Documentation of a combined watershed and stream-aquifer modeling program based on Swat and Modflow

Vol. 2. Added or modified source code

Samuel P. Perkins[a] and Marios Sophocleous[b]

[a]Postdoctoral Research Associate, perkins@kgs.ukans.edu
[b]Senior Scientist, marios@kgs.ukans.edu

Open-File Report No. 99–25, June 1999
revised Jan. 7 2000

Kansas Geological Survey, The University of Kansas
1930 Constant Ave., Lawrence KS 66047

Documentation of a combined watershed and stream-aquifer modeling program based on Swat and Modflow
Volume 2. Added or modified source code

Samuel P. Perkins[a] and Marios Sophocleous[b]
[a]Postdoctoral Research Associate, perkins@kgs.ukans.edu
[b]Senior Scientist, marios@kgs.ukans.edu

Open-File Report No. 99–25, June 1999
rev. Jan 7 2000

Kansas Geological Survey, The University of Kansas
1930 Constant Ave., Lawrence KS 66047

Contents

Coordination of Swat and Modflow: program organization

Swat and Modflow are coordinated either by independent execution with data transfer by file, or by linked execution in which data are passed between programs by reference to arrays in subroutine argument lists. Two sets of subroutines, one for Swat and one for Modflow, were developed for this purpose. On Swat's side, subroutine Preswb was written to provide necessary conversions of Swat's results for use in Modflow. A companion program, Swbavg, provides a way to combine results from Swat to represent spatial heterogeneities existing within the watershed by calculating areally weighted averages of "hydrologic response units" (HRUs), i.e. the various areal components of the watershed with respect to soil type, land use, and management practices. On Modflow's side, a set, or "package" of subroutines referred to as Modswb, was written to make use of Swat's watershed simulations. In addition, Modflow's standard Well and Stream packages have been modified to accommodate the Swat-Modflow (or Potyldr-Modflow) linkage and to incorporate some extensions of those models. Other packages have been written for Modflow essentially to streamline the analysis of results. These include Modrsd, to analyze water level residuals, and Modpost, to provide some postprocessing capabilities that bypass the intermediate steps of writing Modflow's standard output file and then extracting results of interest from that file.

Swat begins by reading the input files associated with a particular case, and opening those with daily values, i.e., weather data; see the manual on input instructions in Appendix A for details. Swat calculates hydrologic fluxes for each subbasin of the watershed on a daily basis and accumulates them over the aquifer time step, designated to be a day, month, year, or an average of all years of simulation. At the end of each aquifer time step (a day, month, or year), Swat calls subroutine Preswb to prepare results for Modflow that are either to be written to a file or passed by reference to an array in a subroutine call to Modflow.

Modflow, coded as subroutine Modflo and called either by Swat or by a small mainline for stand-alone execution, includes the Modswb package, which provides an interface between Swat's lumped parameter model of a watershed and Modflow's distributed parameter model of an aquifer. The Modswb package consists of four subroutines whose names and functions conform to the conventions followed in Modflow. Swb1al allocates memory in Modflow for the watershed-aquifer interface. Swb1rp maps the geographical extent of a watershed's subbasins onto a gridded aquifer domain and the subbasin outflows onto a stream network. Swb1fm converts results obtained from Swat for each subbasin into conditions for Modflow's aquifer solution, including recharge, irrigation, evaporation, and tributary flow to streams. Swb1bd summarizes baseflow and evaporation from the water table for each subbasin. Bothe Swb1fm and Swb1bd make necessary conversions between Swat's hydrologic depths (mm), i.e. flow rates divided by subbasin area, and Modflow's flow rates [L^3/T].

If Swat and Modflow are to be run separately, Modflow calls subroutine Swb1fm in the Modswb package for each time step to read a hydrologic summary from the file written by Hydbal into array SHED; and to set up conditions on which Modflow's solution is based, including recharge, pumping, and lateral inflow to streams.

In a linked execution, Swat calls Modflow through entry points Modflo, to initialize a simulation; Modper, to initialize each stress period, and Modstp, to solve the groundwater equations for a time step. For each time step, Swat calls Modflow according to a procedure that is illustrated below for monthly time steps. Swt2mod is first called to pass the hydrologic summary from array SSUB in Swat to array SHED in Modflow. Modflow is then called via entry point Modstp to solve the aquifer equations, and Mod2swt retrieves baseflow and water table evaporation as hydrologic depths, based on Modflow's solution. Hydbal then writes the summary of hydrologic fluxes to a file, incorporating results from Modflow. The use of Modflow's entry points and Hydbal's subroutines are outlined in the following excerpt of code from file month.h, included in Swat's mainline (SwatMain):

```
      if (iopabs.eq.2) then
        ndays = mndays(mo1,idleap)  !no. days in month (time step)
        write (balttl,'(3i5)') iyr,mo1,ndays    !char balttl*15

            Linked execution: call Modflow for each solution step
        if (iopswt.gt.0) call MODSTP (iopswt,ndays,
     1      kper,nstp,kstp,iyr,msubo,mb,idxsub,ssub)

            Summarize combined results on balance file:
        call PRESWB (iobal,iosum,balttl,mxcomp,msubo,lutot,
     1              idxsub,idxssb,smm,ssub,flu,sw,v,numhru,idxhru,
     1              isolhr,iusehr,soilwt,crpfrc,frcirr(iy),hruwt)
```

```
      end if
```

The remainder of this section provides details on specifying options for the linkage between Swat and Modflow; the data that are passed between the two by file and memory reference; and its implementation.

Program code associated with the SWAT-MODFLOW linkage

Some of the key aspects of the Swat-Modflow linkage have been consolidated into the subroutine Preswb, which is called by Swat at the end of each time step, and does the following:
a) Preswb makes the conversions from hydrologic depths simulated by SWAT (mm) to specify flow rates for boundary conditions in MODFLOW (cfs for the Lower Republican River basin model), previously handled in Modswb;

b) The conceptual models for ground water recharge and tributary inflow, which are given by sums of hydrologic components simulated by Swat, are evaluated as part of Preswb. Previously, this was done as part of Modswb.

c) Spatial weight functions are calculated by Preswb and included as part of the balance file written for each HRU. These weights were previously calculated by a separate program (WEIGHTS). A simplified version of SWBAVG takes an average over the HRUs.

Modflow's memory requirements are mostly handled by its array X, which is in unlabeled common. An exception to this convention was made with the array Shed, which was declared separately, primarily to provide a means of passing data from Swat to Modflow by reference to the array in subroutine Modflo's argument list while making only minimal changes to Swat. The array SHED(30,mb) was declared in SWAT's mainline (included on file swatmod1.h) to be passed to Modflow with the data required for the linkage; subroutine Preswb is called by Swat at the end of each time step to set up array SHED for a subsequent call to Modflow.

For Modflow's stand-alone execution mode, the SHED array is also dimensioned in Modmain, the "stub" mainline that is used to call Modflow. In this mode, SWAT's results are read from the balance file into the SHED array instead of being passed from Swat by reference to array SHED in a subroutine call to Modflow.

Acronyms defined for integrated watershed model code

SWAT: (changes are summarized below).
  PRESWB: summarize stream-aquifer (SWAT-MODFLOW) connections.
  SUBWTS: calculate weight functions for taking averages of HRUs in SWBAVG.
SWBAVG: take area-weighted averages over HRUs.

MODFLOW: added or modified packages

MODFLOW mainline: rewritten as subroutine to implement linkage options.
MODSWB: implements data interface and hydrologic connections.
MODSTR: STREAM package modified for linkage and model requirements.
MODWEL: WELL package modified for linkage and model requirements
MODRSD: Evaluate water table residuals during simulation.
MODPOST: Postprocessing functions; write results for analysis and display.

Flow of execution for modified Swat program

A review of the procedure followed in SWAT to simulate the soil water budget components for each subbasin is outlined in Fig. 1. Subroutines in SWAT were added or modified to coordinate with MODFLOW and to incorporate added options that are specified in the modified Control Codes (*.cod) input file for Swat unless otherwise noted below. Item numbers correspond to superscripts on subroutine names in Fig. 1; options in **bold** type are defined as follows.

1. **Clicon** was modified as follows. Option **iopwfl** allows daily precipitation measurements for all weather stations to be read from one file, and temperature measurements for all stations from another. Option **iopwea** allows reading daily measurements of wind speed, relative humidity, and solar radiation.
2. **Purk18** was modified to include an option to represent a soil profile underlain by bedrock, specified by setting option **iaqufr**=0. In this case, percolation out of the root zone is blocked, resulting in a corresponding increase in subsurface lateral flow and soil water content. This option is associated with the HRU scheme option **iophru**=2, which disaggregates a subbasin into components with or without an underlying aquifer but assumes deep ground water.
3. **Evap8** was modified to call the added subroutine **Penman**, below, based on the added option **ipet**=3.
4. **Penman** was added as an option to calculate potential evaporation according to the procedure recommended in Shuttleworth (1993), which accounts for the effect of long-wave radiation, i.e., black body emissions from earth; and which can make use of daily measurements, if available, of wind speed, relative humidity, and solar radiation (see option **iopwea**).
5. **Crpmd6** was modified to observe a daily maximum irrigation depth, **swminf**, and allow using a threshold on soil water content to trigger irrigation, specified by the option **irr**=3 in the management codes (*.mco) input file.
6. **Swu19** was modified to avoid references to undeclared array locations.
7. **Subbasin** was modified to represent soil water uptake from shallow ground water, specified by option **iaqufr**=2, and associated with the HRU scheme **iophru**=3, which disaggregates shallow and deep ground water components. Simulation uses successive approximation as follows. Evaporation from shallow ground water is given by a previous iteration of Modflow's solution for the current time step, and is redistributed over soil layers for which water content is less than field capacity. The previous iteration of Modflow's solution can be based on the HRU scheme **iophru**=2, summarized above for **Purk18**. The successive approximation approach enables the

```
Read or generate daily weather observations, Clicon[1];
do for each subbasin (subr Subbasin):

   Infiltration and runoff for storms and snowmelt:
      Snowmelt: based on snow depth and temperatures of snow and air;
         Snom5.
      Soil temperature: simulate for each layer; Solt22;
      Runoff: curve number procedure (USDA, 1972), Volq11;
      Transmission loss for ephemeral stream channels (Lane, 1982), subr
         Tran12;
      Infiltration = (precipitation + snowmelt) - (runoff +
         transmission loss).
      Percolation and lateral subsurface flow: apply storage routing to excess
         water content above field capacity (1/3 bar); subr Purk18[2].

   Evaporation, irrigation, and soil water redistribution:
      Evaporative demand from soil and plants, $e_t = es + ep$ Evap8[3]
         Potential evaporation, $e_0$ (added option iopet=3, subr Penman[4])
         Plant uptake, $e_p$, in terms of $e_0$ and leaf area index, lai: $e_p =$
             $e_0 \cdot \min(1, lai/3)$.
         Soil evap., limited by above-ground biomass, cv (t/ha), and plant
             uptake:
             $es = \min[e_0 \exp(-cv/20), e_0 - e_p]$
      Plant growth and irrigation models, Crpmd6[5]
      Supply evap. demand with avail. soil water above wilting pt (15 bar)
             Swu19[6]
      Redistribute uptake from shallow ground water over soil profile,
      Subbasin[7]
   end do
end do
```

Figure 1. Procedure followed in SWAT's daily hydrologic simulation for each basin. Superscripts 1-7 indicate code changes or additions.

coupling of Swat and Modflow solutions through shallow ground water evaporation while allowing separate execution of Swat for each HRU, SWBAVG to average the HRU results, and Modflow based on the averaged HRUs.

Flow of execution for modified Modflow program

MODFLOW's mainline was rewritten in the form of a subroutine to allow MODFLOW to serve dual modes of stand-alone and linked execution. Fig. 2 summarizes MODFLOW's mainline rewritten as a subroutine. Further details are provided in Perkins and Sophocleous (1999b-c). In linked mode (**iopswt**=1), SWAT calls MODFLOW as a subroutine (Fig. 1 of S&P, 1999). In the stand-alone mode (**iopswt**=0), SWAT's simulation results are passed to MODFLOW by the intermediate

```
begin
  subroutine **Modflo** (main entry point)
    On the initial call to Modflo()
      call Bas1df to define grid size (rows, columns, layers), stress periods, options;
      allocate array space (AL routines);
      call Swb2al (follows Str1al) to allocate arrays for watershed-aquifer linkage;
      Read and prepare data that remain constant throughout simulation (RP routines);
      call Swb2rp (follows Bcf1rp) to associate subbasins with aquifer grid cells;
    end if;

    Do for each stress period kper=1,nper (implement loop with a counter and a go-to)
      define stress period length and divide into time steps;
      Read and prepare data that remain in effect for the stress period (RP routines);

      Do for each time step kstp=1,nstp (implement loop with a counter and a go-to)
        entry Modstp ()  [entry point provided for calls from Swat]
        call Swb2fm (before Bas1ad) to set up conditions based on Swat's results;
          if (iopswt = 0) read Swat's simulation results from the HRU-averaged balance
          file;otherwise, Swat calls Modstp() and passes results by reference to array Shed.
        call Bas1ad to advance time and initial hydraulic heads; calculate time step;
        Do for each iteration of approximating the solution (kiter=1,mxiter):
          formulate finite difference equations (FM routines) and solve equations;
        end iteration on solution;
        calculate budget terms for mass balance (BD routines);
        call Swb2bd to summarize Modflow results;
        call Bas1ot to optionally save and print results;
        if (iopswt > 0) return to Swat
      end time step;
    end stress period;
    entry Modend (to end Modflow execution with call from Swat)
      close files;
end.
```

Figure 2. Flow of execution for MODFLOW mainline rewritten as a subroutine.

"balance" data file for each solution time step. The stand-alone mode of executing MODFLOW is summarized as follows.

Stand-alone Modflow execution

```
  program modmain
    data iopswt/0/      ! set option to run modflow as stand-alone program.
    data cnvtim /1./    ! time conversion factor (to call MODFLOW from SWAT)
    call MODFLO (iopswt, cnvtim, delt, kper, nstp, kstp)
  end
```

In the stand-alone mode, arguments **iopswt** and **cnvtim** are defined in the mainline. The rest are defined within the call to Modflo(). (**delt**: time step [T]; **kper, kstp, nstp**: current stress period, time step, and no. time steps).

Compiling and linking Swat, Swbavg, and Modflow

Source files for the programs described here are zipped as file \zips\srcfiles.zip; to examine these, unzip into \src. Programs were run under Lahey's Fortran 77 (em32). The following compiler switches were specified:

/b array bounds are checked during execution; program bombs if bounds are exceeded.
/d direct files without headers: produces Modflow's binary output in standard form.
/i check subroutine interfaces for consistent arguments.
/l generate line number traceback table to locate run-time errors.
/p protect constant arguments.

In addition, the switch /h ("hard copy") was used to produce the source code listings shown below.

Example: compile modified Swat source file Cliconsp.for.

```
f77l3 /b/d/i/l/p >cliconsp.log
```

Compiler messages are redirected in this example to log file cliconsp.log. The following commands to link Swat and Modflow into executable programs were used with Modflow's source and object files located in c:\gh\modswb, and Swat's source and object files located in c:\gh\modswb\swat. The files to be compiled are indicated by the object files that are to be linked as listed below.

Linking Swatmod (Swat modified to allow linked or separate execution of Modflow)

Swatmod is linked into the executable file Swatmod.exe in directory c:\gh with the following command on DOS batch file Lswatmod.bat:

```
386link @swatmod -symbol -stub runb -exe swatmod.exe
```

Object files to be linked are listed on file swatmod.Lnk, which is referenced on the line above as @swatmod[.Lnk], and which contains the following:

```
! file swatmod.lnk  lists object files for linked Swat and Modflow;
! invoked by lswatmod.bat:  jun 28 99 version
!
swatmain       !main
blockd,readinpt,subbasin !individual Swat modules
clicon         !weather input or generated data
evap8          !potential evaporation (called by subr subbasin, calls Penman)
crpmd          !plant water use (called by subr subbasin, calls swu19)
swu19          !soil water use
swata-e,swatf-o,swatp-r,swats-y ! Swat modules grouped alphabetically
penman         !independent calculation of ref. ET (Penman-Monteith).
preswb         !Swat hydrologic balance; pass fluxes between Swat & Modflow

..\modflo      ! Modflow's main routine, changed into a subroutine called by Swat
..\modbas
..\modbcf
..\modwel      ! modified to compare solution with water level measurements
```

```
..\modevt    ! modified to summarize evap from aquifer for each subbasin
..\modrch
..\modsub    ! includes other packages
..\modutl    ! utility package
..\modpcg2   ! more or less standard Modflow
..\modswb    ! module added to coordinate watershed with stream & aquifer
..\modstr    ! version of STREAM package w/ option for diffusive routing
..\indexx    ! indexed sorting and ranking routines indexr,indexi,irank
..\modpost   ! postprocessing subroutines (for budgets and hydrographs)
..\modrsd    ! read observed water levels and calculate residuals
```

Source code for program Swbavg is contained on a single file with, Swbavg.for, located with Swat's source code. Its compiled object file is linked into an executable file by the command

```
386link swbavg -symbol -stub runb -exe swbavg.exe
```

Linking Modflow for stand-alone execution and coordination with Swat

Modflow is linked into the executable file Modflow.exe in directory c:\gh with the following command on DOS batch file Lmodflow.bat:

```
386link @modflow -symbol  -stub runb -exe modflow.exe
```

This refers to file modflow.Lnk for object files to be linked, given by

```
modmain    ! mainline
modflo,modbas,modbcf,modwel,modevt,modrch,modsub,modutl,modpcg2
modswb,modstr,indexx,modrsd,modpost
```

Flow of execution for Swat and Modflow

<u>Linked Swat-Modflow execution</u>

Options **iopswt** and **iopmod**, read from Swat's input control file (~.cod), affect execution control as follows. For each year of a simulation, Swat simulates watershed hydrology on a daily basis, and summarizes its results at the end of each aquifer time step (a day, month, or year, depending on iopmod). If *iopswt*>0, Swat calls Modflow through entry points Modflo, Modstp, and Modend as shown in the program outline below. Option **iopswt** is passed to Modflow through these entry points to return execution control from Modflow back to Swat just before the next entry point is encountered, as shown below for the modified Modflow program structure. If **iopswt**=0, Swat bypasses calls to Modflow, and calls subroutine Hydbal at the end of each time step to write a hydrologic summary that can be read in a subsequent, separate run of Modflow.

Modified Swat flow of execution:

1. Simulate each hydrologic response unit (HRU) with Swat as follows:

```
begin (Swatmain)
  Initialize simulation;
  Read options for Swat-Modflow linkage (input file *.cod), e.g. aquifer time step;
  For each year, run simulation with daily time steps:
  Read daily weather variables (subr. Clicon);
  Simulate hydrologic response (subr. Subbasin);
    At the end of each aquifer time step:
    call Preswb to prepare results for Swbavg and Modflow:
        a) calculate groundwater recharge and tributary flow from simulation results;
        b) calculate rate of change in storage for soils and ponds;
        c) convert hydrologic flux depths to flow rates in units consistent with Modflow;
        d) calculate weights to be applied to HRUs;
        e) if (iopswt = 0) write balance file to be read by Swbavg for averaging HRUs;
      if (iopswt > 0) call Modflo to solve hydraulic head distribution for the time step;
  end of year;
end.
```

2. Run SWBAVG to take a weighted average over HRUs and provide results as a file to be read by Modflow's Modswb package.

3. Run Modflow, using HRU-averaged results from Swat and SWBAVG to specify conditions for its solution in each time step.

Modified Modflow program structure

The modified structure of Modflow is summarized by the outline shown below, which corresponds to the flowchart in Fig. 13 of Modflow's manual (McDonald et al., 1988). Calls to subroutines in the Modswb package, added to this structure to coordinate Modflow and a watershed model, are identified by bold type. The Modswb package is invoked by setting **iunit**(6) > 0 in Modflow's basic package input; see Modflow manual, p. 4.9. Modflow may be run either as a stand-alone program or from a daily watershed simulation program (Swat), using the option **iopswt**, described above. To allow Modflow to be controlled by calls from Swat at the end of each time step, Modflow's looping structures on stress periods and time steps have been converted from **do** loops to a more primitive but functionally equivalent form using conditional jumps with counters, so as to circumvent Fortran compilers' normally justified prohibition of jumps into a **do** loop.

Flow of execution for modified Modflow program structure:

The following flow of execution in Modflow is intended to serve the dual modes of stand-alone execution (iopswt=0), in which Swat's results are passed to Modflow by way of an intermediate data file (the "balance" file); and linked execution (iopswt=1), in which Swat calls Modflow as a subroutine. The stand-alone mode has been extensively tested and used for application to the Lower Republican River basin model and, in earlier versions, to the Rattlesnake Creek and Wet Walnut Creek watershed models. The linked execution mode was implemented as part of the earliest versions of this linkage for application to the Lower Republican River basin model, but has not been extensively tested, primarily because the linked mode was not of much use because it did not allow accounting for spatial heterogeneity in Swat within the aquifer solution time step. Once this problem has been solved, applications may be found for the linked mode of execution.

```
begin
  subroutine Modflo (main entry point)
    On the initial call to Modflo()
      call Bas1df to define grid size (rows, columns, layers), stress periods, options;
      allocate array space (AL routines);
      call Swb2al (follows Str1al) to allocate arrays for watershed-aquifer linkage;
      Read and prepare data that remain constant throughout simulation (RP routines);
      call Swb2rp (follows Bcf1rp) to associate subbasins with aquifer grid cells;
    end if;

    Do for each stress period kper=1,nper (implement loop with a counter and a go-to)
```

```
        define stress period length and divide into time steps;
        Read and prepare data that remain in effect for the stress period (RP routines);

        Do for each time step kstp=1,nstp (implement loop with a counter and a go-to)
          entry **Modstp** () [entry point provided for calls from Swat]
          call **Swb2fm** (before Bas1ad) to set up conditions based on Swat's results;
            if (iopswt = 0) read Swat's simulation results from the HRU-averaged balance
            file;otherwise, Swat calls Modstp() and passes results by reference to array
            Shed.
          call Bas1ad to advance time and initial hydraulic heads; calculate time step;
          Do for each iteration of approximating the solution (kiter=1,mxiter):
            formulate finite difference equations (FM routines) and solve equations;
          end iteration on solution;
          calculate budget terms for mass balance (BD routines);
          call **Swb2bd** to summarize Modflow results;
          call Bas1ot to optionally save and print results;
          **if (iopswt > 0) return to Swat**
      end time step;
    end stress period;
    **entry Modend** (to end Modflow execution with call from Swat)
      close files;
  end.
```

<u>Stand-alone Modflow execution</u>

In stand-alone mode, calling program ModMain, shown below, calls subroutine Modflo, passing **iopswt**=0 so that the complete structure of Modflow shown above is executed before returning to ModMain.

```
    program modmain
    data iopswt/0/      ! set option to run modflow as stand-alone program.
    data cnvtim /1./    ! time unit conversion factor (for calling MODFLOW from SWAT)
      call MODFLO (iopswt, cnvtim, delt, kper, nstp, kstp)
    stop
  end
```

<u>Definition of argument list passed to subroutine Modflo</u>

In the case of stand-alone execution (**iopswt**=0, above), only arguments **iopswt** and **cnvtim** need be defined by the mainline, as shown; the remaining arguments are defined within the call to subroutine Modflo. For combined Swat-Modflow execution (**iopswt**>0), the argument lists for Modflo, Modper, and Modstp are the same, and are defined as follows.

cnvtim: conversion from Swat time units (days) to Modflow time units [Tmd]; given by **cnvtim = tsec(itmuni)**/86400 [days/Tmd]; **tsec** and **itmuni** are defined below.

cnvtim is calculated in MODSWB subroutine SWB1RP and, if iopswt=1, is passed back to Swat via MODFLO's argument list on Swat's initial call to MODFLO.

itmuni: time unit option (0-5); specified in MODFLOW's Basic Package input file.

tsec: time unit length (sec) corresponding to itmuni as follows:

itmuni	tsec(*itmuni*)	time unit
0:	1	undefined
1:	1	second
2:	60	minute
3:	3600	hour
4:	86400	day
5:	31536000	year (365 days)

delt: time step [T]; not used until after the MODSTP entry point, but the argument list of the first entry point should contain all arguments of ensuing entry points. For the stand-alone option (*iopswt*=0), time step and multiplier are used as specified in MODFLOW's basic package input file, i.e. standard input is default.

kper: index to current stress period;

nstp: number of time steps for current stress period, kper;

kstp: index to current time step.

Preswb: summarize Swat or combined Swat-Modflow results

As indicated above, Preswb does the following at the end of each aquifer time step:

a) calculate groundwater recharge and tributary flow from simulation results;
b) calculate rate of change in storage for soils and ponds;
c) convert hydrologic flux depths to flow rates in units consistent with Modflow;
d) calculate weights to be applied to HRUs;
e) if (**iopswt** = 0) write balance file to be read by Swbavg for averaging HRUs;

Program Swbavg: representing heterogeneity within subbasins

Program Swbavg was written to represent spatial heterogeneities existing within subbasins of a watershed by calculating areally weighted averages of "hydrologic response units" (HRUs), i.e., the various soil types, land uses, and management practices existing within each subbasin, in order to produce composite results from Swat to be used as input for Modflow. For each subbasin and time step of the simulation, the areal fractions corresponding to its HRUs, i.e., components of the watershed's spatial heterogeneity, are calculated. These areal fractions represent weight functions that are distinct for each subbasin, and are used to calculate a weighted average over the HRUs for each hydrologic component (SHED vectors 10-22 in the balance file). For each HRU, or component of the watershed's heterogeneity, a separate case of the watershed model is simulated by Swat, for which all subbasins of the watershed are assumed to be homogeneous in the characteristics of that HRU. Program Swbavg then takes the weighted average over all HRUs to produce a composite of the simulations.

In the case of the Lower Republican River Basin, HRUs represent heterogeneities with respect to six soil types and three land uses. Land use and management practices are represented by three schemes: wheat, sorghum, and fallow rotations with no irrigation ("wsf"); irrigated corn ("irc"); and range and pasture without irrigation ("r/p").

Whereas areal fractions of soil types within each subbasin are naturally fixed with respect to time, their land use areal fractions vary from year to year. To represent this land use change, the initial table of weight functions is updated annually.

Modswb: a Swat-Modflow connection

Modswb associates the subbasins defined in Swat with aquifer grid domains, and watershed outflows with the stream network defined in Modflow; updates water rights appropriations for each subbasin at the beginning of each stress period; maps hydrologic fluxes calculated by Swat onto the aquifer grid, and summarizes for each subbasin the fluxes relevant to Swat's hydrologic balance. Options affecting Modswb execution are summarized below; for details of implementation, see source code in Volume 2. "Modswb" is an acronym for Soil Water Balance, referring to the hydrologic summary of Swat's results written by subroutine Preswb. Modflow's basic package input file invokes the Modswb package by setting the Basic input variable iunit(6) > 0.

Modswb subroutines **Swb2al** and **Swb2rp** obtain input data regarding array space requirements, options, and connections between the watershed, stream network and aquifer grid from the Swb input file; see, e.g., file rptest.swb. If Modflow is run as a stand-alone program, either the Swb file or the simulation's case name provides the name of the balance file written by Swbavg to be read in **Swb2fm** in each time step.

Swb2al: memory allocation for Modswb

At the beginning of the simulation, Modflow calls **Swb2al** to read the first record of the Swb input file to define options and allocate memory in Modflow's common x array, conforming to the conventions used in the memory allocation subroutines for Modflow's standard packages. Key memory requirements are provided by the Shed array, defined above. However, the Shed array itself is not defined in terms of the x array, but instead separately in Modmain and in Swat (file Swatmod1.h). In Swat, the Shed array holds hydrogic components as flow rates, converted in subroutine Preswb, which are then either written to a balance file for separate Swat, Swbavg and Modflow execution, or passed by reference to Modflow (entry point Modstp).

Swb2rp: subbasin-aquifer grid connections and annual irrigation water use

At the beginning of the simulation, Modflow calls **Swb2rp**, in which the input file *.Swb is read to define the following:

(a) associate the outflow from each subbasin of a watershed with tributaries that flow into a stream network specified by Modflow's Stream package;

(b) map the geographical extent of the watershed's subbasins onto the gridded aquifer domain specified by Modflow's Basic and Block-Centered Flow packages;

In each stress period (representing a year), subroutine **Srf2rp** summarizes pumping specified by Modflow's Well input for each subbasin, which is used to assign pumping for each time step in Swb2fm.

Swb2fm: specify tributary flow, recharge, potential evaporation, and irrigation

For each aquifer time step (day, month, or year),Modflow (or Modstp from Swat) calls **Swb2fm** to use results from Swat, including ground water recharge, tributary inflows, irrigation demand, and potential evaporation for shallow ground water, to specify conditions for Modflow's solution. For stand-alone execution, Swb1fm obtains Swat's results by reading the aquifer time step and hydrologic results from the balance file into vectors 10-22 of array Shed; otherwise, these results are passed by reference to array Shed in its call to Modflow's entry point Modstp. The following conditions for Modflow are specified in Swb2fm:

Tributary flow (subbasin outflow)

Subbasin outflow for each subbasin, SHED(22), is defined in subroutine Preswb as a sum of surface and subsurface flow. In Modswb it is assigned as lateral inflow to the stream network as STRM (18) in the STREAM package.

Recharge

Groundwater recharge for each subbasin, SHED(25), is defined in Preswb as a sum of percolation from the soil profile, transmission losses along ephemeral streambeds, and pond seepage. For a given subbasin in which the underlying aquifer is a fraction, f_{aqf}, of the subbasin area, the recharge calculated in Preswb is reduced by the fraction, f_{aqf}, and the remaining fraction, $(1-f_{aqf})$, is regarded as a discrepancy contributing to higher soil moisture, subsurface lateral flow, and evaporation than that shown by Swat's results without an underlying aquifer. As an alternative conceptual model, the entire recharge could be assumed to contribute to ground water along lateral flow routes. The former of these alternatives was chosen because the resulting simulations best fit the calibration targets.

Potential evaporation from water table

Maximum water table evaporation rate, array EVTR in the Evt package, is based on potential evaporation, SHED(21), if Modflow option **ievopt** > 0; otherwise, array EVTR is specified by Modflow input to the Evt module.

Irrigation

Irrigation depth assigned by Swat is assigned on the basis of either schedules (in ~.mgt files) or a threshhold on plant water stress (set in ~.mco files); water stress is defined as the fraction of potential plant water evaporation that is available from the soil. An alternative threshold of soil water content was added to Swat's irrigation model and used for the Lower Republican River Basin model. This threshold was determined as a calibration parameter, using annual irrigation water use estimates based on DWR data as the calibration target. Swat's irrigation model was also modified to provide a daily limit, specified in Swat's ~.cod input file.

Groundwater pumping rates for Modflow depend on Modswb option **irropt** as follows. If **irropt** = 0, the pumping rates are specified by Modflow's Well package input file, and are independent of Swat's results. If **irropt** = 1, the pumping rates given by

Modflow's Well input file are scaled in each time step according to the irrigation assigned by Swat in SHED(11).

Swb2bd: summarize evaporation from water table and baseflow

After the aquifer equations have been solved in Modflow, **Swb2bd** is called to summarize results relevant to the watershed model's hydrologic summary as follows.

Evaporation from water table

Evaporation from the water table is summarized in SHED(18) as a depth for each subbasin, is based on array EVAP calculated in the Evt package, and replaces Swat's calculation of "revap". Average depth to water, SHED(23), is also calculated for each subbasin. If Modflow's evaporation (Evt) package is invoked, the maximum evaporation rate is set by array EVTR according to option ievopt (see Swb1fm above); otherwise, (18) and (23) are not calculated, and Swat's version of (18) is retained.

Baseflow

Baseflow from aquifer to stream, SHED(19) as a depth for each subbasin, is based on the negative of streambed leakage from stream to aquifer, given by STRM(11) in the Stream package, and added over all stream reaches within the subbasin.

Discussion of mass balance terms

The above equations indicate that of the total runoff (13) and subsurface flow (15) leaving the soil control volume, only their contributing fraction are shown entering the stream control volume; a balance on ponds might be useful in showing the fate of their noncontributing fraction.

A distinction is drawn between definitions for recharge in Swat and Modflow. Swat's definition of recharge (17) is restricted to soil water that has dropped through both the soil profile and the vadose zone, and is represented by Swat as delayed percolation from the soil profile. This definition is narrower than that used in Preswb to summarize recharge for Modflow, which includes transmission losses, percolation and pond seepage. Similarly, Preswb combines surface runoff and subsurface lateral flow to represent inflow to tributaries.

Mass balance terms affected by the Swat-Modflow linkage are handled through subroutines in the Modswb package as follows.

1. Swat and Modflow are coordinated to represent annual water use for irrigation through input options **ioplim** for Swat (input file ~.cod) and irropt for Modflow (input file ~.swb). If Swat runs without calling Modflow but option ioplim>0, Swat reads a

maximum annual irrigation depth from the ~.cod input file. Then for each aquifer time step, Swat calculates an irrigation depth (11) subject not only to its own scheduling (~.mgt) and plant stress-based (~.mco) options, but also, for ioplim>0, to specified annual use limits. In the case that Swat calls Modflow (iopswt>0) and ioplim>0, Swat obtains annual irrigation limits from Modflow as follows. Swat calls Modflow's entry point Modper at the beginning of each stress period (year), and Modswb subroutine Swb1rp calculates irrigation depth (9) for each subbasin, given by total reported use according to the Well package input file, divided by subbasin area. Swat then retrieves these irrigation depths (subroutine Passflx) to limit annual irrigation as described above.

2. For each aquifer time step, Swb1fm sets up the conditions for Modflow's solution that are based on the hydrologic components calculated by Swat. Irrigation depth (11), subject to limits described above, are treated as follows. If option irropt=0 (specified by Modswb's input file), irrigation pumping is given as read from the Well input file, and Swat's irrigation depth is ignored. Otherwise (irropt>0), the pumping rate of each well within a subbasin is scaled by the ratio of depths given by SHED vectors (11) and (9), so that total groundwater pumping in each subbasin corresponds to that assigned by Swat. The irrigation depth calculated by Swat is part of Swat's hydrologic summary that is obtained by Modflow as described above.

3. Components of Swat's hydrologic summary are combined to represent tributary inflow (22) and groundwater recharge (25) for Modflow's stream-aquifer model as described below, using definitions that are somewhat specific to the Rattlesnake Creek watershed model, and which are invoked by setting option **jkkopt** > 0.

4. Swb1bd calculates the hydrologic components revap(18) and baseflow (19) that depend on Modflow's aquifer head solution. Revap, in Swat's terminology, is based on evapotranspiration from the water table calculated by the Evt package, and baseflow is based on the negative of streambed leakage calculated by the Stream package. For a combined Swat-Modflow run, these components as calculated by Modflow appear in the balance file; otherwise, Swat's version of these components are included, which are based on a lumped model of groundwater with relatively specific assumptions and application (Arnold, 1994).

Swatmain: Swat mainline (file Swatmain.for)

```
c file swatmain.for  spp  7/99 Swat mainline, revised for coupling to Modflow
cspp $debug
      program main
      include 'common.f'
      character*80 titldum, a*1
      character*13 monthin, dayin
c
      parameter (iswbav=0) !indicates that code on swatmod2.h has been included
c             in Swat instead of Swbavg (iswbav=1) for selective execution
c             of file initializing code in Swat.
      include 'swatmod1.h'  !spp: declarations for Swat-Modflow linkage
      boy = 2. * box
      write(*,2231)
2231  format(' SWAT: Soil and Water Assessment Tool for PCs',/,
     3       1x,'Program reading from file.cio ... executing')
      print *,'KGS version of SWAT94.2 with Modflow connection '//
     1        'specified by input file *.cod;'
      print *,'see description of modified input.  --spp Aug 3 95'
CREAD
c       Note: argument added to subr. Open to return std. output file name,
c       which is used to name output files associated with Swat-Modflow
c       linkage; see note on file Swatmod1.h and naming on file Swatmod2.h.
      call open (stdout)
c     TITLE = PROBLEM DESCRIPTION
      write (kw,12200) prog
      write (kw,12300) title

c       Note: use variable incode to define input device no. for Swat's *.cod
c       input file so that code on swatmod2.h can be used in both Swat and
c       Swbavg to read the *.cod file; Swbavg uses std input (dev. 5).
      incode=10
      read (incode,9900) titldum
      print '(a)',' '//titldum
c****format changed for pc swat
c     read (10,10200) nbyr, iyr, lu, ipd, nsim, msim, ign, iwst, isst,
c    *   ires, igraf, irain, itemp, iresq, idaf, idal, iprn, iprp, iopt,
c    *   ipet, ilog
      read(incode,10200) nbyr,iyr,lu,ipd,nsim,msim,ign,iwst,isst,ires,
     1     igraf,irain,itemp,iresq,idaf,idal,iprn,iprp,iopt,ipet
10200 format(20i4)
      print '(20i4)', nbyr,iyr,lu,ipd,nsim,msim,ign,iwst,isst,ires,
     1     igraf,irain,itemp,iresq,idaf,idal,iprn,iprp,iopt,ipet
      iyrbgn = iyr
      iyrend = iyr + nbyr-1
ccc   read(10,10200) ilog  !Swat option (y=1,n=0) to take log10 of monthly outflow for
graphing
c                          (see code on file Swatmod2.h, included below,
c                           for modified line of input data.)
c         !additional data read from *.cod file for Swat-Modflow linkage:
      include 'swatmod2.h'  !spp: *.cod input extension for Swat-Modflow linkage
c     CALCULATE THE NUMBER OF SUBBASINS, REACHES, AND RESERVOIRS
      nrch = 0
      nres = 0
      lu = 0
      lubtot = 0
      do 90 idum = 1, mhyd
        read (9,5200) a, icodes(idum), ihouts(idum), inum1s(idum),
     *       inum2s(idum), inum3s(idum), rnum1s(idum), inum4s(idum)
        if (a.eq.'*') go to 90
        if (icodes(idum).eq.0) go to 100
c
c       icode = 1  subbasin command
c
        if (icodes(idum).eq.1) then
          nbgis(inum1s(idum)) = inum4s(idum)
          nbigs(inum1s(idum)) = inum3s(idum)
          lu = lu + 1
          if (rnum1s(idum).gt.1.e-6) then
             lubtot = lubtot + 1
             nbigis(inum3s(idum)) = inum4s(idum)
          endif
        end if
c
c       icode = 2  route command
c
        if (icodes(idum).eq.2) then
          nrch = nrch + 1
        end if
c
c       icode = 3  route reservoir command
c
        if (icodes(idum).eq.3) then
          nres = nres + 1
        end if
c
c       icode = 7  read in monthly flows in average daily cms
c
```

```
        if (icodes(idum).eq.7) then
          read (9,5201) monthin
 5201     format (10x,6a)
csp       call caps(monthin)
          open (18,file=monthin)
          read (18,9900) titldum
          write (kw,9900) titldum
          read (18,5202) noyrs
          write (kw,5202) noyrs
 5202     format (i6)
          do 5203 iy = 1, noyrs
c***format changed for pc swat
c             read (18,5204) (flomon(inumls(idum),iy,mo),mo = 1,12)
            read(18,5204)(flomon(inumls(idum),iy,mo),mo=1,6)
            read(18,5204)(flomon(inumls(idum),iy,mo),mo=7,12)
5204        format(6e12.4)
            write (kw,5204) (flomon(inumls(idum),iy,mo),mo = 1,12)
 5203     continue
          close (18)
c****format changed for pc swat
c 5204     format (12e12.4)
        end if
c
c       icode = 8  read in EPIC daily epd file
c
        if (icodes(idum).eq.8) then
          dart(ihouts(idum)) = rnumls(idum)
          read (9,5201) dayin
cspp      call caps(dayin)
          open (45+inumls(idum),file=dayin)
          do 5205 ii = 1, 6
            read (45+inumls(idum),9900) titldum
 5205     continue
        end if
c
c       icode = 9  output to eve file for input to another SWAT run
c
        if (icodes(idum).eq.9) then
c         dart(inumls(idum)) = rnumls(idum)
          do 5206 ii = 1, 6
            write (7,9900) titldum
 5206     continue
        end if
c
c       icode = 10  read in daily values with water in cms and rest in tons
c
        if (icodes(idum).eq.10) then
          read (9,5201) dayin
csp       call caps(dayin)
          open (55+inumls(idum),file=dayin)
          do 5207 ii = 1, 6
            read (55+inumls(idum),9900) titldum
 5207     continue
        end if
   90 continue
c*****close unit 9 routin file
  100 close (9)
c****open unit 9 statistical file and save
      open (9,file=statin)
      nbsb = lu
      nrch = lu
      lutot = lu
      do 20 i = 1, lu
        nfert(i) = 1
        npest(i) = 1
        nirr(i) = 1
        bs(i) = 0.
        smq(i) = 0.
        syq(i) = 0.
        sq(i) = 0.
        smsq(i) = 0.
```

```
          sysq(i) = 0.
          ssq(i) = 0.
          potet(i) = 0.
          sym(i) = 0.
          syy(i) = 0.
          sy(i) = 0.
          s1(i) = 0.
          s2(i) = 0.
          irn(i) = 0.
          air(i) = 0.
          sbp(i) = 0.
          ssbp(i) = 0.
          nrf(i) = 0
          tv(i) = 0.
   20   continue
        if (ign.ne.0) then
          call gcycl3(ign,xx)
        end if
        lu1 = lu + 1
        xlu = lu
        flu(lu1) = 1.
        xn = nbyr
        nbmo = nbyr * 12
c           Note: add ffcb to arg list in subr's read and readinpt.--spp jun95
        call read(ign,ffcb)
        if (iresq.eq.1) call readlk
        close (10)
        close (11)
        close (12)
        close (13)
        close (14)
c       close (15)
        read (2,10100) nrgage, ntgage
        read (2,10000) (rfile(j),j = 1,18)
        read (2,10000) (tfile(j),j = 1,18)
        read (2,10000) (resvo(j),j = 1,6)
10000   format (6a)
10100   format (2i4)
        do 31 j = 1, 18
csp       call caps(rfile(j))
csp       call caps(tfile(j))
csp       call caps(resvo(j))
   31   continue
c add ffcb to arg lists of subr's read (above) and readinpt: --spp jun95
        call readinpt(ffcb)
        if (iprn.eq.0) then
          write (kw,10300) sww
          write (6,8500)
          write (6,8600)
          do 30 j = 1, lu
            write (6,8700) j, cla(1,j), sil(j), san(j), ek(j), wn(1,j),
     *          wpo(1,j), ap(1,j)
   30     continue
          write (6,8800)
          write (6,8900)
          do 40 j = 1, lu
            write (6,9000) j, (psz(i,j),i = 1,5)
   40     continue
        end if
        fru = frd * da9
        fpu = fpd * da9
        do 60 i = 1, lu
          xx = 2.65 * (1.-por(1,i)) / 10.
          if (wn(1,i).le.0.) wn(1,i) = 1000. * cbn(1,i) * xx
          if (wpo(1,i).le.0.) wpo(1,i) = .125 * wn(1,i) * xx
          if (ap(1,i).le.0.) ap(1,i) = 30. * xx
   60   continue
        do 70 k = 1, nns
          sm(k) = 0.
          smy(k) = 0.
          smm(k) = 0.
   70 continue
      je = 1
      mo1 = 1
      v1 = 100.
      vb = 0.
      rz1 = 1000.
      ncn = 0
      immo = 0
      cn = cn1
      iw = 1
      v1 = randn(k2)
      v3 = randn(k3)
      v5 = randn(k4)
      v7 = randn(k5)
C     *compute channel routing coefficients*
      do 80 j = 1, lu
        if(ch1(2,j).gt.0.) call ttcoef(j)
   80 continue
      do 110 j = 1, nres
        if (resvo(j).ne.'              ') then
          open (9+j,file=resvo(j))
          call readres(j)
        end if
  110 continue
      call open3 (iopwf1)
c     write (4,5600)
C********WRITE HEADING TO REACH/RES/SUBBASIN OUTPUT split to three files nbs
c     unit 3 = subbasin output file
      write (3,12200) prog
      write (3,12300) title
c****write heading to .sbs file unit=3
      write (3,6501)
 6501 format (//6x,'SEG GIS BIG MGT MON ',' AREA KM2 ',
     *     '  PREC MM ','   SNOF MM',
     *     '   SNOM MM','   SURQ MM','   LATQ MM','   GW Q MM',
     *     '    REV MM',' GWPER MM','   GWRE MM','   WYLD MM',
     *     '   PERC MM','     ET MM','  TLOSS MM','   SYLD MM',
     *     '   PEVP MM','   PSEP MM','   PARI MM','   PINF MM',
     *     '  PSINF MM','   POUT MM','  PSOUT MM','    IRR MM',
     *     '   SAQU MM','   DAQU MM','  ETPOT MM','    SGW MM ',
     *     '   DGW MM  ','   WSTRS   ','TMPSTRS ','    NSTRS  ',
     *     ' PSTR MM','   ORGN KH','   PYLD KH','  NSURQ KH',
     *     '  NSUBQ KH','   SOLP KH','    NUP KH','   NO3L KH',
     *     '    PUP KG','   LABP KH','   STAP KH','   NAPP KH',
     *     '   PAPP KH','   NFIX KH','   DENT KH','  HMAON KH',
     *     '   ASON KH','  HMAOP KG','  MFRSN KH','  MFRSP KH',
     *     '     SW MM  ',' BIOM KH ','    LAI KH''   YLD KH')
c     unit 77 = big subbasin output file
      write (77,12200) prog
      write (77,12300) title
c****write heading to .bsb file unit=77
      write (77,6502)
 6502 format (//8x,'SUB GIS   MON ',' AREA KM2 ',
     *     '  PREC MM ','   SNOM MM',
     *     '   SURQ MM','   WYLD MM','  ETPOT MM','     ET MM',
     *     '   SYLD KH','   ORGN KH','   PYLD KH','   SNO3 KH',
     *     '   LABP KH','   GWQ  MM','   SW   MM','   LAI    ',
     *     '   BIOM TH','   YLD  TH')
c     unit 8 = reach output file
      write (8,12200) prog
      write (8,12300) title
      write (8,12301)
* spp fix 3rd line of the following format stmt: add a comma after t46.
12301 format (//7x,'BIG GIS MON',t21,'FLOW IN cms',
     *     t32,'FLOW OUT cms',
     *     t46,'SED IN t',t58,'SED OUT t',t70,'SED CON ppm',t82,
     *     'ORGN IN kg',t94,'ORGN OUT kg',t106,'ORGP IN kg',t118,
     *     'ORGP OUT kg',t130,'EVAP cms',t142,'TLOSS cms',t153,
     *     'PSTREAM cms',t166,'DIVER cms',t177,'RET FLOW cms',t166,
     *     'NIT IN kg',t177,'NIT OUT kg',t190,'SOLP IN kg',t202,
     *     'SOLP OUT kg')
```

```
c     unit 1 = reservoir output file
      write (1,12200) prog
      write (1,12300) title
C     BEGIN RUNOFF SIMULATION.  NBYRS = NO YRS OF SIMULATION
      do 540 iy = 1, nbyr
      icalyr = iyrbgn + iy-1
      if (MOD(icalyr,4).eq.0) then
        idleap=2
      else
        idleap=1
      end if
      iyr = icalyr
      include 'swatmod3.h' !if iopswt=1, call Modflow; initialize balance file

        do 541 jj = 1, lu
          npest(jj) = 1
          nfert(jj) = 1
          nirr(jj) = 0
          gplt(jj) = 0.
          yldanu(jj) = 0.
          dmanu(jj) = 0.
          xlai(jj) = 0.
*
          irnyr(jj)=0           !init. no. irrig. applications --spp
  541   continue
        do 542 jj = 1, lubtot
          wadm(jj) = 0.
          walai(jj) = 0.
          wayld(jj) = 0.
  542   continue
        jj = 0
        if (iw.eq.iy) then
          write (kw,12200) prog
          write (kw,12300) title
          write (kw,10400)
C********WRITE HEADING TO PESTICIDE OUTPUT
          if (iprp.ne.0) write (5,12200) prog
          if (iprp.ne.0) write (5,12300) title
          if (iprp.ne.0) write (5,5100)
          iw = iw + 3
        end if
        mo = 1
        nt = 1                          ! assume non-leap year
        if (mod(iyr,4).eq.0) nt = 0     ! leap year
        ida = 1  !the NEXT day; used to find corresp. month in subr xmon23
        nd = 366 - nt
        id1 = 1
        if (iy.eq.1.and.idaf.gt.0) id1 = idaf
        if (iy.eq.nbyr.and.idal.gt.0) nd = idal
        ndyr=nd
       call TIME (chtime)
       write (*,2232) iy,iyr,chtime // ' days ',id1,nd
2232   format(2(2i5,1x,a),3(i5,1x,a))
C       DAILY LOOP
        do 530 i = id1, ndyr
          julday = i
          yrdays = FLOAT(icalyr) + FLOAT(julday)/FLOAT(366-nt) - 1900
          numday = numday + 1  !Reset to zero every time Modflow is called.

          call clicon (iopwf1,fmttmp,fmtpcp,fmtrad,iorad,
     1                    itsim,ipsim,julday,icalyr)

          ndmo(mo) = ndmo(mo) + 1
          dtot = dtot + 1.
          bst = 0.
          f1 = 0.
          ssb(8) = 0.
          ssb(9) = 0.
          ssb(10) = ra(j)
          snob = 0.
          snoev = 0.
          pr = 0.
          ipr = 0
          do 130 j = 1, lu
            do 120 k = 1, nptot(j)
              zq(k,j) = 0.
              zl(k,j) = 0.
  120       continue
  130     continue
          do 121 j = 1, lubtot
            wsw(j) = 0.
            wdm(j) = 0.
            wlai(j) = 0.
            wyld(j) = 0.
  121     continue
c         $$$$$$ NEW ROUTING STRUCTURE $$$$$$

          do 200 idum = 1, mhyd
            icode = icodes(idum)
            ihout = ihouts(idum)
            inum1 = inum1s(idum)
            inum2 = inum2s(idum)
            inum3 = inum3s(idum)
            rnum1 = rnum1s(idum)
            inum4 = inum4s(idum)
            if (icode.le.0) go to 210
            go to (140,150,160,170,180,190,191,192,193,194), icode
cc140       call subbasin(icode,ihout,inum1,inum2,inum3,rnum1) -orig. call--spp

c revised call to subr subbasin (see swatmod1.h for argument list definition):

140     call subbasin (icode,ihout,inum1,inum2,inum3,rnum1,
     1      julday,iaqufr,ioirr,iprirr,ioevt,iprevt,
     1      iopswm,wsfmax,swminf,evtgw(inum1),egwuna(inum1))

            go to 200

  150       call route(icode,ihout,inum1,inum2,inum3,rnum1,nrch)
            go to 200
  160       call routres(icode,ihout,inum1,inum2,inum3)
            go to 200
  170       ihout = ihouts(idum-1)
            icodep = icodes(idum-1)
            call transfer(icode,ihout,inum1,inum2,inum3,rnum1,inum4,
     *          icodep)
            go to 200
  180       call add(icode,ihout,inum1,inum2,inum3)
            go to 200
  190       call routsub(icode,ihout,inum1,inum2,inum3)
            go to 200
  191       call recmon(icode,ihout,inum1,inum2,inum3,rnum1)
            go to 200
  192       call recepic(icode,ihout,inum1,inum2,inum3,rnum1)
            go to 200
  194       call recday(icode,ihout,inum1,inum2,inum3,rnum1)
            go to 200
  193       call save(icode,ihout,inum1,inum2,inum3,rnum1)
  200     continue
  210     ssb(1) = rmn
          if (fl.gt.0.) then
            ncn = ncn + 1
            ssb(2) = ssb(2) / fl
          end if
          wy = ssb(3) + ssb(4) + ssb(104) - ssb(33) - ssb(38) + 1.e-20
c         if (ires.eq.1) wy = ssb(28) / af + 1.e-20
          ssb(6) = wy
          ssb(12) = ssb(12) / da9
          ssb(11) = ssb(11) / da9
          do 220 k = 1, nns
            smm(k) = smm(k) + ssb(k)
  220     continue
c         WRITE (13,345)julday,lul,SSB(1),SSB(3),SSB(4),SSB(6),SSB(5),
c         1     SSB(7),SSB(12),SSB(35),SSB(40),SSB(41),SSB(42),SSB(43),
c         2     SSB(44),SSB(45),SSB(46),(SDA(K+30,LU1),K=1,10),VRMM,
c         3     SSB(104)
c         call range2(j)
          if (ipd.eq.1) then
            ii = icl(julday)
c           DAILY WRITE
            write (kw,10500) julday, ssb(1), ssb(3), ssb(4), ssb(6),
     *          ssb(5), ssb(7), ssb(12), ssb(35), ssb(40), ssb(41),
     *          ssb(42), ssb(43), ssb(44), ssb(45), ssb(46), ssb(104)
            do 230 j = 1, lu
              sum = 0.
              do 232 k = 1, 20
                sum = sum + sda(k,j)
  232         continue
              if (sum.gt.0.) then
                if (iprp.ne.0) write (5,5300) j, ' SOL', (sda(k,j),k =
     *              1,10)
                if (iprp.ne.0) write (5,5300) j, ' SOR', (sda(k,j),k =
     *              11,20)
              end if
  230       continue
            sum = 0.
            do 231 k = 1, 10
              kk = k + 46
              jj = k + 73
              sum = sum + ssb(kk)
              sum = sum + ssb(jj)
  231       continue
            if (sum.gt.0. .and. iprp.ne.0) then
                write (5,5400) julday,' SOL',(ssb(k),k = 47,56)
                write (5,5500) julday,' SOR', (ssb(k),k = 74,83)
            end if
          end if
c         write (4,5700) julday, cpest1, cpest2, ssb(62), ssb(63),
c    *        ssb(64), ssb(65), ssb(66), ssb(67), ssb(68), ssb(69),
c    *        ssb(70), phosl, ssb(71), ssb(72), ssb(73), chla, secci,
c    *        tlake

          include 'daily.h'  !summarize daily results to be passed to Modflow
          do 250 k = 1, 20
            do 240 j = 1, lu
              smo(k,j) = smo(k,j) + sda(k,j)
              sda(k,j) = 0.
  240       continue
  250     continue
c           mo = MONTH correspondin to NEXT DAY, ida:
          ida = i + 1     !next day
          call xmon23
ccc    write (*,2232) iy,iyr,chtime // ' days ',id1,ndyr,
ccc   1       'day',i,'mo=',mo,'mo1=',mo1
          if (mo.ne.mo1) then
            xx = julday - jj
            immo = immo + 1
            nd = FLOAT(nc(mo1+1)-nc(mo1))
c           if (iwst.ne.0) wpd(immo) = smm(6)
            if (iwst.ne.0) wpd(immo) = srch(2,iopt) / nd
            if (isst.ne.0) spd(immo) = smm(12)
            xcn = ncn
            smm(2) = smm(2) / (xcn+1.e-20)
            if (smm(2).eq.0.) smm(2) = cn
            smm(8) = smm(8) / xx
            smm(9) = smm(9) / xx
            smm(10) = smm(10) / xx
            smm(32) = smm(6) * smm(6)
            jj = julday
            ncn = 0
            if (ipd.ne.2) then
              do 260 jk = 1, lu
                if (nfull(jk).gt.0.) then
                  write (kw,5900) nfull(jk), jk
                nfull(jk) = 0.
              end if
              if (nempty(jk).gt.0.) then
                write (kw,6000) nempty(jk), jk
                nempty(jk) = 0.
              end if
  260       continue
c           MONTHLY WRITE
            write (kw,10500) mo1, smm(1), smm(3), smm(4), smm(6),
     *          smm(5), smm(7), smm(12), ssb(35), smm(40), smm(41),
     *          smm(42), smm(43), smm(44), smm(45), smm(46), vrmm,
     *          smm(104), smm(108), smm(108)
            if (ipd.eq.0) then
            do 280 j = 1, lutot
              dmt = dm(j) / 1000.
              yldt = (1.-rwt(j)) * dmt * hia(j)
              write (3,6500)j,nbgis(j),nbigs(j),nmgt(j),mo1,
     *           dx(j),(ssub(k,j),k = 1,27)
     *           ,(ssub(k,j),k = 31,53), sw(j), dmt, alai(j), yldt
c    *            , (ssub(k,j),k = 54,60)
  280       continue
            do 281 j = 1, lubtot
              write (77,6503)j,nbigis(j),
     *            mo1,dx(j),(sbsub(k,j),k = 1,
     *            nvbsub), wsw(j), wlai(j), wdm(j), wyld(j)
  281       continue
            do 290 j = 1, nrch
c             ***convert m3 to cms
              nd = FLOAT(nc(mo1+1)-nc(mo1))
              srch(1,j) = srch(1,j) / nd
              srch(2,j) = srch(2,j) / nd
c             take log10 of monthly outflow for graphing
              if (ilog.gt.0) then
                if (srch(2,j).gt.1.) then
                  srch(2,j) = alog10(srch(2,j))
                else
                  srch(2,j) = 0.
                end if
              end if
              srch(10,j) = srch(10,j) / nd
              srch(11,j) = srch(11,j) / nd
              srch(12,j) = srch(12,j) / nd
              srch(13,j) = srch(13,j) / nd
              srch(14,j) = srch(14,j) / nd
c           if (ipd.eq.0) then
c             if (srch(1,j).gt.1.e-6) then
              xx = rnum1s(j) + FLOAT(inum3s(j))
              if(rnum1s(j).gt.0.0 .or. xx.lt.1.e-3) then
              write (8,6300) nbigs(j), nbgis(j), mo1,
     *            (srch(k,j),k = 1,12),
     *            (srch(k,j),k = 14,18)
              endif
  290       continue
            endif

            swmo(mo1) = swmo(mo1) + ssb(35)
            vrmo(mo1) = vrmo(mo1) + vr(1)
            do 300 j = 1, nres
              nd = FLOAT(nc(mo1+1)-nc(mo1))
              sres(1,j) = sres(1,j) / nd
              sres(2,j) = sres(2,j) / nd
c             convert reservoir volume m^3 to ha-m
              vrham = vr(j) / 10000.
              write (1,6400) j, j, mo1, vr(j), (sres(k,j),k = 1,5)
  300       continue
          end if
ccc    write (*,2232) iy,iyr,chtime // ' days ',id1,ndyr,
ccc    1      'day',i,'mo=',mo,'mo1=',mo1,'at monthly.h'
          include 'monthly.h' !summarize monthly results for Modflow
c         CALCULATING MONTHLY AVERAGES
          amo(mo1,1) = amo(mo1,1) + smm(1)
```

```
            amo(mo1,2) = amo(mo1,2) + smm(39)
            amo(mo1,3) = amo(mo1,3) + smm(3)
            amo(mo1,4) = amo(mo1,4) + smm(4)
            amo(mo1,5) = amo(mo1,5) + smm(6)
            amo(mo1,6) = amo(mo1,6) + smm(7)
            amo(mo1,7) = amo(mo1,7) + smm(12)
            amo(mo1,8) = amo(mo1,8) + smm(108)
            do 310 k = 1, nns
              smy(k) = smy(k) + smm(k)
              smm(k) = 0.
  310       continue
            do 330 k = 1, nvsub
              do 320 j = 1, lutot
                sysub(k,j) = sysub(k,j) + ssub(k,j)
                ssub(k,j) = 0.
  320         continue
  330       continue
            do 331 k = 1, nvbsub
              do 321 j = 1, lutot
                sybsub(k,j) = sybsub(k,j) + sbsub(k,j)
                sbsub(k,j) = 0.
  321         continue
  331       continue
            do 350 k = 1, nvrch
              do 340 j = 1, nrch
                syrch(k,j) = syrch(k,j) + srch(k,j)
                srch(k,j) = 0.
  340         continue
  350       continue
            do 370 k = 1, nvres
              do 360 j = 1, nres
                syres(k,j) = syres(k,j) + sres(k,j)
                sres(k,j) = 0.
  360         continue
  370       continue
            do 390 k = 1, 20
              do 380 j = 1, lu
                syr(k,j) = syr(k,j) + smo(k,j)
                smo(k,j) = 0.
  380         continue
  390       continue
            do 400 k = 1, lu
              syq(k) = syq(k) + smq(k)
              sysq(k) = sysq(k) + smsq(k)
              syy(k) = syy(k) + sym(k)
              smq(k) = 0.
              smsq(k) = 0.
              sym(k) = 0.
  400       continue
            if (mo.le.mo1) then
              smy(2) = smy(2) / 12.
              smy(9) = smy(9) / 12.
              smy(10) = smy(10) / 12.
              smy(8) = smy(8) / 12.
ccc           TAV=(SMY(8)+SMY(9))/2.
ccc           print '(3(1x,a,f8.1))', 'smy(8)=',smy(8),'smy(9)=',smy(9),
ccc  1            'tav=',tav
c             ANNUAL WRITE
              write (kw,10600) iyr, smy(1), smy(3), smy(4), smy(6),
     *            smy(5), smy(7), smy(12), ssb(35), smy(40), smy(41),
     *            smy(42), smy(43), smy(44), smy(45), smy(46), vrmm,
     *            smy(104), smy(108)
C*******SSB,SMM,SMY(47-56) = DISSOLVED PORTION FOR PESTICIDES 1-10
C*******SSB,SMM,SMY(74,83) = SORBED PORTION FOR PESTICIDES 1-10
C*******LEACHED  84-93 FOR PESTICIDES 1-10
c             sum = 0.
c             do 235 j = 1, lu
c             do 236 k = 1, 20
c             sum = sum + smo(k,j)
c             236          continue
c             235        continue
```

```
c             if (sum.gt.0.) then
c             do 410 j = 1, lu
c             if (iprp.ne.0) write (5,5300) j, ' SOL', (syr(k,j),k =
c             *              1,10)
c             if (iprp.ne.0) write (5,5300) j, ' SOR', (syr(k,j),k =
c             *              11,20)
c             410        continue
c             if (iprp.ne.0) write (5,5400) iyr, ' SOL', (smy(k),k = 47,
c             *            56)
c             if (iprp.ne.0) write (5,5500) iyr, ' SOR', (smy(k),k = 74,
c             *            83)
c             endif
c             write (4,5800) iyr, cpest1, cpest2, smy(62), smy(63),
c     *           smy(64), smy(65), smy(66), smy(67), smy(68), smy(69),
c     *           smy(70), phos1, smy(71), smy(72), smy(73), chla,
c     *           secci, tlake
ccc    write (*,2232) iy,iyr,chtime // ' days ',id1,ndyr,
ccc   1      'day',i,'mo=',mo,'mo1=',mo1,'at annual.h'
              include 'annual.h' !summarize annual results for Modflow
              if (ipd.eq.0.or.ipd.eq.2) then
              do 420 j = 1, lutot
                write (3,6500)j,nbgis(j),nbigs(j),nmgt(j),iyr,
      *             dx(j),(sysub(k,j),k= 1,27)
      *             ,(sysub(k,j),k = 31,53), sw(j), dmanu(j), xlai(j),
      *             yldanu(j)
c     *             yldanu(j), (sysub(k,j),k = 54,60)
                dmaa(j) = dmaa(j) + dmanu(j)
                xlaia(j) = xlaia(j) + xlai(j)
                yldaa(j) = yldaa(j) + yldanu(j)
  420         continue
              do 421 j = 1, lubtot
                write (77,6503)j,nbigis(j),
      *             iyr,dx(j),(sybsub(k,j),k=1,
      *             nvbsub), wsw(j), walai(j), wadm(j), wayld(j)
                waadm(j) = waadm(j) + wadm(j)
                waalai(j) = waalai(j) + walai(j)
                waayld(j) = waayld(j) + wayld(j)
  421         continue
              do 430 j = 1, nrch
c**convert from m3 to cms
                syrch(1,j) = syrch(1,j) / 12.
                syrch(2,j) = syrch(2,j) / 12.
                syrch(10,j) = syrch(10,j) / 12.
                syrch(11,j) = syrch(11,j) / 12.
                syrch(12,j) = syrch(12,j) / 12.
                syrch(13,j) = syrch(13,j) / 12.
                syrch(14,j) = syrch(14,j) / 12.
c               if (syrch(1,j).gt.1.e-6) then
                  xx = rnum1s(j) + FLOAT(inum3s(j))
                  if(rnum1s(j).gt.0.0 .or. xx.lt.1.e-3) then
                  write (8,6300) nbigs(j), nbgis(j), iyr,
      *               (syrch(k,j),k = 1,12),
      *               (syrch(k,j),k = 14,18)
                  endif
  430         continue
              do 440 j = 1, nres
                write (1,6400) j, j, iyr, (syres(k,j),k = 1,nvres)
  440         continue
              endif

              do 450 k = 1, lu
                sq(k) = sq(k) + syq(k)
                ssq(k) = ssq(k) + sysq(k)
                sy(k) = sy(k) + syy(k)
                syq(k) = 0.
                sysq(k) = 0.
                syy(k) = 0.
  450         continue
              do 460 k = 1, nns
                sm(k) = sm(k) + smy(k)
                smy(k) = 0.
  460         continue
              do 500 k = 1, nvsub
                do 470 j = 1, lutot
                  stsub(k,j) = stsub(k,j) + sysub(k,j)
                  sysub(k,j) = 0.
  470           continue
  500         continue
              do 473 k = 1, nvbsub
                do 472 j = 1, lutot
                  stbsub(k,j) = stbsub(k,j) + sybsub(k,j)
                  sybsub(k,j) = 0.
  472           continue
  473         continue
              do 501 k = 1, nvrch
                do 480 j = 1, nrch
                  strch(k,j) = strch(k,j) + syrch(k,j)
                  syrch(k,j) = 0.
  480           continue
                do 490 j = 1, nres
                  stres(k,j) = stres(k,j) + syres(k,j)
                  syres(k,j) = 0.
  490           continue
  501         continue
              do 520 k = 1, 20
                do 510 j = 1, lu
                  stot(k,j) = stot(k,j) + syr(k,j)
                  syr(k,j) = 0.
  510           continue
  520         continue
              iyr = iyr + 1
            end if
          end if
          mo1 = mo
  530   continue
c******NANCY SWAT
        do 531 isb = 1, lu
          iix = nro(isb)
          iiz = icr(isb)
          nro(isb) = nro(isb) + 1
          if (nro(isb).gt.nrot(isb)) then
            nro(isb) = 1
          end if
          icr(isb) = 1
          nop(isb) = 1
       if(igro(isb).eq.1) then
          g(isb) = 0.
          hufh(isb) = 0.
          hia(isb) = 0.
          phu(nro(isb),icr(isb),isb) = phu(iix,iiz,isb)
          ncr(nro(isb),icr(isb),isb) = ncr(iix,iiz,isb)
          hitar(nro(isb),icr(isb),isb) = hitar(iix,iiz,isb)
          ncrops(iix,iiz,isb) = ncrops(iix,iiz,isb) + 1
       endif
  531   continue
c******NANCY SWAT
  540 continue
      dfpq = 0.
      dfpy = 0.
      dfrq = 0.
      dfry = 0.
      tir = 0.
      do 550 j = 1, lu
        if (irr(j).ne.0.) then
          tir = tir + air(j) * flu(j)
        end if
  550 continue
      sww = sww + snobi - snob
      if (nsim.ge.3) then
        sm(37) = 0.
      end if
C     CHECKING FINAL SOIL WATER BALANCE
```

```
      call swb120(sm(1),sm(3),sm(7),sm(4),sm(5),ssb(35),sm(38),tir)
      if (fpu.ne.0.) then
c       CHECKING POND WATER & SEDIMENT BALANCE
        spflow = sm(20)*da*1.e3 !pond seepage: volume (mpy by basin area)
ccccc   call vb121(sm(19),sm(20),sm(21),sm(22),sm(23),vpw,vpy,sm(13),
        call vb121(sm(19),spflow,sm(21),sm(22),sm(23),vpw,vpy,sm(13),
     *       sm(14),sm(15),v,cs,fpu,nbsb)
        dfpq = vpw
        dfpy = vpy
      end if
      if (fru.ne.0.) then
c       CHECKING RESERVOIR WATER & SEDIMENT BALANCE
        call vb121(sm(24),sm(25),sm(26),sm(27),sm(28),vrw,vry,sm(16),
     *       sm(17),sm(18),vr,csr,fru,nbsb)
        dfrq = vrw
        dfry = vry
      end if
      sv = 0.
      if (idaf+idal.gt.0) then
        xn = dtot / 365.
      else
        xn = nbyr
      end if
      xda = xn * da9
      do 560 j = 1, lu
        xis = nrf(j)
        ssbp(j) = ssbp(j) - sbp(j) * sbp(j) / (xis+.01)
        sbp(j) = sbp(j) / xn
        ssbp(j) = sqrt(ssbp(j)/(xis-1.))
        sy(j) = sy(j) / (flu(j)*xda)
        sq(j) = sq(j) / xn
        ssq(j) = ssq(j) / xn
        potet(j) = potet(j) / xn
        if (fp(j).ne.0.) then
          sv = sv + v(j)
          xx = flu(j) * fp(j)
          v(j) = v(j) / (xx*af)
          cs(j) = cs(j) * 1.e6
        end if
  560 continue
      tot = 0.
      do 570 j = 1, nres
        tot = tot + vr(j)
        xz = 1.
        if (ires.eq.1) xz = 1.
        vr(j) = vr(j) / (xz*af)
        csr(j) = csr(j) * 1.e6
  570 continue
      if (fpu.ne.0.) then
        zf = fpu * 10.
        sv = sv / zf
      end if
      if (fru.ne.0.) then
        fz = fru * 10.
        tot = tot / fz
      end if
      write (kw,12200) prog
      write (kw,12300) title
      write (kw,9500)
      write (kw,11300)

c     WRITE AVERAGE ANNUAL VALUES TO REACH AND SUBBASIN FILES
      if (ipd.eq.0.or.ipd.eq.2) then
      do 504 k = 1, nvsub
        do 505 j = 1, lutot
          stsub(k,j) = stsub(k,j) / xn
 505    continue
 504  continue
      do 507 j = 1, lutot
        dmaa(j) = dmaa(j) / xn
        xlaia(j) = xlaia(j) / xn
```

```
          yldaa(j) = yldaa(j) / xn
          write (3,6500) j,nbgis(j),nbigs(j),nmgt(j), nbyr,
     *       dx(j),(stsub(k,j),k = 1,27),
     *       (stsub(k,j),k = 31,53), sw(j), dmaa(j), xlaia(j), yldaa(j)
c    *        , (stsub(k,j),k = 54,60)
  507 continue

        do 502 k = 1, nvrch
          do 503 j = 1, nrch
            strch(k,j) = strch(k,j) / xn
  503     continue
  502 continue
        do 506 j = 1, nrch
c             if (strch(1,j).gt.1.e-6) then
          xx = rnum1s(j) + FLOAT(inum3s(j))
          if(rnum1s(j).gt.0.0 .or. xx.lt.1.e-3) then
             write (8,6300) nbigs(j), nbgis(j), nbyr,
     *       (strch(k,j),k = 1,12),
     *       (strch(k,j),k = 14,18)
           endif
  506 continue

        do 511 k = 1, nvbsub
          do 509 j = 1, lubtot
            stbsub(k,j) = stbsub(k,j) / xn
  509     continue
  511 continue
        do 508 j = 1, lubtot
          waalai(j) = waalai(j) / xn
          waayld(j) = waayld(j) / xn
          waadm(j) = waadm(j) / xn
          write(77,6503)j,nbigis(j),nbyr,
     *                  dx(j),(stbsub(k,j),k=1,nvbsub)
     *                  ,wsw(j), waalai(j), waadm(j), waayld(j)
  508 continue
        endif


        if (iprn.eq.0) then
          nx = 0
          do 580 j = 1, lu
            if (ns(j).gt.nx) nx = ns(j)
  580     continue
          write (kw,9600) (sid(i),i = 1,nx)
          do 590 j = 1, lu
            nn = ns(j)
C           WRITE FINAL SOIL WATER STORAGE
            write (kw,11000) j, (st(i,j),i = 1,nn)
  590     continue
          write (6,9100)
          do 600 j = 1, lu
            write (kw,9200) j, sw(j)
            irn(j) = irn(j) / xn
            air(j) = air(j) / xn
  600     continue
          write (kw,11400) ssb(35)
          write (kw,11500) rzl
          write (kw,9700)
          do 610 i = 1, lu
C           WRITE FINAL POND AND RESERVOIR STORAGE
            write (kw,7900) i, v(i), cs(i), vr(i), csr(i)
  610     continue
          write (kw,11600) sv
          write (kw,11700) tot
          write (6,9300)
          do 620 j = 1, lu
C           WRITE IRRIGATION OUTPUT
            write (kw,9400) j, irn(j), air(j)
  620     continue
        end if
        write (kw,12000) sww
```

```
      write (kw,10700)
      write (kw,12100) dfpq, dfpy
      write (kw,10800)
      write (kw,12100) dfrq, dfry
      xwy = nbmo
      sm(32) = sm(32) - sm(6) * sm(6) / xwy
      sdv = sqrt(sm(32)/(xwy-1.))
      tw = sm(6) / xwy
      do 630 k = 1, 18
        sm(k) = sm(k) / xn
  630 continue
      do 640 i = 36, nns
        sm(i) = sm(i) / xn
  640 continue
c     do 660 k = 1, nptot(j)
      do 660 k = 1, 10
        pdb(k) = 0.
        do 650 j = 1, lu
          pdb(k) = pdb(k) + (dkg(k,j)+dkf(k,j)) * flu(j)
  650   continue
        pdb(k) = pdb(k) / xn
        pab(k) = pab(k) / xn
  660 continue
      atp = sm(15) / da9
      atr = sm(18) / da9
      pl = sm(33) / xn
      tl = sm(34) / xn
      if (fpu.ne.0.) then
        do 670 k = 19, 23
          sm(k) = sm(k)/xn       !flux w.r.t. basin area (component in ~.bal)
          smpond(k) = sm(k)/fpd  !flux w.r.t. pond area (printed in ~.std)
  670   continue
        sm(13) = sm(13) / fpu
        sm(14) = sm(14) / fpu
      end if
      if (fru.ne.0.) then
        zxx = fz * xn
        do 680 k = 24, 28
          sm(k) = sm(k) / zxx
  680   continue
        if (ires.eq.1) fru = da9
        sm(16) = sm(16) / fru
        sm(17) = sm(17) / fru
      end if
c*****write average crop information to std output file
      write (kw,12200) prog
      write (kw,12300) title
      write (kw,13000)
13000 format (35x,'Average Crop Values',/,20x,'Crop 1',15x,'Crop 2',15x,
     *    'Crop 3'/,14x,'Yld',7x,'Biomass',4x,'Yld',7x,'Biomass',4x,
     *    'Yld',7x,'Biomass',/,13x,'(t/ha)',5x,'(t/ha)',5x,'(t/ha)',5x,
     *    '(t/ha)',5x,'(t/ha)',5x,'(t/ha)'/)
      do 13001 j = 1, lu
        write (kw,13003) j
13003   format (2x,'Sub',i4,/)
        do 13001 nnro = 1, nrot(j)
          do 13002 nicr = 1, 3
c****yldn = average value for yld
c****dm2n = average value for biomass
            yldn(nnro,nicr,j) = yld(nnro,nicr,j) / (
     *          ncrops(nnro,nicr,j)+1.e-6)
            dm2n(nnro,nicr,j) = dm2(nnro,nicr,j) / (
     *          ncrops(nnro,nicr,j)+1.e-6)
13002     continue
          write (kw,13004) nnro, (yldn(nnro,nicr,j),dm2n(nnro,nicr,j),
     *        nicr = 1,3)
13004     format (1x,'Rotation',1x,i3,6(f8.1,2x))
13001 continue
      write (kw,10900)
      do 700 k = 1, 20
        do 690 j = 1, lu
```

```
          stot(k,j) = stot(k,j) / xn
  690   continue
  700 continue
      do 720 k = 1, 20
        do 710 j = 1, lu
          aabv(k) = aabv(k) + stot(k,j) * flu(j)
  710   continue
  720 continue
c     if (iprp.ne.0) write (5,6600)
      do 740 j = 1, lu
C       WRITE AVERAGE ANNUAL SUBBASIN DATA
        write (kw,12400) j, sbp(j), sq(j), ssq(j), sy(j), potet(j)
c     if (iprp.ne.0) write (5,6700) j, ' SOL', (stot(k,j),k = 1,10)
c     if (iprp.ne.0) write (5,6700) j, ' SOR', (stot(k,j),k = 11,20)
  740 continue
c     if (iprp.ne.0) write (5,6800)
c     if (iprp.ne.0) write (5,6900) ' SOL', (aabv(k),k = 1,10)
c     if (iprp.ne.0) write (5,6900) ' SOR', (aabv(k),k = 11,20)
      write (kw,8000)
      dmba = dmba / xn
      do 760 j = 1, 12
        xmm = FLOAT(ndmo(j)) / FLOAT(nc(j+1)-nc(j))
        if (xmm.le.0.) go to 770
        swmo(j) = swmo(j) / xn
        vrmo(j) = vrmo(j) / xn
        do 750 k = 1, 7
          amo(j,k) = amo(j,k) / xmm
  750   continue
        write (kw,7800) j, (amo(j,k),k = 1,8)
  760 continue
  770 if (iprn.eq.0) then
        write (kw,9800)
        write (kw,11800) sm(2), vb, vl
C       WRITE MEAN AND ST DV OF PRED DAILY WATER YLD.
        write (kw,11900) tw, sdv
      end if
C     WRITE STRESS DAYS
      sdw = sdw / xn
      sdt = sdt / xn
      sdn = sdn / xn
      sdp = sdp / xn
c          summarize avg annual results for Modflow:
      include 'avgann.h'    !option (iopmod=0)
      write (kw,7000) sdw, sdt, sdn, sdp
      write (kw,12200) prog
      write (kw,12300) title
C     WRITE AVERAGE ANNUAL BASIN VALUES
      data ioptrc/1/
      if (ioptrc.eq.0) then
          write (kw,11100) sm(1), sm(39), sm(36), sm(3), sm(4), sm(104),
     *        sm(105), sm(106), sm(107), sm(6), sm(5), sm(7), sm(109),
     *        sm(38), sm(11), sm(12), sm(19), sm(20), sm(21), sm(22),
     *        sm(13), sm(23), sm(14), (sm(k),k = 24,27),
     *        sm(16), sm(28), sm(17)
      else
          write (kw,'(t10,a,/)') 'AVE ANNUAL BASIN VALUES'
11101     format (t15,a,f8.1,a)
11102     format (20(t15,a,f8.2,a/))
11103     format (20(t20,a,f8.3,a/))
11104     format (20(t25,a,f8.3,a/))
11106     format (t20,a,f8.5)
          write (kw,11102)
     1        'PRECIP sm(1)....................',sm(1),' MM',
     1        'SNOW FALL sm(39)................',sm(39),' MM',
     1        'SNOW MELT sm(36)................',sm(36),' MM',
     1        'PRED SURFACE Q sm(3)............',sm(3),' MM',
     *        'SUB-SUR Q sm(4).................',sm(4),' MM',
     1        'GROUNDWATER Q sm(104)...........',sm(104),' MM',
     1        'REVAP sm(105)...................',sm(105),' MM',
     1        'GW PERC sm(106).................',sm(106),' MM',
     1        'GW RECHARGE sm(107).............',sm(107),' MM',
     1        'PRED H20 YLD sm(6)...............',sm(6),' MM',
     1        'DEEP PERC sm(5)..................',sm(5),' MM',
     1        'ET sm(7).........................',sm(7),' MM',
     1        'PET sm(109)......................',sm(109),' MM',
     *        'TRANS LOSSES sm(38)..............',sm(38),' MM',
     *        'TOTAL SUB-BASIN SED YLD sm(11)...',sm(11),' T/HA',
     *        'BASIN SED YLD sm(12).............',sm(12),' T/HA',
     1    'POND BUDGET'
          write (kw,'(1x,a)') 'Pond fluxes refer to '//
     1        'noncontributing area of basin;'
          write (kw,11106)
     1             'Areal fraction of basin fpd=',fpd
          write (kw,11103)
     1             'EVAPORATION sm(19)..........',smpond(19),' MM',
     1             'SEEPAGE sm(20)..............',smpond(20),' MM',
     *             'RAINFALL ON POOL sm(21).....',smpond(21),' MM',
     1    'INFLOW'
          write (kw,11104)
     1                  'Q sm(22)...............',smpond(22),' MM',
     1                  'Y sm(13)...............',sm(13),' T/HA'
          write (kw,'(t20,a)') 'OUTFLOW'
          write (kw,11104)
     1                  'Q sm(23)...............',smpond(23),' MM',
     1                  'Y sm(14)...............',sm(14),' T/HA'
          write (kw,'(t15,a)') 'RESERVOIR BUDGET'
          write (kw,11103)
     1             'EVAPORATION sm(24)..........',sm(24),' MM',
     1             'SEEPAGE sm(25)..............',sm(25),' MM',
     *             'RAINFALL ON POOL sm(26).....',sm(26),' MM'
          write (kw,'(t20,a)') 'INFLOW'
          write (kw,11104)
     1                  'Q sm(27)...............',sm(27),' MM',
     1                  'Y sm(16)...............',sm(16),' T/HA'
          write (kw,'(t20,a)') 'OUTFLOW'
          write (kw,11104)
     1                  'Q sm(28)...............',sm(28),' MM',
     1                  'Y sm(17)...............',sm(17),' T/HA'
      end if
c      write (kw,11111) dmba, (swmo(mo),mo = 1,12), (vrmo(mo),mo = 1,12)
c11111 format (f12.2,/,12f12.4,/,12e12.4)
      write (kw,11200) pl, atp, tl, atr
      write (kw,12200) prog
      write (kw,12300) title
      write (kw,7100)
      sbpup = sbpup / xn
      write (kw,7200) sm(40), sm(41), sm(42), sm(45), sm(43), sm(46),
     *    sm(44), sbpup
      spal = spal / xn
      spas = spas / xn
      sfn = sfn / xn
      sfp = sfp / xn
      sdnit = sdnit / xn
      sfix = sfix / xn
      shmn = shmn / xn
      srwn = srwn / xn
      shmp = shmp / xn
      srmn = srmn / xn
      srmp = srmp / xn
      write (kw,7300) spal, spas, sfn, sfp, sfix, sdnit
      write (kw,7400) shmn, srwn, shmp, srmn, srmp
      write (kw,7500)
c     do 800 k = 1, nptot(j)
      do 800 k = 1, 10
C       SM(K+46)=SM(K+46)-SM(K+56)
C*****CONVERT FROM KG/HA TO MG/HA
C       SM(K+46)=SM(K+46)*1.E6
C       SM(K+56)=SM(K+56)*1.E6
        sm(k+83) = sm(k+83) * 1.e6
        sm(k+93) = sm(k+93) * 1.e6
        pab(k) = pab(k) * 1.e6
        pdb(k) = pdb(k) * 1.e6
```

```
C******************************
        write (kw,7600) k, sm(k+46), sm(k+73), sm(k+83), pab(k),
     *      pdb(k), sm(k+93)
        pfp = 0.
        pfg = 0.
        do 790 j = 1, lu
          pfp = pfp + ffp(k,j) * flu(j)
          nn = ns(j)
          do 780 l = 1, nn
            pfg = pfg + gp(k,j,l) * flu(j)
  780     continue
  790   continue
C**********CONVERT FROM KG/HA TO MG/HA
        pfp = pfp * 1.e6
        pfg = pfg * 1.e6
        write (kw,7700) pfp, pfg
  800 continue
C     WRITE MAX AND MIN VALUES TO GRAPH FILE
C     do 348 j=1,lu
C     write(14,346) (xmin(k,j),k=1,25)
C     write(14,346) (xmax(k,j),k=1,25)
C348  continue
C     write(14,346) (xmin(k,lu1),k=1,28)
C     write(14,346) (xmax(k,lu1),k=1,28)
C346  format(28f10.3)
      if (iwst.ne.0.or.isst.ne.0) then
        kk = nbyr * 12
        sump = 0.
        summ = 0.
        do 810 i = 1, kk
c         write (70,777) i,wob(i),wpd(i)
c         777     format (i5,2f8.3)
          read (9,8400,end=805,err=805,iostat=iostat) wob(i), sob(i)
805       if (iostat.ne.0) then
            print '(3(1x,a,i4))','End of streamflow file at i=',i,
     1            'dev. 9; skip comparison.'
            go to 825
          end if
          wob(i) = wob(i) * 86400 / 35.3146
          sob(i) = sob(i) / 35.3146
c         convert m^3/month to m^3/s
c c       wob(i) = wob(i) / 2592000.
c c       wpd(i) = wpd(i) / 2592000.
c c       write(7,777) wob(i), wpd(i)
c         777     format(2f16.6)
          summ = summ + wob(i)
          sump = sump + wpd(i)
  810   continue
        do 811 i = 1, 4
          close (i)
  811   continue
        if (iwst.ne.0) then
          iyr = iyr - nbyr
          write (kw,12200)
          write (kw,12300) title
          write (kw,8100)
          write (kw,8200)
C         COMPUTE STATS ON MEASURED AND PREDICTED MONTHLY WATER YIELDS
          if (summ.gt.0.) call vald25(wob,wpd,kk)
        end if
        if (isst.ne.0) then
          if (iwst.ne.1.or.isst.ne.1) then
            iyr = iyr - nbyr
          end if
          write (kw,12200)
          write (kw,12300) title
          write (kw,8300)
          write (kw,8200)
          kk = nbyr * 12
C         COMPUTE STATS ON MEASURED AND PREDICTED MONTHLY SEDIMENT YIELDS
          if (sump.gt.0.) call vald25(sob,spd,kk)
          end if
        end if
825     continue
c       close(9)
c       open(51,file='opt.out')
c       write (51,5101) pminm, rsq, xnash
c5101 format ('runoff meas - pred      = ',f12.4,/,
c       *               'runoff r-sqared        = ',f12.4,/,
c       *               'runoff nash-sutcliffe  = ',f12.4)
c       close(51)
        do 812 i = 1, 20
          close (i)
  812 continue
        stop
 5100 format (10x,'mg = 1.E6 KILOGRAMS',//,40x,
     *    '****************** PESTICIDE NUMBER *******************',/,
     *    40x,
     *    '------------------          mg/HA        ---------------------',/,
     *    2x,'SUB',t19,'1',t31,'2',t43,'3',t55,'4',t67,'5',t79,'6',t91,
     *    '7',t103,'8',t115,'9',t126,'10',/)
 5200 format (a1,9x,5i6,f6.3,i9)
 5300 format (1x,i4,a4,1x,10f12.2)
 5400 format (/,1x,i4,a4,1x,10f12.2)
 5500 format (1x,i4,a4,1x,10f12.2,/)
 5600 format (1x,38('-'),t39,'TOXIC (mg/M**3)',t54,28('-'),1x,'|',t84,13
     *    ('+'),t96,'PHOSPHORUS (mg/L)',17('+'),/,1x,t8,'CONC',t16,
     *    'CONC',/,t9,'IN',t16,'IN',t36,'REACT',t57,'REACT',t111,
     *    'CHLORO A',t120,'SECCI',t127,'TEMP',/,t8,'LAKE',t15,'BOTTOM',
     *    t24,'IN',t30,'OUT',t36,'LAKE',t43,'VOLAT.',t50,'SETTLE',t57,
     *    'BOTTOM',t64,'RESUS',t71,'BURY',t78,'DIFUS',t85,'CONC',t93,
     *    'IN',t100,'OUT',t106,'LOSS',t112,'(MG/L)',t121,'(M)',t127,
     *    '(C)')
 5700 format (i4,16f7.1,2f7.2)
 5800 format (/i4,16f7.1,2f7.2/)
 5900 format (t5,'EMERGENCY SPILLWAY IS EXCEEDED',i3,' DAYS IN AREA ',i3
     *    )
 6000 format (t5,'RESERVOIR STORAGE IS ZERO',i3,' DAYS IN AREA ',i3)
 6100 format (/,1x,i4,2a4,10f12.2)
 6200 format (1x,i4,2a4,10f12.2,/)
 6300 format ('REACH ',i4,1x,i8,1x,i5,18e12.4)
 6400 format ('RES   ',i4,1x,i8,1x,i4,15e12.4)
 6500 format ('SUBAS ',i4,1x,i8,1x,i4,1x,i4,1x,i4,62f10.3)
 6503 format ('BISUB ',i4,1x,i8,1x,i4,18f10.3)
 6600 format (//,50x,'AVERAGE ANNUAL SUBBASIN DATA ',/,40x,
     *    '****************** PESTICIDE NUMBER ********************',/,
     *    40x,
     *    '------------------        mg/HA        ---------------------',/,
     *    2x,'SUB',t19,'1',t31,'2',t43,'3',t55,'4',t67,'5',t79,'6',t91,
     *    '7',t103,'8',t115,'9',t126,'10',/)
 6700 format (1x,i4,a4,1x,10f12.2)
 6800 format (//,53x,'AVERAGE ANNUAL BASIN DATA ',/,40x,
     *    '****************** PESTICIDE NUMBER ********************',/,
     *    40x,
     *    '------------------        mg/HA        ---------------------',/,
     *    5x,t17,'1',t29,'2',t41,'3',t53,'4',t65,'5',t77,'6',t89,'7',
     *    t101,'8',t113,'9',t124,'10',/)
 6900 format (1x,a4,1x,10f12.2)
 7000 format (/,t5,' AVE ANNUAL BASIN STRESS DAYS',/,t15,
     *    ' WATER STRESS DAYS = ',f8.2,/,t15,
     *    ' TEMPERATURE STRESS DAYS = ',f8.2,/,t15,
     *    ' NITROGEN STRESS DAYS = ',f8.2,/,t15,
     *    ' PHOSPHORUS STRESS DAYS = ',f8.2)
 7100 format (t15,'AVE ANNUAL BASIN VALUES')
 7200 format (//,t15,'NUTRIENTS',/,t20,'ORGANIC N =  ',f8.3,' (KG/HA)',
     *    /,t20,'ORGANIC P =  ',f8.3,' (KG/HA)',/,t20,
     *    'NO3 YIELD (SQ) =  ',f8.3,' (KG/HA)',/,t20,
     *    'NO3 YIELD (SSQ) = ',f8.3,' (KG/HA)',/,t20,'SOL P YIELD = ',f8
     *    .3,' (KG/HA)',/,t20,'NO3 LEACHED =  ',f8.3,' (KG/HA)',/,t20,
     *    'N UPTAKE =  ',f8.3,' (KG/HA)',/,t20,'P UPTAKE = ',f8.3,
     *    ' (KG/HA)')
 7300 format (t20,'ACTIVE TO LABILE P FLOW =  ',f8.3,' (KG/HA)',/,t20,
     *     'ACTIVE TO STABLE P FLOW =  ',f8.3,' (KG/HA)',/,t20,
     *     'N FERTILIZER APPLIED = ',f8.3,' (KG/HA)',/,t20,
     *     'P FERTILIZER APPLIED = ',f8.3,' (KG/HA)',/,t20,
     *     'N FIXATION = ',f8.3,' (KG/HA)',/,t20,'DENITRIFICATION = ',f8
     *     .3,' (KG/HA)')
 7400 format (t20,'HUMUS MIN ON ACTIVE ORG N =  ',f8.3,' (KG/HA)',/,t20,
     *     'ACTIVE TO STABLE ORG N =  ',f8.3,' (KG/HA)',/,t20,
     *     'HUMUS MIN ON ACTIVE ORG P = ',f8.3,' (KG/HA)',/,t20,
     *     'MIN FROM FRESH ORG N = ',f8.3,' (KG/HA)',/,t20,
     *     'MIN FROM FRESH ORG P = ',f8.3,' (KG/HA)')
 7500 format (//,t15,'PESTICIDES')
 7600 format (/,t15,i2,t20,'DISSOLVED = ',f15.4,' (mg/HA)',/,t20,
     *     'SORBED = ',f15.4,' (mg/HA)',/,t20,
     *     'LEACHED (PAST ROOT ZONE) = ',f15.4,' (mg/HA)',/,t20,
     *     'APPLIED = ',f15.4,' (mg/HA)',/,t20,'DECAYED = ',f15.4,
     *     ' (mg/HA)',/,t20,'SUBSURFACE = ',f15.4,' (mg/HA)')
 7700 format (t20,'FINAL PEST ON PLANT = ',f15.4,' (mg/HA)',/,t20,
     *     'FINAL IN GROUND = ',f15.4,' (mg/HA)')
c
 7800 format (i5,14f9.2)
 7900 format (i4,4f10.1)
 8000 format (///,t17,'AVE MONTHLY BASIN VALUES',/t19,'SNOW',t37,'SUB',
     *     t46,'WATER',/t4,'MO',t11,'R',t19,'FALL',t28,'SUR Q',t37,
     *     'SUR Q',t46,'YIELD',t55,'ET',t66,'Y',t76,'EO',/t10,'(MM)',t19,
     *     '(MM)',t28,'(MM)',t37,'(MM)',t46,'(MM)',t55,'(MM)',t64,
     *     '(T/HA)',t75,'(MM)')
 8100 format (/,10x,'MONTHLY AND ANNUAL WATER STATISTICS')
 8200 format (15x,'MONTHLY STATISTICS',/)
 8300 format (/,10x,'MONTHLY AND ANNUAL SEDIMENT STATISTICS')
 8400 format (10e12.4)
 8500 format ('1',/t20,'SOIL SURFACE LAYER')
 8600 format (/,t67,'ORG',t78,'ORG',t89,'SOLUBLE',/,t68,'N',t80,'P',t92,
     *     'P',/,t65,'(G/M3)',t77,'(G/M3)',t89,'(G/M3)',/t3,
     *     'SUB-BASIN     CLAY            SILT            SAND           K')
 8700 format (6x,i2,6x,f5.2,10x,f5.2,7x,f6.2,7x,f5.2,4f12.2)
 8800 format (//,29x,'SEDIMENT SIZE DISTRIBUTION',/,8x,'SUB-BASIN',23x,
     *     'SIZE(MM)')
 8900 format (/,20x,'SAND        SILT        CLAY      SM AG        L AG',/,21
     *     x,'.20          .01         .002        .03         .50',/)
 9000 format (9x,i4,6x,5(f6.3,4x))
 9100 format (/6x,'TOTAL SOIL WATER')
 9200 format (i5,11f10.1)
 9300 format (//t21,'IRRIGATION - AVE. ANNUAL',/,t10,'SUB-BASIN',t27,
     *     'NO.OF ',t44,'VOLUME',/,t14,'NO.',t24,'APPLICATIONS',t42,
     *     'APPLIED(MM)',/)
 9400 format (11x,i4,9x,i4,13x,f8.3)
 9500 format (t5,'FINAL VALUES'/)
 9600 format ('+',/,3x,11(2x,a4))
 9700 format (//t22,'FINAL CONTENTS'/t11,'-----PONDS-----',4x,
     *     '---RESERVOIRS---'/t11,'WATER',t21,'SED',t31,'WATER',t41,'SED'
     *     /t2,'SUB-BASIN VOL',t20,'CONC',t32,'VOL',t40,'CONC'/t4,'NO',
     *     t12,'(MM)',t20,'(PPM)',t32,'(MM)',t40,'(PPM)')
 9800 format (///t5,'BASIN STATISTICS')
 9900 format (a)
c***format changed for pc swat
c10200 format (22i4)
10300 format (/t10,'INITIAL COMPOSITE ST = ',f7.1,' MM')
10400 format (t24,'SUB',t31,'WATER',t38,'PERCO',t52,'SED',t63,
     *     'ORGANIC ORGANIC',t79,'SURQ',t86,'SOL',t92,'CROP',t98,'SSQ',
     *     t102,'PERC',t114,'RES.',/,t10,'R',t17,'SURQ',t23,'SURQ',t31,
     *     'YIELD',t38,'LATE',t44,'ET',t51,'YIELD',t59,'SW',t66,'N',t73,
     *     'P',t80,'NO3',t87,'P',t93,'NO3',t98,'NO3',t103,'NO3',t114,
     *     'VOL',t127,'EO',/,t9,'(MM)',t17,'(MM)',t23,'(MM)',t31,'(MM)',
     *     t38,'(MM)',t43,'(MM)',t50,'(T/HA)',t58,'(MM)',t83,
     *     '--------(KG/HA)--------',t114,'(MM)',t127,'(MM)')
10500 format (i5,13f7.2,2f5.2,1x,4f8.2)
10600 format (/i5,13f7.2,2f5.2,1x,4f8.2//)
10700 format (/t10,'POND BALANCE')
10800 format (/t10,'RESERVOIR BALANCE')
10900 format (/t5,'SUB-BASIN STATISTICS'//t17,'AVE ANNUAL VALUES'/t32,
     *     'SUB'/t2,'SUB-BASIN RAIN',t21,'SUR Q',t31,'SUR Q',t43,'Y',/t4,
     *     'NO',t12,'(MM)',t22,'(MM)',t32,'(MM)',t41,'(T/HA)')
11000 format (i4,11f6.1)
c     146 FORMAT (/T15,'STD DEV OF RAIN STORMS')
11100 format (t10,'AVE ANNUAL BASIN VALUES'//t15,'PRECIP = ',f8.1,' MM'/
     *     t15,'SNOW FALL =',f8.2,' MM'/t15,'SNOW MELT = ',f8.2,' MM'/
     *     t15,'PRED SURFACE Q = ',f8.2,' MM'/t15,'SUB-SUR Q =',f8.2,
     *     ' MM'/t15,'GROUNDWATER Q = ',f8.2,' MM'/t15,'REVAP =',f8.2,
     *     ' MM'/t15,'GW PERC = ',f8.2,' MM'/t15,'GW RECHARGE =',f8.2,
     *     ' MM'/t15,'PRED H20 YLD = ',f8.2,' MM'/t15,'DEEP PERC =',f8.2,
     *     ' MM'/t15,'ET = ',f8.1,' MM'/t15,'PET = ',f8.1,'MM'/t15,
     *     'TRANS LOSSES = ',f8.2,' MM'/t15,
     *     'TOTAL SUB-BASIN SED YLD =  ',f8.3,' T/HA'/t15,
     *     'BASIN SED YLD = ',f8.3,' T/HA'/t15,'POND BUDGET'/t20,
     *     'EVAPORATION = ',f8.3,' MM'/t20,'SEEPAGE = ',f8.3,' MM'/t20,
     *     'RAINFALL ON POOL =',f8.3,' MM'/t20,'INFLOW'/t25,'Q = ',f8.3,
     *     ' MM'/t25,'Y = ',f8.3,' T/HA'/t20,'OUTFLOW'/t25,'Q = ',f8.3,
     *     ' MM'/t25,'Y = ',f8.3,' T/HA'/t15,'RESERVOIR BUDGET'/t20,
     *     'EVAPORATION = ',f8.3,' MM'/t20,'SEEPAGE = ',f8.3,' MM'/t20,
     *     'RAINFALL ON POOL = ',f8.3,' MM'/t20,'INFLOW'/t25,'Q = ',f8.3,
     *     ' MM'/t25,'Y = ',f8.3,' T/HA'/t20,'OUTFLOW'/t25,'Q = ',f8.3,
     *     ' MM'/t25,'Y = ',f8.3,' T/HA')
11200 format (t15,'YIELD LOSS FROM PONDS'/t20,'Q = ',f7.3,' MM'/t20,
     *     'Y = ',f7.3,' T/HA'/t15,'YIELD LOSS FROM RESERVOIRS'/t20,
     *     'Q = ',f7.3,' MM'/t20,'Y = ',f7.3,' T/HA')
11300 format (1x,'SUB-BASIN',t30,'SOIL WATER FOR LAYER NO'/t4,'NO')
11400 format (/t10,'FINAL COMPOSITE ST = ',f7.1,' MM')
11500 format (t10,'MIN INDIVIDUAL WATER ST = ',f8.1,' MM')
11600 format (t10,'FINAL COMPOSITE POND ST = ',f7.2,' MM')
11700 format (t10,'FINAL COMPOSITE RESERVOIR ST = ',f7.2,' MM')
11800 format (/t10,'CN--MEAN = ',f6.3,5x,'MAX = ',f6.3,5x,'MIN = ',f6.3)
11900 format (/t10,'PRED MO WATER YLD'/t15,'MEAN = ',f7.2,' MM'/t15,
     *     'ST DEV = ',f7.2,' MM')
12000 format (/t10,'SOIL WATER BALANCE = ',e16.6,' MM')
12100 format (t10,'Q = ',e16.6,' MM',5x,'Y = ',e16.6,' T/HA')
12200 format ('1',/t5,20a4,t105,2(a2,'/'),a2,5x,2(i2,':'),i2)
12300 format (/(t5,20a4))
12400 format (i5,4f10.1,4x,4f10.1)
      end

c      integer function fpehandler(sig,code,context)
c      integer sig, code, context(5)
c      call abort()
c      end
```

Files included in Swat's mainline for the Swat-Modflow linkage

Listings for these files shown below were taken from the compiler listing of Swat's mainline, and have the extension ".h" (Swatmod1-3, daily, monthly, annual, avgann).

Swatmod1.h: data declarations and file device numbers

```
c-begin file swatmod1.h:  May 1999  spp
c define additional arrays and variables for Swat/Modflow linkage and for
c Swat modifications related primarily to daily weather data and simulation
c of irrigation and HRUs.
c
c        variables for HRU option: set for up to mb=100 subbasins, 50 years,
c        mxsoil=6 soils, mxuse=6 land uses, mxhru=30 HRUs.
      character*160 recinp            !use for header in ~.cod file

      parameter (mxuse=6,mxsoil=6,mxhru=30)
      dimension shed(30,0:mb)       !array passed to Modflow if iopswt > 0
      character*8 namcas, namhru(mxhru)
c        case name for HRU-averaged results calculated in Swbavg:
      character*8 casavg
      character*8 aqflbl, crplbl, crpnam(mxuse), soilnm(mxsoil)
      dimension frcirr(50), crpfrc(mb)
      dimension cropwt(mb,mxuse)    !land use factors for hru weights
      dimension soilwt(mb,mxsoil)   !soil type factors for hru weights
      dimension crpsum(mb)  !sum of areal fractions over land uses for each subbasin
      dimension idxsol(mxhru), idxuse(mxhru)
      dimension hruwts(mb,mxhru), sumwts(mb), hrubas(mxhru)

      integer numhru !no. HRUs required to represent spatial heterogeneity;
c             specified on extended *.cod input file, read from swatmod2.h
      integer idxhru !index to HRU; specified by matching item from list of
c               HRU names on file *.cod with case name on file *.cio.

      integer idxaqf(mxhru)     !indicate whether (y=1,n=0) soil water can
c             percolate to ground water; otherwise, block percolation for
c             lowest soil layer nn in subr Percl6.  Pass iaqufr=idxaqf(idxhru)
c             through subroutine argument lists from Main.  Hierarchy:
c                 Swatmain-->Subbasin-->Purkl8-->Percl6)

      real aqfsub !areal fraction of subbasin with underlying aquifer
      real aqfshl !shallow fraction of aquifer area
c           aqfsub for subbasin j is given by shed(7,j), which is read from
c       modified *.cod input file (swatmod2.h) to initialize; and, along with aqfshl,
c       from the *.shl file at the beginning of each gw time step, which
c       depends on iopmod.
c           aqfshl for subbasin j is given by shed(26,j), which is read from
c       the *.shl file at the beginning of each time step.
      character chdate*8, chtime*11
c         note: parameter mb=max. no. subbasins (declared in file common.f)
c                         ml=max. no. soil layers (coordinate w/ max in SWB)
      character*220 hdrout !for header in balance and summary files
      character balttl*15
c       STDOUT, name of Swat std output file, is read from 'File.cio' in
c       subr Open.  stdout was added to subr Open arg list to be passed back
c       to main so that it could be used to name the balance file, nambal, and
c       balance summary file, namsum (in swatmod3.h).
      character*13 stdout, namshl, nambal, namsum, namirr, namrad,
     1             namevt, namwts
      data stdout /' '/
c         special array to show pond fluxes (sm(j), j=19 to 23) w.r.t. pond area
c         in ~.std output:
      dimension smpond(25)
c         formats for temperature, precip, and optional add'l weather data file:
      character*20 fmttmp, fmtpcp, fmtrad
c         msubo variables, mb subbasins: defined in file common.f.
c         msubo and mb are also used to dimension ssub, sysub and stsub.
      dimension ssprv(msubo,mb)  !previous day's values of array ssub
      dimension sdsub(msubo,mb)  !daily flux values by subbasin
      dimension irnyr(mb)  ! no. irrigation applications for current year
      dimension vprev(mb)        !pond volume at beginning of time step
      dimension swprev(mb)       !soil water content at beginning ot time step
c   arguments:
c   j = subbasin index
c   wsfmax = daily max. added water (mm/operation) for auto irrig; set in ~.mco.
c   swminf = min. soil water as fraction of available water capacity below
c           which irrigation will take place during growing season.
c   swtrig = soil water trigger: minimum soil moisture (total in profile, mm)
c           below which irrigation will be applied if option irr(j)=3, set
c           in the ~.mco file.  swtrig is calculated in Crpmd as follows:
c               if (igro(j).eq.1.and.g(j).gt.0.) then
c                   swtrig = FLOAT(igro(j))*sumfc(j)*swminf
c   evtgw = evaporation rate from water table, sub j, calculated in
c           MODFLOW's subr EVT1FM (and EVT1BD) at beginning of the gw
c           time step.
c   crvnum = current curve number, updated in subr subbasin.
c       evaporation from water table as calculated by MODFLOW
      dimension evtgw(mb)
c       component of evtgw not redistributed over soil profile (subr subbasin)
      dimension egwuna(mb)
      parameter (mxcomp=13) !max no. hydrologic components passed between SWAT & MODFLOW
      dimension idxssb(mxcomp),idxsub(mxcomp)
      integer mndays(12,2), ndayr(2)
      integer iophru !option for one of three schemes:
cc    data iophru /2/ !iophru is now read from *.cod input file (swatmod2.h);
c                       see also swatmod3.h and definitions in subr Preswb.
c               =1: original 18-HRU scheme for 6 soils and 3 land use types
c                       (dryland crop, irrigated crop, and grassland);
c               =2: (a) 24-HRU scheme for 6 soils and 4 land use/landform types:
c                 (c1: nonalluvial dryland crops; c2: alluvial irrigated crops;
c                 (c3: alluvial grassland; c4: nonalluvial grassland)
c                    (b) 10-HRU scheme that, for the Lower Republican River
c                        Basin model, is equivalent to the 24-hru scheme above.
c                =3: refinement of the above scheme, dividing the alluvial
c                  aquifer area into shallow and deep components; the shallow
c                  component is associated with evaporation from gw according
c                  to Modflow's simulation.  For this case, Modflow's results
c                  are read (file device ioshl).
      data fshall /1./  !factor multiplied by shallow aquifer fraction
c                  AND by flow rate for evap from shallow gw.  This is normally
c                  set to 1. Set to zero to check HRU scheme 3: the same
c                  results as HRU scheme 2 should be obtained for fshall=0.
c                  Use in Swatmod3.h (included in Main) and in subr. Preswb
      data (ndayr(j),j=1,2) /365,366/ !no. days in non-leap and leap years
      data (mndays(j,1),j=1,12) /
     1     31,28,31, 30,31,30, 31,31,30, 31,30,31/ !days/mon (non-leap years)
      data (mndays(j,2),j=1,12) /
     1     31,29,31, 30,31,30, 31,31,30, 31,30,31/ !days/mon (leap years)
      data (idxsub(j),j=1,12) / 1,22,12, 4,13, 5,11, 9, 7, 6,16,25/ !subbasin vectors
c          (note: 13th, Qtrib, is given in Preswb.)
      data (idxssb(j),j=1,12) / 1, 0, 7, 3,38, 4,5,107,105,104,20,108/!basin vectors
c            corresponding list of vectors in arrays swtflx and shed:
cccc  data (idxmod(j),j=1,12) /10,11,12,13,14,15,16,17,18,19,20,21/
c            Subbasin data fields written to MODFLOW summary files:
c             pass from Swat's ssub to Modflow's shed:
c                 irr,et,surf Q,lat Q,perc_rz,pond_seepage,wetland_seepage.

ccccc data (dscmod(j),j=0,3) /'avg annual','annual','monthly','daily'/
c     2.  nominal atm. pressure at site (kPa): z=elev (m)            (4.4.12)
      pnom(zelev) = 101.3*((293.-0.0065*zelev)/293.)**5.256
c         pressure conversions:
c         1 atm = 101.32 kPa (kPa = kilopascal; 1 pascal = 1 N/m^2)
c               = 1013.2 millibars = 760 mm Hg = 10.33 m water
c               = 14.7 psia      = 29.92 in Hg = 33.9 ft water
c-----------------------------------------------------------------------
c     iopmod time frame    subbasin j     total basin
c-----------------------------------------------------------------------
c     0      avg annual    stsub(k,j)     sm(L)
c   +-1      annual        sysub(k,j)     smy(L)
c   +-2      monthly       smsub(k,j)     smm(L)
c   +-3      daily         ssub(k,j)(*)   ssb(L)
      call DATE (chdate)
      call TIME (chtime)
      iobal=90 !output: balance file to be read by SWBAVG or MODFLOW
      ioshl=91 !input: shallow gw file, written by MODFLOW for iopshl>0;
c                      see swatmod2.h.
      iopcp=92
      iotmp=93
      iomat=94
      iosum=95 !for summary file opened in swatmod3.h, named after balance file)
      ioirr=96 !for daily irrigation (opened in swatmod3.h; named after balance file)
c               in subr Crpmd where file is written if itrace > 0)
      iorad=97
      ioevt=98 !for ET daily file opened in swatmod3.h; ioevt is also set to 98
c               in subr Penman where file is written if itrace > 0)
      iowts=99 !output device no. for HRU weights written for each time step.
      open (unit=iopcp,file='swatpcp.log',status='unknown')
      open (unit=iotmp,file='swattmp.log',status='unknown')
      open (unit=iomat,file='swatmat.log',status='unknown')
      print '(1x,a)', chdate//' at '//chtime//' Program SWAT '//
     1                '(additional subbasin files written --spp, KGS)'
      call TIMER(itbgn)
```

Swatmod2.h: additional data read from Swat *.cod file for Swat-Modflow linkage

```
c-begin file swatmod2.h:  May 1999  spp
      print '(1x,a)','Read added options from Swat *.cod(10)'//
     1              ' input file.'
      print '(1x,a)','(File swatmod2.h, included in Swat mainline)'
c
c Set option iopswt internally here instead of in the modified *.cod input
c file until the option has been implemented.
c (iopswt is an option, yes=1,no=0, to call Modflow as a subroutine at
c the end of each time step)
      iopswt=0 !set option to run Swat and Modflow with separate executions.
c Added data read from Swat's *.cod input file:
      read (incode,*) ilog,iopmod,numhru,iophru,iopwea,iopwfl,
     1  iprirr,iprevt,iopswm,swminf,wsfmax,cnvlen !(9i4,2f6.0,f8.0)

      iopmod = MAX0 (-3, MIN0 (3,iopmod))  ! keep iopmod value within range
      print '(9i4,f6.3,f6.1,f8.4)', ilog,iopmod,numhru,iophru,
     1  iopwea,iopwfl,iprirr,iprevt,iopswm,swminf,wsfmax,cnvlen

      data depmpy, widmpy /1.e-3, 1000./ !depmpy(m/mm), widmpy(m/km)
c       conversion factor used for pond volumes (from Modflow units to Swat
c       units, e.g. (0.3048 m/ft)^3 (used in subr. Preswb):
      cnvvol = cnvlen*cnvlen*cnvlen

      read (incode,*) fmttmp, fmtpcp, fmtrad, namrad, namshl     !(3a20, 2a13)
      print '(4(1x,a/))','fmttmp='//fmttmp,'fmtpcp='//fmtpcp,
     1                  'fmtrad='//fmtrad,'namrad='//namrad
      print '(1x,a)', 'Shallow gw input file name '//namshl
      numhru=MAX0(numhru,1)        !make sure that numhru is at least 1

      if (iopswt.eq.0.and.numhru.gt.1) then
        read (incode,*) nsoils
        print '(1x,a,i2)','nsoils=',nsoils
        read (incode,*) aqflbl, crplbl, (soilnm(isoil),isoil=1,nsoils)
        print '(8(1x,a))',aqflbl,crplbl,(soilnm(isoil),isoil=1,nsoils)
c         for each subbasin, read areal fractions of cropland and each soil type:
        do j=1,Lu
          read (incode,*) idx, shed(7,j), crpfrc(j),
     1              (soilwt(j,isoil),isoil=1,nsoils)
          print '(i3,8f9.4)', j, shed(7,j), crpfrc(j),
     1      (soilwt(j,isoil),isoil=1,nsoils)
        end do
c         irrigation area as pct. of basin in each year of simulation period,
c         and convert to areal fractions:
        read (incode,*) (frcirr(j), j=1,nbyr)
        print '(1x,a,i5,1x,a)','Year Irrigated areal fraction'
        do j=1,nbyr
          frcirr(j) = frcirr(j)/100.
          iy = iyrbgn+j-1
          print '(i5,f8.5)', iy,frcirr(j)
        end do
c
c Land use classes: 1= nonirrigated crop rotation; 2=irrigated crop rotation;
c 3=grassland.  This list neglects some important uses e.g. in urban areas.
c
c Associate hrus with corresp. soil type, land use, and case names;
c the case name is given by the standard output file name prefix.
        idxhru=0
        read (incode,'(a)') recinp  !read heading for table
        print '(a)',' '//recinp(2:60)
        do i=1,numhru
          read (incode,*) idx,idxsol(i),idxuse(i),idxaqf(i),namhru(i)
          print '(4i5,1x,a)', idx, idxsol(i), idxuse(i), idxaqf(i),
     1                        namhru(i)
        end do
c           the following name is used in Swbavg (iswbav=1) but not in Swat:
        read (incode,*) casavg  !file name for results averaged over HRUs in
c                               Swbavg (iswbav=1); not used in Swat (iswbav=0).
c
```

```
        if (iophru.le.1) then      !HRU model option 1
          crpnam(1) = 'nonirrig'   !no irrig. e.g. wheat,sorghum,fallow rotation
          crpnam(2) = 'irrig'      !irrigated crops
          crpnam(3) = 'noncrop'    !range, pasture, and other noncrop land uses
        else if (iophru.eq.2) then !option 2 distinguishes subbasin with aquifer
          crpnam(1) = 'wsf-upld'  !non-irrigated crops in upland, no aquifer
          crpnam(2) = 'irr-aqfr'  !irrigated cropland over alluvial aquifer
          crpnam(3) = 'grs-aqfr'  !grasslands over alluvial aquifer
          crpnam(4) = 'grs-upld'  !grass in upland, no aquifer
        else  !option 3 distinguishes aquifer with shallow & deep components
          crpnam(1) = 'wsf-upld'  !non-irrigated crops in upland, no aquifer
          crpnam(2) = 'irr-aqdp'  !irrigated cropland, deep alluvial aquifer
          crpnam(3) = 'grs-aqdp'  !grasslands, deep alluvial aquifer
          crpnam(4) = 'grs-upld'  !grass in upland, no aquifer
          crpnam(5) = 'irr-aqsh'  !irrigated cropland, shallow alluvial aquifer
          crpnam(6) = 'grs-aqsh'  !grasslands, shallow alluvial aquifer
        end if
      end if          ! (iopswt.eq.0 and numhru.gt.1)

c       execute the following for Swat (iswbav=0) but not Swbavg (iswbav=1):
      if (iswbav.eq.0) then
        if (iopwea.gt.0) then
          if (namrad.eq.' ') namrad = 'SWATIN.RAD'
          print *,'Open weather input file for solar rad., r.h., & u10'
          print *,' weather input file name: '//namrad//
     1            ' format='//fmtrad
          open (unit=iorad,file=namrad,status='unknown')
        else
c           device no. passed as zero to subr Clicon indicates that the
c           additional weather data are not to be read.
          iorad = 0
        end if

c       Balance and balance summary output file names
c       (files opened in Swatmod3.h):
        idxdot=INDEX(stdout,'.')
        if (idxdot.gt.1) then
          namcas = stdout(1:idxdot-1)
          nambal = stdout(1:idxdot) // 'bal'
          namsum = stdout(1:idxdot) // 'sum'
          namirr = stdout(1:idxdot) // 'irr'
          namevt = stdout(1:idxdot) // 'evt'
        else
          namcas = 'SWATOUT'
          nambal = 'SWATOUT.BAL'
          namsum = 'SWATOUT.SUM'
          namirr = 'SWATOUT.IRR'
          namevt = 'SWATOUT.EVT'
        end if
        print '(4(1x,a/))','nambal='//nambal,'namsum='//namsum,
     1                     'namcas='//namcas
c         (for HRU scheme option iophru=3): name of file with results
c         written by previous MODFLOW solution containing evap from
c         shallow aquifer, shallow aquifer fraction, and active fraction of
c         subbasin.  These results are used to couple Swat and Modflow
c         solutions by successive approximation.
        if (namshl.eq.' ') then
          if (idxdot.gt.1) then
            namshl = stdout(1:idxdot) // 'shl'
          else
            namshl = 'SWATOUT.SHL'
          end if
        end if

        print '(i5,1x,a)',Ilog,'Ilog'
        print '(i5,1x,a)',iopmod,'Iopmod, ground water time step '//
     1    'option: 3(daily), 2(monthly), 1(annual), 0(avg annual)'
        print '(i5,1x,a)',iopswt,'Iopswt, linkage option: '//
     1    '0(separate execution of Swat & Modflow); 1(linked exec.)'
        print '(1x,a)', 'Weather options:'
        print '(i5,1x,a)',iopwea,'Iopwea (y=1,n=0): read, rather than'//
     1    ' synthesize, daily solar radiation, rel. hum., & wind speed.'

        print '(1x,a)', 'Irrigation options '//
     1    '(control specified by Irr in *.mco files):'

        print '(i5,1x,a)',iopswm,'Iopswm: irrigation threshold is '//
     1    'either (=1) soil water content or (=0) plant stress factor'

        print '(1x,f8.2,1x,a)',wsfmax,'=wsfmax (mm/day), max. daily '//
     1                     'irrigation depth (vol/irrig. area)'
        print '(1x,f8.2,1x,a)',swminf,'=swminf, soil water content '//
     1                      'threshold as fraction of col. ht. of awc'
        print '(3(1x,a/))',
     1    'Hydrologic balance output file nambal = '//nambal,
     1    'Temperature input format fmttmp = '//fmttmp,
     1    'Precipitation input format fmtpcp = '//fmtpcp
        if (iopwea.gt.0) then
          print '(3(1x,a/))',
     1      'Radiation input file namrad = '//namrad,
     1      'Radiation input format fmtrad = '//fmtrad
        end if

c         scan list of HRUs; look for a match between HRU names, namhru(i),
c         and NAMCAS, the prefix for the standard output file name read from
c         FILE.CIO.

        if (iopswt.eq.0.and.numhru.gt.1) then
          do i=1,numhru
            if (idxhru.eq.0.and.INDEX(namcas,namhru(i)).gt.0) then
              idxhru = i
              isolhr = idxsol(i)
              iusehr = idxuse(i)
              iaqufr = idxaqf(i) !0: soil is underlain by bedrock;
c                                  1: by deep aquifer; 2: by shallow aquifer.
            end if
          end do
          if (idxhru.ge.1) print '(1x,a,i3,1x,a)',
     1        ' HRU index ',idxhru, namhru(idxhru)
c
c       Set up output file names based on case name, casavg, for HRU average:
          if (idxhru.eq.1) then
            if (casavg.eq.' ') then
              namwts = 'FILE.WTS'
            else
              lenwrd = LEN(casavg)
              idxwts = INDEX(casavg,' ') - 1
              if (idxwts.le.0) idxwts=8
              namwts = casavg(1:idxwts) // '.wts'
            end if
            open (unit=iowts,file=namwts,status='unknown')
            write (iowts,'(a15,a5,26(2x,a8))') ' year mon days',
     1          '  sub',(namhru(idx),idx=1,numhru),'sum(wts)','crpsum'

c             For the first HRU simulated by Swat (idxhru=1; see list on
c             the modified Codes input file), calculate weights for all
c             HRUs and write a file of weights to be read by Swbavg.
            idxhr1 = 1
            idxhrn = numhru
          else
c             For all other HRUs (idxhru > 1), calculate only the weights
c             corresponding to the HRU simulated by SWAT; see note in Preswb
            idxhr1 = idxhru
            idxhrn = idxhru
            iowts = 0 !pass to Subwts to skip writing weights
          end if !write table of weight functions for the first HRU in the list
        end if ! (iopswt.eq.0 and numhru.gt.1)
      end if   ! (iswbav=0, i.e., Swatmod2.h is included in Swat, not Swbavg)

      print '(3(1x,a,i2))', 'Swatmod2.h: Iophru=',iophru,
     1    'Numhru=',numhru,'Idxhru=',idxhru
```

Swatmod3.h: initialize results summary for Modflow; begin stress period

```
c-begin file swatmod3.h:  May 1999  spp
      if (iy.eq.1) then

        bsarea = da*(widmpy/cnvlen)**2 !convert basin area to Modflow units
        aqfbas = 0.
        do j=1,lu
          aqfbas = aqfbas + shed(7,j)*flu(j) !aquifer areal fraction of basin
          shed(1,j) = flu(j)   !subbasin area as fraction of basin area
          shed(8,j) = fp(j)    !noncontributing area as fraction of subbasin area
          vprev(j) = v(j)        !pond volume (m^3) at beginning of time step
          swprev(j) = sw(j)      !soil water (mm) at beginning of time step
          evtgw(j) = 0.
          egwuna(j)=0.
        end do
c          evap from gw is zero except for hrus with shallow aquifer, which
c          are distinguished for the third hru scheme (option iophru=3);
c          see manual or subr. Preswb for definitions of iophru and iaqufr.
        aqfrat = frcirr(iy)/aqfbas
        print '(1x,a,i5,1x,a,f6.4)', 'iy=',iy,'frcirr=',frcirr(iy)
        print '(2(1x,a,f6.4))','aqfbas=',aqfbas,'frcirr/aqfbas=',aqfrat

        if (iopswt.eq.0 .and. iophru.eq.3) then
          print '(1x,a,i2,a)', 'Open shallow gw input file '//namshl//
     1      ', unit ',ioshl
          open (unit=ioshl,file=namshl,status='unknown')
          read (ioshl,'(a)') recinp
          print '(1x,a)',recinp(2:80) // ' evtgw(mm) shed(1,j)'//
     1        '|---shlfrc-----|---cnvfac-----'

c          set up evaporation from gw (evtgw) for each subbasin based on
c          Modflow's solution for the first time step:
          print '(1x,a,i5)','Lu=',Lu
          do j=1,lu
            read (ioshl,'(6i5,5f10.0)',iostat=iostat,err=105,end=105)
     1        idxyr,idxper,idxstp,isub,ngract,ngrshl,
     1        shed(26,j),shed(7,j),shed(18,j),shed(28,j),shed(29,j)
c example:
cear  per  stp  sub  act shal frc shall frc activ   evap-gw  shal dtw   deep dtw
c977    1    1    1   16    3 0.1875000 0.0732544      0.57      3.29      22.16

c            shallow aquifer as areal fraction of BASIN:
c            use this to convert ET-gw from flow rate, given by Modflow,
c            to a depth, evtgw(j), for an HRU with shallow ground water
c            in subbasin j:
            shlfrc = shed(1,j)*shed(7,j)*shed(26,j) !subfrc*aqffrc*shlfrc
            if (1.+shlfrc.gt.1.) then
              cnvfac = (cnvlen/depmpy)*86400./(shlfrc*bsarea)  ![L^3/T] to mm
              evtgw(j) = cnvfac*shed(18,j)        !evap from gw (mm)
            end if
c                fshall, defined in swatmod1.h, is normally 1; set fshall=0
c                to see if HRU scheme 3 reduces to HRU scheme 2.
c                Artificially reduce shallow fraction, shed(26,j):
            if (fshall.lt.1.) then
                shed(26,j) = fshall*shed(26,j)
                evtgw(j) = fshall*evtgw(j)
            end if
            print '(6i5,2f10.7,4f10.2,f10.7,2e15.7)',
     1        idxyr,idxper,idxstp,j,ngract,ngrshl,
     1        shed(26,j),shed(7,j),shed(18,j),shed(28,j),shed(29,j),
     1        evtgw(j),shed(1,j),shlfrc,cnvfac
          end do
105       continue
        end if
      end if
c            call Modflow each stress period:
      if (iopswt.gt.0) then
        call MODFLO (iopswt,iopmod,delt,nbyr,kper,
     1               nstp,kstp,iyr,bsarea,mb,shed)

      else if (iopswt.eq.0 .and. iy.eq.1) then
        print '(1x,a,i2,a)', 'Open file '//nambal//', unit ',iobal
        print '(5(1x,a,i5))', 'Swatmod3.h: iobal=',iobal,
     1    'Lutot=',Lutot,'isolhr=',isolhr,'iusehr=',iusehr
        open (unit=iobal,file=nambal,status='unknown')
        write (iobal,'(2i5,f15.0,f10.7,2i5,1x,a)') nbyr, iopmod,
     1    bsarea, fpd, idxhru, numhru, namcas

c         header record for balance file:
        write  (iobal,'(a)') '  sub sub fract noncontrb HRU fract'//
     1    '    soilwt    cropwt soil Luse soil name  land use'

c Compute weights to be used in SWBAVG to take a weighted average over HRUs:
c Subr Subwts is called to do this here, at the beginning of the first time
c step, and just prior to calling subr Preswb at the end of each month or
c year, depending on iopmod (see monthly.h and annual.h).

        if (numhru.gt.1) then
          iozero=0 !substitute for iowts, weights output file device no.,
c                   to skip writing to weights file for this initial call.
          call Subwts (iozero,balttl,mxuse,mxsoil,mb,Lutot,iophru,mxhru,
     1      numhru,idxhru,idxhru,idxsol,idxuse,idxaqf,shed,frcirr(iy),
     1      aqfbas,crpfrc,cropwt,crpsum,soilwt,hruwts,sumwts,hrubas)

          do j=1,Lutot
            write (iobal,'(i5,5f10.7,2i5,2(2x,a8),2f10.7)')
     1        j,flu(j),fp(j),hruwts(j,idxhru),
     1        soilwt(j,isolhr),cropwt(j,iusehr),
     1        isolhr,iusehr,soilnm(isolhr),crpnam(iusehr)
          end do
        end if
c
        hdrout = ' Year  mon days     delt (sec)  sub'//
     1   '     precip     irrig     ETact     runoff     XMloss'//
     1   '       QLAT      PERC   recharg     et-gw   baseflo'//
     1   '    pondsep     ETpot   tribflo    dSW/dt   dPND/dt'//
     1   ' et-gw(undistrib.)'

        write  (iobal,'(a)') hdrout  !195 chars header for results in each time step

c         open the annual summary file (ext. sum), named according to
c         the balance file (ext. bal):
        print '(1x,a,i2,a)', 'Open file '//namsum//', unit ',iosum
        open (unit=iosum,file=namsum,status='unknown')
        write  (iosum,'(a)') hdrout  !160-chars header for results in each time step

c       Note: daily results for irrigation and evaporation are written to
c       the following two files subject to options that are currently
c       set inside subroutines crpmd and evap8, respectively (variable itrace).

        if (iprirr.gt.0) then
          print '(1x,a,i2,a)', 'Open daily irrigation file '//namirr//
     1                         ', unit ',ioirr
          open (unit=ioirr,file=namirr,status='unknown')
            hdrout='  sub Year Jul igro  pcp,mm   ETpot ETplant'//
     1           '  ETsoil SW,mm swtrig,mm  irr,mm sw/sumfc      ws'
          write  (ioirr,'(a)') hdrout(1:68)  !68 chars header for simulated irrigation
        end if  !iprirr > 0

        if (iprevt.gt.0) then
ccc       print '(1x,a,i2,a)', 'Open daily Penman evap file '//namevt//
ccc  1                        ', unit ',ioevt
          open (unit=ioevt,file=namevt,status='unknown')
ccc       hdrout = '  sub Year  Jul  rad,ly u10,m/s rh,frac tmpav,C'//
ccc  1      ' dTsol,C  albedo ETPOTmm ETshort  ETlong  ETsoil ETvpdef'
          write  (ioevt,'(a)') hdrout(1:103)  !103 chars header for subr Penman pot. ET
        end if  !iprevt > 0
      end if  ! (iopswt.eq.0 .and. iy.eq.1)
```

Daily.h: summarize daily results for Modflow (iopmod=3)

```
c begin file daily.h, included in Swatmain: daily fluxes --spp
          if (iopmod.eq.3) then
c                 calculate daily flux as change in accumulator ssub:
              do k = 1, nvsub
                do j = 1, lutot
                  sdsub(k,j) = ssub(k,j) - ssprv(k,j)
                  ssprv(k,j) = ssub(k,j)
                end do
              end do
              write (balttl,'(3i5)') iyr,mo1,ida   !char balttl*15
              ndays=1  !Swat time step (1 day) to be passed to MODFLOW
c                !note: basin summaries (idxssb,ssb) not passed to Preswb

              call PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,
     1          depmpy,cnvvol,bsarea,mxcomp,msubo,mb,lutot,idxsub,
     1          sdsub,shed,sw,v,swprev,vprev,iaqufr,iophru,
     1          ioshl,evtgw,egwuna)

              if (iopswt.gt.0) call MODSTP (iopswt,iopmod,delt,
     1            nbyr,kper,nstp,kstp,iyr,bsarea,mb,shed)
          end if
```

Monthly.h: summarize monthly results for Modflow (iopmod=2)

```
c begin file monthly.h, included in Swatmain: monthly fluxes --spp
              if (iopmod.eq.2) then
                ndays = mndays(mo1,idleap)  !no. days in month
                write (balttl,'(3i5)') iyr,mo1,ndays    !char balttl*15

c               print '(1x,a,i3,1x,a,f8.5)','Monthly.h: iy=',iy,
c    1                'frcirr(iy)=',frcirr(iy)
c                !basin summary smm and index idxssb are not passed to Preswb
                if (iy.eq.1.and.mo1.eq.1) then
                  print '(1x,a)','Monthly.h:' // balttl
                  print '(5(1x,a,i3))','mxcomp=',mxcomp,'msubo=',msubo,
     1                'numhru=',numhru,'idxhru=',idxhru
                  print '(5(1x,a,i3))','iaqufr=',iaqufr,
     1                'iophru=',iophru,'isolhr=',isolhr,'iusehr=',iusehr
                  print '(5(1x,a,i3))','iopswt=',iopswt,'iobal=',iobal,
     1                'iosum=',iosum,'ndays=',ndays
                  print '(5(1x,a,i3))','mxsoil=',mxsoil,'ioshl=',ioshl
                end if
c
c Compute weights to be used in SWBAVG to take weighted average over HRUs:
                if (numhru.gt.1) call Subwts (iowts,balttl,mxuse,
     1            mxsoil,mb,Lutot,iophru,mxhru,numhru,idxhr1,idxhrn,
     1            idxsol,idxuse,idxaqf,shed,frcirr(iy),aqfbas,
     1            crpfrc,cropwt,crpsum,soilwt,hruwts,sumwts,hrubas)
                                                                      args
                call PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,     6
     1            depmpy,cnvvol,bsarea,mxcomp,msubo,mb,Lutot,idxsub,     8
     1            ssub,shed,sw,v,swprev,vprev,iaqufr,iophru,             8
     1            ioshl,evtgw,egwuna)                                    3

                if (iopswt.gt.0) call MODSTP (iopswt,iopmod,delt,
     1              nbyr,kper,nstp,kstp,iyr,bsarea,mb,shed)
              end if
```

Annual.h: summarize annual results for Modflow (iopmod=1)

```
c begin file annual.h, included in Swatmain: annual fluxes --spp
      if (iopmod.eq.1) then
        ndays=ndayr(idleap)    !time step (sec) to be passed to MODFLOW
        write (balttl,'(i5,5x,i5)') iyr,ndays      !char balttl*15
c         !annual basin summary smy and index idxssb are not passed to Preswb
c
c Compute weights to be used in SWBAVG to take weighted average over HRUs:
        if (numhru.gt.1) call Subwts (iowts,balttl,mxuse,mxsoil,
     1    mb,Lutot,iophru,mxhru,numhru,idxhrl,idxhrn,idxsol,idxuse,
     1    idxaqf,shed,frcirr(iy),aqfbas,crpfrc,cropwt,crpsum,soilwt,
     1    hruwts,sumwts,hrubas)

        call PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,
     1      depmpy,cnvvol,bsarea,mxcomp,msubo,mb,lutot,idxsub,
     1      sysub,shed,sw,v,swprev,vprev,iaqufr,iophru,
     1      ioshl,evtgw,egwuna)

        if (iopswt.gt.0) call MODSTP (iopswt,iopmod,delt,
     1      nbyr,kper,nstp,kstp,iyr,bsarea,mb,shed)
      end if
```

Avgann.h: summarize simulation results for Modflow (iopmod=0)

```
c begin file avgann.h, included in Swatmain: avg annual fluxes --spp

c ---- write balance for the Swat-Modflow connection (device iobal):
      write (balttl,'(i5,1h-,i4)') iyrbgn,iyrend      !char balttl*15
c         !avg annual basin summary sm and index idxssb are not passed to Preswb

        call PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,
     1      depmpy,cnvvol,bsarea,mxcomp,msubo,mb,lutot,idxsub,
     1      stsub,shed,sw,v,swprev,vprev,iaqufr,iophru,
     1      ioshl,evtgw,egwuna)

c                 pass average annual fluxes between SWAT and MODFLOW:
      if (iopmod.eq.0.and.iopswt.gt.0) call MODSTP (iopswt,iopmod,
     1      delt,nbyr,kper,nstp,kstp,iyr,bsarea,mb,shed)
```

Subr. PRESWB: prepare Swat's results for SWBAVG and MODFLOW

```
c file preswb.for subroutines Preswb and Subwts spp
c
      subroutine PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,         6
     1  depmpy,cnvvol,bsarea,ncomp,msubo,mb,Lutot,idxsub,ssub,shed,     10
     1  sw,volpnd,swprev,vprev,iaqufr,iophru,ioshl,evtgw,egwuna)         9

c usage (monthly time steps):
c                call PRESWB (iopswt,iobal,iosum,balttl,ndays,cnvlen,    6
c     1            depmpy,cnvvol,bsarea,mxcomp,msubo,mb,Lutot,idxsub,    8
c     1            ssub,shed,sw,v,swprev,vprev,iaqufr,iophru,            8
c     1            ioshl,evtgw,egwuna)                                   3
c
c Preswb: prepare Swat's hydrologic results as flow rates to be either
c written to a averaged by SWBAVGpassing write summary of hydrologic components
c to "balance" file to be read by MODSWB package in MODFLOW.

c argument list declarations and definitions:
      integer iopswt !Swat-Modflow linking mode:
c       0 (separate): Swat writes balance files for SWBAVG, which writes
c         file of averaged results for Modflow to read;
c       1 (linked): Swat calls Modflow as a subroutine.
```

```
      integer iobal !output device no. for balance file
      integer iosum !output device no. for summary file
      character*(*) balttl !15-char title string;
c         time step description: yr, mo, days (3i5)
      integer ndays !input: no. days in aquifer time step.
      real cnvlen    !input: length conversion factor from Modflow's std. unit
c         to Swat's std. unit (m); defined by input to  *.cod.
c         If Modflow's data set uses feet, cnvlen = 0.3048 (m/ft).
c         If Modflow's data set uses meters, then cnvlen = 1.

      real depmpy    !conversion from Swat's units of hydrologic depths (mm),
c       to std. length unit (m): depmpy = 1e-3 (m/mm), defined in Swatmod2.h.

      real cnvvol    !conversion from Swat's units for pond volumes (cubic
c       meters) to Modflow's volume units; given on included file swatmod2.h by
c                 cnvvol = cnvlen*cnvlen*cnvlen.

c         Note 1. depmpy and widmpy  are defined on the included file
c       swatmod2.h, where cnvlen and other added data are read from the *.cod
c       input file.  widmpy, not used in Preswb, converts Swat's length units
c       used for basin area (km) to std. length unit (m): widmpy = 1000 (m/km)
c       to give bsarea (see below).

c         Note 2. Swat expresses some other areal quantities as hectares
c       (1 ha = 10^4 m^2 = (100 m)^2, or 1 km^2 = 100 ha); but none of these
c       quantities is involved in the present linkage.
c
c         Note 3. Hydrologic depth is accumulated water volume per unit subbasin
c       area given by d = Q*delt/Asub, where Q*delt is accumulated water volume,
c       the product of mean flow rate Q and aquifer time step delt, and
c       Asub is subbasin area.

      real bsarea     !basin area in Modflow's units, given on included file
c       swatmod3.h using cnvlen and widmpy to convert da (km^2) according to
c            bsarea = da*(widmpy/cnvlen)**2

c         Input arguments set in Swatmod1.h:
      integer ncomp !no. hydrologic components from Swat used in linkage.
      integer msubo !no. vectors in ssub; param. def. on included file common.f
      integer mb    !max. no. subbasins; parameter defined on file common.f
      integer Lutot !no. subbasins for this case; specified on *.cod input file.
      dimension idxsub(ncomp)   !index to vector in SWAT subbasin array ssub
ccccc dimension idxssb(ncomp)   !index to vector in SWAT basin array ssb
c       EXCEPT for icomp = 2 (none given for irrigation).  Not currently used;
c       basin-wide summaries are given by adding converted flow rates
c       (see shed) over subbasins.
c
c       Arrays of hydrologic components given by Swat's simulation for this
c       time step:
ccccc dimension ssb(109)        !SWAT basin array (basin-wide avg results)
c         for subbasin L=1 to Lutot = no. subbasins:
      dimension ssub(msubo,mb) !SWAT subbasin array (msubo=60 vectors)
c              contains simulation results as depths (mm), i.e., volumetric
c              content per unit area of subbasin.

c         Note on locations of hydrologic components icomp = 1 to 13 in
c         Modflow's Shed array (passed from Preswb as shed):
c     dimension idxmod(ncomp)   !index to vectors in MODFLOW subbasin array SHED
c        (not defined here; instead, vectors in Shed are indexed (below) by
cc           i = idx + 9 !index to shed vectors 10-22 for idx from 1 to 13.
c
c       Array of hydrologic results converted to flow rates in Modflow's units
c       to be either written to balance file or passed to Modflow (30 vectors
c       for each subbasin):
      dimension shed(30,0:mb)
      dimension swtflx(30)  !basin-wide summary of flow rates in shed.

ccc   dimension flu(lutot)      !subbasin area as fraction of basin
ccc                         (given by shed(1,j); assigned in Main(swatmod3.h)
ccc   dimension fp(lutot)       !noncontributing areal fraction of subbasin
ccc                         (given by shed(8,j); assigned in Main(swatmod3.h)
ccc   dimension aqfsub(lutot)  !areal fraction of each subbasin with aquifer
ccc                     (given by shed(26,j); assigned in Main(swatmod3.h)
      dimension sw(mb)       !soil water depth (mm)
      dimension volpnd(mb)  !pond volume (cubic meters)
      dimension swprev(mb)  !soil water depth at beginning of time step
      dimension vprev(mb)  !pond volume at beginning of time step

      integer iophru !option for one of three schemes:
c               =1: original 18-HRU scheme for 6 soils and 3 land use types
c                     (dryland crop, irrigated crop, and grassland);
c               =2: 2 possible schemes, depending on how soils are handled.
c                 Land use definitions are the same for both of these:
c                (c1: nonalluvial dryland crops; c2: alluvial irrigated crops;
c                (c3: alluvial grassland; c4: nonalluvial grassland)
c                a) 24-HRU scheme for 6 soils and 4 land use/landform types:
c                b) 10-HRU scheme taking into account soil and land use
c                 dependencies.  Alluvial land uses (irrigated corn and
c                 grasslands) are associated only with alluvial soils (1:Carr
c                 and 2:Muir); upland land uses (dryland crop rotation and
c                 grassland) are associated only with non-alluvial soils
c                 3:Crete, 4:Kips, and 5:(Hastings and Hedville).  Hastings
c                 and Hedville are combined into one HRU soil factor with
c                 Hastings soils only in subbasins 1-4 and Hedville soils in
c                 subbasins 6-9.
c                =3: refinement of the above scheme, dividing the alluvial
c                 aquifer area into shallow and deep components; the shallow
c                 component is associated with evaporation from gw according
c                 to Modflow's simulation.  For iophru=3, this is read from
c                 file device ioshl at the end of this subroutine for the
c                 following time step; values for initial time step are read
c                 in swatmod3.h (included in main).  Evaporation, given by
c                 Modflow results as a flow rate, is converted to a hydrologic
c                 depth, evtgw, with respect to the shallow aquifer area.

      integer iaqufr !indicates (y>0,n=0) that soil water can percolate
c        out of the soil profile to an underlying aquifer; otherwise, the
c        soil profile is assumed to be underlain by bedrock, blocking
c        percolation below soil.

c        evap from gw is zero except for hrus with shallow aquifer, which
c        are distinguished for the third hru scheme (option iophru=3).
c
c        For the three HRU options, the input option iaqufr indicates
c        aquifer status as follows:
c    iaqufr =0: soil is underlain by bedrock, not by an aquifer, so that
c               percolation stops at the bottom of the soil profile;
c    iaqufr =1: soil is underlain by a deep aquifer, i.e., one that does
c               not contribute to plant or soil water uptake; aquifer areal
c               fraction of subbasin is read from the *.cod input file.
c    iaqufr =2: soil is underlain by a shallow aquifer; plant or soil water
c               uptake is based on Modflow's model of evapotranspiration
c               from shallow ground water.  For option iophru=3, Modflow's
c               results for areal fractions and evapotranspiration are read
c               by Swat in each time step.
c
      dimension evtgw(mb), egwuna(mb)
c
c internal definitions for conceptual models for recharge and tributary inflow:

      dimension qrech(100) !recharge: sum of (perc, xmloss, pond seepage)
      dimension qtrib(100) !tributary flow: sum of (surface runoff, lateral subsurface
flow)
c
      delt = 86400.*FLOAT(ndays)         !time step (sec) for Modflow
      covdt = (depmpy/cnvlen)/delt   !convert from [L](mm) to [L/T] (ft/sec)
c
      tir=0.
      swbasn=0.
      voltot=0.
      do j=1,lutot
        voltot = voltot + volpnd(j)       !basin pond volume
```

```
         swbasn = swbasn + shed(1,j)*sw(j) !basin soil water depth
         tir = tir + shed(1,j)*ssub(22,j)     !basin irrigation depth
       end do
c
c              hydrologic results are passed from watershed simulator,
c              subr Preswb, by call to Modflow via entry Modstp as follows
       swtflx(5)=0.
       swtflx(6)=0.
       do i=10,22
         swtflx(i)=0.
       end do

c      recharge: sum of xmloss, perc, and pond seepage (Preswb).
c              vertical component: with underlying aquifer (fraction frcaqf);
c              lateral component: without underlying aquifer (fraction 1-frcaqf),
c                  add to tributary flow.
c      tributary flow: sum of surface & subsurface runoff + (1-frcaqf)*Qrech
c              areal fraction of subbasin with underlying aquifer (Swb2rp):
c
       qtrbas=0.
       do j=1,Lutot
         if (iophru.eq.1) then
c            recharge = xmloss(11) + percolation(13) + pond seepage(16)
c            (applied to fraction, aqfsub, of subbasin with underlying aquifer)
           aqfsub = shed(7,j)
           qrech(j) = aqfsub*(ssub(11,j) + ssub(13,j) + ssub(16,j))
c
c            tributary outflow = surface runoff(4) + subsurface lateral flow(5)
c               + pond outflow(20) + [recharge components outside aquifer area]
           qtrib(j) = ssub(4,j) + ssub(5,j) + ssub(20,j)
     1        + (1.-aqfsub)*(ssub(11,j) + ssub(13,j) + ssub(16,j))
c
c            Notes:
c            a)  This scheme previously did not include the last term
c              shown for tributary inflow, qtrib(j), corresponding to recharge
c              components outside the aquifer area.  Its inclusion is somewhat
c              reasonable but not entirely satisfactory since it is based on
c              a hydrologic model that assumes an underlying aquifer for the
c              entire subbasin.  Alternatives have been implemented to
c              distinguish whether the soil profile is underlain by bedrock
c              or an aquifer (iophru=2), and whether the aquifer is shallow
c              or deep (iophru=3).
         else
c             HRU schemes 2 and 3 handle aquifer areal fraction as HRU weight:
c           recharge = xmloss(13) + percolation(13) + pond seepage(16);
c           tributary outflow = surface runoff(4) + subsurface lateral flow(5):
c                             + pond outflow(20)
c           (note: seepage from wetlands, ssub(55,j), is another possible
c            source of ground water recharge; and wetlands outflow, ssub(59,j),
c            is another possible source of tributary flow.)

           qrech(j) = ssub(11,j) + ssub(13,j) + ssub(16,j)
           qtrib(j) = ssub(4,j) + ssub(5,j) + ssub(20,j)

           if (iaqufr.eq.0) then
             qtrib(j) = qtrib(j) + qrech(j) !no aquifer
             qrech(j) = 0.
           end if
         end if
         qtrbas = qtrbas + shed(1,j)*qtrib(j)

         caovdt = covdt*bsarea*shed(1,j)  !convert from [L](mm) to [L^3/T] (cfs)
c
c                 shallow aquifer as areal fraction of BASIN:
         shlfrc = shed(1,j)*shed(7,j)*shed(26,j)  !subfrc*aqffrc*shlfrc
         if (iaqufr.eq.2 .and. shlfrc.gt.0. .and. egwuna(j).gt.0.) then
c            convert undistributed evap from shallow gw back to flow rate:
           cnvfac = covdt*shlfrc
           egwuna(j) = egwuna(j)*cnvfac  !undistributed et-gw from gw [L^3/T]
         end if
c          convert storage components to rates of change in storage:
```

```
c             rate of change in pond volume: convert m^3 to [L^3/T]
        shed(5,j) = (volpnd(j) - vprev(j))/(cnvvol*delt)
        swtflx(5) = swtflx(5) + shed(5,j)
c             rate of change in soil water: convert mm to [L^3/T]
        shed(6,j) = (sw(j) - swprev(j))*caovdt
        swtflx(6) = swtflx(6) + shed(6,j)
c           retain pond volume and soil water content for next time step:
        vprev(j) = volpnd(j)
        swprev(j) = sw(j)

        do idx=1,13
          i = idx + 9                !index to shed vectors 10-22
          if (i.eq.12) then
c           include evaporation components from both land(12) and ponds(15):
            shed(i,j) = (ssub(12,j) + ssub(15,j))*caovdt
          else if (i.eq.17) then                ! (idx=8)
            shed(i,j) = qrech(j)*caovdt  ! Recharge
          else if (i.eq.18) Then
            shed(i,j) = evtgw(j)*caovdt  ! evap from gw (Modflow results)
          else if (i.eq.22) then              ! (idx=13)
            shed(i,j) = qtrib(j)*caovdt  !tributary flow from subbasin
          else                       !for idx from 1 to 12
            iswt=idxsub(idx) !index to SWAT vectors (see idxsub in ModMain)
            shed(i,j) = ssub(iswt,j)*caovdt  !other component fluxes
          end if
          swtflx(i) = swtflx(i) + shed(i,j)
        end do    !i=10,22
      end do        !j=1,Lutot

      if (iopswt.eq.0) then

c           balance file (flow rates):
        do j=1,Lutot
          write (iobal,'(a15,f15.0,i5,16f10.2)') balttl,delt,j,
     1       (shed(i,j),i=10,22),shed(6,j),shed(5,j),egwuna(j)
        end do    ! j = 1, Lutot

c           summary file (flow rates):
        write (iosum,'(a15,f15.0,5x,15f10.2)') balttl,delt,
     1       (swtflx(i),i=10,22),swtflx(6),swtflx(5)
      end if      ! (iopswt .eq. 0)

c       set up evaporation from gw (evtgw) for each subbasin based on
c     Modflow's solution for the next time step.
c       Note: Only HRUs specified as having shallow gw, i.e., iophru=3
c     and iaqufr=2, have evaporation from shallow gw that is to be
c     redistributed over the soil profile in the next time step.
c       But this is also read for HRUs with deep gw that are part
c     of the same scheme, i.e., iophru=3 and iaqufr=2, in order to
c     get the correct areal fraction corresponding to deep aquifer,
c     given by aqfrac = 1. - shed(26,j) for subbasin j; see Preswb.

      data initlz /0/
        if (initlz.eq.0 .and. iophru.eq.3)
     1    print '(1x,a)','Preswb: evt-gw for next time step'
        if (iopswt.eq.0 .and. iophru.eq.3 .and. iaqufr.gt.0) then
          do j=1,Lutot
            read (ioshl,'(6i5,5f10.0)',iostat=iostat,err=100,end=100)
     1        idxyr,idxper,idxstp,isub,ngract,ngrshl,
     1        shed(26,j),shed(7,j),shed(18,j),shed(28,j),shed(29,j)
c           shlfrc = shallow aquifer as areal fraction of BASIN:
            shlfrc = shed(1,j)*shed(7,j)*shed(26,j)  !subfrc*aqffrc*shlfrc
            if (iaqufr.eq.2 .and. shlfrc.gt.0.) then
c               integrate flow rate over one day and divide by area of
c               shallow aquifer.
              cnvfac = (cnvlen/depmpy)*86400./(shlfrc*bsarea)  ![L^3/T] to mm
              evtgw(j) = cnvfac*shed(18,j)     !daily evap from gw (mm)
            else
              evtgw(j) = 0.
            end if
            egwuna(j) = 0. !initialize remaining undistributed evtgw(j)
            data fshall /1./ !temporary; also set in Swatmod1.h to initialize.

c             fshall, set above and in swatmod1.h, is normally 1;
c             set to zero to see if HRU scheme 3 reduces to HRU scheme 2.
c             Artificially reduce shallow fraction, shed(26,j):
            if (fshall.lt.1.) then
                shed(26,j) = fshall*shed(26,j)
                evtgw(j) = fshall*evtgw(j)
            end if

            if (initlz.eq.0) print '(6i5,2f10.7,4f10.2,f10.7,2e15.7)',
     1        idxyr,idxper,idxstp,j,ngract,ngrshl,
     1        shed(26,j),shed(7,j),shed(18,j),shed(28,j),shed(29,j),
     1        evtgw(j),shed(1,j),shlfrc,cnvfac
          end do
100       if (iostat.ne.0) print '(1x,a)','Preswb: end of '//
     1      'evap-gw results from Modflow; balttl='//balttl
        end if
        if (initlz.eq.0) initlz=1
      RETURN
      end
```

Subr. SUBWTS: calculate HRU weights for each subbasin

```
subroutine Subwts (iowts,balttl,mxuse,mxsoil,mxsub,nwshed,
     1    iophru,mxhru,numhru,idxbgn,idxend,idxsol,idxuse,idxaqf,
     1    shed,frcirr,aqfbas,crpfrc,cropwt,crpsum,soilwt,hruwts,
     1    sumwts,hrubas)
c
c calculate HRU weights for each subbasin.
c
      integer iowts !weights output file device no.; write only if iowts > 0.
      character*(*) balttl !15-char title string;
c         time step description: yr, mo, days (3i5)

      integer mxuse  !max. no. land use/landform types to be distinguished;
c                     see description of HRU schemes 1-3 for details.
      integer idxsol(mxhru)    !soil type id for each HRU
      integer idxuse(mxhru)    !land use id for each HRU
      integer idxaqf(mxhru)    !aquifer status id for each HRU
      real frcirr     !irrigated areal fraction of basin for current year.
      real aqfbas     !areal fraction of basin underlain by aquifer
      dimension crpfrc(mxsub) !areal fraction of cropland in each subbasin
c         land use and soil type factors for HRU weights:
      dimension cropwt(mxsub,mxuse)  !areal fractions for up to 4 land uses
      dimension soilwt(mxsub,mxsoil) !areal fractions for each soil by subbasin

      integer idxbgn, idxend !use to calculate weights for a selected range
c       of HRUs.  If idxbgn =1 and idxend=numhru, then weights for all HRUs
c       are calculated.  Select only one HRU to calculate weights by setting
c       idxbgn = idxend = idxhru as specified by input to the codes input file.
c       In calling routine (Swatmain, code on either swatmod3.h or at end of
c       each time step, e.g. monthly.h).
c         if idxhru = 1, set idxbgn=1 and idxend=numhru to calculate all weights
c            and write to weights file;
c         if idxhru > 1, set idxbgn = idxend = idxhru to calculate weights for
c            only the HRU being simulated by SWAT.
c
      integer isolhr !index to soil type associated with this HRU (idxhru).
      integer iusehr !index to land use (or crop rotation) assoc. w/ the HRU
c       Note: isolhr and iusehr are values from index arrays idxsol and idxuse
c       that identify the soil type and land use associated with the HRU idxhru;
c       these associations are read from the extended *.cod input file if
c       numhru > 1.

c     land use definitions for three HRU schemes (iophru=1 to 3):
```

```
c 1 (numuse=3): land use and soil type are assumed to be independent.
c      iuse=1:non-irrigated cropland; 2:irrigated cropland; 3:grassland.

c 2 (numuse=4): Alluvial valley and non-alluvial uplands are disaggregated.
c     Soil dependencies:
c   Carr and Muir soils are alluvial soils; the other four are in the uplands.
c     Land use dependency:
c   Irrigated cropland is assumed to be restricted to the alluvial valley,
c   and non-irrigated cropland to the uplands.

c 3 (numuse=6): Alluvial valley with underlying aquifer is further
c   distinguished by shallow and deep components.
c
      dimension shed(30,0:mxsub)

      dimension crpsum(nwshed)
      dimension hruwts(mxsub,mxhru) !areal weight corresponding to this HRU
c       (idxhru), given by product of soilwt(isolhr) and cropwt(iusehr).
      dimension sumwts(mxsub), hrubas(mxhru)

c       model assumption: irrigated cropland is within alluvial valley
c       with underlying aquifer.

      aqfrat = frcirr/aqfbas
c       Calculate land use weight components for each subbasin.
      do j=1,nwshed
        if (iophru.eq.1) then
          numuse=3 !number of land use types
          cropwt(j,1) = crpfrc(j) - frcirr   !non-irrigated cropland
          cropwt(j,2) = frcirr                   !irrig. cropland
          cropwt(j,3) = 1. - crpfrc(j)           !grassland
        else
          numuse=4 !number of land use types
          aqfsub = shed(7,j)          !alluvial fraction of subbasin area
          aqfirr = aqfrat*aqfsub
          cropwt(j,2) = aqfrat*aqfsub           !alluvial irrig. cropland
          cropwt(j,3) = aqfsub - cropwt(j,2)    !alluvial grassland
          cropwt(j,1) = crpfrc(j) - cropwt(j,2)  !no aquifer, nonirrig. crops
          cropwt(j,4) = 1. - aqfsub - cropwt(j,1) !no aquifer, grassland

          if (iophru.eq.3) then

            numuse=6 !revise number of land use types

c           Revise weights to distinguish shallow and deep aquifer components.
c           First define shallow aquifer fractions as components 5 and 6;
c           then redefine components 2 and 3 as deep aquifer.

            shalfr = shed(26,j)         !shallow fraction of aquifer area
            cropwt(j,5) = cropwt(j,2)*shalfr  !shallow aquifer, irrig. crops
            cropwt(j,6) = cropwt(j,3)*shalfr  !shallow aquifer, grassland

            deepfr = 1. - shed(26,j)    !deep fraction of aquifer area
            cropwt(j,2) = cropwt(j,2)*deepfr  !deep aquifer, irrig. crops
            cropwt(j,3) = cropwt(j,3)*deepfr  !deep aquifer, grassland
          end if
        end if
        crpsum(j)=0.
        do iuse=1,numuse
          crpsum(j) = crpsum(j) + cropwt(j,iuse)
        end do
      end do
c
c        Initialize check on sum of HRU weights, which should add to 1:
      do j=1,nwshed
        sumwts(j)=0.
      end do

      if (idxbgn.eq.1 .and. idxend.eq.numhru) then
        do idx=1,numhru
          hrubas(idx)=0.
          end do
        end if

c           Calculate HRU weights as products of areal fractions for soil type,
c           isolhr, and land use, iusehr, according to one of the HRU schemes
c           1-3 as defined above.

        do idx=idxbgn,idxend
          isolhr = idxsol(idx)  !soil type id for this hru
          iusehr = idxuse(idx)  !land use id for this hru
          iaqufr = idxaqf(idx)  !aquifer status id for this hru

          do j=1,nwshed
            hruwts(j,idx) = soilwt(j,isolhr)*cropwt(j,iusehr) ! HRU weights
c
c               sum of HRU weights for each subbasin (should add to 1):
            sumwts(j) = sumwts(j) + hruwts(j,idx)
c
c               average over basin for each HRU:
            if (idxbgn.eq.1 .and. idxend.eq.numhru)
     1        hrubas(idx) = hrubas(idx) + shed(1,j)*hruwts(j,idx)
          end do   !j=1,nwshed    (for each subbasin)
        end do     !idx=1,numhru  (for each HRU)

c           Write table of HRU weights to file for reference in SWBAVG.
        if (iowts.gt.0) then
          do j=1,nwshed
            write (iowts,'(a,i5,26f10.7)') balttl,j,
     1        (hruwts(j,idx),idx=1,numhru),sumwts(j),crpsum(j)
          end do
        end if   !iowts > 0
        return
        end
```

SWBAVG: take average over HRUs simulated by SWAT.

Notes: (a) Included files common.f, swatmod1.h, and swatmod2.h are not listed here; see their listings above. (b) Compiler warnings for variables declared in common.f and Swatmod1.h but not used in Swbavg are not shown either.

```
c file swbavg.for: Take weighted average over HRUs given by results on
c balance files corresponding to HRU cases and associated soil type and land
c use areal fractions defined on Swat's extended *.cod input file, which is
c also read by Swbavg to direct the averaging.
c
c Note: The code for this routine is closely related to code added to Swat
c for the Swat-Modflow linkage.  Included files common.f, swatmod1.h, and
c swatmod2.h (below) are also included in the modified Swat mainline.  common.f
c is a component of Swat v.94.  swatmod1.h makes the data declarations for the code added
for
c the Swat-Modflow linkage, and swatmod2.h reads Swat's modified *.cod input
c file.
c
        program Swbavg
        include 'common.f'  !original source for Swat

        parameter (iswbav=1) !indicates that code on swatmod2.h has not been included in
c                 Swat, but rather Swbavg (iswbav=1) to skip execution of
c                 file initializing code for Swat.
        include 'swatmod1.h' !also included in Swat's mainline
        character*20 hrufil(50),filwts,fillog,filout,filsum,fildep
        character*132 headwt
        dimension inpfil(50)
        dimension flux(15,100),
     1  flxavg(15,100), swavg(100), pvavg(100), wtsum(100),
     1  flxbas(15),depbas(15)
```

```
c       read the modified Swat Control Codes input file as std input.
      incode=5 !device no. to read *.cod input file from std input (keyboard)
      read (incode,'(a)') recinp(1:80)                   !read line 1 of *.cod
      print '(a)', ' '//recinp(2:78)
      read(incode,10200) nbyr,iyr,lu,ipd,nsim,msim,ign,iwst,isst,ires,
     1    igraf,irain,itemp,iresq,idaf,idal,iprn,iprp,iopt,ipet   !line 2
10200  format(20i4)
      print '(20i4)', nbyr,iyr,lu,ipd,nsim,msim,ign,iwst,isst,ires,
     1    igraf,irain,itemp,iresq,idaf,idal,iprn,iprp,iopt,ipet
      iyrbgn = iyr
      iyrend = iyr + nbyr-1
      include 'swatmod2.h' !also included in Swat's mainline to read remainder of *.cod
c
      do i=1,numhru
c       Balance and summary output file names
c       (files opened in Swatmod3.h):
        idxdot=INDEX(namhru(i),' ')
        if (idxdot.gt.1) then
          hrufil(i) = namhru(i)(1:idxdot-1) // '.bal'
        else
          hrufil(i) = namhru(i) // '.bal'
        end if
        inpfil(i) = 10 + i
        print '(4i5,2(1x,a))', idx, idxsol(i), idxuse(i), idxaqf(i),
     1                        namhru(i),hrufil(i)
        end do
c-------------------------------------------------------------------
      do i=1,numhru
        open (unit=inpfil(i),file=hrufil(i),status='unknown')

c written in Preswb:
cc      write (iobal,'(2i5,f15.0,f10.7,2i5,1x,a)') nbyr, iopmod,
cc   1    bsarea, fpd, idxhru, numhru, namcas

c read in Swbavg:
        read (inpfil(i),'(2i5,f15.0,f10.7,2i5,1x,a)') nyr, iopmod,
     1      bsarea, fpd, idxhru, nhru, namcas
        read (inpfil(i),'(a)') recinp(1:81)
      end do
c       Set up output file names based on case name, casavg, for HRU average:
      idxdot=INDEX(casavg,' ')
        if (idxdot.gt.1) then
          fillog = casavg(1:idxdot-1) // '.log'
          filout = casavg(1:idxdot-1) // '.bal'
          filsum = casavg(1:idxdot-1) // '.sum'
          fildep = casavg(1:idxdot-1) // '.dep'
          filwts = casavg(1:idxdot-1) // '.wts'
        else
          fillog = casavg // '.log'
          filout = casavg // '.bal'
          filsum = casavg // '.sum'
          fildep = casavg // '.dep'
          filwts = casavg // '.wts'
        end if

      iowts = 1
      print '(1x,a)','weights input file = '//filwts
      print '(1x,a,i3)','iowts=',iowts
      open (unit=iowts,file=filwts,status='unknown')
      read (iowts,'(a)') headwt
      iolog = 7
      print '(1x,a)','weights summary file = '//fillog
      print '(1x,a,i3)','iolog=',iolog
      open (unit=iolog,file=fillog,status='unknown')
      iobal = 8
      print '(1x,a)','output balance file = '//filout
      print '(1x,a,i3)','iobal=',iobal
      open (unit=iobal,file=filout,status='unknown')

      write (iobal,'(2i5,f15.0,f10.7,2i5,1x,a)') nbyr, iopmod,
     1    bsarea, fpd, idxhru, numhru, casavg
      write (iobal,'(a)') recinp(1:81)
      print '(2i5,f15.0,f10.7,2i5,1x,a)', nbyr, iopmod,
     1    bsarea, fpd, idxhru, numhru, casavg
c         initialize basin-wide summary output file:
      iosum = 9
      print '(1x,a)','output summary file = '//filsum
      print '(1x,a,i3)','iosum=',iosum
      open (unit=iosum,file=filsum,status='unknown')
      iodep = 10
      open (unit=iodep,file=fildep,status='unknown')

      print '(a)', ' '//recinp(2:81)
      print '(5(1x,a,i5))','numhru=',numhru,'Lu=',Lu
      do j=1,lu
        swavg(j) = 0.
        pvavg(j) = 0.
        wtsum(j) = 0.
      end do
      do i=1,numhru
        do j=1,lu

c written in Preswb:
cc        write (iobal,'(i5,3f10.7,2f10.4,2i5,2(2x,a8))') j,flu(j),
cc   1        fp(j),hruwt(j),soilwt(j,isolhr),cropwt(iusehr),

c lines 1-3 of carr-irm.bal:
c  18    2   27658944500. 0.0199367    2   24 carr-irm
c sub sub fract noncontrb HRU fract    soilwt    cropwt soil Luse soil name  land use
c   1 0.2201500 0.0343000 0.0016744    0.0920    0.0182    1    2  Carr       irrig

c read in Swbavg:
          read (inpfil(i),'(i5,5f10.0,2i5,2(2x,a8))')
     1        idx,flu(j),fp(j),hruwts(j,i),solwt,crpwt,isol,iuse
          write (iolog,'(i5,3f10.7,2f10.4,3i5,2(2x,a8))')
     1        idx,flu(j),fp(j),hruwts(j,i),solwt,crpwt,isol,iuse,i
          wtsum(j) = wtsum(j) + hruwts(j,i)        !sum of weights over all HRUs for
subbasin j
        end do
        read (inpfil(i),'(a)') hdrout
      end do
      print '(a5,25(2x,a8))',' sub','   total',
     1        (namhru(i),i=1,numhru)
      do j=1,Lu
        write (iobal,'(i5,5f10.7,2i5,2(2x,a8))')
     1        j,flu(j),fp(j),wtsum(j)
cc      write (iolog,'(i5,5f10.7,2i5,2(2x,a8))')
cc   1        j,flu(j),fp(j),wtsum(j)
cc      print '(i5,25f10.7)', j, sumwts(j),(hruwts(j,i),i=1,numhru)
      end do

      write  (iobal,'(a)') hdrout  !header for results [L^3/T] each step, subbas
      write  (iosum,'(a)') hdrout  !header for results [L^3/T] each step, basin
      write  (iodep,'(a)') hdrout  !header for results (mm) each year, basin
      nstep=0
      iostat=0
      do while (iostat.eq.0)
        nstep=nstep+1
        do j=1,Lu
          swavg(j) = 0.
          pvavg(j) = 0.
          do k=1,15
            flxavg(k,j)=0.
          end do
        end do
        do j=1,Lu
          read (iowts,'(a15,i5,26f10.7)',iostat=iostat,end=100,err=100)
     1        balttl,isub,(hruwts(j,idx),idx=1,numhru),sumwts(j)
        end do

        do i=1,numhru
```

```
          do j=1,Lu

c written in Preswb:
cc        write (iobal,'(a15,f15.0,i5,15f10.2,f10.7)') balttl,delt,j,
cc   1      (shed(i,j),i=10,22),shed(6,j),shed(5,j),hruwt(j)

c read in Swbavg:
            read (inpfil(i),'(a15,f15.0,i5,15f10.2,f10.7)',
     1        iostat=iostat,end=100,err=100) balttl,delt,idx,
     1          (flux(k,j),k=1,15)
            do k=1,15
              flxavg(k,j) = flxavg(k,j) + hruwts(j,i)*flux(k,j)
            end do
          end do
        end do
c
        do j=1,Lu
c written in Preswb:
cc        write (iobal,'(a15,f15.0,i5,15f10.2,f10.7)') balttl,delt,j,
cc   1      (shed(i,j),i=10,22),shed(6,j),shed(5,j),hruwt(j)

c written in Swbavg:
          write (iobal,'(a15,f15.0,i5,15f10.2,f10.7)') balttl,delt,j,
     1        (flxavg(k,j),k=1,15),sumwts(j)
        end do
c
c         basin-wide summary:
        do k=1,15
          flxbas(k)=0.
        end do
        do j=1,Lu
          do k=1,15
            flxbas(k) = flxbas(k) + flxavg(k,j)
          end do
        end do
        write (iosum,'(a15,f15.0,5x,15f10.2)') balttl,delt,
     1      (flxbas(k),k=1,15)
        read (balttl,'(2i5)') iyr,month
c
c annual summary of hydrologic fluxes as depths:
        do k=1,15
          depth = flxbas(k)*delt/bsarea
          if (month.eq.1) then
            depbas(k) = depth
          else
            depbas(k) = depbas(k) + depth
          end if
        end do
        if (iopmod.eq.1 .or. iopmod.eq.2.and.month.eq.12)
     1    write (iodep,'(i5,10x,f15.0,5x,15f10.3)') iyr,delt,
     1      (depbas(k),k=1,15)
      end do
100   print '(2(1x,a,i5))','Swbavg: completed averages for',
     1                      nstep,'steps.'
      stop
      end
```

MODMAIN: stub mainline to call Modflow (file modmain.for)

```
c file modmain.for  run modflow as a stand-alone program.
c
      program modmain
c
      integer iopswt
      parameter (mxshed=100)
      dimension shed(30,0:mxshed) !passed to MODFLOW via call to entry MODSTP
ccc   integer idxsub(12)

c         locations in array SHED for hydrologic fluxes (Modswb packag3):
ccc   data (idxsub(j),j=1,12) /10,11,12,13,14,15,16,17,18,19,20,21/
c         locations in array SSUB for hydrologic fluxes (passed from Swat):
cSWAT:data (idxsub(j),j=1,12) /1,22,12,4,13,5,11,9,7,6,16,25/
ccc   data iopswt/0/ !set option to run modflow as stand-alone program.
c
c file modflo.for: main subroutine for MODFLOW to be run either stand-alone
c (called by modmain on modmain.for with iopswt=0) or under the time-s:ep
c control of a calling routine (in this case the mainline for SWAT, '94 version,
c with iopswt=1).       --spp Apr 21 95
c
c     --also with diffusive wave stream router that reads streambed hydraulic
c    conductivity rather than streambed conductance.   --spp jul 20 93
c------------------------------------------------------------------------
c          added arguments: msubo, mb, shed(msubo,mb)

      call MODFLO (iopswt,iopmod,ndays,delt,nper,kper,
     1             nstp,kstp,ibalyr,areain,mxshed,shed)
      stop
      end
```

Subr. MODFLO: main rewritten as a subroutine (file modflo.for)

```
c file modflo.for: main subroutine for MODFLOW to be run either stand-alone,
c called by a stub mainline on modmain.for (iopswt=0); or under the tine-step
c control of a calling routine (iopswt=1) as described below for SWAT v. '94
c        --spp Apr 21 95
c------------------------------------------------------------------------
      subroutine MODFLO (iopswt,iopmod,ndays,delt,nper,kkper,
     1                   nstp,kkstp,icalyr,areain,mxshed,shed)

c usage in Modflow mainline:
cc    call MODFLO (iopswt,iopmod,ndays,delt,nper,kper,
cc   1             nstp,kstp,ibalyr,areain,mxshed,shed)
c
c usage in Swat mainline:
c (a) call main entry point for each stress period:
c     if (iopswt.gt.0) call MODFLO (iopswt,iopmod,delt,
c    1            nbyr,kper,nstp,kstp,iyr,bsarea,mb,fmodfl)
c
c (b) call second entry point at end of each time step:
c     if (iopswt.gt.0) call MODSTP (iopswt,iopmod,delt,nbyr,
c    1            kper,nstp,kstp,iyr,bsarea,mb,fmodfl)

c    this is MODFLOW's mainline, rewritten as a subroutine to allow
c the following option.
c----------------------------------------------------
c iopswt = controlling option to either run modflow stand-alone (iopswt=0)
c     or as a subroutine (iopswt=1), in this first case called by SWAT as
c     a coupled watershed-stream-aquifer model.
c
c Calling subroutines
c    Modmain, in stand-alone mode (iopswt = 0);
c    Swatmain, under control of watershed simulation code e.g. Swat (iopswt > 0).

c     Argument list (mod='*' indicates that modified argument is returned)
```

```
c     (type, name, dimensions, mod, units, definition)

      integer iopswt  ! option to run Modflow either of two ways:
c             =0: run Modflow in stand-alone mode by calling this from the
c  "stub" mainline on file Modmain.for.  In this case, the added entry point
c  Modstp() and the return that precedes it is bypassed.
c             >0: run Modflow under control of a watershed simulation program.
c
c Operation of Modflow under control of Swat (file swatmain.for):
c Note: To allow calls from Swat (or other code) to enter the loop on time steps
c (entry point Modstp), the loops have been replaced by equivalent but more
c primitive code using "go to" statements to circumvent compiler objections.
c
c DO for each stress period kper = 1 to nper (stress period = 1 year):
c   call Modflo() at beginning of stress period (file swatmod3.h)
c       On first call (initlz=0), initialize simulation.
c       Modflo calls standard RP* routines to initialize data for the
c           stress period, primarily water rights.
c       Return to Swatmain before entering loop on time steps.
c
c   DO for each aquifer time step kstp = 1 to nstp (step = day, month, or year):
c     Call Modstp() at end of time step (daily.h, monthly.h, or annual.h)
c         Call SWB2FM to specify conditions in terms of watershed simulation;
c         Enter iterative loop to call FM* routines and solve stream-aquifer system;
c         Return at end of loop on time steps.
c   end DO (for each time step)
c end DO (for each stress period, i.e., year)
c     --spp apr 21 1995

      integer kper   ! index to current stress period from 1 to nper
      integer nstp   ! no. time steps in current stress period
      integer kstp   ! index to current time step from 1 to nstp
      integer icalyr ! calendar year

      dimension shed(30,0:mxshed)  !array of watershed simulation results for
c                                  30 vectors by mb subbasins.
c Parameters and common blocks:
      integer mxlen
      parameter (mxlen=800000,mxstep=366)
      COMMON X(mxlen)
      COMMON /FLWCOM/LAYCON(80)

      real delt

C     MAIN CODE FOR MODULAR MODEL --     9/1/87
C          BY MICHAEL G. MCDONALD AND ARLEN W. HARBAUGH
C-----VERSION 1638 24JUL1987 MAIN1
C     ***************************************************************
C
C        SPECIFICATIONS:
C     ---------------------------------------------------------------
      CHARACTER*4 HEADNG,VBNM
      character*3 budlbl(9)  !initialize in POST1RP (label invoked packages)
      DIMENSION HEADNG(32),VBNM(4,20),VBVL(4,20),volnet(2,20),IUNIT(24)
c         volnet is now calculated in BAS1OT for L=1,msum as:
ccc       volnet(1,L) = vbvl(1,L) - vbvl(2,L)  !net volume (in-out)
ccc       volnet(2,L) = vbvl(3,L) - vbvl(4,L)  !net rate (in-out)
      dimension timstp(5),timper(5),timsim(5)
      dimension totinp(2),totout(2),diff(2),pdiff(2)
ccccc dimension strmin(365)  !monthly streamflow minimum for iadcod=12
c-----------------------------------------------------------------------------
c---------- Allow at least 20 hydrographs of mxstp time values each:
      parameter (mxser=20)
      logical logopn
c         !for each time step, option to write results in Surfer format (HYD1RP):
      dimension iopstp(mxstep)
      DOUBLE PRECISION DUMMY
*
        integer itbgn, itend, timsec
        character*30 filstm,filbal,namchr
        character chdate*8, chtime*11, case*8, chrext*3
        character*50 inpfil(0:20)
        character *3 inpext(0:20)
        character *30 inpdsc(0:20)
        character*60 outdsc(12)
        character*13 outfil(12)
        character*3 outext(12)
        integer iopost(12),iopout(12) !device numbers for std. and nonstandard output
files
        character *11 unffil(11)
        integer lstcbc(11),iundev(11)
c           Associate list with "CB" input variables as follows:
c           1) ihedun  2) iddnun  3) ibcfcb  4) iwelcb  5) idrncb  6) irchcb
c           7) ievtcb  8) irivcb  9) istcb1 10) istcb2 11) ighbcb
        equivalence           (lstcbc(1),ihedun), (lstcbc(2),iddnun),
     1    (lstcbc(3),ibcfcb), (lstcbc(4),iwelcb), (lstcbc(5),idrncb),
     1    (lstcbc(6),irchcb), (lstcbc(7),ievtcb), (lstcbc(8),irivcb),
     1    (lstcbc(9),istcb1), (lstcbc(10),istcb2), (lstcbc(11),ighbcb)
*
      EQUIVALENCE (DUMMY,X(1))
*           Data for opening input files:'
        data (inpext(j), inpdsc(j),j=0,17) /
     1      'bas', 'Basic package',                                   0
     1      'bcf', 'Block-centered flow',                             1
     1      'wel', 'Well package',                                    2
     1      'drn', 'Drain package',                                   3
     1      'riv', 'River package',                                   4
     1      'evt', 'Evapotranspiration',                              5
     1      'swb', 'Soil water balance',                            ! 6
     1      'ghb', 'General head boundary',                           7
     1      'rch', 'Recharge',                                        8
     1      'sip', 'Strongly implicit procedure',                     9
     1      'pcg', 'Conjugate gradient',                             10
     1      'sor', 'Successive overrelaxation',                      11
     1      'oc',  'Output control',                                 12
     1      'str', 'Streamflow',                                     13
     1      'XXX', 'surface water diversions: obs.',                !14
     1      'pos', 'Postprocessors',                                !15
     1      'obs', 'Water level observations',                      !16
     1      'plt', 'simulation plots'/                              !17
c
        data (outext(j), outdsc(j),j=1,12) /
     1  'prn', 'Modflow standard output',
     1  'log', 'Response log',
     1  'reg', 'Reduced gw or surface water diversions',            !Welz2fm (2),
Str2fm (13)
     1  'swm', 'Balances (watershed and stream); irrigation',       !Swb2bd (6)
     1  'shl', 'Output: evaporation from shallow ground water',     !Swb2bd (6)
     1  'bud', 'cumulative volumes or flow rates (inflow and outflow)', !Postlot (15)
     1  'net', 'total net (in-out) cumulative volumes or flow rates',   !Postlot (15)
     1  'ntz', 'net (in-out) volumes or rates for zones',           !Postloz (15)
     1  'hyd', 'stream and aquifer hydrographs for selected nodes', !Hydlot (15)
     1  'dat', 'solution arrays in "x,y, hnew(x,y)" format',        !Hydlot (15)
     1  'rsd', 'annual summary of hydraulic head residuals',        !Rsdlwl (16)
     1  'mea', 'individual ground water level residuals'/          !Rsdlwl (16)

c       device numbers for the above files:
      data (iopost(j),j=1,12) /
     1  116,117,218,219,220,221,222,223,224,225,226,227/
      data (iopout(j),j=1,12) /2*1,10*0/
      equivalence (iout,iopost(1)), (iolog,iopost(2)),
     1  (ioreg,iopost(3)), (ioswm,iopost(4)), (ioshl,iopost(5)),
     1  (iobud,iopost(6)),
     1  (ionet,iopost(7)), (ionetz,iopost(8)), (iohyd,iopost(9)),
     1  (iodat,iopost(10)), (iobs, iopost(11)), (iomeas,iopost(12))
c iout iolog (ioreg ioswm) (iobud ionet ionetz iohyd iopdat) (iobs iomeas)
c Open files: 1&2 at beginning;
c              3-5 if Modwel(2), Modswb(6) or Modstr(13) packages are invoked;
c              6-10 for Modpost(15) postprocessing package;
c              11-12 for Modrsd(16) package.
c           Data for opening unformatted files for cell-wise flow:
c           iundev(11) = array of unit no.'s for unformatted files
```

```
      data (iundev(j),j=1,11) /
     1    151,152,153,154,155,156,157,158,159,160,161/
      data ioterm /6/
c       note: subr LEGAL2LL in file modloc.for uses unit no. 91 to open
c       a data file with (lat,long) coords for sections in KANSAS.
      data (unffil(j),j=1,11) / 'modhed.unf', 'modddn.unf',
     1    'modbcf.unf', 'modwel.unf', 'moddrn.unf',
     1    'modrch.unf', 'modevt.unf', 'modriv.unf',
     1    'modst1.unf', 'modst2.unf', 'modghb.unf'/
C     ---------------------------------------------------------------
      data initlz /0/
      if (initlz.eq.0) then     !initial call to Modflow
        initlz = 1
c           begin timing execution here:
        call TIMER (itbgn)
        print '(5(1x,a,i8))', 'entry MODFLO: LCSHED=',lcshed,
     1    'icalyr=',icalyr,'kper=',kper,'nstp=',nstp,'kstp=',kstp
        call DATE (chdate)
        call TIME (chtime)
C1------SET SIZE OF X ARRAY. REMEMBER TO REDIMENSION X.
        LENX=mxlen
C2------ASSIGN BASIC INPUT UNIT AND PRINTER UNIT.
        INBAS=1
c       IOUT=iopost(1)
c       iolog=iopost(2)
* spp
        print *,'MODDW (MODFLOW with diffusive wave router)'//
     1         chdate//' at '//chtime
        print *,'File extensions for output (.PRN) and record '//
     1         'of run (.LOG) are attached to case name.'
        print *,'Enter case name, 8 characters max [MODFLOW]: '
        read (*,'(a)') case
        if (case.eq.' ') case='MODFLOW'
        idxlen=LENSTR(case)

c         names of additional data files:

ccc  1  'stm', 'Streamflow mass balance diagnostics',                !Str2fm (13)
        filbal=case(1:idxlen) //'.BAL'    !Input: default balance file name (pass to
Swb2rp)
c
        do j=1,2
          outfil(j) = case(1:idxlen) // '.' // outext(j)
          open (unit=iopost(j),file=outfil(j),status='unknown')
          print '(1x,a,i4,2(1x,a))','Open',iopost(j),outfil(j),outdsc(j)
        end do
        write (iolog,'(a)') case//' = MODFLOW case name (8 chars)'//
     1    chdate//' at '//chtime//'  file '//outfil(2)

        filstm = case(1:idxlen) // '.stm' !Output: streamflow mass balance diagnostics
        open (unit=iostrm,file=filstm,status='unknown')

        inpfil(0)=case(1:idxlen)
        call GETNAM (iolog,inbas,inpdsc(0),inpext(0),inpfil(0))
        iutemp=inbas
        ifile=0
        open (unit=inbas,file=inpfil(0),status='OLD',err=10)
c
C3------DEFINE PROBLEM__ROWS,COLUMNS,LAYERS,STRESS PERIODS,PACKAGES
        CALL BAS1DF(ISUM,HEADNG,NPER,ITMUNI,TOTIM,NCOL,NROW,NLAY,
     1            NODES,INBAS,IOUT,IUNIT)
c
        write (iout,'(3(1x,a))') ' idx Unit',
     1    'File name                    ','Package'
        i=0
        write (iout,'(2i5,2(1x,a))') i,inbas,inpfil(i),inpdsc(i)
        do i=1,17
          ifile=i
          if(iunit(i).gt.0 .and. inpext(i).ne.'XXX')then
            inpfil(i)=case(1:idxlen)
            call GETNAM (iolog,iunit(i),inpdsc(i),inpext(i),inpfil(i))
            open(unit=iunit(i),file=inpfil(i),status='OLD',err=10)
            write (iout,'(2i5,2(1x,a))') i,iunit(i),inpfil(i),inpdsc(i)
          end if
        end do
        print 120, outfil(1),iout,'output (report) file'
120     format (/,a30,1x,' unit ',i3,1x,a)
C4------ALLOCATE SPACE IN "X" ARRAY.
        CALL BAS1AL(ISUM,LENX,LCHNEW,LCHOLD,LCIBOU,LCCR,LCCC,LCCV,
     1    LCHCOF,LCRHS,LCDELR,LCDELC,LCSTRT,LCBUFF,LCIOFL,INBAS,
     1    ISTRT,NCOL,NROW,NLAY,IOUT)
        IF(IUNIT(1).GT.0) CALL BCF1AL(ISUM,LENX,LCSC1,LCHY,
     1    LCBOT,LCTOP,LCSC2,LCTRPY,IUNIT(1),ISS,
     2     NCOL,NROW,NLAY,IOUT,IBCFCB)
        IF(IUNIT(2).GT.0) CALL WEL2AL(ISUM,LENX,LCWELL,MXWELL,NWELLS,
     1     iopsat,satlo,sathi,ratgnd,ratsrf,IUNIT(2),IOUT,IWELCB)
        IF(IUNIT(3).GT.0) CALL DRN1AL(ISUM,LENX,LCDRAI,NDRAIN,MXDRN,
     1                IUNIT(3),IOUT,IDRNCB)
        IF(IUNIT(8).GT.0) CALL RCH1AL(ISUM,LENX,LCIRCH,LCRECH,NRCHOP,
     1             NCOL,NROW,IUNIT(8),IOUT,IRCHCB)
c         add arguments LCDTW and LCEVAP to subr EVT1AL;
c         LCevtr,LCdtw and LCevap are passed to SWB1AL (below). (spp 1/3/96)
        CALL EVT1AL(ISUM,LENX,LCIEVT,LCEVTR,LCEXDP,LCSURF,LCDTW,
     1      LCEVAP,NCOL,NROW,NEVTOP,IUNIT(5),IOUT,IEVTCB)
        CALL RIV1AL(ISUM,LENX,LCRIVR,MXRIVR,NRIVER,
     1             IUNIT(4),IOUT,IRIVCB)
        CALL STR2AL(ISUM,LENX,LCSTRM,ICSTRM,LCsmad,icanst,MXSTRM,
     1    NSTREM,IUNIT(13),IOUT,ISTCB1,ISTCB2,mxseg,mxtrib,NSS,NTRIB,
     1    NDIV,ICALC,CONST,LCTBAR,LCTRIB,LCIVAR,LSQseg,Ldxseg,
     1    LTOseg,LCBseg,LCEseg,Lrcseg,LTRseg,LCLseg,Lactsg,Lacnsg,
     1    Lavgsg,Lminsg,Lmaxsg,Laresg,Ltrvsg,Lckdt,Lsigsg,LCCstr,
     1    Lxcord,Lycord,ncol,nrow,nlay,itoprc,ibotrc,adjbed)

        call SWB2AL(ISUM,LENX,LCSHED,ICSHED,LCWSHD,LCBSHD,LCEVTR,
     1   LCdtw,LCevap,LCrchi,nwshed,nlay,nrow,ncol,iunit,IOUT)

        IF(IUNIT(7).GT.0) CALL GHB1AL(ISUM,LENX,LCBNDS,NBOUND,MXBND,
     1             IUNIT(7),IOUT,IGHBCB)
        IF(IUNIT(9).GT.0) CALL SIP1AL(ISUM,LENX,LCEL,LCFL,LCGL,LCV,
     1           LCHDCG,LCLRCH,LCW,MXITER,NPARM,NCOL,NROW,NLAY,
     2           IUNIT(9),IOUT)
*         from \modflux\modflow.for: conjugate gradient method
        IF(IUNIT(10).GT.0) CALL PCG2AL(ISUM,LENX,LCV,LCSS,LCP,LCCD,
     1         LCHCHG,LCLHCH,LCRCHG,LCLRCH,MXITER,ITER1,NCOL,NROW,NLAY,
     2         IUNIT(10),IOUT,NPCOND)
        IF(IUNIT(11).GT.0) CALL SOR1AL(ISUM,LENX,LCA,LCRES,LCHDCG,
     1    LCLRCH,LCIEQP,MXITER,NCOL,NLAY,NSLICE,MBW,IUNIT(11),IOUT)

        data instzn /1/ !indicates that the Zone Budget code is installed
        call POST1AL(ISUM,LENX,LCidry,LCXLOC,LCYLOC,LCZLOC,LCKT,LC23,
     1    ICTser,ICLser,ICRser,ICCser,ICXser,ICVser,NCOL,NROW,NLAY,
     1    mxser,numser,idxtim,idxvol,instzn,iopzon,mxzone,
     1    IOUT,iunit(15))
        if (instzn.eq.1)  call ZON1AL(ISUM,LENX,iopzon,mxzone,
     1    LCIzon,LCIzbd,LCIzbf,LCIzin,LCIzou,LCIxfi,LCIxfo,LCItiz,
     1    LCItoz,LCIncz,LCIacz,NCOL,NROW,NLAY,IOUT,inpost)

        CALL RSD1AL(ISUM,LENX,LCobs,MXobs,numobs,ioprsd,dtwmax,
     1      iunit(16),IOUT,iobscb)

C5------IF THE "X" ARRAY IS NOT BIG ENOUGH THEN STOP.
        IF(ISUM-1.GT.LENX) then
          print '(2(1x,a,i8))','MODFLOW: X array length mxlen=',lenx,
     1        '; increase mxlen to at least',isum
          STOP
        end if
c         determine which additional files should be opened:
        if (iunit(2).gt.0.or.iunit(6).gt.0.or.iunit(13).gt.0) then
          iopout(3)=1  !Well & Strm: diversions w/ reduced pumping due to operating &
supply limits
          iopout(4)=1  !ModSwb: write summary file in Swb2bd
          iopout(5)=1  !ModSwb: write evaporation results (~.sh1) in Swb2bd
        end if
        if (iunit(15).gt.0) then         !postprocessing package, Modpost
          if (idxvol.gt.0) then
            iopout(6)=1 !budget component file
            iopout(7)=1 !net budget components file
            if (instzn.gt.0.and.iopzon.gt.0) iopout(8)=1 !zone budget file
          end if
          iopout(9)=1                    !hydrograph file
          iopout(10)=1                    !gw head solution file
        end if
        if (iunit(16).gt.0) then         !residuals package, Modrsd
          iopout(11)=1                   !annual summary of residuals file
          iopout(12)=1                   !individual residuals file
        end if
c         Open files:
        do j=3,12
          if (iopout(j).gt.0) then
            outfil(j) = case(1:idxlen) // '.' // outext(j)
            open (unit=iopost(j),file=outfil(j),status='unknown')
            print '(1x,a,i4,2(1x,a))',
     1         'Open',iopost(j),outfil(j),outdsc(j)
          end if
        end do
c
C6------READ AND PREPARE INFORMATION FOR ENTIRE SIMULATION.
        CALL BAS1RP(X(LCIBOU),X(LCHNEW),X(LCSTRT),X(LCHOLD),ISTRT,
     1    INBAS,HEADNG,NCOL,NROW,NLAY,NODES,VBVL,X(LCIOFL),IUNIT(12),
     1    IHEDFM,IDDNFM,IHEDUN,IDDNUN,IOUT)
        print '(1x,a,i3)','modflo after bas1rp: istrt=',istrt
        IF(IUNIT(1).GT.0) CALL BCF1RP(X(LCIBOU),X(LCHNEW),X(LCSC1),
     1          X(LCHY),X(LCCR),X(LCCC),X(LCCV),X(LCDELR),
     2           X(LCDELC),X(LCBOT),X(LCTOP),X(LCSC2),X(LCTRPY),
     3           IUNIT(1),ISS,NCOL,NROW,NLAY,NODES,IOUT)
        IF(IUNIT(6).GT.0)
     1    call SWB2RP (iunit,shed,x(ICSHED),x(LCBSHD),X(LCIBOU),
     1      X(LCDELR),X(LCDELC),x(LCwshd),areain,mxstrm,nwshed,
     1      iopmod,iopswt,irropt,ievopt,ioprch,rchmpy,evapir,welmpy,
     1      iadcod,frseep,itmuni,filbal,nstp,nlay,nrow,ncol,nper,iout)
        IF(IUNIT(9).GT.0) CALL SIP1RP(NPARM,MXITER,ACCL,HCLOSE,X(LCW),
     1          IUNIT(9),IPCALC,IPRSIP,IOUT)
c         from \modflux\modflow.for: conjugate gradient method
        IF(IUNIT(10).GT.0) CALL PCG2RP(MXITER,ITER1,HCLOSE,RCLOSE,
     1         NPCOND,NBPOL,RELAX,IPRPCG,IUNIT(10),IOUT,MUTPCG,IPCGCD)
        IF(IUNIT(11).GT.0) CALL SOR1RP(MXITER,ACCL,HCLOSE,IUNIT(11),
     1         IPRSOR,IOUT)
        do i=1,11
          if (lstcbc(i).gt.0) then
              inquire (unit=lstcbc(i),opened=logopn,name=namchr)
              print '(1x,a,i3,1x,a,13)',
     1            'unit',lstcbc(i),'logopn=',logopn
              if (logopn) then
                  print '(3(1x,a,i3))',
     1                'File '//namchr//' is open on unit',lstcbc(i)
cccccc            lstcbc(i) = iundev(i)
              else
                  write (chrext,'(i3)') lstcbc(i)
                  do j=1,2
                      if (chrext(j:j).eq.' ') chrext(j:j) = '0'
                  end do
                  unffil(i) = case(1:idxlen) // '.'// chrext
                  print '(3(1x,a,i3))','open dev. ',lstcbc(i),
     1                'unformatted file '//unffil(i)
                  open (unit=lstcbc(i),file=unffil(i),
     1                form='UNFORMATTED')
              end if
          end if
        end do
C7------SIMULATE EACH STRESS PERIOD.
        kper=0
        kstp=0
      end if
```

```
c
300   kper = kper + 1
CCCCCCDO 300 KPER=1,NPER
      KKPER=KPER
C7A-----READ STRESS PERIOD TIMING INFORMATION (unless Modflow is called by Swat)
      CALL BAS1ST(NSTP,DELT,TSMULT,PERTIM,KKPER,INBAS,IOUT)
C7B-----READ AND PREPARE INFORMATION FOR STRESS PERIOD.
      IF(IUNIT(2).GT.0) CALL WEL2RP(X(LCWELL),NWELLS,MXWELL,nwread,
     1        ratgnd,ratsrf,welmpy,iadcod,icalyr,nrow,ncol,nlay,
     1        IUNIT(2),IOUT)
      IF(IUNIT(3).GT.0) CALL DRN1RP(X(LCDRAI),NDRAIN,MXDRN,IUNIT(3),
     1                 IOUT)
      IF(IUNIT(8).GT.0) CALL RCH1RP(NRCHOP,X(LCIRCH),X(LCRECH),
     1            X(LCDELR),X(LCDELC),NROW,NCOL,IUNIT(8),IOUT)
      IF(IUNIT(5).GT.0) then
        CALL EVT1RP(NEVTOP,X(LCIEVT),X(LCEVTR),X(LCEXDP),X(LCSURF),
     1            X(LCDELR),X(LCDELC),NCOL,NROW,IUNIT(5),IOUT)
c             initialize depth to water table for watershed connection
        if (kkper.eq.1) CALL EVTINIT (NEVTOP,x(LCIEVT),x(LCSURF),
     1            X(LCIBOU),X(LCHNEW),x(LCDTW),NCOL,NROW,NLAY,IOUT)
      end if
      IF(IUNIT(4).GT.0) CALL RIV1RP(X(LCRIVR),NRIVER,MXRIVR,IUNIT(4),
     1            IOUT)

      IF(IUNIT(13).GT.0) then
        CALL STR2RP(X(LCSTRM),X(ICSTRM),NSTREM,mxstrm,itmpst,ncol,
     1      nrow,nlay,IUNIT(13),IOUT,X(LCTBAR),NDIV,mxseg,mxtrib,NSS,
     1      NTRIB,X(LCIVAR),x(LSQseg),x(LDXseg),x(LTOseg),x(LCBseg),
     1      x(LCEseg),x(Lrcseg),x(LTRseg),x(LCLseg),x(Lacnsg),
     1      x(Lavgsg),x(Lminsg),x(Lmaxsg),x(Laresg),x(LCIBOU),
     1      x(LCCstr),x(LCHOLD),isgtop,isgbot,ICALC,IPTFLG,
     1      kkper,irdcnd,numinp,iroute,nrstep,adjbed)
c
c           Associate points of diversions with stream reaches for supply;
c           surface water (streamflow) diversions are included in Well file.
        IF(iunit(2).gt.0 .and. IUNIT(6).GT.0)
     1    CALL SRF2RP(x(ICSHED),shed,x(LCwshd),X(LCWELL),x(LCCstr),
     1       x(LCBSHD),X(LCIBOU),kkper,nwshed,mxwell,nwells,nwread,
     1       nstrem,nlay,nrow,ncol,iout)

c                 call orig. stream module (Prudic) for steady state case;
c                 Muskingum-Cunge diffusive wave routing for transient case:
        if (kkper.eq.1) then
          istrbd=0 !initialize streamflow and streambed leakage:
          CALL STR2FM(NSTREM,X(LCSTRM),X(ICSTRM),x(LCSMAD),X(LCHNEW),
     1       X(LCHCOF),X(LCRHS),X(LCIBOU),x(LCCstr),MXSTRM,NCOL,NROW,
     1       NLAY,mxseg,mxtrib,X(LCTBAR),X(LCTRIB),X(LCIVAR),
     1       x(LCBseg),x(LCEseg),x(Lrcseg),x(LTRseg),x(Ltrvsg),
     1       x(Lckdt),x(Lsigsg),ICALC,CONST,delt,sreduc,nswred,
     1       irdcnd,kkiter,kkstp,kkper,iroute,istrbd,ievopt,
     1       IOUT,iostrm,ioreg)
        end if  !kkper = 1
      end if    !iunit(13) > 0 (stream package is invoked)
      IF(IUNIT(7).GT.0) CALL GHB1RP(X(LCBNDS),NBOUND,MXBND,IUNIT(7),
     1            IOUT)
c           Associate points of diversions with subbasins for scaling irrigation;
      IF(IUNIT(6).GT.0 .and. itmpst.gt.0) call SWB2STR (shed,
     1    x(ICSHED),x(LCBSHD),X(LCIBOU),x(ICSTRM),x(LCSTRM),
     1    x(Lrcseg),x(LTRseg),X(LCTBAR),x(Laresg),mxstrm,mxseg,mxtrib,
     1    nstrem,nwshed,nlay,nrow,ncol)
      if (iunit(15).gt.0) then
        if (kkper.eq.1) then
          initxy=0  !indicates that grid coord's xx,yy,zz have not been initialized.
          call POST1RP (msum,iunit,idxtim,budlbl,iobud,ionet)
          if (instzn.eq.1.and.iopzon.gt.0)
     1      call ZON1RP (msum,idxtim,iopzon,mxzone,nzones,
     1          nlay,nrow,ncol,x(LCIBOU),x(LCIzon),x(LCIzbd),
     1          x(LCIzbf),x(LCIxfi),x(LCIxfo),x(LCIncz),x(LCIacz),
     1          budlbl,iunit(15),ionetz,iout)
        end if    !kkper = 1
        print '(1x,a,i3)','modflo before hyd1rp: istrt=',istrt
```

```
c       idmain = 0 identifies calls to HYD1RP and HYD1OT from modflo,
c           where drawdown is calculated from beginning and latest heads;
c       idmain=1 identifies corresponding calls from POSTMD97, where
c           drawdown is read from the output (unless starting heads are read
c           from *.bas).
        data idmain /0/
        CALL HYD1RP (X(LCSTRT),ISTRT,X(LCIBOU),mxstrm,nstrem,
     1      x(ICSTRM),KKPER,NCOL,NROW,NLAY,x(LCdelr),x(LCdelc),
     1      x(LCXLOC),x(LCYLOC),x(LCZLOC),idxtim,iunit(15),iohyd,
     1      mxser,numser,x(ICTser),x(ICLser),x(ICRser),x(ICCser),
     1      x(ICXser),istper,iopvec,idaxis,islice,nstp,iopstp,
     1      initxy,idmain)
        print '(1x,a,i3)','modflo after hyd1rp: istrt=',istrt
      end if
      IF(IUNIT(16).GT.0) CALL RSD1RP(X(LCOBS),x(LCIBOU),ncol,nrow,
     1          nlay,mxobs,numobs,iyrper,IUNIT(16),IOUT,iobscb)

C7C-----SIMULATE EACH TIME STEP.
      kstp = 0
      if (iopswt.gt.0) RETURN ! from initializing call to entry pt MODFLO
c                               for SWAT-MODFLOW coupling
c         (if iopswt=0) skip multiple entry pt MODSTP, called from SWAT:
      go to 200
c
      entry MODSTP (iopswt,iopmod,delt,nper,kkper,nstp,kkstp,
     1              icalyr,areain,mxshed,shed)
c
c     usage from Swat: call Modstp at end of each time step.
c
200   kstp = kstp + 1
C*****DO 200 KSTP=1,NSTP
      KKSTP=KSTP
c         added soil water balance module:
      if (iunit(6).gt.0) then
          call SWB2FM (shed,x(ICSHED),x(LCBSHD),X(LCIBOU),
     1      x(ICSTRM),x(LCSTRM),X(LCDELR),X(LCDELC),X(LCEVTR),
     1      X(LCRECH),x(LCwshd),mxstrm,nstrem,nwshed,irropt,ievopt,
     1      ioprch,rchmpy,evapir,iopswt,delt,tsmult,kkper,kkstp,
     1      icalyr,nlay,nrow,ncol,iunit)

          if (iunit(2).gt.0) call WELZ2FM(nwells,mxwell,x(LCwell),
     1      shed,x(LCwshd),x(LCstrm),x(LCIBOU),x(LCBSHD),
     1      x(LCHnew),x(LCBOT),irropt,iopsat,satlo,sathi,
     1      ratgnd,ratsrf,srfdiv,qoprds,greduc,sreduc,ngwred,nswred,
     1      kkstp,kkper,nstrem,nwshed,NCOL,NROW,NLAY,ioreg)
      end if
c
c     test option to treat part of "evaporation from shallow ground water"
c     as seepage to the stream channel (subr. EVT2STR is in MODSWB);
c     frseep is read in subr SWB2RP (MODSWB package).  Note: if HRU scheme
c     3 is used, frseep should be passed back to soil water balance simulator
c     where only the fraction (1-frseep) of contribution from shallow gw
c     should be redistributed over the soil profile.

      if (iunit(5).gt.0.and.iunit(13).gt.0 .and. 1.+frseep.gt.1.)
     1    call EVT2STR (nlay,nrow,ncol,nstrem,mxstrm,numevt,
     1    x(icstrm),x(Lcstrm),frseep,x(Lcevap))

c         option to read stream inflows for each time step beyond
c         the first time step in each stress period:
      IF(IUNIT(13).GT.0 .and. numinp.gt.0 .and. kkstp.gt.1)
     1  CALL STR2STP (NSTREM,X(LCSTRM),x(LCCstr),MXSTRM,NCOL,NROW,NLAY,
     1      kkstp,kkper,numinp,iunit(13),IOUT)
c
C7C1----CALCULATE TIME STEP LENGTH. SET HOLD=HNEW.
      CALL BAS1AD(DELT,TSMULT,TOTIM,PERTIM,X(LCHNEW),X(LCHOLD),KKSTP,
     1             NCOL,NROW,NLAY)

C7C2----ITERATIVELY FORMULATE AND SOLVE THE EQUATIONS.
      DO 100 KITER=1,MXITER
      KKITER=KITER
```

```
C7C2A---FORMULATE THE FINITE DIFFERENCE EQUATIONS.
      CALL BAS1FM(X(LCHCOF),X(LCRHS),NODES)
      IF(IUNIT(1).GT.0) CALL BCF1FM(X(LCHCOF),X(LCRHS),X(LCHOLD),
     1          X(LCSC1),X(LCHNEW),X(LCIBOU),X(LCCR),X(LCCC),X(LCCV),
     2          X(LCHY),X(LCTRPY),X(LCBOT),X(LCTOP),X(LCSC2),
     3          X(LCDELR),X(LCDELC),DELT,ISS,KKITER,KKSTP,KKPER,NCOL,
     4          NROW,NLAY,IOUT)
c
      IF(IUNIT(2).GT.0) CALL WEL2FM(NWELLS,MXWELL,X(LCRHS),X(LCWELL),
     1           X(LCIBOU),NCOL,NROW,NLAY)
      IF(IUNIT(3).GT.0) CALL DRN1FM(NDRAIN,MXDRN,X(LCDRAI),X(LCHNEW),
     1         X(LCHCOF),X(LCRHS),X(LCIBOU),NCOL,NROW,NLAY)
      IF(IUNIT(8).GT.0) CALL RCH1FM(NRCHOP,X(LCIRCH),X(LCRECH),
     1            X(LCRHS),X(LCIBOU),NCOL,NROW,NLAY)
      IF(IUNIT(5).GT.0) CALL EVT1FM(NEVTOP,X(LCIEVT),X(LCEVTR),
     1            X(LCEXDP),X(LCSURF),X(LCRHS),X(LCHCOF),X(LCIBOU),
     1            X(LCHNEW),NCOL,NROW,NLAY)
      IF(IUNIT(4).GT.0) CALL RIV1FM(NRIVER,MXRIVR,X(LCRIVR),X(LCHNEW),
     1            X(LCHCOF),X(LCRHS),X(LCIBOU),NCOL,NROW,NLAY)
c        version of STR2FM for diffusive flood routing:
      IF(IUNIT(13).GT.0) then
c            call orig. stream module (Prudic) for steady state case:
        istrbd=1 !solution not finished yet
        CALL STR2FM(NSTREM,X(LCSTRM),X(ICSTRM),x(LCSMAD),X(LCHNEW),
     1    X(LCHCOF),X(LCRHS),X(LCIBOU),x(LCCstr),MXSTRM,NCOL,NROW,
     1    NLAY,mxseg,mxtrib,X(LCTBAR),X(LCTRIB),X(LCIVAR),x(LCBseg),
     1    x(LCEseg),x(Lrcseg),x(LTRseg),x(Ltrvsg),x(Lckdt),x(Lsigsg),
     1    ICALC,CONST,delt,sreduc,nswred,irdcnd,kkiter,kkstp,kkper,
     1    iroute,istrbd,ievopt,IOUT,iostrm,ioreg)
      end if
      IF(IUNIT(7).GT.0) CALL GHB1FM(NBOUND,MXBND,X(LCBNDS),X(LCHCOF),
     1            X(LCRHS),X(LCIBOU),NCOL,NROW,NLAY)
C7C2B---MAKE ONE CUT AT AN APPROXIMATE SOLUTION.
      IF(IUNIT(9).GT.0) CALL SIP1AP(X(LCHNEW),X(LCIBOU),X(LCCR),X(LCCC),
     1     X(LCCV),X(LCHCOF),X(LCRHS),X(LCEL),X(LCFL),X(LCGL),X(LCV),
     2     X(LCW),X(LCHDCG),X(LCLRCH),NPARM,KKITER,HCLOSE,ACCL,ICNVG,
     3     KKSTP,KKPER,IPCALC,IPRSIP,MXITER,NSTP,NCOL,NROW,NLAY,NODES,
     4     IOUT)

c        from \modflux\modflow.for: conjugate gradient method
      IF(IUNIT(10).GT.0) CALL PCG2AP(X(LCHNEW),X(LCIBOU),X(LCCR),
     1     X(LCCC),X(LCCV),X(LCHCOF),X(LCRHS),X(LCV),X(LCSS),X(LCP),
     2     X(LCCD),X(LCHCHG),X(LCLHCH),X(LCRCHG),X(LCLRCH),KITER,
     3     NITER,HCLOSE,RCLOSE,ICNVG,KSTP,KPER,IPRPCG,MXITER,ITER1,
     4     NPCOND,NBPOL,NSTP,NCOL,NROW,NLAY,NODES,RELAX,IOUT,MUTPCG,
     5     IPCGCD,STEPL,DELT,IUNIT(15),IP)
      IF(IUNIT(11).GT.0) CALL SOR1AP(X(LCHNEW),X(LCIBOU),X(LCCR),
     1     X(LCCC),X(LCCV),X(LCHCOF),X(LCRHS),X(LCA),X(LCRES),X(LCIEQP),
     2     X(LCHDCG),X(LCLRCH),KKITER,HCLOSE,ACCL,ICNVG,KKSTP,KKPER,
     3     IPRSOR,MXITER,NSTP,NCOL,NROW,NLAY,NSLICE,MBW,IOUT)
C7C2C---IF CONVERGENCE CRITERION HAS BEEN MET STOP ITERATING.
      IF(ICNVG.EQ.1) GO TO 110
  100 CONTINUE
      KITER=MXITER
      kkiter=kiter
  110 CONTINUE
      if (kkstp*kkper.eq.1) write (*,'(1x,a)') 'Period step iteration'
      write (*,'(1x,3i5)') kkper,kkstp,kkiter
C7C3----DETERMINE WHICH OUTPUT IS NEEDED.
      CALL BAS1OC(NSTP,KKSTP,ICNVG,X(LCIOFL),NLAY,
     1  IBUDFL,ICBCFL,IHDDFL,IUNIT(12),IOUT)
C7C4----CALCULATE BUDGET TERMS. SAVE CELL-BY-CELL FLOW TERMS.
      MSUM=1
      IF(IUNIT(1).GT.0) CALL BCF1BDZ(VBNM,VBVL,MSUM,X(LCHNEW),X(LCIBOU),
     1 X(LCHOLD),X(LCSC1),X(LCCR),X(LCCC),X(LCCV),X(LCTOP),X(LCSC2),
     1 DELT,ISS,NCOL,NROW,NLAY,KKSTP,KKPER,IBCFCB,ICBCFL,X(LCBUFF),IOUT,
     1 iopzon,mxzone,x(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou),x(LCIzbf))
      IF(IUNIT(2).GT.0) CALL WEL2BDZ(NWELLS,MXWELL,VBNM,VBVL,MSUM,
     1  X(LCWELL),X(LCIBOU),DELT,NCOL,NROW,NLAY,KKSTP,KKPER,IWELCB,
     1  ICBCFL,X(LCBUFF),IOUT,
     1  iopzon,mxzone,X(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))
```

```
      IF(IUNIT(3).GT.0) CALL DRN1BDZ(NDRAIN,MXDRN,VBNM,VBVL,MSUM,
     1      X(LCDRAI),DELT,X(LCHNEW),NCOL,NROW,NLAY,X(LCIBOU),
     1      KKSTP,KKPER,IDRNCB,ICBCFL,X(LCBUFF),IOUT,
     1      iopzon,mxzone,X(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))
      IF(IUNIT(8).GT.0) CALL RCH1BDZ(NRCHOP,X(LCIRCH),X(LCRECH),
     1      X(LCIBOU),NROW,NCOL,NLAY,DELT,VBVL,VBNM,MSUM,
     1      KKSTP,KKPER,IRCHCB,ICBCFL,X(LCBUFF),IOUT,
     1      iopzon,mxzone,X(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))
c          note: modified EVT1BD argument list --spp 12/30/95
      IF(IUNIT(5).GT.0) CALL EVT1BDZ(NEVTOP,X(LCIEVT),X(LCEVTR),
     1      X(LCEXDP),X(LCSURF),X(LCIBOU),X(LCHNEW),x(LCDTW),x(LCEVAP),
     1      NCOL,NROW,NLAY,DELT,VBVL,VBNM,MSUM,KKSTP,KKPER,IEVTCB,
     1      ICBCFL,X(LCBUFF),IOUT,ioterm,
     1      iopzon,mxzone,x(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))
      IF(IUNIT(4).GT.0) CALL RIV1BDZ(NRIVER,MXRIVR,X(LCRIVR),X(LCIBOU),
     1      X(LCHNEW),NCOL,NROW,NLAY,DELT,VBVL,VBNM,MSUM,
     2      KKSTP,KKPER,IRIVCB,ICBCFL,X(LCBUFF),IOUT,
     1      iopzon,mxzone,X(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))

      if (iunit(13).gt.0) then
        istrbd=2 !stream solution is finished
        CALL STR2FM(NSTREM,X(LCSTRM),X(ICSTRM),x(LCSMAD),X(LCHNEW),
     1    X(LCHCOF),X(LCRHS),X(LCIBOU),x(LCCstr),MXSTRM,NCOL,NROW,
     1    NLAY,mxseg,mxtrib,X(LCTBAR),X(LCTRIB),X(LCIVAR),x(LCBseg),
     1    x(LCEseg),x(Lrcseg),x(LTRseg),x(Ltrvsg),x(Lckdt),x(Lsigsg),
     1    ICALC,CONST,delt,sreduc,nswred,irdcnd,kkiter,kkstp,kkper,
     1    iroute,istrbd,ievopt,IOUT,iostrm,ioreg)

        CALL STR2BDZ(NSTREM,X(LCSTRM),X(ICSTRM),X(LCIBOU),
     1    MXSTRM,NCOL,NROW,NLAY,DELT,VBVL,VBNM,MSUM,KKSTP,KKPER,
     1    ISTCB1,ISTCB2,ICBCFL,X(LCBUFF),IOUT,ICALC,IPTFLG,
     1    iopzon,mxzone,x(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))

        if (icbcfl.ne.0.and.(istcb1.gt.0 .or. istcb2.gt.0))
     1    close (unit=iundev(i))
      end if
c            summarize water level residuals (calc. - obs. heads)
      if (iunit(16).gt.0) call RSD1WL (nresid,numobs,mxobs,iyrper,
     1    x(LCobs),x(LCIBOU),x(LCHNEW),x(LCBOT),dtwmax,ioprsd,
     1    KKPER,KKSTP,nstp,NCOL,NROW,NLAY,iomeas,iobs)

      IF(IUNIT(7).GT.0) CALL GHB1BDZ(NBOUND,MXBND,VBNM,VBVL,MSUM,
     1      X(LCBNDS),DELT,X(LCHNEW),NCOL,NROW,NLAY,X(LCIBOU),
     1      KKSTP,KKPER,IGHBCB,ICBCFL,X(LCBUFF),IOUT,
     1      iopzon,mxzone,X(LCIzon),x(LCIzbd),x(LCIzin),x(LCIzou))

C7C5---PRINT AND OR SAVE HEADS AND DRAWDOWNS. PRINT OVERALL BUDGET.

      CALL BAS1OT(X(LCHNEW),X(LCSTRT),ISTRT,X(LCBUFF),X(LCIOFL),MSUM,
     1  X(LCIBOU),VBNM,VBVL,volnet,KKSTP,KKPER,DELT,PERTIM,TOTIM,ITMUNI,
     1  NCOL,NROW,NLAY,ICNVG,IHDDFL,IBUDFL,IHEDFM,IHEDUN,IDDNFM,IDDNUN,
     4  timstp,timper,timsim,totinp,
     1  totout,diff,pdiff,IOUT,iopzon,nzones,mxzone,x(LCIzbd),x(LCIzbf),
     1  x(LCIxfi),x(LCIxfo),x(LCItiz),x(LCItoz),x(LCIncz),x(LCIacz),
     1  bud1bl)
c          Soil water balance package
      if (iunit(6).gt.0)
     1  call SWB2BD (shed,x(ICSHED),x(LCBSHD),X(LCIBOU),x(ICSTRM),
     1   x(LCSTRM),x(LCwshd),x(LCdtw),x(LCevap),X(LCDELR),X(LCDELC),
     1   itoprc,ibotrc,mxstrm,nstrem,nwshed,delt,timsim(idxtim),volnet,
     1   srfdiv,qoprds,greduc,sreduc,ngwred,nswred,kkper,kkstp,
     1   icalyr,nlay,nrow,ncol,iopswt,ioswm,ioshl)
      if (iunit(15).gt.0) then
        call POST1OT (MSUM,VBVL,volnet,KKSTP,KKPER,kkiter,delt,
     1    timsim(idxtim),totinp,totout,diff,pdiff,idxvol,iobud,ionet)

        if (instzn.gt.0 .and. iopzon.gt.0 .and. ibudfl.gt.1)
     1      call POST1OZ (MSUM,mxzone,x(LCIzbd),x(LCIxfi),x(LCIxfo),
     1        x(LCItiz),x(LCItoz),x(LCIncz),x(LCIacz),KSTP,KPER,kiter,
     1        timsim(idxtim),idxvol,ionetz)
```

```
        CALL HYD1OT (X(LCHNEW),X(LCBUFF),X(LCSTRT),ISTRT,X(LCIBOU),
     1    x(LCidry),mxstrm,nstrem,x(LCSTRM),x(LCTOP),x(LCBOT),
     1    X(LCDELR),X(LCDELC),x(LCXLOC),x(LCYLOC),x(LCZLOC),
     1    x(LC23),x(LCKT),x(LCCR),x(LCCC),x(LCCV),KKSTP,KKPER,
     1    NCOL,NROW,NLAY,timsim(idxtim),delt,iohyd,mxser,numser,
     1    x(ICTser),x(ICLser),x(ICRser),x(ICCser),x(ICXser),x(ICVser),
     1    iopvec,idaxis,islice,nstp,iopstp,iodat,idmain)
      end if

C7C6----IF ITERATION FAILED TO CONVERGE THEN STOP.
      IF(ICNVG.EQ.0) then
        print '(1x,a)',
     1    'Solution failed to converge; Modflow stops here.'
        STOP
      end if
      if (iopswt.gt.0) RETURN ! from entry pt MODSTP for SWAT-MODFLOW coupling
      if (kstp.lt.nstp) go to 200
CCCCCCCCCCC200 CONTINUE
      if (kper.lt.nper) go to 300
CCCCCCCCCCC300 CONTINUE
320   continue
      do 340 i=1,9
        if (lstcbc(i).gt.0) close (unit=iundev(i))
340     continue
C8------END PROGRAM
      call TIMER(itend)
c           time spent in execution (s):
      timsec = 0.01*(itend-itbgn)
      itmhrs = timsec/3600.
      timrem = timsec - 3600.*itmhrs
      itmmin = timrem/60.
      timrem = timrem - 60.*itmmin
      write (ioterm,410) 'Modflow: Execution time (s) =',timsec
410   format (5(1x,a,1x,f10.2,/))
      write (ioterm,400) '(hrs:min:sec) =',itmhrs,itmmin,timrem
400   format (1x,a,i3,':',i2,':',f5.2)
      close (unit=iolog)
      if (iopswt.gt.0) RETURN ! from entry pt MODEND for SWAT-MODFLOW coupling
      STOP
10    print '(2(1x,a,1x,i3))',
     1          'File '//inpfil(ifile)//' not found; unit=',
     1          iutemp,'specified in IUNIT field ',ifile
      write (iolog,'(2(1x,a,1x,i3))')
     1          'File '//inpfil(ifile)//' not found; unit=',
     1          iutemp,'specified in IUNIT field ',ifile
      STOP
      END
```

SUBROUTINE GETNAM - Source file Listing

```
      subroutine GETNAM (iolog,iodev,descrp,filext,filnam)
c
        integer iodev, iolog, nchar
        character *(*) descrp, filext, filnam
        character*8 namdft
c
        namdft=filnam
        if (namdft.eq.' ') namdft='MODFLOW'
        nchar=LENSTR(namdft)
1       print 5,descrp,' unit ',iodev,' ext. ',filext,' name ['//
     1          namdft(1:nchar)//']: '
5       format (1x,2a,1x,i3,1x,3a)
        filnam=' '
        read (*,'(a)') filnam
        if (filnam.eq.' ') filnam=namdft
        write (iolog,100) filnam, iodev, descrp
100     format (a30,1x,' unit ',i3,1x,a)
        return
      end
```

FUNCTION LENSTR - Source file Listing

```
        integer function LENSTR (string)
*
*       LENSTR returns the length of character variable 'string'
*       minus the blank-filled space to its right.
*       LENSTR is similar to the intrinsic function LEN(string), which
*       returns the declared length of the character variable 'string'.
*
        character *(*) string
        character *255 str
        integer slash, delete, length, i
        data slash /33/,delete/127/
*
        str = string
        length = LEN(str)
C*****  print *,'string='//str(1:20)
C*****  print *,'length=LEN(str)=',length
        i=0
        do while (i.lt.length)
            LENSTR=i
            i=i+1
            ischar=ICHAR(str(i:i))
C******     print '(1x,a,3i5)',str(i:i),ischar,slash,delete
            if (ischar.le.slash.or.ischar.ge.delete) go to 10
        end do
        LENSTR = i
10      continue
C****** print 20, 'str(1:',LENSTR,') = '//str(1:LENSTR)
20      format (5(1x,a,1x,i5))
        return
        end
```

Modswb package (file modswb.for)

Definitions

```
c file modswb.for perkins  Jun 99: revised WC version for Repub. R. basin.
c                             later version: Oct 98 for Walnut Creek Basin
c                             early version: Feb 15 95 for Republican R. basin
c Summary of subroutines:
c subr SWB2AL: allocate space in X array for subbasins (shed,ished)
c              and watershed map (ibshed).
c subr SWB2RP: Set up subbasin zone-aquifer grid correspondence;
c               associate outflow from each subbasin with a reach of
c               the stream network defined with the modified Stream package.
c subr EVT2STR: (test) treat a fraction frseep of "evaporation from shallow
c               ground water" as flow from seepage faces into the stream
c               channel if one is associated with the grid cell.
c subr SWB2STR:
c subr SWB2FM: convert simulation results for each subbasin to stream and
c               aquifer fluxes in each time step.
c subr SWB2BD: summarize baseflow and evaporation from shallow gw.
c               -has dependencies regarding water level observations that need
c                to be moved to the MODRSD package.
c
c MODSWB package interdependencies:
c-----------------------------------------------------------------------
c Use of this package in its current form requires invoking the following
c packages:
c EVT: define array evtr in terms of potential evaporation;
c RCH: specify spatial distribution of recharge within each subbasin with
c       input to the Recharge package; scale this distribution with
c       recharge due to percolation from the root zone, transmission losses
c       along ephemeral streams, and seepage from ponds.
c WEL (modified; well array expanded to well(20,mxwell).  Modified input format
c       distinguishes the following:
c       bccode = "*" in col. 41 indicates imaginary wells to specify
c               boundary conditions or recharge;
c       usecod in col. 42 indicates source as either ground water ("G") or
c               surface water ("S"); surface water sources require a
c               correspondence with a stream reach to supply the diversion.
c STR (modified): specify lateral surface flows for stream network based on
c       tributary inflows from subbasins and surface water diversions.
c       The routing procedure for the Stream package was modified to include
c       surface lateral flows.
c
c-----------------------------------------------------------------------
c ~.SWB input file definitions:
c-----------------------------------------------------------------------
c
c     SWB2AL
c 1. read (FREE): nwshed
c    nwshed = no. subbasins in watershed;
c
c     SWB2RP
c 2. read (FREE): irropt ievopt ioprch rchmpy evapir welmpy iadcod
c irropt = option (y>0,n=0) to specify irrigation pumping rates for both ground
c          and surface water diversions with results of watershed simulation;
c       =1: Scale Qapp, appropriations for irrigation diversions within each
c         subbasin, by the total irrigation demand for the subbasin according
c         to the watershed simulation; i.e., apply the scaling factor s = Qirr/Qapp
c         to the appropriated rate of each individual diversion.
c       =2: define scaling factor in terms of totals for entire basin rather
c         than individual subbasins.
c ievopt = option (y=1,n=0) to include evaporation based on results from
c         watershed model:
c       = 1 or 3: specify potential evap from ground surface (array EVTR in MODFLOW's
c         EVT package) with the potential evaporation flux from watershed simulation.
c       = 2 or 3: calculate evaporation from stream surface according to potential
```

```
c         evaporation rate given by watershed simulation results.
c ioprch = option for how to specify recharge:
c       =0: use values specified by Modflow's recharge input file;
c       =1: use depth calculated by watershed model results for corresp. subbasin;
c       =2: use depth calculated by watershed model results for basin.
c             previous version of ioprch (from Rattlesnake version of program):
c             option (y=1,n=0) to distribute recharge within each subbasin
c             according to the recharge package input file array RECH.
c rchmpy = multiplier to be applied to recharge (for sensitivity)
c         change from free to fixed format (3, below) for std version?
c evapir = fraction of irrigation pumping lost to evaporation
c effirr (dropped?) = irrigation efficiency; i.e., the fraction evapir = (1 - effirr)
c             is assumed to be lost to evaporation.
c welmpy = multiplier to be applied to pumping rates (for sensitivity).
c iadcod = administrative option code, applied in subr WEL1F0.
c iwtsub (DROP for now) = option to scale irrig. diversions either by:
c             (iwtsub=0) basin avg depth of irrig. calculated by SWAT, or
c             (iwtsub>0) subbasin depth of irrig. calculated by SWAT.
c
c 3. read (FREE): nambal, cnvlen, depmpy, widmpy
c nambal = name of the balance file with results of hydrologic model.
c     if nambal = ' ', then use the default based on the case name with
c     extension '.BAL'.  This allows the same SWB file (above) to be used
c     for all cases if the balance file names are based on the case names.
c cnvlen = conversion factor for std units of length from Modflow to hydrologic
c     model. Ex. Modflow (ft) to Swat (m): cnvlen = 0.3048 (m/ft).
c depmpy conversion for model results from hydrologic depth to std unit of length.
c     Ex. Swat: Hydrologic depths are in mm, std length units are m;
c     so depmpy = 1.e-3 (m/mm)
c widmpy conversion for model results from units of length used for land surface
c     areas to std unit of length.
c     Ex. Swat: Land surface areas are given in km^2.  The corresponding units
c     of length are km, and std length units are m; so widmpy = 1.e+3 (m/km)
c
c 4. read (U2DINT): ibshed
c ibshed(ncol,nrow): 2-d array that identifies the subbasin corresponding to
c     each grid cell.
c
c 5. read (a) heading for data to associate subbasin outflow with stream reaches:
c 6. read (FREE) idx isact irowout icolout isbnxt
c idx       index to subbasin
c isact     indicates whether (yes:1,no:2) subbasin is active.  Hydrologic
c           results are read from the balance file only for active subbasins.
c irowout   row of grid cell corresponding to subbasin outflow location
c icolout   column of grid cell corresponding to subbasin outflow location
c isbnxt    next subbasin in lateral flow routing sequence
c <end of ~.SWB input file>
c
c <open ~.BAL input file>
c
c ~.BAL input file definitions:
c-----------------------------------------------------------------------------
c     SWB2RP (cont'd)
c 1. Read (FREE) nyrs iopmod iopcnv da fpd
c nyrs = no. years to simulate; should agree with no. stress periods, nper, in
c             the basic package input file.
c iopmod = time step option (0=avg annual, 1=annual, 2=monthly, 3=daily)
c iopcnv = option regarding form of hydrologic results:
c   =0: assume form according to Swat-Modflow linkage for Repub. R. basin model;
c   >0: assume results have already been converted and combined to give
c       each subbasin's tributary inflows, gw recharge, evaporation, and both
c       surface and gw diversions.
c da = basin area, units of hydrologic model.  Ex. Swat units are km^2.
c fpd = noncontributing fraction of basin

c 2. read (a) heading for the following:
c 3. read (FREE) for each subbasin:  idx fract noncontrib swinit pndinit
c idx = subbasin index
c fract = subbasin area as fraction of basin
c noncontrib = noncontributing fraction of basin, i.e., fraction of subbasin
c       area that drains to ponds.
```

```
c swinit = initial soil water content (mm)
c pndinit = initial pond volume (cu. m)
c
c 4. read (a) heading for hydrologic results in each time step:
c     SWB2FM
c 5. read (FREE) for each time step:
c    do for each ACTIVE subbasin (i.e., each subbasin with isact>0, above):
c       iyr imo ndays isub precip irrig ET_act surq  xmloss ~
c           perc gwre ET_gw basflo pseep ET_pot SW    pndvol
c    end do
c---------------------------------------------------------
c
c        Notes on input data definitions for Walnut Creek version, Oct. 98:
c
c  Summary of tributaries with corresponding watershed, streamside,
c  aquifer grid coordinates, stream (segment and reach) into which
c  the tributary flows, and tributary width a little upstream from
c  its mouth:
c         tributary input format should be '(4i5,1x,a)':
c         for each subbasin, read whether subbasin index is active;
c         (row,col) location for runoff from subbasin; areal fraction;
c         initial pond volume (ac-ft), soil water content (ft).
c stpfil = name of data file with fluxes to be read for each time step
cYear  mon days  sub precip,ft irrig,cfs  ETact,ft  ETpot,ft   perc,ft       Qtrib,cfs
Pseep,cfs SWcont,ft dVpond,af  irrig,ft  irdep,ft IrArea,sqft
c1960    1   31        0.08358  0.00000  0.01976  0.04372  0.00000         0.00000
0.00257  0.96291     0.000   0.00000  0.00000    15685.
c                  1   0.16417  0.00000  0.03033  0.06392  0.00000         0.00000
0.00000  1.53850     0.000   0.00000  0.00000        0.
c                  2   0.13750  0.00000  0.02750  0.06408  0.00000         0.00000
0.00000  1.49542     0.000   0.00000  0.00000        0.
c---------------------------------------------------------
c definitions for input to swb file (new version oct 21 98 spp):
c idx     subbasin index from 1 to nwshed subbasins
c isact   indicates whether (y=1,n=0) data are to be read for this subbasin
c         index in each time step in SWB1F0.
c (irow,icol)  grid location assigned for subbasin runoff to stream network
c isbnxt next subbasin in routing sequence; used to assign pond seepage
c         in subbasins without active aquifer grid cells to recharge in next
c         subbasin in routing sequence. ished(3,i) for subbasin i.
c cntfrc areal fraction of subbasin contributing to runoff
c subfrc subbasin area as fraction of basin
c subare subbasin area [L^2] (here, ft^2)
c pndini initial pond volume (acre-ft)
c swinit initial soil water content as depth (ft)
c---------------------------------------------------------
c sub   act   row   col cntrb frc areal frc    area ft^2 pnd ac-ft swcont ft
c   1     1    25   126     1.0000 0.0083100  367853614.     0.000     1.400
c   2     1    15   128     1.0000 0.0040550  179500169.     0.000     1.263
c    nsbact = no. "active" subbasins" (some subbasin numbers are just
c                   placeholders, e.g. 46-50)
c------------------------------------------------------------------------------
c
c Integer array ISHED definitions:
c  1  isact: 1 indicates active subbasin.
c  2  istrec:  index to record for stream (segment, reach) corresponding to
c     the grid cell specified by (row, column) in the ~.SWB input file and
c     retained in this array as vectors 4 and 5, below.
c  3  isbnxt:  next subbasin in routing sequence; used to assign pond seepage
c     in subbasins without active aquifer grid cells to recharge in next
c     subbasin in routing sequence.
c  4  row of grid cell designated as point of outflow from subbasin
c  5  column of grid cell designated as point of outflow from subbasin
c  6  no. cells in each subbasin with nonzero value for ibound(ic,ir,1);
c     evaluated in subr SHALLOW as vector numshd(*), and corresponds to
*     gridded area of each subbasin given by shed(3,*).
*  7  subset of ished(6,*) w/ shallow gw, defined by
*     (dtw(ic,ir) < exdp(ic,ir), where exdp = extinction depth,
*     defined in MODEVT; ished(7,*) is evaluated in subr SHALLOW as
*     vector nshzon.
c  8  no. grid cells in subbasin with stream reaches
```

```
c  9  no. active grid cells in subbasin with stream reaches
* 10  no. pumping wells (Qw < 0)in each subbasin for the stress period
c
c Real array SHED definitions:
c  1  subbasin areal fraction (read)
c  2  subbasin area [L^2] (read)
c  3  gridded subbasin area (see also shed(7,*) and ished(6,*), below).
c     Defined as area of each subbasin w/ nonzero value for ibound;
c     evaluated in subr SHALLOW as vector sharea(*).
c  4  irrigation [L] based on WELL package input as follows:
c            Qsub*dt = total volume assigned in Well package; sum well(5,*)
c      over wells in subbasin isub; store Qsub*dt as shed(4,*).  Then
c            dsub = Qsub*dt/subbasin area; store as shed(9,*).
c  5  initial pond storage (acre-ft)
c  6  initial soil water content (ft)
c  7  gridded active subbasin area as fraction of actual subbasin area;
c     used as correction factor to gridded areal weights e.g. for
c     recharge assignment (see subr SWB1F0).
c  8  noncontributing areal fraction of subbasin, which contributes
c     instead to ponds
c  9  irrigation flux: depth corresponding to quantity (volume) assigned
c     in WELL input file (see shed(4,*) def. above.
c
c vectors 10-25: hydrologic results of watershed simulation passed to Modflow
c 10        precipitation
c 11        applied irrigation [L] (used to scale pumping if irropt>0)
c 12        ET from ground surface [L] (maximum rate for evap from water table)
c 13        surface runoff [L] to tributaries leaving subbasin
c 14        transmission loss [L]
c 15        lateral flow [L] to tributaries leaving subbasin
c 16        percolation from the root zone
c 17        groundwater recharge: includes percolation(16), pond seepage(20),
c           and transmission losses (14).
c   hydrologic components also passed to Modflow, but superseded by
c   values calculated by Modflow:
c 18        evaporation from water table (replaces "revap" from Swat); see
c           also (21).
c 19        baseflow (calculated as streambed leakage in STREAM package);
c           the negative of streambed leakage from stream reaches.
c 20        pond seepage into aquifer
c 21        potential et given by watershed simulation.  If ievopt>0, MODSWB uses
c           this to calculate max. evaporation rate from water table; otherwise,
c           the EVT package input array EVTR is used as the maximum rate.
c 22        net tributary inflows, including surface runoff (13), subsurface
c           lateral flow (15), and surface water diversions, which are
c           subtracted from net tributary inflow (22) in subr Welz2fm in each
c           time step.
c 23        gw pumping other than for irrigation
c 24        gw pumping for irrigation
c 25        recharge assigned to subbasins with no aquifer
c 26        stream length within each subbasin
c 27        stream length through active grid cells within each subbasin
c 28        QirrigAvg: avg pumping rate of wells with nonzero pumping as
c           specified by the ~.WEL input file for the stress period
c 30        QirrigMax: max pumping rate of wells as specified by the ~.WEL
c           input file for the stress period.
```

Subr. Swb2al: allocate array storage for watershed connections to stream-aquifer grid

```
c
      SUBROUTINE SWB2AL(ISUM,LENX,LCSHED,ICSHED,LCWSHD,LCBSHD,Lcevtr,
     1  LCdtw,LCevap,LCrchi,nwshed,nlay,nrow,ncol,iunit,IOUT)

C-----VERSION    1 01MAR1995 SWB1AL
C     ******************************************************************
C     ALLOCATE ARRAY STORAGE FOR WATERSHEDS
C     ******************************************************************
C
```

```
C     SPECIFICATIONS:
C     ---------------------------------------------------------------
      dimension iunit(24)
C     ---------------------------------------------------------------
C
      in = iunit(6)
C1------IDENTIFY PACKAGE AND INITIALIZE SUBBASINS.
      WRITE(IOUT,1) IN
    1 FORMAT(1H0,'SWB  -- SOIL WATER BALANCE PACKAGE V.1, 3/01/95',
     1'INPUT READ FROM UNIT',I3)
C
C2------ READ:
      if (iunit(6).gt.0) read (in,*) nwshed                   !Line 1
      nwshed = MAX0 (nwshed,1) !min no. subbasins =1 for dimensioning
      print '(5(1x,a,i5))','SWB1AL: nwshed=',nwshed
C
C3------SET LCSHED EQUAL TO ADDRESS OF FIRST UNUSED SPACE IN X.
  200 LCSHED=ISUM
C3------SPACE NEEDED FOR real watershed subbasin array shed(30,mxshed):
      ISPA=30*(1+nwshed)
      ISUM=ISUM+ISPA
C
C4------SPACE NEEDED FOR integer watershed array ished(10,mxshed):
      ICSHED=ISUM
      ISPB=10*(1+nwshed)
      ISUM=ISUM+ISPB
c
      LCWSHD=isum     !index to array swtflx(30), basin-wide avg fluxes.
      ispb2=30
      isum=isum+ispb2
C
C6------CALCULATE AMOUNT OF SPACE NEEDED FOR watershed-aquifer grid index.
      LCBSHD=isum
      write (iout,'(1x,a,i8)')
     1     'watershed-aquifer grid ibshed index LCBSHD =',lcbshd
      ispd=ncol*nrow
      isum=isum+ispd

c         If evap module isn't invoked, add space for max evap rate,
c         dtw and evap arrays:
      if (iunit(5).eq.0) then
          ispi1 = ncol*nrow
          lcevtr = isum        !x(lcevtr): evtr(ncol,nrow)
          isum = isum + ispi1
          ispi2 = ncol*nrow
          lcdtw  = isum        !x(lcdtw):  dtw(ncol,nrow)
          isum = isum + ispi2
          ispi3 = ncol*nrow
          lcevap = isum        !x(lcevap): evap(ncol,nrow)
          isum = isum + ispi3
      else
          ispi1 = 0
          ispi2 = 0
          ispi3 = 0
      end if
      ispi = ispi1 + ispi2 + ispi3
c         space for initial recharge matrix whose distribution is to be
c         maintained for each time step by scaling element values by
c         (psi_s/psi_r), where psi_s = subbasin recharge flux given by watershed
c         simulation results, and psi_r = subbasin recharge flux corresponding
c         to the initial recharge matrix specified in the recharge input file;
c         psi_r = Q_r/A, where:
c             Q_r = sum over active nodes i,j in subbasin [rech(i,j)*area(i,j)]
c             A   = sum over active nodes i,j in subbasin [area(i,j)]
      lcrchi=isum
      ispj = ncol*nrow
      isum=isum+ispj
c         total space used in x by this package:
      ISP=ISPA+ISPB+ISPD+ispi+ispj
C
C9------PRINT AMOUNT OF SPACE USED BY THIS PACKAGE.
```

```
      WRITE (IOUT,8)ISP
    8 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED FOR SWB')
      ISUM1=ISUM-1
      WRITE(IOUT,9)ISUM1,LENX
    9 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I7)
      IF(ISUM1.GT.LENX) WRITE(IOUT,10)
   10 FORMAT(1X,'    ***X ARRAY MUST BE DIMENSIONED LARGER***')
C
C10-----RETURN.
      RETURN
      END
```

Subr. Swb2rp: initialize watershed connections to stream-aquifer grid

```
c
      subroutine SWB2RP (iunit,SHED,ISHED,IBSHED,ibound,delr,delc,
     1 swtflx,areain,mxstrm,nwshed,iopmod,iopswt,irropt,ievopt,
     1 ioprch,rchmpy,evapir,welmpy,iadcod,frseep,itmuni,filbal,
     1 nstp,nlay,nrow,ncol,nper,iout)
c
c usage in Modflow main:
c       IF(IUNIT(6).GT.0)
c    1    call SWB2RP (iunit,shed,x(ICSHED),x(LCBSHD),X(LCIBOU),
c    1      X(LCDELR),X(LCDELC),x(LCwshd),areain,mxstrm,nwshed,
c    1      iopmod,iopswt,irropt,ievopt,ioprch,rchmpy,evapir,welmpy,
c    1      iadcod,frseep,itmuni,filbal,nstp,nlay,nrow,ncol,nper,iout)
c
C-----VERSION  1   1May1995 SWB2RP
C     *****************************************************************
C     READ watershed DATA:  INCLUDES:
c     a) list of watersheds: corresponding areas and outflow location
c     b) association matrix: subbasin ID For each grid cell
C     *****************************************************************
C
C     SPECIFICATIONS:
C     -----------------------------------------------------------------
      character filbal*(*) !default name for balance file passed from main
      character aname*4, nambal*30, recinp*132
c       time step options: indexed by iopmod (value from 0 to 3)
      character*10 opstep(0:3)
      dimension tsec(0:5)

      DIMENSION iunit(24),SHED(30,0:nwshed),ISHED(10,0:nwshed),
     1   IBSHED(ncol,nrow),ibound(ncol,nrow,nlay), DELR(ncol),
     1   DELC(nrow),swtflx(30),aname(6,1)

      DATA ANAME(1,1),ANAME(2,1),ANAME(3,1),ANAME(4,1),ANAME(5,1),
     1 ANAME(6,1) /'Subb','basi','n gr','id m','atri','x   '/

      data (opstep(j),j=0,3) /'avg annual','annual','monthly','daily'/
C     -----------------------------------------------------------------
c         time period (sec) for each time unit specified in Modflow by itmuni
c         (use 1 here for unspecified option itmuni=0); itmuni is returned
c         from initializing call to Modflow.
c                    undefined, sec, min, hour,  day,    year(365 days)
      data (tsec(j),j=0,5) /1.,    1., 60., 3600., 86400., 31536000./

C
      in = iunit(6)
C
      print '(5(1x,a,i5))','SWB2RP: nwshed=',nwshed,'ncol=',ncol,
     1     'nrow=',nrow,'nlay=',nlay,'mxstrm=',mxstrm,'nper=',nper,
     1     'nstp=',nstp,'in=',in
c
c         This routine initializes variables that are to remain
c         constant over all stress periods:
        read (in,*) nambal,irropt,ievopt,ioprch,rchmpy,evapir,
     1      welmpy,iadcod,frseep                            !Line 2
```

```
          print '(5(1x,a,i5))','SWB1AL: nwshed=',nwshed,
     1      'irropt=',irropt,'ievopt=',ievopt,'ioprch=',ioprch,
     1      'iadcod=',iadcod
c         option to use balance file default name based on case name from main:
        if (nambal.eq.' ') nambal = filbal

        print '(1x,a,i3,1x,a,f12.1)','Modflow itmuni=',itmuni,
     1        'tsec(itmuni)=',tsec(itmuni)

        nsbact=0
        ilay=1         !assume layer 1
        contrb=0.      !calculate average contributing fraction for basin

c           for each subbasin read idx, isact, irow, icol, isbnxt:
        read (in,'(1x,a)') recinp  !heading
cccc    write (iout,'(1x,a)') recinp
        do i=1,nwshed
          read (in,*) idx,ished(1,i),ished(4,i),ished(5,i),ished(3,i)
        end do

c         read watershed zones array (ibshed) to the STREAM file (iostr):
        CALL U2DINT(ibshed,ANAME(1,1),NROW,NCOL,1,IN,IOUT)

c Note: use subr Srf2rp in this package to associate pour point (irow,icol)
c with stream reach.  Srf2rp is called after points of diversion have been
c defined in subr Wel2rp and the stream network has been defined in Str2rp.
c
        do i=1,nwshed
          isact = ished(1,i)       !1 indicates active subbasins
          isbnxt = ished(3,i)      !index to next subbasin in routing sequence
          ished(2,i)=0 !initialize reference to stream reach for subbasin outflow
          ished(6,i)=0      !initialize no. active grid cells
          shed(3,i)=0.        !initialize active gridded area
          ished(8,i)=0
          ished(9,i)=0
          shed(26,i)=0.
          shed(27,i)=0.
          if (isact.gt.0) then
            nsbact = nsbact + 1 !count no. of "active" subbasins
            if (isbnxt.eq.0) then
              ished(3,i) = i
            else
              ished(3,i) = isbnxt
            end if
          end if    ! isact > 0
        end do
        close (unit=in)

c         Read initial data from balance file; for linked execution (iopswt=1),
c         these data are passed to Modflow from Swat by reference in arg. lists.
c         iopmod and iopswt are read from top of *.swb file (1st or 2nd line)
        if (iopswt.eq.0) then
          print '(1x,a)','Open balance file '//nambal
          open (unit=in,file=nambal,status='unknown')
c
c             The following read corresponds to a write in Swatmod3.h:
c             read rec. 1 of balance file:
          read (in,*) nyrs,iopmod,areain   ! '(2i5,f15.0)'
c             Read heading for subbasin fractions and initial conditions:
          read (in,'(a)') recinp
          swtflx(8)=0.
c           for each active subbasin, read areal fraction, noncontributing fraction:
          do i=1,nwshed
            if (ished(1,i).gt.0) read (in,*) idx, shed(1,i), shed(8,i)
          end do
c              Read heading for fluxes to be read in SWB2FM:
          read (in,'(a)') recinp
        end if
        print '(1x,a)', opstep(iopmod) // 'time steps'
c             convert basin area from units in balance file to Modflow units:
        swtflx(2) = areain   !basin area: areain (ft^2)
```

```
c
        swtflx(8)=0.  !initialize noncontributing area as fraction of basin
        do i=1,nwshed
            if (ished(1,i).gt.0) then
              swtflx(8) = swtflx(8) + shed(1,i)*shed(8,i) !noncontrib. fract.
c                                  runoff goes to ponds in this fraction.
              shed(2,i) = shed(1,i)*swtflx(2)  !subbasin area (ft^2)
              cntfrc = 1. - shed(8,i) !contributing fraction = 1 - noncontrib. fraction
              contrb = contrb + shed(1,i)*cntfrc !mean contributing fraction
c                            weighted by subbasin area (fraction of basin)
            end if
          end do
c
        print '(2(1x,a,i3))','SWB2RP: no. active subbasins =',nsbact

c         accumulate gridded area by subbasin:
        nactsm=0              !no. active grid cells
        activa=0.             !active grid cell area
        do ir=1,nrow
          do ic=1,ncol
            if (ibound(ic,ir,1).gt.0) then
              nactsm=nactsm+1                         !no. active nodes in basin
              izone = ibshed(ic,ir)                   !subbasin corresp. w/ grid cell
              ished(6,izone)=ished(6,izone)+1         !no. active nodes in zone
              acell = delr(ic)*delc(ir)               !area of grid cell
              activa=activa+acell
              shed(3,izone) = shed(3,izone) + acell !active gridded area in zone
            end if
          end do
        end do
        actfrc = activa/swtflx(2)
        swtflx(3) = activa !total active gridded area

        print '(1x,a,f12.5,1x,a,g15.7)', 'areain=',swtflx(2)

        print '(1x,a,i5,1x,a,f9.5)', 'SWB2RP:',
     1    nactsm,'active cells; active areal fraction=',actfrc
        print '(1x,a,f8.5)',
     1    'Noncontributing fraction of basin=',swtflx(8)
        print '(a)', '  sub  act subbas area actgrd area activ frc'//
     1               ' anod nrch strlen,ft acrv stract,ft'//
     1               ' act.nodes srec  row  col  seg  rch'
        do i=1,nwshed
            if (shed(2,i).gt.0.) then
              shed(7,i) = shed(3,i)/shed(2,i)
            else
              shed(7,i) = 0.
            end if
            istrec = ished(2,i)    !index to record for stream segment,reach
            print 130, i,ished(1,i),shed(2,i),
     1        shed(3,i),shed(7,i),ished(6,i),ished(8,i),shed(26,i),
     1        ished(9,i),shed(27,i),istrec
130         format (2i5,2f12.0,f10.6,i5,2(i5,f10.0),i10,5i5)
        end do
      RETURN
      END
```

Subr. Srf2rp: associate pts of subbasin runoff and surface diversions with stream reaches

```
c-----------------------------------------------------------------------------
      subroutine SRF2RP (ished,shed,swtflx,well,idxstr,ibshed,ibound,
     1 kkper,nwshed,mxwell,nwells,nwread,nstrem,nlay,nrow,ncol,iout)
c usage:
c       Associate (a) "pour points", i.e., points of outflow from subbasins,
c       with stream reaches for tributary inflow;
c       and (b) points of diversions with stream reaches for supply.
c       Surface water (streamflow) diversions are included in Well file.
```

```
cc      IF(IUNIT(2).GT.0) CALL SRF2RP(x(ICSHED),shed,x(LCwshd),
cc    1    X(LCWELL),x(LCCstr),x(LCBSHD),X(LCIBOU),kkper,nwshed,
cc    1    mxwell,nwells,nwread,nstrem,nlay,nrow,ncol,iout)

      dimension ISHED(10,0:nwshed),SHED(30,0:nwshed),swtflx(30),
     1          well(20,mxwell),
     1          idxstr(ncol,nrow,nlay),
     1 IBSHED(ncol,nrow),ibound(ncol,nrow,nlay)
c         nradius = the radius (no. grid cells) of search for a stream reach
c         if a well's grid coordinates (col,row,layer) do not correspond to
c         a reach indexed by idxstr(col,row,layer).
      data nradius /3/
c
        ilay=1
c (a) Associate "pour points", i.e., points of outflow from subbasins,
c   with stream reaches for tributary inflow:
        if (kkper.eq.1) then
          print '(1x,a)',' subbasin active(1) strm rec nxt subbas'//
     1                   '  grid row  grid col (subbasin outflow)'//
     1                   '--Srf2rp'
          do j=1,nwshed
            isact = ished(1,j)       !1 indicates active subbasins
            if (isact.gt.0) then
              idxrch = 0
              isbnxt=ished(3,j)    !index to next subbasin in routing sequence
              irow = ished(4,j)    !row of grid cell given as pour point for subbasin i
              icol = ished(5,j)    !col of grid cell given as pour point for subbasin i
c               record corresp to stream segment and reach:
              if (1.le.irow.and.irow.le.nrow .and.
     1            1.le.icol.and.icol.le.ncol)
     1          ished(2,j) = idxstr(icol,irow,ilay)
              print '(6i10)',j,(ished(i,j),i=1,5)
            end if
          end do
        end if
c
c (b) For each ground or surface water diversion, L:
c   If its location corresponds to a stream reach, given by a positive
c   value for idxstr(col,row,layer); and a correspondence between
c   the diversion and a stream reach is not already defined by
c   a positive value for well(10,L) (obtained from the Well input file
c   with modified format in Wel2rp), associate the diversion with the
c   stream reach given by idxstr(col,row,layer).  This applies to both
c   gw and surface water sources.


      write (iout,'(a)') '    Layer       Row    Column  Qwr(cfs)'//
     1     '    pmprat   Dsstrm  Rch Wryr bc=*  Src  Use'
      do L = 1,nwread
        irow=WELL(2,L)
        icol=WELL(3,L)
        ilay=WELL(1,L)
        istrch = idxstr(icol,irow,ilay)
        if (well(10,L).eq.0) well(10,L) = istrch
c           if corresp. stream reach is defined, distance to stream is
c           within the grid cell, so call it half a grid cell:
        if (well(10,L).gt.0) then
c           approximate distance from well to stream reach with a nominal
c           value of 1/2 grid cell width:
          if (well(9,L).eq.0.) well(9,L) = 0.5
        else
c           search for stream reach within nradius grid cells; take first one found.
          do ir = irow-nradius, irow+nradius
            if (1.le.ir.and.ir.le.nrow) then
              do ic = icol-nradius, icol+nradius
                if (1.le.ic.and.ic.le.ncol) then
                  if (ir.ne.irow .or.ic.ne.icol) then
                    istrch = idxstr(ic,ir,ilay)
                    if (istrch.gt.0) then
                      well(10,L) = istrch  !index to stream reach within nradius cells
                      dc = icol - ic
                      dr = irow - ir
                      well(9,L) = SQRT(dc*dc + dr*dr)  !distance to reach (grid coords)
                    end if
                  end if
                end if
              end do
            end if
          end do
        end if
        Qwr = well(5,L)
        ibcflg = well(6,L)
        pmprat = well(13,L)
        isourc = well(7,L)  ! 1:surface, 0:gw (corresponds to DWR source, S(surface),
G(ground)
        iusdwr = well(8,L) !DWR use code (3: irrig; 0: nonirrigation)
        dsstrm = well(9,L)
        istrch = well(10,L)
        iwryr = well(11,L)
        if (well(7,L).gt.0.and.well(10,L).le.0)
     1    write (iout,'(1x,a,i5,a)')'Srf2rp: Surface water '//
     1        'diversion, Well input item L =',L,
     1        ' has no associated stream reach.'
        write (iout,'(3i10,3f10.5,5i5)') ilay,irow,icol,Qwr,
     1      pmprat,dsstrm,istrch,iwryr,ibcflg,isourc,iusdwr
      end do
c-----------------------------------------------------------------------
c               Irrigation (both ground water and surface water diversions):
c         shed(4,j):   total irrigation pumping rate [L^3/T] for each subbasin as
c                      accumulated over wells read from ~.WEL file (MODFLOW)
c
c         For each stress period (assumed to be 1 yr), initialize total pumping
c         rate for each subbasin as specified in MODFLOW's WELL input file (~.wel).
c             The negative sign is introduced to cancel the sign difference
c         between the irrigation flux (positive) and MODFLOW's
c         well flow rate (negative, i.e. pumping out of the aquifer).
c
      swtflx(4)=0.
      do i=1,nwshed
        shed(4,i)=0. ! accumulator for negative of subbasin well pumping rates
      end do
      do L=1,nwells
        il = well(1,L)
        ir = well(2,L)
        ic = well(3,L)
cccc              ibound(ic,ir,il)     !0:inactive; -1: const. head
        izone = ibshed(ic,ir)        !subbasin
        iusdwr = well(8,L) !DWR use code (3: irrig; 0: nonirrigation)
        Q = well(5,L)
        if (ibound(ic,ir,il).gt.0 .and.izone.gt.0 .and.
     1      iusdwr.eq.3.and.Q.lt.0.) then
          swtflx(4)=swtflx(4)-Q
          shed(4,izone) = shed(4,izone) - Q
        end if
      end do
      print '(1x,a,i3,(1x,g15.7,1x,a))',
     1        'SRF2RP: stress period',kkper,swtflx(4),
     1        'total pumping in active nodes and zones'

      return
      end
```

Subr Swb2str: area of basin drained by each stream reach

```
c---------------------------------------------------------------------------
      subroutine SWB2STR (SHED,ISHED,IBSHED,ibound,istrm,strm,
     1  nrcseg,NTRSEG,ITRBAR,areasg,mxstrm,mxseg,mxtrib,nstrem,
     1  nwshed,nlay,nrow,ncol)
c
```

```
c Purpose:  Determine the areal fraction of the basin drained by each stream
c reach.  Method: Follow the routing sequence of stream reaches L used in
c subr Str2fm and accumulate the areal fractions given by strm(26,L)
c corresponding to drained subbasin areas in Swb2rp.

C-----VERSION  1   1May1995 SWB2STR
C     ******************************
C     READ watershed DATA:  INCLUDES:
c     a) list of watersheds: corresponding areas and outflow location
c     b) association matrix: subbasin ID For each grid cell
C     *****************************************************************C
C                                                                      C
C     SPECIFICATIONS:
c         call SWB2STR (x(LCSHED),x(ICSHED),x(LCBSHD),X(LCIBOU),
c    1     x(ICSTRM),x(LCSTRM),x(Lcrseg),x(LTRseg),X(LCTBAR),
c    1     x(Laresg),mxstrm,mxseg,mxtrib,nstrem,nwshed,nlay,nrow,ncol)
c
C     -----------------------------------------------------------------C
      DIMENSION iunit(24),SHED(30,0:nwshed),ISHED(10,0:nwshed),
     1   IBSHED(ncol,nrow),ibound(ncol,nrow,nlay),
     1   ISTRM(10,mxstrm), strm(30,mxstrm),
     1   nrcseg(mxseg),ntrseg(mxseg),ITRBAR(mxseg,mxtrib),areasg(mxseg)
c
        do i=1,nwshed
          if (ished(1,i).gt.0) then
            istrec = ished(2,i)  !index to record for stream segment,reach
            if (1.le.istrec.and.istrec.le.nstrem) then
c                 subbasin associated with tributary floww to stream reach i:
              istrm(10,istrec) = i
c                 accumulate areal fraction of basin drained by this reach:
              strm(26,istrec) = strm(26,istrec) + shed(1,i)
c                 Note: a subsequent call to Swb2str will go through the
c                 routing sequence of stream reaches to give the areal fraction
c                 of the basin drained by each stream reach.
            end if
          end if
        end do
C2------PROCESS EACH CELL IN THE STREAM LIST.
C
        frcbas = 0.
        do ii=1,nstrem
C3------DETERMINE LAYER, ROW, COLUMN OF EACH REACH.
          il=istrm(1,ii)
          ir=istrm(2,ii)
          ic=istrm(3,ii)
          izone = ibshed(ic,ir)
          if (izone.gt.0.and.izone.le.nwshed) then
            ished(8,izone) = ished(8,izone) + 1 !no. cells w/ stream reaches
            shed(26,izone) = shed(26,izone) + strm(12,ii) !stream reach length
c               stream reach length through active grid cells in each subbasin:
            if (ibound(ic,ir,il).gt.0) then
              ished(9,izone) = ished(9,izone) + 1 !no. active cells w/ stream reaches
              shed(27,izone) = shed(27,izone) + strm(12,ii)
            end if
          end if
c               areal fraction of basin drained by each stream reach:
          ISTSG=ISTRM(4,ii)
          NREACH=ISTRM(5,ii)
C
C6------SET FLOWIN EQUAL TO STREAM SEGMENT INFLOW IF FIRST REACH.
          IF(NREACH.eq.1) then
c7------[replaced by c7a, below]--spp
C
c8------IF SEGMENT IS A DIVERSION, COMPUTE FLOW OUT OF UPSTREAM REACH.
cc          IF(IDIVAR(ISTSG).gt.0) then
cc            NDFLG=IDIVAR(ISTSG)     !index to upstream segment (source of diversion)
cc            DUM=ARTRIB(NDFLG)-FLOWIN
cc            IF(DUM.GE.0.0) then
cc              ARTRIB(NDFLG)=DUM     !update outflow from upstream segment
cc              lend = iendsg(ndflg)  !index to strm rec corresp to last reach of
upstream segment
```

```
cc              strm(9,lend) = dum    !update outflow from upstream reach
cc            else
cc              FLOWIN=0.
cc            end if
cc          end if
cc----
C9-----SUM TRIBUTARY OUTFLOW AND USE AS INFLOW INTO DOWNSTREAM SEGMENT.C
c         (Note: inflow < 0 should be a redundant, unnecessary indicator)          if
(ntrseg(istsg).gt.0) then
            DO ITRIB=1,ntrseg(istsg)
              INODE=ITRBAR(ISTSG,ITRIB)
              IF(INODE.gt.0) then
                strm(26,ii) = strm(26,ii) +areasg(INODE)
              end if
            end do
c                                                                   c
C10-----IF REACH >1, SET INFLOW EQUAL TO OUTFLOW FROM UPSTREAM REACH.  C
          else if (nreach.gt.1) then
            strm(26,ii) = strm(26,ii-1) + strm(26,ii)
          else
            print '(2(1x,a,i5))','Swb2Str, rec ii=',ii,
     1         ': nreach = ',nreach,'istsg=',istsg
          end if
          if (nreach.eq.nrcseg(istsg)) areasg(istsg) = strm(26,ii)
        end do  !ii=1,nstrem
      return
      end
c---------------------------------------------------------------------
c
c     test option to treat part of "evaporation from shallow ground water"
c     as seepage to the stream channel (subr. EVT2STR is in MODSWB):
c     NOTE: this option is not coordinated with the coupling option;
c     until this is done, this shouldn't be used with the coupled
c     HRU scheme 3.

c     if (iunit(5).gt.0.and.iunit(13).gt.0 .and. 1.+frseep.gt.1.)
c    1    call EVT2STR (nlay,nrow,ncol,nstrem,mxstrm,numevt,
c    1    x(icstrm),x(Lcstrm),frseep,x(Lcevap)

      subroutine EVT2STR (nlay,nrow,ncol,nstrem,mxstrm,numevt,
     1                    istrm,strm,frseep,evap)

      dimension ISTRM(10,mxstrm),STRM(30,mxstrm),evap(ncol,nrow)
      numevt=0
      do L=1,nstrem
          il = istrm(1,L)
          ir = istrm(2,L)
          ic = istrm(3,L)
          if (MIN0(il,ir,ic).lt.1) then
              print '(1x,a,i5,1x,a,3i5)','stream reach L=',L,
     1            'grid cell not defined: (il,ir,ic)=',il,ir,ic
              stop
          end if
          if (evap(ic,ir).gt.0.) then   !baseflow (negative of streambed leakage)
            numevt=numevt+1
c               assume a fraction frseep of evaporation from shallow
c               ground water ends up instead as seepage to the stream.
            strm(18,L) = strm(18,L) + frseep*evap(ic,ir)
          end if
      end do
      return
      end
```

<u>subr Swb2fm: call each time step to specify conditions in terms of watershed simulation</u>

```
c---------------------------------------------------------------------
      subroutine SWB2FM (SHED,ISHED,IBSHED,ibound,istrm,strm,delr,delc,
     1    EVTR,RECH,swtflx,mxstrm,nstrem,nwshed,irropt,ievopt,ioprch,
     1    rchmpy,evapir,iopswt,delt,tsmult,kkper,kkstp,icalyr,
     1    nlay,nrow,ncol,iunit)

c usage in Main:
c         call SWB2FM (shed,x(ICSHED),x(LCBSHD),X(LCIBOU),
c    1      x(ICSTRM),x(LCSTRM),X(LCDELR),X(LCDELC),X(LCEVTR),
c    1      X(LCRECH),x(LCwshd),mxstrm,nstrem,nwshed,irropt,ievopt,
c    1      ioprch,rchmpy,evapir,iopswt,delt,tsmult,kkper,kkstp,
c    1      icalyr,nlay,nrow,ncol,iunit)
c
C-----VERSION  1  26May1995 SWB2FM                                    C
C     ****************************************************************C
C     READ watershed simulation results for current time step.
C     ****************************************************************C
C     SPECIFICATIONS:                                                 C
C     ----------------------------------------------------------------C
      dimension ibound(ncol,nrow,nlay),IBSHED(ncol,nrow),
     1  SHED(30,0:nwshed),ISHED(10,0:nwshed),EVTR(ncol,nrow),
     1  RECH(ncol,nrow),DELR(ncol),DELC(nrow),ISTRM(10,mxstrm),
     1  STRM(30,mxstrm),swtflx(30),iunit(24)

      data itrace /1/
      data itrshl /0/      !option (y=1,n=0) to show evap for indiv. shallow nodes
C     ---------------------------------------------------------------
C
C1A-----IF mxshed IS LESS THAN 1 THEN MODSWB IS INACTIVE:
      IF(nwshed.LT.1) RETURN
      in = iunit(6)
c         initialize the lateral inflow (and diversion) vector in STRM:
      do i=1,nstrem
          strm(18,i)=0.
      end do
      do i=1,nwshed
          shed(17,i)=0.  !initialize recharge assigned to each subbasin
      end do
c-------------------------------------------------------------------
          aqfare = 0.   !initialize cumulative aquifer gridded area as check
          vrchsm=0.    !accumulate recharge volume over basin
          swtflx(25)=0.
c           no. subbasins with no active grid cells but assigned recharge:
          nrcnon=0
c
c           increase in pumping rate required to meet evaporative loss:
          evpfac = 1./(1.-evapir)

c Recharge(17) and tributary flow(22) are calculated in Preswb;
c evap from shallow gw(18) and baseflow(19) are calculated in Modflow, either
c here (below) or in Modstr, in the case of baseflow.  Stream surface evap is
c also calculated in Modstr.  Irrigation requirements according to the
c watershed model's simulation for each subbasin (Swat) are supplied by ground
c and surface water sources, which are calculated in Modwel for each point of
c diversion subject to both operating capacity and water supply constraints.

      if (iopswt.eq.0) then
c             read hydrologic results from watershed simulation
          do j=1,nwshed
            if (ished(1,j).gt.0)    !read fluxes for active subbasins only:
     1        read (in,*) icalyr,imo,ndays,delt,isub,
     1            (shed(i,j),i=10,22),shed(6,j),shed(5,j)
          end do
      end if

c     Note:
c         hydrologic results are passed from Swat array to Modflow array Shed
c         in subr Preswb for subsequent call to Modflow.

      tsmult=1. ! overrides value given in ~.bas file for stress period,
c                 so that time step length will be as specified by calling
c                 routine (SWAT in this case).

      nrcnon=0
      qrcnon=0.
      swtflx(5) = 0. !initialize change in pond storage
      swtflx(6) = 0. !initialize change in soil water storage
      do i=10,22
        swtflx(i)=0.
      end do
      swtflx(25)=0.
      do j=1,nwshed
        if (ished(1,j).gt.0) then   !read fluxes for active subbasins only:

          swtflx(5)   = swtflx(5) + shed(5,j)  !rate of change in pond storage over basin
          swtflx(6)   = swtflx(6) + shed(6,j)  !rate of change in soil water content over
basin
          do i=10,22
c               accumulate total basin flow rate for flux component i:
            swtflx(i) = swtflx(i) + shed(i,j)
          end do
c                 shed(9,j): scaling factor s=Qirr/Qapp, applied to ground
c                 and surface water diversions for irrigation in WELZ2FM;
c       Qirr = assigned irrigation (cfs): shed(11,*), read as subbasin flux
c       Qapp = appropriated irrig. (cfs): shed(4,*), accumulated in SWB1RP.
c                 convert simulated irrigation flux to cfs

          shed(9,j) = 0.
          if (irropt.eq.1 .and. 1.+shed(4,j).ne.1.) then
            shed(9,j) = evpfac*shed(11,j)/shed(4,j)
          else if (irropt.eq.2 .and. 1.+swtflx(4).ne.1.) then
            shed(9,j) = evpfac*swtflx(11)/swtflx(4)
          end if
c-----------------------------------------------------------------------

          aqfare = aqfare + shed(3,j)

c            Tributary flow destination: add to stream as lateral inflow;
c            diversions are subtracted from strm(18,*) in STR2FM as
c            part of the modified streamflow routing procedure, where
c            the supply limit is observed.

          istrec = ished(2,j)          !index to stream (segment,reach)
          if (1.le.istrec.and.istrec.le.nstrem)
     1      strm(18,istrec) = strm(18,istrec) + shed(22,j)
        end if
      end do
c-----------------------------------------------------------------------
      if (1.+swtflx(4).ne.1.) swtflx(9) = swtflx(11)/swtflx(4)
c-----------------------------------------------------------------------
c               Recharge and evaporation:
c         Recharge (L**3/T) = recharge flux [L] * (grid cell area)/delt;
c         similarly for evaporation at ground surface (array evtr).
c         These arrays supersede, for this time step, those based on the
c         recharge and evaporation input files and calculated by
c         RCH1RP and EVT1RP, respectively, for the stress period.
      do irow=1,nrow
        do icol=1,ncol
          if (ibound(icol,irow,1).gt.0) then
            izone=ibshed(icol,irow)
            if (izone.gt.0) then !apply recharge, evap within zones:
              if (1.+shed(3,izone).gt.1.) then
                afract = delr(icol)*delc(irow)/shed(3,izone)
c                 recharge (sum of percolation + xm loss + pond seepage):
                if (ioprch.gt.0)
     1            rech(icol,irow) = afract*shed(17,izone)
c
c                 max evaporation rate [L^3/T]: set at the subbasin's
c                 potential evaporation rate.
                if (ievopt.eq.1 .or. ievopt.eq.3)
     1            evtr(icol,irow) = afract*shed(21,izone)
              end if    ! shed(3,izone) > 0  (subbasin area is positive)
            end if      ! ibshed(icol,irow) > 0     (within a defined subbasin)
          end if        ! ibound(icol,irow,1) > 0   (node is active)
        end do
      end do
c           potential evaporation from stream reaches:
      if (ievopt.ge.2) then
        DO L=1,NSTREM
          IL=ISTRM(1,L)
          irow=ISTRM(2,L)
          icol=ISTRM(3,L)
          izone=ibshed(icol,irow)
          if (izone.gt.0) then
c               evaporation rate [L/T] from stream reach L:
            if (1.+shed(2,izone).gt.1.)
     1        strm(21,L) = shed(21,izone)/shed(2,izone)
          end if
        end do
      end if
      RETURN
      end
```

subr. Swb2bd: summarize results for output file *.swm and to be returned to Swat

```
c-----------------------------------------------------------------------------
      subroutine SWB2BD (SHED,ished,IBSHED,ibound,istrm,strm,swtflx,
     1  dtw,evap,delr,delc,itoprc,ibotrc,mxstrm,nstrem,nwshed,delt,
     1  timsim,volnet,srfdiv,qoprds,greduc,sreduc,ngwred,nswred,
     1  kkper,kkstp,icalyr,nlay,nrow,ncol,iopswt,ioswm,ioshl)
c
c usage:  call soil water balance module from main subroutine modflo().
c
C-----VERSION  1  26May1995 SWB2BD                                   C
      DIMENSION SHED(30,0:nwshed),ISHED(10,0:nwshed),IBSHED(ncol,nrow),
     1  ibound(ncol,nrow,nlay),ISTRM(10,mxstrm),STRM(30,mxstrm),
     1  swtflx(30),dtw(ncol,nrow),evap(ncol,nrow),
     1  delr(ncol),delc(nrow)
      dimension volnet(2,20) !initlz. in BAS1OT; pass to swb2bd.
c
C1A-----IF mxshed IS LESS THAN 1 THEN MODSWB IS INACTIVE:
      IF(nwshed.LT.1) RETURN
c
c         calculate baseflow for each subbasin as a flux rate (Modflow L/T)
c         with respect to total subbasin area:
c     print '(1x,a,i5/,1x,a)', 'SWB1BD: nstrem =',nstrem,
c    1      'zone     L    il    ir    ic iseg irch strflow strmLkg'
c
c begin summary of evaporation from ground water
      do izone=1,nwshed
          ished(6,izone)=0   !no. active grid cells in subbasin
          ished(7,izone)=0   !no. grid cells with shallow gw
          shed(18,izone)=0.  !evaporation from shallow ground water
          shed(19,izone)=0.  !baseflow (negative of streambed leakage)
          shed(26,izone)=0.  !areal fraction of active grid cells with
c                             evaporation from shallow gw
          shed(3,izone)=0.  !area of active grid cells this time step
          shed(7,izone)=0.  !active grid cells as fraction of subbasin
          shed(28,izone)=0.  !mean dtw of shallow gw (cells w/ evap-gw)
          shed(29,izone)=0.  !mean dtw  of deep gw (cells w/o evap-gw)
      end do
      swtflx(26)=0.
      swtflx(3)=0.
      swtflx(28)=0.
      swtflx(29)=0.
c 26       areal fraction of active grid cells with shallow ground water
c 27       area of active grid cells for this time step
c 28       mean DTW for gw nodes
      numevt=0
      do ir=1,nrow
        do ic=1,ncol
          if (ibound(ic,ir,1).gt.0) then
            izone = ibshed(ic,ir)
            if (izone.gt.0) then
              ished(6,izone) = ished(6,izone) + 1
              acell = delr(ic)*delc(ir)
              if (evap(ic,ir).gt.0.) then   !baseflow (negative of streambed leakage)
                numevt=numevt+1
                ished(7,izone) = ished(7,izone) + 1
                shed(18,izone) = shed(18,izone) + evap(ic,ir)
                shed(26,izone) = shed(26,izone) + acell !area of shallow gw
                shed(28,izone) = shed(28,izone) + dtw(ic,ir)
              else
                shed(29,izone) = shed(29,izone) + dtw(ic,ir)
              end if
              shed(3,izone) = shed(3,izone) + acell   !area of active grid
            end if
          end if
        end do
      end do
      do izone=1,nwshed
        nshal = ished(7,izone) !no. cells with dtw < extinction depth (Modevt)
        ndeep = ished(6,izone) - nshal !no. cells  w/ dtw > extinction depth
        if (nshal.gt.0) then
          swtflx(18) = swtflx(18) + shed(18,izone) !total evap from shallow gw
          swtflx(26) = swtflx(26) + shed(26,izone) !area of shallow gw
          shed(28,izone) = shed(28,izone)/FLOAT(nshal) !arith.mean DTW of shallow gw
        end if
        if (ndeep.gt.0)
     1    shed(29,izone) = shed(29,izone)/FLOAT(ndeep) !arith.mean DTW of deep gw
        if (ished(6,izone).gt.0) then
          swtflx(3) = swtflx(3) + shed(3,izone) !active gridded area of gw
          shed(26,izone) = shed(26,izone)/shed(3,izone) !shallow gw area as fraction of
active grid
          shed(7,izone) = shed(3,izone)/shed(2,izone) !active area as fraction of
subbasin
        end if
      end do
      swtflx(26) = swtflx(26)/swtflx(3) !areal fraction of shallow gw
c end summary of evaporation from ground water
      Qnet=0.
      evpstr=0.
      strflo=0.
      swtflx(19)=0.
      do L=1,nstrem
          il = istrm(1,L)
          ir = istrm(2,L)
          ic = istrm(3,L)
          if (MINO(il,ir,ic).lt.1) then
              print '(1x,a,i5,1x,a,3i5)','stream reach L=',L,
     1            'grid cell not defined: (il,ir,ic)=',il,ir,ic
              stop
          end if
          izone = ibshed(ic,ir)
          if (izone.gt.0) then    !baseflow (negative of streambed leakage)
              shed(19,izone) = shed(19,izone) - strm(11,L) ! baseflow for subbasin
              swtflx(19) = swtflx(19) - strm(11,L)          ! baseflow for basin
              evpstr = evpstr + strm(22,L)  !evap from stream reach (if ievopt = 2
          end if
          Qnet = Qnet + (strm(10,L) - strm(9,L))
      end do
      if (1.le.itoprc.and.itoprc.lt.ibotrc.and.ibotrc.le.nstrem)
     1    stryld = strm(9,ibotrc) - strm(9,itoprc)
      if (1.+swtflx(2).gt.1.) then
        actfrc = swtflx(3)/swtflx(2)
      else
        actfrc = 0.
      end if
c
c         note: swtflx(22) is net tributary inflow, including surface &
c         subsurface runoff and surface water diversions.
c
c         yield = sum of net tributary flow (runoff - diversions),
c                 evaporation from stream surface, and baseflow):
      yield = swtflx(22) + evpstr + swtflx(19)

c       net change in watershed stg = P(precip) - E(evap) - R(yield) - G(net gw inflow)
c       P is an areal avg of NWS data; R is calculated but can be compared
c       with the difference between values at Concordia and Clay Center given
c       by USGS gage measurements averaged over the time step (month).

      gwnet = volnet(2,2)   ! net gw inflow = constant heads net inflow
c     net flux  = precip     - evap     - yield + gwnet:
      dsshed = swtflx(10) - swtflx(12) - yield + gwnet
c         net change in soil water storage:
c         d(sw)/dt = infilt. - evap. - subsurf. lat. flow - perc_rz (+ et_gw)

      dsgw = -volnet(2,1)        !rate of change in gw storage, d(gw)/dt
      dsstg = swtflx(6) + dsgw + swtflx(5)  !sw, gw, ponds
      dscrep = dsshed - dsstg

      if (kkper.eq.1.and.kkstp.eq.1) write (ioswm,'(a)') '  per  stp'//
     1  '      timsim    actfrc  Qout-Qin net yield trbfl(22)'//
     1  '    srfdiv    evpstr basfl(19) rchnaq:25 et-gw(18)'//
     1  ' Qirr/Qapp tgwirr:24 tgwnon:23'//
     1  ' Opr.reduc gw reduct sw reduct n.gw n.sw'//
     1  '     gwnet     dsshed      dsgw     dspond     dssoil'//
     1  '     dsstg     dscrep'

      write (ioswm,'(2i5,g12.4,f10.6,14f10.2,2i5,8f10.2)')
     1 kkper,kkstp,timsim,actfrc,stryld,yield,swtflx(22),
     1 srfdiv,evpstr,swtflx(19),swtflx(25),swtflx(18),swtflx(9),
     1 swtflx(24),swtflx(23),Qoprds,Greduc,Sreduc,ngwred,nswred,
     1 gwnet, dsshed, dsgw, swtflx(5), swtflx(6), dsstg, dscrep

      if (iopswt.eq.0) then
c         write results for evaporation from shallow ground water for
c         active subbasins, i.e., ished(1,j) > 0, as with balance file.
        if (kkper.eq.1.and.kkstp.eq.1) write (ioshl,'(a)')
     1  ' year  per  stp  sub  act shal'//
     1  ' frc shall frc activ   evap-gw  shal dtw  deep dtw'
        do j=1,nwshed
          if (ished(1,j).gt.0)
     1      write (ioshl,'(6i5,2f10.7,3f10.2)')
     1         icalyr,kkper,kkstp,j,ished(6,j),ished(7,j),
     1         shed(26,j),shed(7,j),shed(18,j),(shed(i,j),i=28,29)
        end do
      end if
      RETURN
      END
```

Modstr: modified Stream package (file modstr.for)

Definitions

```
c file modstr.for  perkins & Sophocleous
c modified to allow either diffusive wave or Prudic's stream routing
C                                                                    C
C definitions for vectors in ISTRM and STRM: (from STR1AL on modstrdw, modswb96)
c     stream data in ISTRM (increased from 5 to 10 vectors):
c  1 layer   k
c  2 row     i
c  3 column  j
c  4 segment
c  5 reach
c added:
c  6 no. of pumping wells for which this is closest stream reach.
c  7 indicates whether reach is active this time step.
c  8 indicates whether Muskingum-Cunge (1) or Prudic-type (0) routing
c    was applied to the reach.
c  9 IQFLG, assigned for each reach in STR2FM as follows:
c      0: if head in aquifer is above the streambed bottom elevation;
c      1: if either (a) head in aquifer is below the streambed bottom
c         elevation or (b) leakage from streambed to aquifer exceeds
c         reach inflow + surface lateral inflow to reach, in which case
c         leakage is reset to the sum of these.
c    NOTE: the value of IQFLG assigned in the second iteration of the
c    solution is used to determine the state of coupling (0 for coupled,
c    1 for uncoupled) for succeeding iterations in a given time step.
c 10 indicates whether subbasin ISUB's point of outflow is associated
c    with this stream reach (irec) and, if so, istrm(10,irec) = isub.
c

c Array STRM(j,L) vector list j=1 to 30 for stream reaches L=1 to mxstrm:
c     (stream data array STRM was increased from 11 to 30 vectors.)
c  1 streamflow input (specified for segment's top reach)
c  2 stage (computed): avg for reach
c  3 bed conductance, cstr = strm(3,L):
c             if irdcnd = 0 then
c               cstr is applied as read from input file;
c             else cstr is calculated; see strm(14,L).
c  4 hb=bed bottom elev
c  5 ht=bed top elev
c  6 W = stream base width
c  7 bed longitudinal slope
c  8 n=Manning coef
c  9 reach outflow
c 10 reach inflow
c 11 ql = leakage (rev.: calculate as time-centered.)
c added:
c 12 L = reach length
c 13 if icalc=2 (call subr YQ for depth): reciprocal
c                  side slope for symmetrical trapezoidal channel (cot alpha);
c             if icalc=3 (call subr YQGAGE for depth): index to gaging station
c                  as follows for now: 1=Concordia, 2=Clay Center on Republican R.,
c                  3=Albert KS gage on Walnut Creek.
c 14 Ks = streambed hydraulic conductivity, applied to flow in main
c             channel (perim < wdmain); see strm(3,*)
c 15 Ks_high: hydraulic conductivity applied to flow outside main
c             channel, perim > wdmain, where
c             perim = wetted perimeter, and wdmain is width specified for
c             low hydraulic conductivity channel
c 16 possibly a 2nd Manning coefficient for flow in side channels
c               when perim > wdmain as in (15).
c 17 stream depth (STR2FM)
c 18 qs = net lateral inflow due to surface runoff and interflow;
c                  includes diversions for appropriated water use (see
c                  SURFACE package), but not baseflow; see strm(11,L).
```

```
c 19 main channel width: set to twice the base width, strm(6,L), in STR2RP.
c 20 hydraulic gradient: if iqflg=0, hygrad = hstr-hnew; otherwise, hstr-strm(4,*)
c 21 potential evaporation for time step as depth[L] = rate[L/T]*delt
c 22 evaporation for time step subtracted from reach as flow rate (STR2FM)
c 23 (*) reach outflow for previous time step's solution
c 24 (*) reach inflow for previous time step's solution
c 25 orig. streambed top elevation
c 26 fraction of subbasin drained by this reach
c 27 head on streambed (Str2rp, 410)
c
c Proposed implementation of streamflow routing method to account for
c storage with a modified version of Prudic's routing for a solution
c with daily time steps that does not meet conditions for applying
c Muskingum-Cunge routing
c
c 28-30: experimental, to model storage in stream channels under routing
c    option (?), for a time step delt that is short enough that storage
c    effects in streamflow could be significant, but too long to apply
c    Muskingum-Cunge routing, which would typically require time steps
c    in the range of 10-30 minutes to meet grid Peclet and Courant conditions.
c    A daily time step for the Walnut Creek basin model would qualify, since
c    a flood wave would take several days to travel through the basin.
c    For this case, the following variation on Prudic's routing method is
c    to be tested.  Continuity applied to reach L is expressed as
c                      dS/dt = Qin - Qout + Qlat.
c    To incorporate storage in the routing procedure, keep the term on the
c    left, i.e.,
c                      Qout(L,t) = Qin + Qlat - dS/dt,
c    where the additional term on the right is associated with a flood wave
c    upstream approximated by a step change in streamflow at the beginning of
c    the time step given by
c                      dS/dt(L,t) = dQout(L-idist,t-delt),
c     which is determined after the stream-aquifer solution converges at the
c     end of the previous time step as outlined below.
c
c 28 dQout = change in reach outflow with respect to previous time step:
c          = strm(9,L) - strm(23,L).
c    This is updated only after the coupled stream-aquifer solution has
c    converged, i.e., for istrbd=2; and at the end of the routing procedure
c    for reach L.  As a result, the change in storage dS/dt for reach index L
c    for the current time step is given, for a kinematic wave, by dQout for
c    reach index idorig (defined below), which represents the change in
c    outflow upstream at the beginning of the current time step, i.e., the end of the
c    previous time step, that arrives at reach L at the end of this time step.
c
c 29 dS/dt = rate of change in storage for the reach; approximated for a
c    kinematic wave resulting from a step change in streamflow, dQout,
c    occurring upstream at the beginning of the time step (end of the
c    previous time step).
c
c 30 iddest = index to reach associated with the destination of a flood wave,
c    given by the step change dQout (strm(28,L), at the end of this
c    time step and whose leading edge arrives at the downstream reach iddest
c    at the end of the next time step.  This is initialized to zero for every
c    reach L and remains zero unless the travel time from reach L to the
c    basin outflow is less than the solution time step, in which case iddest
c    identifies the reach downstream from reach L corresponding to a travel
c    time of delt.

c    Example:
c    if delt = 86400 s (1 day), avg reach length dx = 2000 ft,
c    and avg ck = 1 ft/s, a kinematic flood wave will travel from reach L
c    a distance s = ck*delt = 86400 ft, or s/dx = 43.2 reaches,
c    so that ircsto = L + idist, where idist = s/dx rounded up.  Note that the
c    distance travelled by the flood wave in terms of reaches, s/dx, is simply
c    the Courant number, i.e., Cr = ck*delt/dx.  So if Cr = 1, i.e., the Courant
c    condition for a kinematic wave solution were met (with delt = 2000 s in
c    this case), then the wave would travel a distance of 1 reach, and the M-C
c    solution would apply.
c
c Array STRMAD(j,L) vector list j=1 to 7 for stream reaches L=1 to mxstrm:
```

```
c j= 1  top width B
c    2  wetted perimeter P
c    3  cross-sectional area of flow A
c    4  avg flow velocity v = Q/A
c    5  travel time tr = dx/v, where dx = reach length
c    6  kinematic wave speed ck = dQ/dA
c    7  hydraulic diffusion Dh = Q/(2*B*S0)
c-----------------------------------------------------------------------
c STREAM package input data changes:
c     (subr's STR1AL, STR1RP, STR1FM, STR1BD)
c
c itmp:
c     > or = 0: itmp = no. reaches active during the current stress period;
c     = -1: use stream data from the previous stress period;
c     = -2: except for NUMINP stream nodes, use stream data from the previous
c           stress period; see NUMINP, below.
c irdflg:
c iptflg:
c irdcnd:
c         stream channel geometry options
c     If irdcnd = 0 (default): streambed conductance, C, is applied as read from
c     field 51:60 of the first record for each reach.  As described in the STREAM
c     manual, a rectangular channel with steady flow is assumed, and input data
c     values for streambed conductance can be approximated by
c         C = Ks*Ls*P/T = strm(3,*), where:
c             Ks = streambed saturated hydraulic conductivity;
c             Ls = stream reach length;
c             P = wetted perimeter;
c             T = strm(5,*)-strm(4,*) = streambed thickness.
c     If irdcnd > 0, then Ks is applied as read from field 51-60 of the second
c     record for each reach, and conductance C is calculated as shown above.
c     For this option, data are read from f10 fields 4 and 5 (columns 31-40
c     and 41-50) of the second record of stream
c     reach input data as described below;
c     Field 4: Reach length, Ls=strm(12,*);
c     Field 5:
c         if icalc=2, read reciprocal side slope for a symmetrical trapezoidal
c             channel, 1/c = strm(13,*); then subr YQ calculates depth given
c             discharge for a trapezoidal channel under unsteady flow
c             conditions.  P is approximated by the stream surface top width,
c             B, if iperim=0, which is set internally in subr STR1DW; otherwise,
c             P is based on a trapezoidal channel.
c         if icalc=3, read a gaging station identifier as follows:
c             1=Concordia, 2=Clay Center on Repub. R., 3=Albert KS gage on Walnut Creek.
c             subr YQGAGE calculates depth given discharge.
c numinp:
c     If ITMP = -2 (see above), then for each time step (step 1 in STR1RP
c     and remaining steps in STR1FM), streamflow data are read for NUMINP
c     previously defined stream nodes, in the standard format used for the
c     first six variables in the first record read for each reach, as follows:
ccc       READ(IN,'(5i5,f15.0)') ilay,irow,icol,iseg,irch,strflo
c     This allows inflow hydrographs to be updated in each time step.
c iroute:
c     streamflow routing options:
c     0: Apply default stream routing procedure by Prudic, subr STR1FM;
c     1: Apply modified stream routing procedure by Prudic, subr STR1DW;
c     2: Apply diffusive wave routing where Cr-Pe conditions are met;
c        also subr STR1DW.
c nrstep:
c     No. streamflow routing time steps for each aquifer time step: this is
c     assumed to be 1 for all routing options (above).  In the case of routing
c     option 2, the Cr-Pe conditions are checked for each stream reach and
c     time step; if these conditions are met, diffusive routing is applied;
c     otherwise, the modified Prudic routing procedure (option 1) is followed.
c     The tradeoff is that while flood wave travel time is neglected for each
c     reach handled this way, there's no worry over solution errors that can
c     occur for nrstep > 1 under conditions of strong coupling, i.e. for
c     large conductance, C, as described elsewhere (Perkins and Koussis, 1996).
```

Subr. STR2AL: allocate arrays.

```
c-----------------------------------------------------------------------
c
      SUBROUTINE STR2AL(ISUM,LENX,LCSTRM,ICSTRM,LCSMAD,icanst,MXSTRM,
     1  NSTREM,IN,IOUT,ISTCB1,ISTCB2,mxseg,mxtrib,NSS,NTRIB,
     1  NDIV,ICALC,CONST,LCTBAR,LCTRIB,LCIVAR,LSQSEG,LDXSEG,
     1  LTOSEG,LCBSEG,LCESEG,Lrcseg,LTRSEG,LCLSEG,Lactsg,Lacnsg,Lavgsg,
     1  Lminsg,Lmaxsg,Laresg,Ltrvsg,Lckdt,Lsigsg,LCCstr,
     1  Lxcord,Lycord,ncol,nrow,nlay,itoprc,ibotrc,adjbed)
C                                                                      C
C-----VERSION    1 23OCT1987 STR1AL                                    C
C     ****************************************************************C
C     ALLOCATE ARRAY STORAGE FOR STREAMS                               C
C     ****************************************************************C
C                                                                      C
C     SPECIFICATIONS:                                                  C
C     ----------------------------------------------------------------C
C                                                                      C
C1------IDENTIFY PACKAGE AND INITIALIZE NSTREM.                        C
      WRITE(IOUT,1) IN
    1 FORMAT(1H0,'STRM -- STREAM PACKAGE, VERSION 1, 10/23/87',
     1'INPUT READ FROM UNIT',I3)
      NSTREM=0
C                                                                      C
C2------ READ MXSTRM, NSS, NTRIB, ISTCB1, AND ISTCB2:                  C
      if (in.gt.0) then
          READ(IN,3)MXSTRM,NSS,NTRIB,NDIV,ICALC,CONST,ISTCB1,ISTCB2,
     1          itoprc,ibotrc,adjbed
    3     FORMAT(5I10,F10.0,2I10,2i5,f10.0)
ccc       example from repavgm7.str (c:\swat\repub):
ccc     76         1         0         0         3      1.49        -1         0   file
repavgm7.str jun 26 96
ccc     76         0         0         1         1         1         1  itmp irdflg
iptflg irdcnd numinp iroute nrstep
          IF(MXSTRM.LT.1)MXSTRM=1
          IF(NSS.LT.0)NSS=0
          mxseg = MAX0(nss,1)
          mxtrib= MAX0(ntrib,1)
      else
          mxstrm = 1
          mxseg  = 1
          mxtrib = 1
          nss = 0
          ntrib = 0
      end if
      WRITE(IOUT,4)MXSTRM,NSS,NTRIB
    4 FORMAT(1H ,'MAXIMUM OF',I5,' STREAM NODES'//1X,'NUMBER OF STREAM S
     1EGMENTS IS ',I5//1X,'NUMBER OF STREAM TRIBUTARIES IS ',I5//)
      IF(NDIV.GT.0) WRITE(IOUT,5)
    5 FORMAT(1H ,'DIVERSIONS FROM STREAMS HAVE BEEN SPECIFIED')
      IF(ICALC.GT.0) WRITE(IOUT,6) CONST
    6 FORMAT(1H ,'STREAM STAGES WILL BE CALCULATED USING A CONSTANT OF
     1',F10.4)
      IF(ISTCB1.GT.0) WRITE(IOUT,7) ISTCB1,ISTCB2
    7 FORMAT(1X,'CELL BUDGETS WILL BE SAVED ON UNITS',I3,'AND',I3)
C                                                                      C
C3------SET LCSTRM EQUAL TO ADDRESS OF FIRST UNUSED SPACE IN X.        C
  200 LCSTRM=ISUM
C4------CALCULATE AMOUNT OF SPACE NEEDED FOR STRM LIST.                C
      ISPA=30*MXSTRM
      ISUM=ISUM+ISPA
C                                                                      C
C5------CALCULATE AMOUNT OF SPACE NEEDED FOR ISTRM LIST.               C
      ICSTRM=ISUM
      ISPB=10*MXSTRM
      ISUM=ISUM+ISPB
C5------CALCULATE AMOUNT OF SPACE NEEDED FOR STRMAD LIST.              C
      LCSMAD=ISUM
      ISPB1=10*MXSTRM
      ISUM=ISUM+ISPB1
C5------CALCULATE AMOUNT OF SPACE NEEDED FOR anstrm LIST.               C
      icanst=ISUM
      ispb2=2*MXSTRM
      ISUM=ISUM+ispb2
C                                                                      C
C6------CALCULATE AMOUNT OF SPACE NEEEDED FOR ITRBAR LIST.              C
      LCTBAR=ISUM
      ispc = mxseg*mxtrib
      ISUM=ISUM+ISPC
C                                                                      C
C7------CALCULATE AMOUNT OF SPACE NEEDED FOR ARTRIB LIST.               C
      LCTRIB=ISUM
      ispd = mxseg
      ISUM=ISUM+ISPD
C                                                                      C
C8------CALCULATE AMOUNT OF SPACE NEEDED FOR IDIVAR LIST.               C
      LCIVAR=ISUM
      ISPE  =mxseg
      ISUM  =ISUM+ISPE
cx------(added): space for isqseg(nss), ibgseg(nss), iendsg(nss),
cx               nrcseg(nss), ntrseg(nss), seglen(nss), Lactsg(nss),
cx               i.e., 8*mxseg:
      LSQSEG=isum    !isqseg(nss), routing sequence of segment for strm
      isum=isum+mxseg
      LDXSEG=isum    !idxseg(nss), inverse of isqseg: index to segment corresp. to its
routing sequence
      isum=isum+mxseg
      LTOSEG=isum    !itoseg(nss), destination segment to which this segment drains
      isum=isum+mxseg
      LCBSEG=isum    !ibgseg(nss), index to 1st reach in each segment for strm
      isum=isum+mxseg
      LCESEG=isum    !iendsg(nss), index to last reach in each segment for strm
      isum=isum+mxseg
      Lrcseg=isum    !nrcseg(mxseg), no. reaches in each segment j=1,mxseg
      isum=isum+mxseg
      LTRSEG=isum    !ntrseg(mxseg), no. tributary segments into top reach of segment
j=1,mxseg
      isum=isum+mxseg
      LCLSEG=isum    !seglen(mxseg), segment length
      isum=isum+mxseg
      Lactsg=isum    !iactsg(mxseg), indicate active segments (reaches w/ nonzero Qin,
Qout, or Qlkg)
      isum=isum+mxseg
      Lacnsg=isum    !nacseg(mxseg), no. reaches w/ positive ibound array values
      isum=isum+mxseg
      Lavgsg=isum    !dhavsg(mxseg), avg head on streambed h_bed - h_aqf for active
aquifer nodes
      isum=isum+mxseg
      Lminsg=isum    !dhmnsg(mxseg), min head on streambed h_bed - h_aqf for active
aquifer nodes
      isum=isum+mxseg
      Lmaxsg=isum    !dhmxsg(mxseg), max head on streambed h_bed - h_aqf for active
aquifer nodes
      isum=isum+mxseg
      Laresg=isum    !areasg(mxseg), basin area drained by this segment (sum of subbasins
draining to this segment)
      isum=isum+mxseg
      Ltrvsg=isum    !tvtmsg(mxseg), travel time for kinematic wave for simulated flow
rate this time step
      isum=isum+mxseg
      Lckdt=isum     !ckdtsg(mxseg), mu = 0.5*(ck1+ckn)*delt: approx. avg for segment w/
n reaches
      isum=isum+mxseg
      Lsigsg=isum    !sigseg(mxseg), sigma = 2*SQRT[0.5*(D1+Dn)*delt]: approx. avg for
segment w/ n reaches
      isum=isum+mxseg
c         set up (x,y)-coordinates in subr STR2SG:
      Lxcord=isum    !xcoord(ncol), x-coordinate of cell centers for each column along a
row
      isum=isum+NCOL
      Lycord=isum     !ycoord(nrow), y-coordinate of cell centers for each column along a
row
      isum=isum+NROW
      nrcl = ncol*nrow*nlay     !no. nodes in aquifer grid
      LCCstr=isum  !idxstr(nrcl): index to stream reach for each cell in array
      isum = isum + nrcl
      ISP=ISPA+ISPB+ispb1+ispb2+ISPC+ISPD+ISPE+9*mxseg+nrcl
C                                                                       C
C9------PRINT AMOUNT OF SPACE USED BY STREAM PACKAGE.                   C
      WRITE (IOUT,8)ISP
    8 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED FOR STREAMS')
      ISUM1=ISUM-1
      WRITE(IOUT,9)ISUM1,LENX
    9 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I7)
      IF(ISUM1.GT.LENX) WRITE(IOUT,10)
   10 FORMAT(1X,'    ***X ARRAY MUST BE DIMENSIONED LARGER***')
C                                                                       C
C10-----RETURN.                                                         C
      RETURN
      END
```

Subr. STR2RP: read data to begin each stress period.

```
c
       SUBROUTINE STR2RP(STRM,ISTRM,NSTREM,MXSTRM,itmp,ncol,nrow,nlay,
     1  IN,IOUT,ITRBAR,NDIV,mxseg,mxtrib,NSS,NTRIB,IDIVAR,
     1  isqseg,idxseg,itoseg,ibgseg,iendsg,nrcseg,ntrseg,seglen,
     1  nacseg,dhavsg,dhmnsg,dhmxsg,areasg,ibound,idxstr,
     1  hold,isgtop,isgbot,ICALC,IPTFLG,
     1  kkper,irdcnd,numinp,iroute,nrstep,adjbed)
c
cc      CALL STR2RP(X(LCSTRM),X(ICSTRM),NSTREM,mxstrm,itmpst,ncol,nrow,
cc   1  nlay,IUNIT(13),IOUT,X(LCTBAR),NDIV,mxseg,mxtrib,NSS,NTRIB,
cc   1  X(LCIVAR),x(LSQseg),x(LDXseg),x(LTOseg),x(LCBseg),
cc   1  x(LCEseg),x(Lrcseg),x(LTRseg),x(LCLseg),x(Lacnsg),x(Lavgsg),
cc   1  x(Lminsg),x(Lmaxsg),x(Laresg),x(LCIBOU),x(LCCstr),
cc   1  x(LCHOLD),isgtop,isgbot,ICALC,IPTFLG,
cc   1  kkper,irdcnd,numinp,iroute,nrstep,adjbed)

C-----VERSION  1   23OCT1987  STR2RP                                     C
C      *****************************************************************C
C      READ STREAM DATA:  INCLUDES SEGMENT AND REACH NUMBERS, CELL       C
C          SEQUENCE OF SEGMENT AND REACH, FLOW INTO MODEL AT BOUNDARY,  C
C          STREAM STAGE, STREAMBED CONDUCTANCE, AND STREAMBED TOP AND   C
C          BOTTOM ELEVATIONS                                            C
C      *****************************************************************C
C
C      SPECIFICATIONS:
C      -----------------------------------------------------------------C
       DIMENSION STRM(30,MXSTRM),ISTRM(10,MXSTRM),ITRBAR(mxseg,mxtrib),
     1   IDIVAR(mxseg),isqseg(mxseg),idxseg(mxseg),
     1   itoseg(mxseg),ibgseg(mxseg),iendsg(mxseg),nrcseg(mxseg),
     1   ntrseg(mxseg),seglen(mxseg),nacseg(mxseg),dhavsg(mxseg),
     1   dhmnsg(mxseg),dhmxsg(mxseg),areasg(mxseg),
     1   ibound(ncol,nrow,nlay),idxstr(ncol,nrow,nlay),
     1   hold(ncol,nrow,nlay)
       character*8 txtout
C      -----------------------------------------------------------------C
C
C1A-----IF MXSTREAM IS LESS THAN 1 THEN STREAM IS INACTIVE. RETURN.     C
       IF(MXSTRM.LT.1) RETURN
C
C1B-----READ ITMP(NUMBER OF STREAM CELLS OR FLAG TO REUSE DATA).        C

       READ(IN,'(8i10)') ITMP,IRDFLG,IPTFLG,
     1    irdcnd,numinp,iroute,nrstep,isgtop
    1 FORMAT(9I10)
       print '(9i10)', ITMP,IRDFLG,IPTFLG,
     1     irdcnd,numinp,iroute,nrstep,isgtop
c
C2A-----IF ITMP <0 THEN REUSE DATA FROM LAST STRESS PERIOD.              C
      IF(ITMP.GE.0)GO TO 50
      WRITE(IOUT,2)
    2 FORMAT(1H0,'REUSING STREAM NODES FROM LAST STRESS PERIOD')
      if (itmp.eq.-2) then
          write (iout,'(/,1x,a,i3,1x,a)')
     1        'STR1RP: Read',numinp,'stream nodes to update '//
     1        'stream inflow.'
          do j = 1,numinp
              READ(IN,'(5i5,f15.0)') ilay,irow,icol,iseg,irch,strflo
c               look for previously defined occurrence of this stream reach:
              ii = idxstr(icol,irow,ilay)
              if (1.le.ii.and.ii.le.nstrem) then
                strm(1,ii) = strflo
                print '(5i5,f15.0,1x,a,i5)',
     1            ilay,irow,icol,iseg,irch,strflo,'ii=',ii
              else
                print '(a/,a)',' Str2rp: stream reach not defined '//
     1            'for updated streamflow:',
     1             ' kper kstp  lay  row  col  seg  rch streamflow'
                istp=1
                print '(7i5,f15.7)',
     1              kkper,istp,ilay,irow,icol,iseg,irch,strflo
              end if
          end do
      end if
      RETURN
C                                                                       C
C2B-----IF ITMP=> ZERO THEN IT IS THE NUMBER OF STREAM REACHES.         C
   50 NSTREM=ITMP
C                                                                       C
C3A-----IF NSTREM>MXSTRM THEN STOP.                                     C
      IF(NSTREM.LE.MXSTRM)GO TO 100
      WRITE(IOUT,99)NSTREM,MXSTRM
   99 FORMAT(1H0,'NSTREM(',I4,') IS GREATER THAN MXSTRM(',I4,')')
      STOP
C                                                                       C
C3B-----PRINT NUMBER OF STREAM CELLS IN THIS STRESS PERIOD.              C
  100 IF(IRDFLG.EQ.0) WRITE(IOUT,3)NSTREM
    3 FORMAT(1H0,//1X,I5,' STREAM NODES')
C                                                                       C
C4------IF THERE ARE NO STREAM CELLS THEN RETURN.                        C
      IF(NSTREM.EQ.0) RETURN
      data initlz /0/
      if (initlz.eq.0) then
          initlz=1
          if (iroute.eq.0) then
              print '(1x,a)','Apply default stream routing procedure'//
     1              ' by Prudic, subr STR2FM.'
          else if (iroute.eq.1) then
              print '(1x,a)','Apply modified stream routing procedure'//
     1              ' by Prudic, subr STR2FM.'
          else
              print '(1x,a)','Apply diffusive wave routing '//
     1              'where Cr-Pe conditions are met.'
          end if
      end if
      do iseg = 1,nss
        nacseg(iseg)=0
        dhavsg(iseg)=0.
        dhmnsg(iseg)=0.
        dhmxsg(iseg)=0.
        areasg(iseg)=0.
      end do
C5------READ AND PRINT DATA FOR EACH STREAM CELL.                        C
      IF(IRDFLG.EQ.0) WRITE(IOUT,4)
    4 FORMAT(1H ,3X,'LAYER   ROW    COL    SEGMENT   REACH   STREAMFLOW
     1     STREAM    STREAMBED     STREAMBED BOT  STREAMBED TOP',
     2 ' htop-hold',/27X,
     2'NUMBER    NUMBER                        STAGE    CONDUCTANCE       ELEVAT
     3ION      ELEVATION',/3X,110('-'))
      DO II=1,NSTREM
c           initialize storage in reach, strm(29,ii), whether or not storage
c           option, iopstg, is invoked for routing and handled by subr Str2stg,
c           which is called by Modflow.
          if (kkper.eq.1) strm(29,ii) = 0.

          READ(IN,5) (istrm(j,ii),j=1,5),(strm(j,ii),j=1,5)
    5     FORMAT (5I5,F15.0,4F10.0)
          il=istrm(1,ii)
          ir=istrm(2,ii)
          ic=istrm(3,ii)

c             add:
          idxstr(ic,ir,il) = ii  !index to stream reach, given aquifer node(ir,ic,il).
c             adjust streambed elevation (sensitivity factor from str2al):
          strm(4,ii) = strm(4,ii) + adjbed
          strm(5,ii) = strm(5,ii) + adjbed
c            for active nodes:
c            head on streambed and statistics by segment (mean, min, max):
          iseg = istrm(4,ii)
          if (ibound(ic,ir,il).gt.0 .and. iseg.gt.0) then
            nacseg(iseg) = nacseg(iseg)+1
            strm(27,ii) = strm(5,ii) - hold(ic,ir,il) !head on streambed
            dh = strm(27,ii)
            write (txtout,'(f8.1)') dh
            dhavsg(iseg) = dhavsg(iseg) + dh
            if (nacseg(iseg).eq.1) then
              dhmnsg(iseg) = dh
              dhmxsg(iseg) = dh
            else
              dhmnsg(iseg) = dhmnsg(iseg) + dh
              dhmxsg(iseg) = dhmxsg(iseg) + dh
            end if
          else
            txtout = ' '
            strm(27,ii) = 0.
          end if
          if (irdflg.eq.0) WRITE(IOUT,6)
     1      (istrm(j,ii),j=1,5), (strm(j,ii),j=1,5),txtout
    6       FORMAT(1X,3X,I4,2I7,2I9,7X,G11.4,G12.4,G11.4,4X,2G13.4,
     1             2x,a)
      end do
C                                                                      C
C6----READ AND PRINT DATA IF STREAM STAGE IS CALCULATED.                C
      if (irdflg.eq.0) WRITE(IOUT,7)
    7 FORMAT(1H0,3X,'LAYER',3X,'ROW',4X,'COL    ',' SEGMENT',3X,
     1      'REACH',8X,'STREAM',13X,'STREAM',10X,'ROUGH',
     1      3x,'length',3x,' side slope ',' hydr. cond.',/27X,'NUMBER',
     1      3X,'NUMBER',8X,'WIDTH',14X,'SLOPE',10X,'COEF.',/3X,110('-'))
      irtseq=0  !initialize segment routing sequence
      do ii=1,nstrem
c          added:
c       13: (icalc=2:) side slope for trapezoidal section;
c           (icalc=3:) index to set of channel flow characteristics
c                                 based on gaging station data, including:
c                                 depth-streamflow, channel width.v. depth,
c                                 channel section area v. depth, and
c                                 main channel width
c       14: Ks_main, used for main channel flow to calculate conductance cstr;
c       15: Ks_side, used to calculate conductance component for
c                       flood flow outside main channel
c       19: main channel width, corresponding to strm(14,*).
c       20: reach length (for irdcnd > 0, i.e. Ks is read);
          if (icalc.gt.0) then
            read (in,'(8f10.0)') (strm(j,ii),j=6,8),
     1              (strm(j,ii),j=12,15)
c               guess for main channel width: base width
            strm(19,ii) = strm(6,ii)
          end if
          iseg = istrm(4,ii)
          ireach = istrm(5,ii)
          if (ireach.eq.1) then
              ibgseg(iseg) = ii      !index to beginning reach of segment
              nrcseg(iseg) = 1       !initialize count of reaches in segment
              irtseq=irtseq+1        !increment segment routing sequence
              isqseg(iseg) = irtseq  !retain segment's routing sequence
              idxseg(irtseq) = iseg  !index to segment corresp. to routing sequence
              seglen(iseg) = strm(12,ii)
          else
              nrcseg(iseg) = nrcseg(iseg) + 1 !continue count of reaches in segment
              seglen(iseg) = seglen(iseg) + strm(12,ii)
          end if
          iendsg(iseg) = ii  !update index to ending reach of segment
          if (irdflg.eq.0) then
            if (irdcnd.eq.0) then
              WRITE(IOUT,9) (istrm(j,ii),j=1,5),(strm(j,ii),j=6,8)
            else
              WRITE(IOUT,9) (istrm(j,ii),j=1,5),(strm(j,ii),j=6,8),
     1          (strm(j,ii),j=12,15)
 9            FORMAT(4X,I4,2I7,2I9,7X,G12.4,4X,G13.4,4X,5G12.4)
            end if
          end if
      end do
C                                                                       C
C7------INITIALIZE ALL TRIBUTARY SEGMENTS TO ZERO.                      C
  300 DO 320 iseg=1,NSS
      DO 320 itrb=1,NTRIB
      ITRBAR(iseg,itrb)=0
  320 CONTINUE
C                                                                       C
C8-----INITIALIZE DIVERSION SEGMENT ARRAY TO ZERO.                      C
      DO 325 iseg=1,NSS
      IDIVAR(iseg)=0
  325 CONTINUE
C                                                                       C
C9-----READ AND PRINT TRIBUTARY SEGMENTS.                               C
      IF(NTRIB.LE.0) GO TO 343
      IF(IRDFLG.EQ.0) WRITE(IOUT,10)NTRIB
   10 FORMAT(1H0,30X,'MAXIMUM NUMBER OF TRIBUTARY STREAMS IS ',I5,//1X,
     1 20X,'STREAM SEGMENT',15X,'TRIBUTARY STREAM SEGMENT NUMBERS')
      DO 340 iseg=1,NSS
      READ(IN,11) (ITRBAR(iseg,JK),JK=1,NTRIB)
   11 FORMAT(10I5)
      IF(IRDFLG.EQ.0) WRITE(IOUT,12) iseg,(ITRBAR(iseg,JK),JK=1,NTRIB)
   12 FORMAT(20X,I5,20X,10I5)
      ntrseg(iseg) = 0
      do jk=1,ntrib
        inode = itrbar(iseg,jk)
        if (inode.gt.0) then
          ntrseg(iseg) = ntrseg(iseg) + 1   !no. tributaries into segment
          itoseg(inode) = iseg              !identify downstream segment for tributary
        else
c                escape loop as soon as zero-valued segment index is encountered:
          go to 340
        end if
      end do
  340 CONTINUE
C                                                                       C
C10----READ AND PRINT DIVERSION SEGMENTS NUMBERS.                       C
  343 IF(NDIV.LE.0) GO TO 350
      IF(IRDFLG.EQ.0) WRITE(IOUT,13)
   13 FORMAT(1H0,10X,'DIVERSION SEGMENT NUMBER',10X,
     1        'UPSTREAM SEGMENT NUMBER')
      DO 345 IK=1,NSS
      READ(IN,14) IDIVAR(IK)
   14 FORMAT(I10)
      IF(IRDFLG.EQ.0) WRITE(IOUT,15) IK,IDIVAR(IK)
   15 FORMAT(20X,I5,28X,I5)
  345 CONTINUE
C                                                                       C
```

```
C11----SET FLOW OUT OF REACH, FLOW INTO REACH, AND FLOW THROUGH        C
C      STREAM BED TO ZERO.                                             C
  350 DO 360 II =1,NSTREM
      STRM(9,II)=0.0
      STRM(10,II)=0.0
      STRM(11,II)=0.0
  360 CONTINUE
      do irtseq=1,nss
        iseg=idxseg(irtseq)      !segment corresp. to routing sequence
c           complete calculation of mean head on streambed for each segment:
c           (under conditions of starting heads and no streamflow but
c           saturated streambed)
        if (nacseg(iseg).gt.0)
     1    dhavsg(iseg) = dhavsg(iseg)/FLOAT(nacseg(iseg))
      end do
C                                                                      C
C12------
      RETURN
      END
```

Subr. STR2STP (added): read inflows to specified stream segments in each time step.

```
c
      SUBROUTINE STR2STP(NSTREM,STRM,idxstr,MXSTRM,
     1    NCOL,NROW,NLAY,kkstp,kkper,numinp,in,iout)
c
c usage:
c     CALL STR2STP (NSTREM,X(LCSTRM),x(LCCstr),MXSTRM,NCOL,NROW,NLAY,
c    1      kkstp,kkper,numinp,iunit(13),IOUT)
C
C-----VERSION   2 06FEB1990 STR2STR
C-----CHANGED 4 LINES REGARDING CELLS OUTSIDE MODEL BOUNDARIES
C     ****************************************************************
C     ****************************************************************
C
C     SPECIFICATIONS:
C     ----------------------------------------------------------------
C
      DIMENSION STRM(30,MXSTRM),Idxstr(NCOL,NROW,NLAY)
C     ----------------------------------------------------------------
C
C1------IF NSTREM<=0 THERE ARE NO STREAMS. RETURN.
      IF(NSTREM.LE.0)RETURN
c
      if (kkstp.gt.1 .and. numinp.gt.0) then
          print '(3(1x,a,i4))',
     1        'STR2STP: Read',numinp,'stream nodes to update '//
     1        'stream inflow for time step',kkstp
          print '(a)',' lay  row  col  sg reach     streamflow'//
     1                ' strm rec.'
          do j = 1,numinp
              READ(IN,'(5i5,f15.0)') ilay,irow,icol,iseg,irch,strflo
c               look for previously defined occurrence of this stream reach:
              if (MINO(icol,irow,ilay).ge.1 .and. icol.le.ncol .and.
     1            irow.le.nrow. and. ilay.le.nlay) then
                ii = idxstr(icol,irow,ilay)
              else
                ii = 0
              end if
              if (1.le.ii.and.ii.le.nstrem) then
                strm(1,ii) = strflo
                print '(5i5,f15.0,1x,a,i5)',
     1            ilay,irow,icol,iseg,irch,strflo,'ii=',ii
              else
                print '(a/,a)',' Str2fm: stream reach not defined '//
     1            'for updated streamflow:',
     1             ' kper kstp  lay  row  col  seg  rch streamflow'
                print '(7i5,f15.7)',
     1              kkper,kkstp,ilay,irow,icol,iseg,irch,strflo
              end if
          end do
      end if
      return
      end
```

Subr. STR2FM: perform streamflow routing and calculate streambed leakage.

```
c
      SUBROUTINE STR2FM(NSTREM,STRM,ISTRM,strmad,HNEW,HCOF,RHS,IBOUND,
     1  idxstr,MXSTRM,NCOL,NROW,NLAY,mxseg,mxtrib,ITRBAR,ARTRIB,IDIVAR,
     2  ibgseg,iendsg,nrcseg,ntrseg,ckdtsg,tvtmsg,sigseg,ICALC,CONST,
     1  delt,sreduc,nreduc,irdcnd,kkiter,kkstp,kkper,iroute,
     1  istrbd,ievopt,iout,iostrm,ioreg)
C
C-----VERSION   2 06FEB1990 STR2FM
C-----CHANGED 4 LINES REGARDING CELLS OUTSIDE MODEL BOUNDARIES
C     ******************************************************************
C     ADD STREAM TERMS TO RHS AND HCOF IF FLOW OCCURS IN MODEL CELL
C     ******************************************************************
C
C     SPECIFICATIONS:
C     ------------------------------------------------------------------
C
      DOUBLE PRECISION HNEW
      DIMENSION STRM(30,MXSTRM),ISTRM(10,MXSTRM),strmad(10,mxstrm),
     1  HNEW(NCOL,NROW,NLAY),HCOF(NCOL,NROW,NLAY),RHS(NCOL,NROW,NLAY),
     2  IBOUND(NCOL,NROW,NLAY),Idxstr(NCOL,NROW,NLAY),
     3  ITRBAR(mxseg,mxtrib),ARTRIB(mxseg),IDIVAR(mxseg),
     4  ibgseg(mxseg),iendsg(mxseg),nrcseg(mxseg),ntrseg(mxseg),
     5  tvtmsg(mxseg),ckdtsg(mxseg),sigseg(mxseg)

      character optbal*1
      data optbal /'Y'/ !G: overall stream balance for each time step
      data iopstg /0/ !don't use option to include change in storage
      data itrace /0/ !internal diagnostic option to print summary of
c                       results after solution has been reached (istrbd=2).
C     ------------------------------------------------------------------
C
C1------IF NSTREM<=0 THERE ARE NO STREAMS. RETURN.
      IF(NSTREM.LE.0)RETURN

      if (istrbd.eq.0 .and. optbal.eq.'G') then
          write (iostrm,'(a)') ' kper kstp    delt     '//
     1       '    sumyld     ds/dt     suminp     sumout     sumtrb'//
     1       '    sumevt    sumlkg     qLinac      xLsum     xLinac'
cc        write (iostrm,'(2i5,g12.4,8f10.2,2f12.0)') kkper,kkstp,delt,
cc   1      sumyld,dsdt,suminp,sumout,sumtrb,sumevt,sumlkg,
cc   1      qLinac,xLsum,xLinac
c                      write mass balance profile:
      else if (istrbd.eq.2 .and. optbal.eq.'Y') then
          write (iostrm,'(a,2i5)') '  row  col  rch'//
     1       '     dS/dt    Qin:10    Qout:9  Qtrib:18    Qlkg:11'//
     1       'Prv Qin:24   Qout:23 Qavg(x,t) RchLen:12    depth'//
     1       '    hstr     haqf     hygrad Ibou Actv Iqflg',kkstp,kkper
      end if
c
c
      nreduc=0      !initialize no. stream reaches with reduced diversions
      sreduc=0.
      qLact=0.      !total streambed leakage from active grid cells
      xLact=0.      !total length of stream reaches through active grid cells
      qLinac=0.
      xLinac=0.
      if (istrbd.eq.2 .and. optbal.eq.'G') then
        suminp=0.   !total channel outflow from stream reaches
        sumout=0.   !total channel inflow to stream reaches
        sumtrb=0.   !total tributary inflow
        sumevt=0.  !total evaporation along stream channel
        sumlkg=0.  !total streambed leakage including inactive grid cells
        xLsum =0.  !total length of stream reaches including inactive cells
      end if
C2------PROCESS EACH CELL IN THE STREAM LIST.
      if (itrace.gt.0.and.
     1        (istrbd.eq.0.or.istrbd.eq.2)) then
        print '(/,3(1x,a,i3))',' Strmfm: istrbd=',istrbd,
     1    'kkper=',kkper,'kkstp=',kkstp,'kkiter=',kkiter
        print '(a)',' Rec  Lay  Row  Col  Seg  Rch'//
     1          ' Actv Coup Haquifr  Hstream   Depth  AdjTop'//
     1          ' Hstr-Hold StrbedLkg  TribFlow    Inflow'//
     1          '      Qavg   Outflow     Flolat      Qevt    Topwid'//
     1          ' gage'
      end if
C3------DETERMINE LAYER, ROW, COLUMN OF EACH REACH.
      qevtsm=0.
      DO L=1,NSTREM
        IL=ISTRM(1,L)
        IR=ISTRM(2,L)
        IC=ISTRM(3,L)
C
C4----06FEB1990, CHECK FOR CELLS OUTSIDE MOVED TO C12, C16 AND C18.
C
C5------DETERMINE STREAM SEGMENT AND REACH NUMBER.
        ISTSG=ISTRM(4,L)      !index to stream segment
        NREACH=ISTRM(5,L)     !index to stream reach within segment(1=top,
nrcseg(istsg)=bottom)
C
C6------SET FLOWIN EQUAL TO STREAM SEGMENT INFLOW IF FIRST REACH.
        IF(NREACH.eq.1) then
          FLOWIN=STRM(1,L)
c7------[replaced by c7a, below]--spp
C
C8------IF SEGMENT IS A DIVERSION, COMPUTE FLOW OUT OF UPSTREAM REACH.
          IF(IDIVAR(ISTSG).gt.0) then
            NDFLG=IDIVAR(ISTSG)       !index to upstream segment (source of diversion)
            DUM=ARTRIB(NDFLG)-FLOWIN
            IF(DUM.GE.0.0) then
              ARTRIB(NDFLG)=DUM       !update outflow from upstream segment
              lend = iendsg(ndflg)    !index to strm rec for the last reach of upstream seg
              strm(9,lend) = dum      !update outflow from upstream reach
            else
              FLOWIN=0.
            end if
          end if
cc----
C9-----SUM TRIBUTARY OUTFLOW AND USE AS INFLOW INTO DOWNSTREAM SEGMENT.C
c        (Note: is inflow < 0 redundant and unnecessary?)
          if (ntrseg(istsg).gt.0) then
            FLOWIN =0.
            DO ITRIB=1,ntrseg(istsg)
              INODE=ITRBAR(ISTSG,ITRIB)
              IF(INODE.gt.0) FLOWIN=FLOWIN+ARTRIB(INODE)
            end do
          end if
C
        if (istrbd.ne.1) trvseg = 0. !initialize kinematic wave travel time for segment
C10-----IF REACH >1, SET INFLOW EQUAL TO OUTFLOW FROM UPSTREAM REACH.
        else
          FLOWIN=STRM(9,L-1)
        end if
        STRM(10,L)=FLOWIN                   !reach inflow
c            surface lateral diversions can't exceed channel inflow:
        if (strm(18,L).lt.0.) then
          divnet = ABS(strm(18,L)) !net lat. surface outflow = |trib.inflow - diversions|
          if (divnet.gt.strm(10,L)) then
            Qreduc = divnet - strm(10,L)
            sreduc = sreduc + Qreduc
            nreduc = nreduc + 1
            strm(18,L) = -strm(10,L)
```

```
              if (istrbd.eq.2) then
                isourc=1
                write (ioreg,'(2i5,i4,4x,3i4,8x,2i4,70x,f10.5)')
     1              kkper,kkstp,nreduc,il,ir,ic,isourc,L,qreduc
              end if
c               gw version:
c             write (iout,'(2i5,9i4,5f10.3,3f10.3)') kkper,kkstp,
c    1            nreduc,L,il,ir,ic,izone,iusdwr,isourc,istrch,
c    1            well(4,L),well(5,L),well(9,L),well(13,L),Qmax,
c    1            satthk,qfract,qreduc
            else
              Qreduc = 0.
            end if
          end if
c         Calculate depth based on time- and space-centered streamflow, Qavg,
c         given by the average of inflow and outflow for both the present and
c         previous time steps.  This differs from the original version (Prudic),
c         which considers only the present time step.
c         Terms for this time step (i.e., at the end of this time step):
c           (a) strm(10,L): inflow, prev. solution iterate;
c           (b) strm(9,L):  outflow, prev. solution iterate;
c         Terms for previous time step (i.e., at the start of this time step:
c           (c) strm(23,L): outflow, previous time step;
c           (d) strm(24,L): inflow, previous time step.
c               Qin          Qout         Qin_prv      Qout_prv
          if (istrbd.gt.0) then
            if (istrbd.eq.1.and.kkiter.eq.1) then
              Qavg = (strm(24,L) + strm(23,L))/2.  !first iteration of solution
            else
              Qavg = (strm(10,L)+strm(9,L)+strm(24,L)+strm(23,L))/4.
            end if
          else
            Qavg = strm(10,L)  !set equal to inflow for beginning of first time step
          end if
          haqf = hnew(ic,ir,iL)   !aquifer head [L]

C11----COMPUTE STREAM STAGE IN REACH IF ICALC IS GREATER THAN 0.       C
          IF(ICALC.gt.0) then  !option: given flow rate, Qavg, calculate stream
c           depth, stage, and width of hydraulic connection with aquifer, baqf;
c           return the coordinates y(Q) and B(Q) for flood stage,
c           (qflood, yflood, bflood);
c          calculate coupled (aquifer head-dependent) and uncoupled components
c          of streambed conductance and streambed leakage.

            call STRDEP (mxstrm,strm,L,icalc,const,Qavg,delt,depth,
     1          topwid,perim,area,v,ck,diffus,yflood,bflood)

            strm(2,L) = depth + strm(5,L)     !stream stage (surface elevation)
          end if
          hstr = strm(2,L)        !stream stage [L]

          call STRLKG (ibound(ic,ir,iL),irdcnd,IQFLG,mxstrm,kkiter,L,
     1        istrm,strm,Qavg,depth,perim,hstr,haqf,hygrad,
     1        cstr,flobot,flosid,yflood,bflood)

c option (y=1,n=0) to include leakage from inactive cells in stream routing:
          data ioplkg /0/
          if (ibound(ic,ir,il).le.0) then
            qLinac = qLinac + flobot
            xLinac = xLinac + strm(12,L)
            if (ioplkg.eq.0) flobot = 0.  !opt: exclude inactive cell leakage from routing
          else
c             streambed leakage and stream length associated with active grid cells:
            qLact = qLact + flobot
            xLact = xLact + strm(12,L)
          end if
C                                                                     C
c             apply routing option (1: Prudic; 2: diffusive wave); routine
c             returns outflow (flowot) and flag (istrm(8,L)) indicating whcih
c             routing option was applied. (Prudic is applied if conditions on
c             Peclet and Courant numbers are not satisfied, or for a call based
c           on initial or steady state conditions.
c
c           Evaporation loss from reach: include if ievopt > or = 2 (see Modswbwc):
        if (ievopt.ge.2) then
          Qevt = -strm(21,L)*topwid*strm(12,L)
        else
          qevt = 0.
        end if
        if (kkiter.le.1) then
          strm(22,L) = Qevt
        else if (kkiter.le.3) then
          strm(22,L) = (strm(22,L) + Qevt)/2. !avg with previous iteration
        end if
        qevtsm=qevtsm+strm(22,L)
c         Lateral inflow = tributary inflow - streambed leakage - evaporation:
        flolat = strm(18,L) - flobot - flosid + strm(22,L)
        if (iopstg.gt.0) then
          chgstg = strm(29,L)   !change in storage
        else
          chgstg = 0.
        end if
        call MODROUTE (istrbd,iroute,strm(12,L),delt,ck,diffus,
     1         peclet,crnt,theta,istrm(8,L),flolat,chgstg,
     1         strm(24,L),strm(23,L),flowin,flowot)

        strm(9,L) = flowot               !reach outflow
        strm(11,L) = flobot+flosid       !streambed leakage
c7a----     update outflow from this segment to that from this reach:
        artrib(istsg) = flowot
C                                                                      C
C18----RETURN TO STEP 3 IF STREAM INFLOW IS LESS THAN OR EQUAL TO ZERO C
C       AND LEAKAGE IS GREATER THAN OR EQUAL TO ZERO OR IF CELL        C
C       IS NOT ACTIVE--IBOUND IS LESS THAN OR EQUAL TO ZERO.           C

        if (istrbd.eq.1) then  !coupled stream-aquifer solution in progress
          IF (FLOWIN+strm(18,L).gt.0.0 .or. strm(11,L).lt.0.0) then
            if (IQFLG.eq.0) then
C19------------ IF HEAD > BOTTOM THEN ADD TERMS TO RHS AND HCOF.
c                 add conductance*stage to rhs, conductance to hcof:
              RHS(IC,IR,IL)=RHS(IC,IR,IL)-CSTR*HSTR
              HCOF(IC,IR,IL)=HCOF(IC,IR,IL)-CSTR
            else
C20----------- IF HEAD < BOTTOM THEN ADD streambed leakage ONLY TO RHS.
C
              RHS(IC,IR,IL)=RHS(IC,IR,IL)-FLOBOT
            end if
c             decoupled component of leakage includes two possible components:
c a)          head-independent component, flobot = cstr*(hstr-haqf), in main channel;
c b)          infiltration from side channels, flosid; works like recharge
c             (compare recharge package):
            if (1.+flosid.ne.1.) RHS(IC,IR,IL)=RHS(IC,IR,IL)-flosid
          end if
        else
c               Qin           Qout          Qin_prv       Qout_prv
ccc       Qavg = (strm(10,L) + strm(9,L) + strm(24,L) + strm(23,L))/4.
c             retain solution for next time step:
          strm(23,L) = strm(9,L)    !reach outflow
          strm(24,L) = strm(10,L)   !reach inflow

c------        indicate active reaches:
          if (flowot.gt.0.or.flowin.gt.0.or.ABS(flobot).gt.0.) then
            istrm(7,L)=1
          else
            istrm(7,L)=0
          end if
          strm(17,L) = depth   !stream depth
          strm(20,L) = hygrad !hydraulic gradient across streambed
          if (1.+ck.gt.1.) then
            trvtim = strm(12,L)/ck    !flood wave time of travel through reach
            trvseg = trvseg + trvtim !accumulate flood wave travel time through segment
          else
```

```
            trvtim=0.
          end if
c           additional streamflow characteristics (passed to PLT1OUT):
          strmad(1,L) = topwid        !B, top width
          strmad(2,L) = perim         !P, wetted perimeter
          strmad(4,L) = v             !v = Q/A, flow velocity
          strmad(5,L) = trvtim        !tr = dx/ck, flood wave travel time
          strmad(6,L) = ck            !ck = dQ/dA, kinematic wave speed
          strmad(7,L) = diffus        !Dh = Q/(2*B*S0), hydraulic diffusion
          if (istrbd.eq.2.and.nreach.eq.nrcseg(istsg)) then
            Ltop = ibgseg(istsg)      !index to top reach of segment
            cktop = strmad(6,Ltop)
            diftop = strmad(7,Ltop)
c             avg top & bottom reach values for kinematic wave speed and diffusion coef.:
            ckdtsg(istsg) = 0.5*(cktop + ck)*delt                !mu = ckavg*delt
            sigseg(istsg) = 2.*SQRT(0.5*(diftop + diffus)*delt)!sigma=2.*SQRT(Davg*delt)
          end if
          if (istrbd.eq.2.and.istrm(7,L).eq.1) then  !mass balance for active reaches
            if (optbal.eq.'G') then
              suminp = suminp + strm(10,L) !inflow
              sumout = sumout +  strm(9,L) !outflow
              sumtrb = sumtrb + strm(18,L) !tributary inflow
              sumlkg = sumlkg + strm(11,L) !streambed leakage
              sumevt = sumevt + strm(22,L) !evaporation from stream surface
            else if (optbal.eq.'Y') then
              dsdt = strm(10,L) - strm(9,L) + strm(18,L) - strm(11,L)
c             write (iostrm,790) ir,ic,L, dsdt,strm(10,L),strm(9,L),
c    1          strm(18,L),strm(11,L),strm(24,L),strm(23,L),
c    1          Qavg,strm(12,L),depth,hstr,haqf,hygrad,
c    1          ibound(ic,ir,iL),istrm(7,L),istrm(9,L)
790           format (3i5,9f10.2,4f8.2,3i5)
            end if  !optbal
          end if
ccc       else if (istrbd.eq.0) then
c           print '(a)',' Rec  Lay  Row  Col  Seg  Rch'//
c    1          ' Actv Coup Haquifr  Hstream   Depth  AdjTop'//
c    1          ' Hstr-Hold StrbedLkg  TribFlow    Inflow  Qavg'//
c    1          '      Qavg   Outflow    Flolat    Topwid gage'

          if (itrace.gt.0.and.
     1        (istrbd.eq.0.or.istrbd.eq.2)) then
            idgage=strm(13,L)
            print '(8i5,4f8.2,4f10.3,5f10.3,i5)',
     1        L,iL,ir,ic,istsg,nreach,istrm(7,L),istrm(9,L),
     1        haqf,hstr,depth,strm(25,L),
     1        hygrad,strm(11,L),strm(18,L),strm(10,L),
     1        Qavg,strm(9,L),flolat,Qevt,topwid,idgage
          end if  !istrbd,istrm(7,L)
        end if  !istrbd
      end do  !L=1,nstrem

      if (istrbd.eq.2) then
        print '(1x,a,f10.2)','Str2fm: sum(Qevt) = ',qevtsm
        if (optbal.eq.'G') then
          dsdt = suminp - sumout + sumtrb - sumlkg - sumevt
          sumyld = sumout - suminp
          write (iostrm,'(2i5,g12.4,8f10.2,2f12.0)') kkper,kkstp,delt,
     1      sumyld,dsdt,suminp,sumout,sumtrb,sumevt,sumlkg,
     1      qLinac,xLsum,xLinac
        end if
      end if
C                                                                      C
C22-----RETURN.                                                        C
      RETURN
      END
```

Subr. STR2BDZ: summarize budget for streambed leakage, including zone option.

```
      SUBROUTINE STR2BDZ(NSTREM,STRM,ISTRM,IBOUND,MXSTRM,NCOL,NROW,NLAY,
     1  DELT,VBVL,VBNM,MSUM,KSTP,KPER,ISTCB1,ISTCB2,ICBCFL,BUFF,IOUT,
     1  ICALC,IPTFLG,iopzon,mxzone,izonbd,vbvlzn,ratzin,ratzou)

C-----VERSION  2  06FEB1990 STR2BDZ                                     C
C-----CHANGED 4 LINES REGARDING CELLS OUTSIDE MODEL BOUNDARIES          C
C     ******************************************************************C
C     CALCULATE VOLUMETRIC BUDGET FOR STREAMS                           C
C     ******************************************************************C
C                                                                       C
C     SPECIFICATIONS:                                                   C
C     ------------------------------------------------------------------C
      CHARACTER*4 VBNM,TEXT,STRTXT
      DIMENSION STRM(30,MXSTRM), ISTRM(10,MXSTRM),
     1    IBOUND(NCOL,NROW,NLAY),VBVL(4,20),       VBNM(4,20),
     1    BUFF(NCOL,NROW,NLAY),  izonbd(ncol,nrow,nlay),
     1    vbvlzn(4,20,0:mxzone), ratzin(0:mxzone), ratzou(0:mxzone)
      DIMENSION TEXT(4),STRTXT(4)
      DATA TEXT(1),TEXT(2),TEXT(3),TEXT(4) /'  ST','REAM',' LEA','KAGE'/
      DATA STRTXT(1),STRTXT(2),STRTXT(3),STRTXT(4) /'STRE','AM F',
     1                                             'LOW ','OUT '/
C     ------------------------------------------------------------------C
C                                                                       C
C1------SET IBD=1 IF BUDGET TERMS SHOULD BE SAVED ON DISK.              C
      IBD=0
      RATIN = 0.
      RATOUT = 0.
      if (iopzon.gt.0) call ZONINI (mxzone,ratzin,ratzou)
C                                                                       C
C2------IF NO REACHES, KEEP ZEROS IN ACCUMULATORS.                      C
      IF(NSTREM.EQ.0) GO TO 600
C                                                                       C
C3A-----TEST TO SEE IF CELL-BY-CELL TERMS ARE NEEDED.                   C
      if ((ICBCFL.ne.0).and.(ISTCB1.gt.0)) then
C3B-----CELL-BY-CELL TERMS ARE NEEDED, SET IBD AND CLEAR BUFFER.        C
        IBD = 1
        DO IL=1,NLAY
          DO IR=1,NROW
            DO IC=1,NCOL
              BUFF(IC,IR,IL)=0.
            end do
          end do
        end do
      end if
C                                                                       C
C4------IF THERE ARE STREAMS THEN ACCUMULATE LEAKAGE TO OR FROM THEM.   C
      DO L=1,NSTREM
C                                                                       C
C5---DETERMINE REACH LOCATION.                                          C
        IL=ISTRM(1,L)
        IR=ISTRM(2,L)
        IC=ISTRM(3,L)
*-------------------------code from str1bd starts here:
        if (ibound(ic,ir,iL).gt.0) then
          flobot = strm(11,L)
C20----IF LEAKAGE FROM STREAMS IS TO BE SAVED THEN ADD RATE TO BUFFER.  C
          IF(IBD.EQ.1) BUFF(IC,IR,IL)=BUFF(IC,IR,IL)+FLOBOT
C                                                                       C
C21----DETERMINE IF FLOW IS INTO OR OUT OF MODEL CELL.                  C
C       SKIP ESTIMATE OF LEAKAGE FROM STREAM IF LEAKAGE IS ZERO.        C

          if (flobot.lt.0.) then
C22-----SUBTRACT FLOW RATE FROM RATOUT IF AQUIFER DISCHARGES TO STREAM.C
            RATOUT=RATOUT-FLOBOT
          else if (flobot.gt.0.) then
C23-----ADD FLOW RATE TO RATIN IF STREAM DISCHARGES TO AQUIFER.         C
            RATIN=RATIN+FLOBOT
          end if
c                   zone budget option (MODPOST package):
          if (iopzon.gt.0) call ZONACC (flobot,
     1      izonbd(ic,ir,il),mxzone,ratzin,ratzou)
```

```
         end if
       end do
C                                                                        C
C24-----IF BUDGET TERMS WILL BE SAVED THEN WRITE TO DISK.                C
       IF(IBD.EQ.1) CALL UBUDSV(KSTP,KPER,TEXT,ISTCB1,BUFF,NCOL,NROW,
      1                          NLAY,IOUT)
C                                                                        C
C25A-----MOVE RATES INTO VBVL FOR PRINTING BY MODULE BAS_OT.             C
   600 VBVL(3,MSUM)=RATIN
       VBVL(4,MSUM)=RATOUT
C                                                                        C
C25B-----MOVE PRODUCT OF RATE AND TIME STEP INTO VBVL ACCUMULATORS.      C
       VBVL(1,MSUM)=VBVL(1,MSUM)+RATIN*DELT
       VBVL(2,MSUM)=VBVL(2,MSUM)+RATOUT*DELT
       if (iopzon.gt.0)
      1     call ZONSUM(msum,mxzone,delt,ratzin,ratzou,vbvlzn)
C                                                                        C
C25C-----MOVE BUDGET TERM LABELS INTO VBNM FOR PRINTING BY BAS_OT.       C
       VBNM(1,MSUM)=TEXT(1)
       VBNM(2,MSUM)=TEXT(2)
       VBNM(3,MSUM)=TEXT(3)
       VBNM(4,MSUM)=TEXT(4)
C                                                                        C
C26-----INCREASE BUDGET TERM COUNTER BY ONE.                             C
       MSUM=MSUM+1
C                                                                        C
C27-----RESET IBD COUNTER TO ZERO.                                       C
       IBD=0
C28----IF STREAM OUTFLOW FROM EACH REACH IS TO BE STORED ON DISK         C
C      THEN STORE OUTFLOW RATES TO BUFFER.                               C
       IF((ICBCFL.EQ.0).OR.(ISTCB2.LE.0)) GO TO 625
       IBD = 1
       DO 605 IL=1,NLAY
       DO 605 IR=1,NROW
       DO 605 IC=1,NCOL
   605 BUFF(IC,IR,IL)=0.
C                                                                        C
C29-----SAVE STREAMFLOWS OUT OF EACH REACH ON DISK.                      C
       DO 615 L=1,NSTREM
       IC=ISTRM(3,L)
       IR=ISTRM(2,L)
       IL=ISTRM(1,L)
       IF(IBOUND(IC,IR,IL).LE.0) GO TO 615
       BUFF(IC,IR,IL)=BUFF(IC,IR,IL)+STRM(9,L)
   615 CONTINUE
       CALL UBUDSV(KSTP,KPER,STRTXT,ISTCB2,BUFF,NCOL,NROW,NLAY,IOUT)
C                                                                        C
C30-----PRINT STREAMFLOW RATES AND LEAKAGE FOR EACH REACH.               C
   625 IF((ISTCB1.GE.0).OR.(ICBCFL.LE.0)) GO TO 800
       IF(IPTFLG.GT.0) GO TO 800
       IF(ICALC.GT.0) GO TO 700
       WRITE(IOUT,650)
   650 FORMAT(1H0,12X,'LAYER',6X,'ROW',5X,'COLUMN',5X,'STREAM',4X,
      1'REACH',6X,'FLOW INTO',4X,'FLOW INTO',6X,'FLOW OUT OF'/43X,
      2      'NUMBER',3X,'NUMBER',4X,'STREAM REACH',4X,'AQUIFER',
      3      6X,'STREAM REACH'//)
       DO 690 L=1,NSTREM
       IL=ISTRM(1,L)
       IR=ISTRM(2,L)
       IC=ISTRM(3,L)
       WRITE(IOUT,675)IL,IR,IC,ISTRM(4,L),ISTRM(5,L),
      1     STRM(10,L),STRM(11,L),STRM(9,L)
   675 FORMAT(1X,5X,5I10,8X,G9.3,5X,G9.3,7X,G9.3)
   690 CONTINUE
       GO TO 800
 700   WRITE(IOUT,710)
   710 FORMAT(1H0,12X,'LAYER',6X,'ROW',5X,'COLUMN',5X,'STREAM',4X,
      1'REACH',6X,'FLOW INTO',4X,'FLOW INTO',6X,'FLOW OUT OF',5X,
      1'HEAD IN',1X,'TRIBUTARY',1X,'IQFLG'/,
      2 3X,      'NUMBER',3X,'NUMBER',4X,'STREAM REACH',4X,'AQUIFER',
      2 6X,'STREAM REACH',5X,'STREAM',1X,'NET INFLOW'//)
      DO 750 L=1,NSTREM
        IL=ISTRM(1,L)
        IR=ISTRM(2,L)
        IC=ISTRM(3,L)
        if (ISTRM(7,L).EQ.1)    !print results for active reaches
     1    WRITE(IOUT,775) IL,IR,IC,ISTRM(4,L),ISTRM(5,L),STRM(10,L),
     1    STRM(11,L),STRM(9,L),STRM(2,L),STRM(18,L),ISTRM(9,L)
  775     FORMAT (1X,5I10,8X,G9.3,5X,G9.3,7X,G9.3,4X,F9.2,5X,G9.3,I5)
  750 CONTINUE
  800 CONTINUE
C                                                                       C
C31-----RETURN.                                                         C
      RETURN
      END
```

Subr. STRDEP: given streamflow, calculate stream depth

```
c-------------------------------------------------------------------------------
      subroutine STRDEP (mxstrm,strm,L,icalc,const,Q,dt,depth,topwid,
     1  perim,area,v,ck,diffus,yflood,bflood)
c
      dimension strm(30,mxstrm)

c L: index to reach
c         arguments passed to subr YQSTRM and YQ to calculate normal depth
c         given flow rate for a rectangular or trapezoidal cross section, resp.:
c (qflood, yflood, bflood) coordinates of Q(y) and B(y) at flood stage;
c     to date, implemented only for YQWALNUT; remains otherwise undefined.
c iperim: option to represent wetted perimeter by top width (iperim=0)
c           to approximate hydraulic radius in Manning's equation;
c           otherwise (iperim <>0) bottom and side walls will be included.
      data itrdp1/1/
      slope   = ABS(strm(7,L))           !streambed slope (abs. value)
      if (icalc.eq.1) then
c               solve for depth of a rectangular section with steady flow
c               (Manning's eqn as solved in Prudic's STREAM (1989))
        data iprect/1/ !rectangular: perim = b+2y (Prudic convention)
        base    = strm(6,L)       !stream base width [L] (trapezoid section)
        rmning = strm(8,L)        !Manning roughness number
        call YQSTRM (itrdp1,iprect,base,slope,rmning,const,
     1            Q,depth,topwid,perim,area,v,ck)
      else if (icalc.eq.2) then
c               solve for depth of a trapezoidal section with unsteady flow
c         in YQ, use the steady state derivative to evaluate celerity
c         ck and friction slope sf, to take a Newton step, and
c         to compute a convergence tolerance epsq on flow rate,
c         since epsq = epsh*(dq/dy) = epsh*(B*ck) for top width B.
        data iptrap/0/ !trapezoidal: approximate wetted perimeter with top width
        data epsh/0.005/   !depth error tolerance on solution
        base    = strm(6,L)       !stream base width [L] (trapezoid section)
        sidslp = strm(13,L)       !trapezoidal section side slope (1/c)
        rmning = strm(8,L)        !Manning roughness number
        if (dt.gt.0) then         !unsteady flow in a trapezoidal channel
          dqdt = (strm(10,L) - strm(24,L))/dt    !time rate of change of inflow from prev.
time step
        else
          dqdt = 0.
        end if
        call YQ (itrdp1,iptrap,base,slope,sidslp,rmning,const,epsh,
     1        Q,dqdt,epsq,errq,y0inf,depth,sf,topwid,perim,area,v,ck)
      else   !if icalc = 3: idgage, passed in the side slope input field,
*                                   otherwise used for a trapezoidal section by
*                                   YQ (for icalc=2), identifies the gaging station
*                                   whose characteristics are to be used.
ccc            idgage: 1=Concordia, 2=Clay Center, Repub. R. (subr YQGAGE);
c                                     3=Albert KS on Walnut Creek (subr YQWALNUT)
        idgage = strm(13,L)
        if (idgage.le.2) then
```

```
          call YQREPUB (itrdp1,idgage,Q,depth,topwid,perim,area,v,ck,
     1                  bmain,qflood,yflood,bflood)
        else
          call YQWALNUT (itrdp1,idgage,Q,depth,topwid,perim,area,v,ck,
     1                  bmain,qflood,yflood,bflood)
          strm(19,L) = bmain !main channel width (use with low Ks in StrLkg)
        end if
      end if
c     diffus  =Dh = Q/(2*B*S0), hydraulic diffusion coefficient:
      if (topwid.gt.0.and.slope.gt.0.) diffus = Q/(2.*topwid*slope)
      return
      end
```

Subr. STRLKG: given hydraulic gradient across streambed, calculate leakage.

```
c-----------------------------------------------------------------------
      subroutine STRLKG (ibound,irdcnd,IQFLG,mxstrm,kkiter,L,
     1    istrm,strm,Q,depth,perim,hstr,haqf,hygrad,
     1    cstr,flobot,flosid, yflood, bflood)

      dimension istrm(10,mxstrm), STRM(30,MXSTRM)
      data itrace /1/
c   Input:
c ibound = value of activity array element ibound(ic,ir,iL) (1=active node)
c irdcnd:
c     If irdcnd > 0, then Ks is applied as read from field 51-60 of the second
c     record for each reach, and conductance C is calculated as follows:
c     Given
c             Ks = streambed saturated hydraulic conductivity for
c                  flow widths < bmain;
c             Ks_high = streambed hydraulic conductivity for channel outside
c                  the low-flow width bmain;
c             Ls = stream reach length;
c             P = wetted perimeter;
c             T = strm(5,*)-strm(4,*) = streambed thickness.
c         if (P < bmain) then
c             Keq = Ks
c         else
c             wdfrac = bmain/perim
c             Keq = wdfrac*Ks + (1-wdfrac)*Ks_high
c         end if
c         C = Keq*Ls*P/T

c         ref. Eqn. 2.32, Freeze and Cherry (1979).
c-----------------------------------------------------------------------
c IQFLG: flag as follows:
c          0: indicates that stream and aquifer are hydraulically coupled,
c             i.e., aquifer head haqf > streambed bottom elevation;
c          1: aquifer head is below streambed bottom elevation.
c mxstrm   max. no. stream reaches for dimensioning STRM & ISTRM
c L        index to stream reach
c istrm(10,mxstrm)
c strm(10,mxstrm)
c perim:  wetted perimeter [L]
c bmain: channel base width, strm(19,*), within which low Ks is used;
c         outside, high Ks is applied
c hstr = strm(2,L) = stream stage [L]
c haqf = hnew(ic,ir,iL) = aquifer head [L]
c   Output:
c hygrad: hydraulic gradient across streambed [L]
c cstr: streambed conductance [L^2/T]
c flobot: streambed leakage [L^3/T]
c flosid: infiltration due to flow out of banks
c
c             Streambed leakage: calculate for active nodes.
      flobot=0.
      flosid=0.
      hygrad=0.
      if (ibound.le.0) RETURN  !ignore leakage if aquifer head is unknown

C14----LEAKAGE IF HEAD IN CELL IS above STREAMBED BOTTOM:
      if (kkiter.le.2) then
        if (haqf.gt.strm(4,L) .and. ibound.gt.0) then
          IQFLG=0                     !coupled stream-aquifer solution
          hygrad = hstr - haqf        !stage - aquifer head
        else
c               hydraulic gradient limited by streambed bottom elevation:
          IQFLG=1                     !uncoupled stream-aquifer solution
          hygrad = hstr - strm(4,L)   !stage - streambed bottom elev.
        end if
c           note: iqflg is retained below as istrm(9,L).
      else
        iqflg = istrm(9,L)
        if (iqflg.eq.0) then          !coupled stream-aquifer solution
          hygrad = hstr - haqf        !stage - aquifer head
        else
c               hydraulic gradient limited by streambed bottom elevation:
          hygrad = hstr - strm(4,L)   !stage - streambed bottom elev.
        end if
      end if
      bedthk = strm(5,L) - strm(4,L)  !streambed thickness T
      if (irdcnd.gt.0) then
c               bmain: value for main channel width corresp. to
c               low hydraulic conductivity of silted streambed.
        wdks = strm(19,L)   !channel width with high Ks (STR2RP)
        wdmain = AMIN1 (perim,bflood) !limit coupled leakage to width at flood stage
        if (wdmain.le.wdks) then
c           conductance for main channel flow: C=Ks_low*perim*dx/T
          rkeqv = strm(14,L)           !use low Ks within main channel
        else
c                equiv. conductivity through parallel layers:
          rkeqv = (wdks*strm(14,L) + (wdmain-wdks)*strm(15,L))/wdmain
        end if

        strlen = strm(12,L)          !stream reach length
c                 Streambed conductance:
        cstr = rkeqv*perim*strlen/bedthk
c
        if (perim.gt.bflood) then
            davg = (depth - yflood)/2.   !avg depth above flood stage
            flosid = (strlen/bedthk)*strm(15,L)*(perim-bflood)*davg
        end if
      end if
      flobot = cstr*hygrad
      strm(3,L) = cstr
C15----LEAKAGE OUT OF REACH IS LIMITED BY STREAM INFLOW.
c              leakage can't exceed channel + lateral surface inflows:
      flomax = strm(10,L)+strm(18,L)
      if (flobot+flosid.gt.flomax) then
        excess = flobot+flosid - flomax
        if (flobot.le.flomax) then
          flosid = flosid - excess
        else
          flosid = 0.
          flobot = flomax
          IQFLG=1
        end if
      end if
      istrm(9,L) = iqflg
      return
      end
```

Subr. MODROUTE: given inflow and lateral flow, evaluate outflow.

```
c-----------------------------------------------------------------------------
      subroutine MODROUTE (istrbd,iroute,dx,dt,ck,diffus,peclet,crnt,
     1    theta,iflgdw,flolat,chgstg,Qinprv,Qoutpr,flowin,flowot)
c usage:
cc      call MODROUTE (istrbd,iroute,strm(12,L),delt,ck,diffus,
cc   1        peclet,crnt,theta,istrm(8,L),flolat,chgstg,
cc   1        strm(24,L),strm(23,L),flowin,flowot)
c flolat = lateral inflow: tributary inflow - surf. water div. + baseflow
c         quasi-linear options:
c               optdif='V': let diffusion coefficient, Dh=Q/(2*B*S0), vary;
c               lineqn=0:   let wave speed, ck=dQ/dA, vary.
          data lineqn/0/, optdif/'V'/
c                calculate Muskingum-Cunge numbers:
c            Pe = ck*dx/Dh indicates relative significance of diffusion and
c            advection; advection dominated regime: Pe > 2.
c            Dh = Q/(2*B*S0),          hydraulic diffusion coefficient
c            where:
c              Q = discharge [L^3/T]
c              v = Q/A
c              B = top width (width of reach at water surface)
c              S0 = streambed slope.
c             Three conditions must be met here in order to use Muskingum-Cunge
c             routing (iroute=2) instead of this version of Prudic routing
c             (iroute=1):
c             (a) this isn't an initializing call to STR1DW, i.e. istrbd > 0;
c             (b) Pe > 2, i.e. advection dominates diffusion; and
c             (c) Courant should be within the limits [1-2/Pe, 1+2/Pe].
          iflgdw=0 !flag to use Prudic-type mass balance for routing.
          if (istrbd.gt.0.and.iroute.eq.2) then
            if (1.+diffus.gt.1.) then
              peclet = ck*dx/diffus
            else
              peclet = 1.e8
            end if
c             Courant no., ratio of solution characteristic (ck, wave speed) to
c             grid characteristic (dx/dt), where dx=reach length, dt=time step.
            crnt    = ck*dt/dx
            if (peclet.ge.2 .and. 1.-2./peclet.le.crnt .and.
     1          crnt.le.1.+2./peclet) iflgdw = 1
          end if
          if (iflgdw.eq.1) then
c                 criteria were met for Muskingum-Cunge routing, so use that
c                 method; otherwise (iroute=1), use Prudic-type mass balance.
c                 Limit theta to advection-dominated region (Peclet>2):
c                 theta = 1/2 - 1/Pe, a numerical diffusion matching parameter;
c                 numerical diffusion coef. Dn = ck*dx*(1/2 - theta).
              theta = AMAX1 (0., 0.5 - 1./peclet)
              denom = 2. + crnt - 2.*theta
              cunge1 = (crnt + 2.*theta)/denom
              cunge2 = (crnt - 2.*theta)/denom
              cunge3 = (-crnt + 2.*(1.-theta))/denom
c               streambed leakage flobot is time-centered, based on avg of
c               present and previous aquifer head solutions.
c                 Muskingum-Cunge routing:
              flowot = AMAX1(0., cunge1*Qinprv + cunge2*flowin
     1          + cunge3*Qoutpr + (cunge1+cunge2)*flolat)
          else
c               mass balance following that used by Prudic:
              flowot = AMAX1(0., flowin + flolat)
c               mass balance following that used by Prudic but with storage:
cccccccccc    flowot = AMAX1(0., flowin + flolat - chgstg)
          end if
      return
      end
```

<u>Subr. YQSTRM: evaluate normal depth for steady streamflow in rectangular channel.</u>

```
c-----------------------------------------------------------------------
c file yq2.for  spp  compute normal depth of a trapezoidal channel.
```

```
      subroutine YQSTRM (itrace,iperim,base,slope,rmning,const,
     1           flow,depth,topwid,perim,area,v,ck)
c usage:
cc            call YQSTRM (itrdpl,iprect,base,slope,rmning,const,
cc   1           Q,depth,topwid,perim,area,v,ck)

      xnum = flow*rmning            ! qavg*n
      dnom = const*base*SQRT(slope) ! C*W*sqrt(S0)
      depth= AMAX1(0.,(XNUM/DNOM)**0.6)       ![(n/C)*qavg/(W*sqrt(S0))]^(3/5)
      topwid = base                 ! B = base
      if (iperim.eq.0) then
        perim = base
      else
        perim = base + 2.*depth       ! P = base + 2*y
      end if
      area = base*depth             ! A = B*y
      if (1.+area.gt.1.) then
          v = flow/area             ! v = Q/A
      else
          v = 0.
      end if
      ck = 1.67*v                   ! approx. kinematic wave speed
      return
      end
```

<u>Subr. YQ: evaluate normal depth for unsteady streamflow in trapezoidal channel.</u>

```
c-----------------------------------------------------------------------------
        subroutine YQ (itrace,iperim,base,slope,cotphi,
     1                 rmning,cm,epsh,q,dqdt,epsq,errq,y0,y,sf,
     1                 topwid,perim,area,v,ck)
c
c usage:
cc      call YQ (itrdpl,iptrap,base,slope,sidslp,
cc   1      rmning,const,epsh,Q,dqdt,epsq,errq,y0inf,
cc   1      depth,sf,topwid,perim,area,v,ck)

c       Given flow rate Q in a trapezoidal channel, solve for the depth y.
c       This routine applies Newton's method to accelerate a fixed
c       point iterative procedure that converges strongly by itself without
c       the aid of the Newton step.
        data numax/10/     !max no. iterations

* iperim: option to represent wetted perimeter by top width (iperim=0)
*           to approximate hydraulic radius in Manning's equation;
*           otherwise (iperim <>0) bottom and side walls will be included.
        cscphi = SQRT(1.+cotphi*cotphi)
        sf = slope
*           initialize fixed pt iterative step
        y1=0
*           initialize Newton step
        yguess=0.
        erry=1.e6
        nu=0
        do while (nu.lt.numax.and.ABS(erry).gt.epsh)
            nu=nu+1
*               given guess at depth yguess, solve y(Q):
            call YQS (iperim,rmning,cm,sf,q,base,cotphi,cscphi,
     1                topwid,perim,area,yguess,y1)

*               for y1(Q)=guess at depth, solve y0(Q) for dQ/dt=0,
*               i.e. sf=slope:
            call YQS (iperim,rmning,cm,slope,q,base,cotphi,cscphi,
     1                topwid,perim0,area0,y1,y0)
*               Evaluate y2=y(Q+dq) and y3=y(Q-dq) in order to
*               approximate by central differences the derivative dQ/dy
*               about Q under steady state conditions (dQ/dt=0).
*               Steady state conditions are represented for this purpose
*              by passing the bed slope to subr QYTRAP to compute y2 and y3.
*              Use the steady state derivative to evaluate celerity
*              ck and friction slope sf, to take a Newton step, and
*              to compute a convergence tolerance epsq on flow rate,
*              since epsq = epsh*(dq/dy) = epsh*(B*ck) for top width B.
          dltq=.01*Q
          q2=Q + dltq
*              for y1(Q)=guess at depth, solve y2(Q+dQ) for dQ/dt=0,
*              i.e. sf=slope:
          call YQS (iperim,rmning,cm,slope,q2,base,cotphi,cscphi,
     1          topwd2,perim2,area2,y1,y2)
          q3=Q - dltq
*              for y1(Q)=guess at depth, solve y3(Q-dQ) for dQ/dt=0, as above:
          call YQS (iperim,rmning,cm,slope,q3,base,cotphi,cscphi,
     1          topwd3,perim3,area3,y1,y3)
*
*              central difference approximation to derivative dQ/dy
*              about y=y1:
          dq = q2-q3
          dqdy =dq/(y2-y3)
*              channel width at water surface:
          topwid = base + 2.*cotphi*y1
*              celerity (wave speed):
          ck=dqdy/topwid
*              compute a convergence tolerance on flow rate epsq
*              based on epsh to test for convergence on streambed leakage.
*              Since epsq = epsh*(dq/dy) and ck = (1/B)(dq/dy),
*              we can write epsq = epsh*(B*ck), where B=channel surface
*              width and ck = kinematic wave speed.
          epsq = epsh*(topwid*ck)
*              compute friction slope for unsteady flow:
          if (dqdt.ne.0.) then
*                  estimate dy/dt from dq/dt and dq/dy:
              dydt = dqdt/dqdy
              sf= AMAX1 (slope/2., slope + dydt/ck)
          else
              dydt=0.
              sf = slope
          end if
*              compute flow rate q1(y1) based on guess y1 and improved
*              estimate of friction slope sf:
          call QYTRAP (iperim,rmning,cm,sf,base,cotphi,cscphi,
     1             y1,topwid,perim,area,radius,v,q1)

*              take a Newton step:
          y1=y1-(q1-q)/dqdy
          erry=y1-yguess
*              compute flow rate q1(y1) based on guess y1 and improved
*              estimate of friction slope sf:
          call QYTRAP (iperim,rmning,cm,sf,base,cotphi,cscphi,
     1             y1,topwid,perim,area,radius,v,q1)
          errq=q1-q
          if (itrace.ge.2) then
              if (nu.eq.1)
     1            print '(1x,a)', '    q         y1     dq/dt   '//
     1              'wave spd frc slope chg in y  top width'//
     1              ' perimeter       area    v(m/s)'
              print 120, q,y1,dqdt,ck,sf,erry,topwid,perim,area,v
120           format (f10.2,f10.4,f10.5,f8.2,2f10.6,4f10.4)
          end if
          yguess=y1
      end do
*         update flow-dependent parameters:
      y=y1
      topwid = base + 2.*cotphi*y          ! surface width
      perim=base+2.*cscphi*y               ! wetted perimeter
      area=y*(base+cotphi*y)               ! cross sectional area
      if (iperim.eq.0) then
          radius = area/topwid
      else
          radius=area/perim
      end if
      v=Q/area
      ck=dqdy/topwid                      ! celerity (kinematic wave speed)
      return
      end
```

Subr. YQS

```
* ==========================================================================
*
      subroutine YQS (iperim,rmning,cm,slope,q,base,cotphi,cscphi,
     1                topwid,perim,area,yguess,y)
*
      if (base.le.0.) then
*             triangle: explicit for steady state
          f=((rmning/cm)*q/SQRT(slope))**(3./8.)
          y = f*(2.*cscphi)**.25/cotphi**(5./8.)
          perim = 2.*cscphi*y
          area = cotphi*y*y
*
      else
*             rectangles and trapezoids
          f=(rmning*q/(cm*SQRT(slope)))**.6
          topwid = base + 2.*cotphi*yguess
          perim = base + 2.*cscphi*yguess
c             if iperim=0, R=A/B, else R=A/P:
          if (iperim.eq.0) then
              area = f*topwid**.4
          else
              area = f*perim**.4
          end if
c             solve y(b+my) = area (i.e. my*y + base*y - area = 0)
c             for the positive root y with an alternative form of the
c             quadratic formula (it won't blow up for m=0):
          if (cotphi.eq.0) then
c                 rectangle:
              y=area/base
          else
c                 trapezoid (reduces to area/base for rectangles):
              y = (2.*area)/(base+SQRT(base*base + 4.*cotphi*area))
          end if
      end if
      return
      end
```

Subr. QYTRAP

```
c ==========================================================================
      subroutine QYTRAP (iperim,rmning,cm,slope,base,cotphi,cscphi,
     1                 y,topwid,perim,area,radius,v,q)

      topwid = base + 2.*cotphi*y
      perim=base+2.*cscphi*y
      area=y*(base+cotphi*y)
c         if iperim=0, R=A/B, else R=A/P:
      if (iperim.eq.0) then
          radius = area/topwid
      else
          radius = area/perim
      end if
      v = (SQRT(slope)*cm/rmning)*radius**(2./3.)
      q = v*area
      return
      end
```

Subr. YQREPUB: evaluate stream depth for gaged depth-flow relations (Repub. R.)

```
c-----------------------------------------------------------------------------
      subroutine YQREPUB (itrace,idgage,q,y,topwid,perim,area,v,ck,
     1                    bmain,qflood,yflood,bflood)
c usage:
ccc          call YQREPUB (itrdp1,idgage,Q,depth,
ccc  1                    topwid,perim,area,v,ck)

c          this applies to Concordia (idgage=1) and Clay Center (idgage=2)
c          gaging stations on the Republican River in Kansas:
c          a) stream depth, y(Q), is based on an average of depth rating curves
c          obtained from the ADAPS data base;
c          b) stream width, B(y), and transverse area of flow, A(y), are
c          based on transverse streambed profiles obtained from Butch Laycock,
c          USGS, Lawrence, KS.
c idgage    indicates whether gaging station is at Concordia (idgage=1) or
c          Clay Center (idgage=2).
c slope    =S0, streambed slope.
c q    flow rate (cfs)
c y        stream depth (ft)
c sf       friction slope; sf = slope since dq/dt is neglected.
c topwid   =B, stream width at water surface
c perim    =P, wetted perimeter; approximate by P = B (difference is negligible).
c area     =A, transverse area of flow computed by integrating dA = B*dy.
c v        =Q/A, flow velocity.
c ck       =dQ/dA = (1/B)dQ/dy, kinematic celerity (wave speed);
c                            might be approximated by ck = 1.6*v.
c
      data initlz /0/
      if (initlz.eq.0) then
          initlz=1
          Qmain = 200. !flow rate (cfs) corresponding to Bmain
          ymain = 1.85 !depth (ft) corresponding to Qmain
          Qflood = 18000. !flow rate (cfs) corresponding to Bflood
          yflood = 14. !depth (ft) corresponding to Qflood
      end if
      if (idgage.eq.1.or.idgage.eq.2) then
        if (Q.le.Qflood) then
          y = (1./6.)*Q**0.453
          dQdy = (6./.453)*Q**(1.-0.453)
        else
          y = (10./6.)*Q**0.217
          dQdy = (6./2.17)*Q**(1.-0.217)
        end if
        if (idgage.eq.1) then  ! Concordia
          bmain=130.           ! nominal main channel width
          bflood=206.          ! channel width for nominal flood condition
          if (y.le.1.6) then
              topwid = 9.7 + 70.8*y
              area = (9.7 + 35.4*y)*y
          else
              topwid = 118. + 6.28*y
              area = (118. + 3.14*y)*y - 90.69
          end if
        else if (idgage.eq.2) then    !Clay Center
          bmain=160.           ! nominal main channel width
          bflood=426.          ! channel width for nominal flood condition
          if (y.le.1.6) then
              topwid = 5.66 + 58.6*y*y
              area = (5.66 + (58.6/3.)*y*y)*y
          else
              topwid = 119. + 22.*y
              area = (119. + 11.*y)*y - 129.5
          end if
        end if
        topwid = AMAX1(10.,topwid)
      end if
      perim = topwid
      if (Q.le.1.e-3.or.area.le.1.e-3) then
        v = 0.
      else
        v = Q/area
      end if
      if (topwid.le.1.e-3) then
        ck=0.
      else
        ck = dQdy/topwid           !ck = dQ/dA = (1/B)dQ/dy
      end if
      return
      end
```

Subr. YQWALNUT: evaluate stream depth for gaged depth-flow relations (Walnut Cr.)

```
c----------------------------------------------------------------------------
      subroutine YQWALNUT (itrace,idgage,q,y,topwid,perim,area,v,ck,
     1    bmain,qflood,yflood,bflood)

c usage:
ccc   call YQWALNUT (itrdp1,idgage,Q,depth,topwid,perim,area,v,ck,
ccc  1               bmain,qflood,yflood,bflood)

c         this applies to Albert KS gage on Walnut Creek for idgage=3.
c Power relationships for stream depth, top width, and transverse flow area
c are based on data from the ADAPS "Measurements" report for the Albert gage.
c
c idgage   indicates gaging station at Albert KS on Walnut Creek.
c q   flow rate (cfs)
c y       stream depth (ft)
c topwid  =B, stream width at water surface
c perim   =P, wetted perimeter; approximate by P = B (difference is negligible).
c area    =A, transverse area of flow computed by integrating dA = B*dy.
c v       =Q/A, flow velocity.
c ck      =dQ/dA = (1/B)dQ/dy, kinematic celerity (wave speed);
c                            might be approximated by ck = 1.6*v.
c (qflood,yflood,bflood): knee of B(y) curve at flood stage
      dimension a(3),b(3)
c              coefficients for y=aQ^b for three ranges:
c              Q <= 2016 cfs, 2016 < Q <= 2658.3, Q > 2658.3
      data qp1, yp1, qp2, yp2 /2016., 20.3, 2658.3, 21.6/ !y(Q) ftn boundaries
      data (a(j),j=1,3) /.2207,3.3356,16.0266/ !coefficients for y(Q) = aQ^b
      data (b(j),j=1,3) /.5939,.237,.03795/    !exponents for y(Q) = aQ^b
      data qp3,yp3,bp3 /4530., 22.28, 1940./ !B(Q) ftn boundary
ccccc data qp3,yp3,bp3 /2222.436, 20.72, 122.32/ !B(Q) ftn boundary

      data bmin /5./   !min width = 5 ft
c       which occurs about 13% of the time for daily streamflow;
c       i.e., y(Qmean) = 30 ft (approx.), and P(Q < Qmean) = 0.87 (approx.)
      qflood = qp3 !external copies of (Qp3,yp3,bp3): knee of B(y) at flood stage
      yflood = yp3
      bflood = bp3
      bmain = 30. !width corresp. approximately to mean streamflow,

      If (q .le. 0) Then
        y=0.
        topwid=bmin
        perim=bmin
        area=0.
        v=0.
        ck=0.
      Else
c                            Depth y(Q), ft:
        If (q .le. qp1) Then
          idx=1
        ElseIf (q .le. qp2) Then
          idx=2
        Else
          idx=3
        End If
        y = a(idx)*Q**b(idx)
c                                   Area A(Q), ft**2:
        area = 7.2*Q**0.68
        v = Q/area
c                                   Note: for A=aQ^b, v = (1/a)*Q^(1-b) and ck=v/b.
        ck = v/0.68
c                                   top width B(Q), ft:
        If (Q .le. qp3) Then
          topwid = AMAX1(bmin,6.896 * Q ** 0.37316)
        Else
          topwid = 1.013e-5 * Q ** 2.116
        End If
        perim = topwid          !good approx for B >> y
      End If
      return
      end
```

Modwel: modified Well package (file modwel.for)

Definitions

```
c file modwel.for: modified WELL package
c
c       Modified array WELL:
c         WELL was formerly dimensioned to (4,mxwell); see note in WEL1AL
c         re expansion for MODSWB to allow (5) varying pumping rate with each
c         time step (option irropt in SWB1RP, SWBHYD) and (6) distinguishing
c         between pumping wells and wells that aren't to be weighted by Swat
c         irrigation fluxes under swb option irropt=1.  Such wells are to
c         represent b.c.'s or recharge wells, either of which should not be
c         weighted to correspond to irrigation flux calculated by Swat.--spp
c
c         change from 4*mxwell to 20*mxwell to accommodate (5) variation in
c         pumping rate with time step, to be determined in the watershed
c         module MODSWB, subr SWBHYD; and (6) distinguishing between
c         irrigation wells (=0) and recharge/b.c. wells (=1); see below.
c            (1) layer
c            (2) row
c            (3) column
c            (4) pumping rate as orig. unless irropt>0 (MODSWB), in which
c                case well(5)=rate read for stress period and well(4) is
c                scaled according to subbasin irrigation flux.
c            (5) pumping rate: appropriation.
c            (6) bccode: read as '*', retain as 1 to identify flux b.c. wells
c                not to be weighted by fluxes in module modswb; otherwise (default),
c                set to zero.
c            (7) SOURCE: read as 'S' or 'G', retain as 1:surface, 0:ground water
c            (8) usecod: numeric version of DWR use codes (3:irrigation)
c            (9) dsstrm distance to stream
c           (10) istrch: identifer for closest stream reach; index to order in
c                which the stream reach was read; required for surface water diversions.
c           (11) iwryr: water right year
c           (13) pmprat:  appropriated maximum pumping rate [L^3/T]
c           (14) areair:  appropriated area for irrigation (acres)
c           (15) qappr:   appropriated annual volume of water for irrigation (acre-ft)
c          greduc:  (Wel2fm) total reduction in gw pumping due to supply limits
c          sreduc:  (Str2fm) total reduction in sw diversions due to supply limits
c         revised water use input file format:
c  1-10: i10    ilay   layer
c 11-20: i10    irow   row
c 21-30: i10    icol   column
c 31-40: f10.0  Q      annual avg estimated water use [L^3/T], e.g. cfs
c 41-50  f10.0  pmprat appropriated maximum pumping rate
c 51-60  f10.0  dsstrm distance to stream (mi)
c 61-65  i5     istrch index to stream reach (order in which stream reach is read)
c                      If this is zero, a value from array idxstr is used
c                      to associate a stream reach with the surface water diversion.
c                      (see subr Srf2rp, below).
c 66-70  i5     iwryr  year of application for water right (taken as 1st yr of use)
c 71-80  a10    codes
c    71       a1   bndcnd '*' indicates b.c. well if iopbc>0 (Rattlesnake model)
c    72       a1   wrtype V: vested; A: appropriated; (others, e.g. T=temporary)
c    73       a1   source G=ground (well(7,*)=0), S=surface (streamflow, well(7,*)=1)
c    74-76    a3   usecod DWR use code: 3 for irrigation (default)
c
c  81-88        y_r grid row coordinate
c  89-96        x_c grid column coordinate
c 98-124 a27    wright: wellid

c subroutine usage in modflow:
cc        for each simulation:
cc    IF(IUNIT(2).GT.0) CALL WEL2AL(ISUM,LENX,LCWELL,MXWELL,NWELLS,
cc    1    iopsat,satlo,sathi,ratgnd,ratsrf,IUNIT(2),IOUT,IWELCB)
cc        for each stress period:
cc     IF(IUNIT(2).GT.0) CALL WEL2RP(X(LCWELL),NWELLS,MXWELL,
cc    1            welmpy,iadcod,IUNIT(2),IOUT)
cc        for each time step:
```

SUBROUTINE WEL2AL - Source file Listing

```
c---------------------------------------------------------------------------

      SUBROUTINE WEL2AL (ISUM,LENX,LCWELL,MXWELL,NWELLS,iopsat,
     1                satlo,sathi,ratgnd,ratsrf,IN,IOUT,IWELCB)
C
C-----VERSION 1538 12MAY1987 WEL1AL; oct 98 WEL2AL spp
C     ******************************************************************
C     ALLOCATE ARRAY STORAGE FOR WELL PACKAGE
C     ******************************************************************
C
C1------IDENTIFY PACKAGE AND INITIALIZE NWELLS
      WRITE(IOUT,1)IN
    1 FORMAT(1H0,'WEL1 -- WELL PACKAGE, VERSION 1, 9/1/87',
     1' INPUT READ FROM',I3)
      NWELLS=0
C
C2------READ MAX NUMBER OF WELLS AND
C2------UNIT OR FLAG FOR CELL-BY-CELL FLOW TERMS.
      READ(IN,2) MXWELL,IWELCB,opsat,satlo,sathi,ratgnd,ratsrf
2     FORMAT(2I10,5f10.0)
      iopsat = opsat
ccc   READ(IN,2) MXWELL,IWELCB
ccc 2 FORMAT(2I10)
      WRITE(IOUT,3) MXWELL
    3 FORMAT(1H ,'MAXIMUM OF',I5,' WELLS')
      IF(IWELCB.GT.0) WRITE(IOUT,9) IWELCB
    9 FORMAT(1X,'CELL-BY-CELL FLOWS WILL BE RECORDED ON UNIT',I3)
      IF(IWELCB.LT.0) WRITE(IOUT,8)
    8 FORMAT(1X,'CELL-BY-CELL FLOWS WILL BE PRINTED WHEN ICBCFL NOT 0')
      if (iopsat.gt.0) write (iout,'(1x,a,i5,2(1x,a,f8.2))')
     1    'iopsat=',iopsat,'satlo=',satlo,'sathi=',sathi
      if (ratgnd.gt.0.) write (iout,'(2(1x,a,f8.2))') 'ground '//
     1   'water: max. ratio (irrig. pumping rate/approp.) =',ratgnd
      if (ratsrf.gt.0.) write (iout,'(2(1x,a,f8.2))') 'surface '//
     1   'water: max. ratio (irrig. pumping rate/approp.) =',ratsrf
C
C3------SET LCWELL EQUAL TO LOCATION OF WELL LIST IN X ARRAY.
      LCWELL=ISUM
C-------------------------------------------------------------------------
C4------ADD AMOUNT OF SPACE USED BY WELL LIST TO ISUM.
      ISP=20*MXWELL
      ISUM=ISUM+ISP
C
C5------PRINT NUMBER OF SPACES IN X ARRAY USED BY WELL PACKAGE.
      WRITE(IOUT,4) ISP
    4 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED FOR WELLS')
      ISUM1=ISUM-1
      WRITE(IOUT,5) ISUM1,LENX
    5 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I8)
C
C6------IF THERE ISN'T ENOUGH SPACE IN THE X ARRAY THEN PRINT
C6------A WARNING MESSAGE.
      IF(ISUM1.GT.LENX) WRITE(IOUT,6)
    6 FORMAT(1X,'   ***X ARRAY MUST BE DIMENSIONED LARGER***')
C7------RETURN
      RETURN
      END
```

SUBROUTINE WEL2RP - Source file Listing

```
c
      SUBROUTINE WEL2RP (WELL,NWELLS,MXWELL,nwread,ratgnd,ratsrf,
     1            welmpy,iadcod,icalyr,nrow,ncol,nlay,in,iout)
C
C-----VERSION 1544 22DEC1982 WEL1RP; oct 98 WEL2RP spp
C     ******************************************************************
C     READ NEW WELL LOCATIONS AND STRESS RATES
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      DIMENSION WELL(20,MXWELL)
      character *1 bccode ! =' ' for std wells, '*' for wells to be
c                           distinguished as described above, for which
c                           well(6,j) will be set to 1, 0 otherwise.  This
c                           distinction will be relevant in the modswb module.
      character*1 source    !G: ground water, S: surface water (streamflow)
      character*1 usecod    !DWR use code (3: irrigation)  --spp
      character*10 codes
      character*20 wellid
C     ------------------------------------------------------------------
C
C1------READ ITMP(NUMBER OF WELLS OR FLAG SAYING REUSE WELL DATA)
      READ (IN,1) ITMP,nwread,iwelyr,frcuse
    1 FORMAT(3I10,f10.0)
      print '(2(1x,a,i4),1x,a,f8.3)',
     1    'iwelyr=',iwelyr,'iadcod=',iadcod,'welmpy=',welmpy
      if (itmp.lt.0) then
C1A-----    IF ITMP LESS THAN ZERO REUSE DATA. PRINT MESSAGE AND RETURN.
        WRITE(IOUT,6)
    6   FORMAT(1H0,'REUSING WELLS FROM LAST STRESS PERIOD')
        return
      end if
C
C1B-----ITMP=>0.  SET NWELLS EQUAL TO ITMP.
   50 NWELLS=ITMP
      IF(NWELLS.LE.MXWELL) GO TO 100
C
C2------NWELLS>MXWELL.  PRINT MESSAGE. STOP.
      WRITE(IOUT,99) NWELLS,MXWELL
   99 FORMAT(1H0,'NWELLS(',I4,') IS GREATER THAN MXWELL(',I4,')')
      STOP
C
C3------PRINT NUMBER OF WELLS IN CURRENT STRESS PERIOD.
  100 WRITE (IOUT,2) NWELLS,nwread
    2 FORMAT(1H0,10X,I4,' WELLS; read ',i4,' wells.')
C
C4------IF THERE ARE NO ACTIVE WELLS IN THIS STRESS PERIOD THEN RETURN
cspp    nwread=0 indicates wells have been read; only nwells was adjusted.
      IF(NWELLS.EQ.0 .or. nwread.eq.0) return
C
C5------READ AND PRINT LAYER,ROW,COLUMN AND RECHARGE RATE.
      WRITE(IOUT,3)
    3 FORMAT(1H ,47X,'LAYER    ROW    COL    STRESS RATE   WELL NO.',
     1 'b.c.(*) source use'/,48X,45('-'))
      qbc=0.
      qsum=0.
      qnonir=0.
      qirrig=0.
      qsurf=0.    !surface water diversions
      qgw=0.      !gw diversions
      qirrgw=0.   !gw diversions for irrigation
c     write (iout,'(a)') '   Layer      Row    Column  Q[L^3/T]'//
c    1    '    pmprat   Dsstrm Strch Wryr Codes'
      DO ii=1,NWread
        read (in,'(3i10,3f10.0,2i5,a,2f5.0,2f7.0,2f8.0)')
     1      k,i,j,Qwr,pmprat,dsstrm,istrch,iwryr,codes
ccccc1      areair,depirr,qappr,qmxgpm,yrow,xcol
```

```
        WELL(1,II)=K
        WELL(2,II)=I
        WELL(3,II)=J
        well(11,ii) = iwryr  !year of water right
        well(14,ii) = 0      !initialize sat. thickness status, isthk (WEL2ZFM)
c            isthk = -1 indicates satthk < satlo or recovering but < sathi.

c         wells used to specify cell fluxes (b.c.'s, recharge):
        if (codes(1:1).eq.'*') then
          well(6,ii) = 1
          qbc = qbc + Qwr
        else
          well(6,ii) = 0
        end if
        well(5,ii) = Qwr      !appropriated water use (subj. to use fraction
        well(4,ii) = Qwr
        if (INDEX('Ss',codes(3:3)).gt.0) then
          isourc = 1    !surface water source
        else
          isourc = 0    !ground water source
        end if
        well(7,ii) = isourc  !diversion source (1 = surface water, 0=gw)
        if (pmprat.ne.0.) then
          well(13,ii) = 2.*pmprat !appropriated pumping capacity [L^3/t] (max. rate)
        else
          if (isourc.eq.0) then
            well(13,ii) = ratgnd*ABS(Qwr)
          else
            well(13,ii) = ratsrf*ABS(Qwr)
          end if
        end if
        iwruse=0
        read (codes(4:6),'(i3)',iostat=iostat,err=150) iwruse !integer use codes
150     if(Index('  3IRRirrIrr',codes(4:6)).gt.0) iwruse=3 !Irrigation use
        well(8,ii) = iwruse  !DWR use code
c ----      water use management scenarios: apply to irrigation use only.
c           appropriated water use, subject to sensitivity factor, welmpy,
c           and management scenario code iadcod (see USEMGT)
        if (iadcod.gt.0.and.iwruse.eq.3) call USEMGT (iadcod,welmpy,
     1                 dsstrm,iwryr,icalyr,well(5,ii))
        well(9,ii)  = dsstrm       !distance to stream
        well(10,ii) = istrch       !index to stream reach
c       write (iout,'(3i10,3f10.5,2i5,a,2f5.0,2f7.0,2f8.0)')
c    1      k,i,j,Qwr,pmprat,dsstrm,istrch,iwryr,codes
ccccc1      areair,depirr,qappr,qmxgpm,yrow,xcol

c         categorize water rights (icalyr = simulated calendar year):
        Qsum = Qsum + well(5,ii)
        if (well(6,ii).eq.0) then  !for water rights (not for artificial b.c. wells):
            if (well(8,ii).ne.3) then   !any non-irrigation codes, incl. b.c.'s ("*")
              nonirr = nonirr + 1
              qnonir = qnonir + Qwr
c               irrig. wells more than dssmin (mi) from river and an assoc. river reach
            else
              numirr = numirr + 1       !no. irrigation wells
              qirrig = qirrig + Qwr     !total gw appropriated for irrigation
            end if
            if (well(7,ii).eq.1) then
              qsurf = qsurf + Qwr       !surface water right
            else
              qgw = qgw + Qwr           !ground water right
            end if
        end if                          !bccode <> '*'
      end do
      write (iout,'(2(1x,a,g15.7))') 'WEL1RP totals: qsum=',qsum,
     1                    'q b.c.(*) =',qbc
      write (iout,'(4(1x,a,g15.7))') 'Qirrig=',qirrig,
     1    'Qnonir=',qnonir, 'Qsurf=',qsurf,'Qgw=',qgw
C
C6------RETURN
```

```
 260 RETURN
     END
```

Subr USEMGT: water use management scenarios

```
c------------------------------------------------------------------------
      subroutine USEMGT (iadcod,welmpy,dsstrm,iwryr,icalyr,Qwr)
c ----    implement water use management scenarios---------------
c------------------------------------------------------------------------
c iadcod: use management option: this subroutine handles options 1-11 excepting 6-8,
defined below.
c welmpy: if iadcod > 0,scaling factor applied to water use for each water right;
c         if iadcod < 0, welmpy specifies a use fraction that is to be applied to all
c         water rights for all stress periods (years).
c dsstrm  distance from water right to stream (miles)
c iappno  water right application number;
c icalyr  calendar year of simulation; used for options 15 and 16 (below).
c Qwr     appropriation for a water right
c List of scenarios:
c  1 Scale irrigation appropriations by the factor welmpy;
c    e.g., to reduce water rights by 25%, set iadcod=1 and welmpy=0.75;
c    then water right appropriation Qwr is reduced by (1-welmpy)*Qwr.
c  2 Shut off all water rights applied for after 3/22/74 (applic. numbers > 22310);
c  3 Shut off all water rights applied for after 4/12/84 (applic. numbers > 37147);
c  4 Shut off all water rights within 1/4 mi of stream;
c  5 Shut off all water rights within 1/2 mi of stream;
c   6 Increase irrigated cropland area by 10%;
c  7 Decrease irrigated cropland area by 10%;
c  8 Increase pond storage by 25% (increase pond area by 25%);
c  9 Combine 2 and 4: shut off water rights applied for after 3/22/74 within 1/4 mi of
river;
c 10 Combine 3 and 5: shut off water rights applied for after 4/12/84 within 1/2 mi of
river;
c 11 Combine 9 and 10 as follows: shut off all water rights applied for after 3/22/74
within
c   1/4 mi of river, and all water rights after 4/12/84 within 1/2 mi of river.
c 12 If Qin < Qmin (i.e. inflow at top reach less than min. desired Q),
c        shut off water rights within 1/4 mi of river.
c 13 shut off all water rights.
c 14 If Qin < Qmin (see option 12 for def's), provide deficit with gw pumping
c 15 shut off water rights in 1989; hold Qin constant and tributary inflow
c       to zero for '88-'89.
c 16 shut off water rights in 1988-1989; hold Qin constant and tributary
c       inflow to zero for '88-'89.

c------------------------------------------------------------------------

      if (iadcod.gt.0) then
c             apply administrative options on water rights restrictions to Qnew:
c             = 1: multiply use and appropriation by factor welmpy:
        if (iadcod.eq.1) then
            Qwr = welmpy*Qwr
            return
        end if
        iadmin=0 !assume no use restriction, subject to options below:
        if (iadcod.eq.2) then
c             2: shut off add'l water rights after 3/22/74:
          if (iwryr.ge.1974) iadmin=1
        else if (iadcod.eq.3) then
c             3: shut off add'l water rights after 4/12/84:
          if (iwryr.ge.1974) iadmin=1
        else if (iadcod.eq.4) then
c             4: shut off water rights closer than 1/4 mile from stream
          if (dsstrm.lt.0.25) iadmin=1
        else if (iadcod.eq.5) then
c             5: shut off water rights closer than 1/2 mile from stream
          if (dsstrm.lt.0.5) iadmin=1
        else if (iadcod.eq.9) then
```

```
c             9: shut off add'l water rights after 3/22/74 within 1/4 mi of river
          if (iwryr.ge.1974 .and. dsstrm.lt.0.25) iadmin=1
        else if (iadcod.eq.10) then
c            10: shut off add'l water rights after 4/12/84 within 1/2 mi of river
          if (iwryr.ge.1984 .and. dsstrm.lt.0.5) iadmin=1
        else if (iadcod.eq.11) then
c            11: shut off add'l water rights after 3/22/74 within 1/4 mi of river;
c                shut off add'l water rights after 4/12/84 within 1/2 mi of river
          if ((iwryr.ge.1974 .and. dsstrm.lt.0.25) .or.
     1        (iwryr.ge.1984 .and. dsstrm.lt.0.5)) iadmin=1
        else if (iadcod.eq.15) then
          if (icalyr.eq.1989) iadmin=1
        else if (iadcod.eq.16) then
          if (icalyr.eq.1988.or.icalyr.eq.1989) iadmin=1
        end if
        if (iadmin.eq.1) then
          Qwr = 0.
        end if
      end if   !iadcod > 0
      return
      end
```

Subr WELZ2FM: scale pumping rates according to simulated irrigation

```
c-----------------------------------------------------------------------------
      SUBROUTINE WELZ2FM(NWELLS,MXWELL,WELL,shed,swtflx,strm,IBOUND,
     1  ibshed,hnew,bot,irropt,iopsat,satlo,sathi,ratgnd,ratsrf,
     1  srfdiv,qoprds,greduc,sreduc,ngwred,nswred,kkstp,kkper,
     1  nstrem,nwshed,NCOL,NROW,NLAY,iout)
c usage:
c       if (iunit(6).gt.0) then
c         call WELZ2FM(nwells,mxwell,x(LCwell),x(LCshed),
c     1       x(Lcwshd),x(LCstrm),x(LCIBOU),x(LCBSHD),x(LCHnew),
c     1       x(LCBOT),irropt,iopsat,satlo,sathi,ratgnd,ratsrf,
c     1       kkstp,kkper,nstrem,nwshed,NCOL,NROW,NLAY,ioreg)
c
C-----VERSION 1233 12MAY1987 WEL1FM; oct 98 WEL2FM spp
C
C     ******************************************************************
C     SUBTRACT Q FROM RHS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
c        see note in WEL2RP re expanding well dimension from 4 to 5.--spp
      double precision hnew(ncol,nrow,nlay)
ccccc DIMENSION RHS(NCOL,NROW,NLAY)
      dimension WELL(20,MXWELL),shed(30,0:nwshed),swtflx(30),
     1 strm(30,nstrem),IBOUND(NCOL,NROW,NLAY),ibshed(ncol,nrow),
     1 bot(ncol,nrow,nlay)
C     ------------------------------------------------------------------
      data iopmax /1/ !option limit pumping rate
C1------IF NUMBER OF WELLS <= 0 THEN RETURN.
      IF(NWELLS.LE.0) RETURN
      srfdiv=0.    !surface water diversions
      swtflx(23)=0.  !nonirrigation diversion
      swtflx(24)=0.  !irrigation diversions
      ngwred=0    !no. gw pumping rates reduced to maintain saturated thickness
      nswred=0    !no. sw pumping rates reduced
      qoprds=0.   !total gw+sw pumping reduced to satisfy rated capacity (pumping limit)
      greduc=0.   !total gw pumping reduction to maintain saturated thickness
      sreduc=0.   !total sw pumping
      if (kkper.eq.1.and.kkstp.eq.1) write(iout,'(a)')
     1    ' per step   n Wel Lay Row Col Sub Use Src Rch'//
     1    '           Q       Qwr Rate(DWR)     Satthk    Qfract'//
     1    '     Qreduc    Qopred Qirr/Qapp    Qapprop    Qirrig'
C2------PROCESS EACH WELL IN THE WELL LIST.
      DO L=1,NWELLS
```

```
        IR=WELL(2,L)
        IC=WELL(3,L)
        IL=WELL(1,L)
cccc              ibound(ic,ir,il)    !0:inactive; -1: const. head
C2A-----IF THE CELL IS INACTIVE THEN BYPASS PROCESSING.
        IF(IBOUND(IC,IR,IL).gt.0 .and. well(5,L).ne.0.) then
          iusdwr = well(8,L) !DWR use code (3: irrig; 0: nonirrigation)
          isourc = well(7,L) ! 1:surface, 0:gw (corresponds to DWR source, S(surface),
G(ground)
          well(4,L) = well(5,L)  !nonirrigation use or if Modswb not invoked
c             Vary irrigation pumping rate to meet simulated irrigation demand.
c             Scale factor=Qirr/Qapprop, from SWB2FM
          if (nwshed.gt.0 .and. iusdwr.eq.3 .and. irropt.gt.0) then
              izone = ibshed(ic,ir)       !subbasin
              if (izone.gt.0) well(4,L) = well(5,L)*shed(9,izone)
          end if
c             Observe operating limits: rated pumping capacity given for
c             each water right in DWR file (dwr_058.dbf) for both surface and
c             ground water sources.
          if (iopmax.gt.0) then          !option to limit pumping rates
              Qabs = ABS(well(4,L))
              Qmax = well(13,L)            !well's pumping rate limit
              if (Qmax.lt.Qabs) then    !upper limit < appropriated rate?
                Qopred = Qabs - Qmax      !reduction to meet operating limit
                Qoprds = Qoprds + Qopred !pass this sum to SWB2BD for ~.swm summary
                well(4,L) = SIGN(Qmax,well(4,L))
              else
                Qopred = 0.
              end if
ccc               Qmax = ratsrf*ABS(well(5,L))  !use surface rate max (WEL2AL)
ccc               Qmax = ratgnd*ABS(well(5,L))  !use gw rate max (WEL2AL)
ccc           well(4,L) = SIGN(AMIN1(Qabs,Qmax),well(4,L))
          end if
c
c             observe limit on ground water source: saturated thickness.
c             Limit on surface water source, inflow to a reach, is handled
c             by routing procedure in subr STR2FM (file modstrsp.for).
          satthk = hnew(ic,ir,il) - bot(ic,ir,il)
          if (isourc.eq.0 .and. iopsat.gt.0 .and.
     1        well(4,L).lt.0. .and. satthk.lt.sathi) then
c                     trim rate linearly over range (satlo,sathi):
            Qfract = AMAX1(0.,(satthk-satlo)/(sathi-satlo))
            Qreduc = (1.-Qfract)*ABS(well(4,L)) !reduced pumping for source L
            well(4,L) = Qfract*well(4,L)    !reduced pumping rate
          else
            Qfract=1.
            Qreduc=0.
          end if
          Q = well(4,L)
C
C2B-----   IF source is gw and THE CELL IS VARIABLE HEAD THEN SUBTRACT Q FROM
C          THE RHS ACCUMULATOR.
          if (isourc.eq.0) then            !source is ground water
ccc           RHS(IC,IR,IL)=RHS(IC,IR,IL)-Q  !this is done in Wel2fm, not here.
              if (iusdwr.eq.3) then
                swtflx(24) = swtflx(24) + Q  !irrigation ground water use
              else
                swtflx(23) = swtflx(23) + Q  !nonirrigation ground water use
              end if
              if (Qopred.ne.0. .or. Qreduc.ne.0.) then
                  greduc = greduc + Qopred + Qreduc  !accumulate reduced pumping
                  ngwred = ngwred + 1   !accumulate no. of gw wells reduced
              end if
          else                             !source is surface water
c                 subtract diversion from lateral inflow to stream reach.
c                 Note: this diversion will be supplied if net inflows to the
c                 reach can provide it during routing of the reach in STR2FM.
              istrch = well(10,L)          !index to associated stream reach
              if (1.le.istrch.and.istrch.le.nstrem) then
                strm(18,istrch) = strm(18,istrch) + Q
c                     !surface water diversions in subbasin:
```

```
              srfdiv = srfdiv + Q          !total surface water diversions
              swtflx(22) = swtflx(22) + Q  !add diversions to net trib. inflow
              if (Qopred.ne.0.) then
                  sreduc = sreduc + Qopred
                  nswred = nswred + 1   !accumulate no. of sw diversions reduced
              end if
            else
              print '(2(1x,a,i5))',
     1          'Welz2fm: Well surface water diversion, L=',L,
     1          'is not associated with a stream reach.'
              STOP
            end if
        end if
        if (Qopred.ne.0. .or. Qreduc.ne.0.) then
          nreduc = ngwred + nswred
          write (iout,'(2i5,9i4,10f10.3)') kkper,kkstp,nreduc,
     1      L,il,ir,ic,izone,iusdwr,isourc,istrch,well(4,L),
     1      well(5,L),well(13,L),satthk,qfract,qreduc,Qopred,
     1      shed(9,izone),shed(4,izone),shed(11,izone)
        end if
      end if
    end do
    RETURN
    END
```

SUBROUTINE WEL2FM - set up pumping rates for this time step

```
C-----------------------------------------------------------------------------
      SUBROUTINE WEL2FM(NWELLS,MXWELL,RHS,WELL,IBOUND,NCOL,NROW,NLAY)
C
C-----VERSION 1233 12MAY1987 WEL1FM; oct 98 WEL2FM spp
C
C     ******************************************************************
C     SUBTRACT Q FROM RHS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
*         see note in WEL1RP re expanding well dimension from 4 to 5.--spp
      DIMENSION RHS(NCOL,NROW,NLAY),WELL(20,MXWELL),
     1            IBOUND(NCOL,NROW,NLAY)
C     ------------------------------------------------------------------
C1------IF NUMBER OF WELLS <= 0 THEN RETURN.
      IF(NWELLS.LE.0) RETURN
C
C2------PROCESS EACH WELL IN THE WELL LIST.
      DO 100 L=1,NWELLS
      IR=WELL(2,L)
      IC=WELL(3,L)
      IL=WELL(1,L)
      Q=WELL(4,L)
      isourc = well(7,L)  !source: gw(0) or surface water (1)
C
C2A-----IF THE CELL IS INACTIVE or source is surface water THEN BYPASS PROCESSING.
      IF(IBOUND(IC,IR,IL).LE.0 .or.isourc.gt.0) GO TO 100
C
C2B-----    IF source is gw and THE CELL IS VARIABLE HEAD THEN SUBTRACT Q FROM
C           THE RHS ACCUMULATOR.
          RHS(IC,IR,IL)=RHS(IC,IR,IL)-Q
  100 CONTINUE
C
C3------RETURN
      RETURN
      END
```

SUBROUTINE WEL2BD - summary for overall budget

```
      SUBROUTINE WEL2BD(NWELLS,MXWELL,VBNM,VBVL,MSUM,WELL,IBOUND,DELT,
     1              NCOL,NROW,NLAY,KSTP,KPER,IWELCB,ICBCFL,BUFF,IOUT)
C
C-----VERSION 1509 12MAY1987 WEL1BD; oct 98 WEL2BD spp
C     ******************************************************************
C     CALCULATE VOLUMETRIC BUDGET FOR WELLS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
*         see note in WEL1RP re expanding well dimension from 4 to 5.--spp
      CHARACTER*4 VBNM,TEXT
      DIMENSION VBNM(4,MSUM),VBVL(4,MSUM),WELL(20,MXWELL),
     1          IBOUND(NCOL,NROW,NLAY),BUFF(NCOL,NROW,NLAY)
      DIMENSION TEXT(4)
C
      DATA TEXT(1),TEXT(2),TEXT(3),TEXT(4) /'    ','    ','   W','ELLS'/
C     ------------------------------------------------------------------
C
C1------CLEAR RATIN AND RATOUT ACCUMULATORS.
      RATIN=0.
      RATOUT=0.
      IBD=0
C
C2------IF THERE ARE NO WELLS DO NOT ACCUMULATE FLOW
      IF(NWELLS.EQ.0) GO TO 200
C
C3------TEST TO SEE IF CELL-BY-CELL FLOW TERMS WILL BE RECORDED.
      IF(ICBCFL.EQ.0 .OR. IWELCB.LE.0) GO TO 60
C
C4------IF CELL-BY-CELL FLOWS WILL BE SAVED THEN CLEAR THE BUFFER.
      IBD=1
      DO 50 IL=1,NLAY
      DO 50 IR=1,NROW
      DO 50 IC=1,NCOL
      BUFF(IC,IR,IL)=0.
   50 CONTINUE
C
C5------PROCESS WELLS ONE AT A TIME.
   60 DO 100 L=1,NWELLS
      IR=WELL(2,L)
      IC=WELL(3,L)
      IL=WELL(1,L)
      Q=WELL(4,L)
C
C5A-----IF THE CELL IS EXTERNAL or source is surface water IGNORE IT.
      IF(IBOUND(IC,IR,IL).LE.0 .or. well(7,L).gt.0)GO TO 100
C
C5B-----PRINT THE INDIVIDUAL RATES IF REQUESTED(IWELCB<0).
      IF(IWELCB.LT.0.AND.ICBCFL.NE.0) WRITE(IOUT,900) (TEXT(N),N=1,4),
     1    KPER,KSTP,L,IL,IR,IC,Q
  900 FORMAT(1H0,4A4,'   PERIOD',I3,'   STEP',I3,'   WELL',I4,
     1    '   LAYER',I3,'   ROW ',I4,'   COL',I4,'   RATE',G15.7)
C
C5C-----IF CELL-BY-CELL FLOWS ARE TO BE SAVED THEN ADD THEM TO BUFFER.
      IF(IBD.EQ.1) BUFF(IC,IR,IL)=BUFF(IC,IR,IL)+Q
      IF(Q) 90,100,80
C
C5D-----PUMPING RATE IS POSITIVE(RECHARGE). ADD IT TO RATIN.
   80 RATIN=RATIN+Q
      GO TO 100
C
C5E-----PUMPING RATE IS NEGATIVE(DISCHARGE). ADD IT TO RATOUT.
   90 RATOUT=RATOUT-Q
  100 CONTINUE
C
C6------IF CELL-BY-CELL FLOWS WILL BE SAVED CALL UBUDSV TO RECORD THEM
      IF(IBD.EQ.1) CALL UBUDSV(KSTP,KPER,TEXT,IWELCB,BUFF,NCOL,NROW,
     1                           NLAY,IOUT)
C
C7------MOVE RATES INTO VBVL FOR PRINTING BY MODULE BAS1OT.
  200 VBVL(3,MSUM)=RATIN
      VBVL(4,MSUM)=RATOUT
C
C8------MOVE RATES TIMES TIME STEP LENGTH INTO VBVL ACCUMULATORS.
      VBVL(1,MSUM)=VBVL(1,MSUM)+RATIN*DELT
      VBVL(2,MSUM)=VBVL(2,MSUM)+RATOUT*DELT
C
C9------MOVE BUDGET TERM LABELS INTO VBNM FOR PRINTING.
      VBNM(1,MSUM)=TEXT(1)
      VBNM(2,MSUM)=TEXT(2)
      VBNM(3,MSUM)=TEXT(3)
      VBNM(4,MSUM)=TEXT(4)
C
C10-----INCREMENT BUDGET TERM COUNTER(MSUM).
      MSUM=MSUM+1
C
C11-----RETURN
      RETURN
      END
```

Subr. WEL2BDZ - Well summary for zone budgets

```
c-------------------------------------------------------------------------------
      SUBROUTINE WEL2BDZ(NWELLS,MXWELL,VBNM,VBVL,MSUM,WELL,IBOUND,
     1     DELT,NCOL,NROW,NLAY,KSTP,KPER,IWELCB,ICBCFL,BUFF,IOUT,
     1     iopzon,mxzone,izonbd,vbvlzn,ratzin,ratzou)
C
C-----VERSION 1509 12MAY1987 WEL1BD; oct 98 WEL2BDZ spp
C     ******************************************************************
C     CALCULATE VOLUMETRIC BUDGET FOR WELLS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
*         see note in WEL1RP re expanding well dimension from 4 to 5.--spp
      CHARACTER*4 VBNM,TEXT
      DIMENSION VBNM(4,MSUM),VBVL(4,MSUM),WELL(20,MXWELL),
     1          IBOUND(NCOL,NROW,NLAY),BUFF(NCOL,NROW,NLAY),
     1          izonbd(ncol,nrow,nlay),vbvlzn(4,20,0:mxzone),
     1          ratzin(0:mxzone),ratzou(0:mxzone)
      DIMENSION TEXT(4)
C
      DATA TEXT(1),TEXT(2),TEXT(3),TEXT(4) /'    ','    ','   W','ELLS'/
C     ------------------------------------------------------------------
C
C1------CLEAR RATIN AND RATOUT ACCUMULATORS.
      RATIN=0.
      RATOUT=0.
c zone budget option (MODPOST package): initialize flow rate for each zone
      if (iopzon.gt.0) call ZONINI (mxzone,ratzin,ratzou)
      IBD=0
C
C2------IF THERE ARE NO WELLS DO NOT ACCUMULATE FLOW
      IF(NWELLS.EQ.0) GO TO 200
C
C3------TEST TO SEE IF CELL-BY-CELL FLOW TERMS WILL BE RECORDED.
      IF(ICBCFL.EQ.0 .OR. IWELCB.LE.0) GO TO 60
C
C4------IF CELL-BY-CELL FLOWS WILL BE SAVED THEN CLEAR THE BUFFER.
      IBD=1
      DO 50 IL=1,NLAY
      DO 50 IR=1,NROW
      DO 50 IC=1,NCOL
      BUFF(IC,IR,IL)=0.
   50 CONTINUE
```

```
C
C5------PROCESS WELLS ONE AT A TIME.
   60 DO 100 L=1,NWELLS
      IR=WELL(2,L)
      IC=WELL(3,L)
      IL=WELL(1,L)
      Q=WELL(4,L)
C
C5A-----IF THE CELL IS EXTERNAL or source is surface water IGNORE IT.
      IF(IBOUND(IC,IR,IL).LE.0 .or. well(7,L).gt.0) GO TO 100
C
C5B-----PRINT THE INDIVIDUAL RATES IF REQUESTED(IWELCB<0).
      IF(IWELCB.LT.0.AND.ICBCFL.NE.0) WRITE(IOUT,900) (TEXT(N),N=1,4),
     1    KPER,KSTP,L,IL,IR,IC,Q
  900 FORMAT(1H0,4A4,'   PERIOD',I3,'   STEP',I3,'   WELL',I4,
     1    '   LAYER',I3,'   ROW ',I4,'   COL',I4,'   RATE',G15.7)
C
C5C-----IF CELL-BY-CELL FLOWS ARE TO BE SAVED THEN ADD THEM TO BUFFER.
      IF(IBD.EQ.1) BUFF(IC,IR,IL)=BUFF(IC,IR,IL)+Q
      IF(Q) 90,100,80
C
C5D-----PUMPING RATE IS POSITIVE(RECHARGE). ADD IT TO RATIN.
   80 RATIN=RATIN+Q
      GO TO 95
C
C5E-----PUMPING RATE IS NEGATIVE(DISCHARGE). ADD IT TO RATOUT.
   90 RATOUT=RATOUT-Q
c        zone budget option (MODPOST package):
   95 if (iopzon.gt.0)
     1  call ZONACC (Q,izonbd(ic,ir,il),mxzone,ratzin,ratzou)
  100 CONTINUE
C
C6------IF CELL-BY-CELL FLOWS WILL BE SAVED CALL UBUDSV TO RECORD THEM
      IF(IBD.EQ.1) CALL UBUDSV(KSTP,KPER,TEXT,IWELCB,BUFF,NCOL,NROW,
     1                          NLAY,IOUT)
C
C7------MOVE RATES INTO VBVL FOR PRINTING BY MODULE BAS1OT.
  200 VBVL(3,MSUM)=RATIN
      VBVL(4,MSUM)=RATOUT
C
C8------MOVE RATES TIMES TIME STEP LENGTH INTO VBVL ACCUMULATORS.
      VBVL(1,MSUM)=VBVL(1,MSUM)+RATIN*DELT
      VBVL(2,MSUM)=VBVL(2,MSUM)+RATOUT*DELT
c zone budget option (MODPOST package): accumulate flow rate for each zone
      if (iopzon.gt.0)
     1  call ZONSUM(msum,mxzone,delt,ratzin,ratzou,vbvlzn)
C
C9------MOVE BUDGET TERM LABELS INTO VBNM FOR PRINTING.
      VBNM(1,MSUM)=TEXT(1)
      VBNM(2,MSUM)=TEXT(2)
      VBNM(3,MSUM)=TEXT(3)
      VBNM(4,MSUM)=TEXT(4)
C
C10-----INCREMENT BUDGET TERM COUNTER(MSUM).
      MSUM=MSUM+1
C
C11-----RETURN
      RETURN
      END
```

Modrsd package: ground water level measurements (file modrsd.for)

```
c file modrsd.for: Modflow package to read water level measurements (Rsd1Rp)
c and calculate water level residuals, given by simulated - measured heads (Rsd1wl).
```

Subr. RSD1AL: allocate arrays.

```
c
      SUBROUTINE RSD1AL(ISUM,LENX,LCobs,mxobs,numobs,ioprsd,dtwmax,
     1    IN,IOUT,iobscb)
c usage in modflo:
c     CALL RSD1AL(ISUM,LENX,LCobs,MXobs,numobs,ioprsd,dtwmax,iunit(16),
c    1            IOUT,iobscb)
C
C-----VERSION 1538 12MAY1987 RSD1AL
C     ******************************************************************
C     ALLOCATE ARRAY STORAGE FOR gw observations PACKAGE
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C
C1------IDENTIFY PACKAGE AND INITIALIZE numobs=no. gw level observations
      WRITE(IOUT,1)IN
    1 FORMAT(1H0,'RSD1: GW level observations PACKAGE, V.1, 1987-09-01',
     1' INPUT READ FROM',I3)
      numobs=0
C
C2------READ MAX NUMBER OF observations AND
C2------UNIT OR FLAG FOR CELL-BY-CELL FLOW TERMS.
      if (in.gt.0) then
          READ(IN,2) mxobs,iobscb,ioprsd,dtwmax
2         FORMAT(3I10,f10.0)
      else
          mxobs = 1
      end if
      if (dtwmax.eq.0.) dtwmax = 40. !special case for Repub. R. model
      WRITE(IOUT,3) mxobs
    3 FORMAT(1H ,'MAXIMUM OF',I5,' water level measurements')
      IF(iobscb.ne.0) WRITE(IOUT,8)
    8 FORMAT(1X,'measured water levels WILL BE PRINTED')
      write (iout,'(2(1x,a,i5))') 'ioprsd=',ioprsd
C
C3------SET LCobs EQUAL TO LOCATION OF WELL LIST IN X ARRAY.
      LCobs=ISUM        !gwobs(mxobs,25)
      ISP =25*mxobs
      ISUM=ISUM+ISP
C4------ADD AMOUNT OF SPACE USED BY array gwobs(25*mxobs) to ISUM
C
C5------PRINT NUMBER OF SPACES IN X ARRAY USED BY WELL PACKAGE.
      WRITE(IOUT,4) ISP
    4 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE FOR ',
     1  'measured water levels ')
      ISUM1=ISUM-1
      WRITE(IOUT,5) ISUM1,LENX
    5 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I8)
C
C6------IF THERE ISN'T ENOUGH SPACE IN THE X ARRAY THEN PRINT
C6------A WARNING MESSAGE.
      IF(ISUM1.GT.LENX) WRITE(IOUT,6)
    6 FORMAT(1X,'   ***X ARRAY MUST BE DIMENSIONED LARGER***')
C7------RETURN
      RETURN
      END
```

Subr. RSD1RP: read observations for stress period (year).

```
c-----------------------------------------------------------------------------
      SUBROUTINE RSD1RP (gwobs,ibound,ncol,nrow,nlay,mxobs,numobs,
     1                   iyrper,in,iout,iobscb)

cc    IF(IUNIT(16).GT.0) CALL RSD1RP(X(LCOBS),x(LCIBOU),ncol,nrow,
cc   1          nlay,mxobs,numobs,iyrper,IUNIT(16),IOUT,iobscb)
C
C-----VERSION 1544 22DEC1982 RSD1RP
C     ******************************************************************
C     READ NEW WELL LOCATIONS AND water level measurements
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      DIMENSION gwobs(25,mxobs), ibound(ncol,nrow,nlay)
      character wright*27
      character*1 wruse   !DWR use code (except that 0= dtw measurements only)
      character*10 digits !for comparison with wruse
      character datsrc*5, legloc*20
      data digits /'1234567890'/
C     ------------------------------------------------------------------
C
C1------READ NUMBER OF water level observations and year
      READ (in,1) numobs,iyrper
    1 FORMAT(2I10)
      if (numobs.le.0) return
C
C5------READ LAYER,ROW,COLUMN AND measured water level
      if (iobscb.gt.0) write (iout,'(a)') ' obs  lay  row  col'//
     1    ' year  mon WL elev LS elev     dtw     dtb'
      ii=0
      DO iobs=1,numobs
          read (in,101) ilay,irow,icol,zw,elevls,dtw,dtb,
     1      iyrdtw,imodtw                          ! (3i10,f10.0,5i5)
cccc 1      ,wruse,rwr,cwr,wright
101       format (3i10,f10.0,6i5,1x,a1,2f8.0,1x,a)
cc        1        21         36      1177.8 1190    12       1994    11       3    20.56    35.56
8S 2E13  dbb       W      CLAY CTR SW   "
cc   layer       row      column     wlevel zlnd  dtw  dtb  iyr  mon nobs u      row  column
file_id--TspRngSc  Subsec obs. map
c-----------------------------------------------------------------------------
        if (1.le.ilay.and.ilay.le.nlay .and.
     1      1.le.irow.and.irow.le.nrow .and.
     1      1.le.icol.and.icol.le.ncol) then
         if (ibound(icol,irow,ilay).eq.1) then
          ii = ii + 1              !index to list of observations on active cells
          if (iobscb.gt.0) write (iout,102) ii,ilay,irow,icol,
     1        iyrdtw,imodtw,zw,elevls,dtw,dtb
 102      format (6i5,4f8.2)
          IF (ii.gt.mxobs) then
            WRITE(IOUT,99) ii,mxobs
   99       FORMAT(1H0,'numobs(',I4,') > mxobs(',I4,')')
            STOP
          end if
          gwobs(1,II)=ilay
          gwobs(2,II)=irow
          gwobs(3,II)=icol
          gwobs(4,ii)=zw
          gwobs(5,ii) = elevls
          gwobs(6,ii) = dtw
          gwobs(7,ii) = dtb
          gwobs(8,ii) = iyrdtw
          gwobs(9,ii) = imodtw
c
cc        gwobs(10,ii) = nmeas
cc        if (INDEX(datsrc,'KGS').gt.0) then
cc            gwobs(10,ii)=1 !group 1: KGS WL data;
cc        else if (INDEX(datsrc,'WL').gt.0) then
cc            gwobs(10,ii)=2 !group 2: interpolated WL data (for years 1981, 1996)
cc                           A third group may come out of DWR use reports.
cc          end if
cc          iwruse=INDEX(digits,wruse) !set up numeric version of DWR use code
cc          if (wruse.eq.' ') iwruse = 3 !assume blank field indicates irrigation type
cc          gwobs(11,ii) = iwruse
cc          gwobs(12,ii) = rwr
cc          gwobs(13,ii) = cwr
c           if (wright(1:1).eq.'V') then
c               read (wright(4:7),'(i4)') iappno
c           else if (wright(1:1).eq.'A') then
c               read (wright(2:7),'(i6)') iappno
c           else
c               iappno = 0
c           end if
c           gwobs(14,II)=idwell
          end if  !observation lies within active cell, i.e., ibound(ic,ir,il)=1.
         end if  !observation lies within grid domain
      end do
      numobs = ii  !no. useful observations (located within active cells)
      WRITE (IOUT,2) numobs
    2 FORMAT(1H0,10X,I4,' gw level observations')
      RETURN
      END
```

Subr. RSD1WL: calculate residuals in each time step.

```
c-----------------------------------------------------------------------------
      SUBROUTINE RSD1WL (nresid,numobs,mxobs,iyrper,gwobs,IBOUND,
     1    hnew,bot,dtwmax,ioprsd,kkper,kkstp,nstp,NCOL,NROW,NLAY,
     1    iomeas,iobs)
C
cc summarize water level residuals (simulated - observed heads)
cc       usage in modflo():
cc            call RSD1WL (numobs,mxobs,iyrper,x(LCobs),x(LCIBOU),
cc    1           x(LCHNEW),x(LCBOT),dtwmax,ioprsd,KKPER,KKSTP,nstp,
cc    1           NCOL,NROW,NLAY,iomeas,iobs)
C-----VERSION 1544 31Oct1998 RSD1WL
C     ******************************************************************
c     Compare water level measurements with solution for heads
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      DOUBLE PRECISION HNEW
      DIMENSION HNEW(NCOL,NROW,NLAY),BOT(NCOL,NROW,NLAY),
     1    IBOUND(NCOL,NROW,NLAY),gwobs(25,mxobs)
c-----the following definition of array tsec is copied from file modswb.for.
C     ------------------------------------------------------------------
C4------IF THERE ARE NO ACTIVE WELLS IN THIS STRESS PERIOD THEN RETURN
      data initlz /0/
      IF(numobs.EQ.0) RETURN
C
c         iyrper was read from the well file in WEL1RP;
c
      if (kkstp.eq.1.or.nstp.eq.1) then
          nresid=0   !initialize to compute stats on residuals for each year
          avgsat=0.
          avgdtw=0.
          sumres=0.
          ssqres=0.
          if (kkper.eq.1) then
            ntot=0      !sum over all stress periods
            restot=0.
            ssqtot=0.
          end if
      end if
      if (initlz.eq.0) then
```

```
         initlz = 1              ! write headings once
         write (iomeas,'(1x,a,i5,1x,a,f8.2)')
     1        ' RSD1WL: ioprsd=',ioprsd,'dtwmax=',dtwmax

         write (iomeas,'(a)') '  obs  rsd year kper kstp hnew-zw'//
     1     '    hnew      zw  satthk idtw idtb irow icol o_yr o_mo'//
     1     '  timobs'
         write (iobs,'(3(a,i5))') '  yr '//
     1        '  res  stddev  avgres  tot  sdvtot  avgtot'//
     1        '  avgdtw  avgsat  nstp=',nstp
       end if
c-----------------------------------------------------------------------
       DO II=1,numobs
         ilay = gwobs(1,ii)
         irow = gwobs(2,ii)
         icol = gwobs(3,ii)
         zw   =    gwobs(4,ii)    !observed water table elevation
         dtw =     gwobs(6,ii)    !reported dtw measurement (ft)
         iyrdtw = gwobs(8,ii)     !reported year of dtw measurement
         imodtw = gwobs(9,ii)     !reported month of dtw measurement
         if  (ibound(icol,irow,ilay).gt.0 .and. zw.gt.0. .and.
     1       (ioprsd.eq.0 .or. dtw.le.dtwmax) .and.
     1       (imodtw.eq.kkstp .or. nstp.eq.1)) then
           nresid = nresid + 1
           gwobs(20,ii) = nresid !residual is included; retain this order
           idtw = dtw
           idtb =    gwobs(7,ii)     !reported depth to bottom of well (ft)
           h = hnew(icol,irow,ilay)
           satthk = h - bot(icol,irow,ilay) !saturated thickness
           resid =  h - gwobs(4,ii)  !calc. - meas.
           avgdtw = avgdtw + dtw
           avgsat = avgsat + satthk
           sumres = sumres + resid
           ssqres = ssqres + resid*resid
           ntot = ntot + 1
           restot = restot + resid
           ssqtot = ssqtot + resid*resid
           tmeas = FLOAT(iyrdtw) + (FLOAT(imodtw)-0.5)/12.
           write (iomeas,14) ii,nresid,iyrper,kkper,kkstp,resid,h,zw,
     1       satthk,idtw,idtb,irow,icol,iyrdtw,imodtw,tmeas
 14        format (5i5,4f8.2,6i5,f9.3)
         else
           gwobs(20,ii) = 0   !observation is excluded
         end if
       end do
       if (nstp.eq.1.or.kkstp.eq.12) then
           if (nresid.gt.0) then
               xresid = nresid
               avgres = sumres/xresid
               stddev=0.
               if (nresid.gt.1) stddev = SQRT(ssqres/(xresid-1.))
               avgdtw = avgdtw/xresid
               avgsat = avgsat/xresid

               xtot = ntot
               avgtot = restot/xtot
               sdvtot=0.
               if (ntot.gt.1) sdvtot = SQRT(ssqtot/(xtot-1.))
               write (iobs,200) iyrper,nresid,stddev,avgres,
     1             ntot,sdvtot,avgtot,avgdtw,avgsat
 200           format (i5,2(i5,2f8.2),2f8.2)
           end if
       end if
       RETURN
       END
```

Modpost: postprocessing package (file modpost.for)

Subr. POST1AL: allocate arrays

```
      subroutine POST1AL(ISUM,LENX,LCidry,LCXLOC,LCYLOC,LCZLOC,LCKT,
     1  LC23,ICTser,ICLser,ICRser,ICCser,ICXser,ICVser,NCOL,NROW,NLAY,
     1  mxser,numser,idxtim,idxvol,instzn,iopzon,mxzone,IOUT,inpost)

C-----VERSION 1515 12MAY1987 POST1AL
C     ******************************************************************
C     ALLOCATE SPACE FOR Postprocessor ARRAYS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      character*1 ans, optnet
C     ------------------------------------------------------------------
      nrcl = ncol*nrow*nlay
C
C1------PRINT A MESSAGE IDENTIFYING THE PACKAGE.
      if (inpost.gt.0) then
          WRITE(IOUT,1)inpost
1         FORMAT(1H0,'POST1 -- Postprocessing PACKAGE, VERSION 1, '//
     1        '9/1/97, INPUT READ FROM UNIT',I3)
          ans='S' !default time units (sec)
          call PROMPT2 ('Show elapsed time in [S]ec [M]in [H]our '//
     1              '[D]ay [Y]ears',ans,inpost)
          idxtim=INDEX('SMHDY',ans) !index to time unit
          print '(1x,a)','Enter number of time series (numser), '//
     1        'FREE format: '
          read (inpost,*) numser
          end if
c-------------Budget options
          ans='V'
          call PROMPT2 ('Extract budget cumulative [V]olumes, '//
     1              '[R]ates or [N]either?',ans,inpost)
          idxvol=INDEX('VR',ans)
          if (idxvol.gt.0) then
            !Budget option: N=net difference(in-out), S=Separate in,out components
ccc         ans='N'
ccc         call PROMPT2 ('Show [N]et difference (in-out) or '//
ccc  1        '[S]eparate (in, out) budget components?',ans,inpost)
ccc         idxnet=INDEX('N',ans)    !idxnet=1 for net budget terms, 0 otherwise
            if (instzn.gt.0) then    !Zone Budget option installed? (y=1,n=0)
              print '(1x,a)','iopzon = zone budgets option:'
              print '(1x,a)','if iopzon > 0, max. no. zones '//
     1            'mxzone = iopzon + 1, for zones 0 to iopzon.'
              print '(1x,a)',
     1            'Enter number of budget zones, FREE format:'
              read (inpost,*) iopzon
              mxzone = MAX0(1,iopzon+1) !dimensioning parameter for budget zones
              if (iopzon.gt.0) print '(i5,1x,a)',iopzon,'zones; '//
     1          'enter 2-d array izonbd to define zone domain: '
            end if
      end if
      idxtim=MAX0(1,idxtim)      !index to time unit
      mxser = MAX0(1,numser)     !dimensioning parameter for time series
C
C3------STORE,IN ISOLD, LOCATION OF FIRST UNALLOCATED SPACE IN X.
      ISOLD=ISUM
C
C4------ALLOCATE SPACE FOR ARRAYS.
      if (inpost.gt.0) then
          LCidry=ISUM     !idry(ncol,nrow,nlay): identify dry nodes (idry=1)
          isum=isum+NRCL
          LCXLOC=ISUM     !xx(ncol): x-coordinates for columns
```

```
         ISUM=ISUM+NCOL
         LCYLOC=ISUM     !yy(nrow): y-coordinates for rows
         ISUM=ISUM+NROW
         LCZLOC=ISUM
         ISUM=ISUM+NLAY
         LCKT=ISUM                    !kt(nlay), used in HYD1OT
         ISUM=ISUM+NLAY
         LC23=ISUM                    !
         ISUM=ISUM+NLAY
         ICTser=isum                  !integer idtype(mxser)
         isum=isum+mxser
         ICLser=isum                  !integer layser(mxser)
         isum=isum+mxser
         ICRser=isum                  !integer rowser(mxser)
         isum=isum+mxser
         ICCser=isum                  !integer colser(mxser)
         isum=isum+mxser
         ICXser=isum                  !integer idxrch(mxser)
         isum=isum+mxser
         ICVser=isum                  !real valser(mxser)
         isum=isum+mxser
      end if
      ISP=ISUM-ISOLD
C
C7------PRINT AMOUNT OF SPACE USED.
      WRITE(IOUT,6) ISP
    6 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED BY POS')
      ISUM1=ISUM-1
      WRITE(IOUT,7) ISUM1,LENX
    7 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I8)
      IF(ISUM1.GT.LENX) WRITE(IOUT,8)
    8 FORMAT(1X,'   ***X ARRAY MUST BE DIMENSIONED LARGER***')
      RETURN
      END
```

Subr. ZON1AL: allocate arrays for Zone Budget option.

```
c-----------------------------------------------------------------------
      subroutine ZON1AL(ISUM,LENX,iopzon,mxzone,LCIzon,LCIzbd,LCIzbf,
     1  LCIzin,LCIzou,LCIxfi,LCIxfo,LCItiz,LCItoz,LCIncz,LCIacz,
     1  NCOL,NROW,NLAY,IOUT,inpost)
C-----VERSION 1515 12MAY1987 POST1AL
C     ******************************************************************
C     ALLOCATE SPACE FOR Postprocessor ARRAYS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      nrcl = ncol*nrow*nlay
C
C1------PRINT A MESSAGE IDENTIFYING THE PACKAGE.
      if (inpost.gt.0) then
         WRITE(IOUT,1)inpost
1        FORMAT(1H0,'POST1 -- Postprocessing PACKAGE, VERSION 1, '//
     1         '9/1/97, INPUT READ FROM UNIT',I3)
      end if
      print '(3(1x,a,i5))','ZON1AL: iopzon=',iopzon,'mxzone=',mxzone
C
C3------STORE,IN ISOLD, LOCATION OF FIRST UNALLOCATED SPACE IN X.
      ISOLD=ISUM
C
C4------ALLOCATE SPACE FOR ARRAYS.
c-------------------------add zone budget arrays------------------------------
      Lcizon=isum  ! izonbd(ncol,nrow,nlay): volumetric budget zones > 0
      isum = isum + nrcl
      Lcizbd=isum  ! vbvlzn(4,20,0:mxzone):  vol. budget terms for each zone
      isum = isum + 4*20*(1+mxzone)
      Lcizbf=isum  ! vbznfl(2,0:mxzone,0:mxzone): flow between zones
      isum = isum + 2*(1+mxzone)*(1+mxzone)
      Lcizin=isum  ! ratzin(0:mxzone): accumulate inflow over cells for each zone
      isum = isum + (1+mxzone)
      Lcizou=isum  ! ratzou(0:mxzone): accumulate outflow over cells for each zone
      isum = isum + (1+mxzone)
      LCIxfi=isum  ! xfrinp(2,0:mxzone): flow into zone from other zones
      isum = isum + 2*(1+mxzone)
      LCIxfo=isum  ! xfrout(2,0:mxzone): flow from zone into other zones
      isum = isum + 2*(1+mxzone)
      LCItiz=isum  ! totizn(2,0:mxzone): total flow into zone
      isum = isum + 2*(1+mxzone)
      LCItoz=isum  ! totozn(2,0:mxzone): total flow from zone
      isum = isum + 2*(1+mxzone)
      LCIncz=isum  ! nczone(0:mxzone): no. grid cells in each zone (ibound = 0)
      isum = isum + (1+mxzone)
      LCIacz=isum  ! ncaczn(0:mxzone): no. ACTIVE grid cells in each zone (ibound > 0)
      isum = isum + (1+mxzone)
c-----------------------------------------------------------------------------
      ISP=ISUM-ISOLD
C
C7------PRINT AMOUNT OF SPACE USED.
      WRITE(IOUT,6) ISP
    6 FORMAT(1X,I8,' ELEMENTS IN X ARRAY ARE USED BY POS',
     1             ' for ZONE budget')
      ISUM1=ISUM-1
      WRITE(IOUT,7) ISUM1,LENX
    7 FORMAT(1X,I8,' ELEMENTS OF X ARRAY USED OUT OF',I8)
      IF(ISUM1.GT.LENX) WRITE(IOUT,8)
    8 FORMAT(1X,'    ***X ARRAY MUST BE DIMENSIONED LARGER***')
      RETURN
      END
```

Subr POST1RP: read postprocessing directions

```
c--------------------------------------------------------------------------
      SUBROUTINE POST1RP (msum,iunit,idxtim,budlbl,iobud,ionet)
C-----VERSION 1522 12MAY1997 POST1RP
c usage in modflo:
cc        call POST1RP (msum,iunit,idxtim,budlbl,iobud,ionet)
C***********************************************************************
C     OUTPUT TIME, VOLUMETRIC BUDGET, HEAD, AND DRAWDOWN
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      integer nper
      dimension iunit(24)
Cspp----------------------------------------------------------------------
      character*3 modlbl(15), budlbl(9)
      character*4 aname(6,2), timlbl(0:5)
      integer mxiter
c         to retrieve max. head change for each iteration (subr SSIP1P):
      parameter (mxiter=200)
cccc  real HDCG(MXITER)
cccc  integer LRCH(3,MXITER)
      integer idxlbl(15)
ccc   integer idxcod, ibcfcb
C*         maximum vector length (for either rows or columns):
      integer mxlen
      parameter (mxlen=2000)
      parameter (mxstp=2000)
      integer iobud
c         labels for time units corresponding to idxtim (chosen below):
      data timlbl /'time','secs','mins',' hrs','days',' yrs'/
c         labels for modules coresponding to i/o unit codes:
c         (note: STR is in slots 13 and 14 to cover both std version
c         of MODFLOW and the Doherty version of MODFLOW.)
      data (modlbl(j),j=1,15) /'BCF','WEL','DRN','RIV','EVT',
     1        'XXX','GHB','RCH','SIP','GHB','SOR','OC ','STR','STR',
     1        'POS'/
c          sequence in which items are added to budget:
c          WEL DRN RCH EVT RIV STR GHB.
      data (idxlbl(j),j=1,7) /2,3,8,5,4,13,7/
      DATA ANAME(1,1),ANAME(2,1),ANAME(3,1),ANAME(4,1),ANAME(5,1),
     1   ANAME(6,1) /'    ',' Bu','dget',' zon','es a','rrau'/
c-----------------------------------------------------------------------
cccc  inpodst = iunit(15)
*           McDonald & Harbaugh (USGS) standard MODFLOW uses iunit(13)
*           to indicate STREAM package, but Doherty's version uses iunit(14).
*           copy iunit(14) to iunit(13) if iunit(14)>0.
      budlbl(1)='STO'
      budlbl(2)='CON'
      msum=2
*           sequence in which items are added to budget in MODFLOW:
c            2   3   8   5   4   13  7
c           WEL DRN RCH EVT RIV STR GHB.
      do j=1,7
        i = idxlbl(j)
        if (iunit(i).gt.0) then
          msum=msum+1
          budlbl(msum) = modlbl(i)
          print '(2(1x,i3,1x,a3))', msum,budlbl(msum),i,modlbl(i)
        end if
      end do
c         In MODFLOW, msum ends up 1 larger than the number of individual budget
c         entries due to the way msum is incremented.  Do the same thing here
c         so that POST1OT can be called by both POSTMD97 and MODFLOW:
      msum1=msum
      msum=msum+1
c          INPUT and OUTPUT components budget file:
      write (iobud,329) ' stp per itr  -----Delt-----'//timlbl(idxtim),
     1    (budlbl(j),j=1,msum1),'inp',
     1    (budlbl(j),j=1,msum1),'out','I-O','pct'
329   format(a35,25(9x,a3))
c          NET budget file:
      write (ionet,'(a,9(1x,a,1x,6hin-out,1x))')
     1    ' stp per itr -----Delt-----      '//timlbl(idxtim)//
     1    '  % diff  tot in-out', (budlbl(j),j=1,msum1)
      return
      end
```

Subr. ZON1RP: read Zone Budget directions

```
c-----------------------------------------------------------------------
      SUBROUTINE ZON1RP (msum,idxtim,iopzon,mxzone,nzones,
     1   nlay,nrow,ncol,ibound,izonbd,vbvlzn,vbznfl,xfrinp,xfrout,
     1   nczone,ncaczn,budlbl,inpost,ionetz,iout)
C-----VERSION 1522 12MAY1997 POST1RP
c usage in modflo:
c     call ZON1RP (msum,idxtim,iopzon,mxzone,nzones,
c     1        nlay,nrow,ncol,x(LCIBOU),x(LCIzon),x(LCIzbd),x(LCIzbf),
c     1        x(LCIxfi),x(LCIxfo),x(LCIncz),x(LCIacz),budlbl,iunit(15),
c     1        ionetz,iout)
C***********************************************************************
C     OUTPUT TIME, VOLUMETRIC BUDGET, HEAD, AND DRAWDOWN
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      dimension ibound(ncol,nrow,nlay),
     1    izonbd(ncol,nrow,nlay), vbvlzn(4,20,0:mxzone),
     1    vbznfl(2,0:mxzone,0:mxzone), xfrinp(2,0:mxzone),
     1    xfrout(2,0:mxzone), nczone(0:mxzone), ncaczn(0:mxzone)
Cspp-------------------------------------------------------------------------
        character*3 budlbl(9)
        character*4 aname(6,2), timlbl(0:5)
        integer mxiter
*           to retrieve max. head change for each iteration (subr SSIP1P):
        parameter (mxiter=200)
cccc    real HDCG(MXITER)
cccc    integer LRCH(3,MXITER)
        integer idxlbl(15)
ccc     integer idxcod, ibcfcb
C*           maximum vector length (for either rows or columns):
        integer mxlen
        parameter (mxlen=2000)
        parameter (mxstp=2000)
        integer iobud
c           labels for time units corresponding to idxtim (chosen below):
        data timlbl /'time','secs','mins',' hrs','days',' yrs'/
*           labels for modules coresponding to i/o unit codes:
*           (note: STR is in slots 13 and 14 to cover both std version
*           of MODFLOW and the Doherty version of MODFLOW.)
*           sequence in which items are added to budget:
*           WEL DRN RCH EVT RIV STR GHB.
        data (idxlbl(j),j=1,7) /2,3,8,5,4,13,7/
      DATA ANAME(1,1),ANAME(2,1),ANAME(3,1),ANAME(4,1),ANAME(5,1),
     1   ANAME(6,1) /'    ',' Bu','dget',' zon','es a','rrau'/
c-----------------------------------------------------------------------
C2a-----READ version of BOUNDARY ARRAY (izonbd, to specify the domain of
c       the volumetric budget) ONE LAYER AT A TIME.
      nzones=0
      numneg=0
      if (iopzon.gt.0) then
        DO layer=1,NLAY
          KK=layer
cc        LOC=1+(K-1)*NCR
          CALL U2DINT(izonbd(1,1,kk),ANAME(1,1),NROW,NCOL,KK,
     1            inpost,IOUT)
          do irow=1,nrow
            do icol=1,ncol
              nzones = MAX0(nzones,izone)
              if (izonbd(icol,irow,layer).lt.0) then
                numneg=numneg+1
                izone=0
                izonbd(icol,irow,layer)=0
              else if (izonbd(icol,irow,layer).gt.mxzone) then
                numexc=numexc+1
                izone=mxzone
                izonbd(icol,irow,layer)=mxzone
              end if
              izone=izonbd(icol,irow,layer)
              nczone(izone)=nczone(izone)+1 !no. cells in each zone
              if (ibound(icol,irow,layer).gt.0)
     1          ncaczn(izone) = ncaczn(izone) + 1 !no. ACTIVE cells in zone
            end do
          end do
        end do
        if (numneg.gt.0) print '(2(1x,a,i5))','POST1RP: ',numneg,
     1        'negative values encountered in budget zone definition '//
     1        'array izonbd and set to zero.'
        write (ionetz,'(a,9(1x,a,1x,6hin-out,1x))')
     1    ' zon cells activ stp per itr      '//timlbl(idxtim)//
     1    '  tot in-out  xfr in-out', (budlbl(j),j=1,msum-1)
cc      write (ionetz,325) izone,nczone(izone),kstp,kper,kiter,timsim,
cc   1    totnet(idxvol),xfrnet(idxvol),
cc   1    (volnet(idxvol,j),j=1,msum-1)

      end if
C7------INITIALIZE VOLUMETRIC BUDGET ACCUMULATORS for zones TO ZERO (cf BAS1RP)
  590 DO I=1,20
        DO J=1,4
          do k=0,mxzone
              vbvlzn(j,i,k)=0.
          end do
        end do
      end do
      do i=0,mxzone
          do k=1,2
              xfrinp(k,i)=0.
              xfrout(k,i)=0.
          end do
          do j=0,mxzone
              do k=1,2
                  vbznfl(k,j,i)=0.
              end do
          end do
      end do
      return
      end
```

Subr. HYD1RP: read postprocessing directions for hydrographs

```
c-----------------------------------------------------------------------
      SUBROUTINE HYD1RP (STRT,ISTRT,IBOUND,mxstrm,nstrem,istrm,KPER,
     1  NCOL,NROW,NLAY,delx,dely,xx,yy,zz,idxtim,inphyd,iohyd,mxser,
     1  numser,idtype,layser,rowser,colser,idxrch,istper,
     1  iopvec,idaxis,islice,nstp,iopstp,initxy,idmain)

c Arguments
c idmain = 0 indicates a call to HYD1OT from modflo, where drawdown is
c            calculated from beginning and latest heads;
c        = 1 indicates a call from POSTMD97, where drawdown is read from
c            MODFLOW's standard output file.

      dimension iopstp(nstp),istrm(10,mxstrm),
     1          STRT(NCOL,NROW,NLAY),IBOUND(NCOL,NROW,NLAY)
* marios::c:\sam\modellb\postmod.for  Oct 5 91 modflow postprocessor
*
*****************************************************************
*                                                               *
*                            hyd1rp                             *
*                                                               *
*****************************************************************
c
*       Program MODHYD reads a standard MODFLOW output file
*       and writes head and drawdown values for individual cells
*       for each time step or stress period
*       The program will essentially extract head values for
*           plotting hydrographs.
*
*          --spp Oct 5 1991
*          The original PostMod program was modified to provide
*          the hydrograph output.
*                        modified -- cdb.
*          And modified in Jan '92. --sam
*-------------------------------------------------------------------
c
c---------- Allow at least 20 hydrographs of mxstp time values each:
        integer idtype(mxser),layser(mxser),rowser(mxser),colser(mxser),
     1      idxrch(mxser)
        character*1 chtype(9)
        character*7 descrp(9)
        character*40 descr2(9)
        character*4 timlbl(0:5)
c-----------------------------------------------------------------
      data nserty /9/ !no. series types
      data chtype /'H', 'C', 'D', 'T', 'B', 'S', 'Q','V', 'Z'/
      data descrp /'Head','HeadChg','Drawdwn','SatThck',
     1    'Baseflo','Strmflo','Cellflo','Cellvel','Stage'/
      data (descr2(j),j=1,9) /
     1    'Hydraulic heads', 'Change in heads', 'Drawdown',
     1    'Saturated thickness',
     1    'Baseflow =-(streambed leakage)',
     1    'Streamflow = channel outflow',
     1    'Cell flow rate Q=K*A*dh/dl [L^3/T]',
     1    'Specific flow rate q = Q/A [L/T]',
     1    'Stream stage (avg for reach) [L]'/

c           labels for time units corresponding to idxtim (chosen below):
      data timlbl /'time','secs','mins',' hrs','days',' yrs'/
ccccc inphyd = iunit(15)
c         Set up x,y,z-coordinates for cell-centered nodes:
      print '(1x,a,i5)','HYD1RP: numser=',numser
      if (initxy.eq.0) then
          call CELLOC (ncol,nrow,nlay,delx,dely,xx,yy,zz)
          initxy=1  !indicates that grid coord's have been initialized.
      end if
      if (kper.eq.1) then
        print '(3(1x,a,i4))','HYD1RP: mxstrm=',mxstrm,'nstrem=',nstrem,
     1                                  'ISTRT=',istrt
        if (nlay.eq.1) layer = 1

        print '(1x,a,(/,1x,i2,1x,a))', 'Time series type identifiers: ',
     1       (j,chtype(j)//'='//descrp(j)//' '//descr2(j),j=1,9)
        if (nlay.eq.1) then
            print *,'For each time series, enter row, column '//
     1          'and type 1-9 (FREE):'
        else
            print *,'For each series, enter layer, row, column '//
     1          'and type 1-9 (FREE):'
        end if
        write (iohyd,'(1x,a,i3,1x,a)') 'Extract ',numser,'hydrographs:'
        write (iohyd,'(1x,a)') ' idx  lay  row  col  rch series type'//
     1     ' ibnd starting head'
        do idx=1,numser
          print '(1x,a,i3,a)', 'Enter series',idx,': '
          if (nlay.eq.1) then
              read (inphyd,*,iostat=iostat,err=16)
     1            irowid, icolid, idser
          else
              read (inphyd,*,iostat=iostat,err=16)
     1            layer, irowid, icolid, idser
          end if
c             series types: Head, Change in head, Drawdown, Saturated Thickness,
c                 Baseflow (Streambed leakage), Streamflow (channel outflow),
c                 Flow from each cell.
          if (idser.lt.1.or.idser.gt.nserty) idser = 1
c             If starting heads haven't been saved, don't try to write
c             change in heads unless drawdown has been read from std output:
          if ((idser.eq.2.or.idser.eq.3) .and. istrt.eq.0) idser = 1
          idtype(idx) = idser
16        if (iostat.ne.0 .or. idtype(idx).eq.0 .or.
     1        irowid.lt.1 .or. irowid.gt.nrow .or.
     1        icolid.lt.1 .or. icolid.gt.ncol) then
              print '(3(1x,a,i3))',
     1            'Error on input for time series',idx,
     1            '; row or column index may be out of bounds:'
              print *,'row from 1 to ',nrow,' col from 1 to ',ncol
              STOP
          end if
          if (ibound(icolid,irowid,layer).eq.0 .and. idmain.eq.0)
     1        print '(4(1x,a,i4))','Cell at layer',layer,'row',irowid,
     1            'column',icolid,' is not active: ibound=',
     1            ibound(icolid,irowid,layer)
          idxrch(idx) = 0
          do j=1,nstrem
              if (istrm(2,j).eq.irowid .and.
     1            istrm(3,j).eq.icolid) then
cc                    1st strm reach located at cell (row,col):
                  idxrch(idx) = j
                  go to 20
              end if
          end do
20        layser(idx) = layer
          rowser(idx) = irowid
```

```
              colser(idx) = icolid
              if (istrt.ne.0)
     1          write (iohyd,'(5i5,2(1x,a),i5,g12.4,1x,a)')
     1            idx,layser(idx),rowser(idx),colser(idx),idxrch(idx),
     1            chtype(idtype(idx)),descrp(idtype(idx)),
     1            ibound(icolid,irowid,layer), strt(icolid,irowid,layer)
            end do
120         format (1x,a,t25,20(2h (,a,1h:,i3,1h,,i3,1h)))
            if (numser.gt.0) then
                if (nlay.gt.1) write (iohyd,'(1x,a,t25,20(4x,i3,5x))')
     1              'layer:', (layser(idx),idx=1,numser)
                write (iohyd,120) 'step  per        '//timlbl(idxtim),
     1            (chtype(idtype(idx)),rowser(idx),colser(idx),idx=1,numser)
                kstp=0
                timsim=0.
cc              write (iohyd,'(2i5,g12.4,2x,20g12.4)')
cc   1            kstp,kper,timsim,(valser(idx),idx=1,numser)
            end if
c               options to write solution arrays as "x,y,f(x,y)" or "x,y,z,f(x,y,z)":
c istper = 0: [N]one;
c         = 1: [O]ne: specify only one (time step, stress period)
c         = 2: [I]ndividual: specify all time steps individually (iopstp, below);
c         = 3: [A]ll: write results for all time steps.
c               idaxis = 1, 2, or 3: option to write a LAYER, ROW, or COLUMN,
c                        respectively, of the solution array for each time step and
c                        stress period specified by iopstp(j) (below).
c               islice: index to the plane corresponding to layer, row, or column
c                        (depending on idaxis, above) to be written.
c iopvec: option (for idtype = 7 or 8) to write flow rates either as
c         = 0: one record/node, or
c         > 0: vector originating at the center of node (j,i,k) for
c                      column j, row i, and layer k with location (xc,yr,zl),
c                      where xc=xx(j) based on column widths according to
c                            xx(j) = sum [delx(m),m=1 to j-1] + delx(j)/2;
c                      yr=yy(i) based on row widths dely(i);
c                      zl=head(j,i,k) for top layer k=1, and
c                      zl=top(j,i,k) for lower layers k > 1
c                      given by cell widths as follows:
c                  =1: xc=xx(j), yr=yy(i), and zl=ztop, where
c                          ztop = head(j,i,k) for layer k=1, and
c                               = top(j,i,k) for layer k > 1 (lower layers);
c                  =2:     xc=j-0.5, yr=(nrow-i+1)-0.5, zl=(nlay-k+1)-0.5
c                      The vector ends at location (xc+xq,yr+yq,zl+zq), where
c                          xq=ux*delx(j)/2, yq=uy*dely(i)/2, and zq=uz*dz/w;
c                      (ux,uy,uz) are unit vector components for either the
c                      flow rate Q or darcy velocity q, depending on idtype.
c itype   = 0: No results to be written;
c         = 1: head
c         = 2: head and change in head
c         = 3: drawdown
c         = 4: saturated thickness
c         = 5: baseflow (streambed leakage)
c         = 6: streamflow
c         = 7: flow rate through cells, Q=K*A*dh/dl
c         = 8: Darcy velocity (specific flow rate) through cells,q=Q/A
c         = 9: stream stage (avg for reach)
             read (inphyd,*) istper, idaxis, islice, iopvec
             print '(5(1x,a,i3))','istper=',istper,
     1             'idaxis=',idaxis, 'islice=',islice,'iopvec=',iopvec
             if (idaxis.lt.1.or.idaxis.gt.3) idaxis = 1 !default: layer
             islice =MAX0(1,islice)
             if (idaxis.eq.1) then
                 islice = MIN0(islice,nlay)
             else if (idaxis.eq.2) then
                 islice = MIN0(islice,nrow)
             else
                 islice = MIN0(islice,ncol)
             end if
c                initialize output control vector iopstp in first stress period:
             do j=1,nstp
                 iopstp(j)=0
            end do
        end if
c           read options for each time step of this stress period to write
c           results in Surfer format:
        if (istper.eq.1) then
            if (kper.eq.1) read (inphyd,*) itmstp,iperio,itype
            if (kper.eq.iperio) iopstp(itmstp) = itype
        else if (istper.eq.2) then
            read (inphyd,*) (iopstp(j),j=1,nstp)
        else if (istper.gt.3) then
            do j=1,nstp
                iopstp(j) = 1
            end do
        end if
        return
        end
```

Subr. HYD1OT: write hydrographs to file.

```
c------------------------------------------------------------
        SUBROUTINE HYD1OT (HNEW,BUFF,STRT,ISTRT,IBOUND,idry,mxstrm,nstrem,
     1  strm,top,bot,delx,dely,xx,yy,zz,ltyp23,ktlay,cr,cc,cv,KSTP,KPER,
     1  NCOL,NROW,NLAY,timsim,delt,iohyd,mxser,numser,idtype,layser,
     1  rowser,colser,idxrch,valser,iopvec,idaxis,islice,nstp,
     1  iopstp,iopost,idmain)

c           CALL HYD1OT (X(LCHNEW),X(LCBUFF),X(LCSTRT),ISTRT,X(LCIBOU),
c    1        x(LCidry),mxstrm,nstrem,x(LCSTRM),x(LCTOP),x(LCBOT),
c    1        X(LCDELR),X(LCDELC),x(LCXLOC),x(LCYLOC),x(LCZLOC),
c    1        x(LC23),x(LCKT),x(LCCR),x(LCCC),x(LCCV),KKSTP,KKPER,
c    1        NCOL,NROW,NLAY,timsim(idxtim),iohyd,mxser,numser,x(ICTser),
c    1        x(ICLser),x(ICRser),x(ICCser),x(ICXser),x(ICVser),
c    1        iopvec,idaxis,islice,nstp,iopstp,iopost,idmain)

        double precision hnew
        dimension iopstp(nstp),HNEW(NCOL,NROW,NLAY),STRT(NCOL,NROW,NLAY),
     1      BUFF(NCOL,NROW,NLAY),IBOUND(NCOL,NROW,NLAY),
     1      idry(NCOL,NROW,NLAY),strm(30,mxstrm),
     1      CR(NCOL,NROW,NLAY), CC(NCOL,NROW,NLAY),CV(NCOL,NROW,NLAY),
     1      top(ncol,nrow,nlay), BOT(NCOL,NROW,NLAY),
     1      delx(ncol),dely(nrow), xx(ncol),yy(nrow),zz(nlay),
     1      ltyp23(nlay),ktlay(nlay)
        character*13 chvser(20), chval
c
        COMMON /FLWCOM/LAYCON(80)
c
c Arguments
c idmain = 0 indicates a call to HYD1OT from modflo, where drawdown is
c            calculated from beginning and latest heads;
c        = 1 indicates a call from POSTMD97, where drawdown is read from
c            MODFLOW's standard output file.
c
c---------- Allow at least 20 hydrographs of mxstp time values each:
        integer idtype(mxser),layser(mxser),rowser(mxser),colser(mxser),
     1          idxrch(mxser)
        dimension valser(mxser)
        dimension u(3)  !normalized (fractional) x-,y-,z-components of flow rate
c           (florat) [L^3/T] or Darcy velocity [L/T] returned by DARCY
        dimension ds(4) !x-,y-,z-components and magnitude of displacement vector [L]
c           (returned by DARCY if iopvel>0, i.e. Darcy velocity is calculated.
c-----------------------------------------------------------------------
        character optns*3,opaxis*1,outfil*30
        character*26 axlbl(3)
        character*1 chtype(8)
        character*7 descrp(8)
        character*1 outopt, chform(4)  !file identifier
        character*1 chdry(0:1)
        data chdry /' ','D'/
```

```
      data optns /'LRC'/
      data (axlbl(j),j=1,3) /'   x(col)       y(row)     ',  !write layer
     1                       '   x(col)     z(layer)     ',  !write row
     1                       '   y(row)     z(layer)     '/  !write column
      data chform /'H','H','Q','V'/
      data outopt /'S'/
      data chtype /'H', 'C', 'D', 'T', 'B', 'S', 'Q','V'/
      data descrp /'Head','HeadChg','Drawdwn','SatThck',
     1    'Baseflo','Strmflo','Cellflo','Cellvel'/
      data initlz /0/
c-----------------------------------------------------------------------
C2-----initialize arrays ltype23 and ktlay for subr DARCY (flow rate calculations)
      if (initlz.eq.0) then
        initlz=1
        kt = 0
        DO K=1,NLAY
          if (k.lt.nlay) then
              IF(LAYCON(K).EQ.3 .OR. LAYCON(K).EQ.2) then
                  ltyp23(k) = 1
                  KT=KT+1
              else
                  ltyp23(k) = 0
              end if
              ktlay(k) = kt
          end if
        end do
      end if
c-----------------------------------------------------------------------
      do idx=1,numser                                !write time series
          layer = layser(idx)
          ir = rowser(idx)                           !row index
          ic = colser(idx)                           !column index
          itype = idtype(idx)                        !hydrograph type
          valser(idx) = 0.
          if (ibound(ic,ir,layer).ne.0.or.idry(ic,ir,layer).ne.0) then
            L = idxrch(idx)                           !index to reach
            call GETVAL(ic,ir,layer,HNEW,STRT,BUFF,istrt,IBOUND,CR,CC,
     1        CV,TOP,BOT,DELX,DELY,STRM,NCOL,NROW,NLAY,L,mxstrm,iopvel,
     1        ltyp23,ktlay,florat,u,ds,delt,itype,idmain,valser(idx),
     1        chvser(idx))
          end if
      end do
      if (numser.gt.0) write (iohyd,'(2i5,g13.5,20a13)')
     1        kstp,kper,timsim,(chvser(idx),idx=1,numser)
c---------from postmod:-------------------------------------------------
c
      opaxis = optns(idaxis:idaxis)
      itype=iopstp(kstp)
      if (itype.ge.1.and.itype.le.6) then
          do layer=1,nlay
            do ir=1,nrow
              do ic=1,ncol
                if (ibound(ic,ir,layer).ne.0) then
                  call GETVAL(ic,ir,layer,HNEW,STRT,BUFF,istrt,IBOUND,
     1              CR,CC,CV,TOP,BOT,DELX,DELY,STRM,NCOL,NROW,NLAY,L,
     1              mxstrm,iopvel,ltyp23,ktlay,florat,u,ds,delt,itype,
     1              idmain,val,chval)
                  buff(ic,ir,layer) = val
                end if
              end do
            end do
          end do
cc        print '(5(1x,a,i3))','HYD1OT: kstp=',kstp,'kper=',kper,
cc   1      'iopvec=',iopvec,'idaxis=',idaxis,'islice=',islice,
cc   1      'nstp=',nstp,'nlay=',nlay,'nrow=',nrow,'ncol=',ncol
          print '(1x,a)','opaxis = '// opaxis
          write (outfil,'(a1,2i3,a2,i3)')
     1        chtype(itype),kstp,kper,opaxis//'.DAT'
          do j=2,8
              if (outfil(j:j).eq.' ') outfil(j:j) = '0'
          end do
```

```
ccc        print '(5(1x,a,i5))','Write heads for step',kstp,
ccc   1         'period',kper,'to device',iopost,'file '// outfil
ccccccccc  open (unit=iopost,file=outfil,status='unknown')
           write (iopost,'(2(1x,a,i3),a)')
     1          'step',kstp,'period',kper,': file '// outfil
           if (idaxis.eq.1) then     !write (x,y) in layer planes
ccccccccc    do layer = 1,nlay
                layer = islice
                if (itype.eq.2) then !for change in heads, write heads also
                  write (iopost,'(1x,a,i3)') axlbl(idaxis)//'    '//
     1              descrp(itype-1)//'       '// descrp(itype) //
     1              '    layer',layer
                else
                  write (iopost,'(1x,a,i3)') axlbl(idaxis)//
     1              descrp(itype) //'      layer',layer
                end if
                do ir = 1,nrow
                   do ic=1,ncol
                     ibnd = ibound(ic,ir,layer)
                     if (ibnd.ne.0) then
                       if (itype.eq.2) then
                         write (iopost,203) xx(ic), yy(ir),
     1                      hnew(ic,ir,layer),buff(ic,ir,layer),
     1                      ir,ic,chdry(idry(ic,ir,layer))
203                      format (1x,2g12.4,2g15.7,2i5,1x,a)
                       else
                         write (iopost,202) xx(ic), yy(ir),
     1                      buff(ic,ir,layer),
     1                      ir,ic,chdry(idry(ic,ir,layer))
202                      format (1x,2g12.4,g15.7,2i5,1x,a)
                       end if
                     end if
                   end do            ! column ic
                end do               ! row ir
cccccccccc   end do                  ! layer
           else if (idaxis.eq.2) then  !write (x,z) in row planes
cccccccccc   do ir = 1,nrow
                ir = islice
                if (nlay.eq.1) then
                    write (iopost,'(1x,a,i3)') '    x(col)    '//
     1                descrp(itype) //'      row',ir
                else
                    write (iopost,'(1x,a,i3)') axlbl(idaxis)//
     1                descrp(itype) //'      row',ir
                end if
                do layer = 1,nlay
                  do ic=1,ncol
                    ibnd = ibound(ic,ir,layer)
                    if (ibnd.gt.0) then
                        if (nlay.eq.1) then
                            write(iopost,201) xx(ic), buff(ic,ir,layer)
201                         format (1x,2g13.5)
                        else
                            write(iopost,202) xx(ic), zz(layer),
     1                          buff(ic,ir,layer),ic,layer
                        end if
                    end if
                  end do             ! column ic
                end do               ! layer
cccccccccc   end do                  ! row ir
           else   !(idaxis.eq.3)       !write (y,z) in column planes
cccccccccc   do ic=1,ncol
                ic = islice
                if (nlay.eq.1) then
                    write (iopost,'(1x,a,i3)') '    y(row)    '//
     1                descrp(itype) //'      col',ic
                else
                    write (iopost,'(1x,a,i3)') axlbl(idaxis)//
     1                descrp(itype) //'      col',ic
                end if
                do layer = 1,nlay
                do ir = 1,nrow
                  ibnd = ibound(ic,ir,layer)
                  if (ibnd.gt.0) then
                      if (nlay.eq.1) then
                          write(iopost,201) yy(ir), buff(ic,ir,layer)
                      else
                          write(iopost,202) yy(ir), zz(layer),
     1                        buff(ic,ir,layer),ir,layer
                      end if
                  end if
                end do           ! row ir
              end do             ! layer
cccccccccc  end do               ! column ic
          end if
ccccccc   close (unit=iopost)
      else if (itype.eq.7.or.itype.eq.8) then
          write (outfil,'(a1,2i3,a2,i3)')
     1        chtype(itype),kstp,kper,opaxis//'.DAT'
          do j=2,8
              if (outfil(j:j).eq.' ') outfil(j:j) = '0'
          end do
ccccccccc open (unit=iopost,file=outfil,status='unknown')
          write (iopost,'(2(1x,a,i3),a)')
     1        'step',kstp,'period',kper,': file '// outfil
          if (nlay.eq.1) then
              write (iopost,'(1x,a,i3)') '   x(col)    y(row) '//
     1           '      '//chtype(itype)//'     '//
     1           '        qx         qy h(ic,ir,il)  col  row'
          else
              write (iopost,'(1x,a,i3)') '   x(col)    y(row) '//
     1           '   z(lay)      '//chtype(itype)//'     '//
     1           '        qx         qy         qz       h(ic,ir,il)'//
     1           '  col  row  lay'
          end if
c           iopvel=0 for itype=7, flow rates; iopvel=1 for itype=8, velocity
          iopvel=itype-7
          call SBCF2B(HNEW,IBOUND,CR,CC,CV,TOP,BOT,DELX,DELY,xx,yy,
     1      ltyp23,ktlay,ncol,nrow,nlay,iopvel,iopvec,idaxis,islice,
     1      iopost)
      end if
      return
      end
```

Subr. SBCF2B: evaluate flow components (a version of Modflow subr. SBCF1B)

```
c-----------------------------------------------------------------------------
      SUBROUTINE SBCF2B(HNEW,IBOUND,CR,CC,CV,TOP,BOT,DELX,DELY,xx,yy,
     1  ltyp23,ktlay,NCOL,NROW,NLAY,iopvel,iopvec,idaxis,islice,iopost)
C
C-----VERSION 1548 12MAY1997 spp SBCF2B
c     NOTE: This is a postprocessing version of SBCF1B. Changes:
c     1)  Buffer array (buff) is not used;
c     2)  Flow components are computed in one pass of the heads array,
c         and written in "x,y,f(x,y)" or "x,y,z,f(x,y,z)" format.
c     3) Options:
c        iopvel=0: write flow rate Q=K*A*dh/dl;
c              >0: write specific flow rate (Darcy velocity, v=K*dh/dl);
c                  also calculate displacement vector, ds = v*delt:
c                  (ds(j),j=1,4) give x-,y-,z-components and magnitude
c                  |ds| = |v|*delt, respectively.
c     4) iopvec>0: write flow rate components in vector form for plotting.
C
C     ******************************************************************
C     COMPUTE FLOW ACROSS EACH CELL WALL
C     ******************************************************************
C
C     SPECIFICATIONS:
C     ------------------------------------------------------------------
```

```
      DOUBLE PRECISION HNEW
C
      DIMENSION HNEW(NCOL,NROW,NLAY), IBOUND(NCOL,NROW,NLAY),
     1     CR(NCOL,NROW,NLAY), CC(NCOL,NROW,NLAY),
     2     CV(NCOL,NROW,NLAY), TOP(NCOL,NROW,NLAY),BOT(ncol,nrow,nlay),
     3     DELX(ncol),DELY(nrow),xx(ncol),yy(nrow),
     4     ltyp23(nlay),ktlay(nlay),u(3),ds(4)

      COMMON /FLWCOM/LAYCON(80)
C     -----------------------------------------------------------------
C
C2-----FOR EACH CELL CALCULATE FLOW THRU RIGHT FACE & STORE IN BUFFER
      kt = 0
      DO K=1,NLAY
          if (k.lt.nlay) then
              IF(LAYCON(K).EQ.3 .OR. LAYCON(K).EQ.2) then
                  ltyp23(k) = 1
                  KT=KT+1
              else
                  ltyp23(k) = 0
              end if
              ktlay(k) = kt
          end if
      end do
c----------------------------------------------------------------------
      if (idaxis.eq.1) then     !write (x,y) in layer planes
ccccc   do k=1,nlay
          k = islice
          DO I=1,NROW
              DO J=1,NCOL
                  IF(IBOUND(J,I,K).gt.0) then
                      call DARCY (j,i,k,HNEW,IBOUND,CR,CC,CV,TOP,BOT,
     1                    DELX,DELY,NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,
     1                    florat,u,ds,delt)
                      call PLTFLO (j,i,k,HNEW,IBOUND,TOP,BOT,DELX,DELY,
     1                    xx,yy,NCOL,NROW,NLAY,iopvec,iopost,florat,u)
                  end if
              end do    !j=1,ncol
          end do    !i=1,nrow
ccccc   end do    !k=1,nlay
      else if (idaxis.eq.2) then  !write (x,z) in row planes
cccccccccc  do ir = 1,nrow
          i = islice
          DO k=1,nlay
              DO J=1,NCOL
                  IF(IBOUND(J,I,K).gt.0) then
                      call DARCY (j,i,k,HNEW,IBOUND,CR,CC,CV,TOP,BOT,
     1                    DELX,DELY,NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,
     1                    florat,u,ds,delt)
                      call PLTFLO (j,i,k,HNEW,IBOUND,TOP,BOT,DELX,DELY,
     1                    xx,yy,NCOL,NROW,NLAY,iopvec,iopost,florat,u)
                  end if
              end do    !j=1,ncol
          end do    !k=1,nlay
      else   !(idaxis.eq.3)      !write (y,z) in column planes
cccccccccc  do ic=1,ncol
          j = islice
          DO k=1,nlay
              DO i=1,nrow
                  IF(IBOUND(J,I,K).gt.0) then
                      call DARCY (j,i,k,HNEW,IBOUND,CR,CC,CV,TOP,BOT,
     1                    DELX,DELY,NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,
     1                    florat,u,ds,delt)
                      call PLTFLO (j,i,k,HNEW,IBOUND,TOP,BOT,DELX,DELY,
     1                    xx,yy,NCOL,NROW,NLAY,iopvec,iopost,florat,u)
                  end if
              end do    !i=1,nrow
          end do    !k=1,nlay
      end if
C
C3-----RECORD CONTENTS OF BUFFER
```

```
c      CALL UBUDSV(KSTP,KPER,TEXT(1),IBCFCB,BUFF,NCOL,NROW,NLAY,IOUT)
c      CALL UBUDSV(KSTP,KPER,TEXT(5),IBCFCB,BUFF,NCOL,NROW,NLAY,IOUT)
c      CALL UBUDSV(KSTP,KPER,TEXT(9),IBCFCB,BUFF,NCOL,NROW,NLAY,IOUT)
c
       RETURN
       END
```

subr. GETVAL

```
c-----------------------------------------------------------------------
       subroutine GETVAL(ic,ir,layer,HNEW,STRT,BUFF,istrt,IBOUND,
     1  CR,CC,CV,TOP,BOT,DELX,DELY,STRM,NCOL,NROW,NLAY,LREACH,mxstrm,
     1  iopvel,ltyp23,ktlay,florat,u,ds,delt,itype,idmain,val,chval)
c-----------------------------------------------------------------------
c usage in HYD1OT:
cc           call GETVAL(ic,ir,layer,HNEW,STRT,BUFF,istrt,IBOUND,CR,
cc    1            CC,CV,TOP,BOT,DELX,DELY,STRM,NCOL,NROW,NLAY,L,mxstrm,
cc    1            iopvel,ltyp23,ktlay,florat,u,ds,delt,itype,idmain,valser(idx))
c    j=column,i=row,k=layer; lreach = index to order in which reach was read.

       DOUBLE PRECISION HNEW
       character*(*) chval
c
       DIMENSION HNEW(NCOL,NROW,NLAY), STRT(NCOL,NROW,NLAY),
     1     BUFF(NCOL,NROW,NLAY),IBOUND(NCOL,NROW,NLAY),
     1     CR(NCOL,NROW,NLAY), CC(NCOL,NROW,NLAY),
     2     CV(NCOL,NROW,NLAY), TOP(NCOL,NROW,NLAY),BOT(ncol,nrow,nlay),
     3     DELX(ncol),DELY(nrow),STRM(30,mxstrm),
     1     ltyp23(nlay),ktlay(nlay), u(3), ds(4)
       h=hnew(ic,ir,layer)
       h0=strt(ic,ir,layer)
cc     print '(4(1x,a,i2),2(1x,a,f8.2))','GETVAL: ic=',ic,'ir=',ir,
cc    1     'layer=',layer,'itype=',itype,'h=',h,'h0=',h0
       if (itype.eq.1) then
           val = h                                 !head
       else if (itype.eq.2.or.itype.eq.3) then     !head change or drawdown
           if (idmain.eq.1) then
               val = buff(ic,ir,layer)             !drawdown for postmd97
           else if (istrt.ne.0) then
               val = strt(ic,ir,layer) - h         !drawdown for modflo
           end if
           if (itype.eq.2) val = -val               !head change (-drawdown)
       else if (itype.eq.4.and.idmain.eq.0) then
           val = h - bot(ic,ir,layer)               !sat. thickness (modflo)
       else if (itype.eq.5.and.Lreach.gt.0) then
           val = strm(11,Lreach)  !streambed leakage  (baseflow)
       else if (itype.eq.6.and.Lreach.gt.0) then
           val = strm(9,Lreach)   !streamflow (channel outflow from reach)
       else if (itype.eq.9.and.Lreach.gt.0) then
           val = strm(2,Lreach)   !stream stage (avg for reach)
       else if (itype.eq.7.and.idmain.eq.0.and.
     1          ibound(ic,ir,layer).gt.0) then
           iopvel=0          !flow rate [L^3/T] (modflo)
           call DARCY (ic,ir,layer,HNEW,IBOUND,CR,CC,CV,TOP,BOT,DELX,
     1         DELY,NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,florat,u,ds,delt)
           val = florat
       else if (itype.eq.8.and.idmain.eq.0.and.
     1          ibound(ic,ir,layer).gt.0) then
           iopvel=1          !velocity [L/T] (specific flow rate; modflo)
           call DARCY (ic,ir,layer,HNEW,IBOUND,CR,CC,CV,TOP,BOT,DELX,
     1         DELY,NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,florat,u,ds,delt)
           val = florat
       end if
       if (6.le.itype.and.itype.le.8) then
           write (chval,'(e13.5)') val
       else
           write (chval,'(f13.2)') val
       end if
```

```
      return
      end
```

Subr. DARCY: evaluate Darcy's Law for intercell flow.

```
c-----------------------------------------------------------------------------
      subroutine DARCY (j,i,k,HNEW,IBOUND,CR,CC,CV,TOP,BOT,DELX,DELY,
     1    NCOL,NROW,NLAY,iopvel,ltyp23,ktlay,florat,u,ds,delt)
c-----------------------------------------------------------------------------
c   j=column,i=row,k=layer:

      DOUBLE PRECISION HNEW,HD
c
      DIMENSION HNEW(NCOL,NROW,NLAY), IBOUND(NCOL,NROW,NLAY),
     1     CR(NCOL,NROW,NLAY), CC(NCOL,NROW,NLAY),
     2     CV(NCOL,NROW,NLAY), TOP(NCOL,NROW,NLAY),BOT(ncol,nrow,nlay),
     3     DELX(ncol),DELY(nrow),ltyp23(nlay),ktlay(nlay), u(3), ds(4)

      double precision dxdz,dydz,dxdy,q,qr,qf,ql
      qr=0.
      qf=0.
      ql=0.
      u(1)=0.
      u(2)=0.
      u(3)=0.
      if (k.eq.1) then       !sat. thickness
          ztop = hnew(j,i,k)
      else
          ztop = top(j,i,k)  !meaningful only if layer type (laycon) is 2 or 3
      end if
      dz = ztop - bot(j,i,k) !valid for top layer if unconfined (laycon=1)
      dxdz = delx(j)*dz
      dydz = dely(i)*dz
      dxdy = delx(j)*dely(i)
      if (j.lt.ncol) then        !qr=flow through right face
          if (IBOUND(J+1,I,K).gt.0) then
              qr=(HNEW(J,I,K)-HNEW(J+1,I,K))*cr(j,i,k)     !flow rate [L^3/T]
              if (iopvel.gt.0) qr = qr/dydz                !velocity [L/T]
          end if
      end if
      if (i.lt.nrow) then        !qf=flow through front face
          if (IBOUND(J,I+1,K).gt.0) then
              qf=(HNEW(J,I,K)-HNEW(J,I+1,K))*cc(j,i,k)     !flow rate [L^3/T]
              if (iopvel.gt.0) qf = qf/dxdz                !velocity [L/T]
          end if
      end if
      if (k.lt.nlay) then        !ql=flow through lower face
          if (IBOUND(J,I,K+1).gt.0) then
              HD=HNEW(J,I,K+1)
              IF(ltyp23(k).eq.0) then
                  hd=hnew(j,i,k+1)
              else
                  hd=AMAX1(hnew(j,i,k+1),top(j,i,ktlay(k)+1))
              end if
              ql = (HNEW(J,I,K)-HD)*cv(j,i,k)              !flow rate [L^3/T]
              if (iopvel.gt.0) ql = ql/dxdy                !velocity [L/T]
          end if
      end if
      q = DSQRT(qr*qr + qf*qf + ql*ql)    !magnitude of flow rate or velocity
      florat=q              !return single precision version
      if (1.+q.gt.1.) then
          u(1) =  qr/q      !fraction in x direction (increasing column index)
          u(2) = -qf/q      !fraction in y direction (decreasing row index)
          u(3) = -ql/q      !fraction in z direction (decreasing layer index)
      end if
c       displacement vector [L] associated with Darcy velocity [L/T]:
      if (iopvel.gt.0) then
        ds(1) =  qr*delt    !x component (increasing column index)
        ds(2) = -qf*delt   !y component (decreasing row index)
        ds(3) = -ql*delt   !z component (decreasing layer index)
        ds(4) =   q*delt   !magnitude
      end if
      return
      end
```

Subr. PLTFLO

```
c-----------------------------------------------------------------------------
      subroutine PLTFLO (j,i,k,HNEW,IBOUND,TOP,BOT,DELX,DELY,
     1    xx,yy,NCOL,NROW,NLAY,iopvec,iopost,florat,u)
c-----------------------------------------------------------------------------
c   j=column,i=row,k=layer:

      DOUBLE PRECISION HNEW
c
      DIMENSION HNEW(NCOL,NROW,NLAY), IBOUND(NCOL,NROW,NLAY),
     2     TOP(NCOL,NROW,NLAY),BOT(ncol,nrow,nlay),
     3     DELX(ncol),DELY(nrow),xx(ncol),yy(nrow),u(3)

      if (k.eq.1) then       !sat. thickness
          ztop = hnew(j,i,k)
      else
          ztop = top(j,i,k)  !meaningful only if layer type (laycon) is 2 or 3
      end if
      dz = ztop - bot(j,i,k) !valid for top layer if unconfined (laycon=1)
      if (iopvec.eq.1) then
          xq = xx(j) + u(1)*delx(j)/2.
          yq = yy(i) + u(2)*dely(i)/2.
          zq = ztop + u(3)*dz/2.
      else if (iopvec.eq.2) then
          xc = j - 0.5
          yr = (nrow-i+1) - 0.5
          zl = (nlay-k+1) - 0.5
      end if
      if (nlay.eq.1) then
          if (iopvec.eq.0) then
              write (iopost,'(3g12.4,2f10.5,g12.4,2i5)')
     1            xx(j),yy(i),florat,u(1),u(2),hnew(j,i,k),j,i
          else if (iopvec.eq.1) then
              write (iopost,'(/,3g12.4,2f10.5,g12.4,2i5)')
     1            xx(j),yy(i),florat,u(1),u(2),hnew(j,i,k),j,i
              write (iopost,'(2g12.4)') xq,yq
          else if (iopvec.eq.2) then
              write (iopost,'(/,2f10.5,2g12.4,2i5)')
     1            xc,yr,florat,hnew(j,i,k),j,i
              write (iopost,'(2f10.5)') xc+u(1),yr+u(2)
          end if
      else                                 !multi-layer case
          if (iopvec.eq.0) then
              write (iopost,'(4g12.4,3f10.5,g12.4,3i5)') xx(j),yy(i),
     1            ztop,florat,u(1),u(2),u(3),hnew(j,i,k),j,i,k
          else if (iopvec.eq.1) then
              write (iopost,'(/,4g12.4,3f10.5,g12.4,3i5)') xx(j),yy(i),
     1            ztop,florat,u(1),u(2),u(3),hnew(j,i,k),j,i,k
              write (iopost,'(3g12.4)') xq,yq,zq
          else if (iopvec.eq.2) then
              write (iopost,'(/,3f10.5,2g12.4,3i5)')
     1            xc,yr,zl,florat,hnew(j,i,k),j,i,k
              write (iopost,'(3f10.5)') xc+u(1),yr+u(2),zl+u(3)
          end if
      end if
      return
      end
```

Subr. CELLOC

```
c-------------------------------------------------------------------
      subroutine CELLOC (ncol,nrow,nlay,delx,dely,xx,yy,zz)

        integer ncol,nrow,nlay
        real delx(ncol),dely(nrow),xx(ncol),yy(nrow),zz(nlay)
        real dhalf,dprev
*
        character ans*1
*****   real trpy(50)
        integer j
*
***121  write (*,*)' Enter coordinates for node at (col 1, row 1): '
***     read (*,*,err=121) xx(1),yy(1)
        ans='N'
ccccc   call PROMPT ('Use unit cell dimensions (row,col widths)?',ans)
        if (ans.eq.'Y') then
            do j=1,ncol
                delx(j)=1.
            end do
            do i=1,nrow
                dely(i)=1.
            end do
        end if
c-------    x-coordinates:
        xmin = 0.
        dhalf = delx(1)/2.
        xx(1) = xmin + dhalf
        do j=2,ncol
            dprev = dhalf
            dhalf = delx(j)/2.
            xx(j) = xx(j-1) + dprev + dhalf
        end do
        xmax = xx(ncol) + dhalf
c-------    y-coordinates:
        ymin = 0.
        dhalf = dely(nrow)/2.
        yy(nrow) = ymin + dhalf
        do j=nrow-1,1,-1
            dprev = dhalf
            dhalf = dely(j)/2.
            yy(j) = yy(j+1) + (dprev + dhalf)
        end do
        ymax = yy(1) + dhalf
c-------    z-coordinates:
        if (nlay.gt.1) then
            zmin = 0.
            dhalf = 1./2.
            zz(nlay) = zmin + dhalf
            do j=nlay-1,1,-1
                dprev = dhalf
                dhalf = 1./2.
                zz(j) = zz(j+1) + (dprev + dhalf)
            end do
            zmax = zz(1) + dhalf
        end if
c-------
        print '(2(1x,a,f10.3))','x-axis range: xmin=',xmin,
     1      ' xmax=',xmax
        print '(2(1x,a,f10.3))','y-axis range: ymin=',ymin,
     1      ' ymax=',ymax
        if (nlay.gt.1) print '(2(1x,a,f10.3))',
     1      'z-axis range: zmin=',zmin,' zmax=',zmax
        return
        end
```

Subr. POST1OT: write solution arrays to output file.

```
c---------------------------------------------------------------------
      SUBROUTINE POST1OT (MSUM,VBVL,volnet,KSTP,KPER,kiter,delt,
     1    timsim,totinp,totout,diff,pdiff,idxvol,iobud,ionet)
C-----VERSION 1522 12MAY1997 POST1OT
C*********************************************************************
C     OUTPUT TIME, VOLUMETRIC BUDGET, HEAD, AND DRAWDOWN
C     ****************************************************************
C
C        SPECIFICATIONS:
C     ----------------------------------------------------------------
      dimension totinp(2),totout(2),diff(2),pdiff(2)
      DIMENSION VBVL(4,20), volnet(2,20)
c         volnet is now calculated in BAS1OT for L=1,msum as:
ccc       volnet(1,L) = vbvl(1,L) - vbvl(2,L)  !net volume (in-out)
ccc       volnet(2,L) = vbvl(3,L) - vbvl(4,L)  !net rate (in-out)
C     ----------------------------------------------------------------
      if (idxvol.eq.0) RETURN
      msum1 = msum - 1
      idxinp = idxvol
      if (idxvol.eq.2) idxinp=3
      idxout = idxinp+1
c         INPUT and OUTPUT components budget file:
      write (iobud,330) kstp,kper,kiter,delt,timsim,
     1    (vbvl(idxinp,j),j=1,msum1),totinp(idxvol),
     1    (vbvl(idxout,j),j=1,msum1),totout(idxvol),
     1    diff(idxvol),pdiff(idxvol)
330   format(3i4,g15.7,23g12.4)
c         NET budget file:
      write (ionet,325) kstp,kper,kiter,delt,timsim,
     1    pdiff(idxvol),diff(idxvol),
     1    (volnet(idxvol,j),j=1,msum1)
325   format (3i4,g15.7,g12.4,f8.2,10e12.4)
      return
      end
```

Subr. POST1OZ: summarize budgets for zones

```
c---------------------------------------------------------------------
      SUBROUTINE POST1OZ (MSUM,mxzone,VBVLZN,xfrinp,xfrout,
     1 totizn,totozn,nczone,ncaczn,KSTP,KPER,kiter,timsim,idxvol,
     1 ionetz)
C-----VERSION 1522 12MAY1987 BAS1OZ
C*********************************************************************
C     OUTPUT TIME, VOLUMETRIC BUDGET, HEAD, AND DRAWDOWN
C     ****************************************************************
C
C        SPECIFICATIONS:
C     ----------------------------------------------------------------
      DIMENSION VBVLZN(4,20,0:mxzone),
     1    xfrinp(2,0:mxzone), xfrout(2,0:mxzone),
     1    totizn(2,0:mxzone), totozn(2,0:mxzone),
     1    nczone(0:mxzone), ncaczn(0:mxzone)
      dimension volnet(2,9), xfrnet(2), totnet(2)
C     ----------------------------------------------------------------
      if (idxvol.eq.0) RETURN
      msum1 = msum - 1
      idxinp = idxvol
      if (idxvol.eq.2) idxinp=3
      idxout = idxinp+1
      do izone=0,mxzone
       if (nczone(izone).gt.0) then
        DO L=1,MSUM
          volnet(1,L) = vbvlzn(1,L,izone) - vbvlzn(2,L,izone)  !net volume (in-out)
          volnet(2,L) = vbvlzn(3,L,izone) - vbvlzn(4,L,izone)  !net rate (in-out)
          end do
          xfrnet(idxvol) = xfrinp(idxvol,izone)-xfrout(idxvol,izone)
          totnet(idxvol) = totizn(idxvol,izone)-totozn(idxvol,izone)
325       format (i4,i6,3i4,f10.2,12e12.4)
cc        write (iobud,330) izone,kstp,kper,kiter,timsim,
cc   1      (vbvlzn(idxinp,j,izone),j=1,msum1),
cc   1          xfrinp(idxvol,izone),totizn(idxvol,izone),
cc   1      (vbvlzn(idxout,j,izone),j=1,msum1),
cc   1          xfrout(idxvol,izone),totozn(idxvol,izone),
cc   1      diff(idxvol),pdiff(idxvol)
330       format(4i4,23g12.4)
          write (ionetz,325) izone,nczone(izone),ncaczn(izone),
     1      kstp,kper,kiter,timsim,totnet(idxvol),xfrnet(idxvol),
     1      (volnet(idxvol,j),j=1,msum1)
         end if    !nczone(izone) > 0
        end do    !izone=0,mxzone
        return
        end
```

Subrs. Zonini, Zonacc, Zonsum, Zstgini, Zstgsum, Zonxfr, Zbfxfr, Prompt2

Subr. ZONINI

```
c-----------------------------------------------------------------------
      subroutine ZONINI (mxzone,ratzin,ratzou)
c initialize rates for zone budgets; call from budget subroutines x1BDZ
c for x=WEL, DRN, RCH, EVT, RIV, STR, GHB.
      dimension ratzin(0:mxzone),ratzou(0:mxzone)
      do iz=0,mxzone
          ratzin(iz)=0.
          ratzou(iz)=0.
      end do
      return
      end
```

Subr. ZONACC

```
c-----------------------------------------------------------------------
      subroutine ZONACC (Q,iz,mxzone,ratzin,ratzou)
c accumulate flow rate for zone iz; call from budget subroutines x1BDZ
c for x=WEL, DRN, RCH, EVT, RIV, STR, GHB.
      dimension ratzin(0:mxzone),ratzou(0:mxzone)
      if (Q.lt.0.) then
        ratzou(iz) = ratzou(iz) - Q
      else if (Q.gt.0.) then
        ratzin(iz) = ratzin(iz) + Q
      end if
      return
      end
```

Subr. ZONSUM

```
c-----------------------------------------------------------------------
      subroutine ZONSUM (msum,mxzone,delt,ratzin,ratzou,vbvlzn) !add rates for each zone
c accumulate flow rates for zone budgets; call from budget subroutines x1BDZ
c for x=WEL, DRN, RCH, EVT, RIV, STR, GHB.
      dimension vbvlzn(4,20,0:mxzone),
     1    ratzin(0:mxzone),ratzou(0:mxzone)
      do iz=0,mxzone
          vbvlzn(1,MSUM,iz)=vbvlzn(1,MSUM,iz)+ratzin(iz)*DELT
          vbvlzn(2,MSUM,iz)=vbvlzn(2,MSUM,iz)+ratzou(iz)*DELT
          vbvlzn(3,MSUM,iz)=ratzin(iz)
          vbvlzn(4,MSUM,iz)=ratzou(iz)
      end do
      return
      end
```

Subr. ZSTGINI

```
c-----------------------------------------------------------------------
      subroutine ZSTGINI (mxzone,ratzin,ratzou,vbznfl)
c initialize stg & xfr rates for each zone; call from BCF1BDZ (orig. BCF1BD).
      dimension ratzin(0:mxzone),ratzou(0:mxzone),
     1          vbznfl(2,0:mxzone,0:mxzone)
      do iz=0,mxzone
        ratzin(iz)=0.
        ratzou(iz)=0.
        do izone=0,mxzone
          do j=1,2
            vbznfl(j,iz,izone)=0.  !initialize zone transfers
          end do
        end do
      end do
      return
      end
```

Subr. ZSTGSUM

```
c-----------------------------------------------------------------------
      subroutine ZSTGSUM (msum,mxzone,delt,ratzin,ratzou,vbvlzn)
c add rates for each zone; call from BCF1BDZ (orig. BCF1BD)
      dimension vbvlzn(4,20,0:mxzone),
     1    ratzin(0:mxzone),ratzou(0:mxzone)
      do iz=0,mxzone
          vbvlzn(1,MSUM,iz)=vbvlzn(1,MSUM,iz)+ratzin(iz)
          vbvlzn(2,MSUM,iz)=vbvlzn(2,MSUM,iz)+ratzou(iz)
          vbvlzn(3,MSUM,iz)=ratzin(iz)/DELT
          vbvlzn(4,MSUM,iz)=ratzou(iz)/DELT
      end do
      return
      end
```

Subr. ZONXFR

```
c-----------------------------------------------------------------------------
      subroutine ZONXFR(nlay,nrow,ncol,idxarg,mxzone,buff,izonbd,vbznfl)

c interzone transfers
c called in MODBCF package subr SBCF1BZ (orig. SBCF1B)
c idxarg: interzone transfer indicator: between columns(1), rows(2), or layers(3).

      dimension BUFF(NCOL,NROW,NLAY)
      dimension izonbd(ncol,nrow,nlay) !budget zone index for each grid cell
      dimension vbznfl(2,0:mxzone,0:mxzone)   !interzone transfers
      if (idxarg.eq.1) then
        do k=1,nlay
          do i=1,nrow
            do j=1,ncol-1       !interzone transfers between columns j and j+1
              call ZBFXFR (buff(j,i,k),
     1            izonbd(j,i,k),izonbd(j+1,i,k),mxzone,vbznfl)
            end do
          end do
        end do
      else if (idxarg.eq.2) then
        do k=1,nlay
          do i=1,nrow-1         !interzone transfers between rows i and i+1
            do j=1,ncol
              call ZBFXFR (buff(j,i,k),
     1            izonbd(j,i,k),izonbd(j,i+1,k),mxzone,vbznfl)
            end do
          end do
        end do
      else if (idxarg.eq.3) then
        do k=1,nlay-1           !interzone transfers between layers k and k+1
          do i=1,nrow
            do j=1,ncol
              call ZBFXFR (buff(j,i,k),
     1              izonbd(j,i,k),izonbd(j,i,k+1),mxzone,vbznfl)
            end do
          end do
        end do
      end if
      return
      end
```

Subr. ZBFXFR

```
c-----------------------------------------------------------------------------
      subroutine ZBFXFR (rbuff,nz,nza,mxzone,vbznfl)
c         called in MODPOST package subr ZONXFR: transfers between zones
      dimension vbznfl(2,0:mxzone,0:mxzone)
c
      if (nz.ge.0.and.nza.ge.0.and.nz.ne.nza) then
          IF(rbuff.LT.0.) THEN
c                       a negative transfer from zone nz to zone nza:
                  VBZNFL(2,NZ,NZA)=VBZNFL(2,NZ,NZA)-RBUFF
c                       equals a negative transfer to zone nza from zone nz:
                  VBZNFL(1,NZA,NZ)=VBZNFL(1,NZA,NZ)-RBUFF
          ELSE if (rbuff.gt.0.) then
                  VBZNFL(1,NZ,NZA)=VBZNFL(1,NZ,NZA)+RBUFF
                  VBZNFL(2,NZA,NZ)=VBZNFL(2,NZA,NZ)+RBUFF
          END IF
      end if
      return
      end
```

Subr. PROMPT2

```
c-----------------------------------------------------------------------
      subroutine PROMPT2 (query,answer,inpdev)
c
      character *(*) query,answer
      character *1 ansdft
      integer inpdev,stdout
      data stdout /6/
c
      write (stdout,100) query, answer
100   format (1x,a,'[',a,']: ')
      ansdft = answer
      answer = ' '
      read (inpdev,'(a)') answer
      if (answer.eq.' ') answer = ansdft
      i = ICHAR(answer)
      if (i.ge.ICHAR('a') .and. i.le.ICHAR('z'))
     1    answer(1:1) = CHAR(i-32)
      return
      end
```

Swat routines modified for weather, irrigation, and HRU options

Subr. Clicon: read or simulate daily weather data (file cliconsp.for)

```
c file clicon.for: subr clicon (using formats from '93 version of SWAT v.1)

      subroutine clicon (iopwfl,fmttmp,fmtpcp,fmtrad,iorad,
     1                   itsim,ipsim,julday,icalyr)
c
c read or simulate daily weather data.
c
c Argument list:
c iorad  (added) input device for reading daily weather variables
c        associated with the following option, iopwea (below).
c        Iopwea is read from the *.cod file (source: swatmod2.h, inserted
c        into Swat's mainline).  If iopwea = 0, set iorad to zero which will
c        indicate in this routine that these additional data are not to be read.
c
c itsim and ipsim indicate whether (y=1,n=0) daily temperature min & max and
c       precipitation have been simulated due to missing data.
c julday Julian day of year
c icalyr calendar year
c
c Notes on added weather-related options (see swatmod2.h for list):

c a) The following arguments are defined in Swatmod2.h ("included" in mainline):
c     fmttmp fmtpcp ithsta
c
c b) Argument ithsta was previously passed to subr cliconsp to indicate that
c Theissen weights are to be read from Swatmod2.h (main) and applied here.
c Code to average according to Theissen weights has been disabled here, so
c switch ithsta is not passed to subr clicon now, but is set internally
c to zero here.
c
c c) iopwfl = 1: option to read daily precipitation for all stations from a
c    single data file with device no. 10; and to similarly read daily
c    temperature min and max for all stations from a single data file,
c    device no. = 28.
c
c d) Changed cliconsp format for reading daily precipitation and temperature
c data back to that fo the Swat '93 version:  read one daily rainfall value
c per record and two daily temperature values (max and min) per record.
c The (5x,2f10.1) format used here (5000 format) is the same as the '93
c version's 10300 format.
c
c e) The additional daily weather variables for radiation, relative humidity,
c and wind speed u10 (at 10 m ALS) were used as inputs to the added subroutine
c Evap8, to calculate potential evaporation for a reference crop
c according to the equation recommended by Shuttleworth in Handbook of
c Hydrology (Maidment, ed., 1993).

C     THIS SUBROUTINE CONTROLS WEATHER INPUTS

      character*(*) fmttmp,fmtpcp,fmtrad
      include 'common.f'
      parameter (mxsta=20) ! max. no. stations avg'd within each subbasin
c         Theissen averaging arrays (disabled; see also ithsta)
c     dimension nstpcp(mxsta),istpcp(mxsta,mb), rainwt(mxsta,mb),
c    1          nsttmp(mxsta),isttmp(mxsta,mb), tempwt(mxsta,mb)
      data initlz /0/   ! indicate whether routine has been initialized.
      data ithsta /0/   ! indicates whether Theissen weights are to be applied.
      itravg=0
      if (initlz.eq.0) then
          initlz = 1
          itravg = 1 ! set trace switch to show results of first day's
*             interpolation of precip stations.
          itfree = INDEX(fmttmp,'FREE')
```

```
         irfree = INDEX(fmtpcp,'FREE')
         isfree = INDEX(fmtrad,'FREE')

c              disabled code for Theissen weights:
c            if (ithsta.ne.0) then
c                read precip & temperature gage weights for each subbasin:
c                do ir=1,lu                              !Theissen wts for Precip
c                  read (10,*) nstpcp(ir),(istpcp(k,ir),k=1,nstpcp(ir)),
c    1                                    (rainwt(k,ir),k=1,nstpcp(ir))
c                do ir=1,lu                              !Theissen wts for Temp
c                  read (10,*) nsttmp(ir),(isttmp(k,ir),k=1,nsttmp(ir)),
c    1                                    (tempwt(k,ir),k=1,nsttmp(ir))
c                end do
c            end if
         print '(2(1x,a,i3))','Subr clicon: ntgage=',ntgage,
     1        'nrgage=',nrgage
         print '(1x,a,30i3)','itgage=',(itgage(k),k=1,lu)
         print '(1x,a,30i3)','irgage=',(irgage(k),k=1,lu)
      end if
*        flags ipsim and itsim indicate whether (y=1,n=0) precipitation or
*        temperature values have been simulated for the current day.
      ipsim=0
      itsim=0
      precip = 0.
      do 10 k = 1, lu
        subp(k) = 0.
   10 continue
      lsim = 0
      if (msim.ne.2.and.msim.ne.4) then
ccccc      if (nwsdat.eq.1) then
ccccc          call nwsinp (jday,
C       TMX = DAILY MAXIMUM TEMPERATURE FOR EACH SUBBASIN
C       TMN = DAILY MINIMUM TEMPERATURE FOR EACH SUBBASIN
c spp   changed code back to read one daily pair of temperature values
c spp   (max and min, resp.) per record; see note above.
CREAD

c spp          kk1 = 10 * (k-1) + 1
c spp          kk2 = kk1 + 9
c****format changed for pc version
c              read (27+k,5000) (txmeas(l),tnmeas(l),l = kk1,kk2)
c spp          read(27+k,5000) (txmeas(l),l=kk1,kk2)
c spp          read(27+k,5000) (tnmeas(l),l=kk1,kk2)
c
c spp         use simplified version of previous read (above) --spp
         if (iopwfl.eq.0) then
           do k=1,ntgage
             if (itfree.gt.0) then
               read (27+k,*) txmeas(k),tnmeas(k)
             else
               read (27+k,fmttmp) txmeas(k),tnmeas(k)
             end if
           end do
         else
           if (itfree.gt.0) then
               read (27+1,*) jday,idxyr,
     1             (txmeas(kfld),tnmeas(kfld),kfld=1,ntgage)
               if (jday.ne.julday.or.idxyr.ne.icalyr) then
                 print '(1x,a)','Clicon() date discrepancy '//
     1             'on temperature data:'
                 print '(2(1x,a,2i5))',
     1             'simulation Julian date=',julday,icalyr,
     1             'temp. data Julian date=',jday,idxyr
               end if
           else
               read (27+1,fmttmp) (txmeas(kfld),tnmeas(kfld),
     1             kfld=1,ntgage)
           end if
         end if
5000     format(5x,10f5.1)
        do 40 k = 1, lu
```

```
         if (ithsta.le.0) then
             if (ntgage.eq.0.or.itgage(k).eq.0) then
                 tmx(k) = -99.
                 tmn(k) = -99.
             else
                 tmx(k) = txmeas(itgage(k))
                 tmn(k) = tnmeas(itgage(k))
             end if
         end if
c            disabled theissen weighting:
c        else
c            tmx(k)=0.
c            tmn(k)=0.
c            do kfract = 1,nsttmp(k)
c                itk = isttmp(kfract,k)
c                tmx(k) = tmx(k) +txmeas(itk)*tempwt(kfract,k)
c                tmn(k) = tmn(k) +tnmeas(itk)*tempwt(kfract,k)
c                if (itravg.gt.0)
c            end do
c        end if
         if (tmx(k).lt.-97..or.tmn(k).lt.-97.) then
            itsim=1
            mm = msim
            msim = 2
            nn = nsim
            nsim = 1
            call clgen4(k,dd,x1)
            nsim = nn
            msim = mm
            tmx(k) = tmxg
            tmn(k) = tmng
          end if
   40  continue
       if (msim.ne.3) then
         do 50 k = 1, lu
           tmx(k) = tmx(1)
           tmn(k) = tmn(1)
   50    continue
       end if
     end if
     if (nsim.ne.2.and.nsim.ne.4) then
C      SUBP = DAILY RAINFALL FOR EACH SUBBASIN
CREAD
c spp   changed code to read one daily rainfall value per record;
cc     see note above.
cccc          kk1 = 10 * (k-1) + 1
cccc          kk2 = kk1 + 9
cccc          read (9+k,5000) (rmeas(l),l = kk1,kk2)
        if (iopwfl.eq.0) then
          do k=1,nrgage
            if (irfree.gt.0) then
              read (9+k,*) rmeas(k)
            else
              read (9+k,fmtpcp) rmeas(k)
            end if
          end do
        else
          if (irfree.gt.0) then
              read (9+1,*) jday,idxyr,(rmeas(kfld),kfld=1,nrgage)   ! -spp
              if (jday.ne.julday.or.idxyr.ne.icalyr) then
                print '(1x,a)','Clicon() date discrepancy '//
     1             'on precipitation data:'
                print '(2(1x,a,2i5))',
     1             'simulation Julian date=',julday,icalyr,
     1             'prcp. data Julian date=',jday,idxyr
              end if
          else
              read (9+1,fmtpcp) (rmeas(kfld),kfld=1,nrgage)  ! -spp
          end if
        end if
```

```
          do 80 k = 1, lu
            if (ithsta.le.0) then
                if (nrgage.eq.0.or.irgage(k).eq.0) then
                    subp(k) = -99.
                else
                    subp(k) = rmeas(irgage(k))
                end if
            end if
c               disabled theissen weighting:
c           else
c               subp(k)= 0.
c               do kfract = 1,nstpcp(k)
c                   irk = istpcp(kfract,k)
c                       check to see if any station values are missing:
c                   if (rmeas(irk).lt.-97.) then
c                       subp(k) = -99.  !don't average missing values
c                       go to 60
c                   else
c                       subp(k) = subp(k) +rmeas(irk)*rainwt(kfract,k)
c                   end if
c               end do
c60               continue !jump out of above loop to here if station value is missing
c             end if
            if (subp(k).lt.-97.) then
              ipsim = 1
              nn = nsim
              nsim = 2
              mm = msim
              msim = 1
              call clgen4(k,dd,xl)
              nsim = nn
              msim = mm
              subp(k) = precip
            end if
   80     continue
        else
          if (nsim.ne.4) then
            rmn = 0.
            do 90 k = 1, lu
              rmn = rmn + subp(k) * flu(k) * p2(k)
   90       continue
          else
            rmn = precip
            if (rmn.ne.0.) then
              if (snob.le.0..or.tmxg.le.0.) then
                sml = 0.
              else
                sml = 4.57 * tmxg
                if (sml.gt.snob) sml = snob
              end if
              call alph9(1)
              du = 4.605 / (-2.*alog(1.-r1))
              if (du.lt..5) du = .5
              due = du ** (-.1478)
              v12 = randn(k8)
              v13 = randn(k9)
              xx = v12 * xsl + sx
              yy = v13 * ysl + ys
              sum = 0.
              do 100 k = 1, lu
                dia = (xx-xij(k)) ** 2 + (yy-yij(k)) ** 2
                ada = .7854 * dia
                sl(k) = 1. - ada * due / (337.5+1.0935*ada)
                sum = sum + sl(k)
  100         continue
              rto = xlu * rmn / sum
              rmn = 0.
              do 110 k = 1, lu
                subp(k) = sl(k) * rto
                rmn = rmn + subp(k) * flu(k) * p2(k)
  110         continue
```

```
          else
            do 120 k = 1, lu
              subp(k) = 0.
  120       continue
          end if
        end if
      end if
      do 130 k = 1, nsb
        ssb(k) = 0.
  130 continue
      if (msim.eq.4) then
        tmx(1) = tmxg
        tmn(1) = tmng
        do 140 l = 2, lu
          tmx(l) = tmxg + (vobmx(mo,l)-obmx(mo,j))
          tmn(l) = tmng + (vobmn(mo,l)-obmn(mo,j))
  140   continue
      end if
      if (msim.ne.1.and.msim.ne.3) then
        itsim=1
        do 150 l = 1, lu
          tmx(l) = tmxg
          tmn(l) = tmng
  150   continue
      end if
      lsim = 1
      do 160 j = 1, lu
        precip = subp(j)
        call clgen4(j,dd,x1)
  160 continue
*           added option to read radiation (ly/d), relative humidity (pct)
*           and wind speed (m/s measured 2m above ground):
      if (iorad.gt.0) then
          if (isfree.gt.0) then
              read (iorad,*) radmea,relhum,wndspd
          else
              read (iorad,fmtrad) radmea,relhum,wndspd
          end if
          if (relhum.gt.-97. .and. relhum.le.100.) relhum = relhum/100.
          do k=1,Lu
              if (radmea.gt.-97. .and. radmea.lt.9999.) ra(k) = radmea
              if (relhum.gt.-97. .and. relhum.lt.9999.) rhd(k) = relhum
              if (wndspd.gt.-97. .and. wndspd.lt.9999.) u10(k) = wndspd
          end do
      end if
      rmn = 0.
      do 170 k = 1, lu
c       subp(k) = subp(k) * (1. - rfinc(k,mo))
        subp(k) = subp(k) * (1. - rfinc(k,mo)/100.)
        rmn = rmn + subp(k) * flu(k) * p2(k)
        tmx(k) = tmx(k) + tmpinc(k,mo)
        tmn(k) = tmn(k) + tmpinc(k,mo)
        ra(k) = ra(k) + radinc(k,mo)
        rhd(k) = rhd(k) + huminc(k,mo)
        co2(k) = co2(k) + co2inc(k,mo)
  170 continue
      return
c****format changed for pc swat
c 5000 format (5x,20f5.1)
      end
```

Subr. Subbasin

```
* file subbasin.for  modified by spp
cspp $debug
      subroutine subbasin (icode,ihout,inum1,inum2,inum3,rnum1,
     1           julday,iaqufr,ioirr,iprirr,ioevt,iprevt,iopswm,
     1           wsfmax,swminf,evtgw,egwuna)
```

```
c revised call to subr subbasin (see swatmod1.h for argument list definition):
c
c  arguments:
c  inum1: subbasin index, copied to internal variable j
c  added arguments (read from modified *.cod input file by code on Swatmod2.h,
c  inserted into main):
c
c  iaqufr: indicates whether soil water is allowed to percolate out of
c  soil profile; argument added by spp.
c  If iaqufr=0, soil profile is assumed to be underlain by bedrock;
c  if iaqufr>0, soil water is not blocked by bedrock and can percolate
c          down to bedrock.  Pass to subr Purk18 and from there to Perc16.
c     iaqufr=1, aquifer is considered deep so that evap from gw is negligible.
c     iaqufr=2, aquifer is considered shallow and evap from gw is read from
c           file written by previous iteration of Modflow execution.
c  ioirr: device number for daily irrigation file subject to following option
c  iprirr: option (yes: iprirr=j; no: iprirr=0 for subbasin j):
c         write daily irrigation file in subr. Crpmd (Main->Subbasin->Crpmd).
c  ioevt: device number for daily evaporation file subject to following option
c  iprevt: option (yes: iprevt=j; no: iprevt=0 for subbasin j):
c         write daily evaporation file in subr. Penman
c        (Main->Subbasin->Evap->Penman).
c  iopswm: option for automatic irrigation (irr(j) = 1:
c    iopswm = 1: use threshold on soil water content, sw(j) < swminf;
c    iopswm = 0: use threshold on plant water stress factor, ws(j) < wsf(j).
c  wsfmax = daily max. irrigation (mm/operation) for auto. irrigation options
c      irr(j)=1 or irr(j)=3, set in the ~.mco (management codes file), and
c      applied in subr crpmd.
c  swminf = soil water threshold as a fraction of available field capacity
c          given as a column height, below which irrigation will be applied
c          for option irr(j)=3, set in the ~.mco file; swminf is applied in
c          subr crpmd.
c  evtgw = evaporation rate from water table, sub j, calculated in
c          MODFLOW's subr EVT1FM (and EVT1BD) at beginning of the gw
c          time step, summarized for subbasin, and returned as evtgw;
c          passed to routines for evaporation, soil water routing, and plant
c          water uptake.
c
      include 'common.f'
C     THE FOLLOWING VARIABLES ARE OUTPUT ON DAILY, MONTHLY, YEARLY
C     AND AVERAGE ANNUAL BASIS.  FOR EXAMPLE, SSB1=YEARLY DAILY
C     PRECIPITATION, SMM1=TOTAL MONTHLY PRECIPITATION, SMY1=TOTAL
C     YEARLY PRECIPITATION, AND SM1=AVERAGE ANNUAL PRECIPITATION.
C     SM1  = PRECIPITATION (MM)
C     SM3  = SURFACE RUNOFF (MM)
C     SM4  = SUBSURFACE FLOW (MM)
C     SM5  = PERCOLATE OUT OF SOIL PROFILE (MM)
C     SM6  = TOTAL WATER YIELD (MM)
C     SM7  = EVAPOTRANSPIRATION (MM)
C     SM10 = SOLAR RADIATION (LY)
C     SM12 = SEDIMENT YIELD (T/HA)
C     SM36 = SNOW FALL (MM)
C     SM37 = SNOW MELT (MM)
C     SM38 = CHANNEL TRANSMISSION LOSSES (MM)
C     THE FOLLOWING VARIABLES ARE PRINTED OUTPUT ONLY ON AN AVERAGE
C     ANNUAL BASIS.
C     SM11 = TOTAL SUB-BASIN SEDIMENT YIELD (T/HA)
C     SM13 = SEDIMENT INFLOW TO PONDS (T/HA)
C     SM14 = SEDIMENT OUTFLOW FROM PONDS (T/HA)
C     SM14 = SEDIMENT OUTFLOW FROM PONDS (T/HA)
C     SM16 = SEDIMENT INFLOW TO RESERVOIRS (T/HA)
C     SM17 = SEDIMENT OUTFLOW FROM RESERVOIRS (T/HA)
C     SM19 = POND EVAPORATION (MM)
C     SM20 = POND SEEPAGE (MM)
C     SM21 = RAINFALL ON PONDS (MM)
C     SM22 = WATER INFLOW TO PONDS (MM)
C     SM23 = WATER OUTFLOW FROM PONDS (MM)
C     SM24 = RESERVOIR EVAPORATION (MM)
C     SM25 = RESERVOIR SEEPAGE (MM)
C     SM26 = RAINFALL ON RESERVOIRS (MM)
```

```
c      SM27 = WATER INFLOW TO RESERVOIRS (MM)
c      SM28 = WATER OUTFLOW FROM RESERVOIRS (MM)
c      COMPUTE RUNOFF VOLUME FOR EACH SUBAREA.  LU = NO IF SUBAREAS
c      ssub
c      SUBBASIN                     CROP GROWTH                  NUT BAL
c      1 prec(mm)                   29                          35 org n yield(kg/ha)
c      2 snowfall(mm)               30                          36 sed p yield(kg/ha)
c      3 snowmelt(mm)               31 water stress days (sdw)  37 no3 in sur q(kg/ha)
c      4 surface q(mm)              32 temp stress days (sdt)   38 no3 in sub sur
q(kg/ha)
c      5 lateral q(mm)              33 n stress days (sdn)      39 sol p yield(kg/ha)
c      6 groundwater q(mm)          34 p stress days (sdp)      40 n uptake(kg/ha)
c      7 revap(mm)                                              41 no3 leached(kg/ha)
c      8 groundwater perc(mm)                                   42 p uptake(kg/ha)
c      9 groundwater rechrg(mm)                                 43 active to labile
p(kg/ha)
c      10 water yld(mm)                                         44 active to stable
p(kg/ha)
c      11 perc root zone(mm)                                    45 n fert applied(kg/ha)
c      12 ET(mm)                                                46 p fert applied(kg/ha)
c      13 trans losses(mm)                                      47 n fixation(kg/ha)
c      14 sed yld(t/ha)                                         48 denitrification(kg/ha)
c      15 pond evap(mm)                                         49 humus min act org
n(kg/ha)
c      16 pond seepage(mm)                                      50 act to stable org
n(kg/ha)
c      17 pond rainfall(mm)                                     51 humus min act org
p(kg/ha)
c      18 pond inflow(mm)                                       52 min from fresh org
n(kg/ha)
c      19 pond sed. inflow(t/ha)                                53 min from fresh org
p(kg/ha)
c      20 pond outflow(mm)                                      54 soil water(mm)
c      21 pond sed. outflow(t/ha)                               55 biomass(kg/ha)
c      22 irrigation app(mm)                                    56 leaf area index
c      23 irr from shallow aquifer(mm)                          56 leaf area index
c      24 irr from deep aquifer(mm)
c      25 potential et(mm)
c      26 shallow gw storage(mm)
c      27 deep gw storage(mm)
c      28
       j = inum1
c
c      if the subbasin is water compute only pet and et
c      using Priestly-Taylor and a coefficient
c
       if (be(ncr(1,1,j)).le.0.) then
         tk = tx(j) + 273.
         alb = .08
         d = exp(21.255-5304./tk) * 5304. / tk ** 2
         gma = d / (d+.68)
         ho = ra(j) * (1.-alb) / 58.3
         aph = 1.28
         pe = aph * ho * gma
         aet = .7 * pe
         ssub(1,j) = ssub(1,j) + subp(j) *p2(j)
         ssub(12,j) = ssub(12,j) + aet
         ssub(25,j) = ssub(25,j) + pe
         ssb(1) = ssb(1) + subp(j) * p2(j) * flu(j)
         ssb(7) = ssb(7) + aet
         sbp(j) = sbp(j) + subp(j)
       else

       nn = ns(j)
       qd = 0.
       qi = 0.
       ql = 0.
       ssf = 0.0
       yd = 0.
       qtl = 0.
       pq = 0.
```

```
      qdr = 0.
      twl = 0.
      sml = 0.
      c = 0.
      yon = 0.
      yph = 0.
      ysp = 0.
      yno3 = 0.
      uno3 = 0.
      ssfn = 0.
      percn = 0.
      ws(j) = 1.
      ts = 1.
      strsn = 1.
      strsp = 1.
      ssb(8) = ssb(8) + tmx(j) * flu(j)
      ssb(9) = ssb(9) + tmn(j) * flu(j)
      tx(j) = (tmx(j)+tmn(j)) / 2.
      aff = af * flu(j)
      bcv(j) = cv(j) / (cv(j)+exp(7.563-1.297e-4*cv(j)))
      if (sno(j).ne.0.) then
        if (sno(j).le.120.) then
          xx = sno(j) / (sno(j)+exp(6.055-.3002*sno(j)))
        else
          xx = 1.
        end if
        bcv(j) = amax1(xx,bcv(j))
      end if
      call solt22(zz,j)
c-------------------------------------------------------------
C     IF(NSIM.LT.3) PRECIP=RMN
C     IF(NSIM.GE.3) PRECIP=SUBP(J)*P2(J)
      precip = subp(j) * p2(j)
      ssub(1,inum1) = ssub(1,inum1) + precip
C     SDA(1,J)=PRECIP
      sbp(j) = sbp(j) + precip
      ssbp(j) = ssbp(j) + precip * precip
      if (precip.gt.0.) nrf(j) = nrf(j) + 1
*-------------------spp
c     call volq11(j)
cc    subroutine volq11(j)
C     THIS SUBROUTINE PREDICTS DAILY RUNOFF GIVEN DAILY PRECIPITATION
C     AND SNOW MELT USING A MODIFIED SCS CURVE NUMBER APPROACH
C     QD = SURFACE RUNOFF IN MM
C******common.f
ccccc include 'common.f'
      sum = 0.
      do k = 1, nn
        sum = sum + st(k,j)
      end do
      xx = wf(1,j) - wf(2,j) * sum
      if (xx.lt.-20.) xx = -20.
      if (xx.gt.20.) xx = 20.
      r2 = smx(j) * (1.-sum/(sum+exp(xx)))
      if (t(2,j).le.0.) r2 = smx(j) * (1.-exp(-.000862*r2))
      crvnum = 25400. / (r2+254.)
      r2 = 25400. / crvnum - 254.
      bb = .2 * r2
      pb = precip - bb
      if (pb.gt.0.) then
        qd = pb * pb / (precip+.8*r2)
      end if
cc    return
cc    end
*-------------------spp
      if (tx(j).le.0.) then                   !temperature is below freezing
        ssb(39) = ssb(39) + precip * flu(j)
        ssub(2,inum1) = ssub(2,inum1) + precip
        sno(j) = sno(j) + precip
        snow = precip
        precip = 0.
        else
          if (sno(j).gt.0.) call snom5(j)
          if (precip.ne.0.) then
c           COMPUTE RUNOFF - QD IN MM
            call volq11(j)
            qi = qd
            fl = fl + flu(j)
            ssb(2) = ssb(2) + cn * flu(j)
            smq(j) = smq(j) + qi
            ssub(4,inum1) = ssub(4,inum1) + qi
            ssb(3) = ssb(3) + qi * flu(j)
            if (qd.gt.0.0001.and.precip.gt.0.) then
              if (ipr.le.0) then
                call alph9(j)
                ipr = 1
              end if
c             COMPUTE PEAK RATE - PR IN M**3/S
              call pkq10(j)
              data prmin /1.e-6/
              if (qd.gt.0..and.pr.gt.prmin) then
                iv = 1
                vo = qd * da * flu(j) * 1000.
                dur = vo / (pr*3600.)
                if (dur.gt.24.) dur = 24.
c               COMPUTE TRANSMISSION LOSSES
                if(vo.gt.0..and.pr.gt.prmin) call tran12(j)
                qtl = ql - qd
                if (qtl.lt.0.) then
                  qd = ql
                  qtl = 0.
                end if
                ssb(38) = ssb(38) + qtl * flu(j)
                ssub(13,inum1) = ssub(13,inum1) + qtl
c               add transmission losses, qtl, to shallow aquifer storage
                shallst(j) = shallst(j) + qtl
                call ysed13(j,0)
              end if
            end if
            if (cn.ge.vl) then
              if (cn.lt.vb) go to 10
              vb = cn
            else
              vl = cn
            end if
          end if
        end if
   10 rain = precip - qd
        snob = snob + sno(j) * flu(j)
        ssub(3,inum1) = ssub(3,inum1) + sml
        ssb(36) = ssb(36) + sml * flu(j)
        cf(j) = 0.01 * (1.-sw(j)/(sw(j)+exp(7.0-0.11*sw(j))))
        do 20 k = 1, nptot(j)
          z0(k,j) = 0.
          ssfp(k,j) = 0.
   20 continue
c
c     PERFORM SOIL WATER ROUTING

        call purk18(j,iaqufr)  !Percolation: argument iaqufr added: spp
        ssf = prk
        call pestlch(j)
c     SUM LEACHED PESTICIDES
        do 30 k = 1, nptot(j)
          k83 = k + 83
          ssb(k83) = ssb(k83) + z0(k,j) * flu(j)
          k93 = k + 93
          ssb(k93) = ssb(k93) + ssfp(k,j) * flu(j)
   30 continue
c     COMPUTE NITRATE LEACHING
        call nlch(j)
        ssb(5) = ssb(5) + sep * flu(j)
```

```
      ssub(11,inum1) = ssub(11,inum1) + sep
      smsq(j) = smsq(j) + ssf

c
c       test code for simulating uptake from shallow water table --spp
      if (iaqufr.eq.2.and.evtgw.gt.0)
     1  call Capflow (j,nn,ml,mb,st,ul,fc,evtgw,egwuna)

C     COMPUTE eo = potential (crop reference) ET.

      call evap8 (j,julday,ioevt,iprevt)

      ssb(108) = ssb(108) + eo * flu(j)
C     APPLY FERTILIZER-CHECK DAY AND HEAT UNITS
   40 if (ida.eq.ifert(icr(j),nfert(j),j)) go to 41
      if (igro(j).eq.0) then
        if (gplt(j).gt.phun(icr(j),nfert(j),j)) go to 41
      else
        if (g(j).gt.phun(icr(j),nfert(j),j)) go to 41
      endif
      go to 42
   41 ssub(45,inum1) = ssub(45,inum1) + fn(nro(j),nfert(j),j)
      ssub(46,inum1) = ssub(46,inum1) + fph(nro(j),nfert(j),j)
      call fert(j)
      go to 40
42    continue

C     COMPUTE CROP GROWTH
c
c       Note: Automatic irrigation option (irr(j) = 1, specified in the
c       ~.mco file), limits daily irrigation in crpmd6 to a column height of
c       wsfmax (mm), which is specified in the *.cod file. --spp

      call crpmd6 (j,julday,ioirr,iprirr,iopswm,wsfmax,swminf)

csp42 call crpmd6(j)    ! (original call)  --spp

      ssub(45,inum1) = ssub(45,inum1) + ano3
      ssub(46,inum1) = ssub(46,inum1) + tfp
      ssub(31,inum1) = ssub(31,inum1) + (1.-ws(j))
      ssub(32,inum1) = ssub(32,inum1) + (1.-ts)
      ssub(33,inum1) = ssub(33,inum1) + (1.-strsn)
      ssub(34,inum1) = ssub(34,inum1) + (1.-strsp)
      uw = ep + es
      ssb(7) = ssb(7) + uw * flu(j)
      ssub(12,inum1) = ssub(12,inum1) + uw
      ssub(25,inum1) = ssub(25,inum1) + eo
C     COMPUTE N AND P MINERALIZATION
      call nmnim(j)
      call npmin(j)
      ssub(42,inum1) = ssub(42,inum1) + uap
      ssub(43,inum1) = ssub(43,inum1) + rmn1tl
      ssub(44,inum1) = ssub(44,inum1) + roctl
      ssub(47,inum1) = ssub(47,inum1) + wfx
      ssub(48,inum1) = ssub(48,inum1) + wdntl
      ssub(49,inum1) = ssub(49,inum1) + hmntl
      ssub(50,inum1) = ssub(50,inum1) + rwntl
      ssub(51,inum1) = ssub(51,inum1) + hmptl
      ssub(52,inum1) = ssub(52,inum1) + rmn2tl
      ssub(53,inum1) = ssub(53,inum1) + rmptl
C     COMPUTE GROUND WATER CONTRIBUTION
      call gwmod(j)
*         use evap from water table as calculated by subr EGWSWAT, based on
*         initial dtw, but otherwise the same as calculated in Modflow.
      revap = evtgw
      ssub(6,inum1) = ssub(6,inum1) + gwq(j)
      ssb(104) = ssb(104) + gwq(j) * flu(j)
*         coordinate evap from gw (evtgw) with this:
      ssub(7,inum1) = ssub(7,inum1) + revap
      ssb(105) = ssb(105) + revap * flu(j)
      ssub(8,inum1) = ssub(8,inum1) + gwseep
      ssb(106) = ssb(106) + gwseep * flu(j)
      ssub(9,inum1) = ssub(9,inum1) + gwchrg
      ssb(107) = ssb(107) + gwchrg * flu(j)
      ssb(109) = ssb(109) + eo * flu(j)
C     APPLY PESTICIDE
      call apply(j)
C       COMPUTE PESTICIDE WASHOFF
      if (precip.ge.2.54) call washp(j)
C     COMPUTE PESTICIDE DECAY
      call decay(j)
      ssb(42) = ssb(42) + yno3 * flu(j)
      ssub(37,inum1) = ssub(37,inum1) + yno3
      ssb(44) = ssb(44) + uno3 * flu(j)
      ssub(40,inum1) = ssub(40,inum1) + uno3
C     IF(J.EQ.2)WRITE(6,642) J,IDA,UNO3
C642  FORMAT(' !N UPTAKE!  J,IDA,UNO3 ',2I6,F12.3)
      ssb(45) = ssb(45) + ssfn * flu(j)
      ssub(38,inum1) = ssub(38,inum1) + ssfn
      ssb(46) = ssb(46) + percn * flu(j)
      ssub(41,inum1) = ssub(41,inum1) + percn
      ssb(37) = ssb(37) + snoev * flu(j)
      sw(j) = 0.
      qdr = qd + ssf + gwq(j)
c     ADJUST RUNOFF FOR WETLAND STORAGE
      if (fw(j).ne.0.) then
        pq = yd * fw(j)
        yd = yd - pq
        ssb(16) = ssb(16) + pq
        ssub(34,inum1) = ssub(34,inum1) + pq
        call wetlan(j)
        ssb(33) = ssb(33) + pq * flu(j)
        ssb(24) = ssb(24) + ev
        ssub(58,inum1) = ssub(58,inum1) + ev
        if (q1-qd.lt.0.) q1 = qd
        ssb(25) = ssb(25) + sp
        ssub(55,inum1) = ssub(55,inum1) + sp
c       add wetland seepage to shallow aquifer convert from m^3 to mm
        shallst(j) = shallst(j) + sp / (1000.*da*flu(j))
        ssb(26) = ssb(26) + rl
        ssub(56,inum1) = ssub(56,inum1) + rl
        ssb(27) = ssb(27) + qdr
        ssub(57,inum1) = ssub(57,inum1) + qdr
        ssb(28) = ssb(28) + o
        ssub(59,inum1) = ssub(59,inum1) + o
        twl = pq
        yd = yd + yp
        ssb(17) = ssb(17) + yp
        ssub(60,inum1) = ssub(60,inum1) + yp
        ssb(18) = ssb(18) + amps
      end if
      qdr = qdr - twl
c     ADJUST RUNOFF FOR POND/RESERVOIR STORAGE
      if (fp(j).lt.1.0.and.fp(j).gt.0.) then
        pq = yd * fp(j)
        yd = yd - pq
        ssb(13) = ssb(13) + pq
        ssub(19,inum1) = ssub(19,inum1) + pq
        call pond14(j)
        ssb(33) = ssb(33) + pq * flu(j)
*           accumulate as flux(mm) over subbasin area for
*           pond evap (ev), seepage (sp), and rainfall (rl), inflow (q1)
*           and outflow (o) as follows:
c               d(mm) = Q(m^3)/(1000*da(km^2)*flu(j)).
c           see also note below regarding inconsistency
*           of using q1 in mass balance (subr pond14) but accumulating
*           qdr to represent inflow: --spp apr 9 96
        evflux = ev/(1000.*da*flu(j))   !evap (ev: eqn 247, p 60, manual)
        spflux = sp/(1000.*da*flu(j))   !seepage (sp: eqn 248)
        rlflux = rl/(1000.*da*flu(j))   !rainfall
        q1flux = q1/(1000.*da*flu(j))   !inflow to ponds
        oflux = o/(1000.*da*flu(j))     !outflow from ponds
```

```
          ssb(19) = ssb(19) + ev / (1000.*da)
          ssub(15,inum1) = ssub(15,inum1) + evflux
          if (q1.lt.qd) q1 = qd
*             change pond seepage from flow rate (m^3/day) to flux,mm/day:
c             divide by subbasin area, da(km^2).
          ssb(20) = ssb(20) + sp / (1000.*da)
          ssub(16,inum1) = ssub(16,inum1) + spflux
c         add pond seepage to shallow aquifer convert from m^3 to mm
          shallst(j) = shallst(j) + spflux
          ssb(21) = ssb(21) + rl/ (1000.*da)
          ssub(17,inum1) = ssub(17,inum1) + rlflux
*             note: I changed the following two sums to accumulate q1 instead
*             of qdr, since q1 is the inflow used in the mass balance in
*             subr ponds; shouldn't the same variable be accumulated here? -spp
          ssb(22) = ssb(22) + q1/ (1000.*da)
          ssub(18,inum1) = ssub(18,inum1) + q1flux
          ssb(23) = ssb(23) + o/ (1000.*da)
          ssub(20,inum1) = ssub(20,inum1) + oflux
          twl = pq
          yd = yd + yp
          ssb(14) = ssb(14) + yp
          ssub(21,inum1) = ssub(21,inum1) + yp
          ssb(15) = ssb(15) + amps
          qdr = qdr - twl
        end if
c       SUBTRACT WATER USE FROM RESERVOIRS, STREAMS, SHALLOW & DEEP AQUIFERS
        call watuse(j)
        do 50 k = 1, nn
          if (st(k,j).lt.rzl) rzl = st(k,j)
          sw(j) = sw(j) + st(k,j)
   50   continue
c       bs(j) = bs(j) + ssf
c       ssf = bs(j) * brt
c       bs(j) = bs(j) - ssf
c       bst = bst + bs(j) * flu(j)
        yd = yd + ssf * aff * css(j)
        ssub(5,inum1) = ssub(5,inum1) + ssf
        ssb(4) = ssb(4) + ssf * flu(j)
        ssb(11) = ssb(11) + yd
        ssb(35) = ssb(35) + sw(j) * flu(j)
        sym(j) = sym(j) + yd
        ydi = yd
        if(AMIN1(qd,pr,rp,precip).gt.1.e-6) then
ccccc   if (qd.gt.0..and.pr.gt.prmin) then   !further restricted on preceding line --spp
          call enrsb(j)
          if (qi.gt.0.) call pestq(j)
          if (yd.gt.0.) call pesty(j,0)
          call orgn(j,0)
          call psed(j,0)
          call solp(j)
          call delr24(j)
        end if
        ssub(14,inum1) = ssub(14,inum1) + yd / (100.*da*flu(j))
        ssb(12) = ssb(12) + yd
        ssb(40) = ssb(40) + yon * flu(j)
        ssub(35,inum1) = ssub(35,inum1) + yon
        ssb(41) = ssb(41) + yph * flu(j)
        ssub(36,inum1) = ssub(36,inum1) + yph
        ssb(43) = ssb(43) + ysp * flu(j)
        ssub(39,inum1) = ssub(39,inum1) + ysp
        do 60 k = 1, nptot(j)
C****CHANGE
          k46 = k + 46
          ssb(k46) = ssb(k46) + zq(k,j) * flu(j) * 1.e6
          k73 = k + 73
          ssb(k73) = ssb(k73) + zl(k,j) * flu(j) * 1.e6
          sda(k,j) = zq(k,j) * 1.e6 * flu(j) * da / 10.
          sda(k+10,j) = zl(k,j) * 1.e6 * flu(j) * da * 100.
          sda(k+20,j) = (zq(k,j)+zl(k,j)) * 1.e6
          sda(k+30,lu1) = ssb(k46) + ssb(k73)
   60   continue
```

```
c     wysb = qi + ssf + gwq(j) - pq - qtl
      wysb = qdr
      ssub(10,inum1) = ssub(10,inum1) + wysb
      sda(41,j) = precip
c     sda(42,j) = wysb
      sda(42,j) = wysb * dart(ihout) * 1000.
      sda(43,j) = yd
      sda(44,j) = yon
      sda(45,j) = yph
      sda(46,j) = yno3 + ssfn
      sda(47,j) = ysp
      sda(48,j) = eo
      sda(49,j) = pr
c     call range1(j)
      do 70 ii = 1, 9
        i41 = ii + 40
        varoute(ii,ihout) = sda(i41,j)
   70 continue
      do 80 ii = 1, 10
        i10 = ii + 9
        i21 = ii + 20
        varoute(i10,ihout) = sda(i21,j)
   80 continue
      endif
      if (inum3.gt.0) then
        wflu(inum3) = wflu(inum3) + flu(j)
        wprecip(inum3) = wprecip(inum3) + precip * flu(j)
        wcklsp(inum3) = wcklsp(inum3) + cklsp(j) * flu(j)
        wsml(inum3) = wsml(inum3) + sml * flu(j)
        wqd(inum3) = wqd(inum3) + qd * flu(j)
        wwy(inum3) = wwy(inum3) + wysb * flu(j)
        wssf(inum3) = wssf(inum3) + ssf * flu(j)
        wppq(inum3) = wppq(inum3) + pq * flu(j)
        wyd(inum3) = wyd(inum3) + yd * flu(j)
        weo(inum3) = weo(inum3) + eo * flu(j)
        wuw(inum3) = wuw(inum3) + uw * flu(j)
        wyno3(inum3) = wyno3(inum3) + yno3 * flu(j)
        wssfn(inum3) = wssfn(inum3) + ssfn * flu(j)
        wysp(inum3) = wysp(inum3) + ysp * flu(j)
        wgp(inum3) = wgp(inum3) + gp(1,j,1) * flu(j)
        wwn(inum3) = wwn(inum3) + wn(1,j) * flu(j)
        wwpo(inum3) = wwpo(inum3) + wpo(1,j) * flu(j)
        wlai(inum3) = wlai(inum3) + alai(j) * flu(j)
        dmt = dm(j) / 1000.
        wdm(inum3) = wdm(inum3) + dmt * flu(j)
        wyld(inum3) = wyld(inum3) + (1.-rwt(j)) * dmt * hia(j)
     *                 *  flu(j)
        wgwq(inum3) = wgwq(inum3) + gwq(j) * flu(j)
        wsw(inum3) = wsw(inum3) + sw(j) * flu(j)
        if (ida.eq.365) then
          walai(inum3) = walai(inum3) + xlai(j) * flu(j)
          wadm(inum3) = wadm(inum3) + dmanu(j) * flu(j)
          wayld(inum3) = wayld(inum3) + yldanu(j) * flu(j)
        endif
c     wap(inum3) = wap(inum3) + ap(1,j) * flu(j)
c     wwno3(inum3) = wwno3(inum3) + wno3(1,j) * flu(j)
      end if
      if (rnum1.gt.0.) then
        precip = wprecip(inum3) / wflu(inum3)
        sml = wsml(inum3) / wflu(inum3)
        qd = wqd(inum3) / wflu(inum3)
        wysb = wwy(inum3) / wflu(inum3)
        ssf = wssf(inum3) / wflu(inum3)
        ppq = wppq(inum3) / wflu(inum3)
        yd = wyd(inum3) / wflu(inum3)
        eo = weo(inum3) / wflu(inum3)
        uw = wuw(inum3) / wflu(inum3)
        yno3 = wyno3(inum3) / wflu(inum3)
        ssfn = wssfn(inum3) / wflu(inum3)
        ysp = wysp(inum3) / wflu(inum3)
        gwf = wgwq(inum3) / wflu(inum3)
```

```
          wsw(inum3) = wsw(inum3) / wflu(inum3)
          wdm(inum3) = wdm(inum3) / wflu(inum3)
          wlai(inum3) = wlai(inum3) / wflu(inum3)
          wyld(inum3) = wyld(inum3) / wflu(inum3)
          wcklsp(inum3) = wcklsp(inum3) / wflu(inum3)
          if (ida.eq.365) then
            walai(inum3) = walai(inum3) / wflu(inum3)
            wadm(inum3) = wadm(inum3) / wflu(inum3)
            wayld(inum3) = wayld(inum3) / wflu(inum3)
          endif
          sbsub(1,inum3) = sbsub(1,inum3) + precip
          sbsub(2,inum3) = sbsub(2,inum3) + sml
          sbsub(3,inum3) = sbsub(3,inum3) + qd
          sbsub(4,inum3) = sbsub(4,inum3) + wysb
          sbsub(5,inum3) = sbsub(5,inum3) + eo
          sbsub(6,inum3) = sbsub(6,inum3) + uw
          sbsub(12,inum3) = sbsub(12,inum3) + gwf
          if (precip.gt.1.e-6.and.qd.gt.1.e-6) then
            call alph9(j)
            call pkq10(j)
c           call tran12(j)
            if (yd.gt.1.e-6.and.pr.gt.1.e-6) then
              aff = 1000. * wflu(inum3) * da
              da9 = 100. * wflu(inum3) * da
              call ysed13(j,inum3)
              call enrsb(j)
              call orgn(j,inum3)
              call psed(j,inum3)
              call pesty(j,inum3)
              sbsub(7,inum3) = sbsub(7,inum3) + yd/(100.*da*wflu(inum3))
              sbsub(8,inum3) = sbsub(8,inum3) + yon
              sbsub(9,inum3) = sbsub(9,inum3) + yph
              sbsub(10,inum3) = sbsub(10,inum3) + yno3 + ssfn
              sbsub(11,inum3) = sbsub(11,inum3) + ysp
            end if
          end if
          da9 = 100. * da
c         ssub(10,inum1) = ssub(10,inum1) + wysb
          varoute(1,ihout) = precip
          varoute(2,ihout) = wysb * wflu(inum3) * da * 1000.
          varoute(3,ihout) = yd
          varoute(4,ihout) = yon
          varoute(5,ihout) = yph
          varoute(6,ihout) = yno3 + ssfn
          varoute(7,ihout) = ysp
          varoute(8,ihout) = eo
          varoute(9,ihout) = pr
          wflu(inum3) = 0.
          wprecip(inum3) = 0.
          wcklsp(inum3) = 0.
          wsml(inum3) = 0.
          wqd(inum3) = 0.
          wwy(inum3) = 0.
          weo(inum3) = 0.
          wuw(inum3) = 0.
          wssf(inum3) = 0.
          wppq(inum3) = 0.
          wyd(inum3) = 0.
          wgp(inum3) = 0.
          wwn(inum3) = 0.
          wwpo(inum3) = 0.
          wgwq(inum3) = 0.
          if (ipd.eq.1) then
            write (77,5100) inum3, nbgis(j), nbigs(j),
     *          ida, precip, sml, qd, wysb,
     *          eo, uw, yd / (100.*da*rnum1), yon, yph, yno3 + ssfn, ysp
          end if
        end if
c*****write daily file .sbs file unit=3
        if (ipd.eq.1) then
          write (3,5000) j, nbgis(j), nbigs(j), nmgt(j), ida, da*flu(j),
     *      precip, snow, sml, qi, ssf,
     *      gwq(j), revap, gwseep, gwchrg, wysb, sep, uw, qtl, yd / (
     *      100.*da*flu(j)), evflux, spflux, rlflux, qlflux, pq, oflux,
     *      yp, ssub(22,inuml),
     *      ssub(23,inuml), ssub(24,inuml), eo, ssub(26,inuml),
     *      ssub(27,inuml), (1.-ws(j)), (1.-ts), (1.-strsn), (1.-strsp),
     *      yon, yph, yno3, ssfn, ysp, uno3, percn, uap, rmnltl, roctl,
     *      fn(nro(j),nfert(j),j), fph(nro(j),nfert(j),j), wfx, wdntl,
     *      hmntl, rwntl, hmptl, rmn2tl, rmptl, sw(j), dm(j), alai(j)
      end if
      return
c5000 format ('SUBBAS',i4,i9,3i5,53f10.3)
c5000 format ('SUBAS ',i4,1x,i8,1x,i4,1x,i4,1x,i4,62f10.3) ! for Swat 94.2
 5000 format ('SUBAS ',i4,1x,i8,1x,i4,1x,i4,1x,i4,62f15.3) ! changed by spp
 5100 format ('BIGSUB',i4,i9,2i5,14f10.3)
      end
```

Subr. Capflow: simulate uptake from shallow gw into soil profile (iaqufr=2)

```
      subroutine Capflow (j,nn,ml,mb,st,ul,fc,evtgw,egwuna)
c
c       test code for simulating uptake from shallow water table --spp
c       HRU codes:
c iaqufr=0:soil is underlain by bedrock; percolation out of root zone is
c          blocked in subr Purk18, increasing lateral flow, evap., and
c          soil water content.
c       =1: soil is underlain by deep aquifer; percolation flows to aquifer.
c       =2: soil is underlain by shallow aquifer; water evaporates from
c           shallow ground water.

      dimension st(ml,mb)  !available soil water content by layer & subbasin
      dimension ul(ml,mb)  !drainable porosity by layer and subbasin
      dimension fc(ml,mb)  !available capacity by layer and subbasin

      egwrem=evtgw
      do k = nn,1,-1
        if (st(k,j).lt.ul(k,j)) then
          swrise = AMIN1 (egwrem, fc(k,j) - st(k,j)) !uptake
ccccc     swrise = AMIN1 (egwrem, ul(k,j) - st(k,j)) !uptake
          st(k,j) = st(k,j) + swrise
          egwrem = egwrem - swrise
        end if
      end do
c         accumulate evap-gw not redistributed over soil profile; this
c         represents a loss not accounted for in the overall watershed
c         balance (dS/dt = precip - evap - yield + gwnet).
      if (egwrem.gt.0.) egwuna = egwuna + egwrem
      return
      end
```

Subr. Purk18: percolation, modified to represent soil over bedrock (iaqufr=0, *.cod)

```
cspp $debug
      subroutine purk18(j,iaqufr)
C     THIS SUBROUTINE IS THE MASTER PERCOLATION COMPONENT.  IT DIVIDES
C     EACH LAYER'S FLOW INTO 4 MM SLUGS AND MANAGES THE ROUTING PROCESS
C     CALLS PERC AND PKRN
c
C******common.f
      include 'common.f'
      dimension vvv(11)
      data amt /4./
      n2 = 0
      prk = 0.
      do 10 jj = 1, nn
        flat(jj,j) = 0.
        po(jj,j) = 0.
```

```
   10 continue
        vvv(1) = st(1,j) - fc(1,j)
        if (vvv(1).ge.0.) then
          st(1,j) = fc(1,j)
        else
          vvv(1) = 0.
        end if
        vvv(1) = vvv(1) + rain
        if (vvv(1).gt.0.) then
          n1 = 1
          n2 = 1
        end if
        do 20 k = 2, nn
          vvv(k) = st(k,j) - fc(k,j)
          if (vvv(k).gt.0.) then
            st(k,j) = fc(k,j)
            if (n2.gt.0) go to 20
            n2 = 1
            n1 = k
          else
            vvv(k) = 0.
          end if
   20 continue
        vvv(nn+1) = 0.
        tot = 0.
        add = 0.
   30 if (n2.ne.0) then
          n2 = 0
          do 40 j1 = n1, nn
            sum = 0.
            j2 = j1 + 1
            if (j1.eq.nn) j2 = nn
            if (vvv(j1).gt.0.) then
              if (n2.le.0) then
                n2 = 1
                n3 = j1
              end if
              su = vvv(j1)
              if (su.gt.amt) su = amt
              st(j1,j) = st(j1,j) + su
              sep = 0.
c             call pkrnl7(j)
C             SEP = PERCOLATION BY CRACK FLOW
              st(j1,j) = st(j1,j) - sep
              sum = sep
              if (su.ne.sep) then
                if (st(j1,j).gt.fc(j1,j)) then
                  call perc16(j)
C                 SEP = PERCOLATION
C                 PRK = LATERAL SUBSURFACE FLOW
c
c                 Change to represent the case where the soil profile is
c                 underlain by bedrock instead of a vadose zone and aquifer:
c                 set percolation out of bottom soil layer to zero.
c                 This case is indicated by iaqufr=0; iaqufr is given by
c                 iaqufr = idxaqf(i) for HRU i as given by the modified *.cod
c                 input file read by code on Swatmod2.h, included in Main.--spp

                  if (j1.eq.nn.and.iaqufr.eq.0) sep = 0.
                  st(j1,j) = st(j1,j) - sep - prk
                  sum = sum + sep
                  add = add + prk
                end if
                flat(j1,j) = flat(j1,j) + prk
                po(j1,j) = po(j1,j) + sep
              end if
              vvv(j1) = vvv(j1) - su
              j2 = j1 + 1
              vvv(j2) = vvv(j2) + sum
            end if
   40    continue
```

```
        tot = tot + sum
        n1 = n3
        go to 30
      end if
      sep = tot
      prk = add
      return
      end
```

Subr. Evap8: modified to include additional Penman option (ipet=3, *.cod)

```
      subroutine evap8 (j,julday,ioevt,iprevt)
C     THIS SUBROUTINE COMPUTES THE AMOUNT OF SOIL EVAPORATION AND THE
C     POTENTIAL PLANT EVAPORATION USING RITCHIE'S MODEL
c        Note: modified to include additional Penman evaporation option --spp
C******common.f
      include 'common.f'
      dimension tsoil(mb)
      data esd /300./, cej / -5.e-5/, u /6./, sumet /0./, sumeo /0./

c     ncrop = ncr(nro(j),icr(j),j)

      p = precip - qd
      eaj = exp(cej*(cv(j)+.1))
      tk = tx(j) + 273.
      tkk = tk * tk
c     d = exp(21.255-5304./tk) * 5304. / tkk
c     gma = d / (d+.68)
      if (sno(j).le.5.) then
        alb = salb(j)
        if (alai(j).gt.0.) alb = .23 * (1.-eaj) + salb(j) * eaj
      else
        alb = .6
      end if

C     tx        -      average daily temp
c     pb        -      barometric pressure
c     igro      -      flag to determine if crop growing
c     cht       -      canopy height
c     u10       -      wind speed at 10m
c     vpd       -      vapor pressure deficit
c     vpth     -       vapor pressure threshhold (crop parameter)
c     vpd2     -       second point for determining deficit
c     gsi       -      stomatal conductance/resistance
c     co2       -      co2 concentration in ppm
c     ea        -      saturated vapor pressure
c     ed        -      sat. vap. pres. corrected for humidity
c     alai      -      leaf area index
c     rc        -       canopy resistence
c     rv        -
c     rho       -
c     rhd       -      relative humidity (fraction) (blank to generate)

      pb = 101.3 - elev(j) * (0.01152-5.44e-7*elev(j))
      gma = 6.595e-4 * pb
      xl = 2.501 - 2.2e-3 * tx(j)
      ea = 0.1 * exp(54.879-5.029*alog(tk)-6790.5/tk)
      ed = ea * rhd(j)
      vpd = ea - ed
      dlt = ea * (6790.5/tk-5.029) / tk
      xx = dlt + gma

      ralb1 = ra(j) * (1.0-alb)
      rbo = (0.34-0.14*sqrt(ed)) * 4.9e-9 * (tk**4)
      rto = ra(j) / rmx
      rn = (ralb1*.0419-rbo*(0.9*rto+0.1))
      x2 = rn * dlt
c     ee(tk) = 0.1 * exp(54.879 - 5.029 * alog(tk) - 6790.5 / tk)
```

```
c
c     CONVERT DEW POINT TO RELATIVE HUMIDITY
c
c     uplm = dewpt(1)
c     do 1234 mo = 2, 12
c     if (dewpt(i) .gt. uplm) uplm = dewpt(mo,j)
c1234 continue
c
c     if (uplm .ge. 1) then
c     do 1235 i= 1, 12
c     rhmo(mo,j) = ee(dewpt(mo,j) + 273.0) / ee(tx(j) + 273.0)
c1235 continue
c
c     endif
c-------------------------------------------------------------------
c         Added option: Penman evaporation --spp
c
      if (ipet.eq.3) then
c           use calculated temperature values for top two soil layers to
c           approximate soil temperature at 10 cm depth:
          tsprev = tsoil(j)
          tsoil(j) = (2.*t(1,j) + t(2,j))/3.
          if (tsprev.gt.0.) then
            dtsoil = tsoil(j) - tsprev
          else
            dtsoil = 0.
          end if

          call PENMAN (j,iyr,julday,alb,elev(j),pb,ylt(j),tmn(j),
     1        tmx(j),tx(j),dtsoil,ra(j),u10(j),rhd(j),vpd,eo,
     1        ioevt,iprevt)

      else if (ipet.eq.2) then

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     C
C     HARGREAVES POTENTIAL EVAPOTRANSPIRATION METHOD               C
C     C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        ramm = rmx * 0.0419 / xl
        eo = .0032 * ramm * (tx(j)+17.8) * (tmx(j)-tmn(j)) ** 0.6

        go to 10

      else

        if (ipet.eq.1) then


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C         C
C         PENMAN-MONTEITH POTENTIAL EVAPOTRANSPIRATION METHOD              C
C         C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


          rho = 0.01276 * pb / (1.0+0.00367*tx(j))
          if (u10(j).lt.0.01) u10(j) = 0.01
          if (igro(j).le.0) then
            ep = 0.0
          else

            if (cht(j).le.8.0) then
              uzz = u10(j)
              zz = 10.0
            else

              zz = cht(j) + 2
              uzz = u10(j) * alog(zz/0.0005) / 9.9035
            end if

            x1 = alog10(cht(j)+0.01)
            zo = 10.0 ** (0.997*x1-0.883)
            zd = 10.0 ** (0.979*x1-0.154)
            rv = 6.25 * (alog((zz-zd)/zo)) ** 2 / uzz
            x3 = vpd - vpth(ncr(nro(j),icr(j),j))
            if (x3.gt.0.0) then
              fvpd = amax1(0.1,1.0-vpd2(ncr(nro(j),icr(j),j))*x3)
            else

              fvpd = 1.0
            end if

            g1 = gsi(ncr(nro(j),icr(j),j)) * fvpd
            if (alai(j).lt..05) then
              alai(j) = alai(j) + .05
            end if
            rc = 2.0 / ((alai(j)+0.01)*g1*(1.4-0.4*co2(j)/330.0))
            ep = (x2+86.66*rho*vpd/rv) / (x1*(dlt+gma*(1.0+rc/rv)))
          end if

          rv = 350.0 / u10(j)
          eo = (x2+86.66*rho*vpd/rv) / (x1*xx)
c         write(70,2000)ida,eo,ep
c2000     format(i4,2x,2f8.3)

          if (ep.gt.eo) eo = ep
          go to 10

        else

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c         c
c         PRIESTLEY - TAYLOR POTENTIAL EVAPOTRANSPIRATION METHOD    c
c         c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


          gma = d / (d+.68)
          d = exp(21.255-5304./tk) * 5304. / tk ** 2
          ho = ra(j) * (1.-alb) / 58.3
          aph = 1.28
C         EO = POTENTIAL ET (MM)
C         ES = SOIL EVAPORATION (MM)
C         EP = PLANT TRANS. (MM)
          eo = aph * ho * gma
c       sumeo = sumeo + eo


        end if
      end if

   10 eos = eo * eaj
      if (s1(j).ge.u) go to 40
      sp = s1(j) - p
      if (sp.gt.0.) then
        s1(j) = sp
        go to 30
      end if
   20 s1(j) = 0.
   30 s1(j) = s1(j) + eos
      su = s1(j) - u
      if (su.le.0.) then
        es = eos
        go to 50
      end if
      es = eos - .4 * su
      s2(j) = .6 * su
      tv(j) = (s2(j)/3.5) ** 2
      go to 50
   40 sb = p - s2(j)
      if (sb.ge.0.) then
        p = sb
        s1(j) = u - p
        tv(j) = 0.
        if (s1(j).lt.0.) go to 20
        go to 30
      end if
      tv(j) = tv(j) + 1.
      es = 3.5 * sqrt(tv(j)) - s2(j)
      if (p.le.0.) then
        if (es.gt.eos) es = eos
      else
        esx = 0.8 * p
        if (esx.le.es) esx = es + p
        if (esx.gt.eos) esx = eos
        es = esx
      end if
      s2(j) = s2(j) + es - p
      tv(j) = (s2(j)/3.5) ** 2
   50 if (es.le.0.) es = 0.
      if (ipet.ne.1) then
        if (alai(j).le.3.0) then
          ep = alai(j) * eo / 3.
        else
          ep = eo
        end if
      end if
      et = es + ep
      if (eo.lt.et) then
        et = eo
        es = et - ep
      end if
      xx = es
      if (sno(j).lt.es) then
        xx = xx - sno(j)
        snoev = snoev + sno(j) * flu(j)
        sno(j) = 0.
        do 60 j2 = 1, nn
          if (z(j2,j).gt.esd) go to 80
          if (st(j2,j).gt.xx) go to 70
          xx = xx - st(j2,j)
          st(j2,j) = 0.
   60   continue
        go to 90
   70   st(j2,j) = -xx + st(j2,j)
        go to 110
   80   j1 = j2 - 1
        yy = 0.
        if (j1.gt.0) yy = z(j1,j)
        rto = st(j2,j) * (esd-yy) / (z(j2,j)-yy)
        if (rto.gt.xx) go to 100
        xx = xx - rto
        st(j2,j) = st(j2,j) - rto
   90   es = es - xx
        et = et - xx
        go to 110
  100   st(j2,j) = st(j2,j) - xx
      else
        sno(j) = sno(j) - es
        snoev = snoev + es * flu(j)
      end if
  110 potet(j) = potet(j) + eo
      pet(j) = eo
      sumet = sumet + et
      sumeo = sumeo + eo
c     write (70,1000)ida,eo,et,ep,es,sumet,alai(1),dm(1),sumeo
c1000 format (i4,2x,6f8.3,f10.2,f8.2)
      return
      end
```

Subr. Penman: calculate Penman evaporation (file penman.for)

```
c file penman.for  spp aug 94  calculate pot. evaporation
c rev. 1995 to be called by SWAT

      subroutine PENMAN (isub,iyr,julday,albedo,elev,patm,phideg,
     1  tmin,tmax,tavg,dtsoil,stlang,u10,relhum,vpdef,etref,
     1  ioevt,iprevt)

cccc 1    rs,ra,snj,rlong,shflux,etref)
c       components of reference et: (1) short wave (incoming),
c       (2) long wave (emissions), (3) soil heat flux, (4) vapor pressure deficit.
      dimension etcomp(4)

c usage in subr Evap8 (Swat):
cc        call PENMAN (j,iyr,julday,alb,elev(j),pb,ylt(j),tmn(j),
cc   1        tmx(j),tx(j),dtsoil,ra(j),u10(j),rhd(j),vpd,eo)
c
c Note: If internally set itrace = isub (subbasin index value),
c then daily values are written to file with device no. ioevt=98.
c
c Compute evaporation estimates (mm/day); numbered equations refer to
c Handbook of Hydrology, 1993, Maidment, ed., especially Ch. 4, Evaporation,
c Shuttleworth; see also sec. 13.2, pp. 13.22-13.25.
c      Notes: relative humidity value is assumed to be available.
ccc   iemiss=0: assume rel.humidity is available for estimating emissivity;
c     otherwise (iemiss=1), estimate on basis of temperature.
c     Here, rel. humidity is simulated, whereas temperature is measured.
c
c Arguments:
c  IN int isub   subbasin index
c  IN int iyr    calendar year
c  IN int julday Julian day
c  IN re  albedo reflectivity
c  IN re  elev   land surface elevation (m); used to estimate atmos. pressure, patm
c OUT re  patm   atmospheric pressure (kPa)
c  IN re  phideg site latitude (decimal degrees): convert internally to phirad (radians)
c  IN re  tmin   day's minimum temperature (deg C)
c  IN re  tmax   day's maximum temperature (deg C)
c  IN re  tavg   day's avg temperature (deg C)
c  IN re  dtsoil change from previous day in soil temperature (deg C) at 0.1 m depth
c  IN re  stlang incoming terrestrial solar radiation (at land surface (langleys).
c              NOTE: converted internally to stj (MJ/day)
c  IN re  u10     day's avg wind speed (m/s)  measured at 10 m ALS;
c              NOTES: a) use this to approx. u2, wind speed at 2 m ALS, assuming
c                        logarithmic wind speed profile and u2 = 0.749*u10.
c                     b) set minimum at 1 m/s (arbitrary)
c  IN re  relhum relative humidity (fraction)
c OUT re  vpdef  vapor pressure deficit
c OUT re  etref reference crop evaporation (4.2.31 or 4.4.14), version of the
c           Penman-Montheith equation recommended by the Handbook of Hydrology
c  IN int ioevt    device no. initialized in Swatmod1 for file opened in
c                  Swatmod3.h (included in Main)
c  IN int iprevt option to print daily evaporation results if iprevt = isub,
c                for subbasin isub.
c
c internal:
c  IN int itrace indicate whether (y=isub,n=0) to write diagnostics
c OUT re (etcomp(j),j=1,4): components of reference crop evaporation, etref:
c            j=1: short wave component;
c            j=2: long wave component;
c            j=3: soil heat flux component;
c            j=4: vapor pressure deficit component.
c
c      removed arguments:
c int iemiss  0: relative humidity (%) is available, and is used to estimate
c                emissivity; otherwise (iemiss=1), temperature would be used
```

```
c                to estimate emissivity.
c re  hc      crop height (m): used to evaluate actual et by (4.2.27): see et (below).
c          NOTE: now set internally as 0.12 m.
c re  dxleaf leaf area index; if input as zero, calculated by 4.2.23 for grass.
c re  cfcref coefficient replacing the value 1.26 in (4.2.38) req'd for eprta
c               to equal epmref (above).
c re  s0j     day's extraterrestrial incoming radiation (MJ/m^2/day) through a
c               plane tangent to a sphere enclosing the earth's atmosphere.
c re  epot    potential open water evaporation (4.2.30)

c Checks and notes on internal variables:
      data itrace /0/ !internal option to print summary of results at end
cccc  data ioevt /98/ ! i/o device no. set to match the value set in
c                       Swatmod1.h for the file opened in Swatmod3.h.
      data iemiss /0/ !estimate emissivity on basis of relative humidity.
c ra           aerodynamic resistance (4.2.25; see also eqn 13.2.1) checks
c              for wind speed u2=5 m/s:
c                  crop      mean height hc (m)    ra (s/m)
c                  grass           0.1             45
c                  agricultural    1.              18
c                  forest         10.               6.5
c daymax       max. possible daylight hours (N, eqn 4.4.1) accuracy is within 0.1 h.
c
c s0m          extraterrestrial solar radiation S0 in MJ/m^2/day was backed out
c              from (4.4.4) by multiplying by lambda (i.e. htvap): calculated
c              to compare with s0j above.  Note that (4.4.4) must assume a
c              nominal value for heat of vaporization, so that s0j (above)
c              should be more rigorous than s0m.  A comparison of these two
c              for 1993 Sandyland data (actually it's independent of these data)
c              showed a mean error 100*(s0m-s0j)/s0j of -1.2 pct ranging from
c              -2.6 pct to 0.6 pct (small compared to other error sources
c              in evaporation calculation). S0 (mm/day) by (4.4.4) is accurate
c              to within 0.1 mm.
c                  Checks from Maidment (p.4.31) for April 15 (Julian day=105):
c                  latitude (deg)    N (daymax, h)    S0 (mm/day)
c                      30 N              12.7            15.0
c                       0                12.0            15.1
c                      30 S              11.3            11.2
c-----------------------------------------------------------------------------
      real etref
ccc   character*60 refnam
ccc   data refnam /'(4.2.31) Penman-Monteith reference crop evap'/
c         coefficients for estimating incoming solar radiation st given
c         extraterrestrial solar radiation s0 and fraction of cloud cover n/N:
      data as, bs /0.25, 0.50/
c         est. albedo (reflectance): (now pass as argument)
cccc  data albedo /0.23/
c         Stefan-Boltzmann constant (MJ/m^2/K^4) for long-wave radiation:
      data stefan /4.903e-9/
c         solar constant, langleys/min:
      data gsc /1.957/
c         convert radiation in langleys (cal/cm^2) to MJ/m^2:
c             (note: based on 1 J = 0.239 cal)
      data radcnv /23.9/
c         soil heat capacity (MJ/(m^3*C)) for average moist soil (Handbook,p.4.10):
      data cpsoil /2.1/
c         presumed soil depth (m) for temperature change dTsoil:    (4.2.17)
      data dpsoil /0.15/
c             specific heat of moist air (kJ/(kg*C)) (p. 4.13):
      data cp /1.013/
c             mixing ratio (check term): ratio of molecular weight of water
c             vapor to that for dry air:
      data epsiln /0.622/
c         coefficient for psychrometric constant gamma: pscoef=1.e-3*cp/epsiln
      data pscoef /1.6286e-3/
      data iopchk /0/ !option to write out table with entries like T. 4.4.3
      data initlz /0/
c------------------------------
c         saturation vapor pressure (kPa) given temperature (deg C): (4.2.2)
c         (approximation for Clausius-Clapeyron eqn (3.3.3))
      satvp(T) = 0.6108*EXP(17.27*T/(237.3+T))
```

```
c              evaluate derivative delta = d(es)/dT at T (kPa/degC):  (4.2.3)
      desdt(es,T) = 4098.*es/(237.3+T)**2
c          estimated density (kg/m^3) of moist air (eqn 4.2.4, Shuttleworth,
c          Ch. 4, Evaporation, Handbook of Hydrology, ed. by Maidment '93), for
c          atmospheric pressure P (kPa) and air temperature T (deg C):
      airden(P,T) = 3.486*P/(275.+T)
c          estimated atmospheric pressure P (kPa) as a function of elevation z (m)
c          leaf area index for clipped grass of height 0.05 < hc < 0.15 m:
      dxgras(hc) = 24.*hc
c          leaf area index for alfalfa of height 0.10 < hc < 0.50 m:
      dxalfa(hc) = 5.5 + 1.5*ALOG(hc)
c          nominal atm. pressure at site (kPa): z=elev (m)          (4.4.12)
      pnom(z) = 101.3*((293.-0.0065*z)/293.)**5.256
c          (The above function is used in the calling routine ETSAND)
c          pressure conversions:
c          1 atm = 101.32 kPa (kPa = kilopascal; 1 pascal = 1 N/m^2)
c                = 1013.2 millibars = 760 mm Hg = 10.33 m water
c                = 14.7 psia      = 29.92 in Hg = 33.9 ft water
c
      if (initlz.eq.0) then
          initlz = 1
          pi = 4.*ATAN(1.)
          twopi = 2.*pi
          degrad = 180./pi      !deg/radian
          gscday = gsc*24.*60.
c              solar constant (MJ/m^2/day):
          gscj = gsc*24.*60./radcnv
          govrpi=gscj/pi
c              coef. used in (4.2.17-18) for daily change in soil heat content:
          cpdp = cpsoil*dpsoil
          print 110,'solar constant (MJ/m^2/day) gscj=',gscj,
     1        'gscj/pi=',govrpi
          print 110,'soil heat flux coef cp*dp in (4.2.17) =',cpdp
      end if
c-------------------------------------------------------
cccc  conversions to tailor argument list to Swat's units:
c            synthesized wind speed is converted from 10m to 2m for subr PENMAN
c            in subr clicon (file cliconsp.for); if wind speed was read there
c            (iopwea>0), then it was assumed to be u10 (10m above ground).
          u2 = 0.749*u10          !approx. wind speed at 2 m ALS
          phi = phideg/degrad  !site latitude: deg to radians
c              revised Penman so that relative humidity is given as a fraction
ccccc     relhum = rhd(j)
c--------------------------------------------------------
c              convert terrestrial solar radiation from langleys (cal/cm^2)
c              to MJ/m^2 (based on 1 J = 0.239 cal):
          stj = stlang/23.9
cccc
c              nominal atmospheric pressure (kPa):
      patm = PNOM(elev)
c              mean daily air temperature (deg K):
      tavgk = tavg + 273.2
c              julday = Julian day
      djulia = julday
      yrfrac = djulia/365.
c          dr = earth-sun distance as fraction of (mean?)            (4.4.5)
      dr = 1.+0.033*COS(twopi*yrfrac)
c              solar declination (radians)                           (4.4.3)
      dlt = 0.4093*SIN(twopi*yrfrac - 1.405)
c              sunset hour angle (radians):                          (4.4.2)
      ws = ACOS(-TAN(phi)*TAN(dlt))
c              max. possible daylight hours (N):                     (4.4.1)
      daymax = 24.*(ws/pi)
c
c              extraterrestrial solar radiation S0 (MJ/m^2/day):     (ce751)
      s0j = (ws*SIN(phi)*SIN(dlt) + COS(phi)*COS(dlt)*SIN(ws))*
     1      dr*gscday/(pi*radcnv)
c
c              htvap = latent heat of vaporization (MJ/kg) (lambda):  (4.2.1)
c                      base this on mean daily temperature tavg (deg C),
c                      but is defined in terms of Ts, water surface temperature.
```

```
      htvap = 2.501 - 0.002361*tavg
c             psychrometric constant (kPa/deg C):                  (4.2.28)
      gamma = pscoef*patm/htvap
c          day's avg saturated water vapor pressure (kPa) (see p. 4.33):
      esmin = SATVP(tmin)
      esmax = SATVP(tmax)
      esavg = (esmin + esmax)/2.
c             estimate avg vapor pressure deficit (kPa):             (4.4.6)
      vpdef = esavg*(1.-relhum)

c             evaluate avg derivative delta = d(es)/dT at T (kPa/degC)   (4.2.3)
      delta = (DESDT(esmin,Tmin) + DESDT(esmax,Tmax))/2.
c
c     To convert solar radiation from MJ/m^2/day to mm/day, divide by
c     htvap, the latent heat of vaporization (MJ/kg), (4.2.1).

c     Explanation (p. 4.8): for net radiation Rn (MJ/m^2/day), the
c     equivalent depth of evaporated water is Rn/(rho_w*lambda) (m),
c     where rho_w= density of water (approx. 1000 kg/m^3) and
c           lambda=latent heat of vaporization (htvap, above, MJ/kg).
c     So the equivalent evaporated depth (mm) is (1000/rho_w)*Rn/lambda,
c     or approximately Rn/lambda.
c
c     Extraterrestrial solar radiation, s0m (MJ/m^2/day), is backed out
c     from (4.4.4) by multiplying by lambda (i.e. htvap); see
c     notes on s0m in argument list description.
c
cccc  s0m = htvap*15.392*dr*
cccc 1          (ws*SIN(phi)*SIN(dlt) + COS(phi)*COS(dlt)*SIN(ws))

c             absorbed short-wave radiation:                       (4.2.5)
      snj = (1.-albedo)*stj

c         intermediate calculations for net long-wave radiation:
c             est. fraction of available extraterrestrial solar radiation:
      radfrc = stj/s0j
c             estimate fraction of cloud cover dayfrc (n/N) based on st and s0:
c             rearrange equation used to estimate st = s0*(as + bs*n/N) (4.2.6)
c             into the form  [(st/s0) - as]/bs = n/N; nominal values for
c             coefficients as, bs are in data statement.  Limit this fraction
c             to the range [0,1].
      dayfrc = AMIN1(1., AMAX1(0., (radfrc - as)/bs ))
c             cloudiness factor f using nominal coefficients:      (4.2.12)
      f = 0.9*dayfrc + 0.1
c         use the alternate calculation (4.2.9) if relative humidity isn't
c         available; e.g. for Swat, temperature comes from data, but relative
c         humidity was simulated.
      if (iemiss.eq.0) then
c             unsaturated vapor pressure: from def. RH = e(T)/es(T):
         eunsat= esavg - vpdef
c             est. net emissivity from vapor pressure (kPa):        (4.2.8)
         emiss = 0.34 - 0.14*SQRT(eunsat)
      else
c            alt: est. net emissivity from avg temperature (deg C):  (4.2.9)
         emiss = -0.02 + 0.261*EXP(-7.77e-4*tavg*tavg)
      end if
c             net long-wave radiation (MJ/m^2/day):                 (4.2.7)
      rlong = -f*emiss*stefan*tavgk**4

c             est. soil heat flux (MJ/m^2/day):                    (4.2.18)
      shflux = cpdp*dtsoil
c         energy available for evaporation (mm/day) = net radiation - soil heat flux
c             net radiation (MJ/m^2/day) = snj + rlong;            (4.2.13)
      avail = (snj + rlong - shflux)/htvap
c             available energy coefficient for potential evaporation from
c             open water (4.2.30), for the Priestley-Taylor eqn (4.2.38-39),
c             and the Kimberly Penman combination equation (4.2.33):
      f1p = delta/(delta+gamma)

c             evaluate reference crop evaporation:                 (4.2.31)
c             coef's f1rc and f2rc are both functions of (T,u2,z).
```

```
c             variation on psychrometric constant gamma (above):    (4.2.32)
      gstar = gamma*(1.+0.33*u2)
c             available energy coef.:                               (4.2.15)
      f1rc = delta/(delta+gstar)
c             vapor pressure deficit coef.:                         (4.2.16)
      f2rc = (gamma/(delta+gstar))*900.*u2/(tavg+275.)
c             est. reference crop evaporation (mm/day):             (4.4.14)
      etref = f1rc*avail + f2rc*vpdef
c
      if (isub.gt.0.and.iprevt.eq.isub) then
c             ref. et components:
        etcomp(1) = f1rc*(snj/htvap)     !short wave contribution (mm/day)
        etcomp(2) = f1rc*(rlong/htvap)   !long wave contribution (mm/day)
        etcomp(3) = f1rc*(-shflux/htvap) !soil heat flux contribution (mm/day)
        etcomp(4) = f2rc*vpdef  !vapor pressure deficit contribution (mm/day)

        write (ioevt,'(3i5,f8.1,f8.2,f8.3,f8.1,f8.2,f8.3,5f8.2)')
     1     isub,iyr,julday,stlang,u10,relhum,tavg,dtsoil,albedo,
     1     etref,(etcomp(j),j=1,4)

c example from file carr-irm.evt:
c sub Year  Jul   rad,ly u10,m/s rh,frac tmpav,C dTsol,C  albedo ETPOTmm ETshort  ETlong
ETsoil ETvpdef
c    1 1977    0   160.2     2.7   0.923   -13.3    0.00   0.150    0.22    0.35   -0.19
0.00    0.06

cc      hdrout = '  sub Year  Jul  rad,ly u10,m/s rh,frac tmpav,C'//
cc    1     ' dTsol,C  albedo ETPOTmm ETshort  ETlong  ETsoil ETvpdef'
cc    subroutine PENMAN (isub,iyr,julday,albedo,elev,patm,phideg,
cc    1  tmin,tmax,tavg,dtsoil,stlang,u10,relhum,vpdef,etref)

      end if
c     if (itrace.gt.0) then
c         print 110,'emiss(e)=',emiss,'iemiss=',iemiss
c         print 110,'snj=',snj,'MJ/m^2/d, rlong=',rlong,'MJ/m^2/d'
c         print 110,'shflux=',shflux,'MJ/m^2/d, avail=',avail,'mm/d'
110       format (5(1x,a,f10.3))
c     end if
      return
      end
```

Subr Crpmd6: potential plant growth; modified irrigation model.

Note: this modified routine is on file crpmdspp.for.

```
* file crpmdspp.for  spp revised subr crpmd6.

      subroutine crpmd6 (j,julday,ioirr,iprirr,iopswm,wsfmax,swminf)


C     THIS SUBROUTINE PREDICTS DAILY POTENTIAL GROWTH OF TOTAL PLANT
C     BIOMASS AND ROOTS AND CALCULATES LEAF AREA INDEX.  INCORPORATES
C     RESIDUE FOR TILLAGE FUNCTIONS AND DECAYS RESIDUE ON GROUND
C     SURFACE.  CALLS SUBROUTINE SWU AND ADJUSTS DAILY DRY MATTER TO
C     WATER STRESS CALCULATED IN SWU.
c
c  subroutine argument list:
c  j = subbasin index
c  iopswm: option for automatic irrigation (irr(j) = 1:
c    =1: use threshold on soil water content, sw(j) < swminf;
c    =0: use threshold on plant water stress factor, ws(j) < wsf(j).
c  wsfmax = daily max. added water (mm/operation) for auto irrig; set in ~.mco.
c  swminf = irrigation threshold for soil water content as fraction of
c      available field capacity. Calculation in Crpmd6 for the
c      growing season (igro(j)=1):  swtrig = swminf*sumfc(j) is the
c      minimum soil moisture (total in profile, mm)
c      below which irrigation will be applied if option irr(j)=3, set
c      in the ~.mco file.
```

```
c
C******common.f
      include 'common.f'
c       if itrace = j > 0, write daily summary of irrigation for subbasin j
      data itrace /0/ !write results for subbasin j

      xx = t(2,j)
      sut = st(2,j)/fc(2,j)
      cdg = .9 * xx / (xx + exp(9.93 - .312 * xx)) + .1
      decr = .05 * amin1(cdg, sut)
      rsd(j) = rsd(j) * (1. - decr)
      uap = 0.
      if(tx(j).gt.0.) gplt(j) = gplt(j) + tx(j) / phutot
      cv(j) = .8 * dm(j) + rsd(j)
      if(igro(j).eq.0) then
        if(ipl(nro(j),icr(j),j).gt.0) then
          if(ida.eq.ipl(nro(j),icr(j),j)) go to 10
        else
          if(gplt(j).gt.phup(nro(j),icr(j),j)) go to 10
        endif
        go to 11
c*****planting
   10    igro(j)=1
         g(j) = 0.
         dm(j) = 0.01
         snup(j) = 0.
        swh = 0.
        swp = 0.

         spup(j) = 0.
         hufh(j) = 0.
         hia(j) = 0.
         olai(j) = 0.
         rwt(j) = 0.
         if (cnop(nro(j),icr(j),j).gt.0.)
     *           call curno(cnop(nro(j),icr(j),j),j)
      endif
   11 if(igro(j).eq.1) then
c*******check if day of harvest and kill
c
        if (dayl-1. .lt. daylmn(j) .and.
     *     ird(ncr(nro(j),icr(j),j)).eq.1) go to 12
        if(ihv(nro(j),icr(j),j).gt.0) then
          if(ida.eq.ihv(nro(j),icr(j),j)) go to 12
        else
          if(g(j).gt.phuh(nro(j),icr(j),j)) go to 12
        endif
        go to 17
c*****harvest and kill
   12   igro(j)=0
        g(j) = 0.
        if(hiadl(j) .gt. hi(ncr(nro(j),icr(j),j)))
     *     hiadl(j) = hi(ncr(nro(j),icr(j),j))
        rsd(j) = (1.-rwt(j))*(1.-hiadl(j))*dm(j)+rsd(j)
        fon(j) = rsd(j) * .008
        fop(j) = rsd(j) * .0011
        if (hitar(nro(j),icr(j),j).gt.0.)
     *    hiadl(j) = hitar(nro(j),icr(j),j)
        yield = (1.-rwt(j)) * dm(j) * hiadl(j)
        yld(nro(j),icr(j),j) = yld(nro(j),icr(j),j) + yield
        dmba = dmba + dm(j) * flu(j)
        dm2(nro(j),icr(j),j) = dm(j) + dm2(nro(j),icr(j),j)
        yldanu(j) = yldanu(j) + yield / 1000.
        dmanu(j) = dmanu(j) + dm(j)/ 1000.
        ncrops(nro(j),icr(j),j) = ncrops(nro(j),icr(j),j) + 1
        if (cnop(nro(j),icr(j),j).gt.0.)
     *           call curno(cnop(nro(j),icr(j),j),j)
        dm(j) = 0.
        ws(j) = 1.
        alai(j) = 0.
        hia(j) = 0.
        icr(j)=icr(j)+1
c*******check if day of harvest only - no kill - cutting
c
   17   if(ihvo(nro(j),icr(j),j).gt.0) then
          if(ida.eq.ihvo(nro(j),icr(j),j)) go to 13
        else
          if(g(j).gt.phuho(nro(j),icr(j),j)) go to 13
      endif
      go to 19
c*****harvest only
   13   if(hiad1(j) .gt. hi(ncr(nro(j),icr(j),j)))
     *     hiad1(j) = hi(ncr(nro(j),icr(j),j))
        if (hitar(nro(j),icr(j),j).gt.0.)
     *    hiad1(j) = hitar(nro(j),icr(j),j)
*           daily addition to crop yield:
        yield = (1.-rwt(j)) * dm(j) * hiad1(j)    ! (211,manual)
*           Comparing this with eqns 209-212 in manual implies (I think):
*           rwt(j) = daily root fraction of total biomass (eqn 209, manual);
*           dm(j) = daily total biomass (kg/ha), day i (dm is Btot in manual);
*               Comparing yield eqn with (210), Btot must be given by
*           Btot = dm(j)*hiad1(j)
*           Bag = (1-rwt(j))*dm(j)   = total root biomass (210, manual);
*           yld = amount of crop removed from field (kg/ha) at harvest (211,man)
        yld(nro(j),icr(j),j) = yld(nro(j),icr(j),j) + yield
        dmba = dmba + yield * flu(j)
        dm2(nro(j),icr(j),j) = dm2(nro(j),icr(j),j) + dm(j)
        yldanu(j) = yldanu(j) + yield / 1000.
        dmanu(j) = dmanu(j) + yield/ 1000.
        xx = dm(j)
        dm(j) = dm(j) - yield
        alai(j) = alai(j)*dm(j)/xx
        g(j)=g(j)*dm(j)/xx
c*******check if day of kill
c
   19   if (dayl-1. .lt. daylmn(j) .and.
     *     ird(ncr(nro(j),icr(j),j)).eq.1) go to 18
        if(ikill(nro(j),icr(j),j).gt.0) then
          if(ida.eq.ikill(nro(j),icr(j),j)) go to 18
        else
          if(g(j).gt.phuk(nro(j),icr(j),j)) go to 18
        endif
        go to 16
c*****kill
   18   igro(j)=0
        g(j)=0.
        ncrops(nro(j),icr(j),j) = ncrops(nro(j),icr(j),j) + 1
        dmba = dmba + dm(j) * flu(j)
        dm(j) = 0.
        ws(j) = 1.
        alai(j) = 0.
        icr(j)=icr(j)+1
      endif
   16 if(iop(nro(j),nop(j),j).gt.0) then
        if(ida.eq.iop(nro(j),nop(j),j)) go to 14
      else
        if(g(j).gt.phut(nro(j),nop(j),j)) go to 14
      endif
      go to 15
c*****TILLAGE OPERATION
   14 rsd(j)=rsd(j)*(1.-effmx(nro(j),nop(j),j))
      if(cnop(nro(j),nop(j),j).gt.0)
     *         call curno(cnop(nro(j),nop(j),j),j)
      nop(j)=nop(j)+1
      go to 16

c     Allocate for plant use: (a) egwplt, any remaining evt from gw, and
c     (b) soil water;
c     compute water stress ws = actual plant use/potential plant use.
   15 continue

      call swu19 (j)
      actual = ep

c----------------------------------------------------------------------
c Daily limit wsfmax (mm) on automatic irrigation has been added (spp):
c----------------------------------------------------------------------
c     option added: soil water trigger for irr(j) = 1:
c     Note use of wsf(j):
c       irr(j) = 1 and iopswm = 0: wsf(j) = threshold on plant stress factor;
c       irr(j) = 1 and iopswm > 0: swminf = threshold on available water capacity.
c           During the growing season (igro(j) = 1 and g(j) > 0),
c       this is multiplied by sumfc(j), the soil water column corresponding
c       to field capacity to give the threshold as a column height.
c
      stsum=0.
      addirr = 0.
      addmax = 0.
      swtrig=0.
      swfrac = sw(j)/sumfc(j)

c         irrigation specified according to a schedule:
      if (irr(j).lt.0 .and. ida.eq.iai(nro(j),nirr(j),j)) then
          nirr(j) = nirr(j) + 1
          addmax = ai(nro(j),nirr(j),j)  !max addition specified for this day
c
c           irrigation triggered by threshold on either:
c              low plant stress factor ws(j) <= wsf(j) (option irr(j)=1), or
c              low soil water content, sw(j) < swtrig  (option irr(j)=3).
      else if (irr(j).ge.1 .and. igro(j).eq.1) then
          swtrig = sumfc(j)*swminf
          if (    (iopswm.gt.0 .and. sw(j).le.swtrig)      !soil water threshold option
     1        .or. (iopswm.eq.0 .and. ws(j).le.wsf(j)))    !plant stress threshold option
     1      addmax = wsfmax    !max daily irrigation (mm) for automatic case
      end if
c
c         Apply irrigation if the max is above 0.5 mm:
      if (addmax.gt.0.1) then
          do k = 1, nn
c               Add irrigation to layer k, up to the remainder of the
c               allotted daily maximum, addmax-sum, and the layer's
c               soil water deficit, fc(k,j) - st(k,j):
              if (addmax.gt.addirr) then
                addst = AMIN1(addmax-addirr, fc(k,j) - st(k,j))
                st(k,j) = st(k,j) + addst !accumulate this in layer k
                addirr = addirr + addst   !accumulate daily irrigation
              end if
              stsum = stsum + st(k,j)
          end do
          if (addirr.ge.0.1) then
              irn(j) = irn(j) + 1
              air(j) = air(j) + addirr
              ssub(22,j) = ssub(22,j) + addirr
          end if
c             diagnostics to show daily irrigation:  --spp
c         data initlz /0/
c         if (initlz.eq.0) then
c             initlz = 1
c             print '(a)', ' day  sub nlay'//
c    1          '   sw(j)  swtrig  wsfmax  addmax   stsum  addirr'
c         end if
c         print '(3i5,6f8.2)', iday,j,nn,
c    1          sw(j),swtrig,wsfmax,addmax,stsum,addirr
      end if
cc    hdrout='  sub Year Jul igro  pcp,mm  ETa,mm'//
cc   1          ' SW,mm swtrig,mm  irr,mm sw/sumfc'
c          unit device no. for daily irrigation results; also defined in
c          Swatmod1.h, and file initialized in Swatmod3.h.
cc        hdrout='  sub Year Jul igro  pcp,mm   ETpot ETplant'//
cc   1            ' ETsoil SW,mm swtrig,mm  irr,mm sw/sumfc      ws'
      if (j.gt.0 .and. iprirr.eq.j) write (ioirr,'(4i5,9f8.2)')
     1  j,iyr,julday,igro(j),subp(j),eo,ep,es,sw(j),swtrig,
     1  addirr,swfrac,ws(j)

       if(igro(j).ge.1) then
C        COMPUTE DAILY INCREASE IN HEAT UNITS, delg:
         hutmx = amin1(tx(j),to(ncr(nro(j),icr(j),j)))
*            delg = heat unit index (delg=HUI in manual), ranging from 0 at
*            planting to 1 at maturity (eqn 204, manual):
         delg = (tx(j)-tb(ncr(nro(j),icr(j),j)))/
     *          phu(nro(j),icr(j),j)
*            where phu = potential heat units req'd for maturity of crop.
         if (delg.lt.0.) delg = 0.
         g(j) = g(j) + delg
         tgx = tx(j) - tb(ncr(nro(j),icr(j),j))
         if (tgx.le.0.) then
           ts = 0.
         else
C          COMPUTE TEMPERATURE STRESS - TS
           call tstr7(tgx,j)
         end if
       if(ird(ncr(nro(j),icr(j),j)).eq.1) then
          rd(j) = 2.5 * g(j) * zmx(j)
          if(rd(j).gt.zmx(j)) rd(j)=zmx(j)
       else
          rd(j)=zmx(j)
       endif
         if (ida.lt.idsb.and.ida.gt.idse .and.
     *     g(j).le.dlai(ncr(nro(j),icr(j),j))) then


C          DM = TOTAL BIOMASS - KG/HA; dm=Btot as used in (210, manual)
C          G = FRACTION OF PHU ACCUMULATED; g = accumulated delg, where
*              delg = daily fraction of phu, given by (204, manual)
C          RWT = FRACTION OF ROOT WEIGHT (209, manual)
C          HIA = HARVEST INDEX (212, manual)


C              PAR = PHOTOSYNTHETIC ACTIVE RADIATION  (205, manual; 166, book)
           par = .02092 * ra(j) * (1.-exp(-.65*(alai(j)+.05)))

C             BEADJ = biomass energy adjustment for co2 concentration
       beadj = 100. * co2(j) / (co2(j) + exp(wac21(ncr(nro(j),icr(j),j))
     *- co2(j) *  wac22(ncr(nro(j),icr(j),j))))

C               ruedecl = decline in radiation use efficiency (crop specific)
          ruedecl = amax1(vpd - 1, -0.5)
          beadj = beadj - wavp(ncr(nro(j),icr(j),j)) * ruedecl
           if (beadj.gt.be(ncr(nro(j),icr(j),j)))
     *       beadj = be(ncr(nro(j),icr(j),j))

           ddm = beadj * par
           if (ddm.lt.0.) ddm = 0.

C          CALCULATE N AND P UPTAKE
           call nup(j)
           call npup(j)
           tfp = 0.
           ano3 = 0.
           if(ansf(j).gt.0.) call afert(j)
           xx = dm(j) + ddm
           strsn = 1.
           strsp = 1.
           reg = amin1(ws(j),ts,strsn,strsp)
           if(reg.lt.0.) reg = 0.
           if(reg.gt.1.) reg = 1.
           if (dmtar(nro(j),icr(j),j).gt.0.) then
             ddm = ddm * (dmtar(nro(j),icr(j),j) - dm(j)) /
     *       dmtar(nro(j),icr(j),j)
             reg = 1.
           endif
           dm(j) = dm(j) + ddm * reg
           rwt(j) = (.4-.2*g(j))
```

```
            f = g(j) / (g(j)+exp(dlp1(ncr(nro(j),icr(j),j)) -
     *          dlp2(ncr(nro(j),icr(j),j)) *g (j)))
            cht(j) = chtmx(ncr(nro(j),icr(j),j)) * sqrt(f)
            ff = f - hufh(j)
c            hia(j) = hia(j) + hi(ncr(nro(j),icr(j),j)) * ff
            hufh(j) = f
c            fhu = sin(1.5708*(g(j)-.3)/.3)
c            if (g(j).lt..3.or.g(j).gt..9) fhu = 0.
c            hia(j) = amax1(hia(j)-hi(ncr(nro(j),icr(j),j))*(1.-1./(1.+
c     *        wsyf(ncr(nro(j),icr(j),j))*fhu*(.9-ws(j)))),.01)

            if ( igro(j) .gt. 0.0 ) then

              if (g(j) .gt. 0.5) then

               swh = swh + actual
               swp = swp + potentl
             endif

c             heat = 100. * g(j)
              heat = 100. * (1.-(dlai(ncr(nro(j),icr(j),j))-g(j)))



              dum1=.1
              dum2=.95
              dum3=50.
              dum4=95.
              call ascrv(dum1,dum2,dum3,dum4)

              x1 = 0.05
              x2 = 0.9
              x3 = 10.05
              x4 = 50.90
              call ascrv(x1,x2,x3,x4)

              hia(j) = hi(ncr(nro(j),icr(j),j)) * heat /
     *                (heat + exp(dum1 - dum2 * heat))
              hia(j) = amax1(hia(j),wsyf(ncr(nro(j),icr(j),j)))
              incr = 100. * swh / (swp + 1.e-10)
              dum = incr / (incr + exp(x1 - x2 * incr))
c             incr = amax1(hia(j) - wsyf(ncr(nro(j),icr(j),j)), 0.0)
c             hiadl(j) = dum * incr + wsyf(ncr(nro(j),icr(j),j))
              hiadl(j) = dum * hia(j)
            endif
C           SUM STRESS DAYS
            sdw = sdw + (1.-ws(j)) * flu(j)
            sdt = sdt + (1.-ts) * flu(j)
            sdn = sdn + (1.-strsn) * flu(j)
            sdp = sdp + (1.-strsp) * flu(j)
            if (g(j).le.dlai(ncr(nro(j),icr(j),j))) then
              deltalai = ff * blai(ncr(nro(j),icr(j),j)) * (1.0 -
     *              exp(5.0 * (alai(j) - blai(ncr(nro(j),icr(j),j)))))
     *              * sqrt(reg)
c              wlv = (1.-rwt(j)) * xx
c              f = wlv / (wlv+exp(9.5-.0006*wlv))
              alai(j) = alai(j) + deltalai
              olai(j) = alai(j)
              if(alai(j).gt.xlai(j)) xlai(j) = alai(j)
            else
              alai(j) = 16. * olai(j) * (1.-g(j)) ** 2
            end if

          end if
        end if

        return
        end
```

Subr Swu19: distribute plant evaporation and soil moisture

```
      subroutine swu19(j)
c     THIS SUBROUTINE DISTRIBUTES POTENTIAL PLANT EVAPORATION THROUGH
c     THE ROOT ZONE AND CALCULATES ACTUAL PLANT WATER USE BASED ON SOIL
c     WATER AVAILABILITY. ALSO ESTIMATES WATER STRESS FACTOR FOR USE IN
C******common.f
      include 'common.f'
      dimension u(10)
      potentl = ep
      do 10 k = 1, nn
        u(k) = 0.
   10 continue
      if (ep.le.0.) then
        ws(j) = 1.
      else
        k = 1
        ir = 0
        sum = 0.
cccc    if (ird(ncr(nro(j),icr(j),j)).eq.1) then
cccc      rd(j) = 2.5 * g(j) * zmx(j)
cccc      if (rd(j).le.zmx(j)) go to 20
cccc    end if
cccc    rd(j) = zmx(j)
ccc20   xx = 0.
c-----------------
c-The above 6 lines of code have been rewritten as follows to avoid
c referring to nonexistent array locations (spp):
        rd(j) = zmx(j)
        nroval=nro(j)
        icrval=icr(j)
        if (nroval.gt.0.and.icrval.gt.0) then
          icrop = ncr(nroval,icrval,j)
          if (icrop.gt.0) then          !root depth: eqn (219), manual
            if (ird(icrop).eq.1) rd(j) = zmx(j)*AMIN1(1.,2.5*g(j))
          end if
        end if
        xx = 0.
c end revision-----------------------------------------------------
        do 30 k = 1, nn
          if (ir.gt.0) go to 40
          if (rd(j).le.z(k,j)) then
            gx = rd(j)
            ir = k
          else
            gx = z(k,j)
          end if
          if (rd(j).le.0.) then
            sum = ep / uob
          else
            sum = ep * (1.-exp(-ub*gx/rd(j))) / uob
          end if
          u(k) = sum - xx
          ul4 = ul(k,j) / 4.
          if (st(k,j).lt.ul4) then
            u(k) = u(k) * st(k,j) / ul4
          end if
          if (st(k,j).lt.u(k)) then
            u(k) = st(k,j)
          end if
          st(k,j) = st(k,j) - u(k)
          xx = xx + u(k)
   30   continue
   40   ws(j) = xx / ep
        ep = xx
      end if
      return
      end
```

Appendix. File common.f, variable and array declarations for Swat

```
      character*8 till
c     real mumax,rhoq,kl,kn,kp,lambda0, lambda1, lambda2,
c     *nitriten(mb),nitraten(mb)
C*****COMMON BLOCKS FOR SWRRBWQ AND GRAPHICS ROUTINES
c     mb = max number of subbasins within the big subbasins
c     mbb = max number of big subbasins
c     ml = max number of soil layers
c     mr = max number of reservoirs
c     mp = max number of pesticides
c     mch = max number of channels
c     ma = max number of applications
c     mcr = max number of crops per year
c     mcrdb = max number of crops in the database
c     mpdb = max number of pesticides in the database
c     mtil = max number of tillages in the database
c     mnr = max number of years of rotation
c     mop = max number of operations within one year of rotation
c     mgr = max number of grazings within one year of rotation
c     mrg = max number of rainfall/temperature gages
c     mhyd = max number of hydrograph nodes
c     myr = max number of years of simulation
c     mxmon = max no. months simulated (replaces 360 in some arrays, which
c           limited simulation to 30 yrs; set mxmon to 12*myr.--spp jan 30 96
c****NANCY SWAT
      parameter (mbb = 100)
      parameter (mb = 100)
      parameter (mch = 100)
      parameter (ml = 10)
      parameter (mr = 10)
      parameter (mp = 10)
      parameter (mpdb = 105)
      parameter (ma = 10)
      parameter (mcr = 3)
      parameter (mgr = 3)
      parameter (mcrdb = 37)
      parameter (mtil = 83)
      parameter (mnr = 10)
      parameter (myr = 40)
      parameter (mxmon=12*myr)  !replaces 360 below(see note above)--spp
c     parameter (mop = 5)
      parameter (mrg = 180)
      parameter (mhyd = 3*mb)
      parameter (msubo = 60)
      parameter (mxssb = 109) ! added to dimension basin arrays sm,smm,smy--spp
      common /read1/ xij(mb), yij(mb), vobmx(12,mb), vobmn(12,mb),
     *    wvl(12,mb),sm(mxssb),smm(mxssb),smy(mxssb),cn2(mb),ovn(mb),
     *    css(mb), ecp(mb), stp(mb), sax(mb), saf(mr), sas(mr),
     *    chs(mb), chn(mb), box, clt, cu, nns, nsb, boy, iwst, isst,
     *    sbp(mb), ssbp(mb), title(60), sid(11), prog(20), mon(12),
     *    nrf(mb), amo(12,8), xsl, ysl, sx, yx, wob(mxmon), sob(mxmon),
     *    wpd(mxmon), spd(mxmon), sepp(mb), sepd(mr), ys, a2,snobi,frd,
     *    vrw, vry, cnl, fpd, dat, sumfc(mb), sumul(mb), ssfp(ml,mb),
     *    sl(mb), irgage(mb), itgage(mb), rmeas(mrg), txmeas(mrg),
     *    tnmeas(mrg), nrgage, ntgage, iprn, slsoil(mb), ipr, vl, ipd,
     *    vb, rzl, ncrp, nro(mb), ncr(mnr,mcr,mb), nrot(mb), gplt(mb),
     *    wac21(mcrdb), wac22(mcrdb), co21, co22, tmpmean(12,mb),
     *    hiadl(mb), vpd, potentl, dlp1(mcrdb), dlp2(mcrdb), pt1max,
     *    pt2max, beadj, kevran(10), yldanu(mb), incrcn,
     *    dmanu(mb)
      common /bk1/ st(ml,mb), ul(ml,mb), cla(ml,mb), sc(ml,mb),
     *    hk(ml,mb), fc(ml,mb), t(ml,mb), z(ml,mb), wf(2,mb), g(mb),
     *    rt(mb), abd(mb), ub, uob, sep, prk, sww, nn, su, ns(mb), j1,
     *    j2, lu, ird(mcrdb), cf(mb), bcv(mb), wss(2,mb), rtc(ml,mb),
     *    zmx(mb), ifert(mnr,ma,mb), iap(mnr,ma,mb), iai(mnr,ma,mb),
     *    cnop(mnr,ma,mb), hia(mb), hufh(mb), wsol(mpdb), idap(mb),
     *    nop(mb), ipest(mnr,20,mb), iop(mnr,ma,mb), igro(mb),
*     pname(mpdb), p2(mb), bff, rfile(18), tfile(18), resvo(18)
 common /tar/  hitar(mnr,ma,mb), dmtar(mnr,ma,mb), rfinc(mb,12),
*     tmpinc(mb,12), radinc(mb,12), huminc(mb,12), co2inc(mb,12)
 common /bk2/ s1(mb), s2(mb), sw(mb), gr(mb), tv(mb), eo, tu,
*     dk(mp,mb,ml), wof(mpdb), hl(mpdb), skk(mpdb), efa(mpdb),
*     nptot(mb), gp(mp,mb,ml), spa(mp,mb), dkg(mp,mb), dkf(mp,mb),
*     ffp(mp,mb), cs2(mb), bd(mb), erp(mp,mb), zdb(mp), qi,
*     zq(mp,mb), npno(mp), z1(mp,mb), z0(mp,mb), pab(mpdb),
*     pdb(mp), skoc(mpdb), snam(mb), dx(mb)
 common /bk3/ alai(mb), salb(mb), dm(mb), rd(mb), uw, ep, es, rain,
*     ws(mb), je, i, ida, yon, yph, ysp, yno3, uno3, ssfn, percn,
*     cdg, xmin(28,mb), xmax(28,mb), sut, pminm, xnash, rsq, nvbsub
 common /bk4/ nc(13), iyr, nt, mo, nbyr, ires, ssf, wysb, nvrch,
*     nvres, nvsub, dart(mhyd), rmn, snob, xlu, bst, prq, ppq, fl,
*     brt, flomon(10,myr,12), iy, statin
 common /bk5/ r(366), vmx(mb), v(mb), flu(mb), bp(mb), hc(mb),
*     hcr(mb), fp(mb), cs(mb), sk(mb), smx(mb), ev, sp, rl, ql, o,
*     qd, af, pq, yp, aff, amps, cn, sml, cfp(mb), sfc(mb), rp, rq,
*     precip, da, fn(mnr,ma,mb), pa(mnr,ma,mb), nbgis(mb),
*     fph(mnr,ma,mb), ai(mnr,ma,mb), evp, tlc, diver, rflow,
*     sae(mb), bn(2), nfert(mb), npest(mb), nirr(mb), cbn(ml,mb),
*     vme(mb), bp1(mb), bp2(mb), iflod1(mb), iflod2(mb),
*     ndtarg(mb), saxw(mb), vmxw(mb), saew(mb), vmew(mb), vw(mb),
*     sepw(mb), csw(mb), cfw(mb), hcw(mb), bw1(mb), bw2(mb),
*     frorgn(mnr,ma,mb), frorgp(mnr,ma,mb), lfert(mnr,ma,mb),
*     forgn(mb), forgp(mb), lafert(mb), afw, nbigs(mb), nmgt(mb),
*     nbigis(mbb)
 common /bk6/ vrf(mr), vrs(mr), vr(mb), br1(mr), br2(mr), rrr(mr),
*     fw(mb), csr(mb), cfr(mr), qdr, yd, er, ydi, conn, cpp,
*     resout(mr,12,myr), iresco(mr), mores(mr), iyres(mr)
 common /bk7/ rst(12,3,mb), t0(mb), prw(2,12,mb), obmx(12,mb),
*     obmn(12,mb), obsl(12,mb), cvt(12,mb), cvs(12), wft(12,mb),
*     subp(mb), sno(mb), snoev, k1(4), k2(4), k3(4), k4(4), k5(4),
*     k6(4), k7(4), k8(4), k9(4), k10(4), k11(4), v1, v3, v5, v7,
*     ra(mb), tmxg, tmng, rmx, pit, st0, amp(mb), avt(mb), ts, lsim,
*     nsim, msim, tmx(mb), tmn(mb), tb(mcrdb), to(mcrdb),
*     yld(mnr,mcr,mb), hi(mcrdb), tx(mb), leg(mcrdb), be(mcrdb),
*     wsyf(mcrdb), rsdin(mb), icr(mb), pt2(mcrdb), lw, idsb, idse,
*     dayl, daylmn(mb), vpth(mcrdb), vpd2(mcrdb), gsi(mcrdb),
*     chtmx(mcrdb), cht(mb), dewpt(12,mb), uavm(12,mb), co2(mb),
*     ipet, u10(mb), rhd(mb), elev(mb), rhmo, wavp(mcrdb)
 common /bk8/ kr1, kr2, kr3, kr4, kw, kw1, kw2
 common /bk9/ wi(12,mb), wim(12), tc(mb), al(mb), ab, ab1, rn1, r1,
*     pr, lu1, vo, nmb, tr, ddt, trt, dur, iv, alc(mb)
 common /bk10/ cvm(mcrdb), cv(mb), rsd(mb), ek(mb), rwt(mb),
*     olai(mb), blai(mcrdb), ipl(mnr,mcr,mb), ihv(mnr,mcr,mb),
*     itil(mcr,mb), tf(4), rsdi(4), dm2(mnr,mcr,mb), irr(mb),
*     wsf(mb), irn(mb), air(mb), tilop(8), c, itnum(mtil),
*     effmix(mtil), till(mtil), effmx(mnr,ma,mb), phu(mnr,mcr,mb),
*     phup(mnr,mcr,mb), phutot, phuh(mnr,mcr,mb), phut(mnr,ma,mb),
*     ansf(mb), fnmx(mb), anmx(mb), idmn(mb), ipman(mb), tfp, ano3,
*     phuho(mnr,mcr,mb), phuk(mnr,mcr,mb), ikill(mnr,mcr,mb),
*     ihvo(mnr,mcr,mb), phun(mnr,ma,mb)
 common /bk11/ d50(mb), qp(mb), tbo(mb), tbs, wy, da7, da9
 common /bk12/ chw(2,mb), chd(mch), chl(2,mb), chss(mch),
*     chnn(mch), chk(2,mb), psz(5,mb), chxk(mch), chc(mch),
*     pct(5,mb), rock(mb)
 common /bk13/ xyr(30), yyr(30), imo(30), xmo(30), ymo(30)
 common /bk14/ efi(mb), san(mb), sil(mb), sy(mb), syy(mb),
*     sym(mb), ssq(mb), sysq(mb), smsq(mb), sq(mb), syq(mb),
*     potet(mb),smq(mb),ssb(mxssb),bs(mb),awc(ml,mb),up(ml,mb),
*     wp(ml,mb), por(ml,mb), flat(ml,mb), po(ml,mb), snup(mb),
*     sda(49,mb), smo(20,mb), syr(20,mb), stot(20,mb), aabv(20),
*     nfull(mb), nempty(mb), ndmo(12), dtot, vrmm, idaf, idal,
*     varoute(31,mhyd), srch(18,mch), sres(38,mr), ssub(msubo,mb),
*     syrch(18,mch), syres(38,mr), sysub(msubo,mb), strch(38,mch),
*     stres(38,mr), stsub(msubo,mb), fup(ml,mb), phi(7,mb),
*     sdtsav(mb), sdti, sbsub(14,mbb), sybsub(14,mbb),
*     stbsub(14,mbb), pet(mb)
 common /bk15/ umw(12), eff1(12), efflt(12), flowt(12), td(12),
*     isub1, cpest1, vreac, vvol, vpart, vsetl, vrsup, vmix, cpest2,
     *    vreac2, vbury, dact, vsetlp, phosl, cin, cout, react, volat,
     *    setl, reactb, resus, bury, difus, pin, pout, ploss,
     *    efflq(12), chla, secci, tlake, dlake, sa
      common /bk16/ wn(ml,mb), wmn(ml,mb), hum(ml,mb), ap(ml,mb),
     *    fop(mb), fon(mb), wno3(ml,mb), pmn(ml,mb), op(ml,mb),
     *    spup(mb), wpo(ml,mb), spal, spas, sfn, sfp, sdnit, shmn, srwn,
     *    shmp, srmn, srmp, wt(ml,mb), psp, rtn, cnb, strsn, strsp, sdw,
     *    sdt, sdn, sdp, sfix, sbpup, rmnltl, roctl, wfx, wdntl, hmntl,
     *    rwntl, hmptl, rmn2tl, rmptl, uap
      common /bk17/ icodes(mhyd), ihouts(mhyd), inum1s(mhyd),
     *    inum2s(mhyd), inum3s(mhyd), rnum1s(mhyd), inum4s(mhyd)
      common /bk19/ gwht(mb), gwq(mb), delay(mb), abf(mb), syld(mb),
     *    revapc(mb), rchrgc(mb), revap, gwseep, gwchrg, revapmn(mb),
     *    abf1(mb), revapst(mb), rchrg(mb), yldaa(mb), dmaa(mb),
     *    xlaia(mb), xlai(mb)
      common /bk20/ tp5(mb), tp6(mb), tp24(mb), ylt(mb), yls(mb),
     *    ylc(mb), ffc(mbb), wprecip(mbb), wcklsp(mbb), cklsp(mbb),
     *    wsml(mbb),wqd(mbb),wssf(mbb),wppq(mbb), wyd(mbb), wgp(mbb),
     *    wwn(mbb), wwpo(mbb), dmba, swmo(12), vrmo(12), wyno3(mbb),
     *    wssfn(mbb), wysp(mbb), weo(mbb), wuw(mbb), wwy(mbb),
     *    wflu(mbb), wlai(mbb), wdm(mbb), wyld(mbb), wgwq(mbb),
     *    wsw(mbb), walai(mbb), wadm(mbb), wayld(mbb), waadm(mbb),
     *    waalai(mbb), waayld(mbb)
      common /bk21/ wures(12,mb), wurch(12,mb), wushal(12,mb),
     *    wudeep(12,mb), deepst(mb), igrz(mb), igraz(mr,mgr,mb),
     *    phug(mr,mgr,mb), bmeat(mr,mgr,mb), ndeat(mb),
     *    ndgraz(mr,mgr,mb), ngr(mb), irrsub(mb), wurtn(mb),
     *    irtnsb(mb), wuresn(12,mr), wurtnf(mr), irtnsn(mr),
     *    shallst(mb)

c*****common block to be added to common.f
c     common /qual2e/rs1(mb),rs2(mb),rs3(mb),rs4(mb),rs5(mb),rs6(mb),
c     *rs7(mb),rk1(mb),rk2(mb),rk3(mb),rk4(mb),rk5(mb),rk6(mb),bc1(mb),
c     *bc2(mb),bc3(mb),bc4(mb),ammonian(mb),
c     *organicn(mb),wtemp(mb),organicp(mb),disolvp(mb),bod(mb),abc(mb),
c     *disox(mb),algae(mb),lao,ai0,ai1,ai2,ai3, ai4, ai5, ai6,
c     *rhoq, pn,mumax,rho, kl, kn, kp, lambda0, lambda1, lambda2,
c     *nitriten(mb),nitraten(mb),dq,tfact,graopt,thrk1,thrk2,thrk3,
c     *thrk4,
c     *thrk5,thrk6,thbc1,thbc2,thbc3,thbc4,thrs1,thrs2,thrs3,thrs4,thrs5,
c     *thrs6,thrs7,thgra,thrho


      common /reserv/ nd, storec(7), releasec(7), storek(7),
     *    releasek(7), ls(mb)

c*****NANCY SWAT
      common /bk18/ ipnum(mpdb), yldn(mnr,mcr,mb), dm2n(mnr,mcr,mb),
     *    sol(mpdb), icnum(mcrdb), cpnm(mcrdb), dlai(mcrdb),
     *    rdmx(mcrdb), ncrops(mnr,mcr,mb), summ, sump
c*****NANCY SWAT
      character*16 snam, pname
      character*13 rfile, tfile, resvo, statin
      character*4 tilop, sid, prog, mon, title
```