Contents

Preface
Reference program
Chapter 1 Subprograms
  1.1 Function-subroutine
  1.2 Use of CSPM history functions in function-subroutines
  1.3 Subroutines
Chapter 2 Communication between mainprogram and subprograms
  2.1 List of arguments
  2.2 Blanc COMMON
  2.3 Variables that automatically occur in CSMP-III COMMON
  2.4 Labelled COMMON
  2.5 To transfer data from CSMP-COMMON to labelled COMMON
  2.6 Use of dummy's in labeled COMMON
Chapter 3 FORTRAN statements in CSMP main program
  3.1 Use of REAL and INTEGER labels in main program
Chapter 4
  4.1 To force output at prdel
  4.2 Use of more than 25 fixed variables
References
Appendix: Modifications necessary to execute examples on IBM PC-AT machines

Example contents

<table>
<thead>
<tr>
<th>no</th>
<th>title</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>Reference CSMP program</td>
<td>1</td>
</tr>
<tr>
<td>1a</td>
<td>Function-subroutine with LINFUN</td>
<td>4</td>
</tr>
<tr>
<td>1b</td>
<td>Function-subroutine with FAFGEN</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Function-subroutine with history function</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Subroutine cylinder geometry</td>
<td>11</td>
</tr>
<tr>
<td>4a</td>
<td>Subroutine CSMP way of calling a subroutine</td>
<td>13</td>
</tr>
<tr>
<td>4b</td>
<td>Subroutine FORTRAN way of calling a subroutine in a PROCEDURE</td>
<td>13</td>
</tr>
<tr>
<td>4c</td>
<td>Subroutine FORTRAN way of calling a subroutine in a NOSORT section</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Application of blanc COMMON</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Application of labelled COMMON, cylinder geometry</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>as in example 3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Application of labelled COMMON, conversion of array integral values</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>in double array values for calculations similar in structure and the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reverse</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>To force output at PRDEL when no CSMP PRINT or OUTPUT is used and</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>writing simulation timings to the TTY screen during simulation</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Use of more than 25 variables on FIXED label</td>
<td>28</td>
</tr>
</tbody>
</table>
Preface

The members of the department of Theoretical Production Ecology, in their August 1985 meeting, have decided to develop a (sub-) program library for common use. Subprograms to be included should describe processes and algorithms which are frequently needed by a number of people, who now all develop their own program versions for these processes.

To develop, read and update subprograms, however, one has to have a thorough working knowledge of CSMP-III, FORTRAN and particularly of the combined use of these computer languages. This (first) report aims to summarize some knowledge on the combined use of these computer languages by giving examples of applications in the form of small computer programs.

It is hoped that the examples will give inspiration to develop more structured programs possibly containing less errors, both with respect to conceptual errors and programming errors. The main consequence of such well written programs should be that less time is wasted to understand the modeling concepts of fellow workers.

Though it was tried to develop general CSMP-FORTRAN examples, sometimes specific VAX commands are needed to execute the programs.

In the appendix of this report, a description is given of the modifications necessary to execute the examples on IBM PC-AT machines.

I welcome comments and worked out suggestions from readers.

P.A. Leffelaar
March 1986.
Reference program

The following CSMP-simulation program calculates exponential growth with a temperature dependent relative growth rate. Some examples to clarify the use of subroutines and function-subroutines are derived from this program. (See Simulation of ecological processes by C.T. de Wit and J. Goudriaan, 1978, page 15).

TITLE Reference CSMP program (RELGROW.CSM).
TITLE Expon. growth with temp. dependent rel. growth rate
INITIAL
PARAM AVTMP = 20., AMPTMP = 10.
FUNCTION RGRTB = (0.0,0.00),(10.,0.08),(20.,0.16),...
(30.,0.21),(40.,0.24),(50.,0.25)
TIMER PINTIM = 48., OUTDEL = 1.
METHOD RKS
OUTPUT A, RGR, GR
PI = 4.*ATAN(1.)
DYNAMIC
A = INTGRAL(1.,GR)
GR = RGR*A
RGR = AFGEN(RGRTB,TEMP)
TEMP = AVTMP + AMPTMP*SIN(2.*PI*TIME/24.)
END
STOP
ENDJOB
CHAPTER 1

Subprograms

By using Subprograms it is possible to make larger programs than CSMP allows. Subprograms are written according to the standard FORTRAN rules.

1.1 Function-subroutine

The Function-subroutine is used to develop functional-blocks, which have a single output-variable.

- Output takes place through NAME of function-subroutine or possibly through (labelled) COMMON (see par. 2.2-2.4)
- Type-declaration of NAME in calling subprogram and in definition of function-subroutine necessary.
- At least one argument between brackets should follow NAME of function-subroutine or only brackets, e.g. ()
- In CSMP the function-subroutine's NAME is automatically declared real, also when starting with an I, J, K, L, M, or N.
- CSMP-statements excluded from use in function-subroutines are: INTGRL, IMPL, MODINT, REALPL, CMPXPL, LEDLAG, PIPE, TRANSF, and any user defined macro.
- Lay out of a program with function-subroutine:

```
TITLE ...........

A = NAME(IN1, IN2) + NAME(IN3, IN4)

END
STOP

REAL FUNCTION NAME(INA, INB)
IMPLICIT REAL (A-Z)
DIMENSION ....

RESULT = ..............
NAME = RESULT
RETURN
END

ENDJOB
```

Note: 1. NAME = RESULT is recommended to be the last statement in function-definition for clarity.
2. Variable dimensioning is only possible for variables mentioned in the list of arguments.

- type declaration of NAME on a "REAL"-statement must not be given in CSMP program (see par. 3, 3.1).
- IMPLICIT REAL (A-Z) is used to have similar variable type declaration as in CSMP (namely all variables automatically real, and integer exceptions on FIXED-label). Exceptions in subprogram declared on INTEGER-label.
- All subprograms are placed between labels STOP and ENDJOB.
- ENDJOB should always be typed in first six columns.
- The CSMP translator places the function-subroutine-statement in UPDATE without conversion or expansion, except when a history-function is used (see par. 1.2).

EXAMPLES:

program with function-subroutine (FUNC.CSM) Example 1a.

Example 1a and also the following example (1b) contains a function-subroutine which interpolates linearly between two points, Figure 1.
TITLE Example 1a function-subroutine (FUNC.CSM)
TITLE Exponential growth with temp. dependent rel. growth rate
INITIAL
FIXED N
STORAGE RGRT(10), TMPT(10)
PARAM AVTMP = 20., AMPTMP = 10., N = 6
TABLE TMPT(1-6) = 0.,10.,20.,30.,40.,50.
TABLE RGRT(1-6) = 0.,0.08,0.16,0.21,0.24,0.25
TIMER FINTIM = 48., OUTDEL = 1.
METHOD RKS
OUTPUT A,RGR,GR
PI = 4.*ATAN(1.)
DYNAMIC
A = INTGRL(1.,GR)
GR = RGR*A
RGR = LINFUN(RGRT,TMPT,TEMP,N)
TEMP = AVTMP + AMPTMP*SIN(2.*PI*TIME/24.)
END
STOP

REAL FUNCTION LINFUN (RGRT,TMPT,TEMP,N)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION RGRT(N), TMPT(N)
IF (TEMP.GE.TMPT(1).AND.TEMP.LE.TMPT(N)) GO TO 20
TYPE 10
FORMAT (' Temp out of range')
CALL EXIT
RETURN
20 IF(TEMP.GT.TMPT(1)) GO TO 30
LINFUN = RGRT(1)
RETURN
30 DO 40 I=2,N
IF (TEMP.GT.TMPT(I)) GO TO 40
HULP = RGRT(I-1) + (TEMP-TMPT(I-1))*(RGRT(I)-RGRT(I-1))/$
(TMPT(I)-TMPT(I-1))
40 CONTINUE
RETURN
END
note: 1 In example FUNC.CSM variable dimensioning is used. Dimension of array in subprogram must always be smaller or equal to size of corresponding array in main program. When (n x m)-matrices are used, the first dimension (n) must equal the corresponding number in the calling program, whereas the second dimension (m) must be smaller or equal to the number in the calling program. It is recommended to use fixed dimensioning when using matrices.

note: 2 CALL EXIT subroutine causes program termination, closes all files, and returns control to the operating system.

note: 3 In example 1a an error-message is printed directly on screen, if TEMP is out of range of table TMPT. In FORTRAN this is also accomplished with WRITE(6,10), since number 6 is standard assigned to terminal-screen. In CSMP, however, WRITE(6,10) refers to file FOR06.DAT, therefore TYPE 10 is used in example 1a to send a message directly to screen. The same result is accomplished with WRITE(20,10) in example 1b. Normally, this statement would create a FOR020.DAT file, but if ASSIGN SYS$OUTPUT FOR020 is typed before running this program, the message is send directly to screen.

note: 4 Example 1a and also the following example (1b) contains a function-subroutine which interpolates linearly between two points, Figure 1.

In the following example TABLE RGR TMP(I-12) contains both coordinates for relative growth rate and temperature. The sequence is RGR TMP(I) = rel. growth rate, RGR TMP(2) = temp., RGR TMP(3) = rel. growth rate, etc.

Figure 1 Coordinates used in examples 1a and 1b.

EXAMPLE:

program with function-subroutine (FUNSUB.CSM). Example 1b.
TITLE Example 1b function-subroutine (FUNSUB.CSM)
TITLE Exponential growth with temp. dependent rel. growth rate
INITIAL
FIXED N
STORAGE RGRTMP(12)
PARAM AVTMP = 20., AMPTMP = 10., N = 12
TABLE RGRTMP(1-12)= 0.0,0.0, 10.,0.08, 20.,0.16, 30.,0.21, ...
40.,0.24, 50.,0.25
TIMER FINITIM = 48., OUTDEL = 1.
METHOD RKS
OUTPUT A,RGR,GR
PI = 4.*ATAN(1.)
DYNAMIC
A = INTGRL(1.,GR)
GR = RGR*A
RGR = FAFGEN(RGRTMP,TEMP,N)
TEMP= AVTMP + AMPTMP*SIN(2.*PI*TIME/24.)
END
STOP

REAL FUNCTION FAFGEN(TABLE,X,N)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION TABLE(N)
IF (X .GE. TABLE(1) .AND. X .LE. TABLE(N-1)) GO TO 20
WRITE (20,10)
10 FORMAT(' X out of range')
CALL EXIT
RETURN
20 DO 30 I=1,N-1,2
IF (TABLE(I) .GE. X) THEN
  IF (TABLE(I) .EQ. X) THEN
    Y =TABLE(I+1)
  ELSE
    SLOPE=(TABLE(I+1)-TABLE(I-1)) / (TABLE(I)-TABLE(I-2))
    BETA =TABLE(I+1)-SLOPE * TABLE(I)
    Y =SLOPE * X+BETA
  ENDIF
ENDIF
FAFGEN =Y
RETURN
30 CONTINUE
RETURN
END
ENDJOB
note: - FUNCTION RGRTB and TABLE RGRTMP(1-12) (reference
CSMP-program, and example 1b) both contain the
tabulated function of RGR depending on TEMP.
- FUNCTION is used in combination with
function-generators, like AFGEN.
- In combination with TABLE, STORAGE has to be used to
reserve memory space.
- The TABLE-statement has the advantage that all
individual measuring-points are available for other
calculations (for instance slope).
- Examples 1a and 1b will usually not be needed by the
CSMP user, because AFGEN is available. Examples are
given for illustrative purposes, however, because
the process of linear interpolation is simple and
attention can be focussed on programming aspects.

1.2 Use of CSMP history-functions in function-subroutine

- Some history-functions and their number of memory-locations
are (see for more information CSMP III Program Reference
Manual (IBM), 1975, page 99)

<table>
<thead>
<tr>
<th>function</th>
<th>memory locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFGEN</td>
<td>5</td>
</tr>
<tr>
<td>FUNGEN</td>
<td>10</td>
</tr>
<tr>
<td>NLFGEN</td>
<td>10</td>
</tr>
<tr>
<td>SAMPLE</td>
<td>3</td>
</tr>
<tr>
<td>TWOVAR</td>
<td>12</td>
</tr>
</tbody>
</table>

- Lay-out of program with function-subroutine containing two
CSMP history-functions

```
TITLE ..........
HISTORY NAME(10)
| |
A = NAME(IN1,IN2,IN3,IN4)
| END
STOP

REAL FUNCTION NAME(NLOC,IN1,IN2,IN3,IN4)
IMPLICIT REAL (A-Z)
INTEGER NLOC, ......
| |
A = AFGEN(NLOC,IN1,IN2)
| |
B = AFGEN(NLOC+5,IN3,IN4)
| |
RESULT = A+B
NAME = RESULT
RETURN
END

ENDJOB
```
- Place HISTORY-label at beginning of main program in INITIAL-segment (if used).
- Name on HISTORY-label is name of function-subroutine in which history-function appears.
- Number between brackets specifies number of storage-locations needed for CSMP-function(s) in function-subroutine.
- Integer variable NLOC (Number of LOCations) takes value assigned by CSMP-compiler.

In the UPDATE.FOR line \( RGR = \text{NAME}(RGRTB, TEMP) \) in following example reads: \( RGR = \text{NAME}(1, RGRTB, TEMP) \). Note that number of arguments in call of function-subroutine in CSMP main program is one argument shorter than in definition. The conversion of the statement by the CSMP-compiler with additional information on HISTORY-label gives a similar number of arguments.

**EXAMPLE:**

**Function-subroutine with nested history-function** (PUNCHIS.CSM) example 2

**TITLE** Example 2 function-subroutine with history function (PUNCHIS.CSM)

**TITLE** Exponential growth with temp. dependent rel. growth rate

**INITIAL**

```
HISTORY NAME(5)
``` 

**PARAM**

```
AVTMP=20., AMPTMP=10.
``` 

**FUNCTION**

```
RGRTB=(0.0,0.00),(10.,0.08),(20.,0.16),(30.,0.21), ...
    (40.,0.24),(50.,0.25)
``` 

**TIMER**

```
FINTIM=48., OUTDEL=1.
``` 

**METHOD**

```
RKS
``` 

**OUTPUT**

```
A, RGR, GR
``` 

**PI**

```
= 4.*ATAN(1.)
``` 

**DYNAMIC**

```
A = INTGRL(1., GR)
GR = RGR*A
RGR = NAME(RGRTB, TEMP)
TEMP= AVTMP + AMPTMP*SIN(2.*PI*TIME/24.)
``` 

**END**

**STOP**

**REAL FUNCTION NAME(NLOC,RGRTB,TEMP)**

```
IMPLICIT REAL (A-Z)
INTEGER NLOC
RESULT = NESTED(NLOC,RGRTB,TEMP)
NAME = RESULT
RETURN
END
``` 

**REAL FUNCTION NESTED(NLOC,RGRTB,TEMP)**

```
IMPLICIT REAL (A-Z)
INTEGER NLOC
RESULT = AFGEN(NLOC,RGRTB,TEMP)
NESTED = RESULT
RETURN
END
``` 

**ENDJOB**
1.3 Subroutines

If multiple output is desired the SUBROUTINE form of subprogram must be used.

- Output via list of arguments or possibly through (labelled) COMMON.
- CSMP-statements excluded from use in subroutines are:
  
  INTGRL, IMPL, MODINT, REALPL, CMFXPL, LEDLAG, PIPE, TRANSF, and any user defined macro.

Different call's to a subroutine from a CSMP-program:

- CSMP-call for a subroutine, sortable by the CSMP-compiler is: \( A, B = \text{NAME}(C,D) \)
- The CSMP-compiler converts this call to following statement in UPDATE.FOR: \( \text{CALL NAME}(C,D,A,B) \).
  Output variables \((A,B)\) are thus placed after input variables \((C,D)\) in their original sequence.
- The CSMP-call for a subroutine is only correctly interpreted by the CSMP-compiler if there are two or more variables at the left-hand side of the equal-sign. If a subroutine-call is defined as \( A = \text{NAME}(C,D) \), the compiler interprets this as a call for a function-subroutine. Generally, if there is only one output-variable, use the function-subroutine.
- Otherwise there are two possibilities which lead to correct interpretation of a subroutine if there is just one output-variable:

  * by introducing a dummy-variable:

    \( A, \text{DUMVAR} = \text{NAME}(C,D) \)

  * or by using the FORTRAN-call:

    \( \text{CALL NAME}(C,D,A) \)

- The FORTRAN call-statement must be placed either in a NOSORT-section,

```fortran
NOSORT
  |  CALL NAME(C,D,A)
  |
SORT
```

or in a PROCEDURAL-block

```fortran
PROCEDURE A=PRONAM(C,D)
  |  CALL NAME(C,D,A)
  |
ENDPROCEDURE
```
- Lay-out of program with subroutine having two output-variables

TITLE .............
INITIAL

DYNAMIC
PROCEDURE A,B=PRONAM(C,D)
A,B = NAME(C,D) or < CALL NAME(C,D,A,B)
ENDPROCEDURE

SUBROUTINE NAME(C,D,A,B)
IMPLICIT REAL (A-Z)
INTEGER
RETURN
END
ENDJOB

note: all subroutines and function-subroutines placed after STOP-label are not processed by the CSMP-compiler. Therefore the list of arguments must agree with interpretation of CSMP call-statement in mainprogram (see above).

EXAMPLES:

program with subroutine (GEOM.CSM). Example 3.

programs with different subroutine calls (SUBDUM.CSM, SUBPRO.CSM, and SUBSOR.CSM). Examples 4a-4c.
**Example 3 subroutine** (GEOM.CSM)

**Cylinder geometry**

`TITLE Cylinder geometry`

`INITIAL`

`NO SORT`

`FIXED N`

```plaintext
/ REAL  DIST, DEPTH, RAD, AREA, VOL
/ DIMENSION DIST(20), DEPTH(20), RAD(21)
/ DIMENSION AREA(20), VOL(20)

STORAGE TCOM(20)
```

`PARAM DIAM = 10.E-2, HEIGHT = 2.5E-2, N = 10`

`TABLE TCOM(1-10) = 10*5E-2`

`TIMER FINTIM = 5., PRDEL = 1.`

`PRINT A`

`METHOD RKS`

`PI = 4.*ATAN(1.)`

```
***** Calculation cylinder geometry *****
```

```plaintext
CALL GEOMET(N,DIAM,HEIGHT,PI,TCOM,DIST,DEPTH,RAD,AREA,VOL)
```

`DYNAMIC A = INTGRL(1.,0.1*A)`

`TERMINAL`

```
***** Print uitvoer *****
```

```plaintext
CALL WRIT(N,DIST,DEPTH,RAD,AREA,VOL)
```

END

STOP

---

**SUBROUTINE GEOMET(N,DIAM,HEIGHT,PI,TCOM,DIST, DEPTH,RAD,AREA,VOL)**

```plaintext
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION DIST(N), DEPTH(N), RAD(N+1)
DIMENSION AREA(N), VOL(N), TCOM(N)
```

```
C*** Cylinder geometry
```

```plaintext
DIST(1) = TCOM(1)/2.
DEPTH(1) = TCOM(1)/2.
RAD(1) = DIAM/2.
```

```
DO 10 I=2, N
   DIST(I) = (TCOM(I-1)+TCOM(I))/2.
   DEPTH(I) = DEPTH(I-1)+DIST(I)
```

```plaintext
10 CONTINUE
```

```
DO 20 I=1, N
   RAD(I+1) = RAD(I)-TCOM(I)
   AREA(I) = 2.*PI*RAD(I)*HEIGHT
   VOL(I) = PI*(RAD(I)**2-RAD(I+1)**2)*HEIGHT
```

```plaintext
20 CONTINUE
```

RETURN

END

---

**SUBROUTINE WRIT(N,DIST,DEPTH,RAD,AREA,VOL)**

```plaintext
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION DIST(N), DEPTH(N), RAD(N+1)
DIMENSION AREA(N), VOL(N)
```

```
WRITE(20,10)
```

```plaintext
```

```
DO 30 I=1, N
   WRITE(20,20) DIST(I), DEPTH(I), RAD(I), AREA(I), VOL(I)
```

```plaintext
20 FORMAT (E12.4,3X,E12.4,3X,E12.4,3X,E12.4,3X,E12.4)
```

```plaintext
30 CONTINUE
```

RETURN

END

---

ENDJOB
note: - Cylinder geometry, see figure 2.
- Variables not in CSMP-COMMON are: DIST, DEPTH, RAD, AREA, VOL. Thus not available for "PRINT" or "OUTPUT".
- Therefore, WRITE-routine in FORTRAN necessary.
- Before running this program type

ASSIGN SYS$OUTPUT FOR020

Figure 2. Cylinder geometry in examples 3 and 6.
Example 4a subroutine with dummy in call (SUBDUM.CSM).

Title: Example 4a subroutine with dummy in call (SUBDUM.CSM).

Title: Exponential growth with temp. dependent rel. growth rate

Initial history

Name(S)


Function: RGRTB = (0.0, 0.05), (10., 0.08), (20., 0.16), (30., 0.21), ...

(40., 0.26), (50., 0.25)

Timer: FINTIM = 48., OUTDEL = 1.

Method: RKS

Output: A, RGR, GR

Pi: = 4. * atan(1.)

Dynamic

A: = INTGR(1., OR)

GR: = RGR * A

RGR, DUMMY: NAME(RGRTB, TEMP)

Temp: = AVTMP + AMPTMP * SIN(2. * PI * TIME/24.)

End stop

SUBROUTINE NAME(NLOC, RGRTB, TEMP, RGR, DUMMY)

IMPLICIT REAL (A-Z)

INTEGER NLOC

RGR = AFGR(NLOC, RGRTB, TEMP)

RETURN

END

End job

FORTRAN-call in PROCEDURAL-block (SUBPRO.CSM).

Example 4b.

Title: Example 4b subroutine with a PROCEDURE-block (SUBPRO.CSM).

Title: Exponential growth with temp. dependent rel. growth rate

Initial history

Name(S)


Function: RGRTB = (0.0, 0.05), (10., 0.08), (20., 0.16), (30., 0.21), ...

(40., 0.26), (50., 0.25)

Timer: FINTIM = 48., OUTDEL = 1.

Method: RKS

Output: A, RGR, GR

Pi: = 4. * atan(1.)

Dynamic

A: = INTGR(1., OR)

GR: = RGR * A

PROCEDURE RGR: BLOCK(TEMP, RGRTB)

CALL NAME(RGRTB, TEMP, RGR)

ENDPROCEDURE

Temp: = AVTMP + AMPTMP * SIN(2. * PI * TIME/24.)

End stop

SUBROUTINE NAME(NLOC, RGRTB, TEMP, RGR)

IMPLICIT REAL (A-Z)

INTEGER NLOC

RGR = AFGR(NLOC, RGRTB, TEMP)

RETURN

END

End job

FORTRAN-call in NOSORT-section (SUBSOR.CSM).

Example 4c.

Title: Example 4c subroutine with use of a SORT-section (SUBSOR.CSM).

Title: Exponential growth with temp. dependent rel. growth rate

Initial history

Name(S)


Function: RGRTB = (0.0, 0.05), (10., 0.08), (20., 0.16), (30., 0.21), ...

(40., 0.26), (50., 0.25)

Timer: FINTIM = 48., OUTDEL = 1.

Method: RKS

Output: A, RGR, GR

Pi: = 4. * atan(1.)

Dynamic

A: = INTGR(1., OR)

Temp: = AVTMP + AMPTMP * SIN(2. * PI * TIME/24.)

CALL NAME(RGRTB, TEMP, RGR)

GR: = RGR * A

End stop

SUBROUTINE NAME(NLOC, RGRTB, TEMP, RGR)

IMPLICIT REAL (A-Z)

INTEGER NLOC

RGR = AFGR(NLOC, RGRTB, TEMP)

RETURN

END

End job

END 13
CHAPTER 2

Communication between main program and subprograms.

Communication between main program and subprogram or between two subprograms is established by:

- Lists of arguments
- Blank COMMON
- Labelled COMMON

or a combination of these possibilities.

2.1 List of arguments

- List of arguments or variables is given in call of a function-subroutine or subroutine and corresponds to the definition between labels STOP and ENDJOB. For example:

  - Function-subroutine:
    \[ A = \text{NAME}(C,D) \]
    with
    \[
    \text{REAL FUNCTION NAME}(C,D) \\
    \text{list of arguments: } (C,D )
    \]

  - Subroutine:
    \[ A,B = \text{NAME}(C,D) \]
    or
    \[ \text{CALL NAME}(C,D,A,B) \]
    with
    \[
    \text{SUBROUTINE NAME}(C,D,A,B) \\
    \text{list of arguments: } (C,D,A,B)
    \]

- Continuation of list of arguments in CSMP-program (before labels END, STOP) by three dots ('...'') at the end of line to be continued.
- Continuation of list of arguments in subroutine definition (after label STOP) by placing a '$'-sign in sixth column of the line following the line which is to be continued.
- Continuation of lines with lists of arguments is allowed
upto 8 lines in CSMP-FORTRAN combination.
Sequence of variables mentioned in call- and subroutine-statements must correspond.

NOTE: Communication with lists of arguments is recommended for novice.
See examples 1a, 1b, 2, 3, 4a, 4b, and 4c.

2.2 Blanc Common

Including blanc COMMON in a subprogram means that all variables in previously established CSMP III COMMON are available in that subprogram.

- Lay-out of subroutine with use of blanc COMMON:

```
SUBROUTINE NAME(......
REAL ......
INTEGER ......
COMMON
    | |
RETURN
END
```

note: If blanc COMMON is included the IMPLICIT REAL label can not be used, because all variables which are also used in the mainprogram are already declared real or integer in CSMP-COMMON (e.g. N in example 5). New variables in the subroutine (e.g. I in example 5) must be declared on the appropriate labels.

Use of blanc COMMON is not recommended for novice because the programmer has to have considerable knowledge of how CSMP III is processed.

EXAMPLE:

(For more information on this example see LH Theoretische Teelkunde, Inleidende teksten en practicumopgaven, 1986, page 88 and 89)

Application of blanc COMMON (BLACOM.CSM). Example 5.
**Example 5 application of blanc COMMON (BLACOM.CSM)**

**Flow of heat in a homogenous soil column.**

**INITIAL**

**FIXED**

N

**STORAGE**

FLOW(26), TEMP(25)

**PARAM**

TCOM=0.02, COND=0.42, VHCAP=1.0586, ITMP=20., TAV=20., TAMPL=10.

**PARAM**

N = 25

**TIMER**

FINITIM=345600., OUTDEL=3600., PRDEL=3600.

**METHOD**

RKS

**OUTPUT**

TEMP(1), TEMP(5), TEMP(15), TEMP(25)

**PAGE**

GROUP, NTAB=0, WIDTH=80

**PRINT**

TEMP(1), TEMP(5), TEMP(15), TEMP(25)

**PI**

= 4.*ATAN(1.)

**CALL**

INITMP

**DYNAMIC**

**NO SORT**

**VHTC** = INTG(R(IVHTC), NFLOW, 25)

**CALL**

TMPPR(TMPS)

**CALL**

TMFFLW(TMPS)

**END**

**STOP**

**C*******************************************************************************/

**SUBROUTINE INITMP**

**INTEGER**

I

**COMMON**

DO 10 I=1, N

IVHTC(I)=ITMP*TCOM*VHCAP

10 CONTINUE

RETURN

END

**C*******************************************************************************/

**SUBROUTINE TMPPR(TMPS)**

**INTEGER**

I

**COMMON**

TMPS = TAV+TAMPL*SIN(2.*PI*TIME/86400.)

DO 10 I=1, N

TEMP(I)=VHTC(I)/(TCOM*VHCAP)

10 CONTINUE

RETURN

END

**C*******************************************************************************/

**SUBROUTINE TMFFLW(TMPS)**

**INTEGER**

I

**COMMON**

FLOW(1) = (TMPS-TEMP(1))*COND/(0.5*TCOM)

DO 10 I=2, N

FLOW(I) = (TEMP(I-1)-TEMP(I))*COND/TCOM

10 CONTINUE

FLOW(26)=0.0

RETURN

END

**C*******************************************************************************/

**SUBROUTINE NETFFLW**

**INTEGER**

I

**COMMON**

DO 10 I=1, N

NFLOW(I)=FLOW(I)-FLOW(I+1)

10 CONTINUE

RETURN

END
note: In this example no CSMP-call (see par. 1.3) can be used, because blanc COMMON is used and thus no arguments are listed by which CSMP could sort and interpret the statement. This means that CSMP can not sort the routines, so the user has to enter all routines in the correct sequence in a NOSORT-section.

2.3 Variables that automatically occur in CSMP III COMMON

- all variables at left-hand side of equal-sign.
- all variables on a TABLE-statement.
- all variables on a FUNCTION-statement.
- all variables on a STORAGE-statement.
- all variables on PARAM, INCON-, and CONSTANT-statements.
- all variables on a INTGR1-statement.
- all variables on a TIMER-label.

2.4 Labelled COMMON

Labelled COMMON allows communication between subprograms or mainprogram and subprogram by means of COMMON memory-space.

- Lay-out of subroutine with labelled COMMON:

```
SUBROUTINE NAME(..............
IMPLICIT REAL (A-Z)
INTEGER ..... COMMON /COMNAM/ A(10), B(50,4), C, ....
RETURN
END

SUBROUTINE COMTO(..............
IMPLICIT REAL (A-Z)
INTEGER ..... COMMON /COMNAM/ A(10), B(50,4), C, ....
RETURN
END
```

note: name of COMMON block is placed between slashes (/). It is not necessary to add the labelled COMMON-block in program parts where the variables are not needed. In contradiction to labelled COMMON, blanc COMMON has no list of variables.

- Use always the same variable-names for corresponding variables in COMMON-blocks.
2.5 To transfer data from CSMP-COMMON to labelled COMMON

- Application 1:
  
  * Copy only data you need out of CSMP COMMON into subprogram so you don't have to transfer the whole CSMP COMMON into your subprogram.

- General lay out of copy part of program:

```
TITLE ........
!
PARAM A=... , B=... , C=...
PARAM D=... , E=...
NOSORT
!
CALL TRANSF(A,B,C)
!
END
STOP

SUBROUTINE TRANSF(A,B,C)
!
COMMON /COPY/ AA, BB, CC
AA=A
BB=B
CC=C
!
RETURN
END

ENDJOB
EXAMPLE:

Geometry of a cylinder (FILLCO.CSM). Example 6
TITLE Example 6 application of labelled COMMON (FILLCO.CSM)
TITLE Cylinder geometry
INITIAL
NOSORT
FIXED N
STORAGE TCOM(20)
PARAM DIAM = 10. E-2, HEIGHT = 2.5E-2, N = 10
TABLE TCOM(1:10) = 10*.5E-2
TIMER FINTIM = 5., PRDEL = 1., DELT = .01
PRINT A
METHOD RKS
PI = 4.*ATAN(1.)

***** Copy data from CSMP-COMMON to labelled COMMON *****
CALL FILLCO(N,TCOM)

***** Calculation cylinder geometry *****
CALL GEOMET(N,DIAM,HEIGHT,PI)

DYNAMIC A = INTGRL(1.,0.1*A)

TERMINAL

***** Print uitoever *****
CALL WRIT(N)

END
STOP

SUBROUTINE FILLCO(N,TCOM)
IMPLICIT REAL (A-Z)
INTEGER I,N
DIMENSION TCOM(20)
COMMON /GEOM/ TCOM(20)
DO 10 I=1, N
TCOM(I) = TCOM(I)
10 CONTINUE
RETURN
END

SUBROUTINE GEOMET(N,DIAM,HEIGHT,PI)
IMPLICIT REAL (A-Z)
INTEGER I,N
COMMON /GEOM/ TCOM(20)
COMMON /OUTVAL/ DIST(20), DEPTH(20), RAD(21),
$ AREA(20), VOL(20)

C*** Cylindrical geometry
DIST(1) = TCOM(1)/2.
DEPTH(1) = TCOM(1)/2.
RAD(1) = DIAM/2.
DO 10 I=2, N
DIST(I) = (TCOM(I-1)+TCOM(I))/2.
DEPTH(I) = DEPTH(I-1)+DIST(I)
10 CONTINUE
DO 20 I=1, N
RAD(I+1) = RAD(I)-TCOM(I)
AREA(I) = 2.*PI*RAD(I)*HEIGHT
VOL(I) = PI*(RAD(I)**2-RAD(I+1)**2)*HEIGHT
20 CONTINUE
RETURN
END

SUBROUTINE WRIT(N)
IMPLICIT REAL (A-Z)
INTEGER I,N
COMMON /OUTVAL/ DIST(20), DEPTH(20), RAD(21),
$ AREA(20), VOL(20)
WRITE(20,10)
10 FORMAT (6X, 'DIST',10X,'DEPTH',10X,'RAD',
$ 11X,'AREA',11X,'VOL')
DO 30 I=1, N
WRITE(20,20) DIST(I), DEPTH(I), RAD(I),
$ AREA(I), VOL(I)
20 FORMAT (B12.4,3X,B12.4,3X,B12.4,3X,B12.4,3X,B12.4)
30 CONTINUE
RETURN
END

ENDJOB
note: - Observe that no variable dimensioning can be used in COMMON-blocks.
- Program gives same result as example 3.
- By the action of putting only those variables in a labelled COMMON which are needed from the blank COMMON, it is avoided to include CSMP III COMMON as a whole in a subroutine. Especially for the novice this is good practice.

- Application 2:

* Create possibility to integrate a two dimensional array (see example 7 CONVRT.CSM) by transferring it into a one-dimensional array.

EXAMPLE:

(CONVRT.CSM). Example 7.

A lake is connected with a second and third lake of equal size. River water which is polluted by phosphate and nitrate flows into the first lake. The lakes are well mixed and of constant volume. Thus the polluted water from the first lake enters the second lake and so on. Calculations of both pollutions in each lake are similar in structure, but not of course in magnitude.

List of variables:

AANVSN - amount of water flowing in and out first, second and third lake. \( m^3 d \)

VOLHN - Equal and constant volume of water in the three lakes. \( m^3 \)

CONC1 - concentration of phosphate in riverwater. \( kg m^{-3} \)

CONC2 - concentration of nitrate in riverwater. \( kg m^{-3} \)

HN1(I) - amount of phosphate in lakes. \( kg \)

HN2(I) - amount of nitrate in lakes. \( kg \)

This example is calculating the course of the amounts of the two pollutions in each lake. The relational diagram of the problem is given in figure 3.
Figure 3. Relation diagram of three lakes in series. The first lake is polluted by phosphate and nitrate supplied by riverwater. The water flowing out of lake 1 comes in lake 2 and so on in lake 3.
TITLE Example 7 Application of labelled COMMON (CONVRT.CSM)
TITLE Concentration of pollution in three lakes
INITIAL
FIXED NI,NL
TABLE IHN1(1-3) =3*0.
TABLE IHN2(1-3) =3*0.
PARAM AANVSN =5.67 ,VOLHN =90.E7 ,NI =2
PARAM CONC1 =0.5 ,CONC2 =0.5 ,NL =3
TIME FINTH =180. ,PRDEL =10. ,DELT =2.
PRINT RHN1(1-3), RHN2(1-3), HN1(1-3), HN2(1-3)
METHODE RKSF
DYNAMIC
NOSORT
HNL = INTGRL(IHN1,RHN1,3)
HN2 = INTGRL(IHN2,RHN2,3)
*** Conversion of one dimensional HN1, HN2
*** to two dimensional H(L,1), H(L,2)
CALL CNVRT1(NL,HNL,HN2)
*** Calculation of rates
CALL CALC(NL,NI,AANVSN,CONC1,CONC2,VOLHN)
*** Conversion of two dimensional RH(L,1), RH(L,2)
*** to one dimensional RHN1(L), RHN2(L)
CALL CNVRT2(NL,RHN1,RHN2)
END
STOP

SUBROUTINE CNVRT1(NL,HNL,HN2)
IMPLICIT REAL (A-Z)
INTEGER L,NL
COMMON /CONVRT/ H(3,2)
DIMENSION HN1(3),HN2(3)
DO 10 L=1,NL
H(L,1) = HNL(L)
H(L,2) = HN2(L)
10 CONTINUE
RETURN
END

SUBROUTINE CALC(NL,NI,AANVSN,CONC1,CONC2,VOLHN)
IMPLICIT REAL (A-Z)
INTEGER I,L,NI,NL
COMMON /TRANSF/ RH(3,2)
COMMON /CONVRT/ H(3,2)
DIMENSION FLOWH(4,2)
TC = VOLHN/AANVSN
FLOWH(1,1) = AANVSN*CONC1
FLOWH(1,2) = AANVSN*CONC2
DO 10 L = 1,NL
DO 10 I = 1,NI
FLOWH(L+1,I) = H(L,I)/TC
10 CONTINUE
DO 20 L = 1,NL
DO 20 I = 1,NI
RH(L,I) = FLOWH(L,I)-FLOWH(L+1,I)
20 CONTINUE
RETURN
END

SUBROUTINE CNVRT2(NL,RHN1,RHN2)
IMPLICIT REAL (A-Z)
INTEGER L,NL
COMMON /TRANSF/ RH(3,2)
DIMENSION RHN1(3), RHN2(3)
DO 10 L=1,NL
RHN1(L) = RH(L,1)
RHN2(L) = RH(L,2)
10 CONTINUE
RETURN
END

ENDJOB
2.6 Use of dummy's in labelled COMMON

Dummy’s are used when one or more arrays defined in a labelled COMMON are not of interest in a (sub)program where the labelled COMMON appears.

- Use in program:

If a routine includes the following line:

```
COMMON /GEOMET/ A(5,10), B(500), C(50)
```

and in another routine only array C is of importance the following line could be included there.

```
COMMON /GEOMET/ DGE01(550), C(50)
```

note: DGE01 is a dummy array. It takes the contents of arrays A and B (5*10+500=550). D in DGEOM stands for Dummy, GEO stands for the first three letters of the common-name and 1 denotes that this is the first dummy-array in labelled COMMON. By this action it is immediately clear that only array C is of importance in this subroutine.
CHAPTER 3

FORTRAN-statements in CSMP main-program

- All variables used in a FORTRAN-statement in mainprogram, but not used in a CSMP-statement according to the rules in par. 2.3, are not available in CSMP III COMMON. This means also that these variables are not available for output on OUTPUT- or PRINT-labels and also type-declarations are not taken care of. Output through the FORTRAN WRITE-statement is possible, however.

- FORTRAN-labels in mainprogram must be preceded by a '/'-sign in first column; f.i. /DIMENSION, /DATA, /REAL, /INTEGER. These labels can be continued on next line by placing the FORTRAN continuationmark ('$') before the line which is a continuation of the preceding line.

3.1 Use of REAL- and INTEGER-labels in main-program

- Variables that occur in main-program, but don't occur in CSMP III COMMON apply to normal FORTRAN-rules for declaration.
- This means: variables starting with I, J, K, L, M, or N are always integer. All other variables are real.
- To deviate from these rules use INTEGER- and REAL-labels. It is recommended to use these labels always for variables that don't occur in CSMP COMMON so as to state explicitly what type of variables are used.
- Lay-out of main-program using INTEGER- and REAL-labels:

```
TITLE ..........
INITIAL
FIXED ......
/      REAL ......
/      INTEGER ......
|
END
STOP
ENDJOB
```
note: a slash (') is placed at beginning of line to indicate to the compiler that this is a FORTRAN-statement. This part of the program is not translated by the CSMP-compiler, rather, these lines are directly placed in UPDATE.FOR. Therefore, the R from REAL must start in the 7th column.

- Put all FORTRAN-statements beginning with a slash together at the beginning the program.
- List on FIXED-statement only variables occurring in CSMP III COMMON, otherwise use FORTRAN-statement INTEGER. (See also par. 2.3 "Variables that automatically occur in CSMP III COMMON").

EXAMPLE: (See example 3 GEOM.CSM)
4.1 To force output at PRDEL.

Using CSMP PRINT-statements to print output of results of a variable time-step integration algorithm will cause that output is printed on desired PRDEL's. When output of results is generated by FORTRAN write routines PRDEL and OUTPUT-statements are not used. Then usually output-timings do not match with integration-steps. Then a dummy PRINT-statement can be used to force OUTPUT at PRDEL.

EXAMPLE:

note:-Before running this program on the VAX type
ASSIGN SYS$OUTPUT FOR020. (For alternative see Paragraph 1.1.)

-The subroutine in Example 8 can be used to get simulation timings on the terminal screen during the running of the program. Also other information, from which it can be concluded that further program execution is justified, may be sent to the screen (here "amount of organisms").

to force output on PRDEL's (FORCOUT.CSM). Example 8.

TITLE Example 8 to force output at PRDEL (FORCOUT.CSM)
INITIAL
HISTORY WRIT(3)
FIXED KEEP
PARAM STIME = 0.
TIMER FINTIM = 150., PRDEL = 10.
METHOD RKS
*** Dummy PRINT statement to force OUTPUT at PRDEL's ***
PRINT TIME

*****************************************************************

DYNAMIC
NOSORT
A = INTEGRAL(1.0, 0.1*A)
   CALL WRIT(KEEP, STIME, FINTIM, PRDEL, TIME, A)
END
STOP

*** Display of simulation time on TTY-screen ***
IMPLICIT REAL (A-Z)
INTEGER NLOC, KEEP
S = SAMPLE(NLOC, STIME, FINTIM, PRDEL)
IF (S*KEEP.LT.0.5) RETURN
WRITE(20,10) TIME, A
10 FORMAT(' Simulation time = ',F12.3,
       ' Amount of organisms = ',F12.3)
RETURN
END

ENDJOB
4.2 Use of more than 25 FIXED variables

- FIXED is used in mainprogram to declare variables integer.
- Variables, used in mainprogram and not listed on FIXED-statement are automatically real.
- Upto to 25 variables (or arrays) may be specified on FIXED-label, multiple FIXED-statements are allowed.
- It is possible to specify more then 25 single variables FIXED by using arrays. The array name is declared FIXED and need be filled with data in e.g. a subroutine. How to do this is shown in the following example:

**EXAMPLE:**

Use of more than 25 FIXED variables (FIXED.CSM). Example 9.
TITLE Example 9 Use of more than 25 fixed variables (FIXED.CSM)
INITIAL
NOSORT
STORAGE IA(26)
FIXED IA
/ INTEGER NR1, NR2, NR3, NR4, NR5, NR6, NR7, NR8, NR9
/ INTEGER NR10, NR11, NR12, NR13, NR14, NR15, NR16, NR17, NR18
/ INTEGER NR19, NR20, NR21, NR22, NR23, NR24, NR25, NR26
TABLE IA(1-26) = 101, 102, 103, 104, 105, 106, 107, 108,
109, 110, 111, 112, 113, 114, 115, 116, 117,...
118, 119, 120, 121, 122, 123, 124, 125, 126
CALL INTFIL(IA, NR1, NR2, NR3, NR4, NR5, NR6, NR7, NR8, NR9, NR10,...
NR11, NR12, NR13, NR14, NR15, NR16, NR17, NR18, NR19,...
NR20, NR21, NR22, NR23, NR24, NR25, NR26)
TIMER FINALM = 48., OUTDEL = 1.
METHOD RKS
OUTPUT A
DYNAMIC
A = INTGRL(1., 0.1*A)
TERMINAL
CALL WRIT(NR1, NR5, NR10, NR15, NR20, NR25)
END
STOP

SUBROUTINE INTFIL(IA, NR1, NR2, NR3, NR4, NR5, NR6, NR7, NR8, NR9, NR10,...
NR11, NR12, NR13, NR14, NR15, NR16, NR17, NR18, NR19,...
NR20, NR21, NR22, NR23, NR24, NR25, NR26)
IMPLICIT INTEGER (A-Z)
DIMENSION IA(26)
NR1 = IA(1)
NR2 = IA(2)
NR3 = IA(3)
NR4 = IA(4)
NR5 = IA(5)
NR6 = IA(6)
NR7 = IA(7)
NR8 = IA(8)
NR9 = IA(9)
NR10 = IA(10)
NR11 = IA(11)
NR12 = IA(12)
NR13 = IA(13)
NR14 = IA(14)
NR15 = IA(15)
NR16 = IA(16)
NR17 = IA(17)
NR18 = IA(18)
NR19 = IA(19)
NR20 = IA(20)
NR21 = IA(21)
NR22 = IA(22)
NR23 = IA(23)
NR24 = IA(24)
NR25 = IA(25)
NR26 = IA(26)
RETURN
END

SUBROUTINE WRIT(NR1, NR5, NR10, NR15, NR20, NR25)
IMPLICIT INTEGER (A-Z)
WRITE(20, 20) NR1, NR5, NR10, NR15, NR20, NR25
20 FORMAT (6I6)
RETURN
END

ENDJOB
References


APPENDIX

Modifications necessary to execute examples on IBM PC-AT machines.

- All programs described above have been tested on an IBM PC-AT under DOS 3.0.
- The compiler used for testing was: IBM FORTRAN 2.0 (a subset of the standard FORTRAN 77)
- The CSMP version was named PCSMP.

The testing resulted in some modifications, mainly with respect to I/O and VAX DCL commands. One modification had to be made due to a difference in the CSMP specification.

RELGROW.CSM
- No modifications necessary.

FUNC.CSM
- The statement TYPE is not supported by the compiler.
  TYPE as used in FUNC.CSM normally directs the output to the default output (in most cases the screen). It should be replaced with either WRITE (*,10) or WRITE (0,10).
  * and 0 in these statements both represent the DOS Standard Input or Standard Output; these statements thus act like TYPE.

FUNSUB.CSM
- This program runs on IBM PC-AT, but reacts differently during runtime.

  1. Without modification, program execution will result in a prompt:

     Filename missing or blank. Please enter name UNIT 20?

     This prompt is caused by the statement WRITE (20,10). In this statement, 20 represents an unit number which has to be attached to a file or output device.

     When the prompt is answered with eg. FORO20.DAT, all data will be directed to the file FORO20.DAT.

     when the prompt is answered with eg. CON, all data will be directed to the screen.

     The VAX-VMS command ASSIGN SYS$OUTPUT FORO20 (page 5, note 3) is now actually performed during runtime.

To perform the VAX-VMS command before runtime, one should use the file UPDATE.EXE, produced by the linking process of PCSMP.
Giving the command: UPDATE CON results in sending the output of unit 20 to the screen, whereas giving the command: UPDATE FORO20.DAT
results in sending the output of unit 20 to the file FORO20.DAT.

2. With modification of the program, one fixes the output to a particular destination. This can be done as follows:
   put the statement
   OPEN (20, FILE='FORO20.DAT', STATUS='NEW')
   before the first WRITE (20, ....) statement.
   The OPEN statement will create a file FORO20.DAT to which the output of unit 20 is directed.

   To close the FORO20.DAT file, insert after the last WRITE (20, ....) statement, the statement CLOSE (20).

FUNCHIS.CSM
- No modifications necessary.

GEOM.CSM
- Declaration of array RAD (statement: DIMENSION RAD(N+1)) in SUBROUTINE GEOMET is not allowed. The compiler does not permit an arithmetic expression to define the size of an array.

Several modifications to solve this problem are possible. A structured modification is the following:
   transfer an extra parameter N1 (of type integer) to the subroutine GEOMET, with N1 denoting the size of the array RAD.

For subroutine WRIT a similar modification might be made, but in this particular case it is possible to declare the array RAD with size N (DIMENSION RAD(N)), since the algorithm of WRIT does not refer to array element RAD(N+1).

(see modified listing of example 3, GEOM.CSM)

The problems arising with the statements WRITE(20,10) and WRITE(20,20) may be handled as discussed for FUNSUB.CSM.
The translator of PCSMP requires that subroutine names match the syntax of type REAL. Therefore, the identifier NAME, which matches the syntax of type INTEGER, should be modified in eg. XNAME.

This rule does not apply for subroutines without a list of arguments (see also example BLACOM.CSM).

(see modified listing of example 4a, SUBDUM.CSM)
SUBPRO.CSM
- Modifications as for SUBDUM.CSM.

SUBSOR.CSM
- Modifications as for SUBDUM.CSM.

BLACOM.CSM
- No modifications necessary.
- Note that first and last subroutine match syntax type INTEGER. However, they can be used, since no list of arguments is present (see also SUBDUM.CSM).

FILLCO.CSM
- Modifications as for FUNSUB.CSM.

CONVRT.CSM
- No modifications necessary.

FORCOUT.CSM
- Modifications as for FUNSUB.CSM.

FIXED.CSM
- Modifications as for SUBDUM.CSM, and FUNSUB.CSM.
SIMULATION REPORTS CABO-TT

Reports published in this series

1. UNGAR, E. & H. van KEULEN:

2. CORDOVA, J., F.W.T. PENNING de VRIES & H.H. van LAAR:

3. MARLETTO, V. & H. van KEULEN:
   Winter wheat experiments in The Netherlands and Italy analysed by the SUCROS model. 1984, 61 pp.

4. GENG, S., F.W.T. PENNING de VRIES & I. SUPIT:

5. GENG, S., F.W.T. PENNING de VRIES & I. SUPIT:

6. BENSCHOP, M.:
   TUCROS, een simulatiemodel voor de tulpecultivar "Apeldoorn". 1985, 83 pp.

7. SUPIT, I.:

8. VEN, G.W.J. van de: