UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

EXTRACT: A FORTRAN Program to Retrieve Selected Well History Control System (WHCS) Data

By

Suzanne I. Weston and Thaddeus S. Dyman

Open-File Report 81-29

1981

TABLE OF CONTENTS

INTRODUCTION

The Well History Control System (WHCS) is a large digital data file containing geological and engineering data on more than one million oil and gas wells drilled in the United States. Specific data categories include:

1. well location, identification, and classification,
2. initial potential and production tests,
3. formation tops and bases,
4. core data,
5. drill stem and wireline tests,
6. miscellaneous data including logs, shows, casing description, and hole deviation.

The WHCS file is owned and maintained by Petroleum Information Corporation (PI) of Denver, Colorado, and is directly accessed by the U.S. Geological Survey through a contract with PI.

Applications to WHCS within the U.S. Geological Survey include:

1. resource appraisal and basin evaluation
2. field and reservoir engineering and production problems,
3. regional stratigraphic correlation, and
4. special aquifer and formation studies.

The WHCS file is being used extensively throughout the U.S. Geological Survey, but no large scale effort has yet been made to process this data on Survey computers. To best retrieve and manipulate WHCS data for printed and graphic output, an efficient database management system and related processing routines are required.

As a first step in that direction, EXTRACT was written to selectively retrieve and output stratigraphic and production data for the state of Colorado from WHCS on the Honeywell MULTICS System. EXTRACT is a FORTRAN IV program and, with the exception of calls to five system routines, is system independent.

The purpose of this report* is to present EXTRACT for continued use and modification, and describe the level of programming effort and computer time required for data retrieval. Included with the program code and documentation are program expansion and modification guidelines, a sort parameter analysis, and a sample processing cost analysis.

*Large and small letters in key-in commands (e.g., Whcs, extract) should be typed as shown; however, capital letters are used in descriptions (e.g., EXTRACT) to clarify meaning.

PROGRAM DESCRIPTION

EXTRACT can best be described in terms of its three main functions: extract, sort, and print.

The extract portion of the program reads each 80 character record of the Colorado WHCS file* (reading, consecutively, either single or multisegment files FILE15, FILE16, ...FILEN, excluding FILE41 and FILE42), compares the card class of the record with a set of user-defined card classes and extracts specified data from the record if card classes match, and writes the extracted data to a file (FILE09) using one of three record formats. Three formats are used per well unit rather than just one, due to the large amount of data that could be extracted. The record types currently defined are:

1. all data extracted except initial potential tests (IP's) and production tests (PDT's),
2. initial potential tests, and
3. production tests.

Due to the sorting algorithm, the data to be sorted is written at the beginning of each record.

*The data format used for EXTRACT is known as the PI WHCS Storage File Format. Each record contains the following:

```
    cc  1-14 API number
    cc 15-20 blank
    cc 21-25 card class
    cc 26-80 data.
```

Program EXTRACT currently extracts the following data for each well unit and writes it to FILE09 in approximately this order:

API number,

township,

range,

section,

spot,

state,

county,

well number,

well name,

operator elevation,

drill depth for the Dakota formation (602DKOT ,
602DKOTD, 602DKOTJ),

IP's for the Dakota formation (602DKOT ,
602DKOTD, 602DKOTJ),

PDT's for the Dakota formation (602DKOT ,
602DKOTD, 602DKOTJ)

In case of an abnormal termination during the extract process, a restart capability exists. See Restart Procedure, p. 12.

The sort portion of the program consists of a call to a system sort routine (see Appendix A, Sort Parameter Analysis) which sorts the extracted file (FILE09) in the order defined by file SORTDES, placing the results in a second file (FILE10). The following data is sorted in the order listed:

township,

range,

section,

API number, and

record number

where record number is an internal device for sorting purposes. When the sorting process is completed, FILE09 is deleted by a call to the system routine DELETE.

The print portion of the program reads each record of the sorted file (FILE10) and writes the data contained in that record to a second file (FILE11) in a format that can be easily read by the user. The system routine DPRINT is then called which causes FILE11 to be listed on the printer.

PROGRAM EXECUTION

EXTRACT.FORTRAN is the program source name in the USGS Denver MULTICS directory >udd>Whcs>TDyman for the program EXTRACT (see Appendix D, Program Code) used to extract specific data from the Colorado WHCS file (FILE15, ...FILEN), write the extracted data to FILE09, sort FILE09, write the sorted data to FILE10, and write the sorted data in an easy-to-read format in FILE11.

EXTRACT.FORTRAN consists of main program EXTRACT, which calls the three subroutines WRSUB, SORTP, and RESTRT, the five MULTICS system routines SORT, DELETE, DPRINT, OPEN and CF (Close File), and BLOCK DATA. Refer to figure 1 for an overview of the system.

To compile EXTRACT.FORTRAN, key in:

```
ft extract -card
```

Prior to running EXTRACT, determine if the user space is large enough for the expected output by keying in

```
gq >udd>Whcs
```

To execute the program EXTRACT key in:

```
extract
```

EXTRACT will present a task menu with the following choices:

```
1 = extract, sort, print (default)
2 = extract, sort, print restart
3 = extract
4 = extract restart
5 = sort
6 = print
7 = sort, print
```

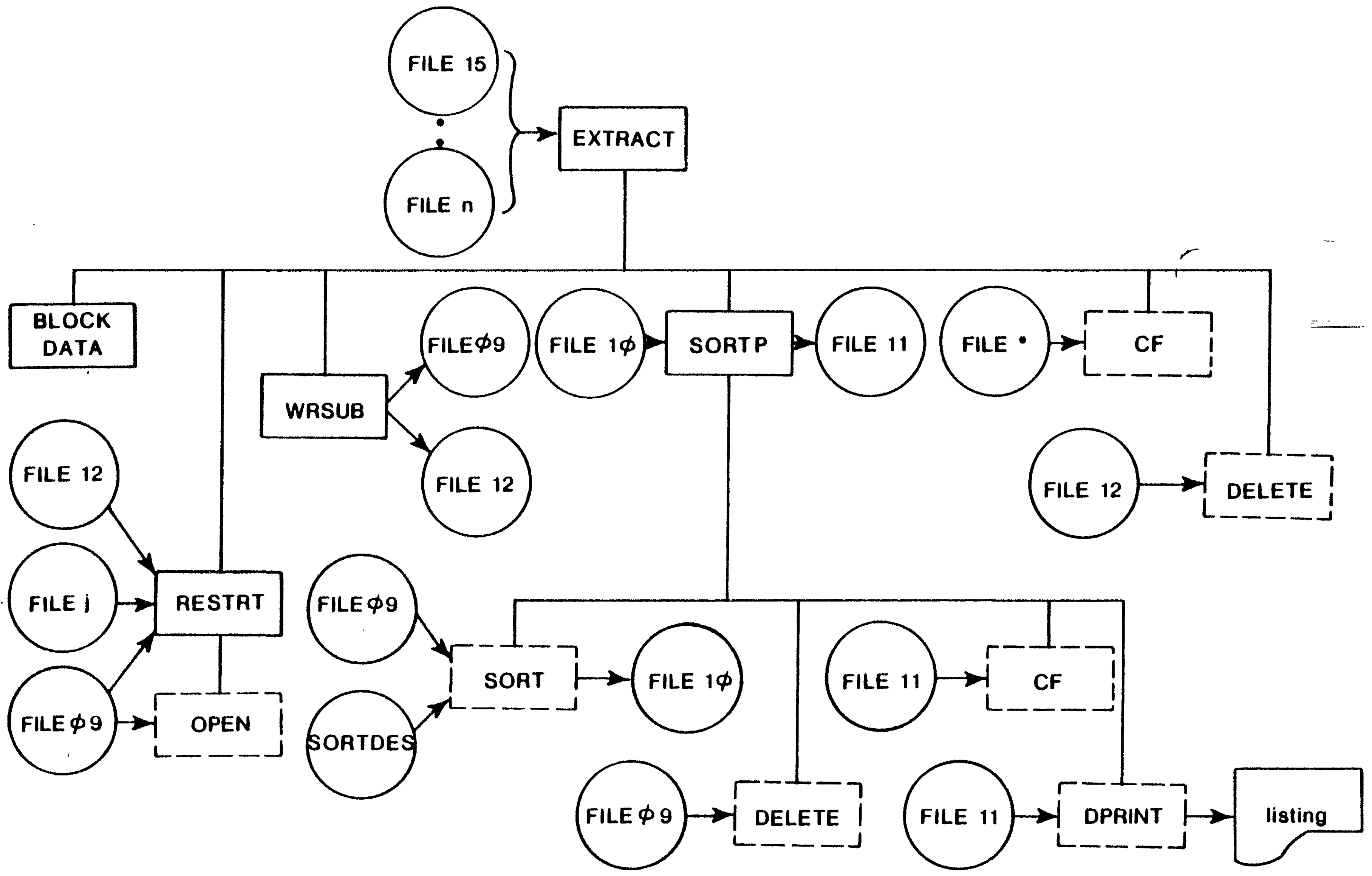

FIGURE 1 SYSTEM OVERVIEW

Decide which task you wish to accomplish and input the appropriate value. The responses 0, 1 and carriage return are equivalent.

Selecting task 1 will result in completing all three program functions: extract, sort, and print (assuming the program terminates normally). Selecting task 3 will result in extracting data only, and the sort and print functions will not be done. Tasks 5 and 6 will cause only sorting and only printing, respectively, and task 7 will cause only the combination of sorting and printing. Refer to Restart Procedure, p. 12, for a description of tasks 2 and 4 and for more information about tasks 5, 6, and 7.

EXTRACT will then ask for the logical FORTRAN unit of the _last_ file to be read, where it is assumed the first file to be read is FILE15. Thus, if 17 is the response, EXTRACT will extract data from the following files in order: FILE15, FILE16, FILE17. The extracted data is always output to FILE09.

So that the program's progress may be monitored, EXTRACT prints various messages on the user's terminal during execution. The first message which appears during the extract process lists the card classes for which data is to be extracted. Then every time data is extracted for 50 well units, the API number of the well unit just completed is printed, and after 1,000 well units have been processed, the value of the well unit counter is printed. (The 50 interval may be changed by modifying variable MESINT. See Appendix C, Definition of Variables). The first message that appears during the sort process indicates that sorting is about to take place. When sorting is completed, the number of records sorted is given and another message indicates that FILE09 is about to be deleted. The first message which appears during the print process indicates that FILE11 is about to be written. As in the extract process, a message appears every time data for 50 (and 1,000) well units has been written. (Variable MESINT defines this interval also). Then a message indicates that FILE11 is being sent to the printer to be listed. A sample terminal session is shown in figure 2.

NOTE

After running EXTRACT, and prior to re-running EXTRACT during the same login session, the following <u>must</u> be keyed in:

```
rl -a
new_proc
```

If not done, data will not be initialized properly.

```
extract
select tasks --
   1= extract, sort, print  (default)
   2= extract, sort, print restart
   3= extract
   4= extract restart
   5= sort
   6= print
   7= sort, print
1
input unit of last input file to be read
16

extracting data for the following card classes --

1 010  1 021  1 1 0  1 2 0  1 2 1  1 3 0  2 1 1  250 1  252 1  254 1
5 1 1
  05121054800000
  05121055300000
  05121055790000
  05121056300000
  05121056800000
  05121057290000
  05121057780000
  05121058280000
  05121058780000
  05121059290000
  05121059790000
  05121060290000
  05121060800000
  05121061300000
  05121061790000
  05121062280000
  05121062780000
  05121063280000
  05121063780000
  05121064280000
well unit      1000
  05121064780000
  05121065290000
  05121065810000
  05121066820000


sorting file09 into file10
```

FIGURE 2. SAMPLE EXTRACT TERMINAL SESSION

```
deleting file09

writing file11 in user format

  05121054800000
  05121055300000
  05121055790000
  05121056300000
  05121056800000
  05121057290000
  05121057780000
  05121058280000
  05121058780000
  05121059290000
  05121059790000
  05121060290000
  05121060800000
  05121061300000
  05121061790000
  05121062280000
  05121062780000
  05121063280000
  05121063780000
  05121064280000
well unit      1000
  05121064780000
  05121065290000
  05121065810000
  05121066820000

printing file11
1 request signalled, 4 already in printer queue 3

STOP
R 1511.1 03/21/79 $21.83 $57.40
```

FIGURE 2. SAMPLE EXTRACT TERMINAL SESSION (continued)

RESTART PROCEDURE

A restart capability has been provided for abnormal program termination. After the termination, respond "no" to any system query regarding file closure. Then key in

```
abc file09
abc file12
rl -a
new_proc
```

If EXTRACT has terminated after the extract process has been completed (i.e., during the sort or print portions), the program may be restarted such that only the remaining functions are executed. Select one of the following tasks from the task menu:

5 = sort

6 = print

7 = sort, print

If there is uncertainty that the extract function has been completed, one of the extract restart tasks may be selected as described below.

If EXTRACT is interrupted during the extract process, the program may be restarted by selecting one of the following tasks from the task menu:

2 = extract, sort, print restart

4 = extract restart

The program retains file unit and record count information at the time of an abnormal termination on FILE12*. When the extract portion of the program is restarted, the appropriate data file is positioned at the first record of the next well unit to be processed, and the output file (FILE09) is positioned immediately after the last record written for the last well unit completely processed. Any combination of multisegment and single segment files can be processed. Obviously, the larger the file, the greater the number of records that must be read to position the file correctly during a restart.

*FILE12 is deleted upon normal program termination.

SAMPLE PROCESSING STATISTICS

Processing statistics for a sample EXTRACT test is as follows:

Input file:	FILE15	11895 records	560 well units
Output files:			
unsorted:	FILE09	935 records	
sorted:	FILE10	935 records	
print:	FILE11	3087 records	
Run cost:			
cost:	$7.90		
date:	1/24/79		
time of day:	1300		

Print cost ($1.31 per 1000 lines):

$15.59	FILE15
$ 2.53	FILE10
$ 4.05	FILE11

Actual processing statistics for one run of the full Colorado WHCS file are:

Input files:	FILE15, ...FILE94	592,821 records
Output files:	FILE09/10	38,000 records
	FILE11	120,000 records

Run cost:	
Extract function (includes restart)	$181.00
Sort function	37.00
Print function	58.00
Miscellaneous costs:	
Miscellaneous terminal costs	30.00
Read tape	84.00
DIVMUL*	110.00
Storage cost ($.20/page/month for 12000 pages public space)	80.00/day
Print cost: FILE11	158.00

Please note a reduction in total run cost can be achieved by modifiying the program source to make FILE09 and FILE10 unformatted. If this is done, the argument list to SORT must be modified (see Appendix A, Sort Parameter Analysis).

*User utility program to convert multisegment files to single segment files. This operation may not be cost effective.

A savings in disk storage (and consequently run time) could also be achieved by rewriting the WHCS file such that a new record is created for each well unit. The new record, the zeroes record (00000), would contain the 14 character well unit API number. The first 20 characters for each of the remaining well unit records could then be deleted, avoiding a redundancy in storage of both the API number and the 5 extra blank characters. At the same time, the file could be written as unformatted, thereby achieving an additional savings.

Appendix A

SORT PARAMETER ANALYSIS

SORT PARAMETER ANALYSIS

EXTRACT calls a system routine, SORT, to sort the data residing in file FILE09 into file FILE10, where the fields sorted are township, range, section, API number and record number (in that order). The routine SORT is specifically used because it sorts multisegment files.

The subroutine call within EXTRACT is:

```
call sort ("-ids","record_stream_ -target vfile_
file09", "-ods","record_stream_ -target
vfile_ file10", "-td", ">udd>Whcs>TDyman",
"-sd","sortdes")
```

where the input file (FILE09) and output file (FILE10) are unstructured (i.e., ASCII), >udd>Whcs>TDyman is the temporary working directory, and SORTDES is the file containing the sort description.

The sort description is:

```
keys:  char(12) 3(18) char(12) 6(18) char(11) 9(18)
       char(14) 0 char(12) 12(9)
```

and can be explained as follows:

The format of each record in FILE09 is

$$\underbrace{\text{A8,A6,}}_{\text{API}}\ \underbrace{\text{3A4,}}_{\text{TWP}}\ \underbrace{\text{3A4,}}_{\text{rge}}\ \underbrace{\text{2A4,A3,}}_{\text{sec}}\ \underbrace{\text{I2,}}_{\text{rec \#}}\ldots$$

Thus, to sort the township field (char(12) 3(18)), for example, note the following:

1. the field is a character string (char);

2. it is 12 characters long (12);

3. it begins in the third word of the record (one word = 4 characters) (3);

4. it begins in the 18th bit of the word (one character = 9 bits) (18);

where counting begins with word 0 and bit 0.

Appendix B

GUIDELINES FOR PROGRAM EXPANSION

GUIDELINES FOR PROGRAM EXPANSION

A. If additional types of data are to be extracted from the Colorado WHCS file, the following guidelines may be helpful in modifying the program:

1. If the program searches for the card class corresponding to the new data, modify the code (e.g., add a DECODE statement) in EXTRACT for that card class.

2. If the program does not search for the card class corresponding to the new data,

   a. add the new card class (in any order) to the KLSD DATA statement in BLOCK DATA;

   b. increment the number of card classes searched for (NKLS DATA statement in BLOCK DATA);

   c. update the KLSD array size in all DIMENSION statements;

   d. add the new section of code extracting the data anywhere in EXTRACT, putting the label for that new section in the GO TO statement (in the proper order) prior to label 130;

   e. if necessary, modify the statement prior to label 130 that checks for 3xx, 4xx and 6xx records.

3. If data is to be extracted for additional formations,

   a. add the new formations to the FORMD DATA statement in BLOCK DATA;

   b. increment the number of formations searched for (NFORM DATA statement in BLOCK DATA);

   c. update the dimensions of all arrays which are dimensioned by NFORM in all CHARACTER and DIMENSION statements (refer to Appendix C, Definition of Variables);

d. change the value 3 corresponding to the present value of NFORM to the new value of NFORM in FORMAT 9110 in WRSUB and FORMAT 9210 in SORTP.

4. Add new variables to the appropriate CHARACTER and/or DIMENSION and COMMON statements in all routines.

5. Decide whether the new data extracted should be output to FILE09 in an existing record type (1/2/3) or whether a new record type should be defined.

   a. If the data is to be output in an existing record type,

      i. modify the existing WRITE and FORMAT statements and update the appropriate data initialization code in WRSUB;

      II. modify the existing READ, WRITE, and FORMAT statements in SORTP.

   b. If a new record type is to be defined,

      i. add the new section of code in WRSUB, putting the label for that new section in the GO TO statement prior to label 1000;

      ii. be sure the argument in the new call to WRSUB in EXTRACT is the correct record type;

      iii. the new record type should include (in this order) the variables API, ITWP, IRGE, ISEC, IREC for later sorting;

      iv. SORTP should be modified to read a record which is other than a type 1, 2, or 3, and then write it out in user format.

6. Update the documentation affected by the program changes.

B. If the extracted file (FILE09) is to be sorted by additional fields, the following guidelines may be helpful:

1. Assuming the new data to be sorted is presently being extracted and written to FILE09, modify the WRITE and FORMAT statements in WRSUB such that the new data is written immediately after record number (variable IREC) for each record type.

2. Modify the READ, WRITE, and FORMAT statements in SORTP for each record type.

3. Modify the sort description in file SORTDES, and the COMMENT after the call to SORT in SORTP. Refer to Appendix A, Sort Parameter Analysis.

4. Update the documentation affected by program and sort description changes.

C. If the current program is to be used to read a WHCS file for a state other than Colorado, the following guidelines may be helpful:

1. Determine if the formats (including card classes) are the same for the Colorado and new WHCS files.

   a. If the formats are the same, the program will not need modification.

   b. If the formats are different, some of the FORMAT statements will probably need modification.

   c. If the card classes are different, refer to the directions in Section A, above.

d. Presently, there are 83 possible records defined for a 2xxyy (xx=01-49) or a 5xxyy (xx=01-99) card class, where yy=01-53,70-99. If more records are defined for the new file,

   i. modify the MAX25R DATA statement in BLOCK DATA;

   ii. update the dimensions of all arrays which are dimensioned by MAX25R in all CHARACTER and DIMENSION statements (refer to Appendix C, Definition of Variables);

   iii. change the value 83 corresponding to the present value of MAX25R to the new value of MAX25R in FORMAT 9210 in WRSUB and FORMAT 9250 in SORTP.

2. Update the documentation affected by program changes.

Appendix C

DEFINITION OF VARIABLES

DEFINITION OF VARIABLES

Name	Dimension	Format	Type/ Common*	Description
API	2	A8,A6	EIGHT	API number extracted from the 10010 record
APII	2	A8,A6	EIGHT	API number read from every record
BLNK	-	A8	EIGHT	=b̸b̸b̸b̸b̸b̸b̸b̸ (8 blanks) used to initialize data
BLNK55	-	A55	FFIVE	=b̸b̸...b̸b̸ (55 blanks) used to initialize data
DAT	-	A55	FFIVE	All data (except API number and card class) read from every record
DDEP	NFORM	A5	EIGHT	Formation drill depths extracted from the 2xxyy records (xx=50,52, 54,yy=01-99)
DDP	3	A5	EIGHT	The three formation drill depths read from each of the 2xxyy records (xx=50,52,54,yy=01-99)

*e.g., API is declared CHARACTER*8 and is in COMMON /EIGHT/

DEFINITION OF VARIABLES (continued)

Name	Dimension	Format	Type/ Common	Description
FORM	3	A8	EIGHT	The three formations read from each of the 2xxyy records (xx=50,52, 54,yy=01-99)
FORMD	NFORM	A8	EIGHT	The specified (uppercase) formations for which data is to be extracted.**
FORME	NFORM	A8	EIGHT	The formations for which drill depths were extracted from the 2xxyy records (xx=50,52,54, yy=01-99)
ICNTY	-	A3	FOUR	County extracted from the API number (10010 record)
IDTP	-	A4	FOUR	The type (LOG, SPL, DRLR) of drill depth read from the 2xxyy records (xx=50,52,54,yy=01-99)

**Values are specified in BLOCK DATA

DEFINITION OF VARIABLES (continued)

Name	Dimension	Format	Type/ Common	Description
IDTYP	NFORM	A4	FOUR	The types (LOG, SPL, DRLR) of drill depths extracted from the 2xxyy records (xx=50,52,54,yy=01-99)
IFORM	NFORM	I1	DIM	Flag corresponding to each specified formation for which a drill depth is to be extracted from the 2xxyy records (xx=50,52,54, yy=01-99)(0=not extracted, 1 = extracted)
IN	-	I2	MISC	FORTRAN unit of the current input file (= 15, 16,...LASTIN)
IOUT	-	I1	MISC	FORTRAN unit of the sorted output file (=9)**
IOUTS	-	I1	MISC	FORTRAN unit of the sorted output file (=10)**

DEFINITION OF VARIABLES (continued)

Name	Dimension	Format	Type/ Common	Description
IP	-	I2	MISC	Flag corresponding to a specified formation for which IP's are to be extracted from 2xxyy (xx=01-49) or 5xxyy (xx=01-99) records (yy=01-53,70-99)(0 = not extracted, 1-NFORM=extracted)
IPDT	-	I2	MISC	Flag corresponding to a specified formation for which PDT's are to be extracted from 5xxyy records (xx= 01-99, yy=01-53,70-99) (0 = not extracted, 1-NFORM = extracted)
IREC	-	12	MISC	Well unit record counter. Used to sort multiple records for a single well unit
IRGE	3	3A4	FOUR	Range extracted from the 10021 record
ISAV	-	12	MISC	FORTRAN unit of the file on which file unit and record count information is written (=12)**

Name	Dimension	Format	Type/ Common	Description
ISEC	3	2A4,A3	FOUR	Section extracted from the 10021 record
ISTE	-	A2	FOUR	State extracted from the API number (10010 record)
ITASK	-	I1	MISC	Task selected by the user at the beginning of the program which specifies the function(s) to be performed. The task menu is: 1 = extract, sort, print (default) 2 = extract, sort, print restart 3 = extract 4 = extract restart 5 = sort 6 = print 7 = sort, print

DEFINITION OF VARIABLES (continued)

Name	Dimension	Format	Type/ Common	Description
ITWP	3	3A4	FOUR	Township extracted from the 10021 record
ITYP	-	A2	FOUR	The type of 5xx01 record read (=IP,PT, . . .)
ITYPD	-	A2	FOUR	The value set (=IP) in BLOCK DATA against which to evaluate the type of 5xx01 record read (variable ITYP)**
KLS	3	I1,2I2	DIM	Card class read from every record
KLSD	3,NKLS	I1,2I2	DIM	The specified card classes for which data is to be extracted**
KLSIP	3,MAX25R	I1,2I2	DIM	The card class of each record of IP's extracted
KLSPDT	3,MAX25R	I1,2I2	DIM	The card class of each record of PDT's extracted

Name	Dimension	Format	Type/ Common	Description
LASTIN	-	I2	MISC	FORTRAN unit of the last file to be read (the first file is assumed to be unit 15)
LPR	-	I2	MISC	FORTRAN unit of the "print" file (=11)**
MAX25R	-	I2	MISC	Maximum number of records defined for card class 2xxyy (xx=01-49) or 5xxyy (xx=01-99), (yy=01-53,70-99)**
MESINT	-	I5	MISC	Number of well units processed prior to printing a message on the user's terminal during program execution**
NFORM	-	I2	MISC	Number of specified formations for which data is to be extracted**
NFORME	-	I2	MISC	Number of formations for which drill depths were extracted from the 2xxyy records (xx=50,52,54, yy=01-99)

Name	Dimension	Format	Type/ Common	Description
NIP	-	I2	MISC	Number of records of IP's extracted for a specified formation (max = MAX25R)
NKLS	-	I2	MISC	Number of specified card classes for which data is to be extracted**
NMWELL	5	5A4	FOUR	The well name extracted from the last 102yy record (yy=00-99) which exists for each well unit
NOWELL	3	2A4,A2	FOUR	The well number extracted from the last 102yy record (yy=00-99) which exists for each well unit
NPDT	-	I2	MISC	Number of records of PDT's extracted for a specified formation (max = MAX25R)
NRECI	-	I10	MISC	Counter for number of records read from the current input file
NRECO	-	I10	MISC	Counter for number of records written to FILE09

DEFINITION OF VARIABLES (continued)

Name	Dimension	Format	Type/ Common	Description
NWELLU	-	I10	MISC	Counter for number of well units processed
OPELEV	2	2A8	EIGHT	Operator elevation (RB/CB/CF/RT/KB/DE/FS, GR) extracted from the 10300 record
PDT	MAX25R	A55	FFIVE	Each record of PDT's extracted for a specified formation from 5xxyy records (xx=01-99,yy=01-53,70-99)
SPOT	4	4A8	EIGHT	Spot extracted from the 10100 record
XIP	MAX25R	A55	FFIVE	Each record of IP's extracted for a specified formation from 2xxyy (xx=01-49) or 5xxyy (xx=01-99) records (yy=01-53,70-99)

Appendix D

EXTRACT PROGRAM CODE

```
c       program extract
c
c   program to read colorado whcs file and, for each well/unit,
c     extract the following data, write it (in approximately this
c     order) to an output file, sort the output file, and
c     write the sorted data in an easy-to-read format:
c      API number
c      TWP
c      rge
c      sec
c      spot
c      state
c      county
c      well number
c      well name
c      operator elevation
c      drill depth for 602dkot (dakota) formation  (dkot , dkotd, dkotj)
c      IPs for 602dkot formation  (dkot , dkotd, dkotj)
c      PDTs for 602dkot formation  (dkot , dkotd, dkotj)
c
c   input and extract output files are ordered by API number
c   sort and print output files are ordered by TWP, rge and sec
c   input record length is 80 characters (multiple records per
c     well/unit)
c   output record length is variable (multiple records per
c     well/unit)
c
        external sort(descriptors)
        external delete(descriptors)
        external dprint(descriptors)
        external cf(descriptors)
c
        character*4 iste,icnty,idtp,idtyp(3),itwp(3),irge(3),isec(3),
      *   nowell(3),nmwell(5),ityp,itypd
        character*8 api(2),apii(2),opelev(2),spot(4),
      *   form(3),forme(3),formd(3),ddp(3),ddep(3),blnk
        character*55 xip(83),pdt(83),dat,blnk55
c
        dimension kls(3),klsd(3,11),iform(3),klsip(3,83),klspdt(3,83)
c
        common /four/ iste,icnty,idtp,idtyp,itwp,irge,isec,
      *   nowell,nmwell,ityp,itypd
        common /eight/ api,apii,opelev,spot,
      *   form,forme,formd,ddp,ddep,blnk
        common /ffive/ xip,pdt,dat,blnk55
        common /dim/ kls,klsd,iform,klsip,klspdt
        common /misc/ in,iout,iouts,lpr,nkls,nwellu,irec,nform,nforme,
      *   ip,nip,ipdt,npdt,max25r,mesint,itask,nreci,nreco,isav,lastin
c
50      continue
c   ******************************************************************
c   select tasks
c
        write(6,9000)
9000    format(" select tasks --",/
      *         "    1= extract, sort, print   (default)",/
      *         "    2= extract, sort, print restart",/
      *         "    3= extract",/
      *         "    4= extract restart",/
      *         "    5= sort",/
```

```
     *         "   6= print",/
     *         "   7= sort, print")
      read(5,9001) itask
9001  format(i1)
      if(itask.lt.0 .or. itask.gt.7) go to 50
c  itask ge 5 = sort and/or print
      if(itask.ge.5) go to 220
c
c  define a variable with 55 blanks
      encode(blnk55,9004) (blnk,i=1,7)
9004  format (6a8,a7)
c
      write(6,9007)
9007  format(" input unit of last input file to be read")
      read(5,9008) lastin
9008  format(i2)
c  itask eq 2 or 4 = restart
      if(itask.ne.2 .and. itask.ne.4) go to 60
c  position input and output files
      call restrt (in,iout,isav,nreci,nreco,lastin)
c  check if finisned with extract function (unit gt last unit)
      if(in.gt.lastin) go to 210
      go to 70
60    continue
c  extract (not restart) -- set input file unit as 15
      in = 15
c
70    continue
c  access variable klsd because of block data requirements
      write(6,9009) ((klsd(i,j),i=1,3),j=1,nkls)
9009  format(//" extracting data for the following card",
     *  " classes --",//,(10(3i2,1x)))
c
100   continue
c  **********************************************************************
c  read a record
c
      read(in,9010,end=200) apii,kls,dat
9010  format(a8,a6,5x,i1,i2,i2,a55)
      nreci = nreci+1
c
110   continue
c  if this is a new well/unit, write data (record type 1) for the
c    previous well/unit, and initialize values
      if(apii(1).ne.api(1) .or. apii(2).ne.api(2)) call wrsud(1)
c
c  skip core test records (3xx), drill stem test records (4xx),
c    and general information records (6xx)
      if(kls(1).eq.3 .or. kls(1).eq.4 .or. kls(1).eq.6) go to 100
c
c  search for specific card classes
      do 130 index=1,nkls
      if(kls(1).ne.klsd(1,index)) go to 130
      if(kls(2).ne.klsd(2,index)) go to 130
      if(kls(3).ne.klsd(3,index)) go to 130
c  we have a match -- go to the appropriate program section
      go to (1000,2000,3000,4000,4000,5000,6000,7000,7000,7000,
     *  8000), index
130   continue
c  no match -- read next record
```

```
      go to 100
c
200   continue
c  ****************************************************************************
c  end of file
c
c  is the file just read the last file to be input
      if(in.eq.lastin) go to 210
c  increment input file unit
      in = in+1
c  skip units 41 and 42
      if(in.eq.41) in = 43
c  reset input record counter
      nreci = 0
c  go read another file
      go to 100
c
210   continue
c  write data (record type 1) for last well/unit
      call wrsub(1)
c  itask eq 3 or 4 = extract only
      if(itask.eq.3 .or. itask.eq.4) go to 230
c
220   continue
c  sort output file 9 into file 10, then "print" file 10 in an
c    easy-to-read format (file 11)
      call sortp
c
230   continue
c  close all files
      call cf ("-all")
c  delete scratch save file if tasks performed include extract
      if(itask.lt.5) call delete ("file12")
      stop
c
1000  continue
c  ****************************************************************************
c  card class 10010
c
c  extract API number, state, county
c
c  API number
      api(1) = apii(1)
      api(2) = apii(2)
c
c  state, county
      decode(api,9110) iste,icnty
9110  format(a2,a3)
      go to 100
c
2000  continue
c  ****************************************************************************
c  card class 10021
c
c  extract TWP, rge, sec
c
      decode(dat,9210) itwp,irge,isec
9210  format(3a4,3a4,2a4,a3)
      go to 100
c
```

```
3000  continue
c   **************************************************************
c   card class 10100
c
c   extract spot
c
      decode(dat,9310) spot
9310  format(15x,4a8)
      go to 100
c
4000  continue
c   **************************************************************
c   card class 102yy  (yy=00-99)
c
c   extract the last well number and well name given for a
c     specific well/unit
c
      decode(dat,9410) nowell,nmwell
9410  format(24x,2a4,a2,5a4)
4100  continue
      read(in,9010,end=4200) apii,kls,dat
      nreci = nreci+1
c   if this is still a 102 record, go decode info
      if(kls(1).eq.klsd(1,index) .and. kls(2).eq.klsd(2,index))
     *  go to 4000
c   this is a different card class
      go to 110
c
4200  continue
c   end of file - increment unit if not the last unit
      if(in.eq.lastin) go to 210
      in = in+1
      if(in.eq.41) in = 43
      go to 4100
c
5000  continue
c   **************************************************************
c   card class 10300
c
c   extract operator elevation (RB/CB/CF/RT/KB/DE/FS, GR)
c
      decode(dat,9510) opelev
9510  format(2a8)
      go to 100
c
6000  continue
c   **************************************************************
c   card class 2xx01-2xx53, 2xx70-2xx99  (xx=01-49)
c
c   extract IPs for nform specified formations (formd) if possible.
c   save rec 2xx01 in case the formation on rec 2xx02-2xx09 is
c     the desired formation.  if it is, also save recs 2xx10-2xx99.
c
c   save rec 2xx01
      xip(1) = dat
      do 6050 i=1,3
      klsip(i,1) = kls(i)
6050  continue
      ik = 1
c
```

```
6100  continue
c  loop through all records in this card class
      ik = ik+1
6125  continue
      read(in,9010,end=6300) apii,kls,dat
      nreci = nreci+1
c  is this still rec 2xx  (xx=1-49)
      inewxx = 0
      inew = 1
      if(kls(1).ne.klsd(1,index) .or. kls(2).gt.49) go to 6400
      inew = 0
c  is this 2xx the same as the last record (i.e., is
c    this a 2xx01 record)
      inewxx = 1
      if(kls(2).ne.klsip(2,ik-1)) go to 6400
      inewxx = 0
      xip(ik) = dat
      do 6150 i=1,3
      klsip(i,ik) = kls(i)
6150  continue
      nip = ik
c  have we already found a match (i.e., on formation)
      if(ip.ne.0) go to 6100
c  are we past the 2xx02-2xx09 records
      if(kls(3).gt.9) go to 6100
c  see if these 2xx records apply to the desired formation
      decode(dat,9620) form(1)
9620  format(a8)
c  if we have a match, set a flag
      do 6200 if=1,nform
      if(form(1).ne.formd(if)) go to 6200
      ip = if
      go to 6100
6200  continue
      go to 6100
c
6300  continue
c  end of file - increment unit if not the last unit
      if(in.eq.lastin) go to 210
      in = in+1
      if(in.eq.41) in = 43
      go to 6125
c
6400  continue
c  if IPs were extracted, write the record and initialize the arrays
      if(ip.ne.0) call wrsub(2)
c  if this is a 2xx01 record, go process it
      if(inewxx.eq.1 .and. kls(1).eq.2) go to 6000
c  if this is a 5xx01 record, go process it
      if(inewxx.eq.1 .and. kls(1).eq.5) go to 8000
c  no more 2xx records
      if(inew.eq.1) go to 110
      go to 100
c
7000  continue
c  ***********************************************************************
c  card class 2xxyy  (xx=50,52,54, yy=01-99)
c
c  extract drill depth for nform specified formations (formd)
c  idtyp is the type of drill depth extracted:
```

```
c       log  = log top     (rec 250)
c       spl  = sample top  (rec 252)
c       drlr = driller top (rec 254)
c
c  if we have extracted drill depths for all formations, skip
c     this section
      if(nforme.eq.nform) go to 100
      ik = 1
      go to 7200
c
7100  continue
c  loop through all records in this card class
      ik = ik+1
7150  continue
      read(in,9010,end=7600) apii,kls,dat
      nreci = nreci+1
c  is this still rec 2xx  (xx=50,52,54)
      if(kls(1).ne.klsd(1,index) .or. kls(2).ne.klsd(2,index)) go to 110
7200  continue
      decode(dat,9710) idtp,(form(i),ddp(i),i=1,3)
9710  format(a4,3(2x,a8,1x,a5))
c  does this record contain the specified formations
      do 7500 if=1,nform
c  have we already found a match
      if(iform(if).eq.1) go to 7500
      do 7400 i=1,3
      if(form(i).ne.formd(if)) go to 7400
c  set flag, increment counter, and save formation, drill type and depth
      iform(if) = 1
      nforme = nforme+1
      forme(nforme) = formd(if)
      idtyp(nforme) = idtp
      ddep(nforme) = ddp(i)
      go to 7500
7400  continue
7500  continue
      go to 7100
c
7600  continue
c  end of file - increment unit if not the last unit
      if(in.eq.lastin) go to 210
      in = in+1
      if(in.eq.41) in = 43
      go to 7150
c
8000  continue
c  ******************************************************************
c  card class 5xx01-5xx53, 5xx70-5xx99  (xx=01-99)
c
c  extract IPs for nform specified formations (formd) whether or
c    not done previously at label 6000
c  extract POTs for nform specified formations (formd)
c  in both extractions, save rec 5xx01 in case the formation
c    on rec 5xx02-5xx09 is the desired formation.  if it is, also
c    save recs 5xx10-5xx99.
c
c  if type is IP, use code at label 6000 to process IP records
      decode(dat,9810) ityp
9810  format(a2)
      if(ityp.eq.itypd) go to 6000
```

```
c
c  records are PDT
c  save rec 5xx01
      pdt(1) = dat
      do 8050 i=1,3
      klspdt(i,1) = kls(i)
8050  continue
      ik = 1
c
8100  continue
c  loop through all records in this card class
      ik = ik+1
8125  continue
      read(in,9010,end=8300) apii,kls,dat
      nreci = nreci+1
c  is this still rec 5xx  (xx=1-99)
      inewxx = 0
      inew = 1
      if(kls(1).ne.klsd(1,index)) go to 8400
      inew = 0
c  is this 5xx the same as the last record (i.e., is
c    this a 5xx01 record)
      inewxx = 1
      if(kls(2).ne.klspdt(2,ik-1)) go to 8400
      inewxx = 0
      pdt(ik) = dat
      do 8150 i=1,3
      klspdt(i,ik) = kls(i)
8150  continue
      npdt = ik
c  have we already found a match (i.e., on formation)
      if(ipdt.ne.0) go to 8100
c  are we past the 5xx02-5xx09 records
      if(kls(3).gt.9) go to 8100
c  see if these 5xx records apply to the desired formation
      decode(dat,9620) form(1)
c  if we have a match, set a flag
      do 8200 if=1,nform
      if(form(1).ne.formc(if)) go to 8200
      ipdt = if
      go to 8100
8200  continue
      go to 8100
c
8300  continue
c  increment unit if not the last unit
      if(in.eq.lastin) go to 210
      in = in+1
      if(in.eq.41) in = 43
      go to 8125
c
8400  continue
c  if PDTs were extracted, write the record and initialize the arrays
      if(ipdt.ne.0) call wrsub(3)
c  if this is a 5xx01 record, go process it
      if(inewxx.eq.1) go to 8000
c  no more 5xx records
      if(inew.eq.1) go to 110
      go to 100
c
```

```
      end
      subroutine wrsub(kode)
c
c  subroutine to write extracted colorado whcs data to an output
c    file and to initialize all variables file and arrays
c
      character*4 iste,icnty,idtp,idtyp(3),itwp(3),irge(3),isec(3),
     *  nowell(3),nmwell(5),ityp,itypd
      character*8 api(2),apii(2),opelev(2),spot(4),
     *  form(3),forme(3),formd(3),ddp(3),ddep(3),blnk
      character*55 xip(83),pdt(83),dat,blnk55
c
      dimension kls(3),klsd(3,11),iform(3),klsip(3,83),klspdt(3,83)
c
      common /four/ iste,icnty,idtp,idtyp,itwp,irge,isec,
     *  nowell,nmwell,ityp,itypd
      common /eight/ api,apii,opelev,spot,
     *  form,forme,formd,ddp,ddep,blnk
      common /ffive/ xip,pdt,dat,blnk55
      common /dim/ kls,klsd,iform,klsip,klspdt
      common /misc/ in,iout,iouts,lpr,nkls,nwellu,irec,nform,nforme,
     *  ip,nip,ipdt,npdt,max25r,mesint,itask,nreci,nreco,isav,lastin
c
c  branch to the type of record to be written
      go to (1000,2000,3000), kode
c
1000  continue
c  ***************************************************************
c  record type 1
c
c  write API number, TWP, rge, sec, record number,
c    total number of records for this well/unit,
c    spot, state, county, well number, well name, operator elevation,
c    number of formation drill depths,
c    formation, drill type, drill depth for nforme formations
c
      nrec = irec
      irec = 1
c  if we haven't processed any well/units yet, skip to data initialization
      if(nwellu.eq.-1) go to 1100
      if(nforme.eq.0) nforme = 1
      nreco = nreco+1
      write(iout,9110) api,itwp,irge,isec,irec,
     *  nrec,spot,iste,icnty,nowell,nmwell,opelev,nforme,
     *  (forme(i),idtyp(i),ddep(i),i=1,nforme)
9110  format(a8,a6,3a4,3a4,2a4,a3,i2,
     *  i2,4a8,a2,a3,2a4,a2,5a4,2a8,i2,3(a8,a4,a5))
c
c  every mesint well/units, write api number to users terminal
      nw = nwellu+1
      if(mod(nw,mesint).eq.0) write(6,9115) api
9115  format(3x,a8,a6)
c  every 1000 well/units, write to users terminal
      if(mod(nw,1000).eq.0) write(6,9120) nw
9120  format(" well/unit",i10)
c
c  save input file unit, record number of the last record
c    read from the input file for the most recently processed
c    well/unit, and record number of the last record
c    written to file09 for the most recently processed
```

```
c     well/unit
      nr = nreci-1
      rewind isav
      write(isav,9130) in,nr,nreco
9130  format(i2,2i10)
c
c  data initialization
1100  continue
      nwellu = nwellu+1
      iste = blnk
      icnty = blnk
      nforme = 0
      do 1200 i=1,2
      api(i) = apii(i)
      opelev(i) = blnk
1200  continue
      do 1300 i=1,3
      itwp(i) = blnk
      irge(i) = blnk
      isec(i) = blnk
      nowell(i) = blnk
1300  continue
      do 1400 i=1,4
      spot(i) = blnk
1400  continue
      do 1500 i=1,5
      nmwell(i) = blnk
1500  continue
      do 1600 i=1,nform
      iform(i) = 0
      forme(i) = blnk
      idtyp(i) = blnk
      ddep(i) = blnk
1600  continue
c  if no wells have been processed yet, initialize the rest of the data
      if(nwellu.eq.0) go to 2100
      return
c
2000  continue
c  **************************************************************
c  record type 2
c
c  write API number, TWP, rge, sec, record number,
c     formation, number of IP sets, and card class and
c     IPs for that formation for nip sets
c
      irec = irec+1
      nreco = nreco+1
      write(iout,9210) api,itwp,irge,isec,irec,
     *  forme(ip),nip,((klsip(j,i),j=1,3),xip(i),i=1,nip)
9210  format(a3,a6,3a4,3a4,2a4,a3,i2,
     *  a8,i2,33(i1,2i2,a55))
c
c  data initialization
2100  continue
      do 2200 i=1,max25r
      xip(i) = blnk55
2200  continue
      ip = 0
      nip = 0
      if(irec.eq.1) go to 3100
      return
c
3000  continue
c ********************************************************************
c  record type 3
c
c  write API number TWP, rge,sec, record number,
c    formation, number of PDT sets, and card class and
c    PDTs for that formation for npdt sets
c
      irec = irec+1
      nreco = nreco+1
      write(iout,9210) api,itwp,irge,isec,irec,
     *  formd(ipdt),npdt,((klspdt(j,i),j=1,3),pdt(i),i=1,npdt)
c
c  data initialization
3100  continue
      do 3200 i=1,max25r
      pdt(i) = blnk55
3200  continue
      ipdt = 0
      npdt = 0
      return
c
      end
      subroutine sortp
c
c  subroutine to sort file 9 into file 10, then "print" file 10
c    in an easy-to-read format (file 11)
c
      external sort(descriptors)
      external delete(descriptors)
      external cf(descriptors)
      external dprint(descriptors)
c
      character*4 iste,icnty,idtp,idtyp(3),itwp(3),irge(3),isec(3),
     *  nowell(3),nmwell(5),ityp,itypd
      character*8 api(2),apii(2),opelev(2),spot(4),
     *  form(3),forme(3),formd(3),ddp(3),ddep(3),blnk
      character*55 xip(83),pdt(83),dat,blnk55
c
      dimension kls(3),klsd(3,11),iform(3),klsip(3,83),klspdt(3,83)
c
      common /four/ iste,icnty,idtp,idtyp,itwp,irge,isec,
     *  nowell,nmwell,ityp,itypd
      common /eight/ api,apii,opelev,spot,
     *  form,forme,formd,ddp,ddep,blnk
      common /ffive/ xip,pdt,dat,blnk55
      common /dim/ kls,klsd,iform,klsip,klspdt
      common /misc/ in,iout,iouts,lpr,nkls,nwellu,irec,nform,nforme,
     *  ip,nip,ipdt,npdt,max25r,mesint,itask,nreci,nreco,isav,lastin
c
c ********************************************************************
c  jump to the selected task
c
c  itask eq 6 = print only
      if(itask.eq.6) go to 2000
c
c ********************************************************************
```

```
c  sort file 9 into file 10
c
c  original order is API number
c  new order is TWP, rge, sec, API number, record number
c
      write(6,9010)
9010  format(//" sorting file09 into file10")
      rewind iout
c
      call sort ("-ids","record_stream_ -target vfile_ file09",
     *           "-ods","record_stream_ -target vfile_ file10",
     *           "-td",">udd>Whcs>TDyman","-sd","sortdes")
c  note that "sortdes" is the file containing the sort
c    description, which is:
c     keys: char(12) 3(18) char(12) 6(18) char(11) 9(18)
c           char(14) 0 char(2) 12(9);
c
c  delete file09
      write(6,9020)
9020  format(//" deleting file09")
c     call delete ("file09")
c
c  itask eq 5 = sort only
      if(itask.eq.5) go to 3000
c
2000  continue
c  *******************************************************************
c  read the sorted file 10 and write it to print file 11
c    in an easy-to-read format
c
c  initialize counter for number of well/units processed
      nwellu = 0
      write(6,9205)
9205  format(//" writing file11 in user format"//)
      rewind iouts
c
2100  continue
c  read record type 1
      read(iouts,9210,end=2300) api,itwp,irge,isec,irec,
     *  nrec,spot,iste,icnty,nowell,nmwell,opelev,nforme,
     *  (forme(i),lotyp(i),ddep(i),i=1,nforme)
9210  format(a3,a6,3a4,3a4,2a4,a3,i2,
     *  i2,4a8,a2,a3,2a4,a2,5a4,2a8,i2,3(a8,a4,a5))
c
c  increment counter for number of well/units processed
      nwellu = nwellu+1
c
c  write well/unit number, stars
      write(lpr,9220) nwellu
9220  format(1x,i10,2x,60(2h**))
c  every mesint well/units, write api number to users terminal
      if(mod(nwellu,mesint).eq.0) write(6,9222) api
9222  format(3x,a3,a6)
c  every 1000 well/units, write to users terminal
      if(mod(nwellu,1000).eq.0) write(6,9225) nwellu
9225  format(" well/unit",i10)
c
c  write record type 1
      write(lpr,9230) api,itwp,irge,isec,
     *  spot,iste,icnty,nowell,nmwell,opelev
```

```
9230  format(a8,a6,3a4,3a4,2a4,a3,
     *  4a8,a2,a3,2a4,a2,5a4,2a8)
      if(forme(1).ne.blnk)
     *  write(lpr,9240) (forme(i),idtyp(i),ddep(i),i=1,nforme)
9240  format(14x,6(a8,1x,a4,a5,1x))
c
c  if there is only one record for this well/unit, go read
c    the next record
      if(nrec.eq.1) go to 2100
c
c  loop through the rest of the records for this well/unit
      do 2200 ir=2,nrec
c  read record type 2/3
      read(iouts,9250) api,itwp,irge,isec,irec,
     *  form(1),nip,((klsip(j,i),j=1,3),xip(i),i=1,nip)
9250  format(a8,a6,3a4,3a4,2a4,a3,i2,
     *  a8,i2,83(i1,2i2,a55))
c
c  write record type 2/3
      write(lpr,9260) form(1),xip(1),
     *  ((klsip(j,i),j=1,3),xip(i),i=1,nip)
9260  format(14x,a8,1x,a3,/(14x,i1,2i2,1x,a55))
2200  continue
      go to 2100
c
2300  continue
c  close and print file11
      write(6,9270)
9270  format(//" printing file11")
      call cf ("file11")
c     call dprint ("-he","FILE11","file11")
c
3000  continue
c  ****************************************************************
c  end of sort/print
c
      return
      end
      subroutine restrt (in,iout,isav,nreci,nreco,lastin)
c
c  subroutine to position input and output files when restarting
c    program extract after an interruption in the extract process
c  the input file is positioned at the first record of the
c    well/unit being processed at the time of the interruption
c  the output file is positioned immediately after the last
c    record of the most recently completed well/unit
c
c  get input file unit, input file record, output file record
      read(isav,9010) in,nreci,nreco
9010  format(i2,2i10)
c  check if finished with extract function (unit gt last unit)
      if(in.gt.lastin) go to 300
c
      write(6,9020)
9020  format(//" positioning input and output files for restart")
c
c  ****************************************************************
c  position input file
c
      do 100 i=1,nreci
      read(in,9110)
9110  format(1x)
100   continue
c
c  *******************************************************************
c  position output file
c
c     open output file with extend option
      open (9,attach="record_stream_ -target vfile_ file09 -extend",
     *  form="formatted")
      do 200 i=1,nreco
      read(iout,9110)
200   continue
c
300   continue
c  *******************************************************************
c  end of restrt
c
      return
      end
      block data
c
c  block data subprogram for program extract
c
      character*4 iste,icnty,idtp,idtyp(3),itwp(3),irge(3),isec(3),
     *  nowell(3),nmwell(5),ityp,itypd
      character*8 api(2),apii(2),opelev(2),spot(4),
     *  form(3),forme(3),formd(3),ddp(3),ddep(3),blnk
      character*55 xip(83),pdt(83),dat,blnk55
c
      dimension kls(3),klsd(3,11),iform(3),klsip(3,83),klspdt(3,83)
c
      common /four/ iste,icnty,idtp,idtyp,itwp,irge,isec,
     *  nowell,nmwell,ityp,itypd
      common /eight/ api,apii,opelev,spot,
     *  form,forme,formd,ddp,ddep,blnk
      common /ffive/ xip,pdt,dat,blnk55
      common /dim/ kls,klsd,iform,klsip,klspdt
      common /misc/ in,iout,iouts,lpr,nkls,nwellu,irec,nform,nforme,
     *  ip,nip,ipdt,npdt,max25r,mesint,itask,nreci,nreco,isav,lastin
c
c  fortran i/o units
      data iout,iouts,lpr,isav /9,10,11,12/
c  miscellaneous
      data blnk/"        "/
      data api /2*" "/
c  counter for number of well/units processed
      data nwellu /-1/
c  counter for number of records input from one file
      data nreci /0/
c  counter for number of records written to file09
      data nreco /0/
c  number of well/units processed before printing message
c    to users terminal
      data mesint /50/
c  maximum number of records of card class 2xxyy (xx=01-49) or
c    5xxyy (xx=01-99).  here yy=01-53, 70-99 = max25r records.
      data max25r /33/
c  number of formations searched for
      data nform /3/
```

```
c  formations searched for
       data formd /"602DKOT ","602DKOTD","602DKOTJ"/
c  designation on 5xx01 record for IP data
       data itypd /"IP"/
c  number of card classes searched for
       data nkls /11/
c  card classes searched for
       data klsd /1,00,10, 1,00,21, 1,01,00, 1,02,00, 1,02,01, 1,03,00,
     *             2,01,01, 2,50,01, 2,52,01, 2,54,01, 5,01,01/
c
c
       end
```