ILLINOIS PUBLIC WATER SUPPLY DATA BASE
AND PWS/SEARCH
REFERENCE MANUAL

by

Carl G. Lonnquist

prepared for
Illinois Department of Transportation
Division of Water Resources
Contract No. 1-5-39433
Springfield, Illinois

Champaign, Illinois
December, 1982
# TABLE OF CONTENTS

I. General Information ................................................. 1
   Accessing the Cyber 175. ............................................. 3
   Running the PWS/SEARCH Program ................................. 4
   Line Printer Output .................................................. 4

II. Preparing Tables of Search Results ............................... 6
   SDP command .......................................................... 6
   ADP command .......................................................... 6
   FND command ......................................................... 6
   ADD command .......................................................... 7
   REQ command .......................................................... 8
   PKY command .......................................................... 8
   TAB command .......................................................... 9
   TAL command ......................................................... 9
   CMP command .......................................................... 9
   HED command ........................................................ 10

III. Modifying the Database ........................................... 11
   SID command .......................................................... 11
   ENT command ........................................................ 12
   ALL command ........................................................ 12
   LST command ......................................................... 12
   DEL command ........................................................ 13
   CID command ........................................................ 13
   Making Permanent Changes ......................................... 13
   END command ........................................................ 14

IV. Database Variables ................................................ 15
   Facility variables .................................................... 16
   Community Variables ................................................ 17

V. Program Description ............................................... 18
   Data file structure .................................................. 21

VI. Program Listing .................................................... 24
I. GENERAL INFORMATION

The Illinois Public Water Supply Data Base and (PWS/SEARCH) is a system to store, search for and display data on public water supply facilities compiled by the Department of Transportation, Division of Water Resources (DWR) and supplemented with data from the Illinois State Water Survey (ISWS). The intent of the system is to make the information in the computer files available to the novice user without the need for programming or text editing skills. Access to the data base must be arranged through DWR.

The data-base can be thought of as a large two dimensional array of information. Each row contains information about a single water supply facility or a community. The facilities are assigned an identification number that is a variation of the facility numbers used by the Environmental Protection Agency (IEPA). The community rows are identified by appending a sequence number to the number of the supplying facility. Each column contains a particular item of information identified by a variable name.

The primary method of retrieving information from the system is to generate a table, consisting of subsets of the rows and columns of the data-base. The program maintains two lists that define a table. The display list is set by the user directly and contains the variables of interest to the user. The facility list is set indirectly by the user as he specifies search commands. The user may request simple summary statistics of the numeric variables and full headings for tables with rows that require more than one line to print.
When information about a particular facility or community is desired it can be selected by its identification number. In this mode all of the available information can be displayed and changes to the data-base made. Commands are also provided to create or delete facilities or communities, reassign a community to a different supplying facility, and add, modify or delete data by variable name.

Briefly, the steps required to run the data-base program are:

1. Log on to the Cyber system at the University of Illinois.
2. Access the PWS/SEARCH program.
3. Run the program using data-base commands.
4. Make any changes to the data-base permanent if desired.
5. Log off the Cyber system.
ACCESSING THE CYBER 175

To access the system the user needs the following:

(1) A terminal with modem set at 300 or 1200 Baud and full-duplex.
(2) A telephone.
(3) A valid user number, codeword, and charge number issued by the University of Illinois.

The following steps will log the user on the Cyber 175.

(1) Dial (217)333-4000 for 300 Baud or (217)333-4001 for 1200 Baud.
(2) Insert the handset in the modem.
(3) Press the RETURN key until the computer responds.
(4) After the computer types 'SIGNON:' type your user number.
(5) After the computer types 'PASSWORD:' type your codeword.
(6) after the computer types "CHARGE/RECOVER:' type 'BILL,dept,PSnnnn', substituting the correct department code for 'dept' and the charge number for 'nnnn'.

Note: Each line that the user enters is followed by a carriage return. This signals the computer that you have finished typing and is required after every line.

If all has worked properly the user will be logged on and can enter any Cyber command including those required to run the data-base program. The Cyber prompts for commands by typing a slash(/). The user enters the commands one at a time after a prompt and terminates each line with a carriage return. The timesharing section is ended by signing off with the BYE command.
RUNNING PWS/SEARCH

A copy of the program that accesses the data-base is obtained and is executed by entering the following two commands.

GET,RUN/UN=3PUBUSE/PW=codeword
RUN

If the user is logged in under 3PUBUSE the GET command is not necessary.

When PWS/SEARCH is ready to accept a new instruction it will prompt the user with a question mark. All commands are three characters in length and are typed at the beginning of the line. If the command requires arguments they should follow the command with one intervening space. The command line is terminated with a carriage return.

To stop the program the user enters a null line by pressing the carriage return key before typing any characters after a prompt or by using the END command.

LINE PRINTER OUTPUT

Users can route the output of the program to the local file OUT by using the following form of the RUN command:

RUN,OUT

The user will be prompted for and enter commands as usual but will receive no output at the terminal. After ending the program the file
OUT can be printed by typing:

    PRINT,OUT

Copying this Users Guide

A copy of this Users Guide can be obtained by typing:

    GET,HELP/UN=3PUBUSE
    TYPE,HELP/AS
II. PREPARING TABLES OF SEARCH RESULTS

There are three groups of commands used to generate tables. The first group (SDP, ADP) define the list of display variables (column headings). The second group (FND, ADD, REQ) is used to construct the facility list (rows in the table). The third group (TAB, TAL, CMP, HED) effects the printing of the table.

**SET DISPLAY COMMAND—SDP** ..................................................

SDP variable-1,variable-2,...,variable-n

The SDP command sets the columns for tables to the list of variables typed as arguments on the command line. The variables are typed starting in column five and are separated by commas with no spaces allowed. A total of ten facility and ten community variables can be specified.

**ADD DISPLAY COMMAND—ADP** ..................................................

ADP variable1,variable2,...,variable-n

The ADP command adds variables to the display list.

**FIND COMMAND—FND** .............................................................

FND variable=value
FND variable=value1-value2
The FND command sets the facility list to include all public water supplies that have a column entry equal to a specified value or within a range of values.

FND CNTY=CHAMPAIGN

sets the facility list to all public water supplies in Champaign county.

FND PT2=0-10000

sets the facility list to all public water supplies that serve a population up to and including 10000 based on the 1980 census.

Community variables can also be used. The entry in the facility list is the supplier of the selected community and not the community itself.

FND PSW=500-1000

sets the facility list to all facilities that serve communities with populations between 500 and 1000.

The FND command can be used with alphabetic variables.

FND TOWN=SAVOY

finds the facility that serves the community of Savoy.

ADD COMMAND—ADD

ADD variable=value
ADD variable=valuel-value2

The ADD command adds additional facilities to the facility list. ADD selects facilities in the same way as the FND command but does not clear the list of previously entered facilities.
The two command sequence:

```
FND CNTY=CHAMPAIGN
ADD CNTY=VERMILLION
```

sets the list to all facilities in Vermillion as well as Champaign counties.

REQUIRE COMMAND—REQ

```
REQ variable=value
REQ variable=valuel-value2
```

The REQ command deletes from the facility list any facilities that do not have an appropriate value for the variable.

The three command sequence:

```
FND CNTY=CHAMPAIGN
ADD CNTY=VERMILLION
REQ ATR=1-2
```

sets the facility list to all facilities in Champaign and Vermillion counties that are deficient or marginal in treatment.

PKY COMMAND--PKY

```
PKY
```

The PKY prints a compact list of the facility numbers that are in the facility list.
The TAB command prints a table of the parameters in the display list as column headings and the facilities in the facility list as rows. If no parameters have been specified only the facility numbers will be listed. If community variables are included in the display list, they will be displayed for all communities served by the facilities in the facility list.

WARNING: If the facility list is large the printout could be lengthy. Currently there is no graceful way to stop a long printout.

The TAL command is a combination of the TAB and ALL commands. After constructing a facility list the user can display all the available information about the facilities and associated communities.

The CMP command will turn on or turn off summary statistics at the end of a table. At the start of the program this switch is off.
The minimum, maximum, average, total, and the number of occurrences of all numeric variables are displayed. Missing values are ignored and the printout is suppressed if there is only one value present in the table.

HEADING SWITCH—HED .................................................................

HED

The HED command will turn on or off headings before each line of data displayed by the TAB command. This switch is off at the start of the program. It needs to be set only for multiple line tables that would be hard to read otherwise.
III. MODIFYING THE DATA BASE

The commands in this section refer to the current facility or community. The current facility is the last displayed in a table or referenced in a SID or CID command. The SID command allows the user to change the current facility by entering a new identification number. The LST and ALL commands list information about the current facility. The ENT command adds, changes or deletes individual data items, the CID command changes the identification number, and the DEL command deletes the record completely.

SET ID COMMAND—SID ..........................

    SID nnnnnnnnn
    SID nnnnnnnn/ss

The SID command is used to point to a particular facility or community before using the ALL, LST, or ENT commands.

    SID 10290200
    SID 10290200/3

To point to a facility (as in the first example) the facility number is entered starting in column five. A community is identified by typing the facility number that supplies it followed by a slash (/) and the sequence number.

When the SID command refers to an identifying number not in the data-base a 'RECORD NOT FOUND' message is printed. If the user enters data with the ENT command after this message a new facility or community will be created.
ENTER COMMAND—ENT

ENT variable=value
ENT variable=

After a facility or community has been selected by the SID command, data may be entered or modified with the ENT command. Numeric values can be typed in any format acceptable to FORTRAN. Character string values are entered immediately after the equal sign and may contain blanks.

The second form of the ENT command (without a value) is used to delete data for the specified variable. The variable reverts to missing status.

LIST ALL COMMAND—ALL

ALL

The ALL command lists all the available information about the facility or community selected by the SID command.

LIST COMMAND—LST

LST

The LST command lists only the information in the display list for the facility or community selected by the SID command.
DELETE COMMAND—DEL

DEL

The DEL command deletes the facility or community that has been specified by a SID command. To prevent loss of data the DEL command can be preceded by the ALL command for a complete record of the data deleted.

CHANGE ID COMMAND—CID

CID nnnnnnnnn
CID nnnnnnnnn/ss

The CID command changes the ID number of the data specified by a SID command to the facility or community indicated by the CID command. This command can be used to reassign a community to a different facility.

MAKING CHANGES PERMANENT

The RUN procedure makes copies of the data files for the use of the data-base program. This allows the user to experiment with the ENT, DEL and CID commands (which modify the data) without harming the permanent data files.

If the user has made changes to the data-base, wishes to make them permanent, and is logged in as 3PUBUSE type 'STORE' as a Cyber command
before ending the terminal session. The STORE command should not be used if the program has terminated abnormally.

END COMMAND—END ..............................................................

END

The END command ends execution of the data-base program.
IV. DATABASE VARIABLES

The following table lists the variable names that are used with the database commands, the FORTRAN 77 format and a brief description. If the format begins with an 'A' the variable is a character string with the number of available characters following the 'A'. An 'I' format is a integer followed by the number of spaces reserved to print it. The 'F' format is a numeric format that allows digits to the right of the decimal point. The numbers to the right of the 'F' indicates the total number of spaces reserved for the variable and the number of digits to the right of the decimal point.
### Facility Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM</td>
<td>A31</td>
<td>System Name</td>
</tr>
<tr>
<td>OWNER</td>
<td>A31</td>
<td>Owner</td>
</tr>
<tr>
<td>CNTY</td>
<td>A10</td>
<td>County</td>
</tr>
</tbody>
</table>

**Total Population:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT1</td>
<td>I6</td>
<td>1970 Census</td>
</tr>
<tr>
<td>PTSP</td>
<td>I6</td>
<td>Specific Census</td>
</tr>
<tr>
<td>PT2</td>
<td>I6</td>
<td>1980 Census</td>
</tr>
<tr>
<td>PT3</td>
<td>I6</td>
<td>1990 Estimate</td>
</tr>
<tr>
<td>PT4</td>
<td>I6</td>
<td>2000 Estimate</td>
</tr>
<tr>
<td>PTBE</td>
<td>I6</td>
<td>Breakeven</td>
</tr>
</tbody>
</table>

**Pumpage:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYR</td>
<td>I2</td>
<td>Current Year</td>
</tr>
<tr>
<td>CPU</td>
<td>F8.3</td>
<td>Average daily MGD (with Dec.)</td>
</tr>
<tr>
<td>OCYR</td>
<td>I2</td>
<td>Old Current Year</td>
</tr>
<tr>
<td>OCPU</td>
<td>F8.3</td>
<td>Old Average daily MGD</td>
</tr>
<tr>
<td>CD</td>
<td>I3</td>
<td>GPCD</td>
</tr>
<tr>
<td>FYR</td>
<td>I4</td>
<td>Projection Year</td>
</tr>
<tr>
<td>FPU</td>
<td>F8.3</td>
<td>Average Daily MGD (with Dec.)</td>
</tr>
<tr>
<td>FD</td>
<td>I4</td>
<td>GPCD</td>
</tr>
<tr>
<td>BEYR</td>
<td>I4</td>
<td>Breakeven Year</td>
</tr>
<tr>
<td>BEPU</td>
<td>F8.3</td>
<td>Average Daily MGD (with Dec.)</td>
</tr>
<tr>
<td>BED</td>
<td>I4</td>
<td>GPCD</td>
</tr>
<tr>
<td>PUT</td>
<td>F8.2</td>
<td>Pumping Time (with Dec.)</td>
</tr>
<tr>
<td>CAP</td>
<td>F8.2</td>
<td>Plant Capacity MGD (with Dec.)</td>
</tr>
</tbody>
</table>

**Storage:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>F8.4</td>
<td>Ground MG (with Dec.)</td>
</tr>
<tr>
<td>SE</td>
<td>F8.3</td>
<td>Elevated MG (with Dec.)</td>
</tr>
<tr>
<td>ST</td>
<td>F8.3</td>
<td>Total MG (with Dec.)</td>
</tr>
</tbody>
</table>

**Assessment:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASO</td>
<td>I1</td>
<td>Source *</td>
</tr>
<tr>
<td>MGDS</td>
<td>F8.3</td>
<td>MGD</td>
</tr>
<tr>
<td>ATR</td>
<td>I1</td>
<td>Treatment *</td>
</tr>
<tr>
<td>MGDT</td>
<td>F8.3</td>
<td>MGD</td>
</tr>
<tr>
<td>AST</td>
<td>I1</td>
<td>Storage *</td>
</tr>
<tr>
<td>MGST</td>
<td>F8.3</td>
<td>MG</td>
</tr>
<tr>
<td>AWD</td>
<td>I1</td>
<td>Withdrawal *</td>
</tr>
<tr>
<td>MGDW</td>
<td>F8.3</td>
<td>MGD</td>
</tr>
</tbody>
</table>
* Deficient—1
Marginal—2
Adequate—3
Breakeven—4

Source:

TYPE A4 Type of System **

** Groundwater—GRND
Surfacewater—SURF
Combined—GRSF

Groundwater Systems:

#W I2 Number of Wells
Q I8 Total Discharge
YLD F8.2 Aquifer Yield MGD (with Dec.)
AQDIS A72 Aquifer Description

Surface Water Systems:

SOURCE A36 Source
RESNAME A40 Impounding Reservoir Name
YR I4 Year
RESCAP A10 Reservoir Capacity
RESSA A10 Surface area
SEDRATE A8 Annual Sediment Rate
DA A10 Drainage Area
RESYLD A6 Reservoir Yield

COMMUNITY VARIABLES

Variable Format Description

TOWN A30 Community Name
P1 I6 1970 Census
PSP I6 Specific Census
P2 I6 1980 Census
P3 I6 1990 Estimate
P4 I6 2000 Estimate
PBE I6 Breakeven
V. PROGRAM DESCRIPTION AND LISTING

As an aide to reading the FORTRAN source code a brief description of the subroutine entry points follows.

START

START constructs the file information table, FIT, with a call to FILEIS and opens the file checking the error status. The variable information, VAR, is read from the record with key equal to 1.

INDEX

INDEX set up the file for reads from the alternate index file, QFILEA, by setting the keyword (RKW) to 1, the key position within that word (RKP) to 0 and the index switch to 'YES'. Word and character counts start from zero so the key location is in the first trailer record. This implies that the corresponding fields in all trailer records are included in the index. This allows for a variable record size and indexed data beyond the minimum record size and is the reason trailer count records were chosen for the data-base.

DATA

DATA set the file to read data from QFILE by setting the keyword to zero and the relative key position to 10. This is an impossible value and the code for a nonimbeded primary key.
FINISH

FINISH calls SID with a zero key to cause the writing to the file of any pending changes to the data, closes the file and stops the program.

Facility List Maintenance

Subroutine ADD decodes the 'variable=value' part of the FND, ADD and REQ commands and calls LOAD to add the selected facility numbers to the facility list. IVAR is searched for the variable name to determine the variable number and the data type. The start and the end of the retrieval range is constructed in the same 1 word format as used for data storage. If the data type is character string and a range is given on the command line the start of the range is truncated to the number of characters before the separating minus sign. The remaining characters are binary zero which sort low. The end of the range is blank filled which sorts high. This allows major key retrievals from the alternate index file without special programming.

LOAD positions the index file to the first value equal or greater than the start of the range. Primary keys values are retrieved in blocks of 200 until the end of the range is exceeded. The community sequence number is masked out and bit 1 (counting from 0 at the right) is set to indicate a new arrival in the facility list. If 200 positions are not available, PACK is called to condense the list.

PACK combines duplicate facility numbers with an inclusive or on the low order flag bits. Bit position 0 is set by MARK and indicates the
facility was in the list before the current command. Bit 1 is set for numbers referenced by the current command. For the FND and ADD commands this distinction is ignored and all the facilities are included. The REQ command calls the subroutine REQ which eliminates all entries that have either of these bits clear.

SETID

The subroutine SETID manages the reading and writing of data records. The data for one facility or community is read and held until a request with a different identification number is made. The SID command calls this routine directly and the table commands make a series of calls to retrieve data for each row of information displayed.

Before reading new records SETID replaces records that have been modified by the ENT command. As each record is read the data is copied to the array IDATA by variable number for easy access by the .LIST subroutine.

The entry point STORE is called by the ENT command and modifies the data in IDATA and scans the trailer records in the REC array for the corresponding entries. Trailer records are added and deleted as necessary and the record flagged for replacement. The entry DELETE removes the current pair of data records from the file and sets KEY to zero indicating no valid data in memory. NEW reads the key of the next record in sequential order from the file and is used by the main program to determine if more communities are associated with a facility.
LIST

LIST formats and displays tabular data for the LST, ALL, TAB and TAL commands. Data for the current facility or community is read from the array IDATA filled by the SETID routine. The value of MRK determines the selection of variables for display and whether headings are to be used. Summary statistics are accumulated if CMP is set.

For the LST and TAB commands the ID of the current record is checked to select the correct display list and the order of the variables in the table is fixed. When processing ALL and TAL commands the variable definition list IVAR is scanned for variable entries for which data is available in IDATA. For each selected variable the data type and format is picked out of the low order bits of the variable definition. If there is not enough room on the current line it is printed. For numeric data a format string is constructed and the data written on the output line.

The entry CMPON toggles the summary statistics switch, CLRCMP clears the arrays used for the summary and LISTCMP prints the results.

DATA FILE STRUCTURE

The data are stored in an Extended Multiply Indexed Sequential file as described in

Cyber Record Manager
Advanced Access Methods
Version 2
Reference Manual
The data file, with local file name QFILE, contains the information about the facilities and communities in the system and the variable definitions. The records in the file can be written, read, replaced, or deleted directly with Record Manager calls when the facility number/community seq. no.) is known. The TAB, SID, ENT, CID, and DEL commands read or manipulate records in this file.

The alternate index file, QFILEA, is an inverted file maintained by the Record Manager software. For variable/value pair entered in the data-base a list of the records that contain the pair is kept. During execution of the FND, ADD, and REQ commands this file is read and the facility number embedded in the record key used to construct the facility list.

Primary Key Format

The primary key for records in QFILE is a 60 bit integer. It is composed of three binary fields.

1) Facility Number  bits 10-58,
2) Community Seq. No. bits 3-9,
3) Record Seq. No. bits 0-2.

With zero reserved for the supplying facility there can be up to 127 communities associated with one facility. The record sequence number is set to zero for records with indexed data and to one for unindexed data records.
Record Format

The data are stored in trailer count records with a 10 character word) header and trailer length. The alternate key control character is in the first character position and the trailer count in the last three positions of the header word. If the control character is 'K' the data items contained are indexed in the alternate index file.

Trailer Record Format

Each trailer record holds one variable number/value pair and has three fields:
1) Variable number bits 48-60,
2) Toggled sign bit bit 47,
3) Data field bits 0-47.

Numeric data are stored in normalized floating point form with the low order 12 bits discarded. What remains is a 12 bit sign and an exponent and 36 bits of precision (approx. 10 decimal digits). Alphabetic strings are stored 8 6-bit characters per word.

The high order bit of the data field (sign bit) is toggled so that positive and negative numeric data will sort in the proper order. All alphabetic characters in display code have their high bit clear. Alphabetic strings that are left justified and begin with A-Z will sort in display code order.
VI. PROGRAM LISTING

.PROC, DB.
* FEBRUARY 16, 1983
8
* ILLINOIS STATE WATER SURVEY
* P. O. Box 5050 Station A
* Champaign, IL 61820
* Attn: Carl Lonnquist
* (217) 333-4968
FTN5, I=SOR, L=0, REW, OPT.
REWIND, LIB, RIP.
FILE(QFILE, FO=IS, ORG=NEW, RT=T, HL=10, TL=10, CP=7, CL=3, MNR=10)
FILE(QFILE, MRL=2000, KT=I, KL=10, XN=QFILEA)
LDSET(FILES=QFILE)
$LOAD, LIB.
$LOAD (LG0)
NOGO (RIP)
* DATA.SOR.

PROGRAM DB(INPUT=65, OUTPUT, TAPE7=OUTPUT)
CHARACTER CMD*100, CM*3, TMPP*20
COMMON /KEYS/KMAX, NKEY, KEYS(1199)
COMMON /FILE/FIT(35), IOPN, DUMMY(0:128), ID
COMMON /DSP/NV(2), NFS(2,10)
MRKA=0
CALL START

1 READ(*, '(A)', END=999) CMD
CM=CMD(:,3)

2 IF(CM.EQ.'SID') THEN
* FIND SEPERATOR
   NN=INDEX(CMD, '/')
   IF(NN.EQ.0) NN=20
* MOVE FACILITY AND COMMUNITY FIELD TO FIXED LOCATIONS
   TMPP(1:10)=CMD(4:NN-1)
   TMPP(11:) = CMD(NN+1:)
* READ VALUES AND CONSTRUCT KEY
   READ(TMPP, '(2I10)', ERR=900) KK, L
   K=SHIFT(KK, 10)+SHIFT(L, 3)
* READ RECORDS
   CALL SETID(K, FAIL)
   IF(FAIL.NE.0) PRINT*, KK, '/', L, ' RECORD NOT FOUND'
   GO TO 1
ENDIF

IF(CM.EQ.'ADD') THEN
* CLEAR CURRENT RECORDS
CALL SETID(O, FAIL)
* ADD TO FACILITY LIST

    CALL ADD(CMD(5:))

* REPORT NUMBER OF FACILITIES

    CM='STA'
    GO TO 2
ENDIF

IF(CM.EQ.'FND')THEN

    * CLEAR FACILITY LIST
        NKEY=0

    * CHANGE TO ADD COMMAND
        CM='ADD'
        GO TO 2
ENDIF

IF(CM.EQ.'STA')THEN

    PRINT*,NKEY,' FACILITIES IN LIST'
    GO TO 1
ENDIF

IF(CM.EQ.'PKY')THEN

    READ(CMD(5:),'(I10)')L
    L=MINO(L,NKEY)
    IF(L.EQ.0)L=NKEY
    PRINT*,(I,KEYS(I)/2**10,I=1,L)
    GO TO 1
ENDIF

IF(CM.EQ.'REQ')THEN

    IF(NKEY.EQ.0)THEN
        PRINT*,'FIC LIST EMPTY'
        GO TO 1
    ENDIF

    * SET BIT O'S IN FACILITY LIST
        CALL MARK

    * CLEAR CURRENT RECORD
        CALL SETID(O,FAIL)

    * ADD TO FACILITY LIST
        CALL ADD(CMD(5:))

    * DELETE ENTRIES WITHOUT BITS 0 & 1 SET
        CALL REQ
        CM='STA'
        GO TO 2
ENDIF

IF(CM.EQ.'SDP')THEN

    CALL SETDSP(CMD(5:))
    GO TO 1
ENDIF

IF(CM.EQ.'ADP')THEN
CALL ADDDSP(CMD(5:))

GO TO 1
ENDIF

IF(CM.EQ.'LST')THEN
    CALL LIST(l)
    GO TO 1
ENDIF

IF(CM.EQ.'ALL')THEN
    CALL LIST(-l)
    GO TO 1
ENDIF

IF(CM.EQ.'TAB')THEN
    MRK=1
    * CLEAR COMPUTE ARRAY
    CALL CLRCMP
    DO 80 I=1,NKEY
    * READ FACILITY RECORDS
    KKK=KEYS(I).AND.MASK(50)
    CALL SETID(KKK,FAIL)
    * PRINT FACILITY VARIABLES
    CALL LIST(MRK)
    * IF COMMUNITY VARIABLES REQUESTED
    IF(NV(2).GT.0)THEN
        DO 60 II=1,127
        * CHECK FOR MORE COMMUNITIES
        CALL NEXT(KK)
        IF(KKK.NE.(MASK(50).AND.KK))GO TO 70
        CALL SETID(KK,FAIL)
        IF(FAIL.NE.0.0)GO TO 70
        * IF FIRST COMMUNITY REQUIRE HEADINGS
        IF(II.EQ.1)THEN
            MRK=1
        ELSE
            MRK=MRKA
        ENDIF
        60 CALL LIST(MRK)
    70 IF(II.EQ.1)THEN
        MRK=MRKA
    ELSE
        MRK=1
    ENDIF
    ELSE
        MRK=MRKA
    ENDIF
    80 CONTINUE
    * PRINT SUMMARY
    CALL LISTCMP
    GO TO 1
ENDIF

IF(CM.EQ.'ENT')THEN
    CALL STORE(CMD(5:))
    GO TO 1
ENDIF

IF(CM.EQ.'CMP')THEN
    CALL CMPON
    GO TO 1
ENDIF

IF(CM.EQ.'HED')THEN
    IF(MRKA.EQ.0)THEN
        MRKA=1
        PRINT*, 'FULL HEADINGS'
    ELSE
        MRKA=0
        PRINT*, 'PARTIAL HEADINGS'
    ENDIF
    GO TO 1
ENDIF

IF(CM.EQ.'CID')THEN
    * DECODE KEY
    NN=INDEX(CMD,'/')
    IF(NN.EQ.0)NN=20
    TMPP(1:10)=CMD(4:NN-1)
    TMPP(11:)=CMD(NN+1:)
    READ(TMPP,'(2110)',ERR=900)KK,L
    K=SHIFT(KK,10)+SHIFT(L,3)
    IF(K.LT.100)THEN
        PRINT *, 'BAD ID'
        GO TO 1
    ENDIF
    * DELETE RECORDS UNDER OLD ID
    CALL DELETE
    * SET NEW ID
    CALL NEWID(K)
    GO TO 1
ENDIF

IF(CM.EQ.'DEL')THEN
    CALL DELETE
    GO TO 1
ENDIF

IF(CM.EQ.'TAL')THEN
    DO 180 I=1,NKEY
        * READ FACILITY RECORDS
        KKK=KEYS(I).AND.MASK(50)
        CALL SETID(KKK,FAIL)
    180 CONTINUE
ENDIF
* PRINT FACILITY VARIABLES

CALL LIST(-1)
DO 160 II=1,127
* CHECK FOR MORE COMMUNITIES
CALL NEXT(KK)
IF(KKK.NE.(MASK(50).AND.KK))GO TO 180
CALL SETID(KK,FAIL)
IF(FAIL.NE.0.0)GO TO 180
160 CALL LIST(-1)
180 CONTINUE
GO TO 1
ENDIF
IF(CM.EQ.'END')GO TO 999
PRINT*,'UNKNOWN COMMAND-',CMD
GO TO 1
900 PRINT*,'BAD ID VALUE'
GO TO 1
* WRITE MODIFIED RECORDS AND CLOSE FILE
999 CALL FINISH
END
.PROC.USLIB.
FTN5, I=SOR,L=0,B=LIB,REW,OPT=3.
.DATA.SOR.

SUBROUTINE START

COMMON /FILE/FIT(35),IOPN,DUMMY(0:128),ID
COMMON /KEYS/KMAX,NKEY,EYES(1199)
COMMON /VAR/Z,VAR(128)
KMAX=1199
ID=0
CALL FILEIS(FIT,L"LFN",L"QFILE")
CALL OPENM(FIT,L"I-0")
IOPN=1
IF(IFETCH(FIT,L"ES").NE.0)THEN
  PRINT *,'UNABLE TO OPEN'
  STOP
ENDIF
NKEY=0
CALL GET(FIT,Z,1)
RETURN

ENTRY INDEX
CALL STOREF(FIT,L"RKP",0)
CALL STOREF(FIT,L"RKW",1)
CALL STOREF(FIT,L"NDX",L"YES")
IOPN=2
RETURN

ENTRY DATA
CALL STOREF(FIT,L"RKP",10)

CALL STOREF(FIT,L"RKW",0)
CALL STOREF(FIT,L"NDX",L"NO")
I0PN=1
RETURN

ENTRY FINISH
CALL SETID(O,FAIL)
CALL CLOSEM(FIT)
STOP
END

SUBROUTINE REQ

COMMON /KEYS/KMAX,NKEY,KEYS(1199)
IF(NKEY.EQ.0)RETURN
K=0
DO 30 I=1,NKEY
IF((KEYS(I).AND.3).EQ.3)THEN
  K=K+1
  KEYS(K)=KEYS(I)-2
ENDIF
30 CONTINUE
NKEY=K
RETURN
ENTRY MARK
DO 40 I=1,NKEY
40 KEYS(I)=(KEYS(I).AND.MASK(50))+1
RETURN
END

SUBROUTINE SORT(K,N)

DIMENSION K(*)
1 MOD=0
DO 20 I=2,N
IF(K(I-1).GT.K(I))THEN
  KK=K(I)
  K(I)=K(I-1)
  K(I-1)=KK
  MOD=1
ENDIF
20 CONTINUE
IF(MOD.NE.0)GO TO 1
END

SUBROUTINE ADD(CMD)
CHARACTER CMD(*)
COMMON /VAR/DUMMY, IVAR(128)
K=INDEX(CMD, '=')
IF(K.LE.1)RETURN
* LOOK UP VARIABLE NAME
L=BOOL(CMD(:K-1))
DO 10 I=1,128
IF(((IVAR(I).XOR.L).AND.MASK(48)).EQ.0)GO TO 20
10 CONTINUE
PRINT *, 'VARIABLE NOT FOUND-', CMD(:K-1)
RETURN
20 KK=INDEX(CMD(K+1:), '-')
IF(KK.EQ.0)THEN
   LOW=CMD(K+1:)
   HIGH=LOW
   NBIT=48
ELSE
   LOW=CMD(K+1:K-1+KK)
   HIGH=CMD(K+KK+1:)
   NBIT=MIN0(48, 6*(KK-1))
ENDIF
IF(((IVAR(I).AND.SHIFT(1,11)).NE.0)THEN
   NUMERIC VARIABLE
   READ(LOW, '(F20.0)', ERR=30)X1
   READ(HIGH, '(F20.0)', ERR=30)X2
   CALL LOAD(I, X1, X2)
   RETURN
ELSE
   CALL LOAD(I, (BOOL(LOW).AND.MASK(NBIT)), BOOL(HIGH))
ENDIF
RETURN
30 PRINT *, 'BAD NUMERIC FIELD-', CMD
RETURN
END

SUBROUTINE LOAD(IV, KL, KH)
INTEGER HIGH
COMMON /KEYS/KMAX, NKEY, KEYS(1199)
COMMON /FILE/FIT, IOPN, IDATA(0:128), DUMMY
KEY(I, IX)=SHIFT(((MASK(48).AND.(MASK(1).XOR.IX)).OR.I), 48) •
CALL INDEX
LOW=KEY(IV, KL)
HIGH=KEY(IV, KH)
CALL STOREF(FIT, L'REL', L'GE')
CALL STARTM(FIT, LOW)
CALL STOREF(FIT, L'REL', L'GT')
1 IF(KMAX-NKEY.LT.200)THEN
   CALL PACK
IF (KMAX-NKEY.LT.200) THEN
    PRINT *, 'KEY BUFFER OVERFLOW'
    RETURN
ENDIF
CALL GETN(FIT, KEYS(NKEY+1), HIGH)
N=IFETCH(FIT, L"PTL")
DO 2 I=1,N
2 KEYS(NKEY+I)=(KEYS(NKEY+I).AND.MASK(50))+2
NKEY=NKEY+N
IFP=IFETCH(FIT, L"FP")
IF (IFP.EQ.0) GO TO 1
CALL PACK
CALL STOREF(FIT, L"REL", L"EQ")
RETURN
END

SUBROUTINE SETID(NKEY, FAIL)
INTEGER REC(0:200,3), NP(3), NMOD(3)
CHARACTER B*10, STR(*), BUF*100
COMMON /FILE/FIT(35), IOPN, IDATA(0:128), KEY
COMMON /VAR/Z, IVAR(128)
DATA NP,NMOD,REC/609*0/
* SETUP FILE FOR DATA
CALL DATA
FAIL=0
IF (NKEY.EQ.KEY) RETURN
IF (KEY.NE.O) THEN
* CHECK IF RECORDS ARE MODIFIED
DO 10 I=1,2
10 IF (NMOD(I).NE.O) THEN
* CONSTRUCT KEY
LKEY=KEY+I-1
* WRITE HEADER RECORD
WRITE(B,'(I10)')NP(I)
* IF KEYED RECORD SET CONTROL CHARACTER
IF ((LKEY.AND.1).EQ.0) B(1:1)='K'
REC(0,I)=BOOL(B)
* ATTEMPT REPLACE
CALL REPLC(FIT, REC(0,I), 0, LKEY)
IF (IFETCH(FIT, L"ES").NE.0) THEN
* IF REPLACE FAILS USE PUT
CALL PUT(FIT, REC(0,I), 0, LKEY)
IF (IFETCH(FIT, L"ES").NE.0) PRINT *, BOOL(IFETCH(FIT, L"ES"))
ENDIF
* CLEAR MODIFIED FLAG
NMOD(I)=0
ENDIF
CONTINUE

ENDIF

IF (NKEY.EQ.0) RETURN
* SET NEW KEY
KEY=NKEY
*CLEAR DATA ARRAY
  DO 15 I=0,128
15 IDATA(I)=L" "
   FAIL=1.0
* READ NEW RECORDS
  DO 20 I=1,2
     LKEY=KEY+I-1
     NMOD(I)=0
     NP(I)=0
     CALL GET(FIT,REC(0,I),LKEY)
     IF (IFETCH(FIT,L"ES").EQ.0) THEN
       FAIL=0
* READ TRAILER RECORD COUNT
     WRITE(B,'(A10)')REC(0,I)
     READ(B,'(1X,I9)')NP(I)
* TRANSFER DATA TO ARRAY
  19 K=1,NP(I)
* ROTATE VARIABLE TO LOW BITS
* AND DATA TO HIGH BITS
* TOGGLE HIGH ORDER DATA BIT
* MASK VARIABLE NUMBER AND TRANSFER
19 IDATA((K.Z.AND.O"177")=K.Z.XOR.MASK(1)
ENDIF
20 CONTINUE
RETURN

ENTRY DELETE
IF (KEY.LT.100) THEN
   PRINT *, 'BAD ID'
   RETURN
ENDIF
CALL DLTE(FIT,KEY)
CALL DLTE(FIT,KEY+1)
KEY=0
RETURN

ENTRY NEWID(NKEY)
KEY=NKEY
  DO 25 I=1,2
25 NMOD(I)=1
RETURN

ENTRY NEXT(NKEY)
CALL GETN(FIT,REC(0,3),NKEY)
ENTRY STORE(STR)
BUF=STR
* FIND EQUAL SIGN
N=INDEX(BUF,'=')
IF(N.EQ.0)RETURN
* PULL OUT VARIABLE NAME
KVAR=BOOL(BUF(:N-1))
* LOOKUP VARIABLE NAME
DO 30 I=1,128
IF(((KVAR.XOR.IVAR(I)).AND.MASK(48)).EQ.0)GO TO 40
30 CONTINUE
PRINT *, 'VARIABLE NOT FOUND'
RETURN
* DETERMINE IF ALPHA OR NUMERIC
40 IF((SHIFT(IVAR(I),-11).AND.1).EQ.1)THEN
   READ(BUF(N+1:),'F20.0)',ERR=45) IDATA(I)
   GO TO 50
45 PRINT *, 'BAD NUMERIC VALUE'
   RETURN
I2=I
ELSE
* DETERMINE NUMBER OF ALPHA WORDS
   I2=(SHIFT(IVAR(I)-O"20",-7).AND.O"17")+I
   READ(BUF(N+1:),'(10A8)') (IDATA(K),K=I,I2)
ENDIF
* COPY MODIFIED DATA TO REC ARRAYS
DO 70 K=I,I2
* SET RECORD NUMBER
   IC=(IVAR(K).AND.3)+1
* SCAN RECORD FOR THE VARIABLE
   DO 60 KK=1,NP(IC)
      IF(K.EQ.(SHIFT(REC(KK,IC),12).AND.O"177"))GO TO 65
60 CONTINUE
* IF NOT FOUND ADD AT END
   NP(IC)=NP(IC)+1
65 REC(KK,IC)=SHIFT(((IDATA(K).XOR.MASK(1)).AND.MASK(48))+K,48)
   IF(IDATA(K).EQ.1)THEN
      " OR.IDATA(K).EQ.MASK(60))THEN
* IF DATA BLANK OR MISSING DELETE TRAILER
   REC(KK,IC)=REC(NP(IC),IC)
   NP(IC)=NP(IC)-1
   IDATA(K)=L" 
ENDIF
* SET MODIFIED FLAG
   NMOD(IC)=1
70 CONTINUE
RETURN
END
SUBROUTINE PACK

COMMON /KEYS/KMAX,NKEY,KEYS(1199)
IF(NKEY.EQ.0)RETURN
K=1
CALL SORT(KEYS,NKEY)
DO 10 I=1,NKEY
  * CHECK FOR EQUAL FACILITY FIELD
  IF(((KEYS(K).XOR.KEYS(I)).AND.MASK(50)).EQ.0)THEN
    KEYS(K)=KEYS(K).OR.KEYS(I)
  ELSE
    K=K+1
    KEYS(K)=KEYS(I)
  ENDIF
10 CONTINUE
NKEY=K
RETURN
END

SUBROUTINE SETDSP(CMD)

CHARACTER CMD*(*)
COMMON /DSP/NV(2),NFS(2,10)
COMMON /VAR/Z,IVAR(128)
* CLEAR DISPLAY LIST
NV(1)=NV(2)=0
ENTRY ADDDSP(CMD)
K=1
L=LEN(CMD)
10 IF(K.GE.L)RETURN
  IF(CMD(K:).EQ.' ')RETURN
  * SCAN FOR SEPERATOR
  N=INDEX(CMD(K:),',',')
  IF(N.EQ.0)THEN
    N=L
  ELSE
    N=K+N-2
  ENDIF
  KV=BOOL(CMD(K:N))
  * LOOKUP VARIABLE NAME
  DO 30 I=1,128
  IF(((KV.XOR.IVAR(I)).AND.MASK(48)).EQ.0)GO TO 40
30 CONTINUE
  PRINT *, 'VARIABLE NOT FOUND-',CMD(K:N)
  K=N+2
  GO TO 10
40 IT=1+(SHIFT(IVAR(I),-3).AND.1)
  IF(NV(IT).LT.10)THEN
* ENTER VARIABLE NUMBER IN LIST

\[
\begin{align*}
\text{NV}(\text{IT}) &= \text{NV}(\text{IT}) + 1 \\
\text{NFS}(\text{IT}, \text{NV}(\text{IT})) &= 1
\end{align*}
\]

ELSE

\[
\begin{align*}
\text{PRINT}^{*}, & '\text{TOO MANY DISPLAY VARIABLES}' \\
\text{RETURN}
\end{align*}
\]

ENDIF

K = N + 2
GO TO 10
END

SUBROUTINE LIST(MRK)

* MRK = -1 LIST ALL VARIABLES WITH HEADINGS
* MRK = 0 LIST DISPLAY VARIABLES—NO HEADINGS
* MRK = 1 LIST DISPLAY VARIABLES WITH HEADINGS

CHARACTER DLINE*(100), TITLE*(100), TMP*10
COMMON /FILE/FIT(35), IOPN, DATA(0:128), ID
COMMON /VAR/Z, IVAR(128)
COMMON /DSP/NV(2), NFS(2,10)
DIMENSION NOP(2,10), TOT(2,10), PMIN(2,10), PMAX(2,10)
DATA CMP/0.0/
DATA NOP/20*0/

NLIM=79
IT=13
ILST=1
IF (MRK.NE.0) TITLE=''
DLINE='

* WRITE FACILITY NO.
WRITE(DLINE(1:8),'(I8)') SHIFT(ID,-10)
* WRITE COMMUNITY SEQ. NO.
IF ((SHIFT(ID,-3).AND.O"77").NE.0) THEN
    ILST=2
    WRITE(DLINE(9:11),'(1H/,I2.2)')(O"77".AND.SHIFT(ID,-3))
ELSE
    IF (MRK.NE.0) TITLE(1:8)='FACILITY'
ENDIF

* SCAN LIST OF DISPLAY VARIABLES
NVV=NV(ILST)
IF (MRK.LT.0) THEN

* LIST ALL VARIABLES
NVV=128
CMP=0
ENDIF
DO 500 I=1,NVV
IF (MRK.LT.0) THEN
    IV=I
    KZZ=IVAR(IV)
ENDIF
DO 500 I=1,NVV
IF (MRK.LT.0) THEN
    IV=I
    KZZ=IVAR(IV)
ENDIF
DO 500 I=1,NVV
IF (MRK.LT.0) THEN
    IV=I
    KZZ=IVAR(IV)
ENDIF

**SKIP VACANT, CONTINUATION, AND MISSING VARIABLES**

```fortran
IF((KZZ.AND.MASK(48)).EQ.L"VACANT ")GO TO 500
IF((KZZ.AND.MASK(48)).EQ.L"CONT. ")GO TO 500
IF(DATA(I).EQ.L")GO TO 500
IF(I+(SHIFT(KZZ,-3).AND.1).NE.ILST)GO TO 500
ELSE
    IV=NFS(ILST,I)
    KZZ=IVAR(IV)
ENDIF
IF((SHIFT(KZZ,-11).AND.1).EQ.1)THEN
    * PICK NUMERIC FIELD LENGTH & DECIMAL CNT.
    NCF=SHIFT(KZZ,-7).AND.O"17"
    ND=SHIFT(KZZ,-4).AND.O"7"
    * DETERMINE OFFSET TO RIGHT JUSTIFY NUMERIC LABELS
    DO 400 IS=1,7
        IF(((SHIFT(KZZ,6*IS).XOR.L" ").AND.MASK(6)).EQ.0)GO TO 410
    400 CONTINUE
    410 NCF=MAXO(NCF,IS)
    IS=NCF-IS
ELSE
    * PICK ALPHA STRING LENGTH
    IS=0
    ND=-1
    NCF=SHIFT(KZZ,-4).AND.O"177"
ENDIF
* IF NO ROOM ON CURRENT LINE-PRINT
IF(IT+NCF-1.GT.NLIM)THEN
    IF(MRK.NE.0)WRITE(7,'(A79)')TITLE
    WRITE(7,'(A79)')DLINE
    DLINE=' '
    IF(MRK.NE.0)TITLE=' '
    IT=8
ENDIF
* INSERT VAR. NAME IN HEADING
IF(MRK.NE.0) WRITE(TITLE(IT+IS:IT+7+IS),'(A8)')KZZ
IF(ND.GE.0)THEN
    IF((DATA(IV).XOR.L")")NE.0)THEN
        IF(CMP.NE.0)THEN
            IF(NOP(ILST,I).EQ.0)THEN
                TOT(ILST,I)=PMIN(ILST,I)=PMAx(ILST,I)=DATA(IV)
            ELSE
                TOT(ILST,I)=TOT(ILST,I)+DATA(IV)
                PMIN(ILST,I)=AMIN1(PMIN(ILST,I),DATA(IV))
                PMAx(ILST,I)=AMAX1(PMAx(ILST,I),DATA(IV))
            ENDIF
            NOP(ILST,I)=NOP(ILST,I)+1
        ENDIF
        * CONST. FORMAT AND WRITE FIELD
        IF(ND.GT.0)THEN
```

This code snippet is a part of a larger FORTRAN program, which appears to be handling data fields, skipping vacant and continuation variables, and managing the formatting and printing of these fields. It includes logic for determining the length of numeric fields, calculating offsets for right-justifying numeric labels, and handling alpha string lengths. The code also includes logic for printing data when there is no room on the current line, inserting variable names in headings, and calculating summary statistics such as total, minimum, and maximum values.
WRITE(TMP,'(2H(F,I2.2,1H.,I1,1H))',ERR=999)NCF,ND
WRITE(DLINE(IT:IT+NCF),TMP,ERR=999)DATA(IV)
ELSE
WRITE(TMP,'(2H(I,I2.2,1H))',ERR=999)NCF
WRITE(DLINE(IT:IT+NCF),TMP,ERR=999)IFIX(DATA(IV))
ENDIF
ENDIF
ELSE
* CAL. NO. OF ALPHA WORDS IN STRING
NW=SHIFT(NCF-1,-3)+IV
WRITE(DLINE(IT:IT+NCF+10),'(10A8)')(DATA(II),II=IV,NW)
ENDIF
* SET NEXT AVAILABLE CHAR. POS. IN LINE
IT=IT+NCF+1
500 CONTINUE
998 IF(MRK.NE.0)WRITE(7,'(A79)')TITLE
WRITE(7,'(A79)')DLINE
RETURN
999 PRINT *,'ERROR IN FORMATING'
PRINT *,NCF,ND,'FIELD SIZES'
PRINT*,I,IV.AND.MASK(0),TMP,DATA(IV).AND.MASK(0)
GO TO 998
ENTRY CLRCMP
IF(CMP.EQ.0.0)RETURN
DO 600 ILST=1,2
DO 600 I=1,NV(ILST)
600 NOP(ILST,I)=0
RETURN
ENTRY LISTCMP
IF(CMP.EQ.0.0)RETURN
LAB=0
DO 700 ILST=1,2
DO 700 I=1,NV(ILST)
IF(NOP(ILST,I) .GT.DTHEN
IF(LAB.EQ.0)THEN
LAB=1
WRITE(7,710)' VARIABLE NO TOTAL MINIMUM MAXIMUM'
1 ' AVERAGE'
710 FORMAT(/2A)
ENDIF
WRITE(7,720)IVAR(NFS(ILST,I)),NOP(ILST,I),TOT(ILST,I),PMIN(ILST,I)
1 ,PMAX(ILST,I),TOT(ILST,I)/NOP(ILST,I)
720 FORMAT(1X,A8,I5,4E11.4)
ENDIF
700 CONTINUE
RETURN
ENTRY CMPON

IF (CMP.EQ.0.0) THEN
  CMP=1.0
  PRINT*,'CMP ON'
ELSE
  CMP=0.0
  PRINT*,'CMP OFF'
ENDIF
RETURN
END