file for companion samples, subroutine *Outloo* (Section 11.25) will locate only the first one it encounters. Getting a list of codes helps unscramble what has happened.

#### 9.4.10 Option 9: List All Between Limits

This routine does just what you might expect: It lists all entries between record extremes chosen by the analyst.

#### 9.4.11 Option 10: Correct Pu

This is one of several places that allows correction of the isotopic composition of plutonium samples to a given date. It can also be done in program *MSC* (See Section 8.10.9).

### 9.5 Automatic Calculation of Averages

This feature was available on the PDP 11/23 but is not available on the VAX. It was used so seldom that the effort necessary for the conversion didn't seem worthwhile. In the event, however, that something like this is needed in the future, it seems desirable to enter a brief description of the protocol here; a full listing of the code (in Subroutine *Rep3*) can be found in the three-ring binder holding listings of all PDP 11/23 programs.

The idea behind automatically calculating averages was to spare the operator the tedium of identifying manually which samples were to be averaged. Such a feature made good sense when we had three instruments humming along at forty hours a week each. To make the feature work, the analyst had to conform to certain conventions. Samples whose identification codes had the first five characters the same were deemed to be replicate analyses and therefore eligible for averaging. The program had logic in it that resulted in averaging the results of all replicate analyses for one element first and then a second, and so on. It was important for the analyst to have specified reasonable record limits in *Out* file to avoid unwanted output.

This routine was used extensively in the early years of the program but declined with the sample load. It'd be nice to think it'll be needed again. Who knows?

## 10. MISCELLANEOUS PROGRAMS

There are several other programs that were written to support isotope ratio measurement activities. These programs and a short description of them are listed in Table 10.1.

---

Table 10.1 Miscellaneous Programs

<u>Program</u>	<u>Description</u>
Atwtpc	Converts ratios to atom percents
IDA	Isotope dilution analysis calculations
Putz	Corrects Pu isotopics to a given date

---

### 10.1 Program Atwtpc

This program converts atom ratios to atom percents. Input is accomplished using subroutine `Elemental_Input` (Section 11.12). Weight percents are also calculated.

### 10.2 Program IDA

IDA standards for Isotope Dilution Analysis. Such analyses performed on instruments other than those designed at ORNL have no provision in the software to perform the necessary calculations; two examples are the IMMA SIMS instrument and the VG-354. Why isotope dilution wasn't built into VG's software remains a mystery.

#### 10.2.1 Isotope Dilution

This section is very similar to that given in Section 11.1 of the Operator's Manual. It is included here for convenience.

Isotope dilution is a very powerful technique for determining the amount of an element present in the sample. It is the most accurate method available in many cases, particularly at low concentrations. It is a very important aspect of our work. For an isotope dilution analysis, a known amount of an isotopically enriched spike is added to a known amount of sample, whose isotopic composition is different from the spike's. The ratio of the sample isotope to the spike isotope in the mixture of spike and sample is measured. This value, along with various other parameters, is fed into the isotope dilution equation and the concentration of the element in the sample calculated. The details of this calculation are discussed below.

##### 10.2.1.1 The Isotope Dilution Equation

The isotope dilution equation has several algebraically equivalent forms; we use one first developed by Hintenberger.<sup>8</sup> Here it is!

$$C = (W_t/W_m) (A_m/A_t) (a_{tk}/a_{mk}) (R_m - R_t)/(R_s - R_m)$$

where:

- C = concentration
- W = weights
- A = atomic weights
- a = atom percents (or atom fractions)
- R = ratios

and the subscripts are defined as follows:

- s = unspiked sample
- t = spike (tracer)
- m = mixture of sample and spike.

The ratios ( $R_x$ ) are defined:

$$R_x = a_{xi}/a_{xk}$$

where x is s, t, or m as defined above and i and k refer to the two isotopes of the isotope dilution ratio. Isotope i is usually the isotope of high abundance in the sample and k the isotope of high abundance in the spike, but using them (consistently!) in the reverse definition is valid algebraically. For example, for uranium, i is usually 238 and k 233.

Note that the units of amounts of spike and sample will define the units of the result. Note also that all the terms except those containing  $R_m$  are constant for a series of replicate analyses.

The ratios  $R_x$  are called the isotope dilution ratios. They are the ratios of the abundance of sample isotope to that of the spike isotope for each of the three components. For uranium, they would usually be  $^{238}\text{U}/^{233}\text{U}$ .

#### 10.2.1.2 Reverse Isotope Dilution

Reverse isotope dilution is a special case of isotope dilution. It is most often encountered when the analyst is calibrating the concentration of a spike. When reverse isotope dilution is called for, the variables designated by subscripts *s* and *t* in the equations above are interchanged. This has the effect of making the unspiked sample the spike and the spike the sample.

An option is provided in the program to do this automatically. It is merely a convenience for the analyst; there is no necessity ever to employ the reverse isotope dilution feature. To use it saves manual entry of information into the *Spike* and *Out* files. For a normal isotope dilution analysis, the program assumes information for the spike is in the *Spike* file and information for the unspiked sample is in the *Out* file. When a spike is being calibrated, its isotopic and other information is in the *Spike* file, but, for reverse isotope dilution calculations, it is actually the sample. Similarly, the material most often used to

calibrate uranium spikes is NIST U-950 (natural); its isotopic composition is kept in the permanent companion portion of the *Out* file. To circumvent the need to enter the spike being calibrated in the *Out* file and NIST U-950 in the *Spike* file, the concept of reverse isotope dilution was introduced.

To use reverse isotope dilution, a 1 must be entered for that index in the set-up part of *MSR* or after the fact using program *File*. Enter the file record number in the *Spike* file occupied by the spike being calibrated. Enter the six-character code that specifies NIST U-950 in the *Out* file; in the past, this has been NBS950. Enter the weights as though it was a conventional isotope dilution analysis. Isotope dilution calculations proceed as normal. Output is in the usual format.

IDA provides two ways of entering the information necessary. One is by strictly manual entry of all parameters--and there are a lot! The second accesses the *Spike* and *Out* files for spike and sample information, respectively. The latter method usually saves time, especially if the necessary information is already in the files or if a large number of samples are to be processed. The manual method is to be preferred if only one or two samples are to be processed and spike and sample information would have to be entered in any case.

IDA accesses several subroutines, which are listed in Table 10.2.

---

Table 10.2 Subroutines in IDA

Read\_File\_Info  
 Set\_IDA\_Params  
 Read\_Manual\_Input  
 IDA\_Calc  
 Print\_Results  
 Read\_Mix\_Ratios  
 List\_Instructions

---

These subroutines are described below.

IDA uses a structure to carry information between subroutines. Its contents are given in Table 10.3.

---

Table 10.3 Structure for IDA

```
STRUCTURE /ida_rec/
  INTEGER*2 spike_rec,
  1      unspk_rec,
  2      nr_samp,
  3      iso_diln_ratio(2),

  REAL*4  samp_wgt,
  1      spike_wgt,
  2      diln_wgt,
  3      aliq_wgt,
  4      spike_ratio,
  5      unspk_ratio,
  6      mix_ratio(10),
  7      spike_at_wgt,
  8      unspk_at_wgt,
  9      spike_at_pct_denom,
  1     unspk_at_pct_denom,
END STRUCTURE
```

---

Variable names within the structure are supposed to be descriptive of what they represent.

#### 10.2.2 Subroutine Read\_File\_Info

This routine is accessed when the analyst chooses input from disk files. Values for the record numbers in *Spike* and *Out* are read. Weight information for the spiked sample is also entered in this routine.

#### 10.2.3 Subroutine Set\_IDA\_Params

This routine assigns values to various parameters needed in isotope dilution analysis when data for the spike and unspiked sample are to be read from disk files. Most of these assignments deal with variables carried in the *ida* structure (Table 10.3). The indicated record in each file is read. Note that it is possible to perform the calculations without an unspiked companion. This requires that the spike isotope not be present in the sample, a situation not obtaining for most elements; two such elements are U and Th, however, so we use this feature a fair amount.

#### 10.2.4 Subroutine Read\_Manual\_Input

This routine opens a window and accepts keyboard entry of the 12 variables required for the calculations. Provision is made to allow correction of an erroneous entry.

### 10.2.5 Subroutine IDA\_Calc

This routine performs the isotope dilution calculation using the Hintenberger equation<sup>8</sup> (described above and in Section 11.1 in the Operator's Manual).

### 10.2.6 Subroutine Print\_Results

This is the output routine for IDA. Normal output is to the printer, but it can be directed to the terminal instead by compiling with /D\_LINES option (see lines 31 and 32 in the listing).

### 10.2.7 Subroutine Read\_Mix\_Ratios

This routine reads values for the experimentally determined isotope dilution ratio (e.g., 238/233 for U) in the spiked sample. The array holding those values is dimensioned 10, so data for that many replicate analyses can be processed.

### 10.2.8 Subroutine List\_Instructions

This routine's sole function is to list abbreviated instructions to the screen. It is designed to remind someone of what he or she may have forgotten and not to serve as full-fledged instructions for the neophyte.

## 10.3 Program Putz

Putz (German for clean) is intended to mean Plutonium Time Zero. It is a stand-alone program that fulfills the same function as calls to subroutines made from various larger programs (*File* and *MSC*). This one allows entry of any known isotopic composition of Pu and the date for which it is valid. It also provides the ability to correct NIST Pu-947 to any chosen date.



## 11. DHS\_LIB LIBRARY

DHS\_Lib is a formal Fortran library containing generally useful subroutines and functions. They are used throughout the suite of main line programs (*MSR, MSC, File, Rep*). A list of the subroutines appears in Table 11.1.

---

Table 11.1 DHS\_Lib Subroutines

<u>Subroutine</u>	<u>Description</u>
Add_Blanks	Adds blanks to end of string
Afapct	Converts atom fraction to atom percent and vice versa
Amax	Finds the largest element of a real array
Arrsum	Calculates the sum of a real array
Atpct	Calculates atom percents from ratios
Aver	Calculates average of a real array
Blank_String	Fills a string variable with blanks
Check	Checks if a string is filled with blanks or not
DDat	Returns the date in U.S. format
Dec	Decrements an integer
Devin	Reads in output device number
Elemental_Input	Reads in element information
Iaver	Calculates average of an integer array
Inc	Increments an integer
Llsqf	Linear least squares fit
Look	Locates element in <i>Massab</i> file
Max	Locates maximum of integer array
MaxReal	Locates maximum of real array
Min	Locates minimum of integer array
MinReal	Locates minimum of real array
Nday	Calculates number of days between two dates
Niscal	Determines number of isotopes and number of ratios
Non_Zero_Els	Determines number of non-zero elements in an array
Odd	Determines if an integer is odd or even
Outlie	Checks array for outliers
Outloo	Finds a sample record in file Out
Qsort	Quicksort; arranges elements of an array in order
Stdev	Calculates standard deviation and other statistics
Swap_Integer	Interchanges two integers
Swap_Real	Interchanges two real numbers
Swap_String	Interchanges two strings
Tzero	Calculates Pu isotopic composition at a given date
Ucase	Converts a string to upper case
Upmtsm	Updates array of uncalculated samples
Upout	Updates Out file bookkeeping records
Wgpct	Calculates weight percents
Xmass	Correlates exact mass with nominal mass array

---

## 11.1 Subroutine Add\_Blanks (str, size, n)

Arguments: str: string variable  
 size: size of string  
 n: integer number of blanks to add

This routine adds n blanks to the end of string variable *str*. It is useful in maintaining tidy screen output.

## 11.2 Subroutine Afapct (ix, niso, ab)

Arguments: ix: integer index = 1 to convert atom pct to atom fraction  
 = 2 to convert atom fraction to atom pct  
 niso: integer number of isotopes  
 ab: real abundance array (fraction or percent).

This routine was written to convert atom percents to atom fractions and vice versa. Some files have isotopic abundances stored in one format, some in the other, and calculations require that all concentrations be in the same units.

An index (*ix*) is used to indicate what units you desire. A value of 1 yields atom fractions; a value of 2 gives atom percents. One of the weaknesses of Fortran makes it necessary to transmit the number of isotopes as an argument for use as a DO loop limit. There are checks to prevent performing the operation when abundances are already in the desired units; atom fractions cannot have any abundance greater than 1.0, and it is safe to assume atom percents must have at least one abundance greater than 1.0%. While this is not mathematically exact (we could theoretically have 101 components all less than 1%), in the real world of isotopes it is uniformly true.

## 11.3 Subroutine AMax (npt, array, ib, rmax)

Arguments: npt: number of points in the array  
 array: array of real numbers  
 ib: location of array's maximum value  
 rmax: maximum value

This routine locates the position of the maximum value in a single-dimensional real array. It returns both the position (subscript value) and the maximum value itself. See also Subroutines *Max* (Section 11.18) and *MaxReal* (Section 11.19).

## 11.4 Subroutine Arrsum (n, arr, sum)

Arguments: n: array size  
 arr: array of real numbers  
 sum: real sum of array

This routine calculates the real sum of a single-dimensional array of real numbers.

## 11.5 Subroutine Atpct (niso, r, apct, moms, mass)

Arguments: niso: number of isotopes  
 r: ratios (array of real numbers)  
 apct: atom pct (array of real numbers)  
 moms: mass of denominator of ratios  
 mass: array of masses of isotopes (integer)

This routine calculates atom percent abundances from the values of isotopic ratios transmitted as input. The ratios must all have the same mass in the denominator. For uranium, for example, the ratios sent might be 233/238, 234/238, 235/238, and 236/238. The process is complicated by the fact that the array of ratios may or may not include that of the denominator to itself (obviously 1.00). A value of 0 for *moms* indicates that this value of 1.0 is included; any other value is assumed to be the mass number of the denominator. In the latter case, 1.0 is inserted at the appropriate location in the array before atom percents are calculated. Doing this keeps intact the sequential correlation of subscript with mass number.

## 11.6 Subroutine Aver (ni, r, avg, sd, sdm, civ)

Arguments: ni: number of points  
 r: array of real numbers  
 avg: average of r  
 sd: standard deviation  
 sdm: standard error of the mean  
 cir: 95% confidence interval

This subroutine calculates the real average, standard deviation, standard error of the mean, and the 95% confidence interval for an array of real numbers. It calls *Arrsum* (Section 11.4) and *Stdev* (Section 11.29) which do most of the work. Definitions of the statistical terms are given in Section 11.29.

## 11.7 Subroutine Blank\_String (str)

Argument: str: string variable

This subroutine fills a string with blanks. It is useful to call before input of a variable to eliminate meaningless characters from appearing on the display.

## 11.8 Subroutine Check (nd, name, ich)

Argument: nd: number of characters in variable name  
 name: string variable  
 ich: index = 1 for no change  
       = 2 if name is changed

This subroutine was written to help input of strings when editing a file. In edit mode, the programs often use entry of the RETURN key alone to indicate that the original value should be retained. This routine reads a variable string and checks for a non-blank character.

If a non-blank character is encountered, the value of the input string is assigned to the variable name. If no non-blank character is found, variable name remains unaltered.

This routine is seldom used on the VAX-PC system.

#### 11.9 Subroutine Ddat (idat)

Argument: idat: array of integers holding the date

This routine returns the date in normal U.S. convention (mm-dd-yy). DEC's house routine originally gave dd-mm-yy (still the European convention).

#### 11.10 Subroutine Dec (int, nr)

Arguments: int: number  
nr: number to decrement by

This routine decrements an integer (*int*) by the specified amount (*nr*). Compare Subroutine *Inc* (Section 11.15).

#### 11.11 Subroutine Devin (idev)

Argument: idev: device code (integer)

This routine asks for a device code number (6 for printer, 7 for terminal) and returns the value interval at the keyboard. Values other than 6 are returned as 7. It uses the normal terminal screen for input.

#### 11.12 Subroutine Elemental\_Input( )

This rather complex routine is no longer in common use. It accepted input associated with a given element: symbol, isotopic mass and abundances, etc. It works only for the standard terminal screen and not for a custom window. It has been superseded by other routines.

#### 11.13 Subroutine Getich (ich)

Argument: ich: index (integer) accepted from the keyboard

This routine has an index value usually associated with editing. It is not much used with the VAX, because it uses the standard terminal screen for input.

#### 11.14 Subroutine Iaver (ni, ir, avg, sd)

Arguments: ni: number of points  
ir: array of integers  
avg: real average of integer array  
sd: standard deviation

This routine calculates the average and standard deviation of an integer array. It complements subroutine *Aver* (Section 11.6), which does the same for real numbers. Both the average and standard deviation are returned as real variables.

#### 11.15 Subroutine Inc (int, nr)

Arguments: int: integer to be incremented  
nr: amount int will be incremented

This routine increases an integer by a specified amount. For example, Inc (x, 5) serves the same purpose as  $x = x + 5$ . Compare subroutine *Dec*.

#### 11.16 Subroutine Llsqf (npt, x, y, slope, yint, sd)

Arguments: npt: number of points  
x: array of x values (real)  
y: array of y values (real)  
slope: slope of best-fit line (real)  
yint: y intercept of slope (real)  
sd: standard deviation (real)

Llsqf stands for Linear Least Squares Fit. The best linear fit to an array of points (x and y values) is calculated and returned as a slope and intercept; its standard deviation is also returned. The equations for the calculations were taken from *The Handbook of Chemistry and Physics*, 45th Edition, p. A164.

#### 11.17 Subroutine Look (iel, nis, xmas, mass, ab)

Arguments: iel: element symbol (CHARACTER\*2)  
nis: number of isotopes  
xmas: array of exact masses (real)  
mass: array of nominal masses (integer)  
ab: array of abundances (real)

This routine locates an element (as defined by the symbol) in the *Massab* file (Section 4.6) and returns information associated with it.

#### 11.18 Subroutine Max (isz, nar, ib, mx)

Arguments: isz: number of points in array (size)  
nar: array of integers  
ib: location of maximum  
mx: value of maximum

This routine complements subroutine *Amax* (Section 11.3). It identifies the maximum element in a one-dimensional integer array and returns its location (subscript number) and value. See also Subroutine *Min* (Section 11.20).

## 11.19 Subroutine MaxReal (size, array, locn, max)

Arguments: size: integer size of array  
 array: array of real numbers  
 locn: location of maximum (integer)  
 max: maximum value (real)

This routine is identical to *AMax* (Section 11.3). Sometime over the course of the years, I must have forgotten the existence of one of them. It locates the position and value of the maximum of a one-dimensional array of real numbers.

## 11.20 Subroutine Min (isz, nar, ib, mx)

Arguments: isz: number of points in array  
 nar: array of integers  
 ib: location of minimum  
 mx: value of minimum

This subroutine identifies the minimum element in a one-dimensional array of integers and returns its location and value. Compare to *Max* (Section 11.18).

## 11.21 Subroutine MinReal (size, array, locn, min)

Arguments: size: integer size of array  
 array: array of real numbers  
 locn: location of minimum (integer)  
 min: value of minimum (real)

This routine complements *MaxReal*. It returns the location and value of the minimum in a one-dimensional real array.

## 11.22 Function Nday (jdat, idat)

Arguments: jdat: initial data  
 idat: final data

This integer function calculates the number of days between two dates. It is particularly useful in calculating isotopic compositions adjusted for radioactive decay (e.g., Pu). It is called from Subroutine *Tzero* (Section 11.33) for that purpose.

Note that the dates must be in the format mm-dd-yy. A negative value for *Nday* is perfectly legitimate; it simply means that the final date chronologically precedes the initial. In this case, you would be correcting the isotopic composition to some specified date in the past.

## 11.23 Subroutine Niscal (mrat, msum, nis, nrat)

Arguments:   mrat:   array of numerators and denominators of ratios (integer)  
               msum:   array of masses  
               nis:     number of isotopes  
               nrat:   number of ratios

This subroutine is required to address one of Fortran's weaknesses (no built-in function to tell you how many positions in an array are filled; Modula-2's HIGH function does this). The program searches first the *mrat* array and then the *mass* array, seeking the first occurrence of a zero in each. The variables *nrat* and *nis* are returned as one less than these subscripts. Note that this means no embedded zeros are allowed in the *mrat* and *mass* arrays. There is no reason why it would be desirable, so this constitutes no real limitation.

## 11.24 Function Non\_Zero\_Els (array)

Arguments:   array:  array of real numbers

This function (Integer\*2) returns what is assumed to be the number of non-zero elements in a real array. It determines this value by locating the first occurrence of a zero--embedded zeros are thus not provided for. The maximum number of elements allowed is 100; this can be altered simply by changing the PARAMETER statement to assign the desired value to *max\_size*.

## 11.25 Function Odd (nr)

Argument:   nr:     integer number

This function determines whether or not *nr* is odd, returning a 0 if it is odd, a 1 if it is even.

## 11.26 Subroutine Outlie (npt, rato, avg, sd, sdm, cir, kstar)

Arguments:   npt:   number of points in the array  
               rato:   array of real numbers (usually isotopic ratios)  
               avg:    average of the array  
               sd:     standard deviation  
               sdm:   standard error of the mean  
               cir:    95% confidence interval  
               kstar:  array of asterisks and blanks

This subroutine identifies statistical outliers in an array of real numbers. A variable associated with the real array, *kstar*, identifies an outlying value by assigning an asterisk (\*) to it (rather than a blank).

This is a one-pass operation. Once a new average and standard deviation have been calculated, no further rejection tests are applied.

Consultation with C. K. Bayne, a professional statistician, led to the use of the studentized deviate test (Nair's statistic) as found in J. H. Sheesley, *J. Quality Tech.* 9(1), 38-41 (1977). The table of t values is from F. E. Grubb and G. Beck, *Technometrics* 14(4), 848-850 (Table 1, upper 2.5% significance level). If there are more than 25 points, a simple test against twice the standard deviation is used.

Once outliers have been identified, they are eliminated from the array and the average recalculated for return to the calling program. All the statistical parameters are also recalculated and returned.

#### 11.27 Subroutine Outloo (*samp\_id*, *out\_rec*, *indx*)

Arguments: *samp\_id*: 6-character sample identification code  
*out\_rec*: record number in *Out* file  
*indx*: = 1 to look up existing sample code  
= 2 to insert new sample code in *Bkout*

This routine serves two functions. One is to locate the position in the *Out* file of a sample whose data are already there. The second is to insert a new sample code in file *Bkout*.

The *Out* file contains results for the 3000 samples most recently run. Many calculations (e.g., isotope dilution) require isotopic abundance information from samples stored in this file. Information for the sample is located by comparing each successive six-character sample identification code with the desired one until the two are identical. This can require up to 3000 comparisons, and reading each sample code in turn using a separate disk read operation takes far too long to be convenient. For this reason, a companion file to *Out* is maintained. It is named *Bkout.dat* and contains the 3000 six-character codes of the samples in *Out* in sequence. In search mode, *Outloo* reads these codes from the disk in 1000-sample blocks and searches the array for the target value. It returns either the record number in *Out* where information for the sample is stored or a zero if the sample is not found.

In insertion mode, *Outloo* inserts the value of the new sample identification code (*samp\_id*) in the record specified by *out\_rec*. The new array is then written to disk. See also subroutine *Upout* (Section 11.36).

#### 11.28 Subroutine Qsort (*inx*, *n*, *a*, *index*)

Arguments: *inx*: an index; = 1 for an increasing array  
= 2 for a decreasing array  
*n*: number of points  
*a*: array of real numbers  
*index*: array of integers

Ordering an array is one of the oldest problems addressed by computers. It is an ideal application of recursion (having a subroutine call itself), but unfortunately Fortran doesn't provide this ability, and a more complex routine is required. The one used here was

adapted from G. Beech, "Successful Software for Small Computers," John Wiley and Sons, New York, 1982, pp. 129-131.

A random real array (*a* in the argument string) is organized into an increasing or decreasing array as indicated by the value of *inx*. An array of integer index values is also sorted, each index following its associated real value. This allows ready identification of information associated with the original real number. For example, if an array of ratios is sorted, using the index as a pointer allows correct identification of the standard deviation (an unsorted array) associated with each ratio.

It is up to the programmer to set up the index array running from 1 to *n* before calling *Qsort*.

#### 11.29 Subroutine Stdev (ni, r, avg, sd, sdm, cir)

Arguments: ni: number of points  
 r: real array  
 avg: real average  
 sd: standard deviation  
 sdm: standard error of the mean  
 cir: 95% confidence interval

This routine calculates the standard deviation, standard error of the mean, and the 95% confidence interval for an array of real numbers; the average of the array and the number of points are required as input.

The equation used to calculate the standard deviation is:

$$sd = \left[ \frac{\sum(x_i - x)^2}{n-1} \right]^{1/2}$$

where  $x_i$  are the individual values and  $x$  the average of  $n$  points. This is the equation used in the ubiquitous hand calculators.

The standard error of the mean is then  $sd/n^{1/2}$ . The 95% confidence interval is  $t(n) * sdm$  where  $t(n)$  is the appropriate value from a t-table. The t-table used here is from M. Kastenbaum, "Lecture Series in Statistics and Probabilities: Lecture IV: Confidence Regions," July 1968, p. 22. This table is available from many other sources.

#### 11.30 Subroutine Swap\_Integer (a,b)

Arguments a, b: integers

There are many times in programming where it is necessary to interchange the values of two variables. This subroutine does so for integers. See also `Swap_Real` and `Swap_String` (Sections 11.29 and 11.30).

#### 11.31 Subroutine `Swap_Real` (a, b)

Argument: a, b: REAL

This routine interchanges the values of two real variables.

#### 11.32 Subroutine `Swap_String` (a, b)

Argument: a, b: CHARACTER \*(\*)

This routine interchanges values of two string variables. Note that 80 is the maximum string length that can be accommodated, although it would be simple enough to alter it by changing appropriately the CHARACTER \*80 type for the variable *hold*.

#### 11.33 Subroutine `Tzero` (iel, niso, ab, jdat, idat, atw, iday, ato)

Arguments: iel: element symbol  
 niso: number of isotopes  
 ab: abundances (real array)  
 jdat: initial date  
 idat: final date  
 atw: atomic weight at idat  
 iday: number of days between jdat and idat  
 ato: atoms at idat

This routine calculates the isotopic composition of an element at a specified date. It is currently implemented only for plutonium, but it would be relatively simple to extend its application to other elements. Required input information is the number of isotopes, (*niso*), the isotopic composition of Pu (*ab*) at a specified date (*jdat*), and the date for which you want the composition (*idat*). Information returned is the new composition (*ab*), the atomic weight, the number of days between the two dates, and the number of atoms of Pu at the final date.

Note that the two dates are arrays of three integers in the sequence mm-dd-yy. The number of days is calculated by subroutine *Nday* (Section 11.22).

#### 11.34 Function `Ucase` (string)

Argument: string (CHARACTER \*(\*))

This function (CHARACTER \*(\*)) changes a string to all upper case characters. The function itself is typed CHARACTER \*(\*) to make it applicable to any length of string variable. VAX Fortran's LEN function is used to determine the length. It is convenient to use *Ucase* when you want to force a variable to all upper case characters or prior to making a comparison for which case is irrelevant (such as an element symbol).

## 11.35 Subroutine Upmtsm (indx, nr, mch)

Arguments: indx: an index defined below  
 nr: record number  
 mch: instrument identification code

The routine handles bookkeeping chores for the array holding record numbers in file *Samp* of samples to be calculated. The specific action taken is controlled by the variable *indx* as defined in Table 11.2.

---

Table 11.2 Values of indx in Upmstm

<u>Value</u>	<u>Result</u>
1	Delete record number from array
2	Insert record number in array
3	Insert record number in array and update <i>Samp</i> file.

---

A value of 1 is most often sent from *C2* in *MSC* (Section 8.22), when data for the sample are being processed for the first time. *Indx* = 2 most often happens when editing the *Samp* file in program *File*. A value of 3 is sent from program *Rec* (Chapter 7) when new data are being transferred to the *Samp* file.

## 11.36 Subroutine Upout (lsam, nr)

Arguments: lsam: six-character sample code  
 nr: record number in *Out* file

This routine updates the *Out* and *Bkout* files by inserting new information in them. It operates only on record numbers and sample codes; it does not write mass spectrometric results to *Out*. The program first searches to see if the sample (identified by *lsam*) is in the *Out* file; if it is, no action is taken and the record number holding the data returned as *nr*. If it isn't found, the value of the last record used in *Out* is incremented by one and *lsam* is inserted in *Bkout* (by using Subroutine *Outloo*, Section 11.27). If incrementing the last record used results in a value greater than 3000, the value is reset to 51--remember that the first 50 records of *Out* are reserved for permanent companions.

## 11.37 Subroutine Wgpct (nis, apct, emas, awsam, wpct)

Arguments: nis: number of isotopes  
 apct: atom percent array  
 emas: exact mass array  
 awsam: atomic weight  
 wpct: weight percent array

This routine calculates weight percents and atomic weight from a specified composition in atom percents. Subroutine *Afapct* (Section 11.2) is called to convert atom fractions to percents if necessary.

11.38 Subroutine *Xmass* (nis, niso, mass, xmas, exmas)

Arguments:   nis:    number of isotopes in sample  
              niso:   number of isotopes in *Massab* file  
              mass:   array of nominal masses (sample)  
              xmas:   real array of exact masses (*Massab*)  
              exmas:  real array of exact masses (sample)

It sometimes happens that fewer masses are scanned for an analysis than are listed in the *Massab* file for the analyte element. The most frequent such occurrence is for uranium, where 233 is in *Massab* to allow easy accommodation of spiked samples but which may or may not be scanned during an unspiked analysis. The purpose of this routine is to assign the correct exact masses to the isotopes monitored during the analysis.

## 12. SCREEN\_LIB LIBRARY

Screen\_Lib is a library of subroutines and functions that addresses input and output through the terminal. It is a formal Fortran library and must be specified as such to the linker.

One of the inconveniences of dealing with DEC's windows (also true of most other operating systems) is that they will accept only strings as input or output. Much of our work involves numerical information, so conversion subroutines were written to ease the strain for the programmer. There are also routines for moving the cursor, for opening and closing windows and screens (virtual displays and pasteboards in DEC's lingo), and for erasing all or part of a window. The contents of Screen\_Lib are listed in Table 12.1

---

Table 12.1 Contents of Screen\_Lib

<u>Subroutine or Function</u>	<u>Description</u>
AnyKey	Pauses program until a key is struck
Create_Pasteboard	Creates a pasteboard (screen)
Create_Virtual_Display	Creates a virtual display (window)
Create_Virtual_Keyboard	Activates the keyboard for input
Create_Window	Creates both a pasteboard and a virtual display
Delete_Virtual_Display	Removes virtual display
Delete_Virtual_Keyboard	Removes keyboard
Erase_Display	Erases all or part of a display
Erase_Line	Erases one line of text
Erase_Pasteboard	Erases and saves pasteboard
Erase_Screen	Clears the screen
Erase_Total_Display	Erases entire display (window)
Get_Index	Special case of Get_Integer for indices
Get_Input	Reads string input from keyboard
Get_Int	Reads up to 6 characters as an integer
Get_Integer	Reads an integer from a special window
Get_Real	Reads a real number
Get_Record	Special case of Get_Integer for record numbers
Mask	Opens an input window with messages displayed
Paste_Virtual_Display	Pastes a virtual display to a pasteboard
Put_Int	Outputs an integer to a display
Read_El_Sym	Reads an element symbol
Trim	Generates a string devoid of trailing blanks
Unpaste_Virtual_Display	Removes and saves display
Write_Msg	Writes a message to the display

---

Several of the subroutines do nothing but mimic those provided by DEC but are more convenient for the programmer to use. Most of the routines whose names begin with *Create\_*, *Delete\_*, and *Erase\_* are of this sort. In DEC's recommended usage, each call to one of these routines required two statements: one to effect the required action, the other to

verify valid execution. In addition, each call involved a prefix (SMG\$ or LIB\$). Another irritation was that all integers had to be typed INTEGER\*4. All this got tiresome, so routines that were more user-friendly were developed. As an example, consider the subroutine *Delete\_Virtual\_Display*. The executable code consists of two lines:

```
Status = SMG$Delete_Virtual_Display (vdid)
IF(.NOT.) status CALL LIB$signal (% val status)
```

Not only are these lines tedious to type, but incessant inclusion of the second line reduces the readability of the code.

### 12.1 INTEGER\*4 vs. INTEGER\*2

All integer variables generated within the programs described in this manual are typed INTEGER\*2. This decision was made to retain compatibility with data files already in existence of the old PDP 11/23. Unfortunately, however, DEC subroutines generally require INTEGER\*4 variables. This mismatch has required a certain amount of legerdemain in the programming. For example, in subroutine *Create\_Pasteboard*, two of the arguments are the size of the pasteboard in rows and columns. DEC's *SMG\$Create\_Pasteboard* routine requires these to be INTEGER\*4, necessitating a couple of assignment statements to effect the necessary conversion from INTEGER\*2. INTEGER\*2 variables that will be converted to INTEGER\*4 often have names ending in 2--e.g., row2.

The only exceptions to this approach are for the ubiquitous variables identifying pasteboards and virtual displays. These variables are typed INTEGER\*4 at all occurrences. They are also consistently named; pasteboard variables have *pbid* and virtual displays *vdid* associated with their variable names.

### 12.2 Subroutine AnyKey (pbid)

This routine is convenient to use when you want to halt the program to allow the user to read a window of information. It opens a small window, displays the message 'Hit any key to continue,' and, once the user hits a key, closes the window and proceeds with execution.

### 12.3 Subroutine Create\_Pasteboard (pbid, row2, col2)

```
Arguments:  pbid:
             row2: size of pasteboard in rows (INTEGER*2)
             col2: size of pasteboard in columns (INTEGER*2)
```

This routine creates a pasteboard (called a screen in many operating systems) in which virtual displays (windows) can be opened.

#### 12.4 Subroutine Create\_Virtual\_Display (vdid, size\_row2, size\_col2, attributes2, title)

Arguments: vdid  
 size\_row2: size of display in rows (INTEGER\*2)  
 size\_col2: size of display in columns (INTEGER\*2)  
 attributes2: See below (INTEGER\*2)  
 title: CHARACTER\*(\*)

This subroutine creates a virtual display (often called a window in other operating systems) for terminal I/O. It can have a title; if no title is desired, set the string variable to 'NONE.' One of the arguments controls what DEC calls the 'attributes' of the display. The meanings of the various values this variable can assume are given in Table 12.2.

---

Table 12.2 Values for Attributes

<u>Value</u>	<u>Description</u>
0	Default attributes
1	Bold
2	Reverse video
4	Blink
8	Underline

---

Use the sum of these values to get more than one of them. Using 15, for example, would give a reverse video display that is in bold, blinking, underlined characters. Sounds wild!

#### 12.5 Subroutine Create\_Virtual\_Keyboard (kbid)

Argument: kbid: keyboard id (INTEGER\*4)

To effect input through a virtual display, the keyboard must be activated. If you want to use the keypad, that must be activated, also. This routine does both.

#### 12.6 Subroutine Create\_Window (pbid, vdid, size\_row, size\_col, cor\_row, cor\_col, attributes, title)

Arguments: pbid  
 vdid  
 size\_row: size of window in rows  
 size\_col: size of window in columns  
 cor\_row:  
 cor\_col: row and column of upper left corner of display  
 attributes: See Section 12.4 for description

To use a window for input, it is necessary both to create a virtual display and to paste it. This routine does both by calling *Create\_Virtual\_Display* (Section 12.4) and *Paste\_Virtual\_Display* (Section 12.21).

#### 12.7 Subroutine *Delete\_Virtual\_Display* (vdid)

This routine deletes a virtual display. DEC defines "delete" as removing the visible display and also its image from memory. Contrast this with *Unpaste\_Virtual\_Display* (Section 12.25), which only removes the visible display and retains its image in memory. To recreate a display that has been deleted, it is necessary to go through the entire display-generating process again, starting with *Create\_Virtual\_Display*.

See also *Delete\_Virtual\_Keyboard* (Section 12.8), which should be executed just before *Delete\_Virtual\_Display*.

#### 12.8 Subroutine *Delete\_Virtual\_Keyboard* (kbid)

Argument: kbid: keyboard id (INTEGER\*4)

This routine deletes a virtual keyboard. It should be executed each time you are done with it. Failing to do this will exhaust the allocated event flags for screen-associated variables and result in a fatal error.

#### 12.9 Subroutine *Erase\_Display* (vdid, row\_begin2, col\_begin2, row\_end2, col\_end2)

Arguments: vdid  
row\_begin2, col\_begin2: row and column to start erasing  
row\_end2, col\_end2: row and column to end erasing

This routine erases all or part of a display. Among the arguments are four that specify the starting and ending rows and columns for the erasure. If either the starting row or starting column is specified as zero, the entire display will be erased; this conforms to DEC usage in their SMG\$ erase\_display subroutine.

If you wish to clear the entire display, using *Erase\_Total\_Display* (Section 12.13) is easier.

#### 12.10 Subroutine *Erase\_Line* (vdid, row2, col2)

Arguments: vdid  
row2  
col2: row and column to start

This routine erases one line, beginning at a specified column. If this column is 1, the entire line is erased.

## 12.11 Subroutine Erase\_Pasteboard (pbid, vdid)

This routine erases a pasteboard; it also saves the physical screen as a precaution. It should be used to remove the pasteboard before leaving a program.

## 12.12 Subroutine Erase\_Screen

This is an old subroutine that simulates "clear screen" routines in many computers. It accomplishes this by executing a string of mystic ASCII characters via a write statement.

## 12.13 Subroutine Erase\_Total\_Display (vdid)

This routine erases an entire display; to erase portions of one, use Erase\_Display (Section 12.9) or *Erase\_Line* (Section 12.10). To remove the display from the pasteboard but retain its contents in memory, use *Unpaste\_Virtual\_Display* (Section 12.25).

## 12.14 Function Get\_Index (pbid, row, col)

Argument: row, col: location of corner of input window

This integer function is a special case of *Get\_Integer* (Section 12.17). It opens a special window with the title 'Edit Index' and with the message 'Index no. to correct, 0 to proceed.' It accepts an integer as input, which it returns to the calling program.

This is a handy routine used to facilitate input of a value for the edit index. It is highly specialized and not of much use otherwise. *Get\_Integer* (Section 12.17) and *Get\_Int* (Section 12.16) are general routines for integer input.

## 12.15 Subroutine Get\_Input (pbid, vdid, nr\_char2, row2, col2, input, text\_size2)

Arguments: pbid  
vdid  
nr\_char2: maximum number of characters  
row2,col2: location of upper left corner of input box  
input: CHARACTER\*(\*)-input string  
text\_size2: length of input

This is a completely general routine that accepts string input. It is used in *Get\_Integer*, *Get\_Int*, and *Get\_Real* for entry of information. Its arguments include the number of characters for the input field, the location (in row and column) of that field, and a variable called *text\_size*. This last is converted to a DEC INTEGER\*4 variable and is required in DECODE statements.

The routine opens a virtual keyboard and creates a reverse video window beginning at the specified location, and having the specified width. It accepts the input string, changes back to normal video, deletes the virtual keyboard, and returns to the calling program.

12.16 Function `Get_Int` (pbid, vdid, row, col)

Arguments: pbid  
 vdid  
 row,col: location of upper left corner of input field

This integer function accepts integer input in a display specified as an argument (*vdid*). The location of the input field is specified as arguments. Its width is fixed at 6; this is not a variable accessible in the Call statement. It uses subroutine *Get\_Input* (Section 12.15) to read a string variable, which is DECODED to an integer and returned to the calling program. Compare to *Get\_Integer* (Section 12.17).

12.17 Function `Get_Integer` (pbid, title, msg, row, col)

Arguments: pbid  
 title: prompt message for input  
 row,col: location of upper left corner of input field

This integer function opens a 3x45 window for input of an integer. It accepts a title and a message as arguments. The location of the display is specified in the argument string. The width of the input field is 6. The display is deleted before return to the calling program. Compare to *Get\_Int* (Section 12.16).

12.18 Function `Get_Real` (pbid, vdid, row, col)

Arguments: pbid  
 vdid  
 row,col: location of upper left corner of input field

This real function serves the same purpose for real numbers as *Get\_Int* (Section 12.16) does for integers. The input string from *Get\_Input* (Section 12.5) is converted to a real number via a DECODE statement. The format used is F10.0, but, as always with Fortran, the presence of a decimal point in the input determines its location.

12.19 Function `Get_Record` (pbid, row, col)

Arguments: pbid  
 row,col: location of upper left corner of input field

Like *Get\_Index* (Section 12.14), this is a specialized use of *Get\_Integer*. A separate display is opened that displays the message "Record no., 0 to stop:" It is frequently used to get the record number in a disk file the user wants to access.

12.20 Subroutine Mask (pbid, vdid, size\_row, size\_col, cor\_row, cor\_col, attributes, title, first\_line, nr\_prompt, prompt\_array)

Arguments: pbid  
 vdid  
 size\_row, size\_col: size of virtual display  
 cor\_row, cor\_col: location of upper left corner of display  
 attributes: see Table 12.2  
 title: for display  
 first\_line: first row for prompts  
 nr\_prompts: number of prompt messages  
 prompt\_array: array of messages to be displayed in window

The function of this subroutine is to open an input mask, complete with prompting messages, on the terminal screen. The size of the display is specified in the argument string. So are its attributes--see Table 12.2 for a list. Other arguments include the display title, the first line upon which you want text to appear, the number of prompting messages, and a string array of the prompts themselves. A value of zero for the number of prompts eliminates writing them to the display. This can also be accomplished by setting prompt\_array [1] = 'NONE.'

A display is opened and the prompting messages (if any) written.

Once control returns to the calling program, the display is ready to accept input via *Get\_Input*, *Get\_Int*, or *Get\_Real*.

12.21 Subroutine Paste\_Virtual\_Display (pbid, vdid, cor\_row2, cor\_col2)

Arguments: pbid  
 vdid  
 cor\_row2, cor\_col2: location of upper left corner of display

This routine "pastes" the contents of a display to the pasteboard. The pasting operation, according to DEC's usage, is the one that makes the display visible on the terminal monitor.

This routine is extremely useful in situations where you are manipulating multiple windows of information. (See Subroutine *Samp\_Edit* in program *File*, Section 4.9, for an example.) Combining *Paste\_Virtual\_Display* with its reciprocal operator, *Unpaste\_Virtual\_Display*, provides an easy means of bringing whichever window of information you want to the fore.

2.22 Subroutine Put\_Int (vdid, row, col, int, digits)

Arguments: vdid  
 row, col: location of output field  
 int: integer to be output  
 digits: width of output field

This routine writes an integer to a specified display at a specified location. The width of the output field is one of arguments; the integer is right-adjusted in the field.

12.23 Function `Read_El_Sym` (pbid, vdid, row, col)

Arguments:   pbid  
              vdid  
              row,col:       location of input field

This is a `CHARACTER*2` function that reads a two-character element symbol and returns it to the calling program. *Get\_Input* (Section 12.15) is used to read the two characters. They are converted to upper case (Subroutine *Ucase*, Section 11.32) to eliminate case problems with symbol matching (e.g., BaþBA in ASCII).

12.24 Subroutine `Trim` (out\_string, in\_string, size2)

Arguments    out\_string, in\_string: `CHARACTER*(*)`  
              size2:               length of string

This routine removes trailing blanks from a string and returns it and the length of the remaining string.

This routine has not been successful. The size returned is often ridiculous (e.g., 538968047--a rather long string!); worse than that it has been known to trash active DO loop indices and other variables. This problem remains unsolved at the time of this writing.

The solution, of course, is to adjust the length of strings in explicit assignment statements.

12.25 Subroutine `Unpaste_Virtual_Display` (pbid, vdid)

This routine removes a display from the terminal monitor and saves the contents in memory. In conjunction with *Paste\_Virtual\_Display* (Section 12.21), it can be used to bring any selected display of a multi-display program to the foreground.

To remove a display from the monitor and from memory, use *Delete\_Virtual\_Display* (Section 12.7). To erase a display and retain its frame for other use, call *Erase\_Display* (Section 12.9) or *Erase\_Total\_Display* (Section 12.13).

12.26 Subroutine `Write_Msg` (vdid, msg, row2, col2, attributes2)

Arguments:   vdid  
              msg:           message to be displayed  
              row2,col2:   location on display where message starts  
              attributes:   See Table 12.2

This routine writes any given message to a display designated in the argument string. Its location and attributes (Table 12.2) are also specified.

## 13. DISK\_IO LIBRARY

Disk\_IO is a formal Fortran library holding routines that facilitate disk operations. Most of the modules address one of two situations: They read or write a specified record in the file; or they read or write a bookkeeping record. In the former case, a structure is used to transmit data from one routine to another. In all cases, the file is opened upon entry to the subroutine and closed upon leaving.

The programs in the library are summarized in Table 13.1.

---

Table 13.1 Contents of Disk\_IO

<u>Subroutine</u>	<u>Description</u>
Record_Check	Verifies record number is valid
DTBias_IO	Disk read-write for DT.Bias.DAT
Ratio_IO	" " " Ratio.DAT
Massab_IO	" " " Massab.DAT
Spike_IO	" " " Spike.DAT
Samp_IO	" " " Samp.DAT
Stand_IO	" " " Stand.DAT
Sumcts_IO	" " " Sumcts.DAT
Last_Out	Bookkeeping record for Out.DAT
Last_Sumcts	" " " Sumcts.DAT
Last_Samp	" " " Samp.DAT

---

To describe each routine individually would be unnecessarily redundant. Each routine whose name ends `_IO` has a similar construction; each routine whose name begins `Last_` has a similar construction different from that of the `_IO` routines. These two types of programs are described generically below.

### 13.1 `_IO` Subroutines

These routines access the file indicated by their names to read or write a record. Each has three arguments: a read-write flag, a record number, and a structure. The read-write flag (*read\_flag*: LOGICAL\*2) is set to `.TRUE.` to read and `.FALSE.` to write. The structure, of course, varies with the file being accessed.

Upon entering the routine, the value of the record number is checked to see if it is valid. A short subroutine, *Record\_Check*, was written to accomplish this. Arguments to *Record\_Check* are the record number, the largest valid value of the record number, and a LOGICAL\*2 variable that is set to `.TRUE.` if the record number is valid and to `.FALSE.` otherwise. An error message is sent to the terminal when an invalid record is encountered. For valid record numbers, the appropriate record is read or written; if read, its contents are returned to the calling program.

### 13.2 LAST\_ Subroutines

These routines read or write the bookkeeping record of the file indicated in the names. There are two arguments: a read-write flag (.TRUE. for read and .FALSE. for write) and the number of the last record used. The bookkeeping record for each file is fixed (e.g., 3001 for *Out*) and its number is not needed as an argument.

## REFERENCES

1. D. H. Smith, "New FORTRAN Programs to Acquire and Process Isotopic Mass Spectrometric Data," ORNL/TM-7002. September 1979.
2. D. H. Smith, "New FORTRAN Computer Programs to Acquire and Process Isotopic Mass Spectrometric Data," ORNL/TM-8356, August 1982.
3. D. H. Smith and H. S. McKown, "New FORTRAN Computer Programs to Acquire and Process Isotopic Mass Spectrometric Data: Operator's Manual," ORNL/TM-12196, 1993.
4. D. H. Smith and D. E. Goeringer, ORNL/TM-10770, May 1988.
5. D. H. Smith, H. S. McKown, W. H. Christie, R. L. Walker, and J. A. Carter, ORNL/TM-5485, June 1976.
6. D. H. Smith, ORNL/TM-9842, November 1985.
7. S. Deron, International Atomic Energy Agency, personal communication.
8. H. Hintenberger in "Electromagnetically Enriched Isotopes and Mass Spectrometry," M. L. Smith, ed., Academic Press, NY, 1958, pp. 177-189.
9. D. H. Smith and J. A. Carter, *Int. J. Mass Spectrom. Ion Phys.* **40**, 211 (1981).
10. ASTM Annual Book of Standards, Part 45: Nuclear Standards, American Society for Testing and Materials, Philadelphia, PA, 1981, pp. 978-987.
11. C. K. Bayne and D. H. Smith, *Int. J. Mass Spectrom. Ion Processes* **59**, 315 (1984).



DISTRIBUTION

1. C. M. Barshick
2. R. M. Coleman
3. D. C. Duckworth
4. E. H. McBay
- 5-9. H. S. McKown
10. S. A. McLuckey
11. R. L. McPherson
12. L. R. Riciputi
13. W. D. Shults
- 14-18. D. H. Smith
19. M. L. Turner
20. R. E. Valiga
21. Patent Section
22. Laboratory Records
23. Laboratory Records-RC
24. Central Research Library
25. Document Reference Section
- 26-27. Office of Scientific and Technical Information, Oak Ridge, TN 37830
28. Assistant Manager of Energy Research and Development, Oak Ridge, TN 37830