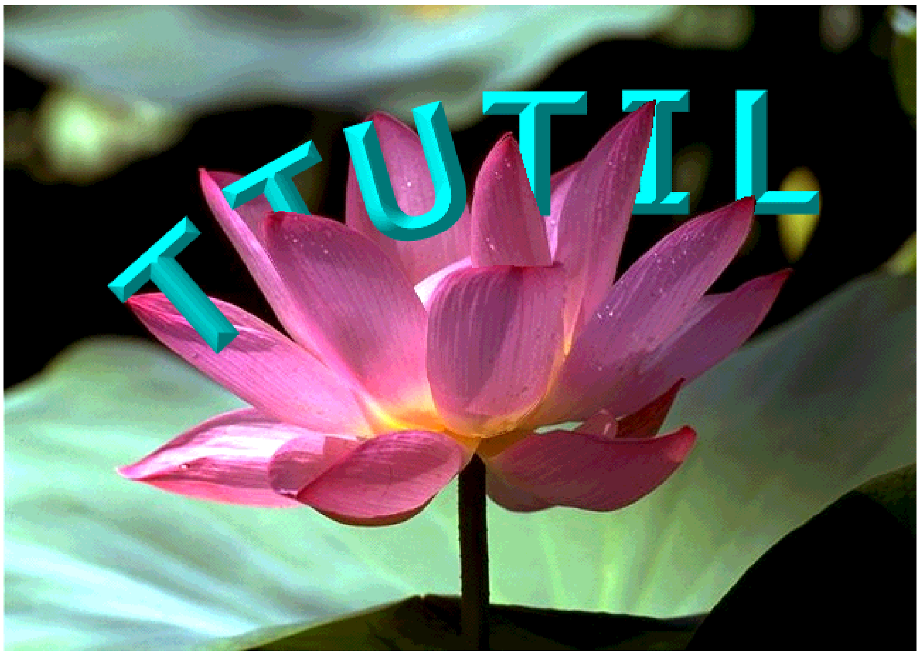
Reference manual of the FORTRAN utility library TTUTIL v. 4

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



Plant Research International, Wageningen.

March 2000. Report 5. Updated for TTUTIL 4.13 (as PDF version only, April 2002)

Table of Contents

1. Introduction

Over the past years we developed solutions for ever recurring problems with respect to file input, file output, character strings and file handling in Fortran simulation models. Each time we paid a little more attention than strictly necessary for the problem at hand. This has resulted in a slowly growing set of subroutines and functions that proved to be useful in almost any Fortran program. Most of these routines have gone through a number of revisions and a lot of new routines have been added. The set of routines became the utility library TTUTIL. The solutions that are offered by TTUTIL have saved many researchers enormous amounts of time and the library is now used in more than a few hundred places all around the world.

Through our experience on different hardware and software platforms, we are able to garantee a very high degree of portability to other machines. TTUTIL has been used sucessfully on VAX/VMS, OpenVMS for AXP, Prospero Fortran 77 for Atari 1024 ST, Microsoft Fortran 5.1 (MS-DOS), Microsoft Professional Powerstation (MS-DOS, Windows), GNU Fortran for Unix, Sun Fortran, LS Fortran for Macintosh v.2, Digital Visual Fortran v.5 for Windows 95 and Windows NT, Compaq Visual Fortran v.6.1 for Windows 95 and Windows NT and Absoft Fortran-90 for the Macintosh.

This is the last version of TTUTIL which fully complies with the FORTRAN-77 standard. Future versions may contain elements of the Fortran 90 language. We do not foresee a complete redesign of TTUTIL in Fortran 90 or 95 style. We will maintain, however, the functionality described in this report with current and new Fortran compilers.

This report serves as a reference manual in using the library. We hope that others will find the library a useful tool in improving their programs.

This report replaces CABO/TT report no. 20 (Rappoldt & van Kraalingen, 1990).

Authors of this report:

D.W.G. van Kraalingen
Alterra
P.O. Box 47
6700 AA Wageningen
The Netherlands
e-mail: d.w.g.vankraalingen@alterra.wag-ur.nl

C. Rappoldt
Alterra
P.O. Box 47
6700 AA Wageningen
The Netherlands
e-mail: c.rappoldt@alterra.wag-ur.nl

2. General description

2.1. Product perspective

The TTUTIL library is a collection of FORTRAN-77 utility routines for string handling, file i/o and screen i/o. Many routines are utilities in the sense that they do not make use of any mathematical or numerical method, do not contain measured data and do not depend on assumptions concerning some described system. Utilities simply perform their task with respect to input, output, string handling, file handling etc. They are tools for writing reliable and readable FORTRAN programs.

2.2. Product identification

Product:	TTUTIL (floppy disk + manual)
Version:	Version 4.13
Purpose:	FORTRAN-77 utility library
Author(s)	Daniël van Kraalingen, C. Rappoldt

2.3. Supported platforms

The TTUTIL library is available for the following platforms (note however, that the source code is capable of running on a wider range of platforms):

Windows NT/95/98 on Intel processors:
- Digital Visual Fortran v.5.0a-d
- Compaq Visual Fortran v.6.1, v.6.5

Macintosh:
Absoft Pro Fortran 6.0 (www.absoft.com) consisting of:
- Fortran 77 compiler for PowerMac version 4.5 (1998)
- Fortran 90 compiler for PowerMac version 2.0 (1998)

The linker can link Fortran 90 programs with Fortran 77 libraries, so there is no immediate need for using a Fortran 90 compiled TTUTIL library (see however the use of the function LINT in "knowns bugs").

Language Systems Fortran 77 consisting of:
- compiler for 68k Macs version 3.3 (1993)
- compiler for PowerMac version 1.2 (1996)

The products of Language systems have been taken over by Fortner Research (www.fortner.com). They have given up on this compiler, however, and Language Systems Fortran is no longer available.

Both the Language Systems and Absoft compilers work from Apple’s MPW shell. They allow the Fortran programs to be linked as either MPW tools or as standalone applications. Absoft claims its compilers are link compatible with MetroWorks CodeWarrior in PPC mode.

2.4. Availability

The TTUTIL library is available by submitting an e-mail request to one of the authors. Usually a nominal fee may be charged for delivery.

2.5. Hard and software limitations

The TTUTIL system is written in FORTRAN-77 with a few commonly accepted extensions to the standard. It is therefore easiest to use the TTUTIL library from a Fortran program, however, experienced programmers should have no difficulty calling TTUTIL routines from other programming languages.

3. The structure of the TTUTIL library

Table 3.1 gives a classification of the TTUTIL routines. Closely related routines have names often beginning with the same acronym. For instance, the "DEC" routines **DECCHK**, **DECINT**, **DECREA** and **DECREC** are used for decoding character strings into real or integer values.

Table 3.1 can be used for an efficient search through the library if you are looking for a routine that solves a specific programming problem. The routine descriptions, given in Section 10, provide further information on the individual subroutines and functions. In general, there will be no need for any further documentation. A few groups, however, require a more detailed introduction. These are the RD routines for reading data files with a convenient format (Section 4 and 5), the ENT routines for interactive variable entry (Section 6), the OUT routines for easy output programming (Section 7) and the routines for message and error handling (Section 8).

Table 3.1 Available TTUTIL routines

Category	Available routines
Reading of TTUTIL format datafiles:	rdinit, rdpars, rddtmp
	rdsets, rdfrom
	rdinqr, rdinqr2, rdinlv, rdinne, rdindt, rdinar
	rdscha, rdsdou, rdsint, rdslog, rdsrea, rdstim
	rdacha, rdadou, rdaint, rdalog, rdarea, rdatim
	rdfcha, rdfdou, rdfint, rdflog, rdfrea, rdftim
	rdador, rdainr, rdarer
	rdfdor, rdfinr, rdfrer
	rdsdor, rdsinr, rdsrer
	rdmdef, rdmcha, rdmdou, rdmint, rdmlog, rdmrea, rdmtim
Writing of TTUTIL format datafiles:	wrinit
	wracha, wraint, wralog, wrarea, wratim, wradou
	wrscha, wrsint, wrslog, wrsrea, wrstim, wrsdou
Interactive input:	entcha, entdch, entint, entdin, entrea, entdre, entdou,
	entddo, entyno, entdyn, enttim, entdti
Output to file:	copfl2
	outar2
	outcom
	outdat, outplt
	outsel
File and unit handling:	delfil, extens, flexist, flname
	fopeng, fopens
	getun, getun2, usedun
Character string handling:	addinf, addint, addrea, addref, addstf, addstr
	ilen, istart
	lowerc, upperc
	remove
	str_copy
	words

	rchrsrc
Decoding of character strings to values:	decchk, decdou, decint, decrea, decrec
Messages and Errors	fatalerr, warning, messini, messinq, messwrt, openlogf
Version routines	ttuver, ver4.13
Numeric functions:	fcnsw, insw, intgrl, limit, lint, lint2, movavr, notnul, reaand, reanor
Date/time:	dtardp, dtdpar, dtdpst, dtfsecmp, dtfsedp, dtleap
‘Raw’ file I/O:	getrec, recread, recread_init, recread_term
List search and sorting:	ifindc, ifindi, sfindg sortch, sortin
Random number generation:	iunifl, unifl
Miscellaneous:	ambusy, chktsk, timer2
Internal routines:	dtsys, rddata, rddecd, rddeci, rddecl, rddect, rderr, rdindx, rdlex, rdsctb, rdtmp1, rdtmp2, swpi4

See Section 13 for a description of the routines that have been removed in this version.

4. General concept of RD routines

4.1. A simple example

The ordinary method for reading data from a file consists of a number of READ statements, each reading data from a record of the file. That method clearly requires that the sequence of READ statements is consistent with the contents of the file. Moreover, array lengths have to be known in the program or have to be read as separate data items. The ordinary reading method, moreover, requires accurate positioning of the data items otherwise multiplication or division by 10 can sometimes occur. In general, much time is invested in debugging such "simple" input sections.

The solution suggested sometimes in textbooks on FORTRAN is to read data from file as character strings and to perform the decoding in the program. This method, however, requires a considerable programming effort and a need was felt for generally applicable input routines that allow a great deal of flexibility and provide robustness. The RD routines are designed to do just this. They enable the construction of clear, short and robust input sections consisting of CALL's only and the construction of robust, powerfull, self-explanatory datafiles.

The general idea is that the input file contains both the variable name *and* the associated value(s). 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 the user can request the value from the datafile by supplying the name of the requested variable (of course after having defined which data file to use). As an example the statement:

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

requests the subroutine `RDSREA` to extract the value of `WLVI` from the data file and assign it 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 actually read from a temporary file which is created after syntax check and analysis of the data file). An example datafile is given in Listing 4.1. Note that the comment lines are actually part of the data file

Listing 4.1 Example datafile demonstrating various syntax forms

```
* example data file
N   = 10                            ! single value
BB  = 0, 2, 4, 6                    ! array of four elements
CCC = 10., 20.,                     ! array continued on next line
      30., 40.
DD  = 100*10.                       ! array of 100 elements
EE  = 10.; FF = 20.; G = 30.        ! more than one variable on a single line
```

Listing 4.2 reads the values of `CCC`, `BB`, `EE`, `FF` and `N` respectively from the above listed data file `INPUT.DAT`. Also note that it is not necessary to read the values in the same sequence as they occur on the file, just as it is not necessary to read every variable in the file. In the declarations section the parameters ILBMAX and ILCMAX specify the declared lengths of the arrays BB and CC. A further explanation is given below the listing.

Listing 4.2 Example illustrating the use of some RD routines.

```
*     declarations
      INTEGER N,ILBMAX,ILB,ILCMAX,ILC
      PARAMETER (ILBMAX=100,ILCMAX=100)
      REAL EE,FF
      REAL BB(ILBMAX),CCC(ILCMAX)

*     example of input section
      CALL RDINIT (30,40,'INPUT.DAT')                   <-  Initialization of reading
      CALL RDSREA ('EE' ,EE)                            <-  read single value
      CALL RDSREA ('FF' ,FF)                            <-  read single value
      CALL RDAREA ('CCC',CCC,ILCMAX,ILC)                <-  read array
      CALL RDAREA ('BB' ,BB ,ILBMAX,ILB)                <-  read array
      CALL RDSINT ('N'  ,N )                            <-  read single value
      CLOSE (30)                                        <-  close reading
```

The statement:

```
CALL RDINIT (30, 40, 'INPUT.DAT')
```

calls the RDINIT routine that 1) opens the file INPUT.DAT using unit=31, 2) analyses the data file, 3) creates a temporary file from the data file using unit=30, 4) closes the data file (leaving 30 used for the temporary file !!), and 5) sends all error messages that have occurred to a log file (with unit=40). After this RDINIT call, the numerical values (including arrays) can be acquired through several RD routines from the library TTUTIL. Note that input sections starting with RDINIT cannot be nested (see Section 9.1).

The values of two real variables EE and FF are read by means of two calls to RDSREA. The first argument of this subroutine is the name of the variable, written as a character constant. Using that name, the routine RDSREA identifies the value to be assigned to the variable. So the character string in the CALL should correspond to the variable name in the data file.

Routine RDAREA reads arrays of real values. In the above example it is called two times for reading the arrays CCC and BB from file. Note that the declared length (=maximum length) is an input argument of RDAREA and the actual array length is an output argument. Clearly, the number of values in the file should not exceed the declared length. This is checked by the RD system and leads to a fatal error message. Finally, the value of the single integer variable N is read using RDSINT. The CLOSE statement disables reading from the datafile.

In the example of Listing 4.1 and Listing 4.2 only INTEGER and REAL datatypes are used. The TTUTIL datafile syntax, however, allows several more datatypes (nl. DOUBLE PRECISION, CHARACTER, LOGICAL, 'Date/time', and 'Missing' datatypes). In Listing 4.3 examples are given of these datatypes.

Listing 4.3 Examples of other supported datatypes

```
D7 = .35D+3                               ! DOUBLE PRECISION
L1 = .TRUE.                               ! LOGICAL
S3 = 'ABC'//'DEF'//'GHI'//'JKL'           ! CONCATENATED STRINGS
DT14 = 26-DEC-1905_12:34:23.070           ! Absolute Date/time
AI = 1234, -, 1444                        ! INTEGER array with one element missing
```

4.2. Reading tables and arrays with fixed lengths

Another very powerfull feature of TTUTIL is that data can be organised in tables, showing the conceptual relationship among several variables. In Listing 4.4 a 'normal' datafile is shown:

Listing 4.4 Standard way of defining array values

```
THICKNESS = 10. , 30. , 50. , 70.
FIELD_CAP = 0.31, 0.33, 0.34, 0.38
```

But it can more elegantly and more clearly, be written as in Listing 4.5:

Listing 4.5 Tabular way of defining array values

```
THICKNESS  FIELD_CAP
   10.       0.31
   30.       0.33
   50.       0.34
   70.       0.38
```

The RD calls to read this from first or the second datafile are identical as shown in Listing 4.6:

Listing 4.6 Program to read both formats of the datafile

```
      INTEGER MAX_NL, NL                                ! Maximum number of layers
      PARAMETER (MAX_NL=10)
      REAL THICKNESS(MAX_NL), FIELD_CAP(MAX_NL)
      ...
*     read arrays
      CALL RDAREA ('THICKNESS', THICKNESS, MAX_NL, NL) ! determine # of layers
      CALL RDAREA ('FIELD_CAP', FIELD_CAP, MAX_NL, NL) ! check # of layers
```

This program reads exactly the same from the first or the second datafile. As was discussed earlier, the RDAREA call reads a REAL array and returns the number of data found in the datafile. Often when working with compartments or layers, several parameters have to be specified for *each* compartment or layer. For instance in the example above, *each* soil layer needs to have a thickness and a water content at field capacity. The 'tabular' way of organising this does provide protection against incomplete specifications (e.g. six thicknesses but five water contents). There is, however, a set of special CALL's that, instead of returning the number of values, checks for a given number of values. This is the category of RDF routines.

The program in Listing 4.7 checks the number of elements of FIELD_CAP against the number found in THICKNESS. The number of data elements is returned in the first RDAREA call through NL, in the next call, to RDFREA, NL is the required number of data elements to be found.

Listing 4.7 Program that checks for same number of elements among soil arrays

```
      INTEGER MAX_NL, NL                                ! Maximum number of layers
      PARAMETER (MAX_NL=10)
      REAL THICKNESS(MAX_NL), FIELD_CAP(MAX_NL)
      ...
*     read arrays
      CALL RDAREA ('THICKNESS', THICKNESS, MAX_NL, NL)
      CALL RDFREA ('FIELD_CAP', FIELD_CAP, MAX_NL, NL)
```

4.3. Using missing values

As shown in Listing 4.3 missing values can be defined by a '-' in the datafile. Dependent on the datatype of the elements around the missing value (in case of an array), or the datatype of the call with which the missing value is read, a missing value code is returned. (See the reference section on the details of these values). The user, however, can override the default return value of missing elements by using a special call such as RDMREA (-300.). For each datatype, RDM routines are available to modify the missing value behaviour. The defaults for missing values are restored for all datatypes at once by a call to RDMDEF.

4.4. Getting information about a variable

In some cases you want information about a variable other than its value(s). For instance you might want to know whether it is present in the datafile, its number of values, its datatype, or whether it is an array or a scalar variable. Several routines are available to do just that, see the reference part of this manual.

Probably the most common kind of information that is necessary in a program is to know whether a variable exists at all in a datafile. Sometimes it is not absolutely required for a specific variable to be available in a datafile, especially in cases where a default behaviour is wanted. For instance if output options are defined in a datafile, we want the program to read them and behave accordingly. If output options are not defined, default output options can be choosen. A special RD call exists to find out if a variable exists in the datafile nl. RDINQR. This is demonstrated in Listing 4.8.

Listing 4.8 Program that finds out the presence of a variable and takes action accordingly

```
      LOGICAL RDINQR
      CHARACTER*80 WEATHER_DIRECTORY
      ...
      IF (RDINQR ('WEATHER_DIRECTORY')) THEN
*        weather directory defined in data file
         CALL RDSCHA ('WEATHER_DIRECTORY',WEATHER_DIRECTORY)
      ELSE
*        weather directory not defined in data file, use default path
         WEATHER_DIRECTORY = 'C:\SYS\WEATHER\'
      END IF
```

4.5. Range checks on input

Often the value of one or more elements in the datafile is restricted to a certain range (e.g. relative humidities between 0 and 100%). The RD routines can check for this with several datatypes nl. `REAL`, `DOUBLE PRECISION` and `INTEGER` (Date/Time not yet implemented). The `RDSRER` routine for instance reads a single REAL and checks the value on the datafile against a lower and upper limit. If the value is not within this range, an error will occur. In Listing 4.9 an example program is shown.

Listing 4.9 Program that checks the input of a REAL variable.

```
      REAL X
      ...
      CALL RDSRER ('X', 0., 100., X)
```

4.6. Making reruns with the RD routines

The use of the RD routines for input has the additional advantage of a built in "rerun facility". Calculations often need to be carried out for different values of input variables. Suppose that something is calculated using the input variables `BB`, `EE` and `FF` from the program described in Listing 4.2 Suppose the calculations have to be repeated for different combinations of `BB` and `EE`. Then a so called *rerun file* can be created by the user, containing the desired combinations of `BB` and `EE`. For example:

<start of rerun file `RERUNS.DAT`>

```
* example rerun file
BB = 1, 3, 5, 7   ;  EE = 10.                     ! set 1
BB = 2, 4         ;  EE = 10.                     ! set 2 (with a short array)
BB = 0, 2, 4, 6   ;  EE = 30.                     ! set 3
BB = 1, 3, 5, 7,                                  ! set 4 (with a long array)
     8, 9, 8, 9   ;  EE = 30.
BB = 2, 4, 5, 7   ;  EE = 30.                     ! set 5
```

<end of file>

A rerun file thus consists of sets of variables. The *order of the variables should be identical* in all sets. The other syntax rules are identical to those of an ordinary data file. Actual array lengths may differ between sets (see the example above). A great advantage of the rerun facility is that most of the program code remains unchanged when a model is modified to be able to do reruns ! Listing 4.10 shows how this works.

Listing 4.10 Program that demonstrates how to program a rerun loop

```
*     open logfile and read rerun file
      CALL FOPENS (40,'RERUNS.LOG','NEW','DEL')
      CALL RDSETS (20,40,'RERUNS.DAT',INSETS)
*     runs
      DO ISET = 0,INSETS
*        select rerun set
         CALL RDFROM (ISET,.TRUE.)

*        an ordinary input section:
         CALL RDINIT (30,40,'INPUT.DAT')
         CALL RDAREA ('BB' ,BB ,ILBMAX,ILB)
         CALL RDSREA ('EE' ,EE)
         CALL RDSREA ('FF' ,FF)
         CLOSE (30)

*        calculations
         ........
      END DO
      CLOSE (20)
```

With a call to **FOPENS** a log file is opened, which is used for writing a report on rerun file usage. In the call to **RDSETS** the rerun file is analysed and a short report is written to the log file (unit 40). When the rerun file is not present or empty the output variable INSETS is set to zero. Otherwise the number of rerun sets is returned in INSETS (in the above example there are 5 rerun sets). By means of the call to **RDFROM** in the DO–loop, a certain set is selected. Selection of set zero means that the contents of the original data file will be used. The input section for reading the values of BB, EE and FF *is just a usual input section* for reading variables from a data file. The RD routines, however, internally check whether reruns are being made and whether a non–zero set was selected. In that case, for variables occurring in the rerun file, the data file contents are *replaced* by the contents of the rerun file. Since this is a rather hidden activity, *each replacement is reported to the log file*.

The file "RERUNS.DAT" may be absent or present. If the file is absent, the above program section will carry out only one run using the contents of the data file.

The rerun facility has a global character, i.e. variables stored in different data files may occur in a single rerun file. Within the above DO–loop, for instance, the variables BB and EE could be read from different input files by writing two separate input sections (each containing a call to **RDINIT**). As a consequence, the use of identical variable names in different input files leads to problems when reruns are made for that variable. Then the value of both variables will be replaced by the contents of the rerun file. Both replacements will be reported to the produced log file. Before a rerun is started, a check is done to see if all the variables of the preceding set were used. If this is not the case, it is assumed that there is a typing error in the data files and the simulation is halted.

4.7. Note when using reruns and the RDINIT routine

In Listing 10 a rerun loop is made around some straightforward calculations. With each new rerun loop, however, a call to `RDINIT` is made which would imply parsing and checking a datafile that is, under most circumstances already parsed and checked. To avoid this inefficient behaviour, `RDINIT` will recover the processed contents of a previous datafile from a .TMP file. The second `RDINIT` call with the same datafile is therefore considerably faster than the first call.

In some cases, however, it is mandatory that the datafile is parsed and checked again, e.g. in cases where a program generates datafiles for input to itself. In that case, instead of RDINIT, RDPARS calls should be used that do not recover any previous information from .TMP files.

4.8. Summary of available interface calls

4.8.1. Routines for opening and closing files

RDINIT	prepares new data file for reading, tries to recover previous ones
RDPARS	prepares new data file for reading, never recovers previous ones
RDSETS	prepares new rerun data file for reading, never recovers previous ones
RDFROM	selects a specific rerun set from a rerun file
RDDTMP	deletes .TMP files known to the RD system

4.8.2. Basic RD routines for reading data

Data type	Single	Unknown length	Prescribed length	Set value for missing data
`REAL`	`RDSREA`	`RDAREA`	`RDFREA`	`RDMREA`
`DOUBLE PRECISION`	`RDSDOU`	`RDADOU`	`RDFDOU`	`RDMDOU`
`INTEGER`	`RDSINT`	`RDAINT`	`RDFINT`	`RDMINT`
`CHARACTER`	`RDSCHA`	`RDACHA`	`RDFCHA`	`RDMCHA`
`LOGICAL`	`RDSLOG`	`RDALOG`	`RDFLOG`	`RDMLOG`
`DATE/TIME`	`RDSTIM`	`RDATIM`	`RDFTIM`	`RDMTIM`

4.8.3. Routines that perform range checks

Data type	Single	Unknown length	Prescribed length
`REAL`	`RDSRER`	`RDARER`	`RDFRER`
`DOUBLE PRECISION`	`RDSDOR`	`RDADOR`	`RDFDOR`
`INTEGER`	`RDSINR`	`RDAINR`	`RDFINR`
`CHARACTER`	not useful	not useful	not useful
`LOGICAL`	not useful	not useful	not useful
`DATE/TIME`	to be	to be	to be

5. Reference manual of data file syntax

5.1. Variable name syntax

Variable names in TTUTIL format data files can be up to 31 characters long. They are case-insensitive and should begin with a letter (a-z). Letters (a-z), digits (0-9) and the underscore (_) character can be used after the first letter. Variable names can occur more than once in rerun files, in standard datafiles, however, they can occur only once.

Examples:

```
X   = 5.
X_1 = 5.
X234567890123456789012345678901 = 5.
X23456789012345678901234567890_ = 5.
X2345678901234567_901234567890_ = 5.
abcdefg = 5.
```

Invalid examples:

```
3X = 5.
X&X@X = 5.
```

5.2. Definitions of data types

5.2.1. REAL and DOUBLE PRECISION data type

The specification of a `REAL` or `DOUBLE PRECISION` data type is very similar to how it would be done in a Fortran program. Both contain one decimal point (`.`) and may have a positive or negative exponent (denoted with an `E` or `e` for `REAL`'s or `D` or `d` for `DOUBLE PRECISION` datatype). The plus sign (+) is optional and indicates a positive number. The minus sign (`-`) is required to indicate a negative number.

Although both floating point types have a different range and accuracy in Fortran programs we handle them identically 'inside' the read routines of the TTUTIL library as a `DOUBLE PRECISION` data type. In other words, a floating point number without an exponent or with an E exponent is stored internally as a `DOUBLE PRECISION` number. When the user request for the value through a REAL RD call, a conversion to `REAL` is done. They are stored in 8 bytes on the .TMP file.

So,

```
X1 = 5.
X2 = 5.E3
X3 = 5.D3
```

are all stored in the same way in the .TMP file. They can be requested through `RDSREA` and `RDSDOU` calls.

The valid range for the `REAL` and `DOUBLE PRECISION` data type is:

-1.E+38 to -1.E-38, 0, 1.E-38 to 1.E+38

Up to 15 digits are decoded, more digits gives a 'loss of accuracy' error.

Examples

```
R1 = 3.
R2 = 3.5
R3 = 3.5E3
R4 = 0.35
R5 = -.35
R6 = .35E3
R7 = .35E+3
R8 = .35E+21
R9 = 0.123456789012345E-20
```

Invalid examples:

```
R1 = 3..
R2 = 3E3
```

5.2.2. INTEGER data type

Integers are stored in 4 bytes in the .TMP file. An integer is a whole number with no decimal point. Integers may have positive and negative values, negative integers must begin with a minus sign (-). The plus sign (+) is optional for positive integers.

The valid range for integers is between -2147483647 and +2147483647.

Examples:

```
I1 = 1
I2 = 1111111111
I3 = -4444
```

Invalid examples:

```
I1 = 0.
I2 = 92147483647
I3 = -2E0
```

5.2.3. LOGICAL data type

Logicals are stored in 4 bytes on the .TMP file. Their value should be either .TRUE. or .FALSE. (=case insensitive).

Examples:

```
L1 = .TRUE.
L2 = .FALSE.
L3 = .TrUe.
L4 = .fAlSe.
```

Invalid examples:

```
L1 = TRUE
L2 = false
L3 = 'TrUe'
```

5.2.4. CHARACTER data type

The `CHARACTER` data type is defined in the data file in a manner similar to Fortran. They should be between quotes and should have ASCII value between decimal 32 and 127. Separate character strings can be concatenated to form one string with the `//` operator.

Examples:

```
S1 = 'A'
S2 = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
S3 = 'ABC'//'DEF'//'GHI'//'JKL'
```

Invalid examples:

```
S1 = A
S2 = 'Aàâäã'
S3 = 'ABC'+'DEF'+'GHI'+'JKL'
```

5.2.5. Date/time data type

Date/time data are stored in 8 bytes on the .TMP file. They are returned to the user as a DOUBLE PRECISION data type. The integer part of the value is defined as the number of days since 1 January 1900, the fractional part corresponds to the fractional day (e.g. noon is 0.5).

The date/time data type consists of a date part and/or a time part. If both are defined they should be separated with an underscore character (_).

Possible formats for the data part are:

```
jjjj/mm/dd      e.g. 1994/08/31
jjjj-mm-dd      e.g. 1994-08-31
dd/mmm/jjjj     e.g. 3/sep/1994, also 3/september/1994
dd-mmm-jjjj     e.g. 3-sep-1994, also 3-september-1994
```

Examples:

```
DT1 = 1905/12/26
DT2 = 1905-12-26
```

```
DT3 = 1905/0012/0026
DT4 = 26/DEC/1905
DT5 = 26-DEC-1905
DT6 = 26-decEMBER-1905
```

In the time part, at least hours and minutes should be available. The number of hours may exceed 24.

Possible formats are:

```
hh:mm                e.g.  11:20
hh:mm:ss             e.g.  11:20:45
hh:mm:ss.xxxxxxxxx   e.g.  11:20:45.453
```

Examples:

```
DT7  = 12:34
DT8  = 12:34:23
DT9  = 12:34:23.070
DT10 = 25:34
DT11 = 25:34:23.070
```

Both date part and time can be combined into one value by the underscore character (_).

Examples:

```
DT12 = 26-DEC-1905_12:34
DT13 = 26-DEC-1905_12:34:23
DT14 = 26-DEC-1905_12:34:23.070
```

5.2.6. 'Missing' data type

A value can be made missing by a dash character (-).

Examples:

```
M = -
```

See the RDMDEF routine for the default values.

5.3. Defining arrays

Arrays can be specified by repeating values of the same data type, separated by a comma, a space or an end_of_line. Also arrays, or part of arrays, can be specified by a multiplication factor followed by the asterisk '*' character. The data types of the elements of an array must be identical, except that the Missing data type can occur in an array.

As has been discussed in Chapter 1, arrays can also be written as a table.

Examples:

```
A1 = 1234, -1, 1444, 10
A2 = 1234  -1  1444  10
A3 = 1234, -1,
     1444, 10
A4 = 100*1234, 10*-1, 1444, 10
A5 = 100*1234, -, 1444, 10

MTR       MTD       MTI     MTL     MTS          MTDT
3.        3.D0      123    .TRUE.  'AAA'     26-DEC-1905_12:34:23.070
3.        3.D0      123    .TRUE.  'AAA'     26-DEC-1905_12:34:23.070
3.        3.D0      123    .TRUE.  'AAA'     26-DEC-1905_12:34:23.070
```

An array with just a single element is distinguished from a scalar value in one of the following ways:

- Use of a multiplier asterisk, as in:
  ```
  A1 = 1 * 7.3
  ```
- When given as part as a table with at least two columns, as in:
  ```
  A1     A2
  5      7
  ```

A degenerate array (consisting of a single element) cannot be read by RDS* routines. Similarly, a scalar value cannot be read by RDA* routines.

5.4. Comment lines

Comment lines start with '`*`' in the first column, or '`!`' in any column. The remainder of that line is then ignored. They can occur anywhere in the program even in tables.

Examples:

```
* example
AI = 1234, -1, 1444, 10             ! first specification
AI = 1234  -1  1444  10
AI = 1234, -1,                      ! first specification<EOL>
     1444, 10
AI = 100*1234, 10*-1, 1444, 10

MTR   MTD       MTI     MTL    MTS      MTDT
3.   3.D0       123   .TRUE.  'AAA'     26-DEC-1905_12:34:23.070  ! first line
3.   3.D0       123   .TRUE.    -       26-DEC-1905_12:34:23.070
* last line
3.   3.D0         -   .TRUE.  'AAA'     26-DEC-1905_12:34:23.070
```

5.5. Separation of specifications

Different specifications can be separated by the semicolon character '`;`'. Also an end_of_line when followed by a variable name is a valid separator.

Examples:

```
EE1  = 10.; FF1 = 20.; G1 = 30. <EOL>
EE1  = 10. <EOL>
FF1 = 20.  <EOL>
G1 = 30.   <EOL>
```

6. The ENT routines

The usual way to obtain interactive input from the user is to write a question to the screen and to read the answer from the screen. Exactly that is the function of the simple routines ENTCHA, ENTINT and ENTREA. They can be used to ask for a character string, an integer value and a real value, respectively. For instance, the statement

```
      CALL ENTREA ('Size of square',SIZE)
```

writes the question "Size of square" to the screen and the number returned is assigned to the real variable SIZE. Several such calls together form a relatively short program section for interactive input. Successive questions are written neatly below each other and the cursor is always in column 53 of the screen, independent of question length.

Somewhat less trivial are the subroutines ENTDCH, ENTDIN and ENTDRE. Again, the three routines are meant for entering a character string, an integer value or a real value, respectively. As an additional input argument, however, they accept a default value. The default value is returned to the calling program when the user does not type in a new value and presses the <Enter> key only. The three ENTD routines write the default value between square brackets following the question. For instance, the statement

```
      CALL ENTDRE ('Size of square',2.300,SIZE)
```

causes the following line being written to the screen:

```
                                    Size of square [2.3]:
```

The user either supplies a new value or just presses <Return> to accept the default. Note that the second argument (the default value) may also be a variable. The variable SIZE could be used, for instance, as the second and third argument of ENTDRE. Than the (current) value of SIZE is used as the default answer. In section 5.1 the use of that trick to simplify testing of newly written subroutines is illustrated.

7. The OUT routines

The OUT routines can be used to generate neat output tables with a minimum of programming effort. During calculations the name and value of a variable can be sent to routine OUTDAT which behaves as a temporary output storage. It stores the received output in a temporary file. After completion of the calculations, a table can be constructed from the gathered data by means of a special call to OUTDAT. The table can be used as the final result or can be imported into a spreadsheet or a statistical program.

The use of OUTDAT is illustrated in the example program TEST below. A table and a printplot are created of the sine and cosine of x between 0 and π.

```
01         PROGRAM TEST
02         IMPLICIT REAL (A-Z)
03         INTEGER IX
04         PARAMETER (PI=3.141597)
05
06   *     initialize output ; X is independent
07         CALL OUTDAT (1, 20, 'X', 0.0)
08
09         DO 10 IX = 0,20
10            X = FLOAT(IX) * PI/20.0
11            SINX = SIN (X)
12            COSX = COS (X)
13   *        repeated output calls
14            CALL OUTDAT (2, 0, 'X'   , X   )
15            CALL OUTDAT (2, 0, 'SINX', SINX)
16            CALL OUTDAT (2, 0, 'COSX', COSX)
17   10    CONTINUE
18
19   *     table construction
20         CALL OUTDAT ( 4, 0, 'sine + cosine', 0.0)
21
22   *     printplot contruction
23         CALL OUTPLT ( 1, 'SINX')
24         CALL OUTPLT ( 1, 'COSX')
25         CALL OUTPLT ( 7, 'sine + cosine')
26
27   *     delete temporary
28         CALL OUTDAT (99, 0, ' ', 0.0)
29         STOP
30         END
```

Listing 7.1 shows the output produced by this example program. The first parameter of the routines OUTDAT and OUTPLT is a task parameter. The first CALL to OUTDAT in line 7 of the above program (with ITASK=1) specifies that X will be the independent variable and that unit=20 can be used for the output file. Subsequent calls in lines 14,15 and 16 with ITASK=2 instruct OUTDAT to store the incoming names and 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 terminal call to OUTDAT in line 20 (with ITASK=4) instructs the routine to

Listing 7.1 Output produced by example program TEST in the text.

```
*-----------------------------------------------------------------------------
* Run no.:  1, (Table output)
* sine + cosine

       X         SINX         COSX

  .00000       .00000       1.0000
  .15708       .15643       .98769
  .31416       .30902       .95106
  .47124       .45399       .89101
  .62832       .58779       .80902
  .78540       .70711       .70711
  .94248       .80902       .58778
  1.0996       .89101       .45399
  1.2566       .95106       .30902
  1.4137       .98769       .15643
  1.5708       1.0000     -0.19809E-05
  1.7279       .98769      -.15644
  1.8850       .95106      -.30902
  2.0420       .89101      -.45399
  2.1991       .80902      -.58779
  2.3562       .70710      -.70711
  2.5133       .58778      -.80902
  2.6704       .45399      -.89101
  2.8274       .30901      -.95106
  2.9845       .15643      -.98769
  3.1416     -0.45280E-05  -1.0000


             sine + cosine

             Variable  Marker  Minimum value  Maximum value
             --------  ------  -------------  -------------
             SINX         1      -0.4528E-05      1.000
             COSX         2       -1.000          1.000

             Scaling: Common      -1.000          1.000

 X

  .00000     I-------------------------------1-------------------------------2
  .15708     I               I               I    1          I               2
  .31416     I               I               I         1     I             2 I
  .47124     I               I               I              1I            2  I
```

```
.62832     I               I               I               I  1      2     I
.78540     I               I               I               I      *        I
.94248     I               I               I               I  2      1     I
1.0996     I               I               I              2I            1  I
1.2566     I               I               I         2     I             1 I
1.4137     I               I               I    2          I               1
1.5708     I-------------------------------2-------------------------------1
1.7279     I               I          2    I               I               1
1.8850     I               I     2         I               I             1 I
2.0420     I               I2              I               I            1  I
2.1991     I            2  I               I               I         1     I
2.3562     I        2      I               I               I      1        I
2.5133     I     2         I               I               I  1            I
2.6704     I  2            I               I              1I               I
2.8274     I 2             I               I         1     I               I
2.9845     2               I               I    1          I               I
3.1416     2-------------------------------1-------------------------------I
```

create an output table using the information stored in the temporary file. Dependent on the value of the task variable, different output formats are chosen. Tab-delimited format (for a spreadsheet) can be generated with ITASK=5, two column format with ITASK=6. The string between quotes is written above the output table.

The calls to OUTPLT in line 23 and 24 (with ITASK=1) instruct the routine to put "SINX" and "COSX" in a graph (up to 25 variables can be printed per plot). The subsequent call with ITASK=7 causes OUTPLT to create the plot. Two different widths of the printplot are possible, 80 and 132 columns, and two different types of scaling, a common scale and individual scales (see Table 7.1). The process can be repeated to create several print plots based on the same output data. The final call to OUTDAT (in line 28 with ITASK=99) deletes the temporary file.

Table 7.1 The task variable that should be supplied to OUTPLT to generate the different print plot types

	Width	
Scaling:	132	80
Individual	4	6
Common	5	7

8. Messages and Error handling

A Fatal error condition in any of the TTUTIL routines results in a call like:

```
      CALL FATALERR ('MODULE', 'message on what went wrong')
```

which displays

```
ERROR in MODULE: message on what went wrong
```

After displaying the error a STOP statement is executed. This is the only STOP statement in the library. If necessary the STOP can be easily replaced by some other (machine dependent) procedure halting program execution. Uniformity in error handling can be achieved by calling FatalERR from user routines as well.

Warnings and general messages can be displayed using calls in the same style to WARNING and MESSWRT. These routines return to the calling program.

The default output for FATALERR, WARNING, MESSWRT and all other TTUTUL routines is the screen. Only RDINIT and RDSETS accept a logfile unit number for displaying messages. This can be changed by calling MESSINI at some point in the user program, but usually at the very begin of it. For instance,

```
      CALL MESSINI (.FALSE., . TRUE., IUNIT)
```

disables screen output and enables logfile output to unit IUNIT, which must be the unit associated with an open, sequential and formatted file (e.g. use FOPENS). The use of MESSINI affects all TTUTIL routines that generate messages, including FATALERR, WARNING and MESSWRT. Only the ENT* routines are not affected, since they were designed for screen and keyboard i/o. The screen and logfile settings are available to any (user) routine by

```
      CALL MESSINQ (TOSCR, TOLOG, IUNIT)
```

If the two logicals and the unit IUNIT are subsequenty used to control local output, a uniform program behaviour is realized in a simple way.

8.1. A logfile with version and author in a single call

A logfile usually contains the name of the program generating it, a version number and the name of the author(s). It is also convenient to have the time of the program run stored in the header of the logfile. All this is realized by means of a single call to OPENLOGF at the beginning of the program run. With

```
      call OPENLOGF (TOSCR, ‘Test’, ProgNam, ‘1.34_Beta’, ‘Daan and Kees’, .true.)
```

a logfile TEST.LOG is created and a logfile header is composed containing the date and time of file creation, program name, Version_String and Author_Name_String. Note that version number is passed as a string which means that any sort of version identification will do. Optionally, with the last argument of the OPENLOGF call, the header includes the version of the TTUTIL library used. OPENLOGF generates a logfile unit number in the range [41,99] and calls MESSINI to initialize message output as described above. Hence, if OPENLOGF is used, subroutine MESSINI need not to be called anymore. The logfile unit number is available via routine MESSINQ.

8.2. RD* routines and logfile use

After a MESSINI or OPENLOGF call, the routines RDINIT and RDSETS neglect their second argument, which is a logfile unit number. Instead, all RD* routines follow the instructions given in the MESSINI or OPENLOGF call. The only way to revert to default behaviour is to reset MESSINI by

```
      CALL MESSINI (logical, .TRUE., 0)
```

The combination of a .TRUE. for logfile use and a zero unit number resets MESSINI (first argument arbitrary). All TTUTIL routines revert to their default behaviour which is output to screen in combination with logfile units passed to RDINIT and RDSETS as arguments in the call.

9. Known problems

9.1. Illegal nesting of input sections

There are basically two types of input sections in the TTUTIL library. There are sections starting with RDINIT or RDPARS and ending with CLOSE(unit), and there are sections starting with RECREAD_INIT and ending with RECREAD_TERM. These input procedures cannot be nested within themselves. For instance the following RDINIT, CLOSE(unit) input section is *illegal* (irrespective of the values of the unit numbers):

```
CALL RDINIT (20,40,'INPUT1.DAT')
   <possible reading of values from file>
CALL RDINIT (30,40,'INPUT2.DAT')
   <possible reading of values from file>
CLOSE (30)
   <possible reading of values from file>
CLOSE (20)
```

The same applies to input sections using the RECREAD_INIT, RECREAD_TERM routines. However, it is possible to nest RECREAD_INIT, RECREAD_TERM calls within an RDINIT, CLOSE(unit) input section. The reverse is not possible.

All this applies equally to situations where input sections are within called subprograms. In general this means that calls to large subprograms from within input sections should be avoided.

9.2. Closing RD* input files

The following erroneous construction is not yet properly detected by the RD* routines:

```
CALL RDINIT (20,40,'INPUT1.DAT')
   <possible reading of values from file>
CLOSE (20)
OPEN(20,FILE=…)
<continue reading of values from file>
```

The illegal construction here is that unit 20 is closed and re-assigned to another file within the input section. In this situation the RD routines do not give a proper error message.

9.3. Compiler specific problems

9.3.1. Digital Visual Fortran and Compaq Visual Fortran

Use of the function ILEN leads to the following warning, which should be ignored by the user:

```
Warning: Arguments' data types are incompatible with intrinsic procedure, assume
EXTERNAL.   [ILEN]
```

9.3.2. All Macintosh MPW Fortran compilers

If a Fortran application is linked as an MPW tool, interactive screen input with the ENT* routines does not work. A fix requiring a small change in ttutil routine ENTCHA is available from the
authors. Most programs which read a few values or strings from the keyboard will be linked as applications, however. In that case there is no problem.

9.3.3. Macintosh Absoft Fortran 90 compiler

The use of the TTUTIL function LINT in Fortran 90 source code leads to problems with the Absoft Fortran 90 compiler. It classifies LINT as an intrinsic function and gives an error message. This compiler bug is fixed by the inlusion of the following explicit interface in the subprogram in which LINT is being used"

```
INTERFACE
   FUNCTION lint (Table, iltab, x)
   REAL :: lint
   INTEGER, INTENT(IN) :: lint
   REAL, DIMENSION(iltab), INTENT(IN) :: Table
   REAL, INTENT(IN) :: X
   END FUNCTION lint
END INTERFACE
```

The program may still be linked with a Fortran 77 compiled TTUTIL library. This problem may have been solved in the latest version, Absoft Fortran Pro 7.0. We do not know yet.

10. Reference manual of interface calls

10.1. Reading of TTUTIL format datafiles

Routine: RDINIT

Purpose:	Initializes a data file in TTUTIL data file format for subsequent reading with other RD routines. Initialization consist of checking the syntax of the datafile (parsing), and generation of a temporary file from which actual reading can take place with the other RD routines. If a datafile is initialized more than once with RDINIT, and the file for temporary storage is not deleted by the user, then, for speed reasons, re-parsing will not take place. If re-parsing must take place, then RDPARS should be used to initialize the data file, instead of RDINIT. Note that input sections starting with RDINIT or RDPARS cannot be nested (see Section 9.1).
Usage:	call RDINIT (IUNIT, IULOG, DATFIL)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDPARS, RDDTMP

Arguments:	Meaning	Data type	I and/or O
IUNIT:	Free unit number used to open random access file for I/O (used for temporary storage), IUNIT+1 is used to open the data file DATFIL (this unit is closed after reading)	I4	I
IULOG:	>0, Open unit number of logfile, used for data file syntax errors. =0, Nothing is done with a logfile	I4	I
DATFIL:	Name of data file to be read	C*(*)	I

Routine: RDPARS

Purpose:	Initializes a data file in TTUTIL data file format for subsequent reading with other RD routines. See RDINIT for a discussion on the difference between RDINIT and RDPARS. Note that input sections starting with RDINIT or RDPARS cannot be nested (see Section 9.1).
Usage:	call RDPARS (IUNIT, IULOG, DATFIL)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINIT, RDPARS

Arguments:	Meaning	Data type	I and/or O
IUNIT:	Free unit number used to open random access file for I/O (used for temporary storage), IUNIT+1 is used to open the data file DATFIL (this unit is closed after reading)	I4	I
IULOG:	>0, Open unit number of logfile, used for data file syntax errors. =0, Nothing is done with a logfile	I4	I
DATFIL:	Name of data file to be read	C*(*)	I

Routine: RDDTMP

Purpose:	Deletes the temporary files created by the RD routines. In most situations all temporaries can be deleted. The name of a possibly used rerun file is known locally in RDDATA and there is a list of data files available which have been opened with RDINIT. The TMP files belonging to this list are deleted as far as they are still there.
Usage:	call RDDTMP (IUNIT)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINIT, RDPARS

Arguments:	Meaning	Data type	I and/or O
IUNIT:	Unit number used to delete the temporary files (should not currently be assigned to any file)	I4	I

Routine: RDSETS

Purpose:	Initializes the RD system 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 analyzed with RDINIT or RDPARS and read with the RDS*, RDA*, RDM*, or RDF* routines.
Usage:	call RDSETS (IUNIT, IULOG, SETFIL, INS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINIT, RDPARS, RDFROM

Arguments:	Meaning	Data type	I and/or O
IUNIT	Free unit number used to open random access file for I/O (used for temporary storage), IUNIT+1 is used to open the data file DATFIL (this unit is closed after reading)	I4	I
IULOG:	>0, Open unit number of logfile, used for data file syntax errors. =0, Nothing is done with a logfile	I4	I
SETFIL	Name of rerun file containing sets. When an empty string is supplied, the rerun facility is disactivated.	C*(*)	I
INS	Number of sets on file (when exists minimum 1)	I4	O

Routine: RDFROM

Purpose:	Instructs the RD system (and the user interfaces RDS*, RDA*, RDM*, and RDF*) to use the IS-th set from the rerun file. Note that set 0 (zero) means that standard 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. See also RDSETS.
Usage:	call RDFROM (IS, FATAL)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSETS

Arguments:	Meaning	Data type	I and/or O
IS	=0, data file contents selected, disable replacement >0, set number selected	I4	I
FATAL	=.false., non-used variables gives text to logfile =.true., non-used variables gives fatal error	L4	I

Routine: RDINQR

Purpose:	Returns a flag for the presence of variable XNAME in the current data file. Presence on rerun file of the variable is not determined.
Usage:	<logical variable> = RDINQR (XNAME)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINQR2, RDINLV

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
RDINQR	= .true., Variable occurs in current data file =.false., Variable not present	L4	O

Routine: RDINQR2

Purpose:	Returns a flag for the presence of variable XNAME in the current data file, if no rerun set is selected, the variable is checked in the standard datafile, otherwise the variable is looked up in the selected rerun set.
Usage:	<logical variable> = RDINQR2 (XNAME)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINQR, RDINLV

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
RDINQR	=.true., Variable occurs in selected data =.false., Variable not present	L4	O

Routine: RDINLV

Purpose:	Returns a list of variables, if no rerun set is selected, the names of the standard datafile are returned, otherwise the names of the rerun file are returned.
Usage:	call RDINLV (SETFLAG, VARLIS, VARLIS_MN, VARLIS_AN)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINQR, RDINQR2

Arguments:	Meaning	Data type	I and/or O
SETFLAG	Flag through which is indicated whether information from a possibly active rerun set is wanted: =.true., information from active set is returned =.false., information from standard datafile is returned	L4	I
VARLIS	Character array of dimension VARLIS_MN in which list of variables is returned. Note that variable names in the data files can be up to 31 characters wide. Declare the length of VARLIS accordingly.	C*(*)	O
VARLIS_MN	Maximum number of names that can be returned	I4	I
VARLIS_AN	Actual number of variable names returned in list	I4	O

Routine: RDINNE

Purpose:	Returns number of elements of a variable, if no rerun set is selected, information from the standard datafile is returned, otherwise information of the variable in the particular rerun set is returned.
Usage:	call RDINNE (VAR_NAME, NO_EL)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINLV

Arguments:	Meaning	Data type	I and/or O
VAR_NAME	Name for which information is requested	C*(*)	I
NO_EL	Number of elements of VAR_NAME	I4	O

Routine: RDINDT

Purpose: Returns the data type of a variable, if no rerun set is selected, information from the standard datafile is returned, otherwise information of the variable in the particular rerun set is returned. Returned data types can be:

I	-	Integer
F	-	Floating point
L	-	Logical
C	-	Character string
T	-	Date/time
-	-	Missing data type

Usage:	call RDINDT (VAR_NAME, DATA_TYPE)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINLV

Arguments:	Meaning	Data type	I and/or O
VAR_NAME	Name for which information is requested	C*(*)	I
DATA_TYPE	Data type of VAR_NAME	C*(*)	O

Routine: RDINAR

Purpose:	Returns a flag whether a variable is an array in the datafile.
Usage:	<logical variable> = RDINAR (VAR_NAME)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDINLV

Arguments:	Meaning	Data type	I and/or O
VAR_NAME	Name for which information is requested	C*(*)	I

Routine: RDSCHA

Purpose:	Reads a single character string value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSCHA (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable to be read	C*(*)	I
X	Value of variable	C*(*)	O

Routine: RDSDOU

Purpose:	Reads a single DOUBLE PRECISION value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSDOU (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSDOR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R8	O

Routine: RDSINT

Purpose:	Reads a single INTEGER value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSINT (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSINR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	I4	O

Routine: RDSLOG

Purpose:	Reads a single LOGICAL value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSLOG (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
XNAME:	Name of variable	C*(*)	I

X:	Value of variable	L4	O

Routine: RDSREA

Purpose:	Reads a single REAL value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSREA (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSRER

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R4	O

Routine: RDSTIM

Purpose:	Reads a single TIME value from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS.
Usage:	call RDSTIM (XNAME, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R8	O

Routine: RDACHA

Purpose:	Reads an array of CHARACTER values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDACHA (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFCHA

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	C*(*)	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDADOU

Purpose:	Reads an array of DOUBLE PRECISION values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDADOU (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFDOU, RDADOR, RDFDOR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDAINT

Purpose:	Reads an array of INTEGER values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDAINT (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFINT, RDAINR, RDFINR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	I4	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDALOG

Purpose:	Reads an array of LOGICAL values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDALOG (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFLOG

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	L4	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDAREA

Purpose:	Reads an array of REAL values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDAREA (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFREA, RDARER, RDFRER

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R4	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDATIM

Purpose:	Reads an array of TIME values from a TTUTIL format data file. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND.
Usage:	call RDATIM (XNAME, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFTIM

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDFCHA

Purpose:	Reads a fixed number of elements into a CHARACTER array from a TTUTIL format data file. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFCHA (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDACHA

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	C*(*)	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFDOU

Purpose:	Reads a fixed number of elements into a DOUBLE PRECISION array from a TTUTIL format data file. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFDOU (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDADOU, RDADOR, RDFDOR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFINT

Purpose:	Reads a fixed number of elements into an INTEGER array from TTUTIL format data. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFINT (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAINT, RDAINR, RDFINR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	I4	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFLOG

Purpose:	Reads a fixed number of elements into a LOGICAL array from TTUTIL format data. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFLOG (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDALOG

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	L4	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFREA

Purpose:	Reads a fixed number of elements into a REAL array from TTUTIL format data. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFREA (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAREA, RDARER, RDFRER

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R4	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFTIM

Purpose:	Reads a fixed number of elements into a TIME array from TTUTIL format data. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message.
Usage:	call RDFTIM (XNAME, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDATIM

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDADOR

Purpose:	Reads an array of DOUBLE PRECISION values from a TTUTIL format data file and carries out a range check on the returned values. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDADOR (XNAME, XMIN, XMAX, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDADOU, RDFDOU, RDFDOR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	R8	I
XMAX	Maximum acceptable value	R8	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDAINR

Purpose:	Reads an array of INTEGER values from a TTUTIL format data file and carries out a range check on the returned values. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDAINR (XNAME, XMIN, XMAX, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAINT, RDFINT, RDFINR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	I4	I
XMAX	Maximum acceptable value	I4	I
X	Array itself of dimension ILDEC	I4	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDARER

Purpose:	Reads an array of REAL values from a TTUTIL format data file and carries out a range check on the returned values. The reading should be initialized with RDINIT or RDPARS. The number of values on file is returned as IFND. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDARER (XNAME, XMIN, XMAX, X, ILDEC, IFND)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAREA, RDFREA, RDFRER

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	R4	I
XMAX	Maximum acceptable value	R4	I
X	Array itself of dimension ILDEC	R4	O
ILDEC	Declared length of X	I4	I
IFND	Number of values found on file	I4	O

Routine: RDFDOR

Purpose:	Reads a fixed number of elements into a DOUBLE PRECISION array from a TTUTIL format data file and carries out a range check on the returned values. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDFDOR (XNAME, XMIN, XMAX, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDADOU, RDFDOU, RDADOR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	R8	I
XMAX	Maximum acceptable value	R8	I
X	Array itself of dimension ILDEC	R8	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFINR

Purpose:	Reads a fixed number of elements into an INTEGER array from a TTUTIL format data file and carries out a range check on the returned values. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDFINR (XNAME, XMIN, XMAX, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAINT, RDFINT, RDAINR

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	I4	I
XMAX	Maximum acceptable value	I4	I
X	Array itself of dimension ILDEC	I4	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDFRER

Purpose:	Reads a fixed number of elements into a REAL array from a TTUTIL format data file and carries out a range check on the returned values. Data file reading should be initialized with RDINIT or RDPARS. A number of values on file different from IVALS results in an error message. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDFRER (XNAME, XMIN, XMAX, X, ILDEC, IVALS)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDAREA, RDFREA, RDARER

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of array, for which data are on file	C*(*)	I
XMIN	Minimum acceptable value	R4	I
XMAX	Maximum acceptable value	R4	I
X	Array itself of dimension ILDEC	R4	O
ILDEC	Declared length of X	I4	I
IVALS	Number of values to be present on file	I4	I

Routine: RDSDOR

Purpose:	Reads a single DOUBLE PRECISION value from a TTUTIL format data file and carries out a range check on the returned value. The reading should be initialized with RDINIT or RDPARS. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDSDOR (XNAME, XMIN, XMAX, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSDOU

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
XMIN	Minimum acceptable value	R8	I
XMAX	Maximum acceptable value	R8	I
X	Value of variable	R8	O

Routine: RDSINR

Purpose:	Reads a single INTEGER value from a TTUTIL format data file and carries out a range check on the returned value. The reading should be initialized with RDINIT or RDPARS. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDSINR (XNAME, XMIN, XMAX, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSINT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
XMIN	Minimum acceptable value	I4	I
XMAX	Maximum acceptable value	I4	I
X	Value of variable	I4	O

Routine: RDSRER

Purpose:	Reads a single REAL value from a TTUTIL format data file and carries out a range check on the returned value. The reading should be initialized with RDINIT or RDPARS. If a value on the datafile is missing and the value for missing data that will be returned by this routine is outside the valid range, then this is not flagged.
Usage:	call RDSRER (XNAME, XMIN, XMAX, X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDSREA

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
XMIN	Minimum acceptable value	R4	I
XMAX	Maximum acceptable value	R4	I
X	Value of variable	R4	O

Routine: RDMDEF

Purpose:	Resets the value returned for missing data to the default values. Being –99.99 for reals, -99 for integers, -99.99D00 for double precision and date/time variables, '- MISSING -' for character strings, and .FALSE. for logicals.
Usage:	call RDMDEF
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMCHA, RDMREA, RDMDOU, RDMINT, RDMLOG, RDMTIM

Arguments:	Meaning	Data type	I and/or O
none			

Routine: RDMCHA

Purpose:	Sets the value to be returned for missing CHARACTER strings.
Usage:	call RDMCHA (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	C*(*)	I

Routine: RDMDOU

Purpose:	Sets the value to be returned for missing DOUBLE PRECISION reals.
Usage:	call RDMDOU (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	R8	i

Routine: RDMINT

Purpose:	Sets the value to be returned for missing INTEGERs.
Usage:	call RDMINT (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	I4	I

Routine: RDMLOG

Purpose:	Sets the value to be returned for missing LOGICALs.
Usage:	call RDMLOG (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	L4	I

Routine: RDMREA

Purpose:	Sets the value to be returned for missing REALs.
Usage:	call RDMREA (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	R4	I

Routine: RDMTIM

Purpose:	Sets the value to be returned for missing DATE/TIME.
Usage:	call RDMTIM (X)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDMDEF

Arguments:	Meaning	Data type	I and/or O
X	Missing value to be returned when a value is missing on the data file	R8	I

10.2. Writing of TTUTIL format datafiles

Routine: WRINIT

Purpose:	Initializes the WR system to write data in TTUTIL format to a datafile. The specified file is left open for writing by the other WR routines. After closing the file, it can be reread with the RDINIT or RDPARS routine.
Usage:	call WRINIT (UNIT_X, FILE)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
UNIT_X	Unit number to open file with	I4	I
FILE	File name to use for output	C*(*)	I

Routine: WRACHA

Purpose:	Writes the character array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRACHA (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	C*(*)	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRADOU

Purpose:	Writes the double precision array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRADOU (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	R8	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRAINT

Purpose:	Writes the integer array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRAINT (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	I4	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRALOG

Purpose:	Writes the logical array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRALOG (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	L4	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRAREA

Purpose:	Writes the real array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRAREA (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	R4	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRATIM

Purpose:	Writes the date/time array XNAME to the output file in TTUTIL format. The output file has to be openened first with WRINIT.
Usage:	call WRATIM (XNAME, X, ILDEC, IFND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Array with values of dimension ILDEC	R8	I
ILDEC	Declared length of X	I4	I
IFND	Number of values to write to file	I4	I

Routine: WRSCHA

Purpose:	Writes a single CHARACTER value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSCHA (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	C*(*)	I

Routine: WRSDOU

Purpose:	Writes a single DOUBLE PRECISION value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSDOU (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R8	I

Routine: WRSINT

Purpose:	Writes a single INTEGER value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSINT (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I

X	Value of variable	I4	I

Routine: WRSLOG

Purpose:	Writes a single LOGICAL value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSLOG (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	L4	I

Routine: WRSREA

Purpose:	Writes a single REAL value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSREA (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R4	I

Routine: WRSTIM

Purpose:	Writes a single date/time value to a data file. The output file has to be openened first with WRINIT.
Usage:	call WRSTIM (XNAME, X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WRINIT

Arguments:	Meaning	Data type	I and/or O
XNAME	Name of variable	C*(*)	I
X	Value of variable	R8	I

10.3. Interactive input

Routine: ENTCHA

Purpose:	Interactive entry of a character string. Writes the text QUEST on screen as a "question" and returns the entered string to the calling program.
Usage:	call ENTCHA (QUEST, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTDCH

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
X	Entered character string	C*(*)	O

Routine: ENTDCH

Purpose:	Interactive entry of a CHARACTER string with a default. Writes the text QUEST on screen as a "question" and returns the entered string to the calling program.
Usage:	call ENTDCH (QUEST, SDEF, S)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTCHA

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
SDEF	Default string, assumed when <Return> is given	C*(*)	I
S	Entered CHARACTER string	C*(*)	O

Routine: ENTINT

Purpose:	Interactive entry of an INTEGER number Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTINT (QUEST, IX)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTDIN

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
IX	Entered number	I4	O

Routine: ENTDIN

Purpose:	Interactive entry of an INTEGER number with a default. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTDIN (QUEST, IXDEF, IX)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTINT

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
IXDEF	Default value assumed when <Return> is given	I4	I
IX	Entered INTEGER number	I4	O

Routine: ENTREA

Purpose:	Interactive entry of a REAL number. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTREA (QUEST, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTDRE

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
X	Entered REAL number	R4	O

Routine: ENTDRE

Purpose:	Interactive entry of a REAL number with a default. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTDRE (QUEST, XDEF, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ENTREA

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
XDEF	Default value assumed when <Return> is given	R4	I
X	Entered REAL number	R4	O

Routine: ENTDOU

Purpose:	Interactive entry of a DOUBLE PRECISION number. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTDOU (QUEST, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTDDO

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
X	Entered DOUBLE PRECISION number	R8	O

Routine: ENTDDO

Purpose:	Interactive entry of a DOUBLE PRECISION number with a default. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program.
Usage:	call ENTDDO (QUEST, XDEF, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTDOU

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
XDEF	Default value assumed when <Return> is given	R8	I
X	Entered DOUBLE PRECISION number	R8	O

Routine: ENTYNO

Purpose:	Interactive entry of a boolean Yes / No, given by hitting a single Y or N key. Writes the text QUEST on screen as a "question" and returns the entered value to the calling program. A ‘Y’ is returned as a .TRUE., a ‘N’ as .FALSE..
Usage:	CALL ENTYNO (QUEST, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTDYN

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
X	.TRUE. when Y was entered, .FALSE. when N was entered	L4	O

Routine: ENTDYN

Purpose:	Interactive entry of a boolean Yes / No, given by hitting a single Y or N key, with a default. Writes the text QUEST on screen as a "question" and returns the entered value to the calling program. A ‘Y’ is returned as a .TRUE., a ‘N’ as .FALSE..
Usage:	call ENTDYN (QUEST, XDEF, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTYNO

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
XDEF	Default value assumed when <Return> is given	L4	I
X	.TRUE. when Y was entered, .FALSE. when N was entered	L4	O

Routine: ENTTIM

Purpose:	Interactive entry of a date / time value. Writes the text QUEST on screen as a "question" and returns the entered number to the calling program. Allowed formats are given in Section 5.2.5.
Usage:	call ENTTIM (QUEST, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTDTI, date and time routines in Section 10.11.

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
X	Entered Date / Time value	R8	O

Routine: ENTDTI

Purpose:	Interactive entry of a date / time value with a default. Writes the text QUEST on screen as a "question" and returns the entered data / time to the calling program. Allowed formats are given in Section 5.2.5.
Usage:	call ENTDTI (QUEST, XDEF, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	1-July-1999
See also:	ENTTIM, date / time routines in Section 5.2.5.

Arguments:	Meaning	Data type	I and/or O
QUEST	Character string, for instance 'Give the value for P'	C*(*)	I
XDEF	Default value assumed when <Return> is given	R8	I
X	Entered Date / Time value	R8	O

10.4. Output to file

Routine: COPFL2

Purpose:	Copies the contents of a file to an output file with unit number IOUT (the output file should already be open and is left open). The input file is opened by COPFL2 and closed after the contents has been copied. If the input file does not exist, nothing is done.
Usage:	call COPFL2 (IIN, FILE, IOUT, HEADER)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WR* routines

Arguments:	Meaning	Data type	I and/or O
IIN	Unit number to be used to open input file with	I4	I
FILE	File name of input file	C*(*)	I
IOUT	Unit number of file where input file should be copied to	I4	I
HEADER	Flags if information header should be written to the output file: =.true., header is written =.false., header is not written	L4	I

Routine: OUTAR2

Purpose:	This routine transfers the contents of an array to the subroutine OUTDAT which can handle only single names and values. The OUTAR2 call works like a series of calls to OUTDAT with single array elements. For example the following calls to OUTDAT: CALL OUTDAT (2,0,’ A(1) ’,A(1)) CALL OUTDAT (2,0,’ A(2) ’,A(2)) CALL OUTDAT (2,0,’ A(3) ’,A(3)) can be abbreviated by a single call with OUTAR2: CALL OUTAR2 (‘A’,A,1,3,1,3) Array subscripts between –99 and 999 are accepted.
Usage:	call OUTAR2 (NAME, ARRAY, LDEC, UDEC, LST, UST)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	OUTDAT, OUTPLT

Arguments:	Meaning	Data type	I and/or O
NAME	Name of array to be written	C*(*)	I
ARRAY	Array itself of at least dimension I2	R4	I
LDEC	Lower declared bound of ARRAY	I4	I

UDEC	Upper declared bound of ARRAY	I4	I
I1	Array element where output should start	I4	I
I2	Array element where output should finish	I4	I

Routine: OUTCOM

Purpose:	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. Repeated input of the same string is neglected. For 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.
Usage:	call OUTCOM (STR)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	OUTDAT, OUTPLT

Arguments:	Meaning	Data type	I and/or O
STR	Text string	C*(*)	I

Routine: OUTDAT

Purpose:	Can be used to generate output tables in files from within simulation models. It should be initialized first, to define the name of the independent variable (the variable printed in the leftmost column) and to set unit numbers (ITASK=1). The name and value of the data are stored by a series of calls with ITASK=2. Each call supplies one name and value to the OUTDAT system. The stored data can be converted to an output file by a call with ITASK=4, 5, or 6. Storage, prior to generating a table is on disk. A maximum of 500 names can be stored, the number of values depends on free disk space. After generation of the output, OUTDAT is ready to be initialized again, if necessary with another independent variable. This initialization, storing of and creation of the output table may be repeated many times. Another possibility is to repeatedly initialize OUTDAT (ITASK=1) and store data (ITASK=2) but to have all the output tables created by a single call to OUTDAT shortly before termination of the program. In that case ITASK values of 14, 15 and 16 should be used. If the reruns are carried out with the RD system, then the selected set number will be printed above the generated output tables.
Usage:	CALL OUTDAT (ITASK, IUNIT, VARNAME, VARVALUE)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	OUTPLT

Arguments:	Meaning		Data type	I and/or O
ITASK	=1,	Initializes the OUTDAT system, opens a temporary file for storage and stores the name of the independent variable and the unit numbers for file I/O. OUTDAT may be reinitialized after an output ‘session’, even with a new independent variable name.	C*(*)	I
	=2,	Stores the name and value of the variable in the temporary file.		
	=3,	Selects the variables for writing to the output file.		
	=4,	Generates an output table in the output file, the independent variable and the first 8 dependent variables in the first block of output, the independent variable and the second set of 8 variables in the second block etc. untill all variables have been printed.		
	=5,	Generates a similar table, but now the values and names are separed by tab characters and the output blocks have a maximum of 100 independent variables. This output is useful for importing in spreadsheets.		
	=6,	Generates a two column output table, the independent variable and the first dependent variable. Below that the independent variable and the second dependent variable		

	etc.		
	=14, Creates output tables like ITASK=4 but now for all sets in the temporary output file.		
	=15, idem 14 but now with spreadsheet output		
	=16, idem 14 but now with two column output		
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 for I/O to the temporary file.	I	I
VARNAME	String, name of variable or meaningfull text, (up to 36 characters will be used). If ITASK is 4, 5, 6, 14, 15, or 16 this string will be written to the output file as title (not limited to 36 characters then).	C*(*)	I
VARVALUE	Value of variable (only used at ITASK=2).	R4	I

Routine: OUTPLT

Purpose:	Designed to be used in conjunction with OUTDAT, which is used to write variable names and values to a temporary file. OUTPLT is used to printplot a selection of the stored variables. By repeated calls to the OUTPLT subroutine with ITASK=1, names of variables 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). The printplot pertains to the last set of the temporary file. If one wants multiple plots for all sets in the temporary file, ITASK values of 14, 15, 16 or 17 should be used. For example, define DTGA and WSO to be plotted and create printplot using wide format, common scaling: CALL OUTPLT (1,'DTGA') CALL OUTPLT (1,'WSO') CALL OUTPLT (5,'Plot title')
Usage:	call OUTPLT (ITASK, RN)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	OUTDAT

Arguments:	Meaning	Data type	I and/or O
ITASK	Defines task of the subroutine:	I4	I
	=1, instruct the routine to store variable names for use in the printplot		
	=4, wide format, individual scale plot of last set		
	=5, wide format, common scale plot of last set		
	=6, small format, individual scale plot of last set		
	=7, small format, common scale plot of last set		
	=14, wide format, individual scale plot of all sets		
	=15, wide format, common scale plot of all sets		
	=16, small format, individual scale plot of all sets		

	=17, small format, common scale plot of all sets 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	when ITASK = 1: name of variable to be plotted, up to 36 characters will be used. The value of the variable must have been stored by previous calls to OUTDAT. When ITASK <>1 the text is printed above the plot(s).	C*(*)	I

Routine: OUTSEL

Purpose:	Performs a sequence of OUTDAT calls in order to generate the table(s) specified in the variable selection array PRSEL. When there are no variables selected, a single call to OUTDAT produces table(s) using default variable order. This routine avoids having to do multiple calls to OUTDAT to select variable to appear in the output
Usage:	call OUTSEL (PRSEL, IMNPRS, INPRS, IPFORM, MESSAG)
Author(s):	Daniël van Kraalingen, Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	OUTDAT

Arguments:	Meaning	Data type	I and/or O
PRSEL	Character string array with names of selected variables	C*(*)	I
IMNPRS	Declared length of array PRSEL	I4	I
INPRS	Used length of array PRSEL	I4	I
IPFORM	Controls OUTDAT table format (see OUTDAT header)	I4	I
MESSAG	Text message for OUTDAT call	C*(*)	I

10.5. File and unit handling

Routine: DELFIL

Purpose:	Deletes the specified file name and can (if flag is turned on) give an error if the file does not exist
Usage:	call DELFIL (FILE_NAME, ERR)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FLEXIST

Arguments:	Meaning	Data type	I and/or O
FILE_NAME	Name of file to be deleted	C*(*)	I

ERR	If .TRUE. causes fatal error when the file does not exist	L4	I

Routine: EXTENS

Purpose: Changes extension of filename. Output filename is in uppercase characters. The old extension is the part of the filename that follows a dot in the filename (.). A dot within a directory name is neglected (bracket], colon :, backslash \, slash / on VAX, Macintosh, MS-DOS and Unix respectively). The input filename does not necessarily has to have an extension. For example:

FILEIN	NEWEXT	FILEOU
Name	dat	NAME.DAT
Name.dat	log	NAME.LOG
Name.dat	log	NAME.LOG
Name.dat	.	NAME
DISK$USER:[AC.MINE]Name	dat	DISK$USER:[AC.MINE]NAME.DAT
HD:Mine.Old:Name	dat	HD:MINE.OLD:NAME.DAT
C:\MINE.OLD\Name	dat	C:\MINE.OLD\NAME.DAT
D:/MINE.OLD/Name	dat	D:/MINE.OLD/NAME.DAT

Usage:	call EXTENS (FILEIN, NEWEXT, ICHECK, FILEOU)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FOPENS

Arguments:	Meaning	Data type	I and/or O
FILEIN	Input filename with old or without extension	C*(*)	I
NEWEXT	New extension, is set to uppercase	C*(*)	I
ICHECK	=1, check on equal output and input extension =0, no check	I4	I
FILEOU	Output filename with new extension in uppercase	C*(*)	O

Routine: FLEXIST

Purpose:	Returns a flag whether the supplied filename exists
Usage:	<logical variable> = FLEXIST (FILE_NAME)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DELFIL

Arguments:	Meaning	Data type	I and/or O
FILE_NAME	Filename to check	C*(*)	I
FLEXIST	Flag whether file exists or not	L4	O

Routine: FLNAME

Purpose:	Prepares file name for opening (for now only carries out a LOWERC call). This routine is called inside from TTUTIL from every routine which accepts a file name. This routine ensures that conflicts will not arise on file systems which are case sensitive.
Usage:	call FLNAME (STRING)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FOPENS

Arguments:	Meaning	Data type	I and/or O
STRING	File name to be processed	C*(*)	I/O

Routine: FOPENG

Purpose: Opens formatted, unformatted or binary files with sequential or direct access. Using FOPENG garantees portability on platforms which have a case sensitive file system. This is achieved by a lowercase operation of the file name inside the FOPENG routine. For example:

```
CALL FOPENG (20,'a.dat','old','fs',0,' ')
```
Opens an existing formatted sequential file.

```
CALL FOPENG (20,'a.dat','new','ud',10,'unk')
```
Creates a new, unformatted, direct access file with a record length of 10 bytes ; in case a file a.dat already exists the routine needs either permission to overwrite or a new filename

```
CALL FOPENG (20,'a.dat','new','ud',10,'del')
```
Creates new, unformatted, direct access file with a record length of 10 bytes ; a possibly existing file a.dat is deleted

Usage:	call FOPENG (IUNIT, FILNAM, STATUS, TYPE, IRECL, PRIV)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FOPENS

Arguments:	Meaning	Data type	I and/or O
IUNIT	Unit number used to open file	I4	I
FILNAM	Name of the file to be opened. Inside FOPENG, FILNAM is converted to lowercase.	C*(*)	I
STATUS	Status of the file	C*(*)	I

	='old', existing file is opened		
	='new', new file is created (see PRIV)		
	='rdo', existing file is opened with write protection ; this only works on the VAX, on other machines RDO=OLD		
TYPE	String containing code for FORM keyword (F,U or B) and code for ACCESS keyword (S or D).	C*(*)	I
	='F', formatted file		
	='U', unform atted file		
	='B', binary filetype (Microsoft Fortran only)		
	='S', sequential access		
	='D', direct access		
IRECL	Record length of direct access files in BYTES (see also machine dependent parameter IWLEN below) may be dummy value in case of sequential files	I4	I
PRIV	Privilege ; in case status='new' and file exists:	C*(*)	I
	='del', old file is overwritten		
	='nod', old file saved, program stopped		
	='unk', in case file exists, one may either overwrite it or give a new filename (interactive choice)		

Routine: FOPENS

Purpose: Opens a formatted, sequential file (the normal ascii text files) by internally calling FOPENG. See FOPENG for details pertaining to formatted sequential files. For example:

CALL FOPENS (20,'a.dat','old',' ')
Opens an existing formatted sequential file

CALL FOPENS (20,'a.dat','new','unk')
Creates a new, formatted, sequential file ; in case a file a.dat already exists the routine asks permission to overwrite.

CALL FOPENS (20,'a.dat','new','del')
Creates new, formatted, sequential file ; a possibly existing file a.dat is deleted

Usage: call FOPENS (IUNIT, FILNAM, STATUS, PRIV)
Author(s): Kees Rappoldt, Daniël van Kraalingen
Version: 1.0, TTUTIL version 4.13
Date: 30-September-1997
See also: FOPENG

Arguments:	Meaning	Data type	I and/or O
IUNIT	Unit number used to open file	I4	I
FILNAM	Name of the file to be opened	C*(*)	I
STATUS	Status of the fileI	C*(*)	I
	='old', existing file is opened		
	='new', new file is created (see PRIV)		

	='rdo', existing file is opened with write protection ; this only works on the VAX, on other machines RDO=OLD		
PRIV	Privilege, in case status='new' and file exists:	C*(*)	I
	='del', old file is overwritten		
	='nod', old file saved, program stopped		
	='unk' in case file exists, one may either overwrite it or give a new filename (interactive choice)		

Routine: GETUN

Purpose:	Gets a free unit number within range IST and IEND (inclusive), an error occurs if a free unit number cannot be found.
Usage:	<integer variable> = GETUN (IST, IEND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	GETUN2, USEDUN

Arguments:	Meaning	Data type	I and/or O
IST	Unit number where search should start	I4	I
IEND	Idem where search should end	I4	I
GETUN	Returned free unit number	I4	O

Routine: GETUN2

Purpose:	Gets a range of NUM free unit number within range IST and IEND (inclusive). An error occurs If a free unit number cannot be found.
Usage:	<integer variable> = GETUN2 (IST, IEND, NUM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	GETUN, USEDUN

Arguments:	Meaning	Data type	I and/or O
IST	Unit number where search should start	I4	I
IEND	Idem where search should end	I4	I
NUM	Number of required consecutive free unit numbers	I4	I
GETUN2	First of sequence of free unit numbers	I4	O

Routine: USEDUN

Purpose:	Checks a range of unit numbers whether they are in use for file i/o. Checking starts at the value of IST and ends at the value of IEND (inclusive). If that unit is used, a report on which file uses which unit is written to the screen. If IST is greater than IEND, the search goes from high numbers to low numbers.
Usage:	call USEDUN (IST, IEND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	GETUN, GETUN2

Arguments:	Meaning	Data type	I and/or O
IST	Unit number where to start search	I4	I
IEND	Unit number where to end search	I4	I

10.6. Character string handling

Routine: ADDINF

Purpose:	Adds integer I to existing STRING, using format FORM. Routine does *not* remove leading and trailing spaces. Also updates the significant length. For example: CALL ADDINF (STRING,SIGLEN,25,'I4.4') Adds the value 25 with the I4.4 format to STRING, the significant length of STRING is increased by 4. If the input string is 'example', then after the call, the string is 'example0025'.
Usage:	call ADDINF (STRING, SIGLEN, I, FORM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDINT, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING (is updated after concatenation)	I4	I/O
I	Integer to be added	I4	I
FORM	Format to be used (without brackets)	C*(*)	I

Routine: ADDINT

Purpose:	Adds integer I to existing STRING, without leading and trailing spaces. Also updates the significant length. For example: CALL ADDINT (STRING,SIGLEN,25) Adds the value 25 to STRING, the significant length of STRING is increased by 2. If the input string is ‘example’, then after the call, the string is ‘example25’.
Usage:	call ADDINT (STRING, SIGLEN, I)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDINF, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING	I4	I/O
I	Integer to be added	I4	I

Routine: ADDREA

Purpose:	Adds real R to existing STRING, using format FORM. Routine removes leading and trailing spaces. Also updates the significant length. For example: CALL ADDREA (STRING,SIGLEN,25.’f3.0’) Adds the value 25. to STRING using the format f3.0, the significant length of STRING is increased by 3. If the input string is ‘example’, then after the call, the string is ‘example25.’.
Usage:	call ADDREA (STRING, SIGLEN, R, FORM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDREF, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING	I4	I/O
R	Real to be added	R4	I
FORM	Format to be used (without brackets)	C*(*)	I

Routine: ADDREF

Purpose:	Adds real R to existing STRING, using format FORM. Routine does *not* remove leading and trailing spaces. Also updates the significant length. For example (a '-' signifies a space): CALL ADDREF (STRING,SIGLEN,25.'f8.0') Adds the value 25. to STRING using the format f3.0, the significant length of STRING is increased by 8. If the input string is 'example', then after the call, the string is: 'example-----25.'.
Usage:	call ADDREF (STRING, SIGLEN, R, FORM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDREA, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING	I4	I/O
R	Real to be added	R4	I
FORM	Format to be used (without brackets)	C*(*)	I

Routine: ADDSTF

Purpose:	Adds string TMP to existing STRING, WITH leading and trailing spaces. Also updates the significant length. For example (a '-' signifies a space): CALL ADDSTF (STRING,SIGLEN,'---example---') Adds the string '---example---' to STRING, the significant length of STRING is increased by 13. If the input string is 'example', then after the call, the string is: 'example---example---'.
Usage:	call ADDSTF (STRING, SIGLEN, TMP)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDSTR, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING	I4	I/O
TMP	String to be added	C*(*)	I

Routine: ADDSTR

Purpose:	Adds string TMP to existing STRING, *without* leading and trailing spaces. Also updates the significant length. For example (a '-' signifies a space): CALL ADDSTR (STRING,SIGLEN,'---example---') Adds the string 'example' to STRING, the significant length of STRING is increased by 7. If the input string is 'example', then after the call, the string is: 'exampleexample'.
Usage:	call ADDSTR (STRING, SIGLEN, TMP)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADDSTF, STR_COPY

Arguments:	Meaning	Data type	I and/or O
STRING	String to which concatenation should take place	C*(*)	I/O
SIGLEN	Significant length of STRING	I4	I/O
TMP	String to be added	C*(*)	I

Routine: ILEN

Purpose:	Determines the significant length of a string. A zero is returned if the string is empty. As this version of TTUTIL is the last fully FORTRAN-77 compatible library, you are advised to change calls to ILEN into calls to the Fortran 90 intrinsic function LEN_TRIM as soon as you have migrated to the Fortran 90 environment.
Usage:	<integer variable> = ILEN (STRING)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ISTART

Arguments:	Meaning	Data type	I and/or O
STRING	Input string	C*(*)	I
ILEN	Returned significant length	I4	O

Routine: ISTART

Purpose:	Determines the first significant character of a string. If the string contains no significant characters, a zero is returned. The functionality of this routine is similar to ILEN, except that now the first non-space character is returned instead of the last.
Usage:	<integer variable> = ISTART (STRING)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ILEN

Arguments:	Meaning	Data type	I and/or O
STRING	Input string	C*(*)	I
ISTART	Returned position of first non-space character	I4	I

Routine: LOWERC

Purpose:	Converts a string to lowercase characters
Usage:	call LOWERC (STRING)
Author(s):	Daniël van Kraalingen , Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	UPPERC

Arguments:	Meaning	Data type	I and/or O
STRING	Character string	C*(*)	I/O

Routine: UPPERC

Purpose:	Converts a string to uppercase characters
Usage:	call UPPERC (STRING)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	LOWERC

Arguments:	Meaning	Data type	I and/or O
STRING	character string	C*(*)	I/O

Routine: REMOVE

Purpose:	Replaces all unwanted characters in a string with a <space>. For example, if "e" is removed from "bicentennial", the result is "bic nt nnial".
Usage:	call REMOVE (STRING, CHR)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
STRING	String that is used	C*(*)	I/O
CHR	Character to be removed	C*1	I

Routine: STR_COPY

Purpose:	Copies an input string to an output string. The added value is that the routine checks that the significant part of the input string fits on the output string.
Usage:	call STR_COPY (SOURCE_S, TARGET_S, OK)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	ADD* routines

Arguments:	Meaning	Data type	I and/or O
SOURCE_S	Input string	C*(*)	I
TARGET_S	Output string	C*(*)	O
OK	Flag whether input string fits on output string	L4	O

Routine: WORDS

Purpose:	Looks for start and end positions of words on a string. Valid separators can be supplied by the user. A sequence of separators is treated as one separator. For example if the string is: ‘a,b, c’ and commas and spaces are used as separator, WORDS will find the first word at position 1 to 1, the second word at position 3 to 3 and the third word at position 9.
Usage:	call WORDS (RECORD, ILW, SEPARS, IWBEG, IWEND, IFND)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DECREC

Arguments:	Meaning	Data type	I and/or O
RECORD	Character string where words are to be located	C*(*)	I
ILW	Maximum number of words which can be found (=declared length of IWBEG and IWEND arrays)	I4	I
SEPARS	String containing separator characters	C*(*)	I
IWBEG	Integer array of dimension ILW containing start positions on return	I4	O
IWEND	Integer array of dimension ILW containing end positions on return	I4	O
IFND	Integer containing the number of words found	I4	O

Routine: RCHRSRC

Purpose:	Takes a character backward from a string and checks whether it matches with one of the characters supplied by the user. If a match is not found, the next character from the input string is compared. For example if the string is ‘myexample’, and the characters to find are ‘ma’, then a match is found at the m on position 6.
Usage:	<integer variable> = RCHRSRC (CHARS, STRING, POSBEG,POSEND)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	IFINDC, SFINDG

Arguments:	Meaning	Data type	I and/or O
CHARS	String of characters with which match may occur (the string ‘ma’ in the above example)	C*(*)	I
STRING	String where match is looked for in backward position (the string ‘myexample’ in the above example)	C*(*)	I
POSBEG	Position where search should stop (inclusive)	I4	I
POSEND	Position where search should start (inclusive)	I4	I
RCHRSRC	Returned position of match	I4	O

10.7. Decoding of character strings to values

Routine: DECCHK

Purpose:	Checks if a string is a number (a number here is defined as either a real or integer value)
Usage:	<logical variable> = DECCHK (STRING)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	2-September-1999
See also:	DECINT, DECREA

Arguments:	Meaning	Data type	I and/or O
STRING	Input string, NO trailing or leading blanks !	C*(*)	I
DECCHK	Returned value is .true. if the string is a real or integer number, otherwise a .false. is returned	L4	O

Routine: DECDOU

Purpose:	Decodes a number from a character string into a double precision variable
Usage:	call DECDOU (IWAR, STRING, VALUE)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	13-March-2002
See also:	DECCHK, DECINT, DECREA

Arguments:	Meaning	Data type	I and/or O
IWAR	In case of error IWAR = 1, otherwise IWAR = 0, can be used to check the success of the decoding	I4	O
STRING	Input string, NO trailing or leading blanks are allowed	C*(*)	I
VALUE	Double precision value read from string, if decoding failed (when IWAR is returned as 1), a zero for VALUE is returned.	R8	O

Routine: DECINT

Purpose:	Decodes an integer number from a character string into an integer variable
Usage:	call DECINT (IWAR, STRING, IVALUE)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DECCHK, DECREA, DECDOU

Arguments:	Meaning	Data type	I and/or O
IWAR	In case of error IWAR = 1, otherwise IWAR = 0, can be used to check the success of the decoding	I4	O
STRING	Input string, NO trailing or leading blanks are allowed	C*(*)	I
IVALUE	Integer value read from string, if decoding failed (when IWAR is returned as 1), a zero for IVALUE is returned	I4	O

Routine: DECREA

Purpose:	Decodes a number from a character string into a single precision real value
Usage:	call DECREA (IWAR, STRING, VALUE)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DECCHK, DECINT, DECDOU

Arguments:	Meaning	Data type	I and/or O
IWAR	In case of error IWAR = 1, otherwise IWAR = 0, can be used to check the success of the decoding	I4	O
STRING	Input string, NO trailing or leading blanks are allowed	C*(*)	I
VALUE	Real value read from string, if decoding failed (when IWAR is returned as 1), a zero for VALUE is returned.	R4	O

Routine: DECREC

Purpose:	Locates and decodes from the character string RECORD at most ILX real numbers. Numbers are separated by blanks(s) and/or comma(s).
Usage:	call DECREC (RECORD, ILX, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	WORDS, DECREA, DECINT, DECCHK

Arguments:	Meaning	Data type	I and/or O
RECORD	Character string	C*(*)	I
ILX	Number of REAL numbers to be decoded	I4	I
X	Array of dimension ILX containing the decoded results	R4	O

10.8. Messages and Errors

Routine: FATALERR

Purpose:	Writes an error message. If screen output is enabled it holds the screen until the <ENTER> key is pressed. Then execution is terminated.
Usage:	call FATALERR (ROUTINE, MESSAG)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	MESSWRT, WARNING, MESSINI

Arguments:	Meaning	Data type	I and/or O
ROUTINE	String containing the routine name	C*(*)	I
MESSAG	String containing the message	C*(*)	O

Routine: MESSINI

Purpose:	Sets screen and logfile use for all TTUTIL routines. After MESSINI, the RD* routines neglect the logfile units in RDINIT and RDSETS calls. MESSINI can be reset to default behaviour by setting TOLOG=.true. and UNLOG=0.
Usage:	call MESSINI (TOSCR, TOLOG, UNLOG)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	18-April-2002
See also:	MESSINQ

Arguments:	Meaning	Data type	I and/or O
TOSCR	Flag controlling message display on screen	L4	I
TOLOG	Flag controlling message display on logfile	L4	I
UNLOG	Unit number of open logfile	I4	I

Routine: MESSINQ

Purpose:	Inquiry for screen and logfile settings
Usage:	call MESSINI (TOSCR, TOLOG, UNLOG)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	18-April-2002
See also:	MESSINI

Arguments:	Meaning	Data type	I and/or O
TOSCR	Flag controlling message display on screen	L4	O
TOLOG	Flag controlling message display on logfile	L4	O
UNLOG	Unit number of open logfile	I4	O

Routine: MESSWRT

Purpose:	Writes a message to screen and/or logfile. Execution is not terminated.
Usage:	call MESSWRT (ROUTINE, MESSAG)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	18-April-2002
See also:	FATALERR, WARNING, MESSINI

Arguments:	Meaning	Data type	I and/or O
ROUTINE	String containing the routine name	C*(*)	I
MESSAG	String containing the message	C*(*)	O

Routine: WARNING

Purpose:	Writes a warning message to screen and/or logfile. Execution is not terminated.
Usage:	call WARNING (ROUTINE, MESSAG)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FATALERR, MESSWRT, MESSINI

Arguments:	Meaning	Data type	I and/or O
ROUTINE	String containing the routine name	C*(*)	I
MESSAG	String containing the message	C*(*)	I

Routine: OPENLOGF

Purpose:	Opens a logfile, writes date/time, program name, version and author in it and (optional) a TTUTIL version message. Then MESSINI is called and the generated logfile unit is available via MESSINQ.
Usage:	call OPENLOGF (TOSCR, NAME, PROGNAM, VRSSTR, AUTHOR, TTNOTE)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	7-May-2002
See also:	MESSINI, MESSINQ

Arguments:	Meaning	Data type	I and/or O
TOSCR	Flag controlling message display on screen	L4	I
NAME	Name of logfile, extension .log is added if absent	C*(*)	I
PROGNAM	Name of calling program	C*(*)	I
VRSSTR	Version string	C*(*)	I
AUTHOR	Author(s)	C*(*)	I
TTNOTE	Flag for TTUTIL version message	L4	I

10.9. Version routines

Routine: TTUVER

Purpose:	Verifies that the minimal TTUTIL version required by the calling program matches the linked version of the TTUTIL library. Also returns the current TTUTIL version.
Usage:	call TTUVER (MIN_V, CUR_V)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	VER4_12

Arguments:	Meaning	Data type	I and/or O
MIN_V	Minimum version of TTUTIL required for the calling program	R4	I
CUR_V	Current version of TTUTIL	R4	O

Routine: VER4_13

Purpose:	This routine is a dummy routine meant to be able to see in a TTUTIL library file what the version of that library is.
Usage:	
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	TTUVER

Arguments:	Meaning	Data type	I and/or O
none			

10.10. Numeric functions

Routine: FCNSW

Purpose:	Input switch depending on sign of X1 ; function is equivalent to the CSMP-FCNSW. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = FCNSW (X1, X2, X3, X4)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	INSW

Arguments:	Meaning	Data type	I and/or O
X1	Variable upon which the test is done	R4	I
X2	Value of FCNSW in case X1 < 0	R4	I
X3	Value of FCNSW in case X1 = 0	R4	I
X4	Value of FCNSW in case X1 > 0	R4	I
FCNSW	Returned value	R4	O

Routine: INSW

Purpose:	Input switch depending on sign of X1 ; function is equivalent to the CSMP-INSW. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = INSW (X1, X2, X3)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	FCNSW

Arguments:	Meaning	Data type	I and/or O
X1	Identifier upon which the test is done	R4	I
X2	Value of INSW in case X1 < 0	R4	I
X3	Value of INSW in case X1 >= 0	R4	I
INSW	Returned value	R4	O

Routine: INTGRL

Purpose:	Function value = STATE + RATE * DELT. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = INTGRL (STATE, RATE, DELT)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
STATE	Old state	R4	I
RATE	Rate of change per unit time	R4	I
DELT	Time step	R4	I
INTGRL	Function name, new state	R4	O

Routine: LIMIT

Purpose:	Returns value of X limited within the interval [MIN,MAX]. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = LIMIT (MIN, MAX, X)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	INSW, FCNSW

Arguments:	Meaning	Data type	I and/or O
MIN	Interval lower boundary	R4	I
MAX	Interval upper boundary	R4	I
X	Variable that should be limited	R4	I
LIMIT	Limited value	R4	O

Routine: LINT

Purpose:	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. The preferred routine for linear interpolation, however, is LINT2 which gives better error and warning texts. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = LINT (TABLE, ILTAB, X)
Author(s):	Daniël van Kraalingen , Kees Rappoldt,
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	LINT2

Arguments:	Meaning	Data type	I and/or O
TABLE	A one-dimensional array of dimension ILTAB with paired data: x1,y1,x2,y2, etc.	R4	I
ILTAB	Number of elements in the array TABLE	I4	I
X	The value at which interpolation should take place	R4	I
LINT	Function name, result of the interpolation	R4	O

Routine: LINT2

Purpose:	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. LINT2 is the preferred routine for linear interpolation. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = LINT2 (TABNAM, TABLE, ILTAB, X)
Author(s):	Daniël van Kraalingen , Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	LINT

Arguments:	Meaning	Data type	I and/or O
TABNAM	Table name	C*(*)	I
TABLE	A one-dimensional array of dimension ILTAB with paired data:	R4	I

	x1,y1,x2,y2, etc.		
ILTAB	Number of elements in the array TABLE	I4	I
X	The value at which interpolation should take place	R4	I
LINT2	Function name, result of the interpolation	R4	O

Routine: MOVAVR

Purpose:	Calculates a moving average of the last IP points, MOVAVR can keep simultaneous moving averages of 10 different variables, distinguished by their names. For each variable, a maximum moving average number of 100 points can be handled.
Usage:	call MOVAVR (ITASK, NAME, IP, IN, OUT)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
ITASK	Task to be carried out: =1, initialize =2, calculate moving average	I4	I
NAME	Variable name for which moving average should be calculated	C*(*)	I
IP	Number of previous values (including the current value) on which moving average should be calculated	I4	I
IN	New value	R4	I
OUT	Moving average for NAME (if MOVAVR has not yet stored IP values for NAME, then the moving average is calculated from the fewer number of points)	R4	O

Routine: NOTNUL

Purpose:	This function can be used to avoid divide by zero errors in divisions. The function result is defined as: NOTNUL = X, when X <> 0 NOTNUL = 1, when X = 0. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = NOTNUL (X)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	LIMIT, INSW, FCNSW

Arguments:	Meaning	Data type	I and/or O
X	Input argument	R4	I
NOTNUL	Function result	R4	O

Routine: REAAND

Purpose:	This function emulates the CSMP function AND. It is similar to a 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. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = REAAND (X1, X2)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	REANOR

Arguments:	Meaning	Data type	I and/or O
X1	first argument	R4	I
X2	second argument	R4	I
REAAND	Function result	R4	O

Routine: REANOR

Purpose:	This function emulates the CSMP function NOR. It is similar to the 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. (See also Rappoldt and van Kraalingen, 1996).
Usage:	<real variable> = REANOR (X1, X2)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	REAAND

Arguments:	Meaning	Data type	I and/or O
X1,	first argument	R4	I
X2	second argument	R4	I
REANOR	Function result	R4	O

10.11. Date/time

Routine: DTARDP

Purpose:	Converts DATEA, FSEC representation of a Date/Time to the double precision representation. This routine is the opposite from DTDPAR.
Usage:	call DTARDP (DATEA, FSEC, DPDTTM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DTDPAR, DTFSEDP

Arguments:	Meaning	Data type	I and/or O
DATEA	Integer array of six elements containing year, month, day, hour, minute and seconds	I4	I
FSEC	Fractional seconds	R4	I
DPDTTM	Converted date/time to a double precision date/time (represented as the number of days since 1/1/1900)	R8	O

Routine: DTDPAR

Purpose:	Converts the double precision representation of a Date/Time to DATEA, FSEC representation. This routine is the opposite from DTARDP.
Usage:	call DTDPAR (DPDTTM, DATEA, FSEC)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DTARDP, DTFSEDP

Arguments:	Meaning	Data type	I and/or O
DPDTTM	Date/time as a double precision date/time (represented as the number of days since 1/1/1900)	R8	I
DATEA	Integer array of six elements containing year, month, day, hour, minute and seconds	I4	O
FSEC	Fractional seconds	R4	O

Routine: DTDPST

Purpose:	Write the double precision representation of date/time to a string using a format specification
Usage:	call DTDPST (FORM, DPDTTM, STRNG)
Author(s):	Daniël van Kraalingen
Version:	1.1, TTUTIL version 4.13
Date:	6-April-1998
See also:	

Arguments:	Meaning	Data type	I and/or O
FORM	Format to use. A string containing one or more of the following specifiers with separator characters: YEAR writes the year as a four digit year MONTH writes the month as a two digit number MONTHLT writes the month with the full name MONTHST writes the month with a three character name DAY writes a two digit day number HOUR writes a two digit hour MINUTE writes a two digit minute SECONDS writes a two digit seconds FSECONDS writes the fractional seconds as a six digit number, must appear behind a seconds specifier For example, if the format ‘YEAR-MONTHST-DAY HOUR:MINUTE:SECONDS.FSECONDS’ is used on noon Sept 27, 1998, STRNG will be: ‘1998-Sep-27 12:00:00.000000’	C*(*)	I
DPDTTM	Date/time as a double precision date/time (represented as the number of days since 1/1/1900)	R8	I
STRNG	Output string with formatted date/time	C*(*)	O

Routine: DTFSECMP

Purpose:	Compares two dates in FSE format (FSE format is an integer for year and another integer for the day of year)
Usage:	<integer variable> = DTFSECMP (IYEAR1, IDOY1, IYEAR2, IDOY2)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
IYEAR1	Year of first date	I4	I

IDOY1	Day of year of first date	I4	I
IYEAR2	Year of second date	I4	I
IDOY2	Day of year of second date	I4	I
DTFSECMP	=-1, first date is earlier than second date =0, first date is equal to second date =1, first date is later than second date	I4	O

Routine: DTFSEDP

Purpose:	Converts date/time in FSE format to the double precision representation
Usage:	call DTFSEDP (IYEAR, IDOY, DOY, DPDTTM)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DTDPAR, DTARDP

Arguments:	Meaning	Data type	I and/or O
IYEAR	Year to convert	I4	I
IDOY	Integer day number to convert	I4	I
DOY	Real day number to convert	R4	I
DPDTTM	Converted date/time as double precision variable	R8	O

Routine: DTLEAP

Purpose:	Determines whether YEAR is a leap year or not, taking into account the official leap year logic
Usage:	<logical variable> = DTLEAP (YEAR)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
YEAR	Year to use	I4	I
DTLEAP	Flag whether YEAR is a leap year	L4	O

Routine: DTNOW

Purpose:	Returns the systems date and time as an integer array of six elements containing year, month, day, hour, minute and seconds
Usage:	call DTNOW (DATEA)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	DTARDP

Arguments:	Meaning	Data type	I and/or O
DATEA	Array of length 6 containing year, month, day, hour, minute and seconds	I4	O

10.12. ‘Raw’ file I/O

Routine: GETREC

Purpose:	Reads a record from an open file skipping comment lines. Comment lines have an asterisk (*) in their first or second (!!) column (with a space in the first).
Usage:	call GETREC (IUNIT, RECORD, EOF)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RECREAD

Arguments:	Meaning	Data type	I and/or O
IUNIT	Unit of opened file used for reading	I4	I
RECORD	Returned record	C*(*)	O
EOF	Flag whether the end of the file is encountered	L4	O

Routine: RECREAD

Purpose:	Returns a record from an input file opened with RECREAD_INIT. The advantage of this routine over the normal method of reading with a Fortran READ statement is that RECREAD is much faster when you are working with the Microsoft 5.1 or 1.0 compiler or the Digital Visual Fortran 5.0 compiler (due to a special mode used in the OPEN statement). Also with these compilers, RECREAD is able to determine whether the supplied string was long enough to hold every character of the input record. If not the IWAR variable is set to 1. With other compilers, RECREAD can also be used but without these advantages.
Usage:	call RECREAD (STBUF, RECLEN, STBLEN, EOF, IWAR)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	GETREC

Arguments:	Meaning	Data type	I and/or O
STBUF	String supplied by the user to which the input record is copied	C*(*)	O
RECLEN	Declared length of STBUF	I4	I
STBLEN	Significant length of string STBUF	I4	O
EOF	End_of_file flag, on EOF, this routines closes the file	L4	I
IWAR	Set to 1 if input record overflows STBUF, otherwise IWAR is zero	I4	O

Routine: RECREAD_INIT

Purpose:	Initializes sequential input from a file for subsequent reading with RECREAD. Note that input sections starting with RECREAD_INIT cannot be nested (see Section 9.1).
Usage:	call RECREAD_INIT (UNIT, INPUT_FILE)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
UNIT	Unit to use for subsequent reading with RECREAD	I4	I
INPUT_FILE	File to be used by RECREAD	C*(*)	I

Routine: RECREAD_TERM

Purpose:	Closes the open file in case it is not closed by the RECREAD routine
Usage:	call RECREAD_TERM
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
none			

10.13. List search and sorting

Routine: IFINDC

Purpose:	Finds position of a string in an array with strings ; when the string does not occur in the list a zero value is returned. Declared length of string array and string to search should be the same. Searching can take place from the beginning to end or the reverse.
Usage:	<integer variable> = IFINDC (NAMLIS, ILDEC, IST, IEND, NAME)
Author(s):	Kees Rappoldt, Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	SFINDG, IFINDI

Arguments:	Meaning	Data type	I and/or O
NAMLIS	Character string array of dimension ILDEC, the "list"	C*(*)	I
ILDEC	Declared length of array NAMLIS	I4	I
IST	Array element where search should start	I4	I
IEND	Array element where search should end	I4	I
NAME	Name to be found in the list	C*(*)	I
IFINDC	Pointer to matching array element	I4	O

Routine: IFINDI

Purpose:	Similar to IFINDC but now searching for an integer. When the integer is not in the list, a zero value is returned.
Usage:	<integer variable> = IFINDI (ILIS, ILDEC, IST, IEND, IINP)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	IFINDC, SFINDG

Arguments:	Meaning	Data type	I and/or O
ILIS	Integer array of dimension ILDEC, the "list"	I4	I
ILDEC	Declared length of array ILIS	I4	I
IST	Array element where search should start	I4	I
IEND	Actual size of the list	I4	I
IINP	Integer to be found in the list	I4	I
IFINDI	Pointer to matching array element	I4	O

Routine: SFINDG

Purpose:	This is a more powerful version of IFINDC. It finds the position of a text in an array of texts. Whereas IFINDC looks for an exact match including leading and trailing spaces, SFINDG has several ways to find a match, nl. an exact match, a match at the beginning, a match at the end, and a match anywhere in the string. Also SFINDG never takes into account trailing spaces.
Usage:	call SFINDG (NAMLIS, ILDEC, IST, IEND, NAME, ISTYPE, IFINDG, IMATCH)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	IFINDC, IFINDI

Arguments:	Meaning	Data type	I and/or O
NAMLIS	Character string array, the “list”	C*(*)	I
ILDEC	Declared length of NAMLIS	I4	I
IST	Array element where search should start	I4	I
IEND	Array element where search should end	I4	I
NAME	Character string to be found in NAMLIS	C*(*)	I
ISTYPE	Type of search to be carried out	I4	I
	=1, NAME should match NAMLIS exactly		
	=2, NAME should match beginning of NAMLIS		
	=3, NAME should match end of NAMLIS		
	=4, NAME can match at any character position		
IFINDG	Element number where match was found	I4	O

IMATCH	Character position of NAMLIS(IFINDG) where match was found	I4	O

Routine: SORTCH

Purpose:	Returns alphabetical order of an array of character strings. The order is returned as an integer index array (the character string array is left unchanged). Adapted from routine INDEXX of Numerical Recipes. The routine is left unchanged as much as possible.
Usage:	call SORTCH (N, ARRIN, INDX, Q)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	SORTIN

Arguments:	Meaning	Data type	I and/or O
N	Array size of ARRIN and INDX	I4	I
ARRIN	Character string array of dimension N	C*(*)	I
INDX	Integer array of dimension N containing the sorted order	I4	O
Q	A character variable with the same length as the elements of ARRIN. This is used internally to SORTCH as a help variable, but should be declared in the calling program.	C*(*)	I

Routine: SORTIN

Purpose:	Returns alphabetical order of an array of integers. The order is returned as an integer index array (the integer array is left unchanged). Adapted from routine INDEXX of Numerical Recipes. Routine is left unchanged as much as possible.
Usage:	call SORTIN (N, ARRIN, INDX)
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	SORTCH

Arguments:	Meaning	Data type	I and/or O
N	Array size of ARRIN and INDX	I4	I
ARRIN	Character string array of dimension N	C*(*)	I
INDX	Integer array of dimension N containing order	I4	O

10.14. Random number generation

Routine: IUNIFL

Purpose:	Generates integer uniformly distributed numbers between a lower and an upper bound (inclusive). See description of UNIFL for a more detailed discussion.
Usage:	<integer variable> = IUNIFL (LOWB, UPPB)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	UNIFL

Arguments:	Meaning	Data type	I and/or O
LOWB	Lower bound (inclusive)	I4	I
UPPB	Upper bound (inclusive)	I4	I
IUNIFL	Random integer	I4	

Routine: UNIFL

Purpose: High quality pseudo random number generator. This pseudo random generator is fully based on FUNCTION UNIFL in the second edition (1987) of Bratley et al. (see below). A logical INIT has been added to the original program in order to include seeds in the program (implicit initialization). This generator is of the so called combined type. It 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 algorithm originates from L'Ecuyer (1986). In Bratley et al. (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.

Usage:	<real variable> = UNIFL()
Author(s):	Kees Rappoldt
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	IUNIFL

Arguments:	Meaning	Data type	I and/or O
UNIFL	Pseudo-random uniformly distributed variate	R4	O

10.15. Miscellaneous

Routine: AMBUSY

Purpose:	Stores and returns message numbers belonging to routine names. Internal use is made in the TTUTIL routines to find out whether or not other routines have been called. Also the selected set number from the RDFROM routine is made available to OUTDAT.
Usage:	call AMBUSY (ITASK, ROUTINE, ICODE)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	RDFROM, OUTDAT

Arguments:	Meaning	Data type	I and/or O
ITASK	Function control =1, store routine name and code =2, get code belonging to routine name	I4	I
ROUTINE	String containing the routine name	C*(*)	I
ICODE	Code	I4	I/O

Routine: CHKTSK

Purpose:	The function of this routine is to check the new task and previous task of simulating subroutines. This routine is normally called from an FSE type of model. (See Van Kraalingen, 1995).
Usage:	call CHKTSK (ROUTINE, IULOG, ITOLD, ITASK)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
ROUTINE	Name of routine from which CHKTSK is called	C*(*)	I
IULOG	Unit for log file where to write errors to	I4	I
ITOLD	Value of previous task	I4	I
ITASK	Value of new task	I4	I

Routine: TIMER2

Purpose:	This subroutine updates TIME and related variables each time it is called with ITASK=2. It will set TERMNL to .TRUE. if FINTIM is reached. OUTPUT is flagged when TIME is a multiple of PRDEL. When PRDEL=0, no output is flagged ! When IYEAR < 1500, IYEAR will not be increased because in that case climate data are used. The routine should be initialized first by a call with ITASK=1. The first six arguments will then be made local. Leap years are handled correctly. This routine is normally called from the FSE driver, and not directly by the user. (See Van Kraalingen, 1995).
Usage:	call TIMER2 (ITASK, STTIME, DELT, PRDEL , FINTIM, IYEAR, TIME, DOY, IDOY, TERMNL, OUTPUT)
Author(s):	Daniël van Kraalingen
Version:	1.0, TTUTIL version 4.13
Date:	30-September-1997
See also:	

Arguments:	Meaning	Data type	I and/or O
ITASK	Task that the routine should carry out: =1, Initialize, store variable for processing with ITASK=2 =2, Increase TIME by DELT, update other time variables	I4	I
STTIME	Start day of simulation (1 <= STTIME <= 365, 366 in leap years, leap years are not flagged when IYEAR < 1500)	R4	I
DELT	Time step of simulation (multiple of 1 or 1/DELT = integer e.g. 0.25, 1/3, 0.5, 1, 2, 3)	R4	I
PRDEL	Time between successive outputs (must be zero, equal to DELT or multiple of DELT)	R4	I
FINTIM	Finish time of simulation (counted from start of simulation !)	R4	I
IYEAR	Start year with ITASK=1 and current year with ITASK=2, not updated when IYEAR < 1500	I4	I/O
TIME	Time from start of simulation (starts at value of STTIME)	R4	O
DOY	Day number within YEAR (REAL version)	R4	O
IDOY	Day number within YEAR (INTEGER version)	I4	O
TERMNL	Flag that indicates if FINTIM has been reached	L4	O
OUTPUT	Flag that indicates if past TIME is a multiple of PRDEL	L4	O

10.16. Internal routines

Not listed, see Section 11.3 for a list of names.

11. Reserved symbol names

11.1. General

It is advisable, to never give your own subroutines, functions, COMMON blocks, BLOCK DATA sections or main programs a name that also occurs within the TTUTIL library, unless you really understand the scope and workings of these names. Relevant names in TTUTIL in this respect are the routine names, the internal routine names, COMMON block name and names of BLOCK DATA sections. All names except the routine names (which can be found in Section 10) are listed in this Section.

11.2. Reserved common block names

COMMON block names in TTUTIL (alphabetically):

```
BUF_READ_COM1
BUF_READ_COM2
MESSINF
RDFIL1
RDFIL2
RDREC1
RDREC2
RDSTA
RDTBL1
RDTBL2
RDTOK1
RDTOK2
TTUDD1
TTUDD2
TTURR1
TTURR2
WR_SYS_1
WR_SYS_2
```

11.3. Names of internal TTUTIL routines

```
DTSYS
ENTHLP
PARSWORD
RDDATA
RDERR
RDERRI
RDINDX
RDLEX
RDSCTB
```

```
RDTMP1
RDTMP2
SWPI4
```

11.4. Names of BLOCK DATA sections

```
RECREAD_DATA
WRINIT_DATA
```

12. Capacity settings of TTUTIL read routines

The following limits exist in the TTUTIL read routines:

Quantity	Maximum
General	
Significant record length on TTUTIL format data file+1 (for end of line processing)	1024
Maximum length of variable names (in characters)	31
Maximum number of columns in a table	40
Data files	
Maximum number of variable names in a data file	400
Maximum number of values per variable name	no limit
Rerun files	
Maximum number of variable names in each set of a rerun file	40
Maximum number of values per variable name	no limit
Maximum number of sets in a rerun file	no limit

13. Removed routines

Because of little use of routines, or because of outdated functionality, we have decided to remove several routines from the current version of the TTUTIL library, relative to the one described in Rappold & van Kraalingen (1990). However, the source files are still available for these routines.

The removed routines are (alphabetically):

Routine name	Brief description of functionality	Why removed
AFINVS	Determines the inverse of a one-dimensional array with (x,y) values, a so called AFGEN/LINT table.	Very limited use.
BOXMUL	Randomly distributed numbers following Standard Normal probability distribution.	Dedicated, public domain libraries available.
CLS	Clears the screen of MS-DOS based systems (when ANSI.SYS is loaded).	Very limited use of MS-DOS, and very limited used of screen functions in MS-DOS
COPFIL	Copies a text file and appends it to another file.	Replaced by COPFIL2 with more functionality.
ERROR	Displays an error message on the screen and stops execution of the program.	Name conflicted with C-library, routine has been renamed to FATALERR.
EUDRIV	Euler integrator.	Moved to DRIVERS library.
FOPEN	Opens formatted sequential access files.	Name conflicted with C-library, routine has been renamed to FOPENS.
GAMMA	Randomly distributed numbers following Gamma distribution.	Dedicated, public domain libraries available.
GETCH	Reads a text file character by character.	Very limited use.
MOFILP	Moves the file pointer across blocks of text that start with '*'.	Routine is redundant with new RD routines for reading data from text files.
OUTARR	Outputs an array ot OUTDAT system.	Replace by OUTAR2 with more functionality.
PLTFUN	Creates a TTPLOT text file from a one-dimensional array with (x,y) values, a so called AFGEN/LINT table.	Very limited use.
PLTHIS	Creates a TTPLOT text file with a histogram of the entered data.	Very limited use.
POS	Positions the cursor of MS-DOS based systems at a particular (x,y) location of the screen. (Only when ANSI.SYS is loaded).	Very limited use of MS-DOS, and very limited used of screen functions in MS-DOS
RK4A	Part of Runge-Kutta integrator.	Moved to DRIVERS library.
RKDRIV	Part of Runge-Kutta integrator.	Moved to DRIVERS library.
RKQCA	Part of Runge-Kutta integrator.	Moved to DRIVERS library.
STRIP	Strips unwanted characters (CHR) from a character string.	Violates the FORTRAN-77 rule that string variables cannot appear on both sides of an assignment. Possibly, this function is reintroduced into the first version of the TTUTIL library for Fortran-90

		compilers.
TIMER	Clock for simulation models	Required change in time calculation, has been reprogrammed as TIMER2.

14. References

Kraalingen, D.W.G. van, 1995.
The FSE system for crop simulation, version 2.1. Quantitative Approaches in Systems Analysis No. 1. DLO Research Institute for Agrobiology and Soil fertility; The C.T.de Wit graduate school for Production Ecology. Wageningen. The Netherlands. 58 pp. (available on request).

Rappoldt, C. & D.W.G. van Kraalingen, 1990.
FORTRAN utility library TTUTIL. Simulation Report CABO-TT no. 20. Centre for Agrobiological Research and Dept. of Theoretical Production Ecology, Wageningen, The Netherlands, 54 pp. (available on request).

Rappoldt, C., and D.W.G. van Kraalingen, 1996.
The Fortran Simulation Translator, FST version 2.0. Quantitative Approaches in Systems Analysis No. 5. DLO Research Institute for Agrobiology and Soil fertility; The C.T.de Wit graduate school for Production Ecology. Wageningen. The Netherlands. 178 pp. (available on request).