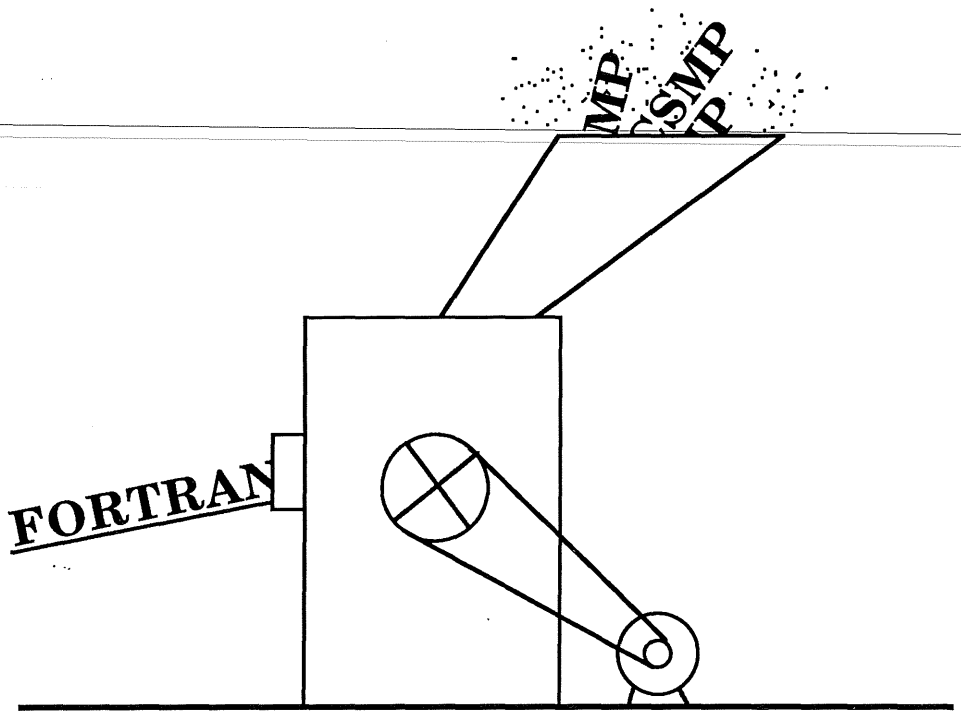
Simulation Reports CABO-TT

The FORTRAN version of CSMP MACROS
(Modules for Annual CROp Simulation)

D.W.G. van Kraalingen and F.W.T. Penning de Vries

Simulation Report CABO-TT nr. 21



A joint publication of

Centre for Agrobiological Research (CABO)

and

Department of Theoretical Production Ecology (TPE), Agricultural University

Wageningen / Los Baños 1990

DRL2SU.FOR 89
SOIL.DAT 94
SUPSEV.FOR 95
Appendix B : MACROS library 97
Appendix C : TTUTIL library 107
Appendix D : Weather reading program 129
Appendix E : RECOVER program 143
Appendix F : Error messages from MACROS library 145

Acknowledgements

The SARP project (CABO/TPE/IRRI) is acknowledged for providing the necessary funding and the creation of good working circumstances. The following persons contributed much to the final result. C. Rappoldt who developed the rerun facility of the model in Wageningen while the work was taking place on the IRRI institute in Los Baños, The Philippines. M. van den Berg for providing the necessary computer communications across the world, and for good computing facilities. H. Drenth is acknowledged for carefully reading the draft manual.

1 Introduction

The series of MACROS modules for simulation of crop growth and water consumption was developed for crop production research. The documentation to the modules (Penning de Vries *et al.*, 1989) also serves students as a textbook to crop modeling. The modules are written in the simulation language CSMP (Continuous System Modeling Program, IBM, 1975, see references).

CSMP has many nice features that are very helpful to modelers, and it can be run on mainframe and IBM-PC compatible computers. However, CSMP is not available at most universities and research institutes. By providing the modules also in FORTRAN, we aim at reaching a much larger group of potential crop modelers. The FORTRAN-77 language is well defined and good compilers are widely available for many different computers. Models in this language are therefore portable to all computers for which a FORTRAN-77 compiler is available. There are also crop modelers who prefer the use of the FORTRAN language over CSMP. The FORTRAN version of MACROS presented here will hopefully encourage the exchange of the MACROS crop simulation programs.

This technical report documents the guiding principles of the translation of the modules, differences between the CSMP and the FORTRAN versions, recommendations for further extensions, requirements for simulation and the complete listing of all modules. As much as possible, the structure and names of the CSMP modules have been retained in the FORTRAN modules, so that the original documentation can continue to serve as a manual for the scientific background. FORTRAN subroutines and functions provide the user of the FORTRAN modules with many of the practical features that are available in CSMP.

If you are only interested in executing the model, see Chapter 5. If you want to understand the program and its concepts, see Chapters 2, 3, and 4. A description of how changes should be implemented in the model is given in Chapter 6.

The FORTRAN modules are available on a set of floppy disks from SARP-IRRI (IRRI/APA, P.O. Box 933, Manila, Philippines) and from SARP-Wageningen (CABO, P.O. Box 14, 6700 AA Wageningen, Netherlands).

2 Principles of the translation

In this Chapter, the rules are discussed that were adhered to in the translation other than the general structure of Euler simulation in FORTRAN (this is outlined in Chapter 3).

The following concepts were applied in the translation:

- **Leave as much as possible of the original programs intact**

The MACROS modules consist of a set of CSMP programs that can be combined to form a complete CSMP program. For example, the module for potential crop growth (L1D) can be run on its own, but can also be combined with a crop transpiration/soil evaporation section (L2C) and a water balance for soils with impeded drainage (L2SS). In addition to this set of modules a library of FORTRAN subroutines and functions is available that calculate driving variables and rates of change. This library will afterwards be called the MACROS library. Examples are the daily photosynthesis computations of the FUPHOT function or the calculation of daylength by SUASTR. It is important to make a distinction between the MACROS library and the CSMP FUNCTION library, as the former is part of the crop growth simulation programs while the latter is provided with the CSMP language.

In the FORTRAN version, the concept of the different crop modules has been retained. However, the modules are now subroutines that have to be merged with a main program (called MACROS.FOR) to obtain an executable program. Most of the variable names and program lines are identical to the CSMP version except that program lines now start in the 7th column in accordance with the FORTRAN syntax rules. Also many of the numeric functions (such as LIMIT) that CSMP provides have been rewritten in FORTRAN. Some of these functions are identical to the CSMP functions, others have been slightly changed (such as AFGEN).

The water balance modules L2SS and L2SU are two different subroutines now that communicate only relevant parameters with the plant, such as water content and transpiration. The interface to the two routines is the same although internally they differ greatly. Both calculate water content as a function of soil parameters and rates of change (transpiration, rain etc.). Some 'driver' code had to be added before and after the modules to make them 'look alike' from the outside. This is reflected in the subroutine names, DRL2SS and DRL2SU (driver of L2SS and L2SU, respectively).

The MACROS library of subroutines and functions (such as FUPHOT and SUASTR) is also used with the translated version. Only some minor modifications were necessary in this library to make it compatible both with the FORTRAN and the CSMP version of MACROS. The changes are given in Chapter 7.

- **Simulation results of the versions should be identical**

Care was taken that at each time step, the results of the FORTRAN version of MACROS and the CSMP version did not differ by more than a rounding error. This was also checked by solving some of the exercises in Penning de Vries *et al.* (1989) using the FORTRAN version of MACROS and comparing the output with the reference output of the exercises.Such checks guarantee that statements have been put in the right order and that the program structure is correct.

- **Hide program complexity that is irrelevant to the simulation as much as possible**

In the FORTRAN program, sometimes complex algorithms are used. Examples are the set of routines that is used to read parameter values from data files, the routines to generate output tables and graphs and the TIMER routine. These routines have a clearly defined task that is easy to understand. The implementation into a FORTRAN program however can be very complex. For example, it is easy to understand that with the following statement: 'CALL RDSREA ('WLVI', WLVI)' you get the value of the parameter WLVI from a data file. The FORTRAN code however is very complex and should not bother the user of the simulation program. These routines are stored in a separate library TTUTIL, the utility library of the Dept. of Theoretical Production Ecology of the University of Wageningen (see Appendix C).

We have also tried to make the program flow straightforward with as few GOTO's as possible. In general, liberal use of GOTO statements is considered bad programming as the GOTO's and the corresponding CONTINUE labels tend to be confusing. The problem in fact is caused by the CONTINUE statement because it represents a location where any section of the program can jump to. In other words with the statement GOTO 10, you will know where the program resumes but at the line 10 CONTINUE you can never be sure from where in the program this statement is used to jump to.

- **Write standard FORTRAN-77 and provide easy transfer to new FORTRAN language definitions**

The FORTRAN version of MACROS was written entirely in FORTRAN-77. This language is well defined (better than Pascal or C) and good compilers are available for many computers and operating systems. (The definition of the language is published in ANSI document X3.9-1978). Also many good textbooks exist from which programming in FORTRAN-77 can be learned. Some of these have been listed in the reference section. The definition of the language is summarized in ter Haar (1983) among others.

The portability of the program is thus greatly improved by adhering to the definition of standard FORTRAN-77, and avoiding compiler extensions that many compilers provide. To

further improve portability among compilers, we have avoided the use of certain features that are part of the standard of the language (such as nested character operations) but that, according to experience, have sometimes not been implemented correctly in the compiler.

At the time of writing of this document, a group of people is working on a new FORTRAN standard dubbed FORTRAN-8X. Although there is not yet full agreement among the committee members, consensus has been reached over a number of features. Among them are advanced control structures such as DO-WHILE and volatile local variables in subroutines and functions. In the FORTRAN version of MACROS we have anticipated these future modifications to the language by emulating a DO-WHILE control structure with IF-ENDIF statements following the guidelines in ter Haar (1983) and by including a SAVE statement in all the subroutines and functions to prevent disappearance of local variables upon return to the calling program. If DO-WHILE becomes part of the language, the emulated DO-WHILE structure can be easily modified:

Emulated DO-WHILE	True DO-WHILE

`10  IF (logical expression) THEN`
`    ...`
`    ...`
`  GOTO 10`
`  END IF`
	`DO WHILE (logical expression)`
	`    ...`
	`    ...`
	`END DO`

Fig. 1: The standard FORTRAN structure to emulate a DO-WHILE loop.

- **Increase transportability by not using large amounts of memory**

The use of large arrays is avoided as these increase RAM-memory requirements. Although programming is often much easier and program execution much faster when arrays are used to solve specific problems, it limits the number of computers on which the program can be run. As disk memory is often much larger than RAM-memory, information is stored in temporary files where possible. This practice limits the total array size of the whole program to about 42kb of memory.

- **Safeguard the program against inaccurate floating point operations**

The definition of standard FORTRAN-77 (as that of most languages) does not specify the algorithms to be used for floating point calculations. Consequently, the results of floating point operations can be somewhat different among compilers. The portability of a program in general is improved if these problems are anticipated and solved.

The inaccuracy is of importance in the TIMER routine which should trigger output to file each time that TIME is a multiple of PRDEL (consequently PRDEL is the time between different outputs). Due to floating point inaccuracy, it is not correct to simply test if TIME is a multiple of PRDEL by using a MOD function. This problem has been solved by using integer variables (see TIMER routine).

3 Euler integration in FORTRAN

Various integration methods can be used in the simulation of continuous systems, ranging from simple rectangular integration (Euler) to higher order integration algorithms (trapezoidal, Runge-Kutta, etc.), possibly with a variable time step. From the point of view of program structure, a program that accommodates Euler integration only, is less complicated easier to understand than one accomodating higher order integration methods and. With such a program structure however, the possibility to use higher order integration methods is lost. Because simulation of crop growth in CSMP often uses Euler integration with a fixed time step of one day and because the program structure is less complicated, we have adopted this integration method in the FORTRAN version.

Order of execution

The following circle (Fig. 2) shows the correct order in which calculations should be executed:

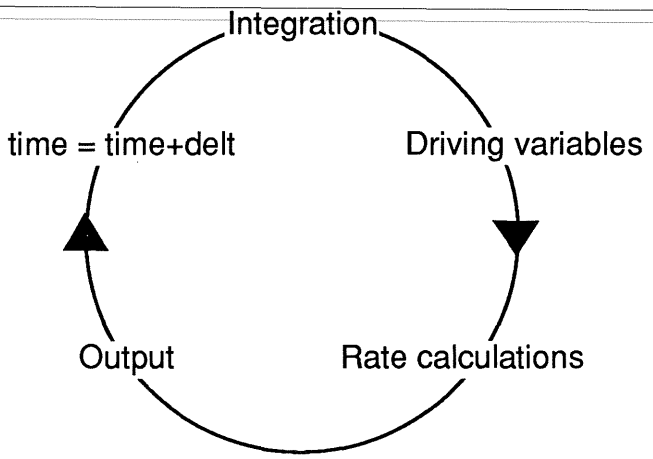


Fig. 2: The order of execution in simulation of continuous systems using Euler integration.

Note that in this sequence, state variables and rates of change correspond to the time for which they are calculated. That this sequence of calculations gives results in FORTRAN and CSMP that are identical is shown for a simple simulation of exponential growth in Listing 1 and 2:

Listing 1: CSMP program of exponential growth and output (only relevant output):

```
TITLE DEMONSTRATION
INCON IH=1.
PARAMETER RGR=0.1
H  = INTGRL (IH, GR)
GR = RGR*H
METHOD RECT
TIMER TIME=0.0, FINTIM=10., DELT=1.0, PRDEL=1.0
```

```
PRINT H, GR
END
STOP
ENDJOB

0TIMER VARIABLES     RECT     INTEGRATION       START TIME  = .00000
    DELT         DELMIN       FINTIM       PRDEL        OUTDEL      DELT
   1.0000     1.00000E-06   10.000       1.000         .00000      1.0000
1  DEMONSTRATION
0  TIME          H             GR
   .000000        1.0000        .10000
   1.00000        1.1000        .11000
   2.00000        1.2100        .12100
   3.00000        1.3310        .13310
   4.00000        1.4641        .14641
   5.00000        1.6105        .16105
   6.00000        1.7716        .17716
   7.00000        1.9487        .19487
   8.00000        2.1436        .21436
   9.00000        2.3579        .23579
   10.0000        2.5937        .25937
1$$$ SIMULATION HALTED FOR FINISH CONDITION  TIME      10.000
1$$$ CONTINUOUS SYSTEM MODELING PROGRAM III   V2.0     EXECUTION OUTPUT
```

Listing 2: FORTRAN program of exponential growth and output

```
      PROGRAM DEMO
      IMPLICIT REAL (A-Z)
      PARAMETER (RGR=0.1, FINTIM=10., DELT=1.0)

      H  = 1.0
      GR = 0.0
      TIME = 0.0

      OPEN (20, FILE='RES.OUT', STATUS='NEW')
      WRITE (20,'(A9,2A13)') 'TIME','H','GR'

10    IF (TIME.LE.FINTIM) THEN
         H  = H+GR*DELT                          <--integration
                                                 <--driving variables (none)
         GR = RGR*H                              <--rate calculation
         WRITE (20,'(3G13.5)') TIME, H, GR       <--output
         TIME = TIME+DELT                        <--time=time+delt
      GOTO 10
      END IF

      STOP
      END
```

```
     TIME            H            GR
   .00000        1.0000        .10000
   1.0000        1.1000        .11000
   2.0000        1.2100        .12100
   3.0000        1.3310        .13310
   4.0000        1.4641        .14641
```

```
5.0000        1.6105        .16105
6.0000        1.7716        .17716
7.0000        1.9487        .19487
8.0000        2.1436        .21436
9.0000        2.3579        .23579
10.000        2.5937        .25937
```

In theory, the sequence in which different state variables are updated is not important because their values should not depend on each other but should be fully determined by the rate variables. In practice however, state variables are sometimes used that are derived from other state variables (e.g. root/shoot ratio or total weight of leaves is weight of dead leaves plus weight of green leaves). It is therefore important to put the state calculations in the right order just as this is necessary for the rate calculations.

The different types of calculations (integration, driving variables and rate calculations) should be strictly separated for correct simulation results. In other words, all states should be updated, then all driving variables should be calculated after which all rates of change should be calculated.

Since the calculations of rates and states cannot be mixed during the simulation but should be executed separately, this would imply merging all state calculations into one block and similarly all rate calculations. Often in a simulation model, different subprocesses are interacting such as a plant transpiring water from a soil. In many cases the interactions among the subprocesses are only weak. For the plant/soil system, in the plant submodel, the water content as a function of rooting depth is needed which is then used to determine water uptake for transpiration as a function of rooting depth. The plant and soil water submodel thus share a limited amount of information whereas the plant and soil water submodels may contain very detailed descriptions of plant growth and soil moisture redistribution with many different rate and state calculations.

Considering this, it is not a good solution to combine all the state calculations from the different subprocesses into one large program section and all rate calculations in another. It is however feasible to separate the state and rate calculations within the subprocess descriptions (such as the plant) and have a calling program decide which of the two to execute. With this method, the states can be calculated separately from the rates, whereas rates and states pertaining to the same subprocess are within the same program. This technique is also discussed in Van Kraalingen and Rappoldt (1989).

This solution is illustrated in Fig. 3. The program lines of the plant and soil water subprocesses are separated into rate and state sections and only one of these is executed during a single call.

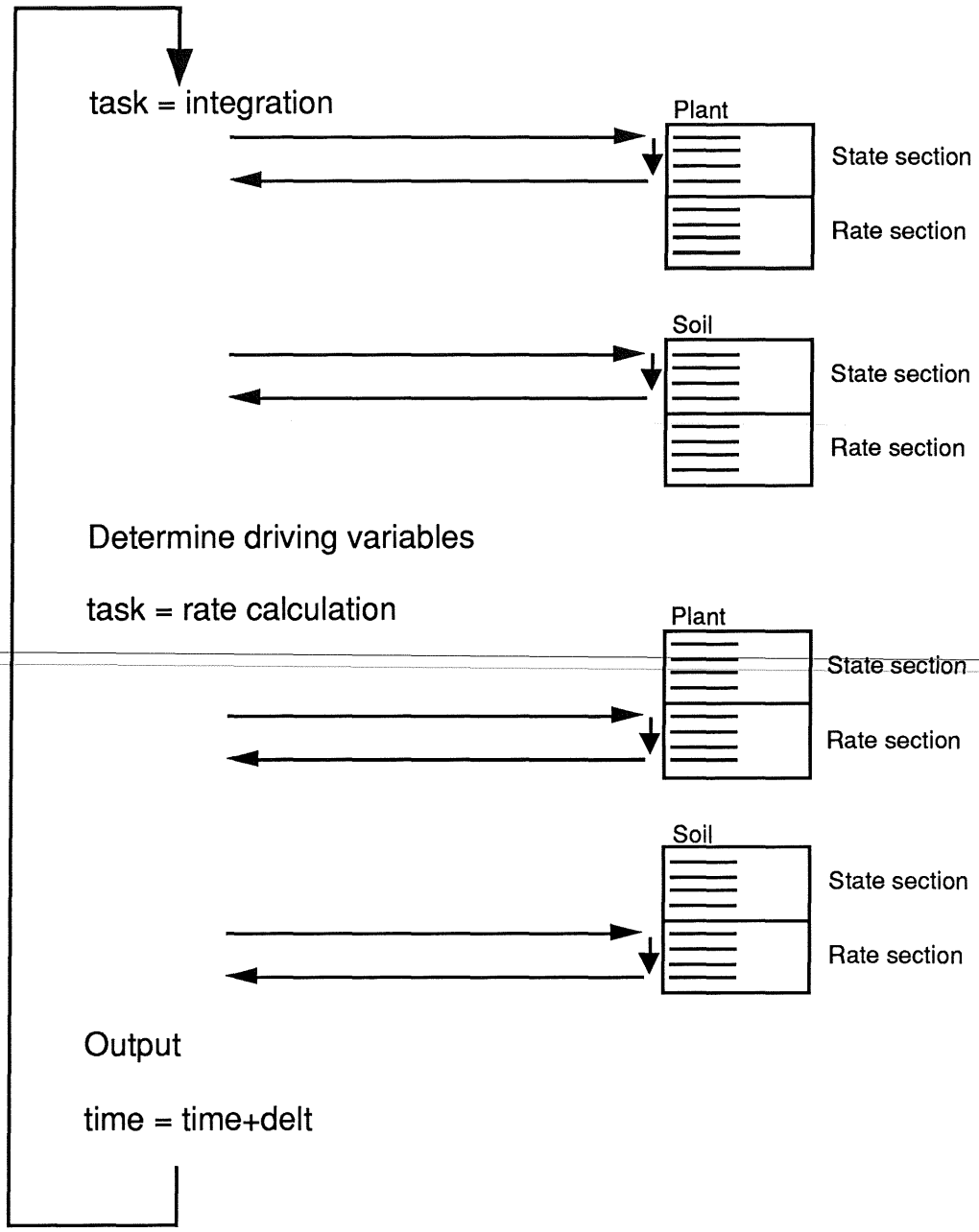


Fig. 3: General structure of incorporating several subprocesses containing integration in a single simulation model.

Some aspects of the simulation circle have not been discussed so far i.e. the way to initialize the states, where to enter the circle it and where to leave it (see Fig. 2).

The most convenient point to leave the circle is between updating time and integration because there time and corresponding rates have been written to the output device and after the time update it seems natural to check whether the finish time (FINTIM) has been exceeded or whether further simulation is required. Consequently, the circle is entered also between time update and integration. The most convenient way to initialize the subprocesses

is to have that controlled by the main program. That makes reruns possible, as in the main program the whole model can be reset to its initial state and run again with different weather data for instance. These refinements are shown in Fig. 4.

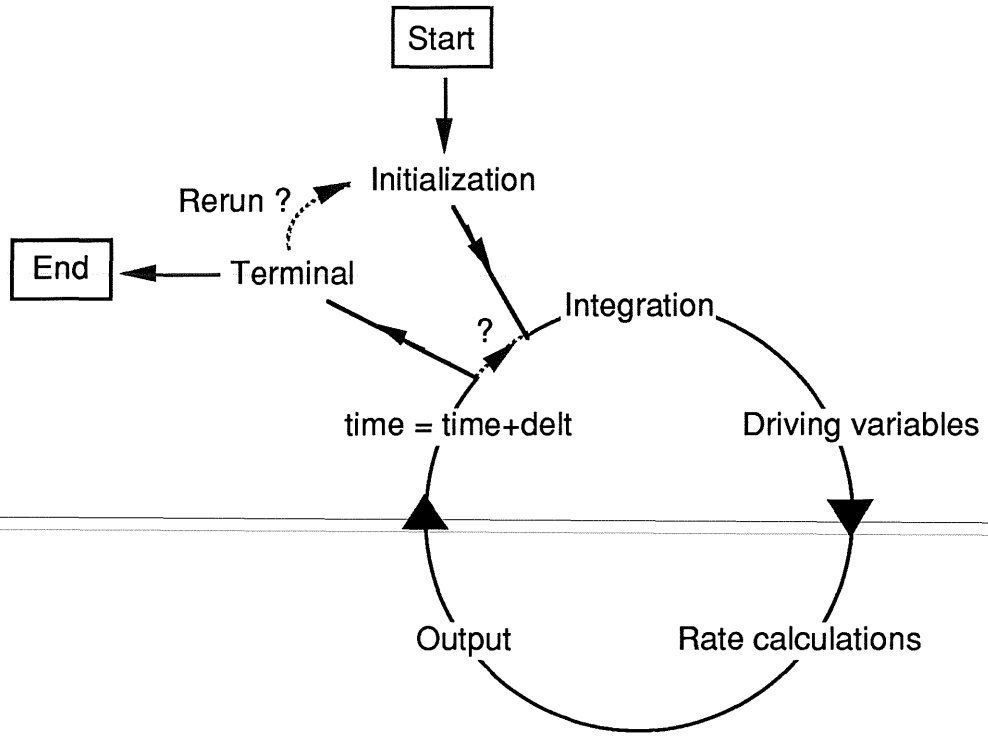


Fig. 4: The order of execution in simulation of continuous systems using Euler integration illustrating where to enter and leave the circle and how reruns are implemented.

The question mark between time update and integration indicates the point where the decision is made about execution of another time step. If not, the model proceeds to the terminal section, else the circle is run once more. After proceeding to the terminal section a decision has to be made whether a rerun is required. The model then has to be re-initialized and a new simulation run is started.

As shown in Fig. 3, the modularity of the subprocess descriptions is conserved by introducing the task concept (the calling program decides what the subroutine should do, either integration or rate calculation). The ability to do reruns requires an external initialization of the different subprocess descriptions which can also be driven by the task variable as well as some terminal calculations that are required (calculation of harvest index etc.). A subprocess description should thus recognize four different tasks: initialization, integration, rate calculation and terminal calculation.

A consequence of this structure is that the first step after initialization is integration. This does no harm if the rates have been set to zero explicitly so that the first integration has no effect on the value of the states. In practice this would imply incorporation of many rate assignments to

zero into the model. This can be avoided by skipping integration if the previous task was initialization, during which the states have been assigned values anyway. The subsequent rate calculation will then use the state variables to initialize the rates of change.

In the next Chapter we will discuss how this rather theoretical outline of continuous simulation using Euler integration is implemented in FORTRAN.

4 Implementation of Euler integration in FORTRAN

In this Chapter the principles of Fig. 3 and Fig. 4, discussed in the previous Chapter, are translated into a full program. Also some more technical details will be discussed to give you some understanding of how the program actually works. A description of the definition of individual subroutines and functions is provided in the Appendices.

At the end of this Chapter you should have a fair understanding of the FORTRAN version of MACROS. For details not explained in the text you are referred to the comment headers of the subprograms or the actual program text. More technical details of the use of subprograms in simulation models can be found in van Kraalingen & Rappoldt (1989).

Control of the time loop

As shown in Fig. 4, after each time step a decision has to be made whether another time step is required or whether the simulation should proceed to the terminal section. One of the criteria to stop the simulation is that the finish time (FINTIM) has been exceeded. In crop growth simulation however, more often simulation is terminated because the crop is mature (DS $\geq$ 2) or some other criterion has been met (e.g. in the MACROS modules, three consecutive days with negative crop assimilation). In other words, termination of the simulation loop should be possible from within each of the subprocesses. This is most conveniently done with a global variable called TERMNL of type LOGICAL that indicates whether the loop should be terminated. The simulation loop should continue as long as this variable is .FALSE. and the criterion is programmed as an emulated DO-WHILE loop. This is shown in the example program below (Listing 3). (The implementation of the rerun facility is not shown here.)

Some other features are also illustrated in this example program:

- **Implementation of the task concept**

The task concept discussed in the previous Chapter is implemented using an INTEGER variable ITASK that can have four different values indicating the required action of the subroutines: 1=initialization, 2=rate calculation, 3=integration and 4=terminal.

- **Time control and output at multiples of PRDEL**

The time control in a simulation program is more complicated than simply the increase of TIME with DELT and this has been hidden in a TIMER subroutine as well as the setting of a flag when output is required (at the start time of the simulation, when TIME is a multiple of PRDEL and at the time that the simulation terminates). The basic actions of the subroutine TIMER are: ITASK=1: check values of FINTIM, DELT, TIME etc. and copy these to variables local to the

subroutine, turn output flag on, ITASK=2: check whether local time variables have the same value as the global time variables, add DELT to TIME, flag if TIME is a multiple of PRDEL using the variable OUTPUT, flag if TIME has exceeded FINTIM using the variable TERMNL.

- **An outline of a plant routine**

In program listing 3, a 'stripped' version of a plant routine is shown. Note that the subroutine consists of four sections corresponding with initialization, rate calculations, integration, and terminal. We have chosen for each subprocess to write its relevant output variables to an output device instead of organising the output from the main program. This has the advantage that fewer variables have to be communicated with the main program and reduces the number of changes that have to be made in the main program when a new plant routine is used. Two program lines need further attention. The last executable statement in the plant routine (ITOLD=ITASK) is necessary to make sure that the first integration (the one following initialization) is dummy. Therefore, the first line after the SAVE statement checks on this and returns execution to the calling program.

Listing 3: Example program of simulation loop driving a hypothetical plant routine.

```
      PROGRAM SMALL
      IMPLICIT REAL (A-Z)
      LOGICAL TERMNL, OUTPUT
      INTEGER ITASK

*     initialization
      TERMNL = .FALSE.
      ITASK  = 1
      TIME   = 0.0
      FINTIM = 100.
      DELT   = 1.0

*     initialization of TIMER and PLANT subroutines
      CALL TIMER (ITASK, DELT, PRDEL, FINTIM, TIME, TERMNL, OUTPUT)
      CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT, ...)

*     run loop as long as TERMNL is not .TRUE.
10    IF (.NOT.TERMNL) THEN

*        integration
         ITASK = 3
         CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT, ...)

*        driving variables
         CALL WEATHR (DATE, weather variables)

*        rate calculation
         ITASK = 2
         CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT,
     &               environmental variables)
```

```
*        time update, TIME multiple of PRDEL ? => OUTPUT = .TRUE.
         CALL TIMER (ITASK, DELT, PRDEL, FINTIM, TIME, TERMNL, OUTPUT)
      GOTO 10
      END IF

*     terminal calculations
      ITASK = 4
      CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT, ...)

      STOP
      END

      SUBROUTINE PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT,
     &                  environmental variables)
      IMPLICIT REAL (A-Z)
      LOGICAL TERMNL, OUTPUT
      INTEGER ITASK, ITOLD
      SAVE

      DATA ITOLD /4/

*     return if first integration
      IF (ITOLD.EQ.1 .AND. ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN
*        initialization of states and parameters
         WLVI = 6.8
         WLV = WLVI
         ...
      ELSE IF (ITASK.EQ.2) THEN
*        rate calculation and output
         GLV = ...
         ...
         IF (OUTPUT .OR. TERMNL) THEN
*           output to file with unit=IUNITO
            WRITE (IUNITO, format) WLV, ...
         END IF
      ELSE IF (ITASK.EQ.3) THEN
*        state calculation and check if TERMNL must be .TRUE.
         WLV = WLV+GLV*DELT
         ...
         IF (DS.GE.2.0) TERMNL = .TRUE.
      ELSE IF (ITASK.EQ.4) THEN
*        terminal calculations
         ...
      END IF

*     save old task
      ITOLD = ITASK
      RETURN
      END
```

The MACROS program and subprocess descriptions

The FORTRAN version of MACROS thus consists of a main program and subroutines that represent the different modules (similar to Listing 3). Because of the complexity of the descriptions of the subprocesses (soil water and plant), and the limited number of interactions between plant and soil, the soil water and plant subprocesses have been separated into different subroutines. The following modules are now available as subroutines:

Plant subprocesses:

L1D	Potential production,
L1DT	Potential production and potential transpiration,
L1QM	Potential production with quarter-day time steps, and morphological development of tillers,
L2DT	Water limited crop production (should be combined with a water balance subroutine,either DRL2SS or DRL2SU, and the potential soil evaporation subroutine SUPSEV).

Soil water subprocesses:

DRL2SS	Water balance of soil with impeded drainage (up to ten compartments),
DRL2SU	Water balance of free draining soil (three compartments),
SUPSEV	Potential soil evaporation (below a standing crop).

Full listings are given in Appendix A

Coupling of water balance and plant subprocesses

As discussed in Chapter 3 complicated subprocess descriptions should be kept separate if the interactions are limited. As is the case with the descriptions of the water balance, potential soil evaporation, and the plant. The water balance modules L2SS and L2SU have been modified in such a way that each can be coupled to the model without any changes in the plant routine or the main program.The sequence of the calls in the MACROS program is shown in Listing 4 using the modules L2DT, SUPSEV, and DRL2SS. Only interaction variables that are effective during a specific task are shown.

Listing 4: Simulation loop showing the variables that are communicated among the subroutines during the different tasks.

```
*     initialization
      ITASK = 1
      CALL DRL2SS (ITASK,  ...,   NL,  TKL, TKLT, ZRTMS,        <- a
     &             WCWP, WCFC, WCST, WDCL, RFSD,   ..., WCLQT)
      CALL L2DT (ITASK, ..., ALVX, PLHT, WDLV)                  <- b

*     run loop as long as TERMNL is not .TRUE.
```

```
10     IF (.NOT.TERMNL) THEN

*         integration
          ITASK = 3
          CALL L2DT (ITASK, ..., ALVX,   PLHT, WDLV)              <- c
          CALL DRL2SS (ITASK, ..., WCLQT)                         <- d

*         driving variables
          < generation of driving variables >

*         rate calculation
          ITASK = 2
          CALL L2DT (ITASK,  ...,                                 <- e
      &                NL,  TKL,  TKLT, ZRTMS,
      &              WCWP, WCFC,  WCST, WCLQT,
      &              TRWL,  ...)
          CALL SUPSEV ( ..., WCLQT(1), WCST(1), RFSD, WDCL,       <- f
      &               WDLV,     ALVX,    PLHT,
      &               EVSC)
          CALL DRL2SS (ITASK, ..., EVSC, ..., TRWL, ...)          <- g

*         time update
          CALL TIMER (...)
       GOTO 10
       END IF
```

The state variables of the plant subprocess are: leaf area index (ALVX, for safety reasons this is a copy of the actual variable ALV), plant height (PLHT) and leaf width (WDLV). The state variable of the water routine is the water content per soil compartment (WCLQT).

The letters of the following subsections refer to Listing 4.

a: The soil water balance is initialized, soil compartment definitions, soil clod size, reflection coefficient for dry soil, and initial water contents (WCLQT) are read and made available to the rest of the program.

b: Plant routine is initialized, initial leaf area index, plant height, and leaf width are made available.

c: Leaf area, plant height and leaf width are integrated (in fact leaf width is constant, nevertheless it is a state variable).

d: Integration of water contents

e: Calculation of water uptake for transpiration per soil compartment (TRWL). Necessary for that are: number of compartments (NL), depth of total soil profile (TKL), thicknesses of individual soil compartment (TKLT), maximum rooting depth in the particular soil (ZRTMS), water contents per compartment at: wilting point (WCWP), field capacity (WCFC), and saturation (WCST), and actual water contents per compartment (WCLQT).

f: Calculation of potential soil evaporation (EVSC), taking into account a standing crop, from water content of first compartment (WCLQT(1)), saturated water content of first compartment (WCST(1)), reflection coefficient of dry soil, size of soil clods, width of leaves, leaf area index, and plant height.

g: Calculation of internal rates of change in content per compartment using potential evaporation and water uptake for transpiration.

Initialization of states and parameters from external data files

Of the four sections distinguished in the plant submodel, the sections for integration, rate calculation and terminal consist of relatively straightforward calculations. The program lines of these sections are very similar to those in CSMP modules as described in Penning de Vries *et al.* (1989). The initialization section however, is very different from that in CSMP and needs further explanation.

As explained in Chapter 3, model parameters have to be given values and states have to be initialized. As shown in Listing 4, this can be done by simple assignments such as WLVI = 6.8. Any change in the value of one of the parameters or initial states however, would then require compilation and linking of the model, a serious drawback compared to CSMP. In CSMP one can run the model with different parameter sets automatically (after the END statement) or after parameter values in the CONTRO.SYS have been changed. To introduce that option in the FORTRAN version of MACROS, initial values are read from external data files.

As you can see in the program text of MACROS, the TIMER routine is not initialized from a data file like the plant and soil routines, but initial values are extracted from a data file in the main program, after which TIMER is initialized with these values.

The values are extracted from the data files using a set of subroutines whose names all begin with RD (e.g. RDSREA means 'read a single real value'). With these routines one can request the value by supplying the name of the requested variable. The statement CALL RDSREA ('WLVI', WLVI) requests the value of WLVI which the subroutine will assign to the variable WLVI. It does so by searching for the line: WLVI = <value> in the data file (in fact, the procedure is slightly different but that does not affect the understanding of the concept of the RD routines. The values are read from a temporary file which is created after syntax check and analysis of the data file). Consequently, the data file consists of variable names and variable values. The syntax of the data files is explained in more detail in Chapter 5.

In the FORTRAN version of MACROS the plant and water balance subroutines have separate data files called PLANT.DAT and SOIL.DAT respectively. During initialization of a subprocess, its corresponding data file is opened, a temporary file is created (using subroutine RDINIT) and the variables are read from the temporary file (using other RD routines). Llisting 5 shows part of the initialization section of a plant routine.

Listing 5: Example illustrating the use of some RD routines.

```
...
IF (ITASK.EQ.1) THEN
   CALL RDINIT (IUNITP, IUNITO, FILEP)
```

```
      CALL RDSREA ('WLVI', WLVI)
      WLV = WLVI
      CALL RDSREA ('PLMXP', PLMXP)
      CALL RDAREA ('PLMTT', PLMTT, ILAR, IPLMTN)
      ...
      CLOSE (IUNITP, STATUS='DELETE')
   ELSE IF (ITASK.EQ.2) THEN
      ...
```

The statement:

```
CALL RDINIT (IUNITP, IUNITO, FILEP)
```

calls the routine that 1) opens the file with variable name FILEP using unit=IUNITP+1 (FILEP is a character string that has been assigned the string 'PLANT.DAT' in the calling program), 2) analyzes the data file, 3) creates a temporary file from the data file using unit=IUNITP, 4) closes the data file (leaving IUNITP used !!), and 5) sends all error messages that occurred to a log file (with unit=IUNITO, this log file must have been opened previously !). After this call, the plant subroutine can acquire the numerical values (including arrays) through three different RD routines, RDSREA (read single real), RDSINT (read single integer), and RDAREA (read array of reals), two of which are given in Listing 5. The CLOSE statement deletes the temporary file that is created by the RD routines.

The implementation of reruns

Often several runs with a crop growth simulation model are required. Examples are the study of crop yields for a number of years, or analysis of the effect of a different value for an input parameter. In CSMP this can be done by repeating the parameter in question after an END statement. (Listing 6).

Listing 6: Example of the rerun facility in CSMP.

```
TITLE DEMONSTRATION
PARAM YEAR=1984.
< model description etc. >
END
PARAM YEAR=1985.
END
PARAM YEAR=1986.
STOP
ENDJOB
```

In Listing 6, in the first run weather data from 1984 are used, additional runs are made using weather data from 1985 and 1986. This facility in CSMP is called the rerun facility. The output of the different runs is merged in the same file for easy comparison.

In the FORTRAN version of MACROS we have included a similar rerun facility. Before this was

implemented one had to make changes in the data files and run the model again (however, without compiling and linking). Each new run would also delete existing output files and consequently, this was considered not an elegant way to do reruns.

The general idea behind this facility is that the data files are retained and that the changes in data are specified in a separate file called 'RERUNS.DAT' which may contain variable names from any of the 'standard' data files. Thus, in the file 'RERUNS.DAT', parameters may occur from soil, plant and TIMER data files. In the first run, the values from the standard data files will be used after which those values will be replaced automatically with the values from the rerun file. Execution will continue until all the rerun sets from 'RERUNS.DAT' have been used. The output of the different runs is merged in one output file. An example file is:

```
WLVI  = 8.0; DSI   = 0.18
WLVI  = 6.8; DSI   = 0.25
WLVI  = 8.0; DSI   = 0.25
```

This specifies three reruns with different values of WLVI and DSI. Unlike in CSMP, variables have to be repeated even if they do not change value. This is explained in more detail in Chapter 5.

It may be deduced from Fig. 4 that the control structure for the reruns should be programmed as a loop around the actual model. In Listing 7 the principle of the reruns is illustrated using the main program of Listing 3 as a basis. The contents of the main loop (IF... until END IF) have not been repeated to shorten the text.

Listing 7: Schematic program showing the implementation of reruns.

```
      PROGRAM SMALL
      IMPLICIT REAL (A-Z)
      LOGICAL TERMNL, OUTPUT
      INTEGER I, ITASK, INSETS

      CALL RDSETS (..., INSETS)

      DO 5 I=0,INSETS
         CALL RDFROM (I, ...)

*        initialization of TIMER and PLANT subroutines
         ITASK  = 1
         TERMNL = .FALSE.
         TIME   = 0.0
         FINTIM = 100.
         DELT   = 1.0
         CALL TIMER (ITASK, DELT, PRDEL, FINTIM, TIME, TERMNL, OUTPUT)
         CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT, ...)

*        run loop as long as TERMNL is not .TRUE.
10       IF (.NOT.TERMNL) THEN
         < main loop contents removed >
```

```
            GOTO 10
            END IF

            ITASK = 4
            CALL PLANT (ITASK, OUTPUT, TERMNL, TIME, DATE, DELT, ...)
5       CONTINUE

        STOP
        END
```

Before entering the simulation loop, the call to RDSETS detects the possible presence of the 'RERUNS.DAT' file. The return variable INSETS contains the number of rerun sets present in the rerun file its value being zero if the rerun file is absent. The subsequent DO-loop runs INSETS+1 times, because the number of runs is always one higher than the number of sets in the rerun file (one run with standard data files + INSETS reruns). The value of the DO-loop counter (I, the set number) is subsequently used in the call to RDFROM to select a parameter set for the simulation. For `I` is zero, the standard data files will be used by the RD routines, for `I` larger than zero, the RD routines will automatically replace values with these from the rerun file. No changes are necessary in the subprocess descriptions as these replacements are internal to the RD routines. To the plant or the soil water balance routines it appears as if the used values originate from the standard data files.

Results from the rerun facility are written to the output file after analysis of the rerun file and after each replacement. If another rerun has to be executed, a check is carried out whether all the variables of the preceding set were used. If this is not the case, a typing error is assumed and the simulation is halted.

Output of simulation results

As shown in Listing 3, output is organized from each subroutine. This avoids large argument lists to communicate output variables to the main program and limits the number of changes in the main program when for instance another plant routine is used with different output variables.

All subroutines write their output to the same output file (MACROS.OUT). By using a set of special routines (the OUT routines), it is written in the form of tables. It is also possible to add printplots of selected variables to that output file. The use of the OUT routines simplifies the generation of output files considerably. The most important routine is OUTDAT. The outline of OUTDAT is shown in Listing 8. In this example, a table and a printplot is created of the function y = sin (X) for X=0, π (with intervals of 0.1 π).

Listing 8: Example program for the use of the OUT routines.

```
      PROGRAM SINE
      IMPLICIT REAL (A-Z)
```

```
      PARAMETER (PI=3.141597)

      CALL OUTDAT (1, 20, 'X', 0.)          <- define X as independent variable
      DO 10 X=0.,PI,PI/10.
         Y = SIN (X)
         CALL OUTDAT (2, 0, 'X', X)         <- store value of X
         CALL OUTDAT (2, 0, 'Y', Y)         <- store value of Y
10    CONTINUE
      CALL OUTDAT ( 4, 0, 'Y=SIN(X)', 0.) <- create output table
      CALL OUTPLT ( 1, 'Y')                 <- define Y to be plotted
      CALL OUTPLT ( 6, 'Y=SIN(X)')          <- create printplot
      CALL OUTDAT (99, 0, ' ', 0.)          <- delete temporary file
      STOP
      END
```

Output of Listing 8:

```
*-------------------------------------------------------------------------
* Run no.:  1, (Table output)
* Y=SIN(X)

       X            Y
   .00000       .00000
   .31416       .30902
   .62832       .58779
   .94248       .80902
   1.2566       .95106
   1.5708       1.0000
   1.8850       .95106
   2.1991       .80902
   2.5133       .58778
   2.8274       .30901
   3.1416      -0.39617E-05

               Y=SIN(X)
               Variable  Marker  Minimum value  Maximum value
               --------  ------  -------------  -------------
               Y            1      -0.3962E-05      1.000
               Scaling: Individual

  X
   .00000      1------------------------------------------------------------I
   .31416      I              I   1          I              I              I
   .62832      I              I              I     1        I              I
   .94248      I              I              I              I   1          I
   1.2566      I              I              I              I           1  I
   1.5708      I              I              I              I              1
   1.8850      I              I              I              I           1  I
   2.1991      I              I              I              I   1          I
   2.5133      I              I              I     1        I              I
   2.8274      I              I   1          I              I              I
   3.1416      1------------------------------------------------------------I
```

The OUTDAT and OUTPLT routines also have a task parameter as input (the first argument in the call statement), similar to the subprocess descriptions. The first call (with ITASK=1) to

OUTDAT specifies that X will be the independent variable and that unit=20 is used for the output file. Subsequent calls with ITASK=2 instruct OUTDAT to store the incoming names and numerical values in a temporary file (with unit=21). The number of values that can be stored is only dependent on free disk space and not on RAM memory. The first call to OUTDAT below the DO-loop (with ITASK=4) instructs the routine to create an output table using the information stored on the temporary file. Different output formats may be chosen, dependent on the value of the task variable. Tab-delimited format (for MS-EXCEL) can be generated with ITASK=5, two column format with ITASK=6. With any ITASK, the string between quotes is written above the output.

The first call to OUTPLT defines that 'Y' should be plotted (up to 25 variables can be plotted per graph). The second call to OUTPLT (ITASK=6) instructs the routine to create a graph using the variable(s) that were defined with ITASK=1. Two different options for the width of the plot are available, 80 and 132 columns, and two different scalings, a common scale for all variables or individual scaling for each of the variables (see Table 1). This procedure can be repeated several times. Separate printplots can be made of dry weights, weather data etc.. The final call to OUTDAT (with 99) deletes the temporary file.

Table 1: Task variable options that should be supplied to OUTPLT to generate the different print plot options.

		Width	
		132	80
Scaling	Individual	4	6
	Common	5	7

Weather data

The weather data used in the model are taken from external files. The weather data file definition however is different from those for the RD routines. In the MACROS program, a weather information system is used that is developed jointly by the Centre for Agrobiological Research and the Dept. of Theoretical Production Ecology. It is especially suited for use in crop growth simulation models. The system is documented in a separate report (van Kraalingen *et al.*, 1990) that can be obtained from the same sources as the documentation of the MACROS modules. This system however is not difficult to understand and we will only give a brief description of it. For more detailed information you are referred to Kraalingen *et al.* (1990). Some introductory paragraphs of the documentation are given here.

The weather infomation system basically consists of two parts: the weather data files and a reading program to retrieve data from those files. A single data file can contain at most daily weather data from one meteorological station for one particular year. The country name (abbreviated), station number and year to which the data refer are reflected in the name of the data file.

The reading program consists of a set of subroutines and functions of which only two are intended to be called by the user (STINFO and WEATHR). The others are internal to the reading program.

A call to the first subroutine (STINFO) defines the country (CNTR), station code (ISTN), year number (IYEAR) and the name of the directory where the weather data can be found (WTRDIR). Also a control parameter should be supplied (IFLAG) to indicate where possible messages of the system should be directed (screen and/or log file) and a name for the log file if that should differ from the default name 'WEATHER.LOG'. The subroutine STINFO returns the location parameters (longitude, LONG, latitude, LAT and altitude, ELV) of the selected meteorological station, and two coefficients of the Ångström formula (A and B) pertaining to the selected station if the radiation data are derived from sunshine hours. The value of a status variable (ISTAT) indicates a possible error or warning (e.g. the requested data file does not exist). The location parameters can be used subsequently to calculate daylength (from latitude) or average air pressure (from altitude).

After this initialization procedure, weather data for specific days can be obtained by calls to the second subroutine (WEATHR) with day number as input parameter. Output of WEATHR consists of six weather variables for that day and the value of the status variable ISTAT indicating a possible error or warning (e.g. missing data, data obtained by interpolation, requested day is out of range, etc.). The six weather variables are daily global radiation (RDTM), minimum and maximum air temperature (TPL and TPH), vapour pressure (HUAA), wind speed (WDS) and rainfall (RAIN). In Listing 9, the weather data from the IRRI/dryland site of 1984 are extracted from the file "PHIL1.984" :

Listing 9: Example of use of the weather information system.

```
      PROGRAM EXTR
      IMPLICIT REAL (A-Z)
      INTEGER IFLAG, ISTN,  IYEAR, ISTAT, IDATE
      CHARACTER WTRDIR*80, CNTR*6

      IFLAG  = 1101
      WTRDIR = ' '        <- weather files on current directory
      CNTR   = 'PHIL'     <- country name
      ISTN   = 1          <- station number
      IYEAR  = 1984       <- year number

      CALL STINFO (IFLAG, WTRDIR, ' ', CNTR, ISTN, IYEAR,
     &             ISTAT,   LONG, LAT,  ELV, A, B)
      < location parameters of station are known >

      DO 10 IDATE=1,365
         CALL WEATHR (IDATE, ISTAT, RDTM, TPL, TPH, HUAA, WDS, RAIN)
         < weather for day= IDATE is known, calculations can be done >
10    CONTINUE
```

```
      STOP
      END
```

STINFO can be called again at any time during program execution to change any of its input parameters. A call to STINFO with identical input parameters is also permitted (in fact this is done regularly in the MACROS model). Similarly, WEATHR can be called repeatedly with any day number between 1 and 365 (or 366 in case of leap-years).

The weather system can generate errors and warnings. Unlike errors from other sections of the model, the weather system never terminates execution of the model. Errors are written to the screen and the log file 'WEATHER.LOG', warnings are written to the log file only.

Errors and warnings

Errors are defined as conditions in which that makes it **impossible** to continue simulation. Examples are, a parameter value not found in a data file, weather data not available for the year requested, or a physically impossible value of a variable such as negative ALV or negative radiation. A warning occurs in case of unlikely events that do, however, not prevent continuation. Examples are, an attempt to search outside the range of the independent variable in an AFGEN function, one or more weather data that are not available for the requested day but can be provided by interpolation, and when some carbon has been found missing by the called carbon balance check. All errors terminate model execution with a message to the screen, in some cases the error is also written to the output file ('MACROS.OUT'). As a rule, all warnings are displayed on the screen and written to the output file while simulation continues.

Errors are handled basically by two routines. The SUERRM subroutine is used for variable-range errors from the MACROS library. All other errors are handled by the routine ERROR of the TTUTIL library (Appendix C). Note that messages generated by SUERRM are all displayed on the screen and written to the output file, whereas messages generated by ERROR are displayed on the screen only. Messages generated by SUERRM consist of numbers that correspond to a list of error messages in Penning de Vries *et al.* (1989). For convenience this list is reproduced in Appendix F. There is no general routine that handles warnings. The following routines are capable of generating warnings:

- from the MACROS library: - FUCCHK (carbon balance check),
  - FUWCHK (water balance check),
- from the TTUTIL library: - AFGEN, OUTDAT, OUTPLT, CHKTSK, RDSETS, RDFROM,
- from the WEATHR system: - WEATHR.

5 Operating the model and its data files

In this Chapter operating the model and the syntax of the corresponding data files are described. If you want to make modifications to the program text you are referred to Chapter 6.

No information is provided here on the concepts underlying the model, the actual programming, or the scientific background. For that you are referred to the preceding Chapters and to Penning de Vries *et al.* (1989). We assume here that you know how to compile, link, and run the program (see also Chapter 8).

The following steps have to be taken to use the model:

1) Decide which module(s) and datafiles to use,

First you have to decide which of the modules you want to run and identify their corresponding datafiles. In Table 2, you find which modules and data files you have to use, dependent on the type of simulation that is required (potential production, water limited production etc.). The following modules and datafiles and their names in Penning de Vries *et al.* (1989) are available as subroutines (in Appendix A you find complete program listings):

MACROS FORTRAN	**MACROS CSMP**	**Meaning**
Plant subprocesses:		
L1D	L1D	Potential production,
L1DT	L1D+L2C	Potential production and potential transpiration,
L1QM	L1Q+TIL	Potential production with quarter-day time steps, and morphological development of tillers,
L2DT	L1D+L2C	Water-limited crop production (should be combined with a water balance subroutine, either DRL2SS or DRL2SU, and the potential soil evaporation subroutine SUPSEV).
Soil water subprocesses:		
DRL2SS	L2SS	Water balance of soils with impeded drainage (up to ten compartments),
DRL2SU	L2SU	Water balance of free draining soils (three compartments),
SUPSEV	incorporated in L2C	Potential soil evaporation (below a standing crop).

Table 2: Combinations of modules to create full simulation programs. Objectives of the simulation dictate wich modules are appropriate.

Module type	<- - - - - - - - FORTRAN MACROS submodels - - - - - - - - ->							<- - - - - data files - - - - ->				<-subr->
Module name	L1D	L1QM	L1DT	L2DT	DRL2SU	DRL2SS	SUPSEV	PLANT	SOIL	WEATHER	TIMER	MACROS
Appendix	A	A	A	A	A	A	A	A	A	A	A	B
Purpose:												
Potential production	•							•		•	•	•
Potential production day-night cycle		•						•		•	•	•
Potential production tillers (rice)		•						•		•	•	•
Potential production and transpiration			•					•		•	•	•
Rainfed production deep water table				•	•		•	•	•	•	•	•
Rainfed production shallow ground water				•		•	•	•	•	•	•	•

Data files:

PLANT.DAT	OSIR36.DAT	Crop data for rice (listing 5 in Penning de Vries *et al.* (1989)),
SOIL.DAT	TYL13LOA.DAT	Data characterizing a loamy soil (listing 10 in Penning de Vries *et al.* (1989)),
TIMER.DAT	-	Timer and location data,
PHIL99.984	BANOS84.TAB	Weather data from Los Baños, 1984, (listing 11 in Penning de Vries *et al.* (1989)),

The subroutines are stored in separate files with the extension '.FOR'. So subroutine L1D is in file 'L1D.FOR'.

2) Append the selected module(s) with the main program,

The subroutine(s) has to be appended to the MACROS main program (in 'MACROS.FOR', remove old subroutines if present) by using an editor. Each subroutine contains a comment header that explains data type, meaning, dimension, and I/O type of the formal parameters. Between the header and the program text, a CALL statement is supplied with an asterisk in the first column. This call statement has to be inserted without the asterisks at **four** indicated positions in the MACROS program (the "`&`" should be in column 6, whereas the `CALL` starts in column 7). Note that the list of arguments in the call and in the subroutine definition are **not** identical for various soil water related routines (DRL2SS, DRL2SS, and SUPSEV).

3) Modify data files if necessary,

Most of the parameters and initial values of the state variables of the various subprocesses are read from data files. This has the advantage that the model does not have to be recompiled and linked if changes are implemented in the data only.

The following data files are used with the MACROS model (values in the data files are from Penning de Vries *et al.* (1989)):

Name:	**Used by:**	**Contents:**
TIMER.DAT	TIMER	timer variables (year, time step etc.), weather station, country,
PLANT.DAT	L1D, L1DT, L1QM, L2DT	plant parameters and initial values of state variables,
SOIL.DAT	DRL2SS, DRL2SU	soil parameters and initial water contents,
RERUNS.DAT	-	timer, plant and soil parameters (used in reruns),
Weather data files	STINFO, WEATHER	daily weather data.

The data files, except the rerun file, are given in Appendix A. Weather data are read from special data files that have a different format (Chapter 4, van Kraalingen *et al.* (1990)).

The data files 'TIMER.DAT', 'PLANT.DAT' and 'SOIL.DAT' have identical formats, and each variable in them may appear only once. The file 'RERUNS.DAT' has basically the same syntax with the exception that it should consist of identical sets of variables. The following syntax rules apply for 'TIMER.DAT', 'PLANT.DAT', and 'SOIL.DAT' :

- the file consists of variable names and numerical values, separated by an '=' sign. So: PLMX = 20., is a valid specification,
- variable names cannot exceed six characters,
- for array variables more than one numerical value may follow the equal sign, separated by commas,
- identical numerical array values may be given as n*<numerical value>,
- variables may appear in the file in any order,
- comment lines start with '*' in the first column, or '!' in any column (remainder of line is ignored),
- continuation character is ',' on preceding line, applies to arrays only,
- variable names and numerical values can be gibven on the same line if separated by a single semicolon ' ; ',
- Only the first 80 characters of each record of the data file are read.

In Listing 10 these rulers are illustrated.

Listing 10: Example of a 'RERUNS.DAT' file:

<start of file data>

```
* example data file
A = 10.                          ! single value
B = 0., 2., 3., 4.               ! array of four elements
C = 10., 20.,                    ! array continued on next line
    30., 40.
D = 100*10.                      ! array of 100 elements
E = 10.; F = 20.; G = 30.        ! more than one parameter on single
                                 ! line
```

<end of file>

If the file 'RERUNS.DAT' is absent, the model will execute only one run using the data from the data files. By creating a rerun file, the model will make additional runs with different parameters and/or initial values for the state variables. The total number of runs therefore is always one greater than the number of rerun sets. The format of the rerun files is identical to that of the other data files with the exception that, variable names may appear more than once in the file. The order and number of the variables should be the same in each set. A new set starts when the first variable is repeated. This is shown in the following example:

<start of file>
```
* example rerun file
PLMXP = 40.; ZWTB = 0.,3.0, 366.,3.0
PLMXP = 47.; ZWTB = 0.,0.5, 366.,0.5
PLMXP = 40.; ZWTB = 0.,0.5, 366.,0.5
```
<end of file>

A difference with reruns in CSMP is that each variable whose value is changed somewhere in the rerun file should be assigned a value in each set even if that value is identical to that in the previous set.

The rerun files have also been discussed from the point of view of the program in Chapter 4.

4) Check the MACROS program for the country code of the weather data (not necessary for weather data from Philippines),

The subroutine that extracts weather data from the weather files (STINFO), needs a country **name**, station **number** and year **number.** The subroutines that extract the values from the data files however, are not able to read names from the data files. It is however, useful to be able to change also the country name through a data file. Therefore, country names are stored in an array in the MACROS model. The country name is now selected with the variable ICNTR in the 'TIMER.DAT' data file. The contents of the array however, cannot be standardized and the user may have to make modifications to add new countries. Currently the array size is 1 and contains only 'PHIL', the country code for the Philippines.

5) Compile the model and link it with the MACROS, TTUTIL and WREAD libraries,

This step is necessary only if you have madechanges to the program text itself. If only data files were modified this step is not necessary.

6) Run the model

The model does not require interactive input during execution. The runs have been specified completely in the data files. During execution the model will display run number, year number and date repeatedly on the screen. Errors and warnings may occur during execution from the weather system and/or from the other modules of the model. In general they consist of one line of text, except for the messages generated by the MACROS library. (see Chapter 3 for a discussion).

7) Examine output

Two output files are created by the model, 'MACROS.OUT' and 'WEATHER.LOG'. 'MACROS.OUT' contains the output of the model with reruns merged below each other in the file. 'WEATHER.LOG' contains all the messages generated by the weather system. By default, all the comment headers of the data files, all warnings and all errors from the weather system are written to the log file.

As explained in Chapter 3, all warnings and some error messages generated by all modules, except the weather system are written to 'MACROS.OUT'. If this happens it is important to note that the output table is created after the run has been terminated and that warnings are written to the output file during simulation so that the output table follows the warnings in the output file.

See Chapter 6 if you want to remove or add a variable to the output table.

9) Error recovery

If execution during a run is terminated by some error from the model, the output file 'MACROS.OUT' will not contain the results of that specific run. The results until the time of occurrence of the error have been written to the temporary file 'RES.TMP' however. This file can be converted into an output table by running the program RECOVER. This program requests an integer number from the user. A standard output table is generated by a '4' (the default), '5' generates a tab-delimited table (meant to be imported in Excel), '6' generates an output of only two columns at a time. The output table will be written to the file 'RECOVER.OUT' so that any existing 'MACROS.OUT' file is not deleted.

Note: If you are not working with Microsoft FORTRAN 4.1 on IBM compatible PC's you have to compile the 'RECOVER.FOR' program with your own compiler because the executable file 'RECOVER.EXE' on floppy disk 1 is possibly not capable of reading a 'RES.TMP' file created by another compiler.

The listing of the RECOVER program is given in Appendix E.

6 How to make changes to an existing subroutine

It is strongly advised not to make changes in the subroutine structure unless you are well acqainted with the procedures. More often changes will be made in the description of the plant or soil subprocesses. In general you have to be more careful changing a FORTRAN program than one in CSMP. If you understand the principles of the program however, implementing the modifications correctly is not difficult. A list of possible modifications will be discussed here. We assume that you know the FORTRAN syntax rules.

If you want to create a new subprocess description (including data file initialization, integration, output, etc). follow the structure of the existing MACROS modules as closely as possible.

Modification of subprocess calculations

First define the modification in terms of program statements. If new variables are introduced, determine the type for each of them (parameter, driving variable, rate of change or state). If the new variables are local to the subroutine, determine for each of the variables the exact position in the text where they should be assigned a value. Parameters and initial values of states are likely to be given their values in the initialization section using one of the RD routines (see Chapter 3). These routines can extract variable values by their name from a data file . Depending on the data type of the variable, different RD routines can be used. With the routines RDSREA, RDSINT, and RDAREA, single reals, single integers and arrays of reals can be read. Driving variables should be assigned a value at the top of the rate section, rates should be defined in the proper order in the rate section. States should be integrated in the integration section.

Especially for rate calculations it is important that they appear in the right order. So **any variable** that appears to the right of the '=' sign of the modification you are making, and that is assigned a value in the rate section, should have been assigned a value **above** the line of the modification, i.e.:

Wrong	Right
`...`	`...`
`ELSE IF (ITASK.EQ.2) THEN`	`ELSE IF (ITASK.EQ.2) THEN`
`...`	`...`
`A = B*...`	`B = ...`
`B = ...`	`A = B*...`
`ELSE IF (ITASK.EQ.3) THEN`	`ELSE IF (ITASK.EQ.3) THEN`
`...`	`...`

If variables are to be communicated among subprocesses, include them also in the list of formal parameters.

Add a variable to the output list

The output list in general should appear at the end of the ITASK=2 section. Output however, should only be done if either the output flag ('OUTPUT'), or if the terminal flag ('TERMNL') is on. Rates, states, and driving variables should be output here. A variable can be added to the output list by simply adding another call to OUTDAT in the output list between the IF-END-IF lines. The variable names in the output file will have the same order as in the output list.

Add a finish condition

Each subprocess can terminate the run by setting TERMNL to .TRUE.. However, in each subprocess description only one place exists where this can be done such that corresponding states, driving variables, and rates are all output to file. This place is at the end of the ITASK=3 section. As shown in the listing of module L1D (Appendix A), DVS > 2 or CELV > 3 sets TERMNL to .TRUE.. Any additional finish condition should be added here, similar to the existing ones.

Add titles to the output file

As shown in the listings of the modules, in the initial section a subroutine OUTCOM is called that accepts a text string. This string is handled as a title by the output routines. Several subprocesses can send their title to the output routines and these titles are printed on top of each output table. There is no objection to several titles from a subprocess. The call to OUTCOM can be repeated several times with different text strings (see listing of L2DT). A total of 25 titles can be handled by the output routines (identical ones are discarded).

Modify print plotted variables

In the terminal sections of the subprocesses, calls to OUTPLT are given. The calls with a '1' as the first argument define a list of variables to be print plotted. The call to OUTPLT with '4', '5', '6', or '7' as the first argument actually creates the print plot for the selected variables. This has been described in more detail in Chapter 3. Variables can simply be modified or added in this section. Up to 25 variables can be print plotted in the same graph.

7 Differences CSMP and FORTRAN versions of MACROS

In the preceding Chapters we have discussed the differences in structure between the CSMP and FORTRAN versions of MACROS. At lower levels however, modifications were also necessary. Sorting of statements in a proper order for calculation is a major difference in the appearance of the programs. The sequence of the program calculations had to be changed since FORTRAN does not sort its statements like CSMP does. In this Chapter the following further differences between CSMP and FORTRAN are treated:

- emulated CSMP functions,
- calculations,
- MACROS library,

Because almost all the program lines of the CSMP modules (not the MACROS library) are put in the ITASK=2 and ITASK=3 sections of the FORTRAN subroutines, most of the differences in the calculations and those due to the emulation of CSMP functions are in those sections. Another source of differences are the AMAX1, AMIN1, and AMOD intrinsic functions. They have been replaced by the generic FORTRAN-77 functions MAX, MIN, and MOD.

Differences in CSMP functions due to emulation in FORTRAN.

AFGEN

- The AFGEN function contains three arguments in FORTRAN instead of two in CSMP (the number of elements of the function is added to the parameter list).
- error and warning messages are different.

AND

The CSMP function is emulated by the FORTRAN function REAAND (REAl AND). The name has been changed to avoid confusion with the logical operator .AND. .

INTGRL

The INTGRL statement is very different due to the structural differences between the CSMP and FORTRAN versions of MACROS. The function has lost much of its function because it does nothing more than: new_state = old_state+rate*DELT. The function has been retained in the program because it helps to distinguish between rate and state variables.

LIMIT

Unlike in CSMP a check is carried out that the lower defined lower limit really is less than the

upper limit.

NOR

The CSMP function is emulated by the FORTRAN function REANOR (REAl NOR). The name has been changed to avoid confusion with the logical operator .OR. .

Differences in the calculations

subroutine L1DT

The calculation of RSLL (line 20, module L2C) has been changed using a temporary variable RSLLT. This avoids confusion in the arguments of the LIMIT function.

old:

```
RSLL =LIMIT(RSLLM,2000.,(CO2E-CO2I)/(PLNA+1.E-10)*...
      (68.4*24.0/1.6)-RSBL-RSTL)
```

new:

```
RSLLT = (CO2E-CO2I)/(PLNA+1.E-10)*(68.4*24.0/1.6)-RSBL-RSTL
RSLL  = LIMIT (RSLLM, 2000., RSLLT)
```

subroutine L1QM

The calculation of CAGCR (line 58, module L1Q) has been changed using a temporary variable WARG.

old:
```
CAGCR = LIMIT(0.,(WAR-0.05*WLV)/(DELT*FADL),(WAR-0.05*WLV)*1.5)
```

new:
```
WARG  = MAX (0.0, (WAR-0.05*WLV)/(DELT*FADL))
CAGCR = LIMIT (0.0, WARG, (WAR-0.05*WLV)*1.5)
```

subroutine L2DT

Identical modifications as in subroutine L1DT.

subroutines DRL2SS and DRL2SU

Additional code has been added to create a standard interface to the plant routines. The lines of module L2SS pertaining to soil water however, have not been changed in subroutine DRL2SS.

Modifications in the MACROS library

A general modification is that all functions have been defined REAL explicitly in the function definition, and SAVE statements are included in each subroutine and function. The library is still compatible with the CSMP modules except for the SAVE statements. These have to be removed in each subroutine and function (put comment character '*' or 'C' in first column).

FUCCHK, FUWCHK and SUERRM

Modifications in the WRITE statements were introduced because unit=6 cannot be used for writing messages in standard FORTRAN. The routine is 'made aware' of the kind of program that is calling it (either CSMP or FORTRAN), using the COMMON block. A run number is added to the messages if the routines are called from FORTRAN.

FUTP

The function FUTP (air temperature as a function of time of the day) could not be used because weather variables in the FORTRAN version of MACROS are not arrays but single variables. Therefore a similar function (FUTP2) was created that uses single variables (FUTP has been retained in the MACROS library).

SUASTR

The variable RDTM in the call to SUASTC has been replaced by '1.'. RDTM is not defined in SUASTR whereas in SUASTC a range check is carried out on the value of RDTM. This causes an error with compilers that do not initialize variables at zero.

SUINTG

A number of lines have been modified because they contained a bug. The first line of:

```
IF(DHH(JTOT).LT.-TINY.OR.JTOT.EQ.0) THEN      <- contains error
   CONTINUE
ELSE
   IX    =INXSAT(JTOT,1)-1
   IF(IX.LT.1) THEN
      CONTINUE
   ELSEIF(FLX(IX+1).GT.-0.1) THEN
      CONTINUE
   ELSEIF(FLXSQ2.LE.0.) THEN
      CONTINUE
   ELSE
      MSAL  =TKL(IX)/2.
      CALL SUWCMS(IX,0,WIX2,MSAL)
      MSACT =MS(IX)
      CALL SUWCMS(IX,0,WIX1,MSACT)
```

```
            IF(WIX2.LE.WIX1) THEN
              CONTINUE
            ELSE
              DLIM=-(WIX2-WIX1)*TKL(IX)/FLX(IX+1)
              DT=AMIN1(DT,AMAX1(DLIM,TINY))
            ENDIF
         ENDIF
      ENDIF
```

contains an error because if JTOT is zero, DHH(JTOT) does not exist. Valid subscripts in this case are from one to ten ! This has been replaced by:

```
      IF (JTOT.GT.0) THEN
         IF (DHH(JTOT).GE.-TINY) THEN
            IX    =INXSAT(JTOT,1)-1
            IF(IX.LT.1) THEN
               CONTINUE
            ELSEIF(FLX(IX+1).GT.-0.1) THEN
               CONTINUE
            ELSEIF(FLXSQ2.LE.0.) THEN
               CONTINUE
            ELSE
               MSAL  =TKL(IX)/2.
               CALL SUWCMS(IX,0,WIX2,MSAL)
               MSACT =MS(IX)
               CALL SUWCMS(IX,0,WIX1,MSACT)
               IF(WIX2.LE.WIX1) THEN
                 CONTINUE
               ELSE
                 DLIM  =-(WIX2-WIX1)*TKL(IX)/FLX(IX+1)
                 DT    =AMIN1(DT,AMAX1(DLIM,TINY))
               ENDIF
            ENDIF
         ENDIF
      ENDIF
```

SUSLIN

Minor modifications have been made that adjust the unit number in the WRITE statement depending on whether it is used from CSMP or FORTRAN.

8 Installing the FORTRAN MACROS program

Requirements to run the FORTRAN MACROS program

There are few requirements to run the MACROS program. Because the program has been developed, using standard FORTRAN compilers, on a VAX 8350, Atari 520ST+, Apple Macintosh II, and an IBM compatible, any standard FORTRAN-77 compiler should be able to compile the program succesfully.

The minimum RAM memory requirement is dependent on the computer and the compiler. A minimum of 512 kb is recommended on any machine.

If you intend serious development work with the MACROS program or any other FORTRAN program, we recommend the use of the FORCHECK program (see references). FORCHECK has much stronger syntax, variable declaration, argument passing and standard FORTRAN checking capacities than most compilers and will save a lot of time in debugging any FORTRAN program.

You are advised to use Microsoft FORTRAN 4.10 or any later version. Some batch files and object libraries are provided on the floppies to be used with this compiler. Note that in the appendices complete source listings of all the relevant programs are provided.

Contents of the disks:

Disk 1:

A:\SYSSARP\PROGDAT\	MACROS programs and data files,
A:\SYSSARP\FORTRAN\	MS compiler batch files, LIST.COM, CHKFLOAT.EXE and RECOVER.FOR and RECOVER.EXE programs,
A:\SYSSARP\SUBFUN\MACROS\	Sources of MACROS library of subroutines and functions,
A:\SYSSARP\WEATHER\	Four weather data files from the Philippines:

Country name	Station name	Country code	Station code	Year
Philippines	IRRI wet station site	PHIL	1	1984
Philippines	IRRI dry station site	PHIL	2	1984
Philippines	Los Baños (UPLB)	PHIL	4	1984
Philippines	From PdV et al. (1989)	PHIL	99	1984

Disk 2:

A:\SYSSARP\SUBFUN\TTUTIL\	Sources of utility library TTUTIL,
A:\SYSSARP\SUBFUN\WEATHER\	Source of the weather data reading program,
A:\MANUAL.TXT	The file 'MANUAL.TXT' contains the unformatted text of the first 9 Chapters of the FORTRAN MACROS documentation. The appendices are not in this text file because they can be found in the FORTRAN source files (on disk 2 and 3).

Disk 3:

A:\SYSSARP\FORTRAN\	MACROS, TTUTIL, and WREAD object libraries for Microsoft FORTRAN 4.1 or higher.

You are suggested to create the same directory structure on the hard disk of the machine you will be working on and copy the files manually into these directories (except MANUAL.TXT).

If you are short of disk space you could omit the A:\SYSSARP\SUBFUN\MACROS\, A:\SYSSARP\SUBFUN\TTUTIL\, and A:\SYSSARP\SUBFUN\WEATHER\, directories because these contain source files that are not used during normal work.

You can also install the set of programs and datafiles by inserting Disk 1 and typing the following commands (supposing the floppy disk is A:):

C:\> CD \	<- moves to root directory of hard disk,
C:\> COPY A:\INSTALL.BAT	<- copies install program to hard disk,
C:\> INSTALL A:	<- runs install batch program which installs files from A:.

If the floppy disk drive has a name different from A: the procedure is essentially the same except that you have to replace A: with the name of the disk drive.

Because there is no way in MS-DOS to check if a directory is present, a message such as 'Directory already exists', or 'Unable to create directory', may occur if one or more of the directories already exist. This will not do harm to the installation process however.

LIST, CHKFLOAT and RECOVER

LIST is a shareware program to display text files, CHKFLOAT is a program to check the proper functioning of a math coprocessor on IBM compatibles (self explanatory), and RECOVER is necessary to create an output file if the simulation was not terminated properly.

Working with Microsoft FORTRAN on IBM compatibles

If you will be working with the Microsoft FORTRAN compiler (4.1 or higher), carry out the following installation procedure:

1) Install the compiler and the FORTRAN library (using the SETUP program from the first disk supplied by Microsoft) **both** on the directory \SYSSARP\FORTRAN\. Create the directory \SYSSARP\TMP\ to store temporary compiler and linker files.

2) Build a library with the following characteristics (also using the SETUP program from the first disk): large memory model, floating point emulator, leave out C and MS FORTRAN 3.30 compatibility. This library will have the name: LLIBFORE.LIB.

3) Add in the AUTOEXEC.BAT file C:\SYSSARP\FORTRAN; to the PATH statement, preferably at the beginning as files will be found faster.

4) Restart your PC.

5) If during compilation and/or linking of the model, you get the message: Out of environment space, (this may be displayed only for a short time) you may have to increase the 'environment space' of the PC. This can be done by adding (or changing) the following line in the CONFIG.SYS:

   SHELL=C:\COMMAND.COM /e:3000 c:\ /p

   Restart your PC again.

   If this does not solve the 'Out of environment' problem, you have to shorten the PATH statement and/or remove some SET statements in the AUTOEXEC.BAT file.

6) Make your own directory from which you will start your work and copy the programs and data files you want to work with from C:\SYSSARP\PROGDAT (this directory is your backup of the original programs).

7) Compilation of FORTRAN files, linking with the libraries, and execution can now be done with the following commands (do not add extension '.FOR'):

FOREXE <file>	plain compilation, linking and execution of ' file.for ',
FOREXE <file> c	compilation only of ' file.for ',
FOREXE <file> s	syntax check and listing of ' file.for '.

   Compilation, linking with the libraries, and debugging can be done with a similar

command:

FOREXED <file>	compilation, linking and debugging of ' file.for ' (requires much more memory !),
FOREXED <file> c	compilation with debug options only of ' file.for '.

After the FOREXE commands, execution can also be invoked by typing the name of the program. Typing the name of the program after FOREXED will execute the program without debugging. Renewed debugging can be invoked by typing:

CV <file>	(CV stands for CodeView, the debugger of the Microsoft languages.)

The executable files created by the FOREXE and FOREXED commands will run on both XT and AT computers. They do not require a coprocessor but will use one if present. Use the CHKFLOAT program to check if a coprocessor is present and if it works properly.

9 References and further reading

IBM, 1975. Continuous System Modeling Program III. General system information manual (GH19-7000) and users manual (SH19-7001-2). IBM Data Processing Division, White Plains, New York.

FORCHECK: A FORTRAN-77 Verifier and Programming aid, E.W. Kruyt, Dept. of Physiology. Leiden University. PO Box 9604. 2300 RC Leiden. The Netherlands.

Haar, L.G.J. ter, 1983. FORTRAN 77, programmers pocket guide. Nederlands Normalisatie Instituut. 47 pp.

Hahn, B.D., Problem solving with FORTRAN-77. Edward Arnold ltd. London. 247 pp.

Kraalingen, D.W.G. van, and Rappoldt, C., 1989. Subprograms in simulation models. Simulation Report CABO-TT nr. 18. Centre for Agrobiological Research and Dept. of Theoretical Production Ecology. Wageningen. The Netherlands. 54 pp. (available on request).

Kraalingen, D.W.G. van, Stol, W., Uithol, P.W.J., Verbeek, M., 1990. User Manual of CABO/TPE Weather System. CABO/TPE internal communication. 27 pp. (available on request).

Meissner, L.P., Organick, E.I., 1984. FORTRAN 77, featuring structured programming. Addison-Wesley publishing company. 500 pp.

Penning de Vries, F.W.T., Jansen, D.M., ten Berge, H.F.M., and Bakema, A., 1989. Simulation of Ecophysiological Processes of Growth in Several Annual Crops. Simulation Monograph 29. PUDOC Wageningen and IRRI, Los Baños. 271 pp.

Rappoldt, C., and Kraalingen, D.W.G. van, 1990. FORTRAN utility library TTUTIL.Simulation Report CABO-TT nr. 20. Centre for Agrobiological Research and Dept. of Theoretical Production Ecology. Wageningen. The Netherlands. 54 pp. (available on request).

Wagener, J.L., 1980. FORTRAN 77, principles of programming. John Wiley & sons. New York. 370 pp.

Appendix A: Modules and data files

In this appendix the listings are given of the main program (called MACROS) the timer data file, the plant subroutines and plant data file, the soil water subroutines and data file, and the potential soil evaporation subroutine.

These programs can be found on disk 1, directory A:\SYSSARP\PROGDAT\.
The file order in this appendix is as follows:

Main program:

MACROS.FOR	main program,
TIMER.DAT	data file containing time and weather variables,

Plant subroutines:

L1D.FOR	potential production,
L1DT.FOR	potential production and transpiration,
L1QM.FOR	potential production with quarter day time steps, and morphological development of tillers,
L2DT.FOR	water-limited production,
PLANT.DAT	data file with parameters and initial values of the states for a rice crop,

Soil water balance subroutines:

DRL2SS.FOR	water balance for soils with impeded drainage,
DRL2SU.FOR	water balance for free draining soils,
SOIL.DAT	data file with soil parameters and initial water contents,

Potential soil evaporation:

SUPSEV.FOR	potential soil evaporation with standing crop.

File: A:\SYSSARP\PROGDAT\MACROS.FOR

```
*-----------------------------------------------------------------------*
*                                                                       *
*                          M A C R O S                                  *
*                  Modules for Annual CROp Simulation                   *
*                                                                       *
*                              Version 1                                *
*                            January 1990                               *
*                       D.W.G. van Kraalingen                           *
*                      F.W.T. Penning de Vries                          *
*                                                                       *
* This is the  main program of the FORTRAN version of  the IRRI/SARP crop *
* growth  simulator  MACROS.  This version is based on  the  CSMP program *
* MACROS  which, together with the  scientific  background, is documented *
* in:                                                                   *
* Penning de Vries, F.W.T., Jansen, D.M., ten Berge, H.F.M.,  and Bakema, *
* A., 1989. Simulation of Ecophysiological Processes of Growth in Several *
* Annual Crops. Simulation Monograph 29. PUDOC Wageningen. 271 pp.      *
*                                                                       *
* The documentation and  listing of this program can  be obtained by sub- *
* mitting  a request to IRRI/MCD,  c/o F. Penning de Vries, P.O. Box 933, *
* Manila,  Philippines,  or  CABO/SARP, P.O. Box 14, 6700 AA  Wageningen, *
* The Netherlands.                                                      *
*                                                                       *
*-----------------------------------------------------------------------*
      PROGRAM MACROS
      IMPLICIT REAL (A-Z)

*     standard declarations
      INTEGER       ITASK, INSETS,    IRUN, I
      INTEGER      IUNITR, IUNITT,  IUNITO, IUNITP
      INTEGER      ISTAT1, ISTAT2,   IDATE, IDATEH, IYEAR, ISTN
      LOGICAL      OUTPUT, TERMNL,  WTRMES,  WTROK
      INTEGER      INCNTR,  ICNTR
      PARAMETER  (INCNTR=1)
      CHARACTER*6 CNTR(INCNTR)
      CHARACTER*1 DUMMY
      CHARACTER*80 WTRDIR,  FILER,   FILET,  FILEO, FILEP

*     special declaration required to write warnings to output file
      COMMON /LOGCOM/ IUNITO, TIME, IRUN

*     declarations for use of DRL2SS and DRL2SU
*     line 1: file I/O unit, number of layers, and maximum number of layers
*       ,, 2: maximum number of layers
*       ,, 3: volumetric water contents
*       ,, 4: thickness, transpiration rates and water contents
*       ,, 5: string that holds name of soil data file

      INTEGER IUNITS, NL, NLMAX
      PARAMETER (NLMAX=10)
      REAL    WCWP(NLMAX),  WCFC(NLMAX),  WCST(NLMAX)
      REAL     TKL(NLMAX),  TRWL(NLMAX), WCLQT(NLMAX)
      CHARACTER*80 FILES

*     Unit numbers for rerun (R), timer (T), output (O), plant data (P)
*     and soil data (S) files. WTRMES flags any messages from
*     the weather system

      IUNITR = 20
      IUNITT = 30
      IUNITO = 40
      IUNITP = 50
      IUNITS = 60
      WTRMES = .FALSE.
```

```
*     idem path for weather data, country name for weather data
*     and file names
      WTRDIR  = 'C:\SYSSARP\WEATHER\'
      CNTR(1) = 'PHIL'
      FILER   = 'RERUNS.DAT'
      FILET   = 'TIMER.DAT'
      FILEO   = 'MACROS.OUT'
      FILEP   = 'PLANT.DAT'
      FILES   = 'SOIL.DAT'

*     open output file, read number of rerun sets
      CALL FOPEN (IUNITO, FILEO, 'NEW', 'DEL')
      CALL RDSETS (IUNITR, IUNITO, FILER, INSETS)

      DO 10 I=0,INSETS

      IRUN = I+1
      WRITE (*,'(A)') ' '

*     select data set
      CALL RDFROM (I, .TRUE.)

*     ----------------------
*     Initialization section
*     ----------------------

      ITASK  = 1
      TERMNL = .FALSE.

      CALL RDINIT (IUNITT, IUNITO, FILET)
      CALL RDSREA ('DATEB', DATEB)
      CALL RDSREA ('FINTIM', FINTIM)
      CALL RDSREA ('PRDEL', PRDEL)
      CALL RDSREA ('DELT', DELT)
      CALL RDSINT ('ICNTR', ICNTR)
      CALL RDSINT ('IYEAR', IYEAR)
      CALL RDSINT ('ISTN', ISTN)
      CLOSE (IUNITT, STATUS='DELETE')

      CALL TIMER (ITASK, DATEB, DELT, PRDEL, FINTIM,
     &            IYEAR,  TIME, DATE, IDATE, TERMNL, OUTPUT)
      CALL OUTDAT (ITASK, IUNITO, 'TIME', TIME)

      CALL STINFO (1101, WTRDIR, ' ', CNTR(ICNTR), ISTN, IYEAR,
     &             ISTAT1, LONG, LAT, ELV, A, B)
      CALL WEATHR (IDATE, ISTAT2, RDTM, TPL, TPH, HUAA, WDS, RAIN)

      WTRMES = WTRMES .OR. (ISTAT1.NE.0) .OR. (ISTAT2.NE.0)
      WTROK  = (ISTAT1.EQ.0).AND.((ISTAT2.GE.0).OR.(ISTAT2.LT.-111111))
      TERMNL = TERMNL.OR..NOT.WTROK

      CALL SUASTR (DATE, LAT, RDTC, DLA, DLP)

*     < insert water balance call here if required >


*     < insert plant call here >


*     --------------------------
*     Dynamic simulation section
*     --------------------------


20    IF (.NOT.TERMNL) THEN

      WRITE (*,'(A,I3,A,I5,A,F7.2)')
     &  ' Run:', IRUN, ', Year:', IYEAR, ', Date:', DATE
```

```
*
*     Integration of rates section
*
      ITASK = 3

*     < insert plant call here >


*     < insert water balance call here if required >


      ITASK = 2

*
*     Calculation of driving variables section
*

      CALL OUTDAT (ITASK, IUNITO, 'TIME', TIME)
      IF (OUTPUT.OR.TERMNL) CALL OUTDAT (ITASK, IUNITO, 'DATE', DATE)

      CALL STINFO (1101, WTRDIR, ' ', CNTR(ICNTR), ISTN, IYEAR,
     &             ISTAT1, LONG, LAT, ELV, A, B)
      CALL WEATHR (IDATE, ISTAT2, RDTM, TPL, TPH, HUAA, WDS, RAIN)

      WTRMES = WTRMES .OR. (ISTAT1.NE.0) .OR. (ISTAT2.NE.0)
      WTROK  = (ISTAT1.EQ.0).AND.((ISTAT2.GE.0).OR.(ISTAT2.LT.-111111))
      TERMNL = TERMNL.OR..NOT.WTROK

*     The next continuous block of statements is only necessary to run L1QM
*     and L2SU under the main program. (weather of previous (P) and
*     next day (N))

      IDATEH = MAX (IDATE-1, 1)
      CALL WEATHR (IDATEH, ISTAT1,
     &             RDTMP, TPLP, TPHP, HUAAP, WDSP, RAINP)
      IDATEH = MIN (IDATE+1, 365)
      CALL WEATHR (IDATEH, ISTAT2,
     &             RDTMN, TPLN, TPHN, HUAAN, WDSN, RAINN)
      WTRMES = WTRMES .OR. (ISTAT1.NE.0) .OR. (ISTAT2.NE.0)
      WTROK  = WTROK.OR.
     &       ((ISTAT1.EQ.0).AND.((ISTAT2.GE.0).OR.(ISTAT2.LT.-111111)))
      TERMNL = TERMNL.OR..NOT.WTROK

      CALL SUASTR (DATE, LAT, RDTC, DLA, DLP)

*
*     Calculation of rates section
*

*     < insert plant call here >


*     < insert potential soil evaporation call here if required >


*     < insert water balance call here if required >


*     time update, check for FINTIM and OUTPUT
      CALL TIMER (ITASK, DATEB, DELT, PRDEL, FINTIM,
     &            IYEAR,  TIME, DATE, IDATE, TERMNL, OUTPUT)

      GOTO 20
      END IF
```

```
*     ----------------
*     Terminal section
*     ----------------

      ITASK = 4

*     Generate output file using normal table format
*     and delete temporary output file.

      CALL OUTDAT (4, 20, 'Macros simulation model',0.)

*     < insert plant call here >


*     < insert water balance call here if required >


*     delete temporary output file
      CALL OUTDAT (99, 0, ' ', 0.)

10    CONTINUE

*     delete temporary rerun file
      IF (INSETS.GT.0) CLOSE (IUNITR, STATUS='DELETE')

      IF (WTRMES) THEN
         WRITE (*,'(A,/,A)')
     &     ' There have been errors and/or warnings from',
     &     ' the weather system, check file WEATHER.LOG'
         WRITE (IUNITO,'(A,/,A)')
     &     ' There have been errors and/or warnings from',
     &     ' the weather system, check file WEATHER.LOG'
         WRITE (*,'(A)') ' Press <RETURN>'
         READ (*,'(A)') DUMMY
      END IF

      STOP
      END
```

File: A:\SYSSARP\PROGDAT\TIMER.DAT

```
*
* Weather variables and time variables
*

ICNTR  = 1         ! Country number of weather data
ISTN   = 99        ! Station number of weather data
IYEAR  = 1984      ! Year of weather data
DATEB  = 197.      ! Start date of simulation
FINTIM = 1000.     ! Finish time of simulation
PRDEL  = 5.        ! Time between consecutive outputs to file
DELT   = 1.        ! Time step of integration
```

File: A:\SYSSARP\PROGDAT\L1D.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE L1D                                                       *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine simulates potential production of crops.    *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                    units  class  *
*  ----   ---- -------                                    -----  -----  *
*  control                                                             *
*  ITASK   I4  Determines action of the subroutine,                    *
*              1=initialization, 2=rate calculation,                   *
*              3=integration, 4=terminal                    -     C,I   *
*  IUNITP  I4  Unit number of plant data file,              -     C,IN  *
*  IUNITO  I4  Unit number of output file                   -     C,IN  *
*  FILEP   C*  Name of plant data file                      -     C,IN  *
*  OUTPUT  L4  Flag that indicates if output to file is                *
*              required,                                    -     C,I   *
*  TERMNL  L4  Flag that indicates if simulation should                *
*              terminate                                    -    C,I,O  *
*  timing                                                              *
*  TIME    R4  Time of simulation                           d      T    *
*  DATE    R4  Day number of simulation                     d      T    *
*  DELT    R4  Time step of integration                     d      T    *
*                                                                      *
*  environment of plant                                                *
*  LAT     R4  Latitude of weather station               degrees   I    *
*  DLP     R4  Photoperiodic daylength of current day      h       I    *
*  RDTM    R4  Global radiation                         J/m2/d     I    *
*  TPL     R4  Minimum temperature                 degrees Celsius I    *
*  TPH     R4  Maximum temperature                 degrees Celsius I    *
*                                                                      *
*  FATAL ERROR CHECKS (execution terminated, message):                 *
*  DELT < 1                                                            *
*  Certain sequences of ITASK, see subroutine CHKTSK                   *
*  Carbon balance check                                                *
*                                                                      *
*  SUBROUTINES and FUNCTIONS called:                                   *
*  from TTUTIL : CHKTSK,  ERROR, RDINIT, RDSREA, RDSINT, RDAREA, OUTCOM,*
*                OUTDAT, OUTPLT,  AFGEN,   INSW, INTGRL,  LIMIT        *
*  from MACROS.LIB: FUPHOT, FUCCHK                                     *
*                                                                      *
*  FILE usage : - Plant definition file FILEP opened and closed for    *
*                 ITASK=1, unit numbers used are IUNITP and IUNITP+1   *
*               - Output file with unit IUNITO for output and warnings *
*-----------------------------------------------------------------------*

*     CALL L1D (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
*    &           TIME,   DATE,   DELT,
*    &            LAT,    DLP,
*    &           RDTM,    TPL,    TPH)

      SUBROUTINE L1D (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
     &                TIME,   DATE,   DELT,
     &                 LAT,    DLP,
     &                RDTM,    TPL,    TPH)
      IMPLICIT REAL (A-Z)

*     Formal parameters

      INTEGER   ITASK, IUNITP, IUNITO
      LOGICAL  OUTPUT, TERMNL
      CHARACTER*(*) FILEP
```

```
*     Standard local declarations

      INTEGER ILAR, ITOLD
      PARAMETER (ILAR=20)

      REAL     PLMTT(ILAR), PLETT(ILAR)
      INTEGER IPLMTN      ,IPLETN

      REAL     CALVT(ILAR), CASTT(ILAR), CASST(ILAR)
      INTEGER ICALVN      ,ICASTN      ,ICASSN

      REAL     LLVT(ILAR), LRTT(ILAR)
      INTEGER ILLVN      ,ILRTN

      REAL     DRVTT(ILAR), DRRTT(ILAR), DRDT(ILAR), SLT(ILAR)
      INTEGER IDRVTN      ,IDRRTN      ,IDRDN      ,ISLN

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:              Old task:
*           initialization         terminal
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done i.e. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('L1D', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L1D, Plant growth at potential production')

*        Initialization section

         IF (DELT.LT.1.0) CALL ERROR ('L1D','DELT too small for L1D')
         CALL RDINIT (IUNITP, IUNITO, FILEP)

*        Initialization of states

         CALL RDSREA ('WLVI', WLVI)
         WLV = WLVI
         CALL RDSREA ('WSTI', WSTI)
         WST = WSTI
         WIR = 0.0
         CALL RDSREA ('WSOI', WSOI)
         WSO = WSOI
         CALL RDSREA ('FEPSO', FEPSO)
         WEPSO = WSO*FEPSO
         WRTI  = WLVI
         WRT   = WLV
         WSS   = WLV+WST+WSO+WIR
         WCR   = WSS+WRT
         WLVD  = 0.0
```

```
      WRTD  = 0.0
      WLVT  = WLV+WLVD
      WLVST = WLVT+WST+WIR
      WLVSO = WLVST+WSO
      HI    = WSO/WSS
      WSTR  = WST+WIR
      PCGT  = 0.0
      PCNT  = 0.0
      RMCT  = 0.0
      RCRT  = 0.0
      CELVN = 0.0
      CALL RDSREA ('SLC', SLC)
      CALL RDAREA ('SLT', SLT, ILAR, ISLN)
      CALL RDSREA ('DSI', DSI)
      DS = DSI
      ALV  = WLV/(SLC*AFGEN (SLT, ISLN, DS))

      CALL RDSREA ('FCLV', FCLV)
      CALL RDSREA ('FCST', FCST)
      CALL RDSREA ('FCSO', FCSO)
      CALL RDSREA ('FCRT', FCRT)

*     Remaining variables to be initialized in state section

      CALL RDSREA ('FSTR', FSTR)
      CALL RDSREA ('FCST', FCST)

*     Variables to be initialized in rate section

      CPEW = 1.0
      DREW = 1.0
      PCEW = 1.0

      CALL RDSREA ('SSC', SSC)
      CALL RDSREA ('PLMXP', PLMXP)
      CALL RDAREA ('PLMTT', PLMTT, ILAR, IPLMTN)
      CALL RDSREA ('PLEI', PLEI)
      CALL RDAREA ('PLETT', PLETT, ILAR, IPLETN)

      CALL RDSREA ('Q10', Q10)
      CALL RDSREA ('TPR', TPR)
      CALL RDSREA ('RMCLV', RMCLV)

      CALL RDAREA ('CASTT', CASTT, ILAR, ICASTN)
      CALL RDAREA ('CASST', CASST, ILAR, ICASSN)
      CALL RDAREA ('CALVT', CALVT, ILAR, ICALVN)

      CALL RDSREA ('CRGLV', CRGLV)
      CALL RDSREA ('CRGST', CRGST)
      CALL RDSREA ('CRGRT', CRGRT)
      CALL RDSREA ('CRGSO', CRGSO)

      CALL RDAREA ('LLVT', LLVT, ILAR, ILLVN)
      CALL RDAREA ('LRTT', LRTT, ILAR, ILRTN)

      CALL RDSREA ('CPGLV', CPGLV)
      CALL RDSREA ('CPGST', CPGST)
      CALL RDSREA ('CPGSO', CPGSO)
      CALL RDSREA ('CPGRT', CPGRT)

      CALL RDAREA ('DRDT',  DRDT, ILAR, IDRDN)
      CALL RDSREA ('DRCV',  DRCV)
      CALL RDAREA ('DRVTT', DRVTT, ILAR, IDRVTN)
      CALL RDSREA ('DRCR',  DRCR)
      CALL RDAREA ('DRRTT', DRRTT, ILAR, IDRRTN)

      CLOSE (IUNITP, STATUS='DELETE')
```

```
      ELSE IF (ITASK.EQ.2) THEN

*        rate calculation section

         TPAV = (TPL+TPH)/2.0
         TPAD = (TPH+TPAV)/2.0
         RDTM = RDTM*1000.

*        Photosynthesis, gross and net
*        Explanation in sections 2.1, 3.3, 3.4

         SLA  = (WLV+0.5*WST*(SLC/SSC))/ALV
         PLMX = PLMXP*AFGEN (PLMTT, IPLMTN, TPAD)*
     &          LIMIT (200., 600., SLA)/SLC
         PLEA = PLEI*AFGEN (PLETT, IPLETN, TPAD)
         PCGC = FUPHOT (PLMX, PLEA, ALV, RDTM, DATE, LAT)
         PCGW = PCGC*PCEW

*        Maintenance respiration
*        Explanation in section 2.3

         TPEM = Q10**((TPAV-TPR)/10.)
         RMLV = WLV*RMCLV*TPEM*0.75
         RMST = WST*0.010*TPEM+WIR*0.0
         RMRT = WRT*0.015*TPEM
         RMSO = MIN (1000., WSO)*0.015*TPEM

         RMMA = 0.2*PCGW*0.5
         RMCR = RMLV+RMST+RMSO+RMRT+RMMA

*        Carbohydrate available for growth, export
*        Explanation in sections 3.2, 2.4, 2.3, 2.2

         LSTR  = INSW ((AFGEN (CASTT, ICASTN, DS)-0.01), WIR*0.1, 0.0)

         CAGCR = PCGW*0.682-RMCR*0.682+LSTR*1.111*0.947
         CAGSS = CAGCR*AFGEN (CASST, ICASSN, DS)*CPEW
         CAGRT = CAGCR-CAGSS
         CAGLV = CAGSS*AFGEN (CALVT, ICALVN, DS)
         CAGST = CAGSS*AFGEN (CASTT, ICASTN, DS)
         CAGSO = CAGSS-CAGLV-CAGST

         CELV  = PCGW-(RMLV+RMST+0.5*RMMA)

*        Growth rates and loss rates
*        Explanation in sections 2.4, 3.2, 2.2

         GLV = CAGLV/CRGLV
         GST = CAGST/CRGST
         GRT = CAGRT/CRGRT
         GSO = CAGSO/CRGSO

         LLV = WLV*AFGEN (LLVT, ILLVN, DS)
         LRT = WRT*AFGEN (LRTT, ILRTN, DS)

*        Respiration due to growth
*        Explanation in section 2.4

         RGLV = GLV*CPGLV
         RGST = GST*CPGST
         RGSO = GSO*CPGSO
         RGRT = GRT*CPGRT
         RLSR = LSTR*1.111*0.053*1.467
         RGCR = RGLV+RGST+RGSO+RGRT+RLSR
         RSH  = RMLV+RMST+RMSO+RMMA+RGLV+RGST+RGSO+RLSR

*        Leaf area
*        Explanation in section 3.3
```

```
          SLN = SLC*AFGEN (SLT, ISLN, DS)
          GLA = GLV/SLN
          LLA = LLV/SLA
          GSA = 0.5*GST/SSC

*         Phenological development of the crop
*         Explanation in section 3.1

          DRED = AFGEN (DRDT, IDRDN, DLP)
          DRV  = DRCV*DRED*DREW*AFGEN (DRVTT, IDRVTN, TPAV)
          DRR  = DRCR*AFGEN (DRRTT, IDRRTN, TPAV)

*         output of states and rates only if it is required

          IF (OUTPUT .OR. TERMNL) THEN
             CALL OUTDAT (2, 0, 'WLV',  WLV)
             CALL OUTDAT (2, 0, 'WLVT', WLVT)
             CALL OUTDAT (2, 0, 'WLVST',WLVST)
             CALL OUTDAT (2, 0, 'WLVSO',WLVSO)
             CALL OUTDAT (2, 0, 'WST',  WST)
             CALL OUTDAT (2, 0, 'WIR',  WIR)
             CALL OUTDAT (2, 0, 'WSO',  WSO)
             CALL OUTDAT (2, 0, 'WRT',  WRT)
             CALL OUTDAT (2, 0, 'GLV',  GLV)
             CALL OUTDAT (2, 0, 'GST',  GST)
             CALL OUTDAT (2, 0, 'GSO',  GSO)
             CALL OUTDAT (2, 0, 'GRT',  GRT)
             CALL OUTDAT (2, 0, 'SLA',  SLA)
             CALL OUTDAT (2, 0, 'PLMX', PLMX)
             CALL OUTDAT (2, 0, 'ALV',  ALV)
             CALL OUTDAT (2, 0, 'DS',   DS)
             CALL OUTDAT (2, 0, 'TPAV', TPAV)
             CALL OUTDAT (2, 0, 'RDTM', RDTM)
             CALL OUTDAT (2, 0, 'PCGT', PCGT)
             CALL OUTDAT (2, 0, 'RCRT', RCRT)
             CALL OUTDAT (2, 0, 'RMCT', RMCT)
          END IF

       ELSE IF (ITASK.EQ.3) THEN

*         integration section

*         Weights of crop components
*         Explanation in sections 3.2, 2.2, 3.4

          WLV = INTGRL (WLV, (GLV-LLV), DELT)

          WST = INTGRL (WST, (GST*(1.0-FSTR)), DELT)

          WIR = INTGRL (WIR, (GST*(FSTR*(FCST/0.444))-LSTR), DELT)
          WSO = INTGRL (WSO, GSO, DELT)

          WEPSO = WSO*FEPSO

          WRT   = INTGRL (WRT, (GRT-LRT), DELT)

          WSS   = WLV+WST+WSO+WIR
          WCR   = WSS+WRT

          WLVD  = INTGRL (WLVD, LLV, DELT)
          WRTD  = INTGRL (WRTD, LRT, DELT)
          WLVT  = WLV+WLVD
          WLVST = WLVT+WST+WIR
          WLVSO = WLVST+WSO
          HI    = WSO/WSS
          WSTR  = WST+WIR
```

```
*       Photosynthesis, gross and net
*       Explanation in sections 2.1, 3.3, 3.4

        PCGT = INTGRL (PCGT, PCGW, DELT)
        PCNT = INTGRL (PCNT, (PCGW-(RMCR+RGCR)), DELT)

*       Respiration
*       Explanation in sections 2.4, 2.3

        RMCT = INTGRL (RMCT, RMCR, DELT)
        RCRT = INTGRL (RCRT, (RMCR+RGCR),DELT)

*       Carbohydrate available for growth, export
*       Explanation in sections 3.2, 2.4, 2.3, 2.2

        CELVN = INTGRL (CELVN, INSW (CELV, 1.0 ,-CELVN/DELT), DELT)

*       Leaf area
*       Explanation in section 3.3

        ALV = INTGRL (ALV, (GLA-LLA+GSA), DELT)

*       Phenological development of the crop
*       Explanation in section 3.1

        DS = INTGRL (DS, INSW ((DS-1.0), DRV, DRR), DELT)
```

```
*       Carbon balance check
*       Explanation in section 3.4

        CKCIN = (WLV-WLVI)*FCLV+(WST-WSTI)*FCST+
     &          (WSO-WSOI)*FCSO+(WRT-WRTI)*FCRT+WIR*0.444
        CKCFL = PCNT*0.2727-(WLVD*FCLV+WRTD*FCRT)
        CKCRD = FUCCHK (CKCIN, CKCFL, TIME)

*       Determine the finish conditions of the simulation

        IF (DS.GE.2.0)     TERMNL = .TRUE.
        IF (CELVN.GE.3.0) TERMNL = .TRUE.

      ELSE IF (ITASK.EQ.4) THEN

*       Define graph for output
        CALL OUTPLT (1, 'WLV')
        CALL OUTPLT (1, 'WLVT')
        CALL OUTPLT (1, 'WLVST')
        CALL OUTPLT (1, 'WLVSO')

*       use common scale, small plot width for output
        CALL OUTPLT (7, 'Macros simulation model')

      END IF

      ITOLD = ITASK

      RETURN
      END
```

File: A:\SYSSARP\PROGDAT\L1DT.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE L1DT                                                      *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine simulates potential production of crops as  *
*           in L1D but also calculates potential crop transpiration.    *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                  units  class    *
*  ----   ---- -------                                  -----  -----    *
*  control                                                              *
*  ITASK   I4  Determines action of the subroutine,                     *
*              1=initialization, 2=rate calculation,                    *
*              3=integration, 4=terminal                  -      C,I    *
*  IUNITP  I4  Unit number of plant data file,            -      C,IN   *
*  IUNITO  I4  Unit number of output file                 -      C,IN   *
*  FILEP   C*  Name of plant data file                    -      C,IN   *
*  OUTPUT  L4  Flag that indicates if output to file is                 *
*              required,                                  -      C,I    *
*  TERMNL  L4  Flag that indicates if simulation should                 *
*              terminate                                  -     C,I,O   *
*                                                                       *
*  timing                                                               *
*  TIME    R4  Time of simulation                         d       T     *
*  DATE    R4  Day number of simulation                   d       T     *
*  DELT    R4  Time step of integration                   d       T     *
*                                                                       *
*  environment of plant                                                 *
*  LAT     R4  Latitude  of weather station            degrees    I     *
*  ELV     R4  Elevation of weather station               m       I     *
*  DLA     R4  Daylength of current day                   h       I     *
*  DLP     R4  Photoperiodic daylength of current day     h       I     *
*  RDTM    R4  Global radiation                         J/m2/d    I     *
*  TPL     R4  Minimum temperature                degrees Celsius I     *
*  TPH     R4  Maximum temperature                degrees Celsius I     *
*  HUAA    R4  Vapour pressure                            kPa     I     *
*  WDS     R4  Average wind speed                         m/s     I     *
*                                                                       *
*  FATAL ERROR CHECKS (execution terminated, message):                  *
*  DELT < 1                                                             *
*  Certain sequences of ITASK, see subroutine CHKTSK                    *
*  Carbon balance check                                                 *
*                                                                       *
*  SUBROUTINES and FUNCTIONS called:                                    *
*  from TTUTIL : CHKTSK,  ERROR, RDINIT, RDSREA, RDSINT, RDAREA, OUTCOM,*
*                OUTDAT, OUTPLT,  AFGEN,   INSW, INTGRL,  LIMIT         *
*  from MACROS.LIB: FUPHOT, FUCCHK, SUEVTR,  FURSC,                     *
*                                                                       *
*  FILE usage : - Plant definition file FILEP opened and closed for     *
*                 ITASK=1, unit numbers used are IUNITP and IUNITP+1    *
*               - Output file with unit IUNITO for output and warnings  *
*-----------------------------------------------------------------------*

*      CALL L1DT (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
*     &             TIME,   DATE,   DELT,
*     &              LAT,    ELV,    DLA,   DLP,
*     &             RDTC,   RDTM,    TPL,   TPH, HUAA, WDS)

      SUBROUTINE L1DT (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
     &                 TIME,   DATE,   DELT,
     &                  LAT,    ELV,    DLA,   DLP,
     &                 RDTC,   RDTM,    TPL,   TPH, HUAA, WDS)
      IMPLICIT REAL (A-Z)
```

```
*     Formal parameters

      INTEGER  ITASK, IUNITP, IUNITO
      LOGICAL OUTPUT, TERMNL
      CHARACTER*(*) FILEP

*     Standard local parameters as in L1D

      INTEGER ILAR, ITOLD
      PARAMETER (ILAR=20)

      REAL     PLMTT(ILAR), PLETT(ILAR)
      INTEGER IPLMTN      ,IPLETN

      REAL     CALVT(ILAR), CASTT(ILAR), CASST(ILAR)
      INTEGER ICALVN      ,ICASTN      ,ICASSN

      REAL     LLVT(ILAR), LRTT(ILAR)
      INTEGER ILLVN      ,ILRTN

      REAL     DRVTT(ILAR), DRRTT(ILAR), DRDT(ILAR), SLT(ILAR)
      INTEGER IDRVTN      ,IDRRTN      ,IDRDN      ,ISLN

*     Extra declarations necessary for L1DT

      REAL     PLHTT(ILAR)
      INTEGER IPLHTN

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:              Old task:
*           initialization         terminal
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done i.e. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('L1DT', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L1DT, Plant growth at potential production,')
         CALL OUTCOM ('with crop transpiration calculations')

*        Initialization section

         IF (DELT.LT.1.0) CALL ERROR ('L1DT','DELT too small for L1DT')
         CALL RDINIT (IUNITP, IUNITO, FILEP)

*        Initialization of states as in L1D

         CALL RDSREA ('WLVI', WLVI)
         WLV = WLVI
```

```
      CALL RDSREA ('WSTI', WSTI)
      WST = WSTI
      WIR = 0.0
      CALL RDSREA ('WSOI', WSOI)
      WSO = WSOI
      CALL RDSREA ('FEPSO', FEPSO)
      WEPSO = WSO*FEPSO
      WRTI  = WLVI
      WRT   = WLV
      WSS   = WLV+WST+WSO+WIR
      WCR   = WSS+WRT
      WLVD  = 0.0
      WRTD  = 0.0
      WLVT  = WLV+WLVD
      WLVST = WLVT+WST+WIR
      WLVSO = WLVST+WSO
      HI    = WSO/WSS
      WSTR  = WST+WIR
      PCGT  = 0.0
      PCNT  = 0.0
      RMCT  = 0.0
      RCRT  = 0.0
      CELVN = 0.0
      CALL RDSREA ('SLC', SLC)
      CALL RDAREA ('SLT', SLT, ILAR, ISLN)
      CALL RDSREA ('DSI', DSI)
      DS = DSI
      ALV  = WLV/(SLC*AFGEN (SLT, ISLN, DS))

      CALL RDSREA ('FCLV', FCLV)
      CALL RDSREA ('FCST', FCST)
      CALL RDSREA ('FCSO', FCSO)
      CALL RDSREA ('FCRT', FCRT)

*     Remaining variables to be initialized in state section as in L1D

      CALL RDSREA ('FSTR', FSTR)
      CALL RDSREA ('FCST', FCST)

*     Remaining variables to be initialized in state section for L1DT

      CALL RDAREA ('PLHTT', PLHTT, ILAR, IPLHTN)
      PLHT = AFGEN (PLHTT, IPLHTN, DS)

*     Variables to be initialized in rate section as in L1D

      CPEW = 1.0
      DREW = 1.0
      PCEW = 1.0

      CALL RDSREA ('SSC', SSC)
      CALL RDSREA ('PLMXP', PLMXP)
      CALL RDAREA ('PLMTT', PLMTT, ILAR, IPLMTN)
      CALL RDSREA ('PLEI', PLEI)
      CALL RDAREA ('PLETT', PLETT, ILAR, IPLETN)

      CALL RDSREA ('Q10', Q10)
      CALL RDSREA ('TPR', TPR)
      CALL RDSREA ('RMCLV', RMCLV)

      CALL RDAREA ('CASTT', CASTT, ILAR, ICASTN)
      CALL RDAREA ('CASST', CASST, ILAR, ICASSN)
      CALL RDAREA ('CALVT', CALVT, ILAR, ICALVN)

      CALL RDSREA ('CRGLV', CRGLV)
      CALL RDSREA ('CRGST', CRGST)
      CALL RDSREA ('CRGRT', CRGRT)
      CALL RDSREA ('CRGSO', CRGSO)
```

```
          CALL RDAREA ('LLVT', LLVT, ILAR, ILLVN)
          CALL RDAREA ('LRTT', LRTT, ILAR, ILRTN)

          CALL RDSREA ('CPGLV', CPGLV)
          CALL RDSREA ('CPGST', CPGST)
          CALL RDSREA ('CPGSO', CPGSO)
          CALL RDSREA ('CPGRT', CPGRT)

          CALL RDAREA ('DRDT',  DRDT, ILAR, IDRDN)
          CALL RDSREA ('DRCV',  DRCV)
          CALL RDAREA ('DRVTT', DRVTT, ILAR, IDRVTN)
          CALL RDSREA ('DRCR',  DRCR)
          CALL RDAREA ('DRRTT', DRRTT, ILAR, IDRRTN)

*         extra for L1DT

          CALL RDSREA ('FIEC',  FIEC)
          CALL RDSREA ('WDLV',  WDLV)

          CLOSE (IUNITP, STATUS='DELETE')

       ELSE IF (ITASK.EQ.2) THEN

*         rate calculation section

          TPAV = (TPL+TPH) /2.0
          TPAD = (TPH+TPAV)/2.0
          RDTM = RDTM*1000.

          CO2E  = 340.*0.88**(ELV/1000.)
          WDSAV = MAX (0.2, WDS)
          WDSAD = 1.33*WDSAV
          VPA   = MIN (FUVP (TPAD), HUAA)

*         Photosynthesis, gross and net
*         Explanation in sections 2.1, 3.3, 3.4

          SLA  = (WLV+0.5*WST*(SLC/SSC))/ALV
          PLMX = PLMXP*AFGEN (PLMTT, IPLMTN, TPAD)*
     &           LIMIT (200., 600., SLA)/SLC
          PLEA = PLEI*AFGEN (PLETT, IPLETN, TPAD)
          PCGC = FUPHOT (PLMX, PLEA, ALV, RDTM, DATE, LAT)

*         Maintenance respiration
*         Explanation in section 2.3

          TPEM = Q10**((TPAV-TPR)/10.)
          RMLV = WLV*RMCLV*TPEM*0.75
          RMST = WST*0.010*TPEM+WIR*0.0
          RMRT = WRT*0.015*TPEM
          RMSO = MIN (1000., WSO)*0.015*TPEM

*---------Potential transpiration and diffusion resistances canopy
*         Explanation in sections 4.1, 4.4

          CO2I = CO2E*FIEC
          RSTL  = FURSC (WDSAD, MIN (2.5, ALV), PLHT, 2.0)
          RSBL  = 0.5*172.*SQRT(WDLV/(WDSAD*0.6))
          RSLLM = (CO2E-CO2I)/(PLMX*0.9+1.E-10)*(68.4/1.6)-10.

          PLNA  = (PCGC/(DLA/24.)-RMLV*0.33)/(MIN (2.5, ALV+1.E-10))
          RSLLT = (CO2E-CO2I)/(PLNA+1.E-10)*(68.4*24.0/1.6)-RSBL-RSTL

          RSLL  = LIMIT (RSLLM, 2000., RSLLT)

          CALL SUEVTR (RDTC, RDTM, 0.25, (DLA/24.), TPAD, VPA,
     &                 RSLL, RSBL, RSTL,
```

```
     &                TRCPR, TRCPD)
          TRC  = TRCPR*(1.0-EXP (-0.5*ALV))+TRCPD*MIN (2.5, ALV)

          PCGW = PCGC*PCEW

          RMMA = 0.2*PCGW*0.5
          RMCR = RMLV+RMST+RMSO+RMRT+RMMA

     *    Carbohydrate available for growth, export
     *    Explanation in sections 3.2, 2.4, 2.3, 2.2

          LSTR  = INSW ((AFGEN (CASTT, ICASTN, DS)-0.01), WIR*0.1, 0.0)

          CAGCR = PCGW*0.682-RMCR*0.682+LSTR*1.111*0.947
          CAGSS = CAGCR*AFGEN (CASST, ICASSN, DS)*CPEW
          CAGRT = CAGCR-CAGSS
          CAGLV = CAGSS*AFGEN (CALVT, ICALVN, DS)
          CAGST = CAGSS*AFGEN (CASTT, ICASTN, DS)
          CAGSO = CAGSS-CAGLV-CAGST

          CELV  = PCGW-(RMLV+RMST+0.5*RMMA)

     *    Growth rates and loss rates
     *    Explanation in sections 2.4, 3.2, 2.2

          GLV = CAGLV/CRGLV
          GST = CAGST/CRGST
          GRT = CAGRT/CRGRT
          GSO = CAGSO/CRGSO

          LLV = WLV*AFGEN (LLVT, ILLVN, DS)
          LRT = WRT*AFGEN (LRTT, ILRTN, DS)

     *    Respiration due to growth
     *    Explanation in section 2.4

          RGLV = GLV*CPGLV
          RGST = GST*CPGST
          RGSO = GSO*CPGSO
          RGRT = GRT*CPGRT
          RLSR = LSTR*1.111*0.053*1.467
          RGCR = RGLV+RGST+RGSO+RGRT+RLSR
          RSH  = RMLV+RMST+RMSO+RMMA+RGLV+RGST+RGSO+RLSR

     *    Leaf area
     *    Explanation in section 3.3

          SLN = SLC*AFGEN (SLT, ISLN, DS)
          GLA = GLV/SLN
          LLA = LLV/SLA
          GSA = 0.5*GST/SSC

     *    Phenological development of the crop
     *    Explanation in section 3.1

          DRED = AFGEN (DRDT, IDRDN, DLP)
          DRV  = DRCV*DRED*DREW*AFGEN (DRVTT, IDRVTN, TPAV)
          DRR  = DRCR*AFGEN (DRRTT, IDRRTN, TPAV)

     *    output of states and rates only if it is required

          IF (OUTPUT .OR. TERMNL) THEN
             CALL OUTDAT (2, 0, 'WLV',  WLV)
             CALL OUTDAT (2, 0, 'WLVT', WLVT)
             CALL OUTDAT (2, 0, 'WLVST',WLVST)
             CALL OUTDAT (2, 0, 'WLVSO',WLVSO)
             CALL OUTDAT (2, 0, 'WST',  WST)
             CALL OUTDAT (2, 0, 'WIR',  WIR)
```

```
            CALL OUTDAT (2, 0, 'WSO',  WSO)
            CALL OUTDAT (2, 0, 'WRT',  WRT)
            CALL OUTDAT (2, 0, 'GLV',  GLV)
            CALL OUTDAT (2, 0, 'GST',  GST)
            CALL OUTDAT (2, 0, 'GSO',  GSO)
            CALL OUTDAT (2, 0, 'GRT',  GRT)
            CALL OUTDAT (2, 0, 'SLA',  SLA)
            CALL OUTDAT (2, 0, 'PLMX', PLMX)
            CALL OUTDAT (2, 0, 'ALV',  ALV)
            CALL OUTDAT (2, 0, 'DS',   DS)
            CALL OUTDAT (2, 0, 'TPAV', TPAV)
            CALL OUTDAT (2, 0, 'RDTM', RDTM)
            CALL OUTDAT (2, 0, 'PCGT', PCGT)
            CALL OUTDAT (2, 0, 'RCRT', RCRT)
            CALL OUTDAT (2, 0, 'RMCT', RMCT)
            CALL OUTDAT (2, 0, 'TRC', TRC)
         END IF

      ELSE IF (ITASK.EQ.3) THEN

*        integration section

*        Weights of crop components
*        Explanation in sections 3.2, 2.2, 3.4

         WLV = INTGRL (WLV, (GLV-LLV), DELT)

         WST = INTGRL (WST, (GST*(1.0-FSTR)), DELT)

         WIR = INTGRL (WIR, (GST*(FSTR*(FCST/0.444))-LSTR), DELT)
         WSO = INTGRL (WSO, GSO, DELT)

         WEPSO = WSO*FEPSO

         WRT   = INTGRL (WRT, (GRT-LRT), DELT)

         WSS   = WLV+WST+WSO+WIR
         WCR   = WSS+WRT

         WLVD  = INTGRL (WLVD, LLV, DELT)
         WRTD  = INTGRL (WRTD, LRT, DELT)

         WLVT  = WLV+WLVD
         WLVST = WLVT+WST+WIR
         WLVSO = WLVST+WSO
         HI    = WSO/WSS
         WSTR  = WST+WIR

*        Photosynthesis, gross and net
*        Explanation in sections 2.1, 3.3, 3.4

         PCGT = INTGRL (PCGT, PCGW, DELT)
         PCNT = INTGRL (PCNT, (PCGW-(RMCR+RGCR)), DELT)

*        Respiration
*        Explanation in sections 2.4, 2.3

         RMCT = INTGRL (RMCT, RMCR, DELT)
         RCRT = INTGRL (RCRT, (RMCR+RGCR),DELT)

*        Carbohydrate available for growth, export
*        Explanation in sections 3.2, 2.4, 2.3, 2.2

         CELVN = INTGRL (CELVN, INSW (CELV, 1.0 ,-CELVN/DELT), DELT)

*        Leaf area
*        Explanation in section 3.3
```

```
          ALV = INTGRL (ALV, (GLA-LLA+GSA), DELT)

*         Phenological development of the crop
*         Explanation in section 3.1

          DS   = INTGRL (DS, INSW ((DS-1.0), DRV, DRR), DELT)
          PLHT = AFGEN (PLHTT, IPLHTN, DS)

*         Carbon balance check
*         Explanation in section 3.4

          CKCIN = (WLV-WLVI)*FCLV+(WST-WSTI)*FCST+
     &            (WSO-WSOI)*FCSO+(WRT-WRTI)*FCRT+WIR*0.444
          CKCFL = PCNT*0.2727-(WLVD*FCLV+WRTD*FCRT)
          CKCRD = FUCCHK (CKCIN, CKCFL, TIME)

*         Determine the finish conditions of the simulation

          IF (DS.GE.2.0)    TERMNL = .TRUE.
          IF (CELVN.GE.3.0) TERMNL = .TRUE.

        ELSE IF (ITASK.EQ.4) THEN

*         Define graph for output
          CALL OUTPLT (1, 'WLV')
          CALL OUTPLT (1, 'WLVT')
          CALL OUTPLT (1, 'WLVST')
          CALL OUTPLT (1, 'WLVSO')

*         use common scale, small plot width for output
          CALL OUTPLT (7, 'Macros simulation model')

        END IF

        ITOLD = ITASK

        RETURN
        END
```

File: A:\SYSSARP\PROGDAT\L1QM.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE L1QM                                                      *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine simulates potential production of crops     *
*           with quarter day time steps, and morphological development  *
*           of tillers.                                                 *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                    units  class  *
*  ----   ---- -------                                    -----  -----  *
*  control                                                             *
*  ITASK   I4  Determines action of the subroutine,                    *
*              1=initialization, 2=rate calculation,                   *
*              3=integration, 4=terminal                    -     C,I   *
*  IUNITP  I4  Unit number of plant data file,              -     C,IN  *
*  IUNITO  I4  Unit number of output file                   -     C,IN  *
*  FILEP   C*  Name of plant data file                      -     C,IN  *
*  OUTPUT  L4  Flag that indicates if output to file is                *
*              required,                                    -     C,I   *
*  TERMNL  L4  Flag that indicates if simulation should                *
*              terminate                                    -    C,I,O  *
*                                                                      *
*  timing                                                              *
*  TIME    R4  Time of simulation                           d      T    *
*  DATE    R4  Day number of simulation                     d      T    *
*  DELT    R4  Time step of integration                     d      T    *
*                                                                      *
*  environment of plant                                                *
*  LAT     R4  Latitude  of weather station               degrees  I    *
*  ELV     R4  Elevation of weather station                 m      I    *
*  DLA     R4  Daylength of current day                     h      I    *
*  DLP     R4  Photoperiodic daylength of current day       h      I    *
*  RDTM    R4  Global radiation                           J/m2/d   I    *
*  TPHP    R4  Maximum temperature of previous day   degrees Celsius I  *
*  TPL     R4  Minimum temperature                   degrees Celsius I  *
*  TPH     R4  Maximum temperature                   degrees Celsius I  *
*  TPLN    R4  Minimum temperature of next day       degrees Celsius I  *
*  HUAA    R4  Vapour pressure                            kPa      I    *
*                                                                      *
*  FATAL ERROR CHECKS (execution terminated, message):                 *
*  DELT > 0.25                                                         *
*  Certain sequences of ITASK, see subroutine CHKTSK                   *
*  Carbon balance check                                                *
*                                                                      *
*  SUBROUTINES and FUNCTIONS called:                                   *
*  from TTUTIL : CHKTSK,  ERROR, RDINIT, RDSREA, RDSINT, RDAREA, OUTCOM,*
*                OUTDAT, OUTPLT,  AFGEN,   INSW, INTGRL,  LIMIT, REAAND,*
*                REANOR                                                *
*  from MACROS.LIB: FUPHOT, FUCCHK, SUEVTR,  FURSC,  FUTP2,   FUVP     *
*                                                                      *
*  FILE usage : - Plant definition file FILEP opened and closed for    *
*                 ITASK=1, unit numbers used are IUNITP and IUNITP+1   *
*               - Output file with unit IUNITO for output and warnings *
*-----------------------------------------------------------------------*

*      CALL L1QM (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
*     &            TIME,   DATE,   DELT,
*     &             LAT,    ELV,    DLA,   DLP,
*     &            RDTM,   TPHP,    TPL,   TPH, TPLN, HUAA)

      SUBROUTINE L1QM (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
     &                 TIME,   DATE,   DELT,
     &                  LAT,    ELV,    DLA,   DLP,
```

```
     &                    RDTM,    TPHP,     TPL,    TPH, TPLN, HUAA)
      IMPLICIT REAL (A-Z)

*     Formal parameters

      INTEGER  ITASK, IUNITP, IUNITO
      LOGICAL OUTPUT, TERMNL
      CHARACTER*(*) FILEP

*     Standard local parameters as in L1D
      INTEGER ILAR, ITOLD
      PARAMETER (ILAR=20)

      REAL     PLMTT(ILAR), PLETT(ILAR)
      INTEGER IPLMTN      ,IPLETN

      REAL     CALVT(ILAR), CASTT(ILAR), CASST(ILAR)
      INTEGER ICALVN      ,ICASTN      ,ICASSN

      REAL     LLVT(ILAR), LRTT(ILAR)
      INTEGER ILLVN      ,ILRTN

      REAL     DRVTT(ILAR), DRRTT(ILAR), DRDT(ILAR), SLT(ILAR)
      INTEGER IDRVTN      ,IDRRTN      ,IDRDN      ,ISLN

*     Extra declarations necessary for L1QM

      REAL     PLMHT(ILAR)
      INTEGER IPLMHN

*     Extra declarations necessary for module TIL

      REAL     CNTIT(ILAR), GGRT(ILAR)
      INTEGER ICNTIN,      IGGRTN

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:              Old task:
*           initialization         terminal
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done i.e. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('L1QM', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L1QM, quarter day time steps,')
         CALL OUTCOM ('with tiller, florets and grains')

*        Initialization section
```

```
      IF (DELT.GT.0.25) CALL ERROR ('L1QM','DELT too large for L1QM')
      CALL RDINIT (IUNITP, IUNITO, FILEP)

*     Initialization of states

      CALL RDSREA ('WLVI', WLVI)
      WLV = WLVI
      WAR  = WLV*0.05
      WARR = WAR/(WLV+1.E-10)
      CALL RDSREA ('WSTI', WSTI)
      WST = WSTI
      WSR  = 0.
      CALL RDSREA ('WSOI', WSOI)
      WSO = WSOI
      GSOAV  = 0.
      GSOAVM = 0.
      CALL RDSREA ('FEPSO', FEPSO)
      WEPSO = WSO*FEPSO
      WRTI  = WLVI
      WRT   = WLV
      WSS   = WLV+WST+WSO+WAR+WSR
      WCR   = WSS+WRT
      WIR   = 0.
      WLVD  = 0.
      WRTD  = 0.
      WLVT  = WLV+WLVD+WAR
      WLVST = WLVT+WST+WAR
      WLVSO = WLVST+WSO
      HI    = WSO/WSS
      WSTR  = WST+WSR
      PCGT  = 0.
      PCNT  = 0.
      PCGDV = 0.
      RMCT  = 0.
      RCRT  = 0.
      CELV  = 10.
      CELVN = 0.
      CALL RDSREA ('SLC', SLC)
      CALL RDAREA ('SLT', SLT, ILAR, ISLN)
      CALL RDSREA ('DSI', DSI)
      DS = DSI
      ALV  = WLV/(SLC*AFGEN (SLT, ISLN, DS))

      CALL RDSREA ('FCLV', FCLV)
      CALL RDSREA ('FCST', FCST)
      CALL RDSREA ('FCSO', FCSO)
      CALL RDSREA ('FCRT', FCRT)

*     Remaining variables to be initialized in
*     state section for module TIL

      CALL RDSREA ('WTI', WTI)
      NTII = WLV/WTI
      NTI  = NTII
      NFL  = 0.
      NGR  = 0.
      WGR  = WSO/MAX (NGR, 1000.)

*     Variables to be initialized in rate section as in L1Q

      CALL RDSREA ('SSC', SSC)
      CALL RDSREA ('PLMXP', PLMXP)
      CALL RDAREA ('PLMTT', PLMTT, ILAR, IPLMTN)
      CALL RDAREA ('PLMHT', PLMHT, ILAR, IPLMHN)
      CALL RDSREA ('PLEI', PLEI)
      CALL RDAREA ('PLETT', PLETT, ILAR, IPLETN)

      CALL RDSREA ('Q10', Q10)
```

```
      CALL RDSREA ('TPR', TPR)
      CALL RDSREA ('RMCLV', RMCLV)

      CALL RDAREA ('CASST', CASST, ILAR, ICASSN)
      CALL RDAREA ('CALVT', CALVT, ILAR, ICALVN)
      CALL RDAREA ('CASTT', CASTT, ILAR, ICASTN)

      CALL RDSREA ('CRGLV', CRGLV)
      CALL RDSREA ('CRGST', CRGST)
      CALL RDSREA ('FSTR' , FSTR)
      CALL RDSREA ('CRGRT', CRGRT)
      CALL RDSREA ('GSORM', GSORM)
      CALL RDSREA ('CRGSO', CRGSO)
      CALL RDSREA ('FCLV' , FCLV)

      CALL RDSREA ('CPGLV', CPGLV)
      CALL RDSREA ('CPGST', CPGST)
      CALL RDSREA ('CPGSO', CPGSO)
      CALL RDSREA ('CPGRT', CPGRT)

      CALL RDAREA ('DRDT',  DRDT, ILAR, IDRDN)
      CALL RDSREA ('DRCV',  DRCV)
      CALL RDAREA ('DRVTT', DRVTT, ILAR, IDRVTN)
      CALL RDSREA ('DRCR',  DRCR)
      CALL RDAREA ('DRRTT', DRRTT, ILAR, IDRRTN)

*     Extra for module TIL

      CALL RDSREA ('WGRMX', WGRMX)
      CALL RDSREA ('DSG1', DSG1)
      CALL RDSREA ('DSG2', DSG2)
      CALL RDSREA ('TCFG', TCFG)
      CALL RDAREA ('CNTIT', CNTIT, ILAR, ICNTIN)
      CALL RDSREA ('DST1', DST1)
      CALL RDSREA ('DST2', DST2)
      CALL RDSREA ('TCFT', TCFT)
      CALL RDSREA ('TCDT', TCDT)
      CALL RDSREA ('DSF1', DSF1)
      CALL RDSREA ('DSF2', DSF2)
      CALL RDSREA ('NFLMXT', NFLMXT)
      CALL RDAREA ('GGRT', GGRT, ILAR, IGGRTN)

      CLOSE (IUNITP, STATUS='DELETE')

    ELSE IF (ITASK.EQ.2) THEN

*     rate calculation section

*     Weather data, time and date
*     Explanation in chapter 6 and sections 1.4, 3.4

      RDTM  = RDTM*1000.
      DTIME = TIME-AINT (TIME)
      NIGHT = INSW (REAAND ((DTIME-0.1),(0.6-DTIME))-0.1, 1.0, 0.0)
      FADL  = INSW ((0.5-NIGHT), (2.-DLA/12.), (DLA/12.))
      TPAA  = FUTP2 (DTIME, TPHP, TPL, TPH, TPLN,
     &               0.15, 0.45, 0.90, 0.60)
      RDTM  = INSW ((0.5-NIGHT), 0., RDTM)
      VPD   = MAX (0., (FUVP (TPAA)-HUAA))

*     Photosynthesis, gross and net
*     Explanation in sections 2.1, 3.3, 3.4

      SLA   = (WLV+0.5*WST*(SLC/SSC)) / ALV
      PLMXT = PLMXP*LIMIT (200., 600., SLA) / SLC
      PLMX  = PLMXT*AFGEN (PLMTT, IPLMTN, TPAA)*(1.0-ELV/8000.)*
     &        AFGEN (PLMHT, IPLMHN, 0.75*VPD)*
     &        INSW ((WARR-0.3), 1., 0.3)
```

```
      PLEA  = PLEI*AFGEN (PLETT, IPLETN, TPAA)
      PCGC  = FUPHOT (PLMX, PLEA, ALV, RDTM, DATE, LAT)
      PCGD  = PCGC/(DLA/24.)

*     Respiration due to maintenance
*     Explanation in section 2.3

      TPEM  = Q10**((TPAA-TPR)/10.)
      RMLVN = WLV*RMCLV*TPEM
      RMLVD = WLV*RMCLV*TPEM*0.5
      RMLV  = INSW ((0.5-NIGHT), RMLVN, RMLVD)
      RMST  = WST*0.010*TPEM+WSR*0.0+WAR*0.0
      RMRT  = WRT*0.015*TPEM
      RMSO  = MIN (1000., WSO)*0.015*TPEM

      RMMA = 0.20*PCGDV*0.5
      RMCR = RMLV+RMST+RMSO+RMRT+RMMA

*     Carbohydrate available and consumed for growth, export
*     Explanation in sections 3.2, 2.4, 2.2, 3.4

      WARG  = MAX (0.0, (WAR-0.05*WLV)/(DELT*FADL))
      CAGCR = LIMIT (0.0, WARG, (WAR-0.05*WLV)*1.5)
      CAGSS = CAGCR*AFGEN (CASST, ICASSN, DS)
      CAGRT = CAGCR-CAGSS
      CAGLV = CAGSS*AFGEN (CALVT, ICALVN, DS)
      CAGST = CAGSS*AFGEN (CASTT, ICASTN, DS)
      CAGSO = CAGSS-CAGLV-CAGST

*     Development of grains, florets and tillers in rice
*     (module TIL)

      DRR   = DRCR*AFGEN (DRRTT, IDRRTN, TPAA)
      GFP   = 1.0/(1.33*DRR)
      GGRMN = WGRMX/GFP
      GGRMX = GGRMN*2.0

      NGRP  = CAGCR/GGRMN
      NGRMX = NFL
      DSGR  = REANOR ((DSG1-DS), (DS-DSG2))
      GNGR  = DSGR*MAX (0.0, MIN ((NGRP-NGR),(NGRMX-NGR))/TCFG)

      CNTI = AFGEN (CNTIT, ICNTIN, DS)
      NTIP = CAGCR/CNTI
      DSTF = REANOR ((DST1-DS), (DS-DST2))
      DSTD = REANOR ((DST1-DS), (DS-(DST2+0.15)))
      GNTI = DSTF*MAX (0.0, (NTIP-NTI)/TCFT)
      LNTI = DSTD*MAX (0.0, (NTI-NTIP)/TCDT)

      CNFL  = 0.7*GGRMN
      NFLP  = CAGCR/CNFL
      DSFL  = REANOR ((DSF1-DS), (DS-DSF2))
      NFLMX = NFLMXT*NTI
      GNFL  = DSFL*MIN ((NFLMX-NFL),(NFLP-NFL))

*     Growth rates and loss rates
*     Explanation in sections 2.4, 2.2, 3.2, 3.4

      GLV  = CAGLV/CRGLV
      GST  = CAGST/CRGST*
     &       (CRGST*(1.0-FSTR)/(FSTR*(1.111/0.947-CRGST)+CRGST))
      GRT  = CAGRT/CRGRT

      GSOM = NGR*GGRMX*AFGEN (GGRT, IGGRTN, TPAA)
*     GSOM = (WSO+10.)*GSORM
      GSO  = MIN ((CAGSO/CRGSO), GSOM)
      GSRP = (CAGSS-GLV*CRGLV-GST*CRGST-GSO*CRGSO)*0.947/1.111
      GSR  = INSW (((WST+WSR)*(FSTR+0.10)-WSR), 0., GSRP)
```

```
      MCLV = INSW ((GSOAVM*0.8-GSOAV-10.), 0.0, 1.0)
      LLV  = WLV*0.15*MCLV

      MCRT = MCLV
      LRT  = WRT*0.15*MCRT

      MCSR = INSW ((GSOAVM*1.0-GSOAV-10.), 0.0, 1.0)
      LSR  = WSR*0.20*MCSR

      GAR  = PCGD*0.682-RMCR*0.682+
     &       LSR*1.111*0.947+0.5*LLV*(FCLV/0.400)*0.947

*     Weight of carbohydrates used for growth
*     Explanation in section 2.4

      CUGLV = GLV*CRGLV
      CUGST = GST*CRGST
      CUGSO = GSO*CRGSO
      CUGRT = GRT*CRGRT
      CUGSR = GSR*1.111/0.947
      CUGCR = CUGLV+CUGST+CUGSO+CUGRT+CUGSR

*     Respiration due to growth
*     Explanation in section 2.4

      RGLV = GLV*CPGLV
      RGST = GST*CPGST
      RGSO = GSO*CPGSO
      RGRT = GRT*CPGRT
      RGSR = GSR/0.947*1.111*0.053*1.467
      RLSR = LSR*1.111*0.053*1.467
      RLLV = (LLV*0.5)*(FCLV/0.400)*0.053*1.467

      RGCR  = RGLV+RGST+RGSO+RGRT+RGSR+RLSR+RLLV
      RSH   = RMLV+RMST+RMSO+RMMA+RGLV+RGST+RGSO+RGSR+RLSR+RLLV
      PCNSH = PCGD-RSH

*     Area of leaves
*     Explanation in section 3.3

      SLN = SLC*AFGEN (SLT, ISLN, DS)
      GLA = GLV/SLN
      LLA = LLV/SLA
      GSA = 0.5*GST/SSC

*     Phenological development of the crop
*     Explanation in section 3.1

      DRED = AFGEN (DRDT, IDRDN, DLP)
      DRV  = DRCV*DRED*AFGEN (DRVTT, IDRVTN, TPAA)
      DRR  = DRCR*AFGEN (DRRTT, IDRRTN, TPAA)

*     output of states and rates only if it is required
      IF (OUTPUT .OR. TERMNL) THEN
         CALL OUTDAT (2, 0, 'WLV',  WLV)
         CALL OUTDAT (2, 0, 'WLVT',  WLVT)
         CALL OUTDAT (2, 0, 'WLVST',  WLVST)
         CALL OUTDAT (2, 0, 'WLVSO',  WLVSO)
         CALL OUTDAT (2, 0, 'WST',  WST)
         CALL OUTDAT (2, 0, 'WSO',  WSO)
         CALL OUTDAT (2, 0, 'WRT',  WRT)
         CALL OUTDAT (2, 0, 'WAR',  WAR)
         CALL OUTDAT (2, 0, 'WSR',  WSR)
         CALL OUTDAT (2, 0, 'WARR',  WARR)
         CALL OUTDAT (2, 0, 'GLV',  GLV)
         CALL OUTDAT (2, 0, 'GST',  GST)
         CALL OUTDAT (2, 0, 'GSO',  GSO)
```

```
            CALL OUTDAT (2, 0, 'GRT',  GRT)
            CALL OUTDAT (2, 0, 'GSRP',  GSRP)
            CALL OUTDAT (2, 0, 'GSR',  GSR)
            CALL OUTDAT (2, 0, 'SLA',  SLA)
            CALL OUTDAT (2, 0, 'PLMX', PLMX)
            CALL OUTDAT (2, 0, 'ALV',  ALV)
            CALL OUTDAT (2, 0, 'DS',   DS)
            CALL OUTDAT (2, 0, 'TPAA', TPAA)
            CALL OUTDAT (2, 0, 'PCGT', PCGT)
            CALL OUTDAT (2, 0, 'PCNSH', PCNSH)
            CALL OUTDAT (2, 0, 'RCRT', RCRT)
            CALL OUTDAT (2, 0, 'RMCT', RMCT)
            CALL OUTDAT (2, 0, 'RSH', RSH)
            CALL OUTDAT (2, 0, 'LLV', LLV)
         END IF

      ELSE IF (ITASK.EQ.3) THEN

*        integration section

*        Weight crop components
*        Explanation in sections 3.2, 2.4, 2.2, 3.4

         WLV    = INTGRL (WLV, (GLV-LLV)*FADL, DELT)
         WAR    = INTGRL (WAR, (GAR-CUGCR)*FADL, DELT)
         WARR   = WAR/(WLV+1.E-10)
         WST    = INTGRL (WST, GST*FADL, DELT)
         WSR    = INTGRL (WSR, (GSR-LSR)*FADL, DELT)
         WSO    = INTGRL (WSO, GSO*FADL, DELT)
         GSOAV  = INTGRL (GSOAV,((GSO-GSOAV)/2.)*FADL, DELT)
         GSOAVM = INTGRL (GSOAVM, MAX (0., GSOAV-GSOAVM)*FADL, DELT)

         WEPSO  = WSO*FEPSO

         WRT    = INTGRL (WRT, (GRT-LRT)*FADL, DELT)
         WSS    = WLV+WST+WSO+WAR+WSR
         WCR    = WSS+WRT

         WIR    = INTGRL (WIR, ((GAR-CUGCR)*0.900+(GSR-LSR))*FADL, DELT)

         WLVD   = INTGRL (WLVD, (0.5*LLV)*FADL, DELT)
         WRTD   = INTGRL (WRTD, LRT*FADL, DELT)
         WLVT   = WLV+WLVD+WAR
         WLVST  = WLVT+WST+WSR
         WLVSO  = WLVST+WSO
         HI     = WSO/WSS
         WSTR   = WST+WSR

*        Photosynthesis, gross and net
*        Explanation in sections 2.1, 3.3, 3.4

         PCGT   = INTGRL (PCGT, PCGD*FADL, DELT)
         PCNT   = INTGRL (PCNT, (PCGD-(RMCR+RGCR))*FADL, DELT)
         PCGDV  = INTGRL (PCGDV, (PCGD-PCGDV)/1.0*FADL, DELT)

*        Respiration
*        Explanation in sections 2.4, 2.3

         RMCT   = INTGRL (RMCT, RMCR*FADL, DELT)
         RCRT   = INTGRL (RCRT, (RMCR+RGCR)*FADL,DELT)

*        Carbohydrate available and consumed for growth, export
*        Explanation in sections 3.2, 2.4, 2.2, 3.4

         CELV   = INTGRL (CELV, (PCGD-(RMLV+RMST+0.5*RMMA))*FADL-
     &                    INSW (DTIME-0.01, CELV/DELT, 0.), DELT)
         CELVN  = INTGRL (CELVN, INSW (CELV, 1.0 ,-CELVN/DELT), DELT)
```

```
*       Area of leaves
*       Explanation in section 3.3

        ALV   = INTGRL (ALV, (GLA-LLA+GSA)*FADL, DELT)

*       Phenological development of the crop
*       Explanation in section 3.1

        DS    = INTGRL (DS, INSW ((DS-1.0), DRV, DRR)*FADL, DELT)

*       Carbon balance check
*       Explanation in section 3.4

        CKCIN = (WLV-WLVI)*FCLV+(WST-WSTI)*FCST+
     &          (WSO-WSOI)*FCSO+(WRT-WRTI)*FCRT+WIR*0.444
        CKCFL = PCNT*0.2727-(WLVD*FCLV+WRTD*FCRT)
        CKCRD = FUCCHK (CKCIN, CKCFL, TIME)

*       Development of grains, florets and tillers in rice
*       module TIL

        NTI   = INTGRL (NTI, (GNTI-LNTI)*FADL, DELT)
        NFL   = INTGRL (NFL, GNFL*FADL, DELT)
        NGR   = INTGRL (NGR, GNGR*FADL, DELT)
        NTIPL = NTI/NTII
        WGR   = WSO/(MAX (NGR, 1000.))
```

```
*       determine if the simulation should stop

        IF (DS.GE.2.)      TERMNL = .TRUE.
        IF (CELVN.GE.3.)   TERMNL = .TRUE.
        IF (WGR.GE.WGRMX)  TERMNL = .TRUE.

      ELSE IF (ITASK.EQ.4) THEN

*       define graph for output
        CALL OUTPLT (1, 'WLV')
        CALL OUTPLT (1, 'WLVT')
        CALL OUTPLT (1, 'WLVST')
        CALL OUTPLT (1, 'WLVSO')

*       use common scale, small plot width for output
        CALL OUTPLT (7, 'Macros simulation model')

*       define second graph for output
        CALL OUTPLT (1, 'RSH')
        CALL OUTPLT (1, 'PCNSH')
        CALL OUTPLT (1, 'WARR')
        CALL OUTPLT (1, 'TPAA')

*       use individual scale, small plot width for output
        CALL OUTPLT (6, 'Macros simulation model')

      END IF

      ITOLD = ITASK

      RETURN
      END
```

File: A:\SYSSARP\PROGDAT\L2DT.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE L2DT                                                      *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine simulates water limited production of crops. *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                    units  class  *
*  ----   ---- -------                                    -----  -----  *
*  control                                                             *
*  ITASK   I4  Determines action of the subroutine,                    *
*              1=initialization, 2=rate calculation,                   *
*              3=integration, 4=terminal                    -     C,I   *
*  IUNITP  I4  Unit number of plant data file,              -     C,IN  *
*  IUNITO  I4  Unit number of output file                   -     C,IN  *
*  FILEP   C*  Name of plant data file                      -     C,IN  *
*  OUTPUT  L4  Flag that indicates if output to file is                *
*              required,                                    -     C,I   *
*  TERMNL  L4  Flag that indicates if simulation should                *
*              terminate                                    -    C,I,O  *
*                                                                      *
*  timing                                                              *
*  TIME    R4  Time of simulation                           d      T    *
*  DATE    R4  Day number of simulation                     d      T    *
*  DELT    R4  Time step of integration                     d      T    *
*                                                                      *
*  location and weather environment of plant                           *
*  LAT     R4  Latitude  of weather station              degrees   I    *
*  ELV     R4  Elevation of weather station                 m      I    *
*  DLA     R4  Daylength of current day                     h      I    *
*  DLP     R4  Photoperiodic daylength of current day       h      I    *
*  RDTM    R4  Global radiation                          J/m2/d    I    *
*  TPL     R4  Minimum temperature                  degrees Celsius I    *
*  TPH     R4  Maximum temperature                  degrees Celsius I    *
*  HUAA    R4  Vapour pressure                             kPa     I    *
*  WDS     R4  Average wind speed                          m/s     I    *
*                                                                      *
*  soil environment                                                    *
*  NL      I4  Number of soil layers from soil water balance  -       I    *
*  TKL     R4  Array of thicknesses of soil layers            m       I    *
*  TKLT    R4  Depth of simulated soil                        m       I    *
*  ZRTMS   R4  Maximum rooting depth of soil                  m       I    *
*  WCWP    R4  Array of water contents at wilting point    cm3/cm3    I    *
*  WCFC    R4   ,,   ,,  ,,      ,,    ,, field capacity   cm3/cm3    I    *
*  WCST    R4   ,,   ,,  ,,      ,,    ,, saturation       cm3/cm3    I    *
*  WCLQT   R4   ,,   ,, actual water contents              cm3/cm3    I    *
*                                                                      *
*  plant output data for potential soil evaporation and water balance  *
*  TRWL    R4  Array of actual transpiration per layer     cm3/cm3    O    *
*  ALVX    R4  Leaf area index (external copy !)            m2/m2     O    *
*  PLHT    R4  Plant height                                   m       O    *
*  WDLV    R4  Width of leaves                                m       O    *
*                                                                      *
*  FATAL ERROR CHECKS (execution terminated, message):                 *
*  DELT < 1                                                            *
*  Certain sequences of ITASK, see subroutine CHKTSK                   *
*  Carbon balance check                                                *
*                                                                      *
*  SUBROUTINES and FUNCTIONS called:                                   *
*  from TTUTIL : CHKTSK,  ERROR, RDINIT, RDSREA, RDSINT, RDAREA, OUTCOM, *
*                OUTDAT, OUTPLT,  AFGEN,   INSW, INTGRL,  LIMIT, REAAND *
*  from MACROS.LIB: FUPHOT, FUCCHK, SUEVTR,  FURSC,   FUWS             *
*                                                                      *
*  FILE usage : - Plant definition file FILEP opened and closed for    *
```

```
*               ITASK=1, unit numbers used are IUNITP and IUNITP+1     *
*             - Output file with unit IUNITO for output and warnings   *
*------------------------------------------------------------------------*

*      CALL L2DT (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
*     &             TIME,   DATE,   DELT,
*     &              LAT,    ELV,    DLA,   DLP,
*     &             RDTC,   RDTM,    TPL,   TPH, HUAA, WDS,
*     &               NL,    TKL,   TKLT, ZRTMS,
*     &             WCWP,   WCFC,   WCST, WCLQT,
*     &             TRWL,   ALVX,   PLHT,  WDLV)

      SUBROUTINE L2DT (ITASK, IUNITP, IUNITO, FILEP, OUTPUT, TERMNL,
     &                 TIME,   DATE,   DELT,
     &                  LAT,    ELV,    DLA,   DLP,
     &                 RDTC,   RDTM,    TPL,   TPH, HUAA, WDS,
     &                   NL,    TKL,   TKLT, ZRTMS,
     &                 WCWP,   WCFC,   WCST, WCLQT,
     &                 TRWL,   ALVX,   PLHT,  WDLV)
      IMPLICIT REAL (A-Z)

*     Formal parameters

      INTEGER  ITASK, IUNITP, IUNITO
      LOGICAL OUTPUT, TERMNL
      CHARACTER*(*) FILEP

*     Standard local declarations as in L1D

      INTEGER ILAR, ITOLD
      PARAMETER (ILAR=20)

      REAL     PLMTT(ILAR), PLETT(ILAR)
      INTEGER IPLMTN      ,IPLETN

      REAL     CALVT(ILAR), CASTT(ILAR), CASST(ILAR)
      INTEGER ICALVN      ,ICASTN      ,ICASSN

      REAL     LLVT(ILAR), LRTT(ILAR)
      INTEGER ILLVN      ,ILRTN

      REAL     DRVTT(ILAR), DRRTT(ILAR), DRDT(ILAR), SLT(ILAR)
      INTEGER IDRVTN      ,IDRRTN      ,IDRDN      ,ISLN

*     Extra declarations as in L1DT

      REAL     PLHTT(ILAR)
      INTEGER IPLHTN

*     Extra declarations for L2DT

      REAL     DRWT(ILAR), WSET(ILAR)
      INTEGER IDRWTN     ,IWSETN

      INTEGER I, NL
      REAL  TKL(NL),  WCWP(NL), WCFC(NL)
      REAL WCST(NL), WCLQT(NL), TRWL(NL)

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:             Old task:
*           initialization        terminal
```

```
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done nl. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('L2DT', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L2DT, plant growth dependent on soil water')

*        Initialization section

         IF (DELT.LT.1.0) CALL ERROR ('L2DT','DELT too small for L2DT')
         CALL RDINIT (IUNITP, IUNITO, FILEP)

*        Initialization of states as in L1D

         CALL RDSREA ('WLVI', WLVI)
         WLV = WLVI
         CALL RDSREA ('WSTI', WSTI)
         WST = WSTI
         WIR = 0.0
         CALL RDSREA ('WSOI', WSOI)
         WSO = WSOI
         CALL RDSREA ('FEPSO', FEPSO)
         WEPSO = WSO*FEPSO
         WRTI = WLVI
         WRT  = WLV
         WSS  = WLV+WST+WSO+WIR
         WCR  = WSS+WRT
         WLVD  = 0.0
         WRTD  = 0.0
         WLVT  = WLV+WLVD
         WLVST = WLVT+WST+WIR
         WLVSO = WLVST+WSO
         HI    = WSO/WSS
         WSTR  = WST+WIR
         PCGT  = 0.0
         PCNT  = 0.0
         RMCT  = 0.0
         RCRT  = 0.0
         CELVN = 0.0
         CALL RDSREA ('SLC', SLC)
         CALL RDAREA ('SLT', SLT, ILAR, ISLN)
         CALL RDSREA ('DSI', DSI)
         DS = DSI
         ALV  = WLV/(SLC*AFGEN (SLT, ISLN, DS))
         ALVX = ALV

         CALL RDSREA ('FCLV', FCLV)
         CALL RDSREA ('FCST', FCST)
         CALL RDSREA ('FCSO', FCSO)
         CALL RDSREA ('FCRT', FCRT)

*        Remaining variables to be initialized in state section as in L1D

         CALL RDSREA ('FSTR', FSTR)
         CALL RDSREA ('FCST', FCST)
```

```
*           integration             rate calculation
*           rate calculation        initialization, integration
*           terminal                <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done nl. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('L2DT', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L2DT, plant growth dependent on soil water')

*        Initialization section

         IF (DELT.LT.1.0) CALL ERROR ('L2DT','DELT too small for L2DT')
         CALL RDINIT (IUNITP, IUNITO, FILEP)

*        Initialization of states as in L1D

         CALL RDSREA ('WLVI', WLVI)
         WLV = WLVI
         CALL RDSREA ('WSTI', WSTI)
         WST = WSTI
         WIR = 0.0
         CALL RDSREA ('WSOI', WSOI)
         WSO = WSOI
         CALL RDSREA ('FEPSO', FEPSO)
         WEPSO = WSO*FEPSO
         WRTI = WLVI
         WRT  = WLV
         WSS  = WLV+WST+WSO+WIR
         WCR  = WSS+WRT
         WLVD = 0.0
         WRTD = 0.0
         WLVT = WLV+WLVD
         WLVST = WLVT+WST+WIR
         WLVSO = WLVST+WSO
         HI   = WSO/WSS
         WSTR = WST+WIR
         PCGT = 0.0
         PCNT = 0.0
         RMCT = 0.0
         RCRT = 0.0
         CELVN = 0.0
         CALL RDSREA ('SLC', SLC)
         CALL RDAREA ('SLT', SLT, ILAR, ISLN)
         CALL RDSREA ('DSI', DSI)
         DS = DSI
         ALV  = WLV/(SLC*AFGEN (SLT, ISLN, DS))
         ALVX = ALV

         CALL RDSREA ('FCLV', FCLV)
         CALL RDSREA ('FCST', FCST)
         CALL RDSREA ('FCSO', FCSO)
         CALL RDSREA ('FCRT', FCRT)

*        Remaining variables to be initialized in state section as in L1D

         CALL RDSREA ('FSTR', FSTR)
         CALL RDSREA ('FCST', FCST)
```

```
*       Remaining variables to be initialized in state section for L2DT

        TPS  = (TPL+TPH)/2.0
        CALL RDSREA ('ZRTI', ZRTI)
        ZRT  = ZRTI
        CALL RDAREA ('PLHTT', PLHTT, ILAR, IPLHTN)
        PLHT = AFGEN (PLHTT, IPLHTN, DS)

*       Variables to be initialized in rate section as in L1D

        CALL RDSREA ('SSC', SSC)
        CALL RDSREA ('PLMXP', PLMXP)
        CALL RDAREA ('PLMTT', PLMTT, ILAR, IPLMTN)
        CALL RDSREA ('PLEI', PLEI)
        CALL RDAREA ('PLETT', PLETT, ILAR, IPLETN)

        CALL RDSREA ('Q10', Q10)
        CALL RDSREA ('TPR', TPR)
        CALL RDSREA ('RMCLV', RMCLV)

        CALL RDAREA ('CASTT', CASTT, ILAR, ICASTN)
        CALL RDAREA ('CASST', CASST, ILAR, ICASSN)
        CALL RDAREA ('CALVT', CALVT, ILAR, ICALVN)

        CALL RDSREA ('CRGLV', CRGLV)
        CALL RDSREA ('CRGST', CRGST)
        CALL RDSREA ('CRGRT', CRGRT)
        CALL RDSREA ('CRGSO', CRGSO)

        CALL RDAREA ('LLVT', LLVT, ILAR, ILLVN)
        CALL RDAREA ('LRTT', LRTT, ILAR, ILRTN)

        CALL RDSREA ('CPGLV', CPGLV)
        CALL RDSREA ('CPGST', CPGST)
        CALL RDSREA ('CPGSO', CPGSO)
        CALL RDSREA ('CPGRT', CPGRT)

        CALL RDAREA ('DRDT',  DRDT, ILAR, IDRDN)
        CALL RDSREA ('DRCV',  DRCV)
        CALL RDAREA ('DRVTT', DRVTT, ILAR, IDRVTN)
        CALL RDSREA ('DRCR',  DRCR)
        CALL RDAREA ('DRRTT', DRRTT, ILAR, IDRRTN)

*       Extra variables to be initialized in rate section as in L1DT

        CALL RDSREA ('FIEC',  FIEC)
        CALL RDSREA ('WDLV',  WDLV)

*       Extra variables to be initialized in rate section for L2DT

        CALL RDSREA ('WSSC', WSSC)
        CALL RDSREA ('WFSC', WFSC)
        CALL RDAREA ('DRWT', DRWT, ILAR, IDRWTN)
        CALL RDSREA ('ZRTMC', ZRTMC)
        CALL RDSREA ('GZRTC', GZRTC)
        CALL RDAREA ('WSET', WSET, ILAR, IWSETN)

        CLOSE (IUNITP, STATUS='DELETE')

      ELSE IF (ITASK.EQ.2) THEN

*       rate calculation section

        TPAV = (TPL+TPH) /2.0
        TPAD = (TPH+TPAV)/2.0
        RDTM = RDTM*1000.
```

```
      CO2E  = 340.*0.88**(ELV/1000.)
      WDSAV = MAX (0.2, WDS)
      WDSAD = 1.33*WDSAV
      VPA   = MIN (FUVP (TPAD), HUAA)

*     Photosynthesis, gross and net
*     Explanation in sections 2.1, 3.3, 3.4

      SLA  = (WLV+0.5*WST*(SLC/SSC))/ALV
      PLMX = PLMXP*AFGEN (PLMTT, IPLMTN, TPAD)*
     &       LIMIT (200., 600., SLA)/SLC
      PLEA = PLEI*AFGEN (PLETT, IPLETN, TPAD)
      PCGC = FUPHOT (PLMX, PLEA, ALV, RDTM, DATE, LAT)

*     Maintenance respiration
*     Explanation in section 2.3

      TPEM = Q10**((TPAV-TPR)/10.)
      RMLV = WLV*RMCLV*TPEM*0.75
      RMST = WST*0.010*TPEM+WIR*0.0
      RMRT = WRT*0.015*TPEM
      RMSO = MIN (1000., WSO)*0.015*TPEM

*     Potential transpiration and diffusion resistances canopy
*     Explanation in sections 4.1, 4.4

      CO2I = CO2E*FIEC

      RSTL  = FURSC (WDSAD, MIN (2.5, ALV), PLHT, 2.0)
      RSBL  = 0.5*172.*SQRT(WDLV/(WDSAD*0.6))
      RSLLM = (CO2E-CO2I)/(PLMX*0.9+1.E-10)*(68.4/1.6)-10.

      PLNA  = (PCGC/(DLA/24.)-RMLV*0.33)/(MIN (2.5, ALV+1.E-10))
      RSLLT = (CO2E-CO2I)/(PLNA+1.E-10)*(68.4*24.0/1.6)-RSBL-RSTL

      RSLL  = LIMIT (RSLLM, 2000., RSLLT)

      CALL SUEVTR (RDTC, RDTM, 0.25, (DLA/24.), TPAD, VPA,
     &             RSLL, RSBL, RSTL,
     &             TRCPR, TRCPD)
      TRC = TRCPR*(1.0-EXP (-0.5*ALV))+TRCPD*MIN (2.5, ALV)

*     at this point potential tranpiration and assimilation
*     are calculated.

      TRRM = TRC/(ZRT+1.0E-10)

      TRW  = 0.
      ZLL  = 0.
      DO 10 I=1,NL
*        calculation of WSE for crops except rice
*        WSE  = FUWS (TRC, ALV, WCLQT(I), WSSC, WFSC, WCWP(I),
*    &                WCFC(I), WCST(I))

*        calculation of WSE for rice only
         RWCL = (WCLQT(I)-WCWP(I))/(WCST(I)-WCWP(I))
         WSE = AFGEN (WSET,IWSETN,RWCL)

         ZRTL = MIN (TKL(I), MAX ((ZRT-ZLL),0.0))
         WLA  = MAX (0.0, (WCLQT(I)-WCWP(I))*TKL(I)*1000.)
         TRWL(I) = MIN (WSE*ZRTL*TRRM, WLA/DELT)
         TRW  = TRW+TRWL(I)
         WSEE = INSW ((REAAND (WCLQT(I)+0.05-WCST(I),ZRT-0.2)-0.5),
     &                WSE, 0.)
         IF (ZRT.LT.(ZLL+TKL(I)) .AND. ZRT.GE.ZLL) WSERT = WSEE
         ZLL  = ZLL+TKL(I)
10    CONTINUE
```

```
*       Effects of water stress

        PCEW = TRW/(TRC+1.E-10)
        DREW = AFGEN (DRWT, IDRWTN, TRW/(TRC+1.E-10))
        CPEW = MIN (1.0, (0.5+TRW/(TRC+1.E-10)))

        PCGW = PCGC*PCEW

        RMMA = 0.2*PCGW*0.5
        RMCR = RMLV+RMST+RMSO+RMRT+RMMA

*       Carbohydrate available for growth, export
*       Explanation in sections 3.2, 2.4, 2.3, 2.2

        LSTR  = INSW ((AFGEN (CASTT, ICASTN, DS)-0.01), WIR*0.1, 0.0)

        CAGCR = PCGW*0.682-RMCR*0.682+LSTR*1.111*0.947
        CAGSS = CAGCR*AFGEN (CASST, ICASSN, DS)*CPEW
        CAGRT = CAGCR-CAGSS
        CAGLV = CAGSS*AFGEN (CALVT, ICALVN, DS)
        CAGST = CAGSS*AFGEN (CASTT, ICASTN, DS)
        CAGSO = CAGSS-CAGLV-CAGST

        CELV  = PCGW-(RMLV+RMST+0.5*RMMA)

*       Growth rates and loss rates
*       Explanation in sections 2.4, 3.2, 2.2

        GLV = CAGLV/CRGLV
        GST = CAGST/CRGST
        GRT = CAGRT/CRGRT
        GSO = CAGSO/CRGSO

        LLV = WLV*AFGEN (LLVT, ILLVN, DS)
        LRT = WRT*AFGEN (LRTT, ILRTN, DS)

*       Respiration due to growth
*       Explanation in section 2.4

        RGLV = GLV*CPGLV
        RGST = GST*CPGST
        RGSO = GSO*CPGSO
        RGRT = GRT*CPGRT
        RLSR = LSTR*1.111*0.053*1.467
        RGCR = RGLV+RGST+RGSO+RGRT+RLSR
        RSH  = RMLV+RMST+RMSO+RMMA+RGLV+RGST+RGSO+RLSR

*       Leaf area
*       Explanation in section 3.3

        SLN = SLC*AFGEN (SLT, ISLN, DS)
        GLA = GLV/SLN
        LLA = LLV/SLA
        GSA = 0.5*GST/SSC

*       Phenological development of the crop
*       Explanation in section 3.1

        DRED = AFGEN (DRDT, IDRDN, DLP)
        DRV  = DRCV*DRED*DREW*AFGEN (DRVTT, IDRVTN, TPAV)
        DRR  = DRCR*AFGEN (DRRTT, IDRRTN, TPAV)

*       root growth

        TERT = AFGEN (PLMTT, IPLMTN, TPS)
        GZRT = GZRTC*WSERT*TERT
        ZRTM = MIN (ZRTMC, ZRTMS, TKLT)
```

```
*       output of states and rates only if it is required

        IF (OUTPUT .OR. TERMNL) THEN
           CALL OUTDAT (2, 0, 'PCGC', PCGC)
           CALL OUTDAT (2, 0, 'PCEW', PCEW)
           CALL OUTDAT (2, 0, 'DREW', DREW)
           CALL OUTDAT (2, 0, 'CPEW', CPEW)
           CALL OUTDAT (2, 0, 'WLV' , WLV)
           CALL OUTDAT (2, 0, 'WLVT', WLVT)
           CALL OUTDAT (2, 0, 'WLVST',WLVST)
           CALL OUTDAT (2, 0, 'WLVSO',WLVSO)
           CALL OUTDAT (2, 0, 'WST',  WST)
           CALL OUTDAT (2, 0, 'WIR',  WIR)
           CALL OUTDAT (2, 0, 'WSO',  WSO)
           CALL OUTDAT (2, 0, 'WRT',  WRT)
           CALL OUTDAT (2, 0, 'ZRT',  ZRT)
           CALL OUTDAT (2, 0, 'GLV',  GLV)
           CALL OUTDAT (2, 0, 'GST',  GST)
           CALL OUTDAT (2, 0, 'GSO',  GSO)
           CALL OUTDAT (2, 0, 'GRT',  GRT)
           CALL OUTDAT (2, 0, 'SLA',  SLA)
           CALL OUTDAT (2, 0, 'PLMX', PLMX)
           CALL OUTDAT (2, 0, 'ALV',  ALV)
           CALL OUTDAT (2, 0, 'DS',   DS)
           CALL OUTDAT (2, 0, 'TPAV', TPAV)
           CALL OUTDAT (2, 0, 'RDTM', RDTM)
           CALL OUTDAT (2, 0, 'PCGT', PCGT)
           CALL OUTDAT (2, 0, 'RCRT', RCRT)
           CALL OUTDAT (2, 0, 'RMCT', RMCT)
           CALL OUTDAT (2, 0, 'TRC', TRC)
        END IF

      ELSE IF (ITASK.EQ.3) THEN

*       integration section

*       Weights of crop components
*       Explanation in sections 3.2, 2.2, 3.4

        WLV = INTGRL (WLV, (GLV-LLV), DELT)

        WST = INTGRL (WST, (GST*(1.0-FSTR)), DELT)

        WIR = INTGRL (WIR, (GST*(FSTR*(FCST/0.444))-LSTR), DELT)
        WSO = INTGRL (WSO, GSO, DELT)

        WEPSO = WSO*FEPSO

        WRT   = INTGRL (WRT, (GRT-LRT), DELT)

        WSS   = WLV+WST+WSO+WIR
        WCR   = WSS+WRT

        WLVD  = INTGRL (WLVD, LLV, DELT)
        WRTD  = INTGRL (WRTD, LRT, DELT)

        WLVT  = WLV+WLVD
        WLVST = WLVT+WST+WIR
        WLVSO = WLVST+WSO
        HI    = WSO/WSS
        WSTR  = WST+WIR

*       Photosynthesis, gross and net
*       Explanation in sections 2.1, 3.3, 3.4

        PCGT = INTGRL (PCGT, PCGW, DELT)
        PCNT = INTGRL (PCNT, (PCGW-(RMCR+RGCR)), DELT)
```

```
*       Respiration
*       Explanation in sections 2.4, 2.3

        RMCT = INTGRL (RMCT, RMCR, DELT)
        RCRT = INTGRL (RCRT, (RMCR+RGCR),DELT)

*       Carbohydrate available for growth, export
*       Explanation in sections 3.2, 2.4, 2.3, 2.2

        CELVN = INTGRL (CELVN, INSW (CELV, 1.0 ,-CELVN/DELT), DELT)

*       Leaf area
*       Explanation in section 3.3

        ALV = INTGRL (ALV, (GLA-LLA+GSA), DELT)
        ALVX = ALV

*       Phenological development of the crop
*       Explanation in section 3.1

        DS   = INTGRL (DS, INSW ((DS-1.0), DRV, DRR), DELT)
        PLHT = AFGEN (PLHTT, IPLHTN, DS)

*       Carbon balance check
*       Explanation in section 3.4
```

```
        CKCIN = (WLV-WLVI)*FCLV+(WST-WSTI)*FCST+
     &          (WSO-WSOI)*FCSO+(WRT-WRTI)*FCRT+WIR*0.444
        CKCFL = PCNT*0.2727-(WLVD*FCLV+WRTD*FCRT)
        CKCRD = FUCCHK (CKCIN, CKCFL, TIME)

*       soil temperature and rooting depth

        TPS = INTGRL (TPS, (TPAV-TPS)/5.0, DELT)
        ZRT   = INTGRL (ZRT, GZRT*REAAND (ZRTM-ZRT,1.0-DS), DELT)

*       Determine the finish conditions of the simulation

        IF (DS.GE.2.0)    TERMNL = .TRUE.
        IF (CELVN.GE.3.0) TERMNL = .TRUE.

      ELSE IF (ITASK.EQ.4) THEN

*       Define graph for output
        CALL OUTPLT (1, 'WLV')
        CALL OUTPLT (1, 'WLVT')
        CALL OUTPLT (1, 'WLVST')
        CALL OUTPLT (1, 'WLVSO')

*       use common scale, small plot width for output
        CALL OUTPLT (7, 'Macros simulation model')

      END IF

      ITOLD = ITASK

      RETURN
      END
```

File: A:\SYSSARP\PROGDAT\PLANT.DAT

```
*
* Plant data
*

* TITLE OSIR36.DAT ORYZA SATIVA, RICE, CV IR36
**PHOTOSYNTHESIS AND RESPIRATION; TABLES 3,4,5,8,11,23

PLMXP = 47. ; PLEI  = 0.50
PLMTT = 0.0,0.0,  10.,0.0, 25.,1.00, 30.,1.00, 42.,0.0,  45.,0.0
PLMHT = 0.0,1.00, 1.0,1.0, 2.0,0.99, 3.0,0.86, 4.0,0.71
PLETT = 0.0,1.0,  15.,1.0, 25.,0.90, 35.,0.60, 45.,0.2,  50.,0.01

CRGLV= 1.326; CRGST= 1.326; CRGSO = 1.462; CRGRT = 1.326
CPGLV= 0.408; CPGST= 0.365; CPGSO = 0.357; CPGRT = 0.365
FCLV = 0.419; FCST = 0.431; FCSO = 0.487 ; FCRT = 0.431
RMCLV= 0.02 ; TPR  = 25.  ; Q10  = 2.

**CONSISTENCY CHECK: 12/30*CRGLV=1.0*FCLV+12/44*CPGLV
**BIOMASS PARTITIONING AND AGING; TABLES 7,17

CALVT = 0.0,0.51, 0.5,0.51, 0.6,0.47, 0.7,0.32,
        0.8,0.26, 1.0,0.00, 1.1,0.00, 2.5,0.00
CASTT = 0.0,0.49, 0.5,0.49, 0.6,0.53, 0.7,0.68,
        0.8,0.74, 1.0,1.00, 1.1,0.27, 1.2,0.00, 2.1,0.0
CASST = 0.0,0.86, 0.5,0.86, 0.6,0.86, 0.7,0.95,
        0.8,0.94, 1.0,0.89, 1.1,1.00, 2.5,1.00
FSTR  = 0.25; FEPSO = 0.8; GSORM = 0.5
LLVT = 0.0,0.0, 1.0,0.0, 1.3,0.007, 1.8,0.012, 2.5,0.012
LRTT = 0.0,0.0, 1.0,0.0, 1.3,0.011, 1.8,0.010, 2.5,0.010

**PHENOLOGICAL DEVELOPMENT; TABLES 12,13,14,15,16,19,20,21

DRCV  = 0.013; DRCR  = 0.028
DRVTT = 10.,0.10, 19.,0.80, 25.,1.00, 27.,1.10,
        32.,1.20, 40.,1.00, 45.,1.00
DRRTT = 10.,0.45, 19.,0.75, 25.,0.90, 28.,1.00,
        30.,1.10, 40.,1.10, 45.,1.10
DRDT  = 0.0,1.0,  24.,1.0
DRWT  = 0.0,1.0,   1.,1.0
SLC   = 370.; SSC   = 1000.; WDLV  = 0.015
SLT   = 0.0,0.82, 0.6,1.0, 2.1,1.0
PLHTT = 0.0,0.0 , 1.0,1.0, 2.1,1.0

**WATER RELATIONS AND ROOT GROWTH; TABLES 22,24,25

WSSC  = 0.5; WFSC  = 1.0; FIEC  = 0.65
ZRTMC = 0.7; GZRTC = 0.03
WSET = -1.0,0.0, 0.0,0.0, 1.0,1.0, 2.0,1.0

**TILLER, FLORET AND GRAIN DEVELOPMENT

DST1 = 0.30; DSF1 = 0.70; DSG1 = 0.95
DST2 = 0.75; DSF2 = 0.95; DSG2 = 1.15
TCFT = 15. ; TCFF = 7.  ; TCFG = 3.  ; TCDT = 10.
WTI  = 1.0E-5 ; NFLMXT = 100. ; WGRMX = 23.5E-6
GGRT = 10.0,0.0, 15.0,0.0, 18.0,0.75,
       23.0,1.0, 27.0,0.9, 40.0,0.0
CNTIT = 0.0,5.0E-6,  0.3,5.0E-6, 0.75,25.0E-6,
        1.0,75.0E-6, 2.1,75.0E-6

**INITIALIZATION
WLVI  = 6.8;  WSTI  = 6.8; WSOI  = 0.0
DSI   = 0.18; ZRTI  =0.20
```

File: A:\SYSSARP\PROGDAT\DRL2SS.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE DRL2SS                                                    *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine creates a simple interface for crop growth  *
*           simulation models written in FORTRAN to the soil water      *
*           balance module L2SS and SUSAWA.                             *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                      units  class *
*  ----   ---- -------                                     -----  ----- *
*  control                                                              *
*  ITASK   I4  Determines action of the subroutine,                     *
*              1=initialization, 2=rate calculation,                    *
*              3=integration, 4=terminal                     -     C,I  *
*  IUNITS  I4  Unit number of soil data file,                -     C,IN *
*  IUNITO  I4  Unit number of output file                    -     C,IN *
*  FILES   C*  Name of soil data file                        -     C,IN *
*  OUTPUT  L4  Flag that indicates if output to file is                 *
*              required,                                     -     C,I  *
*  TERMNL  L4  Flag that indicates if simulation should                 *
*              terminate                                     -    C,I,O *
*                                                                       *
*  timing                                                               *
*  TIME    R4  Time of simulation                            d      T   *
*  DATE    R4  Day number of simulation                      d      T   *
*  DELT    R4  Time step of integration                      d      T   *
*                                                                       *
*  soil description available after initial call                       *
*  NL      I4  Number of soil layers from soil water balance -      O   *
*  TKL     R4  Array of thicknesses of soil layers           m      O   *
*  TKLT    R4  Depth of simulated soil                       m      O   *
*  ZRTMS   R4  Maximum rooting depth of soil                 m      O   *
*  WCWPX   R4  Array of water contents at wilting point    cm3/cm3  O   *
*  WCFCX   R4   ,,   ,,  ,,      ,,    ,, field capacity    cm3/cm3  O   *
*  WCSTX   R4   ,,   ,,  ,,      ,,    ,, saturation        cm3/cm3  O   *
*  WDCL    R4  Width of soil cloth                           m      O   *
*  RFSD    R4  Reflection coefficient for dry soil           -      O   *
*                                                                       *
*  dynamic input and output                                             *
*  EVSC    R4  Potential evaporation rate                    mm/d   I   *
*  RAIN    R4  Rainfall / irrigation rate                    mm/d   I   *
*  TRWL    R4  Array of actual transpiration per layer     cm3/cm3  I   *
*  EVSW    R4  Actual (realized) evaporation rate            mm/d   O   *
*  WCLQT   R4  Array of actual water contents              cm3/cm3  O   *
*                                                                       *
*  FATAL ERROR CHECKS (execution terminated, message):                  *
*  Certain sequences of ITASK, see subroutine CHKTSK                    *
*  Checks on consistency of soil layer definition                       *
*  Checks on consistency of soil texture properties                     *
*  Water balance check                                                  *
*                                                                       *
*  SUBROUTINES and FUNCTIONS called:                                    *
*  from TTUTIL : CHKTSK, OUTCOM, OUTDAT, OUTPLT, RDINIT, RDSREA, RDSINT,*
*                RDAREA,  ERROR,  AFGEN, REAAND,   INSW, INTGRL         *
*  from MACROS.LIB: SUSAWA, FUWCHK, FUWCMS                              *
*                                                                       *
*  FILE usage : - Soil definition file FILES opened and closed for      *
*                 ITASK=1, unit numbers used are IUNITS and IUNITS+1    *
*               - Output file with unit IUNITO for output and warnings  *
*-----------------------------------------------------------------------*

*      CALL DRL2SS (ITASK, IUNITS, IUNITO, FILES, OUTPUT, TERMNL,
*     &             TIME,   DATE,   DELT,
```

```
*     &                NL,     TKL,    TKLT, ZRTMS,
*     &              WCWP,    WCFC,    WCST,  WDCL,    RFSD,
*     &              EVSC,    RAIN,    TRWL,  EVSW,  WCLQT)

      SUBROUTINE DRL2SS (ITASK, IUNITS, IUNITO, FILES, OUTPUT, TERMNL,
     &                   TIME,    DATE,    DELT,
     &                     NL,     TKL,    TKLT, ZRTMS,
     &                  WCWPX,   WCFCX,   WCSTX,  WDCL ,  RFSD,
     &                   EVSC,    RAIN,    TRWL,  EVSW , WCLQT)
      IMPLICIT REAL (A-Z)

*     Formal parameters
      INTEGER ITASK, IUNITS, IUNITO
      INTEGER NL
      CHARACTER*(*) FILES
      LOGICAL OUTPUT, TERMNL

      REAL  TKL(10), WCWPX(10), WCFCX(10), WCSTX(10)
      REAL TRWL(10), WCLQT(10)

*     Local and common declarations

      INTEGER I, ITOLD, ITYL, NL1, NL2, NL3, IMND, IND(6)
      REAL TYL(10), WCLEQI(10), WCLMQI(10), WCLQTI(10), WCLCH(10)
      PARAMETER (IMND=20)
      REAL KMSA1T(IMND), KMSA2T(IMND), KMSMXT(IMND), KSTT(IMND)
      REAL MSWCAT(IMND), WCSTT(IMND)

      COMMON /SLDPTH/ ZL(10)
      COMMON /VOLWAT/ WCAD(10), WCFC(10), WCST(10), WCWP(10)
      COMMON /HYDCON/ KMSMX(10), KMSA1(10), KMSA2(10), KST(10)
      COMMON /PFCURV/ MSWCA(10)

*     ground water table
      INTEGER ILZMAX, IZWTB
      PARAMETER (ILZMAX=40)
      REAL ZWTB(ILZMAX)

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:              Old task:
*           initialization         terminal
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done nl. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('DRL2SS', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('L2SS, water balance with impeded drainage')
```

```
*       Initialization section

        CALL RDINIT (IUNITS, IUNITO, FILES)
        CALL RDSREA ('CSC2', CSC2)
        CALL RDSREA ('CSA', CSA)
        CALL RDSREA ('CSB', CSB)
        CALL RDSREA ('DTFX', DTFX)
        CALL RDSREA ('DTMIN', DTMIN)
        CALL RDSREA ('DTMX1', DTMX1)

*       soil compartment data:
*       maximum rooting depth layer thickness, layer soil type
*       and initial moisture content

        CALL RDSREA ('ZRTMS', ZRTMS)
        CALL RDSINT ('NL', NL)
        CALL RDAREA ('TKL', TKL, 10, NL1 )
        CALL RDAREA ('TYL', TYL, 10, NL2)
        CALL RDAREA ('WCLMQI', WCLMQI, 10, NL3)

        IF (NL.LT.1) CALL ERROR ('DRL2SS', 'Number of layers < 1')
        IF (NL.GT.NL1.OR.NL.GT.NL2.OR.NL.GT.NL3) CALL ERROR
     $   ('DRL2SS','NL greater than layer definition')

*       other soil characteristics and option switches
        CALL RDSREA ('WDCL',WDCL)
        CALL RDSREA ('RFSD',RFSD)
        CALL RDSREA ('WL0MX',WL0MX)
        CALL RDSREA ('WL0QTI',WL0QTI)

        CALL RDAREA ('KMSA1T', KMSA1T, IMND, IND(1))
        CALL RDAREA ('KMSA2T', KMSA2T, IMND, IND(2))
        CALL RDAREA ('KMSMXT', KMSMXT, IMND, IND(3))
        CALL RDAREA ('KSTT', KSTT, IMND, IND(4))
        CALL RDAREA ('MSWCAT', MSWCAT, IMND, IND(5))
        CALL RDAREA ('WCSTT', WCSTT, IMND, IND(6))

        DO 5 I=2,6
           IF (IND(1).NE.IND(I)) CALL ERROR ('DRL2SS',
     &       'inconsistent number of soil properties')
5       CONTINUE

*       groundwater interpolation table + table length
        CALL RDAREA ('ZWTB', ZWTB, ILZMAX, IZWTB)

        CALL RDSREA ('WCLIS', WCLIS)

*       delete temporary file used by the input routines
        CLOSE (IUNITS, STATUS='DELETE')

*       initialize tables
        DO 10 I=1,NL
           ITYL = NINT (TYL(I))
           IF (ITYL.LE.0 .OR. ITYL.GT.IMND) CALL ERROR
     &          ('DRL2SS', 'soil type number incorrect')
           KST(I)   = KSTT(ITYL)
           KST(I)   = KSTT(ITYL)
           KMSMX(I) = KMSMXT(ITYL)
           KMSA1(I) = KMSA1T(ITYL)
           KMSA2(I) = KMSA2T(ITYL)
           MSWCA(I) = MSWCAT(ITYL)
           WCST(I)  = WCSTT(ITYL)
           WCFC(I)  = FUWCMS (I,100.0)
           WCWP(I)  = FUWCMS (I,1.6E4)
           WCAD(I)  = FUWCMS (I,1.0E7)

*          copies to calling program
           WCSTX(I) = WCST(I)
```

```
            WCFCX(I) = WCFC(I)
            WCWPX(I) = WCWP(I)
10        CONTINUE

*         groundwater
          ZWI = AFGEN (ZWTB, IZWTB, DATE)
          WL0QT = WL0QTI

*         remaining variables are set to zero
          DO 20 I=1,NL
            WCLQT(I) = 0.
            TRWL(I)  = 0.
20        CONTINUE
          EVSC = 0.

*         initial call to waterbalance subroutine

          CALL SUSAWA (1,WCLQT,WL0QT,NL,TRWL,EVSC,RAIN,ZWI,
     &                 TKL,TYL,1.0,DTMIN,DTMX1,DTFX,WL0MX,ZEQT,
     &                 CSA,CSB,CSC2,WCLCH,WL0CH,WCLEQI,EVSW,RUNOF,DRSL,
     &                 WCUMCH,ZECH,TKLT)

          WCUMI  = 0.0
          WCLISC = REAAND (-WCLIS, (ZWI-TKLT))-0.5

*         initial water contents of the layers and total water content

          DO 30 I=1,NL
            WCLQTI(I) = INSW (WCLISC, WCLEQI(I), WCLMQI(I))
            WCLQT(I)  = WCLQTI(I)
            WCUMI     = WCUMI+WCLQTI(I)*TKL(I)*1000.
30        CONTINUE

*         initialization of rest

          WCUM = WCUMI

          WCL1  = WCLQTI(1)
          WCST1 = WCST(1)
          ZEQTI = 0.02*(1.0-WCL1/WCST1)
          ZEQT  = ZEQTI

          DRSLCU = 0.0
          EVSWCU = 0.0
          RAINCU = 0.0
          RNOFCU = 0.0
          TRWCU  = 0.0
          WCUMCO = 0.0
          WL0CO  = 0.0
          CKWIN  = 0.0
          CKWFL  = 0.0

        ELSE IF (ITASK.EQ.2) THEN

*         Rate calculation section

          ZW = AFGEN (ZWTB, IZWTB, DATE)

          CALL SUSAWA (2,WCLQT,WL0QT,NL,TRWL,EVSC,RAIN,ZW,
     &                 TKL,TYL,1.0,DTMIN,DTMX1,DTFX,WL0MX,ZEQT,
     &                 CSA,CSB,CSC2,WCLCH,WL0CH,WCLEQI,EVSW,RUNOF,DRSL,
     &                 WCUMCH,ZECH,TKLT)

*         total transpiration
          TRW = 0.
          DO 40 I=1,NL
            TRW = TRW+TRWL(I)
40        CONTINUE
```

```
      IF (OUTPUT.OR.TERMNL) THEN
         CALL OUTDAT (2, 0, 'EVSW', EVSW)
         CALL OUTDAT (2, 0, 'EVSC', EVSC)
         CALL OUTDAT (2, 0, 'TRW', TRW)
         CALL OUTDAT (2, 0, 'DRSL', DRSL)
         CALL OUTDAT (2, 0, 'RAIN', RAIN)
         CALL OUTDAT (2, 0, 'RUNOF', RUNOF)
         CALL OUTDAT (2, 0, 'ZEQT', ZEQT)
         CALL OUTDAT (2, 0, 'ZW', ZW)
         CALL OUTDAT (2, 0, 'WL0QT', WL0QT)
         CALL OUTDAT (2, 0, 'WCLQT(1)', WCLQT(1))
         CALL OUTDAT (2, 0, 'WCLQT(3)', WCLQT(3))
         CALL OUTDAT (2, 0, 'WCLQT(5)', WCLQT(5))
         CALL OUTDAT (2, 0, 'WCLQT(7)', WCLQT(7))
         CALL OUTDAT (2, 0, 'WCUM', WCUM)
      END IF

   ELSE IF (ITASK.EQ.3) THEN

*     Integration section

      WL0QT  = INTGRL (WL0QT, WL0CH, DELT)
      DO 50 I=1,NL
         WCLQT(I) = INTGRL (WCLQT(I), WCLCH(I), DELT)
50    CONTINUE

      WCUM   = INTGRL (WCUM, WCUMCH*1000., DELT)
      ZEQT   = INTGRL (ZEQT, ZECH, DELT)
      DRSLCU = INTGRL (DRSLCU, -DRSL, DELT)
      EVSWCU = INTGRL (EVSWCU, -EVSW, DELT)
      RAINCU = INTGRL (RAINCU, RAIN, DELT)
      RNOFCU = INTGRL (RNOFCU, -RUNOF, DELT)
      TRWCU  = INTGRL (TRWCU, -TRW, DELT)
      WCUMCO = INTGRL (WCUMCO, WCUMCH, DELT)
      WL0CO  = INTGRL (WL0CO, WL0CH*1000., DELT)

*     water balance check
      CKWIN  = INTGRL (CKWIN, (WCUMCH+WL0CH)*1000., DELT)
      CKWFL  = INTGRL (CKWFL, (RAIN-RUNOF-EVSW-TRW-DRSL), DELT)
      CKWRD  = FUWCHK (CKWFL, CKWIN, TIME)

   ELSE IF (ITASK.EQ.4) THEN

*     Terminal section

      CALL OUTPLT (1, 'EVSW')
      CALL OUTPLT (1, 'EVSC')
      CALL OUTPLT (1, 'TRW')
      CALL OUTPLT (1, 'DRSL')
      CALL OUTPLT (1, 'RAIN')
      CALL OUTPLT (1, 'RUNOF')

      CALL OUTPLT (6, 'Various water amounts')

      CALL OUTPLT (1, 'WCLQT(1)')
      CALL OUTPLT (1, 'WCLQT(3)')
      CALL OUTPLT (1, 'WCLQT(5)')
      CALL OUTPLT (1, 'WCLQT(7)')
      CALL OUTPLT (7, 'Water contents for different layers')

      CALL OUTPLT (1,'WCUM')
      CALL OUTPLT (7,'Total water in profile')

      CALL OUTPLT (1, 'ZRT')
      CALL OUTPLT (1, 'ZEQT')
      CALL OUTPLT (1, 'ZW')
      CALL OUTPLT (1, 'WL0QT')
```

```
   CALL OUTPLT (6, 'Various other variables')

END IF

ITOLD = ITASK

RETURN
END
```

File: A:\SYSSARP\PROGDAT\DRL2SU.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE DRL2SU                                                    *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine creates a simple interface for crop growth  *
*           simulation models written in FORTRAN to the soil water      *
*           balance module L2SU.                                        *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                    units  class  *
*  ----   ---- -------                                    -----  -----  *
*  control                                                             *
*  ITASK   I4  Determines action of the subroutine,                    *
*              1=initialization, 2=rate calculation,                   *
*              3=integration, 4=terminal                    -     C,I   *
*  IUNITS  I4  Unit number of soil data file,               -     C,IN  *
*  IUNITO  I4  Unit number of output file                   -     C,IN  *
*  FILES   C*  Name of soil data file                       -     C,IN  *
*  OUTPUT  L4  Flag that indicates if output to file is                *
*              required,                                    -     C,I   *
*  TERMNL  L4  Flag that indicates if simulation should                *
*              terminate                                    -    C,I,O  *
*                                                                       *
*  timing                                                              *
*  TIME    R4  Time of simulation                           d      T    *
*  DATE    R4  Day number of simulation                     d      T    *
*  DELT    R4  Time step of integration                     d      T    *
*                                                                       *
*  soil description available after initial call                       *
*  NL      I4  Number of soil layers from soil water balance -     O    *
*  TKL     R4  Array of thicknesses of soil layers          m      O    *
*  TKLT    R4  Depth of simulated soil                      m      O    *
*  ZRTMS   R4  Maximum rooting depth of soil                m      O    *
*  WCWP    R4  Array of water contents at wilting point  cm3/cm3   O    *
*  WCFC    R4   ,,   ,,  ,,      ,,    ,, field capacity  cm3/cm3   O    *
*  WCST    R4   ,,   ,,  ,,      ,,    ,, saturation      cm3/cm3   O    *
*  WDCL    R4  Width of soil cloth                          m      O    *
*  RFSD    R4  Reflection coefficient for dry soil          -      O    *
*                                                                       *
*  dynamic input and output                                            *
*  EVSC    R4  Potential evaporation rate                  mm/d    I    *
*  RAIN    R4  Rainfall / irrigation rate                  mm/d    I    *
*  RAINN   R4  Rainfall / irrigation rate of next day      mm/d    I    *
*  TRWL    R4  Array of actual transpiration per layer  cm3/cm3   I    *
*  EVSW    R4  Actual (realized) evaporation rate          mm/d    O    *
*  WCLQT   R4  Array of actual water contents           cm3/cm3   O    *
*                                                                       *
*  FATAL ERROR CHECKS (execution terminated, message):                  *
*  Certain sequences of ITASK, see subroutine CHKTSK                    *
*  DELT < 1                                                             *
*  Water balance check                                                  *
*                                                                       *
*  SUBROUTINES and FUNCTIONS called:                                    *
*  from TTUTIL : CHKTSK, OUTCOM, OUTDAT, OUTPLT, RDINIT, RDSREA, ERROR, *
*                INSW, INTGRL                                           *
*  from MACROS.LIB: FUWCHK                                              *
*                                                                       *
*  FILE usage : - Soil definition file FILES opened and closed for      *
*               ITASK=1, unit numbers used are IUNITS and IUNITS+1      *
*             - Output file with unit IUNITO for output and warnings    *
*-----------------------------------------------------------------------*

*     CALL DRL2SU (ITASK, IUNITS, IUNITO, FILES, OUTPUT, TERMNL,
*    &             TIME,   DATE,   DELT,
```

```
*     &                NL,    TKL,   TKLT, ZRTMS,
*     &              WCWP,   WCFC,   WCST,  WDCL, RFSD,
*     &              EVSC,   RAIN,  RAINN,  TRWL,
*     &              EVSW,  WCLQT)

      SUBROUTINE DRL2SU (ITASK, IUNITS, IUNITO, FILES, OUTPUT, TERMNL,
     &                   TIME,   DATE,   DELT,
     &                     NL,    TKL,   TKLT, ZRTMS,
     &                   WCWP,   WCFC,   WCST,  WDCL, RFSD,
     &                   EVSC,   RAIN,  RAINN,  TRWL,
     &                   EVSW,  WCLQT)
      IMPLICIT REAL (A-Z)

*     Formal parameters
      INTEGER   ITASK,   IUNITS,  IUNITO, NL
      REAL     TKL(3),  WCWP(3), WCFC(3), WCST(3)
      REAL    TRWL(3), WCLQT(3)
      LOGICAL OUTPUT, TERMNL
      CHARACTER*(*)   FILES

*     Standard local variables
      INTEGER ITOLD

      SAVE

      DATA ITOLD /4/

*     The task that the subroutine should do (ITASK) against the task
*     that was done during the previous call (ITOLD) is checked. Only
*     certain combinations are allowed. These are:
*
*           New task:              Old task:
*           initialization         terminal
*           integration            rate calculation
*           rate calculation       initialization, integration
*           terminal               <any old task>
*     Note: there is one combination that is correct but will not cause
*     calculations to be done nl. if integration is required immediately
*     after initialization.

      CALL CHKTSK ('DRL2SU', IUNITO, ITOLD, ITASK)

      IF (ITOLD.EQ.1.AND.ITASK.EQ.3) THEN
         ITOLD = ITASK
         RETURN
      END IF

      TRWL1 = TRWL(1)
      TRWL2 = TRWL(2)
      TRWL3 = TRWL(3)
      TRW = TRWL1+TRWL2+TRWL3

      IF (ITASK.EQ.1) THEN

*        send title to output file

         CALL OUTCOM ('DRL2SU, water balance SAHEL')

*        Initialization section

         IF (DELT.LT.1.0) CALL ERROR
     &      ('DRL2SU','DELT too small for DRL2SU')
         CALL RDINIT (IUNITS, IUNITO, FILES)

*        Initialization of states

         CALL RDSREA ('WCLI1', WCLI1)
         CALL RDSREA ('WCLI2', WCLI2)
```

```
      CALL RDSREA ('WCLI3', WCLI3)

      CALL RDSREA ('TKL1', TKL1)
      CALL RDSREA ('TKL2', TKL2)
      CALL RDSREA ('TKL3', TKL3)
      TKLT = TKL1+TKL2+TKL3

      WL1I = WCLI1*TKL1*1.0E4
      WL2I = WCLI2*TKL2*1.0E4
      WL3I = WCLI3*TKL3*1.0E4
      WL1  = WL1I
      WL2  = WL2I
      WL3  = WL3I
      WCUM = (WL1+WL2+WL3)/10.0

      WCL1 = WL1/(TKL1*1.E4)
      WCL2 = WL2/(TKL2*1.E4)
      WCL3 = WL3/(TKL3*1.E4)

      TRWT  = 0.0
      DSLR  = 1.0
      CKWFL = 0.0

*     Variables to be initialized in rate section

      CALL RDSREA ('FRNOF', FRNOF)

      CALL RDSREA ('WCFC1', WCFC1)
      CALL RDSREA ('WCFC2', WCFC2)
      CALL RDSREA ('WCFC3', WCFC3)
      CALL RDSREA ('WCAD1', WCAD1)
      CALL RDSREA ('WCAD2', WCAD2)
      CALL RDSREA ('WCAD3', WCAD3)
      CALL RDSREA ('EES', EES)

*     Remaining variables to be initialized for interface

      NL = 3

      TKL(1) = TKL1
      TKL(2) = TKL2
      TKL(3) = TKL3

      CALL RDSREA ('WCWP1', WCWP1)
      CALL RDSREA ('WCWP2', WCWP2)
      CALL RDSREA ('WCWP3', WCWP3)

      WCWP(1) = WCWP1
      WCWP(2) = WCWP2
      WCWP(3) = WCWP3

      WCFC(1) = WCFC1
      WCFC(2) = WCFC2
      WCFC(3) = WCFC3

      CALL RDSREA ('WCST1', WCST1)
      CALL RDSREA ('WCST2', WCST2)
      CALL RDSREA ('WCST3', WCST3)

      WCST(1) = WCST1
      WCST(2) = WCST2
      WCST(3) = WCST3

      CALL RDSREA ('ZRTMS', ZRTMS)
      CALL RDSREA ('WDCL', WDCL)
      CALL RDSREA ('RFSD', RFSD)

      CLOSE (IUNITS, STATUS='DELETE')
```

```
      ELSE IF (ITASK.EQ.2) THEN

*        rate calculation section

*        Available and total soil water
*        Explanation in sections 5.2, 5.4

         WLFL1 = RAIN*(1.0-FRNOF)
         WLFL2 = MAX (0.0, WLFL1-(WCFC1*TKL1*1000.-WL1*0.10)/DELT)
         WLFL3 = MAX (0.0, WLFL2-(WCFC2*TKL2*1000.-WL2*0.10)/DELT)
         WLFL4 = MAX (0.0, WLFL3-(WCFC3*TKL3*1000.-WL3*0.10)/DELT)

*        Evaporation
*        Explanation in section 5.2

         EVSH = MIN (EVSC, (WL1*0.0001-WCAD1*TKL1)*1000./DELT+WLFL1)
         EVSD = MIN (EVSC, 0.6*EVSC*(SQRT(DSLR)-SQRT(DSLR-1.))+WLFL1)
         EVSW = INSW (DSLR-1.1, EVSH, EVSD)

         FEVL1 = MAX (WL1-WCAD1*TKL1*1.0E4, 0.0)*
     &                EXP(-EES*(0.25*TKL1))
         FEVL2 = MAX (WL2-WCAD2*TKL2*1.0E4, 0.0)*
     &                EXP(-EES*(TKL1+(0.25*TKL2)))
         FEVL3 = MAX (WL3-WCAD3*TKL3*1.0E4, 0.0)*
     &                EXP(-EES*(TKL1+TKL2+(0.25*TKL3)))
         FEVLT = FEVL1+FEVL2+FEVL3

         EVSW1 = EVSW*(FEVL1/FEVLT)
         EVSW2 = EVSW*(FEVL2/FEVLT)
         EVSW3 = EVSW*(FEVL3/FEVLT)

         IF (OUTPUT.OR.TERMNL) THEN
            CALL OUTDAT (2, 0, 'TRW', TRW)
            CALL OUTDAT (2, 0, 'EVSC', EVSC)
            CALL OUTDAT (2, 0, 'EVSW', EVSW)
            CALL OUTDAT (2, 0, 'WCL1', WCL1)
            CALL OUTDAT (2, 0, 'WCL2', WCL2)
            CALL OUTDAT (2, 0, 'WCL3', WCL3)
            CALL OUTDAT (2, 0, 'RAIN', RAIN)
         END IF

      ELSE IF (ITASK.EQ.3) THEN

*        integration section

         WL1  = INTGRL (WL1, (WLFL1-WLFL2-EVSW1-TRWL1)*10.0, DELT)
         WL2  = INTGRL (WL2, (WLFL2-WLFL3-EVSW2-TRWL2)*10.0, DELT)
         WL3  = INTGRL (WL3, (WLFL3-WLFL4-EVSW3-TRWL3)*10.0, DELT)
         WCUM = (WL1+WL2+WL3)/10.0

         WCL1 = WL1/(TKL1*1.E4)
         WCL2 = WL2/(TKL2*1.E4)
         WCL3 = WL3/(TKL3*1.E4)

         TRWT = INTGRL (TRWT, TRW, DELT)
         DSLR = INTGRL (DSLR, INSW (RAINN-0.5, 1.0, 1.00001-DSLR)/
     &                   DELT, DELT)

*        Water balance check
*        Explanation in section 5.4

         CKWFL = INTGRL (CKWFL, (WLFL1-EVSW-TRW-WLFL4)*10.0, DELT)
         CKWIN = WL1-WL1I+WL2-WL2I+WL3-WL3I
         CKWRD = FUWCHK (CKWFL, CKWIN, TIME)

      ELSE IF (ITASK.EQ.4) THEN
```

```
*        Define graph for output
         CALL OUTPLT (1, 'TRW')
         CALL OUTPLT (1, 'EVSC')
         CALL OUTPLT (1, 'EVSW')
         CALL OUTPLT (7, 'Tranpiration and evaporation')

         CALL OUTPLT (1, 'WCL1')
         CALL OUTPLT (1, 'WCL2')
         CALL OUTPLT (1, 'WCL3')
         CALL OUTPLT (1, 'RAIN')
         CALL OUTPLT (7, 'Water contents and rain')

      END IF

      WCLQT(1) = WCL1
      WCLQT(2) = WCL2
      WCLQT(3) = WCL3

      ITOLD = ITASK

      RETURN
      END
```

File: A:\SYSSARP\PROGDAT\SOIL.DAT

```
*
* Soil characteristics
*

* TITLE STANDARD DATA LOAMY SOIL, DEEP GROUNDWATER TABLE (APRIL 1988)
**DATA FOR MODULE DRL2SU
TKL1  = 0.2;  TKL2  = 0.3;  TKL3  = 0.5
WCFC1 = 0.36; WCWP1 = 0.11; WCAD1 = 0.01; WCST1 = 0.50
WCFC2 = 0.36; WCWP2 = 0.11; WCAD2 = 0.01; WCST2 = 0.50
WCFC3 = 0.36; WCWP3 = 0.11; WCAD3 = 0.01; WCST3 = 0.50
WCLI1 = 0.36; WCLI2 = 0.36; WCLI3 = 0.36

**DATA FOR MODULE DRL2SS
NL     = 10
TKL    = 10*0.10
TYL    = 10*13.
WCLMQI = 10*0.36
WCLIS  = -1.;   WL0MX =0.02
ZWTB   =0.,3.0, 366.,3.0
WL0QTI =0.0
DTMIN  =0.001; DTMX1 =0.1; DTFX  =0.03

**SURFACE AND OTHER SOIL CHARACTERISTICS
FRNOF = 0.0; RFSD  = 0.2; WDCL  = 0.05; ZRTMS = 0.9
EES   = 20.; CSC2  = 0.1; CSA   = 0.15; CSB   = 10.

**CHARACTERISTICS SOIL TYPES 1-20
KMSA1T = 0.1960, 0.1385, 0.0821, 0.0500, 0.0269, 0.0562, 0.0378,
         0.0395, 0.0750, 0.0490, 0.0240, 0.0200, 0.0231, 0.0353,
         0.0237, 0.0248, 0.0274, 0.0480, 0.0380, 0.1045
KMSA2T = 0.08, 0.63,  3.30, 10.90, 15.00,  5.26,  2.10,
        16.40, 0.24, 22.60, 26.50, 47.30, 14.40, 33.60,
         3.60, 1.69,  2.77, 28.20,  4.86,  6.82
KMSMXT = 80.0,  90.0, 125.0, 175.0, 165.0, 100.0, 135.0,
        200.0, 150.0, 130.0, 300.0, 300.0, 300.0, 200.0,
        300.0, 300.0, 300.0,  50.0,  80.0 , 50.0
KSTT = 1120.00, 300.00, 110.00, 50.00, 1.00, 2.30,   .36,
         26.50,  16.50,  14.50, 12.00, 6.50, 5.00, 23.50,
          1.50,    .98,   3.50,  1.30,  .22, 5.30
MSWCAT = 0.0853, 0.0450, 0.0366, 0.0255, 0.0135, 0.0153, 0.0243,
         0.0299, 0.0251, 0.0156, 0.0186, 0.0165, 0.0164, 0.0101,
         0.0108, 0.0051, 0.0085, 0.0059, 0.0043, 0.0108
WCSTT =  0.3950, 0.3650, 0.3500, 0.3640, 0.4700, 0.3940, 0.3010,
         0.4390, 0.4650, 0.4550, 0.5040, 0.5090, 0.5030, 0.4320,
         0.4750, 0.4450, 0.4530, 0.5070, 0.5400, 0.8630
```

File: A:\SYSSARP\PROGDAT\SUPSEV.FOR

```
*-----------------------------------------------------------------------*
*  SUBROUTINE SUPSEV                                                    *
*  Authors: Daniel van Kraalingen                                       *
*  Date   : Jan 1990                                                    *
*  Version: 1                                                           *
*  Purpose: This subroutine calculates potential soil evaporation taking *
*           a standing crop into account. This is a strict rate         *
*           calculating subroutine.                                     *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                    units  class  *
*  ----   ---- -------                                    -----  -----  *
*  RDTC    R4  Global radiation outside atmosphere      J/m2/d    I     *
*  RDTM    R4  Global radiation                         J/m2/d    I     *
*  TPL     R4  Minimum temperature                   degrees Celsius I  *
*  TPH     R4  Maximum temperature                   degrees Celsius I  *
*  HUAA    R4  Vapour pressure                          kPa       I     *
*  WDS     R4  Average wind speed                       m/s       I     *
*  WCL1    R4  Water content of uppermost soil layer  cm3/cm3     I     *
*  WCST1   R4  Water content at saturation of uppermost                 *
*              soil layer                              cm3/cm3    I     *
*  RFSD    R4  Reflection coefficient for dry soil       -        I     *
*  WDCL    R4  Width of soil cloth                       m        I     *
*  WDLV    R4  Width of leaves                           m        I     *
*  ALV     R4  Leaf area index                         m2/m2      I     *
*  PLHT    R4  Plant height                              m        I     *
*  EVSC    R4  Potential evaporation rate               mm/d      O     *
*                                                                       *
*  SUBROUTINES and FUNCTIONS called:                                    *
*  from MACROS.LIB: FUVP, SUEVTR, FUWRED, FURSC                         *
*                                                                       *
*  FILE usage : none                                                    *
*-----------------------------------------------------------------------*

*      CALL SUPSEV (RDTC,    RDTM,  TPL,  TPH, HUAA, WDS,
*     &         WCLQT(1), WCST(1), RFSD, WDCL,
*     &             WDLV,    ALVX, PLHT,
*     &             EVSC)

      SUBROUTINE SUPSEV (RDTC,  RDTM,  TPL,  TPH, HUAA, WDS,
     &                   WCL1, WCST1, RFSD, WDCL,
     &                   WDLV,   ALV, PLHT,
     &                   EVSC)
      IMPLICIT REAL (A-Z)
      SAVE

*     Potential evaporation soil
*     Explanation in section 5.1

      TPAV  = (TPL+TPH)/2.0
      TPAD  = (TPH+TPAV)/2.0
      VPA   = MIN (FUVP (TPAD), HUAA)
      WDSAV = MAX (0.2, WDS)

      RFS  = RFSD*(1.0-0.5*WCL1/WCST1)
      WDSS = FUWRED (WDLV, ALV, PLHT, WDSAV)
      RSBS = 172.*SQRT (WDCL/WDSS)
      RSTS = FURSC (WDSAV, 1.0, (0.1*PLHT), (0.63*PLHT))
      CALL SUEVTR (RDTC, RDTM, RFS, 1.00, TPAV, VPA, 0.0, RSBS, RSTS,
     &             EVSPR, EVSPD)
      EVSC = EVSPR*EXP (-0.5*ALV)+EVSPD

      RETURN
      END
```

Appendix B: MACROS library

In this appendix listings are given of the modified subroutines and functions of Appendix B of Penning de Vries *et al.* (1989).

The source library is stored in four separate files on Disk 1. The directory and file order in this appendix is as follows:

A:\SYSSARP\SUBFUN\MACROS\SNEW.FOR	New and modified functions and subroutines,
A:\SYSSARP\SUBFUN\MACROS\SFUN.FOR	Functions,
A:\SYSSARP\SUBFUN\MACROS\SSUB1.FOR	Subroutines first part,
A:\SYSSARP\SUBFUN\MACROS\SSUB2.FOR	Subroutines second part.

One object library has been created of these four files, compatible with Microsoft FORTRAN 4.10 or higher. This library is stored on Disk 3 and is called:

A:\SYSSARP\FORTRAN\MACROS.LIB

File: A:\SYSSARP\SUBFUN\MACROS\SNEW.FOR

```
*     This file contains one new function (FUTP2) and three
*     strongly modified subroutines and functions.
*     These changes were necessary for the FORTRAN version
*     of the MACROS programs.

      REAL FUNCTION FUCCHK(CKCIN,CKCFL,TIME)
C check on crop carbon balance. used in L1D, L1Q. 03/87
C Version changed to be compatible with FORTRAN version of MACROS
      COMMON /LOGCOM/ IUNITL, DUMMY, IRUN
      SAVE
      FUCCHK=2.0*(CKCIN-CKCFL)/(CKCIN+CKCFL+1.E-10)
      IF(ABS(FUCCHK).GT.0.01) THEN
         IF (IUNITL.EQ.0) THEN
            WRITE (6,20)
     $        ' * * * error in carbon balance, please check * * *'
            WRITE (6,30) ' CKCRD=',FUCCHK,  ' CKCIN=',CKCIN,
     $                   ' CKCFL=',CKCFL,' AT  TIME=',TIME
         ELSE
            WRITE (*,10) ' Run =',IRUN
            WRITE (*,20)
     $        ' * * * error in carbon balance, please check * * *'
            WRITE (*,30) ' CKCRD=',FUCCHK,  ' CKCIN=',CKCIN,
     $                   ' CKCFL=',CKCFL,' AT  TIME=',TIME
            WRITE (IUNITL,10) ' Run =',IRUN
            WRITE (IUNITL,20)
     $        ' * * * error in carbon balance, please check * * *'
            WRITE (IUNITL,30) ' CKCRD=',FUCCHK,  ' CKCIN=',CKCIN,
     $                      ' CKCFL=',CKCFL,' AT  TIME=',TIME
         END IF
      END IF
      RETURN
10    FORMAT (/,/,A,I3)
20    FORMAT (A)
30    FORMAT (A,F6.3,A,F8.2,A,F8.2,A,F6.1)
      END

      REAL FUNCTION FUWCHK(CKWFL,CKWIN,TIME)
C check on soil water balance. used in L2SU, L2SS. 3/87
C Version changed to be compatible with FORTRAN version of MACROS
      COMMON /LOGCOM/ IUNITL, DUMMY, IRUN
      SAVE
      FUWCHK=2.0*(CKWIN-CKWFL)/(CKWIN+CKWFL+1.E-10)
      IF(ABS(FUWCHK).GT.0.01.AND.ABS(CKWIN).GT.0.2) THEN
         IF (IUNITL.EQ.0) THEN
            WRITE (6,20)
     $        ' * * * error in water balance, please check * * *'
            WRITE (6,30) ' CKWRD=',FUWCHK,  ' CKWIN=',CKWIN,
     $                   ' CKWFL=',CKWFL,' AT  TIME=',TIME
         ELSE
            WRITE (*,10) ' Run =',IRUN
            WRITE (*,20)
     $        ' * * * error in water balance, please check * * *'
            WRITE (*,30) ' CKWRD=',FUWCHK,  ' CKWIN=',CKWIN,
     $                   ' CKWFL=',CKWFL,' AT  TIME=',TIME
            WRITE (IUNITL,10) ' Run =',IRUN
            WRITE (IUNITL,20)
     $        ' * * * error in water balance, please check * * *'
            WRITE (IUNITL,30) ' CKWRD=',FUWCHK,  ' CKWIN=',CKWIN,
     $                      ' CKWFL=',CKWFL,' AT  TIME=',TIME
         END IF
      END IF
      RETURN
10    FORMAT (/,/,A,I3)
20    FORMAT (A)
30    FORMAT (A,F6.3,A,F8.2,A,F8.2,A,F6.1)
      END

      SUBROUTINE SUERRM (MNR,X,XMIN,XMAX,NUNIT)
C     modified version of SUERRM to FORTRAN version of MACROS
      IMPLICIT REAL (A-Z)
      INTEGER IUNIT
      LOGICAL UNDER, OVER
      CHARACTER*1 DUMMY

C     special declarations to make log-files possible
      INTEGER IUNITL, IRUN
      COMMON /LOGCOM/ IUNITL, TIME, IRUN
      SAVE

      UNDER = .FALSE.
      OVER  = .FALSE.
      IF ((X.LT.XMIN*0.99).AND.(XMIN.NE.-99.)) UNDER = .TRUE.
      IF ((X.GT.XMAX*1.01).AND.(XMAX.NE.-99.)) OVER  = .TRUE.
      IF (UNDER.OR.OVER) THEN
         IF (IUNITL.EQ.0.AND.NUNIT.GT.0.) THEN
            IUNIT = NINT (NUNIT)
            WRITE (IUNIT,20)
     $        ' *** fatal error in variable or parameter value *** '
            WRITE (IUNIT,20)
     $        ' message number,    value,      minimum,      maximum'
            WRITE (IUNIT,30) MNR,X,XMIN,XMAX
         ELSE
            IUNIT = IUNITL
            WRITE (IUNIT,10) ' Run =',IRUN
            WRITE (IUNIT,20)
     $        ' *** fatal error in variable or parameter value *** '
            WRITE (IUNIT,20)
     $        ' message number,    value,      minimum,      maximum'
            WRITE (IUNIT,30) MNR,X,XMIN,XMAX

            WRITE (*,10) ' Run =',IRUN
            WRITE (*,20)
     $        ' *** fatal error in variable or parameter value *** '
            WRITE (*,20)
     $        ' message number,    value,      minimum,      maximum'
            WRITE (*,30) MNR,X,XMIN,XMAX
            WRITE (*,5)  ' Press <RETURN>'
            READ (*,5) DUMMY
            STOP
         END IF
      END IF
      RETURN
5     FORMAT (A)
10    FORMAT (/,/,A,I3)
20    FORMAT (A)
30    FORMAT (F8.1,8X,G11.5,2X,G11.5,2X,G11.5)
      END

      REAL FUNCTION FUTP2 (DTIME,TPHY,TPL,TPH,TPLT,FA,FB,FC,FD)
C approximates daily course of air temperature. in L1Q. 9/85
C This version was taken from the original version of FUTP and
C changed to use it with the FORTRAN versions of MACROS 11/89.

      IMPLICIT REAL (A-Z)
      SAVE
      IF (DTIME.GE.0 .AND. DTIME.LE.0.2) THEN
         FUTP2 = FA*TPHY+(1.-FA)*TPL
      ELSE IF (DTIME.GT.0.2 .AND. DTIME.LE.0.4) THEN
         FUTP2 = FB*TPH+(1.-FB)*TPL
      ELSE IF (DTIME.GT.0.4 .AND. DTIME.LE.0.6) THEN
         FUTP2 = FC*TPH+(1.-FC)*TPL
      ELSE
         FUTP2 = FD*TPH+(1.-FD)*TPLT
      END IF

      RETURN
      END
```

File: A:\SYSSARP\SUBFUN\MACROS\SFUN.FOR

```
*     This file with functions and subroutines is identical to the
*     Appendix B in the book with the exception that all the functions
*     are made real explicitly (REAL FUNCTION ... instead of FUNCTION...)
*     and in each routine a SAVE statement has been added so that with any
*     compiler, variables retain their values between subsequent call.
*     This may not be necessary in some cases but execution speed
*     increases considerably with a compiler that does not save
*     variables by default (such as MacFortran/020 on the Apple
*     Macintosh). The SAVE statement also garantuees compatibility
*     with future FORTRAN standards.

      REAL FUNCTION FUPHOT(PLMX,PLEA,ALV,RDTM,DATE,LAT)
```

```
C computes canopy photosynthesis. used in L1D, L1Q. 8/87
      IMPLICIT REAL (A-Z)
      INTEGER IT,I
      SAVE
      DATA KDIF/0.7155/,PI/3.1415926/,SCV/0.200/,GAUSR/0.3872893/
      CALL SUERRM(1.1,ALV, 0., 25.,6.)
      CALL SUERRM(1.2,PLMX,0.,100.,6.)
      CALL SUERRM(1.3,PLEA,0., 0.7,6.)
      CALL SUASTC(DATE,LAT,RDTM,RDTC,FRDIF,COSLD,SINLD,DSINBE,SOLC,DLA)
      GDFG  =0.
      IF(PLMX*PLEA*ALV.LE.0.0) GOTO 50
      ALVL  =AMIN1(10.,ALV)
      REFH  =(1.-SQRT(1.0-SCV))/(1.+SQRT(1.-SCV))
      DO 40 IT=1,3
        HOUR  =12.0+DLA*0.5*(0.5+(IT-2)*GAUSR)
        SINB  =AMAX1(0.,SINLD+COSLD*COS(2.*PI*(HOUR+12.)/24.))
        REFS  =REFH*2./(1.+1.6*SINB)
        PAR   =0.5*RDTM*SINB*(1.0+0.4*SINB)/DSINBE
        PARDIF=AMIN1(PAR,SINB*FRDIF*(RDTM/RDTC)*0.5*SOLC)
        PARDIR=PAR-PARDIF
        KDIRBL=(0.5/SINB)*KDIF/(0.8*SQRT(1.-SCV))
        KDIRT =KDIRBL*SQRT(1.-SCV)
        FGROS =0.
        DO 30 I=1,3
          ALVC  =0.5*ALVL+GAUSR*(I-2)*ALVL
          VISDF =(1.-REFS)*PARDIF*KDIF*EXP(-KDIF*ALVC)
          VIST  =(1.-REFS)*PARDIR*KDIRT*EXP(-KDIRT*ALVC)
          VISD  =(1.-SCV)*PARDIR*KDIRBL*EXP(-KDIRBL*ALVC)
          VISSHD=VISDF+VIST-VISD
          FGRSH =PLMX*(1.-EXP(-VISSHD*PLEA/PLMX))
          VISPP =(1.-SCV)*PARDIR/SINB
          IF (VISPP.LE.0.) GO TO 10
          FGRSUN=PLMX*(1.-(PLMX-FGRSH)*(1.-EXP(-VISPP*PLEA/PLMX))/
     $           (PLEA*VISPP))
          GO TO 20
 10       FGRSUN=FGRSH
 20       CONTINUE
          FSSLA =EXP(-KDIRBL*ALVC)
          FGL   =FSSLA*FGRSUN+(1.-FSSLA)*FGRSH
          IF(I.EQ.2) FGL   =FGL*1.6
          FGROS =FGROS+FGL
 30     CONTINUE
        FGROS =FGROS*ALVL/3.6
        IF(IT.EQ.2) FGROS =FGROS*1.6
        GDFG  =GDFG+FGROS
 40   CONTINUE
 50   FUPHOT=GDFG*DLA/3.6
      RETURN
      END

      REAL FUNCTION FURSC(WDS,ALV,PLHT,ZREF)
C calculates canopy resistance upper layers. in L2C. 4/87
      SAVE
      ZR    =AMAX1(ZREF,PLHT+1.)
      D     =AMAX1(0.1,0.63*PLHT)
      ZNOT  =AMAX1(0.05,0.1*PLHT)
      ALVX  =AMAX1(1.,ALV)
      WDSX  =AMAX1(0.2,WDS)
      FURSC =0.74*(ALOG((ZR-D)/ZNOT))**2/(0.16*WDSX)*ALVX
      RETURN
      END

      REAL FUNCTION FUTP(IDATE,DTIME,TPHT,TPLT,FA,FB,FC,FD)
C approximates daily course of air temperature. in L1Q. 9/85
       DIMENSION TPHT(365),TPLT(365)
      SAVE
      IF(IDATE.EQ.366) IDATE=365
      FUTP  =FA*TPHT(MAX0(1,IDATE-1))+(1.-FA)*TPLT(IDATE)
      IF(DTIME.GT.0.2) FUTP  =FB*TPHT(IDATE)+(1.-FB)*TPLT(IDATE)
      IF(DTIME.GT.0.4) FUTP  =FC*TPHT(IDATE)+(1.-FC)*TPLT(IDATE)
      IF(DTIME.GT.0.6) FUTP  =FD*TPHT(IDATE)+
     $                        (1.-FD)*TPLT(MIN0(365,IDATE+1))
      RETURN
      END

      REAL FUNCTION FUVP(TP)
C vapour pressure (kPa) relation to temperature. in L1Q, L2C. 9/85
      SAVE
      FUVP  =0.100*6.11*EXP(17.47*TP/(TP+239.))
      RETURN
      END

      REAL FUNCTION FUWCMS(I,MS)
C converts moisture suction into water contents. in L2SS. 9/87
      REAL MS
      SAVE
      CALL SUWCMS(I,2,WCL,MS)
      FUWCMS=WCL
      RETURN
      END

      REAL FUNCTION FUWRED(WDLV,ALV,PLHT,WDS)
C calculates windspeed near soil surface. in L2C. 9/87
      IMPLICIT REAL (A-Z)
      SAVE
      PLHTX =AMAX1(0.05,PLHT)
      ALVX  =AMAX1(0.01,ALV)
      MIXL  =SQRT(1.2732*AMAX1(0.005,WDLV)/(ALVX/PLHTX))
      A     =SQRT(0.2*ALVX*PLHTX/(2.*MIXL*0.5))
      FUWRED=AMAX1(0.2,WDS)*EXP(-A*(1.0-0.05/PLHTX))
      RETURN
      END

      REAL FUNCTION FUWS(TRC,ALV,WCL,WSSC,WFSC,WCWP,WCFC,WCST)
C computes reduction of water uptake, used in L2SU, L2SS. 5/87
      SAVE
      DATA A,B,ALVMAX/0.76,0.15,2./
      IF(WCL .LE. WCFC) THEN
        SDPF  =1./(A+B*ALVMAX*TRC/(ALV+1.E-10))-(1.-WSSC)*0.4
        IF(WSSC.LT.0.6) THEN
          SDPF  =SDPF+0.025*AMIN1(0.,ALVMAX*TRC/(ALV+1.E-10)-6.)/
     $          (1.+5.*WSSC+4.*WSSC*WSSC)
        ENDIF
        WCX   =WCWP+(WCFC-WCWP)*(1.00-AMIN1(1.,AMAX1(0.,SDPF)))
        FUWSX =(WCL-WCWP)/(WCX-WCWP+1.E-10)
      ELSE
        FUWSX =1.-(1.-WFSC)*(WCL-WCFC)/(WCST-WCFC+1.E-10)
      ENDIF
      FUWS  =AMIN1(1.,AMAX1(0.,FUWSX))
      RETURN
      END
```

File: A:\SYSSARP\SUBFUN\MACROS\SSUB1.FOR

```
*     Minor modification were made in SUASTR and SUINTG.
*     see Chapter 7.

      SUBROUTINE SUASTC(DATE,LAT,RDTM, RDTC,FRDIF,COSLD,
     $           SINLD,DSINBE,SOLC,DLA)
C astronomical standard computations. used in L1D, L1Q. 5/87
      IMPLICIT REAL (A-Z)
      SAVE
      DATA PI/3.1415926/,RAD/0.0174533/
      DEC   =-ASIN(SIN(23.45*RAD)*COS(2.*PI*(DATE+10.)/365.))
      COSLD =COS(DEC)*COS(LAT*RAD)
      SINLD =SIN(DEC)*SIN(LAT*RAD)
      AOB   =SINLD/COSLD
      CALL SUERRM(2.1,DATE,0.,365.,6.)
      CALL SUERRM(2.2,AOB,-1.0,1.0,6.)
      DLA   =12.*(1.+2.0*ASIN(AOB)/PI)
      DSINBE=3600.*(DLA*(SINLD+0.4*(SINLD*SINLD+COSLD*COSLD*0.5))+
     $       12.0*COSLD*(2.0+3.0*0.4*SINLD)*SQRT(1.-AOB*AOB)/PI)
      DSINB =3600.*(SINLD*DLA+24./PI*COSLD*SQRT(1.-AOB**2))
      SOLC  =1370.*(1.0+0.033*COS(2.*PI*DATE/365.))
      RDTC  =SOLC*DSINB
      CALL SUERRM(2.3,RDTM,0.,RDTC,6.)
      ATMTR =RDTM/RDTC
      IF(ATMTR.GT.0.75) FRDIF =0.23
      IF(ATMTR.LE.0.75.AND.ATMTR.GT.0.35) FRDIF =1.33-1.46*ATMTR
      IF(ATMTR.LE.0.35.AND.ATMTR.GT.0.07) FRDIF =1.-2.3*(ATMTR-0.07)**2
      IF(ATMTR.LE.0.07) FRDIF =1.00
      RETURN
      END
```

```
      SUBROUTINE SUASTR(DATE,LAT,RDTC,DLA,DLP)
C computes daylength, daily total radiation clear. in L1D, L1Q. 5/87
      IMPLICIT REAL (A-Z)
      SAVE
      DATA INSP/-4.0/,PI/3.1415926/,RAD/0.0174533/
      CALL SUASTC(DATE,LAT,1.0,RDTC,FRDIF,COSLD,SINLD,DSINBE,SOLC,DLA)
      DLP   =12.*(PI+2.*ASIN((-SIN(INSP*RAD)+SINLD)/COSLD))/PI
      RETURN
      END

      SUBROUTINE SUCONV(SWICH2,TKL,ZL,ZLT,RAIN,ZW,WL0,WL0MX,
     $            RUNOF,TRWL,EVSW,EVSC,DRSL,NL)
C converts units between main program and subroutines. in L2SS. 4/88
      IMPLICIT REAL (A-Z)
      INTEGER I,NL,SWICH2
      DIMENSION TKL(10),ZL(10),TRWL(10)
      SAVE
      F1    =0.01
      F2    =10.0
      IF(SWICH2.EQ.1) F1=1./F1
      IF(SWICH2.EQ.1) F2=1./F2
      DO 10 I=1,NL
         TKL(I)   =TKL(I) *F1
         ZL(I)    =ZL(I)  *F1
         TRWL(I)  =TRWL(I)*F2
 10   CONTINUE
      ZLT   =ZLT  *F1
      ZW    =ZW   *F1
      WL0   =WL0  *F1
      WL0MX =WL0MX*F1
      RAIN  =RAIN *F2
      EVSW  =EVSW *F2
      EVSC  =EVSC *F2
      RUNOF =RUNOF*F2
      DRSL  =DRSL *F2
      RETURN
      END

*-----------------------------------------------------------------
*     Subroutine SUERRM has been removed as it had to be modified
*-----------------------------------------------------------------

      SUBROUTINE SUEVTR(RDTC,RDTM,RF,FRD,TPAD,VPA,RSL,RSB,RST,EVPR,EVPD)
C potential evapotranspiration rates crop, soil. used in L2C. 7/87
      SAVE
      CALL SUERRM(3.1,FRD,0.,1.,6.)
      VPAS  =FUVP(TPAD)
      CALL SUERRM(3.2,VPAS,0.0,12.55,6.)
      CALL SUERRM(3.3,VPA, 0.0,VPAS, 6.)
      SLOPE =4158.6*10.*VPAS/(TPAD+239.)**2
      APSCH =0.67*(RSB+RST+RSL)/(RSB/0.93+RST)
      RLWI  =4.8972E-3*(TPAD+273.)**4*(0.618+0.0365*SQRT(10.*VPA))
      RLWO  =4.8972E-3*1.00*(TPAD+273.)**4
      RDTN  =RDTM*(1.-RF)-(RLWO-RLWI)*(RDTM/(0.75*RDTC))*FRD
      EVPR  =0.001*RDTN*SLOPE/((SLOPE+APSCH)*2390.)
      DRYP  =(VPAS-VPA)*10.*1200./(RSB+RST) *FRD
      EVPD  =86400.*0.001*DRYP/((SLOPE+APSCH)*2390.)
      RETURN
      END

      SUBROUTINE SUGRHD(TKL,NL,HGT,HGB)
C calculates  gravitational head at interfaces. in L2SS. 9/87
      DIMENSION TKL(10),HGT(10),HGB(10)
      SAVE
      HGT(1)    =0.
      HGB(1)    =-TKL(1)
      DO 10 I=2,NL
        HGT(I)    =HGT(I-1)-TKL(I-1)
        HGB(I)    =HGB(I-1)-TKL(I)
 10   CONTINUE
      RETURN
      END

      SUBROUTINE SUINTG(HPBP,SWICH4,SWICH5,HPP,FLX,TKL,WCL,
     $            MS,DTMIN,DTMX1,DTMX2,DTFX,INXSAT,DHH,
     $            JTOT,FLXSQ2,WL0,WL0MX,NL,DELT,DT)
C calculates SAWAH-timestep, integrates rates during day;in L2SS;08/88
      IMPLICIT REAL (A-Z)
      INTEGER NL,I,IX,ITEL1,ITEL2,SWICH4,SWICH5,INXSAT,JTOT
      COMMON /VOLWAT/ WCAD(10),DUMMY1(10),WCST(10),DUMMY2(10)
      DIMENSION WCL(10),HPP(11),INXSAT(10,10)
      DIMENSION WCLRCH(10),FLX(11),TKL(10),MS(10),DHH(10)
      SAVE
      DATA TINY/1.0E-10/
      HPTP  =WL0
      IF(HPTP.LE.0..OR.FLX(1).LE.0.) THEN
         DTSRF =DELT
      ELSE
         DTSRF =AMAX1(TINY,HPTP/(FLX(1)))
      ENDIF
      DT    =AMIN1(DELT,DTSRF,DTMX1,DTMX2)
      DO 10 I=1,NL
         WCLRCH(I)=(FLX(I)-FLX(I+1))/TKL(I)
         IF(WCLRCH(I).LT.-TINY)SATTIM=-(WCL(I)-WCAD(I)-TINY)/WCLRCH(I)
         IF(WCLRCH(I).GT. TINY)SATTIM= (WCST(I)-WCL(I)-TINY)/WCLRCH(I)
         IF(ABS(WCLRCH(I)).LT.TINY) SATTIM=DELT
         DT    =AMIN1(SATTIM,DT)
 10   CONTINUE
      IF(SWICH5.EQ.2) THEN
         DT    =AMIN1(DT,DTFX)
         IF (JTOT.GT.0) THEN
            IF (DHH(JTOT).GE.-TINY) THEN
               IX    =INXSAT(JTOT,1)-1
               IF(IX.LT.1) THEN
                  CONTINUE
               ELSEIF(FLX(IX+1).GT.-0.1) THEN
                  CONTINUE
               ELSEIF(FLXSQ2.LE.0.) THEN
                  CONTINUE
               ELSE
                  MSAL  =TKL(IX)/2.
                  CALL SUWCMS(IX,0,WIX2,MSAL)
                  MSACT =MS(IX)
                  CALL SUWCMS(IX,0,WIX1,MSACT)
                  IF(WIX2.LE.WIX1) THEN
                    CONTINUE
                  ELSE
                     DLIM  =-(WIX2-WIX1)*TKL(IX)/FLX(IX+1)
                     DT    =AMIN1(DT,AMAX1(DLIM,TINY))
                  ENDIF
               ENDIF
            ENDIF
         ENDIF
         GOTO 100
      ELSE
         CONTINUE
      ENDIF
      GOTO 30
 20   DT    =DT/2.
 30   IF(DT.LT.DTMIN) GOTO 100
      IF(ABS(WCLRCH(1)).LT.TINY) GOTO 40
      IF(HPTP.GT.TINY) THEN
         MST1  =-(HPTP-DT*(FLX(1)))
      ELSE
         GOTO 40
      ENDIF
      WCT2  =WCL(1)+DT*FLX(1)/TKL(1)
      IF((WCT2-WCAD(1)).LE.-TINY) GOTO 20
      IF(WCT2.GE.WCST(1)) THEN
         MST2  =WCT2-WCST(1)
      ELSE
         CALL SUWCMS(1,SWICH4,WCT2,MST2)
      ENDIF
      DELZ  =0.5*TKL(1)
      DH    =-MST2+MST1-DELZ
      CH    =-DH*FLX(1)
      IF(CH.LT.-TINY) GOTO 20
 40   DO 70 I=2,NL
         GOTO 60
 50      DT    =DT/2.
 60      ITEL1 =0
         ITEL2 =0
         IF(DT.LT.DTMIN) GOTO 100
```

```
      IF(ABS(WCLRCH(I-1)).LT.TINY.AND.ABS(WCLRCH(I)).LT.TINY) GOTO 70
      IF(ABS(WCLRCH(I-1)).LT.TINY) THEN
         IF(ABS(WCL(I-1)-WCST(I-1)).LT.TINY) THEN
            DELZ  =0.5*TKL(I)
            MST1  =0.
            ITEL2 =1
         ELSE
            DELZ  =0.5*(TKL(I-1)+TKL(I))
            MST1  =MS(I-1)
         ENDIF
      ELSE
         DELZ  =0.5*(TKL(I-1)+TKL(I))
         WCT1  =WCL(I-1)-DT*FLX(I)/TKL(I-1)
         IF((WCT1-WCAD(I-1)).LE.-TINY) GOTO 50
         IF(WCT1.GE.WCST(I-1)) THEN
            MST1  =WCT1-WCST(I-1)
         ELSE
            CALL SUWCMS(I-1,SWICH4,WCT1,MST1)
         ENDIF
      ENDIF
      IF(ABS(WCLRCH(I)).LT.TINY) THEN
         IF(ABS(WCL(I)-WCST(I)).LT.TINY) THEN
            DELZ  =0.5*TKL(I-1)
            MST2  =0.
            IF(HPP(I).GT.TINY) THEN
               IF(HPP(I).GE.TKL(I-1)) ITEL1 =1
                  DELZ  =0.5*TKL(I-1)
                  MST2  =0.
               ENDIF
            ELSE
               DELZ  =0.5*(TKL(I-1)+TKL(I))
               MST2  =MS(I)
            ENDIF
         ELSE
            DELZ  =0.5*(TKL(I-1)+TKL(I))
            IF(ITEL2.EQ.1) DELZ  =0.5*TKL(I)
            WCT2  =WCL(I)+DT*FLX(I)/TKL(I)
            IF((WCT2-WCAD(I)).LE.-TINY) GOTO 50
            IF(WCT2.GE.WCST(I)) THEN
               MST2  =WCT2-WCST(I)
            ELSE
               CALL SUWCMS(I,SWICH4,WCT2,MST2)
            ENDIF
         ENDIF
         DH    =-MST2+MST1-DELZ
         CH    =-DH*FLX(I)
         IF(ITEL1.EQ.1) CH    =+1.
         IF(CH.LT.-TINY) GOTO 50
70    CONTINUE
      GOTO 90
80    DT    =DT/2.
      IF(DT.LT.DTMIN) GOTO 100
90    IF(ABS(WCLRCH(NL)).LT.TINY) GOTO 100
      ITEL1 =0
      ITEL2 =0
      WCT1  =WCL(NL)-DT*FLX(NL+1)/TKL(NL)
      IF((WCT1-WCAD(NL)).LE.-TINY) GOTO 80
      IF(WCT1.GE.WCST(NL)) THEN
         MST1  =WCT1-WCST(NL)
      ELSE
         CALL SUWCMS(NL,SWICH4,WCT1,MST1)
      ENDIF
      DELZ  =0.5*TKL(NL)
      IF(HPBP.GT.TINY.AND.WCL(NL).LT.WCST(NL)) THEN
         IF(HPBP.GE.TKL(NL)) ITEL1  =1
         DELZ  =TKL(NL)
         MST2  =-HPBP
      ENDIF
      DH    =-MST2+MST1-DELZ
      CH    =-DH*FLX(NL+1)
      IF(ITEL1.EQ.1) CH    =+1
      IF(CH.LT.-TINY) GOTO 80
100   WL0   =AMAX1(0.,WL0-DT*FLX(1))
      DO 110 I=1,NL
         WCL(I)   =WCL(I)+DT*WCLRCH(I)
110   CONTINUE
      RETURN
      END

      SUBROUTINE SUMFLP(SWICH3,I,MS,MFLP)
C calculates matrix flux potential. in L2SS. 8/87
      IMPLICIT REAL (A-Z)
      INTEGER I,IG,IX,SWICH3
      COMMON /HYDCON/DUMMY1(10),KMSA1(10),DUMMY2(10),KST(10)
      DIMENSION MSI(8),XGAUS(3),WGAUS(3)
      SAVE
      DATA MSI/0.,10.,50.,250.,750.,1500.,5000.,10000./
      DATA XGAUS/0.112702,0.5,0.887298/
      DATA WGAUS/0.277778,0.444444,0.277778/,TINY/1.E-10/
      MFLP  =0.
      IF(MS.GT.TINY) THEN
         IF(SWICH3.EQ.1) THEN
            MFLP  =(KST(I)/KMSA1(I))*(EXP(-KMSA1(I)*MS)-1.)
         ELSE
            DO 30 IX=2,7
               DMFLP =0.
               IF(MS.GT.MSI(IX-1)) THEN
                  MSX   =AMIN1(MSI(IX),MS)
                  IF(IX.EQ.2) THEN
                     DO 10 IG=1,3
                        X     =MSX*XGAUS(IG)
                        CALL SUMSKM(SWICH3,I,X,KMSX)
                        DMFLP =DMFLP+KMSX*WGAUS(IG)
10                   CONTINUE
                     MFLP  =MFLP-DMFLP*(MSX-MSI(IX-1))
                  ELSE
                     DO 20 IG=1,3
                        X     =MSX*(MSI(IX-1)/MSX)**XGAUS(IG)
                        CALL SUMSKM(SWICH3,I,X,KMSX)
                        DMFLP =DMFLP+X*KMSX*WGAUS(IG)
20                   CONTINUE
                     IF(DMFLP.LE.0.0) GOTO 40
                     MFLP  =MFLP-DMFLP*ALOG(MSX/MSI(IX-1))
                  ENDIF
               ENDIF
30          CONTINUE
         ENDIF
      ENDIF
40    CONTINUE
      RETURN
      END

      SUBROUTINE SUMSKM(SWICH3,I,MS,KMS)
C calculates hydraulic conductivity from suction. in L2SS. 8/87
      IMPLICIT REAL (A-Z)
      INTEGER SWICH3,I
      COMMON /HYDCON/ KMSMX(10),KMSA1(10),KMSA2(10),KST(10)
      SAVE
      DATA TINY,MSAD/1.E-10,1.E7/
      CALL SUERRM(4.1,MS,0.,1.E8,6.)
      IF(MS.LT.MSAD-TINY) THEN
        IF((SWICH3.EQ.2).AND.(MS.GT.KMSMX(I))) THEN
          KMS   =KMSA2(I)*(MS**(-1.4))
        ELSE
          KMS   =KST(I)*EXP(-KMSA1(I)*MS)
        ENDIF
      IF(KMS.LT.TINY) KMS   =0.
      ELSE
        KMS   =0.
      ENDIF
      RETURN
      END

      SUBROUTINE SUPHOL(IN,PLMX,PLEA,ALVL,ALVDL,F1,F2,RDTM,
     $           DATE,LAT,PCGC,PCGCL)
C computes canopy photosynthesis for 1-5 layers. for L1D,L1Q. 8/87
      IMPLICIT REAL (A-Z)
      INTEGER I1,I2,I3,I4,IN
      DIMENSION PLMX(5),PLEA(5),ALVL(5),ALVDL(5),PCGCL(5)
      DIMENSION F1(5),F2(5),F3(5),PHL(5,3)
      SAVE
      DATA PI/3.1415926/,SCV/0.200/,GAUSR/0.3872983/
      CALL SUASTC(DATE,LAT,RDTM,RDTC,FRDIF,COSLD,SINLD,DSINBE,SOLC,DLA)
      KBLTOP=0.97*F1(1)+0.85*F2(1)+0.65*(1.00-F1(1)-F2(1))
```

```
      KDFTOP=KBLTOP*SQRT(1.0-SCV)
      DO 20 I2=1,5
         PCGCL(I2)=0.0
         DO 10 I1=1,3
            PHL(I2,I1)=0.0
10       CONTINUE
         CALL SUERRM(5.1,ALVL(I2), 0.,25.0,6.)
         CALL SUERRM(5.2,ALVDL(I2),0.,25.0,6.)
         CALL SUERRM(5.3,PLMX(I2), 0.,100.,6.)
         CALL SUERRM(5.4,PLEA(I2), 0., 0.7,6.)
20    CONTINUE
      DO 60 I1=-1,1
         HOUR  =12.0+DLA*0.5*(0.5+I1*GAUSR)
         SINB  =AMAX1(0.,SINLD+COSLD*COS(2.*PI*(HOUR+12.)/24.))
         O15   =AMAX1(0.16,0.966*SINB)
         O45   =AMAX1(0.46,0.707*SINB)
         O75   =1.0-0.268*O15-0.732*O45
         OTOP  =F1(1)*O15+F2(1)*O45+(1.00-F1(1)-F2(1))*O75
         REFH  =(1.-SQRT(1.0-SCV))/(1.+SQRT(1.-SCV))
         REFV  =REFH*2.0*OTOP/(OTOP+KDFTOP*SINB)
         PAR   =0.5*RDTM*SINB*(1.0+0.4*SINB)/DSINBE
         PARDIF=AMIN1(PAR,SINB*FRDIF*(RDTM/RDTC)*0.5*SOLC)
         PARDIR=AMAX1(0.,PAR-PARDIF)
         RTDIF =(1.0-REFV)*PARDIF
         RTDIRT=(1.0-REFV)*PARDIR
         RTDIRD=(1.0-SCV )*PARDIR
         SUNPER=RTDIRD/SINB
         FSSL  =1.0
         DO 50 I2=1,IN
            IF ((ALVL(I2)*PLEA(I2)*PLMX(I2)).LE.0.) GOTO 50
            F3(I2)=1.00-F2(I2)-F1(I2)
            O     =F1(I2)*O15+F2(I2)*O45+F3(I2)*O75
            T2DS  =F1(I2)*0.034+F2(I2)*0.25+F3(I2)*0.47+
     $             SINB*SINB*(F1(I2)*0.90+F2(I2)*0.25-F3(I2)*0.42)
            RANGET=SQRT(12.0*AMAX1(0.,T2DS-O*O))
            KBL   =0.97*F1(I2)+0.85*F2(I2)+0.65*F3(I2)
            KDIF  =KBL*SQRT(1.0-SCV)
            KDIRBL=O/SINB
            KDIRT =KDIRBL*SQRT(1.-SCV)
            FGROS  =0.
            DO 40 I3=-1,1
               ALVC  =(0.5+GAUSR*I3)*(ALVL(I2)+ALVDL(I2))
               VISDF =RTDIF*KDIF*EXP(-KDIF*ALVC)
               VIST  =RTDIRT*KDIRT*EXP(-KDIRT*ALVC)
               VISD  =RTDIRD*KDIRBL*EXP(-KDIRBL*ALVC)
               VISSHD=VISDF+VIST-VISD
               FGRSH =PLMX(I2)*(1.-EXP(-VISSHD*PLEA(I2)/PLMX(I2)))
               FGRSUN=0.0
               DO 30 I4=-1,1
                  SN    =O+RANGET*I4*GAUSR
                  VISSUN=VISSHD+SN*SUNPER
                  FGRS  =PLMX(I2)*(1.0-EXP(-VISSUN*PLEA(I2)/PLMX(I2)))
                  IF (I4.EQ.0) FGRS  =FGRS*1.6
                  FGRSUN=FGRSUN+FGRS
30             CONTINUE
               FGRSUN=FGRSUN/3.6
               FSSLA =FSSL*EXP(-KDIRBL*ALVC)
               FGL   =FSSLA*FGRSUN+(1.-FSSLA)*FGRSH
               IF(I3.EQ.0) FGL   =FGL*1.6
               FGROS =FGROS+FGL
40          CONTINUE
            FGROS =FGROS*ALVL(I2)/3.6
            IF(I1.EQ.0) FGROS =FGROS*1.6
            PHL(I2,I1+2)=FGROS*DLA/3.6
            RTDIF =RTDIF *EXP(-KDIF  *(ALVL(I2)+ALVDL(I2)))
            RTDIRT=RTDIRT*EXP(-KDIRT *(ALVL(I2)+ALVDL(I2)))
            RTDIRD=RTDIRD*EXP(-KDIRBL*(ALVL(I2)+ALVDL(I2)))
            FSSL  =FSSL  *EXP(-KDIRBL*(ALVL(I2)+ALVDL(I2)))
50       CONTINUE
60    CONTINUE
      DO 70 I2=1,IN
         PCGCL(I2)=PHL(I2,1)+PHL(I2,2)+PHL(I2,3)
70    CONTINUE
      PCGC  =PCGCL(1)+PCGCL(2)+PCGCL(3)+PCGCL(4)+PCGCL(5)
      RETURN
      END
```

```
      SUBROUTINE SUSAWA(SWICH1,WCLQT,WL0QT,NL,TRWL,EVSC,
     $            RAIN,ZW,TKL,TYL,DELT,DTMIN,DTMX1,DTFX,
     $            WL0MX,ZEQT,CSA,CSB,CSC2,WCLCH,WL0CH,
     $            WCLEQI,EVSW,RUNOF,DRSL,WCUMCH,ZECH,ZLT)
C calculates the soil water balance for 24 h. in L2SS. 04/88
      IMPLICIT REAL (A-Z)
      INTEGER SWICH1,SWICH2,SWICH3,SWICH4,SWICH5
      INTEGER NL,I,INXSAT,JJTOT,JTOT,J
      COMMON /SLDPTH/ ZL(10)
      COMMON /VOLWAT/ WCAD(10),WCFC(10),WCST(10),WCWP(10)
      COMMON /HYDCON/ KMSMX(10),KMSA1(10),KMSA2(10),KST(10)
      COMMON /PFCURV/ MSWCA(10)
      DIMENSION INXSAT(10,10),JJTOT(10),WCL(10),TKL(10),MS(10)
      DIMENSION FLXSTT(11),FLXUNT(11),FLX(11),FLXINT(11)
      DIMENSION HGT(10),HGB(10),HPP(11),DHH(10),TRWL(10),TYL(10)
      DIMENSION WCLQT(10),WCLCH(10),WCLEQI(10)
      SAVE
      DATA TINY1,TINY2,SWICH3,SWICH4,SWICH5/1.E-3,1.E-4,1,1,2/
      DO 10 I=1,NL
         WCLCH(I) =0.
         WCL(I)   =WCLQT(I)
         IF(SWICH1.EQ.1) ZL(I)    =0.
10    CONTINUE
      WL0   =WL0QT
      ZE    =ZEQT*100.
      EVSW  =0.
      DRSL  =0.
      RUNOF =0.
      CALL SUCONV(1,TKL,ZL,ZLT,RAIN,ZW,WL0,WL0MX,RUNOF,TRWL,EVSW,
     $            EVSC,DRSL,NL)
      IF(SWICH1.EQ.1) THEN
         CALL SUGRHD(TKL,NL,HGT,HGB)
         CALL SUSLIN(TYL,NL,TKL,ZW,ZLT,WCL)
         DO 20 I=1,NL
            WCLEQI(I)=WCL(I)
20       CONTINUE
      ELSE
         TIMTOT=0.
         DO 30 I=1,NL
            WCL(I)    =WCL(I)-DELT*TRWL(I)/TKL(I)
            FLXINT(I)=0.
30       CONTINUE
         FLXINT(NL+1)=0.
         DTMX2 =1.
         WL0   =WL0+RAIN*DELT
         HPBP  =ZLT-ZW
         IF(WL0.GE.EVSC*DELT) THEN
            EVSW  =EVSC
            EVSWX =0.
            WL0   =WL0-EVSC*DELT
         ELSEIF(WL0.GT.0.) THEN
            EVSW1 =WL0/DELT
            EVSW2 =AMIN1(EVSC-EVSW1,(WCL(1)-WCAD(1))*TKL(1)/DELT)
            EVSW  =EVSW1+EVSW2
            EVSWX =EVSW2
            WCL(1)=WCL(1)-EVSW2*DELT/TKL(1)
            WL0   =0.
         ELSE
            EVSW1 =EVSC
            EVSW2 =(WCL(1)-WCAD(1))*TKL(1)/DELT
            EVSW3 =CSC2/(ZE+TINY1)
            EVSW  =AMIN1(EVSW1,EVSW2,EVSW3)
            EVSWX =EVSW
            WCL(1)=WCL(1)-EVSW*DELT/TKL(1)
         ENDIF
40       JTOT  =0
         DO 50 J=1,NL
            JJTOT(J) =0
50       CONTINUE
         CALL SUSTCH(WCL,NL,INXSAT,JTOT,JJTOT)
         IF(JTOT.NE.0) THEN
           CALL SUSTHH(INXSAT,JTOT,JJTOT,NL,HGT,HGB,WL0,WL0MX,HPBP,
     $                 HPP,DHH)
           CALL SUSTFL(NL,INXSAT,JTOT,JJTOT,TKL,DHH,FLXSTT,FLXSQ1,FLXSQ2)
         ELSE
            CONTINUE
         ENDIF
```

```
      IF(JTOT.EQ.1.AND.JJTOT(1).EQ.NL) THEN
        FLXUNT(1)=0.
      ELSE
        CALL SUUNST(SWICH3,SWICH4,WCL,TKL,NL,WL0,WL0MX,HPBP,
     $                MS,FLXUNT)
      ENDIF
      CALL SUSEFL(INXSAT,NL,JTOT,JJTOT,FLXSTT,FLXUNT,WL0,WL0MX,
     $            FLXSQ1,FLXSQ2,FLX)
      CALL SUINTG(HPBP,SWICH4,SWICH5,HPP,FLX,TKL,WCL,MS,DTMIN,DTMX1,
     $        DTMX2,DTFX,INXSAT,DHH,JTOT,FLXSQ2,WL0,WL0MX,NL,DELT,DT)
      DO 60 I=1,NL+1
        FLXINT(I)=FLXINT(I)+DT*FLX(I)
60    CONTINUE
      TIMTOT=TIMTOT+DT
      DTMX2 =DELT-TIMTOT
      IF(DTMX2.GT.TINY2) GOTO 40
      IF(WL0.GT.WL0MX) THEN
        RUNOF =WL0-WL0MX
        WL0   =WL0MX
      ENDIF
      DRSL  =FLXINT(NL+1)/DELT
      CALL SUZECA(WCLQT(1),EVSC,RAIN,-EVSWX,FLXINT(2),
     $            WL0,DELT,ZE,CSA,CSB,CSC2)
    ENDIF
    CALL SUCONV(2,TKL,ZL,ZLT,RAIN,ZW,WL0,WL0MX,RUNOF,TRWL,EVSW,
   $          EVSC,DRSL,NL)
    ZECH  =ZE/100.-ZEQT
    WL0CH =WL0-WL0QT
    WCUMCH=0.
    DO 70 I=1,NL
      WCLCH(I)=WCL(I)-WCLQT(I)
      WCUMCH  =WCUMCH+WCLCH(I)*TKL(I)
70  CONTINUE
    RETURN
    END
```

File: A:\SYSSARP\SUBFUN\MACROS\SSUB2.FOR

```
*     A minor modification was made in SUSLIN.
*     see Chapter 7.

      SUBROUTINE SUSEFL(INXSAT,NL,JTOT,JJTOT,FLXSTT,FLXUNT,
     $           WL0,WL0MX,FLXSQ1,FLXSQ2,FLX)
C selects between saturated and unsaturated fluxes; in L2SS; 04/88
      IMPLICIT REAL (A-Z)
      INTEGER INXSAT,JJTOT,JTOT,J,JJ,I,IIN,IOUT,NL
      DIMENSION INXSAT(10,10),JJTOT(10),FLXSTT(11),FLXUNT(11),FLX(11)
      SAVE
      DATA TINY/0.001/
      HPTP=AMIN1(WL0,WL0MX)
      DO 10 I=1,NL+1
        FLX(I)   =FLXUNT(I)
10    CONTINUE
      DO 70 J=1,JTOT
        IF(FLXSTT(J).GT.TINY) THEN
          IOUT  =INXSAT(J,JJTOT(J))+1
          IIN   =INXSAT(J,1)
          FLX(IOUT)=AMIN1(FLXSTT(J),FLXUNT(IOUT))
          IF(IOUT.EQ.NL+1) FLX(IOUT)=FLXSTT(J)
          IF (JJTOT(J).GT.1) THEN
            DO 20 JJ=2,JJTOT(J)
              FLX(INXSAT(J,JJ))=FLX(IOUT)
20          CONTINUE
          ENDIF
          FLX(IIN)=AMIN1(FLX(IOUT),FLXUNT(IIN))
          IF(IIN.EQ.1.AND.HPTP.GT.TINY) FLX(IIN)=FLX(IOUT)
        ELSEIF(FLXSTT(J).LT.-TINY) THEN
          IOUT  =INXSAT(J,1)
          IIN   =INXSAT(J,JJTOT(J))+1
          IF(FLXSQ2.LE.0.) THEN
            FLX(IOUT)=FLXSTT(J)
          ELSE
            IF(FLXUNT(IOUT).LT.-TINY) THEN
              FLX(IOUT)=FLXUNT(IOUT)
            ELSEIF(FLXUNT(IOUT).GT.TINY) THEN
              FLX(IOUT)=AMAX1(FLXSQ2,FLXUNT(IOUT))
            ELSE
              FLX(IOUT)=0.
            ENDIF
          ENDIF
          IF(IOUT.EQ.1) FLX(IOUT)=FLXSTT(J)
          IF(JJTOT(J).GT.1) THEN
            DO 30 JJ=2,JJTOT(J)
              FLX(INXSAT(J,JJ))=AMAX1(FLX(IOUT),FLXSTT(J))
30          CONTINUE
          ENDIF
          FLX(IIN)=AMAX1(FLX(IOUT),FLXSTT(J))
        ELSE
          IIN   =INXSAT(J,1)
          IOUT  =NL+1
          IF(FLXUNT(IIN).LE.-TINY) THEN
            FLX(IIN)=FLXUNT(IIN)
            IF(JJTOT(J).GT.1) THEN
              DO 40 JJ=2,JJTOT(J)
                FLX(INXSAT(J,JJ))=AMAX1(FLXSQ1,FLX(IIN))
40            CONTINUE
            ENDIF
            FLX(IOUT)=AMAX1(FLXSQ1,FLX(IIN))
          ELSEIF(FLXUNT(IIN).GE.TINY) THEN
            FLX(IIN)=AMIN1(FLXUNT(IIN),FLXSQ2)
            IF(JJTOT(J).GT.1) THEN
              DO 50 JJ=2,JJTOT(J)
                FLX(INXSAT(J,JJ))=FLX(IIN)
50            CONTINUE
            ENDIF
            FLX(IOUT)=FLX(IIN)
          ELSE
            FLX(IIN)=0.
            IF(JJTOT(J).GT.1) THEN
              DO 60 JJ=2,JJTOT(J)
                FLX(INXSAT(J,JJ))=0.
60            CONTINUE
            ENDIF
            FLX(IOUT)=0.
          ENDIF
        ENDIF
70    CONTINUE
      DO 80 I=1,NL+1
        FLXUNT(I)=0.
        FLXSTT(I)=0.
80    CONTINUE
      RETURN
      END

      SUBROUTINE SUSLIN(TYL,NL,TKL,ZW,ZLT,WCL)
C routine has been modified to make it compatible with
C FORTRAN version of MACROS.
C derives soil parameters. in L2SS. 03/88
      IMPLICIT REAL (A-Z)
      INTEGER NL,I,IX, IUNIT, IUNITL
      COMMON /VOLWAT/ WCAD(10),WCFC(10),WCST(10),WCWP(10)
      COMMON /SLDPTH/ ZL(10)
      COMMON /LOGCOM/ IUNITL, DUMMY, IRUN
      DIMENSION TYL(10),TKL(10),WCL(10)
      SAVE
      DATA TINY/1.E-10/
      IF (IUNITL.EQ.0) THEN
        IUNIT = 6
      ELSE
        IUNIT = IUNITL
      END IF

      WRITE(IUNIT,20)
      DO 10 I=1,NL
        CALL SUERRM(6.1,TYL(I),1.,20.,6.)
        WRITE(IUNIT,30)I,TYL(I),TKL(I)*.01,
     $        WCAD(I),WCWP(I),WCFC(I),WCST(I)
        IF(I.EQ.1) THEN
          ZL(I)=0.
        ELSE
          ZL(I)=ZL(I-1)+TKL(I-1)
        ENDIF
        MS    =AMAX1(0.,ZW-ZL(I)-0.5*TKL(I))
        CALL SUWCMS(I,2,WCL(I),MS)
```

```
          CALL SUERRM(6.2,WCL(I),WCAD(I),WCST(I),6.)
 10     CONTINUE
        ZLT    =ZL(NL)+TKL(NL)
 20     FORMAT(' SOIL CHARACTERISTICS PER COMPARTMENT: ',/,
       $' COMPARTMENT  TYPE NR    TKL     WCAD     WCWP     WCFC     WCST')
 30     FORMAT(3X,I4,8X,F5.1,3X,F5.3,3X,4(F5.4,3X))
        RETURN
        END

        SUBROUTINE SUSTCH(WCL,NL,INXSAT,JTOT,JJTOT)
C checks for presence of saturated layers. in L2SS. 9/87.
        IMPLICIT REAL (A-Z)
        INTEGER NL,INXSAT,I,J,JTOT,JJTOT,JJ
        COMMON /VOLWAT/ DUMMY1(20),WCST(10),DUMMY(10)
        DIMENSION WCL(10),INXSAT(10,10),JJTOT(10)
        SAVE
        DATA TINY/0.001/
        J      =0
        JJ     =0
        DO 10 I=1,NL
           IF(ABS(WCL(I)-WCST(I)).LT.TINY) THEN
              JJ     =JJ+1
              IF(JJ.NE.1) THEN
                 J      =J+0
              ELSE
                 J      =J+1
                 JTOT   =J
              ENDIF
              INXSAT(J,JJ)=I
              JJTOT(J)=JJ
           ELSE
              JJ     =0
           ENDIF
 10     CONTINUE
        RETURN
        END

        SUBROUTINE SUSTFL(NL,INXSAT,JTOT,JJTOT,TKL,DHH,
       $            FLXSTT,FLXSQ1,FLXSQ2)
C calculates tentative saturated fluxes. in L2SS. 04/88
        IMPLICIT REAL (A-Z)
        INTEGER I,J,N,NL,JTOT,JJTOT,INXSAT,INDX,IA,JA
        COMMON /HYDCON/ DUMMY(30),KST(10)
        DIMENSION A(11,11),TKL(10),INXSAT(10,10),JJTOT(10),B(11)
        DIMENSION DHH(10),FLXSTT(11),DIS(10),INDX(11),KS(10)
        SAVE
        DATA TINY/1.0E-10/, LARGE/-100./,DUMMY2/0./,DUMMY1/0./
        DO 190 J=1,JTOT
           DO 20 IA=1,NL+1
              DO 10 JA=1,NL+1
                 A(IA,JA)=0.
                 B(JA)=0.
 10           CONTINUE
 20        CONTINUE
           IF(ABS(DHH(J)).LT.TINY) THEN
              FLXSTT(J)=0.
           ELSE
              DO 30 IA=1,JJTOT(J)
                 I=INXSAT(J,1)+IA-1
                 KS(IA)=KST(I)
                 DIS(IA)=TKL(I)
 30           CONTINUE
              DO 40 IA=1,JJTOT(J)
                 A(IA,JJTOT(J)+1)=-1.
                 JA=IA
                 A(IA,JA)=-KS(IA)/DIS(IA)
 40           CONTINUE
              DO 50 JA=1,JJTOT(J)
                 A(JJTOT(J)+1,JA)=+1.
 50           CONTINUE
              A(JJTOT(J)+1,JJTOT(J)+1)=0.
              DO 60 JA=1,JJTOT(J)
                 B(JA)=0.
 60           CONTINUE
              B(JJTOT(J)+1)=DHH(J)
              N=JJTOT(J)+1
              CALL SUSTMD(A,N,INDX,D)
              CALL SUSTMS(A,N,INDX,B)
              FLXSTT(J)=B(N)
           ENDIF
           IF(DHH(J).GT.-TINY) THEN
              DO 80 IA=1,NL+1
                 DO 70 JA=1,NL+1
                    A(IA,JA)=0.
                    B(JA)=0.
 70              CONTINUE
 80           CONTINUE
              IF(JJTOT(J).EQ.1) THEN
                 FLXSQ1=LARGE
              ELSE
                 DO 90 IA=1,JJTOT(J)-1
                    I=INXSAT(J,2)+IA-1
                    KS(IA)=KST(I)
                    DIS(IA)=TKL(I)
 90              CONTINUE
                 DO 100 IA=1,JJTOT(J)-1
                    A(IA,JJTOT(J))=-1.
                    JA=IA
                    A(IA,JA)=-KS(IA)/DIS(IA)
 100             CONTINUE
                 DO 110 JA=1,JJTOT(J)-1
                    A(JJTOT(J),JA)=+1.
 110             CONTINUE
                 A(JJTOT(J),JJTOT(J))=0.
                 DO 120 JA=1,JJTOT(J)-1
                    B(JA)=0.
 120             CONTINUE
                 B(JJTOT(J))=DHH(J)+TKL(INXSAT(J,1))
                 N=JJTOT(J)
                 CALL SUSTMD(A,N,INDX,D)
                 CALL SUSTMS(A,N,INDX,B)
                 FLXSQ1=B(N)
              ENDIF
              DO 140 IA=1,NL+1
                 DO 130 JA=1,NL+1
                    A(IA,JA)=0.
                    B(JA)=0.
 130             CONTINUE
 140          CONTINUE
              IF(INXSAT(J,1).EQ.1) THEN
                 FLXSQ2=DUMMY2
              ELSE
                 DO 150 IA=1,JJTOT(J)+1
                    I=(INXSAT(J,1)-1)+IA-1
                    KS(IA)=KST(I)
                    DIS(IA)=TKL(I)
 150             CONTINUE
                 DO 160 IA=1,JJTOT(J)+1
                    A(IA,JJTOT(J)+2)=-1.
                    JA=IA
                    A(IA,JA)=-KS(IA)/DIS(IA)
 160             CONTINUE
                 DO 170 JA=1,JJTOT(J)+1
                    A(JJTOT(J)+2,JA)=+1.
 170             CONTINUE
                 A(JJTOT(J)+2,JJTOT(J)+2)=0.
                 DO 180 JA=1,JJTOT(J)+1
                    B(JA)=0.
 180             CONTINUE
                 B(JJTOT(J)+2)=DHH(J)-TKL(INXSAT(J,1)-1)
                 N=JJTOT(J)+2
                 CALL SUSTMD(A,N,INDX,D)
                 CALL SUSTMS(A,N,INDX,B)
                 FLXSQ2=B(N)
              ENDIF
           ENDIF
 190    CONTINUE
        RETURN
        END

        SUBROUTINE SUSTHH(INXSAT,JTOT,JJTOT,NL,HGT,HGB,
       $            WL0,WL0MX,HPBP,HPP,DHH)
C identifies hydraulic head across saturated layers. in L2SS. 04/88
        IMPLICIT REAL (A-Z)
```

```
      INTEGER I,NL,INXSAT,J,JTOT,JJTOT
      DIMENSION INXSAT(10,10),JJTOT(10),HHT(10),HHB(10),HGT(10),HGB(10)
      DIMENSION HPP(11),DHH(10)
      SAVE
      DATA TINY/1.E-5/
      HPTP  =AMIN1(WL0,WL0MX)
      DO 10 I=1,11
         HPP(I)=0.
   10 CONTINUE
      DO 20 J=1,JTOT
         EXTRAT=0.
         IF(INXSAT(J,1).EQ.1) EXTRAT=HPTP
         EXTRAB=0.
         IF(INXSAT(J,JJTOT(J)).EQ.NL) EXTRAB=HPBP
         HHT(J)=EXTRAT+HGT(INXSAT(J,1))
         HHB(J)=EXTRAB+HGB(INXSAT(J,JJTOT(J)))
         DHH(J)=HHB(J)-HHT(J)
         IF(DHH(J).GT.TINY) HPP(INXSAT(J,1))=DHH(J)
         IF(ABS(DHH(J)).LT.TINY) DHH(J)=0.
   20 CONTINUE
      RETURN
      END

      SUBROUTINE SUSTMD(A,N,INDX,D)
C decomposes matrix A. in L2SS. (PRESS etal, 1986.)
      IMPLICIT REAL (A-Z)
      INTEGER I,J,K,N,IMAX,INDX
      DIMENSION A(11,11),INDX(11),VV(11)
      SAVE
      DATA TINY/1.0E-10/
      D=1.
      DO 12 I=1,N
         AAMAX=0.
         DO 11 J=1,N
            IF(ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
   11    CONTINUE
         VV(I)=1./AAMAX
   12 CONTINUE
      DO 19 J=1,N
         IF(J.GT.1) THEN
            DO 14 I=1,J-1
               SUM=A(I,J)
               IF(I.GT.1) THEN
                  DO 13 K=1,I-1
                     SUM=SUM-A(I,K)*A(K,J)
   13             CONTINUE
                  A(I,J)=SUM
               ENDIF
   14       CONTINUE
         ENDIF
         AAMAX=0.
         DO 16 I=J,N
            SUM=A(I,J)
            IF(J.GT.1.) THEN
               DO 15 K=1,J-1
                  SUM=SUM-A(I,K)*A(K,J)
   15          CONTINUE
               A(I,J)=SUM
            ENDIF
            DUM=VV(I)*ABS(SUM)
            IF (DUM.GE.AAMAX) THEN
               IMAX=I
               AAMAX=DUM
            ENDIF
   16    CONTINUE
         IF(J.NE.IMAX) THEN
            DO 17 K=1,N
               DUM=A(IMAX,K)
               A(IMAX,K)=A(J,K)
               A(J,K)=DUM
   17       CONTINUE
            D=-D
            VV(IMAX)=VV(J)
         ENDIF
         INDX(J)=IMAX
         IF(J.NE.N) THEN
            IF(ABS(A(J,J)).LT.TINY) A(J,J)=TINY
            DUM=1./A(J,J)
            DO 18 I=J+1,N
               A(I,J)=A(I,J)*DUM
   18       CONTINUE
         ENDIF
   19 CONTINUE
      IF(ABS(A(N,N)).LT.TINY) A(N,N)=TINY
      RETURN
      END

      SUBROUTINE SUSTMS(A,N,INDX,B)
C solves a set of linear equations. in L2SS. (PRESS etal, 1986)
      IMPLICIT REAL (A-Z)
      INTEGER I,J,II,LL,N,INDX
      DIMENSION A(11,11),INDX(11),B(11)
      SAVE
      II=0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF(II.NE.0) THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
   11     CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
   12 CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N) THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
   13     CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
   14 CONTINUE
      RETURN
      END

      SUBROUTINE SUUNST(SWICH3,SWICH4,WCL,TKL,NL,WL0,WL0MX,
     $           HPBP,MS,FLXUNT)
C calculates tentative fluxes of unsaturated layers; in L2SS; 04/88
      IMPLICIT REAL (A-Z)
      INTEGER NL,I,SWICH3,SWICH4
      COMMON /HYDCON/ DUMMY1(30),KST(10)
      COMMON /VOLWAT/ WCAD(10),DUMMY2(10),WCST(10),DUMMY3(10)
      DIMENSION WCL(10),TKL(10),DIS(11),FLXUNT(11),MS(10)
      DIMENSION KMS(10),MFLP(10),MFLPQT(10)
      SAVE
      DATA TINY1,TINY2/0.001,0.001/
      HPTP  =AMIN1(WL0,WL0MX)
      IF(WCST(1)-WCL(1).GT.TINY1) THEN
         CALL SUWCMS(1,SWICH4,WCL(1),MS(1))
         CALL SUMFLP(SWICH3,1,MS(1),MFLP(1))
         DIS(1)=0.5*TKL(1)
         IF(HPTP.GT.TINY1) THEN
            DZDH  =DIS(1)/(-MS(1)-HPTP)
            DMFLP =MFLP(1)-0.-KST(1)*HPTP
            FLXUNT(1)=(DZDH-1.)*DMFLP/DIS(1)
         ELSE
            FLXUNT(1)=0.
         ENDIF
      ELSE
         FLXUNT(1)=0.
      ENDIF
      DO 10 I=2,NL
         IF(ABS(WCL(I)-WCST(I)).LT.TINY2) THEN
            IF(ABS(WCL(I-1)-WCST(I-1)).GT.TINY2) THEN
               DIS(I)=0.5*TKL(I-1)
               DZDH  =DIS(I)/(0.+MS(I-1))
               DMFLP =0.-MFLP(I-1)
               FLXUNT(I)=(DZDH-1.)*DMFLP/DIS(I)
            ELSE
               CONTINUE
```

```
            ENDIF
         ELSE
            CALL SUWCMS(I,SWICH4,WCL(I),MS(I))
            CALL SUMFLP(SWICH3,I,MS(I),MFLP(I))
            IF(ABS(WCL(I-1)-WCST(I-1)).LT.TINY2) THEN
               DIS(I)=0.5*TKL(I)
               DZDH  =DIS(I)/(-MS(I)-0.)
               DMFLP =MFLP(I)-0.
               FLXUNT(I)=(DZDH-1.)*DMFLP/DIS(I)
            ELSE
               CALL SUMFLP(SWICH3,I-1,MS(I),MFLPQT(I))
               CALL SUMFLP(SWICH3,I,MS(I-1),MFLPQT(I-1))
               DIS(I)=0.5*(TKL(I)+TKL(I-1))
               DMFLP1=MFLP(I)-MFLPQT(I-1)
               IF((MS(I)-MS(I-1))*(MFLP(I)-MFLPQT(I-1)).GT.-TINY1) THEN
                  DMFLP1=0.
               ENDIF
               DMFLP2=MFLPQT(I)-MFLP(I-1)
               IF((MS(I)-MS(I-1))*(MFLPQT(I)-MFLP(I-1)).GT.-TINY1) THEN
                  DMFLP2=0.
               ENDIF
               IF(ABS(DMFLP1).LT.TINY2.OR.ABS(DMFLP2).LT.TINY2) THEN
                  DMFLP =0.
               ELSE
                  SIGN  =DMFLP1/ABS(DMFLP1)
                  DMFLP =SIGN*SQRT(DMFLP1*DMFLP2)
               ENDIF
               IF(MS(I).NE.MS(I-1)) THEN
                  DZDH  =DIS(I)/(-MS(I)+MS(I-1))
                  FLXUNT(I)=(DZDH-1.)*DMFLP/DIS(I)
               ELSE
                  CALL SUMSKM(SWICH3,I,MS(I),KMS(I))
                  CALL SUMSKM(SWICH3,I-1,MS(I-1),KMS(I-1))
                  KAV   =SQRT(KMS(I)*KMS(I-1))
                  FLXUNT(I)=+KAV
               ENDIF
            ENDIF
         ENDIF
10    CONTINUE
      IF(ABS(WCL(NL)-WCST(NL)).LT.TINY2) THEN
         CONTINUE
      ELSE
         DIS(NL+1)=0.5*TKL(NL)
         CALL SUMFLP(SWICH3,NL,MS(NL),MFLP(NL))
         IF(HPBP.GT.TINY1) THEN
            DZDH  =DIS(NL+1)/(HPBP+MS(NL))
            DMFLP =0.-MFLP(NL)+KST(NL)*(HPBP-0.)
            FLXUNT(NL+1)=(DZDH-1.)*DMFLP/DIS(NL+1)
         ELSE
            MSB   =-HPBP
            CALL SUMFLP(SWICH3,NL,MSB,MFLPB)
            IF(HPBP.NE.-MS(NL)) THEN
               DZDH  =DIS(NL+1)/(HPBP+MS(NL))
               DMFLP =MFLPB-MFLP(NL)
               IF((MSB-MS(NL))*(MFLPB-MFLP(NL)).GT.-TINY1) DMFLP =0.
               FLXUNT(NL+1)=(DZDH-1.)*DMFLP/DIS(NL+1)
            ELSE
               CALL SUMSKM(SWICH3,NL,MS(NL),KMS(NL))
               FLXUNT(NL+1)=+KMS(NL)
            ENDIF
         ENDIF
      ENDIF
      RETURN
      END

      SUBROUTINE SUWCMS(I,SWICH4,WCL,MS)
C relates volumetric water content and suction; in L2SS, 03/87
      IMPLICIT REAL (A-Z)
      INTEGER I,SWICH4
      COMMON /VOLWAT/ WCAD(10),DUMMY1(10),WCST(10),DUMMY2(10)
      COMMON /PFCURV/ MSWCA(10)
      SAVE
      DATA TINY/0.001/
      IF(SWICH4.EQ.1) THEN
         CALL SUERRM(7.1,WCL,WCAD(I),WCST(I),6.)
         MS    =EXP(SQRT(-ALOG(AMAX1(WCAD(I),WCL)/WCST(I))/MSWCA(I)))-1.
      ELSE
         CALL SUERRM(7.2,MS,0.,1.E8,6.)
         WCL =AMAX1(TINY,WCST(I)*EXP(-MSWCA(I)*((ALOG(MS+1.))**2)))
      ENDIF
      RETURN
      END

      SUBROUTINE SUZECA(WCLQT1,EVSC,RAIN,FLX1,FLX2,WL0,
     $           DELT,ZE,AEXP,BEXP,C2)
C calculates the depth of the evaporation front; in L2SS; 04/88
      IMPLICIT REAL (A-Z)
      COMMON /VOLWAT/ WCAD(10),DUMMY1(10),WCST(10),DUMMY2(10)
      SAVE
      DATA TINY1,LARGE/1.E-3,10./
      IF(RAIN.GT.EVSC) THEN
         ZE    =0.
      ELSEIF(FLX2.LT.-TINY1.AND.FLX2.LT.FLX1) THEN
         ZE    =ZE-(FLX1-FLX2)/(0.5*WCST(1))
         IF(ZE.LE.TINY1) THEN
            ZE    =0.
         ENDIF
      ELSEIF(RAIN.GT.0.05) THEN
         CONTINUE
      ELSE
         WI    =WCLQT1/WCST(1)
         WTH   =WI-WCAD(1)/WCST(1)
         IF(WTH.LE.TINY1) THEN
            ZE    =LARGE
         ELSEIF(WL0.GT.TINY1.OR.ABS(WCLQT1-WCST(1)).LT.TINY1) THEN
            ZE    =0.
         ELSE
            C1    =AEXP*(EXP(AMAX1(0.,(WI-0.5)*BEXP))-1.)
            C3    =C2/(WTH+C1/C2)
            ZE    =SQRT(ZE*ZE+2.*C3*DELT)
         ENDIF
      ENDIF
      RETURN
      END
```

Appendix C: TTUTIL

In this appendix listings are given in alphabetic order of the subroutines and functions that are used from the utility library TTUTIL of the Dept. of Theoretical Production Ecology. The full library is supplied on Disk 2, directory A:\SYSSARP\SUBFUN\TTUTIL\. However, only routines that are used in the MACROS program are listed here. More information is given by Rappoldt & van Kraalingen (1990). The listing is in condensed format to save space.

An object library of the full library has been created however, compatible with Microsoft FORTRAN 4.10 or higher. The path to this library is A:\SYSSARP\FORTRAN\TTUTIL.LIB.

```
      REAL FUNCTION AFGEN (TABLE,ILTAB,X)

*     This function is a linear interpolation function. The
*     function also extrapolates outside the defined region in
*     case X is below or above the region defined by TABLE.
*     Extrapolation, however, results in a warning to the screen.
*     This function has been modified to write warnings to a log file.
*     It is a copy of the function LINT of the TTUTIL library.

*     AFGEN - function name, result of the interpolation     O
*     TABLE - a one-dimensional array with paired            I
*             data: x1,y1,x2,y2, etc.
*     ILTAB - the number of elements of the array            I
*             TABLE
*     X     - the value at which interpolation should        I
*             take place
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR

*     Author: Daniel van Kraalingen
*     Date  : January 1990


*     formal parameters
      INTEGER ILTAB
      REAL TABLE(ILTAB), X

*     local variables
      INTEGER I1, ILT, IUP
      REAL SLOPE, TINY
      PARAMETER (TINY=1.E-7)
      LOGICAL ERR, WARN

*     special declarations to make log files possible
      INTEGER IUNITL, IRUN
      REAL TIME
      COMMON /LOGCOM/ IUNITL, TIME, IRUN

      SAVE

*     initialize
      ERR  = .FALSE.
      WARN = .FALSE.

*     check on value of ILTAB
      IF (MOD(ILTAB,2).NE.0 .OR. ILTAB.LE.2) THEN
         WRITE (*,'(A)')
     $    ' Number of elements in AFGEN table not correct !'
         ERR = .TRUE.

      ELSE
         IUP = 0
         DO 10 I1=3,ILTAB,2
*           check on ascending order of X-values in function
            IF (TABLE(I1).LE.TABLE(I1-2)) THEN
               WRITE (*,'(2A,I4)')
     $          ' X-coordinates in AFGEN table not in',
     $          ' ascending order at point',I1
               ERR = .TRUE.
            END IF
            IF (IUP.EQ.0 .AND. TABLE(I1).GE.X) IUP = I1
10       CONTINUE
      END IF

      IF (.NOT.ERR .AND. X.LT.TABLE(1)) THEN
         IUP = 3
         IF ((TABLE(1)-X) .GT. ABS(X)*TINY) THEN
            WARN = .TRUE.
            WRITE (*,'(A,G13.5)')
     $       ' WARNING in AFGEN: X-value below defined region at X=',X

*           modification for log files
            IF (IUNITL.NE.0)  THEN
               WRITE (IUNITL,'(/,/,A,I3,A,G13.5)')
     $          ' Run =',IRUN,', TIME =',TIME
               WRITE (IUNITL,'(A,G13.5)')
     $          ' WARNING in AFGEN: X-value below defined region at X=',X
            END IF
         END IF
      ELSE IF (.NOT.ERR .AND. X.GT.TABLE(ILTAB-1)) THEN
         IUP = ILTAB-1
         IF ((X-TABLE(ILTAB-1)) .GT. ABS(X)*TINY) THEN
            WARN = .TRUE.
            WRITE (*,'(A,G13.5)')
     $       ' WARNING in AFGEN: X-value above defined region at X=',X

*           modification for log files
            IF (IUNITL.NE.0) THEN
               WRITE (IUNITL,'(/,/,A,I3,A,G13.5)')
     $          ' Run =',IRUN,', TIME =',TIME
               WRITE (IUNITL,'(A,G13.5)')
     $          ' WARNING in AFGEN: X-value above defined region at X=',X
            END IF
         END IF
      END IF

      IF (WARN.OR.ERR) THEN
         ILT = MIN (ILTAB/2, 15)
         WRITE (*,'(A,I4,/,A,I2,A)')
     $     ' Number of table elements is',ILTAB,
     $     ' First ',ILT,' pairs are:'

*        modification for log files
         IF (WARN.AND.IUNITL.NE.0) WRITE (IUNITL,'(A,I4,/,A,I2,A)')
     $     ' Number of table elements is',ILTAB,
     $     ' First ',ILT,' pairs are:'

         IF (ILT.GT.0) WRITE (*,'(2G13.5)')
     $      (TABLE(I1),TABLE(I1+1),I1=1,2*ILT,2)

*        modification for log files
         IF (ILT.GT.0.AND.WARN.AND.IUNITL.NE.0)
     $      WRITE (IUNITL,'(2G13.5)')
     $      (TABLE(I1),TABLE(I1+1),I1=1,2*ILT,2)

         IF (ERR) CALL ERROR ('AFGEN','execution terminated')
      END IF

*     interpolation and extrapolation
      SLOPE = (TABLE(IUP+1)-TABLE(IUP-1))/(TABLE(IUP)-TABLE(IUP-2))
      AFGEN  = TABLE(IUP-1)+(X-TABLE(IUP-2))*SLOPE

      RETURN
      END
      SUBROUTINE CHKTSK (MODULE, IULOG, ITOLD, ITASK)

*     The function of this routine is to check the new task
*     and previous task of simulating subroutines.

*     MODULE - Name of module from which CHKTSK is called  I
*     IULOG  - Log file where to write errors to           I
*     ITOLD  - Value of previous task                      I
*     ITASK  - Value of new task                           I
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR

*     Author: Daniel van Kraalingen
*     Date  : December 1989


*     formal parameters
      INTEGER IULOG, ITOLD, ITASK
      CHARACTER*(*) MODULE

**    local parameters
      INTEGER ITABLE(4,4), ISTAT, IMES
      CHARACTER MESSAG(9)*58
      SAVE


*     The messages that can be generated and the severity (warning
*     or error) are coded in a table.
```

```
*     Positive values: WARNING
*     Negative values: ERROR

      DATA ITABLE /1,  0,  0,  7,
     &             2,  3,  0,  0,
     &             2,  0, -5,  8,
     &             0, -4, -6, -9/

      DATA MESSAG/
     &   'Repeated initialization !                                  ',
     &   'No Terminal call done before initialization !             ',
     &   'Repeated rate calculation !                                ',
     &   'Rate calculation impossible after terminal call !          ',
     &   'Repeated integration not allowed !                         ',
     &   'Integration impossible after terminal call !               ',
     &   'Terminal call without simulation !                         ',
     &   'Terminal call after integration, some output may be lost !',
     &   'Repeated terminal call not allowed !                       '/

      IF (ITASK.LT.1.OR.ITASK.GT.4) CALL ERROR (MODULE,'wrong ITASK')
      IF (ITOLD.LT.1.OR.ITOLD.GT.4) CALL ERROR (MODULE,'wrong ITOLD')

      ISTAT = ITABLE(ITASK, ITOLD)
      IMES  = ABS (ISTAT)

      IF (ISTAT.LT.0) THEN
         CALL ERROR (MODULE, MESSAG(IMES))
      ELSE IF (ISTAT.GT.0) THEN
         WRITE (*,'(4A)')
     &     ' WARNING from ',MODULE,': ',MESSAG(IMES)
         IF (IULOG.NE.0) WRITE (IULOG,'(4A)')
     &     ' WARNING from ',MODULE,': ',MESSAG(IMES)
      END IF

      RETURN
      END
      LOGICAL FUNCTION DECCHK (STRING)

*     Checks if string STRING is a number.

*     DECCHK --> .TRUE. if the string is a number                 O
*     STRING - input string, NO trailing or leading blanks !      I
*
*     No subroutines and/or functions called

*     Author: Kees Rappoldt
*     Date  : September 1989


*     Parser status jumptable
*     -----------------------
*     Example string:    -2345.4567E-23
*            status:   12|...3|             (| = go down)
*                        4...55...6789
*
*            STATUS: 0/-  1   2   3   4   5   6   7   8   9
*     CHARACTER:
*     0 - other       -   -   -   -   -   -   -   -   -   -
*     1 - sign        -   2   -   -   -   -   7   -   -   -
*     2 - point       -   3   3   -   5   -   -   -   -   -
*     3 - numeric     -   4   4   5   4   5   8   8   9   -
*     4 - E           -   -   -   -   6   6   -   -   -   -
*     5 - last char   -   -   -   -   4   5   -   -   8   9


*     formal parameter
      CHARACTER*(*) STRING

**    local variables
      INTEGER I,IN,IS,ILEN
      INTEGER ITYP(0:14), JUMPTB(0:9,0:5)
      CHARACTER NUM1*14

      SAVE
      DATA   NUM1 /'+-1234567890.E'/
      DATA   ITYP /0,1,1,3,3,3,3,3,3,3,3,3,3,2,4/
      DATA JUMPTB /0,0,0,0,0,0,0,0,0,0, 0,2,0,0,0,0,7,0,0,0,
     $             0,3,3,0,5,0,0,0,0,0, 0,4,4,5,4,5,8,8,9,0,
     $             0,0,0,0,6,6,0,0,0,0, 0,0,0,0,4,5,0,0,8,9/


*     initialize loop over characters ; set status 1
      ILEN = LEN (STRING)
      I    = 0
      IS   = 1

10    IF (IS.GT.0 .AND. I.LT.ILEN) THEN
         I = I + 1
         IN = INDEX (NUM1,STRING(I:I))
         IS = JUMPTB (IS,ITYP(IN))
      GOTO 10
      END IF

*     test final status
      IS = JUMPTB (IS,5)

      DECCHK = IS.GT.0
      RETURN
      END
      SUBROUTINE DECINT (IWAR,STRING,IVALUE)

*     Decodes an integer number from a character string

*     IWAR   - In case of error IWAR = 1, otherwise IWAR = 0      O
*     STRING - input string, NO trailing or leading blanks !      I
*     IVALUE - Integer value read from string                     O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK

*     Author: Kees Rappoldt
*     Date  : September 1989


*     formal parameters
      INTEGER IWAR,IVALUE
      CHARACTER*(*) STRING

**    local variables + functions used
      INTEGER ILR
      CHARACTER HELP*12,FORM*5
      LOGICAL NOP,DECCHK
      SAVE

      ILR = LEN (STRING)
      NOP = INDEX(STRING,'.').EQ.0 .AND. INDEX(STRING,'E').EQ.0

      IF (DECCHK(STRING).AND.NOP.AND.ILR.LE.11) THEN
*        string is probably valid integer ; decode it
         HELP = ' '
         HELP = ' '//STRING
         WRITE (FORM,'(A,I2,A)') '(I',ILR+1,')'
         READ (HELP,FORM,ERR=10) IVALUE
         IWAR = 0
         RETURN
      END IF

10    CONTINUE
      IWAR   = 1
      IVALUE = 0
      RETURN
      END
      SUBROUTINE DECREA (IWAR,STRING,VALUE)

*     Decodes a real number from a character string

*     IWAR   - In case of error IWAR = 1, otherwise IWAR = 0      O
*     STRING - input string, NO trailing or leading blanks !      I
*     VALUE  - Real value read from string                        O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK

*     Author: Kees Rappoldt
```

```
*     Date  : September 1989


*     formal parameters
      INTEGER IWAR
      REAL VALUE
      CHARACTER*(*) STRING

**    local variables + functions used
      INTEGER ILR
      CHARACTER HELP*21,FORM*7
      LOGICAL DECCHK
      SAVE

      ILR = LEN (STRING)

      IF (DECCHK(STRING) .AND. ILR.LE.20) THEN
*        string is probably valid real ; decode it
         HELP = ' '
         HELP = ' '//STRING
         WRITE (FORM,'(A,I2,A)') '(F',ILR+1,'.0)'
         READ (HELP,FORM,ERR=10) VALUE
         IWAR = 0
         RETURN
      END IF

10    CONTINUE
      IWAR  = 1
      VALUE = 0.0
      RETURN
      END
      SUBROUTINE ERROR (MODULE,MESSAG)

*     Writes an error message to the screen and holds the
*     screen until the <RETURN> is pressed.
*     Then execution is terminated.

*     MODULE  - string containing the module name                I
*     MESSAG  - string containing the message                    I
*
*     No subroutines and/or functions called

*     Author: Daniel van Kraalingen
*     Date  : October 1989


*     formal parameters
      CHARACTER*(*) MODULE, MESSAG

**    local variables
      INTEGER I
      CHARACTER*1 DUMMY
      SAVE



*     fudge construction to fool the compiler about the return statement
      I = 0
      IF (I.EQ.0) THEN
         WRITE (*,'(4A)') ' ERROR in ',MODULE,': ',MESSAG
         WRITE (*,'(A)')  ' Press <RETURN>'
         READ (*,'(A)') DUMMY
         STOP
      END IF

      RETURN
      END
      SUBROUTINE EXTENS (FILEIN,NEWEXT,ICHECK,FILEOU)

*     Changes extension of filename. Output filename is filled
*     with characters of input filename and new extension until
*     end is reached. Output filename is in uppercase characters.
*     The old extension is the part of the filename that follows
*     a dot (.). A dot before a bracket (]) is neglected (VAX).
*     The input filename does not necessarily have an extension.

*     FILEIN - Input filename with old or without extension      I
*     NEWEXT - New extension ; is set to uppercase               I
*     ICHECK = 1 --> check on equal output and input extension   I
*            = 0 --> no check
*     FILEOU - Output filename with new extension in uppercase   O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR, ILEN, UPPERC

*     Author: Kees Rappoldt
*     Date  : October 1989


*     formal parameters
      INTEGER ICHECK
      CHARACTER*(*) FILEIN,FILEOU,NEWEXT

**    local variables + function called
      INTEGER I,ILEXT1,ILEXT2,ILFIL,IP,LOUT,ILEN
      CHARACTER*1 OLD,NEW,CHR
      LOGICAL CHECK, HAAK, FOUND
      SAVE


*     length of input filename and new extension
      ILFIL  = ILEN (FILEIN)
      ILEXT2 = ILEN (NEWEXT)
      LOUT   = LEN (FILEOU)
      IF (ILFIL.EQ.0)  CALL ERROR ('EXTENS','no filename supplied')
      IF (ILEXT2.EQ.0) CALL ERROR ('EXTENS','no new extension supplied')

*     initialize search for "no extension"
      IP     = ILFIL + 1
      ILEXT1 = 0
      HAAK   = .FALSE.
      FOUND  = .FALSE.

*     search for point from the end on
      I = ILFIL
10    IF (I.GT.0 .AND..NOT.(FOUND.OR.HAAK)) THEN
*        check single character
         CHR  = FILEIN(I:I)
         HAAK = CHR.EQ.']'

         IF (CHR.EQ.'.') THEN
*           a (valid) point is found
            IP = I
            ILEXT1 = ILFIL - IP
            FOUND  = .TRUE.
         END IF

         I = I - 1
      GOTO 10
      END IF

*     check and set output name in uppercase
      IF (LOUT.LT.IP+1)
     $ CALL ERROR ('EXTENS','output string too short for new name')
      FILEOU = ' '
      IF (IP.GE.2) FILEOU = FILEIN(1:IP-1)
      CALL UPPERC (FILEOU)
*     set point
      FILEOU(IP:IP) = '.'

*     actual length of new extension
      ILEXT2 = MIN (ILEXT2, LOUT-IP)
*     if old and new extension are equally long ; check !!
      CHECK = (ILEXT1.EQ.ILEXT2) .AND. (ICHECK.NE.0)

      DO 20 I=IP+1,IP+ILEXT2
*        get character new extension and put into output
         NEW = NEWEXT(I-IP:I-IP)
         CALL UPPERC (NEW)
         FILEOU(I:I) = NEW

         IF (CHECK) THEN
*           get character old extension for comparison
            OLD = FILEIN(I:I)
            CALL UPPERC (OLD)
```

```
            IF (OLD.NE.NEW) CHECK=.FALSE.
          END IF
20      CONTINUE

        IF (CHECK) CALL ERROR ('EXTENS','old and new extension equal')

        RETURN
        END
        SUBROUTINE FOPEN (IUNIT,FILE,STATUS,PRIV)

*       Opens a sequential, formatted file
*       after checking on its the existence

*       IUNIT  - unit number used to open file                    I
*       FILE   - name of the file to be opened                    I
*       STATUS - status of the file                               I
*       PRIV   - privilege ; in case status='new' and file exists:  I
*                = 'del' --> old file is overwritten
*                = 'nod' --> old file saved, program stopped
*                = 'unk' --> interactive choice
*
*       Subroutines and/or functions called:
*       - from library TTUTIL: ERROR, ILEN, UPPERC

*       Author: Daniel van Kraalingen
*       Date  : October 1989


*       formal parameters
        INTEGER IUNIT
        CHARACTER*(*) FILE, STATUS, PRIV

**      local variables + function called
        INTEGER ILFIL, ILEN
        CHARACTER CHOICE*1, DUMMY*1, CODE*3
        LOGICAL EXIST,OPENT
        SAVE



*       length of filename, existence and requested status
        ILFIL = ILEN (FILE)
        INQUIRE (FILE=FILE(1:ILFIL),EXIST=EXIST,OPENED=OPENT)
        IF (OPENT) CALL ERROR ('FOPEN','File already opened')
        CODE = STATUS
        CALL UPPERC (CODE)

        IF (CODE.EQ.'OLD') THEN
*         file should exist
          IF (EXIST) THEN
*           open existing file
            OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='OLD')
          ELSE
            WRITE (*,'(3A)') ' File ',FILE(1:ILFIL),' does not exist'
            CALL ERROR ('FOPEN','program stopped')
          END IF

        ELSE IF (CODE.EQ.'NEW') THEN
*         file may exist or not
          IF (.NOT.EXIST) THEN
*           open new file
            OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='NEW')

          ELSE
*           depends on privilege
            CODE = PRIV
            CALL UPPERC (CODE)

            IF (CODE.EQ.'DEL') THEN
*             delete file ; overwriting does not work on MAC
              OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='OLD')
              CLOSE (IUNIT,STATUS='DELETE')
              OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='NEW')

            ELSE IF (CODE.EQ.'UNK') THEN
*             interactive choice
              WRITE (*,'(3A,/,A$)')
     $         ' File ',FILE(1:ILFIL),' already exists',
     $         ' Overwrite (Y/N): '
              READ (*,'(A)') CHOICE

              IF (CHOICE.EQ.'Y'.OR.CHOICE.EQ.'y') THEN
                OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='OLD')
                CLOSE (IUNIT,STATUS='DELETE')
                OPEN (IUNIT,FILE=FILE(1:ILFIL),STATUS='NEW')
              ELSE
                WRITE (*,'(A,/,A)')
     $           ' File not overwritten, program stopped',
     $           ' press <RETURN>'
                READ (*,'(A)') DUMMY
                STOP
              END IF

            ELSE IF (CODE.EQ.'NOD') THEN
              WRITE (*,'(3A)')
     $         ' Existing file ',FILE(1:ILFIL),' is not deleted'
              CALL ERROR ('FOPEN','program stopped')

            ELSE
              CALL ERROR ('FOPEN','Unknown privilege')
            END IF

          END IF
        ELSE
          CALL ERROR ('FOPEN','Unknown file status')
        END IF

        RETURN
        END
        INTEGER FUNCTION IFINDC (NAMLIS,ILDEC,IST,IEND,NAME)

*       Finds number of name in a list with names ; when
*       name is not in the list a zero value is returned
*       Character strings should be of the same length !!

*       IFINDC - element where a match was found                  O
*       NAMLIS - character string array, the "list"               I
*       ILDEC  - declared length of array NAMLIS                  I
*       IST    - array element where search should start          I
*       IEND   - actual size of the list with names               I
*       NAME   - name to be found in the list                     I
*
*       Subroutines and/or functions called:
*       - from library TTUTIL: ERROR

*       Author: Kees Rappoldt
*       Date  : September 1989


*       formal parameters
        INTEGER ILDEC, IST, IEND
        CHARACTER*(*) NAMLIS, NAME
        DIMENSION NAMLIS(ILDEC)

**      local variables
        INTEGER IN, IM, IL1, IL2
        SAVE

*       error check
        IL1 = LEN (NAMLIS(1))
        IL2 = LEN (NAME)
        IF (IL1.NE.IL2) CALL ERROR ('IFINDC','string size mismatch')

        IM = 0
        IN = IST
10      IF (IN.LE.IEND .AND. IM.EQ.0) THEN
          IF (NAMLIS(IN).EQ.NAME) IM = IN
          IN = IN+1
        GOTO 10
        END IF

        IFINDC = IM

        RETURN
        END
```

```
      INTEGER FUNCTION ILEN (STRING)

*     Determines the significant length of a string.
*     If the string is empty a zero is returned.

*     ILEN   - returned length          O
*     STRING - input string             I
*
*     No subroutines and/or functions called

*     Author: Daniel van Kraalingen
*     Date  : October 1989


*     formal parameter
      CHARACTER*(*) STRING

**    local variables
      INTEGER I, L
      SAVE

      L = LEN (STRING)

      DO 10 I=L,1,-1
         IF (STRING(I:I).NE.' ') THEN
            ILEN = I
            RETURN
         END IF
10    CONTINUE

      ILEN = 0

      RETURN
      END
      REAL FUNCTION INSW (X1,X2,X3)

*     Input switch depending on sign of X1 ;
*     function is equivalent to the CSMP-INSW.

*     INSW - returned value                                  O
*     X1   - identifier upon which the test is done          I
*     X2   - value of INSW in case X1 <  0,                  I
*     X3   - value of INSW in case X1 >= 0,                  I
*
*     No subroutines and/or functions used

*     Author: Daniel van Kraalingen
*     Date  : October 1989


*     formal parameters
      REAL X1,X2,X3

**    no local variables
      SAVE

      IF (X1.LT.0.) THEN
         INSW = X2
      ELSE
         INSW = X3
      END IF

      RETURN
      END
      REAL FUNCTION INTGRL (STATE, RATE, DELT)

*     Function value = STATE + RATE * DELT.

*     INTGRL - function name, new state          O
*     STATE  - old state                         I
*     RATE   - rate of change per unit time      I
*     DELT   - time step                         I
*
*     No subroutines of functions called

*     Author: Daniel van Kraalingen
*     Date  : December 1989


*     formal parameters
      REAL STATE, RATE, DELT

**    no local variables
      SAVE

      INTGRL = STATE + RATE * DELT

      RETURN
      END

      INTEGER FUNCTION ISTART (STRING)

*     Determines the first significant character of a string.
*     If the string contains no characters, a zero is returned.

*     ISTART - returned value           O
*     STRING - input string             I
*
*     No subroutines and/or functions called

*     Author: Daniel van Kraalingen
*     Date  : October 1989


*     formal parameter
      CHARACTER*(*) STRING

**    local variables
      INTEGER I, L
      SAVE

      L = LEN (STRING)

      DO 10 I=1,L
         IF (STRING(I:I).NE.' ') THEN
            ISTART = I
            RETURN
         END IF
10    CONTINUE

      ISTART = 0

      RETURN
      END
      REAL FUNCTION LIMIT (MIN,MAX,X)

*     Returns value of X limited within the interval [MIN,MAX]

*     MIN - interval lower boundary     I
*     MAX - interval upper boundary     I
*     X   - Argument of function        I
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR

*     Author: Kees Rappoldt
*     Date  : January 1990


*     formal parameters
      REAL MIN,MAX,X

**    local parameters
      CHARACTER MESSAG*52
      SAVE

      IF (MAX.LT.MIN) THEN
*        minimum is larger than maximum, should generate an error
         WRITE (MESSAG, '(2(A,G11.5))')
     &   'argument error, MIN = ',MIN,', MAX = ',MAX
         CALL ERROR ('LIMIT',MESSAG)
      END IF
```

```
      IF (X.LT.MIN) THEN
*        X below allowed range ; return lower bound
         LIMIT = MIN

      ELSE IF (X.LE.MAX) THEN
*        X in range ; return X
         LIMIT = X

      ELSE
*        X above allowed range ; return upper bound
         LIMIT = MAX

      END IF

      RETURN
      END
      REAL FUNCTION LINT (TABLE,ILTAB,X)

*     This function is a linear interpolation function. The
*     function also extrapolates outside the defined region in
*     case X is below or above the region defined by TABLE.
*     Extrapolation, however, results in a warning to the screen.

*     LINT  - function name, result of the interpolation     O
*     TABLE - a one-dimensional array with paired            I
*             data: x1,y1,x2,y2, etc.
*     ILTAB - number of elements in the array TABLE          I
*     X     - the value at which interpolation should        I
*             take place
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR

*     Author: Daniel van Kraalingen
*     Date  : January 1990


*     formal parameters
      INTEGER ILTAB
      REAL TABLE(ILTAB), X

**    local variables
      INTEGER I1, ILT, IUP
      REAL SLOPE, TINY
      PARAMETER (TINY=1.E-7)
      LOGICAL ERR, WARN
      SAVE

*     initialize
      ERR  = .FALSE.
      WARN = .FALSE.

*     check on value of ILTAB
      IF (MOD(ILTAB,2).NE.0 .OR. ILTAB.LE.2) THEN
         WRITE (*,'(A)')
     $    ' Number of elements in LINT table not correct !'
         ERR = .TRUE.

      ELSE
         IUP = 0
         DO 10 I1=3,ILTAB,2
*           check on ascending order of X-values in function
            IF (TABLE(I1).LE.TABLE(I1-2)) THEN
               WRITE (*,'(2A,I4)')
     $          ' X-coordinates in LINT table not in',
     $          ' ascending order at point',I1
               ERR = .TRUE.
            END IF
            IF (IUP.EQ.0 .AND. TABLE(I1).GE.X) IUP = I1
10       CONTINUE
      END IF

      IF (.NOT.ERR .AND. X.LT.TABLE(1)) THEN
         IUP = 3
         IF ((TABLE(1)-X) .GT. ABS(X)*TINY) THEN
            WARN = .TRUE.
            WRITE (*,'(A,G13.5)')
     $       ' WARNING in LINT: X-value below defined region at X=',X
         END IF
      ELSE IF (.NOT.ERR .AND. X.GT.TABLE(ILTAB-1)) THEN
         IUP = ILTAB-1
         IF ((X-TABLE(ILTAB-1)) .GT. ABS(X)*TINY) THEN
            WARN = .TRUE.
            WRITE (*,'(A,G13.5)')
     $       ' WARNING in LINT: X-value above defined region at X=',X
         END IF
      END IF

      IF (WARN.OR.ERR) THEN
         ILT = MIN (ILTAB/2, 15)
         WRITE (*,'(A,I4,/,A,I2,A)')
     $     ' Number of table elements is',ILTAB,
     $     ' First ',ILT,' pairs are:'
         IF (ILT.GT.0) WRITE (*,'(2G13.5)')
     $      (TABLE(I1),TABLE(I1+1),I1=1,2*ILT,2)
         IF (ERR) CALL ERROR ('LINT','execution terminated')
      END IF

*     interpolation and extrapolation
      SLOPE = (TABLE(IUP+1)-TABLE(IUP-1))/(TABLE(IUP)-TABLE(IUP-2))
      LINT  = TABLE(IUP-1)+(X-TABLE(IUP-2))*SLOPE

      RETURN
      END
      SUBROUTINE OUTCOM (STR)

*     Stores a text string which is written to
*     the output file generated by OUTDAT. A maximum
*     number of 25 strings of 80 characters can be stored.

*     STR - text string      I
*
*     Subroutines and/or functions called:
*     from library TTUTIL: IFINDC, ERROR, ILEN

*     Author: Daniel van Kraalingen
*     Date  : December 1989


*     Example:
*     CALL OUTCOM ('Potential production')
*     CALL OUTCOM ('and water limited production')
*     CALL OUTDAT (4, 0, 'Final output',0.)
*     both text strings will appear in the final output file.


*     formal parameters
      CHARACTER*(*) STR

**    local variables and used functions
      INTEGER ICOM, NCOM, I1, IL, ILU1, ILU2, ILEN, IFINDC
      PARAMETER (NCOM=25)
      CHARACTER*80 COMMNT(NCOM), TMP
      LOGICAL OPEN, TERMNL
      COMMON /OUTCUT/TERMNL, ILU1, ILU2
      SAVE

      DATA ICOM /0/

      IF (STR.EQ.'<INIT$$$>') THEN
         ICOM = 0
      ELSE IF (STR.EQ.'<PRINT$$$>' .AND. ICOM.GT.0) THEN
         INQUIRE (UNIT=ILU2, OPENED=OPEN)
         IF (.NOT.OPEN) CALL ERROR ('OUTCOM','Output file not open')
         DO 10 I1=1,ICOM
            IL = ILEN (COMMNT(I1))
            WRITE (ILU2,'(2A)') ' * ',COMMNT(I1)(1:IL)
10       CONTINUE
      ELSE
         IF (ICOM.LT.NCOM) THEN
            IL  = ILEN (STR)
            TMP = STR
            I1  = IFINDC (COMMNT, NCOM, 1, ICOM, TMP)
            IF (IL.GT.0 .AND. I1.EQ.0) THEN
```

```
            ICOM = ICOM+1
            COMMNT(ICOM) = STR
          END IF
        ELSE
          CALL ERROR ('OUTCOM','Too many comment lines')
        END IF
      END IF

      RETURN
      END
      SUBROUTINE OUTDAT (ITASK, IUNIT, RN, R)

*     Can be used to generate output files from within simulation mod-
els.
*     It should be initialized first, to define the name of the
*     independent variable and to set unit numbers (ITASK=1). The name
*     and value of the data are stored by calls with ITASK=2, supplying
*     the name and the value to OUTDAT. The output file is generated by
a
*     call with ITASK=4, 5, or 6. These generate table output, spread-
sheet
*     output and two column output, respectively.

*     ITASK - integer, can be 1, 2, 4, 5, or 6. 1 initializes the
*             subroutine, opens a temporary file for storage, and stores
*             the name of the independent variable and the unit number.
*             2 stores the name and value of the variable in a temporary
*             file. 4, 5, or 6 generate an output file.  After 4, 5, or
6
*             the subroutine can be used again by using ITASK=1.
*     IUNIT - unit number used for writing to output file. If the unit
*             defined during ITASK=1 is open this is used for output.
*             Otherwise a file 'res.dat' using that unit is created.
*             IUNIT+1 is used I/O to the temporary file.
*     RN    - string, name of variable, (up to 11 characters
*             will be used). If ITASK is 4, 5, or 6, this string
*             will be written to the output file as title
*             (not limited to 11 characters).
*     R     - value of variable (only effective at ITASK=2).
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, FOPEN, IFINDC, ERROR, OUTCOM, UPPERC

*     Author: Daniel van Kraalingen, Kees Rappoldt
*     Date  : December 1989


*     Example:
*
*     CALL OUTDAT (1,20,'TIME',0.)      initialization, TIME is
*                                       made independent variable
*     CALL OUTDAT (2,20,'TIME',TIME)    value of TIME is stored
*     CALL OUTDAT (2,20,'D(11)',D(11)) value of DTGA is stored
*
*     repeated calls to OUTDAT with ITASK=2
*
*     CALL OUTDAT (4,20,'Plottitle',0.)  generate output file with
*                                        table format
*     CALL OUTDAT (6,20,'plottitle',0.)  generate output file with
*                                        two column format
*     CALL OUTDAT (1,20,'TIME',0.)    initialize again
*     etc.


*     formal parameters
      INTEGER ITASK, IUNIT
      REAL R
      CHARACTER*(*) RN

**    Settings to be adjusted for different formats.
*     Numbers to be changed are given in comment lines below,
*     IMNCOL means maximum number of columns (default = 9)
*     IFW    means the field width for a single variable (default = 12)
*            adjust the declared lengths of the format strings if neces-
sary.
*     IMPORTANT: if you make changes you must make explicit all the
*     occurences of n and m in the uncommented declaration. Also the
*     format 'calculations' in the initial section of the routine should
*     not be forgotten.
*     INTEGER IMNC1, IFW
*     PARAMETER (IMNC1=n,IFW=m)
*     CHARACTER FORMS*7, FORMV*13, FORMH*19
*     CHARACTER AVV(255)*m, LXV*m, ZERO*m

      INTEGER IMNC1, IFW
      PARAMETER (IMNC1=9,IFW=12)
      CHARACTER FORMS*7, FORMV*13, FORMH*19
      CHARACTER AVV(255)*12, LXV*12, ZERO*12

*     local variables
      INTEGER ILU1, ILU2, IBLOK, I, I1, I2, I3, I4, I5, IOS, INREC, IR
      INTEGER IFINDC, IAVNL(255), ILXNL, ITOLD, IRUN, IMNC2, ILEN, IB
      REAL LX, LV, LVO
      LOGICAL LTEMP, OPENT, OPEND, YFND, YNFND, INIT, INITF, TERMNL
      LOGICAL FOUND(255)
      CHARACTER LXN*11, LN*11, AVN(255)*11, CHR*1, HULP1*2, HULP2*2
      CHARACTER*22 TEXT(4:6)

      COMMON /OUTCUT/ TERMNL, ILU1, ILU2

      SAVE

      DATA INIT/.FALSE./, INITF /.FALSE./, ITOLD /0/, IRUN /0/
      DATA TEXT /', (Table output)      ',
     &           ', (Spreadsheet output)',
     &           ', (2 column output)   '/

      IF (ITASK.EQ.1) THEN

*       initialization of unit numbers
        ILU1 = IUNIT+1
        ILU2 = IUNIT
        CALL OUTCOM ('<INIT$$$>')

*       increase run number
        IRUN = IRUN+1

*       this section deletes the temporary file
*       see if old file for temporary storage is open, if true
*       delete it by closing it. If the file is not open see if it
*       exists, open it and delete it

        INQUIRE (FILE='RES.TMP', OPENED=OPENT,  EXIST=LTEMP)
        IF (OPENT) THEN
          CLOSE (ILU1, STATUS='DELETE')
        ELSE IF (LTEMP) THEN
          OPEN (ILU1, FILE='RES.TMP', STATUS='OLD',
     &          FORM='UNFORMATTED', ACCESS='SEQUENTIAL')
          CLOSE (ILU1, STATUS='DELETE')
        END IF

*       new file is opened for temporary storage
        OPEN (ILU1,FILE='RES.TMP', STATUS='NEW',
     &        FORM='UNFORMATTED', ACCESS='SEQUENTIAL')

*       logicals indicate that nothing has been read from the file.
        TERMNL = .FALSE.
        YNFND = .FALSE.

*       name of independent variable is assigned to local string

        LXN = RN
        WRITE (ILU1) LXN, R
        INIT = .TRUE.

      ELSE IF (ITASK.EQ.2) THEN

*       check file status first
        IF (.NOT.INIT .OR. TERMNL) CALL ERROR
     &          ('OUTDAT','Initialization not done')

*       values and names are written to file
        LN = RN
        WRITE (ILU1) LN, R
```

```
      ELSE IF (ITASK.EQ.4.OR.ITASK.EQ.5.OR.ITASK.EQ.6) THEN

         IF (ITOLD.EQ.1)
     &   CALL ERROR ('OUTDAT','No variables for output')

*        the routine is instructed to read the temporary file
*        and generate an output file
*        4: Table output, 5: Spreadsheet output, 6: Two column output

*        construct formats first

         IF (.NOT.INITF) THEN

*           output formats are written,
*           IMPORTANT: adjustments must be made in the declaration
*           section

            WRITE (HULP1,'(I2)') IFW
            I = IFW-4
            WRITE (HULP2,'(I2)') I

*           Single value write format (FORMS), variable name header
*           format (FORMH) and value line format FORMV.

            FORMS = '(G'//HULP1//'.5)'
            FORMH = '(A,A'//HULP2//',255(A1,A'//HULP1//'))'
            FORMV = '(255(A1,A'//HULP1//'))'

*           a string containing a zero has to be used in the spreadsheet
*           output sometimes

            ZERO = ' '
            I = LEN (ZERO)
            ZERO(I-1:I) = '0.'

            INITF = .TRUE.

         END IF

*        if temporary file is open close it and open it
*        this seems trivial but is necessary to make recovery of
*        output possible after a run-time error

         INQUIRE (FILE='RES.TMP', OPENED=OPENT,  EXIST=LTEMP)
         IF (OPENT) THEN
            CLOSE (ILU1)
         ELSE IF (.NOT.LTEMP) THEN
            CALL ERROR ('OUTDAT',' temporary file does not exist')
         END IF

         OPEN (ILU1, FILE='RES.TMP', STATUS='OLD',
     &         FORM='UNFORMATTED', ACCESS='SEQUENTIAL')
         TERMNL = .TRUE.

*        check if output file is open, make new one if it is not opened

         INQUIRE (UNIT=ILU2, OPENED=OPEND)
         IF (.NOT.OPEND) CALL FOPEN (ILU2,'RES.DAT','NEW','DEL')

*        character to be written to the output file between the values
*        is assigned
         IF (ITASK.EQ.4) THEN
            CHR = ' '
            IMNC2 = IMNC1
         ELSE IF (ITASK.EQ.5) THEN
            CHR = CHAR(9)
            IMNC2 = IMNC1
         ELSE IF (ITASK.EQ.6) THEN
            CHR = ' '
            IMNC2 = 1
         END IF

*        find number of Y-variable names if it has not been found
*        before

         IF (.NOT.YNFND) THEN

*           read name of independent variable
            READ (ILU1) LXN, LX
            ILXNL = ILEN (LXN)
            INREC = 1
            I1 = 0
            IOS = 0
10          READ (ILU1,IOSTAT=IOS) LN, LV
            IF (IOS.EQ.0) THEN
               INREC = INREC+1

*              check record for new name
               IF (LN.NE.LXN) THEN
*                 record is 'Y' ; check found names
                  IF (IFINDC (AVN,255,1,I1,LN).EQ.0) THEN
*                    new name found
                     I1 = I1 + 1
                     IF (I1.GT.255)
     &                  CALL ERROR ('OUTDAT','too many variables')
                     AVN(I1) = LN
                  END IF
               END IF

               GOTO 10
            END IF

            YNFND = .TRUE.

*           determine lengths of Y-names (I1 is the number of names)
            DO 20 I=1,I1
               IAVNL(I) = ILEN (AVN(I))
               IF (IAVNL(I).EQ.0) THEN
                  WRITE (*,'(A)')
     &              ' WARNING from OUTDAT: zero length variable name'
                  WRITE (ILU2,'(A)')
     &              ' WARNING from OUTDAT: zero length variable name'
                  IAVNL(I) = 1
               END IF
20          CONTINUE
         END IF

*        write line to mark start of new run
         WRITE (ILU2,'(A2,76A1)') ' *',('-',I=1,76)

*        write header to output file and possible extra comment lines
         WRITE (ILU2,'(A,I3,A,/,2A)')
     &     ' * Run no.:', IRUN, TEXT(ITASK), ' * ', RN
         CALL OUTCOM ('<PRINT$$$>')

*        generate output, determine number of blocks first
         IBLOK = 1 + (I1-1)/IMNC2
         DO 30 IB=1,IBLOK
            REWIND (ILU1)

*           number of first and last 'Y' in block
            I2 = 1 + (IB-1) * IMNC2
            I3 = MIN (I2+IMNC2-1,I1)

*           header with variable names is written dependent on ITASK
            IF (ITASK.EQ.6) THEN
               WRITE (ILU2,FORMH)
     &                ' * ',LXN(1:ILXNL),CHR,AVN(I2)(1:IAVNL(I2))
               WRITE (ILU2,'(A)') ' 1 1 1'
               WRITE (ILU2,'(1X, 4A)')
     &                AVN(I2)(1:IAVNL(I2)),'(',LXN(1:ILXNL),')'
            ELSE
               WRITE (ILU2,'(A)') ' '
               WRITE (ILU2,FORMH)
     &                ' ',LXN(1:ILXNL),(CHR,AVN(I)(1:IAVNL(I)),I=I2,I3)
               WRITE (ILU2,'(A)') ' '
            END IF

*           initialize output
            YFND = .FALSE.

            DO 40 I4=1,255
               FOUND(I4) = .FALSE.
40          CONTINUE
```

```
            DO 50 IR=1,INREC
*              read next record
               READ (ILU1) LN, LV

*              see if variable name is the independent variable
               IF (LN.EQ.LXN) THEN
                  IF (YFND.AND.LV.NE.LVO) THEN

*                    internal write of variable value on string
                     WRITE (LXV,FORMS) LVO
                     WRITE (ILU2,FORMV) ' ',LXV,(CHR,AVV(I),I=I2,I3)

*                    initialize search for 'Y' values
                     DO 60 I=I2,I3
                        FOUND(I) = .FALSE.
                        IF (ITASK.EQ.3) THEN
                           AVV(I) = ' '
                        ELSE
                           AVV(I) = ZERO
                        END IF
60                   CONTINUE
                     YFND = .FALSE.
                  END IF
                  LVO = LV
               ELSE
*                 record contains 'Y' ; check names I2...I3
                  I = IFINDC (AVN, 255, I2, I3, LN)
                  IF (I.NE.0) THEN
                     IF (.NOT.FOUND(I)) THEN
*                       Y found, write value to output record
                        FOUND(I) = .TRUE.
                        YFND = .TRUE.
                        WRITE (AVV(I),FORMS) LV
                     ELSE
                        I4 = ILEN (LN)
                        I5 = ILEN (LXN)
                        WRITE (*,'(4A)')
     &                    ' WARNING from OUTDAT: variable ',LN(1:I4),
     &                    ' occurs twice at same value of ',LXN(1:I5)
                        WRITE (ILU2,'(4A)')
     &                    ' WARNING from OUTDAT: variable ',LN(1:I4),
     &                    ' occurs twice at same value of ',LXN(1:I5)
                     END IF
                  END IF
               END IF
50          CONTINUE

*           write last line at E_O_F
            WRITE (LXV,FORMS) LVO
            IF (YFND) WRITE (ILU2,FORMV) ' ',LXV,(CHR,AVV(I),I=I2,I3)
30       CONTINUE

*        add some blank lines if table or spreadsheet output was
*        choosen
         IF (ITASK.EQ.4.OR.ITASK.EQ.5) WRITE (ILU2,'(/,/,/,1X)')

         CLOSE (ILU1)

      ELSE IF (ITASK.EQ.99) THEN

         IF (ITOLD.EQ.99)
     &   CALL ERROR ('OUTDAT','Temporary file already deleted')

*        this option deletes the temporary file

         OPEN (ILU1, FILE='RES.TMP', STATUS='OLD',
     &         FORM='UNFORMATTED', ACCESS='SEQUENTIAL')
         CLOSE (ILU1,STATUS='DELETE')

      ELSE

         CALL ERROR ('OUTDAT','Unknown ITASK')

      END IF

      ITOLD = ITASK

      RETURN
      END

      SUBROUTINE OUTPLT (ITASK, RN)

*     Designed to be used in conjunction with OUTDAT, which is used
*     to write variable name and value to a temporary file. OUTPLT is
*     used to printplot a selection of the stored variables. By repeated
*     calls to the subroutine with ITASK=1, names of variable for which
*     the plot is wanted can be given to the subroutine.
*     By a call with ITASK=4, 5, 6, or 7, printplots are generated with
*     a width of 80 or 132 characters, either with individual scaling
*     or with common scaling (all variables scaled to the smallest and
*     largest value in the data set).

*     ITASK - integer, can be 1, 4, 5, 6, or 7. Repeated calls with a
*             1 instruct the routine to store variable names for
*             use in the printplot. Calls with 4, 5, 6, and 7 generate
*             the printplot with the variables as defined with ITASK=1.
*             4 means wide format, individual scale,
*             5 means wide format, common scale,
*             6 means small format, individual scale,
*             7 means small format, common scale,
*             If the unit defined during ITASK=1 of OUTDAT is open,
*             this is used for output. Otherwise
*             a file 'res.dat' with that unit is created.
*     RN    - string, name of variable, up to 11 characters will be
*             used. The value of the variable must have been stored
*             by previous calls to OUTDAT.
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, FOPEN, IFINDC, ERROR, UPPERC

*     Author: Daniel van Kraalingen
*     Date  : December 1989


*     Example:
*
*     CALL OUTPLT (1,'DTGA')          define DTGA to be plotted
*     CALL OUTPLT (1,'WSO')           define WSO to be plotted
*     CALL OUTPLT (5,'Plot title')    make printplot using
*                                     wide format, common scale


*     formal parameters
      INTEGER ITASK
      CHARACTER*(*) RN

**    local variables and used functions
      INTEGER IN, I, ILU1, ILU2, INREC, I1, I2, I3, IY
      INTEGER IWRITE, IFINDC, ILEN, IOS, IEND
      CHARACTER*11 AVN(25), LXN, LN
      CHARACTER RECORD*109, REC1*109, REC2*109
      CHARACTER MARK(25)*1, LXV*12, CHR*1
      REAL AVMIN(25), AVMAX(25), LV, LVO, VMIN, VMAX, WIDTH
      LOGICAL FOUND(25), FNDONE, OPENT, OPEND, EXIST, INIT, YFND, TERMNL

*     a common with OUTDAT
      COMMON /OUTCUT/ TERMNL, ILU1, ILU2

      SAVE
      DATA IN /0/, INIT /.FALSE./
      DATA MARK /'1','2','3','4','5','6','7','8','9','0',
     &           'A','B','C','D','E','F','G','H','I','J',
     &           'K','L','M','N','P'/

*     'define variable name' option

      IF (ITASK.EQ.1) THEN

         IF (.NOT.INIT) THEN
            IN = 0
            INIT = .TRUE.
         END IF
```

```
*        place new name in array of names, check for duplicates
*        and number of variables

         LXN = RN
         I = IFINDC (AVN, 25, 1, IN, LXN)
         IF (I.EQ.0) THEN
*           new variable found
            IN = IN+1
            IF (IN.GE.25) CALL ERROR ('OUTPLT','Too many variables')
            AVN(IN) = LXN
         END IF

       ELSE IF (ITASK.EQ.4.OR.ITASK.EQ.5.OR.
     &          ITASK.EQ.6.OR.ITASK.EQ.7) THEN

*        ITASK = 4 wide plot,  individual scaling
*        ITASK = 5 wide plot,  common scaling
*        ITASK = 6 small plot, individual scaling
*        ITASK = 7 small plot, common scaling

*        graph output option
         IF (IN.EQ.0) CALL ERROR ('OUTPLT','Initialization not done')

*        make sure that temporary file is opened and rewound

         INQUIRE (FILE='RES.TMP', OPENED=OPENT, EXIST=EXIST)
         IF (OPENT) THEN
            CLOSE (ILU1)
         ELSE IF (.NOT.EXIST) THEN
            CALL ERROR ('OUTPLT','Temporary file not found')
         END IF

         OPEN (ILU1, FILE='RES.TMP', STATUS='OLD',
     &         FORM='UNFORMATTED', ACCESS='SEQUENTIAL')
         TERMNL = .TRUE.

*        check if output file is open, else delete old one and
*        create new one

         INQUIRE (UNIT=ILU2, OPENED=OPEND)
         IF (.NOT.OPEND) CALL FOPEN (ILU2,'RES.DAT','NEW','DEL')

*        set minimum and maximum values to zero, single variables
*        are used for common scaling
         VMIN = 0.
         VMAX = 0.
         DO 20 I=1,25
            AVMIN(I) = 0.
            AVMAX(I) = 0.
            FOUND(I) = .FALSE.
20       CONTINUE

*        read independent variable
         READ (ILU1) LXN, LV
         INREC = 1

*        determine minimum and maximum values from the file
*        and the number of records

         IOS = 0
         FNDONE = .FALSE.
30       READ (ILU1, IOSTAT=IOS) LN, LV
         IF (IOS.EQ.0) THEN
            INREC = INREC+1
            I = IFINDC (AVN, 25, 1, IN, LN)
            IF (I.GT.0) THEN

               IF (.NOT.FOUND(I)) THEN
                  AVMIN(I) = LV
                  AVMAX(I) = LV
                  FOUND(I) = .TRUE.
               END IF

               IF (.NOT.FNDONE) THEN
                  VMIN = LV
                  VMAX = LV
                  FNDONE = .TRUE.
               END IF

               IF (LV.LT.AVMIN(I)) AVMIN(I) = LV
               IF (LV.GT.AVMAX(I)) AVMAX(I) = LV
               IF (LV.LT.VMIN) VMIN = LV
               IF (LV.GT.VMAX) VMAX = LV

            END IF
            GOTO 30
         END IF

*        prepare records
         REC1 = ' '
         REC2 = ' '

         IF (ITASK.EQ.4 .OR. ITASK.EQ.5) THEN
*           wide format plots
            DO 40 I2=2,108
               REC1(I2:I2) = '-'
40          CONTINUE
            REC1(1:1) = 'I'
            REC1(109:109) = 'I'

            DO 50 I2=1,109,18
               REC2(I2:I2) = 'I'
50          CONTINUE
            IEND = 109
            WIDTH = 108.
         ELSE
*           small format plots
            DO 60 I2=2,64
               REC1(I2:I2) = '-'
60          CONTINUE
            REC1(1:1) = 'I'
            REC1(65:65) = 'I'

            DO 70 I2=1,65,16
               REC2(I2:I2) = 'I'
70          CONTINUE
            IEND = 65
            WIDTH = 64.
         END IF

*        write header to output file
         WRITE (ILU2,'(/,14X,A,/)') RN
         WRITE (ILU2,'(14X,4A,/,14X,4A)')
     &      'Variable  ','Marker  ','Minimum value  ','Maximum value',
     &      '--------  ','------  ','--------------  ','--------------'

*        write name, marker, minimum and maximum value
*        dependent on individual or common scaling

         DO 80 I=1,IN
            IF (FOUND(I)) THEN
               IF (AVMIN(I).NE.AVMAX(I)) THEN
                  WRITE (ILU2,'(14X,A11,2X,A,6X,G11.4,4X,G11.4)')
     &               AVN(I),MARK(I),AVMIN(I),AVMAX(I)
               ELSE
                  WRITE (ILU2,'(14X,A11,2X,A,6X,G11.4,4X,G11.4,A)')
     &               AVN(I),MARK(I),AVMIN(I),AVMAX(I),' no range to plot'
               END IF
            ELSE
               WRITE (*,'(4A)') ' WARNING from OUTPLT: ',
     &           'variable ', AVN(I), ' not in temporary file'
               WRITE (ILU2,'(4A)') ' WARNING from OUTPLT: ',
     &           'variable ', AVN(I), ' not in temporary file'
            END IF
80       CONTINUE

*        add common scaling text and values if necessary
         IF (ITASK.EQ.5 .OR. ITASK.EQ.7) THEN
            WRITE (ILU2,'(/,14X,A,5X,G11.4,4X,G11.4)')
     &        'Scaling: Common', VMIN, VMAX
         ELSE
            WRITE (ILU2,'(/,14X,A)')
     &        'Scaling: Individual'
         END IF
```

```
*        write name of independent variable
         WRITE (ILU2,'(/,2X,A,/)') LXN

*        read file again and plot
         REWIND (ILU1)
         YFND = .FALSE.

*        set number of plot lines written to file to zero
*        start plotting

         IWRITE = 0
         RECORD = REC1

         DO 90 I2=1,25
            FOUND(I2) = .FALSE.
90       CONTINUE

         DO 100 I1=1,INREC
            READ (ILU1) LN, LV

*              new name is independent name
               IF (LN.EQ.LXN) THEN
                  IF (YFND.AND.LV.NE.LVO) THEN

                     WRITE (LXV,'(G12.5)') LVO
                     WRITE (ILU2,'(2(1X,A))') LXV, RECORD(1:IENDD)
                     IWRITE = IWRITE+1

*                    take start record dependent on number of
*                    lines written

                     IF (MOD (IWRITE,10).EQ.0) THEN
                        RECORD = REC1
                     ELSE
                        RECORD = REC2
                     END IF

                     YFND = .FALSE.

                     DO 110 I2=1,25
                        FOUND(I2) = .FALSE.
110                  CONTINUE

                  END IF
                  LVO = LV
               ELSE

*                 Y name found
                  I = IFINDC (AVN, 25, 1, IN, LN)
                  IF (I.GT.0) THEN
                     IF (.NOT.FOUND(I)) THEN
                        YFND = .TRUE.
                        FOUND(I) = .TRUE.

*                       if individual scaling was choosen reset
*                       variables

                        IF (ITASK.EQ.4. OR. ITASK.EQ.6) THEN
                           VMAX = AVMAX(I)
                           VMIN = AVMIN(I)
                        END IF

*                       plot only if there is a range
                        IF (VMAX.NE.VMIN) THEN
                           IY = 1+NINT(WIDTH*(LV-VMIN)/(VMAX-VMIN))
                           CHR = RECORD(IY:IY)
                           IF (CHR.EQ.' '.OR.
     &                         CHR.EQ.'I'.OR.
     &                         CHR.EQ.'-') THEN
                              RECORD(IY:IY) = MARK(I)
                           ELSE
                              RECORD(IY:IY) = '*'
                           END IF
                        END IF
                     ELSE
                        I2 = ILEN (LN)
                        I3 = ILEN (LXN)
                        WRITE (*,'(4A)')
     &                    ' WARNING from OUTPLT: variable ',LN(1:I2),
     &                    ' occurs twice at same value of ',LXN(1:I3)
                        WRITE (ILU2,'(4A)')
     &                    ' WARNING from OUTPLT: variable ',LN(1:I2),
     &                    ' occurs twice at same value of ',LXN(1:I3)
                     END IF
                  END IF
               END IF
100      CONTINUE

*        plot last line

         WRITE (LXV,'(G12.5)') LVO
         IF (YFND) WRITE (ILU2,'(2(1X,A))') LXV, RECORD(1:IEND)
         WRITE (ILU2,'(/,/,/,A)') ' '
         INIT = .FALSE.
         CLOSE (ILU1)
      ELSE
         CALL ERROR ('OUTPLT','Wrong ITASK')
      END IF

      RETURN
      END

      SUBROUTINE RDAREA (XNAME,X,ILDEC,IFND)

*     Reads an array of REAL values from a data file,
*     that should be initialized with RDINIT.

*     XNAME  - Name of array, for which data are on file          I
*     X      - Array itself                                       O
*     ILDEC  - Declared length of X                               I
*     IFND   - Number of values found on file                     O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA,  ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN, RDDATA, RDINDX,
*                           UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     Example: The data file below contains values for
*     the variables A, B and ZTB:
*
*         * example
*         A = 3.4   ;   B = 5
*         * the following variable is an array
*         ZTB = 1.0, 3.4,
*               1.2, 4.8,
*               1.4, 5.9
*
*     With this routine one may read the array ZTB by:
*     CALL RDAREA ('ZTB',ZTB,100,ILZ).
*     The actual array length ILZ will get the value 6.
*     In the header of RDINDX a formal description of
*     the data file syntax can be found.
*     The lines of the data file are read until column 80.


*     formal parameters
      INTEGER ILDEC,IFND
      REAL X
      DIMENSION X(ILDEC)
      CHARACTER*(*) XNAME

**    local variables
      INTEGER IS
      SAVE


*     now RDDATA will set the number of values
      IFND = 0
```

```
*     read data
      CALL RDDATA (4,0,0,' ',IS,XNAME,X,ILDEC,IFND)

      RETURN
      END

      SUBROUTINE RDDATA (ITASK,IUNIT,IULOG,FILNAM,IS,XNAME,X,NDEC,NREQ)

*     This is the central subroutine of the complete set of RD*
*     routines in the library TTUTIL. The simple routines RDSINT
*     RDSREA, RDAREA, RDSETS and RDFROM form user interfaces. The
*     actual work is done here. That implies that the headers of
*     all user interfaces also apply to this routine RDDATA. The
*     used ITASK values are:
*       ITASK = 1  <----  RDSETS
*       ITASK = 2  <----  RDFROM
*       ITASK = 3  <----  RDINIT
*       ITASK = 4  <----  RDSINT, RDSREA and RDAREA
*     Rerun and data files both are analysed by routine RDINDX
*     called for ITASK is 1 and 3.
*     Examples of data files can be found in the headers of
*     RDSETS, RDSINT, RDSREA and RDAREA. In the header of RDINDX
*     a formal description of the data file syntax can be found.

*     ITASK  - function control, see the header above                I
*     IUNIT  - Unit number used for the .TMP direct access file      I
*              in which values are written (ITASK=1 or 3).
*              IUNIT+1 is used for the data file itself, which is
*              closed after reading it.
*              Different unit numbers should be used in the calls to
*              RDSETS and RDINIT.
*     IULOG  - >0, Unit number of logfile used for data file syntax  I
*                  errors. When not opened RDINDX.LOG is created.
*              =0, Nothing is done with a logfile.
*              The unit number used with ITASK=1 (RDSETS call)
*              is stored and also used for a report about replaced
*              data file values during calls to RDSINT, RDSREA
*              and RDAREA.
*     FILNAM - Name of file (ITASK=1 or 3)                           I
*     IS     - For ITASK = 1: Number of sets, output                I/O
*              For ITASK = 2: Requested set number, input
*     XNAME  - Name of variable, for which data are on file          I
*     X      - Variable itself, may be array                         O
*     NDEC   - Declared length of X, when not array use 1            I
*     NREQ   - For ITASK=2:                                          I/O
*                =0, non-used variables previous set --> logfile
*                >0, non-used variables --> fatal error
*              For ITASK=4:
*                =0, The number of values on file is returned
*                >0, Requested number of values, should match
*                    the number present on file (checked !!)
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA, RDINDX,  ERROR,
*                            EXTENS,  FOPEN, IFINDC,   ILEN, UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     ======= word size in bytes belonging to unformatted record =======
*     = on VAX this is 4 bytes/word; on ATARI, IBM and MAC 1 byte/word =
      INTEGER IWLEN
      PARAMETER (IWLEN=1)

*     formal parameters
      INTEGER ITASK,IUNIT,IULOG,IS,NDEC,NREQ
      REAL X
      DIMENSION X(NDEC)
      CHARACTER*(*) FILNAM,XNAME

**    local variables rerun file
      INTEGER IUNREP,ILNREP,IPTREP,ILPREP,INFREP,INSETS
      INTEGER ISLOC,INR,IPP,IRREP,IRCHK,IULR,ILBUF,ILR
      REAL RBUF
      PARAMETER (ILBUF=32)
      DIMENSION RBUF(ILBUF)
      CHARACTER REPLIS*6,REPFIL*30
      PARAMETER (ILNREP=40,ILPREP=400)
      DIMENSION REPLIS(ILNREP),IPTREP(0:ILPREP),IRCHK(ILNREP)
      LOGICAL LOGR,REPL

*     local variables data file
      INTEGER IUNDAT,ILNDAT,IPTDAT,INFDAT,IDUM,IND,IRDAT,IULD,ILD
      REAL DBUF
      DIMENSION DBUF(ILBUF)
      CHARACTER DATLIS*6,DATFIL*30
      PARAMETER (ILNDAT=200)
      DIMENSION DATLIS(ILNDAT),IPTDAT(0:ILNDAT)
      LOGICAL LOGD,INIT

*     other local variables
      INTEGER I,J,I1,IX,N,ILX,IMES,IPOS,IREC
      CHARACTER MESSAG*40,AANTAL*6,LXNAME*6
      DIMENSION MESSAG(8)
      LOGICAL ALL,OPEND,OVERWR,EMPTY,THERE,FATAL,SINGLE

*     used functions
      INTEGER ILEN, IFINDC

      SAVE
      DATA ISLOC/0/, INIT/.FALSE./, REPL/.FALSE./, THERE/.FALSE./
      DATA MESSAG /
     $ 'Variable name too long                  ',
     $ 'Variable name not in data file          ',
     $ 'On data file, this variable is an array ',
     $ 'Array on data file has a different size ',
     $ 'Array on data file too large to fit     ',
     $ 'On rerun file, this variable is an array',
     $ 'Array on rerun file has a different size',
     $ 'Array on rerun file too large to fit    '/


      IF (ITASK.EQ.1) THEN
*        read rerun file
         IF (REPL) THEN
*           there may be a previous TMP file open
            INQUIRE (UNIT=IUNIT, OPENED=OPEND)
            IF (OPEND) CLOSE (IUNIT)
         END IF

         I = ILEN (FILNAM)
         EMPTY = I.EQ.0
         IF (.NOT.EMPTY) INQUIRE (FILE=FILNAM, EXIST=THERE)

         IF (EMPTY .OR. .NOT.THERE) THEN
*           disactivate replacements
            REPL = .FALSE.
            IS   = 0
            RETURN
         END IF

*        set (new) local unit number + local logfile unit
         IUNREP = IUNIT
         LOGR   = IULOG.GT.0
         IULR   = IULOG

         CALL RDINDX (IUNREP,.TRUE.,.TRUE.,LOGR,IULR,FILNAM,RBUF,
     $        ILBUF,IWLEN,REPLIS,ILNREP,IPTREP,ILPREP,INFREP,INSETS)

*        reset record number, set REPL flag and copy filename
         IRREP  = 0
         REPL   = .TRUE.
         REPFIL = FILNAM
         ILR    = ILEN (REPFIL)
         CALL UPPERC (REPFIL)

*        return number of sets
         IS    = INSETS

         IF (INSETS.GT.0) THEN
*           report to logfile
            IF (LOGR) THEN
               WRITE (IULR,'(/,1X,3A,I3,A,I3,A,2(/,1X,A))')
```

```
     $          'File ',REPFIL(1:ILR),' contains',INSETS,' set(s) of',
     $           INFREP,' variable(s):',
     $          'name     number of values in each set',
     $          '----     ----------------------------'
                DO 10 I=1,INFREP
                   WRITE (IULR,'(1X,A,3X,40(10I5,/,10X))') REPLIS(I),
     $             (IPTREP((J-1)*INFREP+I) - IPTREP((J-1)*INFREP+I-1),
     $             J=1,INSETS)
10              CONTINUE
                WRITE (IULR,'(1X)')
             END IF

*            reset replace counters
             DO 20 I=1,INFREP
                IRCHK(I) = 0
20           CONTINUE
          END IF


       ELSE IF (ITASK.EQ.2) THEN
*        increase set number
          IF (IS.EQ.ISLOC) RETURN
          IF (.NOT.REPL) CALL ERROR ('RDDATA','no rerun file read')
          FATAL = NREQ.GT.0

          IF (ISLOC.GE.1) THEN
*            report to logfile
             IF (LOGR) WRITE (IULR,'(/,1X,A,I3,A,I3,3A)')
     $        '- Moving from set',ISLOC,' to set',IS,' of ',
     $          REPFIL(1:ILR),':'

*            check variable use of previous set
             ALL = .TRUE.
             DO 30 I=1,INFREP
                IF (IRCHK(I).EQ.0) THEN
                   IF (LOGR) WRITE (IULR,'(3X,A,I3,2A)')
     $              'WARNING: NOT USED from set',ISLOC,': ',REPLIS(I)

                   IF (FATAL) THEN
                      WRITE (*,'(/,1X,A,I3,A,I3,3A,/,1X,3A,I3,A)')
     $                 '- Moving from set',ISLOC,' to set',IS,' of ',
     $                 REPFIL(1:ILR),':',' Variable ',
     $                 REPLIS(I),' of set',ISLOC,' was NOT USED'
                      CALL ERROR ('RDDATA','variable not used')
                   END IF
                   ALL = .FALSE.
                END IF
                IRCHK(I) = 0
30           CONTINUE

             IF (ALL.AND.LOGR) WRITE (IULR,'(3X,A,I3,A)')
     $        '(From set',ISLOC,' all variables were used)'
          END IF

*         change set
          IF (IS.GE.0 .AND. IS.LE.INSETS) THEN
             ISLOC = IS
          ELSE
             CALL ERROR ('RDDATA','illegal set number requested')
          END IF


       ELSE IF (ITASK.EQ.3) THEN
*         initialize data file
          IF (INIT) THEN
*            there may be a previous TMP file open
             INQUIRE (UNIT=IUNIT, OPENED=OPEND)
             IF (OPEND) CLOSE (IUNIT)
          END IF

*         set (new) local unit number + local logfile unit
          IUNDAT = IUNIT
          LOGD   = IULOG.GT.0
          IULD   = IULOG

          CALL RDINDX (IUNDAT,.FALSE.,.TRUE.,LOGD,IULD,FILNAM,DBUF,
     $         ILBUF,IWLEN,DATLIS,ILNDAT,IPTDAT,ILNDAT,INFDAT,IDUM)
*         rest record number, set INIT flag and copy filename
          IRDAT  = 0
          INIT   = .TRUE.
          DATFIL = FILNAM
          ILD    = ILEN (DATFIL)
          CALL UPPERC (DATFIL)


       ELSE IF (ITASK.EQ.4) THEN
*         get data ; error checks
          IF (.NOT.INIT) CALL ERROR ('RDDATA','no data file read')

*         initialize reading and get length of name
          SINGLE = NDEC.EQ.1 .AND. NREQ.EQ.1
          OVERWR = .FALSE.
          IMES   = 0
          ILX    = ILEN (XNAME)

*         get local copy in uppercase and check data file index
          LXNAME = XNAME
          CALL UPPERC (LXNAME)
          IND = IFINDC (DATLIS,ILNDAT,1,INFDAT,LXNAME)

          IF (ILX.GT.6) THEN
*            requested name too long
             IMES = 1
          ELSE IF (IND.EQ.0) THEN
*            name does not occur in data file
             IMES = 2
          ELSE
*            get pointer to first value and number of values N
             I1    = IPTDAT(IND-1) + 1
             N     = IPTDAT(IND) - I1 + 1

*            error checks
             IF (SINGLE .AND. N.GT.1) THEN
                IMES = 3
             ELSE IF (NREQ.GT.0 .AND. N.NE.NREQ) THEN
                IMES = 4
             ELSE IF (N.GT.NDEC) THEN
                IMES = 5
             END IF

             IF (REPL .AND. ISLOC.GT.0) THEN
*               check contents of rerun file ; overwrite ?
                INR    = IFINDC (REPLIS,ILNREP,1,INFREP,LXNAME)
                OVERWR = INR.GT.0
             END IF

             IF (IMES.EQ.0 .AND. OVERWR) THEN
*               get pointer to first value and number of values N
                IPP   = (ISLOC-1) * INFREP + INR
                I1    = IPTREP(IPP-1) + 1
                N     = IPTREP(IPP) - I1 + 1

*               error checks
                IF (SINGLE .AND. N.GT.1) THEN
                   IMES = 6
                ELSE IF (NREQ.GT.0 .AND. N.NE.NREQ) THEN
                   IMES = 7
                ELSE IF (N.GT.NDEC) THEN
                   IMES = 8
                END IF
             END IF
          END IF

          IF (IMES.EQ.0) THEN
             IF (.NOT.OVERWR) THEN
*               read data from normal data file
                DO 40 IX=1,N
*                  get record and position IX-th value
                   I = I1 + IX - 1
                   IREC = 1 + (I-1)/ILBUF
                   IPOS = 1 + MOD (I-1,ILBUF)

                   IF (IREC.NE.IRDAT) THEN
```

```
*                  read new record from file
                   READ (IUNDAT,REC=IREC) DBUF
                   IRDAT = IREC
                END IF

*               get requested value
                X(IX) = DBUF(IPOS)
40           CONTINUE

          ELSE
*            read data from rerun file
             DO 50 IX=1,N
*               get record and position IX-th value
                I = I1 + IX - 1
                IREC = 1 + (I-1)/ILBUF
                IPOS = 1 + MOD (I-1,ILBUF)

                IF (IREC.NE.IRREP) THEN
*                  read new record from file
                   READ (IUNREP,REC=IREC) RBUF
                   IRREP = IREC
                END IF

*               get requested value
                X(IX) = RBUF(IPOS)
50           CONTINUE

*            count overwriting ; message
             IRCHK(INR) = IRCHK(INR) + 1
             IF (LOGR) WRITE (IULR,'(1X,7A,I3)')
     $        '- Value(s) of ',LXNAME,' in ',DATFIL(1:ILD),
     $        ' replaced, using ',REPFIL(1:ILR),' set',ISLOC
          END IF

*         return number of values on file when requested to do so
          IF (NREQ.EQ.0) NREQ = N

       ELSE
*         error message to screen
          IF (SINGLE) THEN
*            a single variable was requested
             WRITE (*,'(/,1X,5A)')
     $        'Reading ',XNAME,' from file ',DATFIL(1:ILD),' :'
          ELSE
*            array values were requested
             IF (NREQ.GT.0) THEN
*               number of values was specified
                WRITE (AANTAL,'(I6)') NREQ
             ELSE
*               requested number unknown (0)
                AANTAL = ' ?????'
             END IF
             WRITE (*,'(/,1X,4A,/,1X,3A,I5,A)')
     $        'Reading',AANTAL,' numbers from file ',
     $        DATFIL(1:ILD),'into array ',XNAME,
     $        ' (declared length',NDEC,') :'
          END IF
          IF (OVERWR) THEN
             WRITE (*,'(3X,A,/,3X,A,I3,2A,/,3X,A)')
     $        'Routine was instructed to replace data file contents ',
     $        'by number(s) in set',ISLOC,' of rerun file ',
     $        REPFIL(1:ILR),
     $        '-- BOTH files should contain valid data --'
          END IF
          CALL ERROR ('RDDATA',MESSAG(IMES))
       END IF

    ELSE
       CALL ERROR ('RDDATA','illegal ITASK value')
    END IF

    RETURN
    END

    SUBROUTINE RDFROM (IS,FATAL)

*   Instructs RDDATA (and RDSINT, RDSREA and RDAREA) to use the
*   IS-th set from the rerun file. Note that set 0 (zero) means that
*   data file values are used. Selecting set 0 does not require
*   a previous call to RDSETS and set 0 may also be selected when
*   no rerun file exists or when it is empty.
*   Warnings are generated on non-used variables of the previous
*   set. If desired this may result in a fatal error (see FATAL).
*   Moving from set 0 to another set, no check is carried out.
*   The header of RDSETS contains further remarks.

*   IS    - =0, data file contents returned, disable replacement  I
*           >0, set number enabled
*   FATAL - .FALSE., non-used variables --> logfile               I
*           .TRUE., non-used variables --> fatal error
*
*   Subroutines and/or functions called:
*   - from library TTUTIL: DECCHK, DECINT, DECREA,  ERROR, EXTENS,
*                          FOPEN, IFINDC,   ILEN, RDDATA, RDINDX,
*                          UPPERC

*   Author: Kees Rappoldt
*   Date  : December 1989


*   formal parameters
    INTEGER IS
    LOGICAL FATAL

**  local variables
    INTEGER IFATAL
    REAL DUMMY(1)
    SAVE

    IF (FATAL) THEN
       IFATAL = 1
    ELSE
       IFATAL = 0
    END IF

*   call for next set
    CALL RDDATA (2,0,0,' ',IS,' ',DUMMY,1,IFATAL)

    RETURN
    END

     SUBROUTINE RDINDX (IUNIT,SETS,TOSCR,TOLOG,IUL,DATFIL,XBUF,ILBUF,
    $                   IWLEN,NAMLIS,ILIND,INDPNT,ILPNT,INFND,INSETS)

*   Produces an index of a data file. The index consists of a list of
*   variable names and an integer array pointing to decoded values
*   on a direct access file. The direct access file is opened
*   for reading at the end of the routine or deleted after errors.

*   IUNIT  - unit number used to open random access file for I/O   I
*            IUNIT+1 used to open data file (closed after reading).
*   SETS   - when .TRUE. more than a single set of values is       I
*            is allowed. The order of the names in all sets
*            should be identical.
*   TOSCR  - flag enabling error message output to screen          I
*   TOLOG  - flag enabling error message output to logfile         I
*   IUL    - unit number of logfile (when TOLOG is set)            I
*            if non-existent, RDINDX.LOG is created
*   DATFIL - name of data file                                     I
*   XBUF   - record buffer direct access file, overwritten        I/O
*   ILBUF  - array length XBUF, number of values on record         I
*   IWLEN  - word length in bytes for unformatted record           I
*   NAMLIS - list of variable names                                O
*   ILIND  - declared size of array NAMLIS                         I
*   INDPNT - points to value in temporary file                     O
*   ILPNT  - declared size of array INDPNT                         I
*   INFND  - number of variable names found                        O
*   INSETS - Number of sets in data file (0 when empty)            O
*
*   Subroutines and/or functions called:
*   - from library TTUTIL: DECCHK, DECINT, DECREA,  ERROR, EXTENS,
*                          FOPEN, IFINDC,   ILEN,  UPPERC

*   Author: Kees Rappoldt
```

```
*     Date  : December 1989


*     The data should be present on file in the form "name = value(s)".
*     Such a string is called a field in the formal description below.
*     On each record (line) of a data file different fields may occur,
*     separated by a semi-colon.
*     A field may be continued on a following line. A name or an
*     individual number, however, should not be interrupted. Continuing
*     a series of numbers, the comma is expected before going to the
*     next line. On all lines comments may be present following an
*     exclamation mark (!). Lines beginning with an asterisk (*) are
*     ignored. Maximum name length is set by the variable type of NAM-
LIS.
*
*          record =   field {; field}
*           field =   name = number {, number}
*            name =   letter{letdig}
*          letdig =   < letter | digit >
*          number =   [repeat*]real
*          repeat =   INTEGER number
*            real =   REAL number (may be without decimal point)
*
*     When SETS is .TRUE. the data file may consist of more than a sin-
gle
*     set of variable names and values. The order of the names in all
*     sets should be identical. In this case the pointer array INDPNT
*     contains INFND pointers for each set.
*     Examples of data files can be found in the headers of RDAREA,
RDSREA,
*     RDSINT and RDSETS.


*     Parser status IS jumptable
*     --------------------------
*       Example field:   NAME   =   4.0   ,    120*105.8    ;
*               status: 1.2...3..4...5..6..4....5..75....6...1..
*
*     Status 0 is the error status. The characters 1,2,3 and 5 (see
*     table) return from error status and the analysis is continued.
*     For the errors 9 to 14 the analysis is not interrupted.
*
*                    STATUS: 0/-  1   2   3   4   5   6   7   8
*     CHARACTER:
*     0 - space              -   1   3   3   4   6   6   -   -
*     1 - semi-colon ;       1   -   -   -   -   1   1   -   -
*     2 - equality sign =    4   -   4   4   -   -   -   -   -
*     3 - comma ,            4   -   -   -   -   4   4   -   -
*     4 - point and + -      -   -   -   -   5   5   -   5   -
*     5 - *                  7   -   -   -   -   7   -   -   -
*     6 - E (not e)          -   2   2   -   -   5   -   -   -
*     7 - rest alphabetic    -   2   2   -   -   -   -   -   -
*     8 - numerical          -   -   2   -   5   5   -   5   -
*     9 - other              -   -   -   -   -   -   -   -   -
*    10 - comment "!"        -   1   3   3   4   1   1   -   -
*    11 - char(26) "E_O_F"   8   1   8   8   8   8   8   8   -


*     formal parameters
      INTEGER IUNIT,IUL,ILBUF,IWLEN,INDPNT,ILIND,ILPNT,INFND,INSETS
      REAL XBUF
      CHARACTER*(*) DATFIL,NAMLIS
      DIMENSION NAMLIS(ILIND),INDPNT(0:ILPNT),XBUF(ILBUF)
      LOGICAL SETS,TOSCR,TOLOG

**    local variables
      INTEGER I,I1,I2,IC,IS,IT,IBN,IOS,INL,IW1,IW2,IERR,IMES,INN,IPOS
      INTEGER IREC,IREP,IVAL,IVOLD,IWAR,IUDF,ILINE,ILTMP,IRECL,ISOLD
      INTEGER ITYP(0:122), JUMPTB(0:8,0:11)
      REAL XVAL
      CHARACTER RECORD*80, HLPREC*81, HD*60
      CHARACTER NBUF*32, MESSAG*28, FORM*23, ERBUF*38, TMPFIL*40
      DIMENSION MESSAG(14),HD(2)
      LOGICAL EOF,COMM,COMPL,RDNAME,THERE

*     functions called
      INTEGER ILEN, IFINDC

      SAVE
      DATA ITYP / 26*9,11,5*9,
     $  0,10,9,9, 9,9,9,9, 9,9,5,4, 3,4,4,9, 8,8,8,8, 8,8,8,8,
     $  8,8,9,1, 9,2,9,9, 9,7,7,7, 7,6,7,7, 7,7,7,7, 7,7,7,7,
     $  7,7,7,7, 7,7,7,7, 7,7,7,9, 9,9,9,9, 9,7,7,7, 7,7,7,7,
     $  7,7,7,7, 7,7,7,7, 7,7,7,7, 7,7,7,7, 7,7,7/
      DATA JUMPTB /
     $  0,1,3,3,4,6,6,0,0, 1,0,0,0,0,1,1,0,0, 4,0,4,4,0,0,0,0,0,
     $  4,0,0,0,0,4,4,0,0, 0,0,0,0,5,5,0,5,0, 7,0,0,0,0,7,0,0,0,
     $  0,2,2,0,0,5,0,0,0, 0,2,2,0,0,0,0,0,0, 0,0,2,0,5,5,0,5,0,
     $  0,0,0,0,0,0,0,0,0, 0,1,3,3,4,1,1,0,0, 8,1,8,8,8,8,8,8,8/
      DATA MESSAG /
     $ 'Variable name expected      ','Illegal character in name   ',
     $ 'Equality sign "=" expected  ','Number expected             ',
     $ 'Illegal character in number ','Comma or semi-colon expected',
     $ 'Number should follow "*"    ','Unexpected End_Of_File      ',
     $ 'Wrong name order in set     ','Variable name already used  ',
     $ 'Unknown name in set         ','Variable name too long      ',
     $ 'Before "*" positive INTEGER ','This is not a number       '/


*     maximum variable name length
      INL = LEN (NAMLIS(1))
      IF (INL.GT.32) CALL ERROR ('RDINDX','too long names')

*     open temporary direct access file ; record length
      IRECL = 4 * ILBUF / IWLEN
      CALL EXTENS (DATFIL,'TMP',1,TMPFIL)
      ILTMP = ILEN (TMPFIL)
      INQUIRE (FILE=TMPFIL(1:ILTMP),EXIST=THERE)
      IF (THERE) THEN
*        delete old file
         OPEN (IUNIT,FILE=TMPFIL(1:ILTMP),ACCESS='DIRECT',
     $     RECL=IRECL,STATUS='OLD')
         CLOSE (IUNIT,STATUS='DELETE')
      END IF
      OPEN (IUNIT,FILE=TMPFIL(1:ILTMP),ACCESS='DIRECT',
     $  RECL=IRECL,STATUS='NEW')

*     open data file
      IUDF = IUNIT + 1
      CALL FOPEN (IUDF,DATFIL,'OLD','NVT')

*     error list header
      HD(1) = 'ERROR(s) reading data from '//DATFIL
      HD(2) = 'LINE   ERROR                                somewhere here'

*     set zero-th element pointer ; initialize parser
      INDPNT(0) = 0
      INFND  = 0
      INSETS = 0
      INN    = 0
      ILINE  = 0
      IS     = 1
      IW1    = 0
      IREP   = 1
      IERR   = 0
      IBN    = 0
      IREC   = 0
      IOS    = 0
      COMPL  = .TRUE.
      EOF    = .FALSE.


10    IF (.NOT.EOF) THEN
*        read next line ; neglect contents of E_O_F record
         READ (IUDF,'(A)',IOSTAT=IOS) RECORD
         ILINE = ILINE + 1
         EOF   = IOS.LT.0
         IF (EOF) RECORD = CHAR(26)

*        terminate copy with comment mark
         I      = ILEN (RECORD) + 1
         HLPREC = RECORD
         HLPREC(I:I) = '!'

*        analyse line ; skip comment lines
```

```
      COMM  = HLPREC(1:1).EQ.'*'
      IPOS  = 0
20    IF (.NOT.COMM) THEN
         IPOS  = IPOS + 1
*        save current status (as message number) ; get new
         ISOLD = IS
         IMES  = ISOLD
         IC    = ICHAR (HLPREC(IPOS:IPOS))
         IF (IC.LE.122) THEN
*           use type aray
            IT = ITYP(IC)
         ELSE
*           character type 9 (including extended ASCII)
            IT = 9
         END IF
         COMM  = IT.EQ.10
         IS    = JUMPTB (ISOLD,IT)

*        (last) set should be complete !
         IF (IT.EQ.11 .AND. .NOT.COMPL) IS = 8

         IF (IS.EQ.0) THEN
*           delete word buffer and repeat counter ; message
            IW1  = 0
            IREP = 1

         ELSE IF ((IS.EQ.2.AND.ISOLD.NE.2).OR.
     $            (IS.EQ.5.AND.ISOLD.NE.5)) THEN
*           a word starts
            RDNAME = IS.EQ.2
            IW1    = IPOS
            IW2    = IPOS

         ELSE IF (IS.EQ.2.OR.IS.EQ.5) THEN
*           a word continues
            IW2 = IPOS

         ELSE IF (IW1.GT.0) THEN
*           a word has been read
            IF (RDNAME) THEN
*              it is a variable name
               IF (IW2-IW1+1.LE.INL) THEN
*                 valid name ; previously used ?
                  NBUF = RECORD (IW1:IW2)
                  CALL UPPERC (NBUF)
                  I = IFINDC (NAMLIS,ILIND,1,INFND,NBUF(1:INL))

                  IF (I.EQ.0 .AND. INSETS.LE.1) THEN
*                    accept new name in index
                     INFND = INFND + 1
                     IF (INSETS.EQ.0) INSETS = 1
                     IF (INFND.GT.ILIND.OR.INFND.GT.ILPNT) CALL ERROR
     $                ('RDDATA','too many different names on file')
                     NAMLIS(INFND) = NBUF(1:INL)
                     IVOLD = INFND
                  ELSE IF (I.GT.0 .AND. SETS) THEN
*                    variable name was previously found
                     IF (I.EQ.1 .AND. COMPL) THEN
*                       start of new set
                        INSETS = INSETS + 1
                     ELSE IF (I.NE.IVOLD+1) THEN
*                       error in variable name order
                        IMES = 9
                     END IF
                     COMPL = I.EQ.INFND
                     IVOLD = I
                  ELSE IF (I.GT.0) THEN
*                    old name illegal (no sets)
                     IMES = 10
                  ELSE
*                    new name illegal (in set)
                     IMES = 11
                  END IF

*                 increase name counter ; initialize value pointer
                  INN = INN + 1
                  IF (INN.GT.ILPNT) CALL ERROR ('RDDATA',
     $                'too many parameter sets on file')
                  INDPNT(INN) = INDPNT(INN-1)
               ELSE
                  IMES = 12
               END IF


            ELSE IF (IS.EQ.7) THEN
*              it must be a repeat value
               CALL DECINT (IWAR,RECORD(IW1:IW2),IVAL)
               IF (IWAR.EQ.0 .AND. IVAL.GT.0) THEN
*                 accept it
                  IREP = IVAL
               ELSE
                  IMES = 13
               END IF

            ELSE
*              it must be a real number
               CALL DECREA (IWAR,RECORD(IW1:IW2),XVAL)
               IF (IWAR.EQ.0) THEN
*                 accept it IREP times
                  DO 30 I=1,IREP
                     IF  (IBN.EQ.ILBUF) THEN
*                       buffer if full ; empty buffer
                        IBN  = 0
                        IREC = IREC + 1
                        WRITE (IUNIT,REC=IREC) XBUF
                     END IF
*                    write value in buffer
                     IBN = IBN + 1
                     XBUF(IBN) = XVAL
30                CONTINUE

*                 update total number of values and pointer
                  INDPNT(INN) = INDPNT(INN) + IREP
*                 reset repeat counter
                  IREP = 1
               ELSE
                  IMES = 14
               END IF
            END IF

*           empty word buffer
            IW1 = 0
         END IF

         IF ((ISOLD.NE.0.AND.IS.EQ.0) .OR. IMES.GE.9) THEN
*           compose and write error message (+header HD)
            IERR = IERR + 1
            IF (IERR.EQ.1) THEN
               IF (TOSCR) WRITE  (*,'(9X,A,/,2X,A,/)')  HD
               IF (TOLOG) THEN
*                 logfile present ?
                  INQUIRE (UNIT=IUL,OPENED=THERE)
                  IF (.NOT.THERE) CALL FOPEN
     $               (IUL,'RDINDX.LOG','NEW','DEL')
                  WRITE (IUL,'(9X,A,/,2X,A,/)') HD
               END IF
            END IF
            I1 = MAX (IPOS-25,1)
            I2 = MIN (I1+37,80)
            I  = IPOS -I1 + 40
            ERBUF = RECORD(I1:I2)
            IF (IMES.EQ.8) ERBUF = ' '
            WRITE (FORM,'(A,I2,A)') '(1X,I5,2(3X,A),/,',I,'X,A)'
            IF (TOSCR) WRITE  (*,FORM)  ILINE,MESSAG(IMES),ERBUF,'^'
            IF (TOLOG) WRITE (IUL,FORM) ILINE,MESSAG(IMES),ERBUF,'^'
         END IF

      GOTO 20
      END IF
   GOTO 10
   END IF

*  close data file
   CLOSE (IUDF)
```

```
      IF (IERR.EQ.0) THEN
*        flush buffer ; close .TMP file and re-open
         WRITE (IUNIT,REC=IREC+1) XBUF
         CLOSE (IUNIT)
         OPEN (IUNIT,FILE=TMPFIL(1:ILTMP),ACCESS='DIRECT',
     $    RECL=IRECL,STATUS='OLD')
      ELSE
*        delete direct access file
         CLOSE (IUNIT,STATUS='DELETE')
         WRITE (*,'(/,1X,I4,1X,A)') IERR,HD(1)
         CALL ERROR ('RDINDX','temporary file deleted')
      END IF

      RETURN
      END

      SUBROUTINE RDINIT (IUNIT,IULOG,DATFIL)

*     Initializes data file reading with the routines RDSINT,
*     RDSREA and RDAREA. An index of the data file is stored
*     as a local array containing variable names. The values
*     are written to a temporary file. After a call to RDINIT
*     the data file itself is closed again.
*     In the header of RDINDX a formal description of the
*     data file syntax can be found. For examples, see the
*     headers of RDSINT, RDSREA and RDAREA.

*     IUNIT  - unit number used to open random access file for I/O   I
*              IUNIT+1 used to open data file (closed after reading).
*     IULOG  - >0, Unit number of logfile used for data file syntax  I
*                  errors. When not opened RDINDX.LOG is created.
*              =0, Nothing is done with a logfile
*     DATFIL - Name of data file                                     I
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA,   ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN,  RDDATA, RDINDX,
*                           UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     formal parameters
      INTEGER IUNIT,IULOG
      CHARACTER*(*) DATFIL

**    local variables
      INTEGER IL,IS
      REAL DUMMY(1)
      SAVE

*     initialize subroutine RDDATA
      IL = 0
      CALL RDDATA (3,IUNIT,IULOG,DATFIL,IS,' ',DUMMY,1,IL)

      RETURN
      END

      SUBROUTINE RDSETS (IUNIT,IULOG,SETFIL,INS)

*     Initializes subroutine RDDATA for reading data from a so called
*     "rerun file" containing sets of variable names with associated
*     values. The sets are used to replace corresponding data items in
*     a normal data file analysed with the routines RDINIT, RDSREA,
*     RDSINT and RDAREA.
*     This facility has a "global" character. A call to RDINIT does not
*     disable the replacement of values. So the sets in a
*     rerun file may contain variable names occurring in different
*     data files. Only a call to RDSETS with empty filename will
*     disactivate the replacement of numbers.
*     After a normal RDSETS call the 0-th set is activated meaning that
*     data file values are used. With RDFROM the sets of the rerun
*     file are actually activated.

*     IUNIT  - unit number used to open random access file for I/O   I
*              IUNIT+1 used to open rerun file (closed after reading).
*     IULOG  - >0, Unit number of logfile used for syntax error      I
*                  messages. When not opened RDINDX.LOG is created.
*              =0, Nothing is done with a logfile
*     SETFIL - Name of rerun file containing sets                    I
*              Empty --> number replacement is disactivated (INS=0)
*     INS    - Number of sets on file (when exists minimum 1)        O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA,   ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN,  RDDATA, RDINDX,
*                           UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     Example:
*     The following rerun file contains two sets of values for the
*     variables PAR1, PAR2 and ZTB:
*
*        * set 1
*        PAR1 = 3.4   ;   PAR2 = 5
*        ZTB = 1.0, 3.4,  1.2, 4.8,  1.4, 5.9
*        *
*        * set 2
*        PAR1 = 3.7   ;   PAR2 = 7
*        ZTB = 1.0, 3.1,  1.4, 9.8
*
*     After calling RDSETS, set 2 is activated by:
*     CALL RDFROM (2,FATAL). Reading then variable PAR1 from any
*     data file (just using RDINIT and RDSREA), the value from
*     the data file will be replaced by 3.7.
*     The lines of the data file are read until column 80.
*
*     The following statements form a typical application. The
*     rerun file RERUNS.DAT is assumed to contain sets of values
*     for parameter PAR1 of (sub)model A and parameter PAR2 of
*     (sub)model B. Both models make use of their own data file
*     containing also other variables. The structure below runs
*     the two models, at first for the values of PAR1 and PAR2 on
*     the data files, then for the sets in RERUNS.DAT. When the
*     file RERUNS.DAT is not there, a normal model run is made.
*
*     *     open logfile
*           CALL FOPEN (40,'LOGFILE.DAT','NEW','DEL')
*           CALL RDSETS (20,40,'RERUNS.DAT',INS)
*
*     *     at first the content of the data files is used (set 0)
*           DO 10 IS=0,INS
*              CALL RDFROM (IS,.TRUE.)
*              ........
*     *        statements occurring in model A:
*              CALL RDINIT (22,40,'MODELA.DAT')
*              CALL RDSREA ('PAR1',PAR1)
*              ........
*     *        statements occurring in model B:
*              CALL RDINIT (22,40,'MODELB.DAT')
*              CALL RDSREA ('PAR2',PAR2)
*              ........
*     10    CONTINUE
*
*     Note the following:
*     - The order of variables should be identical in all sets.
*     - When a non-zero logfile unit number is supplied, the usage of
*       values from the rerun file is reported. Also a change of
*       set number with RDFROM is reported and warnings are given
*       on non-used variables (of the previous set).
*     - In the above example the argument FATAL of RDFROM is set
*       to .TRUE. That leads to a fatal error on non-used variables
*       of the previous set (instead of a warning on logfile only)
*     - The (logfile) unit number used for the report is not
*       overwritten by the unit number in a RDINIT call.
*     - Values on the rerun file can only be used in combination
*       with a data file on which error free values occur for the
*       same variable. There is no access to the rerun file without
*       normal data file reading.
```

```
*     - After RDSETS "set 0" is selected meaning that the data file
*       values are not replaced. "Set 0" may also be selected just
*       using CALL RDFROM (0,FATAL).


*     formal parameters
      INTEGER IUNIT,IULOG,INS
      CHARACTER*(*) SETFIL

**    local variables and used functions
      INTEGER IL, I, ILEN
      REAL DUMMY(1)
      SAVE

*     analyse rerun file using subroutine RDDATA
      IL = 0
      CALL RDDATA (1,IUNIT,IULOG,SETFIL,INS,' ',DUMMY,1,IL)

*     messages to screen
      IF (IULOG.GT.0 .AND. INS.GT.0) THEN
         I = ILEN (SETFIL)
         WRITE (*,'(1X,A,/,1X,A,I3,2A)')
     $    'Message from RDSETS: A logfile report is written',
     $    'about the use of the',INS,' parameter sets on ',
     $     SETFIL(1:I)
      ELSE IF (INS.GT.0) THEN
         WRITE (*,'(1X,2A)') 'WARNING from RDSETS: ',
     $    'no logfile is used !!'
      END IF

      RETURN
      END

      SUBROUTINE RDSINT (XNAME,IX)

*     Reads a single INTEGER value from a data file.
*     The reading should be initialized with RDINIT.

*     XNAME  - Name of variable        I
*     IX     - Value of variable       O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA,   ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN,  RDDATA, RDINDX,
*                           UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     Example: The data file below contains values for
*     the variables A, IB and ZTB:
*
*         * example
*         A = 3.4   ;   IB = 5
*         * the following variable is an array
*         ZTB = 1.0, 3.4,
*               1.2, 4.8,
*               1.4, 5.9
*
*     With this routine one may read the value of IB by:
*     CALL RDSINT ('IB',IB).
*     Reading A results in the value 3 (the nearest integer
*     function is used). Reading the array ZTB with this
*     routines results in an error message. In the header of
*     RDINDX a formal description of the data file syntax
*     can be found.
*     The lines of the data file are read until column 80.


*     formal parameters
      INTEGER IX
      CHARACTER*(*) XNAME

**    local variables
      INTEGER IL,IS
      REAL DUMMY(1)
      SAVE

*     read 1 and only 1 number and convert to INTEGER
      IL = 1
      CALL RDDATA (4,0,0,' ',IS,XNAME,DUMMY,1,IL)

      IX = NINT (DUMMY(1))

      RETURN
      END

      SUBROUTINE RDSREA (XNAME,X)

*     Reads a single REAL value from a data file.
*     The reading should be initialized with RDINIT.

*     XNAME  - Name of variable        I
*     X      - Value of variable       O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECCHK, DECINT, DECREA,   ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN,  RDDATA, RDINDX,
*                           UPPERC

*     Author: Kees Rappoldt
*     Date  : December 1989


*     Example: The data file below contains values for
*     the variables A, B and ZTB.
*
*         * example
*         A = 3.4   ;   B = 5
*         * the following variable is an array
*         ZTB = 1.0, 3.4,
*               1.2, 4.8,
*               1.4, 5.9
*
*     With this routine one may read the value of A by:
*     CALL RDSREA ('A',A).
*     Reading the array ZTB with this routines results in an
*     error message. In the header of RDINDX a formal description
*     of the data file syntax can be found.
*     The lines of the data file are read until column 80.


*     formal parameters
      REAL X
      CHARACTER*(*) XNAME

**    local variables
      INTEGER IL,IS
      REAL DUMMY(1)
      SAVE

*     read 1 and only 1 number !!
      IL = 1
      CALL RDDATA (4,0,0,' ',IS,XNAME,DUMMY,1,IL)

      X = DUMMY(1)

      RETURN
      END

      REAL FUNCTION REAAND (X1, X2)

*     This function emulates the CSMP function AND.
*     REAl AND is similar to logical .AND. except that
*     arguments and results are REAL instead of LOGICAL
*     The definition of the function is:
*     REAAND = 1,  X1 > 0 and X2 > 0
*     REAAND = 0,  else

*     REAAND - Function result         O
*     X1     - first argument          I
*     X2     - second argument         I
*
```

```
*     No subroutines and/or functions used

*     Author: Daniel van Kraalingen
*     Date  : December 1989


*     formal parameters
      REAL X1, X2

**    no local variables
      SAVE

      IF (X1.GT.0. .AND. X2.GT.0.) THEN
         REAAND = 1.
      ELSE
         REAAND = 0.
      END IF

      RETURN
      END
      REAL FUNCTION REANOR (X1, X2)

*     This function emulates the CSMP function NOR.
*     REAl NOR is similar to logical expression
*     .NOT.(logical.OR.logical) except that
*     arguments and results are REAL instead of LOGICAL
*     The definition of the function is:
*     REANOR = 1  when  X1 <=0 and X2 <= 0
*     REANOR = 0  otherwise

*     REANOR - Function result          O
*     X1     - first argument           I
*     X2     - second argument          I
*
*     No subroutines and/or functions called

*     Author: Daniel van Kraalingen
*     Date  : November 1989


*     formal parameters
      REAL X1, X2

**    no local variables
      SAVE

      IF (X1.LE.0. .AND. X2.LE.0.) THEN
         REANOR = 1.
      ELSE
         REANOR = 0.
      END IF

      RETURN
      END
      SUBROUTINE TIMER (ITASK, DATEB, DELT, PRDEL, FINTIM,
     &                  IYEAR, TIME, DATE, IDATE, TERMNL, OUTPUT)

*     This subroutine updates TIME and related variables
*     each time it is called.
*     The routine should be initialized first by a call
*     with ITASK=1. The time variables will be made local.
*     Leap years are not (yet) implemented

*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR

*     formal parameters
      INTEGER ITASK, IYEAR, IDATE
      REAL DATEB, DELT, PRDEL, FINTIM, TIME, DATE
      LOGICAL TERMNL, OUTPUT

*     local variables
      REAL LDATEB, LDELT, LPRDEL, LFINTM, LTIME
      REAL R1, TINY
      INTEGER ILYEAR, IYEARB, IOUT, INSTEP, ITOLD

      PARAMETER (TINY=1.E-5)

      SAVE

      DATA ITOLD/4/

      IF (ITASK.EQ.1) THEN

*        initialization

*        check value of DELT, PRDEL, FINTIM and TIME
         IF (DELT.LE.0.)     CALL ERROR ('TIMER','DELT <= 0')
         IF (PRDEL.LE.0.)    CALL ERROR ('TIMER','PRDEL <= 0')
         IF (PRDEL.LT.DELT) CALL ERROR ('TIMER','PRDEL < DELT')
         IF (FINTIM.LT.0.)   CALL ERROR ('TIMER','FINTIM < 0')
         IF (DATEB.LT.1. .OR. DATEB.GT.365.)
     &       CALL ERROR ('TIMER','DATEB < 1 or DATEB > 365')
         IF (IYEAR.LT.1000 .OR. IYEAR.GT.2100)
     &       CALL ERROR ('TIMER', 'IYEAR < 1000 or IYEAR > 2100')

*        check if PRDEL is a multiple of DELT, IOUT is also used to
*        indicate the number of times that TIME must increase before
*        OUTPUT is set to .TRUE.

         IOUT = NINT (PRDEL/DELT)
          R1   = (PRDEL/DELT-FLOAT (IOUT))/FLOAT (IOUT)
         IF (ABS (R1).GT.TINY) THEN
            CALL ERROR ('TIMER','PRDEL not an exact multiple of DELT')
         END IF

         LDATEB = DATEB
         LDELT  = DELT
         LPRDEL = PRDEL
         LFINTM = FINTIM

         LTIME  = 0.
         TIME   = LTIME
         DATE   = 1.+ MOD (LDATEB+LTIME-1., 365.)
         IDATE  = 1 + MOD ((INT (LDATEB+LTIME+TINY)-1), 365)
         ILYEAR = IYEAR
         IYEARB = IYEAR
         TERMNL = .FALSE.
         OUTPUT = .TRUE.
         INSTEP = 0

      ELSE IF (ITASK.EQ.2) THEN

*        dynamic section

         IF (ITOLD.EQ.4) CALL ERROR ('TIMER', 'initialization required')

*        checks are done if time variables are unchanged
         IF (DATEB.NE.LDATEB)
     &       CALL ERROR ('TIMER','DATEB has been changed illegally')
         IF (DELT.NE.LDELT)
     &       CALL ERROR ('TIMER','DELT has been changed illegally')
         IF (PRDEL.NE.LPRDEL)
     &       CALL ERROR ('TIMER','PRDEL has been changed illegally')
         IF (FINTIM.NE.LFINTM)
     &       CALL ERROR ('TIMER','FINTIM has been changed illegally')
         IF (IYEAR.NE.ILYEAR)
     &       CALL ERROR ('TIMER','IYEAR has been changed illegally')
         IF (TIME.NE.LTIME)
     &       CALL ERROR ('TIMER','TIME has been changed illegally')

         INSTEP = INSTEP+1
         OUTPUT = MOD (INSTEP, IOUT).EQ.0

         LTIME  = LTIME+LDELT
         TIME   = LTIME

         IF (LTIME.GE.LFINTM) OUTPUT = .TRUE.
         IF (LTIME.GE.(LFINTM+LDELT)) TERMNL = .TRUE.

         DATE   = 1.+ MOD (LDATEB+LTIME-1., 365.)
         IDATE  = 1 + MOD ((INT (LDATEB+LTIME+TINY)-1), 365)
         ILYEAR = IYEARB+INT ((LDATEB+LTIME-TINY)/365.)
         IYEAR  = ILYEAR
```

```
      ELSE
         CALL ERROR ('TIMER','wrong ITASK')
      END IF

      ITOLD = ITASK

      RETURN
      END
      SUBROUTINE UPPERC (STRING)

*     Converts string to uppercase characters

*     STRING - character string        I
*
*     No subroutines and/or functions called

*     Author: Daniel van Kraalingen, Kees Rappoldt
*     Date  : October 1989


*     formal parameter
      CHARACTER*(*) STRING

**    local variables
      INTEGER I,IC,L
      SAVE

      L = LEN (STRING)
      DO 10 I=1,L
*        convert lowercase letters
         IC = ICHAR(STRING(I:I))
         IF (IC.GE.97.AND.IC.LE.122) STRING(I:I) = CHAR(IC-32)
10    CONTINUE

      RETURN
      END
```

Appendix D: Weather reading program

For a more detailed description of the weather program see van Kraalingen *et al.* (1990).

```
* ------------------------------------------------------------ *
* weather -- get weathr data
* ------------------------------------------------------------ *
* PURPOSE
*       WEATHER makes 6 measurements (irradiation, minus temperature,
*       maximum temperature, early morning vapour pressure, mean
*       windspeed and precipitation) available for planth growth
*       modeling programs and other programs requiring a realistic
*       "weather environment" on a daily basis.
* AUTHOR
*       M Verbeek
* DESIGN
*       M Verbeek, D v Kraalingen, C ten Cate.
* (c)
*       Center Agro Biological Research (CABO)
*       P.O. Box 14
*       Wageningen
*       Netherlands
* VERSION
*       1
* DATE
*       21 september 1989
* ------------------------------------------------------------ *

* ------------------------------------------------------------ *
* stinfo -- (re)set weathr system parameters
* ------------------------------------------------------------ *
* PURPOSE
*
*       set (or reset) default values for location of data
*       files and the name of the log file to IPATH and ILOG.
*       Return information about data: the coordinates and altitude
*       of the weather station and in what whay the irradiation
*       data where obtained.
* PARAMETERS
*       name    type        description
*       ----    ----        -----------------------------------
*       --- in  ---
*       IFLAG   int         output flags
*       IPATH   C*(*)       PATH to data files
*       ILOG    C*(*)       name of logfile
*       ICNT    C*(*)       name of country
*       ISTN    int         code for station
*       IYEAR   int         year of measurments
*        --- out ---
*       LON     real        longitude
*       LAT     real        latitude
*       ALT     real        altitude
*       A       real        first parameter of radiation conversion
*       B       real        second parameter of radiation conversion
* DEFAULTS
*       PATH:   ' '         Default directory
*       LOG:    ' '         Default name = "weather.log"
*       others: -           no defaults
* RETURNS
* ------------------------------------------------------------ *

      SUBROUTINE STINFO (IFLAG,IPATH,ILOG,ICNT,ISTN,IYEAR,
     &                   STAT,LON,LAT,ALT,A,B)

*     IMPLICIT NONE
*     --------- parameters --------
*     --- in ---
      CHARACTER*(*)     IPATH, ILOG, ICNT
      INTEGER           IFLAG, ISTN, IYEAR
*     --- out ---
      INTEGER           STAT
      REAL              LON, LAT, ALT, A, B
*     --------- constants ---------
*               unit var is not used (file closed)
      INTEGER   FSCLSD
      INTEGER   MAXCNT
      CHARACTER DEFLOG*20
      PARAMETER (
     &          DEFLOG = 'weather.log' ,
     &          MAXCNT =  6       ,
     &          FSCLSD = -1       )
*     --------- external ----------
      INTEGER  WSILEN
      LOGICAL  WSDAOP, WSATTR, WSRDDA, WSFLGS


*     --------- COMMON: WSCFIL ---------------------------------- *
*                     Weathr Subsystem Char FILe names
*                     note: MAXFNM depends on system.
      INTEGER    MAXFNM
      PARAMETER (MAXFNM = 256)

      CHARACTER*(MAXFNM) BPATH, FNAME, LOG
      COMMON   /WSCFIL/  BPATH, FNAME, LOG
*     --------- common wscfil end ------------------------------ *


*     --------- COMMON: WSNFIL  --------------------------------- *
*                     Weathr Subsystem Num  FILe flags and logf. unit.
*     WARNFL  - .TRUE. if to create warings (screen and/or file)
*     ERRFL   - .TRUE. if to create error messages  ( " )

*               Number Of Flags
      INTEGER    NOF
      PARAMETER  (NOF = 4)

      LOGICAL    OF(NOF)
*               unit of logfile
      INTEGER    LUNIT
*               error flag from STINFO to WEATHR (12345 = ok)
      INTEGER    STERR
      LOGICAL    WARNFL, ERRFL

      COMMON /WSNFIL/ OF, LUNIT, WARNFL, ERRFL, STERR
*     --------- common wsnfil end ------------------------------ *

*     --------- local variables ----
      INTEGER   FLAG
*               default logfile name
      LOGICAL   FIRST, DUMMY
*               data unit
      INTEGER   DUNIT
*               datafile identification
      CHARACTER BCNT*(MAXCNT)
      CHARACTER NEWLOG*(MAXFNM)
      INTEGER   BSTN, BYEAR
      REAL      BLON, BLAT, BALT, BA, BB

      SAVE
*     --------- data -------------
      DATA FIRST /.TRUE./
*                nonsense value
      DATA FLAG   /-54321/
      DATA BCNT   /' '/
      DATA BSTN   /-1/
      DATA BYEAR  /-1/
      DATA BLON   /-199./
      DATA BLAT   /-99./
      DATA BALT   /-99./
      DATA BA     /-99./
      DATA BB     /-99./

*     ------- start module --------
      IF (FIRST) THEN
          FIRST = .FALSE.
*         --- init of variables in common done here
          BPATH = '<no path>'
          FNAME = '<no file>'
          LUNIT = FSCLSD
          LOG   = DEFLOG
       ENDIF

*     --- magic value in STERR if exit ok: STERR = 12345,
*     ---                          on error:       =  -1
      STERR = -1

*     --- check IFLAG, if changed, reset FLAGS
      IF (IFLAG .NE. FLAG) THEN
```

```
         IF (.NOT. WSFLGS(IFLAG, STAT)) THEN
*            --- set flag to maximum output and continue
             FLAG = 1111
             DUMMY = WSFLGS(FLAG, STAT)
         ELSE
             FLAG = IFLAG
         ENDIF
      ENDIF

*     --- if ILOG = ' ' the logfile has the default name
      IF (WSILEN(ILOG) .EQ. 0) THEN
         NEWLOG = DEFLOG
      ELSE
         NEWLOG = ILOG
      ENDIF

*     --- see if new log file requested; close old logfile
      IF (NEWLOG .NE. LOG) THEN
         IF (LUNIT .NE. FSCLSD) THEN
            CLOSE(LUNIT)
            LUNIT = FSCLSD
         ENDIF
         LOG = NEWLOG
      ENDIF

*     --- return NIL value's on error
      LON = -199.
      LAT = -99.
      ALT = -99.
      A = -99.
      B = -99.

*     --- if datafile id or PATH changed, initialise buffer
      IF  ((ICNT .NE. BCNT)    .OR. (ISTN .NE. BSTN) .OR.
     &     (IYEAR .NE. BYEAR) .OR. (IPATH .NE. BPATH)) THEN
*        --- open data file, checks parameters
         IF (.NOT. WSDAOP(IPATH,ICNT,ISTN,IYEAR,STAT,DUNIT)) RETURN
*        --- get station location and conversion factors
         IF (.NOT. WSATTR(DUNIT,STAT,LON,LAT,ALT,A,B)) RETURN
*        --- get data in buffer
         IF (.NOT. WSRDDA(DUNIT,LAT, A, B, STAT)) RETURN
*        --- store *valid* parameters in buffer variables
         BPATH = IPATH
         BCNT = ICNT
         BYEAR = IYEAR
         BSTN = ISTN
         BLON = LON
         BLAT = LAT
         BALT = ALT
         BA = A
         BB = B
*        --- close data file
         CLOSE (DUNIT)
         DUNIT = FSCLSD
      ELSE
*        --- buffer already initialised, just return buffer variables
         LON = BLON
         LAT = BLAT
         ALT = BALT
         A   = BA
         B   = BB
      ENDIF

      STAT = 0
      STERR = 12345
      RETURN
      END

*  ------------------------------------------------------------------
*
* weathr -- Get weathr data for IDAY from data file opend with STINFO
*  ------------------------------------------------------------------
*
* SUBROUTINE WEATHR
*       Returns weather data for a day specified in IDAY from a
*       data file opened with STINFO.
* PARAMETERS
*       name     type     description
*       --- in  ---
*       IDAY     int      day for which weather data are returned
*       --- out ---
*       STAT     int      return status: negative = error,
*                                        zero     = ok,
*                                        positive = warning.
*       RIRRAD    real    irradiation
*       RTMIN     real    minus temperature
*       RTMAX     real    maximum temperature
*       REMPR     real    early morning vapour pressure
*       RMWIND    real    mean wind speed
*       RPRECI    real    precipitation
* DESCRIPTION
*       WEATHR reads six "weather parameters" from the buffer filled by
*       STINFO: irradiation, lowest temperature, highest temperature,
*       early morning vapour pressure and precipitation for the
*       specified IDAY.
* STATUS
*       ISTAT is zero when all data are available.
*       The data returned by WEATHR may or may not be original
*       measurements: the status of these values is returned in
*       ISTAT. Each digit in ISTAT represents one status value for
*       one measurement. The order of digits is the same as in
*       the parameter list.  ISTAT is negative when data are missing
*       or an exception has occured.
* DIGIT FLAGS in ISTAT
*       1   value is an original measurement
*       2   value is an interpolation
*       3   value is an estimate
*       4   value is missing
* EXAMPLES
*       0        All values are OK
*       -444441  All values except precipitation are missing
*       3111111  Irradiation is an estimate.
*       2222223  All values are interpolations, precipiation is an
*                estimate.
*       -1       WEATHR called with wrong day number.
* ------------------------------------------------------------------ *

      SUBROUTINE WEATHR(IDAY,
     &           STAT, RIRRAD,RTMIN,RTMAX,REMPR,RMWIND,RPRECI)

*     IMPLICIT NONE

*     -------- parameters ---------
*     --- in ---
      INTEGER IDAY
*     --- out ---
      REAL      RIRRAD, RTMIN, RTMAX, REMPR, RMWIND, RPRECI
      INTEGER   STAT
*     --------- external ----------
*     void      WSITOA
*     void      WSMESS
      INTEGER   WSILEN


*     --------- COMMON: WSCFIL ---------------------------------- *
*                       Weathr Subsystem Char FILe names
*                       note: MAXFNM depends on system.
      INTEGER    MAXFNM
      PARAMETER (MAXFNM = 256)

      CHARACTER*(MAXFNM) PATH, FNAME, LOG
      COMMON   /WSCFIL/  PATH, FNAME, LOG
*     --------- common wscfil end ------------------------------ *

*     --------- COMMON: WSNFIL  --------------------------------- *
*                       Weathr Subsystem Num  FILe flags and logf. unit.
*     OF      - Output File flags
*     OFxxxW  - warning flag,   OFxxxF - fatal error flag
*     OFLOGx  - logfile flag,   OFOUTx - output (screen) flag
*     WARNFL  - .TRUE. if to create warings (screen and/or file)
*     ERRFL   - .TRUE. if to create error messages  ( " )

*                 Number Of Flags
      INTEGER    NOF
```

```
      PARAMETER (
     &           NOF = 4)

      LOGICAL    OF(NOF)
*               unit of logfile
      INTEGER    LUNIT
*               error flag from STINFO to WEATHR (12345 = ok)
      INTEGER    STERR
      LOGICAL    WARNFL, ERRFL

      COMMON /WSNFIL/ OF, LUNIT, WARNFL, ERRFL, STERR
*     --------- common wsnfil end ------------------------------ *

*     --------- COMMON: WSNBUF ------------------------------- *
*                     Weathr Subsystem Num BUFfer
*     --- BFATTR: attributes of data: DFINT, DFEST, DFUNDE
*     --- BFVALS: values
      INTEGER    NRDAYS, NRDP
      PARAMETER (
     &           NRDP = 6      ,
     &           NRDAYS = 366  )

      REAL       BFVALS(NRDAYS, NRDP)
      INTEGER    BFATTR(NRDAYS, NRDP)

      COMMON /WSNBUF/ BFVALS, BFATTR
*     --------- common wsnbuf end ------------------------------ *

*     --------- constants ---------
*     --- DF: Digit Flags returned in ISTAT
*     --- DFINT : interpolated
*     --- DFEST : estimated
*     --- DFUNDE: undefined
      INTEGER    DFINT, DFEST, DFUNDE
      PARAMETER (
     &           DFINT = 2,
     &           DFEST = 3,
     &           DFUNDE = 4)

*               order of measurements in buffer (and file)
      INTEGER    NIRRAD, NTMIN, NTMAX, NEMPR, NMWIND, NPRECI
      PARAMETER (NIRRAD = 1,
     &           NTMIN = 2,
     &           NTMAX  = 3,
     &           NEMPR = 4,
     &           NMWIND = 5,
     &           NPRECI = 6)

*             wrong day number
      INTEGER SPDAY
*             size of message for WSMESS
      INTEGER MAXMSG
*             internal write wrong
      INTEGER SSWEAT
*             weathr called after error from STINFO or STINFO not called
      INTEGER SOINIT
      PARAMETER (
     &        SOINIT = -21  ,
     &        SSWEAT = -902 ,
     &        MAXMSG = 200  ,
     &        SPDAY = -1    )
*     ------ local variables ------
*             attributes
      INTEGER ATTI
      LOGICAL ISMISS, ISESTI, ISINTP
      CHARACTER MSG*(MAXMSG), MTYPE*(MAXMSG), HLP*10

      SAVE
*     ------- start module --------
*     --- return NIL values on error
      RIRRAD = -99.
      RTMIN = -99.
      RTMAX = -99.
      REMPR = -99.
      RMWIND = -99.
      RPRECI = -99.

*     --- signal from STINFO (12345 "magic value": successful exit)
      IF (STERR .NE. 12345) THEN
         STAT = SOINIT
         CALL WSMESS(
     &        'Error in WEATHR: called after error from STINFO or '//
     &        'STINFO not called', STAT)
         RETURN
      ENDIF

*     --- check day
      IF ((IDAY .LT. 1) .OR. (IDAY .GT. NRDAYS)) THEN
         STAT = SPDAY
         IF (ERRFL) THEN
            CALL WSITOA(IDAY,HLP)
            MSG = 'Error in WEATHR: called with wrong day: '//HLP
            CALL WSMESS(MSG, STAT)
         ENDIF
         RETURN
      ENDIF

*     --- read data from buf, keep track of missing data
      STAT = 111111
      ISMISS = .FALSE.
      ISINTP = .FALSE.
      ISESTI = .FALSE.

      ATTI = BFATTR(IDAY, NIRRAD)
      IF (ATTI .NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (100000 * (ATTI-1))
      ENDIF
      RIRRAD = BFVALS(IDAY, NIRRAD)

      ATTI = BFATTR(IDAY, NTMIN)
      IF (ATTI .NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (10000 * (ATTI-1))
      ENDIF
      RTMIN = BFVALS(IDAY, NTMIN)

      ATTI = BFATTR(IDAY, NTMAX)
      IF (ATTI.NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (1000 * (ATTI-1))
      ENDIF
      RTMAX = BFVALS(IDAY, NTMAX)

      ATTI = BFATTR(IDAY, NEMPR)
      IF (ATTI.NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (100 * (ATTI-1))
      ENDIF
      REMPR = BFVALS(IDAY, NEMPR)

      ATTI = BFATTR(IDAY, NMWIND)
      IF (ATTI .NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (10 * (ATTI-1))
      ENDIF
      RMWIND = BFVALS(IDAY, NMWIND)

      ATTI = BFATTR(IDAY, NPRECI)
      IF (ATTI .NE. 1) THEN
          IF (ATTI .EQ. DFUNDE) ISMISS = .TRUE.
          IF (ATTI .EQ. DFEST) ISESTI = .TRUE.
          IF (ATTI .EQ. DFINT) ISINTP = .TRUE.
          STAT = STAT + (ATTI-1)
```

```
      ENDIF
      RPRECI = BFVALS(IDAY, NPRECI)

*     --- check result, create message if necessary and
*         requested.
*         messages:   missing        (printed if ERRFL on)
*                     no missing:    (printed if WARNFL on)
*                       1 -   estimated (not interpolated)
*                       2 -   estimated and interpolated
*                       3 -   interpolated (not estimated)
      IF (STAT .EQ. 111111) THEN
         STAT = 0
*        --- nothing to report
         RETURN
      ELSEIF (ISMISS) THEN
*     --- missing data
         STAT = -STAT
         IF (ERRFL) THEN
            WRITE(MSG, FMT='(3A,I3,1A,I6)', ERR=990)
     &         'Error in WEATHR: missing data, in: ',
     &         FNAME(1:WSILEN(FNAME)),
     &         ' day:', IDAY, ' attr.:', -(STAT)
            CALL WSMESS(MSG, STAT)
         ENDIF
      ELSE
*     --- warning
         IF (WARNFL) THEN
            MTYPE = ' '
            IF ((ISESTI) .AND. (.NOT. ISINTP)) THEN
               MTYPE = 'Warning in WEATHR: estimated'
            ELSEIF ((ISESTI) .AND. (ISINTP)) THEN
               MTYPE = 'Warning in WEATHR: estimated and interpolated'
            ELSE
               MTYPE = 'Warning in WEATHR: interpolated'
            ENDIF
            WRITE(MSG, FMT='(4A,I3,1A,I6)', ERR=990)
     &          MTYPE(1:WSILEN(MTYPE)),' data, in:',
     &          FNAME(1:WSILEN(FNAME)), ' day: ', IDAY,
     &         ' attr.:', STAT
            CALL WSMESS(MSG, STAT)
         ENDIF
      ENDIF
*     ------------------------------
      RETURN

990   CONTINUE
      STAT = SSWEAT
      CALL WSMESS ('Internal error in WEATHR: ', STAT )
      RETURN
      END

* ------------------------------------------------------------- *
* wsflgs -- set system flags
* ------------------------------------------------------------- *
* DESCRIPTION
*       Verify and set output flags to NEWFLG, store seperate
*       flag values in common field OF.
*       Initializes itself when called first time.
* ------------------------------------------------------------- *
      LOGICAL FUNCTION WSFLGS(NEWFLG, STAT)

*     IMPLICIT NONE
*     -------- parameters ---------
*     --- in ---
*               new flag
      INTEGER   NEWFLG
*     --- out ---
      INTEGER   STAT
*     --------- external ----------
*     void      WSMESS


*     --------- COMMON: WSNFIL  ------------------------------ *
*                       Weathr Subsystem Num  FILe flags and logf. unit.
*     OF      - Output File flags
*     OFxxxW  - warning flag,    OFxxxF - fatal error flag
*     OFLOGx  - logfile flag,    OFOUTx - output (screen) flag
*     WARNFL  - .TRUE. if to create warings (screen and/or file)
*     ERRFL   - .TRUE. if to create error messages  ( " )

      INTEGER    OFOUTF,OFOUTW,OFLOGF,OFLOGW, NOF
      PARAMETER (OFOUTF = 1,
     &           OFOUTW = 2,
     &           OFLOGF = 3,
     &           OFLOGW = 4,
*                Number Of Flags
     &           NOF = 4)

      LOGICAL    OF(NOF)
*                unit of logfile
      INTEGER    LUNIT
*                error flag from STINFO to WEATHR (12345 = ok)
      INTEGER    STERR
      LOGICAL    WARNFL, ERRFL

      COMMON /WSNFIL/ OF, LUNIT, WARNFL, ERRFL, STERR
*     --------- common wsnfil end ---------------------------- *

*     --------- constants ---------
*                Status value : Parameter FLAG wrong
      INTEGER   SPFLAG
      PARAMETER (
     &           SPFLAG  = -2      )
*     --------- local variables ---
*               copy of flag
      INTEGER   F
*               "value" of flag  (1000,100,10,1)
      INTEGER   FI
*               local copy of flag list (OF), OF is set after
*               every thing is checked, to avoid inconsistencies.
      LOGICAL   LOF(NOF)
*
      INTEGER   I
      CHARACTER MSG*80
      LOGICAL   FIRST

      SAVE
*     ----------- data ------------
      DATA FIRST /.TRUE./

*     ------- start module --------
      WSFLGS = .FALSE.
      F = NEWFLG

      IF (FIRST) THEN
*        --- initialise output flags (warning/error for stdout/file)
         DO 50, I = 1, NOF
            OF(I) = .FALSE.
50       CONTINUE
*        --- send errors to output to start with
         OF(OFOUTF) = .TRUE.
         ERRFL = .TRUE.
         WARNFL = .FALSE.
      ENDIF

      WRITE(MSG, '(A29,I10)' ) 'Error in STINFO: wrong flag: ', NEWFLG
      IF (F .LT. 0) THEN
         STAT = SPFLAG
         CALL WSMESS( MSG, STAT )
         RETURN
      ENDIF

*     --- loop for flags
*     --- flag "1000" is OFLOGW: warnings to log file
*     --- ...
*     --- flag "   1" is OFOUTF: fatals to output
      FI = 1000
      DO 100, I = OFLOGW, OFOUTF, -1
         IF ((F - FI) .GE. 0) THEN
            F = F - FI
            IF (F .GT. FI) THEN
*              --- not 1
               STAT = SPFLAG
               CALL WSMESS( MSG, STAT )
```

```
               RETURN
            ENDIF
*           --- enable flag I
            LOF(I) = .TRUE.
         ELSE
*           --- disable flag I
            LOF(I) = .FALSE.
         ENDIF
         FI = FI/10
100   CONTINUE

*     --- New flag ok, make it final.
      DO 500, I=1, NOF
         OF(I) = LOF(I)
500   CONTINUE
      ERRFL  = (OF(OFOUTF) .OR. OF(OFLOGF))
      WARNFL = (OF(OFOUTW) .OR. OF(OFLOGW))

      WSFLGS = .TRUE.
      RETURN

      END

* ------------------------------------------------------------ *
* wsdaop -- open new data file, close current (if any)
* ------------------------------------------------------------ *
* DESCRIPTION
*       Checks parameters, constructs a file name , and opens the
*       file.
* REMARKS
*       Although this routine is written in standard fortran, file-
*       name construction is tricky: correct filename syntax is
*       is system dependend.
* ------------------------------------------------------------ *

      LOGICAL FUNCTION WSDAOP( IPATH, CNT, STN, YEAR,
     &                         STAT,  DUNIT)

*     IMPLICIT NONE

*     -------- parameters ---------
*     --- in ---
      CHARACTER IPATH*(*)
      CHARACTER CNT*(*)
      INTEGER   STN
      INTEGER   YEAR
*     --- out ---
      INTEGER   STAT
*             data unit
      INTEGER   DUNIT

*     --------- external ----------
*     viod      WSMESS, WSITOA
      INTEGER   WSILEN
      LOGICAL   WSOPEN

*     -------- constants ----------
*             year is wrong
      INTEGER SPYEAR,
*             country is wrong
     &        SPCNT,
*             station is wrong
     &        SPSTN,
*             can't open data file
     &  SFDATA,
*             internal error in wsdaop
     &  SSDAOP,
*             internal error in wsopen
     &  SSOPEN,
*             maximum size of country
     &  MAXCNT,
*             maximum year
     &  MAXYR,
*             minimum year
     &  MINYR,
*             maximum for station code
     &  MAXSTN
      PARAMETER (
     &      SPYEAR = -3    ,
     &      SPCNT  = -4    ,
     &      SPSTN  = -5    ,
     &      SFDATA = -12   ,
     &      SSOPEN = -903  ,
     &      SSDAOP = -900  ,
     &      MAXCNT = 6     ,
     &      MAXYR  = 1999  ,
     &      MINYR  = 1000  ,
     &      MAXSTN = 99    )


*     --------- COMMON: WSCFIL ------------------------------- *
*                     Weathr Subsystem Char FILe names
*                     note: MAXFNM depends on system.
      INTEGER    MAXFNM
      PARAMETER (MAXFNM = 256)

      CHARACTER*(MAXFNM) PATH, FNAME, LOG
      COMMON   /WSCFIL/  PATH, FNAME, LOG
*     --------- common wscfil end ---------------------------- *


*     ------ local variables ------
      INTEGER            SIZE
      INTEGER            LCNT
      INTEGER            IOSS
      CHARACTER*10       S
      CHARACTER*200      MSG,HMSG
*                              data file name, including path
      CHARACTER*(MAXFNM)         FNDAT, LFNAME, HFNAME

      SAVE
*     ------- start module --------
      WSDAOP = .FALSE.
      LFNAME = ' '

*     ----------------------------
*     --- check parameters
*     --- year is 3 digits only
      IF ((YEAR .LT. MINYR) .OR. (YEAR .GT. MAXYR)) THEN
          MSG= 'Error in STINFO: year: "'
          CALL WSITOA(YEAR, S)
          HMSG = MSG
          MSG = HMSG(1:WSILEN(HMSG))//S(1:WSILEN(S))//
     &          '" is out of range'
          STAT = SPYEAR
          CALL WSMESS( MSG, STAT )
          RETURN
      ENDIF

      LCNT = WSILEN(CNT)

*     --- size of country
      IF ((LCNT .LT. 1) .OR. (LCNT .GT. MAXCNT)) THEN
         STAT = SPCNT
         CALL WSMESS(
     &       'Error in STINFO: wrong string size for country', STAT )
         RETURN
      ENDIF

*     --- station
      IF ((STN .LT. 0) .OR. (STN .GT. MAXSTN)) THEN
         STAT = SPSTN
         CALL WSMESS(
     &       'Error in STINFO: station code out of range', STAT )
         RETURN
      ENDIF
*     ----------------------------


*     ----------------------------
*     --- construct name of file (without path), put in LFNAME
*     --- country part
      LFNAME = CNT(1:LCNT)
      SIZE  = LCNT
```

```
*     --- get station number part of filename
      CALL WSITOA(STN, S)
      HFNAME = LFNAME
      LFNAME = HFNAME(1:SIZE)//S
      SIZE = WSILEN(LFNAME)

*     --- year comes in extension part of filename
      HFNAME = LFNAME
      LFNAME = HFNAME(1:SIZE) // '.'
      SIZE = SIZE + 1

*     --- get year
      S = ' '
      WRITE(S, ERR= 900, FMT='(I3.3)') (YEAR - 1000)
      HFNAME = LFNAME
      LFNAME = HFNAME(1:SIZE) // S
      SIZE = SIZE + 3
*     ----------------------------

*     ----------------------------
*     --- add path to name
      IF (WSILEN(IPATH) .NE. 0) THEN
          FNDAT = IPATH(1:WSILEN(IPATH))  // LFNAME(1:SIZE)
      ELSE
*         --- allow for "default directory"
          FNDAT = LFNAME
        ENDIF
      SIZE = WSILEN(FNDAT)
*     ----------------------------

*     ----------------------------
*     --- open file for reading
      IF (.NOT. WSOPEN(FNDAT(1:SIZE), 'r', DUNIT, STAT, IOSS)) THEN
*        --- failed, report if not internal error
         IF (STAT .EQ. SSOPEN) RETURN
         CALL WSITOA(IOSS, S)
         MSG = 'Error in STINFO: cannot open: "'//FNDAT(1:SIZE)//
     &         '" (system status ='//S(1:WSILEN(S))//')'
         STAT = SFDATA
         CALL WSMESS(MSG, STAT)
         RETURN
      ENDIF
*     ----------------------------
*     --- make path/filename global for error messages
      FNAME = LFNAME
      PATH  = IPATH
      WSDAOP = .TRUE.
      RETURN

*-------------------------------------------------------------  *
*
*     --- Internal write errors
900   CONTINUE
      STAT = SSDAOP
      CALL WSMESS( 'Internal error in STINFO', STAT )
      RETURN

      END

* ------------------------------------------------------------ *
* wsattr -- get attributes of data for weather station
* ------------------------------------------------------------ *
* DESCRIPTION
*       This routine is called after the file is opened.
*       Comment lines are read until a data line is seen. This line
*       must contain 5 (no more, no less) REAL values which
*       represent (in  this order): longitude, latitude and
*       altitude and irradiaton conversion factors A and B.
*       If no conversion is needed (data are already in irradiation
*       per square meter) then A and B must be zero.
* ------------------------------------------------------------ *

      LOGICAL FUNCTION  WSATTR(DUNIT,STAT,LON,LAT,ALT,A,B)

*     IMPLICIT NONE
*     -------- parameters ---------
*     --- in ---
*               data file unit
      INTEGER   DUNIT
*     --- out ---
      INTEGER   STAT
      REAL      LON, LAT, ALT
*               radiation conversion constants
      REAL      A, B
*     ------- constants -----------
*               maximum size of input string
      INTEGER   MAXSTR
*               comment char in column 1
      CHARACTER CHRCOM*1
*               signal comment seen
      INTEGER   SMCOMM
*               signal unexpected end of data file
      INTEGER   SFDEOF
*               signal read error in data file
      INTEGER   SFDERR

      PARAMETER (
     &          CHRCOM = '*' ,
     &          SMCOMM = 1   ,
     &          SFDEOF = -14 ,
     &          SFDERR = -15 ,
     &          MAXSTR = 132 )

*     --------- external ----------
*     void      WSMESS

*     ------ local variables ------
      CHARACTER*(MAXSTR) S
*               local copies of location parms
      REAL      LLON, LLAT, LALT, LA, LB
      SAVE
*     ------- start module --------
      WSATTR = .FALSE.

*     --- loop until location data line is read
100   CONTINUE
*        --- read a string
         READ( UNIT=DUNIT, FMT='(A)', END=900, ERR=910 ) S
         IF(S(1:1) .EQ. CHRCOM) THEN
*            --- print if warn-flag(s) is (are) on
             STAT = SMCOMM
             CALL WSMESS( S, STAT)
*            --- if not equal to old value: error in wsmess.
             IF (STAT .NE. SMCOMM) RETURN
             STAT = 0
             GOTO 100
         ENDIF
*     --- end of loop

*     --- not comment, get data again
      BACKSPACE(UNIT=DUNIT)
      READ(UNIT=DUNIT, FMT=*,END=900,ERR=920) LLON, LLAT, LALT,
     &                                        LA, LB
*     --- read was succesfull, copy to global parms.
      LON = LLON
      LAT = LLAT
      ALT = LALT
      A   = LA
      B   = LB
      WSATTR = .TRUE.
      RETURN

* ------------------------------------------------------------ *
*     read errors:

900   CONTINUE
      STAT = SFDEOF
      CALL WSMESS( 'Error in STINFO: unexpected end of file.', STAT )
      RETURN

910   CONTINUE
      STAT = SFDERR
```

```
      CALL WSMESS( 'Error in STINFO: unexpected read error.', STAT )
      RETURN

920   CONTINUE
      STAT = SFDERR
      CALL WSMESS(
     &    'Error in STINFO: incorrect geografical data line', STAT )
      RETURN

      END

* ------------------------------------------------------------ *
* wsrdda -- read data in buffer
* ------------------------------------------------------------ *
* DESCRIPTION
*       WSRDDA reads the actual weathr data, i.e. the second part
*       of the datafile. First the buffer is filled with the
*       value for "undefined". Then the data are read.
*       The actual reading is done in WSGREC (Get RECord) witch
*       returns data values and attributes if an attribute line
*       preceded the data record.
*       After reading the available data, the buffer is scanned
*       for undefined values. These are replaced with an
*       interpolated value (if possible). If data are in hours sun
*       then a conversion to irradiation per square meter is
*       is performed.
* ------------------------------------------------------------ *

      LOGICAL FUNCTION  WSRDDA( DUNIT, LAT, A, B,
     &                          STAT )

*     IMPLICIT NONE

*     --------- parameters --------
*     --- in ---
      INTEGER   DUNIT
*               latitude and conversion factors for hours sun to
*               irradiation.
      REAL      LAT, A, B
*     --- out ---
      INTEGER   STAT
*     --------- constants ---------
      REAL      VALUND
      PARAMETER (
     &          VALUND = -99.       )

*     --- DF: Digit Flags returned in ISTAT
*     --- DFINT : interpolated
*     --- DFUNDE: undefined
      INTEGER    DFOK, DFINT, DFUNDE
      PARAMETER (DFOK = 1,
     &           DFINT = 2,
     &           DFUNDE = 4)



*     --------- COMMON: WSNBUF ------------------------------- *
*                     Weathr Subsystem Num BUFfer
*     --- BFATTR: attributes of data: DFOK, DFINT, DFEST, DFUNDE
*     --- BFVALS: values
      INTEGER    NRDAYS, NRDP
      PARAMETER (
     &           NRDP = 6      ,
     &           NRDAYS = 366  )

      REAL       BFVALS(NRDAYS, NRDP)
      INTEGER    BFATTR(NRDAYS, NRDP)

      COMMON /WSNBUF/ BFVALS, BFATTR
*     --------- common wsnbuf end ---------------------------- *

*     --------- external ----------
*               get NRDP data points from file
      LOGICAL   WSGREC
*               convert hours sun to irradiation
*     void      WSCONI
*     -------- local variables ----
      INTEGER   TODAY
      INTEGER   I
      INTEGER   COL
*               list of data for day
      REAL      DATLST(NRDP)
*               list of attributes for day
      INTEGER   ATTLST(NRDP)
*               attributes set in data file with attribute line.
      INTEGER   ATTSET(NRDP)
      REAL      BVAL
      REAL      SLOPE
*               in missing part of column
      LOGICAL   INMIS
*               start of missing part of column
      INTEGER   START

      SAVE
*     ------- start module --------
*     --- initialize data buffer
      DO 100, TODAY = 1, NRDAYS
         DO 110, I = 1, NRDP
            BFVALS(TODAY, I) = VALUND
            BFATTR(TODAY, I) = DFUNDE
110      CONTINUE
100   CONTINUE
*     --- initialize attribute-set list to default
      DO 120, I=1, NRDP
            ATTSET(I) = DFOK
120   CONTINUE

*     --- fill buffer, loop until eof or all days read
200   CONTINUE
*        --- get a line with NRDP data
         IF (.NOT. WSGREC( DUNIT, ATTSET,
     &                     STAT,  TODAY, DATLST, ATTLST)) GOTO 300
         DO 250, COL=1, NRDP
            BFVALS(TODAY, COL) = DATLST(COL)
            BFATTR(TODAY, COL) = ATTLST(COL)
250      CONTINUE
         GOTO 200

*     --- end read loop ---
300   CONTINUE
*     --- abnormal exit if status unequal to eof.
      IF (STAT .NE. 0) RETURN

*     --- see if interpolations needed. (attributes undefined)
*     --- don't interpolate for rainfall: last collumn
      DO 400, COL=1, NRDP-1
         INMIS = .FALSE.
         DO 410, TODAY=1, NRDAYS

            IF ((BFATTR(TODAY,COL) .EQ. DFUNDE) .AND.
     &          (.NOT. INMIS)) THEN
*              --- beginning of missing part
               INMIS = .TRUE.
               START = TODAY

            ELSEIF (INMIS .AND. (BFATTR(TODAY,COL) .NE. DFUNDE)) THEN
*              --- at end of missing part
*              --- if not from day 1 interpolate
               INMIS = .FALSE.
               IF (START .NE. 1) THEN
*                 --- compute slope
                  SLOPE = (BFVALS(TODAY, COL) - BFVALS(START-1,COL))/
     &                    (TODAY-START+1)
                  BVAL = BFVALS(START-1,COL)
                  DO 411, I=START, TODAY-1
                     BFVALS(I, COL) = BVAL + SLOPE*FLOAT(I-START+1)
                     BFATTR(I, COL) = DFINT
411               CONTINUE
               ENDIF
            ENDIF
410      CONTINUE
*        --- we don't have to check for missing at end (inmis=.true.)
*        --- because values are already set to "undefined"
400   CONTINUE
```

```
*     --- if data in hours sun, convert to irradiation
      CALL WSCONI(LAT, A, B)

      WSRDDA = .TRUE.
      RETURN

      END

* ------------------------------------------------------------ *
* wsgrec -- get a record: NDPR data with atributes
* ------------------------------------------------------------ *
* DESCRIPTION
*       Datafiles are devided in to parts: a comment section
*       and a data section. This function reads the data section.
*       Two types of data lines are possible: normal data lines
*       and Attribute lines. Attribute lines are marked by the
*       special station value -999 (year and day are ignored
*       but a dummy value must be present). The parameter ATTSET
*       holds the attributes from the last attribute line. The values
*       in ATTLST may differ from ATTSET because data can be missing.
* PARAMETERS
*       name    class  type   description
*       ------------------------------------------------------
*       dunit   in     int    unit of data file
*       attset  in/out int()  attribute's set from last att. line.
*       stat    out    int    exit status (see below)
*       dayrd   out    int    day read
*       datlst  out    real() array (NRDP elements) with data points
*       attlst  out    int()  array (NRDP elements) with attributes
* RETURNS in STAT
*       0       signals success
*       other   signals failure (message is printed)
* RETURNS
*       .FALSE. when done : STAT = 0
*               on error  : STAT = exit error code
*       .TRUE.  got data, more lines expected.
* ------------------------------------------------------------ *

      LOGICAL FUNCTION  WSGREC( DUNIT,
     &                          ATTSET,
     &                          STAT, DAYRD, DATLST, ATTLST)

*     IMPLICIT NONE

      INTEGER   NRDP
      PARAMETER (
     &          NRDP = 6 )

*     -------- parameters ---------
*     --- in ---
*               datafile unit no.
      INTEGER   DUNIT
*     --- in/out ---
*               attributes as specified in datafile.
*               These are in effect until reset in file,
*               and are not affected by missing data.
      INTEGER   ATTSET(NRDP)
*     --- out ---
*               day read
      INTEGER   DAYRD
*               list of data read
      REAL      DATLST(NRDP)
*               list of attributes for this line.
      INTEGER   ATTLST(NRDP)

*     --------- external ----------
*     void      WSMESS

*     --------- constants ---------
      REAL      VALUND
      INTEGER   SFDERR
      INTEGER   SFDEOF

      PARAMETER (
     &          SFDERR = -15       ,
     &          SFDEOF = -14       ,
     &          VALUND = -99.      )
*     --- DF: Digit Flags returned in ISTAT
*     --- DFUNDE: undefined
      INTEGER    DFUNDE
      PARAMETER (DFUNDE = 4)


*     ------ local variables ------
      INTEGER   STAT, I
      INTEGER   YEARRD, STNRD

      SAVE
*     ------- start module --------
      WSGREC = .FALSE.

      READ(UNIT=DUNIT,FMT=*,END=900,ERR=910)
     &             STNRD, YEARRD, DAYRD, DATLST

*     --- Is it a line with data or attributes?
      IF (STNRD .EQ. -999) THEN
*        --- got line with attributes
         DO 100,I=1, NRDP
            ATTSET(I) = NINT(DATLST(I))
100      CONTINUE
*        --- also get next line: MUST have data for these attributes
         READ(UNIT=DUNIT,FMT=*,END=920,ERR=910)
     &             STNRD, YEARRD, DAYRD, DATLST
      ENDIF

*     --- set attributes
      DO 200, I=1, NRDP
         IF (DATLST(I) .LE. VALUND) THEN
            ATTLST(I) = DFUNDE
         ELSE
            ATTLST(I) = ATTSET(I)
         ENDIF
200   CONTINUE

      WSGREC = .TRUE.
      STAT = 0
      RETURN


* ------------------------------------------------------------ *
*     --- labels for read
900   CONTINUE
*     --- eof
      STAT = 0
      RETURN

910   CONTINUE
      STAT =SFDERR
      CALL WSMESS('Error in STINFO: incorrect data file', STAT )
      RETURN

920   CONTINUE
      STAT = SFDEOF
      CALL WSMESS('Error in STINFO: unexpected end of file', STAT )
      RETURN

      END

*
* ------------------------------------------------------------ *
* wsconi -- convert hours sun to irradation
* ------------------------------------------------------------ *
* PUPOSE
*       convert hours sun measurements to irradiation
* DESCRIPTION
*       WSCONI tests if A and B are not zero the measurements
*       are in hours sun and have to be converted to irradiation
*       per square meter.
* ------------------------------------------------------------ *
      SUBROUTINE WSCONI(ILAT, A, B)

*     IMPLICIT NONE
```

```
*      --------- parameters --------
*      --- in ---
       REAL A, B, ILAT
*      --------- constants ---------
*                position of irradiation value in buffer and file
       INTEGER    NIRRAD
       PARAMETER (NIRRAD = 1)

*                PI and conversion factor from degrees to radians
       REAL       PI, RAD
*                undefined data
       REAL       VALUND
       PARAMETER (
     &            PI = 3.141592654  ,
     &            RAD = 0.017453292  ,
     &            VALUND  = -99.0        )
*      --------- external ----------

*      --------- COMMON: WSNBUF ------------------------------ *
*                       Weathr Subsystem Num BUFfer
*      --- BFATTR: attributes of data: DFOK, DFINT, DFEST, DFUNDE
*      --- BFVALS: values
       INTEGER    NRDAYS, NRDP
       PARAMETER (
     &            NRDP = 6      ,
     &            NRDAYS = 366  )

       REAL       BFVALS(NRDAYS, NRDP)
       INTEGER    BFATTR(NRDAYS, NRDP)

       COMMON /WSNBUF/ BFVALS, BFATTR
*      --------- common wsnbuf end ---------------------------- *

*      --------- local -----------
*                length of sun day for particalar latitude.
       REAL       DAYL(NRDAYS)
*                angot for this latitude
       REAL       ANGOT(NRDAYS)
*                last used latitude
       REAL       LAT
       REAL       DAY, DSINB, SC, DEC, SINLD, COSLD, AOB
       INTEGER    IDAY
       SAVE
*      --------- data --------------
*                doesn't exist
       DATA       LAT /100/
*      --------- start module ------

*      --- no conversion needed when less then .01.
*      --- (test for less then .01 instead of 0: this is defacto 0)
       IF ((A .LT. 0.01) .AND. (B .LT. 0.01)) RETURN

       IF (ILAT .NE. LAT) THEN
*      --- new latitude, recompute DAYL and ANGOT.
          LAT = ILAT

          DO 100, IDAY=1, 366

             DAY = FLOAT (IDAY)

*            --- declination of the sun as function of daynumber (DAY)
             DEC = -ASIN (SIN(23.45*RAD)*COS(2.*PI*(DAY+10.)/365.))

*            --- SINLD, COSLD and AOB are intermediate variables
             SINLD = SIN (RAD*LAT)*SIN (DEC)
             COSLD = COS (RAD*LAT)*COS (DEC)
             AOB   = SINLD/COSLD

*            ---  daylength
             DAYL(IDAY) = 12.0*(1.+2.*ASIN (AOB)/PI)

*            ---- integral of sine of solar elevation
             DSINB = 3600.*(DAYL(IDAY)*SINLD+24.*COSLD*
     &                           SQRT(1.-AOB*AOB)/PI)

*            --- solar constant (SC) and daily extraterrestrial radiation
(ANGOT)
             SC = 1370.*(1.+0.033*COS(2.*PI*DAY/365.))
             ANGOT(IDAY) = SC*DSINB/1000.

100       CONTINUE
       ENDIF

*      --- replace hours sun with radiation from daylength and angot
       DO 200, IDAY=1, 366
             IF (BFVALS(IDAY,NIRRAD) .NE. VALUND) THEN
                BFVALS(IDAY,NIRRAD) =
     &             ANGOT(IDAY)*(A+B*BFVALS(IDAY,NIRRAD)/DAYL(IDAY))
             ENDIF
200    CONTINUE

       RETURN
       END

* ------------------------------------------------------------ *
* wsmess -- (don't) print message on output and/or logfile
* ------------------------------------------------------------ *
* PARAMETERS
*       name    class   type    description
*       instr   in      char*   message to print
*       stat    in/out  integ   status value
* PURPOSE
*       WSMESS is used for printing (or not if flags are
*       disabled) warnings and/or error messages to the
*       standard output and/or log file.
* REMARK
*       STAT is written if WSMESS exits with an error.
* STATUS CODES
*        lowest highest description
*        ----------------------------------------------
*         <<    -111114 data are missing, print this code
*       -111111      -1 error detected (e.g. wrong parameters)
*        1       1      (SMCOMM): print a comment line
*        2       111111 not used
*        111112  333333 warning, print this code
* REMARKS
*       WSMESS is designed so that it is most efficient
*       if all output is disabled.
* ------------------------------------------------------------ *
       SUBROUTINE WSMESS (INSTR, STAT)

*      IMPLICIT NONE

*      -------- parameters ---------
*      --- in ---
*                string with message (if error)
       CHARACTER  INSTR*(*)
*      --- in/out ---
*                error-code or warning/missing-code
       INTEGER    STAT
*      --------- external ----------
       INTEGER    WSILEN
       LOGICAL    WSOPEN
*      void       WSITOA

*      --------- COMMON: WSNFIL  ------------------------------ *
*                       Weathr Subsystem Num  FILe flags and logf. unit.
*      OF       - Output File flags
*      OFxxxW   - warning flag,    OFxxxF - fatal error flag
*      OFLOGx   - logfile flag,    OFOUTx - output (screen) flag
*      WARNFL   - .TRUE. if to create warings (screen and/or file)
*      ERRFL    - .TRUE. if to create error messages  ( " )

       INTEGER    OFOUTF,OFOUTW,OFLOGF,OFLOGW, NOF
       PARAMETER (OFOUTF = 1,
     &            OFOUTW = 2,
     &            OFLOGF = 3,
     &            OFLOGW = 4,
*                Number Of Flags
     &            NOF = 4)

       LOGICAL    OF(NOF)
*                unit of logfile
       INTEGER    LUNIT
```

```
*               error flag from STINFO to WEATHR (12345 = ok)
      INTEGER    STERR
      LOGICAL    WARNFL, ERRFL

      COMMON /WSNFIL/ OF, LUNIT, WARNFL, ERRFL, STERR
*     --------- common wsnfil end ---------------------------- *
*     --------- COMMON: WSCFIL ------------------------------- *
*                     Weathr Subsystem Char FILe names
*                     note: MAXFNM depends on system.
      INTEGER    MAXFNM
      PARAMETER (MAXFNM = 256)

      CHARACTER*(MAXFNM) PATH, FNAME, LOG
      COMMON   /WSCFIL/  PATH, FNAME, LOG
*     --------- common wscfil end ---------------------------- *

*     --------- constants ---------
*               signal comment
      INTEGER    SMCOMM
*               write to (open) logfile failed
      INTEGER    SFLOGW
*               open of logfile failed: unknown mode.
      INTEGER    SSOPEN
*               unknown status code passed to WSMESS
      INTEGER    SSUSTA
*               maximum size for messages
      INTEGER    MAXMSG
*               special unit nbr: not a unit number (file is closed)
      INTEGER    FSCLSD
*               can't open log file
      INTEGER    SFLOGF
*               STINFO not called, or wrong initialisation
      INTEGER    SOINIT

      PARAMETER (
     &           SOINIT = -21     ,
     &           SFLOGF = -13     ,
     &           FSCLSD = -1      ,
     &           MAXMSG = 200     ,
     &           SSOPEN = -903    ,
     &           SFLOGW = -16     ,
     &           SSUSTA = -904    ,
     &           SMCOMM = 1       )

*     ------ local variables ------
      INTEGER    IOSS
      CHARACTER  MSG*(MAXMSG)
      CHARACTER  S*10
      LOGICAL    ISERR
*               status of WSMESS (used with WSOPEN)
      INTEGER    MSTAT

      SAVE
*     ------- start module --------

*     --- get level of status: warning or fatal-error
      ISERR = (STAT .LT. 0)

      IF (STAT .EQ. SOINIT) THEN
*        --- error caused by NOT calling STINFO, must print to output
         PRINT *, INSTR(1:WSILEN(INSTR))
         RETURN
      ENDIF

*     --- create a message if output/logfile enabled
      IF (ISERR) THEN
         IF (ERRFL) THEN
*           --- error output enabled, create message
            MSG = ' '
            IF (STAT .LT. -111111)  THEN
*              --- missing data
               MSG = INSTR
            ELSEIF (STAT .GT. -900) THEN
*              --- some kind of error detected: report INSTR
               MSG = INSTR
            ELSE
*              --- internal error, unknown status
               PRINT *, 'Internal error in WEATHR/STINFO: ',
     &                  SSUSTA
               STAT = SSUSTA
               RETURN
            ENDIF
         ELSE
*           --- error output disabled, nothing to do
            RETURN
         ENDIF

*     --- warning
      ELSE
         IF (WARNFL) THEN
*           --- warning output enabled
            MSG = ' '
            IF (STAT .GT. 111111) THEN
*              --- interpolated or artificial data
               MSG = INSTR
            ELSEIF (STAT .EQ. SMCOMM) THEN
*              --- comment line read, copy to MSG
               MSG = INSTR
            ELSE
*              --- internal error, unknown status
               PRINT *, 'Internal error in WEATHR/STINFO: ',
     &                  SSUSTA
               STAT = SSUSTA
               RETURN
            ENDIF

         ELSE
*           --- warning output disabled, nothing to do
            RETURN
         ENDIF
      ENDIF

*     --- write message to output and/or logfile
      IF (ISERR) THEN
         IF (OF(OFOUTF)) PRINT *, MSG(1:WSILEN(MSG))
         IF (OF(OFLOGF)) THEN
*           --- open logfile if closed
            IF (LUNIT .EQ. FSCLSD) THEN
               IF (.NOT. WSOPEN(LOG, 'w', LUNIT, MSTAT, IOSS))
     &                  GOTO 920
            ENDIF
            WRITE(UNIT=LUNIT, FMT='(1X,A)', ERR=990, IOSTAT=IOSS)
     &           MSG(1:WSILEN(MSG))
         ENDIF

      ELSE
         IF (OF(OFOUTW)) PRINT *, MSG(1:WSILEN(MSG))
         IF (OF(OFLOGW)) THEN
            IF (LUNIT .EQ. FSCLSD) THEN
               IF (.NOT. WSOPEN(LOG, 'w', LUNIT, MSTAT, IOSS))
     &                  GOTO 920
            ENDIF
            WRITE(UNIT=LUNIT, FMT='(1X,A)', ERR=990, IOSTAT=IOSS)
     &           MSG(1:WSILEN(MSG))
         ENDIF
      ENDIF

      RETURN

* ------------------------------------------------------------ *
*     ----error labels

*     --- log open failed: write to output instead
920   CONTINUE
*     --- If not internal error in WSOPEN print a message and put
*     --- code number in STAT. Internal errors are handled in WSOPEN.
      IF (MSTAT .EQ. SSOPEN) THEN
         STAT = MSTAT
         RETURN
      ENDIF
      STAT = SFLOGF
      CALL WSITOA(IOSS, S)
      MSG = 'Error in WEATHR/STINFO: cannot open logfile:"' //
     &      LOG(1:WSILEN(LOG)) // '"' //
```

```
     &      ', (system status =' // S(1:WSILEN(S)) // ')'
      PRINT *, MSG
      RETURN
*     ------------------------------

990   CONTINUE
      STAT = SFLOGW
      CALL WSITOA(IOSS, S)
      MSG = 'Error in WEATHR/STINFO: cannot write to logfile:"'//
     &      LOG(1:WSILEN(LOG)) // '"' //
     &      ', (system status =' // S(1:WSILEN(S)) // ')'
      PRINT *, MSG
      RETURN

      END


* ------------------------------------------------------------ *
* wsopen -- open file NAME with MODE, return unit and status
* ------------------------------------------------------------ *
* DESCRIPTION
*       WSOPEN opens the file NAME with access mode MODE.
*       A unit number is assigned to UNUM for accessing the file.
* MODES
*       The following modes are defined:
*       'r'     open file for reading. On multi user systems
*               shareable files can be read.
*       'w'     the file is opend for writing. If the file exists
*               the old version is deleted first.
* RETURNS
*       The function result is .TRUE. if successfull.
*
*       The parameter STAT is used to return errors caused by
*       wrong arguments. IOSS is used to return errors from the
*       environment, e.g. trying to open a file for reading that
*       doesn't exist. These codes are highly system/compiler
*       dependend.
* ------------------------------------------------------------ *

      LOGICAL FUNCTION WSOPEN(NAME, MODE,
     &                        FUNIT, STAT, IOSS)

*     IMPLICIT NONE

*     -------- parameters ---------
*     --- in ---
      CHARACTER     NAME*(*)
*                   valid modes are: 'r' = readonly; 'w' = writeonly
      CHARACTER     MODE*1
*     --- out ---
*                   file handle
      INTEGER       FUNIT
*                   exit status
      INTEGER       STAT
*                   iostatus returned by system
      INTEGER       IOSS
*     --------- external ----------
      INTEGER       WSILEN
*     void          WSFUN
*     ------ local variables ------
      INTEGER       FSCLSD
      INTEGER       SSOPEN
      PARAMETER (
     &           SSOPEN = -903     ,
     &           FSCLSD = -1       )
*                   size of name
      INTEGER   LNAME
      LOGICAL   EXST
      SAVE
*     ------- start module --------
      WSOPEN = .FALSE.
      IOSS = 0
      STAT = 0

*     --- get unit number
      CALL WSFUN(FUNIT)
      LNAME = WSILEN(NAME)

*     --- --- read mode ------------------------------
*     --- open readonly, sequential, existing (old) file
      IF (MODE(1:1) .EQ. 'r') THEN
         OPEN(UNIT=FUNIT,
     &        STATUS='OLD',
     &        IOSTAT=IOSS,
     &        FILE=NAME(1:LNAME))
         IF (IOSS .NE. 0) THEN
            FUNIT = FSCLSD
            RETURN
         ELSE
            WSOPEN = .TRUE.
            RETURN
         ENDIF

*     --- ---- write mode ------------------------
*     --- open new file for sequential writing
      ELSEIF (MODE(1:1) .EQ. 'w') THEN
*        --- to be safe: first delete file if it exists
         INQUIRE(FILE=NAME(1:LNAME), EXIST=EXST)
         IF (EXST) THEN
            OPEN(UNIT=FUNIT,
     &           FILE=NAME(1:LNAME),
     &           STATUS='OLD',
     &           IOSTAT=IOSS )
            CLOSE(UNIT=FUNIT, STATUS='DELETE')
         ENDIF
*        --- now we can open with 'NEW'
         OPEN(UNIT=FUNIT,
     &        FILE=NAME(1:LNAME),
     &        STATUS = 'NEW',
     &        IOSTAT = IOSS )
         IF (IOSS .NE. 0) THEN
            FUNIT = FSCLSD
            RETURN
         ELSE
            WSOPEN = .TRUE.
            RETURN
         ENDIF

*     --- ---- cannot happen ---------------------
      ELSE
           STAT = SSOPEN
           FUNIT = FSCLSD
*          --- must print to output (can't call wsmess)
           PRINT *,
     &       'Error in WEATHER/STINFO: internal error, code:', STAT
           RETURN
      ENDIF
      END

* ------------------------------------------------------------ *
* wsilen -- return significant length of a string
* ------------------------------------------------------------ *
* DESCRIPTION
*       The string is searched from end to begin for the first
*       non-space.
*       If the string is empty 0 is returned.
* HISTORY
*       Author: Daniel van Kraalingen, orginal name: ILEN.
*       Date: Aug 87
* ------------------------------------------------------------ *
      INTEGER FUNCTION WSILEN (STRING)

      CHARACTER*(*) STRING
      SAVE

      DO 10 WSILEN=LEN(STRING),1,-1
         IF (STRING(WSILEN:WSILEN).NE.' ') RETURN
10    CONTINUE

      RETURN
      END

* ------------------------------------------------------------ *
* wsstar -- return first significant character of string
```

```
* ------------------------------------------------------------ *
* DESCRIPTION
*       The string is searched from begin to end for the first
*       non-space.
*       If the string is empty (spaces only) 0 is returned.
* HISTORY
*       Author: Daniel van Kraalingen, orginal name: ISTART.
*       Date: Aug 87
* ------------------------------------------------------------ *
      INTEGER FUNCTION WSSTAR (STRING)

      CHARACTER*(*) STRING

      SAVE

      DO 10 WSSTAR=1,LEN(STRING)
         IF (STRING(WSSTAR:WSSTAR).NE.' ') RETURN
10    CONTINUE

      WSSTAR = 0

      RETURN
      END

* ------------------------------------------------------------ *
* wsfun -- get free unit number
* ------------------------------------------------------------ *
* DESCRIPTION
*       wsfun scans for unused unitnumbers, starting with 92
*       down to 40.
*       The numbers are checked with INQUIRE, so that numbers
*       are returned to the "free-list" by using standard CLOSE.
* RETURNS
*        -1 in FD if no valid unit number found.
* BUGS
*       Inconsequent use causes subtle bugs.
* ------------------------------------------------------------ *
      SUBROUTINE WSFUN(FD)

*     IMPLICIT NONE

*     -------- parameters ---------
      INTEGER   FD

*     ------- constants -----------
*               lowest,highest unit number opened:
      INTEGER   LOWUNT, MAXUNT
      PARAMETER (LOWUNT = 40,
     &           MAXUNT = 92)

*     ------ local variables ------
      LOGICAL   ISOPEN
      SAVE
*     ------- start module --------

*     --- fine unused unit number
      DO 10, FD = MAXUNT, LOWUNT, -1
         INQUIRE(UNIT=FD, OPENED=ISOPEN)
         IF (.NOT. ISOPEN) RETURN
10    CONTINUE

      FD = -1
      RETURN

      END

* ------------------------------------------------------------ *
* wsitoa -- convert integer to (left adjusted) ascii in string *
* ------------------------------------------------------------ *
* DESCRIPTION                                                  *
*       The integer I is written in STRING starting at the     *
*       first postion.                                         *
* ------------------------------------------------------------ *
      SUBROUTINE WSITOA (I, STRING)

*     IMPLICIT NONE

*     -------- parameters ---------
*     --- in ---
      INTEGER   I
*     --- in/out ---
      CHARACTER STRING*(*)
*     --------- external ----------
      INTEGER   WSILEN, WSSTAR

*     ------ local variables ------
      CHARACTER S*80
      INTEGER   IS
      INTEGER   IE
      SAVE
*     ------- start module --------

      WRITE(S, '(I20)' )  I

      IS = WSSTAR(S)
      IE = WSILEN(S)

      STRING = S(IS:IE)
      RETURN

      END
```