Note added in proof d.d. 3/04/90:

1) The name of the function NUMBER has been changed to DECCHK.

2) The following routines were added to the library after printing of this documentation:

   CHKTSK, GAMMA, INTGRL, OUTCOM, RDFROM, RDSETS, REAAND, REANOR.

   A description of these routines is given on disk as comments inside the FORTRAN code.

A new version of the documentation of the library is being prepared now and will be ready for publication within a few weeks. This new documentation can be obtained by submitting a request to:

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands.

Note added in proof d.d. 3/04/90:

1) The name of the function NUMBER has been changed to DECCHK.

2) The following routines were added to the library after printing of this documentation:

   CHKTSK, GAMMA, INTGRL, OUTCOM, RDFROM, RDSETS, REAAND, REANOR.

   A description of these routines is given on disk as comments inside the FORTRAN code.

A new version of the documentation of the library is being prepared now and will be ready for publication within a few weeks. This new documentation can be obtained by submitting a request to:

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands.

Note added in proof d.d. 3/04/90:

1) The name of the function NUMBER has been changed to DECCHK.

2) The following routines were added to the library after printing of this documentation:

   CHKTSK, GAMMA, INTGRL, OUTCOM, RDFROM, RDSETS, REAAND, REANOR.

   A description of these routines is given on disk as comments inside the FORTRAN code.

A new version of the documentation of the library is being prepared now and will be ready for publication within a few weeks. This new documentation can be obtained by submitting a request to:

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands.

Note added in proof d.d. 3/04/90:

1) The name of the function NUMBER has been changed to DECCHK.

2) The following routines were added to the library after printing of this documentation:

   CHKTSK, GAMMA, INTGRL, OUTCOM, RDFROM, RDSETS, REAAND, REANOR.

   A description of these routines is given on disk as comments inside the FORTRAN code.

A new version of the documentation of the library is being prepared now and will be ready for publication within a few weeks. This new documentation can be obtained by submitting a request to:

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands.

FORTRAN utility library TTUTIL

C. Rappoldt and D.W.G. van Kraalingen

Internal report 18:
Dept. of Theoretical Production Ecology,
PO Box 430,
6700 AK Wageningen,
The Netherlands,
November, 1989.

List of Internal Reports of Department of Theoretical Production Ecology:

No:

1 D. Barél, F. van Egmond, C. de Jonge, M.J.Frissel, M. Leistra, C.T. de Wit. 1969. Simulatie van de diffusie in lineaire, cylindrische en sferische systemen. Werkgroep: Simulatie van transport in grond en plant.

2 L. Evangelisti and R. van der Weert. 1971. A simulation model for transpiration of crops.

3 J.R. Lambert and F.W.T. Penning de Vries. 1973. Dynamics of water in the soil-plant-atmosphere system: a model named TROIKA.

4 M. Tollenaar. 1971. De fotosynthetische capaciteit.

4a J.N.M. Stricker. 1971. Berekening van de wortellengte per cm^3 grond.

5 L Stroosnijder en H. van Keulen. 1972. Waterbeweging naar de plantenwortel.

6 Th.J.M. Blom and S.R. Troelstra. 1972. A simulation model of the combined transport of water and heat produced by a thermal gradient in porous media.

7a F.W.T. Penning de Vries and H.H. van Laar. 1972. Products and requirements of synthetic processes. Listing of the model and some test runs.

7b F.W.T. Penning de Vries and H.H. van Laar. 1973. Products, requirements and efficiency of biosynthesis. Listing of the model and some test runs.

8 M. Arbab. 1972. A CSMP-program for computing Thornthwaite's classification of climate.

9 P. A. Leffelaar. 1977. A theoretical approach to calculate the anaerobic volume fraction in aerated soil in view of denitrification.

10 L. van Loon and H. Wösten. 1979. A model to simulate evaporation of bare soils in arid regions.

11 C.J.T. Spitters. 1980. Simulating the vegetation dynamics in Sahelian pastures.

12 M. Buil. 1981. Een studie naar de invloed van een aantal diereigenschappen op de primaire en sekundaire productie in de Sahel.

13 D.J. Onstad. 1982. Application of dynamic programming techniques to optimize pest and disease control in winter wheat.

14 P. van de Sanden. 1982. Langgolvige hemelstraling te Niamey.

15 J. Reijerink. 1985. Een simulatie model voor de denitrificatie van NO_3 to N_2 via de intermediaren NO_2 en N_2O.

15a Yan Ying. 1986. A simulation model for dry matter production of maize based on gas exchange measurements of the whole canopy.

16 C. Rappoldt. 1988. Decoding Quantimet 920 binary image files.

17 D.W.G. van Kraalingen. 1989. A three-dimensional light model for crop canopies.

18 C. Rappoldt and D.W.G. van Kraalingen. 1989. FORTRAN utility library TTUTIL.

CONTENTS

INTRODUCTION

Why FORTRAN ?

There are good reasons for the use of FORTRAN in scientific computation. In the first place the language is well standardized and compilers are available on most computers. In the second place there is an overwhelming amount of well-tested subroutines available which are designed to support scientific computation. Well known examples are the IMSL library (1987) containing routines for special functions, matrix calculations, differential equations and much more. More recently the nice and well documented routines from "Numerical Recipes" (Press et.al. 1986) became available to anyone who buys the book. Further, more than 30 libraries of FORTRAN sources are available from NETLIB, a file server reachable via bitnet. Among these are the famous libraries EISPACK and LINPACK on which many routines from IMSL are based.

The use of many of the library routines is based on the so called passed length declaration of arrays in FORTRAN. The dimension(s) of an array can be passed to a subroutine or function as a normal integer argument. Programming languages that do not support passed length declaration are probably not suitable to serious computational work.

Why utilities ?

In practice there are a few ever returning problems with FORTRAN programs. Reading data from files is often done using very specific formats. That tends to make programs and datafiles hard to adapt and hard to maintain. In textbooks the standard advise is to read input data as a character string and to do the decoding in the program. That is a well meant advise but it requires a considerable programming effort. Further, there are always problems with programming output in a convenient form.

During the last three years we solved our own problems with input and output, with character strings and file handling. Each time we paid a little more attention than neccessary for the problem at hand. That resulted in a slowly growing set of subroutines and functions that proved to be usefull in almost any program. We profited enormously of our investment.

Almost all routines went throught a few revisions. We use now the same sources on 4 types of computers: VAX, IBM PC, Atari and Apple MacIntosh (for two exceptions see below). The set of routines became our utility library TTUTIL.

Working on different machines

Higher programming languages exist to enable programmers to write machine independent and readable programs. Large, maintained FORTRAN programs therefore need to be written in standard FORTRAN-77. We regard as very bad practice the use of, for instance, VAX-FORTRAN language extensions in simulation models. Further, experience learns that compilers running on micro computers are not always as clever as the VAX-FORTRAN compiler. "Difficult" constructions like function calls and string concatenation in subroutine calls sometimes lead to problems. Further, the different compilers all have their own deficiencies. A few examples:

- An IBM does not overwrite an existing file, when the file is opened with status 'NEW'
- The STOP statement on the Apple Macintosh leads to the disappearance of the output window. That implies that fatal error messages cannot be written in the form STOP 'message'.

In the 41 subroutines and functions of TTUTIL we use a conservative programming style. Nested constructions in a single statement are omitted (they are not very clear anyway) and we "programmed around" the compiler deficiencies we met. The subroutines and functions all run on ATARI with the Prospero compiler, on IBM with Microsoft compiler, on Apple MacIntosh with the Absoft compiler and of course on the VAX using the VAX-FORTRAN compiler.

Classification of TTUTIL routines

Below is a rough classification of the TTUTIL routines. Note that some routines occur in more that a single class. In some cases library routines call each other, and in the headers of the routines this has been indicated.

- Interactive input

ENTCHA, ENTDCH, ENTDIN, ENTDRE, ENTINT, ENTREA

- String handling

EXTENS, IFINDC, ILEN, ISTART, NUMBER, REMOVE, STRIP, UPPERC, WORDS

- Decoding of strings to values

DECINT, DECREA, DECREC, NUMBER

- Files

EXTENS, FOPEN

- AFGEN tables

AFINVS, LINT, PLTFUN, PLTHIS

- Input from file

GETCH, GETREC, MOFILP, RDAREA, RDDATA, RDINDX, RDINIT, RDSINT, RDSREA

- Output to file

PLTFUN, PLTHIS, OUTDAT, OUTPLT

- VT100 screen

CLS, POS

- Random numbers

BOXMUL, UNIFL

- Other

ERROR, INSW, LIMIT

Handling of fatal errors

Fatal errors are dealt with by calling the subroutine **ERROR**. That subroutine writes a message to the screen, requires a <Return> from the keyboard and then terminates program execution. That procedure leads to the same result on all machines we know. A STOP statement with a message does not work on an Apple MacIntosh since the output screen disppears on the execution of the STOP statement. Users of TTUTIL can use subroutine ERROR also for their own fatal errors (see the header of **ERROR**).

Non-standard statements and machine dependencies

- The routines **CLS** and **POS** are meant to handle output to a VT100 terminal or to the screen of an IBM PC. They are not meant as general output routines.
- To hold the cursor after writing a question to the screen a "$" is used in **ENTCHA**, **ENTDCH**, **ENTDIN**, **ENTDRE**, **ENTINT**, **ENTREA**, **FOPEN** and **POS**. When the syntax needs to be stricly standard or when the routines are used on an ATARI the "$" needs to be removed. They can be found be searching for the string "A$".
- In **RDDATA** the integer **PARAMETER IWLEN** describes the length in bytes of the words used in records of unformatted direct access files. On the microcomputers ATARI, IBM and Apple this is 1 byte per word (the words are just bytes). On a VAX, however, that is 4 bytes per word. The microcomputer version will run on the VAX, but the temporary files created by the **RDDATA** and related routines will be 4 times as large as needed. A true VAX version (IWLEN=4) won't run on microcomputers, however. No error messages are given and results are unpredictable. Since the temporary files will under most circumstances be small, the microcomputer version is included in the library.

HEADERS OF THE TTUTIL LIBRARY ROUTINES

```
      SUBROUTINE  AFINVS  (FUN,ILF,FUNINV,ILFI,EXIST)

*     Inversion of AFGEN/LINT table FUN by reversing the
*     role of X and Y points and inverting the order of the
*     points when FUN is decreasing.
*     The array lengths in this routine are ILF and ILFI.
*     They can be declared longer, however, in the calling
*     program. Only part of the table is then inverted.
*     The (double) numbers of points ILF and ILFI should
*     be equal. The array FUN from 1 to ILF must describe
*     a table with an existing inverse (checked !!).
*     Note that the actual arguments FUN and FUNINV are
*     allowed to be the same arrays. The table is then
*     inverted without using extra memory

*     FUN    - "AFGEN" table as defined by routine LINT          I
*     ILF    - size of FUN                                      I
*     FUNINV - inverse table                                    O
*     ILFI   - size of FUNINV array (= ILF)                     I
*     EXIST  - Flags the existence of the inverse               O
*              = .TRUE. the inverse exists and is returned
*              = .FALSE. no inverse exists, FUNINV unchanged
*
*     Subroutine called:
*      - from library TTUTIL: ERROR

*     Author: Kees Rappoldt
*     Date  : Oct 89


*     formal parameters
      INTEGER ILF,ILFI
      REAL FUN,FUNINV
      DIMENSION FUN(2,ILF/2),FUNINV(2,ILFI/2)
      LOGICAL EXIST
```

```
      REAL FUNCTION BOXMUL()

*     Generate unit normal deviate by Box-Muller method

*     BOXMUL - pseudo-random standard normal deviate                  O
*
*     The Box-Muller method is based on inversion. No inverse
*     exists of the distribution of a single normal variate.
*     However SQRT(X**2+Y**2) with X and Y both being normal
*     variates, has a distribution which can be inverted.
*
*     This enables elegant generation of normal variates
*     in combination with a generator for uniform variates
*     on (0,1). The commonly used linear congruential
*     generators, however, lead to pathological behaviour
*     (see section 6.7.3 in Bratley (1987)). This is not
*     the case with the generator UNIFL used in this program.
*
*     References:
*     Box, G.E.P., and M.E.Muller. (1958). A note on the
*         generation of random normal deviates.
*         Ann.Math.Stat. 29:610-611.
*     Bratley,P., B.L.Fox and L.E.Schrage. 1983. A guide to
*         simulation. Springer-Verlag New York Inc. 397 pp.
*
*     Subroutines and/or functions called:
*      - from library TTUTIL: UNIFL

*     Author: Kees Rappoldt
*     Date  : October 1989
```

```
      SUBROUTINE  CLS

*     This subroutine clears the screen (IBM screen)
*
*     Author: Daniel van Kraalingen
*     Date:   Oct 1989
*     No subroutines called
```

```
      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 spaces !           I
*     IVALUE - Integer value read from string                          O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: NUMBER

*     Author: Kees Rappoldt
*     Date  : Sept 89


*     formal parameters
      INTEGER IWAR,IVALUE
      CHARACTER*(*) STRING
```

```
      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 spaces !          I
*     VALUE  - Real value read from string                            O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: NUMBER

*     Author: Kees Rappoldt
*     Date  : Sept 89


*     formal parameters
      INTEGER IWAR
      REAL VALUE
      CHARACTER*(*) STRING
```

```
      SUBROUTINE  DECREC  (RECORD,ILX,X)

*     Finds and decodes from the character string RECORD
*     (the first) ILX real numbers.
*     Numbers are separated by space(s) and/or comma(s).

*     RECORD - character string ; numbers max 20 characters          I
*     ILX    - number of REAL numbers to be decoded                  I
*     X      - array containing the results                          O
*
*     Subroutines and functions called:
*     - from library TTUTIL: DECREA, ERROR, NUMBER

*     Author: Kees Rappoldt
*     Date: Sept 87, revised Oct 89


*     formal parameters
      INTEGER ILX
      CHARACTER*(*) RECORD
      REAL X
      DIMENSION X(ILX)
```

```
      SUBROUTINE  ENTCHA  (QUEST,X)

*     Interactive entry of a character string.
*     Writes the text QUEST on screen as a "question" and
*     returns entered string to calling program.

*     QUEST - character string for instance 'name'                  I
*     X     - entered character string                              O

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameters
      CHARACTER QUEST*(*),X*(*)
```

```
      SUBROUTINE  ENTDCH  (QUEST,SDEF,S)

*     Interactive entry of a CHARACTER string with a default.
*     Writes the text QUEST on screen as a "question" and
*     returns entered string to calling program.

*     QUEST - character string for instance 'waarde van P'          I
*     SDEF  - default string, assumed when <Return> is given        I
*     S     - entered CHARACTER string                              O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, ISTART

*     Author: Kees Rappoldt
*     Date: October 1989


*     formal parameters
      CHARACTER*(*) QUEST,SDEF,S
```

```
      SUBROUTINE  ENTDIN  (QUEST,IXDEF,IX)

*     Interactive entry of an INTEGER number with a default.
*     Writes the text QUEST on screen as a "question" and
*     returns entered number to calling program.

*     QUEST - character string for instance 'waarde van P'          I
*     IXDEF - default value assumed when <Return> is given          I
*     IX    - entered INTEGER number                                O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, ISTART

*     Author: Kees Rappoldt
*     Date: May 1989, revised October 1989


*     formal parameters
      INTEGER IXDEF,IX
      CHARACTER*(*) QUEST
```

```
      SUBROUTINE  ENTDRE  (QUEST,XDEF,X)

*     Interactive entry of a REAL number with a default. Writes the
*     text QUEST on screen as a "question" returns entered number.

*     QUEST - character string for instance 'waarde van P'          I
*     XDEF  - default value assumed when <Return> is given          I
*     X     - entered REAL number                                   O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN

*     Author: Kees Rappoldt
*     Date: August 1987, revised March 1988, October 1989


*     formal parameters
      REAL XDEF,X
      CHARACTER*(*) QUEST
```

```
      SUBROUTINE  ENTINT  (QUEST,IX)

*     Interactive entry of an INTEGER number
*     Writes the text QUEST on screen as a "question" and
*     returns entered number to calling program.

*     QUEST - character string for instance 'aantal iteraties'      I
*     IX    - entered number                                        O

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameters
      INTEGER IX
      CHARACTER QUEST*(*)
```

```
      SUBROUTINE  ENTREA  (QUEST,X)

*     Interactive entry of a REAL number.
*     Writes the text QUEST on screen as a "question" and
*     returns entered number to calling program.

*     QUEST - character string for instance 'waarde van P'          I
*     X     - entered REAL number                                   O

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameters
      REAL X
      CHARACTER QUEST*(*)
```

```
      SUBROUTINE  ERROR  (MODULE,MESSAG)

*     This routine 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

*     Author: Daniel van Kraalingen
*     Date:   Oct 1989
*     No subroutines called


*     formal parameters
      CHARACTER*(*) MODULE, MESSAG
```

```
      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 is behind
*     a point (.). A point before a bracket (]) is neglected (VAX).
*     The input filename does not necessarily have an extension.

*     FILEIN - Input name with old extension 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 function called:
*     - from library TTUTIL: ERROR, ILEN, UPPERC

*     Author: Kees Rappoldt
*     Date: March 1988, revised October 1989


*     formal parameters
      INTEGER ICHECK
      CHARACTER*(*) FILEIN,FILEOU,NEWEXT
```

```
      SUBROUTINE  FOPEN  (IUNIT,FILE,STATUS,PRIV)

*     This subroutine opens a sequential, formatted file
*     after doing an inquiry about the existence

*     IUNIT  - unit number to be used                                  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
```

```
      SUBROUTINE  GETCH  (IUNIT,INCHAR,CHARS,EOF)

*     Reads INCHAR characters from a sequential file.
*     the routine resets itself during the first CALL, on an
*     End_Of_File condition and when a new unit number is used.

*     IUNIT  - file unit                                          I
*     INCHAR - number of characters to be read                    I
*     CHARS  - characters read from file                          O
*     EOF    - flags end of file ; CHARS may not be complete      O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: ERROR, ILEN

*     Author: Kees Rappoldt
*     Date: Jan 1989


*     formal parameters
      INTEGER IUNIT,INCHAR
      CHARACTER*(*) CHARS
      LOGICAL EOF
```

```
      SUBROUTINE  GETREC  (IUNIT,RECORD,EOF)

*     Reads record from an open file skipping comment lines.
*     Comment lines have an asterisk (*) in their first column.

*     IUNIT  - unit number              I
*     RECORD - returned record          O
*     EOF    - End_Of_File found        O

*     Author: Kees Rappoldt, Daniel van Kraalingen
*     Date  : October 1989
*     No subroutines and/or functions are used


*     formal parameters
      INTEGER IUNIT
      CHARACTER RECORD*(*)
      LOGICAL EOF
```

```
      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  : Sept 89


*     formal parameters
      INTEGER ILDEC, IST, IEND
      CHARACTER*(*) NAMLIS, NAME
      DIMENSION NAMLIS(ILDEC)
```

```
      INTEGER FUNCTION ILEN (STRING)

*     This function determines the significant length of
*     a string, if the string is empty a zero is returned

*     ILEN   - returned length          O
*     STRING - input string             I

*     Author: Daniel van Kraalingen
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameter
      CHARACTER*(*) STRING
```

```
      REAL FUNCTION INSW (X1,X2,X3)

*     Input switch depending on sign of X1, this
*     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

*     Author: Daniel van Kraalingen
*     Date  : Oct 89
*     No subroutines and/or functions used


*     formal parameters
      REAL X1,X2,X3
```

```
      INTEGER FUNCTION ISTART (STRING)

*     This function determines the first significant character
*     character of a string, if the string contains no characters,
*     a zero is returned.

*     ISTART - returned value          O
*     STRING - input string            I

*     Author: Daniel van Kraalingen
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameter
      CHARACTER*(*) STRING
```

```
      REAL FUNCTION LIMIT (MIN,MAX,X)

*     Returns value of X limited on the interval [MIN,MAX]

*     MIN - interval lower boundary    I
*     MAX - interval upper boundary    I
*     X   - Argument of function       I

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions used


*     formal parameters
      REAL MIN,MAX,X
```

```
*-----------------------------------------------------------------------*
*  REAL FUNCTION LINT                                                   *
*  Authors: Daniel van Kraalingen                                       *
*  Version: 2                                                           *
*  Purpose: This function is a linear interpolation function.  The      *
*           function does not extrapolate : in case of X below or       *
*           above the region defined by TABLE, the first                *
*           respectively the last Y-value is returned and a message     *
*           is generated.                                               *
*  Keywords:Utility, linear interpolation                               *
*                                                                       *
*  FORMAL PARAMETERS:  (I=input,O=output,C=control,IN=init,T=time)      *
*  name   type meaning                                      units  class *
*  ----   ---- -------                                      -----  ----- *
*  LINT    R4  function name, result of the interpolation     =      O   *
*  TABLE   R4  A one-dimensional array with paired            =      I   *
*              data: x,y,x,y, etc.                                      *
*  ILTAB   I4  The number of elements of the array            -      I   *
*              TABLE                                                    *
*  X       R4  The value at which interpolation should        =      I   *
*              take place                                               *
*                                                                       *
*  FATAL ERROR CHECKS (execution terminated, message):                  *
*  TABLE(I) < TABLE(I-2) , for I odd                                    *
*  ILTAB odd                                                            *
*                                                                       *
*  No WARNINGS using the control variable IWAR are generated since      *
*  nobody will check IWAR after each LINT call ; instead an error text  *
*  and table information is written to the screen (number of points and *
*  the table contents).                                                 *
*                                                                       *
*  SUBROUTINES called: ERROR                                            *
*  No FILE's are used (error message with WRITE(*,...)... )             *
*-----------------------------------------------------------------------*
      REAL  FUNCTION  LINT  (TABLE,ILTAB,X)

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

```
      SUBROUTINE  MOFILP  (IUNIT)

*     This subroutine moves the filepointer across comment
*     lines of data files and puts the file pointer at the
*     first non-comment record.
*     Comment lines have an asterisk (*) in their first column.

*     IUNIT - unit number of file      I

*     Author: Daniel van Kraalingen
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameter
      INTEGER IUNIT
```

LOGICAL FUNCTION NUMBER (STRING)

```
*     Checks if the string STRING is a number.

*     NUMBER --> .TRUE. if the string is a number                    O
*     STRING - input string, NO trailing or leading spaces !!!!      I

*     Author: Kees Rappoldt
*     Date  : Sept 89
*     No subroutines and/or functions called


*     formal parameter
      CHARACTER*(*) STRING
```

```
      SUBROUTINE OUTDAT (ITASK, IUNIT, RN, R)

*     This subroutine can be used to generate output files from
*     within simulation models. The routine 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 the
*     routine. Output can be generated by a call with ITASK=4, 5, or 6.
*     These are table output, spreadsheet output and two column output.

*     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. 2 stores the
*             name and value of the variable in the 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 - integer, number used for temporary file, IUNIT+1
*             is used for output file.
*     RN    - string, name of variable, (only six 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 six characters).
*     R     - value of variable (only effective at ITASK=2).

*     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,'DTGA',DTGA) 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.

*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, FOPEN, IFINDC, ERROR, UPPERC

*     Author: Daniel van Kraalingen, Kees Rappoldt
*     Date: Oct 89

*     formal parameters
      INTEGER ITASK, IUNIT
```

```
REAL R
CHARACTER*(*) RN
```

```
      SUBROUTINE  OUTPLT  (ITASK,  RN)

*     this subroutine is meant to be used in conjunction with
*     OUTDAT. OUTDAT should be used to write variable name
*     and value to a temporary file, OUTPLT can be used to
*     printplot a selection of the stored variables. By repeated
*     calls to the subroutine with ITASK=1, variable names can be
*     given to the subroutine for which the plot is wanted. By
*     a call with ITASK=4, 5, 6, or 7, printplots can be made with
*     a width of 80 or 120 characters, and 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 the variable name for
*             use in the printplot. 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,
*     RN    - string, name of variable, only six characters will be
*             used. The value of the variable should have been stored
*             by previous calls to OUTDAT.

*     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

*     Subroutines and/or functions called:
*     - from library TTUTIL: ILEN, FOPEN, IFINDC, ERROR, UPPERC

*     Author: Daniel van Kraalingen
*     Date: Oct 89


*     formal parameters
      INTEGER ITASK
      CHARACTER*(*) RN
```

```
      SUBROUTINE  PLTFUN  (IUNIT,FUN,ILF,ILINE,IMARK,FORM,LEGEND)

*     writes AFGEN/LINT table points to file in TTPLOT format
*        !!!!!!!!!!!! NO CHECKS ARE CARRIED OUT !!!!!!!!!!

*     IUNIT  - unit number of open file                              I
*     FUN    - "AFGEN" table as defined by routine LINT              I
*     ILF    - array length of FUN                                   I
*     ILINE  - line type (see TTPLOT documentation)                  I
*     IMARK  - marker type (see TTPLOT documentation)                I
*     FORM   - data pair FORMAT ; CHARACTER string bv. '(2F8.3)'     I
*     LEGEND - legenda text ; CHARACTER string                       I
*
*     Subroutine called:
*      - from library TTUTIL: ERROR

*     Author: Kees Rappoldt
*     Date  : Oct 89


*     formal parameters
      INTEGER IUNIT,ILF,IMARK,ILINE
      REAL FUN
      DIMENSION FUN(2,ILF/2)
      CHARACTER*(*) FORM,LEGEND
```

```
      SUBROUTINE  PLTHIS  (IUNIT,HIST,ILH,ILINE,IMARK,FORM,
     &                     LEGEND,IDOWN)

*     Writes AFGEN/LINT table points to file in TTPLOT format
*     The table is drawn as a HISTOGRAM using the X values of
*     the table as mid-class points.

*     IUNIT  - unit number of open file                              I
*     HIST   - "AFGEN" table as defined by routine LINT              I
*     ILH    - array length of HIST                                  I
*     ILINE  - line type (see TTPLOT documentation)                  I
*     IMARK  - marker type (see TTPLOT documentation)                I
*     FORM   - data pair FORMAT ; CHARACTER string bv. '(2F8.3)'     I
*     LEGEND - legenda text ; CHARACTER string                       I
*     IDOWN  =1 --> right side of interval is connected to x-axis    I
*            =0 --> this is not done
*
*     Subroutine called:
*      - from library TTUTIL: ERROR

*     Author: Kees Rappoldt
*     Date  : Oct 89


*     formal parameters
      INTEGER IUNIT,ILH,ILINE,IMARK,IDOWN
      REAL HIST
      DIMENSION HIST(2,ILH/2)
      CHARACTER*(*) FORM,LEGEND
```

```
      SUBROUTINE  POS  (IX,IY,TEXT)

*     Positions text on VT100 screen

*     IX   - X-position of cursor (# of columns)                    I
*     IY   - Y-position of cursor (# of rows)                       I
*     TEXT - Text to be written on the screen                       I
*
*     Functions called:
*      - from library TTUTIL: ILEN, ISTART

*     Author: Daniel van Kraalingen
*     Date  : Oct 89


*     formal parameters
      INTEGER IX,IY
      CHARACTER*(*) TEXT
```

```
      SUBROUTINE  RDAREA  (XNAME,X,ILDEC,IFND)

*     Reads an array of REAL values from a datafile. The datafile
*     should be initialized with RDINIT. See header of RDINDX
*     for information on datafile syntax.

*     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: DECINT, DECREA,  ERROR, EXTENS,  FOPEN,
*                            IFINDC,   ILEN, NUMBER, RDDATA, RDINDX,
*                            UPPERC

*     Author: Kees Rappoldt
*     Date  : October 1989


*     formal parameters
      INTEGER ILDEC,IFND
      REAL X
      DIMENSION X(ILDEC)
      CHARACTER*(*) XNAME
```

```
      SUBROUTINE  RDDATA(ITASK,IUNIT,IULOG,DATFIL,
     $                   XNAME,X,NDEC,NREQ)

*     Looks for a variable name on a datafile and reads the data
*     behind it ; for syntax of data file see header RDINDX.FOR.

*     ITASK  - =1, Initialization, An index of the datafile is made
*                  and all values are stored on a *.TMP file.
*              =2, Returns results.
*     IUNIT  - Unit number used for the .TMP direct access file      I
*              in which all values are written.
*              IUNIT+1 is temporarily used for the datafile itself
*     IULOG  - >0, Unit number of logfile used for datafile syntax   I
*                  errors. When not opened RDINDX.LOG is created.
*              =0, Nothing is done with a logfile
*     DATFIL - Name of datafile                                      I
*     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   - >0, Requested number of values, should match        I/O
*                  the number present on file (checked !!)
*              =0, The number of values on file is returned
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECINT, DECREA, RDINDX,  ERROR, EXTENS,
*                            FOPEN, IFINDC,   ILEN, NUMBER, UPPERC

*     Author: Kees Rappoldt
*     Date  : Sept 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,NDEC,NREQ
      REAL X
      DIMENSION X(NDEC)
      CHARACTER*(*) DATFIL,XNAME
```

```
      SUBROUTINE  RDINDX  (IUNIT,TOSCR,TOLOG,IUL,DATFIL,XBUF,
     $                     ILBUF,IWLEN,NAMLIS,INDPNT,ILIND,INFND)

*     Produces index of datafile. The index consits 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 for random access file              I
*              IUNIT+1 used for datafile, closed after reading
*     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
*              when not exists, RDINDX.LOG is created
*     DATFIL - name of datafile                                    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
*     INDPNT - points to value in temporary file                   O
*     ILIND  - declared size of index                              I
*     INFND  - number of variable names found                      O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECINT, DECREA,  ERROR, EXTENS, FOPEN,
*                            IFINDC,   ILEN, NUMBER, UPPERC

*     Author: Kees Rappoldt
*     Date  : Sept 1989


*     Datafile consists of "fields", separated by semi-colon or on
*     different lines. A field may be continuated on a next line. A name
*     or number, however, should not be broken up. Continuing a series
*     of numbers, commas should be used before going to the next line.
*     In all lines comment may be written behind an exclamation mark !
*     Lines beginning with a star * are neglected. Maximum name length
*     is set by the variable type of NAMLIS.
*
*          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)
```

```
*       formal parameters
        INTEGER IUNIT,IUL,ILBUF,IWLEN,INDPNT,ILIND,INFND
        REAL XBUF
        CHARACTER*(*) DATFIL,NAMLIS
        DIMENSION NAMLIS(ILIND),INDPNT(0:ILIND),XBUF(ILBUF)
        LOGICAL TOSCR,TOLOG
```

```
      SUBROUTINE  RDINIT  (IUNIT,IULOG,DATFIL)

*     Initializes subroutine RDDATA for reading data
*     from a user supplied filename. See header of RDINDX
*     for information on datafile syntax.

*     IUNIT  - Unit number used for the direct access file in which  I
*              all values are written.
*              IUNIT+1 is temporarily used for the datafile itself
*     IULOG  - >0, Unit number of logfile used for datafile syntax   I
*                  errors. When not opened RDINDX.LOG is created.
*              =0, Nothing is done with a logfile
*     DATFIL - Name of datafile                                      I
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECINT, DECREA,  ERROR, EXTENS,  FOPEN,
*                            IFINDC,   ILEN, NUMBER, RDDATA, RDINDX,
*                            UPPERC

*     Author: Kees Rappoldt
*     Date  : October 1989


*     formal parameters
      INTEGER IUNIT,IULOG
      CHARACTER*(*) DATFIL
```

```
      SUBROUTINE  RDSINT  (XNAME,IX)

*     Reads a single INTEGER value from a datafile. Datafile
*     should be initialized with RDINIT. See header of RDINDX
*     for information on datafile syntax.
*     Note that reading very large integer values may cause
*     accuracy problems, since all numbers are buffered as
*     floating point numbers.

*     XNAME  - Name of variable        I
*     IX     - Variable itself         O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECINT, DECREA,  ERROR, EXTENS,  FOPEN,
*                            IFINDC,   ILEN, NUMBER, RDDATA, RDINDX,
*                            UPPERC

*     Author: Kees Rappoldt
*     Date  : October 1989


*     formal parameters
      INTEGER IX
      CHARACTER*(*) XNAME
```

```
      SUBROUTINE  RDSREA  (XNAME,X)

*     Reads a single REAL value from a datafile. Datafile
*     should be initialized with RDINIT. See header of RDINDX
*     for information on datafile syntax.

*     XNAME  - Name of variable        I
*     X      - Variable itself         O
*
*     Subroutines and/or functions called:
*     - from library TTUTIL: DECINT, DECREA,  ERROR, EXTENS,  FOPEN,
*                            IFINDC,   ILEN, NUMBER, RDDATA, RDINDX,
*                            UPPERC

*     Author: Kees Rappoldt
*     Date  : October 1989


*     formal parameters
      REAL X
      CHARACTER*(*) XNAME
```

```
      SUBROUTINE  REMOVE  (STRING,CHR)

*     This subroutine removes all unwanted characters (CHR)
*     from a string (STRING), and replaces them with a <space>

*     STRING - string that is used     I,O
*     CHR    - character to be removed  I

*     Author: Daniel van Kraalingen
*     Date  : Oct 89
*     No subroutines and/or functions used


*     formal parameters
      CHARACTER STRING*(*),CHR*1
```

```
      SUBROUTINE  STRIP  (STRING,CHR)

*     This subroutine strips unwanted characters (CHR)
*     from a string (STRING). The right hand side of the
*     string is adjusted i.e. if "e" is stripped from
*     "bicentennial", the result is "bicntnnial"

*     STRING - string that is used      I,O
*     CHR    - character to be removed  I
*
*     Function called:
*      - from library TTUTIL: ILEN

*     Author: Daniel van Kraalingen
*     Date  : Oct 89


*     formal parameters
      CHARACTER STRING*(*),CHR*1
```

```
      REAL FUNCTION UNIFL()

*     High quality pseudo random number generator.

*     UNIFL - pseuso-random uniformly distributed variate                O
*
*     This generator is of the so called combined type. This
*     generator does not behave pathologically with the Box-Muller
*     method for the generation of normal variates, as do the
*     commonly used linear congruential generators (see also
*     comments in FUNCTION BOXMUL).
*
*     The algorithm is:  X(i+1) = 40014 * X(i) mod (2147483563)
*                        Y(i+1) = 40692 * Y(i) mod (2147483399)
*                        Z(i+1) = (X(i+1)+Y(i+1)) mod (2147483563)
*     The random number returned is constructed dividing Z by its
*     range. The period of the generator is about 2.30584E+18.
*
*     The program below has been written using a program UNIFL in
*     Bratley et.al. (1987). A logical INIT has been added in order
*     to include seeds in the program (implicit initialization). The
*     algorithm originates from L'Ecuyer (1986). In Bratley (page 332)
*     more information can be found on seeds and periods of X and Y.
*
*     References:
*     Bratley,P., B.L.Fox, L.E.Schrage. 1983. A guide to simulation
*         Springer-Verlag New York Inc. 397 pp.
*     L'Ecuyer,P. (1986). Efficient and portable combined pseudo-
*         random number generators. Commun. ACM (to appear).

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called
```

```
      SUBROUTINE  UPPERC  (STRING)

*     Converts string to uppercase characters

*     STRING - character string        I

*     Author: Daniel van Kraalingen, Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameter
      CHARACTER*(*) STRING
```

```
      SUBROUTINE  WORDS  (RECORD,ILW,SEPARS,IWBEG,IWEND,IFND)

*     Returns position of start and end of the (first) ILW words
*     in string RECORD. Valid separators are all the characters
*     present in string SEPARS, for instance ' ,' or '/ ,+=-'.

*     RECORD - character string                                  I
*     SEPARS - string containing separator characters            I
*     ILW    - number of words to be found                       I
*     IWBEG  - integer array containing begin positions          O
*     IWEND  - integer array containg end positions              O
*     IFND   - integer containing the number of words            O

*     Author: Kees Rappoldt
*     Date  : Oct 89
*     No subroutines and/or functions called


*     formal parameters
      INTEGER ILW,IWBEG,IWEND,IFND
      DIMENSION IWBEG(ILW),IWEND(ILW)
      CHARACTER*(*) RECORD,SEPARS
```

EXAMPLE PROGRAMS

```
      PROGRAM  EXAMP1

*     test subroutine ASTRO

      IMPLICIT REAL (A-Z)

*     initial defaults
      DAY = 1.0
      LAT = 52.5

10    CONTINUE

*     interactive input
      CALL ENTDRE ('Dagnummmer',DAY,DAY)
      CALL ENTDRE ('Breedte in graden',LAT,LAT)

*     call to tested routine
      CALL ASTRO (DAY,LAT,DAYL,DAYLP,SINLD,COSLD)

*     write results
      WRITE (*,'(1X,A,F8.3,3(/,1X,A,F8.3))')
     $ 'DAYL  = ',DAYL,
     $ 'DAYLP = ',DAYLP,
     $ 'SINLD = ',SINLD,
     $ 'COSLD = ',COSLD

      GOTO 10
      END

      INCLUDE HD40: FORTRAN:TTLIB:ASTRO.FOR
```

```
      PROGRAM  EXAMP2

*     An example of data transformation
*     The user specifies two colums of an input file
*     The two columns need not to exist on all lines
*     When they exist they should contain numbers
*     The numbvers are written to an ouytput file as
*     an X-value and an Y-value. A linear transformation
*     may be applied to the Y-values.

*     declarations
      INTEGER IWMAX,IFND,IWB,IWE,IX,IY,IWARX,IWARY
      PARAMETER (IWMAX=100)
      DIMENSION IWB(IWMAX),IWE(IWMAX)
      REAL X,Y,A,B
      CHARACTER MESSAG*40,FILIN*80,FILOUT*80,RECORD*80
      LOGICAL EOF

*     ask name input file
      CALL ENTCHA ('Filename',FILIN)
      CALL ENTDIN ('Column X',1,IX)
      CALL ENTDIN ('Column Y',2,IY)
      CALL ENTDRE ('Transform Y as A*Y+B, value of A',1.0,A)
      CALL ENTDRE ('And value of B',0.0,B)

*     change extension into 'OUT'
      CALL EXTENS (FILIN,'OUT',1,FILOUT)

*     open files
      CALL FOPEN (40,FILIN,'OLD','NVT')
      CALL FOPEN (41,FILOUT,'NEW','UNK')

*     read until EOF reached
20    CONTINUE
      CALL GETREC (40,RECORD,EOF)
      IF (.NOT.EOF) THEN

*        handle non-comment line
         CALL WORDS (RECORD, IWMAX, ' ,;=', IWB, IWE, IFND)

         IF (IFND.GE.IX .AND. IFND.GE.IY) THEN
*           decode IX-th and IY-th word
            CALL DECREA (IWARX, RECORD(IWB(IX):IWE(IX)), X)
            CALL DECREA (IWARY, RECORD(IWB(IY):IWE(IY)), Y)

            IF (IWARX.EQ.0 .AND. IWARY.EQ.0) THEN
*              correct values and write result to output
               WRITE (41,'(1X,2F12.4)') X, A*Y + B
```

```
            ELSE
*              error
               IF (IWARX.NE.0) MESSAG =
     $          RECORD(IWB(IX):IWE(IX))//' is not a number'
               IF (IWARY.NE.0) MESSAG =
     $          RECORD(IWB(IY):IWE(IY))//' is not a number'
               CALL ERROR ('EXAMP2',MESSAG)
            END IF
         END IF

      GOTO 20
      END IF

      STOP
      END
```

```
      PROGRAM  EXAMP3

*     Generates a scatter plot for Y = A * X + B + (error term)
*     The error term has a normal distribution with mean 0.0 and
*     standard deviation supplied by the user.
*     The x-values are ramdomly chosen on a user supplied interval.
*     The output file contains the calculated (X,Y) pairs with
*     the X-values in increasing order. A TTPLOT header is written
*     above the list of (X,Y) pairs.

*     declarations
      INTEGER I,INDX,N,NMAX
      REAL A,B,BOXMUL,SIGMA,UNIFL,X,Y
      PARAMETER (NMAX=10000)
      DIMENSION X(NMAX),Y(NMAX),INDX(NMAX)


*     ask input
      CALL ENTDRE ('Lower bound X interval', 4.0, X1)
      CALL ENTDRE ('Upper bound X interval', 10., X2)

      CALL ENTDRE ('Equation A*X+B, value of A',0.9,A)
      CALL ENTDRE ('and value of B',-3.0,B)

      CALL ENTDRE ('Standard deviation of error term',0.5,SIGMA)
      CALL ENTDIN ('Number of points',1000,N)

*     error check
      IF (N.GT.NMAX) CALL ERROR ('EXAMP3','N > NMAX')


*     generate points
      DO 10 I=1,N
         X(I) = X1 + (X2-X1) * UNIFL()
         Y(I) = A * X(I)  +  B  +  SIGMA * BOXMUL()
10    CONTINUE

*     sort points with respect to X value
      CALL INDEXX (N,X,INDX)


*     write results to file ; open file, write TTPLOT header
      CALL FOPEN (40,'RES.DAT','NEW','UNK')
      WRITE (40,'(A,9(/,A))')
     $ ' * Generated by EXAMP3.FOR',
     $ ' Straight line with error term',
     $ ' X 1 0 4 10 1 0.5 0 0',
     $ ' x-axis',
```

```
     $ ' Y 1 0 -5 10 5 1 0 0',
     $ ' y-axis',
     $ ' 0 0',
     $ ' *',
     $ ' 1 0 9',
     $ ' using BOXMUL'

      DO 20 I=1,N
         J = INDX(I)
         WRITE (40,'(1X,2F10.5)') X(J),Y(J)
20    CONTINUE

      STOP
      END

*     This include statement is used on the Apple MacIntosh.
*     On other machines this routine should be linked or so
      INCLUDE HD40: FORTRAN:RECIPES:INDEXX.FOR
```

```
      PROGRAM  EXAMP4

*     Checks partitioning factors on a plant datafile
*     Writes results to a TTPLOT file.

*     declarations
      INTEGER I,CROPT,ILMAX,ILFL,ILFS,ILFO,ILF
      REAL EFF,FLTB,FSTB,FOTB,LINT,FRL,FRS,FRO,TOT,SUM,DVS,FUNTB
      PARAMETER (ILMAX=100)
      DIMENSION FLTB(ILMAX),FSTB(ILMAX),FOTB(ILMAX),SUM(ILMAX)
      DIMENSION FUNTB(ILMAX)
      LOGICAL EXIST


*     initialize data reading
      CALL RDINIT (30,40,'Barley.dat')

*     read croptype, for demonstration only
      CALL RDSINT ('CROPT',CROPT)

*     read phoptosynthesis efficiency, for demonstration only
      CALL RDSREA ('EFF',EFF)

*     read partionioning tables
      CALL RDAREA ('FLTB',FLTB,ILMAX,ILFL)
      CALL RDAREA ('FSTB',FSTB,ILMAX,ILFS)
      CALL RDAREA ('FOTB',FOTB,ILMAX,ILFO)
      CLOSE (30,STATUS='DELETE')


*     check partitioning tables
      DO 10 I = 1, ILMAX/2
         DVS = 2.0 * FLOAT(I-1)/FLOAT(ILMAX/2-1)

         FRL = LINT (FLTB,ILFL,DVS)
         FRS = LINT (FSTB,ILFS,DVS)
         FRO = LINT (FOTB,ILFO,DVS)
         TOT = FRL + FRS + FRO

*        check
         IF (ABS(TOT-1.0).GT.0.001) WRITE (*,'(1X,A,F4.2)')
     $     'Partitioning error at DVS=',DVS

*        get sum table
         SUM(2*I-1) = DVS
         SUM(2*I)   = TOT
10    CONTINUE
```

```
*     write tables to TTPLOT plot file
      CALL FOPEN (50,'RES.DAT','NEW','UNK')
      WRITE (50,'(A,6(/,A))')
     $ ' * Generated by EXAMP4.FOR',
     $ ' Partitioning check',
     $ ' X 1 0 0 2 0.5 0.1 0 0',
     $ ' DVS',
     $ ' Y 1 0 0 1.4 0.2 0.1 0 0',
     $ ' Fraction',
     $ ' 0.1 1.3'

      CALL PLTFUN (50,FLTB,ILFL,2,7,'(1X,F4.2,F8.3)','leaves')
      CALL PLTFUN (50,FSTB,ILFS,3,8,'(1X,F4.2,F8.3)','stems')
      CALL PLTFUN (50,FOTB,ILFO,5,1,'(1X,F4.2,F8.3)','storage organs')
      CALL PLTFUN (50,SUM,ILMAX,1,0,'(1X,2F8.3)','total')

      STOP
      END

*
* Datafile belonging to Example 4
* ==============================
* Example of data file that can be read by the
* routines RDSINT, RDSREA and RDAREA.
*
* Variable names are followed by one or more values
* separated by a comma. Repetition of the same value
* may be coded by using 100*8.3, for instance (without
* spaces around the "*"). See the file below and
* a formal description in the header of RDINDX.
*
* Crop characteristics of Barley (Gerrie van de Ven, Egypt)
*
CROPT  = 2     ;     DSL = 2     ; AIRDUC = 0
SPAN   = 22.   ;   TBASE = 0.    ;  RDMCR = 125.
   DLO = 1.0   ;     DLC = 0.
   EFF = 0.40  ;    CFET = 1.00  ;  DEPNR = 4.5

* initial dry weight
TDWI = 100.0  ! in kg/ha

* initial rooting depth
RDI = 10.0    !  cm

* optimum development rates
DVRC1 = 0.034
DVRC2 = 0.058

* optimum rooting depth increase per day
```

```
RRI = 1.2

* extinction coefficient
KDIF = 0.6

* conversion factors
CVL = 0.72 ;  CV0 = 0.73 ;  CVR = 0.72 ;  CVS = 0.69

* partitioning
* ============
FRTB =  0.00,   0.70,     !  fraction to roots
        0.33,   0.50,
        1.00,   0.00,
        2.00,   0.00

FLTB =  0.00,   1.00,     !  fraction to leaves
        0.33,   1.00,
        0.80,   0.30,
        1.00,   0.00,
        2.00,   0.00

FSTB =  0.00,   0.00,     !  fraction to stems
        0.33,   0.00,
        0.80,   0.70,
        1.00,   0.50,
        1.01,   0.00,
        2.00,   0.00

FOTB =  0.00,   0.00,     ! fraction to storage organs
        0.80,   0.00,
        1.00,   0.50,
        1.01,   1.00,
        2.00,   1.00

* specific leaf area table
SLATB =  0.00,  0.0025,
         2.00,  0.0025

* development rate ; temperature dependence
DVRETB =  0.00,   0.00,
         35.00,   1.00,
         45.00,   1.00
```

```
      PROGRAM  EXAMP5

*     This program compares the cumulated assimilation with two
*     different AMAX values over a growing season using a given
*     LAI as a forcing function. Radiation data are actual data.

      IMPLICIT REAL (A-Z)
      PARAMETER (IAR=20)
      REAL LAIT (IAR)
      INTEGER ILAIN, ISTAT, IDAY, IDAYS, IDAYE

      CALL RDINIT (30, 0, 'INPUT.DAT')
      CALL RDAREA ('LAIT', LAIT, IAR, ILAIN)
      CALL RDSREA ('AMAX1', AMAX1)
      CALL RDSREA ('AMAX2', AMAX2)
      CALL RDSREA ('EFF', EFF)
      CLOSE (30, STATUS='DELETE')

      CALL STINFO (11, 'HD40: WEER:DATA:', ' ', 'NL', 1, 1984,
     &             ISTAT, LONG, LAT, ALT, A, B)

      CALL OUTDAT (1, 40, 'DAY', 0.)

      IDAYS = NINT (LAIT(1))
      IDAYE = NINT (LAIT(ILAIN-1))

      DTGA1S = 0.
      DTGA2S = 0.

      DO 10 IDAY=IDAYS,IDAYE
         DAY = FLOAT (IDAY)
         CALL WEATHR (IDAY,ISTAT,
     &                RAD, TMIN, TMAX, VAPOUR, WIND, RAIN)
         RAD = RAD*1000.

         CALL ASTRO (DAY, LAT, DAYL, DAYLP, SINLD, COSLD)

         LAI = LINT (LAIT, ILAIN, DAY)

         CALL TOTASS (DAY, DAYL, AMAX1, EFF, LAI, RAD, SINLD, COSLD,
     &                DTGA)
         DTGA1S = DTGA1S+DTGA

         CALL TOTASS (DAY, DAYL, AMAX2, EFF, LAI, RAD, SINLD, COSLD,
     &                DTGA)
         DTGA2S = DTGA2S+DTGA

         D1D2 = DTGA1S/DTGA2S
```

```
            CALL OUTDAT (2, 0, 'DAY', DAY)
            CALL OUTDAT (2, 0, 'RAD', RAD)
            CALL OUTDAT (2, 0, 'LAI', LAI)
            CALL OUTDAT (2, 0, 'DTGA1S', DTGA1S)
            CALL OUTDAT (2, 0, 'DTGA2S', DTGA2S)
            CALL OUTDAT (2, 0, 'D1D2', D1D2)

10      CONTINUE

        CALL OUTDAT (4, 0, 'Output van voorbeeldprogramma', 0.)

        CALL OUTPLT (1, 'DTGA1S')
        CALL OUTPLT (1, 'DTGA2S')
        CALL OUTPLT (6, 'Output van voorbeeldprogramma')

        CALL OUTPLT (1, 'D1D2')
        CALL OUTPLT (6, 'Output van voorbeeldprogramma')
        CALL OUTDAT (5, 0, 'Output van voorbeeldprogramma', 0.)
        CALL OUTDAT (6, 0, 'Output van voorbeeldprogramma', 0.)
        CALL OUTDAT (99, 0, ' ', 0.)

        STOP
        END

        INCLUDE HD40: FORTRAN:WEER.FOR
        INCLUDE HD40: FORTRAN:TTLIB:ASTRO.FOR
        INCLUDE HD40: FORTRAN:TTLIB:TOTASS.FOR
        INCLUDE HD40: FORTRAN:TTLIB:RADIAT.FOR
        INCLUDE HD40: FORTRAN:TTLIB:ASSIM.FOR
```

Appendix

A remark on AFGEN tables

Some of the programs in TTUTIL are meant to handle so called AFGEN tables. An AFGEN table consists of a number of (X,Y) pairs. The values are stored in an array. The X values in the array should form an increasing series. AFGEN tables are intended to be used by an interpolation routine. For linear interpolation, the function LINT is available in TTUTIL.

There are two ways of declaring an AFGEN table. The traditional method is to declare a one-dimensional array and to use the odd elements as X–values and the even elements as Y-values:

```
      INTEGER ILMAX
      PARAMETER (ILMAX=200)
      REAL TABLE (ILMAX)
```

The array TABLE has room for 100 data pairs. A DO-loop over the first 50 points looks like:

```
      DO 10 I=1,50
         X = TABLE (2*I-1)
         Y = TABLE (2*I)
         .....
10    CONTINUE
```

In some of the TTUTIL routines another method is used to declare an AFGEN table:

```
      INTEGER ILMAX
      PARAMETER (ILMAX=200)
      REAL TABLE (2,ILMAX/2)
```

The array is two-dimensional. The element TABLE(1,I) is the I-th X value, the element TABLE(2,I) is the I-th Y value. The above loop over the first 50 points now becomes:

```
      DO 10 I=1,50
         X = TABLE (1,I)
         Y = TABLE (2,I)
         .....
10    CONTINUE
```

In some applications the one dimensional arrays lead to awkward calculations with array indices. In the second method the X and Y values are directly found from the point number.

The two methods lead to the same values in the same places in computer memory. Therefore, both declaration methods can be used in combination with LINT and the other AFGEN table handling routines. Note that the array length should always be twice the number of (X,Y) pairs and the declaration as a two dimensional array should always use the "2" as the first index and the number of points (the array length divided by 2) as the second.

The routines dealing with AFGEN tables all require the actual array length as an argument. So it is good practice to always make a distinction between the declared length and the actual length, for instance by using ILFMAX and ILF as declared and actual lengths of the table TABLE. Reading datafiles one should check the value of ILF against ILFMAX for instance by means of the statement:

```
IF (ILF.GT.ILFMAX) CALL ERROR ('TEST','ILF > ILFMAX')
```

Such error checks require a single statement only and are frequently used in all routines of TTUTIL. Subroutine **ERROR** handles fatal errors.

Using routine **RDAREA** for reading AFGEN tables from file, the above check can be omitted since it is carried out by **RDAREA**. The interpolation at X=2.1 of an AFGEN table TABLE present on file INPUT.DAT then looks as follows:

```
INTEGER ILFMAX, ILF
PARAMETER (ILFMAX=200)
REAL Y,LINT, TABLE (200)

CALL RDINIT (40,50,'INPUT.DAT')
CALL RDAREA (' TABLE', TABLE,ILFMAX,ILF)
Y = LINT (TABLE,ILF,2.1)
```

See also Example program 4 and the routine headers of RDINIT, RDAREA and LINT for further documentation.

Hiërarchical order of TTUTIL routines

Some linkers scan object libraries only once. Then, object libraries need to have a hiërarchical structure, which implies that lower level routines follow higher level routines. The table below assigns a "level" to all routines from TTUTIL. Higher level routines call lower level ones.

```
AFINVS          2
BOXMUL          2
CLS         1
DECINT          2
DECREA          2
DECREC              3
ENTCHA      1
ENTDCH          2
```

ENTDIN		2			
ENTDRE		2			
ENTINT	1				
ENTREA	1				
ERROR	1				
EXTENS		2			
FOPEN		2			
GETCH		2			
GETREC	1				
IFINDC		2			
ILEN	1				
INSW	1				
ISTART	1				
LIMIT	1				
LINT		2			
MOFILP	1				
NUMBER	1				
OUTDAT			3		
OUTPLT			3		
PLTFUN		2			
PLTHIS		2			
POS		2			
RDAREA					5
RDDATA				4	
RDINDX			3		
RDINIT					5
RDSINT					5
RDSREA					5
REMOVE	1				
STRIP		2			
UNIFL	1				
UPPERC	1				
WORDS	1				

REFERENCES

Anonymous, 1987. IMSL User's manual.

Press, W.H., B.P.Flannery, S.A.Teukolsky and W.T.Vetterling. 1986. Numerical Recipes, the art of scientific computing. Cambridge University Press. 818 pp.