Appendix F

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.

```
               __________________________________
 0  (0)    !                -1                  !
           !____________________________________!
 4  (4)    !                 0                  !
           !____________________________________!
 8  (8)    !                OL                  !
           !____________________________________!
12  (C)    !                BA                  !
           !____________________________________!
16  (10)   !              QUALIF                !
           !____________________________________!
20  (14)   !               LIND                 !
           !____________________________________!
24  (18)   !               UIND                 !
           !____________________________________!
28  (1C)   !  n    !  type   !         d.       !
           !_______!_________!__________________!
           !                                    !
           !__________________                  !
           !      dn-1        !                 !
           !__________________!                 !
           !                                    !
           !          array elements            !
           !____________________________________!
```

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

OL 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 (l(i) : u, ... , l(n) : u(n));
```

Then

```
          d(1) = u(1) - l(1) + 1

          d(i) = d(i-1)*(u(i) - l(i) + 1), i = 2, ..., n - 1

          d(0) = 1 (not present in object)

          LIND := 0;

          for  i := 1 step 1 until n do LIND := LIND + l(i)*d(i-1);

          UIND := 0;

          for  i := 1 step 1 until n do LIND := LIND + u(i)*d(i-1);
```

The computation of the adress of A(i , ..., i(n)) is described by the following algorithm:

```
          t := 0;

          for  k := 1 step 1 until n do t := t + i(k)*d(k-1);

error ("subscriptbounds");
address := t * elementlength + BA;
```

Text object format.

```
          ----------------------------
0       !              -2              !
        !------------------------------!
4       !               0              !
        !------------------------------!
8       !      CL      !      OL       !
        !--------------!---------------!
        !                              !
        !        text contents         !
        !------------------------------!
```

-2 indicates that this is a text object.

CL is the length of the text contents.

OL is the text object length.

```
OL  =   (CL + 12 + 7)// 8 * 8
```