on file in
Doc. Library

8/89

TECHNICAL REPORT

The Warm Core Rings Database Routines

Glenn R. Flierl

54-1422 M.I.T.

Cambridge, MA 02139

(617) 253-4692

Preface:

Given the huge number of (expensive) commerical database software packages around, why write our own routines? My answers are:

1) The warm core rings group works on a wide range of computers. Often the data will be prepared on one's own microcomputer and finally archived on the WHOI VAX. It would be very advantageous for data to be not only in a common format but also to be manipulated with twin programs on the various machines. Currently, we have functionally identical database routines on Commodore, NEC (CPM-86), OSI, PDP11-RSX11M and VAX-VMS with Apple software being prepared. I believe that the ability to use the same procedure locally with small data sets on micros or remotely with the full set on the VAX will make these routines much more useful and data exchange much easier.

2) Many of our operations which meld various data sets will involve interpolating one data set (such as CTD or other high resolution measurements) upon the locations of another data set (e.g. bottle depths). This is not a common function.

3) It is now possible and fairly easy to add new capabilities such as in-line computations of (r,θ): by writing our own routines we can modify and extend them in any direction judged useful by the group as a whole.

4) Time was becoming critical and WHOI did not seem to be moving towards a commercial system. The routines described herein are working now (with some bugs, I presume).

1. Introduction

The Warm Core Rings Database (WCRDB) is a "relational" data base, meaning that all data is treated as tables of numbers; for example, figure 1.1 shows the table *CTDKNO93.01.

The header for each column (8 character limit on the VAX) is stored with the table and serves to identify visually the data in that column. The data must be numeric; values such as -999 can be used to identify missing data items.

Data may be stored either in the user's own workspace or in the WCR data archive. Data tables from the archive are referenced by prefixing a * to the filename. (On the VAX, archive files are in the <WCRDB> area; on micros, archives are on the 2nd disk drive.) Data can be read from either source but writing is only possible in the user's own area. Data will be moved into the archive by request.

There are two forms of storage of data on disk, called either a "file" or a "pseudofile". A "file" just contains headers and data in a simple format; the first line has the number of columns (NC), the next NC lines have the table headings, the following groups of NC lines contain successive rows of the table. All data is in ASCII readable format. (See Appendix 1.)

A "pseudofile" appears to the user to be identical in style (for example KNCTD1.TMP figure 1.2), but in reality does not contain the data itself.

* CTD KN093.01

FIGURE 1.1

SIGTH	PRES	TEMP	SAL	O2	DYNHGT
25.8500	3.0000	8.9270	33.3660	7.0000	0.0060
26.1500	6.0000	8.9190	33.7450	6.9300	0.0120
26.2500	9.0000	9.0180	33.8960	6.8700	0.0180
26.2900	11.0000	9.1080	33.9640	6.5900	0.0210
26.3300	13.0000	9.1590	34.0260	6.3800	0.0250
26.4100	15.0000	9.2910	34.1630	6.1900	0.0280
26.5100	17.0000	9.5130	34.3210	6.1700	0.0310
26.5300	19.0000	9.6300	34.3770	5.9900	0.0340
26.5500	23.0000	9.7810	34.4350	5.8600	0.0400
26.5700	27.0000	9.9260	34.4980	5.8200	0.0460
26.5900	29.0000	10.0460	34.5450	5.7800	0.0490
26.6100	31.0000	10.1270	34.5820	5.6800	0.0520
26.6300	34.0000	10.4010	34.6870	5.5700	0.0560
26.6500	39.0000	10.4740	34.7140	5.4600	0.0630
26.6700	43.0000	10.5370	34.7610	5.5300	0.0680
26.6900	45.0000	10.7070	34.8240	5.5000	0.0710
26.7100	50.0000	10.9440	34.9070	5.4300	0.0780
26.7300	56.0000	11.1210	34.9650	5.4000	0.0860
26.7500	60.0000	11.2910	35.0320	5.3200	0.0910
26.7700	70.0000	11.4540	35.0950	5.2100	0.1040
26.7900	83.0000	11.6160	35.1610	5.2900	0.1210
26.8100	93.0000	11.7540	35.2170	5.1800	0.1330
26.8300	102.0000	11.8550	35.2690	5.1000	0.1440
26.8500	109.0000	11.9070	35.3140	4.9700	0.1530
26.8700	114.0000	11.9030	35.3300	4.8400	0.1590
26.8900	121.0000	11.9820	35.3790	4.7400	0.1670
26.9100	128.0000	11.9700	35.4010	4.6100	0.1760
26.9300	134.0000	11.9840	35.4300	4.4300	0.1830
26.9500	142.0000	11.9970	35.4660	4.3300	0.1920
26.9700	150.0000	11.5400	35.3780	4.1700	0.2010
27.0100	155.0000	11.1700	35.3230	4.0400	0.2060
27.0300	160.0000	11.0710	35.3350	3.8600	0.2120
27.0500	168.0000	10.8680	35.3160	3.7500	0.2200
27.0900	178.0000	10.7150	35.3330	3.6100	0.2300
27.1100	187.0000	10.3710	35.2760	3.6300	0.2390
27.1300	194.0000	10.1100	35.2470	3.6700	0.2460
27.1500	202.0000	9.9470	35.2340	3.6500	0.2540
27.1700	208.0000	9.9140	35.2490	3.5700	0.2590
27.1900	215.0000	9.9190	35.2780	3.4900	0.2660
27.2100	222.0000	9.5770	35.2350	3.4900	0.2720
27.2300	227.0000	9.4010	35.2200	3.5300	0.2770
27.2500	237.0000	9.2070	35.2040	3.5600	0.2860
27.2700	250.0000	8.8760	35.1710	3.6000	0.2970
27.2900	264.0000	8.7250	35.1540	3.6200	0.3080
27.3100	277.0000	8.5320	35.1480	3.6600	0.3190
27.3300	287.0000	8.3910	35.1360	3.6100	0.3270
27.3500	296.0000	8.1440	35.1200	3.6300	0.3340
27.3700	304.0000	7.9440	35.1070	3.7600	0.3400
27.3900	313.0000	7.7660	35.1000	3.8400	0.3470
27.4100	322.0000	7.6080	35.0910	3.9100	0.3530
27.4300	330.0000	7.4110	35.0830	4.0000	0.3590
27.4500	337.0000	7.2000	35.0720	4.0700	0.3640
27.4700	344.0000	7.0020	35.0620	4.1900	0.3690
27.4900	359.0000	6.7910	35.0520	4.3400	0.3780
27.5100	377.0000	6.5890	35.0410	4.4200	0.3900
27.5300	395.0000	6.4160	35.0350	4.4900	0.4010

27.5500	415.0000	6.3360	35.0450	4.5500	0.4130
27.5700	429.0000	6.1810	35.0490	4.6600	0.4210
27.5900	443.0000	6.0390	35.0470	4.7500	0.4290
27.6100	466.0000	5.8600	35.0460	4.8900	0.4410
27.6300	487.0000	5.6570	35.0370	5.0000	0.4530
27.6500	507.0000	5.5120	35.0350	5.0900	0.4630
27.6700	538.0000	5.3120	35.0320	5.2400	0.4780
27.6900	580.0000	5.0730	35.0230	5.4300	0.4980
27.7100	644.0000	4.6910	34.9940	5.6700	0.5270
27.7300	755.1000	4.5170	34.9920	5.7800	0.5770
27.7500	926.9000	4.3590	34.9920	5.8500	0.6530
27.7700	1164.0000	4.1120	34.9820	5.9700	0.7570
27.7900	1403.0000	3.8990	34.9780	6.0100	0.8630
27.8100	1668.0000	3.7310	34.9780	6.0700	0.9790
27.8300	1909.9000	3.5500	34.9730	6.0800	1.0850

FIGURE 1.2

SELECTED DATA FROM 1.1

$100 \leq \text{PRESS} \leq 200$

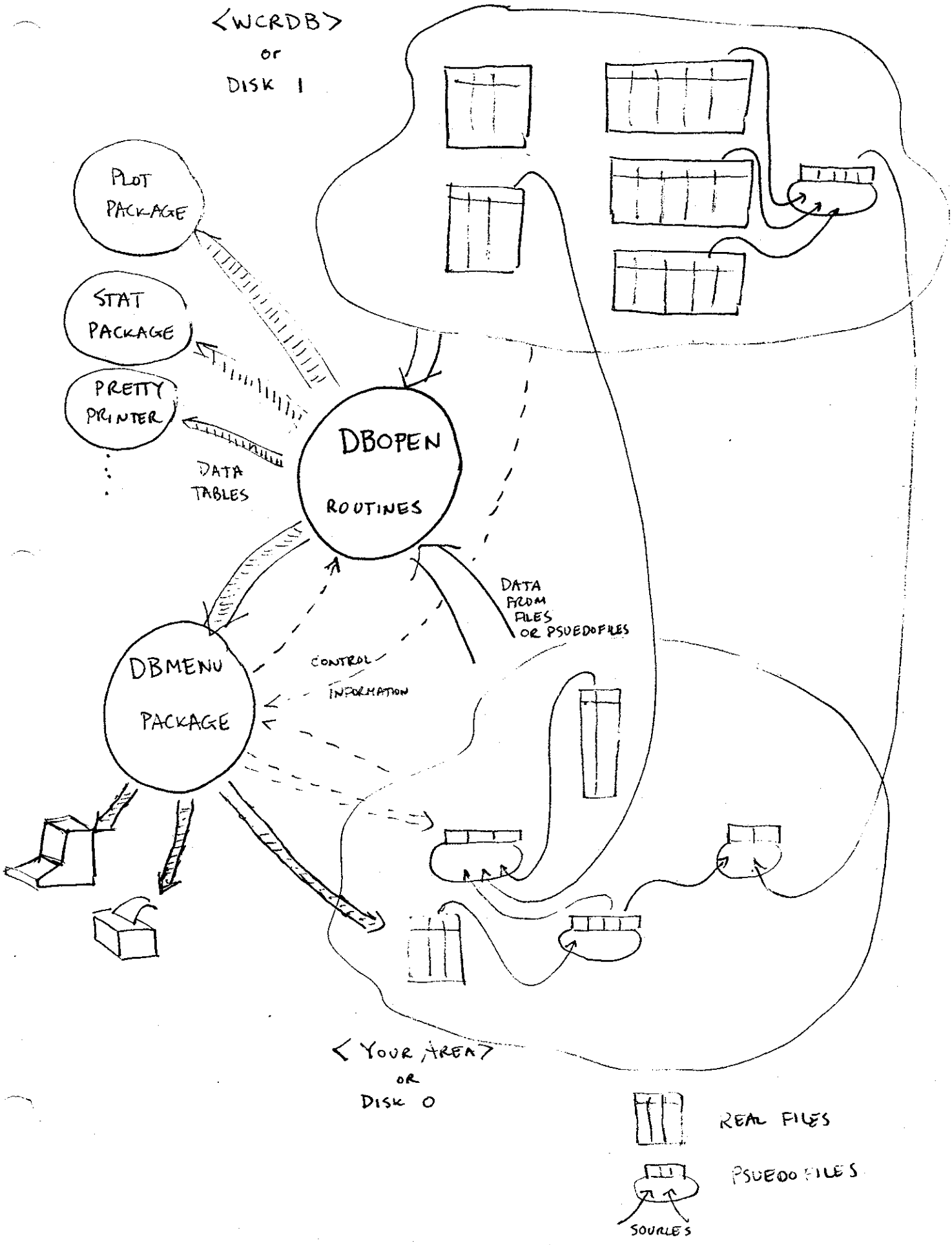
KNCTD1.TMP

PRES	TEMP	SAL	O2
102.0000	11.8550	35.2690	5.1000
109.0000	11.9070	35.3140	4.9700
114.0000	11.9030	35.3300	4.8400
121.0000	11.9820	35.3790	4.7400
128.0000	11.9700	35.4010	4.6100
134.0000	11.9840	35.4300	4.4300
142.0000	11.9970	35.4660	4.3300
150.0000	11.5400	35.3780	4.1700
155.0000	11.1600	35.3230	4.0400
160.0000	11.0710	35.3350	3.8600
168.0000	10.8680	35.3160	3.7500
178.0000	10.7150	35.3330	3.6100
187.0000	10.3710	35.2760	3.6300
194.0000	10.1100	35.2470	3.6700

Instead it contains instructions which tell the database routines how to construct the file from other files. Thus the pseudofile which was printed in figure 1.2 instructs the data base to select all data from *CTDKN093.1 which is in the range 100 to 200 m and display only PRESS, TEMP, SAL, and O2. Compare figures 1.1 and 1.2. A pseudofile exists on disk but only occupies a small region, since it has instructions, not data. (For those interested, pseudofile format is described in Appendix 2.) From the point of view of applications programs (plotting, report-generation, statistical manipulation), a "pseudofile" is as real as a "file" (so long as the applications programs access data through the WCRDB subroutines). Figure 1.3 shows schematically the structure we have just described.

The rest of this document will describe: (1) the types of manipulations and combinations of data sets which are possible with the subroutines and (2) the menu-driven package for setting up pseudofiles, saving them on disk and typing/printing/copying the resulting pseudotable.

FIGURE 1.3


<WCRDB>
or
DISK 1
PLOT PACKAGE
STAT PACKAGE
PRETTY PRINTER
DATA TABLES
DBOPEN ROUTINES
DATA FROM FILES OR PSUEDOFILES
DBMENU PACKAGE
CONTROL INFORMATION
<YOUR AREA>
or
DISK 0
REAL FILES
PSUEDO FILES
SOURCES

2. Operations on files and pseudofiles

Operations will be discussed below in terms of what happens to data from a "source" or "sources". These may be either files on disk, pseudofiles on disk or the results of preceding operations. There is no distinction among these. For example one can take the data which appears to be in the pseudofile KNCTD1.TMP and combine it with other data, e.g. nutrient data, select out particular pressure levels and finally keep only temperature and nitrate values. Conceptually, each operation produces a new pseudofile which can then be used as source for succeeding operations; in actuality the final file is constructed one row at a time. So remember: a "source" could be any of these three things interchangeably. In the examples below we will show only a single step, giving both the source and the results of the operation using made-up small-sized tables.

A) Selection by rows.

The result table will contain only the rows from the source which satisfy some (compound) criterion. For example, Figure 2.1 shows the source file and the result of selecting only rows with 3 < DEPTH and DEPTH < 7. Each part of the criteria is a comparison between the value in any column of the table against any number, with the usual set of comparison operators. These parts can then be combined using AND or OR to yield the full selection criterion. For example one could choose EVNO = 1 OR EVNO = 3; 38 < LAT AND LAT < 39 AND LONG > -69 AND LONG < -68; or TEMP > 10 AND SAL < 35; etc.

A second row selection method involves interpolation based on one particular column. All other columns are interpolated linearly based on the rows which fall on either side of the desired value. Figure 2.2 shows this

FIGURE 2.1

SOURCE

EVNO	DEPTH	TEMP	SAL
1	2	20.5	33.31
1	5	19.8	33.34
1	10	18.2	33.45
2	3	21.2	33.55
2	9	20.5	33.62
3	1	22.2	33.4
3	4	27.6	33.44
3	11	25.5	33.46

RESULT

$3 < \text{DEPTH} < 7$

EVNO	DEPTH	TEMP	SAL
1	5	19.8	33.34
3	4	27.6	33.44

FIGURE 2.2

SOURCE

EVNO	DEPTH	TEMP	SAL
1	2	20.5	33.31
1	5	19.8	33.34
1	10	18.2	33.45
2	3	21.2	33.55
2	9	20.5	33.62
3	1	22.2	33.4
3	4	27.6	33.44
3	11	25.5	33.46

RESULT

DEPTH INTERP. TO 5

EVNO	DEPTH	TEMP	SAL
1	5	19.8	33.34
2	5	20.9666667	33.5733333
3	5	27.3	33.4428572

process with interpolation of the same table to DEPTH of 5 meters. Because we often group stations together, as in the source file shown, interpolation can be done only on ascending or only on descending passes or at any crossing.+ (+ means easily but not currently implemented - let me know if you need to do it.) Figure 2.3 sketches the different results for selection, ascending interpolation, descending interpolation and any crossing interpolation for the value of 5.

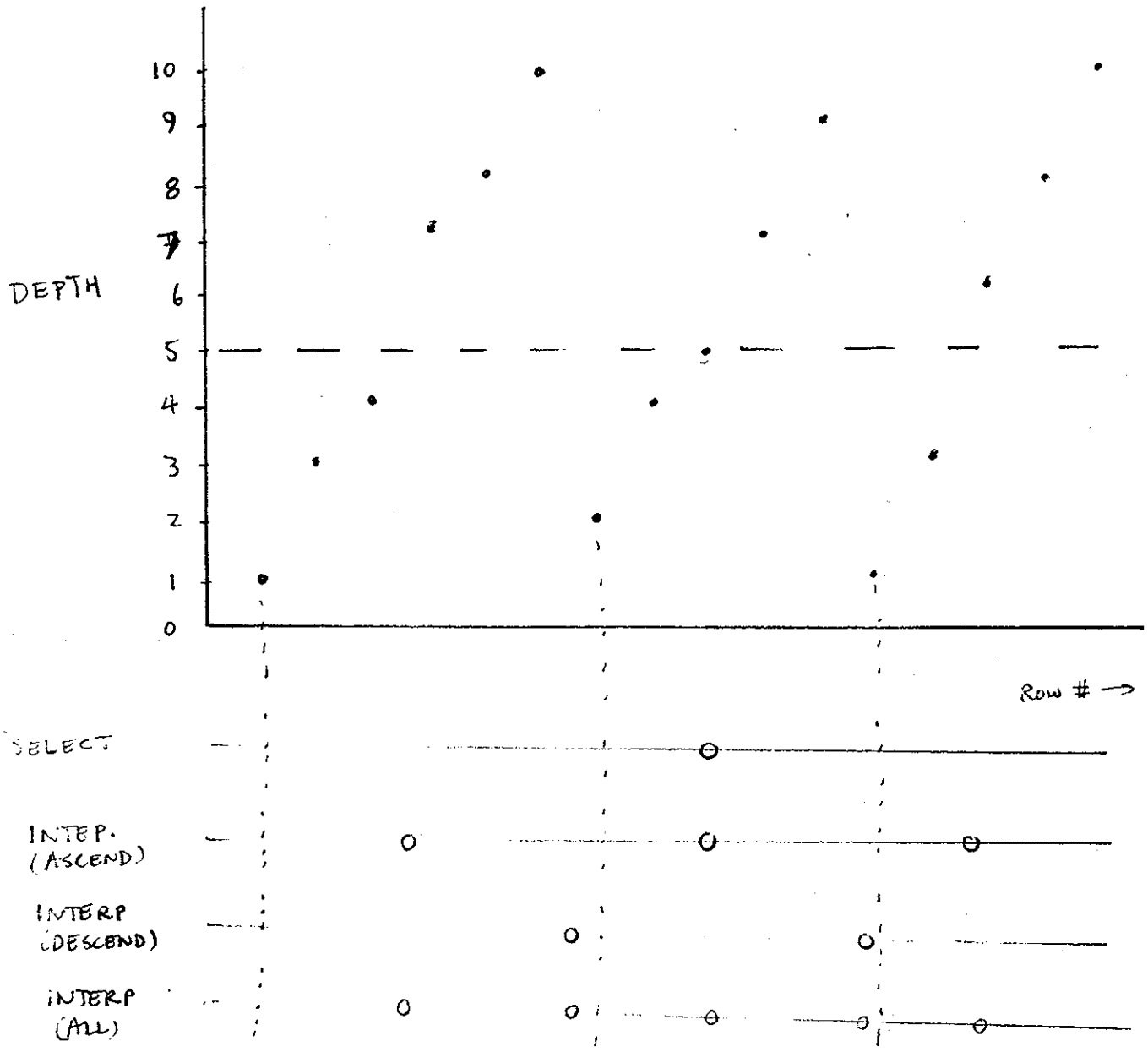
B) Selection of columns.

Where selecting by rows picks out only particular horizontal swaths from the table, selection by columns makes vertical swaths. Figure 2.4 shows an example, selecting SAL and TEMP from the source. It is important to note that the order of selection determines the ordering in the result table so that columns can be easily reordered by this means if some program expects data in a special column order.

C) Joining together several tables.

For our program, the ability to meld together various data sets is absolutely critical. Conceptually, one uses common information to put the two sources together. The simplest case is in figure 2.5 where the EVNO and DEPTH commonality has been used to join together the T-S and NO3 data. For various data from the same bottles, this is the natural way of combining information, using the agreed-upon depth convention. Again a compound criterion EVNO(1) = EVNO(2) and DEPTH(1) = DEPTH(2) has been used to choose the matching. Both fields are in ascending order. (The order is critical in deciding, if one table is missing a row, which of the two sources is the one with the missing row.) If desired we could implement a join assuming

FIGURE 2.3



POINTS INCLUDED IN FINAL TABLE UNDER VARIOUS SELECT BY ROW PROCESSES

FIG. 2.4

EVNO	DEPTH	TEMP	SAL
1	2	20.5	33.31
1	5	19.8	33.34
1	10	18.2	33.45
2	3	21.2	33.55
2	9	20.5	33.62
3	1	22.2	33.4
3	4	27.6	33.44
3	11	25.5	33.46

SOURCE

RESULT

KEEP SAL, TEMP

SAL	TEMP
33.31	20.5
33.34	19.8
33.45	18.2
33.55	21.2
33.62	20.5
33.4	22.2
33.44	27.6
33.46	25.5

FIG. 2.5

EVNO	DEPTH	NO3
1	2	1.5
1	5	1.6
1	10	2.7
2	3	1.3
2	5	1.4
2	9	2.1
3	1	1
3	4	1.7
3	11	2.5

SOURCE 1

EVNO	DEPTH	TEMP	SAL
1	2	20.5	33.31
1	5	19.8	33.34
1	10	18.2	33.45
2	3	21.2	33.55
2	9	20.5	33.62
3	1	22.2	33.4
3	4	27.6	33.44
3	11	25.5	33.46

SOURCE 2

TEMP	SAL	NO3	DEPTH
20.5	33.31	1.5	2
19.8	33.34	1.6	5
18.2	33.45	2.7	10
21.2	33.55	1.3	3
20.5	33.62	2.1	9
22.2	33.4	1	1
27.6	33.44	1.7	4
25.5	33.46	2.5	11

RESULT

JOIN BY

= EVNOS & = DEPTHS

KEEP

TEMP, SAL, NO3, DEPTH

that the second table could be missing data lines and that these should be filled with -999.+ As the final part of the join process, there is a column selection step to pick from the two sources. The process is (hopefully) exactly what you would do if someone handed you the two tables and said, "Make me a new table containing TEMP SAL NO3 and DEPTH".

A second type of join operation involves interpolating one data set (e.g. from some high-resolution device like the CTD) to the levels of another (e.g. bottles). Figure 2.6 shows an example of this process. The first source file serves to specify the desired levels to be obtained by interpolation from the second file. The issues about direction of interpolation apply here also; the program is set up to handle situations where the first file is (say) a set of stations of bottle depths at increasing depths within a station while the second contains the data from the same stations with increasing CTD depths. Flowcharts in Appendix 3 describe selection procedures fully. All you need (I hope) to know is that cases like those in figure 2.7 can all be handled properly; misordering of stations or use of only a few points could be troublesome.

The third type of join operation is designed for selecting data from a list of stations (or event numbers) or filling in a column of information such as latitude/longitude from a source like the event logs. For the previous two cases, at least one new line of data is read from the first source for each line of the result. In contrast, for the fill-in join, new lines are read from the first source only when the match no longer succeeds. The result, then, includes all lines from the second source which match with one line from the first. For example, figure 2.8 shows the process of selecting data from particular stations. The first file contains a list of desired stations which is matched against the station data to yield the result shown. As a second

FIG. 2.6

DEPTH	NO3
2	1.5
5	1.6
10	2.7

SOURCE 1

PRESS	TEMP	SAL
1	20.7	33.3
3	20.35	33.33
5	19.82	33.33
7	19.12	33.38
9	18.58	33.43
11	18.1	33.46

SOURCE 2

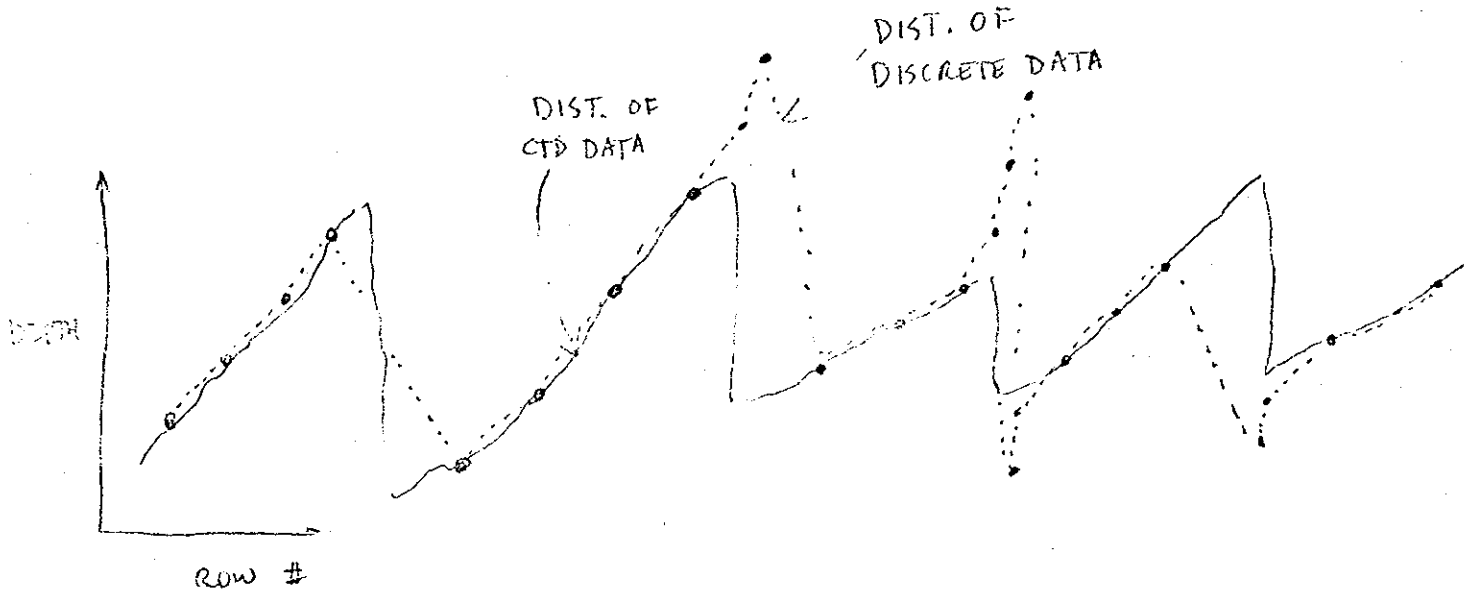
DEPTH	NO3	TEMP	SAL
2	1.5	20.525	33.315
5	1.6	19.82	33.33
10	2.7	18.34	33.445

INTERP. JOIN

DEPTH = PRESS

KEEP DEPTH, NO3, TEMP, SAL

FIG. 2.7



POINTS AT WHICH ROWS WILL OCCUR IN JOINED

FIGURE 2.8

SEL-STN
2
4
5

SOURCE 1

EVNO	DEPTH	TEMP	SAL
1	2	20.5	33.31
1	5	19.8	33.34
1	10	18.2	33.45
2	3	21.2	33.55
2	9	20.5	33.62
3	1	22.2	33.4
3	4	27.6	33.44
3	11	25.5	33.46
4	2	24.8	34.25
4	6	23.2	34.27
4	12	22.1	34.29
5	1	25.5	34.82
5	6	25	34.84
5	9	24.8	34.9
5	15	23.6	34.88

SOURCE 2

RESULT

SEL-STN	DEPTH	TEMP	SAL
2	3	21.2	33.55
2	9	20.5	33.62
4	2	24.8	34.25
4	6	23.2	34.27
4	12	22.1	34.29
5	1	25.5	34.82
5	6	25	34.84
5	9	24.8	34.9
5	15	23.6	34.88

FILL JOIN ASCENDING

SEL-STN = EVNO

example, the positions from an event log (figure 2.9) are appended to all data from the station using the "fill-in join" procedure.

D) Arithmetic operations on columns

Certain operations for combining or modifying the information in selected columns can be done with the data base subroutines. These routines also permit extending the width of the table. As an example, figure 2.10 shows creating a new column BIOMASS which is the sum of two other columns, ZOO BIO and PHYTO BIO. Other functions currently implemented are computing the product of two columns and a linear scaling of the data in one column. The latter can also be used to construct a constant column -- simply use a slope of zero and an intercept of the desired value (e.g. figure 3.3). This is one area where additional capabilities could be important; for example (r,θ) calculations and hydrographic calculations could be added easily.

E) Chaining files head-to-tail

The final type of operation is simply adding one or more sources onto the end of the first source. This allows constructing pseudofiles which represent a whole set of stations; for example one could construct a TS relationship for one of the radial sections. Figure 2.11 shows an example of this operation.

FIGURE 2.9

EVNO	LAT	LONG
1	38.42	-69.73
2	38	-68.35
3	37.25	-70.11

SOURCE 1

EVNO	DEPTH	NO3
1	2	1.5
1	5	1.6
1	10	2.7
2	3	1.3
2	5	1.4
2	9	2.1
3	1	1
3	4	1.7
3	11	2.5

SOURCE 2

RESULT

FILL JOIN

EVNO = EVNO

EVNO	LAT	LONG	DEPTH	NO3
1	38.42	-69.73	2	1.5
1	38.42	-69.73	5	1.6
1	38.42	-69.73	10	2.7
2	38	-68.35	3	1.3
2	38	-68.35	5	1.4
2	38	-68.35	9	2.1
3	37.25	-70.11	1	1
3	37.25	-70.11	4	1.7
3	37.25	-70.11	11	2.5

FIG. 2.10

SOURCE

MOC#	NET#	ZOO BIOM	PHYTO BIOM
110	1	12.5	105.2
110	2	11.3	103.1
110	3	6.5	121.1
110	4	3.1	69.2
110	5	.52	2.2

RESULT

BIOMASS = ZOO + PHYTO

MOC#	NET#	ZOO BIOM	PHYTO BIOM	BIOMASS
110	1	12.5	105.2	117.7
110	2	11.3	103.1	114.4
110	3	6.5	121.1	127.6
110	4	3.1	69.2	72.3
110	5	.52	2.2	2.72

FIG. 2.11

SOURCE 1

PRESS	TEMP	SAL
1	20.7	33.3
3	20.35	33.33
5	19.82	33.33
7	19.12	33.38
9	18.58	33.43
11	18.1	33.46

SOURCE 2

PRESS	TEMP	SAL
2	21.6	33.53
3	21.23	33.54
5	20.89	33.57
7	20.62	33.6
9	20.51	33.62
11	20.48	33.65

RESULT

CHAIN 1 TO 2

PRESS	TEMP	SAL
1	20.7	33.3
3	20.35	33.33
5	19.82	33.33
7	19.12	33.38
9	18.58	33.43
11	18.1	33.46
2	21.6	33.53
3	21.23	33.54
5	20.89	33.57
7	20.62	33.6
9	20.51	33.62
11	20.48	33.65

3) Use of the menu-driven routines for examining files and pseudofiles and constructing pseudofiles.

While one can construct pseudofiles with regular text editors, it is not easy and we have written a menu-driven package to help in this process. While this is the simplest way to learn the system, it is not necessarily the most efficient; perhaps a command-driven package will be added later, which still interfaces to the same database subroutines. Refer again to figure 1.3 where the DBMENU package occupies the position shown, allowing the user to display, print, or write to disk the contents of files or pseudofiles as well as permitting the construction of pseudofiles. Since this will be the primary interface to the database, at least for now, we shall make some comments upon its use (the menus are hopefully fairly easy to understand).

The program is loaded with the usual system procedure. For the VAX, this simply requires typing DBM in response to the $ prompt. This translates to RUN <WCRDB.PGM>DBMENU according to the definition in your LOGIN.COM file. The Microsoft basic version should be invoked with the /F:12 (on the NEC at least) to give maximum file space.

We have illustrated the menus which appear in figure 3.1, which is a record of the process used to construct the pseudofile shown in figure 2.1.

A) Main menu.

FIG. 3.1 (A)

```
OK
RUN "DBMENU"

 0              RESTART                                          MAIN MENU
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE                        A
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                          1

STEP  1
 0              FINISH CONSTRUCTING PSEUDOFILE                   PSEUDOFILE
 1              SELECT ROWS                                      CONSTRUCTION
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)                            MENU
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)                             B
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                         1               SELECT ROWS
SELECT SOURCE FILE
 0              ANOTHER FILE FROM DISK
ENTER CHOICE ( 0 - 0 )?                          0               FROM
FILENAME?                                        TS              "TS"
VARIABLE FOR SELECTION
 1              EVNO
 2              DEPTH
 3              TEMP
 4              SAL
ENTER CHOICE ( 1 - 4 )?                           2              DEPTH
COMPARISON OPERATOR
 1              =
 2              >
 3              <
 4              >=
 5              <=
 6              <>
ENTER CHOICE ( 1 - 6 )?                           2              >
COMPARISON VALUE?                                 3.0            3.0
ADD MORE CONDITIONS
 0              END OF TESTS
 1              AND
 2              OR
ENTER CHOICE ( 0 - 2 )?                           1              &

VARIABLE FOR SELECTION
 1              EVNO
 2              DEPTH
 3              TEMP
 4              SAL                                             DEPTH
ENTER CHOICE ( 1 - 4 )?                           2
```

3.1 (B)

```
COMPARISON OPERATOR
 1              =
 2              >
 3              <
 4              >=
 5              <=
 6              <>
ENTER CHOICE ( 1 - 6 )?                          3
COMPARISON VALUE?                                7.0
ADD MORE CONDITIONS
 0              END OF TESTS
 1              AND
 2              OR
ENTER CHOICE ( 0 - 2 )?                          0

STEP  3
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                         0

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                          3
EVNO            DEPTH          TEMP           SAL
 1               5              19.8           33.34
 3               4              27.6           33.44

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?

OK
```

< 7.0

END OF CONSTRUCTION

SHOW RESULTS

The options here (3.1A) are fairly straightforward. Option 1 (CONSTRUCT PSEUDOFILE) leads into a set of menus for pseudofile construction; upon completion of these procedures, the constructed pseudofile becomes the "current" file which can then be typed, printed or copied. The sub-menus for pseudofiles will be discussed below.

Options 2 (TYPE FROM FILE OR PSEUDOFILE), 3 (PRINT), and 4 (COPY TO DISK) provide a listing of the data in a file or pseudofile. The listing may be directed in tabular form to either your terminal or the printer, or may be written in "real file format" (Appendix 1) to disk. There are several reasons for doing this: constructing a pseudofile can be time-consuming; if one plans to use it frequently, it may be more efficient to write it to disk and use these records as the source for further operations. One is, of course, trading space for time. Secondly, the write to disk option provides a complete real file which can then be transferred to another machine. (While pseudofiles can be copied directly also, they are not useful unless all the required source files are present on the second machine too.)

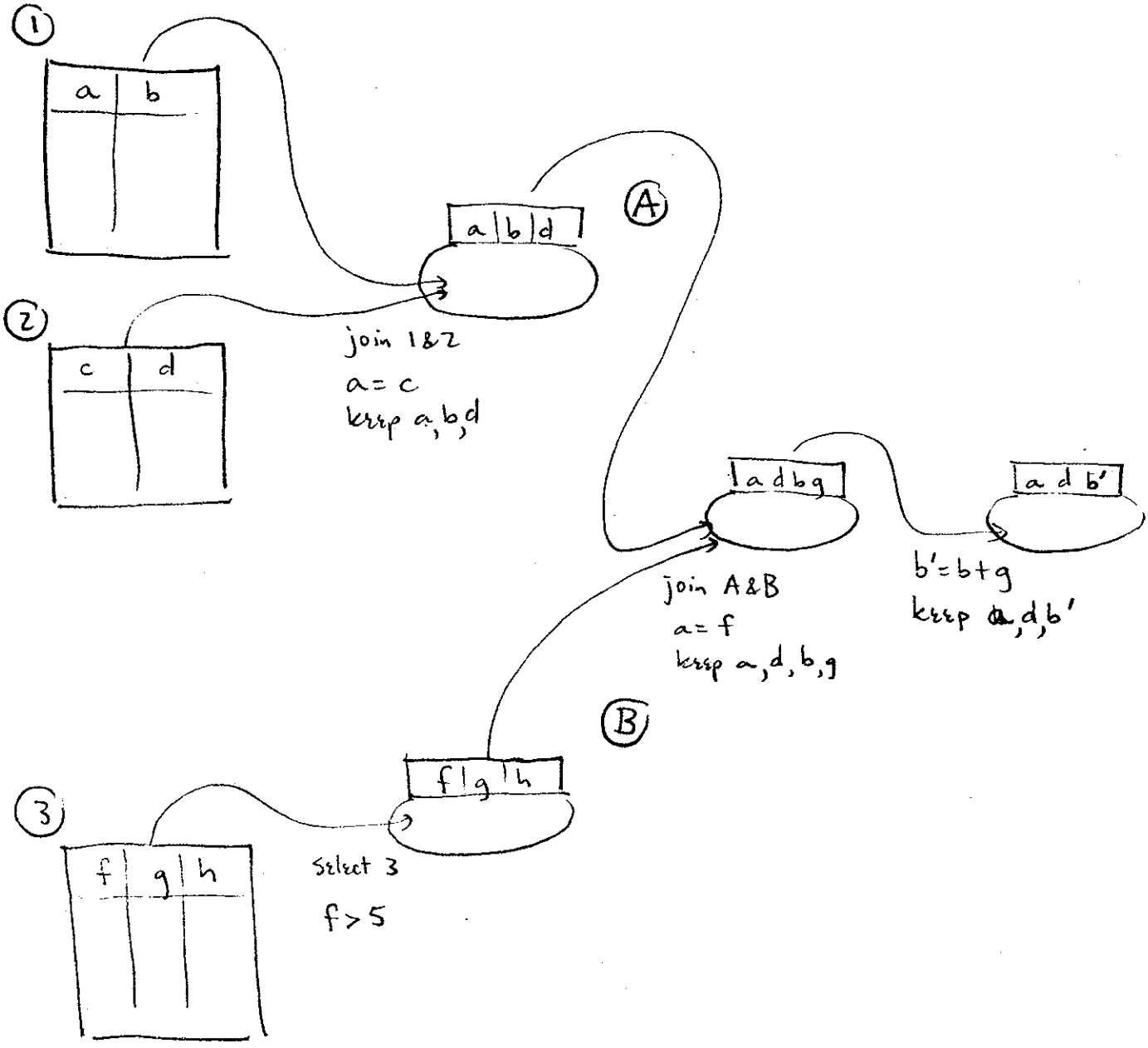
Each invocation of the option 2-4 will output the contents of the "current" file or pseudofile. If no file is current-- at the first pass through the program or after the RESTART option (choice 0)--, the name of the file to be used will be requested.

Option 5 is used after the construction of a pseudofile to save it for future use. In principle, this can be done after construction, after typing,

printing or copying or even after bringing one in from disk. There are several reasons for saving a pseudofile. First, one may wish to use the file a few times but not take the storage space required if one converts it to a real file. Secondly, one may wish to break a complicated series of operations into a simpler set of steps and test the intermediate stages by typing them out. For example, the desired final result might be as sketched in figure 3.2, the sum of columns from a pseudofile which is two other pseudofiles joined together. These two sources are in turn constructed from others. To minimize the chance of error, one could construct the description of pseudofile A, save it to disk and check it by typing. Then one would construct the description of pseudofile B, save and test it. Finally, one would set up the step of merging these two files , doing the sums and selecting the desired columns. The source files for this last stage would be the pseudofiles produced in the earlier steps (see figure 3.2 again). Finally, because of their brevity, pseudofiles may prove an efficient method for exchanging ideas of important relationships among users of the common database.

B) The ways in which the results are constructed from the sources for the various options in the second menu (figure 3.1B)-- the one for constructing pseudofiles-- have been described. One should note that the list is preceded by a line giving the identification number for this step of the process. This is an important number since the results of this step may be used as the source for some future step; in a listing of possible source files, the identification number is used to distinguish these intermediate results.

FIG. 3.2


1
a b
2
c d
a b d
A
join 1&2
a = c
keep a, b, d
a d b g
a d b'
b' = b + g
keep a, d, b'
join A&B
a = f
keep a, d, b, g
B
f g h
3
f g h
Select 3
f > 5

After selecting the operation, one then is prompted for the source file(s) required. In the first example (3.1), there is only one possible choice, 0, which selects a file from disk. The file may name either a pseudofile or a real file; the menu program does not care. Recall that data from the archive are accessed by prefixing a * to the name.

The menus that follow are pretty self-explanatory, requesting the information required to specify the procedure completely. When a comparison value is needed, any floating point number may be entered. For other options at the second menu level, a few comments may prove helpful:

1) Menus offering choice of VARIABLE(S) TO KEEP will allow multiple choices without repeating the list; the order of choices reflects the ordering of columns in the result table. The list of selections is terminated by using the 0 option.

2) In the interpolating, filling join, and chaining procedures, the order of specification of source files is important. In the interpolating join case, the second source is interpolated to the values of the first source. In the filled join, the first source is "filled out" to match the second.

3) For the arithmetical operations, the user is first prompted for the column which will receive the result (which may be either a column in the source to be altered or a new column to be created), then for the operation

to be done, and finally for the operands.

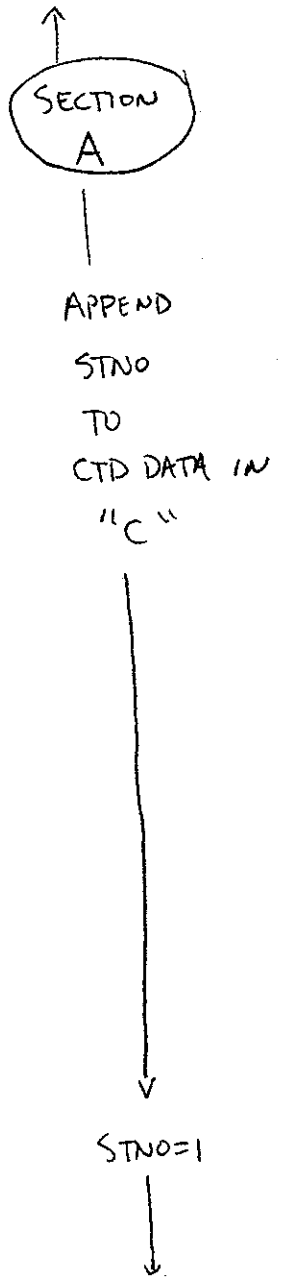
As a second example, consider the procedure in figure 3.3 which shows appending of station numbers to two files and then chaining them together (a miniature version of the procedure for creating the .ALL files). The modifications which add the station numbers (sections A and B in figure 3.3) must be done before chaining the two files-- the processes proceeds from "bottom up." After appending the station number to the first file C (the set of steps marked A on the figure), the same operation is done for the second file D (set of steps marked B) with the source being again "another file from disk." Finally these two results are chained together; for this operation, the sources are the results from step 1 and from step 3.

It is possible to construct a pseudofile which can be used to perform an identical series of operations on many different source files. This is done by responding with a ? to the name of the source file from disk at the time of construction of the pseudofile. The computer will then request the number of columns in the files to be used later and their names (figure 3.4). The pseudofile should be saved before it is used; otherwise the ? wildcard will be replaced by something else in the saved version (probably). Upon use of this pseudofile, the computer will prompt at execution time for the filename to substitute for the ? wildcard.

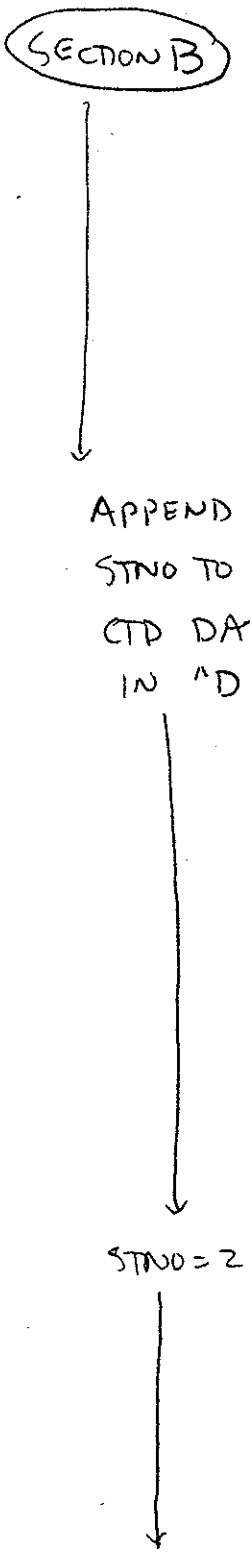
Finally, we show some complicated examples (did you know that the previous ones were simple?): given a list of event numbers (EVENT.DAT) as in figure 3.5A, construct a table of NO3 and O2 at 75m depth (3.5B). This

Fig.(3.3)(A)

```
OK
RUN "DBMENU"

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                          1

STEP  1
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                         11
SELECT SOURCE FILE
 0              ANOTHER FILE FROM DISK
ENTER CHOICE ( 0 - 0 )?                           0
FILENAME?                                        C
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                           1
NEW VARIABLE NAME?                               STNO
FUNCTION TO APPLY
 1              SUM OF TWO COLUMNS
 2              PROD OF TWO COLUMNS
 3              LINEAR TRANSFORMATION TO ONE COLUMN
ENTER CHOICE ( 1 - 3 )?                           3
SLOPE?                                            0
ARGUMENT
 1              PRES
 2              TEMP
 3              SAL
 4              STNO
ENTER CHOICE ( 1 - 4 )?                           1
INTERCEPT?                                        1
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                           0
```



```
STEP  3
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                    11
SELECT SOURCE FILE
 0              ANOTHER FILE FROM DISK
 1              (RESULTS FROM STEP LISTED)
ENTER CHOICE ( 0 - 2 )?                     0
FILENAME?                                   D
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                     1
NEW VARIABLE NAME?                          STNO
FUNCTION TO APPLY
 1              SUM OF TWO COLUMNS
 2              PROD OF TWO COLUMNS
 3              LINEAR TRANSFORMATION TO ONE COLUMN
ENTER CHOICE ( 1 - 3 )?                     3
SLOPE?                                      0
ARGUMENT
 1              PRESS
 2              TEMP
 3              SAL
 4              STNO
ENTER CHOICE ( 1 - 4 )?                     1
INTERCEPT?                                  2
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                     0

STEP  5
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
```



(3.3)(c)

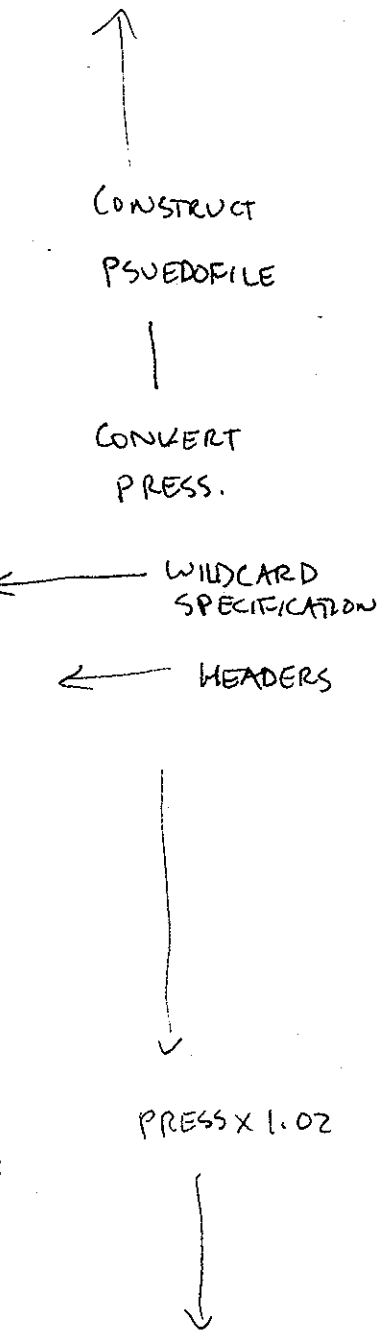
```
 12            CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                                    12
SELECT SOURCE FILE
 0             ANOTHER FILE FROM DISK
 1             (RESULTS FROM STEP LISTED)
 3             (RESULTS FROM STEP LISTED)
ENTER CHOICE ( 0 - 4 )?                                      1
CHAIN TO
SELECT SOURCE FILE
 0             ANOTHER FILE FROM DISK
 3             (RESULTS FROM STEP LISTED)
ENTER CHOICE ( 0 - 4 )?                                      3
 0             END OF CHAIN
 1             CHAIN FURTHER
ENTER CHOICE ( 0 - 1 )?                                      0

STEP  6
 0             FINISH CONSTRUCTING PSEUDOFILE
 1             SELECT ROWS
 2             INTERP ROWS (ASCEND)
 3             INTERP ROWS (DESCEND)
 4             SELECT COLUMNS
 5             JOIN TWO FILES (ASCEND)
 6             JOIN TWO FILES (DESCEND)
 7             INTERP FILES (ASCEND)
 8             INTERP FILES (DESCEND)
 9             FILL OUT JOIN (ASCEND)
 10            FILL OUT JOIN (DESCEND)
 11            ARITHMETIC CONVERSIONS
 12            CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                                     0

 0             RESTART
 1             CONSTRUCT PSEUDOFILE
 2             TYPE FROM FILE OR PSEUDOFILE
 3             PRINT
 4             COPY TO DISK
 5             SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                      3
PRES           TEMP           SAL            STNO
 1              20.7           33.3           1
 3              20.35          33.33          1
 5              19.82          33.33          1
 7              19.12          33.38          1
 9              18.58          33.43          1
 11             18.1           33.46          1
 2              21.6           33.53          2
 3              21.23          33.54          2
 5              20.89          33.57          2
 7              20.62          33.6           2
 9              20.51          33.62          2
 11             20.48          33.65          2

 0             RESTART
 1             CONSTRUCT PSEUDOFILE
 2             TYPE FROM FILE OR PSEUDOFILE
 3             PRINT
 4             COPY TO DISK
```

CHAIN THE LAST TWO RESULTS TOGETHER

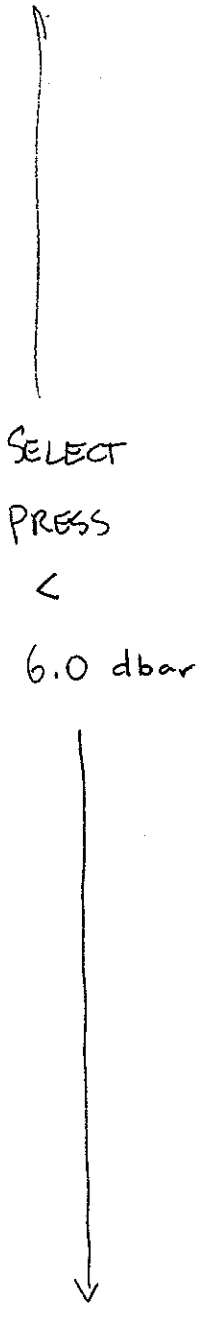
PRINT OUT RESULTS

FIG. 3.4 (A)

```
OK
RUN "DBMENU"

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                  1

STEP  1
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                                 11
SELECT SOURCE FILE
 0              ANOTHER FILE FROM DISK
ENTER CHOICE ( 0 - 0 )?                                   0
FILENAME?                                                 ?
NUMBER OF COLUMNS?                                        3
NAME OF COLUMN 1 ?                                       PRESS
NAME OF COLUMN 2 ?                                       TEMP
NAME OF COLUMN 3 ?                                       SAL
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                                   2
 1              PRESS
 2              TEMP
 3              SAL
ENTER CHOICE ( 1 - 3 )?                                   1
FUNCTION TO APPLY
 1              SUM OF TWO COLUMNS
 2              PROD OF TWO COLUMNS
 3              LINEAR TRANSFORMATION TO ONE COLUMN
ENTER CHOICE ( 1 - 3 )?                                   3
SLOPE?                                                    1.02
ARGUMENT
 1              PRESS
 2              TEMP
 3              SAL
ENTER CHOICE ( 1 - 3 )?                                   1
INTERCEPT?                                                0
```



```
 0              END OF CONVERSIONS
 1              CREATE NEW COLUMN
 2              CHANGE OLD COLUMN
ENTER CHOICE ( 0 - 2 )?                                  0

STEP  3
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                                 1
SELECT SOURCE FILE
 0              ANOTHER FILE FROM DISK
 1              (RESULTS FROM STEP LISTED)
ENTER CHOICE ( 0 - 2 )?                                  1
VARIABLE FOR SELECTION
 1              PRESS
 2              TEMP
 3              SAL
ENTER CHOICE ( 1 - 3 )?                                  1
COMPARISON OPERATOR
 1              =
 2              >
 3              <
 4              >=
 5              <=
 6              <>
ENTER CHOICE ( 1 - 6 )?                                  2
COMPARISON VALUE?                                        6.0
ADD MORE CONDITIONS
 0              END OF TESTS
 1              AND
 2              OR
ENTER CHOICE ( 0 - 2 )?                                   0

STEP  4
 0              FINISH CONSTRUCTING PSEUDOFILE
 1              SELECT ROWS
 2              INTERP ROWS (ASCEND)
 3              INTERP ROWS (DESCEND)
 4              SELECT COLUMNS
 5              JOIN TWO FILES (ASCEND)
 6              JOIN TWO FILES (DESCEND)
 7              INTERP FILES (ASCEND)
 8              INTERP FILES (DESCEND)
 9              FILL OUT JOIN (ASCEND)
 10             FILL OUT JOIN (DESCEND)
 11             ARITHMETIC CONVERSIONS
```



3.4 (c)

```
 12             CHAIN FILES
ENTER CHOICE ( 0 - 12 )?                                    0

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                     5
FILENAME?                                                   B

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                     3
PRESS          TEMP          SAL
FILENAME FOR STEP 3 ?                                       C
 7.14           19.12         33.38
 9.18           18.58         33.43
 11.22          18.1          33.46

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                     0

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?                                     3
FILENAME?                                                   B
PRESS          TEMP          SAL
FILENAME FOR STEP 3 ?                                       D
 7.14           20.62         33.6
 9.18           20.51         33.62
 11.22          20.48         33.65

 0              RESTART
 1              CONSTRUCT PSEUDOFILE
 2              TYPE FROM FILE OR PSEUDOFILE
 3              PRINT
 4              COPY TO DISK
 5              SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE ( 0 - 5 )?

OK
```

SAVE AS FILE B

FILENAME TO SUBST. FOR ?

SECOND RUN

SUBSTITUTE FOR ?

involves data from several sources, since the O2 data is with the CTD stuff while the NO3 data is in the NUT area. There are many ways to accomplish this task, as sketched in figure 3.6; since we presume that it is more efficient to select stations first and then interpolate, we follow the second procedure. (See note below.) Figure 3.7 gives listings of the process for this project.

First, we use the event log and EVENT.DAT to construct a file of event numbers and corresponding CTD numbers, saved as TEMP. Note that some events, such as 420.14, are listed in the log as having two CTD numbers-- 5.6-- (maybe one shallow and one deep?). If this appeared in the TEMP table, we would lose the data from this station, since 5.6 does not appear in the CTD listings. Therefore, we would need to edit TEMP1 to choose 5 or 6, whichever were desired.

Secondly, we shall fill-join EVENT.DAT against the collected nutrients file NUTKN093.ALL and interpolate the result to 75m. This will be saved as TEMP1. A similar procedure is carried out on TEMP versus CTDKN093.ALL to produce TEMP2.

The last step is to join TEMP1 and TEMP2 to produce the table shown in 3.5B. This table can be printed or copied for downloading; the temporary files can be deleted.

If the problem were to be altered to creating a table with all NO3 versus

FIG. 3.5A

```
  FILENAME
EVENT.DAT
EVENT#
 420.0200
 420.1000
 421.0200
```

FIG 3.5B

EVENT#	DEPTH	NO3	O2
420.1000	75.0000	8.9193	5.4333
421.0200	75.0000	10.8532	5.2727

420.02 – no common data available at 75 m

FIGURE 3.6

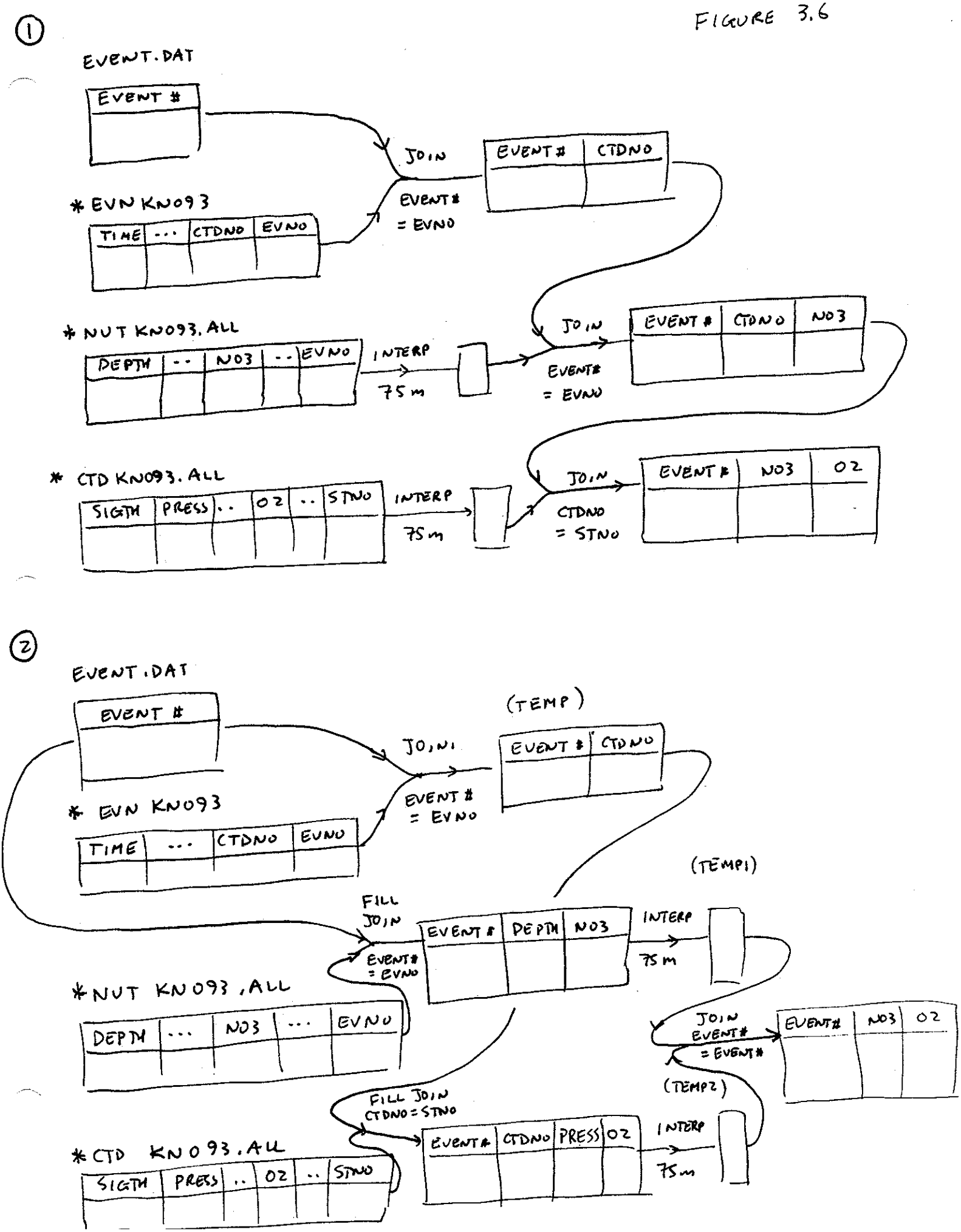

①
EVENT.DAT
EVENT #
JOIN
EVENT# = EVNO
EVENT# | CTDNO
* EVN KN093
TIME | ... | CTDNO | EVNO
* NUT KN093.ALL
DEPTH | .. | N03 | .. | EVNO
INTERP
75 m
JOIN
EVENT# = EVNO
EVENT # | CTDNO | N03
* CTD KN093.ALL
SIGTH | PRESS | .. | O2 | .. | STNO
INTERP
75 m
JOIN
CTDNO = STNO
EVENT # | N03 | O2
②
EVENT.DAT
EVENT #
JOIN
EVENT # = EVNO
(TEMP)
EVENT # | CTDNO
* EVN KN093
TIME | ... | CTDNO | EVNO
(TEMP1)
FILL JOIN
EVENT# = EVNO
EVENT # | DEPTH | N03
INTERP
75 m
*NUT KN093.ALL
DEPTH | ... | N03 | ... | EVNO
JOIN EVENT# = EVENT#
EVENT# | N03 | O2
(TEMP2)
FILL JOIN CTDNO = STNO
EVENT# | CTDNO | PRESS | O2
INTERP
75 m
*CTD KN093.ALL
SIGTH | PRES | .. | O2 | .. | STNO

FIGURE 3.7 (2

```
RUN DBMENU

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

TYPE OUT EVENT.DAT

```
                                                    2

  FILENAME

                                                EVENT.DAT

EVENT#
 420.0200
 420.1000
 421.0200

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

MERGE EVENT.DAT & EVENT LOG TO FIND EVENT# — CTDNO CORRESPONDENCES

```
                                                    1

STEP   1
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)
                                                    5

  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0

  FILENAME

                                                EVENT.DAT

  MERGE WITH
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0

  FILENAME
                                                *EVNKN093
```

```
  VARIABLE FOR KEY
 1 EVENT#
ENTER CHOICE (  1-  1)

COMPARISON OPERATOR
1 =
2 =+-5%
ENTER CHOICE (  1-  2)

  VARIABLE TO COMPARE
 1 TIME
 2 DAY
 3 LAT
 4 LATMIN
 5 LONG
 6 LONGMIN
 7 TYPE
 8 CTDNO
 9 EVNO
ENTER CHOICE (  1-  9)

ADD MORE CONDITIONS
0 END OF CONDITIONS
1 AND
ENTER CHOICE (  0-  1)

  VARIABLE(S) TO KEEP
 1 EVENT#
-1 TIME
-2 DAY
-3 LAT
-4 LATMIN
-5 LONG
-6 LONGMIN
-7 TYPE
-8 CTDNO
-9 EVNO
  0 END OF KEEP CONDITIONS
ENTER CHOICE ( -9-  1)

ENTER CHOICE ( -9-  1)

ENTER CHOICE ( -9-  1)


STEP   4
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
```

1

1

9

0

1

-8

0

KEEP ONLY EVENT# & CTDNO

```
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)
```

0

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)

EVENT#     CTDNO
 420.0200     2.0000
 420.1000     4.0000
 421.0200     7.0000

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)

  OUTPUT FILENAME
  FILENAME
```

TYPE OUT RESULTS

2

SAVE IN TEMP.DAT

4

TEMP.DAT

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROL FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

MERGE EVENT.DAT WITH NUTRIENT DATA

1

```
STEP   1
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES
```

```
ENTER CHOICE (  0- 12)

  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)

  FILENAME

  MERGE WITH
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)

  FILENAME

  VARIABLE FOR KEY
 1 EVENT#
ENTER CHOICE (  1-  1)

COMPARISON OPERATOR
1 =
2 =+-5%
ENTER CHOICE (  1-  2)

  VARIABLE TO COMPARE
 1 DEPTH
 2 TEMP
 3 SAL
 4 NO3
 5 NO4
 6 SI
 7 THETA
 8 SIG-TH
 9 EVNO
ENTER CHOICE (  1-  9)

ADD MORE CON"ITIONS
0 END OF CONDITIONS
1 AND
ENTER CHOICE (  0-  1)

  VARIABLE(S) TO KEEP
 1 EVENT#
-1 DEPTH
-2 TEMP
-3 SAL
-4 NO3
-5 NO4
-6 SI
-7 THETA
-8 SIG-TH
-9 EVNO
  0 END OF KEEP CONDITIONS
```

```
9

0

EVENT.DAT

0

*NUTKNO93.ALL

1

1

9

0
```

BASED ON COMMON EVENT #'S

KEEP EVENT# DEPTH NO3

```
ENTER CHOICE ( -9-  1)
                                                    1
ENTER CHOICE ( -9-  1)
                                                   -1
ENTER CHOICE ( -9-  1)
                                                   -4
ENTER CHOICE ( -9-  1)
                                                    0

STEP   4
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)
                                                    2
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
  1 results from step
ENTER CHOICE (  0-  3)
                                                    1
 VARIABLE FOR SELECTION
 1 EVENT#
 2 DEPTH
 3 NO3
ENTER CHOICE (  1-  3)
                                                    2
 COMPARISON VALUE
                                                 75.0

STEP   5
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES
```

Now INTERPOLATE RESULTS OF LAST STEP TO 75m DEPTH

```
ENTER CHOICE (  0- 12)
```

0

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)

EVENT#     DEPTH      NO3
 420.1000    75.0000     8.9193
 421.0200    75.0000    10.8532

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

2

TYPE RESULTS

4 (could be 5)

SAVE AS TEMP1.DAT

```
  OUTPUT FILENAME
  FILENAME
```

TEMP1.DAT

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

RESET EVERYTHING

0

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

1

```
STEP   1
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
```

MERGE TEMP WITH CTD DATA

1

```
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)
                                                    9

  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0

  FILENAME
                                                TEMP.DAT

  MERGE WITH
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0

  FILENAME
                                              *CTDKN093.ALL

  VARIABLE FOR KEY
 1 EVENT#
 2 CTDNO
ENTER CHOICE (  1-  2)
                                                    2

COMPARISON OPERATOR
1 =
2 =+-5%
ENTER CHOICE (  1-  2)
                                                    1

  VARIABLE TO COMPARE
 1 SIGTH
 2 PRES
 3 TEMP
 4 SAL
 5 O2
 6 DYNHGT
 7 STNO
ENTER CHOICE (  1-  7)
                                                    7

ADD MORE CONDITIONS
0 END OF CONDITIONS
1 AND
ENTER CHOICE (  0-  0)
                                                    0

  VARIABLE(S) TO KEEP
 1 EVENT#
 2 CTDNO
-1 SIGTH
-2 PRES
-3 TEMP
-4 SAL
-5 O2
-6 DYNHGT
-7 STNO
  0 END OF KEEP CONDITIONS
```

BASED ON COMMON CTDNO'S

KEEP EVENT#, PRES, O2

```
ENTER CHOICE ( -7-  2)

ENTER CHOICE ( -7-  2)

ENTER CHOICE ( -7-  2)

ENTER CHOICE ( -7-  2)

ENTER CHOICE ( -7-  2)


STEP   4
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)

  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
  1 results from step
ENTER CHOICE (  0-  3)

 VARIABLE FOR SELECTION
 1 EVENT#
 2 CTDNO
 3 PRES
 4 O2
ENTER CHOICE (  1-  4)

 COMPARISON VALUE


STEP   5
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES
```

1

2

-2

-5

0

2

1

3

75.0

INTERPOLATE LAST RESULT TO 75 m

```
ENTER CHOICE (  0- 12)
```

0

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER !HOICE (  0-  5)
```

2

```
EVENT#     CTDNO      PRES       O2
 420.0200     2.0000    75.0000     5.5136
 420.1000     4.0000    75.0000     5.4333
 421.0200     7.0000    75.0000     5.2727
```

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)

  OUTPUT FILENAME
  FILENAME
```

could be 5 → 4

TEMP2.DAT

```
0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

1

```
STEP   1
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES
```

TYPE RESULTS

SAVE AS TEMP2.DAT

MERGE TEMP1.DAT WITH TEMP2.DAT

```
ENTER CHOICE (  0- 12)
                                                    5
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0
  FILENAME
                                                TEMP1.DAT
  MERGE WITH
  SELECT SOURCE FILE
 0 ANOTHER FILE FROM DISK
ENTER CHOICE (  0-  0)
                                                    0
  FILENAME
                                                TEMP2.DAT
  VARIABLE FOR KEY
 1 EVENT#
 2 DEPTH
 3 NO3
ENTER CHOICE (  1-  3)
                                                    1
COMPARISON OPERATOR
1 =
2 =+-5%
ENTER CHOICE (  1-  2)
                                                    1
  VARIABLE TO COMPARE
 1 EVENT#
 2 CTDNO
 3 PRES
 4 O2
ENTER CHOICE (  1-  4)
                                                    1
ADD MORE CONDITIONS
0 END OF CONDITIONS
1 AND
ENTER CHOICE (  0-  1)
                                                    0
  VARIABLE(S) TO KEEP
 1 EVENT#
 2 DEPTH
 3 NO3
-1 EVENT#
-2 CTDNO
-3 PRES
-4 O2
  0 END OF KEEP CONDITIONS
ENTER CHOICE ( -4-  3)
                                                    1
ENTER CHOICE ( -4-  3)
                                                    2
ENTER CHOICE ( -4-  3)
                                                    3
ENTER CHOICE ( -4-  3)
                                                   -4
ENTER CHOICE ( -4-  3)
```

BASED ON COMMON EVENT #

KEEP EVENT#, DEPTH, NO3, O2

```
STEP   4
0  FINISH CONSTRUCTING PSEUDOFILE
1  SELECT ROWS
2  INTERP ROWS (ASCEND)
3  INTERP ROWS (DESCEND)
4  SELECT COLUMNS
5  JOIN TWO FILES (ASCEND)
6  JOIN TWO FILES (DESCEND)
7  INTERP FILES (ASCEND)
8  INTERP FILES (DESCEND)
9  FILL OUT JOIN (ASCEND)
10 FILL OUT JOIN (DESCEND)
11 ARITHMETIC CONVERSIONS
12 CHAIN FILES

ENTER CHOICE (  0- 12)
```

0

```

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

0

TYPE OUT FINAL RESULTS

```

EVENT#     DEPTH      NO3        O2
 420.1000    75.0000     8.9193     5.4333
 421.0200    75.0000    10.8532     5.2727

0 RESTART
1 CONSTRUCT PSEUDOFILE
2 TYPE FROM FILE OR PSEUDOFILE
3 PRINT
4 COPY TO DISK
5 SAVE PSEUDOFILE DESCRIPTION
ENTER CHOICE (  0-  5)
```

2

STOP

```
TT5  --  STOP

>
```

^Z

O2 data for depths < 75m, some other problems can arise. Presumeably, one would like to use the interpolating join process, based on depth, to produce NO3 and O2 at common depths after selecting the stations, as sketched in 3.8. As long as both input tables to the join step are pretty regular, having data from all stations covering all depth ranges, this procedure will work. Howver the event numbers with nutrient data (420.10, 421.02) do not correspond exactly to those with CTD data (420.02, 420.10, 421.02); thus we would end up interpolating the wrong pair of station data. This will show up if we keep the event numbers from both TEMP1 and TEMP2 number in the final result (figure 3.9). We can correct this problem by editing out the troublesome station (420.02) with either row selection or editing of the original EVENT.DAT. Other difficulties arise because station 420.10 has its last <75m value at 25.7m while station 421.02 begins at 58m. Based on a single column of depth information, we cannot know that we switched stations. We may need a join with both equality of certain columns and interpolation to correct this problem.+

Note-- for operations like this, if there are not too many stations, it may be better to do the NO3-O2 interpolation and depth selection station by station and check them before chaining the results together (figure 3.10).

FIGURE 3.8

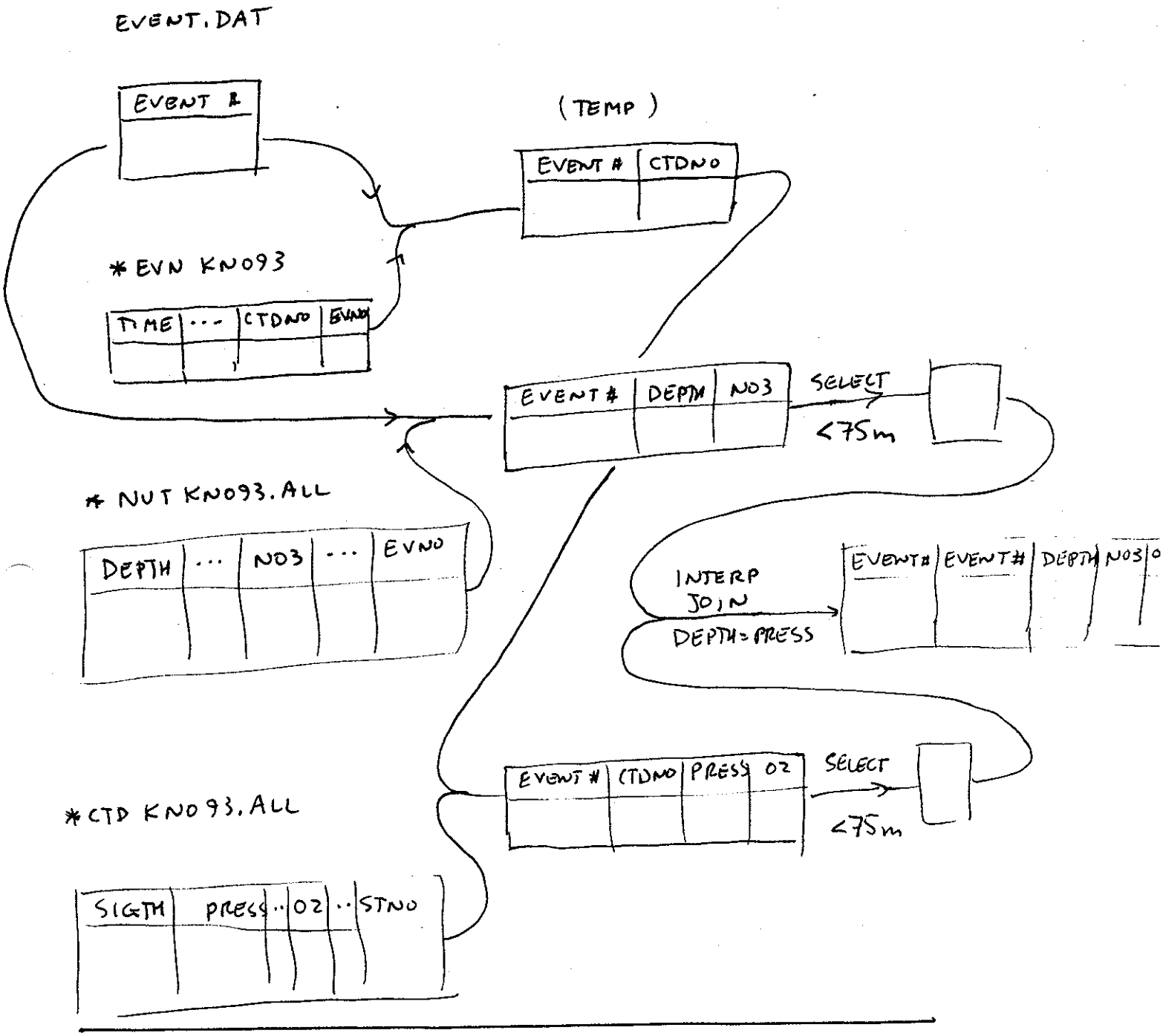
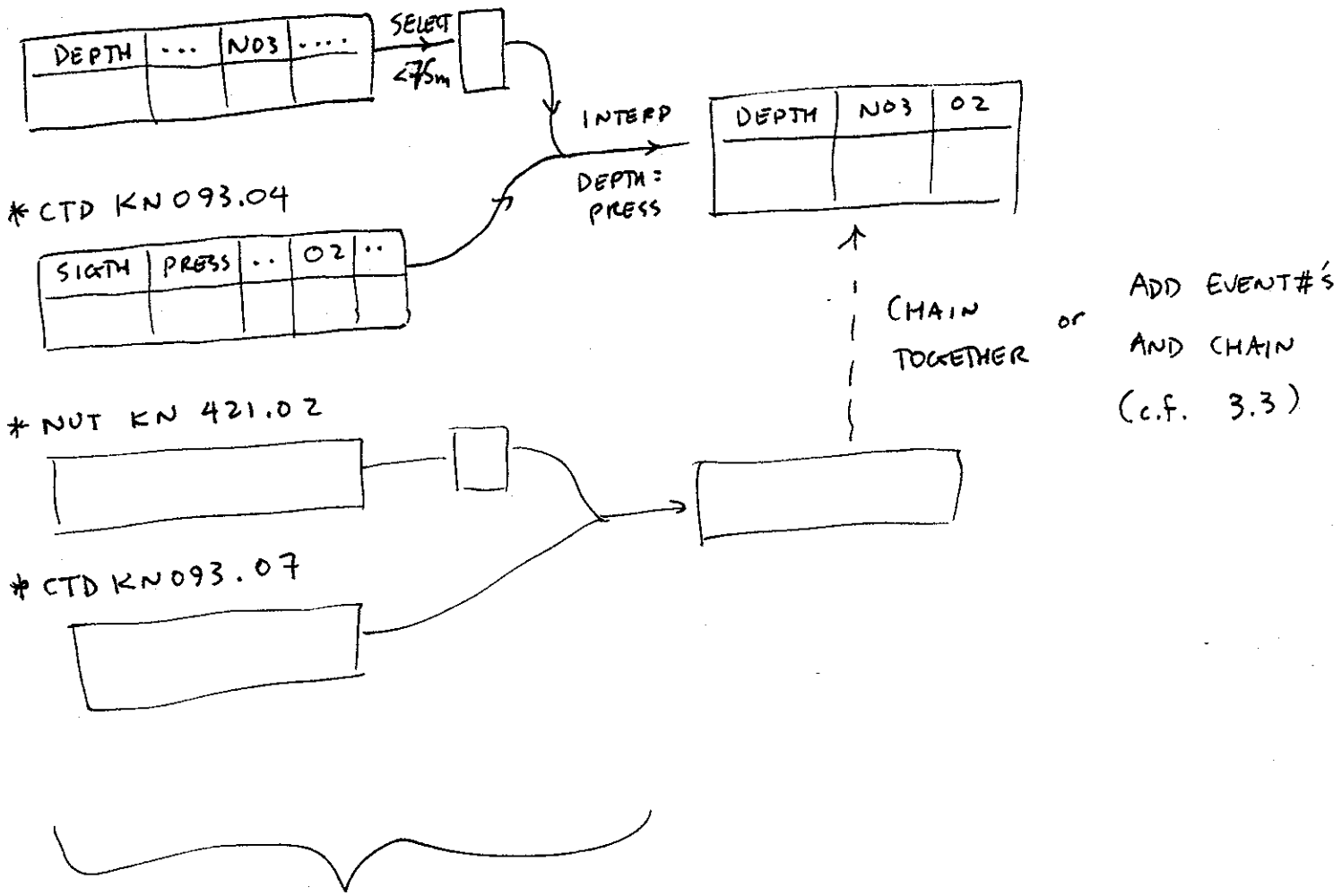


FIGURE 3.9

EVENT#	EVENT#	DEPTH	NO3	O2
420.1000	420.0200	3.2000	2.4900	6.8170
420.1000	420.0200	3.6000	2.4600	6.8510
420.1000	420.0200	8.4000	2.6400	6.9190
420.1000	420.0200	11.3000	3.6900	6.9230
420.1000	420.0200	17.6000	5.1400	6.4060
420.1000	420.0200	25.7000	8.2600	6.1700
421.0200	420.0200	58.0000	8.0800	5.8080

FIG. 3.10



THIS COULD BE DONE WITH WILDCARD SPECIFICATIONS

DEPTH	NO3	O2	EVENT#
8.4000	2.6400	6.9230	420.1000
11.3000	3.6900	7.0227	420.1000
17.6000	5.1400	6.8400	420.1000
25.7000	8.2600	6.3685	420.1000
58.0000	8.0800	5.6100	421.0200

4) Data access on VAX

In order to get data, you need to know its name. For convenience we have organized the data into subdirectories of <WCRDB> and put the translation into the data base routines. All names for data in the archive will be of the form

*TTTSSNNN.NNN

where

TTT = type of data (CTD,NVT, EVN...)

SS = Ship (KN, EN, OC, A2)

NNN.NN = identification number

The identification number depends upon the type of data; it may be the event number or the number of the cruise and station number or cruise.ALL for collected sets. Event number - CTD number correspondences are in the

*EVNsscccc

files where ccc is the cruise number.

These filenames are translated into

<WCRDB.TTT.SS>NNN.NN

Searching is therefore fairly straightforward using the GOTO and DIRE commands from the $ prompt.

GOTO WCRDB provides a listing of types of data

GOTO.CTD then lists all the ships having CTD data

GOTO.KN lists all the CTD data from the Knorr

DIRE 093.* lists all the KN093 CTD data etc.

Note that ^Y will terminate these listings.

We could find all the data labelled by event number 614.02 by doing

DIRE < WCRDB.*.* >614.02

(DIRE < WCRDB.*.KN > 614.02 would also work since only KN data would exist for this particular event number.)

It is hoped that the adoption of this standard form for names will make them easily remembered during the use of DBMENU.

Acknowledgements: This work was supported by an NSF grant to MIT. The Apple version is being prepared by Tim Cowles. Thanks also to Peter Wiebe for comments on earlier (and present) versions of the program and to Terry Joyce and Jane Dunworth-Baker for helping structure the files on the VAX.

Appendix 1. File format

The format used by the database routines is quite simple. Conceptually, the data is organized into tabular form; e.g.

EVNO	DEPTH	TEMP	SAL
602.1	2	12.5	33.42
602.1	5	10.1	33.47
602.1	10	8.9	33.53
603.4	2	13.2	33.44
603.4	8	9.9	33.48

But as actually stored on disk, this file looks like

```
                  ↓ these things are comments- not in file! ↓
4                 ------ this is the number of columns-- number of
EVNO                |     rows (down the page) is not specified
DEPTH               |
TEMP                |----- these are the names of the columns (8
SAL                 |      character limit on VAX)
602.1           |
2               |---------- these are the data from the first row
12.5            |
33.42           |
602.1               |
5                   |-------data from second row
10.1                |
```

```
33.47                |
602.1          |
10             |------------third row
8.9            |
33.53          |
603.4                |
2                    |--------etc.
13.2                 |
33.44                |
603.4
8
9.9
33.48
```

-- A separate line for each piece of data.

The numbers are all read with a free-format read, so that the number of decimal places and leading or trailing blanks are all unimportant.

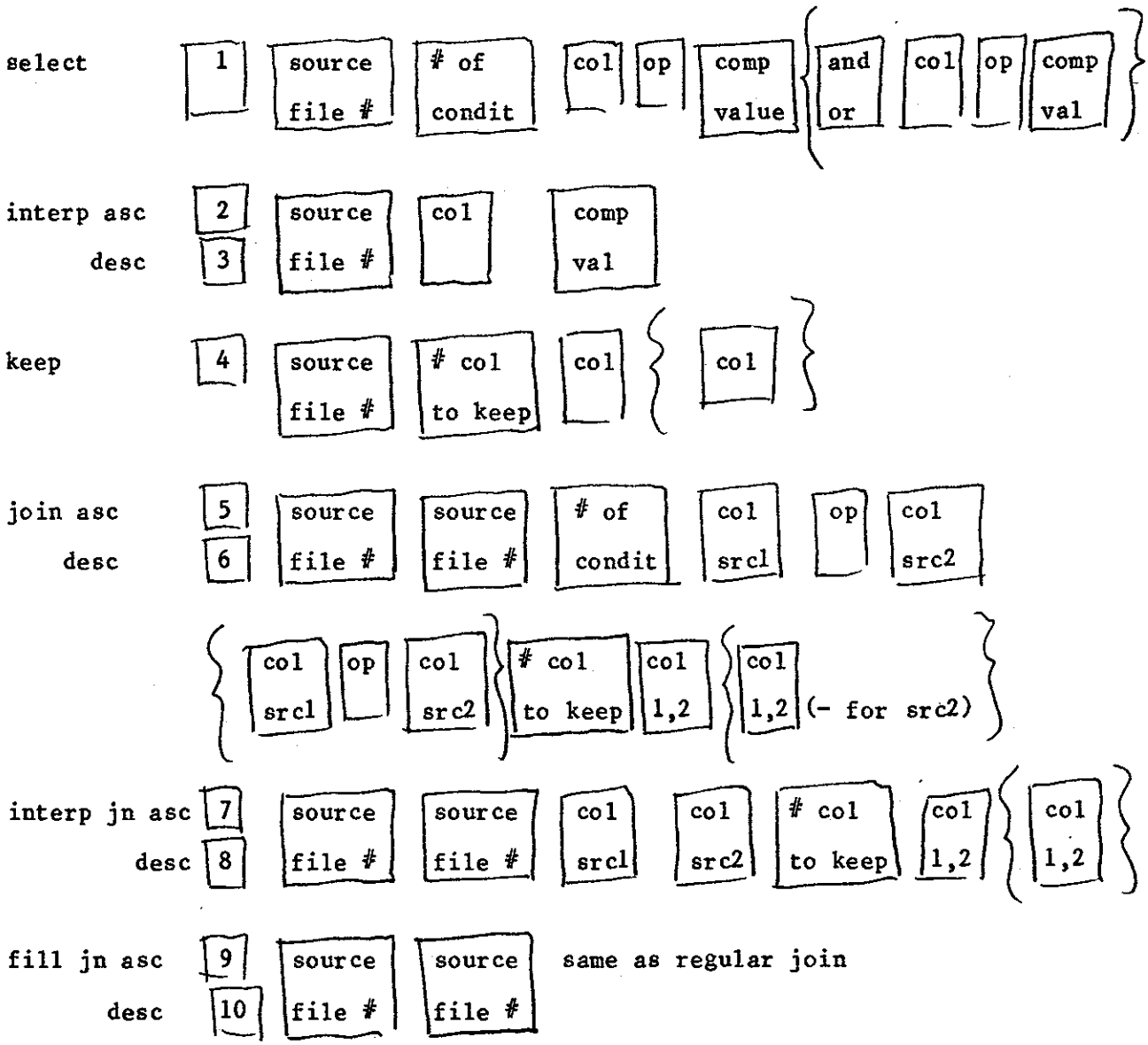
We strongly rcommend the use of a standard value -999 to indicate missing data entries. The number of rows is determined by reaching end of file.

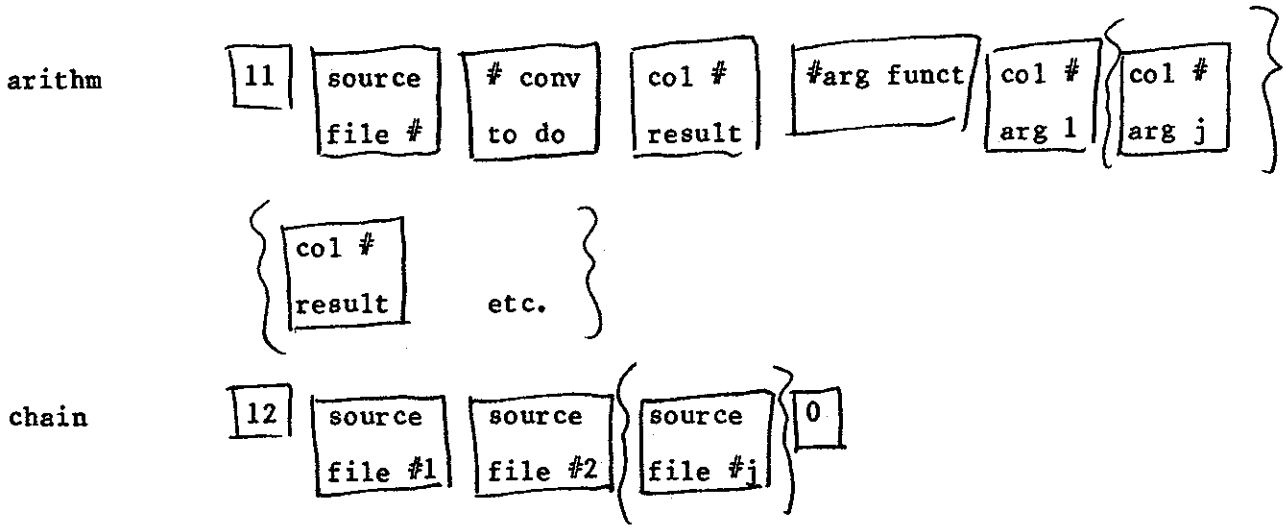
Appendix 2. Pseudofile format

The distinguishing mark of a pseudofile on disk is that the number of columns is the negative of the actual number. Thus the opening subroutine can distinguish immediately which style it is dealing with. The format is:

```
- # of columns

name col. 1
name col. 2
     .
     .
     .

pointer to first pseudoinstruction to execute

# of instruction data items to follow

instruction
data
items
     .
     .
     .

# of subfiles
     filename          &addr in instr array
     # columns    or    #columns
     relocation pointer  relocation pointer
          .
```

The pseudoinstructions look like (stretched out horizontally rather than vertically):





The pseudofile for figure 3.1 is shown in A2.1 while that of 3.3 is in A2.2. Note that the instruction sections correspond fairly closely to the responses to the menus.

Each instruction uses one or more source files. These are located using the "subfile" information. They may be either another file from disk (in which case the filename appears in the pseudofile) or a result from some previous step (in which case the &xxxxxxx form is used with the xxxxxxx pointing to the address in the instruction section which should be called to retrieve this source file). Addresses begin at 1 and increment by 1 for each line.

The relocation pointers tell which line within the instruction referenced this particular subfile. When this pseudofile is loaded in, the instructions will not generally start at location 1 within the I0 or INSTR array; nor will the subfiles be labelled by their original numbers. Thus both the & pointers and the instructions which give subfile numbers must be altered.

The information concerning subfiles (or for that matter primary files) is contained in arrays L5, P5, N5,F0 (Basic) or IWK (1..13) in Fortran. These look like

L5(), IWK(1,)	P5(), IWK(2,)	N5(),IWK(3,)	F0$(), IWK(4..13)
1) File which is not open			
= 0	----	>0 not yet used <0 finished and closed |N5| = # columns	name on disk
2) Open real file			
>0	0 not yet read >0 pointer to region in D array where data will go	>0 # columns	name on disk
3) Pseudofile in memory			
<0 value is - addr. in I0 or INSTR array for this pseudoinstr.	0 not yet read >0 pointer to region in D where results will go.	>0 # columns >0 # columns <0 finished and closed	

FIG. A2.1

PSEUDOFILE FOR FIG 3.1

-4	
EVNO	
DEPTH	
TEMP	
SAL	
1	begin
10	# instr.
1	①
1	
2	
2	Select instr.
2	
3	
1	
2	
3	
7	
1	# files
TS	
4	
2	

FIG. A2.2

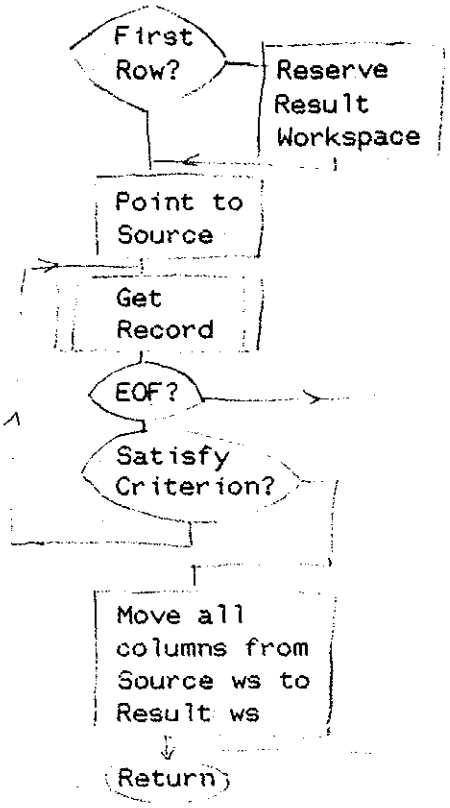
PSEUDOFILE FOR 3.3

-4		
PRESS		
TEMP		
SAL		
STNO		
19	begin	
22	# instr	
11	①	
2		
1		
4	arith instr.	
3		
3		
0		
1		
1		
11	⑩	
4		
1		
4	arith instr.	
3		
3		
0		
1		
2		

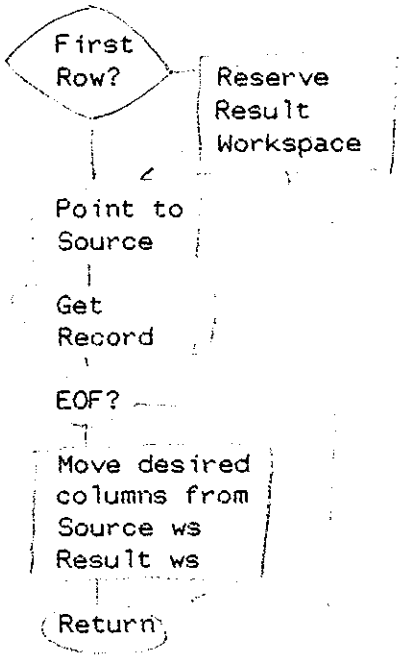
12	⑲	
1	Chain instr	
3		
0		
4		# sub files
&	1	subfile #1 → instr #1
4		
20		
C		subfile #2 → "C" on disk
3		
2		
&	10	subfile # 3 → instr 10
4		
21		
D		subfile # 4 → D on disk
3		
11		

Appendix 3

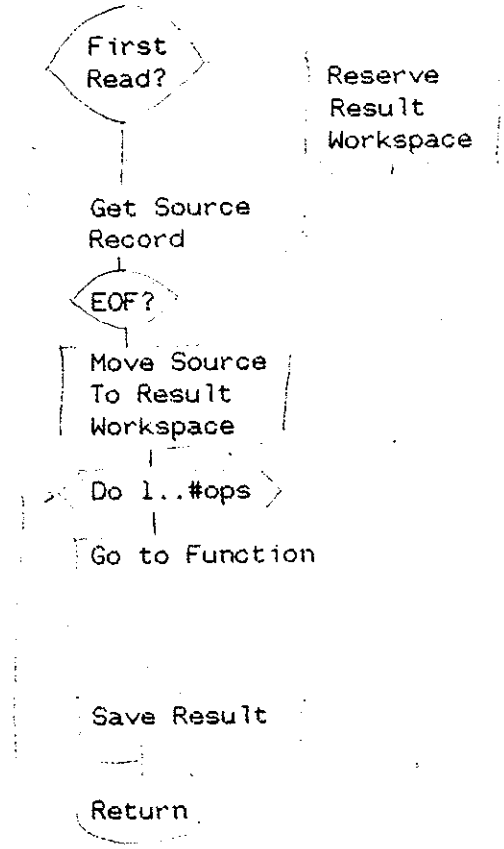
SELECT ROW OPERATION
12000-12490



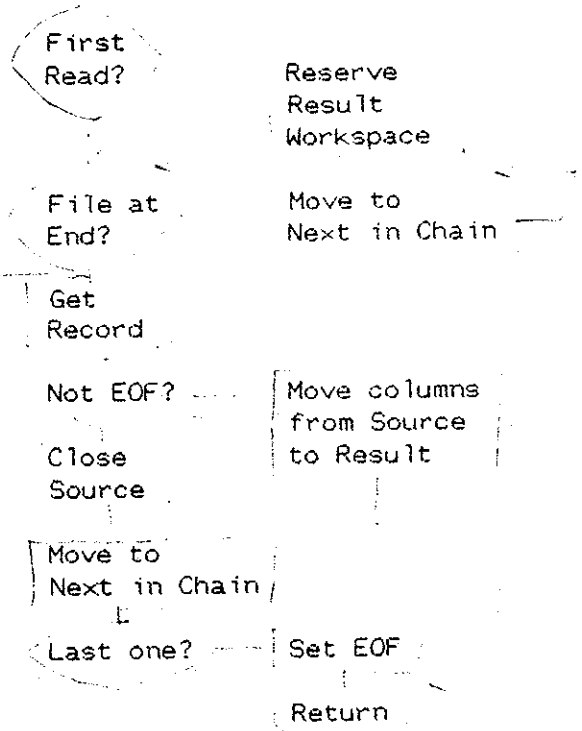
SELECT COLUMN OPERATION
13000-12130



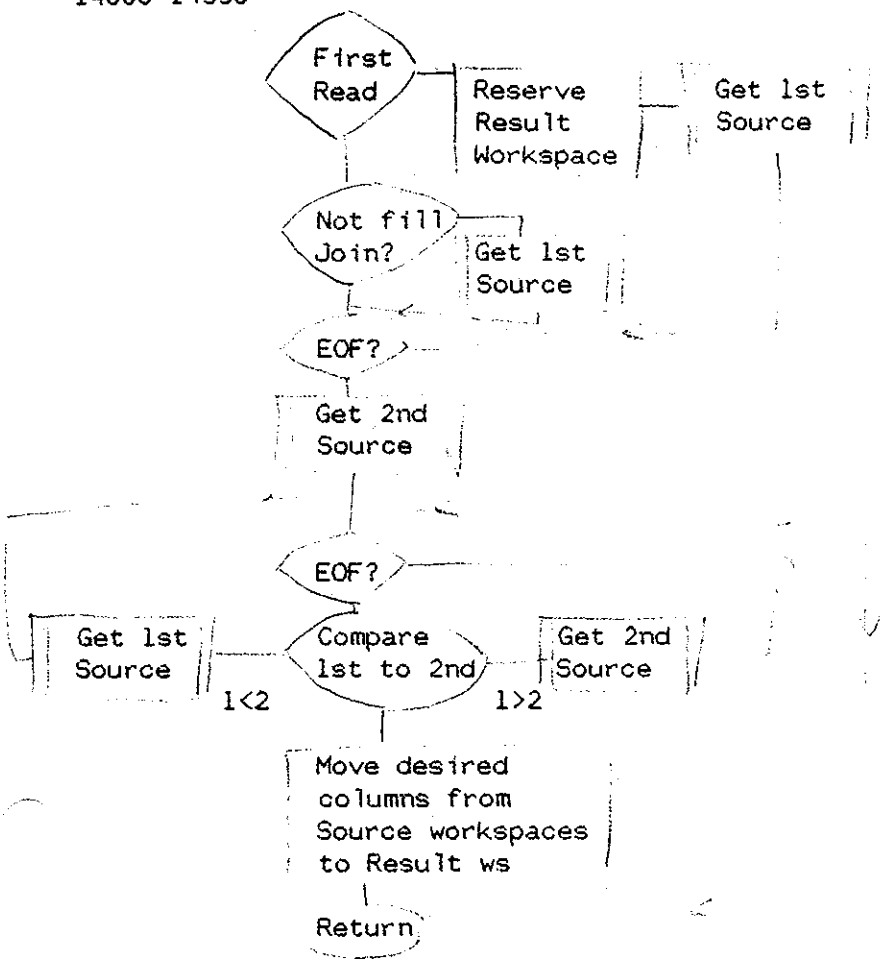
ARITHMETIC CONVERSIONS
15000-15920



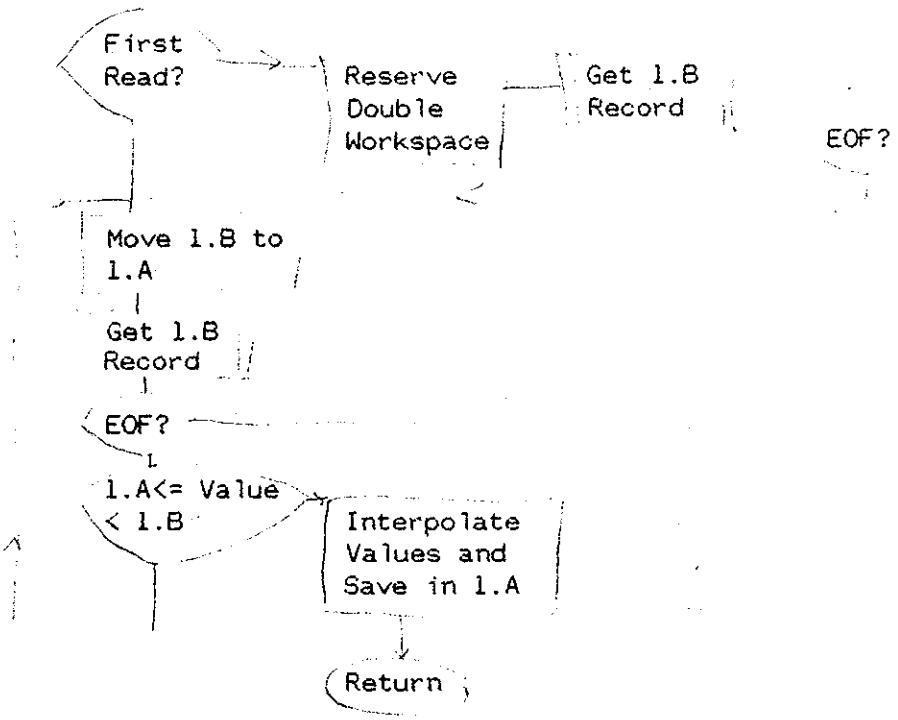
CHAIN FILES
16000-16520



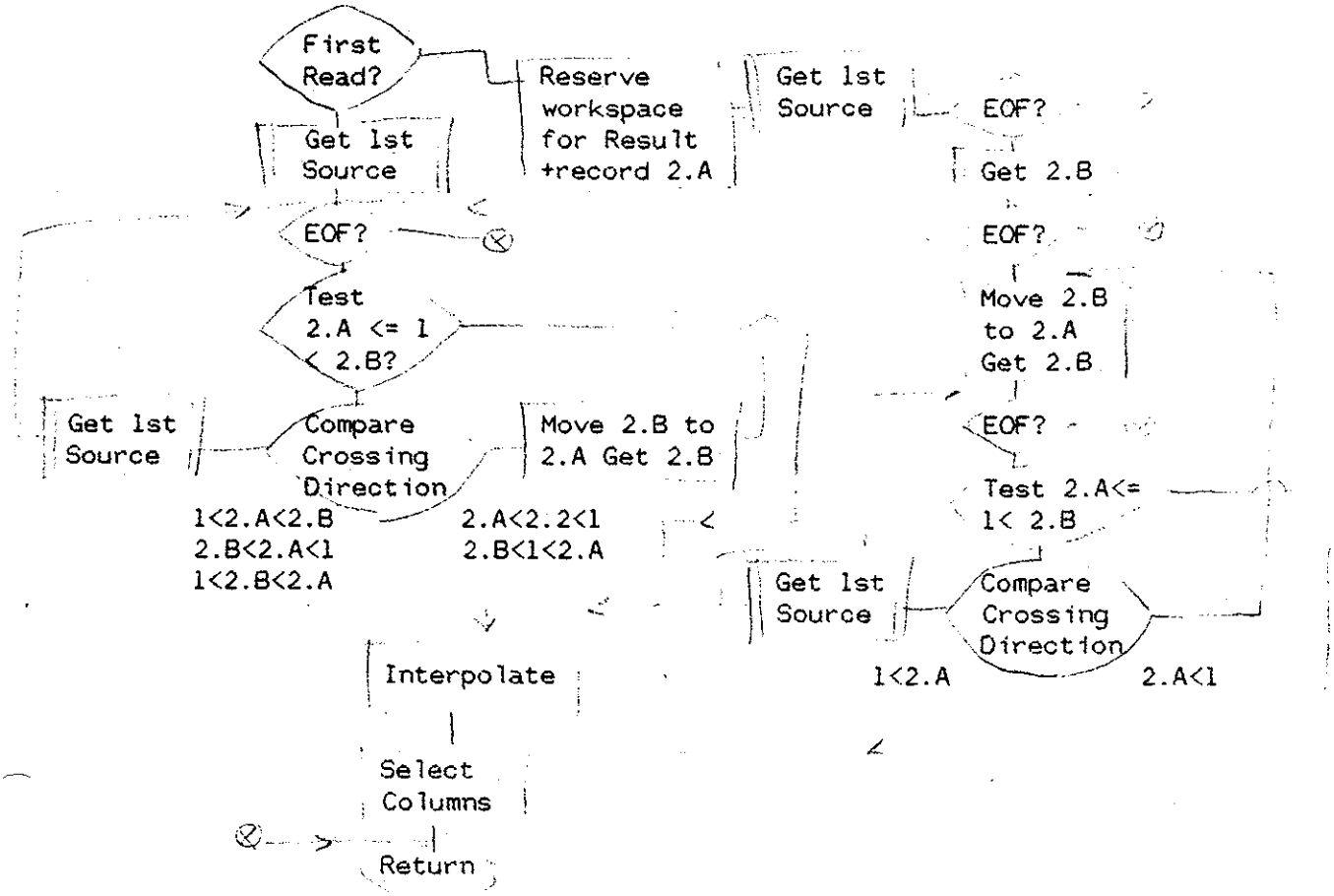
JOIN OR FILL JOIN
14000-14330



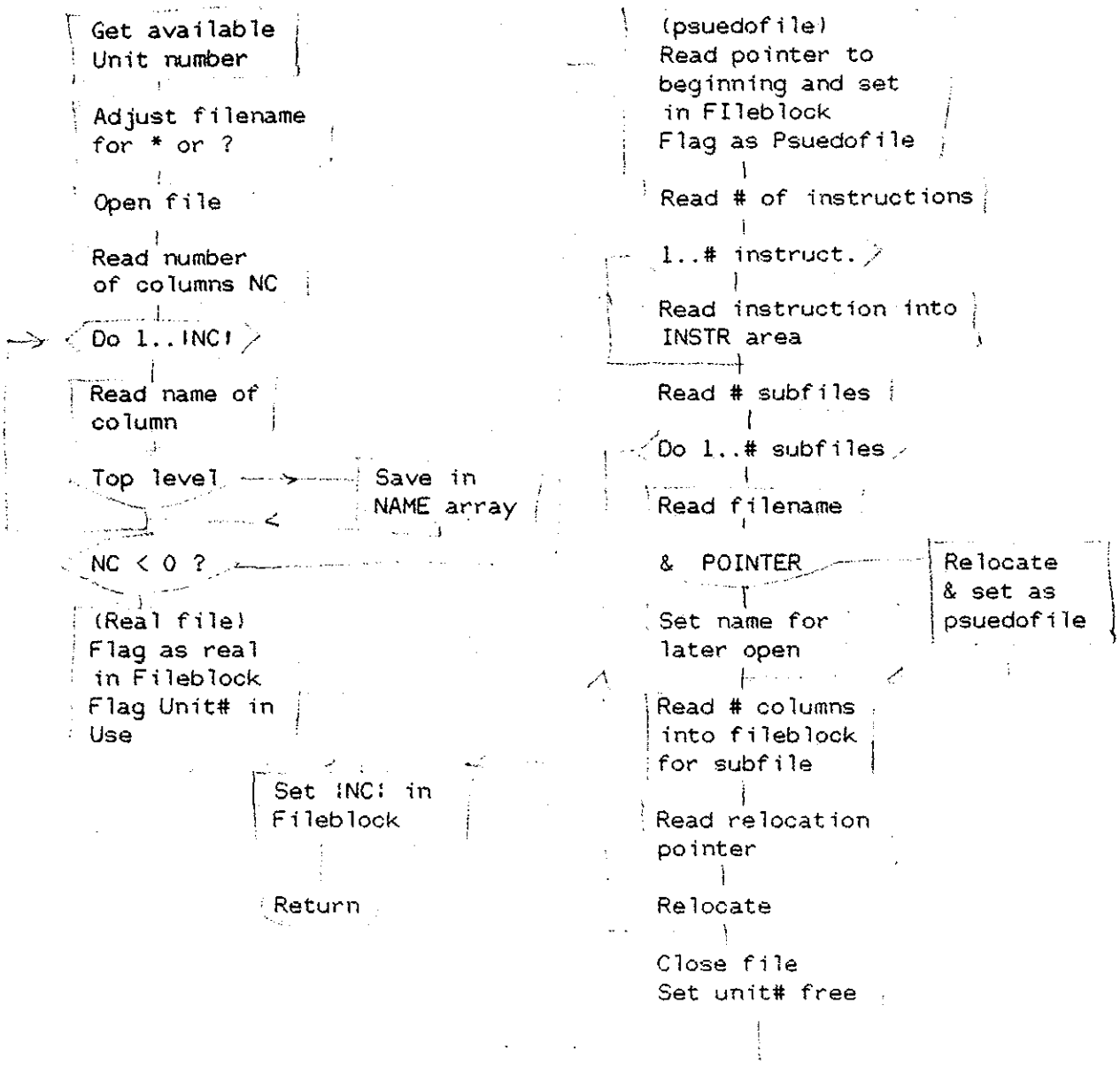
INTERPOLATE ROW
12500-12890



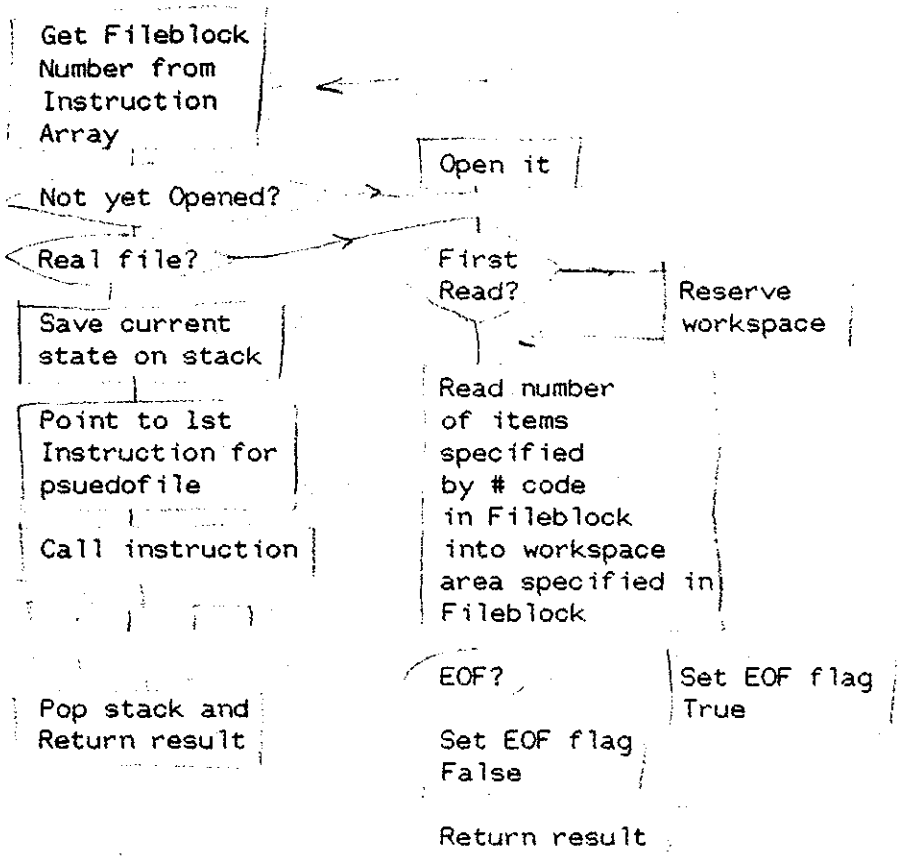
INTERPOLATE FILES
14500-14800



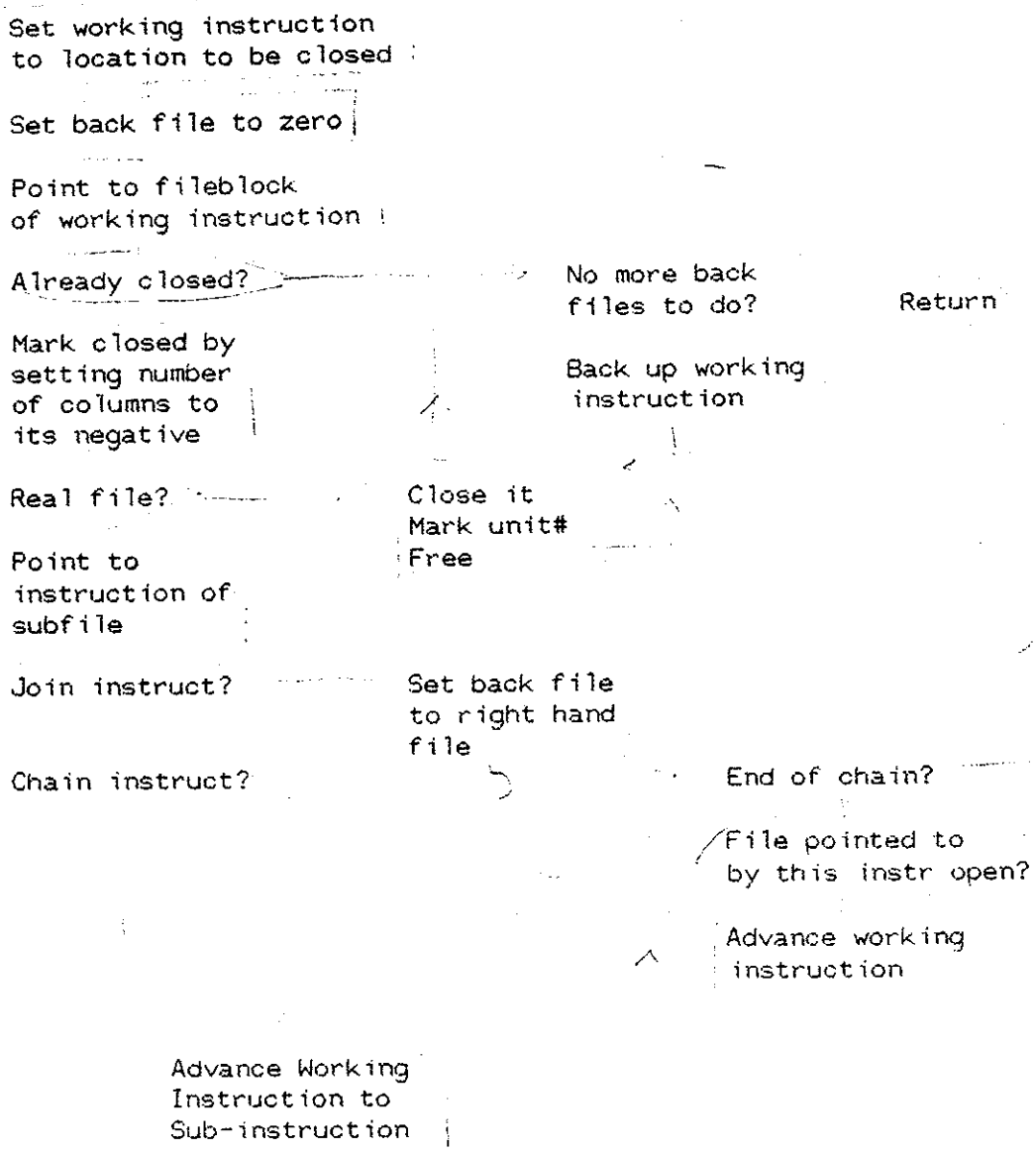
OPEN SUBROUTINE
OPNSUB 10400-10910



GET DATA
DBGET 10210-10390



CLOSE ROUTINE
DBCLOS 16050-16270



Appendix 4. Interfacing other programs to use the WCR database routines

FORTRAN

There are three calls here which must be used in place of corresponding standard FORTRAN i/o statements in your programs:

CALL DBOPEN(FILNAM,NB) opens a file with the name which is contained as ASCII characters in the LOGICAL*1 array FILNAM. The last character must be followed by a binary 0. FILNAM must be dimensioned for 20 characters, but names should be restricted to <13 characters in length. (The subroutine DBGETF(FILNAM) can be called to prompt the user and set up the filename with the trailing 0.) The returned INTEGER NB gives the number of columns in the table.

CALL DBGET(EF,ND,D) gets a line of data into the REAL array D in positions D(1)..D(NB). This is a row of the final table either as read from an actual file or as constructed from 1 to 10 different files. The INTEGER ND is the dimension of the array D which must be large enough to handle all intermediate computions; probably at least 100, but increased as necessary. Some bounds checking is done. For the pseudofiles containing large amounts of CTD data, dimensions of 500-1000 are required. The LOGICAL*1 variable EF returns .TRUE. when end of file condition has been reached.

CALL DBCLOS closes out data base files.

CALL DBCLR clears the workspace for the database routines.

For linking the task, one must include <WCRDB.PGM>DBCOM and <WCRDB.PGM>DBOPEN with the modules one has written. Note that in building an executeable task under RSX11M, sufficient space should be reserved for file buffers, since many pseudofile descriptions may involve opening a fair number of files and later closing them.

The examples shown in figures A4.1-A4.5 show various techniques and problems in interfacing. The first case, DBMEAN (figure A4.1), gives the simplest case: a program to compute means and standard deviations of each column of a table. Space is set aside for the working data array, the filename, the end-of-file flag, and the accumulators for computing means and standard deviations. The routine GETFIL is used to request the filename, the file is opened, and lines are read with sums being made until the end of file. The statistics are then printed and the file is closed. The following figure shows a sample run of this program along with the input data table.

The next problem for interfacing is retrieving the column names. This requires including in the user program the COMMON/DBNAME/ as indicated in the DBTYPE program in A4.3. This program retrieves the filename from the command line >DBT XXXXXX rather than from the user, but otherwise is straightforward.

One can also read multiple files with the database routines as indicated in the DBTST program A4.4. The user must include the COMMON/DBUNIT which passes information concerning the effective unit number of the particular data file and the area in the D array where the result from this file will be deposited. NUNIT and NDATA are set upon open; however, it is important that the DBOPEN call be followed immediately with a DBGET call from that

FIG. A4.1

```
TYPE DBMEAN.FTN
Type 4R0 26-Feb-84 23:31:15
C       COMPUTE MEANS AND STD. DEV. OF COLUMNS
C
C       RESERVE SPACE FOR DATA FROM DB ROUTINES
        DIMENSION D(300)
C       SPACE FOR FILE NAME
        LOGICAL*1 FILNAM(20)
C       END OF FILE FLAG
        LOGICAL*1 EF
C       ACCUMULATORS FOR STATISTICS
        INTEGER CNT(20)
        REAL*4 MEAN(20),STDDEV(20)
C       INITIALIZE ACCUMULATORS
        DATA MEAN,STDDEV,CNT/40*0.0,20*0/

C       REQUEST NAME OF FILE FROM USER
        CALL GETFIL(FILNAM)
C       OPEN FILE  NB=NUMBER OF COLUMNS IN TABLE
        CALL DBOPEN(FILNAM,NB)
C
C       THE NEXT STATEMENT GETS A ROW OF DATA INTO D(1..NB)
C       EF=.TRUE. WHEN AT END OF FILE
C
100     CALL DBGET(EF,300,D)
C       JUMP TO COMPUTE STATS IF END OF FILE
        IF(EF) GOTO 200

C       ACCUMULATE RUNNING MEAN, COUNT, RUNNING SUM OF SQUARE DEVIATIONS
        DO 10 I=1,NB
        IF(D(I).EQ.-999) GOTO 10
        D1=D(I)-MEAN(I)
        CNT(I)=CNT(I)+1
        MEAN(I)=MEAN(I)+D1/CNT(I)
        STDDEV(I)=STDDEV(I)+D1*(D(I)-MEAN(I))
10      CONTINUE
C       GO BACK AND GET NEXT ROW
        GOTO 100

C       ARRIVE HERE AT EOF
C       COMPUTE MEANS AND STD. DEV. FOR EACH COLUMN
200     DO 19 I=1,NB
        D1=0
        IF(CNT(I).LE.1) GOTO 19
        D1=SQRT(STDDEV(I)/(CNT(I)-1))
19      STDDEV(I)=D1
C       PRINT OUT ANSWERS
        WRITE(5,3)(CNT(I),I=1,NB)
3       FORMAT(1X,8(I5,5X))
        WRITE(5,2) (MEAN(I),I=1,NB)
        WRITE(5,2)(STDDEV(I),I=1,NB)
2       FORMAT(1X,8(F9.4,1X))
C       CLOSE THE FILE
        CALL DBCLOS
        END
```

FIG. A 4.2

```
>DBT TS.DAT
BNO        TEMP       SAL        EVNO
   1.0000    20.3000    35.2200    1.0000
   2.0000    20.0000    35.2400    1.0000
   3.0000    18.8000    35.3000    1.0000
   4.0000    18.5000    35.4500    1.0000
   5.0000    18.0000    35.5500    1.0000
   1.0000    20.4000    35.2300    2.0000
   2.0000  -999.0000    35.2700    2.0000
   3.0000    18.9000    35.3300    2.0000
   1.0000    20.6000    35.2700    3.0000
   2.0000    20.2000    35.3300    3.0000
   3.0000    18.5000    35.4400    3.0000
   4.0000    19.0000    35.6600    3.0000
   5.0000    18.0000    35.5500    3.0000

>RUN DBMEAN
  FILENAME
TS.DAT
   13         12         13         13
   2.7692    19.2667    35.3723    2.0000
   1.4233     0.9698     0.1435    0.9129
```

FIG. A4.3

```
>TYPE DBTYPE.FTN
Type 4R0 26-Feb-84 14:06:37
C       RESERVE SPACE FOR DATA AREA
        dimension d(500)
C       END OF FILE FLAG
        LOGICAL*1 EF,c(80)
        INTEGER*2 CALIGN

C       MAX NUMBER OF COLUMN NAMES AND RETURNED NUMBER
        INTEGER*2 NN,NB
C       SPACE FOR NAMES
        REAL*8 NAME(20)
        COMMON/DBNAME/NN,NAME

C       RETRIEVE FILENAME FROM COMMAND LINE

        CALL GETMCR(C(2),iQ)
        IF(iQ.LT.4)STOP ' NO FILENAME SPECIFIED'

C       APPEND BINARY ZERO

        C(iQ+2)=0

C       OPEN THE FILE, GET BACK # COLUMNS NB AND NAMES IN
C       NAME(1..NB)

        CALL DBOPEN(C(6),nb)

C       PRINT OUT NAMES OF COLUMNS

        TYPE 1,(NAME(I),I=1,Nb)
1       FORMAT(1X,10(2X,A8))

C       RETRIEVE ROW OF TABLE. 500 IS SIZEOF WORKING ARRAY.
C       EF IS SET TO .TRUE. AT END OF FILE

90      CALL DBGET(EF,500,D)
         IF(EF)GOTO 100

C       NOT END OF FILE-- TYPE DATA

        TYPE 2,(D(I),I=1,Nb)
2       FORMAT(1X,10(F9.4,1X))

C       GO BACK AND GET NEXT LINE

        GOTO 90

C       END OF FILE-- CLOSE AND EXIT

100      CALL DBCLOS
        END
```

FIG. A 4.4

```
>TYPE DBTST.FTN
Type 4R 19-Feb-84 22:57:53
        DIMENSION D(300)
        LOGICAL*1 FIDOL(20)
        INTEGER*2 NU(3),ND(3),NB(3),NR(3)
        INTEGER*2 NUNIT,NDATA,NBA,LFQ
        LOGICAL*1 EF,EFA(3)
        COMMON/DBUNIT/NUNIT,NDATA
        DATA EFA/3*.FALSE./

        TYPE *,'NUMBER OF FILES'
        ACCEPT *,NRA
        DO 50 NA=1,NRA
        TYPE *,'FILE NAME ',NA
        ACCEPT 1,L,(FIDOL(I),I=1,L)
1       FORMAT(Q,20A1)
        FIDOL(L+1)=0

        CALL DBOPEN(FIDOL,NBA)
        NU(NA)=NUNIT
        ND(NA)=NDATA
        NB(NA)=NBA
        CALL DBGET(EF,300,D)
50      CONTINUE

100     TYPE *,'WHICH FILE'
        ACCEPT *,LFQ
        IF (EFA(LFQ))GOTO 200
        TYPE 131,(D(I),I=ND(LFQ), ND(LFQ)+NB(LFQ)-1)
131     FORMAT(1X,10F10.4)
        NUNIT=NU(LFQ)
        CALL DBGET(EF,300,D)
        IF(.NOT.EF)GOTO 100
        EFA(LFQ)=.TRUE.
        CALL DBCLOS
        GOTO 100

200     TYPE *,'END OF FILE'
        GOTO 100
        END
```

Data is retrieved by executing a

GOSUB 10200 statement.

Again the variable EF is set so that an IF test based on EF will inform the user when to stop reading. The data is returned in the (global) array D which must be DIMed early in the program. For multiple files, the variables NUNIT must be set before calling 10200.

Finally, the file is closed by using a

GOSUB 10300

(with NUNIT set in case of multiple files).

The clear workspace call is

GOSUB 10400

Figures A4.6, A4.7 and A4.8 are the BASIC analogues of the A4.1 to A4.3 respectively. Refer to the previous discussions for clarification of the purpose of various statements. Most of the variables used in the database routines are named with a letter followed by a number (except for I and J) so that if you avoid these names, you shouldn't conflict with my routines; however, it would do no harm to check carefully!

FIG. A 4.6

STATISTICS OF COLUMNS

```
10 REM COMPUTE MEANS AND STD DEV OF COLUMNS
20 REM
30 REM RESERVE SPACE FOR DATA, INITIALIZE
40 DIM D(300):GOSUB 10000
50 REM SPACE FOR ACCUMULATORS
60 DIM CNT(20),MEAN(20),STDDEV(20)
70 REM
80 REM GET FILE NAME
90 INPUT "FILENAME";FI$
100 REM OPEN FILE FI$, RETURN NB = NUMBER OF COLS.
110 GOSUB 10100
120 REM
130 REM GET LINE OF DATA INTO D(1..NB), RETURN EF=TRUE AT EOF
140 GOSUB 10200
150 REM JUMP TO COMPUT STATS AT EOF
160 IF EF THEN 280
170 REM ACCUMULATE STATS
180 FOR I=1 TO NB
190 IF D(I)=-999 THEN 230
200 D1=D(I)-MEAN(I):CNT(I)=CNT(I)+1
210 MEAN(I)=MEAN(I)+D1/CNT(I)
220 STDDEV(I)=STDDEV(I)+D1*(D(I)-MEAN(I))
230 NEXT I
240 REM GO BACK FOR NEXT ROW
250 GOTO 140
260 REM
270 REM ARRIVE HERE AT EOF, COMPUTE STATS
280 FOR I=1 TO NB
290 D1=0:IF CNT(I)>1 THEN D1=SQR(STDDEV(I)/(CNT(I)-1))
300 STDDEV(I)=D1
310 NEXT I
320 REM PRINT ANSWERS
330 FOR I=1 TO NB:PRINT CNT(I),:NEXT I:PRINT
340 FOR I=1 TO NB:PRINT MEAN(I),:NEXT I:PRINT
350 FOR I=1 TO NB:PRINT STDDEV(I),:NEXT I:PRINT
360 REM CLOSE FILE
370 GOSUB 10300
380 END
```

FIG. A 4.7

TYPE DATA TABLE

```
10 REM RESERVE SPACE AND INITIALIZE
20 DIM D(300):GOSUB 10000
30 REM
40 REM GET FILENAME
50 INPUT "FILENAME";FI$
60 REM OPEN AND GET BACK NB (NUMBER OF COLUMNS)
70 GOSUB 10100
80 REM
90 REM PRINT NAMES
100 FOR I=1 TO NB:PRINT N$(I),:NEXT I:PRINT
110 REM
120 REM GET DATA
130 GOSUB 10200
140 REM ON EOF CLOSE FILE AND STOP
150 IF EF THEN GOSUB 10300:END
160 REM ELSE PRINT DATA
170 FOR I=1 TO NB:PRINT D(I),:NEXT I:PRINT
180 GOTO 130
```

FIG. A 4.8

MULTIPLE FILES

```
10 DIM D(300):GOSUB 10000
20 DIM NU(3),ND(3),NB(3),NR(3),EF(3)
30 INPUT "NUMBER OF FILES";NR
40 FOR NA=1 TO NR
50 INPUT "FILENAME";FI$:GOSUB 10100
60 NU(NA)=NUNIT:ND(NA)=ND:NB(NA)=NB
70 GOSUB 10200
80 NEXT NA
100 INPUT "WHICH FILE";LF
110 IF EF(LF) THEN 200
120 FOR I=ND(LF) TO ND(LF)+NB(LF)-1:PRINT D(I),:NEXT I:PRINT
130 NUNIT=NU(LF):GOSUB 10200
140 IF EF THEN EF(LF)=-1
150 GOTO 100
200 PRINT "END OF FILE":GOTO 100
```

Source codes for the FORTRAN and BASIC routines will be available in the <WCRDB.PGM> user area of the WHOI VAX. For the FORTRAN, you need DBOPEN.FOR and DBCOM.FOR while the menu-driven routines are DBMENU.FOR, DBMENU2.FOR; for BASIC you need DBOPEN.BAS (Microsoft) .CBM (Commodore 8032) .APP (Apple). The menu routines are DBMENU.xxx with the same suffixes.