3 Hardware and implementation defined restrictions and capacity limitations.

3.1 Permitted ranges of arithmetic quantities, precision of real number arithmetic.

The ranges of arithmetic quantities are listed in table 3.1.

quantity	maximum	minimum
INTEGER	2**31 - 1	-2**31
SHORT INTEGER	2**15 - 1	-2**15
REAL *)	((1-16)**-6)*16**63	16**-65
LONG REAL *)	((1-16)**-14)*16**63	16**-65

Table 3.1: Ranges of arithmetic quantities.

*) The range of the magnitudes of normalized numbers are given. A true zero is also representable, and the range of the magnitudes of negative numbers is the same as for positive numbers.

Integer arithmetic is performed exactly.

REAL quantities have a precision of 6 hexadecimal digits (around 7 decimal digits), LONG REAL quantities have 14 hexadecimals (around 16 decimal digits). There may be an additional loss of precision due to the rules for truncation of floating point operation results (see (4)).

3.2 Maximum sizes of block instances, arrays and texts.

The maximum size of a block instance is 4096 bytes. The size of a block instance depends on its parameters and declared quantities and can be computed using the information in 3.4.

There is no maximum size of an array, but if the last subscript is ignored, the array must not have more than 2**15 - 1 elements.

A text must not have length greater than 2**15 - 20.

3.3 Program capacity limitations.

Besides the capacity limitations given in the table below there are restrictions on the complexity of an expression, for which no comprehensive estimate can be given. See also 3.2.

Item		max value
Block level	1)	15
Prefix level of class or prefixed block	2)	62
Number of parameters to a procedure or a class (including parameters of prefixes)		127
Number of parameters to Fortran or assembly procedure.		18
For statement nesting level		255
Designational expressions in switch declaration		127
Number of virtuals specified in class (including prefixes)		255
Number of subscripts of array		127
Identifiers declared in one list		127
Number of differently spelled identifiers, compiled in minimum core area		1000
Number of differently spelled identifiers, independent of core area	3)	3072
Redeclaration level capacity	4)	30
Number of fixups	5)	Dep. on part.size
Number of external references	6)	200

1) The block level for a program point is the number of begin-end pairs that enclose it, excluding those of compound statements, but including connection blocks.

2) A class with no prefix has prefix level zero, and a class with prefix has prefix level one higher than its prefix.

3) This is normally 1500 user identifiers as default. In order to obtain the maximum, the compiler CSECT DEFAULT must be modified and the compiler regenerated to increase tha table size.

4) Block level plus the number of enclosing compound statements that have local labels and are also controlled statements or connection blocks.

5) The number of fixups generated by program constructions are:

Item	Fixups
Class declaration, prefixed block	4
Other block	3
then	1
else	1
label	1
inspect	3
when	1
for statement	2
switch	1

3.4 Object program storage requirements.

The SIMULA object program needs storage of three categories:

i) Storage for compiled code (load module).
ii) SIMULA working storage.
iii) System working storage.

The amount needed for i) is fixed during program execution and can be determined from the linkage-editor listing when the object program is created.

The amount of system working storage needed can be determined from (2).

The amount of SIMULA working storage needed is the sum of the storage needed for all referable block incarnations in the program, and a general overhead of 368 bytes (which includes the images of sysin and sysout).

Except for space for declared variables a block takes:

24 bytes if it is not terminated, or if it is an object of a class with local class attributes

24 bytes if it is a scheduled process

24 bytes if it is a procedure made visible through connection or remote referencing.

The block instance itself has an overhead of 8 bytes and the storage needed for parameters and declared quantities can be determined from table 3.2, which gives the number of bytes and the alignment factor for any quantity. Quantities are allocated in the order in which they are declared, then 4 bytes are added for each nesting level of for statements.

*. mode *. *. quant *.	declared or by value		parm.by reference		parm. by name	
	S	A	S	A	S	A
PROCEDURE LABEL,SWITCH	0	0	8	4	8	4
REAL	4	4	-	-	8	4
LONG REAL	8	8	-	-	8	4
INTEGER	4	4	-	-	8	4
SHORT INTEGER	2	2	-	-	8	4
BOOLEAN CHARACTER	1	1	-	-	8	4
REF	4	4	4	4	12	4
TEXT	12	4	12	4	8	4
REF ARRAY	4	4	4	4	12	4
array:s	4	4	4	4	8	4

Table 3.2 Space for quant:s in block instance .

S : size (bytes) for quantity.
A : alignment of quantity.

For a <type> procedure block instance, the result takes the space of a declared quantity of the same type.

Arrays and text blocks are allocated outside the block instance. An array needs the space required for all its elements according to table 3.2, and an overhead of 28+2*n bytes, where n is the number of subscripts of the array. A text block needs 12+n bytes, where n is the length of the main text.

The sizes of block instances, arrays and texts are always rounded upwards to a multiple of 8.