E. REMIDDI

CERN - DATA HANDLING DIVISION
DD/73/10
Hugo Strubbe
April, 1973

# Internal Mechanism of a Schoonschip* Calculation

Schoonschip is a general purpose "Algebraic Manipulation" program. It is designed to do long - but in principle straightforward - analytic calculations. It can be used interactively.

It is very fast in execution and very economical in storage (25K). This is achieved by writing the program almost entirely in (CDC 6000) machine code. Therefore, the representation of the algebraic formulae (list structure) and the calculation mechanism can be set up very efficiently. Input, Output and a few numerical tasks are done in Fortran.

*Schoonschip: A CDC 6600 program for symbolic evaluation of algebraic expressions. CERN preprint by M. Veltman, 1967. CERN Program Library R 201.

(Part of this text to be presented at the Technical Meeting of SEAS in Rimini, Italy, 9-13 April, 1973)

## I. Introduction to Schoonschip

The following text summarises the basic features of Schoonschip:

```
C  SCHOONSCHIP , A PROGRAM FOR ALGEBRAIC MANIPULATIONS.
C  SCHOONSCHIP , WRITTEN IN 1967 BY M. VELTMAN AT CERN.
C  SCHOONSCHIP , VERSION OF JANUARY 1, 1973.

C  SCHOONSCHIP IS A COMPUTER PROGRAM, MAINLY WRITTEN IN COMPASS , THE CDC 6000
 AND 7000 ASSEMBLER LANGUAGE. IT IS ABLE TO PERFORM  ALGEBRAIC MANIPULATIONS
 WHAT MEANS THAT IT CAN SOLVE PROBLEMS OF THE FOLLOWING KIND

 INPUT                                          OUTPUT
 (A+B)**2                                       A**2+B**2+2*A*B
 THIS IS ESSENTIALLY ADDITION AND MULTIPLICATION OF POLYNOMIALS.

 INPUT                                          OUTPUT
 F(A,B)=B**2                                    2*A*B+A**2
 TOGETHER WITH
 F(A,B)=(A+B)**2
 THIS ILLUSTRATES THE SUBSTITUTION FACILITY.

 INPUT                                          OUTPUT
 A**2*B**4*C**3*D                               A**2*Z*D
 TOGETHER WITH
 A**2*B**2=X
 B**4*D**2=Y
 B**4*C**3=Z
 THIS REFLECTS THE POSSIBILITY OF  PATTERN MATCHING .

FINALLY, THE PROGRAM IS ABLE TO DEAL WITH VECTORS, NON COMMUTATIVE QUANTITIES,
GAMMA MATRICES...IT CAN PERFORM TRACE CALCULATIONS,NUMERICAL EVALUATION,COMPLEX
CONJUGATION,...WHAT MAKES IT TO A VERY USEFUL TOOL FOR HIGH ENERGY PHYSICS
CALCULATIONS.
```

What a Schoonschip orogram looks like can be seen from the example shown in Appendix I.

For full details see the manual (which is sent on tape, together with the Schoonschip system and is in fact 100 pages of examples, to be run as a Schoonschip program).

## II. Introduction to Schoonschip's Internal Mechanism

Understanding Schoonschip's internal structure can help one to run Schoonschip more efficiently. Understanding its dump in the case of an error message can help one to figure out where exactly the system stopped.

a) At read-in time, name lists are constructed which establish the correspondence between the print name and the internal numbering of the variables. These name lists are the arrays NVGEH (for vectors), NFGEH (for functions), NVIGEH (for indices), NAGEH (for symbols). The representation of the different kinds of variables is explained in the following text:

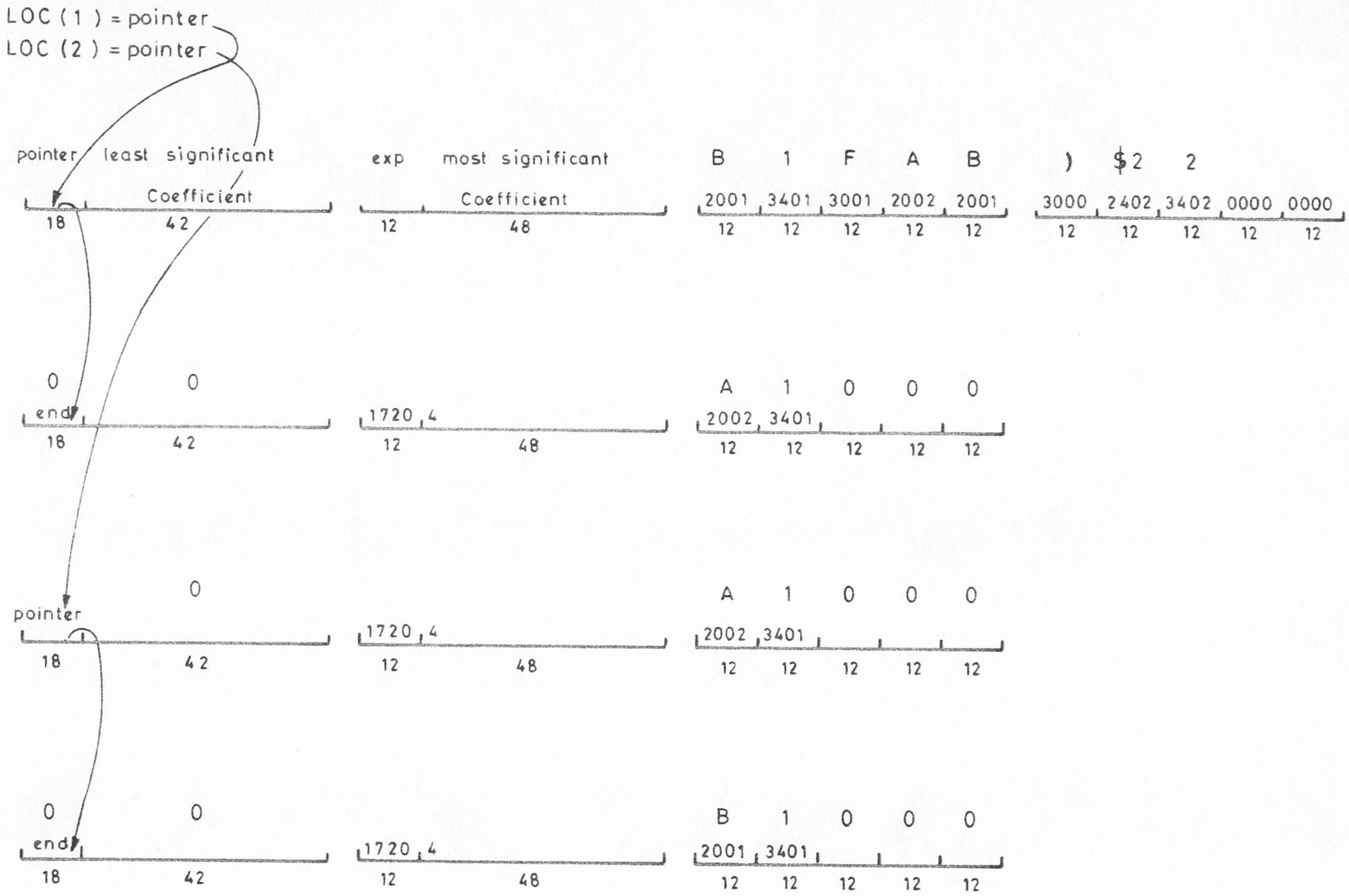
```
C  NOTE ON SCHOONSCHIPS INTERNAL STRUCTURE AND MECHANISM
C  THESE ARE THE CONVENTIONS FOR THE REPRESENTATION OF THE DIFFERENT QUANTITIES
 WHICH CAN OCCUR. EVERY QUANTITY IS  12  BITS LONG.
 THE HEADING  4  BITS SPECIFY ITS TYPE.
 THE TRAILING 8  BITS SPECIFY ITS NAME.
C
 HEADING     KIND                EXAMPLE
 BINARY                          OCTAL

 0000        DUMMY               0007  DUMMY NR 7
 0001        INDEX               0405  INDEX NR 5
 0010        VECTOR              1010  VECTOR NR 8
 0011        OPERATOR            1420  COMPLEX CONJUGATION OPERATOR
 0100        SYMBOL              2001  SYMBOL I
 0101        $ EXPRESSION        2425  $21
 0110        FUNCTION            3020  FUNCTION DP
                                 3000  END OF ARGUMENT LIST OF FUNCTION
 0111        NUMBER              3776  =1
 10**        VECTOR COMPONENT    4142  SECOND COMPONENT OF VECTOR NR 3
             THE  10  IS FOLLOWED BY A 5 BIT VECTOR NAME AND A 5 BIT NUMBER
 11**        DOTPRODUCT          6105  DOTPRODUCT OF VECTOR NR 2 AND VECTOR NR 5
             THE  11  IS FOLLOWED BY TWO 5 BIT VECTOR NAMES

THIS EXPLAINS WHY E.G. 256 SYMBOLS ARE ALLOWED BUT ONLY 32 VECTORS, EACH WITH
AT MOST  32  COMPONENTS.
QUANTITIES  1041  UP TO  1377 INDICATE FUNCTIONS THAT ARE THEMSELVES A
FUNCTION ARGUMENT.
```

b) The list structure of the Input can be seen in the following example:

B*F(A,B)*(A+B)**2+A

The input store is the array IT(2000). LOC(200) is an array that contains a pointer to each expression (e.g. LOC(5) contains the pointer to $5, represented by 2405).

(see following page)




LIST STRUCTURE OF INPUT FOR

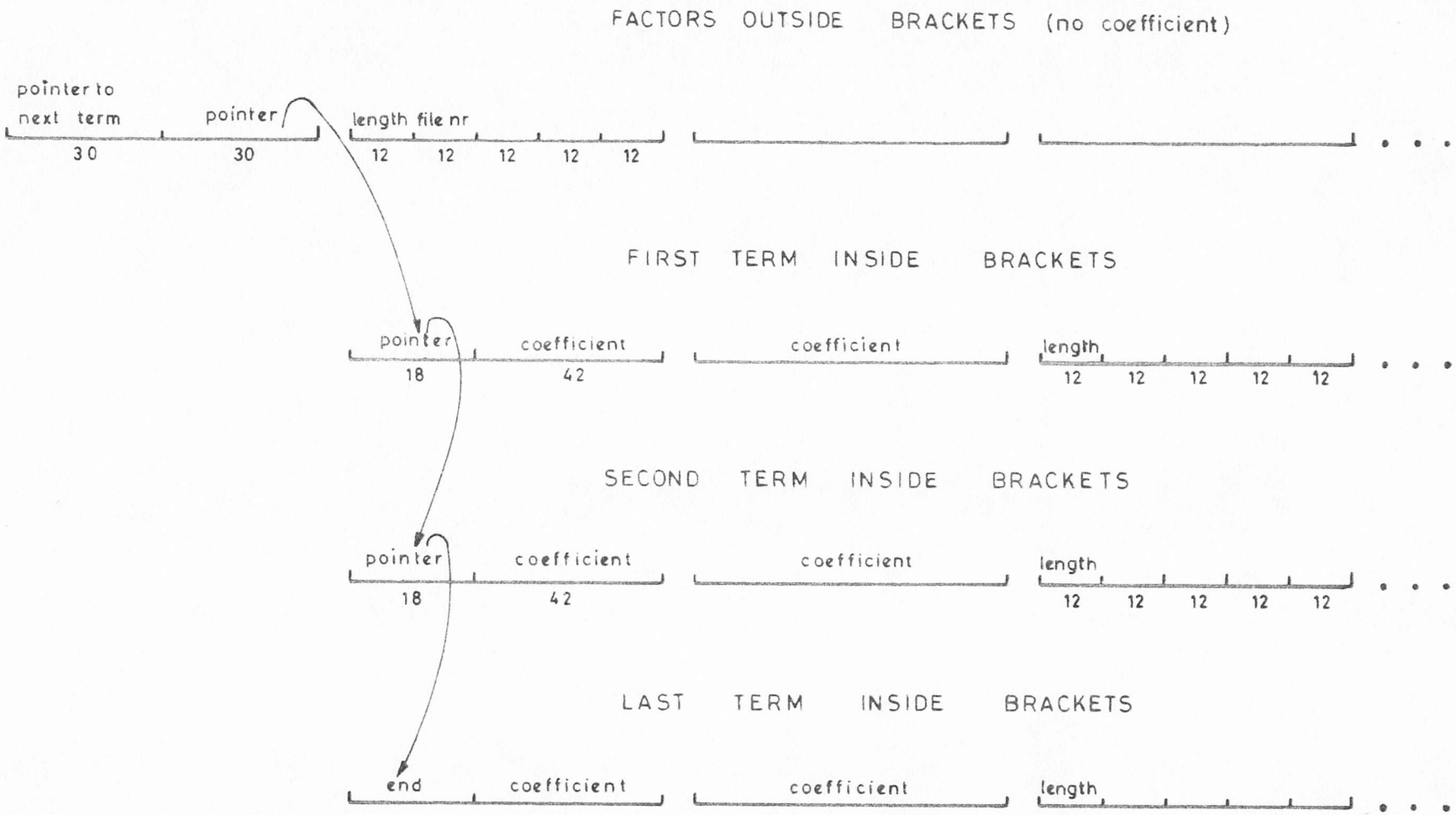
B * F ( A , B ) * ( A + B ) * * 2 + A

Remarks:
- CDC computer words are 60 bits long. Floating point numbers are represented by 12 bits for exponent and signs; 48 bits for mantisse (twice this convention for double precision quantities).
- The formulae are represented in a densely packed way.
- The use of pointers is minimal: only to connect the additive parts.
- In the internal representation, symbols are always followed by an exponent; functions cannot be followed by an exponent.

c) The list structure of the Output is slightly different as it is possible to factor out some quantities:

```
B11*B12*B13*...*(A11+A12+A13+....)
B21*B22*B23*...*(A21+A22+A23+....)
:::
```

Functions and vectors are always factored out by the system. For other quantities it can be requested by the user. As each term is calculated a scan through the present state of the result has to be performed and is optimal when (length of quantities outside brackets)≈(length of quantities inside brackets).




LIST STRUCTURE OF OUTPUT

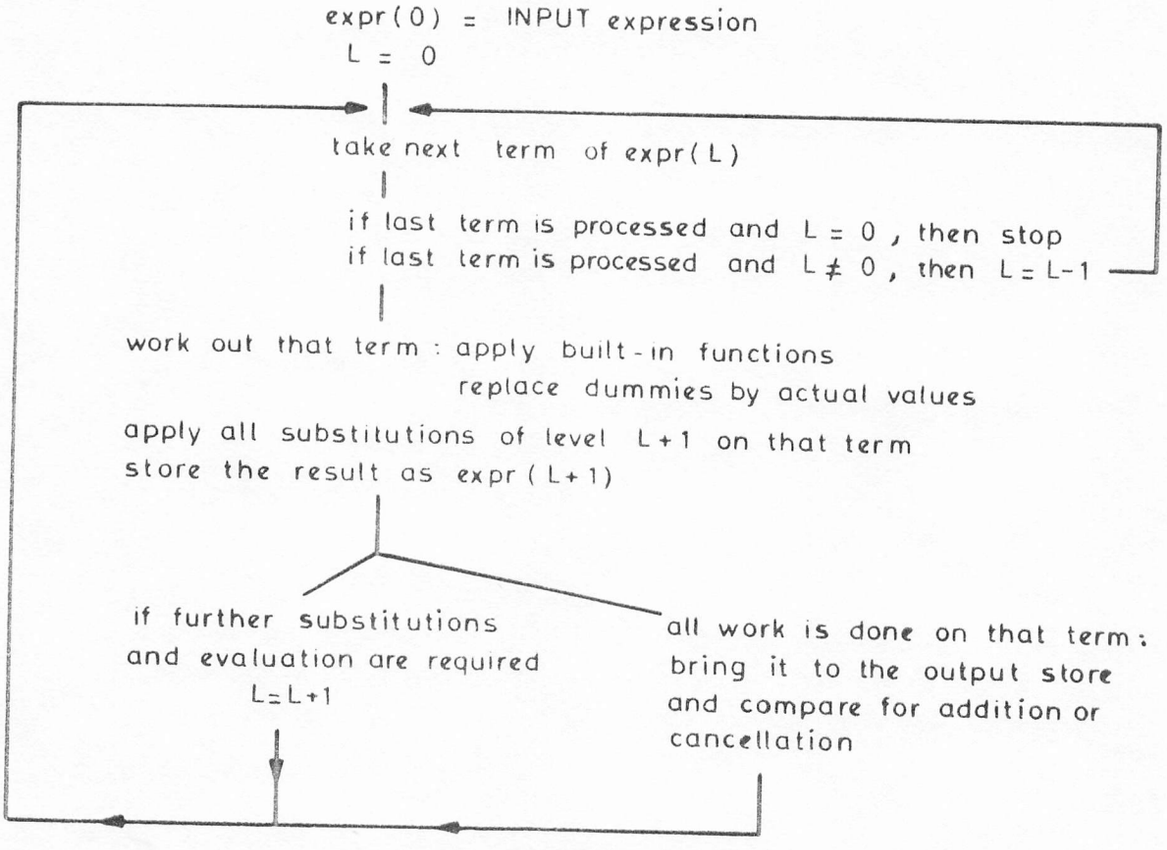
The Output store is the array NU(5000).

d) From Input to Output

Substitutions are performed in a sequential way. Each substitution is performed at a certain level, specified by the user. Working out a pair of brackets is essentially a substitution and requires an additional substitution level.

The calculation mechanism works along the following lines:

notation: L = substitution level.
expr(L) = expression under consideration at level L.

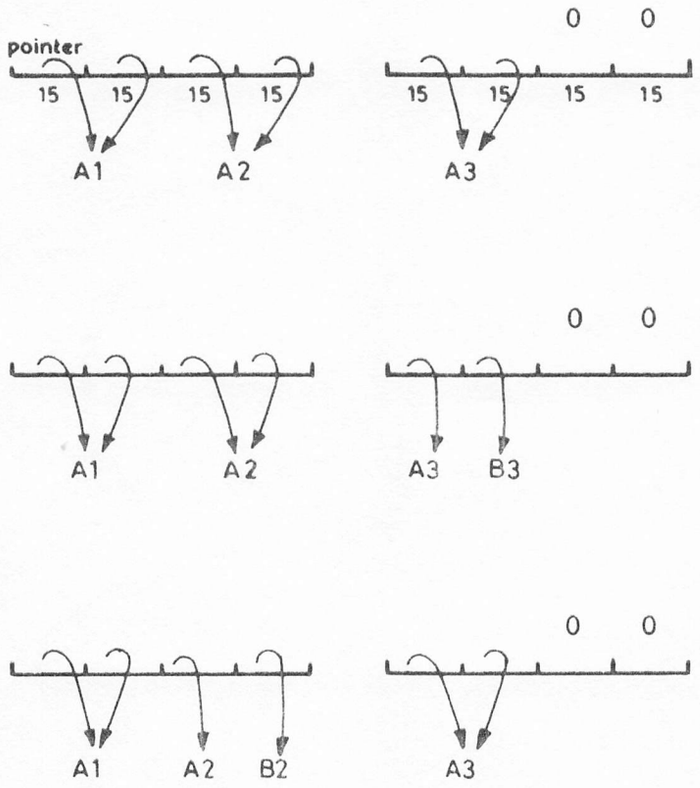



CALCULATION MECHANISM

The administration implied in the statement "take next term of expr(L)" is done by specifying pointers to the first term, and to the current term.

example: (A1+B1)*(A2+B2)*(A3+B3)

is successively referred to by




.....etc.

These "administration words" can easily be recognised in a Schoonschip dump, and allows the user to trace back which was the term under consideration at the moment of the dump.

As soon as some of the expressions involved reside on disk rather than in memory, the situation becomes much more complicated. Such cases are not described here.

e) Use of the Input store

The input store contains (in the following order):

1. The built-in formulae (necessary for dirac algebra, epsilon reduction, etc.)
2. The expression (+ its subexpressions) to be evaluated.
3. The right-hand sides of the substitutions.
4. The substitutions and commands.
   a) control word specifying the level, the expression number of right-hand side and a pointer to the next control word.
   b) left-hand side: one quantity per word.
5. A separator symbol: 7740000000 0000000000
6. The intermediate steps of all substitutions on the last term. i.e. expr(L) followed by its administration words for all L-values involved.

f) Overflow of the Output store

The output mechanism works as follows: The calculation generates a term, which is compared with the terms already present in the output store. This can lead to an addition or to a new term.

Assume the output store is full. The incoming term is compared with the output store. If no addition occurs, the term goes to tape 4. (No test is made whether such a term is already on the tape). When all terms are generated, the output store is emptied (e.g. by printing). Now tape 4 is read in, its terms go into the output store and possible overflow is written on tape 5. Then the function of tape 5 and tape 4 is interchanged, etc., until all terms are printed. (Tape 4 and tape 5 are disk files).

## III. Looking inside Schoonschip

There are three instructions that cause a dump of the internal (or "coded") representations:

1. PRINT CINPUT: dumps after analysis of the input
2. PRINT COUTPUT: dumps at the end of the calculation
3. PRINT ERROR: dumps whenever an error message is given.

Even in correct calculations it is possible to force a dump, in order to see what was going on at a particular moment, by using one of the following tricks:

```
1.  V  P
    N  MU
    Z  EXP=(A+B+C)**3
    ID,A=AA
    10,B=BB*P(MU)
    PRINT ERROR
    .
    .
    .
```

Whenever the index, specified on the N card, is seen in the output as index of a vector, the "error 1001" occurs. Varying the place where P(MU) is appended allows to specify exactly at which point of the calculation the dump is made.

```
2.  N  500
    Z  EXP=(A+B+C)**10
    PRINT ERROR
    .
    .
    .
```

During every calculation the number of "multiplications" or "insertions" is counted. This number is given in the statistics. However, the error "Insert count limit" is forced when

(nr of insertions)=(nr on N card - 300).

In this example the error will occur after 200 multiplications. This feature allows to follow the calculation step-by-step.

N.B. Remember, N 15 means floating point numbers are printed with 15 digits.

N R means floating point numbers are converted to a quotient of 2 integers.

## IV. Comments on a few Schoonschip Facilities

a) Functions are essentially considered as non-commuting quantities. (Special instructions: Allow Inbetween and Allow Disorder can be used to make them commuting anyway).

b) The normal Spinor algebra is built-in.

c) The program always attempts numerical evaluation of an expression before doing algebra on it. E.g. (1+2)**100 is not computed in the same way as (A+B)**100. DX is a built-in function which generates a call to subroutine EXTRA, to be given by the user in Fortran.

d) The user can add his own "built-in functions" (called X expressions).

e) There is no branching possible in a Schoonschip program. However, this can be easily overcome by using selective substitutions. e.g. IFSMA, IFGRE, IFEQU tests on the coefficient of each term and appends a symbol to each successful term. COUNT computes the the weight, N, of each term and appends to it, e.g. F(N) or A**N ... Very complicated pattern matching can then be used to pick out a specific term:

```
A**N+*B**N+*C**M+*F(A,B,N+,M+) = ...
```

f) The "freezing" of the expression

```
Z=V11*V12*....*(A11+A12+...)
 +V21*V22*....*(A21+A22+...)
  ...
```

leads to

```
Z=V11*V12*....*DF(Z,1)
 +V21*V22*....*DF(Z,2)
  ...
```

Now substitutions on $V_{ij}$ can be done (which may involve DF). DF remains unchanged. At the end DF is again replaced by the $A_{ij}$ values (with the command EXPAND).

g) Output as well as input can be stored on disk and used again later. The first feature allows to cut a long calculation into little pieces (write common, enter common, R input, etc.) The second feature is useful in cases where a kind of "DO LOOP" is required (which is not available in Schoonschip). (Tape copy, tape read, etc.)

h) It is up to the user to specify whether expressions have to be kept or deleted after a part of the calculation. If required, a transfer from output store to input store is made. When the expression is too long for the input store, it is automatically stored on disk (keep, delete, *begin, *next, etc.).

APPENDIX I

```
SCHOONSCHIP , VERSION OF JANUARY 1, 1973

   TIME          .40  SECONDS


                  C  EXAMPLE. EXPANSION IN POWER SERIES UP TO TENTH ORDER IN Y .

                  FUNCTION FS,FC
                  C  SERIES TO BE EXPANDED.
                  Z EXPAN=FS((3*Y**2))*(FS((4*Y/3)))**2*FC((2*Y))/Y**4  * A * B
                  C  THIS EXPRESSION IS NOW EXAMINED FOR SUBSTITUTIONS.

                  C  FIRST SUBSTITUTION.  Z+  MEANS, REPLACE FOR ANY VALUE OF  Z .
          L 3     ID,FS(Z+)=Z-Z**3/6+Z**5/120-Z**7/5040+Z**9/362880-Z**11/39916800
                  C  WHEREVER IN EXPAN THE FACTOR  FS  IS SEEN,  FS  IS REPLACED BY THE
                  C  RIGHT HAND SIDE OF THE SUBSTITUTION.

                  C  SECOND SUBSTITUTION.
          L 4     ID,FC(Z+)=1-Z**2/2+Z**4/24-Z**6/720+Z**8/40320-Z**10/3628800
                  C  WHEREVER IN EXPAN THE FACTOR  FC  IS SEEN,  FC  IS REPLACED BY THE
                  C  RIGHT HAND SIDE OF THE SUBSTITUTION.

                  C  AFTER THESE SUBSTITUTIONS, ALL TERMS OF EXPAN HAVE THE FORM
                  C                  A*B*Y**N   WITH N UP TO 50.

                  C  WE WANT TO TRUNCATE THE SERIES AT  Y**10 .
          L 5     ID,A=Y**(-10)
                  C  DIVIDE EXPAN BY  Y**10
          L 6     ID,Y=0
                  C  TERMS  Y**N  WITH  N  POSITIVE ARE SET TO ZERO.
          L 7     ID,B=Y**10
                  C  MULTIPLY EXPAN WITH  Y**10 .

                  * END

SYMBOLS         I=I, Y, A, B.

FUNCTIONS       D, EPF=I, G=I, GI, G5=I, G6=C, G7, UG=C, UBG, DD=U, DB,
                DT, DS=U, DX, DK, DP, DF=U, FS, FC.

RUNNING TIME (SEC)       18.83                              USED    MAXIMUM
TERMS IN OUTPUT              6
GENERATED TERMS             80          INPUT SPACE          352       2000
EQUAL TERMS                 74          OUTPUT SPACE          21       4890
CANCELLATIONS                0          NR. OF EXPR.          67        260
RECORDS WRITTEN              0          ID. REGISTER          11        500
MULTIPLICATIONS          41773          FUNCT. REG.           12        750


 EXPAN =

    + 5.33333 - 1.38272E1*Y**2 + 2.6257*Y**4 + 1.65662E1*Y**6 - 1.13265E1*Y**8 - 3.23859*Y**10  +0

END OF RUN.   TIME       19.28  SECONDS
```