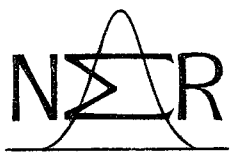
NORSK REGNESENTRAL / NORWEGIAN COMPUTING CENTER

Forskningsvn. 1 B, Blindern, Oslo 3, Norway, telefon (02) 46 69 30



Publication No. S.56.0

360/370 SIMULA

EXTERNAL PROCEDURE LIBRARY

Norwegian Computing Center

1st November 1973

Karel Babcicky

Note: The procedures described in this publication are compatible with the SIMULA Run Time System Release 2.1 and all its successors.

To NCC users of 360/370 SIMULA System at TEAMCO:

Any of the SIMULA external library procedures may be used in a SIMULA program without an explicit declaration, providing that

1) the SIMULA program is compiled using one of NCC's catalogued procedures.

2) %COPY OSSIMULA card is inserted into the program deck at a position where the omitted declarations would be placed otherwise.

Table of contents

CLOCK

function : elapsed time measurement (in hundredths of seconds).

declaration : external assembly integer procedure CLOCK

parameters : none

result : time of day in hundredths of seconds, i.e. the absolute difference between the results of two successive calls is the time which elapsed between the calls.

CPUTIME

function : measures CPU time usage.

declaration : external assembly long real procedure CPUTIME

parameters : none.

result : total CPU time spent since the beginning of the SIMULA program execution, expressed in hundredths of seconds.

DATE

function : provides date information.

declaration : external assembly text procedure DATE

parameters : none.

result : reference to a text object of length 11
whose contents are as follows:

nn␣mmm␣yyyy

where nn is day of the month
mmm are the first three letters of the name of the month
yyyy is the year,

these data refer to the current valid date.

SETTIME

function : schedules duration of program execution in real time.

declaration : external assembly procedure SETTIME

parameters : 1) first parameter must be of (short) integer type; its value is cpu-time in hundredths of seconds for which the program execution should continue after SETTIME call.

2) second parameter (optional) is a label from which the control will resume after the time period specified by the first parameter is exhausted. If this parameter is absent, the program will automatically terminate (with return code = 0); after the specified time period expires.

result : none.

notes :

1. procedure SETTIME works as a dummy procedure if not called from the outermost block level or if the value of the first parameter exceeds the remaining time interval set by RTS parameter TIME.
2. otherwise, there is no interference between the RTS TIME parameter, procedure CPUTIME and SETTIME.
3. a new call on SETTIME issued before the time period set by the previous call expired will override the previous time setting, providing that neither of the calls was rejected due to reasons mentioned in note 1.

TIMEOFDAY

function : access to system clock.

declaration : external assembly text procedure TIMEOFDAY

parameters : none.

result : reference to a text object of length 11 whose contents are as follows:

hh:mm:ss.cc

where hh is hours
mm is minutes
ss is seconds, and
cc is hundredths of seconds,

at the time of the call.

TIMER

function : elapsed time measurement (in timer units).

declaration : external assembly integer procedure TIMER

parameters : none

result : time (expressed in timer units, 1TU = 26.04166 microseconds) which elapsed since the first call on this procedure was made in the program (the first call consequently returns 0).

COPYTEXT

function : special purpose assembly routine to be used for creating copies of text objects when a (text type) external procedure is written in assembler or PL360.

calling sequence :

```
LM   0,2,TEXTDESCRIPTOR  of the text to be copied
L    15,=V(COPYTEXT)
BALR 14,15
USING *,15
```

result : the reference to the created text object is returned in the form of a text descriptor (see Programmer's Guide, Appendix F) in registers R0, R1 and R2.

ENVIRONMENTAL ENQUIRY ROUTINES

function : provide information about a particular implementation of the SIMULA system and thus enable even higher portability of SIMULA programs.

declarations : external assembly integer procedure BITSININ, BITSINRE, BITSINSI, BITSINLR, BITSINRF, BITSINCH, MAXSHORT, MAXINT, BITS

external assembly long real procedure SMALLREAL, MAXREAL, SMALLONG, MAXLONGR

parameters : none

results :

BITSININ	-	number of bits in an integer location
BITSINRE	-	number of bits in a real location
BITSINSI	-	number of bits in a short integer location
BITSINLR	-	number of bits in a long real location
BITSINRF	-	number of bits in a reference location
BITSINCH	-	number of bits in a character location
MAXSHORT	-	maximum short integer value
MAXINT	-	maximum integer value
BITS	-	free storage size in bits
SMALLREAL	-	smallest real magnitude
MAXREAL	-	maximum real magnitude
SMALLONG	-	smallest long real magnitude
MAXLONGR	-	maximum long real magnitude

Note: Every call on the procedure BITS forces a garbage collection prior to free storage checking.

ERROR

function : forces RT-error with controllable diagnostics message.

declaration : external assembly procedure ERROR

parameters : optional text parameter
(text value constant i.e. string is also accepted).

result : no value returned. The call causes a program interruption with return code = 8, accompanied by diagnostics: ZYQ104 and a message equal to the first 16 characters of the parameter (blanks are appended to the right if the parameter text value length is less than 16).
If no parameter was used, the message reads: FORCED ERROR.

note : This procedure may also be used from other than the SIMULA environment. This simplifies programming of error exits from non-SIMULA routines prepared to be used as external procedures by SIMULA programs.
The only requirements then, are that register R1 points to the message text, and R0 holds the routine return address leading back to a SIMULA program. The procedure ERROR recognises itself whether a call was made from a SIMULA program.

FILENAME

function : access to the external name of a data set.

declaration : external assembly text procedure FILENAME

parameter : one and only one parameter, which has to be a valid simple reference to an object of an I/O class.

result : reference to a text object whose value is equal to the parameter NAME of the FILE object. (cf. Common Base, 11.1.1, page 87).

note : the result text value is stripped of blanks before it is returned.

LINECNT

function : access to pagesize of a printfile.

declaration : external assembly [short] integer procedure LINECNT

parameters : one optional parameter, if present it has to be a reference to a printfile object; if absent, SYSOUT is substituted.

result : value of LINESPERPAGE attribute of the printfile which is otherwise inaccessible from the SIMULA environment.

MAXLOC

function : determines actual size (in records) of a directfile.

declaration : external assembly integer procedure MAXLOC

parameter : one and only one parameter, which has to be a valid simple reference to an open directfile object.

result : total number of records in the directfile.

PARM

function : access to EXEC PARM field.

declaration : external assembly text procedure PARM

parameters : none.

result : reference to a text object whose text value is equal to a parameter passed to the SIMULA program in the PARM field on the corresponding EXEC card.

note :

1. Respective parameters in the PARM field are separated by commas or equal signs. However, either of these delimiters may also be a part of the parameter, providing that they are. doubled. Example: if the following EXEC card is used

   ```
   //   EXEC   SIMCLG,PARM.GO='A,B=5,C==3'
   ```

   then the four successive calls on PARM will return the values "A", "B", "5", "C=3" respectively.

2. If no parameters are present in the parm field, or if PARM is called after the whole parameter string was scanned, the returned value is notext

3. If SIMULA RTS control parameters (such as DUMP, TRACE etc.) are used simultaneously, then the user parameters must follow these in the PARM field. The use of a slash to separate these two parameter strings is recommended. However, if this is omitted, the first parameter different from SIMULA RTS parameters signifies the beginning of the user parameter portion.

   Example:

   ```
   //  EXEC  SIMCLG,PARM.GO='DUMP=5/MYDUMP=10'
   ```

GET/PUT

function : un-edited (binary) input (GET) and output (PUT)

declaration : external assembly procedure GET
external assembly procedure PUT

parameters : the first parameter has to be a simple reference to an object of open directfile or infile for GET, or directfile or outfile for PUT. Remaining parameters (maximum 17) may be simple variables or array identifiers of any type but object reference; constants and labels are not allowed.

result : no functional value is returned; instead values of all but the first parameter are transferred between the core and the file referenced by the first parameter in the direction indicated by the procedure in use.

notes :

- total amount of information transferred depends on the type of respective parameters:

  4 bytes for integer and real
  2 bytes for short integer
  1 byte for Boolean or character
  8 bytes for long real

  or the length of the referenced text value for text type parameters.

- all but text arrays are checked for correct type, number of dimensions and total size on input, otherwise it is user responsibility to ensure that the sequence of receiving locations matches the input data pattern.

- the position indicator of the current image determines the starting location of the transferred chunk on the file. The next image is automatically used if the current one cannot accommodate the total amount of data passed.

GET/PUT cont.

It is however user responsibility to bring in and correctly position the first image for GET processing and to output the last image after a PUT call.

GETID

function : location of identifiers in a text or input file.

declaration : external assembly text procedure GETID

parameters : one optional parameter, either of type text or a reference to an object of infile or directfile. If absent, reference to sysin is assumed.

result : a reference to a text object whose value is equal to the identifier found in the text (input/directfile image) passed as parameter, or notext if the next item does not start with a letter.

notes :

1) The position indicator of the parameter (or "file".pos) is taken into account, i.e. the search for the identifier starts from the current position.

2) The blanks preceding the identifier (if any) are skipped.

3) The resulting position indicator setting of the parameter is that following the last letter (digit) of the identifier.

MIN (MAX)

function : returns the minimum value of the actual parameters.

declaration : external assembly <type> procedure MIN
<type> ::= integer|real|
short integer|long real

parameters : up to 18 parameters of type (short) integer, (long) real. Arrays are not allowed as parameters.

result : MIN returns the minimum value of the actual parameters as a <type>-value specified in the declaration.

Note : No fixed point overflow error is recognised if the magnitude of the converted real (result) value is greater than the maximum value of an integer. The latter is then used as the conversion result instead.

MAX follows the description of MIN except that the maximum value of the actual parameters is returned.

READINTO

function : procedure for free-format input to SIMULA programs.

declaration : external assembly procedure READINTO

parameters : up to 18 parameters of type (short) integer, (long) real, or character. Reference type parameter, if used, must be a simple reference to an object of class infile or directfile. Constants are not allowed, but a parameter may be an array identifier of suitable type.

result : new values for actual parameters are obtained one by one from input file (default = sysin). If a parameter is an array identifier, new values are read and assigned for all elements of the array, and the decomposition for multi-dimensional arrays is such that the first subscript varies more frequently than the second, etc. A parameter which is a reference to an open infile or directfile causes all successive read operations to be applied to this file.

WRITEOUT

function : free format output.

declaration: external assembly procedure WRITEOUT

parameters : up to 18 parameters of type (short) integer, (long) real, character or text. Constants and arrays are also allowed as parameters. Reference type parameter, if used, must be a simple reference to an object of class outfile or printfile or directfile.

result : values of actual parameters are output one by one on the attached output file (default = sysout). The output formats for respective values are the following:

integer i	outint(i,12)
real y	outreal(y,5,12)
short integer j	outint(j,6)
long real z	outreal(z,11,18)
character c	outchar(c)
text t	outtext(t)

If the actual parameter is an array identifier, the values of all its elements are output and the decomposition for multi-dimensional arrays is such that the first subscript varies more frequently than the second, etc. A parameter which is a reference to an open printfile, outfile or directfile, causes all successive output operations to be applied to this file.

Each call on WRITEOUT starts output on a new line and outimage is called implicitly if the line image is filled before all parameters are processed.

CODE, HEXDUMP

function : allows for insertion of machine code segments into SIMULA source programs, thus giving access to special machine features that are normally not accessible from a high level language.

Use of the routine may therefore often result in an obsolescence of external assembly procedures that are otherwise used in SIMULA programs for these purposes. If called as HEXDUMP, it will print unformatted hexadecimal dumps of either program or data areas, providing that the inserted code segment loads R0 and R1 with start address and the length (in bytes) of the requested area respectively.

declaration : external assembly procedure CODE
and
external assembly procedure HEXDUMP
respectively.

principles : the list of parameter addresses, the address of which is passed to the procedure by the administration routine ZYQFORT, is searched for literals. These are assumed to represent a bug-free segment of machine code instructions to which control is subsequently passed. Return is achieved by execution of the instruction:

```
BR   14
```

that is inserted by CODE at the end of the users supplied code.

The register situation on entering the inserted code segment is as follows:

R0 = 0
R1 = address of the list of parameter addresses
R2 = address of the location following the last user instruction
R8 = address of the first non-literal parameter address
R9 = same as R1
R10 = starting address of the inserted code
R11 = local display
R12 = current driver address
R14 = return address leading back to CODE
R15 = procedure CODE entry point

The general register save area is pointed to by R15 with a displacement of 24. Only the registers R3-R14 are restored on exit.

external routines : PL360 global procedure DUMP

data sets : SYSOUT - used optionally if a program/data area dump is requested (HEXDUMP entry).

output : an optional (HEXDUMP only) output is a hexadecimal dump of the indicated area that appears on SYSOUT. In addition, since registers R0-R2, as well as floating point registers, are not restored on exit, they may return a result of an arbitrary type according to 360/370 SIMULA standards (cf. PG Appendix F).

parameters : each call on CODE/HEXDUMP may be supplied with up to 18 actual parameters that have to comply with the following rules.

a) all literal parameters at the beginning of the list (conveniently expressed in hexadecimal) will be interpreted as a continuous stream of machine code instructions and must therefore

constitute an executable bug-free code segment. It is essential that, when interpreted numerically, these parameters will represent integer or real values within the permissible ranges (see PG 3.1/2).

b) Parameters other than those representing the machine code instructions have to be right-adjusted in the actual parameter list and the start of these is indicated by the first non-literal parameter.

side-effects : the only registers affected by CODE call are R0, R1, R2, R14 and R15 and eventually floating point registers.

sample calls : the calls below will have the following effect in turn:

- the general register dump will be printed on SYSOUT
- the dump of the object referenced by X will be printed on SYSOUT
- the program area lying between labels L1 and L2 will be dumped on SYSOUT
- the value of lines per page of a printfile object referenced by PRINT is returned as the call result in R0
- DDNAME of directfile object D is returned as result

```
HEXDUMP(#4100F018,#41100040);
HEXDUMP(#58209014,#58102000,#58101000;
        #48101100,#58002000,X);
HEXDUMP(#58108004,#58208000,#18021B12,L1,L2);
CODE(#58108000,#58101000,#48001022,PRINT);
CODE(#58109018,#58101000,#41101013,#5820901C,
     #58202000,#41000000,D,#80001);
```

remarks : it is assumed that a potential user of this routine will be familiar with Appendices F and G of the Programmer's Guide and the 360/370 Principles of Operation publication.

DEBUG

function : provides formatted hexadecimal dumps of relevant core areas under SIMULA object program execution, i.e. it may be conveniently used as a debugging tool for both 360/370 SIMULA programs and 360/370 SIMULA system.

declaration : external assembly procedure DEBUG

principles : the only genuine code of DEBUG takes care of the parameter retrieval and the overall control of the routine functioning. The main goal is accomplished by using the code already present in ZYQERR and ZYQSTORECOLLAPSE to which the control is passed from ZYQDEBUG.

external routines :

ZYQLNO
ZYQERR
ZYQSTORECOLLAPSE
PAGE
WRITE
ZYQCOM } external data segments
ZYQSTCDA }

data sets : SYSOUT - sequential output file used for output of the dump information

output : except for suppressed page effect, the output format is very similar to the optional dump obtained in the case of a run-time error. The amount of the output is controlled by the first actual parameter supplied at a call:

1st par. is greater or equal to :	output
any negative number	<identification line>
0	register dump
1	DEBUG call area dump
1	register area dump
1	displays
6	storage pool } all allocated core
6	notice pool } all allocated core
2	operating chain
3	SQS+LSC of all scheduled processes
4	LSC of all non-terminated objects
5	storage pool } referable structures only
5	notice pool } referable structures only
any negative number	<termination line>

The <identification line> and the <termination line> have the following formats respectively:

ZYQDEBUG CALLED AT CARD dddd [(copy of 2nd parameter)]
END OF DUMP AT CARD dddd

parameters : only the first two actual parameters supplied at a call are recognised and checked for correctness.

1st parameter, if present, has to be literal or a simple variable of type integer or short integer; Otherwise an error message is issued and control returns back to the calling program.

This parameter specifies which information is to be output according to the table above.

2nd parameter, should be a simple text value or a simple text variable. This parameter is optional and no error is recognised if it is completely missing or incorrectly supplied.

The text value referred by this parameter (if any) appears on the identification line enclosed in round brackets.

For the user's convenience, the following equivalence holds:

```
DEBUG; = DEBUG(0);
```

side-effects : a "false" garbage collection is forced if the value of the first parameter is greater than or equal to 2. The extra garbage collection does not affect further execution of the SIMULA program, neither is it recorded in the total number of storecollapses shown in the termination line of the program execution.

sample calls :

```
DEBUG; comment prints register dump only;

DEBUG(-1,"START OF CLASS CAR ACTION");
comment may be used for control flow tracing;

DEBUG(100); comment has the same effect as :-
DEBUG(6);
```

Addition No 1 to NCC Publication 56.0

SETCC

Function: setting of program return code.

Declaration: external assembly procedure SETCC

Parameters: one and only one parameter, which has to be a (short) integer variable/constant with a value in the range 0 - 4095.

Result: no value returned. The call causes a program termination with the return code equal to the parameter value.

Note: omitted parameter or a parameter of incorrect type causes the call to function as a dummy statement. The effect of the call with a parameter value greater than 4095 or negative is unpredictable.