PROGRAMMERS MANUAL FOR A ONE-DIMENSIONAL LAGRANGIAN TRANSPORT MODEL

By David H. Schoellhamer and Harvey E. Jobson

U.S. GEOLOGICAL SURVEY

Water Resources Investigation Report 86-4144

NSTL, Mississippi

1986

UNITED STATES DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to

U.S. Geological Survey
Water Resources Division
Gulf Coast Hydroscience Center
Bldg. 2101
NSTL, Mississippi 39529

Copies of the report can be
purchased from:

Open-File Services Branch
Western Distribution Branch
U.S. Geological Survey
Box 25425, Federal Center
Denver, Colorado 80225
(Telephone [303] 236-7476)

CONTENTS

	Page
Abstract	1
Introduction	1
Lagrangian modeling techniques	3
Parcel tracking	3
Dispersion	6
Inflows	11
Simulation procedure	25
Lagrangian Transport Model subroutines	27
Inflow functions	27
Optional output subroutines	27
Time-Dependent Data System subroutines	29
Subroutine DECAY	30
Decay-coefficient subroutine	31
CON.F77	32
OXYGEN.F77	32
QUAL2.F77	34
Writing a decay-coefficient subroutine	35
Summary	36
References	37
Appendix A. LTM.F77	39
Appendix B. TDDS.F77	58
Appendix C. CON.F77	89
Appendix D. OXYGEN.F77	91
Appendix E. QUAL2.F77	95

ILLUSTRATIONS

Figure 1. Inter-parcel flows	7
2. Example dispersion calculation	10
3. Effect of inflows on dispersion calculation	12
4. Parcel passing tributary inflow	14
5. Parcel stopped with tributary inflow	15
6. Parcel distance exposed to lateral inflow vs. time	17
7. Parcel leaving a reach with lateral inflow	18
8. Parcel stopped wholly within a reach with lateral inflow	20
9. Parcel stopped with upstream boundary in a reach with lateral inflow	21
10. Parcel spanning several reaches	23
11. Downstream parcel boundary in a reach with a lateral inflow	24
12. Grid concentration calculation in unsteady flow	26

TABLES

Page

Table 1. Array dimension adjustments. 4
2. Lagrangian Transport Model subroutines 28

CONVERSION FACTORS

The Lagrangian Transport Model is capable of receiving input in either English or Metric (SI) units, so both unit systems are used in this report. The following conversion factors may be used to convert the units of measurement in this report.

MULTIPLY	BY	TO OBTAIN
feet	0.3048	meters (m)
mile(mi)	1.609	kilometers (km)
square feet (ft^2)	0.0929	square meters (m^2)
cubic feet (ft^3)	0.02832	cubic meters (m^3)
feet per second (ft/s)	0.3048	meters per second (m/s)
feet per hour (ft/hr)	0.3048	meters per hour (m/hr)
miles per hour (mi/h)	1.609	kilometers per hour (km/hr)
square feet per second (ft^2/s)	0.0929	square meters per second (m^2/s)
cubic feet per second (ft^3/s)	0.02832	cubic meters per second (m^3/s)
kilopascal (kpa)	10.00	millibars (mb)
inches of mercury (in Hg)	33.864	millibars (mb)
milligrams per day per foot (mg/day-ft)	3.2808	milligrams per day per meter (mg/day-m)
Langley	1.0000	calories per square centimeter

Temperature Conversion: °F = 1.8°C + 32
°C = (°F - 32)/1.8

PROGRAMMERS MANUAL FOR A ONE-DIMENSIONAL LAGRANGIAN TRANSPORT MODEL

By David H. Schoellhamer and Harvey E. Jobson

ABSTRACT

A one-dimensional Lagrangian transport model for simulating water-quality constituents such as temperature, dissolved oxygen, and suspended sediment in rivers is presented in this Programmers Manual. Lagrangian transport modeling techniques, the model's subroutines, and the user-written decay-coefficient subroutine are discussed in detail. Appendices list the program codes. This Programmers Manual is intended for the model user who needs to modify code either to better adapt the model to a particular need or to use reaction kinetics not provided with the model.

INTRODUCTION

A one-dimensional transport model can be a useful tool for predicting the fate of water-quality constituents in rivers. Such a model simulates the longitudinal movement of constituents as they are transported downstream. Direct modeling of secondary currents, stratification, transverse mixing, and other multi-dimensional phenomena are excluded. For example, the effect of a power plant discharge on water temperature is modeled by assuming that the outflow immediately mixes uniformly across the river cross section and is transported by the cross-sectionally averaged flow velocity. The primary factor in transport modeling is the flow of the river, called convection, which moves the constituents downstream. As convection occurs, the constituents tend to disperse in the longitudinal direction. Other factors may include the mixing of both point and non-point sources and the decay of constituents with time.

Simulation of these processes is accomplished by solving the convection-dispersion equation. This equation is derived by considering the principle of conservation of mass and must be solved by numerical approximation. Most solution techniques use an Eulerian reference frame that fixes the computational nodes in space. Although Eulerian modeling is easy to conceptualize, the computations are fairly difficult and the results can be inaccurate, oscillatory, and unstable. The alternative is a Lagrangian reference frame which eliminates many of the Eulerian difficulties.

Lagrangian transport modeling moves the computational nodes, which are in reality parcels of water, with the flow of the river. Even though the tracking of parcels requires much bookkeeping, the troublesome convection term is eliminated from the convection-dispersion equation. As parcels pass

both point and non-point sources the inflow is mixed and new parcel concentrations are computed. In addition, decay is computed within each parcel during each simulation time step.

A one-dimensional Lagrangian transport model, called the LTM, for simulating water-quality constituents such as temperature, dissolved oxygen, and suspended sediment in rivers is presented in this Programmers Manual. Lagrangian transport modeling techniques, the LTM subroutines, and the decay-coefficient subroutine are discussed and the program codes are listed. This Programmers Manual is intended for the LTM user who needs to modify code either to better adapt the LTM to a particular need or to use reaction kinetics not provided with the LTM. A companion document serves as a Users Manual for the LTM (Schoellhamer and Jobson, 1986). The Users Manual describes transport modeling theory, discusses how to apply the LTM, and presents three example applications.

The LTM can simulate both steady and unsteady streamflow in addition to variable boundary conditions and solute loads. Time-dependent data can be input from the data base of the Time-Dependent Data System (TDDS, Krug and others, written commun., 1983). Input may be in either English or metric units. Output options include storing output either with the TDDS or in direct access files, calculating a root-mean-squared error, calling a user-written plotting subroutine, and writing decay coefficients. The output includes grid concentration, initial parcel concentration, travel time, and change in concentration due to decay, dispersion, lateral inflows, and tributary inflows.

Decay and constituent reactions are modeled by determining decay coefficients in a subroutine that may be written by the user to suit his particular needs. Examples of three different decay-coefficient subroutines are listed in this Programmers Manual and discussed in the LTM Users Manual (Schoellhamer and Jobson, 1986). One of these subroutines uses the reaction kinetics of the QUAL II water-quality model (Roesner and others, 1977a, 1977b), thus providing the advantages of unsteady Lagrangian calculations and the QUAL II reaction kinetics. The user, however, should not use any of the example kinetics if they do not apply to his particular problem. This Programmers Manual describes how to write a decay-coefficient subroutine if the existing routines are not satisfactory.

This Programmers Manual describes the techniques used by the LTM to simulate physical processes, the LTM subroutines, and the decay-coefficient subroutine. The Appendices list the program codes for the LTM, TDDS subroutine, and the three example decay-coefficient subroutines.

LAGRANGIAN MODELING TECHNIQUES

Lagrangian modeling techniques have been developed that compute convection, dispersion, and the effects of point and non-point sources and sinks. These techniques are contained within the Lagrangian Transport Model (LTM).

Parcel Tracking

Parcel tracking is used to account for river convection, but the flow field must first be known. In order to define the flow field for the LTM, the river is discretized by a maximum of fifty non-uniformly spaced Eulerian grid points at which the flow field is given to the LTM by the user. Thus the user has the convenience of defining the system in a Eulerian reference frame and the advantage of Lagrangian calculations. More than fifty grid points can be used if the arrays listed in Table 1 are redimensioned.

For steady flow simulation the required inputs are discharge at the upstream boundary, source/sink discharges, as well as river mile or kilometer, cross-sectional area, and cross-sectional top width at every grid point. Reach averaged areas and top widths are computed for every reach where a reach is defined as the segment of river between two adjacent grid points. Top widths are not used for any calculations in the main LTM program but may be used in the decay-coefficient subroutine, as will be discussed later. The program calculates the average discharge in every reach and divides by the average reach cross-sectional area to determine the average reach velocity. These calculations are performed only once for the entire steady flow simulation.

For unsteady flow simulation, the river mile or kilometer of every grid point is initially input. Then at every time step beginning with time zero, the hydraulic data (velocity, area, top width, and source/sink discharges) at the end of the time step at each grid point are read from an input file. The calculated time or time and reach averaged values are

$$
\begin{aligned}
\bar{A}_{I,J} &= \frac{1}{4}\left(A_{I,J-1} + A_{I,J} + A_{I+1,J-1} + A_{I+1,J}\right) \\
\bar{U}_{I,J} &= \left[(UA)_{I,J-1} + (UA)_{I,J} + (UA)_{I+1,J-1} + (UA)_{I+1,J}\right]/(4\bar{A}_{I,J}) \\
\bar{W}_{I,J} &= \frac{1}{4}\left(W_{I,J-1} + W_{I,J} + W_{I+1,J-1} + W_{I+1,J}\right) \qquad (1) \\
\overline{QIN1}_{I,J} &= \frac{1}{2}\left(QIN1_{I,J-1} + QIN1_{I,J}\right) \\
\overline{QIN2}_{I,J} &= \frac{1}{2}\left(QIN2_{I,J-1} + QIN2_{I,J}\right)
\end{aligned}
$$

Adjustment	Default	Subroutine/Arrays
Number of Grids	50	MAIN/X,IGIN,NGOUT,A,CIN,CINC, DQQ,DX,FLOW,GT,GTR,QIN, TACT,TLPP,TT,U,W
		DECAY/A,U,W
		STORE/GT,GTS
Number of Parcels	500	MAIN/ND,DF,DMF,DQ,PDC,PDF,PH, PT,PTI,PTR,PV,PX
		DECAY/PDC,PT
GRID.DATA Storage	200	STORE/DATA

**NOTE: Array initialization in the main program must also be adjusted in order to change the number of grids (line 390) or parcels (line 375). To adjust the number of time steps in GRID.DATA the record length of GRID.DATA must also be adjusted at line 485 of the main program.

Table 1.--Array Dimension Adjustments

in which $A_{I,J}$ is the cross-sectional area at grid I at the end of time step J, $U_{I,J}$ is the mean velocity at grid I at the end of time step J, $W_{I,J}$ is the top width at grid I at the end of time step J, $QIN1_{I,J}$ is the tributary inflow at grid I at the end of time step J, $QIN2_{I,J}$ is the lateral inflow in the reach downstream from grid I at the end of time step J, and a bar ($\bar{\ }$) indicates an average value computed and used by the program. These calculations are repeated every time step. If the TDDS option is used to input data, grid velocities are calculated from the discharge and area at each grid by the TDDS interface subroutine.

Once the flow field is defined for either the entire simulation or a time step, convection can be computed. The river is divided into discrete volumes called parcels whose contents are assumed to be completely mixed. The upstream boundary of each parcel is constantly tracked. Parcels are moved one at a time by comparing how long it will take the upstream parcel boundary to reach the downstream reach boundary (PDT) to the time remaining for movement within the time step (RDT). If PDT is greater than RDT the parcel will not pass the boundary this time step so the parcel location PX is incremented by U*RDT/DX where PX is in grid coordinates (PX = 2.0 at grid 2) and DX is the length of the reach containing the parcel. If PDT is less than RDT then the parcel passes the boundary and PX is located at the grid, RDT is reduced by PDT, and the parcel is moved at least partially through the new reach at the new average reach velocity. These calculations start with the most downstream parcel and continue until all parcels have been moved until they stop (RDT < PDT).

Three special situations need to be discussed. First, the initial parcel locations are determined by locating parcel number one at the upstream system boundary and using either the steady flow or the unsteady flow field at time zero to determine where the other parcels are, all separated by a travel time equal to the simulation time step DT. The initial parcel concentrations are determined by linear interpolation of the user-provided initial conditions at the grid points. Secondly, when a parcel passes the most downstream grid point it is simply discarded. Finally, one new parcel is introduced at the upstream system boundary every time step, so the numbering of all parcels in the system is incremented by one every time step. Thus a smaller time step creates more parcels within the system.

Several variables are tracked with every parcel. In addition to parcel location PX, parcel volume (PV) in cubic meters, parcel concentration (PT), change in parcel concentration due to each source type (PTR), change in parcel concentration due to dispersion (PDF), initial parcel concentration (PTI), and the time each parcel entered the system (PH) in hours since midnight of day one are all tracked.

The LTM Users Manual (Schoellhamer and Jobson, 1986) discusses how parcel tracking simulates convection with neither the numerical difficulties associated with Eulerian solutions nor the interpolations required by Eulerian-Lagrangian schemes. The disadvantage of having to track several variables within the program is relatively minor by comparison. The steady flow simplification of McBride and Rutherford (1984) is not used because the LTM is intended to be applicable to both steady and unsteady flows.

Dispersion

As parcels move downstream, a two-dimensional velocity profile similar to that shown in Figure 1 will exist because the water further away from the fixed boundaries will be moving faster than the water near the fixed boundaries. Thus the faster moving water in a parcel will flow into the adjacent downstream parcel, and the slower moving water in the downstream parcel will flow into the upstream parcel. The net effect is an exchange flow or an inter-parcel flow that mixes adjacent parcels. Because the parcel boundaries are moved at the mean flow velocity, the parcel volumes are not affected by this exchange flow but the parcel concentrations are affected.

The LTM dispersion equation is derived from this exchange flow mixing concept. The exchange flow rate is defined as a fraction of the river discharge. Thus

$$DQ = DQQ * Q \tag{2}$$

in which DQ is the exchange flow rate, DQQ is the ratio of exchange flow to river discharge, and Q is the river discharge. DQQ is defined by the user in the LTM input for every reach. The mixing equation for parcel K of Figure 1 is

$$\Delta PT = DT*(DQ_K PT_{K-1} - DQ_K PT_K + DQ_{K+1} PT_{K+1} - DQ_{K+1} PT_K)/PV_K \tag{3}$$

in which ΔPT is the change in parcel concentration and DT is the simulation time step. For steady flow PV = Q*DT, so the steady mixing equation simplifies to

$$\Delta PT = DQQ_K PT_{K-1} - (DQQ_K + DQQ_{K+1}) PT_K + DQQ_{K+1} PT_{K+1} \tag{4}$$

The coefficient DQQ can be defined in terms of model and physical parameters by considering the Lagrangian transport equation

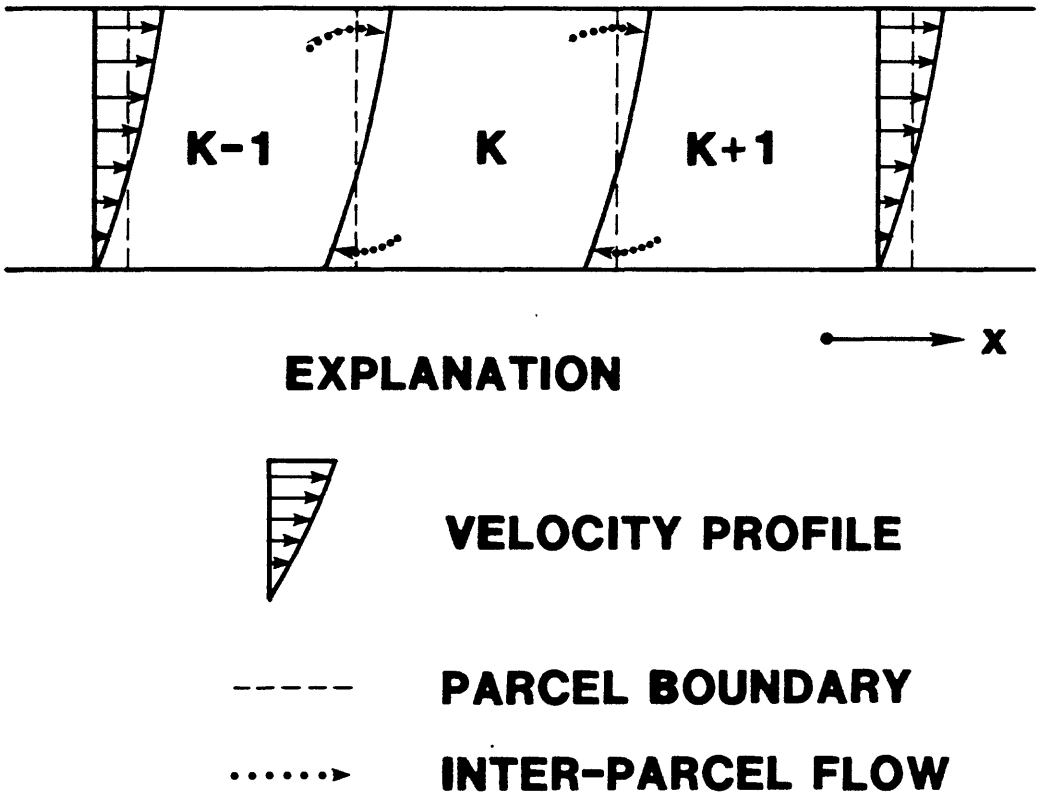


Figure 1.--Inter-parcel flows.

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D_x \frac{\partial C}{\partial x}\right) \tag{5}$$

in which C is concentration, t is the time coordinate, D_x is the longitudinal dispersion coefficient, and x is the longitudinal axis (Schoellhamer and Jobson, 1986). Application of a simple finite difference scheme to the flow condition illustrated in Figure 1 gives

$$\int^{DT} \frac{\partial C}{\partial t}\, dt = \int^{DT} \frac{\left(D_x \frac{\partial C}{\partial x}\right)_K - \left(D_x \frac{\partial C}{\partial x}\right)_{K+1}}{\Delta x}\, dt \tag{6}$$

in which Δx is the parcel length and the subscripts K and K+1 refer to the upstream boundary of parcel K and K+1, respectively. Application of simple finite differences again and the assumption that the time step DT is small gives

$$\Delta PT = DT*[D_{x_K}(C_{K-1} - C_K) + D_{x_{K+1}}(C_{K+1} - C_K)]/\Delta x^2 \tag{7}$$

Multiply both the numerator and denominator by Q=A*U to get

$$\Delta PT = DT*Q*[D_{x_K}(C_{K-1} - C_K) + D_{x_{K+1}}(C_{K+1} - C_K)]/(PV*U*\Delta x) \tag{8}$$

Comparing equations 3 and 8 gives the definition

$$DQQ = \frac{D_x}{U\Delta x} = \frac{1}{P} \tag{9}$$

in which P is the Peclet number, a dimensionless ratio of convection and dispersion. For steady flow, $\Delta x = U*DT$, so

$$DQQ = \frac{D_x}{U^2 DT} = \frac{D_x*DT}{\Delta x^2} = \left(\frac{\sqrt{D_x*DT}}{\Delta x}\right)^2 \tag{10}$$

For a time step, the quantity $\sqrt{D_x*DT}$ is related to the distance an advancing diffusion front travels and Δx is the distance a parcel travels. So DQQ is again shown to be a ratio of convection and dispersion.

Equation 3, however, is not completely satisfactory. If the exchanged volume during a time step is relatively large compared to the parcel volume ($DQ*DT \geq 0.5*PV$) then the equation becomes unstable. This can occur either with steady flows when the chosen time step is too small, causing a larger DQQ, or with unsteady flows when small parcels are in a relatively high discharge flow. Stability of the LTM is guaranteed by subdividing the simulation time step, when necessary, with the following algorithm.

At every parcel boundary the ratio of exchange volume to the minimum adjacent parcel volume is calculated by

$$RATIO = \frac{DQ*DT}{MIN(PV_K, PV_{K-1})} \tag{11}$$

in which MIN is the FORTRAN intrinsic function that selects the minimum of a variable list. Note that RATIO will always equal DQQ for steady flow. For the sample problem shown in Figure 2, the RATIOs at the three boundaries are 0.2, 0.5, and 1.0. Next, the number of time step subdivisions needed for the stable calculation of dispersion (RATIO less than 0.4 for the subdivided time interval) are calculated at each parcel boundary. The number of subdivisions, ND, is an integer power of two. Thus all effective RATIOs will be between 0.1 and 0.4 which is desirable because a RATIO of about 0.3 provides the most accurate simulation of the dispersion term. In the example ND(1)=1, ND(2)=2, and ND(3)=4. Each parcel concentration is updated for dispersion NDMAX times during the time step where NDMAX is the maximum value of ND in the system. The mass flux across each boundary during the time interval DT/NDMAX, however, is calculated ND times. The equation for mass flux is

$$DMF_K = DQ_K*DT*(PT_K + DF_K - PT_{K+1} - DF_{K+1})/NDMAX \tag{12}$$

in which DF is the change in concentration of the parcel due to dispersive fluxes since the beginning of the time step as given by

$$DF_K = (DMF_K - DMF_{K+1})/PV_K + DF_K \tag{13}$$

DQQ = 0.2, Q = 1 m^3/s, DT = 5 sec, mass = 1110.0000

PT = 100 PV = 10 1	PT = 10 PV = 10 2	PT = 0 PV = 2 3	PT = 10 PV = 1 4

RATIO 0.2 0.5 1.0

ND 1 2 4

NDMAX = 4

Subdivided Time Step	DF_1	DMF_2	DF_2	DMF_3	DF_3	DMF_4	DF_4
1		22.50		2.50		-2.50	
	-2.25		2.00		2.50		-2.50
2		22.50		2.50		-1.25	
	-4.50		4.00		4.38		-3.75
3		22.50		2.41		-0.47	
	-6.75		6.01		5.81		-4.22
4		22.50		2.41		0.01	
	-9.00		8.02		7.01		-4.21

PT = 91.0 PV = 10 1	PT = 18.02 PV = 10 2	PT = 7.01 PV = 2 3	PT = 5.79 PV = 1 4

mass = 1110.0000

NOTE: Intermediate results are rounded off to two decimal places

Figure 2.--Example dispersion calculation.

The example problem shows these calculations.

The need for this dispersion scheme can be explained with the example. If the time step was not subdivided, parcels three and four could cause the solution to be unstable because of their small size. Also if the mass flux was calculated only once, then the small parcels would prevent a dispersion front from advancing far enough (parcel three would block interaction between parcels two and four). But because the mass flux is only calculated when needed, parcels three and four do not affect the final concentration of parcel one. Thus separate calculation of mass flux prevents a dispersion front from being moved too far. This scheme also precisely conserves mass.

When subdivision is not needed, as is usually the case, equations 12 and 13 simplify to equation 3. Whenever possible, select a time step such that $0.1 < DQQ < 0.4$ in order to achieve the best results.

When longitudinal dispersion is negligible many solutions to the convection-dispersion equation suffer from artificial numerical dispersion (Cunge and others, 1980, Gray and Pinder, 1976, Grenney and others, 1978,, Gresho and Lee, 1981, Jobson, 1980, van Genuchten and Gray, 1978, and Varoglu and Finn, 1978). But use of the exchange flow concept alleviates this problem in the LTM. If D_x is zero, then DQQ is also zero, so the exchange flow rate is zero and the parcels are simply moved downstream with no dispersion.

A final point is the effect of inflows on the dispersion routine. If a parcel is either receiving a tributary inflow or is at the upstream boundary of a lateral inflow, the load added by the inflow should not disperse into the upstream parcel. Therefore DQ is set to zero when such a situation would otherwise occur, as shown in Figure 3.

Inflows

As the parcels are tracked, both tributary and lateral inflows (point and non-point sources) are mixed with the parcels. When a parcel either passes a grid point or completes movement at the end of a time step the mixing equation applied is

$$DEL = \frac{PT*PV + CIN*DELV}{PT + DELV} \tag{14}$$

in which DEL is the change in parcel concentration, PT is the parcel concentration before mixing, PV is the parcel volume before mixing, CIN is the concentration of inflow, and DELV is the change in parcel volume. In addition, the parcel volume is incremented by DELV and the array PTR is

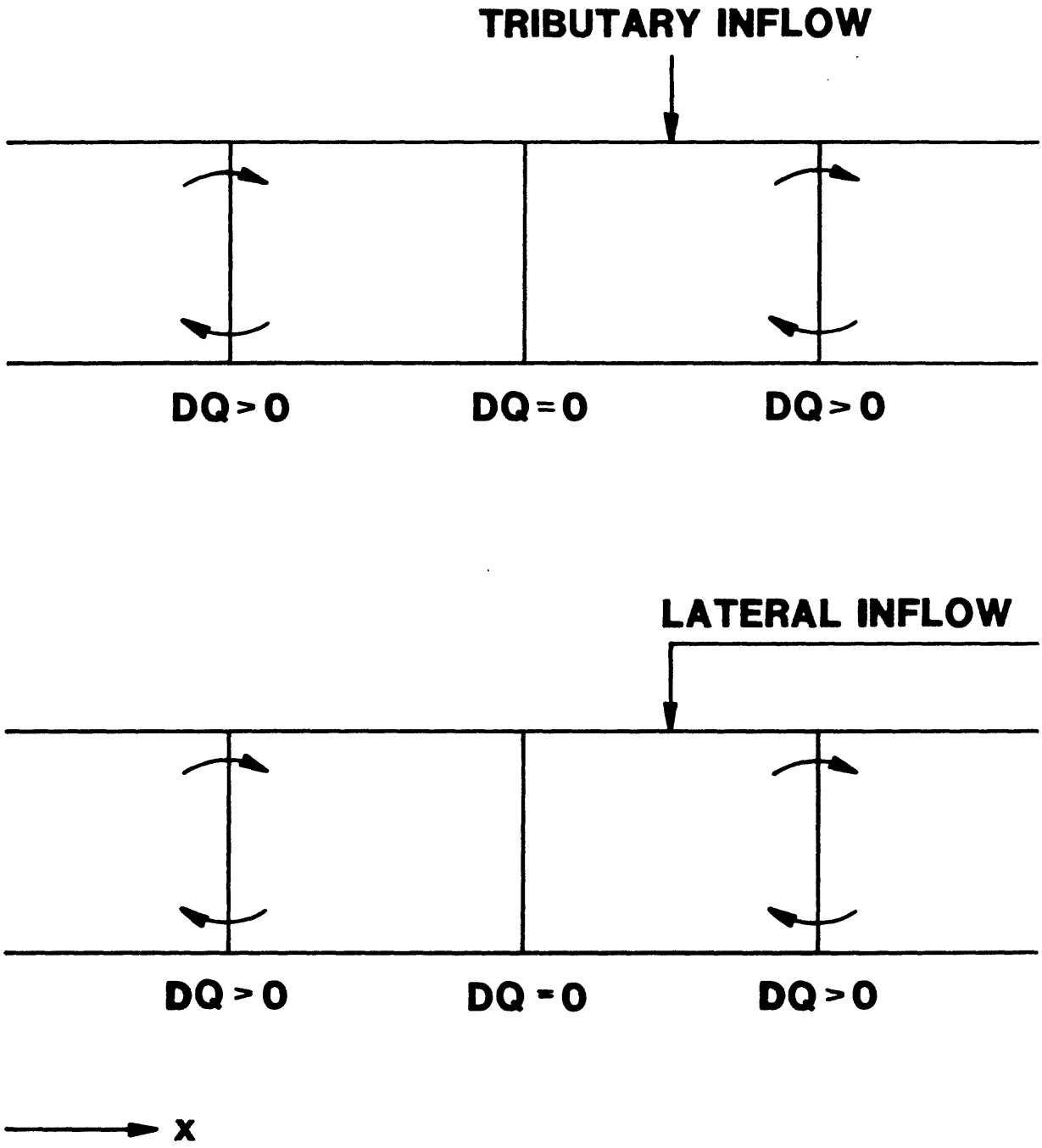


Figure 3.--Effect of inflows on dispersion calculation.

incremented by DEL so that the change in parcel concentration due to both types of sources are recorded. Of the variables needed for mixing, PT, PV and PTR are tracked and known, CIN is given in the input, and DELV must be calculated.

For tributaries, DELV is simply the tributary flow rate multiplied by the time the parcel was exposed to the inflow during the time step. When a parcel of volume PV passes a grid point with a tributary inflow, as in Figure 4, the time of exposure to the inflow is

$$DT - RDT - TLPP(MX) \tag{15}$$

in which DT is the simulation time step, RDT is the time remaining for parcel movement, TLPP is the time the downstream parcel passed the grid point relative to the beginning of the time step, and MX is the grid point with the tributary inflow that the parcel has passed. If the downstream parcel did not pass the grid point during the current time step TLPP equals zero. When a parcel completes movement at the end of a time step (RDT = 0) and is exposed to a tributary inflow, as shown in Figure 5, the time of exposure is

$$DT - TLPP(IX) \tag{16}$$

in which IX is the grid point of the tributary discharging into the stopped parcel.

Non-point inflows are more difficult to model because they require the determination of the parcel length exposed to a lateral inflow as a function of time. The instantaneous inflow volume is

$$d(DELV) = QIN2*\delta*dt \tag{17}$$

in which d(DELV) is the incremental inflow volume, δ is the parcel distance exposed to the lateral inflow, and dt is an incremental time. Integration over the complete time of exposure and the assumption that QIN2 is constant during the time step gives

$$DELV = QIN2* \int_{t1}^{t4} \delta \, dt \tag{18}$$

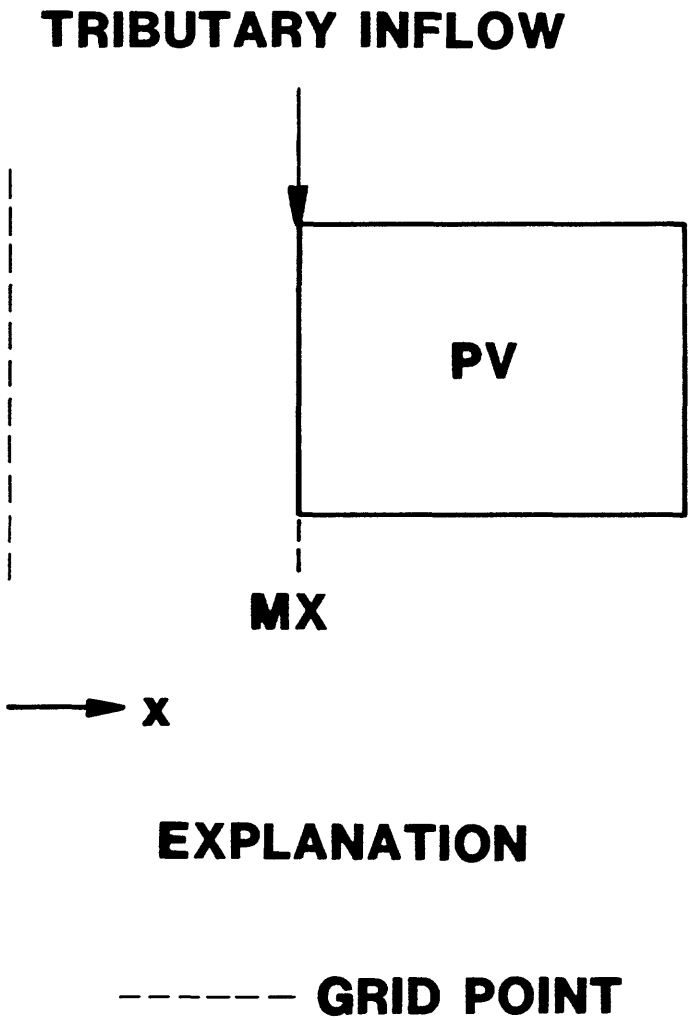


Figure 4.--Parcel passing tributary inflow.

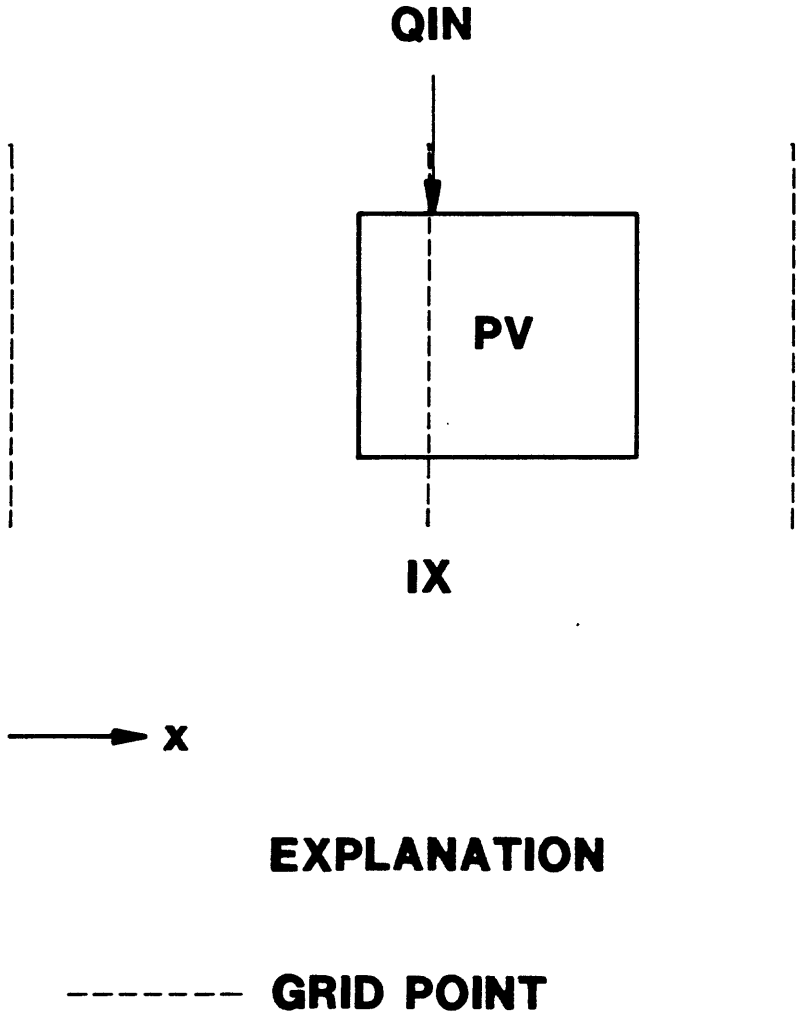


Figure 5.--Parcel stopped with tributary inflow.

The integral is the area under the curve of exposed parcel distance versus time, shown in Figure 6, in which t1 is the time that exposure to lateral inflow begins. As the exposed distance increases, the slope of the curve is $\Delta\delta/\Delta t = U$. This assumes that the incremental inflow during the time step has a negligible effect on the length of the parcel. The maximum distance exposed to the inflow, DMAX, is reached at t2. If the parcel volume is greater than the reach volume, t2 is the time that the downstream parcel boundary leaves the reach, but if the parcel volume is less than the reach volume, t2 is the time that the upstream parcel boundary enters the reach. At time t3 the exposed distance begins to decrease with time. If the parcel volume is greater than the reach volume, t3 is the time the upstream parcel boundary enters the reach, and if the parcel volume is less that the reach volume, t3 is the time that the downstream parcel boundary leaves the reach. As the exposed distance decreases, the slope of the curve is -U until the upstream parcel boundary leaves the reach at time t4. Note that all lower case times are relative to real time, t# > 0.

Figure 6 illustrates the method used by the LTM to compute changes in parcel volume due to non-point inflows. As a parcel leaves a reach with a non-point inflow or stops in such a reach the function AREA computes the area under the curve from the beginning of the time step to the current time T4 (not necessarily equivalent to t4). All upper case times are relative to the time step, $0 \leq T\# \leq DT$ and are used by the LTM. AREA begins at the current location on the curve (T4 and FD = distance exposed to lateral inflow at T4) and proceeds back in time until either T1 or the beginning of the time step is reached. The five values provided to AREA are T4, FD, U, DMAX, and T2. Note that, because this calculation is done within a time step, the previous assumption of constant QIN2 is valid.

When a parcel leaves a reach with lateral inflow the values given to AREA are

```
T4 = DT - RDT

FD = 0.0

U = U(MX-1)

DMAX = MIN(PV(K)/A(MX),DX(MX-1))

T2 = TLPP(MX-1+INT(DMAX/DX(MX-1)))                                  (19)
```

in which INT is the FORTRAN intrinsic function that converts a REAL variable to an INTEGER. Figure 7 shows this situation and AREA curve.

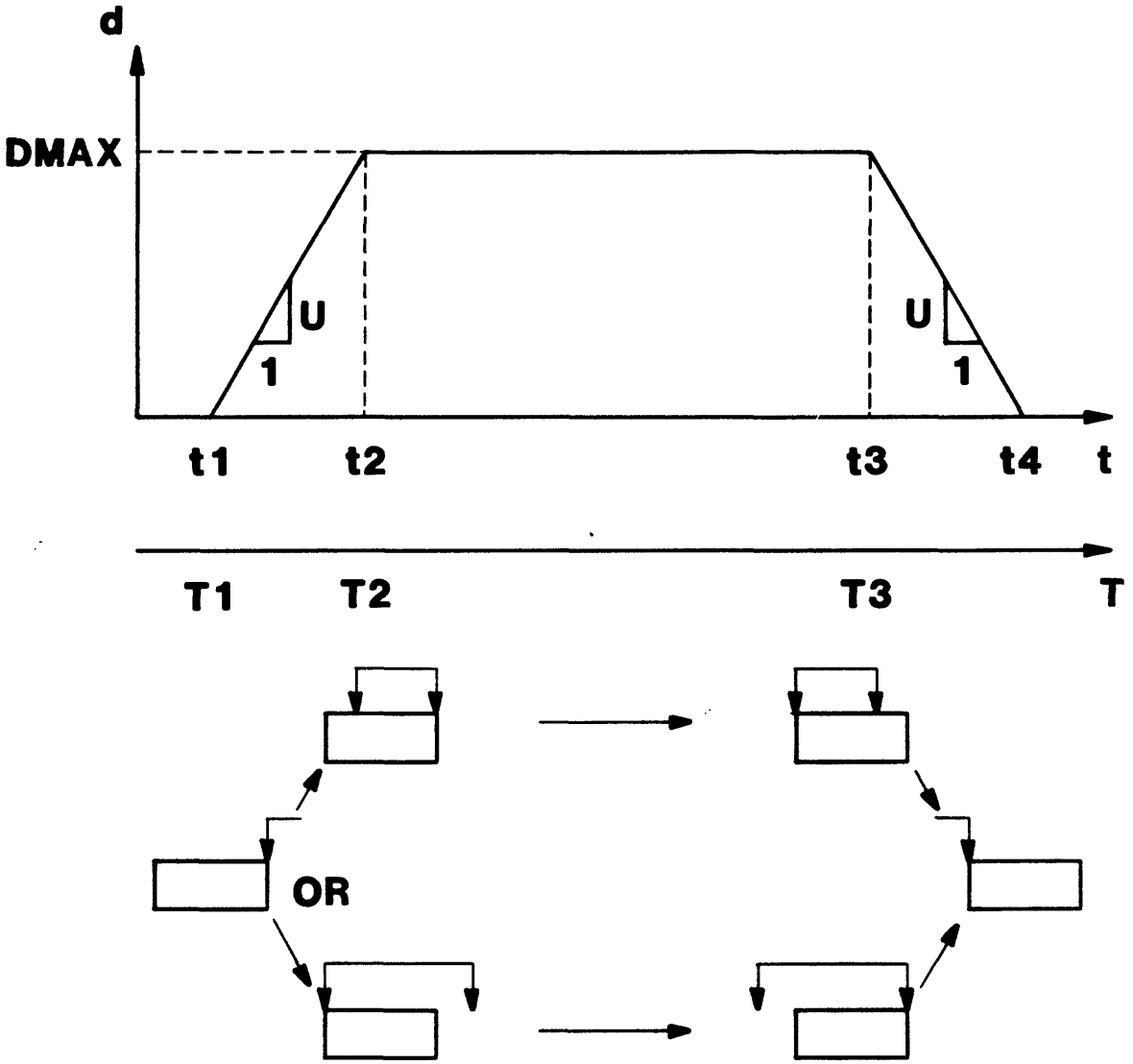


Figure 6.--Parcel distance exposed to lateral inflow vs. time.

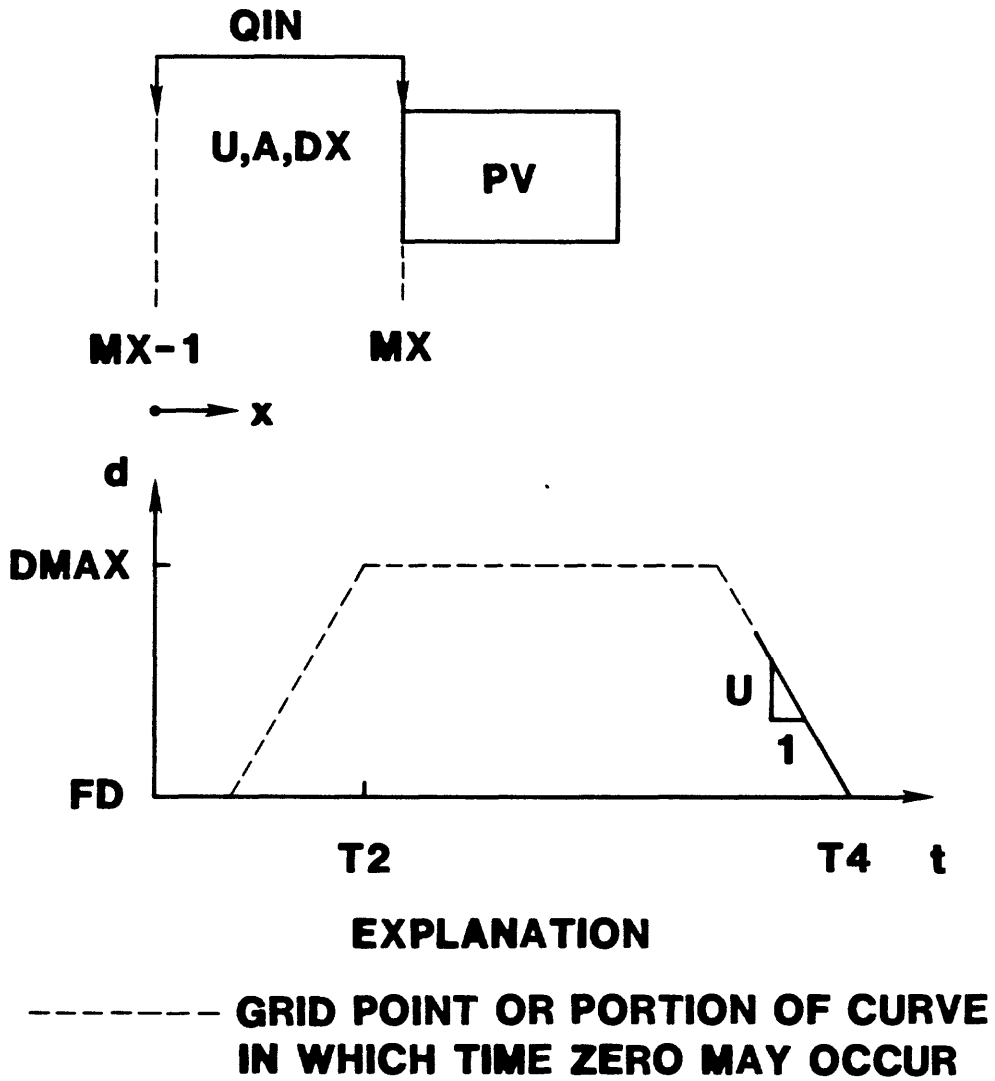


Figure 7.--Parcel leaving a reach with lateral inflow.

When a parcel completes its movement, several different inflow calculations may be necessary. The simplest lateral inflow situation is if the stopped parcel is wholly within the reach with lateral inflow. The values given to AREA are

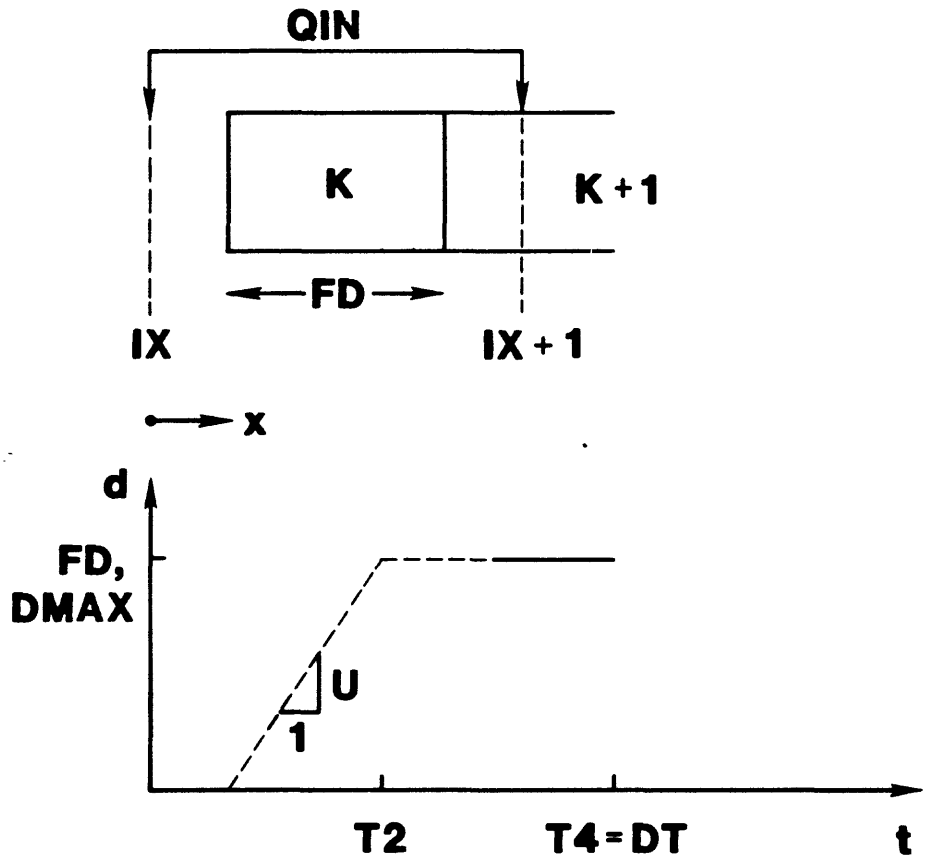
```
T4 = DT

FD = (PX(K+1) - PX(K))*DX(IX)

U = U(IX)

DMAX = FD

T2 = TLPP(IX)                                                    (20)
```

Figure 8 shows this situation and AREA curve.

Another possibility is that the parcel's upstream boundary may be located in a reach with lateral inflow as shown in Figure 9 along with the corresponding AREA curve. If TLPP(IX+1) is less than TLPP(IX) then the parcel volume is greater than the reach volume and the values given to AREA are

```
T4 = DT

FD = (IX + 1 - MAX(PX(K),IX))*DX(IX)

U = U(IX)

DMAX = DX(IX)

T2 = TLPP(IX+1)                                                  (21a)
```

in which MAX is the FORTRAN intrinsic function that selects the maximum from a list of variables. If TLPP(IX+1) is greater than TLPP(IX) then the parcel volume is less than the reach volume and the values given to AREA are



EXPLANATION

------ **GRID POINT OR PORTION OF CURVE IN WHICH TIME ZERO MAY OCCUR**

Figure 8.--Parcel stopped wholly within a reach with lateral inflow.

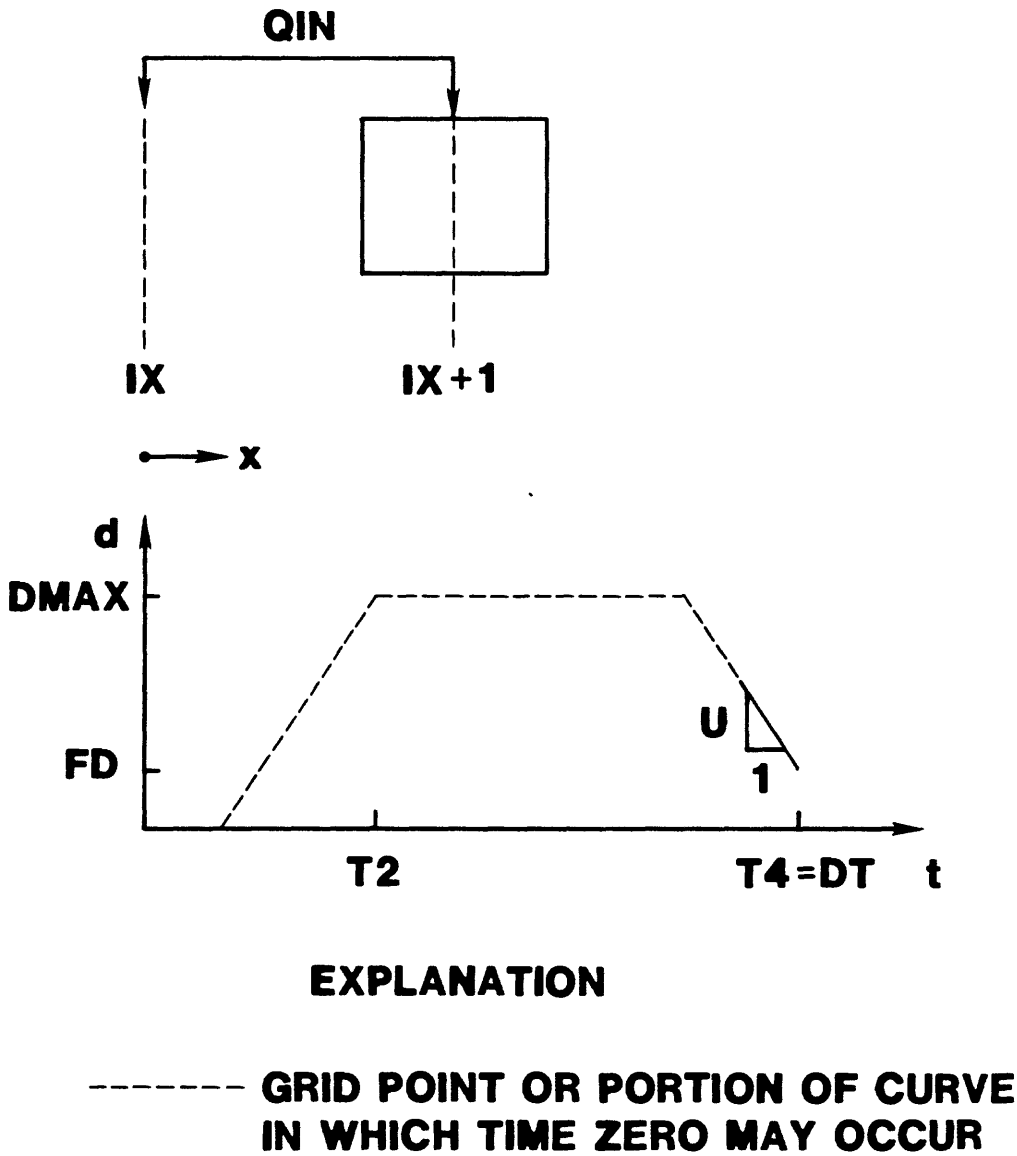


Figure 9.--Parcel stopped with upstream boundary in a reach with lateral inflow.

```
T4 = DT

FD = (IX + 1 - MAX(PX(K),IX))*DX(IX)

U = U(IX)

DMAX = PV(K)/A(IX)

T2 = TLPP(IX)                                                    (21b)
```

Any parcel that finishes the time step containing more than one grid point may also contain several inflows further downstream as shown in Figure 10. The effect of any downstream inflows on the parcel during the current time step must be considered. Proceeding downstream, each reach is checked first for a tributary inflow and then a lateral inflow. For a tributary inflow equation 16 is used to calculate the time of exposure to the inflow. For a lateral inflow a curve similar to that in Figure 8 is constructed by AREA using equation set 21a.

When the reach containing the downstream parcel boundary contains a tributary as shown in Figure 5, the time of exposure to the inflow is given by equation 16. When the downstream reach contains a lateral inflow the values passed to function AREA are

```
T4 = DT

FD = (PX(K+1) - IX)*DX(IX)

U = U(IX)

DMAX = FD

T2 = DT                                                          (22)
```

This situation is shown in Figure 11 along with the corresponding AREA curve.

In addition to calculating parcel concentrations, the concentrations at grid points are calculated separately. If a grid is just upstream from an inflow, as in Figure 5, no effect of the inflow should appear at the grid. But because parcels are uniformly mixed, the parcel concentration is not an appropriate grid concentration for this case. So when a parcel stops, the grid concentration at the most upstream encompassed grid (IX in Figure 5 and IX+1 in Figure 9) is the parcel concentration before mixing plus the effect of a lateral inflow in the reach upstream from the grid. If a parcel encompasses several additional grid points, as in Figure 10, grid concentrations are computed proceeding toward the parcel's downstream

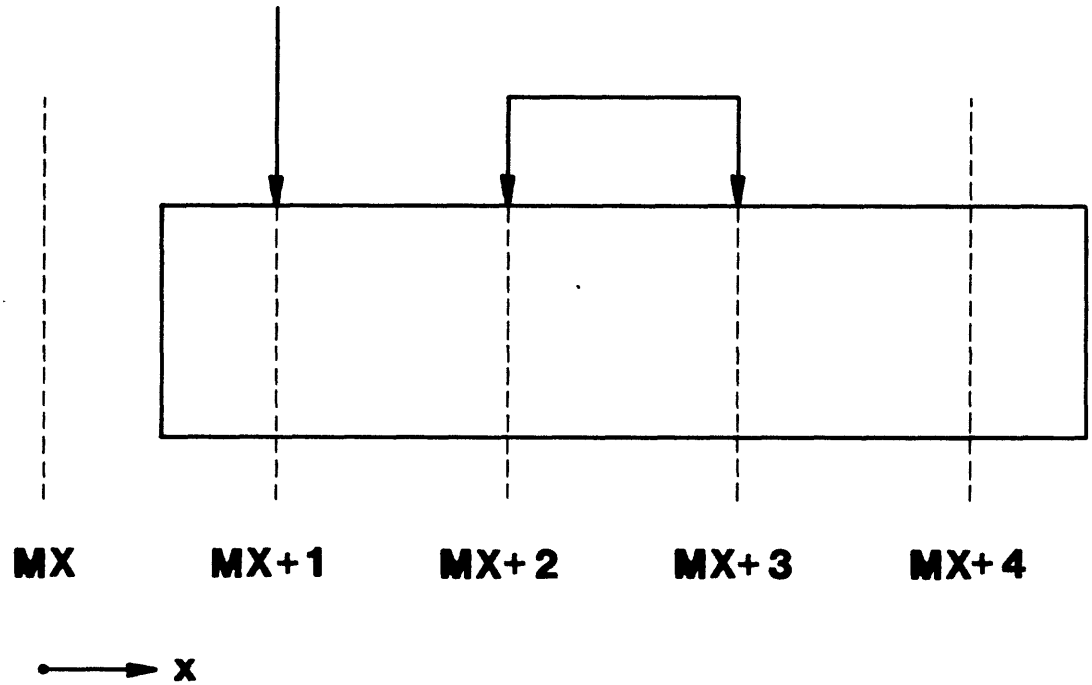


Figure 10.--Parcel spanning several reaches.

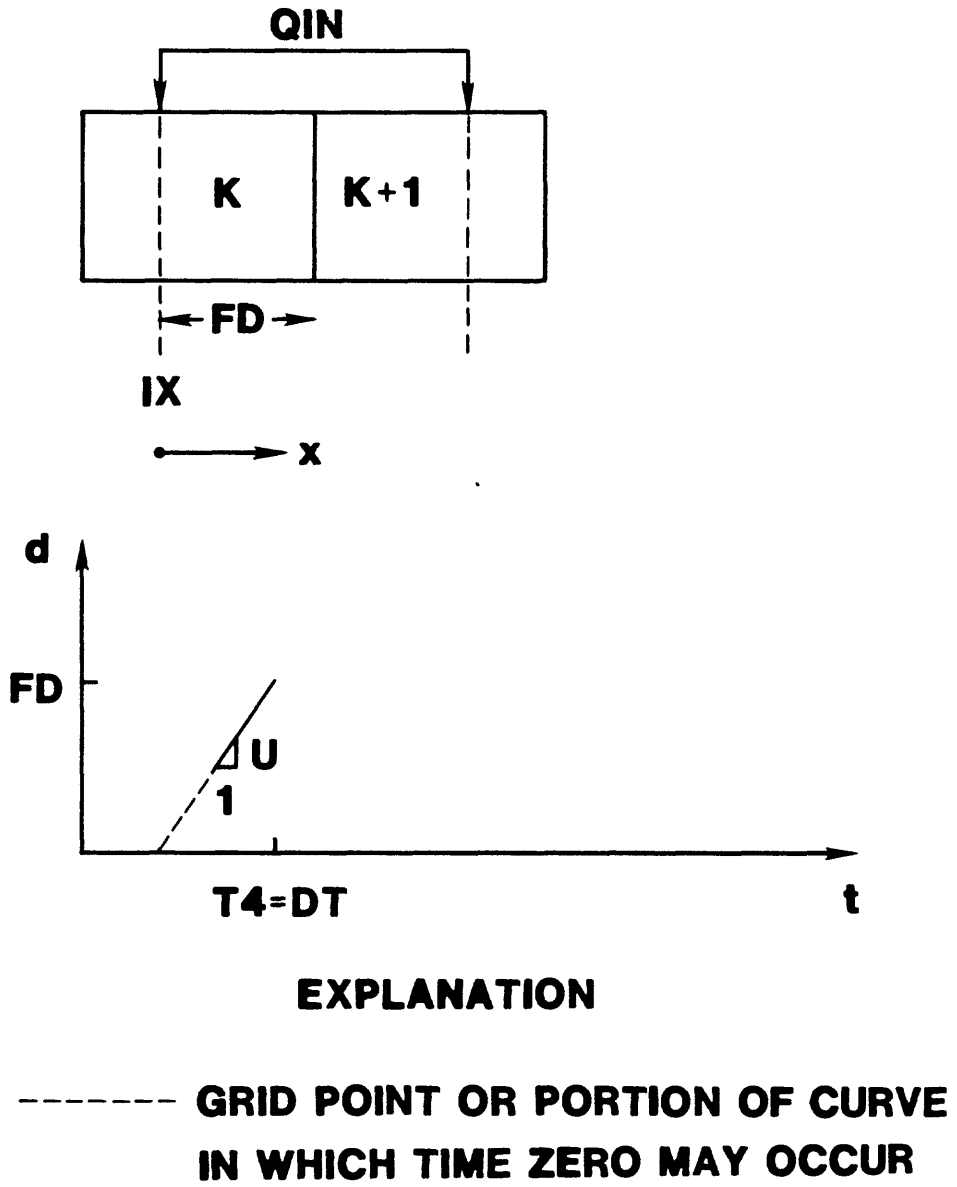


Figure 11.--Downstream parcel boundary in a reach with a lateral inflow.

boundary by considering the cumulative effects of all upstream inflows. For example, at grid MX+2 the unmixed parcel concentration and the tributary inflow determine the grid concentration, while at grids MX+3 and MX+4 the additional effect of the lateral inflow is considered. Calculation of grid concentrations are performed in conjuction with parcel mixing for maximum efficiency. For unsteady flows, however, the effect of an inflow can propagate upstream if the stream velocity decreases because the grid concentrations are recomputed every time step assuming parcels are initially uniformly mixed, as shown in Figure 12.

These algorithms permit consideration of the effects of both point and non-point inflows on parcels. The tributary algorithms are relatively straightforward while the lateral inflow algorithms are more complex due to the need to calculate the curve of exposed parcel distance versus time. Separate grid concentrations are also calculated for increased solution accuracy. Withdrawals are simulated with a negative inflow that has no effect on parcel concentration.

Simulation Procedure

The parcel tracking, dispersion, and inflow algorithms just described are performed by the LTM every simulation time step. The time of day at which the simulation begins (HI), the simulation time step (DT) in hours, and the number of time steps in the simulation (NTS) are chosen by the user. The time step should be chosen such that (1) parcel volumes do not exceed reach volumes, (2) DQQ is less than 0.4 and preferably greater than 0.1, and (3) the number of parcels in the system does not exceed five hundred. Table 1 shows which arrays need to be redimensioned if more parcels are needed. Before the time simulation begins, the input is read (except boundary conditions and unsteady flow field) and processed and the initial conditions are printed.

At the beginning of each time step the unsteady flow field and boundary conditions are read. The boundary conditions, which include upstream concentration and inflow concentrations, are assumed to be the average values during the time step. Next, the effect of dispersion on each parcel (DF) is calculated but the parcel concentrations (PT) are not adjusted until later in the time step. Then all parcels are moved beginning with the most downstream parcel and proceeding to the adjacent upstream parcel. When a parcel either passes a grid point or completes its movement the effect of any inflows it may be receiving is calculated and any decay, which will be discussed later, is also calculated. When a parcel stops movement, DF is added to PT and the grid concentrations are calculated. If output has been requested for the time step the desired grid values are printed and the next time step is begun.

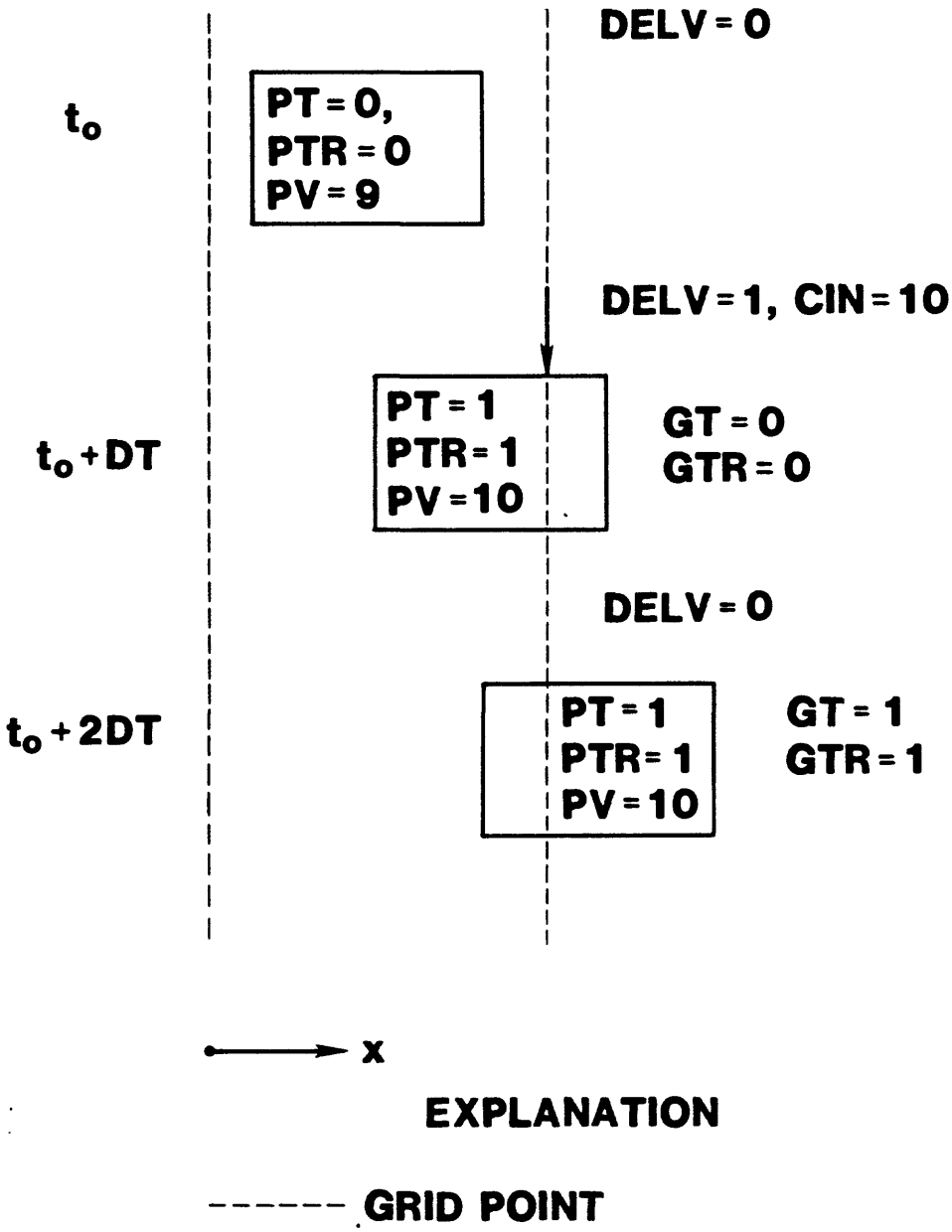


Figure 12.--Grid concentration calculation in unsteady flow.

LAGRANGIAN TRANSPORT MODEL SUBROUTINES

The convection, dispersion, and some of the inflow calculations are performed in the main LTM program in addition to most of the input and output. The remainder of the inflow calculations, some optional input and output including TDDS interfacing, and the effect of decay and constituent interactions within parcels are performed by subroutines and functions described in this section. The user-written subroutine that calculates the decay coefficients will be described later. All subroutines are summarized in Table 2.

Inflow Functions

Two functions are used to help simulate inflows. AREA calculates the area under the exposed parcel length versus time curve for lateral inflows, as described in the previous section. Function DEL calculates the change in parcel concentration with equation 14 and the new parcel volume for either tributary or lateral inflows. Both of these functions are contained in the main LTM source code file LTM.F77 listed in Appendix A.

Optional Output Subroutines

Subroutine STORE stores the grid concentrations for every time step in the array GTS and writes this array to the direct access file GRID.DATA after every fifty time steps. Up to 199 time steps, including data at time zero, can be stored, and if more is needed the arrays listed in Table 1 can be redimensioned. STORE is located in the main LTM source code file LTM.F77. Parcel concentrations and locations can be stored in the direct access file PARCEL.DATA directly from the main LTM program. The formats for both of these files are given in the LTM Users Manual (Schoellhamer and Jobson, 1986).

The user can write a subroutine, called PLOT, that is called from the main LTM program after all calculations are completed but before files are closed. This user-written subroutine is intended to plot data stored in either GRID.DATA or PARCEL.DATA. Another possible use for PLOT is as an LTM post-processor. The call to PLOT in the LTM.F77 has no arguments, but data can be retrieved from either GRID.DATA, PARCEL.DATA, LTM COMMON statements included at the start of PLOT, the TDDS, or other files opened by PLOT. The input variable IPLOT controls subroutine PLOT and the two direct access storage files as described in the LTM Users Manual.

Name	Type	Reference	Purpose
AREA	f	2,A	Area under lateral inflow curve
DADI	e	*,B	Retrieve data from TDDB
DADIO	s	*,B	Initialize TDDS
DADO	e	*,B	Write data to TDDB
DATTIM	s	*,B	TDDS time
DECAY	s	2,A	Runge-Kutta integration for decay
DECAYIN	e	2,A	Read decay coef. output variables
DECAYO	e	2,A	Write decay coefficient output
DECODE	e	*,B	TDDS time
DEL	f	2,A	Mix inflow
DSTIME	s	2,A	Beginning and ending times for TDDS
DTCODE	s	*,B	TDDS time
FINK	s	3,C,D,E	Compute decay coefficients
JDCODE	e	*,B	TDDS time
JULDAT	e	*,B	TDDS time
MOVE	s	*,B	TDDS sort
MOVE1	e	*,B	TDDS sort
MOVE2	e	*,B	TDDS sort
MOVE4	e	*,B	TDDS sort
PLOT	s	2	User-written post-processor
SCRIPT	s	2,B	Convert TDDB data to LTM data
SORTIN	s	*,B	TDDS sort
STORE	s	2,A	Write output to GRID.DATA
TDDS	s	2,B	Read TDDS/LTM interface data
TDDSBC	e	2,B	Retrieve TDDB boundary conditions
TDDSHY	e	2,B	Retrieve TDDB hydraulic data
TDDSO	e	2,B	Write output to TDDB
TDDSRMS	e	2,B	RMS summation with TDDB data
TIME	e	*,B	TDDS time

Notes:
e, entry point
f, function
s, subroutine
A, see Appendix A
B, see Appendix B
C, see Appendix C
D, see Appendix D
E, see Appendix E
2, see section II
3, see section III
*, see Krug and others (written commun., 1983)

Table 2.--Lagrangian Transport Model subroutines

Time-Dependent Data System Subroutines

Krug and others (written commun., 1983) have written a series of subroutines, called the Time-Dependent Data System (TDDS), that store and retrieve data from a time-dependent data base (TDDB). This section will first briefly describe the TDDS subroutines that manipulate the TDDB and then describe the LTM subroutines that interface the LTM to the TDDS.

The main subroutine for manipulating the TDDB is called DADIO. This subroutine must be called before any data manipulations because it initializes several variables. DADIO contains two important entry points. DADI must be called to retrieve data from the TDDB and DADO must be called to store data in the TDDB. DADIO, DADI, and DADO are all called from the LTM/TDDS interface subroutine to be described later in this section.

Subroutine DADIO may call many subroutines and entry points. DATTIM, DECODE, DTCODE, JDCODE, JULDAT, and TIME all manipulate a given time in various ways. MOVE, MOVE1, MOVE2, MOVE4, and SORTIN manipulate and sort data. These subroutines and DADIO, which comprise the TDDS, are at the end of the file TDDS.F77 which is listed in Appendix B.

At the beginning of the file TDDS.F77 is the code needed for the LTM to interface with the Time-Dependent Data System (TDDS). Several entry points are contained in the file TDDS.F77. The main subroutine TDDS is called to read and initialize the TDDS data contained in the file LTM.INPUT described in the LTM Users Manual. The entry point TDDSBC in subroutine TDDS is called by the LTM to retrieve boundary condition data. This data includes upstream boundary concentrations, inflow concentrations, measured concentrations for the TDDS root-mean-squared (RMS) error calculation, and meteorological conditions for the decay-coefficient subroutine. Entry point TDDSHY is called to retrieve hydraulic data including discharge, area, top width, tributary inflows, and lateral inflows. Both TDDSBC and TDDSHY are called for the initial conditions and once every time step. Every ten time steps (0, 10, 20, ...) these entry points call subroutine DADI to retrieve data from the TDDB for the next ten time steps. DADI is called every ten time steps to reduce the I/O time of a program run from that if DADI were called every time step. This increases the required core storage but improves overall program efficiency. This retrieval scheme requires that a TDDS data point fall on every tenth LTM time step, as mentioned in the LTM Users Manual (Schoellhamer and Jobson, 1986).

Entry point TDDSRMS sums the values needed to compute an RMS error with measured data from the TDDS. TDDSRMS is called every time step an RMS calculation is desired, as controlled by the input variable IRMS. Entry point TDDSO sends data stored in GRID.DATA to the TDDB and is called only at the end of the simulation. If output is sent to the TDDB, file GRID.DATA will be created independent of the value of IPLOT which normally determines the output storage options.

Subroutine TDDS calls two other subroutines in order to interface with the TDDS subroutines. Subroutine SCRIPT determines how data that is available at one time step in the TDDB is to be converted to the LTM time step with linear interpolation. Thus the LTM and TDDB time steps do not have to be equal. Subroutine DSTIME determines the beginning and ending times of a data retrieval from the TDDS. This permits the retrieval of enough data for ten LTM time steps. Both SCRIPT and DSTIME are in the file TDDS.F77 after subroutine TDDS and before DADIO.

If the TDDS is not used then the file DUMMY.TDDS.F77, which contains a dummy TDDS subroutine with dummy entry points, may be used instead.

Subroutine DECAY

Subroutine DECAY calculates the decay and interactions of the modeled constituents and is called whenever a parcel passes a grid point or completes movement for the time step. The time span over which decay occurs (DT in DECAY) is the elapsed time interval since the decay was last calculated for the parcel. DECAY sets the decay coefficients XK, CR, and S to zero and calls the decay-coefficient subroutine FINK which calculates the decay coefficients from LTM variables and values read from an external file. Subroutine FINK will be discussed later. These coefficients are used to calculate the decay rate of each constituent at the beginning of the decay time step,

$$\frac{\partial PT}{\partial t} = S_L + \sum_{N=1}^{NEQ} XK_{L,N} \left(PT_N - CR_{L,N}\right) \tag{23}$$

where L is the constituent number, N is a counter, NEQ is the total number of constituents modeled, XK is an exchange coefficient, CR is an equilibrium concentration, and S is a source flux of constituent L.

Then DECAY determines the maximum decay time step (DTM) such that in the time DTM no constituent changes by more that ten percent of the remaining deficit (|PT-CR|) if the deficit is greater than 0.3 concentration units. If the deficit is less than or equal to 0.3 units the decay time interval does not have to be subdivided for that constituent. A simple first order Runge-Kutta integration (also called Euler's method) is used to evaluate the change in each constituent during DTM. The initial parcel concentrations are stored in array PT1 and the parcel concentrations at the end of DTM are predicted with the Euler predictor

$$PT = PT + DKPHR1*DTM \tag{24}$$

FINK is again called using these new parcel concentrations and the decay rates at the end of the time step, DKPHR2, are predicted. The Euler corrector used to calculate the 'correct' parcel concentrations at the end of DTM is

$$PT = PTI + 0.5*DTM*(DKPHR1 + DKPHR2) \quad (25)$$

The changes in parcel concentrations due to decay (PDC) are updated. If the entire duration of decay has not been covered, FINK is called, a new DTM is calculated, and a Runge-Kutta integration is again performed until the entire time of decay has been considered.

Subroutine DECAY also can write the decay coefficients to the output file LTM.OUTPUT. The number of specified coefficient outputs (NCO, maximum of twenty) is read by the main LTM program. If NCO is greater than zero, entry point DECAYIN of subroutine DECAY is called to read both a time and a grid output control value for each specified output. The output time control is accomplished with IPT. If IPT is greater than zero then output will occur at time step IPT, provided IPT is a multiple of IOUT, and if IPT is zero then output will occur at every normal output time step. Similarly, if the output grid control IPG is positive, and contained in NGOUT, output will occur for grid IPG and if IPG equals zero output will occur for all grids. Thus output can be specified for a particular time and grid, all times at one grid, one time at all grids, or all times at all grids.

If NCO is greater than zero, entry point DECAYO is called after the normal output at a grid has been written to determine if decay-coefficient output has been requested for that particular grid and time. If requested, DECAYO writes one table for each coefficient array. FINK is called at the end of the time step to compute the coefficients in the reach downstream from the grid point. This option is useful for checking that the calculated decay coefficients are reasonable and, if not, for determining which coefficient is troublesome.

DECAY-COEFFICIENT SUBROUTINE

The decay-coefficient subroutine FINK calculates the decay coefficients used by subroutine DECAY which was described at the end of the last section. The coefficients, XK, CR, and S, describe how each constituent decays and reacts with other constituents. Their calculation is crucial to successful LTM modeling, so a subroutine may have to be written for each application of the LTM. This section will discuss the three subroutines used as examples in the LTM Users Manual (Schoellhamer and Jobson, 1986) -- conservative constituents with CON.F77, oxygen modeling with OXYGEN.F77, and the QUAL II reaction kinetics in QUAL2.F77. The development of OXYGEN.F77 will be presented in detail. Instructions are given at the end of the section for a user to write his own decay-coefficient subroutine.

CON.F77

The file CON.F77 models up to ten conservative constituents and is listed in Appendix C. This file also contains instructions for writing a FINK subroutine for ten independent constituents that decay with a first order reaction. This subroutine contains the subroutine name, comment statements, two COMMON statements, and one REAL declaration statement. Next are comment statements that delineate the user-defined portion of the subroutine. In this case, this portion contains only the instructions for the first order model and the program immediately returns to DECAY. Because the constituents are conservative, all the decay coefficients should be equal to zero (see equation 23), which they are initially set to in subroutine DECAY, so conservative reaction kinetics are achieved with CON.F77. The dummy FINK subroutine in CON.F77 is a good starting point for a user who writes his own decay-coefficient subroutine, which will be described later.

OXYGEN.F77

The reaction kinetics for a four constituent oxygen model of conservative dye, temperature, biochemical oxygen demand (BOD), and dissolved oxygen (DO) are contained in the decay-coefficient subroutine FINK in file OXYGEN.F77 listed in Appendix D. The reaction kinetics, derived in the LTM Users Manual (Schoellhamer and Jobson, 1986), are

$$\frac{\partial PT(1,KO)}{\partial t} = 0$$

$$\frac{\partial PT(2,KO)}{\partial t} = \frac{-XKX*W(MX)}{CPR*A(MX)}(PT(2,KO) - TA)$$

$$\frac{\partial PT(3,KO)}{\partial t} = -TCK1*PT(3,KO)$$

$$\frac{\partial PT(4,KO)}{\partial t} = -TCKI*PT(3,KO) - \frac{CK4}{A(MX)} + TCK2*(CR(4,4)-PT(4,KO)) \qquad (26)$$

These reaction kinetics are written here with the actual variable names used in the code. Therefore, PT(1,KO) is the dye concentration in parcel KO, t is time, W(MX) is the average cross-sectional top width in the reach downstream from grid MX, A(MX) is the average cross-sectional area in the reach downstream from grid MX, XKX is a function of water temperature and wind speed derived by Jobson (1981), PT(2,KO) is the water temperature

(Celsius) in parcel KO, TA is either the equilibrium water temperature or the air temperature (Celsius), CPR is the product of water density, the specific heat of water, and a unit conversion, PT(3,KO) is the BOD concentration in parcel KO, TCK1 is the temperature dependent BOD decay rate (1/day), PT(4,KO) is the DO concentration in parcel KO, TCK2 is the temperature dependent reaeration coefficient (1/day), CK4 is the benthos source rate for BOD (mg/day/ft), and CR(4,4) is the DO equilibrium concentration.

Both TCK1 and TCK2 are temperature dependent and the dependency is calculated by function T in the code. Function T is

$$T = X(PT - 20.0)^{THETA} \tag{27}$$

in which T is the temperature adjusted coefficient, X is the coefficient at 20°C, THETA is a constant equal to 1.047 for TCK1 and 1.0159 for TCK2, and PT is the water temperature of the parcel.

Because dye is assumed to be conservative, all of the decay coefficients for dye are zero. Temperature, however, decays with nonzero decay coefficients

```
XK(2,2)= -XKX*W(MX)/(A(MX)*CPR)

CR(2,2)=TA                                                    (28)
```

in which XK(2,2) is the decay rate dependent on water temperature, windspeed (both thru XKX), and hydraulics while CR(2,2) is the equilibrium temperature TA. Temperature decay is independent of the concentrations of the other constituents. The nonzero BOD decay coefficient is given by

```
TCK1=T(PT(2,KO),1.047,CK1)

XK(3,3)=-TCK1/24.0                                            (29)
```

in which the temperature dependency function T modifies the BOD decay rate CK1, the equilibrium BOD concentration is zero, and TCK1 is converted from units of 1/day to 1/hour. The nonzero DO decay coefficients are

```
XK(4,4)=-T(PT(2,KO),1.0159,CK2)/24.0

CR(4,4)=468.0/(PT(2,KO)+31.6)

XK(4,3)=-TCK1/24.0                                          (30)

S(4)=-CK4/(24.0*28.316*A(MX))
```

in which an equilibrium DO concentration is calculated with the empirical relationship given in the LTM Users Manual (Schoellhamer and Jobson, 1986), the other constants are for unit conversions, and A is in square feet. Note that the nonzero coefficient XK(4,3) makes the decay of constituent 4 (DO) dependent on the concentration of constituent 3 (BOD) and a source coefficient S is used for the benthic DO demand.

The FINK subroutine in OXYGEN.F77 has comment, COMMON, and declaration statements as found in CON.F77 but the user-defined portion of the subroutine is not empty. Several declaration statements are listed for the user-supplied variables in the subroutine. Only on the first call to FINK will the file OXYGEN.INPUT be read for coefficient input. The meteorological conditions are input via TDDSBC on the first call to FINK of every time step. The code contains instructions on how to input meteorological data from OXYGEN.INPUT if the TDDS is not used. Subroutine FINK concludes with the definition of the decay coefficients. Function T is at the end of OXYGEN.F77.

QUAL2.F77

The file QUAL2.F77 contains a decay-coefficient subroutine that uses the reaction kinetics of the QUAL II water-quality model (Roesner and others, 1977a and 1977b). This file is listed in Appendix E. Schoellhamer (in press) shows that this subroutine allows the LTM to mimic the QUAL II model results for an equally posed problem. The LTM, however, has the advantage of modeling transport with unsteady constituent loads and unsteady flow using Lagrangian calculations. The subroutine is currently set up to use the QUAL II temperature algorithm but the code contains instructions on how to use Jobson's (1981) simplified temperature algorithm instead. File QUAL2.F77 also contains the QUAL II subroutine HSNET which is used to calculate the net solar radiation.

Writing a Decay-Coefficient Subroutine

The user may wish to use only some of the reaction kinetics provided in the given decay-coefficient subroutines, the reaction kinetics provided for a constituent may be unsatisfactory, or none of the three subroutines may simulate the constituents that a user desires to model. Therefore the user may want to write his own set of reaction kinetics. This can be accomplished relatively easily because only the decay-coefficient subroutine has to be adjusted. The LTM was originally designed to force the user to write his own subroutine to prevent the use of the model as a black box (Jobson, 1981). Before attempting to write the subroutine, the user must understand how the decay-coefficient subroutine is used. Refer to the decay section, the DECAY subroutine section, and the OXYGEN.F77 section for more information.

The files CON.F77, OXYGEN.F77, and QUAL2.F77 provide excellent starting points for writing a new decay-coefficient subroutine. Start with QUAL2.F77 if a variation of the QUAL II reaction kinetics is desired, OXYGEN.F77 if either a variation to those kinetics is to be made or data is to be read from a separate input file or with the TDDS, or CON.F77 if the user wishes to start with just the basic FINK structure or a first order decay model for up to ten independent constituents.

Any of the variables in the COMMON statements may be used to write the reaction kinetics. These variables include the concentration of other constituents (PT), cross-sectional area (A), top width (W), mean velocity (U), elapsed simulation time (TIME), initial hour of the simulation (HI), or the time step (J). All of the variables are defined in the comment statements at the beginning of the subroutine. If an LTM variable is needed but not included in the COMMON statements then either the variable can be added to an existing COMMON block or a new COMMON block can be created. Note that all common blocks with the same name must be changed identically through out all of the codes that contain the altered COMMON block. The top width, W, is averaged in every reach and, if the simulation is of unsteady flow, for every time step by the main LTM program. But top width is not used in the main LTM code for any other calculations, so accurate top width data is not needed if it is not used in FINK and, in fact, the array W can be used for any other reach and time averaged value needed in FINK. Depth and wetted perimeter are two examples of data that could be input in place of top width. If the top width data is replaced, the substitution should be noted in field zero of the top width cards in LTM.INPUT.

Data for defining decay coefficients can also be read from a second input file such as OXYGEN.INPUT or QUAL2.INPUT. The subroutine in OXYGEN.F77 contains code that can read a file at the start of the simulation only or at the start of every new time step. File QUAL2.F77 contains code in function HSNET that reads data every three simulation hours.

Another source of data for the decay-coefficient subroutine is the TDDS. Meteorological data types are atmospheric pressure (AP), atmospheric temperature (AT), incoming atmospheric radiation (RA), incoming solar radiation (RS), and wind speed (WS). An example of how to retrieve atmospheric temperature and wind speed data with the TDDS can be found in OXYGEN.F77. Although these data types have specific names, any kind of time-dependent data can be stored under these names if the actual data type stored under the false name is properly and permanently recorded.

The reaction kinetics can be nonlinear because any coefficient can be any function of any variable. For example, if the reaeration rate in equation 26 of the oxygen modeling example were a function of the DO deficit squared instead of varying linearly with the DO deficit, then either XK(4,4) could be proportional to the DO deficit or XK(4,4) could equal zero and the product of the reaeration coefficient and the DO deficit squared could be added to S(4).

A helpful tool for testing a new decay-coefficient subroutine is the decay-coefficient output option discussed in the LTM Users Manual (Schoellhamer and Jobson, 1986). Other debugging aids include the standard output in LTM.OUTPUT and any output options the user writes in the decay-coefficient subroutine.

Simulation of the actual constituents and reaction kinetics in a river may not be possible with the reaction kinetics provided with the LTM code. Use of a user-written decay coefficient subroutine greatly increases the versatility of the LTM. Schoellhamer and Curwick (1986) used the LTM and several subroutines to test various suspended-sediment deposition and erosion functions with Mississippi River data. Jobson (in press) uses the LTM to calculate longitudinal dispersion and desorption coefficients. These two examples and the three examples in the LTM Users Manual show the versatility of the LTM.

SUMMARY

A Programmers Manual for the Lagrangian Transport Model (LTM) has been presented for the LTM user who wishes either to learn the details about the LTM or to make changes to the existing LTM codes. The Lagrangian modeling techniques used to simulate convection, dispersion, and tributary and lateral inflows have been presented in detail. The LTM may call subroutines to calculate and mix inflows, store and output data, and retrieve and write data with a Time-Dependent Data System (TDDS). Decay is simulated with a subroutine that uses a first order Runge-Kutta scheme to integrate constituent decay rates. The coefficients that quantify the constituent decay rates are defined in a subroutine that can and possibly should be written by the user. Instructions and suggestions for writing this subroutine are given. The user-written decay-coefficient subroutine allows the LTM to simulate virtually any constituent and any known reaction

kinetics in a river. This versatility is a valuable asset to the study of riverine water quality. Appendices give the computer codes for the LTM, TDDS interface subroutine, and the three example decay-coefficient subroutines.

REFERENCES

Cunge, J. A., Holly, F. M., and Verwey, A., 1980, Practical aspects of computational river hydraulics: London, England, Pitman Publishing Ltd., p. 320-331.

Gray, W. G., and Pinder, G. F., 1976, An analysis of the numerical solution of the transport equation: Water Resources Research, v. 12, no. 3, p. 547- 555.

Grenney, W. J., Teuscher, M. C., and Dixon, L. S., 1978, Characteristics of the solution algorithms for the QUALII river model: Journal of the Water Pollution Control Federation, v. 50, no. 1, p. 151-157.

Gresho, P. M., and Lee, R. L., 1981, Don't suppress the wiggles--they're telling you something: Computers and Fluids, v. 9, no. 2.

Jobson, H. E., 1980, Comment on A new collocation method for the solution of the convection-dominated transport equation, by Pinder, George E. and Shapiro, Allen, 1979, (in Water Resources Research, v. 15, no. 5, p. 1177-1182): Water Resources Research, v. 16, no. 6, p. 1135-1136.

----- 1981, Temperature and solute-transport simulation in streamflow using a Lagrangian reference frame: U.S. Geological Survey, Water Resources Investigations 81-2, 165 p.

----- in press, Estimation of dispersion and first order rate coefficients by numerical routing: Water Resources Research.

McBride, G. B., and Rutherford, J. C., 1984, Accurate modeling of river pollutant transport: Journal of Environmental Engineering, American Society of Civil Engineers, v. 110, no. 4, p. 808-827.

Roesner, L. A., Giguere, P. R., and Evenson, D. E., 1977a, Computer program documentation for the stream quality model QUAL-II: prepared by Water Resources Engineers Inc., Walnut Creek, Calif., for Southeast Michigan Council of Governments, Detroit, Michigan.

----- 1977b, Users manual for the stream quality model QUAL-II: prepared by Water Resources Engineers, Inc., Walnut Creek, Calif., for Southeast Michigan Council of Governments, Detroit, Michigan.

Schoellhamer, D. H., in press, "Lagrangian Transport Modeling with QUALII Kinetics," Journal of Environmental Engineering, American Society of Civil Engineers.

Schoellhamer, D. H., and Curwick, P. B., 1986, Selected functions for sediment-transport models: Proceedings of the Fourth Federal Inter-Agency Sedimentation Conference, Las Vegas, Nevada, March 24-27, 1986, v. 2, p. 6-157 - 6-166.

Schoellhamer, D. H., and Jobson, H. E., 1986, Users Manual for a One-Dimensional Lagrangian transport model: U.S. Geological Survey Water-Resources Investigation 86-4145, p.

van Genuchten, M Th., and Gray, W. G., 1978, Analysis of some dispersion corrected numerical schemes for solution of the transport equation: International Journal for Numerical Methods in Engineering, v. 12, p. 387-404.

Varoglu, E., and Finn W. D. L., 1978, A finite element method for the diffusion-convection equation: Proceedings of the Second International Conference on Finite Elements in Water Resources, v. 2, p. 4.3-4.20.

APPENDIX A

LTM.F77

Lagrangian Transport Model

```
**************ONE-DIMENSIONAL LAGRANGIAN TRANSPORT MODEL****************
*
***********************************************************************
*                                                                     *
*             ONE-DIMENSIONAL LAGRANGIAN TRANSPORT MODEL              *
*                                                                     *
*               BY DAVID SCHOELLHAMER AND HARVEY JOBSON               *
*                                                                     *
*                   UNITED STATES GEOLOGICAL SURVEY                   *
*                   GULF COAST HYDROSCIENCE CENTER                    *
*                            BUILDING 2101                            *
*                       NSTL STATION, MS  39529                       *
*                                                                     *
*                             MARCH 1985                              *
*                                                                     *
***********************************************************************
C
C
C*****DEFINE FORTRAN UNIT NUMBERS*************************************
C
C     UNIT NUMBER   FILE NAME                PURPOSE
C     -----------   ---------   ----------------------------------------
C         30      UNSTEADY.DATA  CONTAINS ARRAY 'FLOW' FOR EVERY TIME
C                                STEP OF AN UNSTEADY SIMULATION NOT
C                                USING THE TDDS OPTION
C
C         31        GRID.DATA    DIRECT ACCESS FILE THAT STORES GRID
C                                CONCENTRATIONS AT EVERY TIME STEP IF
C                                REQUESTED BY USER OR NEEDED BY TDDS
C
C         32       PARCEL.DATA   DIRECT ACCESS FILE THAT STORES PARCEL
C                                CONCENTRATIONS AND POSITIONS AT EVERY
C                                TIME STEP IF REQUESTED WITH IPLOT
C
C         35        LTM.INPUT    BASIC MODEL INPUT
C
C         36        LTM.OUTPUT   MODEL OUTPUT
C
C         37            -        RESERVED FOR USE IF PUNCH OUTPUT IS
C                                DESIRED FROM THE TDDS
C
C         38            -        DATA STATION REFERENCE FILE FOR TDDS
C
C         39        TOBS.INPUT   CONTAINS OBSERVED CONCENTRATIONS THAT
C                                ARE USED TO CALCULATE RMS ERROR IF
C                                REQUESTED
C
C         98            -        FILE CONTAINING TIME DEPENDENT DATA IF
C                                TDDS OPTION IS USED
C
C   **NOTE: FILE NUMBERING BEGINS AT 30 TO ALLOW USE OF DISSPLA GRAPHICS
C           IN THE USER DEFINED SUBROUTINE PLOT IF DESIRED
C
C   **NOTE: UNITS 37, 38, AND 98 ARE TO BE OPENED EXTERNALLY WHEN NEEDED
C
C
C*****DEFINE FUNCTIONS AND SUBROUTINES*******************************
C
C      NAME                          PURPOSE
C     -------    ----------------------------------------------------
C      AREA      COMPUTE THE AREA UNDER A CURVE OF PARCEL LENGTH
C                EXPOSED TO LATERAL INFLOW VERSUS TIME
C
C      DECAY     COMPUTE EFFECT OF DECAY ON PARCEL CONCENTRATIONS
C
C     DECAYIN    READ INPUT FOR DECAY COEFFICIENT OUTPUT CONTROL
C
C     DECAYO     WRITE DECAY COEFFICIENT OUTPUT
C
C       DEL      COMPUTE CHANGE IN PARCEL CONCENTRATION DUE TO INFLOW
C
C      FINK      COMPUTE DECAY COEFFICIENTS
```

```
C
C      PLOT       USER DEFINED SUBROUTINE TO PLOT RESULTS THAT IS CALLED
C                 IF IPLOT IS NEGATIVE
C
C      STORE      STORES GRID CONCENTRATIONS IN FILE GRID.DATA (UNIT #31)
C                 IF EITHER REQUESTED WITH IPLOT OR NEEDED BY TDDS
C
C      TDDS       INITIALIZES VALUES USED WITH TDDS OPTION
C
C     TDDSBC      RETURNS BOUNDARY CONDITIONS FROM TDDS IF REQUESTED
C
C     TDDSHY      RETURNS ARRAY 'FLOW' FROM TDDS IF REQUESTED
C
C      TDDSO      WRITES GRID CONCENTRATIONS WITH TDDS IF REQUESTED
C
C     TDDSRMS     CALCULATES RMS ERROR FROM TDDS DATA IF IRMS IS NEGATIVE
C
C   **NOTE: THE FIVE TDDS SUBROUTINES ARE CONTAINED IN THE SEPERATE FILE
C           TDDS.F77 AND FINK IS ALSO IN A SEPERATE FILE
C
C
C*****DEFINE ARRAYS*****************************************************
C
C        ARRAY                          PURPOSE
C     ----------- ------------------------------------------------------
C        A(G)      AVERAGE AREA OF REACH DOWNSTREAM FROM GRID G [SQ M]
C
C    >CIN(C,G,I)<  FOR CONSTITUENT C:
C                  IF I=1, CONCENTRATION OF TRIBUTARY INFLOW AT GRID G
C                  IF I=2, CONCENTRATION OF LATERAL INFLOW IN REACH
C                          DOWNSTREAM FROM GRID G
C
C      CINC(G,I)   IF I=1, CONCENTRATION OF TRIBUTARY INFLOW AT GRID G
C                  IF I=2, CONCENTRATION OF LATERAL INFLOW IN REACH
C                          DOWNSTREAM FROM GRID G
C
C       DELG(C)    CHANGE IN GRID CONCENTRATION OF CONSTITUENT C DUE TO
C                  INFLOW
C
C       DF(C,P)    CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C                  DUE TO DISPERSION IN THE CURRENT TIME STEP
C
C       DMF(P)     THE MASS FLUX INTO PARCEL P AT ITS UPSTREAM BOUNDARY
C
C        DQ(P)     FLOW RATE BETWEEN PARCELS P AND P-1 DUE TO VELOCITY
C                  GRADIENTS [CUBIC METERS PER HOUR]
C
C      >DQQ(G)<    DISPERSION FACTOR OF REACH DOWNSTREAM FROM GRID G
C
C        DX(G)     DISTANCE BETWEEN GRIDS G AND G+1 [METERS]
C
C    >FLOW(H,G)<   VALUE OF HYDRAULIC PROPERTY H AT OR BELOW GRID G:
C                       FOR H=1, VELOCITY [METERS OR FEET/HOUR],
C                                CALCULATED FOR STEADY STATE
C                                SIMULATIONS AND BY THE TDDS SUBROUTINE
C                       FOR H=2, AREA [SQ METERS OR FEET]
C                       FOR H=3, TOP WIDTH [METERS OR FEET]
C                       FOR H=4, TRIBUTARY INFLOW JUST BELOW GRID G
C                                [CUBIC METERS OR FEET PER SECOND]
C                       FOR H=5, LATERAL INFLOW IN THE REACH DOWNSTREAM
C                                FROM GRID G [SQ METERS OR FEET PER SEC]
C
C      >GT(C,G)<   CONCENTRATION OF CONSTITUENT C AT GRID G, READ AS AN
C                  INITIAL CONDITION
C
C     GTR(C,G,I)   CHANGE IN CONCENTRATION OF CONSTITUENT C AT GRID G
C                  DUE TO:
C                       FOR I=1, TRIBUTARIES
C                       FOR I=2, LATERAL INFLOWS
C
C      GTRMS(G)    CONCENTRATION OF A CONSTITUENT AT GRID G
C
```

```
C      >IGIN(N,I)<  GRID NUMBERS WITH INFLOW WHERE N=1 TO NIN(I),
C                        I=1 FOR TRIBUTARY INFLOW JUST BELOW THE GRID
C                        I=2 FOR LATERAL INFLOW IN THE DOWNSTREAM REACH
C
C        ND(P)       NUMBER OF SUB-TIME STEPS NEEDED AT THE UPSTREAM
C                    BOUNDARY OF PARCEL P FOR DISPERSION
C
C       NSUM(C)      NUMBER OF SQUARED ERROR CALCULATIONS SUMMED FOR
C                    CONSTITUENT C
C
C      >LABEL(C)<    NAME OF CONSTITUENT C
C
C      >NGOUT(N)<    GRID NUMBERS TO BE INCLUDED IN OUTPUT, N=1 TO NGO
C
C       >NIN(I)<     NUMBER OF GRIDS WITH INFLOW TYPE I,
C                        I=1 FOR TRIBUTARY INFLOW JUST BELOW THE GRIDS
C                        I=2 FOR LATERAL INFLOW DOWNSTREAM OF THE GRIDS
C
C       PARA(C)      CHARACTER NUMBER OF CONSTITUENT C, PARA(10)=0
C
C      PDC(C,P)      CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C                    DUE TO DECAY
C
C      PDF(C,P)      CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C                    DUE TO DISPERSION
C
C        PH(P)       TIME PARCEL P ENTERED THE SYSTEM [HRS SINCE MIDNIGHT
C                    OF DAY ONE]
C
C       PPXT(C)      CONCENTRATION OF CONSTITUENT C AT THE PREVIOUS
C                    (UPSTREAM) PARCEL'S UPSTREAM BOUNDARY
C
C       PT(C,P)      CONCENTRATION OF CONSTITUENT C IN PARCEL P
C
C      PTI(C,P)      INITIAL CONCENTRATION OF CONSTITUENT C IN PARCEL P
C
C     PTR(C,P,I)     CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C                    DUE TO:
C                        FOR I=1, TRIBUTARIES
C                        FOR I=2, LATERAL INFLOW
C
C        PV(P)       VOLUME OF PARCEL P [CUBIC METERS]
C
C        PX(P)       POSITION OF PARCEL P IN EULERIAN GRID UNITS
C
C      >QIN(G,I)<    INFLOW BELOW GRID G DUE TO:
C                        FOR I=1, A TRIBUTARY [CUBIC METERS PER SECOND]
C                        FOR I=2, LATERAL INFLOW [SQ M PER SEC]
C
C       SUM(C)       SUM OF SQUARED ERRORS OF CONSTITUENT C
C
C      >TACT(C,G)<   OBSERVED CONCENTRATION OF CONSTITUENT C AT GRID G
C
C      >TITLE(N)<    TITLE OF SIMULATION
C
C      TRASH(N)      DUMMY ARRAY FOR SUBROUTINE CALL, N=5
C
C       TLPP(G)      TIME AN UPSTREAM PARCEL BOUNDARY LAST PASSED GRID G
C                    DURING THE CURRENT TIME STEP
C
C        TT(G)       TRAVEL TIME FROM GRID ONE TO GRID G [HOURS]
C
C       TUP(C)       CONCENTRATION OF CONSTITUENT C AT THE UPSTREAM
C                    BOUNDARY
C
C        U(G)        AVERAGE VELOCITY IN THE REACH DOWNSTREAM FROM GRID G
C                    [METERS/HOUR]
C
C       >W(G)<       AVERAGE TOP WIDTH IN THE REACH DOWNSTREAM FROM
C                    GRID G [METERS]
C
C       >X(G)<       LOCATION OF GRID G, READ IN RIVER MI OR KM, CONVERTED
```

```
C                    TO METERS DOWNSTREAM OF GRID ONE
C
C   **NOTE: >ARRAY< DENOTES ARRAYS DEFINED VIA MODEL INPUT
C
C
C*****DEFINE VARIABLES*****************************************************
C
C      NAME                          PURPOSE
C     ------  ------------------------------------------------------------
C      CONM   CONSTANT THAT CONVERTS INPUT DATA TO METERS
C
C      CONX   CONSTANT THAT CONVERTS INPUT GRID STATIONS TO METERS
C
C      DELP   CHANGE IN PARCEL CONCENTRATION DUE TO LATERAL INFLOW
C
C      DELV   CHANGE IN PARCEL VOLUME DUE TO INFLOW [CUBIC METERS]
C
C      DMAX   A PARCEL'S MAXIMUM POSSIBLE LENGTH OF EXPOSURE TO A
C             LATERAL INFLOW [METERS]
C
C      >DT<   SIMULATION TIME STEP [HOURS]
C
C      DTT    TIME A PARCEL IS EXPOSED TO A TRIBUTARY INFLOW [HOURS]
C
C       FD    A PARCEL'S LENGTH EXPOSED TO A LATERAL INFLOW AT THE END
C             OF THE CURRENT TIME STEP [METERS]
C
C      >HI<   INITIAL HOUR OF THE SIMULATION
C
C       HR    HOUR (0 TO 24) OF THE CURRENT SIMULATION TIME STEP
C
C       I     COUNTER
C
C      IDAY   DAY OF THE CURRENT SIMULATION TIME STEP
C
C      IEXP   EXPONENT OF TWO USED TO DETERMINE A ND(K)
C
C       IN    COUNTER FOR TYPE OF INFLOW, 1 FOR TRIBUTARY, 2 FOR LATERAL
C
C      INX    NUMBER OF REACHES IN SIMULATION
C
C     >IOUT<  TIME STEP INTERVAL AT WHICH OUTPUT IS DESIRED
C
C    >IPLOT<  DEFINES PLOT OPTION:
C                  IF IPLOT=0, DO NOT STORE DATA
C                  IF ABS(IPLOT)=1, STORE GRID DATA
C                  IF ABS(IPLOT)=2, STORE PARCEL DATA
C                  IF ABS(IPLOT)=3, STORE BOTH GRID AND PARCEL DATA
C                  IF IPLOT<0, CALL USER DEFINED SUBROUTINE PLOT
C                  IF IPLOT=>0, DO NOT CALL PLOT
C
C     >IRMS<  DEFINES ROOT MEAN SQUARED ERROR CALCULATION, COMPUTED
C             EVERY ABS(IRMS) TIME STEPS:
C                  IF IRMS=0, DO NOT CALCULATE AN RMS ERROR
C                  IF IRMS>0, CALCULATE AN RMS ERROR USING TOBS.INPUT
C                  IF IRMS<0, CALCULATE AN RMS ERROR USING TDDS DATA
C
C    >IUNSTD< DEFINES STEADY/UNSTEADY OPTIONS:
C                  IF IUNSTD=0, STEADY STATE SIMULATION
C                  IF IUNSTD=1, UNSTEADY SIMULATION USING UNSTEADY.DATA
C                  IF IUNSTD=-1, UNSTEADY SIMULATION USING TDDS DATA
C                  IF IUNSTD=-2, STEADY SIMULATION USING BOUNDARY
C                                CONDITIONS FROM THE TDDS
C                  IF IUNSTD=-3, UNSTEADY SIMULATION USING HYDRAULIC
C                                VALUES FROM THE TDDS
C
C       IX    UPSTREAM GRID NUMBER, USED FOR INFLOWS
C
C       J     TIME STEP
C
C       K     COUNTER, PRIMARILY PARCEL NUMBER
C
```

```
C       KO     OLD PARCEL NUMBER
C
C       L      COUNTER
C
C    >METRIC< DEFINES THE INPUT DATA UNIT SYSTEM:
C                  IF METRIC=0, ENGLISH SYSTEM (LENGTH=FEET)
C                  IF METRIC=1, METRIC SYSTEM (LENGTH=METER)
C
C       MX     UPSTREAM GRID NUMBER, USED WHEN MOVING PARCELS
C
C      NBC     NUMBER OF BOUNDARY CONDITIONS TO BE CONSIDERED BY TDDSBC
C
C     >NCO<    NUMBER OF COEFFICIENT OUTPUTS SPECIFIED
C
C     NDMAX    MAXIMUM NUMBER OF SUB-TIME STEPS NEEDED FOR DISPERSION
C
C     >NEQ<    NUMBER OF CONSTITUENTS IN SIMULATION
C
C       NG     NUMBER OF GRID AT WHICH OUTPUT IS DESIRED
C
C     >NGO<    TOTAL NUMBER OF GRIDS AT WHICH OUTPUT IS DESIRED
C
C     NLINES   COUNTS NUMBER OF LINES OF OUTPUT ON CURRENT PAGE
C
C      NREC    NUMBER OF RECORD OF A DIRECT ACCESS FILE
C
C       NS     NUMBER OF PARCELS IN SYSTEM
C
C     >NTS<    NUMBER OF TIME STEPS IN THE SIMULATION
C
C    >NXSEC<   NUMBER OF GRID POINTS IN SYSTEM
C
C      PDT     TIME REQUIRED FOR A PARCEL'S UPSTREAM BOUNDARY TO TRAVERSE
C              THE REMAINING PART OF THE REACH
C
C      PDX     FRACTION OF CURRENT REACH REMAINING TO BE TRAVERSED BY THE
C              PARCEL'S UPSTREAM BOUNDARY
C
C      >Q<     INITIAL FLOWRATE AT THE UPSTREAM SYSTEM BOUNDARY [CU M/SEC]
C
C       QX     FLOWRATE AT A GRID POINT [CUBIC METERS PER SECOND]
C
C     RATIO    RATIO OF VOLUME MIXED WITH A NEIGHBORING PARCEL TO PV
C
C      RDT     TIME REMAINING FOR PARCEL MOVEMENT [HOURS]
C
C      RKM     RIVER KILOMETER OF A GRID POINT
C
C      RMS     ROOT MEAN SQUARED ERROR OF SIMULATION
C
C      SPV     VOLUME OF PARCEL AFFECTED BY A TRIBUTARY INFLOW [CU M]
C
C       T      TIME WHEN MAXIMUM PARCEL EXPOSURE TO A LATERAL INFLOW
C              BEGAN [HOURS]
C
C      TIME    ELAPSED TIME OF SIMULATION [HOURS]
C
C     XOLD     OLD EULARIAN LOCATION OF A PARCEL'S UPSTREAM BOUNDARY
C
C      XUP     RIVER MI OR KM OF THE SYSTEM'S UPSTREAM BOUNDARY
C
C   **NOTE: >VARIABLE< DENOTES VARIABLES DEFINED VIA MODEL INPUT
C
C
C*****DIMENSION ARRAYS, OPEN FILES*************************************
C
      COMMON /CDECAY/ KO,MX,PT,PDC
      COMMON /CFINK/ A,U,W,TIME,HI,J
      COMMON         CIN,DF,GTR,PDF
      CHARACTER PARA(10)/'1','2','3','4','5','6','7','8','9','0'/,
     1          TITLE(80)
      CHARACTER*7 LABEL(10)
```

```
      DOUBLE PRECISION X(51)
      INTEGER   IGIN(50,2),ND(500),NGOUT(50),NIN(2),NSUM(10)
      REAL      A(50),CIN(10,50,2),CINC(50,2),DELG(10),DF(10,500),
     1          DMF(500),DQ(500),DQQ(50),DX(50),FLOW(5,50),GT(10,50),
     2          GTR(10,50,2),GTRMS(10),
     3          PDC(10,500),PDF(10,500),PH(500),PPXT(10),PT(10,500),
     4          PTI(10,500),PTR(10,500,2),PV(500),PX(500),QIN(50,2),
     5          SUM(10),TACT(10,50),TRASH(5),TUP(10),TLPP(50),TT(50),
     6          U(50),W(50)
      OPEN(UNIT=35,FILE='LTM.INPUT',FORM='UNFORMATTED',RECL=80)
      OPEN(UNIT=36,FILE='LTM.OUTPUT')
C
C*****ZERO VARIABLES AND ARRAYS*****************************************
C
      DO 100 K=1,500                                                    LINE375
       PX(K)= 0.0
       DQ(K)=0.0
       PV(K)= 0.0
       PH(K)= 0.0
       DO 80 I=1,10
        PT(I,K)=0.0
        PTR(I,K,1)= 0.0
        PTR(I,K,2)=0.0
        PDF(I,K)= 0.0
        PDC(I,K)=0.0
        PTI(I,K)= 0.0
        DF(I,K)= 0.0
   80  CONTINUE
  100 CONTINUE
      DO 140 I=1,50                                                     LINE390
       DO 120 IN=1,2
        DO 110 K=1,10
         GTR(K,I,IN)=0.0
         CIN(K,I,IN)=0.0
  110   CONTINUE
        CINC(I,IN)=0.0
        QIN(I,IN)=0.0
        FLOW(3+IN,I)=0.0
        IGIN(I,IN)=0
  120  CONTINUE
       TLPP(I)=0.0
       DO 130 K=1,10
        GT(K,I)=0.0
  130  CONTINUE
       TT(I)=0.0
       NGOUT(I)=0
       X(I)= 0.0
       DQQ(I)= 0.0
       DX(I)= 0.0
       U(I)= 0.0
       W(I)=0.0
       A(I)= 0.0
  140 CONTINUE
      DO 150 I=1,10
       NSUM(I)=0
       SUM(I)=0.0
  150 CONTINUE
      J=0
      IUNSTD=0
      IRMS=0
      IPLOT=0
C
C*****READ INPUT AND ECHO OUTPUT****************************************
C
      READ(35,3500)(TITLE(K),K=1,80)
 3500 FORMAT(80A1)
      WRITE(36,3600) (TITLE(K),K=1,80)
 3600 FORMAT(1H1,80A1,//)
      READ(35,3505) NXSEC,NIN(1),NIN(2),Q,HI,NTS,DT,IUNSTD,NEQ,METRIC
 3505 FORMAT(10X,3I7,2F7.0,I7,F7.0,3I7)
      IF (MOD(ABS(IUNSTD),2) .NE. 0) THEN
        WRITE(36,3605)
```

```
 3605   FORMAT(' SIMULATION IS OF UNSTEADY FLOW')
       ELSE
        WRITE(36,3610)
 3610   FORMAT(' SIMULATION IS OF STEADY FLOW')
      END IF
      IF (METRIC .EQ. 1) THEN
        WRITE(36,3612)
 3612   FORMAT(' INPUT DATA IS METRIC')
       ELSE
        WRITE(36,3613)
 3613   FORMAT(' INPUT DATA IS ENGLISH')
      END IF
      READ(35,3510) IOUT,NGO,IRMS,IPLOT,NCO
 3510 FORMAT(10X,10I7)
      IF (NGO .LE. 0 .OR. NGO .GE. NXSEC) GO TO 160
      READ(35,3510) (NGOUT(I),I=1,NGO)
      GO TO 200
C     OUTPUT DESIRED AT EVERY GRID
  160 DO 180 I=1,NXSEC
       NGOUT(I)=I
  180 CONTINUE
  200 IF (NCO .GT. 0) CALL DECAYIN(NCO)
      READ(35,3513) (LABEL(I),I=1,NEQ)
 3513 FORMAT(10X,10A7)
      WRITE(36,3615) NTS,DT,HI,NXSEC,Q,IOUT,(NGOUT(I),I=1,NGO)
 3615 FORMAT(' MODEL IS TO RUN',I5,' TIME STEPS EACH',F10.2,' HOURS LONG
     1.',/,' THE SIMULATION STARTS ',F7.4,' HOURS AFTER MIDNIGHT.',
     2/,' THE RIVER IS DISCRETIZED BY',I5,' GRID POINTS.',/,
     3' THE INITIAL UPSTREAM DISCHARGE IS CONSTANT AT',F10.2,
     4' CUBIC METERS OR FEET PER SEC.',/,' EVERY',I5,' TIME STEPS DATA
     5WILL BE PRINTED FOR THE FOLLOWING GRIDS:',/,4(14I5,/))
      WRITE(36,3617) (I,LABEL(I),I=1,NEQ)
 3617 FORMAT(1H0,'THE CONSTITUENTS MODELED ARE:',/,10(I4,3X,A7,/))
      READ(35,3515) (X(I),I=1,NXSEC)
 3515 FORMAT(10X,10F7.0)
      IF (MOD(ABS(IUNSTD),2) .EQ. 0) READ(35,3515)(FLOW(2,I),I=1,NXSEC)
      IF (MOD(ABS(IUNSTD),2) .EQ. 0) READ(35,3515)(FLOW(3,I),I=1,NXSEC)
      DO 220 IN=1,2
       IF (NIN(IN).GT.0) READ(35,3520) (IGIN(I,IN),FLOW(IN+3,IGIN(I,IN))
     1                                 ,I=1,NIN(IN))
 3520  FORMAT(10(10X,5(I7,F7.0),/))
  220 CONTINUE
      READ(35,3515) (DQQ(I),I=1,NXSEC)
      DO 225 K=1,NEQ
       READ(35,3515) (GT(K,I),I=1,NXSEC)
  225 CONTINUE
C
C*****CALL TDDS, OPEN FILES, OR READ UNSTEADY.DATA IF REQUESTED*********
C
      IF (IUNSTD .LT. 0 .OR. IRMS .LT. 0) CALL TDDS(NXSEC,DT,NTS,IPLOT)
      IF (IUNSTD .EQ. 1) OPEN(UNIT=30,FILE='UNSTEADY.DATA',RECL=80)
      IF (MOD(ABS(IPLOT),2) .NE. 0) OPEN(UNIT=31,FILE='GRID.DATA',
     1                                   ACCESS='DIRECT',RECL=400)        LINE485
      IF (ABS(IPLOT) .GE. 2) OPEN(UNIT=32,FILE='PARCEL.DATA',
     1                                   ACCESS='DIRECT',RECL=1000)
      IF (IRMS .GT. 0) OPEN(UNIT=39,FILE='TOBS.INPUT',RECL=80)
      IF (IUNSTD .EQ. -1 .OR. IUNSTD .EQ. -3) CALL TDDSHY(FLOW,0)
      IF (IUNSTD .EQ. -1 .OR. IUNSTD .EQ. -2) THEN
       NBC=4
       DO 235 I=1,NEQ
        IF (I .EQ. NEQ) NBC=9
        CALL TDDSBC(PARA(I),NBC,J,CINC,TUP(I),TRASH(1),TRASH(2),
     1              TRASH(3),TRASH(4),TRASH(5))
        DO 230 K=1,NXSEC
         CIN(I,K,1)=CINC(K,1)
         CIN(I,K,2)=CINC(K,2)
  230   CONTINUE
  235  CONTINUE
      END IF
      IF (IUNSTD .EQ. 1) READ(30,3000) ((FLOW(K,I),K=1,5),I=1,NXSEC)
 3000 FORMAT(10F8.0)
C
```

```
C*****PROCESS INPUT DATA*************************************************
C
      IF (METRIC .EQ. 1) THEN
        CONX=1000.0
        CONM=1.0
       ELSE
        CONX=1609.34
        CONM=0.3048
      END IF
      INX=NXSEC-1
      XUP=X(1)
      X(NXSEC+1)=X(NXSEC)
      QX=Q
      DO 240 I=1,NXSEC
       DX(I)=(X(I)-X(I+1))*CONX
       X(I)=(XUP-X(I))*CONX
       IF (MOD(ABS(IUNSTD),2) .EQ. 0) FLOW(1,I)=3600.0*QX/FLOW(2,I)
       QX= QX+FLOW(4,I)+FLOW(5,I)*DX(I)/CONM
       IF (I .EQ. NXSEC) GO TO 240
       QIN(I,1)=FLOW(4,I)*CONM**3
       QIN(I,2)=FLOW(5,I)*CONM**2
       A(I)=(FLOW(2,I)+FLOW(2,I+1))*CONM**2/2.0
       U(I)=(QX*CONM**3-QIN(I,2)*DX(I)/2.0)*3600.0/A(I)
       W(I)=(FLOW(3,I)+FLOW(3,I+1))*CONM/2.0
  240 CONTINUE
C     ASSUME STEADY FLOW AT J=0 FLOW FOR INITIAL PARCEL VALUES
      PX(1)=1.0
      DO 243 I=1,NEQ
       PPXT(I)=GT(I,1)
  243 CONTINUE
      PH(1)=HI
      NS=2
      RDT=DT
      DO 255 I=1,INX
       PX(NS)=I
       PDX=1.0
  245  PDT=PDX*DX(I)/U(I)
       IF (PDT .LT. RDT) GO TO 250
C      STOPPED IN REACH
       PX(NS)=PX(NS)+RDT*U(I)/DX(I)
       DO 248 K=1,NEQ
        PT(K,NS-1)=PPXT(K)
        PPXT(K)=GT(K,I)+(PX(NS)-I)*(GT(K,I+1)-GT(K,I))
        PT(K,NS-1)=(PT(K,NS-1)+PPXT(K))/2.0
        PTI(K,NS-1)=PT(K,NS-1)
  248  CONTINUE
       PV(NS-1)=PV(NS-1)+0.5*(2.0*FLOW(2,I)+(PX(NS)-I)*(FLOW(2,I+1)
     1            -FLOW(2,I)))*(PX(NS)-MAX(PX(NS-1),FLOAT(I)))*DX(I)
     2            *CONM**2
       PH(NS)=HI
       PDX=I+1-PX(NS)
       NS=NS+1
       RDT=DT
       PX(NS)=PX(NS-1)
       GO TO 245
C      PASSED GRID
  250  PV(NS-1)=PV(NS-1)+0.5*(2.0*FLOW(2,I+1)+(I+1-PX(NS))*(FLOW(2,I)
     1                        -FLOW(2,I+1)))*(I+1-PX(NS))*DX(I)*CONM**2
       RDT=RDT-PDT
  255 CONTINUE
      NS=NS-1
      DO 257 I=1,NEQ
       PT(I,NS-1)=PPXT(I)
       PPXT(I)=GT(I,INX)+(PX(NS)-INX)*(GT(I,INX+1)-GT(I,INX))
       PT(I,NS-1)=(PT(I,NS-1)+PPXT(I))/2.0
       PTI(I,NS-1)=PT(I,NS-1)
       PT(I,NS)=(PPXT(I)+GT(I,NXSEC))/2.0
       PTI(I,NS)=PT(I,NS)
  257 CONTINUE
C     STORE INITIAL VALUES IF REQUESTED OR NEEDED
      IF (MOD(ABS(IPLOT),2) .NE. 0) CALL STORE(0,NTS,NXSEC,NEQ,GT)
      IF (ABS(IPLOT) .GE. 2) THEN
```

```
      DO 259 I=1,NEQ
       WRITE(32'I) (PT(I,K),K=1,NS)
  259  CONTINUE
       WRITE(32'(NEQ+1)) (PX(K),K=1,NS)
      END IF
C
C*****WRITE INITIAL CONDITIONS AND OUTPUT HEADER************************
C
      WRITE (36,3620)
 3620 FORMAT(1H0,30X,'INITIAL  CONDITIONS',//)
      WRITE(36,3625) (I,I=1,NEQ)
 3625 FORMAT(3X,'GRID   RIVER   TRIB. FLOW   VELOCITY',
     1       3X,'AREA   TOP WIDTH   LAT FLOW   DISP FACT',3X,
     2       'INITIAL CONCENTRATIONS OF CONSTITUENTS',/,11X,' KM ',4X,
     3       'CU  M/SEC     M/HR     SQ M    METERS',6X,
     4       'SQ M/S      DQQ',3X,10(3X,I2))
      DO 260 I=1,NXSEC
       RKM=(XUP*CONX-X(I))/1000.00
       WRITE(36,3630) I,RKM,QIN(I,1),(GT(K,I),K=1,NEQ)
 3630  FORMAT(4X,I2,F9.2,F11.3,56X,10(1X,F4.1))
       IF(I.LT.NXSEC) WRITE(36,3635) U(I),A(I),W(I),QIN(I,2),DQQ(I)
 3635  FORMAT(24X,F14.1,2F9.2,F13.6,F10.2)
  260 CONTINUE
      WRITE(36,3640)
 3640 FORMAT(1H1,2X,'TIME',6X,'GRID',4X,'TRAVEL',3X,'CONSTI',1X,' GRID ',
     1       2X,'PARCEL CONC',5X,'CHANGE IN CONC DUE TO',/,
     2       ' DAY  HOUR',2X,'NUMBER',2X,'TIME(HR)',2X,'TUENT',3X,'CONC',
     3       4X,'AT ENTRY',5X,'DECAY',3X,'DISP',3X,'LATQ',3X,'TRIB',/)
      NLINES=3
C
C*****START TIME LOOP**************************************************
C
      DO 780 J=1,NTS
      TIME=J*DT+HI
      IF (MOD(ABS(IUNSTD),2) .EQ. 0) GO TO 320
C
C*****COMPUTE AVERAGE REACH VALUES FOR UNSTEADY FLOW********************
C
      DO 280 I=1,INX
       A(I)=FLOW(2,I)+FLOW(2,I+1)
       U(I)=FLOW(1,I)*FLOW(2,I)+FLOW(1,I+1)*FLOW(2,I+1)
       W(I)=FLOW(3,I)+FLOW(3,I+1)
       QIN(I,1)=FLOW(4,I)
       QIN(I,2)=FLOW(5,I)
  280 CONTINUE
      IF (IUNSTD .EQ. -1 .OR. IUNSTD .EQ. -3) CALL TDDSHY(FLOW,J)
      IF (IUNSTD .EQ. 1) READ(30,3000) ((FLOW(K,I),K=1,5),I=1,NXSEC)
      DO 300 I=1,INX
       A(I)=(A(I)+FLOW(2,I)+FLOW(2,I+1))*0.25*CONM**2
       U(I)=(U(I)+FLOW(1,I)*FLOW(2,I)+FLOW(1,I+1)*FLOW(2,I+1))*CONM**3
     1       /(4.0*A(I))
       W(I)=(W(I)+FLOW(3,I)+FLOW(3,I+1))*0.25*CONM
       QIN(I,1)=(QIN(I,1)+FLOW(4,I))*0.5*CONM**3
       QIN(I,2)=(QIN(I,2)+FLOW(5,I))*0.5*CONM**2
  300 CONTINUE
C
C*****COMPUTE DISPERSION BETWEEN PARCELS*******************************
C
  320 PX(NS+1)=NXSEC
C     CALCULATE DQ'S
      DO 340 K=1,NS
      IX=PX(K)
  340 DQ(K)=ABS(DQQ(IX)*U(IX)*A(IX))
C     SET DQ=0 AT TRIBUTARIES AND UPSTREAM BOUNDARIES OF LATERAL INFLOW
      K=0
      DO 400 IN=1,2
      DO 380 I=1,NIN(IN)
  360 K=K+1
      IF (PX(K) .LT. IGIN(I,IN)) GO TO 360
      K= K-1
      IF (K .EQ. 0) GO TO 380
      DQ(K)= 0.0
```

```
  380 CONTINUE
  400 CONTINUE
C     CALCULATE DF FOR EACH PARCEL
C
C     DETERMINE NUMBER OF TIME STEP SUBDIVISIONS NEEDED FOR EACH PARCEL
      DQ(1)=0.0
      DQ(NS+1)=0.0
      NDMAX=0
      ND(1)=1
      ND(NS+1)=1
      DO 420 K=2,NS
       ND(K)=1
       IEXP=0
       RATIO=DQ(K)*DT/MIN(PV(K-1),PV(K))
       IF (RATIO .GT. 0.0) IEXP=INT(LOG10(RATIO/0.4)/LOG10(2.0)+1)
       IF (IEXP .GT. 0) ND(K)=2**IEXP
       NDMAX=MAX(ND(K),NDMAX)
  420 CONTINUE
C     COMPUTE NEW MASS FLUXES IF NEEDED AND UPDATE ALL CONCENTRATIONS
      DMF(1)=0.0
      DO 463 L=1,NEQ
       DF(L,1)=0.0
       DO 460 I=1,NDMAX
        DO 440 K=1,NS
         IF (I .EQ. 1) DF(L,K+1)=0.0
         IF (MOD(I-1,NDMAX/ND(K+1)) .EQ. 0) DMF(K+1)=DQ(K+1)*DT*
     1       (PT(L,K)+DF(L,K)-PT(L,K+1)-DF(L,K+1))/NDMAX
         DF(L,K)=(DMF(K)-DMF(K+1))/PV(K)+DF(L,K)
         IF (ABS(DF(L,K)) .LT. 1.0E-6) DF(L,K)=0.0
  440   CONTINUE
  460  CONTINUE
  463 CONTINUE
C
C*****READ BOUNDARY CONDITIONS*****************************************
C
      DO 470 I=1,NEQ
       IF (IUNSTD .EQ. -1 .OR. IUNSTD .EQ. -2) THEN
         NBC=4
         CALL TDDSBC(PARA(I),NBC,J,CINC,TUP(I),0.0,0.0,0.0,0.0,0.0)
         DO 465 K=1,NXSEC
          CIN(I,K,1)=CINC(K,1)
          CIN(I,K,2)=CINC(K,2)
  465    CONTINUE
        ELSE
        READ(35,3515) TUP(I),((CIN(I,IGIN(K,IN),IN),K=1,NIN(IN)),
     1                IN=1,2)
       END IF
  470 CONTINUE
C
C*****MOVE PARCELS*****************************************************
C
      NS=NS+1
      DO 680 K=NS,1,-1
       IF (K .EQ. 1) THEN
C       SET VALUES FOR THE NEW PARCEL AT THE UPSTREAM BOUNDARY
        PX(1)=1.0
        PV(1)=FLOW(1,1)*FLOW(2,1)*DT*CONM**3
        PH(1)=J*DT+HI
        DO 475 I=1,NEQ
         PT(I,1)=TUP(I)
         PTI(I,1)=TUP(I)
         PTR(I,1,1)=0.0
         PTR(I,1,2)=0.0
         PDF(I,1)=0.0
         PDC(I,1)=0.0
         GT(I,1)=TUP(I)
  475   CONTINUE
        IX=1
        TLPP(1)=DT
        GO TO 620
       END IF
C
```

```
C       CHECK IF THE PARCEL'S UPSTREAM BOUNDARY PASSED THE NEXT GRID
C
        KO=K-1
        XOLD=PX(KO)
        MX=IFIX(XOLD)
        RDT=DT
        PDX=FLOAT(MX+1)-PX(KO)
  480   PDT=PDX*DX(MX)/U(MX)
        IF(PDT.GE.RDT) GO TO 520
C
C       PASSED GRID
C
        CALL DECAY(PDT,NEQ)
        RDT=RDT-PDT
        MX=MX+1
        XOLD=FLOAT(MX)
        PDX=1.0
        IF (ABS(PH(KO)-HI) .GT. 0.0000001
     1      .OR. KO .EQ. J) TT(MX)=TIME-PH(KO)-RDT
        DTT=DT-RDT-TLPP(MX)
        DO 500 IN=1,2
         IF (ABS(QIN(MX+1-IN,IN)) .LT. 0.0000001) GO TO 500
C        PARCEL LEAVING REACH WITH  INFLOW
         IF (IN .EQ. 1) THEN
C         TRIBUTARY INFLOW
          DELV=QIN(MX,1)*DTT*3600.0
         ELSE
C         LATERAL INFLOW
          DMAX=PV(KO)/A(MX-1)
          IF (DMAX .GT. DX(MX-1)) DMAX=DX(MX-1)
          DELV=QIN(MX-1,IN)*AREA(DT-RDT,0.0,U(MX-1),DMAX,
     1                           TLPP(MX-1+INT(DMAX/DX(MX-1))))
         END IF
         DO 490 I=1,NEQ
          PTR(I,KO,IN)=PTR(I,KO,IN)+
     1                 DEL(PT(I,KO),PV(KO),CIN(I,MX+1-IN,IN),DELV,I,NEQ)
  490    CONTINUE
  500   CONTINUE
        TLPP(MX)=DT-RDT
        IF (MX .NE.NXSEC) GO TO 480
        NS=NS-1
        GO TO 680
C
C       STOPPED IN REACH
C
  520   CALL DECAY(RDT,NEQ)
        PX(K)=XOLD+RDT*U(MX)/DX(MX)
        PV(K)=PV(KO)
        PH(K)=PH(KO)
        DO 525 I=1,NEQ
         PT(I,K)=PT(I,KO)+DF(I,KO)
         PTR(I,K,1)=PTR(I,KO,1)
         PTR(I,K,2)=PTR(I,KO,2)
         PTI(I,K)=PTI(I,KO)
         PDF(I,K)=PDF(I,KO)+DF(I,KO)
         PDC(I,K)=PDC(I,KO)
  525   CONTINUE
C
C       CONSIDER BOTH TRIBUTARY AND LATERAL INFLOW INTO STOPPED PARCEL
C
        IX=INT(PX(K))
        IF (PX(K+1) .LT. IX+1 .AND. K .LT. NS) THEN
         IF (ABS(QIN(IX,2)) .LT. 0.0000001) GO TO 680
C        PARCEL WHOLLY WITHIN REACH WITH LATERAL INFLOW
         FD=(PX(K+1)-PX(K))*DX(IX)
         DELV=QIN(IX,2)*AREA(DT,FD,U(IX),FD,TLPP(IX))
         DO 530 I=1,NEQ
          PTR(I,K,2)=PTR(I,K,2)+DEL(PT(I,K),PV(K),CIN(I,IX,2),DELV,I,NEQ)
  530    CONTINUE
         GO TO 680
        END IF
C       PARCEL PARTIALLY WITHIN REACH
```

```
      DO 535 I=1,NEQ
       GT(I,IX+1)=PT(I,K)
       GTR(I,IX+1,1)=PTR(I,K,1)
       GTR(I,IX+1,2)=PTR(I,K,2)
  535 CONTINUE
  540 IN=2
  560 IF (ABS(QIN(IX,IN)) .LT. 0.0000001) GO TO 600
      IF (IN .EQ. 1) THEN
C      TRIBUTARY INFLOW
       DELV=QIN(IX,1)*(DT-TLPP(IX))*3600.0
       SPV=FLOW(1,IX)*FLOW(2,IX)*(DT-TLPP(IX))*CONM**3
       DO 570 I=1,NEQ
        PTR(I,K,1)=PTR(I,K,1)+DEL(PT(I,K),PV(K),CIN(I,IX,1),DELV,I,NEQ)
        DELG(I)=(GT(I,IX+1)*SPV+CIN(I,IX,IN)*DELV)/(SPV+DELV)
     1            -GT(I,IX+1)
        IF (CIN(I,IX,IN) .LT. -1.0) DELG(I)=0.0
  570  CONTINUE
      ELSE
C      LATERAL INFLOW
       I=0
       IF (TLPP(IX+1) .LE. TLPP(IX) .OR. PX(K) .LT. IX) THEN
         DMAX=DX(IX)
         I=1
        ELSE
         DMAX=PV(K)/A(IX)
       END IF
       FD=(IX+1-MAX(PX(K),IX))*DX(IX)
       DELV=QIN(IX,2)*AREA(DT,FD,U(IX),DMAX,TLPP(IX+I))
C      CALCULATE AFFECT ON PARCEL DUE TO LATERAL INFLOW
       DELV=3600.0*QIN(IX,2)*(0.5*FD*FD/U(IX)+
     1          MAX(0.0,IX-PX(K))*DX(MAX(1,IX-1))*DMAX/U(MAX(1,IX-1)))
       DO 590 I=1,NEQ
        DELP=DEL(PT(I,K),PV(K),CIN(I,IX,2),DELV,I,NEQ)
        PTR(I,K,2)=PTR(I,K,2)+DELP
        GT(I,IX+1)=GT(I,IX+1)+DELP
        GTR(I,IX+1,2)=GTR(I,IX+1,2)+DELP
C       CALC ADDITIONAL AFFECT ON D.S. GRID FROM COMPLETE MIXING THERE
        DELG(I)=(GT(I,IX+1)*PV(K)+CIN(I,IX,2)*DELV)/(PV(K)+DELV)
     1            -GT(I,IX+1)
        IF (CIN(I,IX,2) .LT. -1.0) DELG(I)=0.0
  590  CONTINUE
      END IF
C     ADJUST GRID CONCENTRATION FOR INFLOW
      DO 595 I=1,NEQ
       IF (DELV .LT. 0.0) DELG(I)=0.0
       GT(I,IX+1)=GT(I,IX+1)+DELG(I)
       GTR(I,IX+1,IN)=GTR(I,IX+1,IN)+DELG(I)
  595 CONTINUE
  600 IF (IN .EQ. 1) GO TO 540
C     CHECK PARCEL'S POSITION IN THE NEXT DOWNSTREAM REACH
      IX=IX+1
  620 IN=1
      IF (IX .GE. NXSEC) GO TO 680
      IF (PX(K+1) .LT. IX+1) GO TO 640
C     PARCEL ENTIRELY ENCOMPASSES THE NEXT DOWNSTREAM REACH
      DO 630 I=1,NEQ
       GT(I,IX+1)=GT(I,IX)
       GTR(I,IX+1,1)=GTR(I,IX,1)
       GTR(I,IX+1,2)=GTR(I,IX,2)
  630 CONTINUE
      GO TO 560
C     DOWNSTREAM PARCEL BOUNDARY WITHIN NEXT REACH
  640 DO 660 IN=1,2
       IF (ABS(QIN(IX,IN)) .LT. 0.0000001) GO TO 660
       IF (IN .EQ. 1) THEN
C       TRIBUTARY INFLOW
        DELV=QIN(IX,1)*(DT-TLPP(IX))*3600.0
       ELSE
C       LATERAL INFLOW
        FD=(PX(K+1)-IX)*DX(IX)
        DELV=QIN(IX,2)*AREA(DT,FD,U(IX),FD,DT)
       END IF
```

```
        DO 650 I=1,NEQ
         PTR(I,K,IN)=PTR(I,K,IN)+DEL(PT(I,K),PV(K),CIN(I,IX,IN),DELV,
     1                                I,NEQ)
  650   CONTINUE
  660  CONTINUE
C
C     MOVE NEXT PARCEL
C
  680 CONTINUE
C     PARCEL MOVEMENT COMPLETE, ZERO TLPP
      DO 700 I=1,NXSEC
       TLPP(I)=0.0
  700 CONTINUE
C
C*****COMPUTE RMS ERROR, STORE DATA, WRITE RESULTS*********************
C
      IF (IRMS .EQ. 0) GO TO 713
      IF (MOD(J,ABS(IRMS)) .NE. 0) GO TO 713
       DO 710 I=1,NEQ
        IF (IRMS .LT. 0) THEN
C        CALCULATE RMS ERROR WITH TDDS DATA
         DO 705 K=1,NXSEC
          GTRMS(K)=GT(I,K)
  705    CONTINUE
         CALL TDDSRMS(GTRMS,PARA(I),J,SUM(I),NSUM(I))
        ELSE
C        CALCULATE RMS ERROR WITH TOBS.INPUT
         READ(39,3515) (TACT(I,L),L=1,NGO)
         DO 708 K=1,NGO
          NG=NGOUT(K)
          IF (TACT(I,K) .LT. 0.0) GO TO 708
          SUM(I)=(GT(I,NG)-TACT(I,K))*(GT(I,NG)-TACT(I,K))+SUM(I)
          NSUM(I)=NSUM(I)+1
  708    CONTINUE
        END IF
  710  CONTINUE
C     STORE DATA
  713 IF (MOD(ABS(IPLOT),2) .NE. 0) CALL STORE(J,NTS,NXSEC,NEQ,GT)
      IF (ABS(IPLOT) .GE. 2) THEN
       DO 715 K=1,NEQ
        NREC=J*(NEQ+1)+K
        WRITE(32'NREC) (PT(K,I),I=1,NS)
  715  CONTINUE
       WRITE(32'(NREC+1)) (PX(I),I=1,NS)
      END IF
      IF (MOD(J,IOUT) .NE. 0) GO TO 780
C     WRITE RESULTS FOR THIS TIME STEP
      IDAY=(TIME/24.0)+1
      HR=TIME-FLOAT(IDAY-1)*24.0
      K= 0
      DO 760 I=1,NGO
       NG=NGOUT(I)
       IF (NLINES .LT. 50) GO TO 720
       WRITE(36,3640)
       NLINES=2
  720  K= K+1
       IF (K .GT. NS) GO TO 740
       IF(PX(K).LE.FLOAT(NG)) GO TO 720
  740  K=K-1
       IF (I .EQ. 1) THEN
        WRITE(36,3645)IDAY,HR,NG,TT(NG),(LABEL(L),GT(L,NG),PTI(L,K),
     1                PDC(L,K),PDF(L,K),GTR(L,NG,2),GTR(L,NG,1),
     2                L=1,NEQ)
 3645   FORMAT(1X,I2,F7.2,4X,I2,F11.2,2X,10(A7,F7.2,F10.2,F11.2,
     1          3(1X,F6.2),/,29X))
       ELSE
        WRITE(36,3650) NG,TT(NG),(LABEL(L),GT(L,NG),PTI(L,K),PDC(L,K),
     1                 PDF(L,K),GTR(L,NG,2),GTR(L,NG,1),L=1,NEQ)
 3650   FORMAT(14X,I2,F11.2,2X,10(A7,F7.2,F10.2,F11.2,
     1          3(1X,F6.2),/,29X))
       END IF
       NLINES=NLINES+1+NEQ
```

```
      IF(NCO.GT.0.AND.NG.LT.NXSEC)CALL DECAYO(NCO,J,K,NG,NLINES,NEQ,DT)
      K=K-1
  760 CONTINUE
C
C*****TIME STEP CALCULATIONS COMPLETE***********************************
C
  780 CONTINUE
C
C*****SIMULATION COMPLETED**********************************************
C
C     CALL OPTIONAL OUTPUT SUBROUTINES
      IF (IUNSTD .LT. 0) CALL TDDSO(NEQ)
      IF (IPLOT .LT. 0) CALL PLOT
      IF (IRMS .EQ. 0) GO TO 800
C     COMPUTE AND WRITE RMS ERROR
      WRITE(36,3653)
 3653 FORMAT(1H1,'CONSTITUENT   RMS ERROR')
      DO 790 I=1,NEQ
       IF (NSUM(I) .EQ. 0) GO TO 790
       RMS=SQRT(SUM(I)/NSUM(I))
       WRITE(36,3660) LABEL(I),RMS
 3660  FORMAT(2X,A7,4X,F9.5)
  790 CONTINUE
C     CLOSE FILES
      IF (IRMS .EQ. 1) CLOSE(UNIT=39)
  800 CLOSE(UNIT=35)
      CLOSE(UNIT=36)
      IF (IUNSTD .EQ. 1) CLOSE(UNIT=30)
      IF (MOD(ABS(IPLOT),2) .NE. 0) CLOSE(UNIT=31)
      IF (ABS(IPLOT) .GE. 2) CLOSE(UNIT=32)
      END
C
C*****CALCULATE THE DECAY IN PARCEL CONCENTRATION***********************
C
      SUBROUTINE DECAY(DT,NEQ)
C
C     DEFINE VARIABLES:
C
C      NAME                          PURPOSE
C     ------  ------------------------------------------------------------
C      A(G)   AVERAGE CROSS-SECTIONAL AREA OF THE REACH DOWNSTREAM FROM
C             GRID G [SQUARE METERS]
C
C    CR(E,C)  REFERENCE (EQUILIBRIUM) CONCENTRATION IN EQUATION E FOR
C             CONSTITUENT C
C
C      DEF    CONCENTRATION DEFICIT (PT-CR)
C
C   DKPHR1(C) DECAY RATE OF CONSTITUENT C AT THE START OF THE DECAY
C             TIME STEP [1/HOUR]
C
C   DKPHR2(C) DECAY RATE OF CONSTITUENT C AT THE END OF THE DECAY
C             TIME STEP [1/HOUR]
C
C       DT    DURATION OF DECAY [HOURS]
C
C      DTL    THE TIME REQUIRED FOR TEN PERCENT OF THE DEFICIT TO DECAY
C             [HOURS]
C
C      DTM    THE INCREMENT OF THE DECAY TIME STEP (DTM<=DT) SUCH
C             THAT IN THE TIME DTM THE CONCENTRATION NEVER CHANGES BY MORE
C             THAN TEN PERCENT OF THE REMAINING DEFICIT (PT-CR) OR BY
C             MORE THAN 0.3 UNITS, WHICHEVER IS GREATER [HRS]
C
C       I     COUNTER
C
C       I1    COUNTER
C
C       I2    COUNTER
C
C    IPG(N)   GRID CONTROL FOR SPECIFIED COEFFICIENT OUTPUT N, N=1,NCO,
C                  IF IPG(N)=0, WRITE COEFFICIENT OUTPUT AT ALL GRIDS
```

```
C                                IN NGOUT
C                     IF IPG(N)>0, WRITE COEFFICIENT OUTPUT AT GRID IPG(N)
C
C    IPT(N)    TIME CONTROL FOR SPECIFIED COEFFICIENT OUTPUT N, N=1,NCO,
C                     IF IPT(N)=0, WRITE COEFFICENT OUTPUT AT ALL
C                                  REGULAR OUTPUT TIME STEPS
C                     IF IPT(N)>0, WRITE COEFFICIENT OUTPUT AT TIME STEP
C                                  IPT(N)
C
C       J      SIMULATION TIME STEP
C
C       K      COUNTER
C
C      KKO     PARCEL NUMBER (ARGUEMENT)
C
C       KO     PARCEL NUMBER (COMMON)
C
C       MX     UPSTREAM GRID NUMBER
C
C      NCO     NUMBER OF COEFFICIENT OUTPUTS SPECIFIED
C
C      NEQ     NUMBER OF CONSTITUENTS IN SIMULATION
C
C       NG     NUMBER OF GRID AT WHICH OUTPUT IS DESIRED
C
C     NLINES   NUMBER OF LINES OF OUTPUT ON THE CURRENT PAGE
C
C    PDC(C,P)  CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C              DUE TO DECAY
C
C     PT(C,P)  CONCENTRATION OF CONSTITUENT C IN PARCEL P
C
C     PT1(C)   PARCEL CONCENTRATION OF CONSTITUENT C AT THE BEGINNING
C              OF THE DECAY TIME STEP
C
C      RDT     TIME REMAINING FOR DECAY TO OCCUR [HOURS]
C
C      S(C)    SOURCE FLUX OF CONSTITUENT C [UNITS PER HOUR]
C
C      U(G)    AVERAGE VELOCITY IN THE REACH DOWNSTREAM FROM GRID G
C              [METERS PER HOUR]
C
C      W(G)    AVERAGE TOP WIDTH IN THE REACH DOWNSTREAM FROM GRID G [M]
C
C    XK(C,CC)  EXCHANGE COEFFICIENT FOR CONSTITUENT C DUE TO
C              CONSTITUENT CC
C
      COMMON /CDECAY/ KO,MX,PT,PDC
      INTEGER IPG(20),IPT(20)
      REAL A(50),CR(10,10),DKPHR1(10),DKPHR2(10),PDC(10,500),PT(10,500),
     1     PT1(10),S(10),U(50),W(50),XK(10,10)
      SAVE IPG,IPT
      DO 80 I=1,NEQ
       S(I)=0.0
       DO 60 K=1,NEQ
        CR(K,I)=0.0
        XK(K,I)=0.0
   60  CONTINUE
   80 CONTINUE
      RDT=DT
  100 DTM= RDT
      CALL FINK(CR,S,XK)
C
C     FIND MAXIMUM DECAY TIME STEP
C
      DO 140 I=1,NEQ
       DKPHR1(I)=S(I)
       DO 120 K=1,NEQ
        DKPHR1(I)=DKPHR1(I)+XK(I,K)*(PT(K,KO)-CR(I,K))
  120  CONTINUE
       DTL= RDT
       DEF=PT(I,KO)-CR(I,I)
```

```
      IF (ABS(DKPHR1(I)) .GT. 0.0000001 .AND. ABS(DEF) .GT. 0.3)
     1    DTL=ABS(0.1*DEF/DKPHR1(I))
      IF(DTL.LT.DTM) DTM=DTL
  140 CONTINUE
      IF(DTM.GT. (0.999*RDT)) DTM= RDT
C
C     UPDATE CONCENTRATION
C
      DO 160 I=1,NEQ
       PT1(I)=PT(I,KO)
       PT(I,KO)= PT(I,KO)+DKPHR1(I)*DTM
  160 CONTINUE
      CALL FINK(CR,S,XK)
      DO 200 I=1,NEQ
       DKPHR2(I)=S(I)
       DO 180 K=1,NEQ
        DKPHR2(I)=DKPHR2(I)+XK(I,K)*(PT(K,KO)-CR(I,K))
  180  CONTINUE
       PT(I,KO)=PT1(I)+0.5*(DKPHR1(I)+DKPHR2(I))*DTM
       PDC(I,KO)=PDC(I,KO)+0.5*(DKPHR1(I)+DKPHR2(I))*DTM
  200 CONTINUE
      IF (ABS(DTM-RDT) .LT. 0.0000001) GO TO 220
      RDT= RDT-DTM
      GO TO 100
  220 RETURN
C
C     READ VALUES FOR CONTROL OF COEFFICIENT OUTPUT
C
      ENTRY DECAYIN(NCO)
      READ(35,3500) (IPT(I),IPG(I),I=1,NCO)
 3500 FORMAT(10X,10I7)
      RETURN
C
C     WRITE COEFFICIENT OUTPUT
C
      ENTRY DECAYO(NCO,J,KKO,NG,NLINES,NEQ,DT)
      KO=KKO
      MX=NG
      DO 280 I=1,NCO
       IF (IPT(I) .NE. J .AND. IPT(I) .NE. 0) GO TO 280
       IF (IPG(I) .NE. MX .AND. IPG(I) .GT. 0) GO TO 280
       CALL FINK(CR,S,XK)
       IF (60-NLINES .LT. 3*(NEQ+5)) THEN
        WRITE(36,3600)
 3600    FORMAT(1H1)
        NLINES=1
       END IF
       WRITE(36,3605)
 3605  FORMAT(1H0,39X,'CR(I1,I2)')
       WRITE(36,3610) (I1,I1=1,NEQ)
 3610  FORMAT(1H0,'I2  ',39(1H-),'I1',39(1H-),/,' --',5X,10(I2,6X))
       DO 240 I2=1,NEQ
        WRITE(36,3615) I2,(CR(I1,I2),I1=1,NEQ)
 3615    FORMAT(I3,2X,10F8.3)
  240  CONTINUE
       WRITE(36,3620)
 3620  FORMAT(1H0,39X,'XK(I1,I2)')
       WRITE(36,3610) (I1,I1=1,NEQ)
       DO 260 I2=1,NEQ
        WRITE(36,3615) I2,(XK(I1,I2),I1=1,NEQ)
  260  CONTINUE
       WRITE(36,3625)
 3625  FORMAT(1H0,41X,'S(I1)')
       WRITE(36,3630) (I1,I1=1,NEQ)
 3630  FORMAT(1H0,4X,39(1H-),'I1',39(1H-),/,8X,10(I2,6X))
       WRITE(36,3635) (S(I1),I1=1,NEQ)
 3635  FORMAT(5X,10F8.3)
       NLINES=51
  280 CONTINUE
      RETURN
      END
C
```

```
C
C*****CALCULATE AREA UNDER LENGTH VS TIME CURVE FOR LATERAL INFLOW******
C
      FUNCTION AREA(T4,FD,U,DMAX,T2)
C
C     DEFINE VARIABLES:
C
C      NAME                                PURPOSE
C     ------  ------------------------------------------------------------
C      AREA   AREA UNDER THE LENGTH VS TIME CURVE OF A PARCEL WITHIN A
C             REACH WITH LATERAL INFLOW FOR THE CURRENT TIME STEP [M*S]
C
C       FD    FINAL PARCEL LENGTH EXPOSED TO LATERAL INFLOW [METERS]
C
C      DMAX   MAXIMUM PARCEL LENGTH THAT CAN BE EXPOSED TO LATERAL
C             INFLOW WITHIN THE REACH [METERS]
C
C       T1    TIME WHEN DOWNSTREAM PARCEL BOUNDARY ENTERS REACH WITH
C             LATERAL INFLOW [HOURS]
C
C       T2    TIME WHEN MAXIMUM EXPOSURE TO LATERAL INFLOW BEGAN [HOURS]
C
C       T3    TIME WHEN DOWNSTREAM PARCEL BOUNDARY LEAVES REACH WITH
C             LATERAL INFLOW [HOURS]
C
C       T4    EITHER TIME AT WHICH PARCEL LEAVES REACH WITH LATERAL
C             INFLOW OR SIMULATION TIME STEP [HOURS]
C
C       U     AVERAGE REACH VELOCITY [METERS/HOUR]
C
C     **NOTE: TIMES T1 THRU T4 ARE EVALUATED WITHIN THE CURRENT TIME
C             STEP (0 =< T =< DT)
C
C
      AREA=0.0
      T3=T4-((DMAX-FD)/U)
      IF (T3 .LT. 0.0) THEN
C      PARCEL STARTED TIME STEP STRADLING DOWNSTREAM BOUNDARY OF REACH
       AREA=T4*((FD+U*T4)-0.5*U*T4)
       GO TO 100
      END IF
C     MAXIMUM POSSIBLE LENGTH EXPOSED TO INFLOW DURING TIME STEP
      AREA=(T4-T2)*DMAX-0.5*(T4-T3)*(DMAX-FD)
      IF (ABS(T2) .LT. 0.0000001) GO TO 100
C     MAXIMUM EXPOSURE BEGAN THIS TIME STEP
      T1=T2-(DMAX/U)
      IF (T1 .LT. 0.0) THEN
C       DOWNSTREAM PARCEL BOUNDARY STARTED TIME STEP IN REACH
        AREA=T2*(DMAX-0.5*U*T2)+AREA
       ELSE
C       DOWNSTREAM PARCEL BOUNDARY ENTERED REACH THIS TIME STEP
        AREA=0.5*(T2-T1)*DMAX+AREA
      END IF
C     CONVERT AREA FROM [METERS*HOURS] TO [METERS*SECONDS]
  100 AREA=AREA*3600.0
      RETURN
      END
C
C*****MIX INFLOW INTO PARCELS*****************************************
C
      FUNCTION DEL(PT,PV,CIN,DELV,NCON,NEQ)
C
C     DEFINE VARIABLES:
C
C      NAME                            PURPOSE
C     ------  ------------------------------------------------------------
C      CIN    CONCENTRATION OF INFLOW
C
C      DEL    CHANGE IN PARCEL CONCENTRATION DUE TO INFLOW
C
C      DELV   CHANGE IN PARCEL VOLUME DUE TO INFLOW [CUBIC METERS]
C
```

```
C       NCON    NUMBER OF THE CONSTITUENT BEING MIXED
C
C       NEQ     NUMBER OF CONSTITUENTS BEING SIMULATED
C
C        PT     PARCEL CONCENTRATION
C
C        PV     PARCEL VOLUME [CUBIC METERS]
C
C
      DEL=(PT*PV+CIN*DELV)/(PV+DELV)-PT
      IF (CIN .LT. -1.0) DEL=0.0
      IF (DELV .LT. 0.0) DEL=0.0
      PT=PT+DEL
      IF (NCON .EQ. NEQ) PV=PV+DELV
      RETURN
      END
C
C*****STORE GRID DATA IN A DIRECT ACCESS FILE***************************
C
      SUBROUTINE STORE(J,NTS,NXSEC,NEQ,GT)
C
C     DEFINE VARIABLES:
C
C      NAME                          PURPOSE
C     ------  ------------------------------------------------------
C     DATA(L) CONTAINS ALL PREVIOUS CONCENTRATIONS AT A GRID POINT
C
C     GT(C,G) CONCENTRATION OF CONSTITUENT C AT GRID G
C
C  GTS(C,G,T) CONTAINS CONCENTRATION OF CONSTITUENT C AT GRID G FOR TIME
C             STEP T=J50
C
C        I    COUNTER
C
C        J    TIME STEP
C
C       J50   NUMBER OF TIME STEPS SINCE J+1 WAS LAST A MULTIPLE OF 50
C
C      JLAST  THE LAST TIME STEP WRITTEN TO UNIT 31
C
C        K    COUNTER
C
C        L    COUNTER
C
C       NEQ   NUMBER OF CONSTITUENTS IN SIMULATION
C
C       NREC  RECORD NUMBER IN UNIT 31
C
C       NTS   NUMBER OF TIME STEPS IN SIMULATION
C
C      NXSEC  NUMBER OF GRID POINTS IN SIMULATION
C
C
      REAL DATA(200),GT(10,50),GTS(10,50,50)
      SAVE GTS
      DO 120 I=1,NXSEC
       DO 100 L=1,NEQ
        J50=1+MOD(J,50)
        GTS(L,I,J50)=GT(L,I)
        IF (J50 .LT. 50 .AND. J .LT. NTS) GO TO 100
        NREC=NEQ*(I-1)+L
        JLAST=J-MOD(J,50)
         IF (J .LT. 50) THEN
           WRITE(31'NREC) (GTS(L,I,K),K=1,J+1)
          ELSE
           READ(31'NREC) (DATA(K),K=1,JLAST)
           WRITE(31'NREC) (DATA(K),K=1,JLAST),(GTS(L,I,K),K=1,J50)
         END IF
  100  CONTINUE
  120 CONTINUE
      RETURN
      END
```

APPENDIX B

TDDS.F77

LTM/TDDS Interface Subroutines

```
C
C     DSTIME   COMPUTES BEGINNING AND END TIMES FOR DADI AND DADO CALLS
C
C     DTCODE   CODES A TIME
C
C     SCRIPT   DETERMINES HOW TO CONVERT TDDS DATA INTO MODEL DATA
C
C
C*****DEFINE DATA TYPES*************************************************
C
C     NAME                          DATA TYPE
C     ----   ------------------------------------------------------------
C     > A<   INTEGER*4 CROSS SECTIONAL AREA [SQUARE FEET OR METERS]
C
C      AP    ATMOSPHERIC PRESSURE [KILOPASCALS]
C
C      AT    ATMOSPHERIC TEMPERATURE [CELCIUS]
C
C     > B<   INTEGER*4 TOP WIDTH [FEET OR METERS]
C
C      C#    CONCENTRATION OF CONSTITUENT #, WHERE #=1 TO NEQ (10=0)
C
C      L#    CONCENTRATION OF CONSTITUENT # OF LATERAL INFLOWS, WHERE
C            #=1 TO NEQ (10=0)
C
C      M#    MEASURED RIVER CONCENTRATION OF CONSTITUENT #, WHERE
C            #=1 TO NEQ (10=0)
C
C     > Q<   INTEGER*4 FLOW RATE [CUBIC FEET OR METERS PER SECOND]
C
C     >QL<   LATERAL FLOW RATE [SQUARE FEET OR METERS PER]
C
C     >QT<   TRIBUTARY FLOW RATE [CUBIC FEET OR METERS PER SECOND]
C
C      RA    INCOMING ATMOSPHERIC RADIATION [CAL PER SQ CM PER HOUR]
C
C      RS    INCOMING SOLAR RADIATION [CAL PER SQ CM PER HOUR]
C
C      T#    CONCENTRATION OF CONSTITUENT # OR TRIBUTARY INFLOWS, WHERE
C            #=1 TO NEQ (10=0)
C
C      WS    WIND SPEED [METERS PER SECOND]
C
C   **NOTE: ALL DATA IS STORED AS INTEGER*2 DATA IN HUNDRETHS OF UNITS
C           IN UNIT 98 EXCEPT FOR Q WHICH IS INTEGER*4 IN HUNDRETHS
C           OF CUBIC FEET PER SECOND AND A AND B
C
C   **NOTE: >DATA TYPE< DENOTES HYDRAULIC DATA, ALL OTHER DATA TYPES
C           ARE BOUNDARY CONDITION DATA
C
C
C*****DEFINE ARRAYS*****************************************************
C
C      NAME                          PURPOSE
C     ------   ---------------------------------------------------------
C      BC(K)    CONTAINS NAMES OF BOUNDARY CONDITION DATA SETS
C
C    CIN(K,L)   IF L=1, CONCENTRATION OF TRIBUTRARY INFLOW AT GRID K
C               IF L=2, CONCENTRATION OF LATERAL INFLOW IN REACH
C                       DOWNSTREAM FROM GRID K
C
C    DATA(K)    CONCENTRATION AT A GRID AT TIME STEP K
C
C DATAI2(K,L,M) INPUT DATA TYPE L AT GRID K AND TIME STEP INT(J/10)+M,
C               WHERE M=1 TO 10.  THUS AT TIME STEPS 0, 10, 20... INPUT
C               DATA FOR THE NEXT TEN SIMULATION TIME STEPS IS STORED
C
C DATAI4(K,L,M) INPUT DATA TYPE L AT GRID K AND TIME STEP INT(J/10)+M,
C               WHERE M=1 TO 10.  THUS AT TIME STEPS 0, 10, 20... INPUT
C               DATA FOR THE NEXT TEN SIMULATION TIME STEPS IS STORED
C
C   DELTIM(K)   DIFFERENCE BETWEEN DESIRED SIMULATION TIME AND DATA
```

```
*SUBROUTINE THAT INTERFACES THE LAGRANGIAN TRANSPORT MODEL TO THE TDDS**
*
*********************************************************************
*                                                                   *
*            TIME DEPENDENT DATA SYSTEM INTERFACE SUBROUTINE        *
*                                                                   *
*                       BY DAVID SCHOELLHAMER                       *
*                                                                   *
*                  UNITED STATES GEOLOGICAL SURVEY                  *
*                  GULF COAST HYDROSCIENCE CENTER                   *
*                          BUILDING 2101                            *
*                      NSTL STATION, MS  39529                      *
*                                                                   *
*                             MARCH 1985                            *
*                                                                   *
*********************************************************************
C
C
C*****DEFINE FORTRAN UNIT NUMBERS*************************************
C
C     UNIT NUMBER   FILE NAME                PURPOSE
C     -----------   ---------   ----------------------------------------
C         31        GRID.DATA   DIRECT ACCESS FILE THAT STORES GRID
C                               CONCENTRATIONS AT EVERY TIME STEP IF
C                               REQUESTED BY THE USER OR NEEDED BY TDDS
C
C         35        LTM.INPUT   BASIC MODEL INPUT
C
C         36        LTM.OUTPUT  MODEL OUTPUT
C
C         37            -       RESERVED FOR USE IF PUNCH OUTPUT IS
C                               DESIRED FROM THE TDDS
C
C         38            -       DATA STATION REFERENCE FILE FOR TDDS
C
C         98            -       FILE CONTAINING TIME DEPENDENT DATA
C                               FOR TDDS
C
C   **NOTE: FILE NUMBERING BEGINS AT 30 TO ALLOW USE OF DISSPLA GRAPHICS
C           IN SUBROUTINE PLOT THAT MAY BE CALLED FROM THE MAIN PROGRAM
C
C   **NOTE: UNITS 37, 38, AND 98 ARE TO BE OPENED EXTERNALLY
C
C
C*****DEFINE ENTRY POINTS********************************************
C
C      NAME                        PURPOSE
C      ------   ----------------------------------------------------
C      TDDS     INITIALIZE VALUES, CALL DADIO, READ INPUT
C
C     TDDSBC    RETURN BOUNDARY CONDITIONS FOR THE CURRENT TIME STEP
C               TO THE MODEL
C
C     TDDSHY    RETURN HYDRAULIC PROPERTIES FOR THE CURRENT TIME STEP
C               TO THE MODEL
C
C     TDDSO     WRITE OUTPUT TO UNIT 98
C
C    TDDSRMS    CALCULATE THE SUM OF THE SQUARED ERRORS
C
C
C*****DEFINE SUBROUTINES*********************************************
C
C      NAME                          PURPOSE
C      ------   ----------------------------------------------------
C      DADI     RETRIEVES DATA FROM UNIT 98
C
C      DADIO    PREPARES UNIT 98 FOR DATA TRANSFER
C
C      DADO     SENDS DATA TO UNIT 98
C
C     DECODE    DECODES CODED TIME
```

```
C                TIME FOR VALUE K, K=1 TO 10
C
C   FLOW(K,L)    VALUE OF HYDRAULIC PROPERTY K AT OR BELOW GRID L:
C                     FOR K=1, VELOCITY [FT OR M/HR] FROM CONTINUITY
C                     FOR K=2, AREA [SQUARE FEET OR METERS]
C                     FOR K=3, TOP WIDTH [FEET OR METERS]
C                     FOR K=4, TRIBUTARY INFLOW JUST BELOW GRID L
C                              [CUBIC FEET OR METERS PER SECOND]
C                     FOR K=5, LATERAL INFLOW IN THE REACH DOWNSTREAM
C                              FROM GRID L [SQ FT OR M/SEC]
C
C     GC(K)      TDDS VALUE FOR CONCENTRATION AT A GRID AT TIME STEP K
C
C     GT(K)      CONCENTRATION AT GRID K FOR THE CURRENT TIME STEP
C
C     HY(K)      CONTAINS TDDS NAMES OF HYDRAULIC DATA SETS
C
C     I2(K)      CONTAINS DATA RECEIVED FROM THE TDDS
C
C     I4(K)      CONTAINS INTEGER*4 DATA RECEIVED FROM THE TDDS
C
C >INGRID(K,L)< GRID NUMBERS WITH INPUT DATA TYPE L WHERE K=1 TO 50,
C                L=1 TO NINPUT
C
C  >INTYPE(K)<  INPUT DATA TYPES, K=1 TO NINPUT
C
C  >NINGRD(K)<  NUMBER OF GRIDS WITH AN INPUT DATA TYPE, K=1 TO NINPUT
C
C  >NOUTGR(K)<  NUMBER OF GRIDS WITH AN OUTPUT DATA TYPE, K=1 TO NOUT
C
C >OUTGRD(K,L)< GRID NUMBERS WITH AN OUTPUT DATA TYPE, K=1,NOUTGR(L)
C                AND L=1,NOUT
C
C  >OUTTYP(K)<  OUTPUT DATA TYPES, K=1 TO NOUT
C
C   >RDPDY(K)<  READINGS PER DAY OF TDDS DATA TYPE K, K=1 TO NINPUT
C
C      S(K)      SUBSCRIPTS FOR CONVERTING TDDS DATA TO SIMULATION DATA,
C                K=1 TO 10
C
C   >STAID(K)<  TDDS STATION IDENTIFICATION NUMBER FOR GRID K
C
C    TLCM(K)     CONTAINS 'T', 'L', 'C', AND 'M'
C
C   **NOTE: >ARRAY< DENOTES ARRAYS DEFINED VIA UNIT 35.  IN ADDITION,
C           ALL HYDRAULIC PROPERTIES AND BOUNDARY CONDITIONS TO BE READ
C           BY THIS SUBROUTINE MUST BE PRE-DEFINED WITH THE TDDS
C
C
C*****DEFINE VARIABLES*****************************************************
C
C      NAME                          PURPOSE
C     ------   ------------------------------------------------------------
C      >BD<    BEGINNING DAY OF TDDS DATA TRANSFER
C
C      >BH<    BEGINNING HOUR OF TDDS DATA BEING TRANSFERED
C
C      >BMN<   BEGINNING MINUTE OF TDDS DATA BEING TRANSFERRED
C
C      >BMO<   BEGINNING MONTH OF TDDS DATA BEING TRANSFERRED
C
C      BTIME   CODED BEGINNING TIME OF SIMULATION
C
C      >BY<    BEGINNING YEAR OF TDDS DATA TO BE TRANSFERRED
C
C      CHAR    DATA TYPE CONTAINING MEASURED VALUES FOR RMS CALCULATION
C
C      DSREF   UNIT NUMBER OF DATA STATION REFERENCE FILE (38)
C
C       DT     SIMULATION TIME STEP [HOURS], DUMMY ARGUEMENT
C
C     DTDATA   TIME STEP OF TDDS DATA [HOURS]
```

```
C
C       DTT     SIMULATION TIME STEP [HOURS], STATIC
C
C       EA      VAPOR PRESSURE [KILOPASCALS]
C
C      >ED<     ENDING DAY OF TDDS DATA TO BE TRANSFERRED
C
C      >EH<     ENDING HOUR OF TDDS DATA TO BE TRANSFERRED
C
C      >EMN<    ENDING MINUTE OF TDDS DATA TO BE TRANSFERRED
C
C      >EMO<    ENDING MONTH OF TDDS DATA TO BE TRANSFERRED
C
C      ETIME    CODED ENDING TIME OF SIMULATION
C
C      >EY<     ENDING YEAR OF TDDS DATA TO BE TRANSFERRED
C
C       HL      INCOMING ATMOSPHERIC RADIATION [CAL PER SQ CM PER DAY]
C
C       HS      INCOMING SOLAR RADIATION [CAL PER SQ CM PER DAY]
C
C        I      COUNTER
C
C       IO      INTEGER*2 ZERO (0)
C
C       IG      GRID NUMBER WITH A TDDS INPUT
C
C      IPLOT    DEFINES PLOT OPTION:
C                    IF IPLOT=0, DO NOT STORE DATA
C                    IF ABS(IPLOT)=1, STORE GRID DATA
C                    IF ABS(IPLOT)=2, STORE PARCEL DATA
C                    IF ABS(IPLOT)=3, STORE BOTH GRID AND PARCEL DATA
C                    IF IPLOT<0, MAIN CALLS USER DEFINED SUBROUTINE PLOT
C                    IF IPLOT=>0, DO NOT CALL PLOT
C
C        J      SIMULATION TIME STEP
C
C       JI2     INTEGER*2 JUNK
C
C       JI4     INTEGER*4 JUNK
C
C       JJ      MOD(J,10)+1, NUMBER OF DATA POINT TO BE ASSIGNED TO MAIN
C
C        K      COUNTER
C
C        L      COUNTER
C
C     >LIST<    DADIO LIST OPTION:
C                    IF LIST=-1, PRINT CALENDER YEAR AND DIRECTORY
C                                SUMMARIES
C                    IF LIST=0, DO NOT PRINT SUMMARIES
C                    IF LIST=1, PRINT ONLY THE INDEX DIRECTORY SUMMARY
C
C        M      COUNTER, BUT IN TDDSRMS IT IS THE NUMBER OF THE DATA SET
C               CONTAINING THE APPLICABLE MEASURED VALUES
C
C       N1      NUMBER OF FIRST BOUNDARY CONDITION TO BE READ
C
C       NBC     NUMBER OF BOUNDARY CONDITIONS TO BE CONSIDERED BY TDDSBC
C
C       NDT     NUMBER OF TIME STEPS IN THE SIMULATION, DUMMY ARGUEMENT
C
C       NEQ     NUMBER OF CONSTITUENTS BEING MODELED
C
C    >NINPUT<   NUMBER OF INPUT DATA TYPES FOR THE SIMULATION
C
C     >NOUT<    NUMBER OF OUTPUT DATA TYPES FOR THE SIMULATION
C
C      NREC     NUMBER OF THE RECORD IN UNIT 31 WITH DESIRED GRID DATA
C
C     NSTART    NUMBER OF THE RECORD IN UNIT 31 TO BE READ FIRST FOR
C               THE DESIRED CONSTITUENT
```

```
C
C       NSUM    COUNTS NUMBER OF SQUARED ERROR CALCULATIONS SUMMED
C
C        NTS    NUMBER OF TIME STEPS IN THE SIMULATION, STATIC
C
C        NXS    NUMBER OF GRID POINT IN SYSTEM, STATIC
C
C       NXSEC   NUMBER OF GRID POINTS IN SYSTEM, DUMMY ARGUEMENT
C
C       PARA    CHARACTER CLASSIFICATION OF CONSTIUENT, A DIGIT
C               ('1','2'...'9','0')
C
C      PRINTR   UNIT NUMBER OF PRINTER (36)
C
C      PRTMSG   TELLS TDDS TO PRINT ERROR MESSAGES, LOGICAL (.TRUE.)
C
C       PUNCH   UNIT NUMBER OF THE CARD PUNCH (37)
C
C      RTCODE   RETURN CODE FROM TDDS
C
C       SIMT    DURATION OF SIMULATION [HOURS]
C
C        SUM    SUM OF SQUARED ERRORS FOR RMS CALCULATION
C
C        TA     AIR TEMPERATURE [CELSIUS]
C
C      TDDATA   UNIT NUMBER OF TIME DEPENDENT DATA FILE (98)
C
C       TSPDY   NUMBER OF SIMULATION TIME STEPS PER DAY
C
C        TUP    CONCENTRATION AT THE UPSTREAM BOUNDARY
C
C         V     WINDSPEED [METERS/SECOND]
C
C
C*****ENTRY POINT FOR THE INITIAL CALL*********************************
C
      SUBROUTINE TDDS(NXSEC,DT,NDT,IPLOT)
C
C     DECLARATION STATEMENTS
C
      COMMON /CDADIO/ PRINTR,PUNCH,DSREF,TDDATA,LIST,RTCODE
C
      CHARACTER    TLCM(4)/'T','L','C','M'/
      CHARACTER*2  BC(9),HY(5)/' Q',' A',' B','QT','QL'/,INTYPE(54),
     1             OUTTYP(10)
      INTEGER*2    DATAI2(50,51,10),GC(200),I2(240),INGRID(50,54),
     1             NINGRD(54),NOUTGR(10),OUTGRD(50,10),RDPDY(54),S(10)
      INTEGER*4    I4(120),DATAI4(50,3,10),STAID(50)
      REAL         CIN(50,2),DATA(200),DELTIM(10),FLOW(5,50),GT(50)
C
      CHARACTER    PARA
      CHARACTER*2  CHAR
      INTEGER*2    BD,BH,BMN,BMO,BY,ED,EH,EMN,EMO,EY,I0/0/,JI2,LIST,
     1             RTCODE,TSPDY
      INTEGER*4    BTIME,DSREF/38/,ETIME,PRINTR/36/,PUNCH/37/,TDDATA/98/
      LOGICAL      PRTMSG/.TRUE./
C
      EQUIVALENCE (I4(1),I2(1))
      SAVE BD,BH,BMN,BMO,BTIME,BY,DATAI2,DTT,ED,EH,EMN,EMO,ETIME,EY,
     1     INGRID,INTYPE,NINPUT,NINGRD,NOUT,NOUTGR,NTS,NXS,OUTGRD,
     2     OUTTYP,DATAI4,RDPDY,SIMT,STAID,TSPDY
C
C     SET STATIC VARIABLES
C
      TSPDY=INTS(24.0/DT)
      DTT=DT
      NXS=NXSEC
      NTS=NDT
      SIMT=DT*NDT
C
C     READ NUMBER OF DATA TYPES, TIME INFO, AND STATION INFO
```

```
C
      READ(35,3500) NINPUT,NOUT,LIST
 3500 FORMAT(10X,10I7)
      IF (NOUT .GT. 0) THEN
       IF (IPLOT .EQ. 0 .OR. IPLOT .EQ. 2) IPLOT=IPLOT+1
       IF (IPLOT .EQ. -2) IPLOT=-3
      END IF
      READ(35,3500) BY,BMO,BD,BH,BMN,EY,EMO,ED,EH,EMN
      CALL DTCODE(BY,BMO,BD,BH,BMN,BTIME,JI4,*100)
      CALL DTCODE(EY,EMO,ED,EH,EMN,ETIME,JI4,*100)
  100 READ(35,3505) (STAID(I),I=1,NXS)
 3505 FORMAT(10X,5I14)
C
C     READ DATA TYPES AND GRID NUMBERS
C
      DO 140 I=1,NINPUT
       READ(35,3510) INTYPE(I),RDPDY(I),NINGRD(I),
     1               (INGRID(K,I),K=1,NINGRD(I))
 3510  FORMAT(15X,A2,9I7,4(10X,10I7,/))
C      ASSUME RDPDY=TSPDY AND INPUT AT ALL GRIDS IF NOT READ IN
       IF (RDPDY(I) .EQ. 0) RDPDY(I)=TSPDY
       IF (NINGRD(I) .GT. 0) GO TO 140
       NINGRD(I)=NXS
       DO 130 L=1,NXS
        INGRID(L,I)=L
  130  CONTINUE
  140 CONTINUE
      DO 180 I=1,NOUT
       READ(35,3510) OUTTYP(I),NOUTGR(I),(OUTGRD(L,I),L=1,NOUTGR(I))
       IF (NOUTGR(I) .NE. 0) GO TO 180
C      IF NOT SPECIFIED ASSUME OUTPUT AT ALL GRIDS EXCEPT GRID ONE
       NOUTGR(I)=NXS-1
       DO 170 L=1,NOUTGR(I)
        OUTGRD(L,I)=L+1
  170  CONTINUE
  180 CONTINUE
C
C     CALL DADIO
C
      CALL DADIO
      RETURN
C
C*****ENTRY POINT TDDSHY, RETURN HYDRAULIC VALUES TO MODEL**************
C
      ENTRY TDDSHY(FLOW,J)
      JJ=MOD(J,10)+1
      IF (JJ .EQ. 1) CALL DSTIME(J,SIMT,DTT,BY,BMO,BD,BH,BMN,
     1                           EY,EMO,ED,EH,EMN)
       DO 210 L=1,5
        DO 200 K=1,NINPUT
         IF (INTYPE(K) .NE. HY(L)) GO TO 200
C
C        DATA TYPE IS A HYDRAULIC PROPERTY
C
         IF (JJ .EQ. 1) DTDATA=24.0/RDPDY(K)
         DO 190 I=1,NINGRD(K)
          IF (JJ .NE. 1) GO TO 185
C
C         RETRIEVE DATA FOR THE NEXT TEN SIMULATION TIME STEPS
C
          IF (L .LE. 3) THEN
C
C          RETRIEVE INTEGER*4 DATA
C
           CALL DADI(STAID(INGRID(I,K)),INTYPE(K),BY,BMO,BD,BH,BMN,
     1                EY,EMO,ED,EH,EMN,I4(1),RDPDY(K),IO,PRTMSG)
C
C          CONVERT DATA FROM TDDS TO SIMULATION TIME STEP
C
           CALL SCRIPT(DTT,DTDATA,S,DELTIM)
            DO 182 M=1,10
             IF (J+M-1 .GT. NTS) GO TO 185
```

```
              DATAI4(INGRID(I,K),L,M)=INTL(DELTIM(M)/DTDATA)*
     1                        (I4(S(M))-I4(S(M)-1))+I4(S(M)-1)
  182         CONTINUE
              GO TO 185
             ELSE
C
C             RETRIEVE INTEGER*2 DATA
C
              CALL DADI(STAID(INGRID(I,K)),INTYPE(K),BY,BMO,BD,BH,BMN,
     1                  EY,EMO,ED,EH,EMN,I2(1),RDPDY(K),IO,PRTMSG)
C
C             CONVERT DATA FROM TDDS TO SIMULATION TIME STEP
C
              CALL SCRIPT(DTT,DTDATA,S,DELTIM)
              DO 184 M=1,10
               IF (J+M-1 .GT. NTS) GO TO 185
               DATAI2(INGRID(I,K),K,M)=INTS((DELTIM(M)/DTDATA)*
     1                          (I2(S(M))-I2(S(M)-1))+I2(S(M)-1))
  184         CONTINUE
            END IF
C
C           CONVERT PROPERLY TIMED TDDS DATA TO SIMULATION DATA
C
  185       IF (L .EQ. 1) GO TO 190
            IF (L .LE. 3) THEN
              FLOW(L,INGRID(I,K))=DATAI4(INGRID(I,K),L,JJ)/100.0
             ELSE
              FLOW(L,INGRID(I,K))=DATAI2(INGRID(I,K),K,JJ)/100.0
            END IF
            IF (L .EQ. 2) FLOW(1,INGRID(I,K))=DATAI4(INGRID(I,K),1,JJ)
     1                                  *36.0/FLOW(2,INGRID(I,K))
  190     CONTINUE
C
C         PROCEED TO THE NEXT HYDRAULIC PROPERTY
C
          GO TO 210
  200   CONTINUE
  210  CONTINUE
      RETURN
C
C*****ENTRY POINT TDDSBC, RETURN BOUNDARY CONDITIONS TO THE MODEL*******
C
      ENTRY TDDSBC(PARA,NBC,J,CIN,TUP,TA,V,HS,HL,EA)
      JJ=MOD(J,10)+1
      TUP=0.0
      TA=0.0
      V=0.0
      HS=0.0
      HL=0.0
      EA=0.0
      DO 230 I=1,NXS
       CIN(I,1)=0.0
       CIN(I,2)=0.0
  230 CONTINUE
      DO 240 I=1,4
       BC(I)=PARA//TLCM(I)
       IF (PARA .NE. ' ') BC(I)=TLCM(I)//PARA
  240 CONTINUE
      BC(5)='AT'
      BC(6)='WS'
      BC(7)='AP'
      BC(8)='RA'
      BC(9)='RS'
      N1=1
      IF (NBC .GT. 4 .AND. J .GT. 0) N1=5
      IF (JJ .EQ. 1) CALL DSTIME(J,SIMT,DTT,BY,BMO,BD,BH,BMN,
     1                           EY,EMO,ED,EH,EMN)
      DO 270 I=1,NINPUT
       DO 260 K=N1,NBC
        IF (INTYPE(I) .NE. BC(K)) GO TO 260
C
C        DATA TYPE IS A BOUNDARY CONDITION
```

```
C
        IF (JJ .EQ. 1) DTDATA=24.0/RDPDY(I)
        DO 250 L=1,NINGRD(I)
         IG=INGRID(L,I)
  242    IF (JJ .NE. 1) GO TO 245
C
C        RETRIEVE DATA FOR THE NEXT TEN SIMULATION TIME STEPS
C
         CALL DADI(STAID(IG),INTYPE(I),BY,BMO,BD,BH,BMN,EY,EMO,ED,EH,
     1             EMN,I2(1),RDPDY(I),IO,PRTMSG)
C
C        CONVERT DATA FROM TDDS TO SIMULATION TIME STEP
C
         CALL SCRIPT(DTT,DTDATA,S,DELTIM)
         DO 244 M=1,10
          IF (J+M-1 .GT. NTS) GO TO 245
          DATAI2(IG,I,M)=INTS((DELTIM(M)/DTDATA)*(I2(S(M))-I2(S(M)-1))+
     1                   I2(S(M)-1))
  244    CONTINUE
C
C        CONVERT PROPERLY TIMED TDDS DATA TO SIMULATION DATA
C
  245    IF (K .LE. 2) CIN(IG,K)=DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 3) TUP=DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 5) TA=DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 6) V=DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 7) HS=24.0*DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 8) HL=24.0*DATAI2(IG,I,JJ)/100.0
         IF (K .EQ. 9) EA=DATAI2(IG,I,JJ)/100.0
  250   CONTINUE
C
C       PROCEED TO THE NEXT BOUNDARY CONDITION
C
        GO TO 270
  260  CONTINUE
  270 CONTINUE
      RETURN
C
C*****ENTRY POINT TDDSRMS, COMPUTE ROOT MEAN SQUARED ERROR**************
C
      ENTRY TDDSRMS(GT,PARA,J,SUM ,NSUM)
      CHAR=' M'
      IF (PARA .NE. ' ') CHAR='M'//PARA
C
C     DETERMINE WHICH INPUT DATA TYPE CONTAINS DESIRED MEASURED VALUES
C
      DO 280 M=1,NINPUT
       IF (INTYPE(M) .EQ. CHAR) GO TO 290
  280 CONTINUE
      RETURN
  290 JJ=MOD(J,10)+1
C
C     INCREMENT SUM AND NSUM AT DESIRED GRID POINTS
C
      DO 300 I=1,NINGRD(M)
        IF (DATAI2(INGRID(I,M),M,JJ) .LT. 0) GO TO 300
        SUM=(ABS(GT(INGRID(I,M))-DATAI2(INGRID(I,M),M,JJ)/
     1      100.0))**2.0+SUM
        NSUM=NSUM+1
  300  CONTINUE
      RETURN
C
C*****ENTRY POINT TDDSO, WRITE OUTPUT TO UNIT 98************************
C
      ENTRY TDDSO(NEQ)
      CALL DECODE(BTIME,BY,BMO,BD,BH,BMN,JI4,JI2)
      CALL DECODE(ETIME,EY,EMO,ED,EH,EMN,JI4,JI2)
      DO 330 I=1,NOUT
C
C       DETERMINE RECORD FOR CONSTITUENT CONCENTRATION AT GRID ONE
C
        NSTART=1
```

```
      IF (OUTTYP(I)(:1) .NE. ' ') NSTART=ICHAR(OUTTYP(I)(2:))-176
      IF (NSTART .EQ. 0) NSTART=10
      DO 320 K=1,NOUTGR(I)
C
C       DETERMINE RECORD WITH DESIRED DATA AND READ IT
C
        NREC=NEQ*(OUTGRD(K,I)-1)+NSTART
        READ(31'NREC) (DATA(L),L=1,NTS+1)
C
C       CONVERT FROM SIMULATION DATA TO TDDS DATA
C
        DO 310 M=1,NTS+1
         GC(M)=INTS(DATA(M)*100.0)
  310   CONTINUE
C
C       SEND DATA TO TDDS
C
        CALL DADO(STAID(OUTGRD(K,I)),OUTTYP(I),BY,BMO,BD,BH,BMN,
     1            EY,EMO,ED,EH,EMN,GC(1),TSPDY,IO,PRTMSG)
  320 CONTINUE
  330 CONTINUE
      RETURN
      END
C
C*****SUBROUTINE TO COMPUTE BEGINNING AND END TIMES FOR DADI/O CALL*****
C
      SUBROUTINE DSTIME(J,SIMT,DT,BY,BMO,BD,BH,BMN,EY,EMO,ED,EH,
     1                  EMN)
C
C     DEFINE VARIABLES AND ARRAYS:
C
C      NAME                          PURPOSE
C     ------   ------------------------------------------------------------
C       BD     BEGINNING DAY OF TDDS DATA TRANSFER
C
C       BH     BEGINNING HOUR OF TDDS DATA TRANSFER
C
C       BMN    BEGINNING MINUTE OF TDDS DATA TRANSFER
C
C       BMO    BEGINNING MONTH OF TDDS DATA TRANSFER
C
C       BY     BEGINNING YEAR OF TDDS DATA TRANSFER
C
C      DATET   ELAPSED TIME FROM SIMULATION TIME ZERO TO END TIME OF
C              PLANNED DATA TRANSFER [HOURS], LESS THAN OR EQUAL TO SIMT
C
C     DPERM(K) DAYS IN MONTH K, K=1 TO 12
C
C       DT     SIMULATION TIME STEP [HOURS]
C
C       ED     ENDING DAY OF TDDS DATA TRANSFER
C
C       EH     ENDING HOUR OF TDDS DATA TRANSFER
C
C       EMN    ENDING MINUTE OF TDDS DATA TRANSFER
C
C       EMO    ENDING MONTH OF TDDS DATA TRANSFER
C
C       EY     ENDING YEAR OF TDDS DATA TRANSFER
C
C        J     SIMULATION TIME STEP
C
C      JLAST   SIMULATION TIME STEP FOR THE PREVIOUS CALL OF TIME
C
C      NDTS    NUMBER OF SIMULATION TIME STEPS OVER WHICH DATA IS TO
C              BE RETRIEVED
C
C      SIMT    DURATION OF SIMULATION [HOURS]
C
C
C
C     DIMENSION AND INITIALIZE
```

```
C
      INTEGER*2 BD,BH,BMN,BMO,BY,ED,EH,EMN,EMO,EY
      INTEGER*2 DPERM(12)/31,28,31,30,31,30,31,31,30,31,30,31/
      SAVE JLAST
      DPERM(2)=28+(4-(BY-BY/4*4))/4
      NDTS=J-JLAST
      IF (J .EQ. 0) JLAST=0
  100 IF (J .EQ. JLAST) THEN
C
C     CALC SIMULATION TIME INTERVAL FOR WHICH DATA WILL BE RETRIEVED
C
       DO 110 NDTS=10,0,-1
        DATET=(J+NDTS)*DT
        IF (DATET .LE. SIMT) GO TO 120
  110  CONTINUE
      END IF
C
C     CALCULATE AN ENDING TIME
C
  120 EMN=BMN+INTS(NDTS*DT*60.0+0.01)
      EH=BH
      ED=BD
      EMO=BMO
      EY=BY
  130 IF (EMN .LT. 60) GO TO 140
      EMN=EMN-60
      EH=EH+1
      GO TO 130
  140 IF (EH .LT. 24) GO TO 150
      EH=EH-24
      ED=ED+1
      GO TO 140
  150 IF (ED .LE. DPERM(EMO)) GO TO 170
      ED=ED-DPERM(EMO)
      EMO=EMO+1
  160 IF (EMO .LE. 12) GO TO 150
      EMO=EMO-12
      EY=EY+1
      GO TO 150
  170 IF (EY .LE. 99) GO TO 180
      EY=EY-100
      GO TO 170
C
C     IF J=JLAST, BEGINNING TIME IS CORRECT SO RETURN
C
  180 IF (J .EQ. JLAST) RETURN
C
C     CALCULATED ENDING TIME IS REALLY A NEW BEGINNING TIME
C
      JLAST=J
      BY=EY
      BMO=EMO
      BD=ED
      BH=EH
      BMN=EMN
      GO TO 100
      END
C
C*****SUBROUTINE DETERMINES HOW TO CONVERT DATA FROM TDDS TO*************
C*********************SIMULATION TIME STEP*******************************
C
      SUBROUTINE SCRIPT(DTT,DTDATA,SCRIPT,DELTIM)
C
C     DEFINE VARIABLES AND ARRAYS:
C
C      NAME                          PURPOSE
C     ------   ---------------------------------------------------------
C     DATTIM   TIME FROM CURRENT SIMULATION TIME TO THE TIME OF A TDDS
C              DATA POINT [HOURS]
C
C    DELTIM(K) TIME FROM A TDDS DATA POINT TO THE PRECEDING OR IDENTICAL
C              SIMULATION DATA POINT, BUT RETURNS THE TIME FROM THE TDDS
```

```
C               DATA POINT PRECEDING THE SIMULATION POINT DESIRED TO THE
C               DESIRED SIMULATION POINT [HOURS]
C
C     DTDATA    TIME STEP OF TDDS DATA [HOURS]
C
C       DTT     SIMULATION TIME STEP [HOURS]
C
C        M      COUNTER
C
C    SCRIPT(K) NUMBER OF THE FIRST TDDS DATA POINT AFTER THE CURRENT
C              SIMULATION TIME (IN ARRAY I2 OR I4)
C
C     SIMTIM    TIME FROM THE CURRENT SIMULATION TIME TO A FUTURE
C               SIMULATION TIME STEP [HOURS]
C
C
C     DIMENSION ARRAYS
C
      INTEGER*2 SCRIPT(10)
      REAL      DELTIM(10)
C
C     ASSUME A TDDS DATA POINT EXISTS FOR THE CURRENT TIME
C
      SCRIPT(1)=2
      DELTIM(1)=0.0
C
C     DETERMINE INTERPOLATION PARAMETERS FOR NINE MORE SIM. TIME STEPS
C
      DO 130 M=2,10
       SCRIPT(M)=SCRIPT(M-1)
  100  DATTIM=DTDATA*(SCRIPT(M)-1)
       SIMTIM=DTT*(M-1)
       DELTIM(M)=SIMTIM-DATTIM
       IF (ABS(DELTIM(M)) .LT. 0.01) GO TO 120
       IF (DELTIM(M)) 125,120,110
C
C      DELTIM>0, TRY NEXT TDDS DATA POINT
C
  110  SCRIPT(M)=SCRIPT(M)+1
       GO TO 100
C
C      DELTIM=0, TDDS AND SIMULATION DATA POINTS ARE IDENTICAL
C
  120  SCRIPT(M)=SCRIPT(M)+1
C
C      DELTIM<=0, CALC RETURNING DELTIM, GO TO NEXT SIM. TIME STEP
C
  125  IF (ABS(DELTIM(M)) .GT. 0.01) DELTIM(M)=DELTIM(M)+DTDATA
  130 CONTINUE
      RETURN
      END
*********************************************************************
*                        SUBROUTINES.F77                            *
*                 BY THE SURFACE WATER BRANCH, USGS                 *
*                          JUNE 1985                                *
*********************************************************************
      SUBROUTINE DADIO                                              JL1206
C     SUBROUTINE DADIO (PRINTR,PUNCH,DSREF,TDDATA,LIST,RTCODE)      DD  10
C                                                                   DD  20
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***DD  30
C***                                                               ***DD  40
C***  MAIN ENTRY POINT CALLED TO INITIALIZE LOGICAL UNIT NUMBERS.   ***DD  50
C***  ALSO CALLED TO PRINT CHRONOLOGICAL SUMMARY OF FILED DATA AND  ***DD  60
C***  DIRECTORY OF DATA SETS.                                      ***DD  70
C***                                                               ***DD  80
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***DD  90
C***                                                                DD 100
C***  NOTE:  THE FOLLOWING DATA TYPES CAN NOW BE STORED AND         DD 110
C***         RETRIEVED BY THIS ROUTINE.                             DD 120
C***                                                                DD 130
C***         ' Z' : INTEGER*2  (STAGE)                              DD 140
C***         ' Q' : INTEGER*4  (DISCHARGE)                          DD 150
```

```
C***          'A2' : TWO PARAMETER ASSOCIATED INTEGER*2                    DD 160
C***          'A4' : TWO PARAMETER ASSOCIATED INTEGER*4                    DD 170
C                                                                          DD 180
C # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
C     Group          Parameter          Code            Information
C-----------------------------------------------------------------------------
C  Hydrologic     Water Level            Z      Measured positively above datum.
C                 Point Velocity         V      In terms of speed and direction.
C                 Point Velocity         U      In terms of N&E vector components.
C                 X sec. Velocity       XV      In terms of speed and direction.
C                 X sec. Velocity       XU      In terms of N&E vector components.
C                 Discharge              Q
C                 Lateral Flow Rate     QL
C                 Tributary Flow Rate   QT
C                 Top Width              B      Surface width at cross section.
C                 X-sec. Area            A
C                 Wetted Perimeter      WP
C-----------------------------------------------------------------------------
C  Meteorologic   Air Temperature       AT,TA
C                 Atmospheric Press.    AP,PA
C                 Relative Humidity     RH
C                 Incoming Atmos-       RA
C                  pheric Radiation
C                 Incoming Solar        RS
C                  Radiation
C                 Radition              R0-R9  Atmos. radiation - ten possible
C                                               definitions.
C                 Precipitation          P     In equivalent liquid measure.
C                 Air (Wind) Velocity   AV     In terms of speed and direction.
C                 Air (Wind) Velocity   AU     In terms of N&E vector components.
C                 Gust* (Wind) Vel.     GV     In terms of speed and direction.
C                 Gust* (Wind) Vel.     GU     In terms of N&E vector components.
C            WD
C                 Wind Speed            WS
C-----------------------------------------------------------------------------
C  Water-         Water Temperature      T
C   Quality       pH                    PH
C  (Transport)    Conductivity           C     Adjusted to standard conditions.
C                 Salinity               S
C                 Dissolved Concen-     C0-C9  Ten possible definitions.
C                  tration
C                 Suspended Conc.       K0-K9  Ten possible definitions.
C                 Lateral Conc.         L0-L9  Ten possible definitions.
C                 Measured Conc.        M0-M9  Ten possible definitions.
C                 Tributary Conc.       T0-T9  Ten possible definitions.
C # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
C                                                                          DD 220
      COMMON    /INDEX/ NOENT,STATNO(930),BTIME(930),ETIME(930),           DD 230
     +                  PTIME(930),DUMMY(930),RPERD(930),RDINT(930),       DD 240
     +                  DTYPE(930),ERFLAG(6,930),TRKNO(930),BYTNO(930),    DD 250
     +                  PASS(930)                                          DD 260
      INTEGER   NOENT,STATNO,BTIME,ETIME,PTIME                             DD 270
      INTEGER*2 RPERD,RDINT,ERFLAG,TRKNO,BYTNO,ERF(1),RDPDAY
      CHARACTER*2 DTYPE,TYPE
C      LOGICAL*2 OR *1 NOT STANDARD FORTRAN '77, CHANGED TO *4
      LOGICAL PASS
      COMMON    /DSTRF/ ISTK,FLDSTA(500),DATUM(500)
      INTEGER   FLDSTA
      INTEGER*2 DATUM
      COMMON    /CDADIO/ PRINTR,PUNCH,DSREF,TDDATA,LIST,RTCODE           JL1206
      COMMON    /DATYPE/ LENTRK,NINTRK,NENTRK,LENIND,MAXENT              JL0123
C                                                                          DD 380
      INTEGER   LENIND,MAXENT,MXLN/60/,
     +          LASTDY(12) /31,28,31,30,31,30,31,31,30,31,30,31/,          DD 400
     +          IREADS(6) /5,6,10,15,30,60/,IBAR/'  |'/,
     +          IRCODE(6) /' T |',' U |',' V |',' W |','X |',' Y |'/,      DD 420
     +          LINSEG(2) /'----','---+'/,SYMBOL(6) /'  |','* |',          DD 430
     +          ' # |',
      INTEGER   ICHAR(858) /' MON','TH  ',' 1  ',' 2  ',' 3  ',' 4  ',     DD 450
     +          ' 5  ',' 6  ',' 7  ',' 8  ',' 9  ',' 10 ',' 11 ',' 12 ', DD 460
     +          ' 13 ',' 14 ',' 15 ',' 16 ',' 17 ',' 18 ',' 19 ',' 20 ', DD 470
     +          ' 21 ',' 22 ',' 23 ',' 24 ',' 25 ',' 26 ',' 27 ',' 28 ', DD 480
```

```
     +          ' 29 ',' 30 ',' 31 ','----',32*'---+',792*'    '/
      INTEGER   MONTH(24) /' JAN',' |',' FEB',' |',' MAR','CH |',      DD 500
     +          ' APR','IL |',' MAY',' |',' JUN','E |',' JUL','Y |', JL0814
     +          ' AUG',' |',' SEP','T. |',' OCT',' |',' NOV',' |', DD 520
     +          ' DEC',' |'/                                          DD 530
      CHARACTER*4 DUMMY,BLK/'    '/,YESNO(2)/'YES ',' NO '/
      INTEGER*2 LTRKNO/0/,DAYLMT/90/,BLK2/'  '/,NOTYPS/91/,NOA2TP/8/
      CHARACTER*2     AA/' A'/,BB/' B'/,QQ/' Q'/,ZZ/' Z'/,BLANK/'  '/,
     +                A2/'A2'/,A4/'A4'/,
     + TYPES(91)/' A',' B',' C',' L',' P',' Q',' S',' T',' U',' V',' Z',
     +           'AP','AT','AV','AU','GU','GV','PH','RH','WP','XU','XV',
     +                'C0','C1','C2','C3','C4','C5','C6','C7','C8','C9',
     +                'K0','K1','K2','K3','K4','K5','K6','K7','K8','K9',
     +                'L0','L1','L2','L3','L4','L5','L6','L7','L8','L9',
     +                'R0','R1','R2','R3','R4','R5','R6','R7','R8','R9',
     +                'M0','M1','M2','M3','M4','M5','M6','M7','M8','M9',
     +                'TA','PA','QL','QT','RA','RS',
     +                'T0','T1','T2','T3','T4','T5','T6','T7','T8','T9',
     +                'WD','WS',' M'/
      CHARACTER*2 A2TYPS(8)/' U',' V','AU','AV','GU','GV','XU','XV'/
      REAL      STANO3                                                 DD 600
      INTEGER   PRINTR,PUNCH,DSREF,TDDATA,YR,MO,DA,PDATE,IS,ISKIP,     DD 620
     +          BRANCH,STANUM,STANO1,STANO2,SIZE,NSUB,NSTEP,LSTEP,     DD 630
     +          BT,ET,IBELT,IEELT,BELT,EELT,IELT,     IEELTT,IPELT,    DD 640
     +          ELTBY,ELTIBY,IBYADD,LTENT,NXENT,IELE,JELE,KELE,LELE,   DD 650
     +          NELE,IBYTE,LBYTE,NBYTE,LENTRK,NOY,NOYM1,PY(25),PYR,    DD 660
     +          RV(366),RVL,RVK,IABRVL,IOFF,IEXP,ISUB,ISUB15,IRDS,IXL, DD 670
     +          IXM,KEEP,IERTST,IRDCOD,IIND,NIND,KZ,LN,NEXREC,ITRKNO,  DD 680
     +          IMOD,NENTRK,KK
      INTEGER*2 LIST,RTCODE,STRIP,     RDPDY,DT,IBY,IBMO,IBD,DATA(1),  DD 700
     +          IBH,IBM,IEY,IEMO,IED,IEH,IEM,BY,BMO,BD,BH,BMN,EY,EMO,  DD 710
     +          ED,EH,EMN,BJDAYN,EJDAYN,IPY,IPMO,IPD,IPH,IPM,PJDAYN,IY, DD 720
     +          IRDINT,IBYTNO,LRDPDY,ERRORS(15),IERVAL,DYLMT           DD 730
      LOGICAL   PRTMSG,UPDATE,NODATA,ERROR,FLAGS,OPEN/.FALSE./         DD 740
     1         ,SORTED/.FALSE./
C                                                                      DD 750
      INTEGER   DATAI4(1558)                                           DD 760
      INTEGER*2 DATAI2(3116)
      INTEGER NINTRK,NEXENT,STNUM,LL
      SAVE NELE,LTRKNO,LASTDY,OPEN,SORTED,LELE,ITRKNO,DATAI2,DATAI4,
     1 ICHAR,NEXENT,PDATE,ELTIBY,YR,MO,DA,I,JELE,NEXREC
      EQUIVALENCE (DATAI2(1),DATAI4(1)),(ERFLAG(1,1),ERF(1))
C                                                                      DD 800
C***  STATEMENT FUNCTION : NUMBER OF MINUTES IN YEAR IY (INTEGER*2)    DD 810
      ITIMEY(IY)=525600+(4-(IY-IY/4*4))/4*1440                         DD 830
C                                                                      DD 840
C                                                                      DD 870
      NELE=3116                                                        DD 880
      MAXENT=930
      LENIND=40
      IF (TDDATA .EQ. 98)   GO TO 100                                  DD 890
            WRITE (PRINTR,9299)                                        DD 930
            RTCODE=-7                                                  DD 940
            RETURN                                                     DD 950
C                                                                      DD 960
C***  CHECK IF FILE HAS BEEN OPENED - IF NOT, READ IN INDEX.           DD 970
C                                                                      DD 980
  100 RTCODE=0                                                         DD 990
      IF (OPEN) GO TO 200                                              DD1000
         OPEN=.TRUE.                                                   DD1010
         CALL JULDAT(YR,MO,DA)                                         DD1050
         CALL TIME(IS)                                                 DD1060
         IS=(IS+49)/100                                                DD1070
         IBY=YR                                                        DD1080
         IBMO=MO                                                       DD1090
         IBD=DA                                                        DD1100
         IBH=IS/3600                                                   DD1110
         IBM=(IS-IBH*3600)/60                                          DD1120
         CALL DTCODE(IBY,IBMO,IBD,IBH,IBM,PDATE,BELT,*150)              PRIME
         LENTRK=NELE*2                                                 DD1140
         NENTRK=LENTRK/LENIND                                          DD1150
         NINTRK=MAXENT/NENTRK                                          DD1160
```

```
      IF (MOD(MAXENT,NENTRK).NE.0) NINTRK=NINTRK+1                      DD1170
      IIND=1                                                            DD1190
      NIND=NENTRK                                                       DD1200
      NOENT=0                                                           DD1210
C     RECORD LENGTH IS MEASURED IN WORDS (16 BITS OR 2 BYTES) ON
C     PR1ME COMPUTERS, AND IN BYTES ON THE AMDAHL FOR UNFORMATTED
C     RECORDS.
CIBM  OPEN(UNIT=TDDATA,ACCESS='DIRECT',RECL=6232,ERR=105,IOSTAT=IOS)
      OPEN(UNIT=TDDATA,ACCESS='DIRECT',RECL=3116,ERR=105,IOSTAT=IOS)
      IF(IOS.EQ.0) GO TO 106
  105    WRITE(PRINTR,9322)
         RTCODE=-9
         RETURN
  106 DO 130 J=1,NINTRK                                                 DD1220
         READ (TDDATA,REC=J,IOSTAT=IOS) (STATNO(I),BTIME(I),ETIME(I),
     +                 RPERD(I),DTYPE(I),(ERFLAG(K,I),K=1,6),DUMMY(I),
     +                      PTIME(I),TRKNO(I),BYTNO(I),I=IIND,NIND)
         IF(IOS.GT.0) THEN
            WRITE(PRINTR,9333)
            RTCODE=-10
            RETURN
         ELSE IF(IOS.LT.0) THEN
            GO TO 160
         ENDIF
            DO 110 I=IIND,NIND                                          DD1270
               IF (STATNO(I).EQ.0) GO TO 120                            DD1280
                  RDINT(I)=1440/RPERD(I)                                DD1290
                  NOENT=I                                               DD1300
  110       CONTINUE                                                    DD1310
  120    IF (NIND.EQ.MAXENT) GO TO 140                                  DD1320
            IIND=IIND+NENTRK                                            DD1330
            NIND=NIND+NENTRK                                            DD1340
            IF (NIND.GT.MAXENT) NIND=MAXENT                             DD1350
  130 CONTINUE                                                          DD1360
  140 IF (NOENT.LT.MAXENT) GO TO 180                                    DD1370
         WRITE (PRINTR,9099)                                            DD1380
         RTCODE=-1                                                      DD1390
         RETURN                                                         DD1400
  150 WRITE (PRINTR,9177)                                               DD1410
      RTCODE=-3                                                         DD1420
      RETURN                                                            DD1430
C                                                                       DD1440
C***  NEW FILE - INITIALIZE INDEX                                       DD1450
C                                                                       DD1460
  160 IF(LIST .NE. 2 .AND. LIST .NE. 3) THEN
         WRITE(PRINTR,9133)
         RTCODE=-8
         RETURN
      ELSE
         LIST=0
      ENDIF
      DO 165 K=1,MAXENT
         STATNO(K)=0
         BTIME(K)=0
         ETIME(K)=0
         PTIME(K)=0
         RPERD(K)=0
         DTYPE(K)=BLANK
         TRKNO(K)=0
         BYTNO(K)=0
         DUMMY(K)=BLK
  165 CONTINUE
      N=MAXENT*6
      DO 166 K=1,N
         ERF(K)=0
  166 CONTINUE
      DO 170 J=1,NINTRK                                                 DD1680
         WRITE (TDDATA,REC=J)
  170 CONTINUE                                                          DD1700
  180 NEXENT=NOENT+1
      IF(LIST.EQ.2) LIST=0
      IF(LIST.EQ.3) LIST=1
```

```
      PTIME(NEXENT)=PDATE
C                                                                       DD1740
C***  READ REQUIRED DATA FROM THE DATA STATION REFERENCE FILE           DD1750
C                                                                       DD1760
      DO 190 I=1,500
  185    FORMAT (1X,I8,19X,I5)
  190    READ (DSREF,185,END=199) FLDSTA(I),DATUM(I)
         WRITE (PRINTR,9288)                                            DD1820
         RTCODE=-4                                                      DD1830
         RETURN                                                         DD1840
  199 ISTK=I-1
C                                                                       DD1850
C***  FILE IS NOW OPEN - CHECK IF LIST OPTION REQUESTED                 DD1860
C                                                                       DD1870
  200 IF (LIST.EQ.0) THEN                                               JL0214
         WRITE(PRINTR,9033)                                             JL0214
         RETURN                                                         JL0214
         ENDIF                                                          JL0214
         IF (ISTK.NE.0) GO TO 205
            WRITE (PRINTR,9311)
            RTCODE=-8
            RETURN
  205    IF (NOENT.NE.0) GO TO 210                                      DD1890
            WRITE (PRINTR,9133)                                         DD1900
            RTCODE=-5                                                   DD1910
            RETURN                                                      DD1920
C                                                                       DD1930
C***  LIST OPTION REQUESTED --                                          DD1940
C***                                                                    DD1950
C***       LIST = -1 : PRINT BOTH CALENDAR-YEAR AND DIRECTORY SUMMARIES DD1960
C***       LIST =  0 : DO NOT PRINT INDEX SUMMARIES                     DD1970
C***       LIST = +1 : PRINT ONLY THE DIRECTORY SUMMARY                 DD1980
C                                                                       DD1990
C***  FIND AND SORT ALL YEARS COVERED BY DATA ON FILE                   DD2000
C                                                                       DD2010
  210 NOY=0                                                             DD2020
      IBY=BTIME(1)/10000000                                             DD2030
      IEY=ETIME(1)/10000000                                             DD2040
      NOY=1                                                             DD2050
      PY(1)=IBY                                                         DD2060
      IF (IBY .EQ. IEY)   GO TO 220                                     DD2070
         NOY=2                                                          DD2080
         PY(2)=IEY                                                      DD2090
  220 IF (NOENT .EQ. 1)   GO TO 300                                     DD2100
         DO 270 I=2,NOENT                                               DD2110
            IBY=BTIME(I)/10000000                                       DD2120
            IEY=ETIME(I)/10000000                                       DD2130
            IY=IBY                                                      DD2140
  230       DO 240 J=1,NOY                                              DD2150
               IF (IY.EQ.PY(J)) GO TO 260                               DD2160
  240       CONTINUE                                                    DD2170
            NOY=NOY+1                                                   DD2180
            IF (NOY.LT.26) GO TO 250                                    DD2190
               WRITE (PRINTR,9111)                                      DD2200
               RTCODE=-2                                                DD2210
               RETURN                                                   DD2220
  250       PY(NOY)=IY                                                  DD2230
  260       IF (IY.EQ.IEY) GO TO 270                                    DD2240
               IY=IEY                                                   DD2250
               GO TO 230                                                DD2260
  270 CONTINUE                                                          DD2270
      IF (NOY.EQ.1) GO TO 300                                           DD2280
         NOYM1=NOY-1                                                    DD2290
         K=1                                                            DD2300
         DO 290 I=1,NOY                                                 DD2310
            IF (K.EQ.0) GO TO 300                                       DD2320
               K=0                                                      DD2330
               DO 280 J=1,NOYM1                                         DD2340
                  IF (PY(J).LT.PY(J+1)) GO TO 290                       DD2350
                     K=1                                                DD2360
                     KEEP=PY(J)                                         DD2370
                     PY(J)=PY(J+1)                                      DD2380
```

```
                    PY(J+1)=KEEP                                        DD2390
  280            CONTINUE                                              DD2400
  290       CONTINUE                                                   DD2410
C                                                                      DD2420
C***  INITIALIZE THE CALENDAR-YEAR SUMMARY OUTPUT BUFFER               DD2430
C                                                                      DD2440
  300 DO 320 L=1,2                                                     DD2450
         DO 310 LN=2,24,2                                              DD2460
            ICHAR(L+33*LN)=MONTH(L+LN-2)                               DD2470
            ICHAR(L+33*(LN+1))=LINSEG(L)                               DD2480
  310       CONTINUE                                                   DD2490
  320 CONTINUE                                                         DD2500
C                                                                      DD2510
C***  SET UP THE 'PASS' ARRAY FOR THIS DATA STATION REFERENCE          DD2520
C                                                                      DD2530
      DO 325 L=1,MAXENT
  325 PASS(L)=.TRUE.
      DO 340 I=1,ISTK                                                  DD2570
         STNUM=FLDSTA(I)                                               JL1221
         DO 330 J=1,NOENT                                              DD2590
            IF (STATNO(J) .NE. STNUM)   GO TO 330                      JL1221
               PASS(J)=.FALSE.                                         DD2610
  330       CONTINUE                                                   DD2630
  340 CONTINUE                                                         DD2640
      IF (LIST .GT. 0)   GO TO 540                                     DD2650
C                                                                      DD2660
C***  GENERATE CALENDAR-YEAR SUMMARIES FOR EACH DATA TYPE, FOR EACH    DD2670
C***  YEAR, FOR EACH FIELD STATION.                                    DD2680
C                                                                      DD2690
      DO 530 I=1,ISTK                                                  DD2700
         STNUM=FLDSTA(I)                                               JL1221
      DO 529 J=1,NOY                                                   DD2720
         PYR=PY(J)                                                     DD2730
         LASTDY(2)=29-(MOD(PYR,4)+3)/4                                 DD2740
      DO 528 KK=1,NOTYPS                                               DD2750
         DO 345 L=1,366
  345    RV(L)=SYMBOL(1)
         NODATA=.TRUE.                                                 DD2780
         DO 470 K=1,NOENT                                              DD2790
            IF (PASS(K))   GO TO 470                                   DD2800
            IF (STATNO(K) .NE. STNUM)   GO TO 470                      JL1221
            IF (DTYPE(K) .NE. TYPES(KK))   GO TO 470                   DD2810
            CALL DECODE(BTIME(K),IBY,IBMO,IBD,IBH,IBM,BJDAYN,IBELT)    DD2820
            CALL DECODE(ETIME(K),IEY,IEMO,IED,IEH,IEM,EJDAYN,IEELT)    DD2830
            IF (IBY.NE.PYR.AND.IEY.NE.PYR) GO TO 470                   DD2840
            NODATA=.FALSE.                                             DD2850
            RDPDAY=RPERD(K)                                            JL1221
            IRDINT=RDINT(K)                                            DD2870
            L=0                                                        DD2880
  350       L=L+1                                                      DD2890
            IF (IRDINT .NE. IREADS(L))   GO TO 350                     DD2900
            IRDCOD=IRCODE(L)                                           DD2910
            ISUB=0                                                     DD2920
            ISUB15=0
            IOFF=1                                                     DD2930
            IF (IBY.EQ.IEY) GO TO 370                                  DD2940
               ELTIBY=ITIMEY(IBY)                                      DD2950
               IF (IBY.EQ.PYR) GO TO 360                               DD2960
                  IOFF=(ELTIBY-IBELT)/1440+1                           DD2970
                  IBELT=IRDINT                                         DD2980
                  BJDAYN=1                                             DD2990
                  GO TO 370                                            DD3000
  360             EJDAYN=ELTIBY/1440                                   DD3010
                  IEELT=ELTIBY                                         DD3020
  370       DO 460 L=BJDAYN,EJDAYN                                     DD3030
               IF (ISUB.EQ.0) GO TO 380                                DD3040
               IF (MOD(IOFF+14,15).NE.0) GO TO 410                     DD3050
  380          ISUB=(IOFF+14)/15                                       DD3060
               ISUB15=(ISUB-1)*15                                      DD3070
               DO 385 LL=1,15
  385          ERRORS(LL)=0
               IERVAL=ERFLAG(ISUB,K)                                   DD3100
```

```
                IEXP=14                                                 DD3110
  390           IERTST=2**IEXP                                          DD3120
                IF (IERVAL.LT.IERTST) GO TO 400                         DD3130
                   ERRORS(IEXP+1)=1                                     DD3140
                   IERVAL=IERVAL-IERTST                                 DD3150
  400           IEXP=IEXP-1                                             DD3160
                IF (IEXP.GE.0) GO TO 390                                DD3170
  410           RVL=RV(L)                                               DD3180
                IEELTT=L*1440                                           DD3190
                IF (IEELT.LT.IEELTT) IEELTT=IEELT                       DD3200
                IRDS=(IEELTT-IBELT)/IRDINT+1                            DD3210
                IBELT=IBELT+IRDS*IRDINT                                 DD3220
                IF (RVL.EQ.IBAR) RVL=0                                  DD3230
                IF (IRDS.EQ.RDPDAY) GO TO 430                           JL1221
                   LRDPDY=IABS(RVL)/1000                                DD3250
                   IF (LRDPDY.NE.0) GO TO 420                           DD3260
                      RVL=RDPDAY*1000                                   JL1221
                      GO TO 430                                         DD3280
  420              IF (LRDPDY.EQ.RDPDAY) GO TO 430                      JL1221
C                  **DUAL DENSITY RECORD**
                   RV(L)=SYMBOL(3)                                      DD3300
                   GO TO 460                                            DD3310
  430           RVL=ISIGN(IABS(RVL)+IRDS,RVL)                           DD3320
C               **COMPLETE DAY**
                RV(L)=RVL                                               DD3330
                IF (RVL.LT.0) GO TO 440                                 DD3340
                   IF (ERRORS(IOFF-ISUB15).EQ.0) GO TO 440              DD3350
                   RVL=-RVL                                             DD3360
C                  **MARK ERROR FLAG**
                   RV(L)=RVL                                            DD3370
  440           IABRVL=IABS(RVL)                                        DD3380
                IF (IABRVL-IABRVL/1000*1000.NE.RDPDAY) GO TO 460        JL1221
                   IF (RVL.GT.0) GO TO 450                              DD3400
C                     **COMPLETE DAY, FLAGS**
                      RV(L)=SYMBOL(5)                                   DD3410
                      GO TO 460                                         DD3420
C                  **COMPLETE DAY, NO FLAGS**
  450              RV(L)=IRDCOD                                         DD3430
  460        IOFF=IOFF+1                                               DD3440
  470     CONTINUE                                                     DD3450
          IF (NODATA) GO TO 528                                        DD3460
             DO 490 K=1,366                                            DD3470
                RVK=RV(K)                                               DD3480
C               IF (RVK.GT.288288) GO TO 490                            DD3490
                IF (RVK.LE.IRCODE(6).AND.RVK.GE.IBAR) GO TO 490         JL1228
                   IF (RVK.GT.288287) GO TO 480                         JL1228
C                  IF (RVK.LT.0) GO TO 480                              DD3500
C                     **PARTIAL DAY, NO FLAGS**
                      RV(K)=SYMBOL(2)                                   DD3510
                      GO TO 490                                         DD3520
C                  **PARTIAL DAY, FLAGS**
  480              RV(K)=SYMBOL(4)                                      DD3530
  490        CONTINUE                                                  DD3540
             M=1                                                       DD3550
             DO 520 K=1,12                                             DD3560
                IXL=2+66*K                                              DD3570
                IXM=IXL+33                                              DD3580
                MONEND=LASTDY(K)                                        DD3590
                DO 500 L=1,MONEND                                       DD3600
                   ICHAR(IXL+L)=RV(M)                                   DD3610
                   ICHAR(IXM+L)=LINSEG(2)                               DD3620
                   M=M+1                                                DD3630
  500           CONTINUE                                                DD3640
                KZ=31-LASTDY(K)                                         DD3650
                IF (KZ.LT.1) GO TO 520                                  DD3660
                   DO 510 L=1,KZ                                        DD3670
                      ICHAR(IXL+LASTDY(K)+L)=SYMBOL(6)                  DD3680
                      ICHAR(IXM+LASTDY(K)+L)=LINSEG(2)                  DD3690
  510              CONTINUE                                             DD3700
  520        CONTINUE                                                  DD3710
             STAN01=STNUM/1000000                                      JL1221
             STAN02=STNUM-STAN01*1000000                               JL1221
```

```
            STAN03=STAN02/100.+0.005                                    DD3740
            WRITE (PRINTR,9066)                                         DD3750
            WRITE (PRINTR,9077) YR,MO,DA                                DD3760
            WRITE (PRINTR,9088) TYPES(KK),PYR,STAN01,STAN03,ICHAR       DD3780
  528 CONTINUE
  529 CONTINUE
  530 CONTINUE                                                          DD3850
C                                                                       DD3860
C***  GENERATE THE DIRECTORY SUMMARY -                                  DD3870
C                                                                       DD3880
  540 CALL SORTIN                                                       DD3890
      SORTED=.TRUE.
      WRITE (PRINTR,9000)                                               DD3900
      WRITE (PRINTR,9077) YR,MO,DA                                      DD3910
      WRITE (PRINTR,9011)                                               DD3920
      LN=15                                                             DD3940
      DO 560 I=1,NOENT                                                  DD3950
         IF (PASS(I))   GO TO 560                                       DD3960
            LN=LN+1                                                     DD3970
            STAN01=STATNO(I)/1000000                                    DD3980
            STAN02=STATNO(I)-STAN01*1000000                             DD3990
            STAN03=STAN02/100.+0.005                                    DD4000
            CALL DECODE (BTIME(I),IBY,IBMO,IBD,IBH,IBM,BJDAYN,IBELT)    DD4010
            CALL DECODE (ETIME(I),IEY,IEMO,IED,IEH,IEM,EJDAYN,IEELT)    DD4020
            CALL DECODE (PTIME(I),IPY,IPMO,IPD,IPH,IPM,PJDAYN,IPELT)    DD4030
            IF (IBY .NE. IEY)   IEELT=IEELT+ITIMEY(IBY)                 DD4040
            SIZE=(IEELT-IBELT)/RDINT(I)+1                               DD4050
            L=2                                                         DD4060
            DO 550 J=1,6                                                DD4070
               IF (ERFLAG(J,I) .NE. 0)   L=1                            DD4080
  550       CONTINUE                                                    DD4090
            WRITE (PRINTR,9022) I,STAN01,STAN03,IBY,IBMO,IBD,IBH,IBM,   DD4100
     +                          IEY,IEMO,IED,IEH,IEM,IPY,IPMO,IPD,IPH,  DD4110
     +                          IPM,RPERD(I),SIZE,DTYPE(I),TRKNO(I),    DD4120
     +                          BYTNO(I),YESNO(L)                       DD4130
            IF (LN .LT. MXLN)   GO TO 560
               WRITE (PRINTR,9033)                                      DD4150
               WRITE (PRINTR,9011)                                      DD4160
               LN=7                                                     DD4170
  560 CONTINUE                                                          DD4180
      WRITE (PRINTR,9044)                                               DD4190
      LN=LN+1                                                           DD4194
      IF (LN .GT. MXLN)   RETURN
      DO 570 K=LN,MXLN
         WRITE (PRINTR,9055)                                            DD4210
  570 CONTINUE                                                          DD4220
      RETURN                                                            DD4230
C                                                                       DD4310
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***DD4320
C***                                                                  ***DD4330
C***  SECONDARY ENTRY POINT CALLED TO EXTRACT DATA FROM THE D/A FILE  ***DD4340
C***                                                                  ***DD4350
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***DD4360
C***                                                                    DD4370
C***  RESTRICTION: DATA RETRIEVED DO NOT SPAN A RECORDING DENSITY CHANGE DD4380
C                                                                       DD4390
      ENTRY DADI (STANUM,TYPE,BY,BMO,BD,BH,BMN,EY,EMO,ED,EH,EMN,DATA,   DD4400
     +           RDPDY,STRIP,PRTMSG)                                    JL1206
C                                                                       DD4420
      IF (OPEN)   GO TO 590                                             DD4430
  580    WRITE (PRINTR,9122)                                            DD4440
         RTCODE=-6                                                      DD4450
         RETURN                                                         DD4460
  590 ASSIGN 710 TO BRANCH                                              DD4470
      NODATA=.TRUE.                                                     DD4480
      IF (NOENT .NE. 0)   GO TO 600                                     DD4490
         WRITE (PRINTR,9133)                                            DD4500
         RTCODE=-5                                                      DD4510
         RETURN                                                         DD4520
C                                                                       DD4530
C***  INITIALIZATION AND ARGUMENT ERROR CHECKS FOR BOTH 'DADI' & 'DADO' DD4540
C                                                                       DD4550
```

```
  600 RTCODE=0                                                          DD4560
      IF ((RDPDY .EQ. 288) .OR. (RDPDY .EQ. 240))   GO TO 610            DD4570
      IF ((RDPDY .EQ. 144) .OR. (RDPDY .EQ. 96))   GO TO 610            DD4580
      IF ((RDPDY .EQ. 48) .OR. (RDPDY .EQ. 24))   GO TO 610             DD4590
         IF (PRTMSG)   WRITE (PRINTR,9144)                              DD4600
         RTCODE=1                                                       DD4610
         GO TO 1310                                                     DD4620
  610 IRDINT=1440/RDPDY                                                 DD4630
      IF ((BMN-BMN/IRDINT*IRDINT) .EQ. 0)   GO TO 630                   DD4640
      IF ((EMN-EMN/IRDINT*IRDINT) .EQ. 0)   GO TO 630                   DD4650
  620    IF (PRTMSG)   WRITE (PRINTR,9166)                              DD4660
         RTCODE=3                                                       DD4670
         GO TO 1310                                                     DD4680
  630 CALL DTCODE (BY,BMO,BD,BH,BMN,BT,BELT,*620)                        PRIME
      CALL DTCODE (EY,EMO,ED,EH,EMN,ET,EELT,*620)                        PRIME
      IF (BY .GT. EY)   GO TO 620                                       DD4710
      IF (BY .LT. EY)   EELT=EELT+ITIMEY(BY)                            DD4720
      IF (BELT .GT. EELT)   GO TO 620                                   DD4730
      DO 640 I=1,ISTK                                                   DD4740
         IF (STANUM .EQ. FLDSTA(I))   GO TO 650                         DD4750
  640 CONTINUE                                                          DD4760
         IF (PRTMSG)   WRITE (PRINTR,9155)                              DD4770
         RTCODE=2                                                       DD4780
         GO TO 1310                                                     DD4790
C                                                                       DD4800
C***  CHECK FOR INVALID DATA TYPE                                       DD4810
C***  FIND THE APPROPRIATE 'STEP' VALUE FOR THE DATA TYPE REQUESTED.    DD4820
C                                                                       DD4830
  650 IZCORR=0
      IF (TYPE .EQ. ZZ)   THEN
         IZCORR=DATUM(I)
         NSTEP=1                                                        DD4850
      ELSE IF (TYPE .EQ. QQ .OR. TYPE .EQ. AA .OR. TYPE .EQ. BB .OR.
     1        TYPE .EQ. A2) THEN
         NSTEP=2                                                        DD4880
      ELSE IF (TYPE .EQ. A4)   THEN
         NSTEP=4                                                        DD4940
      ELSE
         DO 660 II=1,NOA2TP
            IF(TYPE .NE. A2TYPS(II)) GO TO 660
            NSTEP=2
            GO TO 700
  660    CONTINUE
         DO 670 II=1,NOTYPS
            IF(TYPE .NE. TYPES(II)) GO TO 670
            NSTEP=1
            GO TO 700
  670    CONTINUE
         WRITE (PRINTR,9188)
         RTCODE=5
         GO TO 1310
      ENDIF
C                                                                       DD4990
C***  'BRANCH' TO THE APPROPRIATE PROCESSING ROUTINE - DADI OR DADO     DD5000
C                                                                       DD5010
  700 GO TO BRANCH, (710,1060)                                          DD5020
C                                                                       DD5030
C***  SET UP DATA BUFFER OF ARGUMENT LIST - INITIALIZE ERROR FLAGS.     DD5040
C                                                                       DD5050
  710 KELE=((EELT-BELT)/IRDINT+1)*NSTEP                                 DD5070
      DO 715 I=1,KELE
  715 DATA(I)=BLK2
      ERROR=.FALSE.                                                     DD5090
      FLAGS=.FALSE.                                                     DD5100
      IBYADD=0                                                          DD5110
C                                                                       DD5120
C***  SET UP ASSIGNED GO TO PARAMETER FOR THE STRIP OPTION --           DD5130
C***  REASON: USED IN LARGE LOOP AND THE ASSIGNED GO TO IS MUCH FASTER. DD5140
C                                                                       DD5150
      ASSIGN 890 TO ISKIP                                               DD5160
      IF (STRIP)   720,740,730                                          DD5170
  720 ASSIGN 850 TO ISKIP                                               DD5180
```

```
      GO TO 740                                                         DD5190
  730 ASSIGN 880 TO ISKIP                                               DD5200
C                                                                       DD5210
C***  SEQUENCE THROUGH THE INDEX TO RETRIEVE THE REQUESTED DATA.        DD5220
C                                                                       DD5230
  740 DO 980 I=1,NOENT                                                  DD5240
         IF (STANUM .NE. STATNO(I))   GO TO 980                         DD5250
         IF (TYPE .NE. DTYPE(I))   GO TO 980                            DD5260
         IF ((ET .LT. BTIME(I)) .OR. (BT .GT. ETIME(I)))   GO TO 980    DD5270
         IF (RPERD(I) .EQ. RDPDY)   GO TO 750                           DD5280
            IF (PRTMSG)   WRITE (PRINTR,9199) RPERD(I)                  DD5290
            RTCODE=6                                                    DD5300
            GO TO 1310                                                  DD5310
C                                                                       DD5320
C***  INDEX ENTRY #I CONTAINS AT LEAST PART OF REQUESTED DATA.          DD5330
C                                                                       DD5340
  750    ITRKNO=TRKNO(I)                                                DD5350
         IBYTNO=BYTNO(I)                                                DD5370
         NODATA=.FALSE.                                                 DD5380
         CALL DECODE (BTIME(I),IBY,IBMO,IBD,IBH,IBM,BJDAYN,IBELT)       DD5390
         CALL DECODE (ETIME(I),IEY,IEMO,IED,IEH,IEM,EJDAYN,IEELT)       DD5400
         IF (IBYADD .EQ. 0)   GO TO 760                                 DD5410
            IBYADD=0                                                    DD5420
            BELT=BELT-ELTIBY                                            DD5430
            EELT=EELT-ELTIBY                                            DD5440
  760    IF (IBY .EQ. IEY)   GO TO 770                                  DD5450
            ELTIBY=ITIMEY(IBY)                                          DD5460
            IEELT=IEELT+ELTIBY                                          DD5470
  770    IF (BY-IBY)   780,800,790                                      DD5480
C                                                                       DD5490
  780       ELTBY=ITIMEY(BY)                                            DD5500
            IBELT=IBELT+ELTBY                                           DD5510
            IEELT=IEELT+ELTBY                                           DD5520
            GO TO 800                                                   DD5530
  790       BELT=BELT+ELTIBY                                            DD5540
            EELT=EELT+ELTIBY                                            DD5550
            IBYADD=1                                                    DD5560
C                                                                       DD5570
C***  READ DATA FROM D/A FILE - COMPUTE BUFFER SUBSCRIPTS NEEDED TO     DD5580
C***  OBTAIN THE REQUESTED DATA STORED ON THIS TRACK.                   DD5590
C                                                                       DD5600
  800    READ (TDDATA,REC=ITRKNO) DATAI4
         NEXREC=ITRKNO+1                                                JL0315
         LTRKNO=ITRKNO                                                  DD5630
         IELE=(IBYTNO+1)/2                                              DD5640
         IELT=IBELT                                                     DD5650
         IF (BELT .LE. IBELT)   GO TO 830                               DD5660
            IELE=IELE+((BELT-IBELT)/IRDINT)*NSTEP                       DD5670
            IF (IELE .LE. NELE)   GO TO 820                             DD5680
  810          IBELT=IBELT+(NELE-(IBYTNO-1)/2)/NSTEP*IRDINT             DD5690
               IBYTNO=1                                                 DD5700
C              INQUIRE (UNIT=TDDATA,NEXTREC=NEXREC)                      PRIME
               ITRKNO=NEXREC                                            DD5710
               GO TO 800                                                DD5720
  820       IELT=BELT                                                   DD5730
  830    JELE=(IBYTNO+1)/2+((IEELT-IBELT)/IRDINT+1)*NSTEP-1             DD5740
         IF (EELT .LT. IEELT)   JELE=JELE-(IEELT-EELT)/IRDINT*NSTEP     DD5750
         IF (JELE .GT. NELE)   JELE=NELE                                DD5760
C                                                                       DD5770
C***  PROCESS DATA ACCORDING TO DATA TYPE                               DD5780
C                                                                       DD5790
         NSUB=(IELT-BELT)/IRDINT*NSTEP+1                                DD5800
         IF (TYPE .EQ. ZZ)   GO TO 840                                  DD5810
         IF (TYPE .EQ. QQ .OR. TYPE .EQ. AA .OR. TYPE .EQ. BB) GO TO 910
         IF (TYPE .EQ. A2)   GO TO 930                                  DD5830
         DO 835 II=1,NOA2TP
            IF (TYPE .EQ. A2TYPS(II)) GO TO 930
  835    CONTINUE
         IF (TYPE .EQ. A4)   GO TO 950                                  DD5840
         GO TO 905
C                                                                       DD5850
C***  I N T E G E R   * 2   S T A G E   D A T A   ***                   DD5860
```

```
C                                                                       DD5870
  840     DO 900 J=IELE,JELE                                            DD5880
             DT=DATAI2(J)                                               DD5890
             IF ((DT .LT. 10000) .AND. (DT .GT. -10000))   GO TO 890    DD5900
             IF (DT .LT. 20000)   ERROR=.TRUE.                          DD5910
             IF (DT .GT. 19999)   FLAGS=.TRUE.                          DD5920
             GO TO ISKIP, (850,890,880)                                 DD5930
C                                                                       DD5940
C***  DATUM IS ERRANT; DETERMINE THE ERROR CONDITION AND STRIP CODE.    DD5950
C                                                                       DD5960
  850        IF (DT)   860,890,870                                      DD5970
C                                                                       DD5980
C***  ERROR CODE -10000 OR -20000                                       DD5990
C                                                                       DD6000
  860        IF (DT .LT. -19999)   DT=DT+10000                          DD6010
             DT=-DT-10000                                               DD6020
             GO TO 890                                                  DD6030
C                                                                       DD6040
C***  ERROR CODE +10000 OR +20000                                       DD6050
C                                                                       DD6060
  870        IF (DT .GT. 19999)   DT=DT-10000                           DD6070
             DT=DT-10000                                                DD6080
             GO TO 890                                                  DD6090
C                                                                       DD6100
C***  STRIP ONLY +20000 ERROR CODE :                                    DD6110
C                                                                       DD6120
  880        IF (DT .GT. 19999)   DT=DT-20000                           DD6130
  890        DATA(NSUB)=DT+IZCORR                                       DD6140
             NSUB=NSUB+1                                                DD6150
  900     CONTINUE                                                      DD6160
          GO TO 970                                                     DD6170
C
C***  I N T E G E R  * 2  D A T A ***
C
  905    DO 906 J=IELE,JELE
            DATA(NSUB)=DATAI2(J)
            NSUB=NSUB+1
  906     CONTINUE
          GO TO 970
C                                                                       DD6180
C***  I N T E G E R   * 4   D A T A   ***                               DD6190
C                                                                       DD6200
  910     DO 920 J=IELE,JELE                                            DD6210
             DATA(NSUB)=DATAI2(J)                                       DD6220
             NSUB=NSUB+1                                                DD6230
  920     CONTINUE                                                      DD6240
          GO TO 970                                                     DD6250
C                                                                       DD6260
C***  A S S O C I A T E D   * 2   D A T A   ***                         DD6270
C                                                                       DD6280
  930     DO 940 J=IELE,JELE                                            DD6290
             DATA(NSUB)=DATAI2(J)                                       DD6300
             NSUB=NSUB+1                                                DD6310
  940     CONTINUE                                                      DD6320
          GO TO 970                                                     DD6330
C                                                                       DD6340
C***  A S S O C I A T E D   * 4   D A T A   ***                         DD6350
C                                                                       DD6360
  950     DO 960 J=IELE,JELE                                            DD6370
             DATA(NSUB)=DATAI2(J)                                       DD6380
             NSUB=NSUB+1                                                DD6390
  960     CONTINUE                                                      DD6400
C                                                                       DD6410
C***  END OF PROCESSING LOOP                                            DD6420
C                                                                       DD6430
  970     IELT=IELT+((JELE-IELE+1)/NSTEP)*IRDINT                        DD6440
          IF (JELE .LT. NELE)   GO TO 980                               DD6450
             IF ((IELT .LE. EELT) .AND. (IELT .LE. IEELT))   GO TO 810  DD6470
  980 CONTINUE                                                          DD6480
C                                                                       DD6490
C***  ALL RELATED INDEX ENTRIES HAVE BEEN PROCESSED - CHECK IF ALL DATA DD6500
C***  REQUESTED WAS RETRIEVED AND SET THE CORRECT RETURN CODE.          DD6510
```

```
C
      IF (.NOT. NODATA)    GO TO 990
         IF (PRTMSG)    WRITE (PRINTR,9211)
         RTCODE=7
         GO TO 1310
  990 DO 1000 J=1,KELE
         IF (DATA(J) .EQ. BLK2)    GO TO 1010
 1000 CONTINUE
      GO TO 1020
 1010 IF (PRTMSG)    WRITE (PRINTR,9277)
      RTCODE=11
 1020 IF (ERROR)    GO TO 1040
         IF (.NOT. FLAGS)    RETURN
            IF (PRTMSG)    WRITE (PRINTR,9266)
            IF (RTCODE .NE. 11)    GO TO 1030
               RTCODE=12
               GO TO 1310
 1030          RTCODE=10
               GO TO 1310
 1040 IF (PRTMSG)    WRITE (PRINTR,9255)
      IF (RTCODE .NE. 11)    GO TO 1050
         RTCODE=13
         GO TO 1310
 1050    RTCODE=4
         GO TO 1310
C
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
C***                                                                     ***
C***  SECONDARY ENTRY POINT CALLED TO STORE DATA INTO THE D/A FILE.       ***
C***                                                                     ***
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
C
      ENTRY DADO (STANUM,TYPE,BY,BMO,BD,BH,BMN,EY,EMO,ED,EH,EMN,DATA,
     +            RDPDY,STRIP,PRTMSG)
C
      IF (.NOT. OPEN)    GO TO 580
         ASSIGN 1060 TO BRANCH
         IF (NOENT .LT. MAXENT)    GO TO 600
            WRITE (PRINTR,9099)
            RTCODE=-1
            RETURN
C
 1060 LTENT=NOENT
      NXENT=LTENT+1
      UPDATE=.FALSE.
C WRITE INDEX TO THE DATA BASE BEFORE STORING NEW DATA IN THE FILE IF
C THE INDEX WAS SORTED IN CORE FOR PRINTOUT PURPOSES
      IF (.NOT.SORTED) GO TO 1065
      SORTED=.FALSE.
      PTIME(NEXENT)=0
      IIND=1
      NIND=NENTRK
      DO 1064 J=1,NINTRK
      WRITE (TDDATA,REC=J) (STATNO(I),BTIME(I),ETIME(I),RPERD(I),DTYPE(I
     1 ),(ERFLAG(K,I),K=1,6),DUMMY(I),PTIME(I),TRKNO(I),BYTNO(I),I=IIND,
     2 NIND)
      IF (NIND.EQ.MAXENT) GO TO 1065
      IIND=IIND+NENTRK
      NIND=NIND+NENTRK
      IF (NIND.GT.MAXENT) NIND=MAXENT
 1064 CONTINUE
 1065 IF (LTENT .EQ. 0)    GO TO 1120
C
C***  CHECK FOR UPDATE PROCESSING:
C
      IF ((BTIME(LTENT) .NE. BT) .OR. (ETIME(LTENT) .NE. ET) .OR.
     +    (PTIME(LTENT) .NE. BT))    GO TO 1090
C
         UPDATE=.TRUE.
         NXENT=LTENT
         NBYTE=TRKNO(NXENT)*LENTRK+BYTNO(NXENT)
         LBYTE=0
```

```
         LTENT=0                                                        DD7080
         DO 1070 I=1,MAXENT                                             DD7090
            IBYTE=TRKNO(I)*LENTRK+BYTNO(I)                              DD7100
            IF (IBYTE .EQ. 0)    GO TO 1080                             DD7110
            IF ((IBYTE .GE. NBYTE) .OR. (IBYTE .LE. LBYTE))   GO TO 1070 DD7120
               LTENT=I                                                  DD7130
               LBYTE=IBYTE                                              DD7140
 1070    CONTINUE                                                       DD7150
 1080    IF (LTENT .EQ. 0)    GO TO 1120                                DD7160
            GO TO 1130                                                  DD7170
C                                                                       DD7180
C***  CHECK FOR PREVIOUSLY STORED DATA FOR GIVEN TIME SPAN.             DD7190
C                                                                       DD7200
 1090 DO 1100 I=1,NOENT                                                 DD7210
         IF (STANUM .NE. STATNO(I))    GO TO 1100                       DD7220
         IF (TYPE .NE. DTYPE(I))    GO TO 1100                          DD7230
         IF ((ET .LT. BTIME(I)) .OR. (BT .GT. ETIME(I)))   GO TO 1100   DD7240
            IF (PRTMSG)    WRITE (PRINTR,9222)                          DD7250
            RTCODE=8                                                    DD7260
            GO TO 1310                                                  DD7270
 1100 CONTINUE                                                          DD7280
C                                                                       DD7290
C***  FIND THE INDEX ENTRY (LTENT) WHICH WAS STORED LAST ON THE FILE -  DD7300
C***  AND THUS FIND THE LAST TRACK USED AND THE NUMBER OF BYTES USED.   DD7310
C                                                                       DD7320
      IF (LTENT .EQ. 1)    GO TO 1130                                   DD7330
         LTENT=1                                                        DD7340
         LBYTE=TRKNO(1)*LENTRK+BYTNO(1)                                 DD7350
         DO 1110 I=2,NOENT                                              DD7360
            IBYTE=TRKNO(I)*LENTRK+BYTNO(I)                              DD7370
            IF (IBYTE .LT. LBYTE)    GO TO 1110                         DD7380
               LTENT=I                                                  DD7390
               LBYTE=IBYTE                                              DD7400
 1110    CONTINUE                                                       DD7410
         GO TO 1130                                                     DD7420
C                                                                       DD7430
C***  NO DATA SETS ARE CURRENTLY STORED - THUS, THIS IS FIRST ENTRY.    DD7440
C                                                                       DD7450
 1120 TRKNO(1)=NINTRK+1                                                 DD7460
      BYTNO(1)=1                                                        DD7470
      ITRKNO=NINTRK+1                                                   DD7480
      LELE=0                                                            DD7490
C                                                                       DD7500
C***  CHECK FOR TIME SPAN GREATER THAN DEFINED LIMIT.                   DD7510
C                                                                       DD7520
 1130 IF (.NOT.UPDATE.AND.NEXENT.GT.NOENT) PTIME(NEXENT)=0
      CALL DECODE (BT,IBY,IBMO,IBD,IBH,IBM,BJDAYN,IBELT)
      CALL DECODE (ET,IEY,IEMO,IED,IEH,IEM,EJDAYN,IEELT)                DD7540
      IF (IBY .LT. IEY)    EJDAYN=EJDAYN+365+(4-(IBY-IBY/4*4))/4        DD7550
      DYLMT=DAYLMT/NSTEP                                                DD7560
      IF ((EJDAYN-BJDAYN+1) .LE. DYLMT)    GO TO 1140                   DD7570
         IF (PRTMSG)    WRITE (PRINTR,9233) DYLMT                       DD7580
         RTCODE=9                                                       DD7590
         GO TO 1310                                                     DD7600
C                                                                       DD7610
C***  CALCULATE PERTINENT PARAMETERS FOR THE 'LTENT' INDEX ENTRY.       DD7620
C                                                                       DD7630
 1140 IF (LTENT .EQ. 0)    GO TO 1170                                   DD7640
         LSTEP=1
         IF (DTYPE(LTENT) .EQ. QQ)    LSTEP=2                           DD7660
         IF (DTYPE(LTENT) .EQ. AA)    LSTEP=2                           DD7660
         IF (DTYPE(LTENT) .EQ. BB)    LSTEP=2                           DD7660
         DO 1142 II=1,NOA2TP
            IF(TYPE.NE.A2TYPS(II)) GO TO 1142
            LSTEP=2
            GO TO 1143
 1142    CONTINUE
         IF (DTYPE(LTENT) .EQ. A2)    LSTEP=2                           DD7670
         IF (DTYPE(LTENT) .EQ. A4)    LSTEP=4                           DD7680
 1143    CALL DECODE (BTIME(LTENT),IBY,IBMO,IBD,IBH,IBM,BJDAYN,IBELT)   DD7690
         CALL DECODE (ETIME(LTENT),IEY,IEMO,IED,IEH,IEM,EJDAYN,IEELT)   DD7700
         IF (IBY .LT. IEY)    IEELT=IEELT+ITIMEY(IBY)                   DD7710
```

```
          IF (.NOT.UPDATE) GO TO 1145
          ITRKNO=TRKNO(NXENT)
          LELE=BYTNO(NXENT)/2
          GO TO 1165
C                                                                       DD7720
C***  FIND TRACK AND BYTE NUMBER WHERE DATA IS TO BE STORED.            DD7730
C                                                                       DD7740
 1145 KELE=((IEELT-IBELT)/RDINT(LTENT)+1)*LSTEP+BYTNO(LTENT)/2          DD7750
      TRKNO(NXENT)=TRKNO(LTENT)+KELE/NELE                               DD7760
      LELE=MOD(KELE,NELE)                                               DD7770
      IMOD=MOD((NELE-LELE),NSTEP)                                       DD7780
      IF (IMOD .EQ.0)   GO TO 1160                                      DD7790
         KELE=((EELT-BELT)/IRDINT+1)*NSTEP+LELE                         DD7800
         IF (KELE .LE. NELE)   GO TO 1160                               DD7810
            LL=LELE+1
            N=LL+IMOD-1
            DO 1150 I=LL,N
 1150       DATAI2(I)=BLK2
            LELE=LELE+IMOD                                              DD7850
            IF ((LELE+NSTEP) .LE. NELE)   GO TO 1160                    DD7860
               TRKNO(NXENT)=TRKNO(NXENT)+1                              DD7870
               LELE=0                                                   DD7880
C                                                                       DD7890
 1160 BYTNO(NXENT)=LELE*2+1                                             DD7900
      ITRKNO=TRKNO(NXENT)                                               DD7910
      IF ((LELE .EQ.0) .OR. (ITRKNO .EQ. LTRKNO))   GO TO 1170          DD7920
 1165    READ (TDDATA,REC=ITRKNO) DATAI4
         NEXREC=ITRKNO+1                                                JL0315
C                                                                       DD7950
C***  PLACE DATA ON FILE                                                DD7960
C                                                                       DD7970
 1170 IELE=LELE+1                                                       DD7980
      IOFF=(RDPDY-(BH*60+BMN)/IRDINT)+1                                 DD7990
      KELE=((EELT-BELT)/IRDINT+1)*NSTEP+LELE                            DD8000
      J=1                                                               DD8010
      IERVAL=0                                                          DD8020
      ERROR=.FALSE.                                                     DD8030
      ISUB=1                                                            DD8040
      IEXP=0                                                            DD8050
 1180 JELE=KELE                                                         DD8060
      IF (JELE .GT. NELE)   JELE=NELE                                   DD8070
C                                                                       DD8080
C***  PROCESS DATA ACCORDING TO DATA TYPE                               DD8090
C                                                                       DD8100
      IF (TYPE .EQ. ZZ)   GO TO 1190                                    DD8110
      IF (TYPE .EQ. QQ .OR. TYPE .EQ. AA .OR. TYPE .EQ. BB)   GO TO 1220
      IF (TYPE .EQ. A2)   GO TO 1240                                    DD8130
      DO 1185 II=1,NOA2TP
         IF(TYPE .EQ. A2TYPS(II)) GO TO 1240
 1185 CONTINUE
      IF (TYPE .EQ. A4)   GO TO 1260                                    DD8140
      GO TO 1215
C                                                                       DD8150
C***  I N T E G E R   * 2   S T A G E   D A T A   ***                   DD8160
C                                                                       DD8170
 1190 DO 1210 I=IELE,JELE                                               DD8180
         DT=DATA(J)                                                     DD8190
         IF (DT .LT. -9999)   ERROR=.TRUE.                              DD8200
         IF ((DT .GT. 9999) .AND. (DT .LT. 20000))   ERROR=.TRUE.       DD8210
         DATAI2(I)=DT                                                   DD8220
         IF (J .NE. IOFF)   GO TO 1200                                  DD8230
            IOFF=IOFF+RDPDY                                             DD8240
            IF (ERROR)   IERVAL=IERVAL+2**IEXP                          DD8250
            ERROR=.FALSE.                                               DD8260
            IEXP=IEXP+1                                                 DD8270
            IF (IEXP .LT. 15)   GO TO 1200                              DD8280
               ERFLAG(ISUB,NXENT)=IERVAL                                DD8290
               ISUB=ISUB+1                                              DD8300
               IEXP=0                                                   DD8310
               IERVAL=0                                                 DD8320
 1200       J=J+1                                                       DD8330
 1210 CONTINUE                                                          DD8340
```

```
      GO TO 1280                                                        DD8350
C
C***  I N T E G E R   * 2   D A T A  ***
C
 1215 DO 1216 I=IELE,JELE
         DATAI2(I)=DATA(J)
         J=J+1
 1216 CONTINUE
      GO TO 1280
C                                                                       DD8360
C***  I N T E G E R   * 4   D A T A  ***                                DD8370
C                                                                       DD8380
 1220 DO 1230 I=IELE,JELE                                               DD8390
         DATAI2(I)=DATA(J)                                              DD8400
         J=J+1                                                          DD8410
 1230 CONTINUE                                                          DD8420
      GO TO 1280                                                        DD8430
C                                                                       DD8440
C***  A S S O C I A T E D   * 2   D A T A  ***                          DD8450
C                                                                       DD8460
 1240 DO 1250 I=IELE,JELE                                               DD8470
         DATAI2(I)=DATA(J)                                              DD8480
         J=J+1                                                          DD8490
 1250 CONTINUE                                                          DD8500
      GO TO 1280                                                        DD8510
C                                                                       DD8520
C***  A S S O C I A T E D   * 4   D A T A  ***                          DD8530
C                                                                       DD8540
 1260 DO 1270 I=IELE,JELE                                               DD8550
         DATAI2(I)=DATA(J)                                              DD8560
         J=J+1                                                          DD8570
 1270 CONTINUE                                                          DD8580
C                                                                       DD8590
C***  CHECK IF NEXT TRACK IS TO BE USED WITH THIS DATA SET.             DD8600
C                                                                       DD8610
 1280 IF (JELE .EQ. NELE)   GO TO 1290                                  DD8620
      IF (UPDATE)   GO TO 1290                                          DD8630
         LL=JELE+1
         DO 1285 I=LL,NELE
 1285    DATAI2(I)=BLK2
C                                                                       DD8670
C***  WRITE TRACK WITH NEW DATA BACK ONTO FILE.                         DD8680
C                                                                       DD8690
 1290 WRITE (TDDATA,REC=ITRKNO) DATAI4
      NEXREC=ITRKNO+1                                                   JL0315
      IF (JELE .LT. NELE)   GO TO 1300                                  DD8720
         KELE=KELE-NELE                                                 DD8730
         IF (KELE .EQ. 0)   GO TO 1300                                  DD8740
            IELE=1                                                      DD8750
C           INQUIRE (UNIT=TDDATA,NEXTREC=NEXREC)                         PRIME
            ITRKNO=NEXREC                                               DD8760
            IF (.NOT. UPDATE)   GO TO 1180                              DD8770
               IF (KELE .GE. NELE)   GO TO 1180                         DD8780
                  READ (TDDATA,REC=ITRKNO) DATAI4
                  NEXREC=ITRKNO+1                                       JL0315
                  GO TO 1180                                            DD8810
C                                                                       DD8820
C***  DATA STORAGE COMPLETED - ADD NEW INDEX ENTRY.                     DD8830
C                                                                       DD8840
 1300 LTRKNO=ITRKNO                                                     DD8850
      IF (ERROR) IERVAL=IERVAL+2**IEXP                                  DD8860
      ERFLAG(ISUB,NXENT)=IERVAL                                         DD8870
      STATNO(NXENT)=STANUM                                              DD8880
      BTIME(NXENT)=BT                                                   DD8890
      ETIME(NXENT)=ET                                                   DD8900
      PTIME(NXENT)=PDATE                                                DD8910
      RPERD(NXENT)=RDPDY                                                DD8920
      RDINT(NXENT)=IRDINT                                               DD8930
      DTYPE(NXENT)=TYPE                                                 DD8940
      DUMMY(NXENT)=BLK                                                  DD8950
      J=(NXENT-1)/NENTRK+1                                              DD8960
      IIND=(J-1)*NENTRK+1                                               DD8970
```

```
      NIND=J*NENTRK                                                     DD8980
      WRITE (TDDATA,REC=J) (STATNO(I),BTIME(I),ETIME(I),RPERD(I),DTYPE(I
     +                ),(ERFLAG(K,I),K=1,6),DUMMY(I),PTIME(I),TRKNO(I),  DD9000
     +                    BYTNO(I),I=IIND,NIND)                          DD9010
      NOENT=NOENT+1                                                     DD9020
      RETURN                                                            DD9030
C                                                                       DD9040
C***  ERROR : PRINT ARGUMENT LIST IF ERROR RTCODE ENCOUNTERED.          DD9050
C                                                                       DD9060
 1310 IF (PRTMSG) WRITE (PRINTR,9244) STANUM,BY,BMO,BD,BH,BMN,EY,EMO,ED, DD9070
     +                                EH,EMN,RDPDY,TYPE                 DD9080
      RETURN                                                            DD9090
C                                                                       DD9100
C***  OUTPUT FORMATS ---                                                DD9110
C                                                                       DD9120
 9000 FORMAT ('1'//T38,'DIRECTORY OF TIME-DEPENDENT DATA SETS IN DIRECT DD9130
     1ACCESS FILE')                                                     DD9140
 9011 FORMAT ('0',T8,116('=')/' ',T8,' DATA |   FIELD    | START OF ENTR DD9150
     1Y |  END OF ENTRY  | PROCESSING TIME | RDS | DATA | TYPE | FILE AD DD9160
     2DRESS |  DATA'/' ',T8,'  SET |  STATION   |',2(16(' '),'|'),17(' ' DD9170
     3),'| PER |  SET | OF  |',14(' '),'|'/' ',T8,'  NO  |  NUMBER   ' DD9180
     4,3('| YR MO DY HR MN '),' | DAY | SIZE | DATA | RECORD  BYTE | FLA JL0814
     5GGED'/' ',T8,116('='))                                            DD9200
 9022 FORMAT (' ',T8,I5,' |',I3,'-',F7.2,3(' |',I3,'/',I2,'/',I2,I3,':', DD9210
     1I2),'  |',I4,' |',I5,' |',A4,' |',I5,' |',I6,' |   ',A4)          DD9220
 9033 FORMAT ('1')                                                      DD9230
 9044 FORMAT (' ',T8,116('='))                                          DD9240
 9055 FORMAT (' ')                                                      DD9250
 9066 FORMAT ('1/////////,T19,'A CALENDAR-YEAR SUMMARY OF THE TRANSIENT DD9260
     1-FLOW, BOUNDARY-VALUE DATA, STORED IN DIRECT ACCESS FILES ')      DD9270
 9077 FORMAT ('0',T44,'PROCESSED BY PROGRAM "DADIO" ; VERSION 85/04/19 ' DD9280
     1/'0',T63,'# # # #'/'0',T55,'PROCESSING DATE ',I2,'/',I2,'/',I2)   DD9290
 9088 FORMAT ('0',T52,30('='),/,'0',T66,A2,/,'0',T02,'CALENDAR ',       DD9300
     1'YEAR  -  19',I2,T111,'STATION NO. ',I2,'-',F7.2/' ',T65,'DAY'/26( JL0814
     2' ',32A4,A3))                                                     JL0814
     2              ' ',T64,'CODES'/' ',T02,'* = PARTIAL RECORD;        JL0814
     3AL RECORD,CONTAINS FLAGGED DATA;  # = DUAL DENSITY RECORD;  $ = CO DD9330
     4MPLETE RECORD,CONTAINS FLAGGED DATA'/' ',T02,'T = 288 READINGS/DAY DD9340
     5;  U = 240 READINGS/DAY;  V = 144 READINGS/DAY;  W = 96 READINGS/D DD9350
     6AY;  X = 48 READINGS/DAY; Y = 24 READINGS/DAY')                   JL0814
 9099 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ TIME-DEPENDENT DATA IND DD9370
     1EX RECORDS ARE FULL; RECONSTRUCT THE DATA FILE $$$$$$$$$$$$$$$$$$$ DD9380
     2$$$$$$$$$$$')                                                     DD9390
 9111 FORMAT (/' $$$$$$$$$$$ MAXIMUM NUMBER OF YEARS (25 YEARS) ALLOWED  DD9400
     1IN THE TIME-DEPENDENT DATA FILE EXCEEDED; CREATE A NEW DATA FILE $ DD9410
     2$$$$$$$$$$$')                                                     DD9420
 9122 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$ TIME-DEPENDENT DATA INDEX RECORDS  DD9430
     1HAVE NOT BEEN READ; CHECK YOUR PROGRAM CALLING SEQUENCE $$$$$$$$$$ DD9440
     2$$$$$$$$$$$')                                                     DD9450
 9133 FORMAT (/' $$$$$$$$$$$$$$$$$$ TIME-DEPENDENT DATA INDEX IS EMPTY; CH DD9460
     1ECK YOUR PROGRAM CALLING SEQUENCE, OR FILE NAMING CONVENTION $$$$$ DD9470
     2$$$$$$$$$$$')                                                     DD9480
 9144 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ REC DD9490
     1ORDING DENSITY IS INVALID $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9500
     2$$$$$$$$$$$')                                                     DD9510
 9155 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ FIEL DD9520
     1D STATION NUMBER IS INVALID $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9530
     2$$$$$$$$$$$')                                                     DD9540
 9166 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DA DD9550
     1TE AND/OR TIME IS ERRANT $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9560
     2$$$$$$$$$$$')                                                     DD9570
 9177 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ SYSTEM SUP DD9580
     1PLIED DATE AND/OR TIME IS ERRANT $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9590
     2$$$$$$$$$$$')                                                     DD9600
 9188 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9610
     1 DATA TYPE IS INVALID $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9620
     2$$$$$$$$$$$')                                                     DD9630
 9199 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ RECORDING DENSITY CO DD9640
     1NFLICTS WITH THAT OF AN INDEX ENTRY (',I3,') $$$$$$$$$$$$$$$$$$$$$$ DD9650
     2$$$$$$$$$$$$$$')                                                  DD9660
 9211 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ NO DATA AV DD9670
     1AILABLE FOR TIME SPAN OF REQUEST $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9680
```

```
     2$$$$$$$$$$$')                                                     DD9690
 9222 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DATA PR DD9700
     1EVIOUSLY STORED FOR TIME SPAN $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ DD9710
     2$$$$$$$$$$$')                                                     DD9720
 9233 FORMAT (/' $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ MAXIMUM RECORD  DD9730
     +LENGTH (',I2,' DAYS) EXCEEEDED BY TIME SPAN $$$$$$$$$$$$$$$$$$$$
```

```
      SUBROUTINE SORTIN                                                 ST 100
C                                                                       ST 110
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***ST 120
C***                                                                  ***ST 130
C***  ROUTINE TO SORT THE TIME-DEPENDENT DIRECT ACCESS DATA FILE      ***ST 140
C***  INDEX IN THE COMMON BLOCK /INDEX/.                              ***ST 150
C***                                                                  ***ST 160
C*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***ST 170
C                                                                       ST 180
      COMMON    /INDEX/ NOENT,STATNO(930),BTIME(930),ETIME(930),        ST 190
     +                  PTIME(930),DUMMY(930),RPERD(930),RDINT(930),    ST 200
     +                  DTYPE(930),ERFLAG(6,930),TRKNO(930),BYTNO(930), ST 210
     +                  PASS(930)                                       ST 220
      CHARACTER*4 DUMMY,DUM
      INTEGER   NOENT,STATNO,BTIME,ETIME,PTIME                          ST 230
      INTEGER*2 RPERD,RDINT,ERFLAG,TRKNO,BYTNO,NOTYPS/91/
C      LOGICAL*2 OR *1 NOT STANDARD FORTRAN '77, CHANGED TO *4
      LOGICAL PASS,PSI
      INTEGER   STI,BTI,ETI,PTI,IP1
      INTEGER*2 RPD,RDI,ERR,TRK,BYT
      LOGICAL   SORTED                                                  ST 290
      CHARACTER*2      DTYPE,DTI,
     + TYPES(91)/' A',' B',' C',' L',' P',' Q',' S',' T',' U',' V',' Z',
     +           'AP','AT','AV','AU','GU','GV','PH','RH','WP','XU','XV',
     +               'C0','C1','C2','C3','C4','C5','C6','C7','C8','C9',
     +               'K0','K1','K2','K3','K4','K5','K6','K7','K8','K9',
     +               'L0','L1','L2','L3','L4','L5','L6','L7','L8','L9',
     +               'R0','R1','R2','R3','R4','R5','R6','R7','R8','R9',
     +               'M0','M1','M2','M3','M4','M5','M6','M7','M8','M9',
     +               'TA','PA','QL','QT','RA','RS',
     +               'T0','T1','T2','T3','T4','T5','T6','T7','T8','T9',
     +               'WD','WS',' M'/
C                                                                       ST 310
C***  INITIALIZATIONS ---                                               ST 320
C                                                                       ST 330
      IF (NOENT .LT. 2) RETURN                                          ST 331
      N=NOENT-1                                                         ST 340
      K=N                                                               ST 350
      SORTED=.FALSE.                                                    ST 360
C                                                                       ST 370
C***  SORTING ORDER:  STATION NUMBER - DATA TYPE - BEGIN TIME           ST 380
C                                                                       ST 390
      DO 180 L=1,N                                                      ST 400
         IF (SORTED)   RETURN                                           ST 410
         SORTED=.TRUE.                                                  ST 420
         DO 170 I=1,K                                                   ST 430
            IP1=I+1                                                     ST 440
C                                                                       ST 450
C***  SORT ON STATION NUMBER ---                                        ST 460
C                                                                       ST 470
            IF (STATNO(IP1)-STATNO(I))   100,140,170                    ST 480
C                                                                       ST 490
C***  SWITCH THE ORDER OF I AND IP1 INDEX ENTRIES.                      ST 500
C                                                                       ST 510
  100       STI=STATNO(I)                                               ST 520
            STATNO(I)=STATNO(IP1)                                       ST 530
            STATNO(IP1)=STI                                             ST 540
  110       DTI=DTYPE(I)                                                ST 550
            DTYPE(I)=DTYPE(IP1)                                         ST 560
            DTYPE(IP1)=DTI                                              ST 570
  120       BTI=BTIME(I)                                                ST 580
            BTIME(I)=BTIME(IP1)                                         ST 590
            BTIME(IP1)=BTI                                              ST 600
            ETI=ETIME(I)                                                ST 610
            ETIME(I)=ETIME(IP1)                                         ST 620
            ETIME(IP1)=ETI                                              ST 630
            PTI=PTIME(I)                                                ST 640
            PTIME(I)=PTIME(IP1)                                         ST 650
            PTIME(IP1)=PTI                                              ST 660
            DUM=DUMMY(I)                                                ST 670
            DUMMY(I)=DUMMY(IP1)                                         ST 680
            DUMMY(IP1)=DUM                                              ST 690
```

```
            RPD=RPERD(I)                                                ST 700
            RPERD(I)=RPERD(IP1)                                         ST 710
            RPERD(IP1)=RPD                                              ST 720
            RDI=RDINT(I)                                                ST 730
            RDINT(I)=RDINT(IP1)                                         ST 740
            RDINT(IP1)=RDI                                              ST 750
            TRK=TRKNO(I)                                                ST 760
            TRKNO(I)=TRKNO(IP1)                                         ST 770
            TRKNO(IP1)=TRK                                              ST 780
            BYT=BYTNO(I)                                                ST 790
            BYTNO(I)=BYTNO(IP1)                                         ST 800
            BYTNO(IP1)=BYT                                              ST 810
            PSI=PASS(I)                                                 ST 820
            PASS(I)=PASS(IP1)                                           ST 830
            PASS(IP1)=PSI                                               ST 840
            DO 130 J=1,6                                                ST 850
               ERR=ERFLAG(J,I)                                          ST 860
               ERFLAG(J,I)=ERFLAG(J,IP1)                                ST 870
               ERFLAG(J,IP1)=ERR                                        ST 880
  130       CONTINUE                                                    ST 890
            SORTED=.FALSE.                                              ST 900
            GO TO 170                                                   ST 910
C                                                                       ST 920
C***  EQUAL STATION NUMBERS - SORT ON DATA TYPE ---                     ST 930
C                                                                       ST 940
  140       IF (DTYPE(I) .EQ. DTYPE(IP1))   GO TO 160                   ST 950
               DO 150 M=1,NOTYPS                                        ST 960
                  IF (DTYPE(I) .EQ. TYPES(M))   GO TO 170               ST 970
                  IF (DTYPE(IP1) .EQ. TYPES(M))   GO TO 110             ST 980
  150          CONTINUE                                                ST 990
C                                                                       ST1000
C***  EQUAL DATA TYPES - SORT ON BEGIN TIME ---                         ST1010
C                                                                       ST1020
  160       IF (BTIME(IP1) .LT. BTIME(I))   GO TO 120                   ST1030
C                                                                       ST1040
C***  COMPARE NEXT TWO ENTRIES -                                        ST1050
C                                                                       ST1060
  170    CONTINUE                                                       ST1070
         K=K-1                                                          ST1080
  180 CONTINUE                                                          ST1090
C                                                                       ST1100
C***  SORT COMPLETE ---                                                 ST1110
C                                                                       ST1120
      RETURN                                                            ST1130
      END                                                               ST1140-
C
C       THIS SUBROUTINE IS USED ON PRIME COMPUTERS TO REPLACE THE AMDAHL
C       SYSTEM CALLS TO JULDAT AND TIME, WHICH RETRIEVE THE SYSTEM DATE
C       AND TIME.
C
      SUBROUTINE JULDAT (YR,MO,DA)
      INTEGER*2 STRING(28),NUM/28/
      INTEGER*4 YR,MO,DA,SEC
      CHARACTER*80 LIDO
      CALL TIMDAT(STRING,NUM)
      WRITE(LIDO,10) STRING(1),STRING(2),STRING(3)
      READ(LIDO,20) MO,DA,YR
   10 FORMAT (3A4)
   20 FORMAT (3I4)
      RETURN
      ENTRY TIME(SEC)
      CALL TIMDAT(STRING,NUM)
      SEC=(STRING(4)*60+STRING(5))*100
      RETURN
      END
C
C- COMBINATION OF USGS SUBROUTINES JULDAT AND TIME
C- RETURNS CURRENT DATE AND TIME TO INTEGER ARRAYS
C
C      JDATE - CURRENT DATE
C            - (1) -- MONTH
C            - (2) -- DAY
```

```
C            - (3) -- LAST 2 DIGITS OF YEAR
C       JTIME - CURRENT MILITARY TIME
C
C- KF   10/83
C
      SUBROUTINE DATTIM( JDATE, JTIME )
      INTEGER    JDATE(3)
      INTEGER *2 STRING(28), NUM / 28 /
      CHARACTER*6 IMAGE
      CALL TIMDAT( STRING, NUM )
      WRITE(IMAGE,1) ( STRING(J), J = 1,3 )
      READ(IMAGE,2) (JDATE(J), J = 1,3 )
      JTIME = ( STRING(4) / 60 ) * 100 + MOD( STRING(4), 60 )
    1 FORMAT( 3A2 )
    2 FORMAT( 3I2 )
      RETURN
      END
      SUBROUTINE MOVE
      LOGICAL*1  L(N),M
      INTEGER*2  J(N),KK
      REAL       A(N),B
C
      ENTRY MOVE1(M,L,N)
   10 DO 15 I=1,N
   15 L(I)=M
      RETURN
C
      ENTRY MOVE2(KK,J,N)
   20 DO 25 I=1,N
   25 J(I)=KK
      RETURN
C
      ENTRY MOVE4(B,A,N)
   40 DO 45 I=1,N
   45 A(I)=B
      RETURN
      END
```

APPENDIX C

CON.F77

Conservative Constituents Decay Coefficient Subroutine

```
C
C*****CALCULATE THE CONSTITUENT COEFFICIENTS*****************************
C
      SUBROUTINE FINK(CR,S,XK)
C
C     DEFINE VARIABLES:
C
C      NAME                              PURPOSE
C     ------  ------------------------------------------------------------
C      A(G)   AVERAGE AREA OF THE REACH DOWNSTREAM FROM GRID G [SQ M]
C
C     CR(E,C) REFERENCE (EQUILIBRIUM CONCENTRATION IN EQUATION E FOR
C             CONSTITUENT C
C
C       J     TIME STEP
C
C       KO    PARCEL NUMBER
C
C       MX    UPSTREAM GRID NUMBER
C
C    PDC(C,P) CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C             DUE TO DECAY
C
C     PT(C,P) CONCENTRATION ON CONSTITUENT C IN PARCEL P
C
C      S(C)   SOURCE FLUX OF CONSTITUENT C [UNITS PER HOUR]
C
C      U(G)   AVERAGE VELOCITY IN THE REACH DOWNSTREAM FROM GRID G
C             [METERS PER HOUR]
C
C      W(G)   AVERAGE TOP WIDTH OF THE REACH DOWNSTREAM FROM GRID G [M]
C
C    XK(C,CC) EXCHANGE COEFFICIENT FOR CONSTITUENT C DUE TO THE
C             CONCENTRATION OF CONSTITUENT CC
C
      COMMON /CDECAY/ KO,MX,PT,PDC
      COMMON /CFINK/      A,U,W,TIME,HI,J
      REAL A(50),CR(10,10),PDC(10,500),PT(10,500),S(10),U(50),W(50),
     1     XK(10,10)
C
C      INSERT USER DEFINED SUBROUTINE FINK HERE
C
C
C     USE THE FOLLOWING 13 STATEMENTS FOR
C     FIRST ORDER DECAY OF UP TO TEN INDEPENDENT CONSTITUENTS
C     I.E. DC(I)/DT = XK(I,I)*C(I), I = 1 TO NEQ
C
C      LOGICAL READ/.FALSE./
C      REAL X(10)
C      SAVE X
C      IF (.NOT. READ) THEN
C       OPEN(UNIT=33,FILE='FOI.INPUT')
C       READ(33,3300) (X(I),I=1,10)
C  33   FORMAT(10X,10F7.0)
C       CLOSE(UNIT=33)
C       READ=.TRUE.
C      END IF
C      DO 100 I=1,10
C       XK(I,I)=X(I)
C 100  CONTINUE
C
C      END OF USER DEFINED SUBROUTINE FINK, RETURN TO DECAY
C
      RETURN
      END
C
C     INSERT USER-DEFINED SUBROUTINES HERE
C
```

APPENDIX D

OXYGEN.F77

Oxygen Modeling Decay Coeficient Subroutine

```
C
C*****CALCULATE THE CONSTITUENT COEFFICIENTS*****************************
C
      SUBROUTINE FINK(CR,S,XK)
C
C     DEFINE VARIABLES:
C
C      NAME                          PURPOSE
C      ------  ------------------------------------------------------------
C      A(G)    AVERAGE AREA OF THE REACH DOWNSTREAM FROM GRID G [SQ M]
C
C       A1     CONSTANT TERM IN WIND FUNCTION [MM PER DAY PER KPA], GIVES
C              EVAPORATION IN MILLIMETERS PER DAY PER KILOPASCAL OF
C              VAPOR GRADIENT
C
C       AL     LATENT HEAT OF VAPORIZATION [CAL PER GRAM]
C
C       B1     MASS TRANSFER COEFFICIENT IN WIND FUNCTION [MM/DAY PER
C              M/SEC PER KPA], GIVES EVAPORATION IN MILLIMETERS PER DAY
C              PER M/SEC OF WINDSPEED AND KILOPASCAL OF VAPOR GRADIENT
C
C      CK1     TEMPERATURE DEPENDENT BOD DECAY RATE, [1/DAY]
C
C      CK2     TEMPERATURE DEPENDENT REAERATION RATE, [1/DAY]
C
C      CK4     BENTHOS SOURCE RATE FOR BOD [MG/DAY/FT]
C
C      CPR     PRODUCT OF SPECIFIC HEAT OF WATER AND DENSITY OF WATER,
C              EQUAL TO 100 [CAL PER M PER SQ CM PER DEGREE CELSIUS]
C
C     CR(E,C)  REFERENCE (EQUILIBRIUM CONCENTRATION IN EQUATION E FOR
C              CONSTITUENT C
C
C      DFT     SLOPE OF VAPOR PRESSURE CURVE [KPA PER DEGREE CELSIUS]
C
C       HI     INITIAL HOUR OF THE SIMULATION
C
C       I      COUNTER
C
C      INX     NUMBER OF REACHES IN SIMULATION
C
C       J      TIME STEP
C
C       K      COUNTER
C
C       KO     PARCEL NUMBER
C
C       MX     UPSTREAM GRID NUMBER
C
C    PDC(C,P)  CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C              DUE TO DECAY
C
C      PSI     WIND FUNCTION [CM PER HOUR PER KILOPASCAL]
C
C     PT(C,P)  CONCENTRATION OF CONSTITUENT C IN PARCEL P
C
C      READ    LOGICAL VARIABLE, TRUE IF INPUT HAS BEEN READ FOR THE
C              SIMULATION
C
C      S(C)    SOURCE FLUX OF CONSTITUENT C [UNITS PER HOUR]
C
C      SIG     STEFAN-BOLTZMAN CONSTANT [CAL PER SQ CM PER HOUR PER
C              (DEGREES KELVIN)**4]
C
C       TA     AIR TEMPERATURE [CELCIUS]
C
C      TAB     ABSOLUTE PARCEL TEMPERATURE [DEGREES KELVIN]
C
C      TCK1    TEMPERATURE ADJUSTED CK1, CARBONACEOUS BOD DECAY RATE
C              [1/DAY]
C
C      TIME    ELAPSED TIME OF SIMULATION [HOURS]
```

```
C
C       U(G)    AVERAGE VELOCITY IN THE REACH DOWNSTREAM FROM GRID G
C               [METERS PER HOUR]
C
C        V      WIND SPEED [METERS PER SECOND]
C
C       W(G)    AVERAGE TOP WIDTH OF THE REACH DOWNSTREAM FROM GRID G [M]
C
C     XK(C,CC)  EXCHANGE COEFFICIENT FOR CONSTITUENT C DUE TO THE
C               CONCENTRATION OF CONSTITUENT CC
C
C       XKX     SURFACE EXCHANGE COEFFICIENT [CAL PER SQ CM PER HOUR
C               PER DEGREE CELCIUS]
C
      COMMON /CDECAY/ KO,MX,PT,PDC
      COMMON /CFINK/ A,U,W,TIME,HI,J
      REAL A(50),CR(10,10),PDC(10,500),PT(10,500),S(10),U(50),W(50),
     1     XK(10,10)
C
C      INSERT USER DEFINED SUBROUTINE FINK HERE
C
      SAVE CK1,CK2,CK4,A1,B1
      REAL TRASH(50,2)
      INTEGER JOLD/0/
      LOGICAL READ/.FALSE./
C
C     READ OXYGEN.INPUT
C
      IF (.NOT. READ) THEN
       OPEN(UNIT=33,FILE='OXYGEN.INPUT')
       READ(33,3300) CK1,CK2,CK4,A1,B1
 3300  FORMAT(10X,5F7.0)
       READ=.TRUE.
      END IF
C
C     READ METEOROLOGICAL CONDITIONS
C
      IF (J .GT. JOLD) THEN
C     USE THE NEXT CARD IF THE TDDS IS USED
       CALL TDDSBC(' ',6,J,TRASH,0.0,TA,V,0.0,0.0,0.0)
C     USE THE NEXT CARD IF THE TDDS IS NOT USED
C      READ(33,3300) TA,V
      END IF
C
C     DYE (CONSERVATIVE)
C
C
C     TEMPERATURE (SIMPLIFIED TEMPERATURE ALGORITHM)
C
      CPR= 100.0
      SIG= 1.171E-7/24.0
      AL= 595.9-0.545*PT(2,KO)
      PSI= (A1+B1*V)/(24.0*10.0)
      TAB= PT(2,KO)+273.16
      DFT=1.1532E11*EXP(-4271.1/(PT(2,KO)+242.63))
     1              /((PT(2,KO)+242.63)**2)
      XKX= 4.0*0.97*SIG*(TAB**3)+AL*PSI*(DFT+0.06)
      XK(2,2)= -XKX*W(MX)/(A(MX)*CPR)
      CR(2,2)=TA
C
C     BOD
C
      TCK1=T(PT(2,KO),1.047,CK1)
      XK(3,3)=-TCK1/24.0
C
C     DISSOLVED OXYGEN
C
      XK(4,4)=-T(PT(2,KO),1.0159,CK2)/24.0
      CR(4,4)=468.0/(PT(2,KO)+31.6)
      XK(4,3)=-TCK1/24.0
      S(4)=-CK4/(24.0*28.316*A(MX))
C
```

```
C     RETURN
C
      JOLD=J
      RETURN
      END
C
C     INSERT USER-DEFINED SUBROUTINES HERE
C
C
C     TEMPERATURE CORRECTION FUNCTION
C
      FUNCTION T(PT,THETA,X)
      T=X*THETA**(PT-20.0)
      RETURN
      END
```

APPENDIX E

QUAL2.F77

QUAL II Reaction Kinetics Decay Coeficient Subroutine

```
C
C*****CALCULATE THE CONSTITUENT COEFFICIENTS*****************************
C
      SUBROUTINE FINK(CR,S,XK)
C
C     DEFINE VARIABLES:
C
C      NAME                              PURPOSE
C     ------  ------------------------------------------------------------
C      A(G)   AVERAGE AREA OF THE REACH DOWNSTREAM FROM GRID G [SQ M]
C
C     CR(E,C) REFERENCE (EQUILIBRIUM CONCENTRATION IN EQUATION E FOR
C             CONSTITUENT C
C
C        J    TIME STEP
C
C       KO    PARCEL NUMBER
C
C       MX    UPSTREAM GRID NUMBER
C
C    PDC(C,P) CHANGE IN CONCENTRATION OF CONSTITUENT C IN PARCEL P
C             DUE TO DECAY
C
C     PT(C,P) CONCENTRATION ON CONSTITUENT C IN PARCEL P
C
C      S(C)   SOURCE FLUX OF CONSTITUENT C [UNITS PER HOUR]
C
C      U(G)   AVERAGE VELOCITY IN THE REACH DOWNSTREAM FROM GRID G
C             [METERS PER HOUR]
C
C      W(G)   AVERAGE TOP WIDTH OF THE REACH DOWNSTREAM FROM GRID G [M]
C
C    XK(C,CC) EXCHANGE COEFFICIENT FOR CONSTITUENT C DUE TO THE
C             CONCENTRATION OF CONSTITUENT CC
C
      COMMON /CDECAY/ KO,MX,PT,PDC
      COMMON /CFINK/      A,U,W,TIME,HI,J
      REAL A(50),CR(10,10),PDC(10,500),PT(10,500),S(10),U(50),W(50),
     1     XK(10,10)
C
C      INSERT USER DEFINED SUBROUTINE FINK HERE
C
C
      LOGICAL READ/.FALSE./
      REAL ALPHAO(50),ALPHA(6),CKNH3(50),CKNO2(50),ALGSET(50),SPHOS(50),
     1     SNH3(50),CK(50,6)
      SAVE ALPHAO,ALPHA,CKNH3,CKNO2,ALGSET,SPHOS,SNH3,CK,GROMAX,RESPRT,
     1     CKN,CKP,CKL,EXCOEF,SONET,PTIME,A1,B1,TA,V
      REAL LAT,LLM,LSM
      SAVE LAT,LLM,LSM,ELEV,JDAT,DAT,AE,BE,DT
      IF (.NOT. READ) THEN
       OPEN(UNIT=33,FILE='QUAL2.INPUT')
       READ(33,3300) INX,DT
 3300  FORMAT(10X,I7,9F7.0)
       READ(33,3305) (ALPHAO(I),I=1,INX)
 3305  FORMAT(10X,10F7.0)
       READ(33,3305) (ALPHA(I),I=1,6),GROMAX,RESPRT
       READ(33,3305) (CKNH3(I),I=1,INX)
       READ(33,3305) (CKNO2(I),I=1,INX)
       READ(33,3305) (ALGSET(I),I=1,INX)
       READ(33,3305) (SPHOS(I),I=1,INX)
       READ(33,3305) (SNH3(I),I=1,INX)
       DO 100 I=1,6
        READ(33,3305) (CK(K,I),K=1,INX)
  100  CONTINUE
       READ(33,3305) CKN,CKP,CKL,EXCOEF
C
C******UTILIZE THE NEXT LINE IF THE QUAL II TEMP ALGORITHM IS USED******
       READ(33,3300) JDAT,LAT,LLM,LSM,ELEV,DAT,AE,BE
C
C****UTILIZE THE NEXT TWO LINES IF THE SIMPLIFIED TEMP ALGORITHM IS USED
C       READ(33,3305) A1,B1
```

```
C        PTIME=HI
C
         READ=.TRUE.
        END IF
C
C       TEMPERATURE (QUAL II ALGORITHM)
C
        S(1)=252.0*HSNET(LAT,LLM,LSM,ELEV,JDAT,DAT,AE,BE,TIME,HI,DT,
     1      PT(1,KO),SONET)*W(MX)/(1000000.0*0.0929*A(MX))
C
C       TEMPERATURE (SIMPLIFIED TEMPERATURE ALGORITHM)
C
C       IF (TIME .GT. PTIME) READ(33,3305) SONET,TA,V
C       PTIME=TIME
C       CPR= 100.0
C       SIG= 1.171E-7/24.0
C       AL= 595.9-0.545*PT(1,KO)
C       PSI= (A1+B1*V)/(24.0*10.0)
C       TAB= PT(1,KO)+273.16
C       DFT=1.1532E11*EXP(-4271.1/(PT(1,KO)+242.63))
C    1                /((PT(1,KO)+242.63)**2)
C       XKX= 4.0*0.97*SIG*(TAB**3)+AL*PSI*(DFT+0.06)
C       XK(1,1)= -XKX*W(MX)/(A(MX)*CPR)
C       CR(1,1)=TA
C
C       ALGAE
C
        DEXCOF=EXCOEF*A(MX)/W(MX)
C
C**UTILIZE THE NEXT THREE LINES IF THE SIMPLIFIED TEMP ALGORITHM IS USED
C        GRO=T(PT(1,KO),1.047,GROMAX)*(PT(5,KO)/(PT(5,KO)+CKN))
C       1            *(PT(6,KO)/(PT(6,KO)+CKP))*(1.0/DEXCOF)
C       2            *LOG((CKL+SONET)/(CKL+SONET*EXP(-DEXCOF)))
C
C*****UTILIZE THE NEXT THREE LINES IF THE QUAL II TEMP ALGORITHM IS USED
        GRO=T(PT(1,KO),1.047,GROMAX)*(PT(5,KO)/(PT(5,KO)+CKN))
     1            *(PT(6,KO)/(PT(6,KO)+CKP))*(1.0/DEXCOF)
     2            *LOG((CKL+SONET/221.4)/(CKL+SONET*EXP(-DEXCOF)/221.4))
C
        TRSPRT=T(PT(1,KO),1.047,RESPRT)
        XK(2,2)=(GRO-TRSPRT-ALGSET(MX)*0.3048*W(MX)/A(MX))/24.0
C
C       AMMONIA NITROGEN
C
        XK(3,2)=ALPHA(1)*TRSPRT/24.0
        XK(3,3)=-T(PT(1,KO),1.047,CKNH3(MX))/24.0
        S(3)=3.28034*SNH3(MX)/(A(MX)*24.0*1000.0)
C
C       NITRITE NITROGEN
C
        XK(4,3)=-XK(3,3)
        XK(4,4)=-T(PT(1,KO),1.047,CKNO2(MX))/24.0
C
C       NITRATE NITROGEN
C
        XK(5,2)=-ALPHA(1)*GRO/24.0
        XK(5,4)=-XK(4,4)
C
C       DISSOLVED OROPHOSPHATE
C
        XK(6,2)=ALPHA(2)*(TRSPRT-GRO)/24.0
        S(6)=3.28034*SPHOS(MX)/(A(MX)*24.0*1000.0)
C
C       BOD
C
        TCK1=T(PT(1,KO),1.047,CK(MX,1))
        XK(7,7)=-(TCK1+CK(MX,3))/24.0
C
C       DISSOLVED OXYGEN
C
        PTF=PT(1,KO)*1.8+32.0
        CR(8,8)=24.89+PTF*(-0.426+PTF*(0.00373-PTF*0.0000133))
```

```
      XK(8,8)=-T(PT(1,KO),1.0159,CK(MX,2))/24.0
      XK(8,7)=-TCK1/24.0
      XK(8,3)=-ALPHA(5)*T(PT(1,KO),1.047,CKNH3(MX))/24.0
      XK(8,2)=(ALPHA(3)*GRO-ALPHA(4)*TRSPRT)/24.0
      XK(8,4)=-ALPHA(6)*T(PT(1,KO),1.047,CKNO2(MX))/24.0
      S(8)=-3.28034*CK(MX,4)/(24.0*1000.0*A(MX))
C
C     COLIFORM
C
      XK(9,9)=-T(PT(1,KO),1.047,CK(MX,5))/24.0
C
C     ARBITRARY NONCONSERVATIVE CONSTITUENT
C
      XK(10,10)=-T(PT(1,KO),1.047,CK(MX,6))/24.0
C
C      END OF USER DEFINED SUBROUTINE FINK, RETURN TO DECAY
C
      RETURN
      END
C
C     INSERT USER-DEFINED SUBROUTINES HERE
C
C
C     TEMPERATURE CORRECTION FUNCTION
C
      FUNCTION T(PT,THETA,X)
      T=X*THETA**(PT-20.0)
      RETURN
      END
C
      FUNCTION HSNET(LAT,LLM,LSM,ELEV,JDAT,DAT,AE,BE,TIME,HI,DT,PTC,
     1               SONET)
C
C                                HSNET COMPUTES THE NET AMOUNT OF HEAT
C                                RADIATION FLUX BEING TRANSFERRED ACROSS
C                                THE AIR-WATER INTERFACE BASED ON AN
C                                ENERGY BUDGET WHICH CONSIDERS SOLAR
C                                RADIATION, ATMOSPHERIC RADIATION, BACK
C                                RADIATION, CONDUCTION, AND EVAPORATION.
C
C
      REAL LLM,LSM,LAT,TRLCD/3.0/,PTIME/999.9/
      INTEGER*2 ISTDY/0/
      SAVE CLOUD,DRYBLB,WETBLB,ATMPR,WIND,SONETS
CCC
CCC NCASI Commentary, HSNET Section A. (QUAL2 Step 1-0)
CCC   A. Compute and/or define required constants.
CCC
      PTF=PTC*1.8+32.0
      IF (ISTDY .EQ. 1) GO TO 36
      IF (ABS(PTIME-TIME) .LT. 0.0000001) GO TO 36
      PTIME=TIME
      PI=3.141628
      CON1=2.0*PI/365.0
      CON2=PI/180.0*LAT
      CON3=180.0/PI
      CON4=23.45*PI/180.0
      CON5=PI/12.0
      CON6=12.0/PI
      DELTSL=(LLM-LSM)/15.0
      SOLCON=438.0
      ELEXP=EXP(-ELEV/2532.0)
      DAYOFY=JDAT
CCC
CCC  NCASI Commentary, HSNET Section B. (QUAL2 Step 2-0)
CCC   B. Begin computations  for calculating the
CCC       net solar radiation term.
CCC
CCC   B.1a Calculate seasonal and daily position
CCC         of the sun relative to the location
CCC         of the basin on the earth's
CCC         surface. (QUAL2 Step 2-1)
```

```
CCC
      REARTH=1.0+0.017*COS(CON1*(186.0-DAYOFY))
      DECLIN=CON4*COS(CON1*(172.0-DAYOFY))
      RR=REARTH**2
      EQTIME=0.000121-0.12319*SIN(CON1*(DAYOFY-1.0)-0.07014)
     *       -0.16549*SIN(2.0*CON1*(DAYOFY-1.0)+0.3088)
      DECLON=ABS(DECLIN)
CC
CC  Replace TAN function with SIN/COS.
CC
      TANA =SIN(CON2)/COS(CON2)
      TANB = SIN(DECLON)/COS(DECLON)
      ACS = TANA*TANB
CC
      IF (ABS(ACS) .LT. 0.0000001) GO TO 8
      X=SQRT(1.0-ACS*ACS)
      X=X/ACS
      ACS=ATAN(X)
      IF (DECLIN.GT.0.0) ACS=PI-ACS
      GO TO 9
    8 ACS=PI/2.0
    9 CONTINUE
CCC
CCC   B.1a Calculate the standard time of
CCC         sunrise (STR) and sunset (STS).
CCC         (QUAL2 Step 2-2)
CCC
      STR=12.0-CON6*ACS+DELTSL
      STS=24.0-STR+2.0*DELTSL
      STB=MOD(TIME-DT,24.001)
      STE=STB+1.0
CCC
CCC   B.3 Test if time to read in local
CCC        climatological data. (QUAL2 Step 2-3)
CCC
      IF(TRLCD .LT. 3.0) GO TO 82
      READ (33,12) CLOUD,DRYBLB,WETBLB,ATMPR,WIND,METRIC,ISTDY
   12 FORMAT (10X,5F7.0,2I7)
      IF(METRIC.EQ.0)GO TO 13
      DRYBLB=DRYBLB*1.8+32.0
      WETBLB=WETBLB*1.8+32.0
      ATMPR=ATMPR*(29.9/1000.)
      WIND=WIND/0.3048
   13 CONTINUE
      WIND=WIND*0.6818
      IF (ISTDY .EQ.1) CLOSE(UNIT=33)
      TRLCD=0.0
CCC
CCC   B.7 Compute vapor pressures (VPWB and
CCC        VPAIR), dew point (DEWPT), AND
CCC        dampening effect of clouds (CNS
CCC        and CNL). (QUAL2 Step 2-4)
CCC
   82 CONTINUE
      TRLCD=TRLCD+DT
      VPWB=0.1001*EXP(0.03*WETBLB)-0.0837
      VPAIR=VPWB-0.000367*ATMPR*(DRYBLB-WETBLB)
     *      *(1.0+(WETBLB-32.0)/1571.0)
      DEWPT=ALOG((VPAIR+0.0837)/0.1001)/0.03
      CS=1.0-0.65*CLOUD**2
      IF (CLOUD.GT.0.9) CS=0.50
      NL=CLOUD*10.0+1.0
      IF (STS.LE.STB.OR.STR.GE.STE) GO TO 35
      IF(STR.GT.STB.AND.STR.LT.STE) GO TO 41
      IF (STS.LT.STE.AND.STS.GT.STB) GO TO 42
CCC
CCC  NCASI Commentary, HSNET Section C. (QUAL2 Step 2-5)
CCC   C. Continue with calculations for solar
CCC       radiation.
CCC
CCC   C.1 Calculate hour angles (TB and TE).
CCC
```

```
      TB=STB-12.0-DELTSL+EQTIME
      TE=STE-12.0-DELTSL+EQTIME
      GO TO 43
   41 TB=STR-12.0-DELTSL+EQTIME
      TE=STE-12.0-DELTSL+EQTIME
      GO TO 43
   42 TB=STB-12.0-DELTSL+EQTIME
      TE=STS-12.0-DELTSL+EQTIME
   43 CONTINUE
      TALT=(TB+TE)/2.0
CCC
CCC   C.2 Compute amount of clear sky, solar
CCC        radiation(SOLAR), and altitude of
CCC        the sun (ALPHA). (QUAL2 Step 2-6)
CCC
      SOLAR=SOLCON/RR*(SIN(CON2)*SIN(DECLIN)*(TE-TB)+CON6*COS(CON2)*
     *      COS(DECLIN)*(SIN(CON5*TE)-SIN(CON5*TB)))
      ALPHA=SIN(CON2)*SIN(DECLIN)+COS(CON2)*COS(DECLIN)*COS(CON5*TALT)
      IF (ABS(ABS(ALPHA)-1.0) .LT. 0.0000001) GO TO 4
      Y=SQRT(1.0-ALPHA*ALPHA)
      Y=ALPHA/Y
      ALPHA=ATAN(Y)
      GO TO 5
    4 IF (ABS(ALPHA+1.0) .LT. 0.0000001) GO TO 6
      ALPHA=PI/2.0
      GO TO 5
    6 ALPHA=-PI/2.0
    5 CONTINUE
      IF (ALPHA.LT.0.01) GO TO 35
CCC
CCC   C.3 Compute absorption and scattering due
CCC        to atmospheric conditions. (QUAL2
CCC        Step 2-7)
CCC
      PWC=0.00614*EXP(0.0489*DEWPT)
      OAM=ELEXP/(SIN(ALPHA)+0.15*(ALPHA*CON3+3.885)**(-1.253))
      A1=EXP(-(0.465+0.0408*PWC)*(0.129+0.171*EXP(-0.880*OAM))*OAM)
      A2=EXP(-(0.465+0.0408*PWC)*(0.179+0.421*EXP(-0.721*OAM))*OAM)
CCC
CCC   C.4 Compute reflectivity coefficient (RS).
CCC        (QUAL2 Step 2-8)
CCC
      GO TO (30,31,31,31,31,31,32,32,32,32,33), NL
   30 AR=1.18
      BR=-0.77
      GO TO 34
   31 AR=2.20
      BR=-0.97
      GO TO 34
   32 AR=0.95
      BR=-0.75
      GO TO 34
   33 AR=0.35
      BR=-0.45
   34 CONTINUE
      RS=AR*(CON3*ALPHA)**BR
CC
CC  Add test for RS greater than 1.0.
CC
      IF(RS.GE.1.0) GO TO 35
CC
CCC
CCC   C.5 Compute atmospheric transmission term (ATC).
CCC
      ATC=(A2+0.5*(1.0-A1-DAT))/(1.0-0.5*RS*(1.0-A1+DAT))
CCC
CCC   C.6 Compute net solar radiaiont for the time
CCC        interval delta t. (QUAL2 Step 2-9)
CCC
      SONETS=SOLAR*ATC*CS*(1.0-RS)
      GO TO 36
   35 SONETS=0.0
```

```
   36 CONTINUE
      SONET=SONETS
      CLC=1.0+0.17*CLOUD**2
CCC
CCC  NCASI Commentary, HSNET Section D. (QUAL2 Step 3-0)
CCC   D. Compute heat fluxes from other terms.
CCC
CCC   D.1 Long wave atmospheric radiation (HA).
CCC
      HA=0.97*1.73E-09*2.89E-06*(DRYBLB+460.0)**6*CLC
CCC
CCC   D.2 Water surface back radiation (HB).
CCC
      VPW=0.1001*EXP(0.03*PTF)-0.0837
      HB=0.97*1.73E-09*(PTF+460.0)**4
CCC
CCC   D.3 Evaporation (HE).
CCC
      EVAP=62.4*(AE+BE*WIND)
      HE=EVAP*(VPW-VPAIR)*(1084.0-0.5*PTF)
CCC
CCC   D.4 Conduction (HC).
CCC
      HC=0.01*EVAP*(DRYBLB-PTF)*(1084.0-0.5*PTF)
CCC
CCC  NCASI Commentary, HSNET Section E.
CCC   E. Compute net heat flux from all sources.
CCC
      HSNET=SONET+HA-HB+HC-HE
CCC
CCC  NCASI Commentary, HSNET Section G.
CCC   G. Return to FINK.
CCC
      RETURN
      END
```