CERN COMPUTER CENTRE

PROGRAM LIBRARY

In addition to LONG WRITE-UP

R 201

15.4.1977

Author: H. Strubbe

# MANUAL FOR SCHOONSCHIP

## Addition to the 1973 Version

4.2 Meaning of built-in functions. Addition to Table 2

a) Numerical functions (i.e. DP, DK, DT, DB) are not worked out as long as their arguments are not numbers. They can therefore appear in the output.

Example : Z Z = DK(N,M)*F(N,M) becomes Z = DK(N,M)*F(N,M)

Z Z = DK(2,3) becomes Z = 0

b) DD(J,K) is a replacement function. All the appearances in the term of J or -J will be replaced by K or -K respectively. If no J is present, the term is unmodified. In both cases, the DD-function disappears after trying its action. A DD-function can never appear in the output.

Example : Z Z = DD(M,N) becomes Z = 1

Z Z = F(M,M,-M,-M)*DD(M,N) becomes Z = F(N,N,-N,-N)

c) D(J,K) is a special replacement function. Its properties are such that it acts like a Kronecker delta function in the case of Einstein summation convention. This means that e.g. $P_\mu S_{\mu\nu}$ should become $P_\nu$ while $P_\mu S_{\nu\lambda}$ should be unchanged.

The rules are: (i) D(J,K) has the same action as D(K,J). The arguments are internally ordered according to the name lists;

(ii) if both of the indices appear an even number of times in the part of the term that precedes the D-function, then no action is taken;

(iii) otherwise, the first occurrence in the term of the first argument is replaced by the second argument.

Example: Z Z=F(N,M)*D(M,K)*DEL(M,J) becomes Z=F(N,K)*DEL(M,J)

Z Z=F(N,M)*DEL(M,J)*D(M,K) is unchanged.

NB: Such cases do not really appear in conventional spinor calculations, where each index occurs at the most twice.

In addition to LONG WRITE-UP

5.2.1. Input from TAPE3 via BLOCKS.

A series of tape commands, listed in the 1973 manual was intended to store SCHOONSCHIP subprograms for later use. These tape commands still exist, but the new BLOCK facility makes the use of subprograms much easier and more flexible.

A BLOCK is a subprogram - with dummy arguments - that is defined once for ever. It is invoked by using a COPY statement, which contains the actual parameters.

BLOCK name (arg1, arg2,.....argN).

The name must be less than 50 characters long. It should contain only letters and digits. Blanks are irrelevent in it. The number of arguments is not restricted. If there are none, the brackets still have to be present.

The statements of the block.

Any (part of a) SCHOONSCHIP statement can appear inside the block, with the exception of BLOCK, COPY, DO, ENDDO.
The statements are stored on TAPE3 for later use, but not executed.

ENDBLOCK.

Terminates the BLOCK.

COPY name (arg1, arg2,.....argN).

A BLOCK with identical name is searched for on TAPE3. Its statements are used as SCHOONSCHIP input, while doing a textual replacement of the dummy BLOCK arguments by the actual COPY arguments. If there are no arguments, the brackets should be present nevertheless.

The number of blocks is unrestricted.
Blocks can be read in at the beginning of the program. They should be terminated by a WRITE BLOCKS statement. Alternatively, they can be stored on disk as a kind of library. In that case they should be present on TAPE3 (by using SCOPE control cards).
In the case where blocks have to be added or replaced, the program should start with

```
ENTER BLOCKS
BLOCK name (arg1, arg2,.....argN)
...
NBLOCK name1, name2,...
WRITE BLOCKS
*BEGIN
```

ENTER BLOC(KS) only 10 characters relevant
reads in the existing blocks of the library

BLOCK name (arg1, arg2,.....argN)
if "name" is an already existing block, this block will replace it.
if "name" did not yet exist, it will be added to the library.

NBLOCK name1, name2,...
deletes the named blocks from the library.

WRITE BLOC(KS) only 10 characters relevant
terminates the updating of the library. A new version is written back on TAPE3. Moreover, the status of the library is printed.

---

5.2.2. Input via DO loops

Another way to generate input for SCHOONSCHIP is the use of a DO loop. This feature copies the input cards, while textually replacing the DO loop index by its value.

DO J=J1, J2, J3.

means: DO for J going from J1 to J2 in steps of J3. If J3 is absent, J3=1 is assumed. J3 can be negative. The test on the value of J is done at the end of the loop (single step DO loop). In other words, DO J=10,0,1 is executed once. The DO statement cannot have a continuation. J1, J2, J3 should be indices with absolute value less than 128 or they can be previously defined loop indices. J, the DO loop index, should not be defined in a namelist.

The statements of the DO loop .

Any (part of a) SCHOONSCHIP statement can appear inside the loop, including DO, COPY and * statements. The statements are copied to memory till the loop terminator is seen.
Then their execution starts.

ENDDO I1, I2, ...

Closes the DO loops with loop indices I1, I2, ...
This statement cannot have a continuation.

---