HYDROLOGY OF THE LOWER LITTLE RED RIVER, ARKANSAS, AND A PROCEDURE FOR ESTIMATING AVAILABLE STREAMFLOW

By Gerald D. Grosz, J.E. Terry, and A.P. Hall

U.S. GEOLOGICAL SURVEY

Water Resources Investigations Report 88-4008

Prepared in cooperation with the

ARKANSAS SOIL AND WATER CONSERVATION COMMISSION

Little Rock, Arkansas

1988

DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
2301 Federal Office Building
700 West Capitol
Little Rock, Arkansas 72201

Copies of this report can be
purchased from:

U.S. Geological Survey
Books and Open-File Reports Section
Federal Center, Building 810
Box 25425
Denver, Colorado 80225

CONTENTS

ILLUSTRATIONS

CONVERSION FACTORS

For use of readers who prefer to use metric (International System) units, rather than the inch-pound units used in this report, the following conversion factors may be used:

Multiply inch-pound unit	By	To obtain metric unit
foot (ft)	0.3048	meter (m)
cubic foot per second (ft^3/s)	0.02832	cubic meter per second (m^3/s)
mile (mi)	1.609	kilometer (km)
square mile (mi^2)	2.590	square kilometer (km^2)
gallon per minute (gal/min)	0.0630	liter per second (L/s)

Temperature in degrees Fahrenheit (°F) can be converted to degrees Celsius (°C) as follows:

$$°C = (°F - 32)\ 5/9$$

Sea level: In this report "sea level" refers to the National Geodetic Vertical Datum of 1929 (NGVD of 1929)--a geodetic datum derived from a general adjustment of the first-order level nets of both the United States and Canada, formerly called "Mean Sea Level of 1929."

HYDROLOGY OF THE LOWER LITTLE RED RIVER, ARKANSAS, AND A PROCEDURE FOR ESTIMATING AVAILABLE STREAMFLOW

By Gerald D. Grosz, J.E. Terry, and A.P. Hall

ABSTRACT

A hydrologic investigation of the lower Little Red River from near Searcy, Arkansas (mile 31.7), to the river's mouth at its confluence with the White River was conducted during 1983 and 1984 to obtain information needed for making streamflow allocation decisions. Data were collected on streamflow, stream altitude, ground-water altitude, and diversion pumping from the Little Red River.

A comparison of surface- and ground-water levels indicates that the Little Red is a gaining stream during summer and fall low periods and is a losing stream during periods of high flow. Diversions from the river during the summer of 1984 exceeded 115 cubic feet per second (the minimum release from Greers Ferry Reservoir 47 miles upstream) for 73 days. Minimum streamflow at Searcy during the period was 195 cubic feet per second. Flow in the Little Red River near Searcy was computed by using a modified stage/fall/discharge relation and stage data collected at Searcy and at Judsonia 6.5 miles downstream using a family of 12 rating curves. A mass balance procedure was developed to be used for estimating the amount of streamflow available along a reach of stream, given a minimum instream flow requirement. This procedure was coded into a computer program that can be invoked interactively as an aid in making streamflow allocation decisions and in maintaining related data bases.

INTRODUCTION

The Little Red River from near Searcy, Arkansas, to its mouth flows through an agricultural area. Several irrigation diversions located along the river remove significant, but unknown amounts of streamflow during the irrigation season. Estimates of streamflow remaining in the Little Red River at specific locations and times and resulting partly from these diversions was not available. Estimates of the streamflow remaining after diversion and conversely the amount of diversion which will allow the maintenance of streamflow at acceptable levels would aid water managers making allocation decisions. The project was done in cooperation with the Arkansas Soil and Water Conservation Commission (ASWCC).

Purpose and Scope

The purposes of this report are to (1) investigate the general hydrology (discharge, ground- and surface-water interaction, and diversions) of the lower Little Red River near Searcy, and (2) develop a simple procedure for estimating the amount of streamflow available for diversion. Field data collection took place from May 1983 through September 1984.

Description of Study Area

The area of this investigation lies primarily within White County, Arkansas (fig. 1). The study area is bounded on the northwest by either the Fall Line (the physiographic boundary between the Interior Highlands and the Coastal Plain) or the area of deposits designated as Tertiary undifferentiated by Counts (1957). The study area parallels the Little Red River and extends approximately 1 mile on each side of the river from north of Searcy to near Judsonia and approximately 5 to 8 miles on each side of the river downstream of Judsonia. The area is bounded on the east by the White River.

The topography of the study area is flat. Land surface altitude in the region ranges from 230 feet near Judsonia to 190 feet around the mouth. The Little Red River flows generally from northwest to southeast, out of the Interior Highlands and onto the Coastal Plain.

The economy of the area is based primarily on agriculture with the major emphasis on the production of crops such as rice, soybeans, and grain sorghum. Rice is the major crop in the area and utilizes the most irrigation water. This irrigation water is obtained from the Little Red River and the alluvial aquifer of Quaternary age.

The Little Red River is regulated by the U.S. Army Corps of Engineers at Greers Ferry Dam, a hydroelectric power and flood control structure 78.8 miles upstream from the mouth. Contained behind this structure is Greers Ferry Reservoir which has a drainage area of 1,153 mi^2 (Sullavan, 1974).

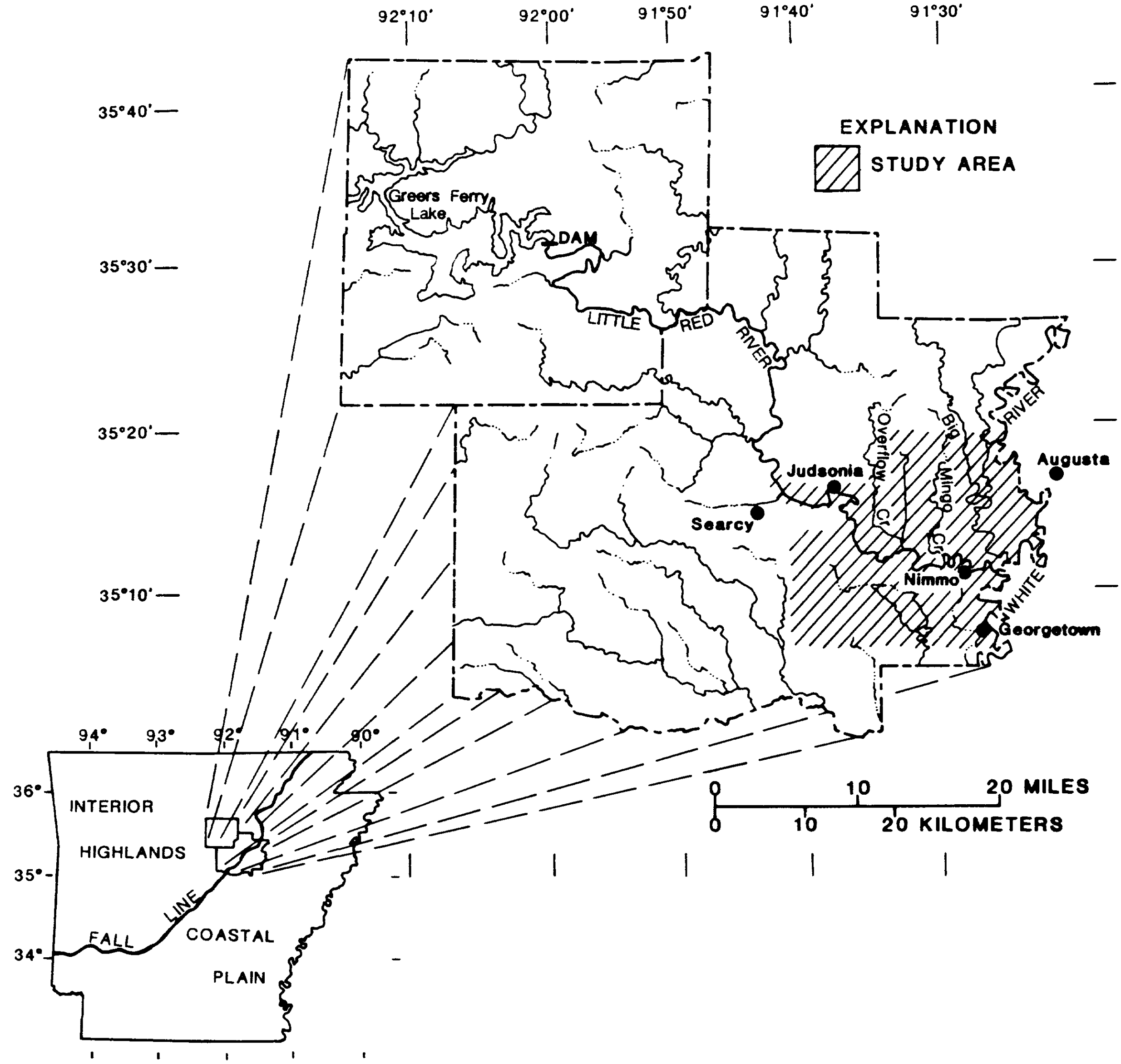


Figure 1.--Location of study area.

The river drains approximately 500 mi^2 (Sullavan, 1974) between the Greers Ferry Dam and the upstream end of the study area. Flows in the Little Red are also affected by varying backwater conditions resulting from high stages on the White River.

The study area contains approximately 150 mi^2 of land that drains directly into the Little Red River. Major tributaries in the area are Overflow Creek and Big Mingo Creek. These streams flow generally from north to south and drain 60.4 mi^2 and 34.8 mi^2, respectively. Both of these tributaries contribute little or no flow during dry periods. Tributaries south and west of the Little Red River are small and intermittent. The topography in much of the area is very flat with swamps, especially along Big Mingo Creek and near the mouth of the Little Red River.

Streamflow in the Little Red River and White River is regulated throughout the year. Consistent high flows occur from January to June. Highly variable flows in the Little Red River during the summer are the result of variable releases from Greers Ferry Dam based on demand for hydroelectric power. The water release policy of the Corps of Engineers for Greers Ferry Dam provides for a minimum release of 115 to 225 ft^3/s from May through October. For air temperatures below 90°F, the minimum release is 115 ft^3/s and as the temperature increases the minimum release increases to 225 ft^3/s. During the remainder of the year the minimum release is 40 ft^3/s. From May 20, 1983, to September 30, 1984, the range of discharge in the Little Red River near Searcy was 60 ft^3/s to 13,200 ft^3/s.

Numerous water diversion structures are located on the Little Red River from near Judsonia to the mouth. Water is diverted from the river primarily during June through September for crop irrigation.

Ground water used in the study area is obtained almost exclusively from alluvial deposits of Quaternary age. These deposits cover the entire study area and generally are less than 150 feet thick. Lithologically, these beds grade from silts and fine-grained materials at the surface to coarser sands and gravels at the base (Counts, 1957). The basal sand and gravel beds are capable of producing much more water than the overlying fine-grained materials. All irrigation wells and many domestic wells are developed in these basal beds with yields generally ranging from 400 to 2,000 gal/min. However, near the Fall Line the basal beds are thin or absent altogether and well yields are much lower.

Alluvial deposits in the study area are underlain by Tertiary materials. These Tertiary beds are generally thin and lenticular and are composed of compact sand and interbedded clay. Wells finished in these beds typically yield only enough water for household or small farm use (Counts, 1957).

Acknowledgments

The assistance of diversion pump owners and observers was essential to obtaining accurate diversion data. Also, the permission of well owners to make well measurements was appreciated.

HYDROLOGIC DATA COLLECTION AND ANALYSES

Surface Water

A stream-gaging station, (station number 07076634), was located on the Little Red River at Judsonia (25.2 miles upstream from the mouth) before this investigation began. This was originally a non-recording gage with data being collected intermittently by the U.S. Weather Service and the U.S. Army Corps of Engineers since the early 1900's. In October of 1981 responsibility for operation of this gaging station was assumed by the U.S. Geological Survey and a continuous stream-stage recorder was installed.

Historically, it has been difficult to establish a stable stage-discharge relationship (rating curve) at the Judsonia gage for high flows (greater than 1,000 ft^3/s). Part of this difficulty is due to variable backwater conditions that result from high stages of the White River.

Because of the need for basic stage/discharge data on the Little Red River, a second continuous stream-stage recorder was established in May 1983 on the Little Red River near Searcy, 31.7 miles upstream from the mouth. It was intended that this station serve as a "base gage" and that the Judsonia station be used as an "auxiliary gage". Data collected at both sites would be used to develop a stage/fall/discharge relation from which discharge could be computed as a function of stage at the base gage and fall (the difference in water-surface altitudes) between the base gage and the auxiliary gage.

As data collection for this investigation progressed it became apparent that some data for the Searcy station would have to be simulated in order to adequately define the stage/fall/discharge relation. (There were not enough high discharge measurements at Searcy to define the upper end of the relation.) Using discharge and stage data from the older Judsonia station and channel cross-section data obtained from the Corps of Engineers, a step-backwater model (Shearman, 1976) was applied to the Little Red River between Judsonia and Searcy. The step-backwater model was adjusted until the discharge measurements at the Searcy station could be reproduced within 5 percent. The step-backwater model was then used to simulate stages at Searcy for higher discharges that had been observed at Judsonia.

All of these data, observed and simulated, were then used to develop a stage/fall/discharge relation which resulted in the "family of curves" shown in figure 2. This relation is now being used to compute discharge in the river as a function of the stage at the base gage and the fall between the water-surface altitudes at the two gages. This discharge is assumed to occur at a low water dam (which acts as a control at lower flows) located between the two gage sites (30.9 miles upstream from the mouth). The validity of the family of curves shown in figure 2 is continually checked and verified as additional discharge measurements are obtained at the Searcy station. Computed mean daily discharge data for the Little Red River near Searcy, station number 07076620, are available from the U.S. Geological Survey WATSTORE computer data base.

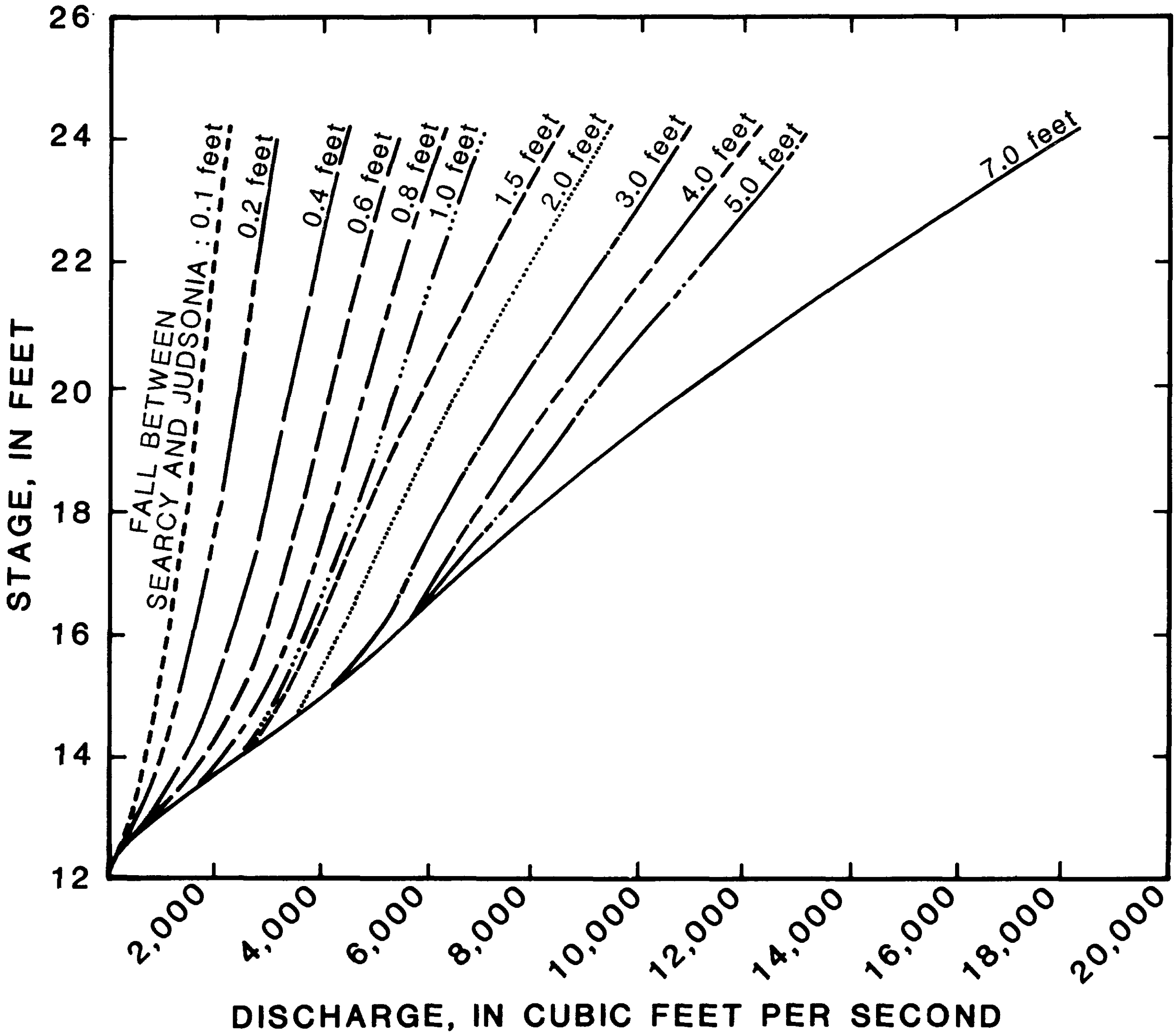


Figure 2.--Stage/fall/discharge relation at Little Red River near Searcy.

On July 18, 1984, a stream-stage recorder was installed at Nimmo, 3.5 miles upstream from the mouth of the Little Red River, to monitor stage/discharge conditions near the mouth. Slow velocities and extremely poor measuring conditions at this site made measuring discharge difficult. However, stage data were recorded at Nimmo through October 2, 1984. A comparison of stage data collected at Nimmo and Judsonia during the period July through September 1984 (fig. 3) indicated that water was moving out of the Little Red River and into the White River; however, this was not verifiable by standard discharge measuring techniques because of the conditions previously described.

Observations made during the summer of 1984 indicated no measurable flow from any tributary into the Little Red River.

Ground Water

Ground-water levels in the vicinity were monitored to investigate the relationship between surface and ground water in the study area. Water levels were measured monthly in 17 wells near the Little Red River. Water levels were referenced to land-surface altitudes surveyed to the nearest 0.01 foot. These measurements and corresponding daily river stage measurements at the five gaging stations were used to develop potentiometric surface maps of the alluvial aquifer in the study area. Figures 4 and 5 show the potentiometric surface for April and July of 1984. Water levels in the monitoring wells declined from 1 to 10 feet between April and July.

To more closely investigate the relation of the alluvial aquifer and the river, two wells equipped with continuous water-level recorders were installed near the river (figs. 4 and 5). Well A is located approximately 100 feet south of the river's low-water channel about 21.3 miles upstream from the river's mouth. Well B is located approximately 2,600 feet south of the river channel about 15.2 miles upstream from the mouth. Figure 6 is a comparison of the water-surface altitudes of the two wells and the Little Red River at Judsonia. At times, the river level was above the water level in both wells and provided some recharge to the alluvial aquifer. During the summer and fall, the river level was below the water level in both wells indicating that ground water was flowing into the river from the surrounding alluvium.

Diversions

The amount of water being diverted from the river needed to be estimated for general information and to aid in the development of an allocation procedure. During the summer of 1984, 16 diversion structures were being used (fig. 7). Discharge tests were run to obtain the pump ratings for the diversions. For those diversions powered by diesel engines, discharge was measured at two different engine speeds. This allowed discharge from the diversion to be calculated from engine speed. For electric powered pumps only one discharge was required because of the constant speed of electric motors.

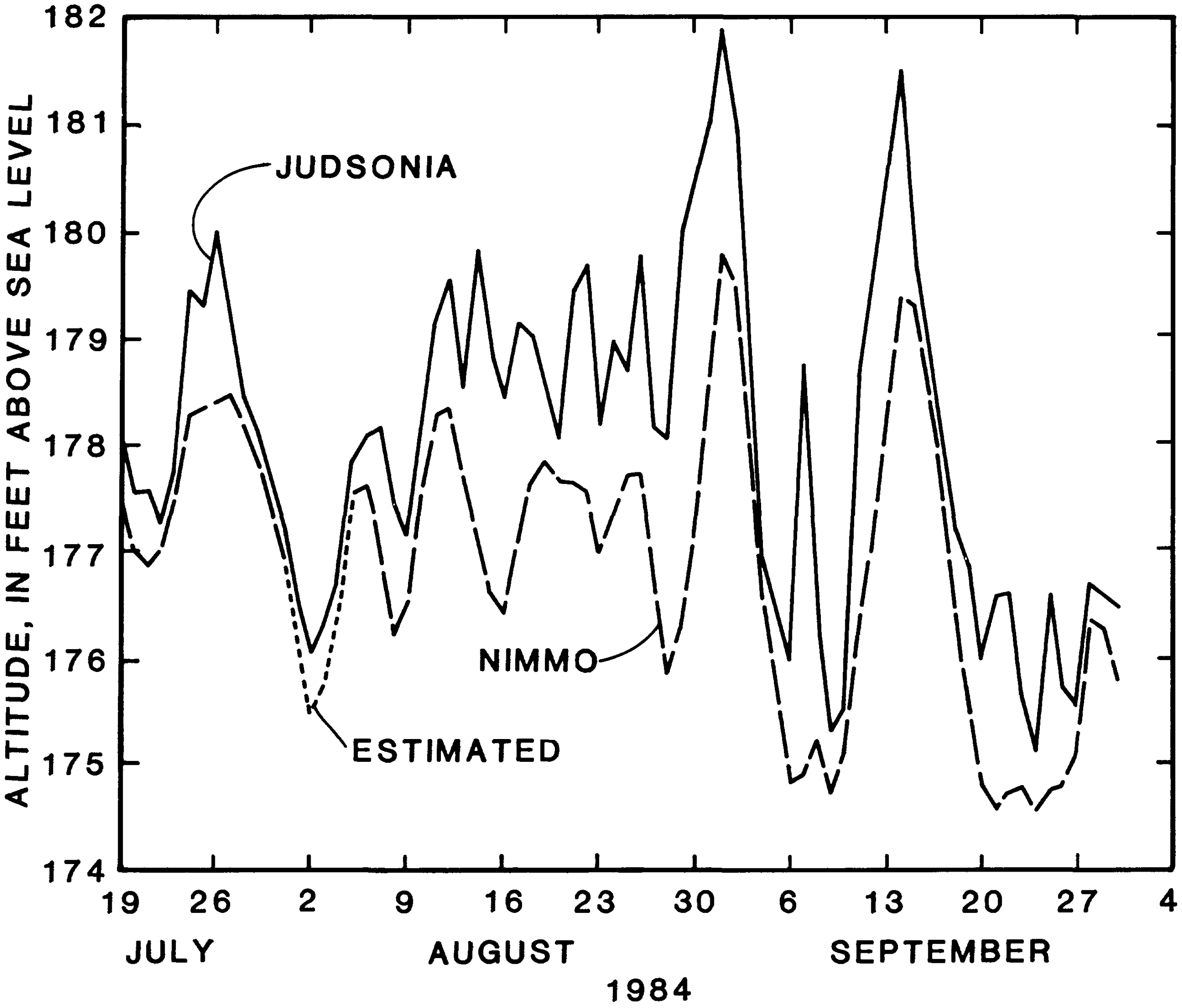
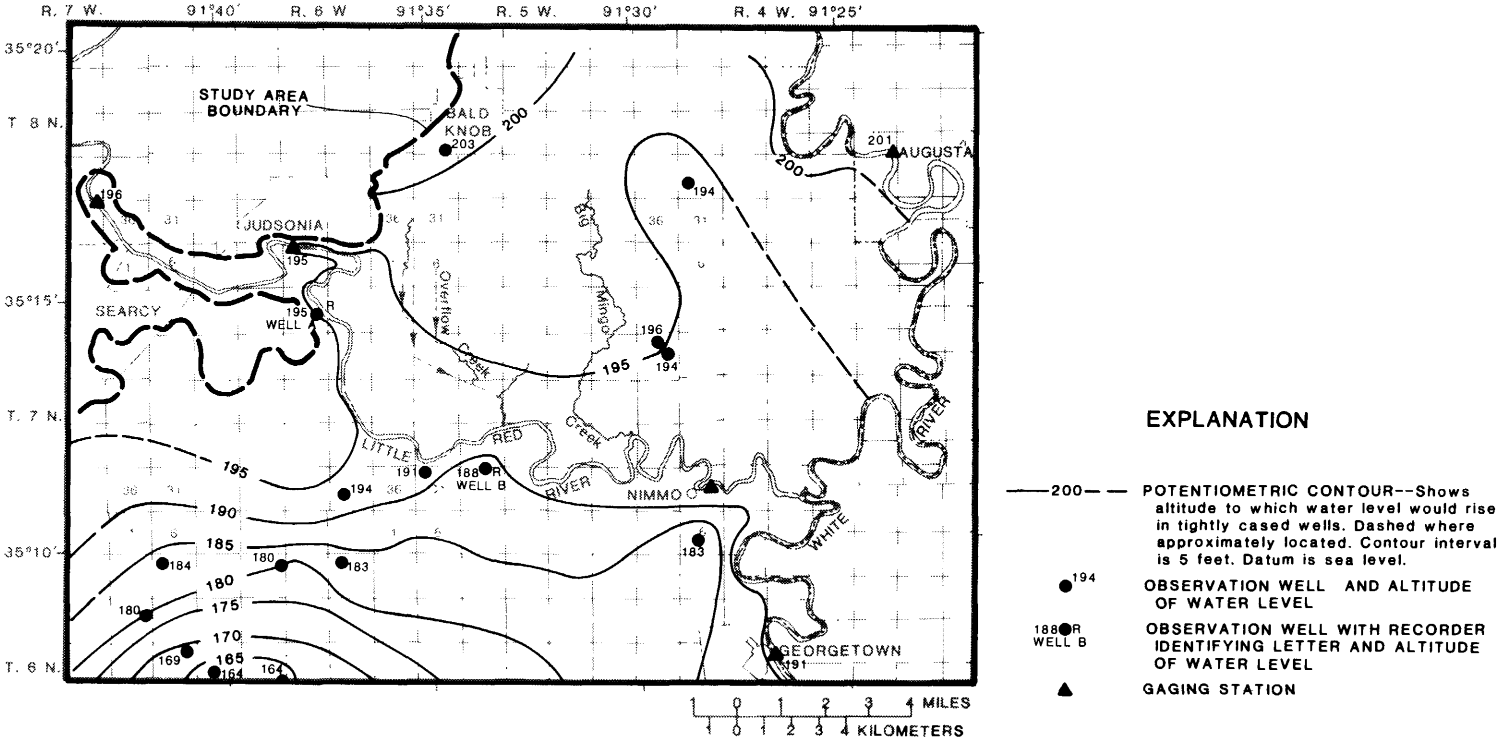


Figure 3.--Hydrographs of water level in the Little Red River at Judsonia and Nimmo.



EXPLANATION

——200— — POTENTIOMETRIC CONTOUR--Shows altitude to which water level would rise in tightly cased wells. Dashed where approximately located. Contour interval is 5 feet. Datum is sea level.

●194 OBSERVATION WELL AND ALTITUDE OF WATER LEVEL

188●R WELL B OBSERVATION WELL WITH RECORDER IDENTIFYING LETTER AND ALTITUDE OF WATER LEVEL

▲ GAGING STATION

Figure 4.--Generalized potentiometric surface of alluvial aquifer, April 1984.

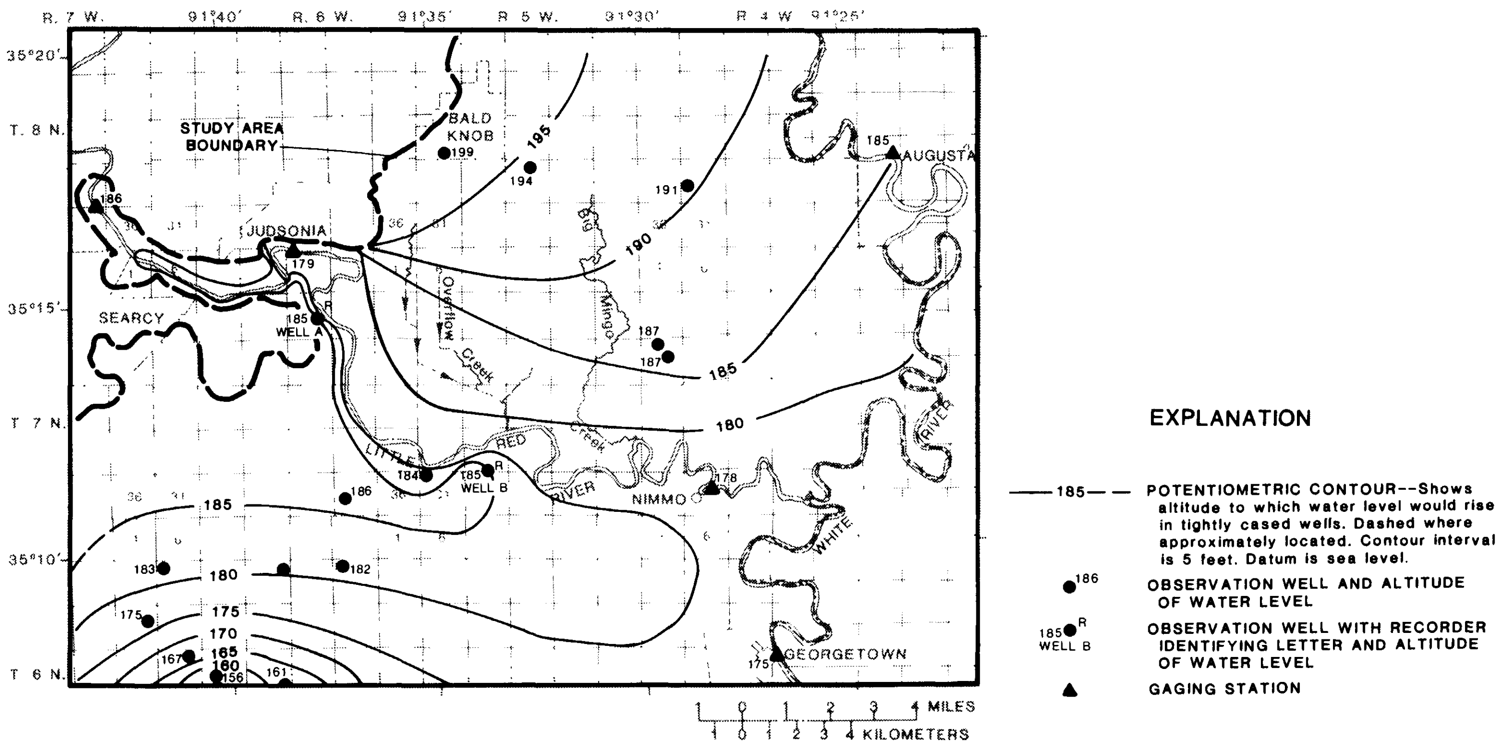


Figure 5.--Generalized potentiometric surface of alluvial aquifer, July 1984.

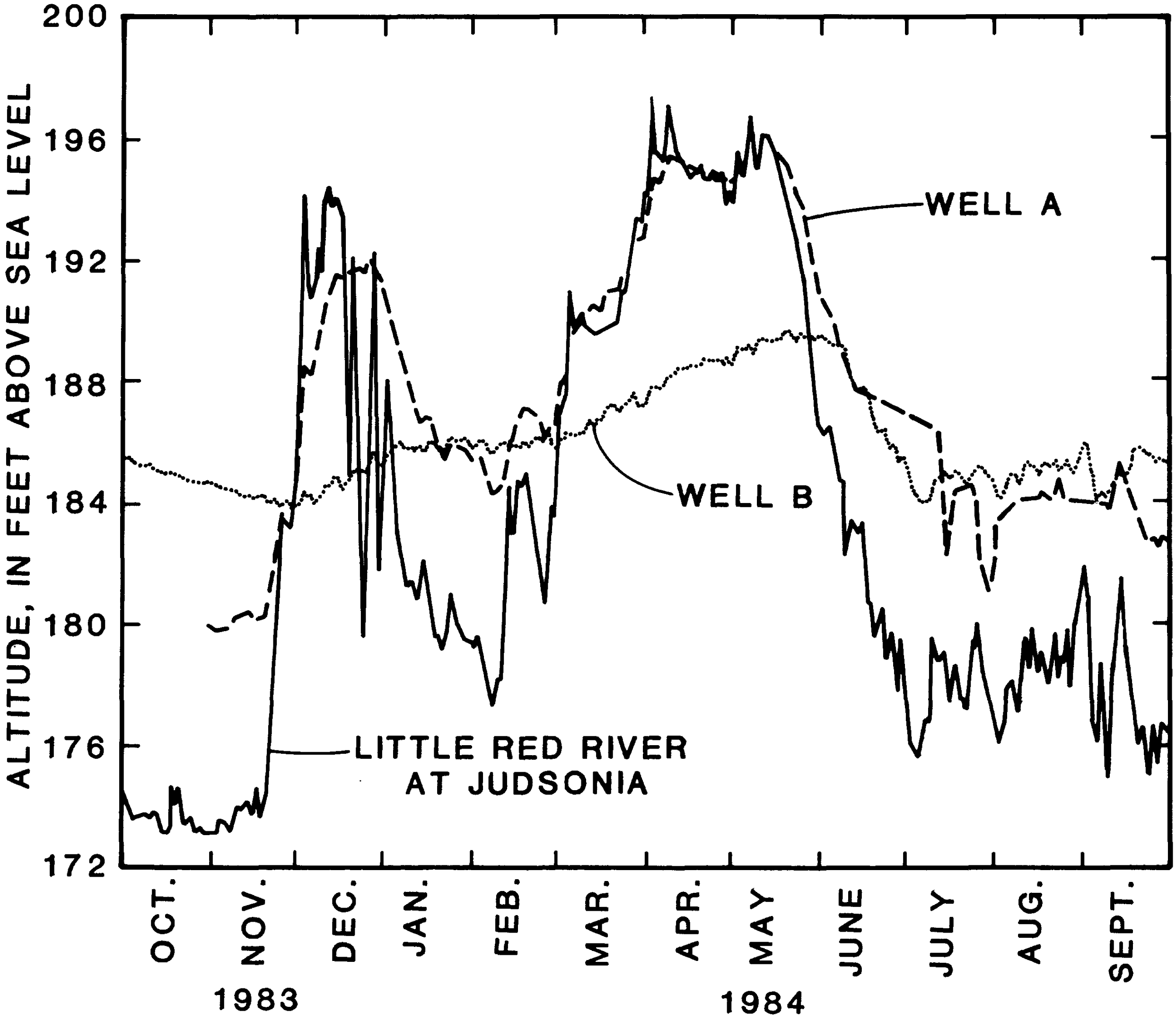


Figure 6.--Hydrographs of water level in the Little Red River and two nearby wells in the alluvial aquifer.

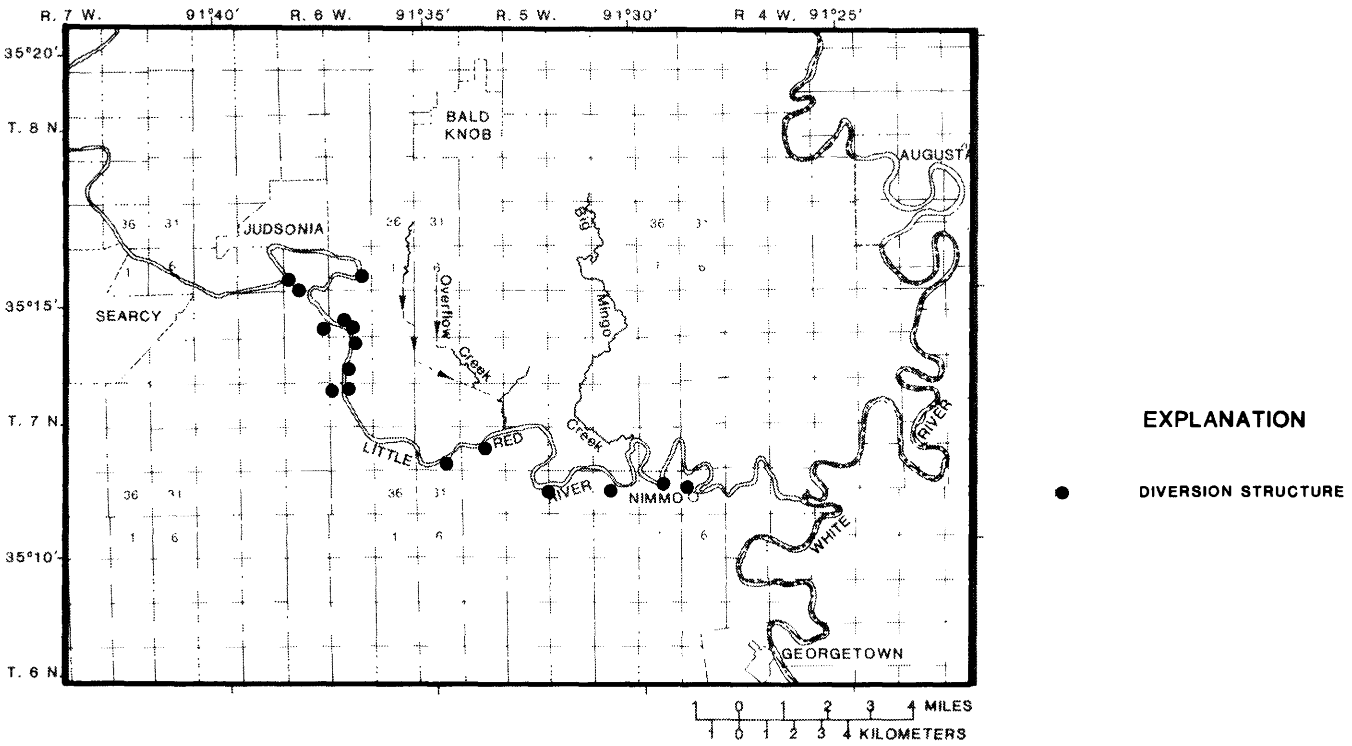


Figure 7.--Location of diversion structures on the Little Red River.

Observers at each diversion site were asked to note any adjustments made to the pumps such as turn on, speed adjustment, turn off, and the corresponding time of any change in order to obtain daily estimates of water pumped. With this pump activity data, estimates were made of the amount of water diverted each day by each diversion. Figure 8 is a summary of the discharges from the 16 diversions.

Potentially, diversion pumping could exceed the flow in the Little Red River. The diversion data for 1984 showed 73 days with diversion pumping greater than 115 ft^3/s, 11 days greater than 200 ft^3/s, and a maximum of 219 ft^3/s. Potential diversion capacity was 269 ft^3/s. The smallest discharge estimated for the Little Red River during the summer of 1984 was 195 ft^3/s. Figure 9 is a comparison of Little Red River discharge and diversion pumping.

Field observations made during the summer of 1984 and discussions with diverters in the project area indicate that there was no return flows from irrigated fields to the Little Red River. There may have been some seepage or leaky pipes in some places but apparently this was insignificant.

PROCEDURE FOR ESTIMATING STREAMFLOW AVAILABLE FOR DIVERSION

Development

A procedure using a computer program (Attachment A) has been developed that can be used to estimate the amount of streamflow available for diversion at any point along a given reach. The program is basically a mass balance that distributes inflows and outflows between two reference points, A and B. For simplicity and a minimum of input data, the program was developed for steady-state conditions.

Data requirements for the procedure are observed flow at the reference points, user-defined minimum flows at the reference points, and the locations and discharges of any diversions or tributaries. Existing tributary or diversion flows may be entered directly and are denoted by a stream index of "c" in output from the program. Existing diversion flows are always entered as negative values. All flow values should represent a point in time and not long-term averages such as weekly, monthly, and so forth. If tributary flows are unknown they may be estimated on the basis of a percentage of the flow at either reference site A or B. In such cases the "tributary percent", the resulting tributary flow, and an appropriate stream index of "A" or "B" are associated with the indicated stream mile. Program output resulting from an availability analysis are displayed where such estimates have been made. A stream index of "u" for undefined, always indicates that no known tributary or diversion has been defined. The user must also specify the stream reach of interest and the length of the computation interval (such as 0.1 mile, 1 mile, 2 miles, and so forth).

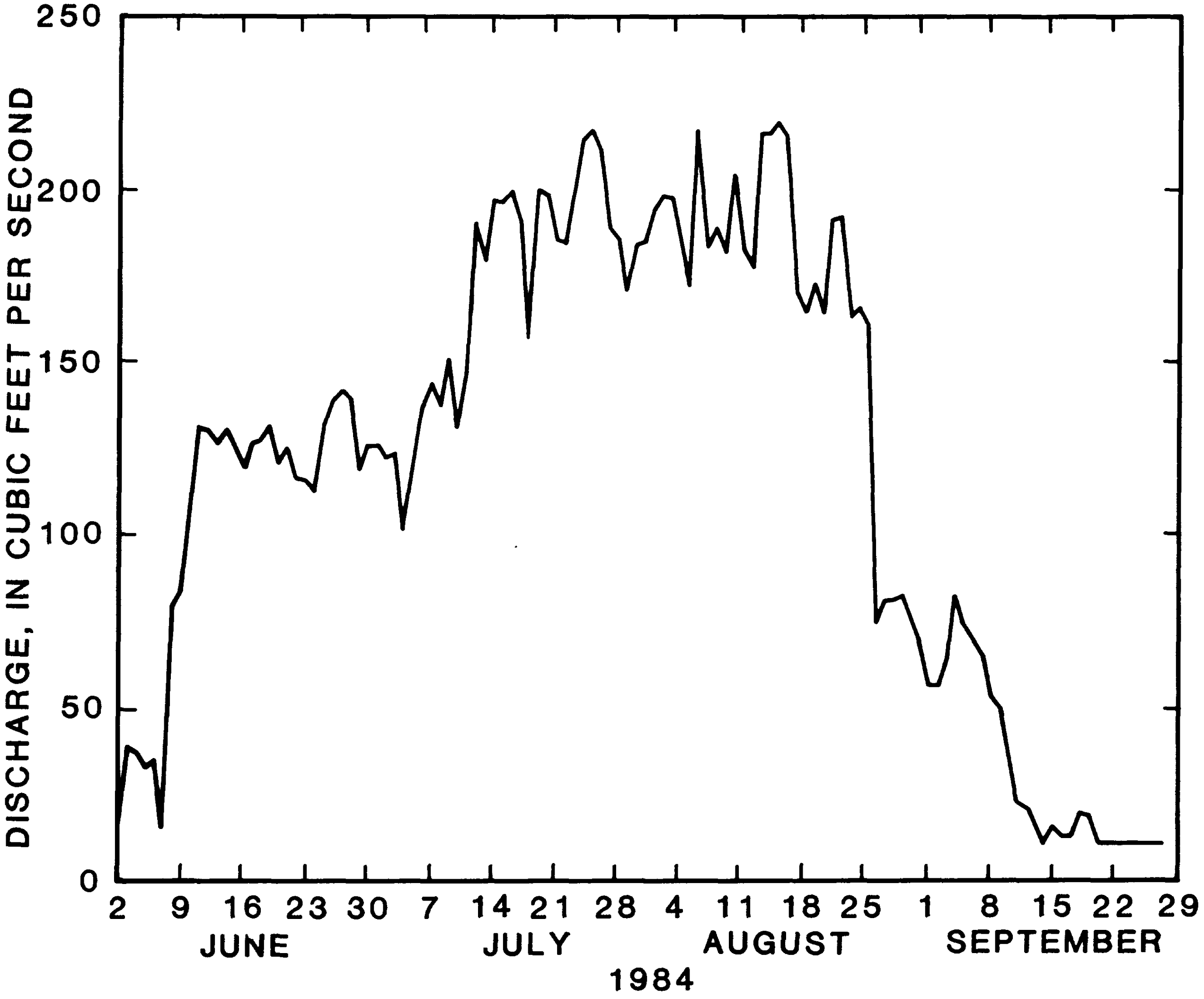


Figure 8.--Estimated daily diversion from Little Red River, summer 1984.

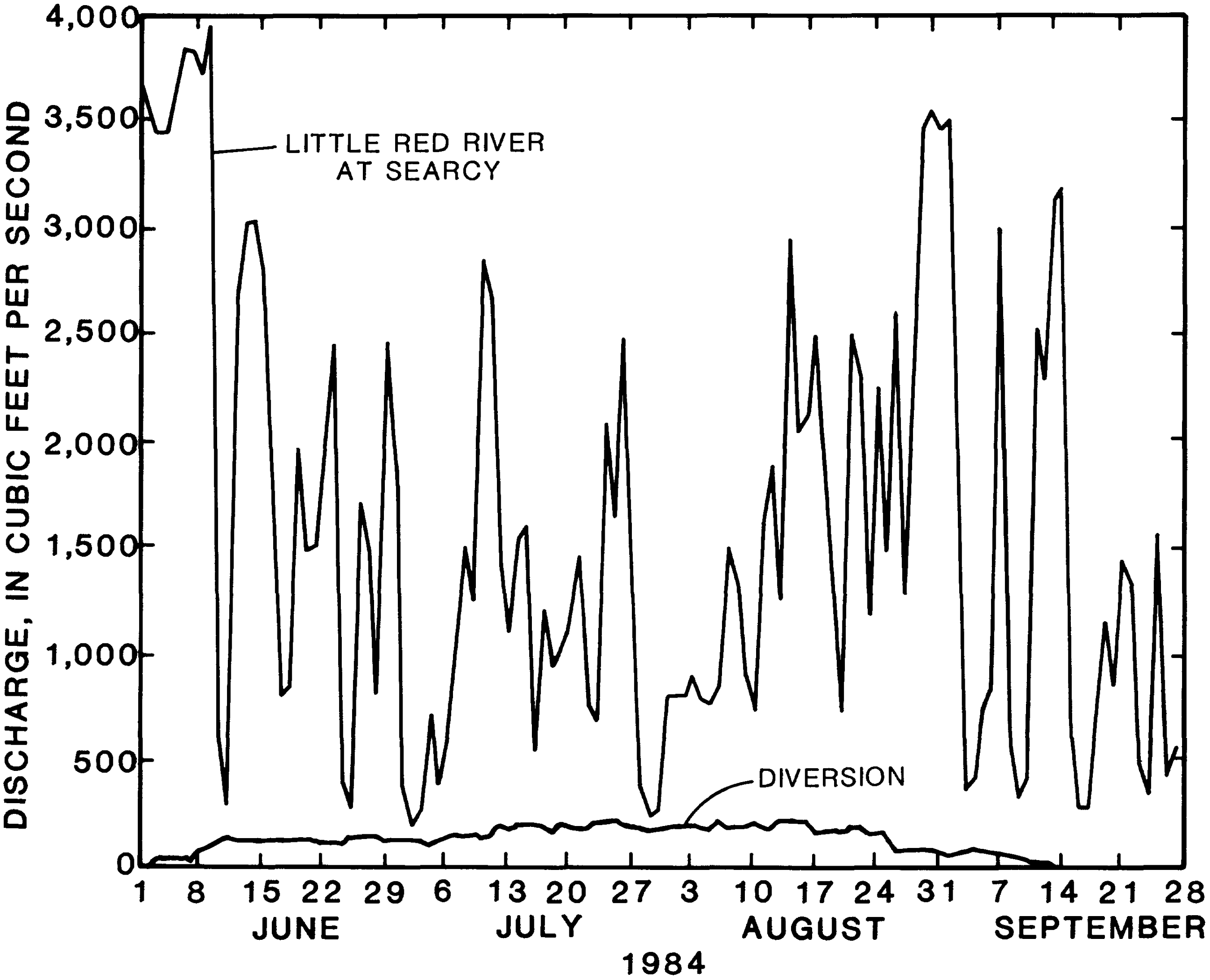


Figure 9.--Comparison of diversion and discharge from the Little Red River, summer 1984.

The following terms are used in the program.

1. BGREF, ENREF - observed flows at the beginning and ending reference points (A and B).

2. BGMN, ENMN - the user-defined minimum flows for A and B.

3. TRB_FLOW (i) - tributary (positive) or diversion (negative) flow at computation interval i.

4. BGN_MILE - relative stream mile location of reference point A.

5. END_MILE - relative stream mile location of reference point B.

6. DIV_PER_MILE - number of computation intervals per mile (i.e. if DIV_PER_MILE is 2, computation interval is 0.5 mile).

7. SUMTRB - sum of all tributaries and diversions.

8. EXCESS - net seepage or overland inflow (positive) or outflow (negative).

9. MNDIFF - difference in user-defined minimum flows between points A and B.

10. CUMAVL - sum of EXCESS and MNDIFF, distributed linearly between points A and b.

11. RESID - incremental residual used in computing estimated available flow for each interval.

12. RESID_TOT - incremental residual used in computing estimated total flow for each interval.

13. AVAIL_FLOW(i) - Estimated flow available for diversion at computation interval i.

14. TOT_FLOW(i) - Estimated streamflow at computation interval i.

15. MIN_FLOW(i) - Estimated minimum flow to be maintained at computation interval i.

The following relations are significant in the program.

EXCESS = ENREF - (BGREF + SUMTRB) (1)

MNDIFF = BGMN - ENMN (2)

CUMAVL = EXCESS + MNDIFF (3)

RESID = CUMAVL/((BGN_MILE - END_MILE)*DIV_PER_MILE) (4)

RESID_TOT = EXCESS/((BGN_MILE - END_MILE)*DIV_PER_MILE) (5)

$$\text{AVAIL\_FLOW}(i) = \text{AVAIL\_FLOW}(i-1)+\text{TRB\_FLOW}(i) + \text{RESID} \tag{6}$$

$$\text{TOT\_FLOW}(i) + \text{TOT\_FLOW}(i-1) + \text{TRB\_FLOW}(i) + \text{RESID\_TOT} \tag{7}$$

$$\text{MIN\_FLOW}(i) = \text{TOT\_FLOW}(i) - \text{AVAIL\_FLOW}(i) \tag{8}$$

The program was developed to require a minimal amount of input data. The intended uses are to approximate the quantity of water available for diversion from a stream segment and to maintain a data file which can be used repetitively.

The program is intended for steady-state applications and will provide better results on relatively short reaches where travel times are less significant and the user-defined minimums do not differ greatly. Inflow (from adjacent aquifers or overland runoff) or outflow (to adjacent aquifers) is computed as a net residual and distributed linearly throughout the stream reach.

The program estimates are based on flowing water in the stream and do not account for, nor consider available, any volume of water in storage as a result of backwater conditions.

Application of Computer Program

The program begins with the data entry which establishes a data file from which repetitive runs can be made. The program allows changes to be made in the data file so that variations of the basic data can be run through the program.

The sum of all tributaries and diversions occurring within a computation interval must be used when inputting diversions and tributaries. For example, consider two 5 ft^3/s diversions located at stream miles 10.2 and 10.4. If the computation interval is 0.5 mile, then -10 ft^3/s would be entered for mile location 10.0.

The program is run to estimate the streamflow available for diversion throughout a specified reach after data input or modifications are complete. The estimates are displayed on the terminal screen and optionally to a data file for further reference.

To illustrate the use of the program, an example using data for the Little Red River on July 29, 1986, has been run through the program with a computation interval of 1 mile. Figure 10 is the reference station data for that date. It should be noted that the minimum of 50 ft^3/s to be maintained was arbitrarily chosen because no minimum flow has been declared by any local, State, or Federal agency. Also, the value of observed flow at point B was selected arbitrarily as no data were available for a downstream station on the Little Red River. Figure 11 displays, in a tabular format, the programs representation of flow conditions in the river based upon existing conditions and selected minimum flows as defined above.

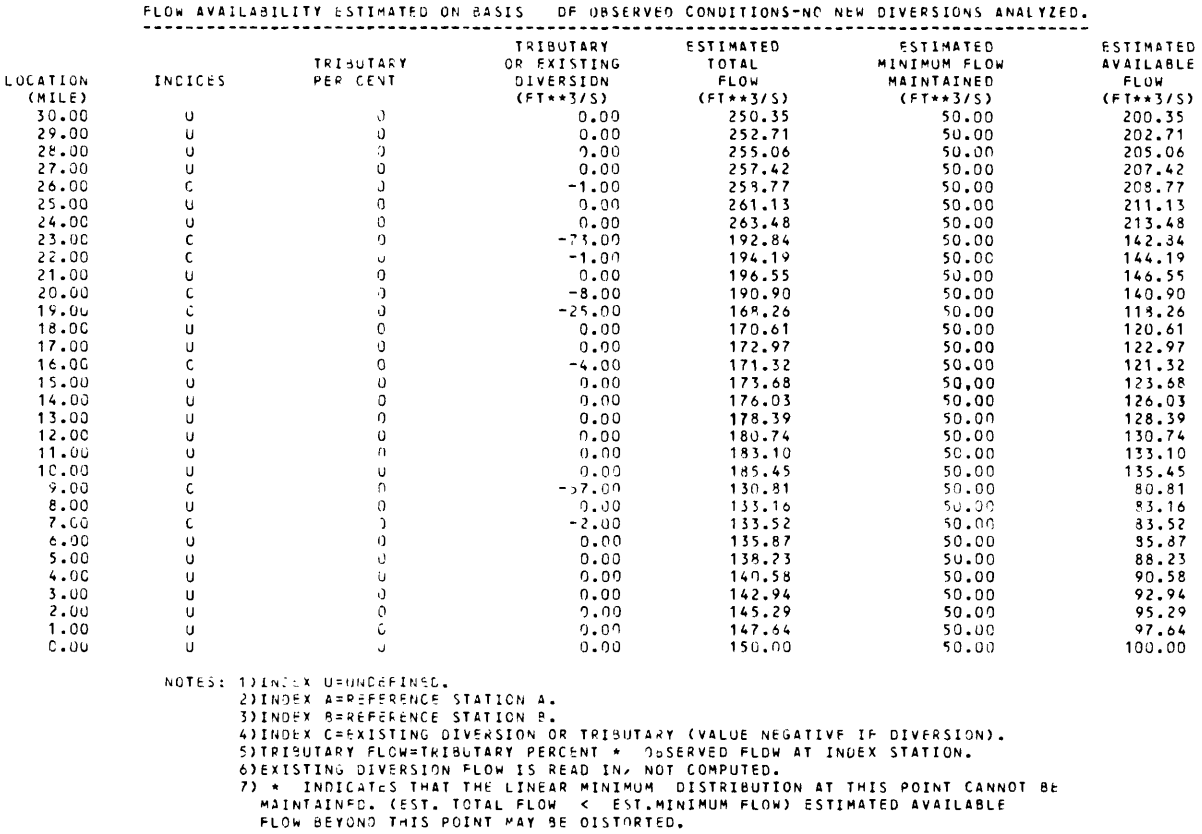
```
                                                   REFERENCE STATION INFORMATION                     FILE NAME: STMDATA
==========================================================================================================================
                                                                  ==
                   STATION A(UPSTREAM)                            ==                STATION B(DOWNSTREAM)
                                                                  ==
                                                                  ==
==========================================================================================================================
      OBSERVED       SPECIFIED       AVAILABLE                    ==      OBSERVED       SPECIFIED       AVAILABLE
        FLOW          MINIMUM          FLOW         LOCATION      ==        FLOW          MINIMUM          FLOW      LOCATION
     (FT**3/S)       (FT**3/S)       (FT**3/S)       (MILE)       ==     (FT**3/S)       (FT**3/S)       (FT**3/S)    (MILE)
        248.00           50.00          198.00         31.00      ==        150.00           50.00          100.00       0.00
==========================================================================================================================

***NOTE***
RESIDUAL FROM DISCHARGE BALANCE BETWEEN STATION A AND STATION B IS DISTRIBUTED LINEARLY.

DIVISION PER MILE IS   1

ENTER (CR) TO CONTINUE:
```

Figure 10.--Example of program output of reference station data.



```
             FLOW AVAILABILITY ESTIMATED ON BASIS   OF OBSERVED CONDITIONS-NO NEW DIVERSIONS ANALYZED.
             -----------------------------------------------------------------------------------------
                                             TRIBUTARY       ESTIMATED         ESTIMATED           ESTIMATED
                                TRIBUTARY   OR EXISTING        TOTAL        MINIMUM FLOW           AVAILABLE
LOCATION      INCICES           PER CENT     DIVERSION         FLOW          MAINTAINED              FLOW
 (MILE)                                      (FT**3/S)       (FT**3/S)       (FT**3/S)             (FT**3/S)
  30.00          U                  0           0.00          250.35            50.00                200.35
  29.00          U                  0           0.00          252.71            50.00                202.71
  28.00          U                  0           0.00          255.06            50.00                205.06
  27.00          U                  0           0.00          257.42            50.00                207.42
  26.00          C                  0          -1.00          258.77            50.00                208.77
  25.00          U                  0           0.00          261.13            50.00                211.13
  24.00          U                  0           0.00          263.48            50.00                213.48
  23.00          C                  0         -73.00          192.84            50.00                142.34
  22.00          C                  0          -1.00          194.19            50.00                144.19
  21.00          U                  0           0.00          196.55            50.00                146.55
  20.00          C                  0          -8.00          190.90            50.00                140.90
  19.00          C                  0         -25.00          168.26            50.00                118.26
  18.00          U                  0           0.00          170.61            50.00                120.61
  17.00          U                  0           0.00          172.97            50.00                122.97
  16.00          C                  0          -4.00          171.32            50.00                121.32
  15.00          U                  0           0.00          173.68            50,00                123.68
  14.00          U                  0           0.00          176.03            50.00                126.03
  13.00          U                  0           0.00          178.39            50.00                128.39
  12.00          U                  0           0.00          180.74            50.00                130.74
  11.00          U                  0           0.00          183.10            50.00                133.10
  10.00          U                  0           0.00          185.45            50.00                135.45
   9.00          C                  0         -57.00          130.81            50.00                 80.81
   8.00          U                  0           0.00          133.16            50.00                 83.16
   7.00          C                  0          -2.00          133.52            50.00                 83.52
   6.00          U                  0           0.00          135.87            50.00                 85.87
   5.00          U                  0           0.00          138.23            50.00                 88.23
   4.00          U                  0           0.00          140.58            50.00                 90.58
   3.00          U                  0           0.00          142.94            50.00                 92.94
   2.00          U                  0           0.00          145.29            50.00                 95.29
   1.00          U                  0           0.00          147.64            50.00                 97.64
   0.00          U                  0           0.00          150.00            50.00                100.00

               NOTES: 1)INDEX U=UNDEFINED.
                      2)INDEX A=REFERENCE STATION A.
                      3)INDEX B=REFERENCE STATION B.
                      4)INDEX C=EXISTING DIVERSION OR TRIBUTARY (VALUE NEGATIVE IF DIVERSION).
                      5)TRIBUTARY FLOW=TRIBUTARY PERCENT *  OBSERVED FLOW AT INDEX STATION.
                      6)EXISTING DIVERSION FLOW IS READ IN, NOT COMPUTED.
                      7) *  INDICATES THAT THE LINEAR MINIMUM  DISTRIBUTION AT THIS POINT CANNOT BE
                        MAINTAINED. (EST. TOTAL FLOW  <  EST.MINIMUM FLOW) ESTIMATED AVAILABLE
                        FLOW BEYOND THIS POINT MAY BE DISTORTED.
```

Figure 11.--Example of program output for analysis with no diversions in place.

An added feature of the program allows for new diversions to be added and their effect on points downstream estimated. New diversions are input much the same as existing diversions except that the amount of diversion is specified as a percentage of the available flow at the location of the new diversion.

As an example, assume we wish to examine the effect of a proposed new diversion to be located at stream mile 2, and that the diversion will be 5 percent of the available flow. Figure 12 displays the programs representation of flow conditions based upon previously described data and the implacement of this new diversion.

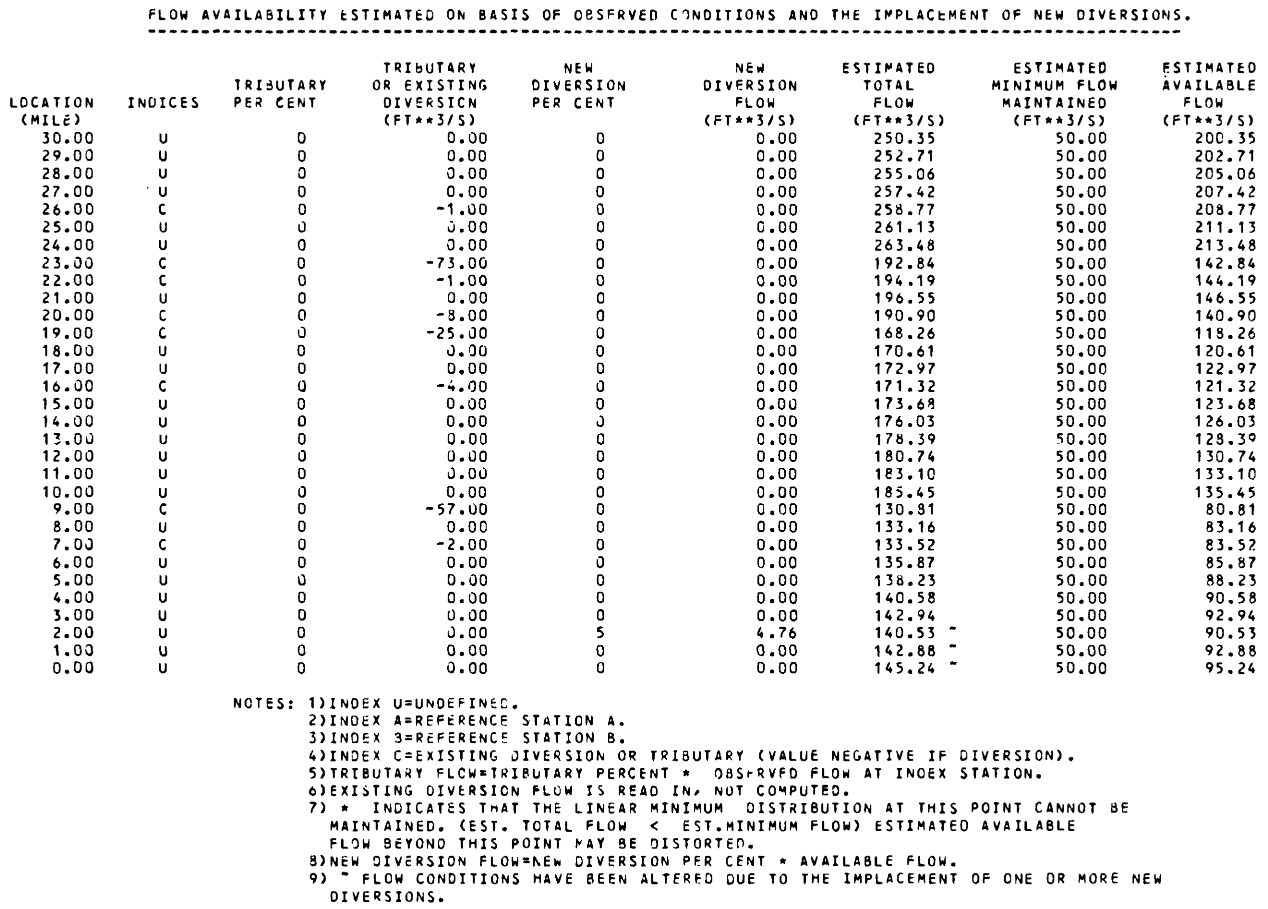
SUMMARY

A stream-stage recorder was installed near Searcy to complement the recorder at Judsonia. A stage/fall/discharge relation was developed for the Little Red River near Searcy using actual and simulated stage/discharge data for the Searcy station. A stage recorder was temporarily installed at Nimmo (near the mouth of the Little Red River), but a stage/discharge relation could not be derived for this site.

Ground-water levels were measured monthly in 17 area wells to develop a local potentiometric surface. Water levels in these wells dropped 1 to 10 feet from April to July 1984. In addition, two wells were equipped with continuous water-level recorders. Data from these recorders showed the Little Red to be a gaining stream during low flow and a losing stream during periods of high flow.

Data on diversion pumping were acquired by using pump activity data from local observers. These data showed a total potential diversion of 269 ft^3/s. During 1984 the maximum actual diversion was estimated to be 219 ft^3/s. This is greater than the specified minimum release from Greers Ferry Dam (115 ft^3/s) and also greater than the lowest streamflow during the summer of 1984 (195 ft^3/s). This shows that total diversion can be greater than streamflow.

A computer program was developed for estimating the amount of streamflow available for diversion along a reach of a stream. An option in the program allows new diversions to be added to examine their effect on points downstream. The minimum hydrologic data necessary for using the program are observed and specified minimum streamflow for upstream and downstream reference points and a complete inventory of tributaries and diversions (both location and discharge).



```
                 FLOW AVAILABILITY ESTIMATED ON BASIS OF OBSFRVED CONDITIONS AND THE IMPLACEMENT OF NEW DIVERSIONS.
                 -------------------------------------------------------------------------------------------------

                                  TRIBUTARY        NEW              NEW       ESTIMATED      ESTIMATED       ESTIMATED
                       TRIBUTARY OR EXISTING    DIVERSION        DIVERSION      TOTAL      MINIMUM FLOW      AVAILABLE
LOCATION   INDICES     PER CENT   DIVERSION     PER CENT           FLOW         FLOW        MAINTAINED         FLOW
 (MILE)                           (FT**3/S)                     (FT**3/S)    (FT**3/S)      (FT**3/S)       (FT**3/S)
   30.00      U           0          0.00          0               0.00        250.35          50.00          200.35
   29.00      U           0          0.00          0               0.00        252.71          50.00          202.71
   28.00      U           0          0.00          0               0.00        255.06          50.00          205.06
   27.00      U           0          0.00          0               0.00        257.42          50.00          207.42
   26.00      C           0         -1.00          0               0.00        258.77          50.00          208.77
   25.00      U           0          0.00          0               0.00        261.13          50.00          211.13
   24.00      U           0          0.00          0               0.00        263.48          50.00          213.48
   23.00      C           0        -73.00          0               0.00        192.84          50.00          142.84
   22.00      C           0         -1.00          0               0.00        194.19          50.00          144.19
   21.00      U           0          0.00          0               0.00        196.55          50.00          146.55
   20.00      C           0         -8.00          0               0.00        190.90          50.00          140.90
   19.00      C           0        -25.00          0               0.00        168.26          50.00          118.26
   18.00      U           0          0.00          0               0.00        170.61          50.00          120.61
   17.00      U           0          0.00          0               0.00        172.97          50.00          122.97
   16.00      C           0         -4.00          0               0.00        171.32          50.00          121.32
   15.00      U           0          0.00          0               0.00        173.68          50.00          123.68
   14.00      U           0          0.00          0               0.00        176.03          50.00          126.03
   13.00      U           0          0.00          0               0.00        178.39          50.00          128.39
   12.00      U           0          0.00          0               0.00        180.74          50.00          130.74
   11.00      U           0          0.00          0               0.00        183.10          50.00          133.10
   10.00      U           0          0.00          0               0.00        185.45          50.00          135.45
    9.00      C           0        -57.00          0               0.00        130.81          50.00           80.81
    8.00      U           0          0.00          0               0.00        133.16          50.00           83.16
    7.00      C           0         -2.00          0               0.00        133.52          50.00           83.52
    6.00      U           0          0.00          0               0.00        135.87          50.00           85.87
    5.00      U           0          0.00          0               0.00        138.23          50.00           88.23
    4.00      U           0          0.00          0               0.00        140.58          50.00           90.58
    3.00      U           0          0.00          0               0.00        142.94          50.00           92.94
    2.00      U           0          0.00          5               4.76        140.53 "        50.00           90.53
    1.00      U           0          0.00          0               0.00        142.88 "        50.00           92.88
    0.00      U           0          0.00          0               0.00        145.24 "        50.00           95.24

                 NOTES: 1)INDEX U=UNDEFINED.
                        2)INDEX A=REFERENCE STATION A.
                        3)INDEX 3=REFERENCE STATION B.
                        4)INDEX C=EXISTING DIVERSION OR TRIBUTARY (VALUE NEGATIVE IF DIVERSION).
                        5)TRIBUTARY FLOW=TRIBUTARY PERCENT *  OBSERVED FLOW AT INDEX STATION.
                        6)EXISTING DIVERSION FLOW IS READ IN, NOT COMPUTED.
                        7) *  INDICATES THAT THE LINEAR MINIMUM  DISTRIBUTION AT THIS POINT CANNOT BE
                          MAINTAINED. (EST. TOTAL FLOW  <  EST.MINIMUM FLOW) ESTIMATED AVAILABLE
                          FLOW BEYOND THIS POINT MAY BE DISTORTED.
                        8)NEW DIVERSION FLOW=NEW DIVERSION PER CENT * AVAILABLE FLOW.
                        9) " FLOW CONDITIONS HAVE BEEN ALTERED DUE TO THE IMPLACEMENT OF ONE OR MORE NEW
                          DIVERSIONS.
```

Figure 12.--Example of program output for analysis with the implacement of a new diversion.

REFERENCES CITED

Counts, H.B., 1957, Ground water resources of parts of Lonoke, Prairie, and White Counties, Arkansas: Arkansas Geological and Conservation Commission Water Resources Circular No. 5, 65 p.

Shearman, J.O., 1976, Computer applications for step-backwater and floodway analyses: U.S. Geological Survey Open-File Report, 103 p.

Sullavan, J.N., 1974, Drainage areas of streams in Arkansas, White River basin: U.S. Geological Survey Open-File Report, 123 p.

ATTACHMENT A

```
      PROGRAM MNPROG
C     *******************************************************************
C     *
C     *                       STREAMFLOW PROGRAM
C     *
C     *           (Developed for the Little Red River Project)
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *   AUTHORS: JOHN E. TERRY AND BARRY LYLE
C     *   DATE: 12/9/86
C     *
C     *   Converted from original PL/1 by C. R. Baxter  1/87-6/87
C     *
C     *   PURPOSE:
C     *
C     *            The program estimates the availability of
C     *            of a stream flow and maintains flow information
C     *            for the stream.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *    ALGORITHM FOR THE MAIN PROGRAM:
C     *
C     *      I. START THE PROCESS.
C     *         A. CALL A PROCEDURE TO INITIALIZE VARIBLES
C     *            AND THE TEMPORARY WORK FILE FOR FLOW INFORMATION.
C     *         B. WHILE THERE IS NO INDICATION TO STOP PROCESSING,
C     *            DO THE FOLLOWING:
C     *
C     *            1. INDICATE THE TYPE OF PROCESSING TO BE PERFORMED.
C     *            2. IF TRIBUTARIES OR DIVERSIONS ARE TO BE CHANGED,
C     *               ENTER THE POSITION OF THE TRIBUTARY OR DIVERSION
C     *               TO CHANGE AND PROCESS.
C     *            3. IF THE BEGINNING OR ENDING REFERENCES ARE TO
C     *             BE CHANGED, PROCESS.
C     *            4. IF WATER AVAILIBILITY DETERMINED, PROCESS.
C     *            5. IF PROCESSING TO STOP,SAVE INFO & CLOSE FILES.
C     *            6. GIVE STATION INFORMATION .
C     *            7. WRONG OPTION WAS ENTERED,TRY AGAIN.
C     *            8. REPEAT STEP B IF B.5 NOT INDICATED.
C     *
C     *         C. DECIDE IF WISH TO CONTINUE WORKING WITH AN OLD
C     *            FILE OR CREATE NEW FILE OR STOP PROCESSING.
C     *         D. REPEAT STEPS I.A THRU I.C IF PROCESSING TO RESUME.
C     *         E. IF WRONG OPT. OR STOP INDICATED,STOP PROCESSING.
C     *     II. DELETE THE TEMPORARY WORK FILE.
C     *    III. STOP THE PROCESS.
C     *
C     ********************************************************************
```

```
C     *
C     *      INDICATION CODES:
C     *
C     *         Y ---------- YES
C     *         N ---------- NO
C     *
C     *         C ---------- TO CHANGE ANY DIVERSIONS OR TRIBUTARIES.
C     *         R ---------- TO CHANGE BEGINNING OR ENDING REFERENCES
C     *         A ---------- TO DETERMINE AMOUNT OF WATER AVAILABLE.
C     *         I ---------- TO GIVE THE STATION INFORMATION.
C     *         E ---------- TO END PROCESSING OF THE CURRENT DATA
C     *
C     *         *** ANY OTHER OPTION IS INVALID AND NO PROCESSING
C     *             WILL OCCUR
C     *******************************************************************
C     *******************************************************************
C     *
C     *    PROCEDURES CALLED BY THE MAIN PROGRAM:
C     *
C     *         ENTER ----------- CREATES NEW FILE AND INITIALIZES THE
C     *                           VARIABLES WITH DATA FROM THE SCREEN OR
C     *                           ACCESSES OLD FILE AND INITIALIZES THE
C     *                           VARIABLES FROM THE FILE.THE PROCEDURE
C     *                           ALSO INTIALIZES THE TEMPORARY WORK
C     *                           FILE WITH FLOW INFORMATION.
C     *
C     *
C     *         CHANGE ---------- MAKES CHANGES OF EXISTING TRIBUTARY OR
C     *                           DIVERSION FLOWS,OR ENTERS NEW
C     *                           DIVERSION FLOWS.
C     *
C     *         BGENCNG --------- CHANGES EITHER THE MAXIMUM FLOW OR
C     *                           MINIMUM FLOW AT STATION A OR B.
C     *
C     *         WTAVL ----------- DETERMINES THE AMOUNT OF WATER
C     *                           AVAILABLE WITH OR WITHOUT DIVERSIONS.
C     *
C     *         CLOSEF ---------- SAVES THE DATA IN PERMANENT FILE AND
C     *                           CLOSES THE FILES.
C     *
C     *         INFO ------------ PRINTS THE STATION INFORMATION
C     *                           ON THE SCREEN.
C     *         ENTR ------------ PRINTS A QUERY AND READS REAL NUMBER
C     *                           DATA FROM THE SCREEN.
C     *
C     *         CRWKFL ---------- CREATES A TEMPORARY WORK FILE.
C     *
C     *         CHECK ----------- CHECKS FOR THE EXISTENCE OF A POSITION
C     *                           IN TEMPORARY WORK FILE.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *      MAJOR VARIABLES:
```

```
C     *
C     *         BGREF --------- MAXIMUM UPSTREAM FLOW AT STATION A.
C     *         BGMN  --------- MINIMUM UPSTREAM FLOW AT STATION A.
C     *         ENREF --------- MAXIMUM DOWNSTREAM FLOW AT STATION B.
C     *         ENMN  --------- MINIMUM DOWNSTREAM FLOW AT STATION B.
C     *         BGN_MILE ------ STATION A LOCATION IN MILES.
C     *         END_MILE ------ STATION B LOCATION IN MILES.
C     *         DIV_PER_MILE -  DIVISIONS/MILE BETWEEN STATIONS A & B.
C     *         SUM_TRB ------- SUM OT THE TRIBUTARY FLOWS.
C     *         POS_SUM_TRB --  POSITIVE SUM OF THE TRIBUTARY FLOWS.
C     *         DATA  --------- REAL NUMBER DATA TO BE ENTERED.
C     *         POSITION ------ THE POSITION BETWEEN STATIONS A AND B
C     *                         TO BE ANALYZED.
C     *         ANSWER -------- BUFFER FOR OPTIONS.
C     *         QUES  --------- QUESTION/QUERY PRINTED AT TERMINAL.
C     *         FILE_NAME ----  NAME OF OLD FILE TO BE PROCESSED OR NEW
C     *                         FILE TO CREATE WITH STREAM FLOW DATA.
C     *         CFILE --------- COPY OF THE OLD FILE.
C     *
C     *******************************************************************

$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN
$INSERT SYSCOM>A$KEYS.INS.FTN

C     *** DECLARATIONS ***

      INTEGER                   FUNIT*2,
     *                          PUNIT*4,
     *                          ARRAY(14)*2,
     *                          LENGTH*2,
     *                          LEN*2,
     *                          STATUS*2,
     *                          FLAG*2,
     *                          FL*2,
     *                          MODE*2

      REAL                      BGREF,
     *                          BGMN,
     *                          ENREF,
     *                          ENMN,
     *                          BGN_MILE,
     *                          END_MILE,
     *                          DIV_PER_MILE,
     *                          SUM_TRB,
     *                          POS_SUM_TRB,
     *                          DATA,
     *                          POSITION


      CHARACTER                 ANSWER*1,
     *                          FANS*1,
     *                          CMD*80,
     *                          ANSWER1*1,
     *                         KEY*8,
     *                         FILE_NAME*32,
     *                         CFILE*60,
     *                         FLOW_INFO*40,
     *                         QUES*40,
     *                         PATHNAME*32

      EQUIVALENCE(FLOW_INFO(1:8),KEY)


      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN

      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE

      COMMON/THIRD/SUM_TRB,POS_SUM_TRB

      COMMON/FOURTH/POSITION

      COMMON/FIFTH/FILE_NAME,PUNIT

      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH


      COMMON/CFL/CFILE

C      ***QUERY FORMAT***

2000  FORMAT(//,T1,'ENTER ONE OF THE FOLLOWING OPTIONS:',
     *        /,T1,'"C"- TO CHANGE ANY DIVERSIONS OR TRIBUTARIES,',
     *        /,T1,'"R"- TO CHANGE BEGINNING OR ENDING REFERENCES,',
     *        /,T1,'"A"- TO DETERMINE THE AMOUNT OF WATER AVAILABLE,',
     *        /,T1,'"I"- TO GIVE THE STATION INFORMATION,',
     *        /,T1,'"E"- TO END PROCESSING OF THE CURRENT DATA,')

2001  FORMAT(10(/,' '))
2010  FORMAT(//,T1,'THE WRONG OPTION WAS ENTERED,TRY AGAIN!')

2020  FORMAT(//,T1,'IF YOU WISH TO CONTINUE WORKING WITH AN ',
     *        /,T1,'OLD FILE OR CREATE A NEW FILE, ENTER "Y",')

2030  FORMAT(//,T1,'YOU ENTERED THE WRONG ANSWER!!',
     *        /,T1,'ALL PROCESSING WILL STOP!')


2040  FORMAT(//,T2,'ENTER (CR) TO CONTINUE: ')
```

```
C     *** ANSWER IS INITIALIZED TO "Y" TO BEGIN PROCESSING. ***


      PUNIT = 80
      PATHNAME = 'FLOW.DAT'


C***** CREATE THE TEMPORARY WORK FILE  FLOW.DAT *****


      CALL CRWKFL
      LENGTH = 8
      MODE = K$GETU +K$RDWR
      ANSWER = 'Y'


10    IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN




C*****CALL ENTER PROCEDURE TO OPEN FILES AND INITIALIZE VARIABLES
C*****AND WORK FILE

         CALL ENTER


C***** IF THE FILE NAME IS GIVEN AS "END" THEN *****
C***** CLOSE WORK FILE AND EXIT PROGRAM        *****

         IF (FILE_NAME .EQ. 'END') THEN
            CALL CLOSM$(FUNIT,STATUS)
            GO TO 3500
         ENDIF

         FL = 0




C***** PROCESS UNTIL  ANSWER = E (STOP) *****


20       IF ((ANSWER .NE. 'E') .AND. (ANSWER .NE. 'e')) THEN
            ASSIGN 100 TO LABEL


C***** PRINT OPTION MENU *****
```

```
100         CALL TONL
            CALL TNOUA('ENTER (CR) TO CONTINUE: ',INTS(24))
            READ '(A)',ANSWER1
            IF (ANSWER1 .EQ. ' ') THEN
               CALL CLEAR
               CALL HEAD
            ELSE
               GO TO 100
            ENDIF


             WRITE(1,2000)
             CALL TNOUA('ENTER OPTION: ',INTS(14))
             READ '(A)',ANSWER

C>>>>>      CHANGE TRIBUTARY OR DIVERSION VALUES AT GIVEN POSITION

             IF ((ANSWER .EQ. 'C') .OR. (ANSWER .EQ. 'c')) THEN
                QUES = 'ENTER POSITION TO CHANGE: '
                LEN = 27
                CALL ENTR(QUES,DATA,LEN)
                POSITION = DATA
                   CALL RNF(POSITION,FL)
                IF (FL .EQ. 0) THEN
                   CALL CHANGE(POSITION,ARRAY)
                ENDIF
                   FL = 0
             ELSE
C>>>>>          CHANGE MAXIMUM OR MINIMUM FLOWS AT A GIVEN REFERENCE

                IF ((ANSWER .EQ. 'R') .OR. (ANSWER .EQ. 'r')) THEN
                   CALL BGENCNG
                ELSE

C>>>>>            DETERMINE STREAM AVAILABILITY

                    IF ((ANSWER .EQ. 'A') .OR. (ANSWER .EQ. 'a'))THEN
                       CALL WTAVL
                    ELSE

C>>>>>                 STOP AND CLOSE FILES

                       IF ((ANSWER .EQ. 'E') .OR. (ANSWER .EQ. 'e')) THEN
                          CALL CLOSEF
                       ELSE

C>>>>>                    PRINT STATION INFORMATION AT TERMINAL

                          IF ((ANSWER .EQ. 'I') .OR. (ANSWER .EQ. 'i'))
     *                    THEN
                             CALL INFO
                          ELSE
```

```
C>>>>>                      INVALID ANSWER MESSAGE

                            WRITE(1,2010)
                         ENDIF
                      ENDIF
                   ENDIF
                ENDIF
             ENDIF

C>>>>>       IF RECORD NOT FOUND,CREATE POSITION OR CHOOSE NEW OPTION


1000         IF (FL .EQ. 1) THEN
                FL = 0
             ENDIF
             GO TO 20
          ENDIF


C>>>      PROMPT TO RETAIN COPY OF ORIG. INPUT FILE OR DELETE IT.

588       CALL TNOU('ENTER "Y" IF YOU WISH TO RETAIN A COPY',INTS(38))
          CALL TNOUA('OF YOUR ORIGINAL INPUT FILE, ELSE(CR):',INTS(38))
          READ '(A)',FANS
          IF (FANS .EQ. ' ') THEN
             IF (CFILE .NE. ' ') THEN
                OPEN(11,FILE=CFILE)
                CLOSE(11,STATUS='DELETE')
             ENDIF
          ELSE
             IF ((FANS .NE. 'Y') .AND. (FANS .NE. 'y')) THEN
                CALL TNOU('INVALID RESPONSE')
                CALL TONL
                GO TO 588
             ENDIF
          ENDIF

C>>>      PROMPT TO REPEAT PROGRAM WITH NEW OR OLD FILE OR EXIT
C>>>      PROGRAM.
          WRITE(1,2020)
          CALL TNOUA('ELSE (CR): ',INTS(12))
          READ '(A)',ANSWER




          GO TO 10
          ENDIF




          IF (ANSWER .NE. ' ') THEN
```

```
              WRITE(1,2030)

       ENDIF



C>>>   PROGRAM ENDING
3500   CALL CLEAR
       WRITE(1,3222)
3222   FORMAT('1m')
       WRITE(1,3221)
3221   FORMAT(10(/),T20,'?31STREAMFLOW AVAILABILITY PROGRAM ENDED')
       WRITE(1,3223)
3223   FORMAT('0m')
       WRITE(1,'(10(/))')

       CALL EXIT
3600   END
```

```
      SUBROUTINE ENTER
C     *******************************************************************
C     *
C     *                     ENTER PROCEDURE
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date:  5/30/87
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  Purpose:
C     *
C     *            The procedure either initializes the temporary
C     *            work file with the contents of an old file or
C     *            creates a new file which contents are also in the
C     *            temporary work file.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  PROCEDURES CALLED:
C     *
C     *                     HEAD -- Prints program headerato screen.
C     *
C     *                     CLEAR -- Clears the screen.
C     *
C     *                     CRTMP -- Creates a copy of the input file
C     *                              with date/time tag.
C     *
C     *                     CHECK -- Checks if given position within
C     *                              range of the stream.
C     *
C     *                     RNF -- Checks the existence of a position
C     *                            in temporary work file.
C     *
C     *******************************************************************
C     *******************************************************************


C     ***MIDAS DECLARATIONS***


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN
```

```
C     *** DECLARATIONS ***

      INTEGER*2                STATUS,
     *                         FLG,
     *                         F,
     *                         I,
     *                         FL,
     *                         FLAGS,
     *                         OFLAG,
     *                         AFLAG,
     *                         LENGTH,
     *                         LEN,
     *                         MODE,
     *                         ARRAY(14),
     *                         FUNIT

      INTEGER*4                MORE_DATA,PUNIT

      REAL                     BGREF,
     *                         BGMN,
     *                         ENREF,
     *                         ENMN,
     *                         BGN_MILE,
     *                         END_MILE,
     *                         DIV_PER_MILE,
     *                         SUM_TRB,
     *                         POS_SUM_TRB,
     *                         FLOW_POSITION,
     *                         POSITION,
     *                         POS,
     *                         TRB_FLOW_PCT,
     *                         TRB_FLOW,
     *                         DVR_FLOW_PCT,
     *                         EMP_FLOW_POS,
     *                         EMP_FLOW_PCT,
     *                         EMP_TRB_FLOW,
     *                         EMP_DVR_PCT,
     *                         DATA


      CHARACTER                FILE_NAME*32,
     *                         CFILE*60,
     *                         ANSWER*3,
     *                         TRB_FLOW_STAT*1,
     *                         EMP_FLOW_STAT*1,
     *                         ANSSAVE*3,
     *                         EMP_FLOW*40,
     *                         EKEY*8,
     *                         FLOW_INFO*40,
     *                         KEY*8,
     *                         CHECK_FLG*3,
     *                         PATHNAME*32,
     *                                  QUES*40



      EQUIVALENCE (EMP_FLOW(1:8),EKEY)

      EQUIVALENCE (FLOW_INFO(1:8),KEY)



      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN

      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE

      COMMON/THIRD/SUM_TRB,POS_SUM_TRB

      COMMON/FIFTH/FILE_NAME,PUNIT

      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH

      COMMON/CFL/CFILE



C     **** INPUT FORMAT ****

1010  FORMAT(T1,F11.0,F11.0,F11.0,F11.0)

1020  FORMAT(T1,F7.0,F7.0,F4.0)

1030  FORMAT(T1,F7.0,F5.0,A1,F11.0,F5.0)



C     **** QUERY AND OUTPUT FORMAT ****
```

```
2010  FORMAT(F7.2,F4.2,A1,F10.4,F4.2)

2020  FORMAT(//,T1,'ENTER "OLD" TO ACCESS OLD FILES OR',
     *        /,T1,'"NEW" TO CREATE NEW FILES: ')

2060  FORMAT(//,T1,'YOU ARE READY TO ENTER THE TRIBS AND DIVERSIONS.',
     *        /,T1,' ',/,T1,'WHEN EVERTHING HAS BEEN ENTERED,',
     *        /,T1,'ENTER 9999 FOR THE FLOW POSITION(MILE).')

2070  FORMAT(//,T1,'THE GIVEN FILE DOES NOT EXIST, PLEASE REENTER.')

2080  FORMAT(//,T1,'ERROR AT POSITION',F6.2,' MIDAS ERROR ',I5)

2110  FORMAT(//,T1,'STATION LOCATIONS HAVE BEEN ENTERED INCORRECTLY.',
     *        /,T1,'STATION A LOCATION MUST BE UPSTREAM ( A LARGER ',
     *        /,T1,'NUMBER) FROM STATION B. PLEASE RE-ENTER.',
     *       //,T1,' ')


2120  FORMAT(//,T1,'THE FILE NOT VALID FOR THIS PROGRAM,PLEASE ENTER',
     *        /,T1,'A NEW FILE NAME OR ENTER "END" TO EXIT',
     *        /,T1,'THE PROGRAM: ')


C>>>>>INITIALIZE

      PATHNAME = 'FLOW.DAT'
      LENGTH = 32
      PUNIT = 80
      SUM_TRB = 0
      POS_SUM_TRB = 0
      OFLAG = 0

      EMP_FLOW = ' '
      EMP_FLOW_PCT = 0.0
      EMP_TRB_FLOW =0.0
      EMP_FLOW_STAT = 'U'

      EMP_DVR_PCT = 0.0


C>>>>>PRINT HEADER AND INDICATE IF USING OLD OR NEW FILE

      CALL CLEAR
      CALL HEAD
10    CALL TNOU('ENTER "OLD" TO ACCESS OLD FILES,', INTS(32))
      CALL TNOUA('OR "NEW" TO CREATE NEW FILES: ',INTS(30))
      READ '(A)', ANSWER
      ANSSAVE = ANSWER
      F = 0
      QUES = 'ENTER FILE NAME:'

      LEN = 17
C     ******OPEN STRM_FLOW DIRECT ACCESS FILE HERE******

      CALL OPENM$(MODE,PATHNAME,LENGTH,FUNIT,STATUS)
      IF (STATUS .NE. 0) THEN
         PRINT '(A)','ERROR IN OPENING'
         PRINT '(I6)',STATUS
         GO TO 3500
      ENDIF



C****>PROCESSING OLD FILE

      IF ((ANSWER .EQ. 'OLD') .OR. (ANSWER .EQ. 'old')) THEN

C>>>>>    CHECK FOR EXISTENCE OF THE FILE

         OFLAG = 1
17       FLG = 1
         PUNIT = 80
         CALL FILECK(QUES,FILE_NAME,FLG,LEN)
         IF ((FILE_NAME .EQ. 'END') .OR. (FILE_NAME .EQ. 'end')) THEN
            GO TO 3500
         ENDIF


         CALL CRTMP(FILE_NAME,CFILE)
C>>>>> INITIALIZE FLOW VARIABLES

         TRB_FLOW = 0.00


C     ******OPEN PRMFILE STREAM FILE HERE******
         OPEN(UNIT=PUNIT,FILE = FILE_NAME,FORM = 'FORMATTED',
     *        ACCESS = 'SEQUENTIAL')



C>>>>>    INITIALIZE REFERENCE VARIABLES

         READ(PUNIT,1010,IOSTAT = MORE_DATA,ERR = 3000,END = 3000)
     *                                BGREF,BGMN,ENREF,ENMN
         READ(PUNIT,1020,ERR = 3000,END = 3500)BGN_MILE,END_MILE,
     *                                       DIV_PER_MILE


C>>>>>    READ FIRST RECORD OF FLOW INFORMATION
```

```
         READ(PUNIT,1030,ERR = 3000,IOSTAT = MORE_DATA) POS,
     *                           TRB_FLOW_PCT,TRB_FLOW_STAT,TRB_FLOW,
     *                           DVR_FLOW_PCT


C>>>>>   INITIALIZE WORK FILE WITH FLOW INFORMATION

         DUMMY = 1/DIV_PER_MILE
         POSITION = BGN_MILE
         FLAGS = FL$RET
15       IF (POSITION .GE. END_MILE) THEN
            EMP_FLOW_POS = POSITION
            WRITE(EMP_FLOW(1:8),'(F8.2)') POSITION
            WRITE(EMP_FLOW(9:12),'(F4.2)') EMP_FLOW_PCT
            WRITE(EMP_FLOW(16:16),'(A1)') EMP_FLOW_STAT
            WRITE(EMP_FLOW(17:26),'(F10.2)') EMP_TRB_FLOW
            WRITE(EMP_FLOW(27:30),'(F4.2)') EMP_DVR_PCT
            CALL ADD1$(FUNIT,EMP_FLOW,EKEY,ARRAY,FLAG,
     *                  INTS(0),INTS(0),INTS(0),
     *                 INTS(0),INTS(0))
            IF (ARRAY(1) .NE. 0) THEN
               PRINT '(A)', 'ERROR IN ADDING RECORD'
               PRINT '(I5)',ARRAY(1)

               GO TO 3500
            ENDIF
            POSITION = POSITION - DUMMY
            GO TO 15
         ENDIF


C>>>>>   PLACE PERMANENT FLOW RECORDS IN WORK FILE

         CHECK_FLG = 'YES'
20       IF (MORE_DATA .EQ. 0) THEN

C>>>>>      CHECK POSITION FOR RANGE

            CALL CHECK(POS,CHECK_FLG)

C>>>>>      IF POSITION IN RANGE, PUT RECORD IN WORK FILE

            IF (CHECK_FLG .EQ. 'YES') THEN
               IF (TRB_FLOW .GT. 0) THEN
                  POS_SUM_TRB = POS_SUM_TRB + TRB_FLOW
               ENDIF
               SUM_TRB = SUM_TRB + TRB_FLOW

               FLAGS = FL$RET + FL$UKY
               FLOW_INFO = ' '
               WRITE(FLOW_INFO(1:8),'(F8.2)') POS
               CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAGS,
     *                    INTS(0),INTS(0),
     *                    INTS(0),INTS(0),INTS(0))
```

```
C>>>>>         INSERT RECORD IS POSITION DOES NOT EXIST

               IF (ARRAY(1) .EQ. 7) THEN
                  WRITE(FLOW_INFO(9:12),'(F4.2)') TRB_FLOW_PCT
                  WRITE(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
                  WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW
                  WRITE(FLOW_INFO(27:30),'(F4.2)') DVR_FLOW_PCT
                  FL = FL$RET
                  CALL ADD1$(FUNIT,EMP_FLOW,EMP_FLOW_POS,ARRAY,FL,
     *                        INTS(0),INTS(0),INTS(0),INTS(0),
     *                        INTS(0))
               ELSE

C>>>>>             UPDATE RECORD WITH FLOW INFO IF POSITION EXISTS
C>>>>>             IN WORK FILE.
                  IF ((ARRAY(1) .EQ. 0) .OR. (ARRAY(1) .EQ. 10)) THEN
                     WRITE(FLOW_INFO(1:8),'(F8.2)') POS
                     DO 7 I = 1,14
                        ARRAY(I) = 0
7                    CONTINUE
                     FL = FL$RET
                     CALL LOCK$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                       INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
                      WRITE(FLOW_INFO(9:12),'(F4.2)') TRB_FLOW_PCT
                     WRITE(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
                     WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW
                     WRITE(FLOW_INFO(27:30),'(F4.2)') DVR_FLOW_PCT
                     FL = FL$USE
                     CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                        INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
                     FL = FL$ULK + FL$USE
                     CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                       INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
                  ELSE
                     WRITE(1,2080) POS,ARRAY(1)
                     GO TO 3500
                  ENDIF
               ENDIF


               READ(PUNIT,1030,IOSTAT = MORE_DATA) POS,TRB_FLOW_PCT,
     *                         TRB_FLOW_STAT,TRB_FLOW,DVR_FLOW_PCT


            ELSE

               GO TO 3000

            ENDIF

            GO TO 20

         ENDIF
```

```
               CLOSE(PUNIT)

               OFLAG = 0

            ELSE


C****>         PROCESSING A NEW FILE

               IF ((ANSWER .EQ. 'NEW') .OR. (ANSWER .EQ. 'new')) THEN


C>>>>>             CHECK FOR FILE EXISTENCE

                   FLG = 2
                   PUNIT = 80
                   CALL FILECK(QUES,FILE_NAME,FLG,LEN)


                   IF ((FILE_NAME .EQ. 'END') .OR. (FILE_NAME .EQ. 'end'))
     *             THEN
                      GO TO 3500
                   ENDIF
                   OPEN(PUNIT,FILE = FILE_NAME, FORM = 'FORMATTED',
     *                  ACCESS = 'SEQUENTIAL')



C>>>>>             INITIALIZE THE REFERENCE DATA FOR THE NEW FILE

                   QUES ='ENTER STATION A(UPSTREAM) FLOW:'
                   LEN = 32
                   CALL ENTR(QUES,DATA,LEN)
                   BGREF = DATA
                   QUES = 'ENTER STATION A MINIMUM FLOW:'
                   LEN = 30
                   CALL ENTR(QUES,DATA,LEN)
                   BGMN = DATA
                   QUES = 'ENTER STATION B(DOWNSTREAM) FLOW:'
                   LEN = 34
                   CALL ENTR(QUES,DATA,LEN)
                   ENREF = DATA
                   QUES = 'ENTER STATION B MINIMUM FLOW:'
                   LEN = 30
                   CALL ENTR(QUES,DATA,LEN)
                   ENMN = DATA
 50                QUES = 'ENTER STATION A LOCATION(MILE):'
                   LEN = 32
                   CALL ENTR(QUES,DATA,LEN)
                   BGN_MILE = DATA
                   QUES = 'ENTER STATION B LOCATION(MILE):'
                   CALL ENTR(QUES,DATA,LEN)
```

```
C>>>>>      BGN_MILE HAS TO BE GREATER THAN END_MILE

            IF (DATA .GE. BGN_MILE) THEN
               WRITE(1,2110)
               GO TO 50
            ENDIF
            END_MILE = DATA
            QUES = 'ENTER DIVISIONS/MILE:'
            LEN = 22
            CALL ENTR(QUES,DATA,LEN)
            DIV_PER_MILE = DATA

C>>>>>      INITIALIZE WORK FILE

            POSITION = BGN_MILE
            DUMMY = 1/DIV_PER_MILE
            FLAGS = FL$RET
16          IF (POSITION .GE. END_MILE) THEN
               WRITE(EMP_FLOW(1:8),'(F8.2)') POSITION
               WRITE(EMP_FLOW(9:12),'(F4.2)') EMP_FLOW_PCT
               WRITE(EMP_FLOW(16:16),'(A1)') EMP_FLOW_STAT
               WRITE(EMP_FLOW(17:26),'(F10.2)') EMP_TRB_FLOW
               WRITE(EMP_FLOW(27:30),'(F4.2)') EMP_DVR_PCT
               CALL ADD1$(FUNIT,EMP_FLOW,EKEY,ARRAY,FLAGS,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
               POSITION = POSITION - DUMMY
               GO TO 16
            ENDIF


C>>>>>      ENTER THE FLOW POSITIONS FOR TRIBUTARIES AND DIVERSIONS

            WRITE(1,2060)
            QUES = 'ENTER FLOW POSITION:'
            LEN = 21
            CALL ENTR(QUES,DATA,LEN)
            FLOW_POSITION = DATA

C       ****CLOSE PRMFILE ****
            CLOSE(PUNIT)




60          IF (FLOW_POSITION .NE. 9999) THEN
               POSITION = FLOW_POSITION
               CALL RNF(POSITION,F)
               IF (F .EQ. 0) THEN
               CALL CHANGE(POSITION,ARRAY)
               ENDIF
               F = 0
500            QUES = 'ENTER POSITION: '
               LEN = 16
               CALL ENTR(QUES,DATA,LEN)
               FLOW_POSITION = DATA
               GO TO 60
            ENDIF


         ENDIF

      ENDIF



C>>>>> IF RETURNED ANSWER BESIDE "OLD" OR "NEW",REPEAT THE QUERY

      IF ((ANSSAVE .NE. 'OLD') .AND. (ANSSAVE .NE. 'NEW') .AND.
     *    (ANSSAVE .NE. 'old') .AND. (ANSSAVE .NE. 'new')) THEN
         GO TO 10
      ENDIF



3000  IF (OFLAG .EQ. 1) THEN
         WRITE(1,2120)
         QUES = ' '
          GO TO 17
      ENDIF



3500  END
```

```
      SUBROUTINE CHANGE(POSITION,ARRAY)
C     ********************************************************************
C     *
C     *                        CHANGE PROCEDURE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter
C     *  Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Purpose:
C     *
C     *            The purpose of this procedure is to change various
C     *            aspects of flow information according to position.
C     *             Data than can be adjusted is the tributary flow
C     *             percent,tributary stat,diversion flow percent,and
C     *             tributary flow.
C     *
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  PROCEDURES CALLED:
C     *
C     *                     CHECK ----- Checks if given position within
C     *                                 range of the stream.
C     *
C     *                     ENTR ------ Prints query and reads real
C     *                                 number data from terminal.
C     *
C     *                     CLEAR ----- Clears the screen.
C     *
C     ********************************************************************
C     ********************************************************************

C     ****MIDAS DECLARATIONS****

$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN
$INSERT SYSCOM>A$KEYS.INS.FTN


C     *****DECLARATIONS*****
```

```
      INTEGER                                  STATUS*2,
     *                                         I*2,
     *                                         MODE*2,
     *                                         LENGTH*2,
     *                                         FLAG*2,
     *                                         FLG*2,
     *                                         FL*2,
     *                                         ARRAY(14)*2,
     *                                         LEN*2,
     *                                         FUNIT*2

      REAL                                     BGREF,
     *                                         BGMN,
     *                                         ENREF,
     *                                         ENMN,
     *                                         BGN_MILE,
     *                                         END_MILE,
     *                                         DIV_PER_MILE,
     *                                         SUM_TRB,
     *                                         POS_SUM_TRB,
     *                                         FLOW_POSITION,
     *                                         POSITION,
     *                                         TRB_FLOW_PCT,
     *                                         DATA,
     *                                         TRB_FLOW,
     *                                         DVR_FLOW_PCT,
     *                                         TRB_PCT

      CHARACTER                                TRB_FLOW_STAT*1,
     *                                         FLOW_INFO*40,
     *                                         KEY*8,
     *                                         ANSWER*1,
     *                                         PATHNAME*32,
     *                                         TRB_STAT*1,
     *                                         QUES*40,
     *                                         CHECK_FLAG*3,
     *                                         CORR_DVR*3

      EQUIVALENCE (FLOW_INFO(1:8),KEY)




      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/THIRD/SUM_TRB,POS_SUM_TRB
      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH
```

```
C     **** QUERY AND OUTPUT FORMAT ****


2010  FORMAT(//,T19,'***PRESENT VALUES FOR MILE:',F8.2,'***')

2020  FORMAT(/,T44,'TRIBUTARY',T65,'NEW')

2030  FORMAT(  T26,'TRIBUTARY',T43,'OR EXISTING',T62,'DIVERSION')

2040  FORMAT(  T11,'INDICES',T26,'PER CENT',T44,'DIVERSION',T62,
     *        'PER CENT')

2050  FORMAT(  T44,'(FT**3/S)')

2060  FORMAT(  T11,60('-'))

2070  FORMAT(/,T14,A1,T28,F5.2,T43,F11.2,T64,F5.2)

2080  FORMAT(8(/,' '),T1,'TYPE "T" TO ENTER EXISTING TRIBUTARY',
     *        /,T1,'OR DIVERSION FLOWS AND "D" TO ENTER NEW ')

2090  FORMAT(//,T1,'ENTER ONE OF THE FOLLOWING: ')

2100  FORMAT(/,T1,'"A"-TO REFERENCE TRIB. FLOW TO OBSERVED',/,T1,
     *             '    FLOW AT STATION A,',
     *        /,T1,'"B"-TO REFERENCE TRIB. FLOW TO OBSERVED',/,T1,
     *             '    FLOW AT STATION B,',
     *        /,T1,'"C"-TO INDICATE AN EXISTING TRIBUTARY,RETURN FLOW,',
     *        /,T1,'    OR DIVERSION.'
     *        /,T1,'"U"-TO REMOVE A TRIBUTARY OR EXISTING ',
     *        /,T1,'    DIVERSION (UNDEFINED).')


2130  FORMAT(/,T1,'ILLEGAL ENTRY, DIVERSION PERCENTAGE',
     *        /,T1,'CANNOT BE GREATER THAT 100! PLEASE REENTER.')




C>>>  Check for range of position

      CHECK_FLAG = 'YES'
      CALL CHECK(POSITION,CHECK_FLAG)


C>>>> If position is within range, do the following.
```

```
      CALL CLEAR
      IF (CHECK_FLAG .EQ. 'YES') THEN

C>>>     Locate record according to position and print its contents
C>>>     to the screen.

         FLG = FL$RET
         WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
         CALL LOCK$(FUNIT,FLOW_INFO,KEY,ARRAY,FLG,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
         READ (FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
         READ (FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
         READ (FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
         READ (FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT
         WRITE(1,2010) POSITION
         WRITE(1,2020)
         WRITE(1,2030)
         WRITE(1,2040)
         WRITE(1,2050)
         WRITE(1,2060)
         WRITE(1,2070) TRB_FLOW_STAT,TRB_FLOW_PCT,TRB_FLOW,
     *                 DVR_FLOW_PCT


C>>>     Enter option of entering existing tributaries or diversions,
C>>>     or new diversions.

         WRITE(1,2080)
         CALL TNOUA('DIVERSION FLOWS: ',INTS(17))
         READ '(A)',ANSWER



C>>>     Process for tributaries or existing diversions


         IF ((ANSWER .EQ. 'T') .OR. (ANSWER .EQ. 't')) THEN

C>>>         Choose option to reference trib. flow to ovserved flow at
C>>>         station A, to reference trib. flow to observed flow at
C>>>         station B, to indicate an existing tributary or diversion,
C>>>         or remove a tributary or existing diversion (undefined).

            WRITE(1,2090)
            WRITE(1,2100)
            CALL TNOUA('ENTER CHOICE: ',INTS(14))
            READ '(A)',TRB_STAT



C>>>         Process for referencing trib. flow at either stat. A or B.
```

```
          IF ((TRB_STAT .EQ. 'A') .OR. (TRB_STAT .EQ. 'a')  .OR.
     *        (TRB_STAT .EQ. 'B') .OR. (TRB_STAT .EQ. 'b')) THEN


C>>>            Enter the tributary flow percent

                QUES = 'ENTER PERCENTAGE:'
                LEN = 18
                CALL ENTR(QUES,DATA,LEN)
                TRB_PCT = DATA



C>>>            Adjust sum of the tributaries and the positive sum of
C>>>            tributaries (if trib. flow > 0) by removing the old
C>>>            tributary flow.  Then recalculate tributary flow using
C>>>            trib stat and new tributary flow percent.  Afterwards,
C>>>            add new tributary flow back into sum of the tributaries
C>>>            and positive sum of the tributaries (if trib. flow > 0).

                IF (TRB_FLOW .GT. 0) THEN
                   POS_SUM_TRB = POS_SUM_TRB - TRB_FLOW
                ENDIF
                SUM_TRB = SUM_TRB - TRB_FLOW

                TRB_FLOW_PCT = TRB_PCT/100
                TRB_FLOW_STAT = TRB_STAT
                IF ((TRB_FLOW_STAT .EQ. 'A') .OR.
     *              (TRB_FLOW_STAT .EQ. 'a')) THEN
                   TRB_FLOW = BGREF * TRB_FLOW_PCT
                ELSE
                   TRB_FLOW = ENREF * TRB_FLOW_PCT
                ENDIF
                IF (TRB_FLOW .GT. 0) THEN
                   POS_SUM_TRB = POS_SUM_TRB + TRB_FLOW
                ENDIF
                SUM_TRB = SUM_TRB + TRB_FLOW



C>>>            Write record with adjusted data back into the work file.

                FLOW_POSITION = POSITION
                WRITE(FLOW_INFO(9:12),'(F4.2)') TRB_FLOW_PCT
                WRITE(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
                WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW
                WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
                FL = FL$USE
                CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                      INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

          ELSE
```

```
C>>>           Process to indicate existing tributary,return flow,
C>>>           or diversion.

               IF ((TRB_STAT .EQ. 'C') .OR. (TRB_STAT .EQ. 'c')) THEN


C>>>              Remove old tributary flow from sum of the tributaries
C>>>              and positive sum of tributaries (if trib.flow > 0).
C>>>              Enter new tributary flow and add new flow back into
C>>>              sum of tribs and the positive sum of the tribs
C>>>              (if trib. flow > 0).  SEt trib stat to C and the trib
C>>>              flow to 0 and write record back into the work file
C>>>              with the adjusted data.

                  IF (TRB_FLOW .GT. 0) THEN
                     POS_SUM_TRB = POS_SUM_TRB - TRB_FLOW
                  ENDIF
                  SUM_TRB = SUM_TRB - TRB_FLOW
                  FLOW_POSITION = POSITION
                  QUES = 'ENTER TRIBUTARY CONSTANT VALUE:'
                  LEN = 32
                  CALL ENTR(QUES,DATA,LEN)
                  TRB_FLOW = DATA
                  IF (TRB_FLOW .GT. 0) THEN
                     POS_SUM_TRB = POS_SUM_TRB + TRB_FLOW
                  ENDIF
                  SUM_TRB = SUM_TRB + TRB_FLOW
                  TRB_FLOW_STAT = 'C'
                  TRB_FLOW_PCT = 0
                  FLOW_POSITION = POSITION
                  WRITE(FLOW_INFO(16:16),'(A1)')TRB_FLOW_STAT
                  WRITE(FLOW_INFO(9:12),'(F4.2)') TRB_FLOW_PCT
                  WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW

                  FL = FL$USE
                  CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                 INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

               ELSE


C>>>        Process to remove a trib. or div.,(undefined).

                  IF ((TRB_STAT .EQ. 'U') .OR. (TRB_STAT .EQ. 'u')) THEN


C>>>                 Make the trib stat "U" (undefined) and the rest of
C>>>                  flow information to 0. Remove the old tributary
C>>>                 from  sum of  tribs and the positive sum of the
C>>>                 tribs (if trib. flow > 0).  Then write the record
C>>>                 back into the work file with adjusted data.

                     TRB_FLOW_PCT = 0
                     TRB_FLOW_STAT = 'U'
                     FLOW_POSITION = POSITION
                     IF (TRB_FLOW .GT. 0) THEN
                        POS_SUM_TRB = POS_SUM_TRB - TRB_FLOW
                     ENDIF
                     SUM_TRB = SUM_TRB - TRB_FLOW
                     TRB_FLOW = 0
                     WRITE(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
                     WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW
                     WRITE(FLOW_INFO(9:12),'(F4.2)') TRB_FLOW_PCT
                     FL = FL$USE
                     CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                        INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

                  ELSE
                     CALL TONL
                     PRINT *,'INVALID ANSWER, NO CHANGES WILL BE MADE.'
                  ENDIF
               ENDIF
            ENDIF

         ELSE

C>>>        Entering new diversions


            IF ((ANSWER .EQ. 'D') .OR. (ANSWER .EQ. 'd')) THEN

C>>>           Enter the diversion flow percent.

               CORR_DVR = 'NO'
10             IF (CORR_DVR .EQ. 'NO') THEN
                  QUES = 'ENTER DIVERSION PERCENTAGE:'
                  LEN = 27
                  CALL ENTR(QUES,DATA,LEN)
                  DVR_PCT = DATA

C>>>              Make sure percent is less than 100

                  IF (DVR_PCT .GT. 100) THEN
                     WRITE(1,2130)
                  ELSE

C>>>                 Put percentage in fractional form and place record
C>>>                 back into work file.

                     DVR_FLOW_PCT = DVR_PCT/100
                     FLOW_POSITION = POSITION
                     WRITE(FLOW_INFO(27:30),'(F4.2)')DVR_FLOW_PCT
                     FL = FL$USE
                     CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *                        INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

                     CORR_DVR = 'YES'
                  ENDIF
                  GO TO 10
               ENDIF
            ELSE
               CALL TONL
               PRINT *,'INVALID ANSWER, NO CHANGES WILL BE MADE.'
            ENDIF
         ENDIF
      ENDIF


C>>>  Make sure the record is unlocked and clear the MIDAS array.

      FL = FL$ULK + FL$USE

      CALL UPDAT$(FUNIT,FLOW_INFO,POSITION,ARRAY,FL,
     *            INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))


      DO 5 I = 1,14
         ARRAY(I) = 0
5     CONTINUE


      END
      SUBROUTINE BGENCNG
C     ********************************************************************
C     *
C     *               CHANGING BEGINNING OR ENDING REFERENCES
C     *                             PROCEDURE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *    AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *    DATE:    12/9/86
C     *
C     *    Converted from the original PLI/I by C. R. Baxter
C     *    Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *   Purpose:
C     *
C     *              The purpose of this procedure is to change either
C     *              maximum or minimum flow(s) at station A or B.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *        ALGORITHM:
C     *
C     *           I.  INDICATE WHETHER REFERENCING STATION A FLOW OR
C     *               STATION B FLOW.
C     *
C     *          II.  IF OPTION CHOSEN EITHER  A OR B  THEN
C     *               DO THE FOLLOWING:
C     *               A. SET STAT TO INDICATED STATION (A OR B).
C     *
C     *               B. ENTER  X  TO CHANGE MAIXIMUM FLOW OR  N
C     *                  TO CHANGE MINIMUM FLOW.
C     *
C     *               C. IF CHANGING MINIMUM FLOW THEN CHANGE
C     *                  STATION A MIN. FLOW IF STAT IS A OR CHANGE
C     *                  STATION B MINIMUM FLOW IF STAT IS B.
C     *
C     *               D. IF CHANGING MAXIMUM FLOW,DO THE FOLLOWING:
C     *
C     *                  1. CHANGE STATION  A  MAXIMUM FLOW IF THE
C     *                     STAT IS A OR CHANGE STATION B MAXIMUM
C     *                     FLOW IF THE STAT IS B.
C     *
C     *                  2. SET THE CHANGE VALUE TO THE INDICATED
C     *                     STATION'S NEW MAXIMUM FLOW.
C     *
C     *                  3. CHANGE THE TRIBUTARY FLOW VALUES IF STAT
```

```
C     *                     MATCHES THE FLOW INFORMATION STAT.
C     *
C     *               E. IF OPTION CHOSEN OTHER 'X' OR 'N', THEN
C     *                  PRINT THAT WRONG ANSWER WAS CHOSEN
C     *                  AND NO CHANGES ARE MADE.
C     *
C     *         III.  IF OPTION WAS CHOSEN OTHER THAN 'A' OR 'B' THEN
C     *               INDICATE THAT WRONG OPTION WAS CHOSEN AND
C     *               NO CHANGES MADE.
C     *
C     *          IV.  STOP PROCESSING.
C     *
C     ******************************************************************
C     ******************************************************************
C     *
C     *         PROCEDURES CALLED:
C     *
C     *            CHNGTMPFL ---------- CHANGES THE TRIBUTARY FLOW
C     *                                 VALUES IF STAT MATCHES THE
C     *                                 FLOW INFORMATION STAT.
C     *
C     *            ENTR --------------- PRINTS QUERY AND READS REAL
C     *                                 NUMBER DATA.
C     *
C     *            HEAD --------------- PRINTS PROGRAM HEAD TO SCREEN.
C     *
C     *            CLEAR -------------- CLEARS THE SCREEN.
C     *
C     ******************************************************************
C     ******************************************************************
C     *
C     *         MAJOR VARIABLES:
C     *
C     *            BGREF -------------- MAXIMUM FLOW AT STATION  A.
C     *            BGMN  -------------- MINIMUM FLOW AT STATION  A.
C     *            ENREF -------------- MAXIMUM FLOW AT STATION  B.
C     *            ENMN  -------------- MINIMUM FLOW AT STATION  B.
C     *            BGN_MILE ----------- STATION  A  LOCATION IN MILES.
C     *            END¯MILE ----------- STATION  B  LOCATION IN MILES.
C     *            DIV¯PER_MILE ------- DIV/MILE BETWEEN STAT. A & B.
C     *            SUM¯TRB¯------------ SUM OF THE TRIBUTARY FLOWS.
C     *            POS¯SUM_TRB -------- POSITIVE SUM OF TRIBUTARY FLOWS.
C     *            DATA¯ -------------- REAL NUMBER DATA ENTERED
C     *                                 AT TERMINAL.
C     *            ANSWER ------------- OPTION BUFFER
C     *            QUES  -------------- QUESTION TO PRINTED AT TERMINAL
C     *            STAT  -------------- 'A' ---TO CHANGE STATION A
C     *                                  MAX OR MIN FLOW.
C     *                                 'B' ---TO CHANGE STATION B
C     *                                  MAX OR MIN FLOW.
C     *
C     ******************************************************************
C     ******************************************************************
```

```
$INSERT SYSCOM>A$KEYS.INS.FTN

C     ********************
C     *** DECLARATIONS ***
C     ********************


      INTEGER                                   LEN*2

      REAL                                      BGREF,
     *                                          BGMN,
     *                                          ENREF,
     *                                          ENMN,
     *                                          BGN_MILE,
     *                                          END_MILE,
     *                                          DIV_PER_MILE,
     *                                          SUM_TRB,
     *                                          POS_SUM_TRB,
     *                                          DATA

      CHARACTER                                 ANSWER*1,
     *                                          STAT*1,
     *                                          QUES*40



      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/THIRD/SUM_TRB,POS_SUM_TRB








C     **********************
C     **** QUERY FORMAT ****
C     **********************


2010  FORMAT(/,T1,'ENTER "A" TO CHANGE STATION A REFERENCE FLOW ',
     *       /,T1,'OR ENTER "B" TO CHANGE STATION B REFERENCE FLOW  ')
2020  FORMAT(/,T1,'ENTER "X" TO CHANGE REFERENCE(MAXIMUM) FLOW OR')
2040  FORMAT(/,T1,'THE WRONG ANSWER WAS ENTERED. ',/,T1,
     *       'NO CHANGES WILL BE MADE.')
```

```
C>>>  Enter option to change a station's (A or B) reference flow.

10    CALL CLEAR
      CALL HEAD
      ANSWER = ' '
      WRITE(1,2010)
      CALL TNOUA('ELSE (CR): ',INTS(11))
      READ '(A)',ANSWER



C>>>  Begin processing the flows.

      IF ((ANSWER .EQ. 'A') .OR. (ANSWER .EQ. 'B')  .OR.
     *    (ANSWER .EQ. 'a') .or. (ANSWER .EQ. 'b')) THEN

C>>>     Set stat to chosen station (A or B)

         STAT = ANSWER



C>>>     Enter "X" to change maximum flow or "N" to change
C>>>     minimum flow.

         WRITE(1,2020)
         CALL TNOUA('"N" TO ENTER MINIMUM FLOW: ',INTS(27))
         READ '(A)', ANSWER

C>>>     If the minimum flow is chosen, enter the minimum flow
C>>>     and set it value to the chosen station's minimum flow.

         IF ((ANSWER .EQ. 'N') .OR. (ANSWER .EQ. 'n')) THEN
            QUES = 'ENTER MINIMUM FLOW:'
            LEN = 20
            CALL ENTR(QUES,DATA,LEN)
            IF ((STAT .EQ. 'A') .OR. (STAT .EQ. 'a')) THEN
               BGMN = DATA
            ELSE
               ENMN = DATA
            ENDIF
         ELSE


C>>>        If maximum flow was chosen, enter the maximum flow and
C>>>        set its value to the chosen station's maximum flow.
C>>>        Reinitialize sum of the tributaries and the positive sum
C>>>        of tributaries to zero (0) and call a routine to adjust
```

```
C>>>        the tributaries flows with respect to its new maximum flow.

            IF ((ANSWER .EQ. 'X') .OR. (ANSWER .EQ. 'x')) THEN
               QUES = 'ENTER MAXIMUM FLOW:'
               LEN = 20
               CALL ENTR(QUES,DATA,LEN)
               IF ((STAT .EQ. 'A') .OR. (STAT .EQ. 'a')) THEN
                  BGREF = DATA
               ELSE
                  ENREF = DATA
               ENDIF

               CHNGVAL = DATA
               SUM_TRB = 0
               POS_SUM_TRB = 0
               CALL CHNGTMFL(STAT,CHNGVAL)
            ELSE

C>>>               Invalid option, no changes made.

                  WRITE(1,2040)
            ENDIF
         ENDIF
      GO TO 10
      ELSE


C>>>      If option is carriage return besides A or B,then exit routine.

         IF (ANSWER .EQ. ' ') THEN
            GO TO 3500
         ELSE

C>>>         Invalid option, no changes made.

            WRITE(1,2040)
            GO TO 10
         ENDIF
      ENDIF

3500  END
```

```
      SUBROUTINE CHNGTMFL(STAT,CHNGVAL)
C     ********************************************************************
C     *
C     *                      CHANGE TRIBUTARY FLOW
C     *                            PROCEDURE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR: JOHN E. TERRY AND BARRY LYLE
C     *  DATE:   12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Purpose:
C     *
C     *              The purpose of this procedure is to change the
C     *              tributary flows according to either the station A
C     *              or station B maximum flow between the beginning
C     *              mile and the end mile.
C     *
C     ********************************************************************
C     *
C     *      VARIABLES:
C     *
C     *         BGN_MILE ---------- STATION A LOCATION IN MILES
C     *         END_MILE ---------- STATION B LOCATION IN MILES.
C     *         DIV_PER_MILE ------ DIV/MILE BETWEEN STATIONS A AND B.
C     *         SUM_TRB  ---------- SUM OF THE TRIBUTARY FLOWS.
C     *         POS_SUM_TRB ------- POSITIVE SUM OF TRIBUTARY FLOWS.
C     *         CHNGVAL  ---------- PARAMETER VALUE OF EITHER BGREF
C     *                             (STATION A MAXIMUM FLOW) OR
C     *                             ENREF(STATION B MAXIMUM FLOW).
C     *         END_STM  ---------- END STREAM POSITION.
C     *         STAT     ---------- INDICATOR FOR EITHER STATION A OR B.
C     *         I        ---------- INDEX
C     *
C     *         FLOW INFORMATION:
C     *
C     *            FLOW_POSITION -- POSITION WHERE FLOW INFO WAS
C     *                             ANALYZED.
C     *            TRB_FLOW_PCT --- PERCENTAGE OF TRIBUTARY FLOW.
C     *            TRB_FLOW_STAT -- 'A' - TO REFERENCE TRIBUTARY FLOW
C     *                              TO OBSERVED FLOW AT STATION A.
C     *                             'B' - TO REFERENCE TRIBUTARY FLOW
C     *                              TO OBSERVED FLOW AT STATION B.
C     *                             'C' - TO INDICATE AN EXISTING TRIBUTA
C     *                                   IRRIGATION RETURN FLOW,OR
```

```
C      *                                         DIVERSION.
C      *                                  'U' - TO REMOVE A TRIBUTARY OR
C      *                                   EXISTING DIVERSION (UNDEFINED).
C      *                TRB_FLOW ------- TRIBUTARY FLOW
C      *                DVR_FLOW_PCT --- PERCENTAGE OF DIVERSION FLOW.
C      *
C      ******************************************************************
C      ******************************************************************




C      ***MIDAS DECLARATIONS***

$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN



C      ********************
C      *** DECLARATIONS ***
C      ********************

       INTEGER                                  MODE*2,
      *                                         STATUS*2,
      *                                         FLAG*2,
      *                                         FL*2,
      *                                         FLG*2,
      *                                         I*2,
      *                                         LENGTH*2,
      *                                         FUNIT*2,
      *                                         ARRAY(14)*2

       REAL                                     BGN_MILE,
      *                                         END_MILE,
      *                                         DIV_PER_MILE,
      *                                         SUM_TRB,
      *                                         POS_SUM_TRB,
      *                                         CHNGVAL,
      *                                         FLOW_POSITION,
      *                                         TRB_FLOW_PCT,
      *                                         TRB_FLOW,
      *                                         DUMMY,
      *                                         DVR_FLOW_PCT



       CHARACTER                                STAT*1,
      *                                         TRB_FLOW_STAT*1,
      *                                         FLOW_INFO*40,
      *                                         KEY*8,
      *                                         PATHNAME*32
      EQUIVALENCE(FLOW_INFO(1:8),KEY)

      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE

      COMMON/THIRD/SUM_TRB,POS_SUM_TRB

      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH



C     ***********************
C     **** OUTPUT FORMAT ****
C     ***********************


2010  FORMAT(F7.2,F4.2,A1,F10.2,F4.2)

C     ***********************************
C     *******BEGIN CHNGTMPFL*************
C     ***********************************




C>>>  Initialize position to the end mile and a dummy variable to
C>>>  the reciprocal of division per mile. The reciprocal will be
C>>>  used as an increment for position.

      POSITION = END_MILE
      DUMMY = 1/DIV_PER_MILE


C>>>  Set MIDAS flag and write the position to the flow info buffer.

      FLAG = FL$RET
      WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION

C>>>
C>>>  While position is less than or equal to beginning mile, process.

10    IF (POSITION .LE. BGN_MILE) THEN


C>>>     Locate the record and read the flow information from the
C>>>     buffer to their variables.
```

```
         WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
         CALL LOCK$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
         READ (FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
         READ (FLOW_INFO(17:26),'(F10.0)')TRB_FLOW
         READ (FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT


C>>>     If the tributary stat is equal to the parameter stat (A or B),
C>>>     then recalculate the tributary flow and write back into the
C>>>     temporary work file.

         IF (TRB_FLOW_STAT .EQ. STAT) THEN
            TRB_FLOW = TRB_FLOW_PCT * CHNGVAL
            FL = FL$USE
            WRITE(FLOW_INFO(17:26),'(F10.2)') TRB_FLOW
            CALL UPDAT$(FUNIT,FLOW_INFO,KEY,ARRAY,FL,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
         ENDIF


C>>>     If tributary flow is positive then add the flow to the sum
C>>>     of positive tributary flows.

         IF (TRB_FLOW .GT. 0) THEN
            POS_SUM_TRB = POS_SUM_TRB + TRB_FLOW
         ENDIF

C>>>     Add the tributary flow to the sum of tributary flows.

         SUM_TRB = SUM_TRB + TRB_FLOW


C>>>     Release the record that was updated.

         FL = FL$ULK +FL$USE
         CALL UPDAT$(FUNIT,FLOW_INFO,KEY ,ARRAY,FL,
     *               INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))



C>>>     Clear the MIDAS array.

         DO 5 I = 1,14
            ARRAY(I) = 0
5        CONTINUE
         FLAG = FL$RET


C>>>     Increment to the next position and repeat.
```

```
         POSITION = POSITION + DUMMY
         GO TO 10

      ENDIF


C>>>  Making sure that last record processed was unlocked.

      FL =FL$ULK +FL$USE
      CALL UPDAT$(FUNIT,FLOW_INFO,POSITION,ARRAY,FL,
     *            INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
      END
      SUBROUTINE WTAVL
C     *******************************************************************
C     *
C     *                         WATER AVAILABLILITY
C     *                              PROCEDURE
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHORS:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:     12/9/86
C     *
C     *  Converted from original PL/I by C. R. Baxter.
C     *  Date: 4/87
C     *
C     *  PURPOSE:
C     *
C     *            The purpose of this procedure is to determine the
C     *            amount of water available with existing
C     *            tributaries,diversions (new and old) or without
C     *            diversions.
C     *******************************************************************
C     *
C     *     ALGORITHM:
C     *
C     *        I. ENTER THE STARTING POSITION AT WHERE THE PROCESSING
C     *           IS TO BEGIN.
C     *
C     *       II. CHECK IF START POSITION IS IN RANGE OF THE STREAM.
C     *           IF POSITION IN RANGE THEN ENTER ENDING POSITION.
C     *
C     *      III. IF THE ENDING POSITION IS IN RANGE, DECIDE IF NEW
C     *           DIVERSIONS NEEDED IN DETERMINING WATER AVAILABILITY.
C     *
C     *       IV. IF NEW DIVERSIONS ARE TO BE USED, DETERMINE WATER
C     *           AVAILABILITY WITH EXISTING TRIBUTARIES & DIVERSIONS &
C     *           NEW DIVERSIONS, ELSE USE ONLY EXISTING TRIBUTARIES &
C     *           DIVERSIONS.
C     *           IF WRONG OPTION CHOSEN, NO PROCESSING IS PERFORMED.
C     *
C     *        V. STOP PROCESSING.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *      INDICATION CODES:
C     *
C     *         'D' ------- DETERMINE WATER AVAILABILITY WITH NEW
C     *                     DIVERSIONS.
C     *         'N' ------- DETERMINE WATER AVAILABILITY WITHOUT NEW
C     *                     DIVERSIONS.
C     *
```

```
C     *          **** ANY OTHER OPTION IS INVALID AND NO CHANGES MADE.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *     PROCEDURES CALLED:
C     *
C     *         WDVR ------------ DETERMINES WATER AVAILABILITY WITH
C     *                           EXISTING TRIBUTARIES,DIVERSIONS,&
C     *                           NEW DIVERSIONS.
C     *         WODVR ----------- DETERMINES WATER AVAILABILITY WITH
C     *                           EXISTING TRIBUTARIES & DIVERSIONS.
C     *                           (NO NEW DIVERSIONS).
C     *
C     *         CHECK ----------- CHECKS  A POSITION TO SEE IF
C     *                           WITHIN RANGE OF THE STREM, THAT IS,
C     *                           THE POSITION IS BETWEEN
C     *                           STATION A & STATION B LOCATIONS.
C     *
C     *         ENTR ------------ PRINTS A QUERY AND READS REAL
C     *                           NUMBER DATA FROM THE TERMINAL.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *     VARIABLES:
C     *
C     *        BGREF ------------ MAXIMUM FLOW AT STATION A.
C     *        BGMN  ------------ MINIMUM FLOW AT STATION A.
C     *        ENREF ------------ MAXIMUM FLOW AT STATION B.
C     *        ENMN  ------------ MINIMUM FLOW AT STATION B.
C     *        BGN_MILE --------- STATION A LOCATION IN MILES.
C     *        END_MILE --------- STATION B LOCATION IN MILES.
C     *        DIV_PER_MILE------ DIVISIONS/MILE BETWEEN STATION A & B.
C     *        SUM_TRB ---------- SUM OF THE TRIBUTARIES(+ OR -).
C     *        POS_SUM_TRB ------ POSITIVE SUM OF THE TRIBUTARIES.
C     *        STRT_POS --------- START POSITION WHCIH PROCESS BEGINS.
C     *        END_POS ---------- END POSITION WHICH PROCESS ENDS.
C     *        POSITION --------- POSITION OF STREAM BEING ANANLYZED.
C     *        DATA  ------------ REAL NUMBER DATA ENTEREDNATRTERMINAL.
C     *        CHECK_FLG -------- FLAG TO INDICATE WHETHER A POSITION
C     *                           IS IN RANGE.
C     *        ANSWER ----------- OPTION BUFFER.
C     *        QUES  ------------ QUERY PRINTED AT TERMINAL.
C     *
C     *******************************************************************
C     *******************************************************************



C     ********************
C     *** DECLARATIONS ***
C     ********************
```

```
      INTEGER                                  WUNIT*4,
     *                                         FLG*2,
     *                                         LEN*2

      REAL                                     BGREF,
     *                                         BGMN,
     *                                         ENREF,
     *                                         ENMN,
     *                                         BGN_MILE,
     *                                         END_MILE,
     *                                         DIV_PER_MILE,
     *                                         SUM_TRB,
     *                                         POS_SUM_TRB,
     *                                         STRT_POS,
     *                                         END_POS,
     *                                         FLOW_POSITION,
     *                                         POSITION,
     *                                         DATA


      CHARACTER                                CHECK_FLG*3,
     *                                         FILENAME*32,
     *                                         QUES*40,
     *                                         ANSWER*1


      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/THIRD/SUM_TRB,POS_SUM_TRB






C          **********************
C          **** QUERY FORMAT ****
C          **********************

2010  FORMAT(/,T1,'ENTER "D" IF YOU WANT DIVERSIONS.')

2015  FORMAT(15(/,' '))
2020  FORMAT(/,T1,'YOU ENTERED THE WRONG ANSWER!!',/,T1,
     *        'NO PROCESSING WILL BE DONE!')
2030  FORMAT(/,T1,'STARTING POSITION SHOULD BE',
     *       /,T1,'UPSTREAM (i.e. GREATER) FROM THE',
     *       /,T1,'END POSITION.  PLEASE RE-ENTER.')
```

```
C     ***** BEGIN WTAVL PROCEDURE *****


C>>>  Enter the option to continue or exit the routine.


C>>>  Enter starting position.

      WUNIT = 85
      CALL CLEAR
      CALL HEAD
10    QUES = 'ENTER STARTING POSITION(UPSTREAM):'
      LEN = 35
      CALL ENTR(QUES,DATA,LEN)
      STRT_POS = DATA



C>>>  Check if starting position is in range of the stream.

      CHECK_FLG = 'YES'
      CALL CHECK(STRT_POS,CHECK_FLG)

C>>>  If starting position is in range, enter the end position.

      IF (CHECK_FLG .EQ. 'YES') THEN
15       QUES = 'ENTER ENDING POSITION(DOWNSTREAM):'
         LEN = 35
         CALL ENTR(QUES,DATA,LEN)
         END_POS = DATA


C>>>     Check if end position is in range of the stream.

         CALL CHECK(END_POS,CHECK_FLG)


C>>>     If end position is in range of the stream, start processing.

         IF (CHECK_FLG .EQ. 'YES') THEN


C>>>        If end postion greater than start position, reenter data.

            IF (END_POS .GT. STRT_POS) THEN
```

```
               WRITE(1,2030)
               CALL TONL
               GO TO 10
            ENDIF



C>>>        Enter option of table with or without diversions.

            WRITE(1,2010)
            CALL TNOUA('IF NOT,ENTER "N": ',INTS(18))
            READ '(A)',ANSWER



C>>>        Enter the name of a output file, else carriage return.

            QUES = 'ENTER OUTPUT FILE,ELSE (CR): '
            FLG = 3
            LEN = 29
            CALL FILECK(QUES,FILENAME,FLG,LEN)


C>>>        If a table with diversions is chosen call procedure WDVR.

            IF ((ANSWER .EQ. 'D') .OR. (ANSWER .EQ. 'd')) THEN
               CALL WDVR(STRT_POS,END_POS,FILENAME,WUNIT)
            ELSE


C>>>           If tabel without diversions chosen,call procedure WODVR.

               IF ((ANSWER .EQ. 'N') .OR. (ANSWER .EQ. 'n')) THEN
                  CALL WODVR(STRT_POS,END_POS,FILENAME,WUNIT)
               ELSE

C>>>              Invalid option, no processing is done.

                  WRITE(1,2020)
               ENDIF
            ENDIF
         ELSE

C>>>        End position is out of range of the stream, reenter.

            GO TO 15
         ENDIF
      ELSE

C>>>     Start position is out of range of the stream, reenter.

         GO TO 10
      ENDIF



3500  END
```

```
      SUBROUTINE WODVR(STRT_POS,END_POS,FILE_NAME,WUNIT)

C     ********************************************************************
C     *
C     *                          WODVR PROCEDURE
C     *
C     *               (Stream Availability Without Diversions)
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter
C     *  Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Purpose:
C     *
C     *           The purpose of this procedure is to produce a linear
C     *           table beginning and ending with user supplied
C     *           positions of available stream flows without
C     *           diversion adjustments.
C     *
C     *           The table is not printed if the combination of
C     *           station A's maximum flow,excess, and sum of the
C     *           tributaries is less than station B's minimum flow.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Procedures called:
C     *
C     *                        CLEAR -- To clear the screen.
C     *
C     *                        HEAD  -- Prints the program heading.
C     *
C     ********************************************************************
C     ********************************************************************


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN


C     *** DECLARATIONS ***

      INTEGER                                  WUNIT*4,
     *                                         FUNIT*2,
*                                         PUNIT*4,
*                                         FLAG*2,
*                                         FL*2,
*                                         ARRAY(14)*2,
*                                         MODE*2,
*                                         STATUS*2,
*                                         VAR*4,
*                                         LENGTH*2


 REAL                                     BGREF,
*                                         BGMN,
*                                         ENREF,
*                                         ENMN,
*                                         BGN_MILE,
*                                         END_MILE,
*                                         DIV_PER_MILE,
*                                         SUM_TRB,
*                                         POS_SUM_TRB,
*                                         STRT_POS,
*                                         END_POS,
*                                         POSITION,
*                                         TRB_FLOW_PCT,
*                                         TRB_FLOW,
*                                         DVR_FLOW_PCT,
*                                         POS,
*                                         AVAL_FLOW,
*                                         CUM_AVL,
*                                         EXCESS,
*                                         RESID,
*                                         RESID_TOT,
*                                         TOT_FLOW,
*                                         MIN_FLOW,
*                                         MN_DIFF,
*                                         POS_TRB_FLOW

 CHARACTER                                TRB_FLOW_STAT*1,
*                                         FILE_NAME*32,
*                                         PATHNAME*32,
*                                         ANSWER*4,
*                                         KEY*8,
*                                         FLOW_INFO*40

 EQUIVALENCE (FLOW_INFO(1:8),KEY)


 COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
 COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
 COMMON/THIRD/SUM_TRB,POS_SUM_TRB
 COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH
```

```
C      *********************************
C      **** QUERY AND OUTPUT FORMAT ****
C      *********************************


2010  FORMAT(//, T23,'FLOW AVAILABILITY ESTIMATED ON BASIS ',
     *       T62, 'OF OBSERVED CONDITIONS-NO NEW DIVERSIONS ANALYZED.')

2020  FORMAT(  T23,89('-'))

2030  FORMAT(  T58,'TRIBUTARY',T74,'ESTIMATED',T94,'ESTIMATED',T113,
     *        'ESTIMATED')

2040  FORMAT(  T39,'TRIBUTARY',T57,'OR EXISTING',T76,'TOTAL',T92,
     *         'MINIMUM FLOW',T113,'AVAILABLE')

2050  FORMAT(  T10,'LOCATION',T24,'INDICES',T39,'PER CENT',T58,
     *       'DIVERSION',T77,'FLOW',T93,'MAINTAINED',T115,'FLOW')

2060  FORMAT(  T12,'(MILE)',T58,'(FT**3/S)',T75,'(FT**3/S)',
     *       T94,'(FT**3/S)',T113,'(FT**3/S)')

2070  FORMAT(  T11,F7.2,T27,A1,T41,I5,T57,F11.2,T73,F11.2,
     *         T92,F11.2,T104,A1,T110,F11.2,T123,A1)

2090  FORMAT(/,T1,'UNDER THE GIVEN CONDITIONS, THE MINIMUM FLOW  ',
     *       /,T1,'(',F11.2,' FT**3/S)',
     *       /,T1,'AT B CANNOT BE MAINTAINED.')

2100  FORMAT(/,T1,'ENTER "LIST" IF YOU WISH TO SEE',
     *       /,T1,'MINIMUM DISCHARGE BALANCE DATA,')
2110  FORMAT(T1,'SPECIFIED MINUMUM FLOW AT STATION A = ',F11.2)

2120  FORMAT(/,T1,'SPECIFIED MINIMUM FLOW AT STATION B = ',F11.2)

2130  FORMAT(/,T1,'SUM OF ALL TRIBUTARY FLOWS = ',F11.2)

2140  FORMAT(/,T1,'LINEAR INFLOW OR OUTFLOW (COMPUTED FROM OBSERVED',
     *       /,T1,'DISCHARGE BALANCE) = ',F11.2)

2150  FORMAT(/,T2,F11.2,'  + ',F11.2,'  + ',F11.2,'  < ',F11.2)
```

```
C     ************************
C     **     BEGIN WODVR    **
C     ************************


C>>>  If a filename was given, open the file.


      IF (FILE_NAME .NE. ' ') THEN
         OPEN(WUNIT, FILE = FILE_NAME, FORM = 'FORMATTED',
     *        ACCESS = 'SEQUENTIAL')
      ENDIF

C>>>  Initilize the total flow to station A's max flow and the minimum
C>>>  flow to station A's minimum flow.

      TOT_FLOW = BGREF
      MIN_FLOW = BGMN


C>>>  Initialize the available flow to the difference between station
C>>>  A's max and minimum flows.

      AVAL_FLOW = BGREF - BGMN


C>>>  initialize the minimum difference as the difference between
C>>>  Station A and B's minimum flows.

      MN_DIFF = BGMN - ENMN


C>>>  Calculate the excess.

      EXCESS = ENREF - (BGREF + SUM_TRB)

      CALL CLEAR


C>>>  Set a dummy variable as the reciprocal of the division per mile.
C>>>  This will be used as a decrement for POSITION.

      DUMMY = 1/DIV_PER_MILE

C***********************************************************************
C**>  If the combination of station A's max flow, the excess, and the
C**>  sum of the tribs. is greater than or equal to station B's minimum
C**>  flow, do the following.
C***********************************************************************

      IF ((BGREF + EXCESS + SUM_TRB) .GE. ENMN) THEN
```

```
C>>>     Write the headings to the tables (terminal and output file(if gi
C>>>   file(if given).
         IF (FILE_NAME .NE. ' ') THEN
            WRITE(WUNIT,2010)
            WRITE(WUNIT,2020)
            WRITE(WUNIT,2030)
            WRITE(WUNIT,2040)
            WRITE(WUNIT,2050)
            WRITE(WUNIT,2060)
         ENDIF

         WRITE(1,2010)
         WRITE(1,2020)
         WRITE(1,2030)
         WRITE(1,2040)
         WRITE(1,2050)
         WRITE(1,2060)



C>>>     Calculate the cumulative available flow, the residual, and the
C>>>     total residual.

         CUM_AVL = BGMN + EXCESS - ENMN
         RESID = CUM_AVL/((BGN_MILE - END_MILE)* DIV_PER_MILE)
         RESID_TOT = EXCESS/((BGN_MILE - END_MILE) * DIV_PER_MILE)



C>>>     Set position to the beginning mile, find the first record
C>>>     according to the position and read the record.

         POSITION = BGN_MILE

         FLAG = FL$RET
         DO 3 I = 1,14
            ARRAY(I) = 0
 3       CONTINUE

         WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
         CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

         READ (FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
         READ (FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
         READ (FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
         READ (FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT

C>>>     Initialize stat to blank.


         STAT = ' '
```

```
C>>>     If the starting position is equal to the beginning mile,
C>>>     write the position and its relative information to the
C>>>     table (terminal and output file(if given).

         IF (STRT_POS .EQ. POSITION) THEN

            WRITE(1,2070) POSITION,TRB_FLOW_STAT,(TRB_FLOW_PCT*100),
     *            TRB_FLOW,TOT_FLOW,MIN_FLOW,STAT,AVAL_FLOW,STAT

            IF (FILE_NAME .NE. ' ') THEN
               WRITE(WUNIT,2070) POSITION,TRB_FLOW_STAT,
     *                           (TRB_FLOW_PCT*100),TRB_FLOW,TOT_FLOW,
     *                           MIN_FLOW,STAT,AVAL_FLOW,STAT
            ENDIF
         ENDIF




C>>>     Continue to process records while the current position is
C>>>     less than the end position.

10       IF (POSITION .GT. END_POS) THEN


C>>>        Decrement down to the next position,find and read the
C>>>        next record according to the position.

            POSITION = POSITION  - DUMMY

            DO 4 I = 1,14
               ARRAY(I) = 0
4           CONTINUE

            FLAG = FL$RET

            WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
            CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *            INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
            READ(FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
            READ(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
            READ(FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
            READ(FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT


C>>>        Re-initialize stat to blank.

            STAT = ' '
```

```
C>>>        Add to the current available flow the residual and the
C>>>        Current tributary flow.  Add the tributary flow and
C>>>        the total residual to the total flow.  Subtract the
C>>>        recalculated available flow from the recalculated total
C>>>        flow and set to the minimum difference.

            AVAL_FLOW = AVAL_FLOW + TRB_FLOW + RESID
            TOT_FLOW = TOT_FLOW + TRB_FLOW + RESID_TOT
            MIN_FLOW = TOT_FLOW - AVAL_FLOW

C>>>        If the available flow is less than 0, set the stat.

            IF (AVAL_FLOW .LT. 0) THEN
               STAT = '*'
            ENDIF




C>>>        If the current position is less than or equal to the
C>>>        starting position, write the position and its relative
C>>>        information to the table.

            IF (POSITION .LE. STRT_POS) THEN
               WRITE(1,2070) POSITION,TRB_FLOW_STAT,
     *                        (TRB_FLOW_PCT*100),TRB_FLOW,TOT_FLOW,
     *                        MIN_FLOW,STAT,AVAL_FLOW,STAT
               IF (FILE_NAME .NE. ' ') THEN
                  WRITE(WUNIT,2070) POSITION,TRB_FLOW_STAT,
     *                              (TRB_FLOW_PCT*100),TRB_FLOW,
     *                              TOT_FLOW,MIN_FLOW,STAT,AVAL_FLOW,
     *                              STAT
               ENDIF
            ENDIF
            GO TO 10
         ENDIF


C>>>     Write a note index at end of the table explaining its
C>>>     characteristics.

         CALL NOTE(FILE_NAME,WUNIT)

      ELSE


C******************************************************************
C**>> If the combination of station A's max flow,the excess, and
C**>  the sum of the tribs. is less than station B's minimum flow
C**>  do the following.
C******************************************************************
```

```
         CALL CLEAR
         CALL HEAD


C>>>     Write a note indicating station B's minimum flow cannot be
C>>>     maintained and write a query letting the user enter the
C>>>     option of seeing the values for station A's max flow,excess,
C>>>     and sum of tributaries, or exit.

         WRITE(1,2090) ENMN
25       WRITE(1,2100)
         CALL TNOUA('OTHERWISE ENTER "N": ',INTS(21))
         READ '(A)',ANSWER


C>>>     If the user chooses to see the values, print them to the
C>>>     screen and to the optional output file.

         IF ((ANSWER .EQ. 'LIST') .OR. (ANSWER .EQ. 'list') .OR.
     *       (ANSWER .EQ. 'LIS') .OR. (ANSWER .EQ. 'lis') .OR.
     *       (ANSWER .EQ. 'LI') .OR. (ANSWER .EQ. 'li') .OR.
     *       (ANSWER .EQ. 'L') .OR. (ANSWER .EQ. 'l')) THEN
            VAR = 1
            CALL CLEAR
            CALL HEAD
               WRITE(VAR,2110) BGMN
               WRITE(VAR,2120) ENMN
               WRITE(VAR,2130) SUM_TRB
               WRITE(VAR,2140) EXCESS
               WRITE(VAR,2150)BGMN,EXCESS,SUM_TRB,ENMN
               IF (FILE_NAME .NE. ' ') THEN
                  VAR = WUNIT
                  WRITE(VAR,2110) BGMN
                  WRITE(VAR,2120) ENMN
                  WRITE(VAR,2130) SUM_TRB
                  WRITE(VAR,2140) EXCESS
                  WRITE(VAR,2150) BGMN,EXCESS,SUB_TRB,ENMN
                ENDIF
         ELSE
            IF ((ANSWER .NE. 'N') .AND. (ANSWER .NE. 'n')) THEN
               GO TO 25
            ENDIF
         ENDIF
      ENDIF
C>>>  If a file was given, close the file.

3500  IF (FILE_NAME .NE. ' ') THEN
         CLOSE(WUNIT)
      ENDIF

      END
      SUBROUTINE WDVR(STRT_POS,END_POS,NAME,WUNIT)
C     ********************************************************************
C     *
C     *                          WDVR PROCEDURE
C     *
C     *                (Stream Availability With Diversions)
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Purpose:
C     *
C     *            The purpose of this procedure is to print a linear
C     *            table beginning and ending with user-given positions
C     *            of available stream flows adjusted by diversions.
C     *            (if any).
C     *
C     *           A table will not be printed if the combination of
C     *           station A's maximum flow,excess, and sum of the
C     *           tributaries is less than station B's minimum flow.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Procedures called:
C     *
C     *                          CLEAR -- To clear the screen.
C     *
C     *                          HEAD  -- Prints the program header.
C     *
C     ********************************************************************
C     ********************************************************************


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN


C     **** DECLARATIONS ****


      INTEGER                                 FUNIT*2,
*                                        I*2,
*                                        WUNIT*4,
*                                        LENGTH*2,
*                                        MODE*2,
*                                        STATUS*2,
*                                        ARRAY(14)*2,
*                                        FLAG*2,
*                                        VAR*4


 REAL                                    BGREF,
*                                        BGMN,
*                                        ENREF,
*                                        ENMN,
*                                        BGN_MILE,
*                                        END_MILE,
*                                        DIV_PER_MILE,
*                                        SUM_TRB,
*                                        POS_SUM_TRB,
*                                        STRT_POS,
*                                        END_POS,
*                                        FLOW_POS,
*                                        TRB_FLOW_PCT,
*                                        TRB_FLOW,
*                                        DVR_FLOW_PCT,
*                                        DVR_FLOW,
*                                        RESID,
*                                        RESID_TOT,
*                                        TOT_FLOW,
*                                        MIN_FLOW,
*                                        MN_DIFF,
*                                        DUMMY,
*                                        CUM_AVL,
*                                        EXCESS,
*                                        POS_TRB_FLOW,
*                                        POSITION



 CHARACTER                               TRB_FLOW_STAT*1,
*                                        NAME*32,
*                                        FLOW_INFO*40,
*                                        KEY*8,
*                                        PATHNAME*32,
*                                        ANSWER*4


 EQUIVALENCE (FLOW_INFO(1:8),KEY)


 COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
```

```
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/THIRD/SUM_TRB,POS_SUM_TRB
      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH




C     *********************************
C     **** QUERY AND OUTPUT FORMAT ****
C     *********************************



2010  FORMAT(/,T17,'FLOW AVAILABILITY ESTIMATED ON BASIS OF OBSERVED',
     *       ' CONDITIONS AND THE IMPLACEMENT OF NEW DIVERSIONS.')

2020  FORMAT(  T17,97('-'))

2030  FORMAT(/,T39,'TRIBUTARY',T56,'NEW',T72,'NEW',T82,'ESTIMATED',
     *       T98,'ESTIMATED',T112,'ESTIMATED')


2040  FORMAT(  T25,'TRIBUTARY',T38,'OR EXISTING',T53,'DIVERSION',
     *       T69, 'DIVERSION',T84,'TOTAL',T96,'MINIMUM FLOW',T112,
     *       'AVAILABLE')

2050  FORMAT(  T4,'LOCATION',T15,'INDICES',T25,'PER CENT',T39,
     *       'DIVERSION',T53,'PER CENT',T72,'FLOW',T85,'FLOW',
     *       T97,'MAINTAINED',T114,'FLOW')

2060  FORMAT(  T5,'(MILE)',T39,'(FT**3/S)',T69,'(FT**3/S)',T83,
     *        '(FT**3/S)',T98,'(FT**3/S)',T112,'(FT**3/S)')

2070  FORMAT(  T5,F7.2,T18,A1,T27,I5,T38,F11.2,T55,I5,T67,F11.2,
     *       T80,F11.2,T92,A1,T96,F11.2,T108,A1,T110,F11.2,T122,A1)

2080  FORMAT(  T25,'       8)NEW DIVERSION FLOW=NEW DIVERSION PER ',
     *       'CENT * AVAILABLE FLOW.')

2085  FORMAT(  T25,'       9) ~ FLOW CONDITIONS HAVE BEEN ALTERED DUE',
     *         ' TO THE IMPLACEMENT OF ONE OR MORE NEW',
     *       /,T25,'          DIVERSIONS.')


2100  FORMAT(/,T20,'THE SPECIFIED MINIMUM FLOW AT STATION B CANNOT',
     *       /,T20,'BE MAINTAINED WITH SPECIFIED NEW DIVERSIONS!')

2110  FORMAT(/,T1,'UNDER THE GIVEN CONDITIONS THE MINUMUM FLOW',
     *       /,T1,'(',F11.2,' FT**3/S) AT B CANNOT BE MAINTAINED.')

2120  FORMAT(/,T1,'ENTER "LIST" IF YOU WISH TO SEE ',
     *       /,T1,'MINIMUM DISCHARGE BALANCE DATA,')
```

```
2130  FORMAT(T1,'SPECIFIED MINIMUM FLOW AT STATION A = ',F11.2)

2140  FORMAT(/,T1,'SPECIFIED MINIMUM FLOW AT STATION B = ' F11.2)

2150  FORMAT(/,T1,'SUM OF ALL TRIBUTARY FLOWS = ', F11.2)

2160  FORMAT(/,T1,'LINEAR INFLOW OR OUTFLOW (COMPUTED FROM OBSERVED',
     *       /,T1,'DISCHARGE BALANCE) = ',F11.2)

2170  FORMAT(/,T2,F11.2,'  + ',F11.2,'  + ',F11.2,'  < ',F11.2)




C     ********************
C     **   BEGIN WDVR   **
C     ********************



C>>>  If a filename was given, open the file.

      IF (NAME .NE. ' ') THEN
      OPEN(WUNIT,FILE = NAME,ACCESS = 'SEQUENTIAL',
     *     FORM = 'FORMATTED')
      ENDIF


      CALL CLEAR

C>>>  Set a dummy variable to the reciprocal of the division per mile.
C>>>  This will be used as a decrement for POSITION.

      DUMMY = 1/DIV_PER_MILE

C>>>  Calculate beginning available flow and the minimum difference.
C>>>  Initialize the total flow with station A maximum flow and the
C>>>  minimum flow with Station A minimum flow.
C>>>  Calculate the excess.

      AVAL_FLOW = BGREF - BGMN
      MN_DIFF = BGMN - ENMN
      TOT_FLOW = BGREF
      MIN_FLOW = BGMN
      EXCESS = ENREF - (BGREF + SUM_TRB)


C*************************************************************************
C**>  If sum of Station A max flow, the excess and the sum of tribs.
C**>  is greater than or equal to Station B's minimum flow,then do
C**>  the following.
C*************************************************************************
```

```
      IF ((BGREF + EXCESS + SUM_TRB) .GE. ENMN) THEN

C>>>>    Write table headings to the output file, if file was given.

         IF (NAME .NE. ' ') THEN
            WRITE(WUNIT,2010)
            WRITE(WUNIT,2020)
            WRITE(WUNIT,2030)
            WRITE(WUNIT,2040)
            WRITE(WUNIT,2050)
            WRITE(WUNIT,2060)
         ENDIF

C>>>     Write table headings to the terminal.

         WRITE(1,2010)
         WRITE(1,2020)
         WRITE(1,2030)
         WRITE(1,2040)
         WRITE(1,2050)
         WRITE(1,2060)



C>>>     Initialize stat indicators to blank

         STAT = ' '
         STATS = ' '



C>>>     Initialize position to the beginning mile location.

         POSITION = BGN_MILE


C>>>     Clear the midas array to 0.

         DO 3 I = 1,14
            ARRAY(I) = 0
3        CONTINUE


C>>>     Find first record according to beginning mile and read record.

         FLAG = FL$RET
         WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
         CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *        INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

         READ(FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
         READ(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
         READ(FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
         READ(FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT


C>>>     Calculate the diversion flow.

         DVR_FLOW = DVR_FLOW_PCT * AVAL_FLOW



C>>>     If the given starting position for the table is equal to the
C>>>     beginning mile, write its relative information to the table
C>>>     (terminal and to output file,if given.)

         IF (STRT_POS .EQ. POSITION) THEN

            IF (NAME .NE. ' ') THEN
               WRITE(WUNIT,2070) POSITION,TRB_FLOW_STAT,
     *                           (TRB_FLOW_PCT*100),TRB_FLOW,
     *                           (DVR_FLOW_PCT*100),DVR_FLOW,TOT_FLOW,
     *                           STATS,MIN_FLOW,STAT,AVAL_FLOW,STAT
            ENDIF

            WRITE(1,2070) POSITION,TRB_FLOW_STAT,(TRB_FLOW_PCT*100),
     *                    TRB_FLOW,(DVR_FLOW_PCT*100),DVR_FLOW,
     *                    TOT_FLOW,STATS,MIN_FLOW,STAT,AVAL_FLOW,STAT



         ENDIF


C>>>     Calculate the cumulative available flow, the residual,
C>>>     and the total residual.

         CUM_AVL = BGMN + EXCESS - ENMN
         RESID = CUM_AVL/((BGN_MILE - END_MILE) * DIV_PER_MILE)
         RESID_TOT = EXCESS/((BGN_MILE - END_MILE) * DIV_PER_MILE)


C>>>     Do while position is greater than the given end position.

10       IF (POSITION .GT. END_POS) THEN

C>>>         Decrement position, initialize stat, clear midas array, and
C>>>         find and read next record according to position from the
C>>>         work file.

            POSITION = POSITION - DUMMY
            STAT = ' '

            DO 5 I = 1,14
               ARRAY(I) = 0
5           CONTINUE
```

```
          FLAG = FL$RET
          WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
          CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *         INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

          READ(FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
          READ(FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
          READ(FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
          READ(FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT

C>>>      Add the residual and the current tributary flow to the
C>>>      available flow.

          AVAL_FLOW = AVAL_FLOW + RESID + TRB_FLOW

C>>>      Calculate the diversion flow for the present position.

          DVR_FLOW = AVAL_FLOW * DVR_FLOW_PCT

C>>>      If diversion flow percentage is greater than 0, set stat.

          IF (DVR_FLOW_PCT .GT. 0) THEN
             STATS = '~'
          ENDIF

C>>>      Subtract diversion flow from the current available flow,
C>>>      and recalculate the total flow and the minimum flow.

          AVAL_FLOW = AVAL_FLOW - DVR_FLOW
          TOT_FLOW = TOT_FLOW + TRB_FLOW - DVR_FLOW + RESID_TOT
          MIN_FLOW = TOT_FLOW - AVAL_FLOW


C>>>      If the available flow is less than 0, set another stat.

          IF (AVAL_FLOW .LT. 0) THEN
             STAT = '*'
          ENDIF



C>>>      If the current position is less than or equal to the
C>>>      starting position , write the position and its relative
C>>>      information to the table.

          IF (POSITION .LE. STRT_POS) THEN
             WRITE(1,2070) POSITION,TRB_FLOW_STAT,
     *                     (TRB_FLOW_PCT*100),TRB_FLOW,
     *                     (DVR_FLOW_PCT*100),DVR_FLOW,TOT_FLOW,
     *                     STATS,MIN_FLOW,STAT,AVAL_FLOW,STAT


             IF (NAME .NE. ' ') THEN
                WRITE(WUNIT,2070) POSITION,TRB_FLOW_STAT,
     *                            (TRB_FLOW_PCT*100),TRB_FLOW,
     *                            (DVR_FLOW_PCT*100),DVR_FLOW,
     *                            TOT_FLOW,STATS,MIN_FLOW,STAT,
     *                            AVAL_FLOW,STAT
               ENDIF
            ENDIF
            GO TO 10

         ENDIF


C>>>     Write note indexes to the table that explain some of its
C>>>     characteristics.

         CALL NOTE(NAME,WUNIT)
         IF (NAME .NE. ' ') THEN
            WRITE(WUNIT,2080)
            WRITE(WUNIT,2085)
         ENDIF
         WRITE(1,2080)
         WRITE(1,2085)
      ELSE


C***********************************************************************
C**>  If sum of station A max flow, excess, and sum of tribs. is less
C**>  than station B's minimum flow, do the following.
C***********************************************************************

         CALL CLEAR
         CALL HEAD

C>>>     Write a note indicating station B's minimum flow cannot be
C>>>     maintained and write a query letting the user enter the
C>>>     option of seeing the values for Station A max flow,excess,and
C>>>     sum of tribs.,or exit.

         WRITE(1,2110) ENMN
25       WRITE(1,2120)
         CALL TNOUA('OTHERWISE ENTER "N": ',INTS(21))
         READ '(A)',ANSWER


C>>>     If the user chooses to see the values, print them to the
C>>>     screen and to the optional output file.

         IF ((ANSWER .EQ. 'LIST') .OR. (ANSWER .EQ. 'list') .OR.
     *       (ANSWER .EQ. 'LIS') .OR. (ANSWER .EQ. 'lis') .OR.
     *       (ANSWER .EQ. 'LI') .OR. (ANSWER .EQ. 'li') .OR.
     *       (ANSWER .EQ. 'L') .OR. (ANSWER .EQ. 'l')) THEN
            CALL CLEAR
            CALL HEAD
            VAR = 1
```

```
               WRITE(VAR,2130) BGMN
               WRITE(VAR,2140)ENMN
               WRITE(VAR,2150) SUM_TRB
               WRITE(VAR,2160) EXCESS
               WRITE(VAR,2170) BGMN,EXCESS,SUM_TRB,ENMN
               IF (NAME .NE. ' ') THEN
                  VAR = WUNIT
                  WRITE(VAR,2130) BGMN
                  WRITE(VAR,2140) ENMN
                  WRITE(VAR,2150) SUM_TRB
                  WRITE(VAR,2160) EXCESS
                  WRITE(VAR,2170) BGMN,EXCESS,SUM_TRB,ENMN
               ENDIF
         ELSE
            IF ((ANSWER .NE. 'N') .AND. (ANSWER .NE. 'n')) THEN
               GO TO 25
            ENDIF
         ENDIF
      ENDIF


C>>>  If a file was given, close the file.

3500  IF (NAME .NE. ' ') THEN
         CLOSE(WUNIT)
      ENDIF

      END
      SUBROUTINE NOTE(NAME,WUNIT)
C     *******************************************************************
C     *
C     *                        NOTE PROCEDURE
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHOR: JOHN E. TERRY AND BARRY LYLE
C     *  DATE:   12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date:  4/87
C     *
C     *  PURPOSE:
C     *
C     *            This procedure attaches an explanatory note to the
C     *            report produced in either procedure STMODVR or
C     *            STMWDVR.
C     *
C     *******************************************************************
C     *******************************************************************

C     **** DECLARATIONS ****

      INTEGER                        WUNIT*4

      CHARACTER                      NAME*32




C     ********************
C     **   NOTE FORMAT  **
C     ********************

2010  FORMAT(/,T25,'NOTES: 1)INDEX U=UNDEFINED.',
     *       /,T25,'       2)INDEX A=REFERENCE STATION A.',
     *       /,T25,'       3)INDEX B=REFERENCE STATION B.',
     *       /,T25,'       4)INDEX C=EXISTING DIVERSION OR TRIBUTARY',
     *                      T74,'(VALUE NEGATIVE IF DIVERSION).',
     *       /,T25,'       5)TRIBUTARY FLOW=TRIBUTARY PERCENT * ',
     *                     T70,'OBSERVED FLOW AT INDEX STATION.',
     *       /,T25,'       6)EXISTING DIVERSION FLOW IS READ IN, ',
     *                      T70,'NOT COMPUTED.',
     *       /,T25,'       7) *  INDICATES THAT THE LINEAR MINIMUM ',
     *                      T73,'DISTRIBUTION AT THIS POINT CANNOT BE',
     *       /,T25,'         MAINTAINED. (EST. TOTAL FLOW  <  EST. ',
     *                       T71,'MINIMUM FLOW) ESTIMATED AVAILABLE ',
     *        /,T25,'         FLOW BEYOND THIS POINT MAY BE DISTORTED.')


      IF (NAME .NE. ' ') THEN
         WRITE (85,2010)
      ENDIF
      WRITE(1,2010)


      END
      SUBROUTINE CLOSEF

C     ********************************************************************
C     *
C     *                           CLOSE FILES
C     *                            PROCEDURE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter
C     *  Date: 5/30/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  Purpose:
C     *
C     *            The purpose of this procedure is to place the flow
C     *            information of a stream into its old or new file
C     *            and close the files.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *      VARIABLES:
C     *
C     *         BGREF -------------- MAXIMUM FLOW AT STATION A.
C     *         BGMN  -------------- MINIMUM FLOW AT STATION A.
C     *         ENREG -------------- MAXIMUM FLOW AT STATION B.
C     *         ENMN  -------------- MINIMUM FLOW AT STATION B.
C     *         BGN_MILE ----------- STATION A LOCATION IN MILES.
C     *         END_MILE ----------- STATION B LOCATION IN MILES.
C     *         DIV_PER_MILE ------- DIVISION/MILE BETWEEN STAT. A & B.
C     *         POSITION ----------- A POSITION WHERE INFO IS LOCATED.
C     *         KEY ---------------- CHARACTER FORM OF POSTION.
C     *         FILE_NAME ---------- NAME FOR PERMANENT FILE.
C     *
C     *         FLOW INFORMATION:
C     *
C     *            POSITION -------- FLOW POSITION.
C     *            TRB_FLOW_PCT ---- PERCENTAGE OF TRIBUTARY FLOW.
C     *            TRB_FLOW -------- TRIBUTARY FLOW.
C     *            TRB_FLOW_STAT --- TRIBUTARY FLOW STAT:
C     *                              'A' - TO REFERENCE TRIBUTARY FLOW
C     *                               TO OBSERVED FLOW AT STATION A.
C     *                              'B' - TO REFERENCE TRIBUTARY FLOW
C     *                              TO OBSERVED FLOW AT STATION B.
C     *                              'C' - TO INDICATE AN EXISTING
```

```
C      *                             TRIBUTARY,IRRIGATION RETURN FLOW,
C      *                             OR DIVERSION.
C      *                             'U' - TO REMOVE A TRIBUTARY OR
C     *                              EXISTING DIVERSION (UNDEFINED).
C     *            DVR_FLOW_PCT ---- PERCENTAGE OF DIVERSION FLOWS.
C     *
C     *******************************************************************
C     *******************************************************************




C     ***MIDAS DECLARATIONS***


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN
$INSERT SYSCOM>ERRD.INS.FTN



C     **** DECLARATIONS ****


      INTEGER                                  ARRAY(14)*2,
     *                                         STATUS*2,
     *                                         LENGTH*2,
     *                                         MODE*2,
     *                                         PUNIT*4,
     *                                         FUNIT*2,
     *                                         FLAG*2,
     *                                         FLAGS*2

      REAL                                     BGREF,
     *                                         BGMN,
     *                                         ENREF,
     *                                         ENMN,
     *                                         BGN_MILE,
     *                                         END_MILE,
     *                                         DIV_PER_MILE,
     *                                         TRB_FLOW_PCT,
     *                                         TRB_FLOW,
     *                                         DVR_FLOW_PCT,
     *                                         POSITION


      CHARACTER                                TRB_FLOW_STAT*1,
     *                                         FILE_NAME*32,
     *                                         PATHNAME*32,
     *                                         KEY*8,
     *                                         FLOW_INFO*40
```

```
      EQUIVALENCE(FLOW_INFO(1:8),KEY)
      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/FIFTH/FILE_NAME,PUNIT
      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH




C     *********************************
C     **** QUERY AND OUTPUT FORMAT ****
C     *********************************



2010  FORMAT(T1,F11.4,T12,F11.4,T23,F11.4,T34,F11.4)

2020  FORMAT(T1,F7.2,T8,F7.2,T15,I4)

2030  FORMAT(T1,F7.2,T8,F5.2,T13,A1,T14,F11.4,T25,F5.2)

2040  FORMAT(/////,T1,'PLEASE WAIT....CLOSING FILES')


C>>>  Print message that files are being closed.

      WRITE(1,2040)


C>>>  Open the permanent file to hold flow data
      OPEN(PUNIT,FILE=FILE_NAME)
      CLOSE(PUNIT,STATUS='DELETE')

      OPEN(PUNIT,FILE = FILE_NAME,FORM = 'FORMATTED',
     *     ACCESS = 'SEQUENTIAL')

C>>>  Write station A's maximum and minimum flows,station B's
C>>>  maximum and minimum flows,the beginning mile, the end mile, and
C>>>  division per mile to the permanent file.

      WRITE(PUNIT,2010) BGREF,BGMN,ENREF,ENMN
      WRITE(PUNIT,2020) BGN_MILE,END_MILE,DIV_PER_MILE


C>>>  Clear the MIDAS array.

      DO 4 I = 1,14
         ARRAY(I) = 0
```

```
4     CONTINUE


C>>>  Retrieve the first record of flow information from the temporary
C>>>  work file.

      FLAGS = FL$RET + FL$FST
      WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION
      CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAGS,
     *           INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))


C>>>  Continue to read records until no records can be found.
C>>>  Write to the permanent file only those records which
C>>>  not undefined unless they have a diversion flow percent
C>>>  not equal to zero (0) .

10    IF (ARRAY(1) .NE. 7) THEN

         READ (FLOW_INFO(1:8),'(F8.0)') POSITION
         READ (FLOW_INFO(9:12),'(F4.0)') TRB_FLOW_PCT
         READ (FLOW_INFO(16:16),'(A1)') TRB_FLOW_STAT
         READ (FLOW_INFO(17:26),'(F10.0)') TRB_FLOW
         READ (FLOW_INFO(27:30),'(F4.0)') DVR_FLOW_PCT

         IF ((TRB_FLOW_STAT .NE. 'U') .OR. (DVR_FLOW_PCT .NE. 0.0))THEN
            WRITE(PUNIT,2030) POSITION,TRB_FLOW_PCT,TRB_FLOW_STAT,
     *                        TRB_FLOW,DVR_FLOW_PCT
         ENDIF

         FLAG = FL$RET + FL$USE + FL$PLW
         CALL NEXT$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAG,
     *              INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))

         GO TO 10
      ENDIF


C>>>  Clear the MIDAS array.

      DO 20 I = 1,14
         ARRAY(I) = 0
20    CONTINUE


C>>>  Beginning with the first record, starting deleting the records
C>>>  from the work file.

      CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FL$RET+FL$FST,
     *           INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
      READ(FLOW_INFO(1:8),'(F8.0)') POSITION

11    IF ((ARRAY(1) .NE. 7) .AND. (POSITION .LE. BGN_MILE)) THEN
```

```
          READ(FLOW_INFO(1:8),'(F8.0)') POSITION
          ARRAY(1) = 0
          CALL DELET$(FUNIT,FLOW_INFO,KEY,ARRAY,FL$USE,
     *                INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
          CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FL$RET+FL$PLW,
     *               INTS(0),INTS(0),INTS(0),INTS(0),INTS(0))
          READ(FLOW_INFO(1:8),'(F8.0)') POSITION
          GO TO 11
      ENDIF


C>>>  Close the files.

      CLOSE(PUNIT)
      CALL CLOSM$(FUNIT,STATUS)


      END
      SUBROUTINE HEAD
C     *******************************************************************
C     *
C     *                          HEAD SUBROUTINE
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHOR: JOHN E. TERRY
C     *  DATE:   12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter
C     *  Date: 5/30/87
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  Purpose:
C     *
C     *              The purpose of this procedure is to print
C     *              the program header to the screen and change
C     *              the screen from 80 chars to 132 characters.
C     *
C     *******************************************************************
C     *******************************************************************



C     *****OUTPUT FORMAT*****

2000  FORMAT(   T1, '?3h' )
2010  FORMAT(/ ,T42,'1m   ','STREAMFLOW AVAILABILITY',
     *  //, T55,'AND',
     *     //,T48,'DATA BASE MANAGER',
     *    //,T37,    '                PROGRAM 0m15;1H')


C>>>  Write program header to the screen as screen size is changed
C>>>  from 80 to 132 characters.

      WRITE(1,2000)
      WRITE(1,2010)

      END
      SUBROUTINE CLEAR
C     ***************************************************************
C     *                                                             *
C     *                      CLEAR PROCEDURE                        *
C     *                                                             *
C     ***************************************************************
C     ***************************************************************
C     *                                                             *
C     *  AUTHOR: JOHN E. TERRY                                      *
C     *  DATE:   12/9/86                                            *
C     *                                                             *
C     *  Converted from the original PL/I by C. R. Baxter           *
C     *  Date: 5/30/87                                              *
C     *                                                             *
C     ***************************************************************
C     ***************************************************************
C     *                                                             *
C     *  Purpose:                                                   *
C     *                                                             *
C     *              The purpose of this procedure is to clear      *
C     *              the screen.                                    *
C     *                                                             *
C     ***************************************************************
C     ***************************************************************


C     ***OUTPUT FORMAT***

2000  FORMAT('2JH')

C>>>  Print the soft keys/escape characters to clear the screen

      WRITE(1,2000)
      END
      SUBROUTINE CRTMP(FILENAME,TEMP)
C     ********************************************************************
C     *
C     *                       CREATE TEMPORARY FILE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR: C. R. BAXTER
C     *  DATE: 5/87
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  PURPOSE:
C     *
C     *           The purpose of this subroutine is to make a copy
C     *           of an old file entered with the copy having
C     *           a date/time tag.
C     ********************************************************************
c     ********************************************************************

C     *****DECLARATIONS*****


      INTEGER          A(28)*2,I

      CHARACTER        D*15,TEMP*60,FILENAME*32,REC*80

      D = ' '


C>>>  Call system subroutine TIMDAT to retrieve the date and time,
C>>>  then substring the date and time into a string called  D.

      CALL TIMDAT(A,INTS(28))
      D(1:1) = '.'
      D(8:8) = '.'
      WRITE(D(2:3),'(A2)') A(1)
      WRITE(D(4:5),'(A2)') A(2)
      WRITE(D(6:7),'(A2)') A(3)
      WRITE(D(9:12),'(I4)')A(4)
      WRITE(D(13:14),'(I2)') A(5)


C>>>  Place zeroes (0) in D string where blanks are located.

      DO 1 I = 1,15
         IF (D(I:I) .EQ. ' ') THEN
            D(I:I) = '0'
         ENDIF
1     CONTINUE
```

```
C>>>  Determine the length of the filename.

      I = 1
10    IF (FILENAME(I:I) .NE. ' ') THEN
         IF (I .LE. 32) THEN
            I = I + 1
            GO TO 10
         ENDIF
      ENDIF


C>>>  Since the total number of characters in a filename can be no more
C>>>  than 32 and the length of the date/time tag is 15 characters,only
C>>>  the first 17 characters of the filename can be used for the copy.

      I = I - 1
      IF (I .LE. 17) THEN
      TEMP = FILENAME(1:I)//D
      ELSE
      TEMP = FILENAME(1:17)//D
      ENDIF


C>>>  Open the old file and its copy, copy the records from the
C>>>  old file to the copy, and close both files.

      OPEN(5,FILE=TEMP)
      OPEN(10,FILE=FILENAME)

20    READ(10,'(A80)',END=3500) REC
      WRITE(5,'(A80)') REC
      GO TO 20

3500  CLOSE(5)
      CLOSE(10)

      END
      SUBROUTINE CRWKFL
C     *******************************************************************
C     *
C     *                      SUBROUTINE CRWKFL
C     *
C     *                      (CREATE WORK FILE)
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHOR:  C. R. BAXTER
C     *  DATE: 2/87
C     *
C     *  PURPOSE:
C     *
C     *           This procedure creates a MIDAS files that will be
C     *           used as a temporary work file by program STMFLOW.
C     *
C     *******************************************************************
C     *******************************************************************

$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN


C     *****DECLARATIONS*****


      INTEGER*2      PRIDEF(6),SECDEF(6,17)
      INTEGER*2      I,J
      INTEGER*2      ERRCOD(2)
      CHARACTER*32   NAME


C>>>  Initialize the primary and secondary key flags.  Since there are
C>>>  no secondary keys, all seventeen are 0.

      PRIDEF(1) = M$WORD + M$ASTR
      PRIDEF(2) = 4
      PRIDEF(3) = 20
      PRIDEF(4) = 0
      PRIDEF(6) = 0
      PRIDEF(5) = 0


      DO 3 I = 1,17
         DO 4 J = 1,6
            SECDEF(J,I) = 0
4        CONTINUE
3     CONTINUE
```

```
C>>>  Create the temporary MIDAS work file FLOW.DAT.

      NAME = 'FLOW.DAT'

      CALL KX$CRE(NAME,INTS(32),M$NRNW,INTS(0),PRIDEF,SECDEF,ERRCOD)

      END
      SUBROUTINE CHECK(POSITION,CHECK_FLAG)

C     ******************************************************************
C     *
C     *                          CHECK PROCEDURE
C     *
C     ******************************************************************
C     ******************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date: 2/87
C     *
C     *  PURPOSE:
C     *
C     *            The purpose of this procedure is to determin if a
C     *            given position is within range of the stream.
C     *
C     *
C     ******************************************************************
C     *
C     *   ALGORITHM:
C     *
C     *      I. IF THE GIVEN POSITION IF BETWEEN OR EQUAL TO
C     *         STATIONS A AND B, THEN SET FLAG TO YES.
C     *
C     *     II. IF POSITION IS OUT OF RANGE, THEN SET FLAG TO NO.
C     *
C     *    III. STOP PROCESSING.
C     *
C     ******************************************************************
C     ******************************************************************
C     *
C     *   VARIABLES:
C     *
C     *      BGN_MILE ------------ STATION A LOCATION IN MILES.
C     *      END_MILE ------------ STATION B LOCATION IN MILES.
C     *      DIV_PER_MILE -------- DIVISIONS PER MILE.
C     *      POSITION ------------ POSITION BEING CHECKED FOR RANGE.
C     *      CHECK_FLG ----------- FLAG INDICATING WHETHER POSITION
C     *                            IS IN RANGE.
C     *
C     ******************************************************************
C     ******************************************************************

C     **** DECLARATIONS ****


      REAL                                        BGN_MILE,
     *                                            END_MILE,
     *                                             DIV_PER_MILE,
     *                                             POSITION

      CHARACTER                                    CHECK_FLAG*3

      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE


2010  FORMAT(/,T1,'THE ENTERED POSITION:',F7.2,' IS OUT OF RANGE',
     *       /,T1,'OF THE STREAM')


C>>>  If the position is in range, set the check flag to yes, otherwise
C>>>  set the check flag to no and print an error message.

      IF ((POSITION .LE. BGN_MILE) .AND. (POSITION .GE. END_MILE)) THEN
         CHECK_FLAG = 'YES'
      ELSE
         CHECK_FLAG = 'NO '
         WRITE(1,2010) POSITION
      ENDIF

      RETURN
      END
      SUBROUTINE RNF(POSITION,FLG)

C     ********************************************************************
C     *
C     *                           RNF PROCEDURE
C     *                         (Record Not Found)
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  C. R. BAXTER
C     *  DATE: 1/87
C     *  MODIFIED: 4/5/87
C     *
C     *  PURPOSE:
C     *
C     *            The purpose of this procedure is to determine
C     *            whether or not a position entered is valid in
C     *            in regard to the given streamflow data.  If a
C     *            is not valid, a message will be printed indicating
C     *            that the record was not found and for the user to
C     *            reenter another position or option.
C     *
C     ********************************************************************
C     ********************************************************************


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN



C     *****DECLARATIONS*****


       INTEGER                                 I*2,
     *                                         J*2,
     *                                         ARRAY(14)*2,
     *                                         MODE*2,LENGTH*2,
     *                                         FLAGS*2,
     *                                         FL*2,FLG*2,
     *                                         FUNIT*2

       CHARACTER                               PATHNAME*32,
     *                                         FLOW_INFO*40,
     *                                         KEY*8

       REAL                                    POSITION


       EQUIVALENCE(FLOW_INFO(1:8),KEY)
      COMMON/SIXTH/PATHNAME,FUNIT,MODE,LENGTH
2000  FORMAT(/,T1,'1m**THIS RECORD DOES NOT EXIST**0m')

2005  FORMAT(//,T1,'THE POSITION IS NOT VALID OR CONSISTENT IN ',
     *      /,T1,'REGARD TO THE INFORMATION GIVEN. PLEASE    ',
     *      /,T1,'START AGAIN WITH ANOTHER POSITION OR OPTION.')



C>>>  Initialize search flag to 1 and MIDAS array to 0.

      FL = 1
      DO 4 I = 1,14
         ARRAY(I) = 0
4     CONTINUE

C>>>  Locate the position in the work file.

      FLAGS = FL$RET + FL$UKY

      WRITE(FLOW_INFO(1:8),'(F8.2)') POSITION

      CALL FIND$(FUNIT,FLOW_INFO,KEY,ARRAY,FLAGS,$3000,
     *            INTS(0),INTS(0),INTS(0),INTS(0))

C>>>  If position is found, set flag to 0.

      FL = 0


C>>>  If position is not located, print RECORD NOT FOUND and message.

3000  IF (FL .EQ. 1) THEN
      WRITE(1,2000)
      WRITE(1,2005)
      ENDIF

C>>>  Set the parameter flag to the search flag. The parameter flag
C>>>  will be returned to the routine from which it was called.

      FLG = FL


      END
      SUBROUTINE FILECK(QUES,FILE_NAME,FLG,LEN)
C     ********************************************************************
C     *
C     *                       FILE CHECK PROCEDURE
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  AUTHOR:  C. R. BAXTER
C     *  DATE: 2/87
C     *
C     *  PURPOSE:
C     *
C     *             The purpose of this procedure is to check for the
C     *             existence of files.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  DEFINITION OF FILE FLAGS (FLG):
C     *
C     *             FLG = 1  -- Check for existence of an old file.
C     *                   2  -- Check for nonexistence of a new file.
C     *                   3  -- Check for nonexistence of
C     *                         a new output file and the option of
C     *                         no file (carriage return)
C     *
C     ********************************************************************
C     ********************************************************************


C     ***SYSTEM DECLARATIONS***


$INSERT SYSCOM>PARM.K.INS.FTN
$INSERT SYSCOM>KEYS.INS.FTN
$INSERT SYSCOM>A$KEYS.INS.FTN


C     *****DECLARATIONS*****

      INTEGER                       EFLAG*2,FLG*2,PUNIT*4,LEN*2

      CHARACTER                     FILE_NAME*32,QUES*40,ANS*1

      LOGICAL                       EXISTS
```

```
C     *****QUERY FORMAT*****


2000  FORMAT(//,T1,'ENTER "Y" IF YOU WISH TO ENTER A NEW ',
     *       /,T1,'FILE NAME, ELSE (CR) AND THE OLD FILE WILL')

2001  FORMAT(//,T1,'1m***FILE DOES NOT EXIST***0m')

2002  FORMAT(//,T1,'1m***FILE EXISTS***0m')


C>>>  Initialize the error flag to 0, logic variable to true, and temp
C>>>  file unit to 5.

      EFLAG = 0
      EXISTS = .TRUE.
      PUNIT = 5


C>>>  Enter the file name

100   CALL TONL
      CALL TNOUA(QUES,LEN)
      READ '(A)', FILE_NAME

C>>>  If a carriage return was made without entering a file name and
C>>>  the file flag does not indicate a check for a blank name,
C>>>  reenter the file name.


      IF ((FILE_NAME .EQ. ' ') .AND. (FLG .NE. 3)) THEN
          GO TO 100
      ENDIF


C>>>  If the user enters END or a carriage return (FLG = 3) then exit
C>>>  routine.

      IF ((FILE_NAME .EQ. 'END') .OR. (FILE_NAME .EQ. ' ')) THEN
         GO TO 3500
      ENDIF


C>>>  Check for the existence of the file entered.

      INQUIRE(FILE = FILE_NAME, ERR = 200,EXIST = EXISTS)


C>>>  Check for the existence of an old file.  If it does not exist,
C>>>  print a message indicating nonexistence and then have the user
C>>>  reenter another file name.

      IF (FLG .EQ. 1) THEN
```

```
         IF (.NOT. EXISTS) THEN
            CALL TONL
            WRITE(1,2001)
            GO TO 100
         ENDIF
      ELSE

C>>>     Check for nonexistence of a new file.If it does exist, print
C>>>     option of entering another file name or overwriting the old
C>>>     file name.

         IF ((FLG .EQ. 2) .OR. (FLG .EQ. 3)) THEN
            IF (EXISTS) THEN
               CALL TONL
               WRITE(1,2002)
               WRITE(1,2000)
               CALL TNOU('BE OVERWRITTEN: ',INTS(16))
               READ '(A)',ANS

               IF (ANS .EQ. ' ') THEN
                  OPEN(UNIT=PUNIT,FILE= FILE_NAME)
                  CLOSE(PUNIT,STATUS='DELETE')
               ENDIF
               IF (ANS .NE. ' ') THEN
                  GO TO 100
               ENDIF
            ENDIF
         ENDIF
      ENDIF


C>>>  If file name has been successfully entered,set error flag to 1.

      EFLAG = 1


C>>>  If error flag is 0, this indicates an illegal format in the file
C>>>  name and it will have to be reentered.

200   IF (EFLAG .EQ. 0) THEN
         CALL TONL
         CALL TNOU('ILLEGAL FORMAT IN FILE NAME',INTS(27))
         GO TO 100
      ENDIF

3500  END
```

```
      SUBROUTINE ENTR(QUES,DATA,LEN)
C     *******************************************************************
C     *
C     *                       ENTR PROCEDURE
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date: 2/87
C     *
C     *  PURPOSE:
C     *.
C     *            The purpose of this procedure is to allow
C     *            interactive entering of real number data.
C     *
C     *******************************************************************
C     *******************************************************************
C     *
C     *     VARIABLES:
C     *
C     *          DATA -------- REAL NUMBER DATA TO BE ENTERED.
C     *          QUES  ------- QUESTION TO BE PRINTED AT TERMINAL.
C     *
C     *******************************************************************


C     *******************************
C     ******** DECLARATIONS *********
C     *******************************


      REAL                                     DATA

      INTEGER                                  FLG,LEN*2
      CHARACTER                                QUES*40




C>>>     Print the query and read the data, checking for errors.


      FLG = 0
10       CALL TONL
         CALL TNOUA(QUES,LEN)
         READ(1,'(F11.0)',ERR = 20) DATA
         FLG = 1
```

```
C>>>      If invalid entry, print error message and repeat query.

20        IF (FLG .EQ. 0) THEN
             CALL TONL
             CALL TNOU('ILLEGAL ENTRY, PLEASE REENTER.', INTS(30))
             GO TO 10
          ENDIF

      QUES = ' '

      END
      SUBROUTINE INFO
C     ********************************************************************
C     *
C     *                         INFO PROCEDURE
C     *
C     ********************************************************************
C     *
C     *  AUTHOR:  JOHN E. TERRY AND BARRY LYLE
C     *  DATE:    12/9/86
C     *
C     *  Converted from the original PL/I by C. R. Baxter.
C     *  Date: 2/87
C     *
C     *
C     *  PURPOSE:
C     *
C     *           The purpose of this procedure is to print the
C     *           station information of a given stream to the screen.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *    ALGORITHM:
C     *
C     *       I. WRITE THE STATION INFORMATION HEADINGS INDICATING
C     *          WHICH DATA FILE IS USED.
C     *
C     *      II. DETERMINE THE INITIAL WATER AVAILABLE AT STATION  A
C     *          AND STATION  B.
C     *
C     *     III. WRITE THE FOLLOWING TO THE SCREEN(FORMATTED):  STATION A
C     *          MAXIMUM,STATION B MAXIMUM,STATION A MINIMUM,
C     *          STATION B MIN,  STATION A LOCATION,STATION B LOCATION,
C     *          AND WATER AVAILABLE AT STATION A AND B.
C     *
C     *      IV. WRITE NOTE INDICATION THAT DISCHARGE BALANCE BETWEEN
C     *          STATIONS A AND B IS DISTRIBUTED LINEARLY.
C     *
C     *       V. STOP PROCESSING.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *  PROCEDURES CALLED:
C     *
C     *                      CLEAR --- Clears the screen.
C     *
C     ********************************************************************
C     ********************************************************************
C     *
C     *    VARIABLES:
C     *
C     *       BGREF ------------ STATION A MAXIMUM FLOW.
```

```
C     *       BGMN  ------------ STATION A MINIMUM FLOW.
C     *       ENREG ------------ STATION B MAXIMUM FLOW.
C     *       ENMN  ------------ STATION B MINIMUM FLOW.
C     *       BGN_MILE --------- STATION A LOCATION IN MILES.
C     *       END_MILE --------- STATION B LOCATION IN MILES.
C     *       DIV_PER_MILE ----- DIVISION/MILE BETWEEN STATIONS A & B.
C     *       AVALA ------------ WATER AVAILABLE AT STATION A.
C     *       AVALB ------------ WATER AVAILABLE AT STATION B.
C     *       FILE_NAME -------- NAME OF THE FILE WHICH IS FROM OR
C     *                          WILL BE PLACED.
C     *
C     *******************************************************************
C     *******************************************************************


C     **********************
C     **** DECLARATIONS ****
C     **********************

      INTEGER                                  PUNIT*4

      REAL                                     BGREF,
     *                                         BGMN,
     *                                         ENREF,
     *                                         ENMN,
     *                                         BGN_MILE,
     *                                         END_MILE,
     *                                         DIV_PER_MILE,
     *                                         AVALA,
     *                                         AVALB

      CHARACTER                                FILE_NAME*32


      COMMON/FIRST/BGREF,BGMN,ENREF,ENMN
      COMMON/SECOND/BGN_MILE,END_MILE,DIV_PER_MILE
      COMMON/FIFTH/FILE_NAME,PUNIT


C     **** OUTPUT FORMAT ****


2005  FORMAT(10(/,' '))
2010  FORMAT(T50,'REFERENCE STATION INFORMATION',T104,
     *        'FILE NAME: ',A15)

2020  FORMAT(T1,130('='))
```

```
2030  FORMAT(T64,'==')

2040  FORMAT(T22,'STATION A(UPSTREAM)',T64,'==',T85,
     *       'STATION B(DOWNSTREAM)')

2050  FORMAT(T64,'==',/,T64,'==')

2060  FORMAT(T1,130('='))

2070  FORMAT(T8,'OBSERVED',T22,'SPECIFIED',T37,'AVAILABLE',T64,
     *        '==',T73,'OBSERVED',T87,'SPECIFIED',T102,'AVAILABLE')

2080  FORMAT(T10,'FLOW',T23,'MINIMUM',T39,'FLOW',T51,'LOCATION',T64,
     *       '==',T75,'FLOW',T88,'MINIMUM',T104,'FLOW',T113,'LOCATION')

2090  FORMAT(T7,'(FT**3/S)',T22,'(FT**3/S)',T37,'(FT**3/S)',T52,
     *       '(MILE)',T64,'==',T72,'(FT**3/S)',T87,'(FT**3/S)',T102,
     *       '(FT**3/S)',T114,'(MILE)')

2100  FORMAT(4X,F11.2,4X,F11.2,4X,F11.2,6X,F7.2,T64,'==',4X,F11.2,
     *       4X,F11.2,4X,F11.2,3X,F7.2)

2110  FORMAT(T1,130('='))

2120  FORMAT(//,T1,'***NOTE***')

2130  FORMAT(  'RESIDUAL FROM DISCHARGE BALANCE BETWEEN STATION A ',
     *         'AND STATION B IS DISTRIBUTED LINEARLY.')

2140  FORMAT( /,T1,'DIVISION PER MILE IS ',I3)


      CALL CLEAR

C>>>  Print the station information headings.

      WRITE(1,2010) FILE_NAME
      WRITE(1,2020)
      WRITE(1,2030)
      WRITE(1,2040)
      WRITE(1,2050)
      WRITE(1,2060)
      WRITE(1,2070)
      WRITE(1,2080)
      WRITE(1,2090)


C>>>  Determine intitial available flow at station A and station B.

      AVALA = BGREF - BGMN
      AVALB = ENREF - ENMN
```

```
C>>>  Print the station information of the given stream.

      WRITE(1,2100) BGREF,BGMN,AVALA,BGN_MILE,ENREF,ENMN,
     *              AVALB,END_MILE
      WRITE(1,2110)
      WRITE(1,2120)
      WRITE(1,2130)
      WRITE(1,2140) DIV_PER_MILE

      END
```

☆ U.S. GOVERNMENT PRINTING OFFICE 1988—548-200/80006