INTERNAL REPRESENTATION OF DATA STRUCTURES

In order to use Assembly or Fortran procedures in a SIMULA program one must know how data and data structures are represented internally. Variables of type INTEGER, SHORT INTEGER, REAL, LONG REAL, and CHARACTER have their obvious internal representations: fullword, halfword, single precision floating point, double precision floating point and EBCDIC character.

A BOOLEAN is X'00' for FALSE and X'01' for TRUE.

A REF (... variable is the fullword block instance address, or, if none, X'00FF0000', and an array is the fullword array object address.

A text variable is represented within a block as a 3-word text descriptor: The first word is the address of the text storage block (text object), the second is the address of the first byte of the text -1, the third word is divided into two halfwords: the first is the length of the text and the second is the position indicator.

In a block the quantities are allocated in the same sequence as they are declared, with the spaces and alignments given in table 3.2. The first quantity of a block is allocated at the displacement 8 from the blocks starting address.
Array object format.

-1

0 (0)

4 (4)

8 (8)

12 (C)

16 (10)

20 (14)

24 (18)

28 (1C)

-1 in the first word indicates that this is an array object.

QL is the array object length.

BA is the address of the element A(0,0,...,0).

QUALIF is a word identifying the qualification of a REF array, or unused.

n number of subscripts.

d dope vector

LIND lower index

UIND upper index

type array type code (App. G).
Dope vector and index checking.

Assume the array declaration

\[
A \left( \{i\} : u, \ldots, \{n\} : u(n) \right);
\]

Then

\[
d(1) = u(1) - l(1) + 1
\]

\[
d(i) = d(i-1) \times (u(i) - l(i) + 1), \quad i = 2, \ldots, n - 1
\]

\[
d(0) = 1 \text{ (not present in object)}
\]

\[
LINC := 0;
\]

for \( i := 1 \) step 1 until \( n \) do

\[
LINC := LINC + l(i) \times d(i-1);
\]

\[
UINC := 0;
\]

for \( i := 1 \) step 1 until \( n \) do

\[
UINC := UINC + u(i) \times d(i-1);
\]

The computation of the address of \( A(\{i\}, \ldots, \{n\}) \) is described by the following algorithm:

\[
t := 0;
\]

for \( k := 1 \) step 1 until \( n \) do

\[
t := t + l(k) \times d(k-1);
\]

error ("subscript bounds");

address := \( t \times \text{element length} + BA \);
Text object format.

-2 Indicates that this is a text object.

CL is the length of the text contents.

DL is the text object length.

\[ DL = \frac{(CL + 12 + 7)}{8} \times 8 \]