Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	148903

TITLE

HELSCAM MODIFICATIONS

System Number:

Patron Number:

Requester:

Notes:

DSIS Use only:

Deliver to: FF

DEPARTMENT OF NATIONAL DEFENCE
CANADA

OPERATIONAL RESEARCH AND ANALYSIS

DIRECTORATE OF LAND OPERATIONAL RESEARCH

Research Note 94/7

HELSCAM Modifications

by

J.M. Thompson

November 1994

OTTAWA, CANADA

National Défense
Defence nationale

Operational Research and Analysis

CATEGORIES OF PUBLICATION

ORA Reports are the most authoritative and most carefully considered publications issued. They normally embody the results of major research activities or are significant works of lasting value or provide a comprehensive view on major defence research initiatives. ORA Reports are approved by DGOR and are subject to peer review.

ORA Project Reports record the analysis and results of studies conducted for specific sponsors. This category is the main vehicle to report completed research to the sponsors and may also describe a significant milestone in ongoing work. They are approved by DGOR and are subject to peer review. They are released initially to sponsors and may, with sponsor approval, be released to other agencies having an interest in the material.

Directorate Research Notes are issued by directorates. They are intended to outline, develop or document proposals, ideas, analysis or models which do not warrant more formal publication. They may record development work done in support of sponsored projects which could be applied elsewhere in the future. As such they help serve as the corporate scientific memory of the directorate.

Department of National Defence

Canada

Operational Research and Analysis

Directorate of Land Operational Research

Research Note 94/7

HELSCAM Modifications

by

J.M. Thompson

Ottawa, Canada November 1994

Abstract

This paper documents some changes that were made to the FORTRAN code of HELSCAM. The HELSCAM runs for the Future Air Defence Requirements project involved complicated scenarios which required a tempest computer with a fast floating-point processor. Thus the initial changes were instituted to enable HELSCAM to run on a Sun workstation under the UNIX operating system. When the actual simulations were run some bugs in the code were discovered and rectified. This research note also discusses the future of HELSCAM.

Résumé

Ce rapport documente des changements qui ont été faits au code FORTRAN du programme HELSCAM. Les passage-machines de HELSCAM faits pour le project sur les besoins futurs de la défence aérienne comportaient des scénarios complexes qui demandaient un ordinateur "tempest" avec processeur à virgule flottante rapide. Les changements initiaux furent institués pour permettre à HELSCAM de fonctionner sur une station de travail Sun avec système d'exploitation UNIX. Durant les simulations, on découvrit des bogues dans le programme. Ce rapport discute aussi du futur de HELSCAM.

Table of Contents

Pages

List of Tables

Pages

Introduction

1. The HElicopter SCenario Assessment Model (HELSCAM) was originally developed by the Directorate of Land Operational Research (DLOR) as a tool to assist in a study of helicopter operations. Since the original model was developed it has undergone a number of revisions. These changes are documented in Ref. 1. When HELSCAM was employed for the Future Air Defence Requirements Study (Ref. 2) several further modifications took place. Firstly the HELSCAM simulation core was ported to a Sun workstation, and subsequently, because the program was used with very different scenarios than it had been previously, several bugs were revealed in the code and some limitations of the program became apparent. HELSCAM can be a very useful tool, but it has not yet been fully tested and debugged. No simulation model can handle all possible scenarios, or take into account future developments. Thus HELSCAM must first be fully tested, and then there must be enhancements made to the simulation core.

2. This paper is intended to document the changes that have been made to the HELSCAM core, and to suggest areas where further development should take place.

Porting HELSCAM to Run Under UNIX

3. There were very few difficulties encountered in compiling HELSCAM with Sun's FORTRAN compiler. In the translation of the code to run on the Sun one objective was to make the code as close to being standard ANSI FORTRAN as was possible. Any code adhering to the ASCII standards for FORTRAN can be ported from one platform to another with relative ease.

4. The first step in compiling HELSCAM on the Sun was to create a Makefile to compile and link HELSCAM. A Makefile describes explicitly how to build each module of a program as well as the final executable. The Makefile shows the object files and source files as well as the interdependencies between them. The Makefile uses the list of source and object files to compile each section and then links them to form HELSCAM. The Makefile is implemented with a *make* command which performs only the steps that are required to bring HELSCAM up to date. The list of dependencies is used so that if one of the source codes is changed only the other sections dependent on the changed source are updated. The Makefile is appended at Annex A.

5. The major changes required to compile HELSCAM were the replacement of function calls which are not standard UNIX FORTRAN functions, and changes to the OPEN statements.

6. The PC random number generating function *RND()* was replaced by the UNIX function *rand()*. This function appears in: *acq.for; comm.for; dam_ info.for; rap.for;* and *simulati.for.* The function *rand()* returns real values in the range 0.0 through 1.0. There are other random number generators available on the Sun which are theoretically better, but *rand()* is widely available on UNIX systems and thus better for portability. Consideration could be given to including a subroutine to generate random numbers within HELSCAM. This would avoid the dependency on a system function. The problem would be supplying a seed to start the random sequence since it is not always desirable to start with the same seed.

7. For $r = rand(k)$, if $k = 0$ then r is the next number in the sequence; if $k = 1$ the sequence is restarted and the first number returned, and if $k > 0$ then k is used as a seed for a new sequence, returning the first number.

8. The initialization of the sequence of random numbers occurs in *simulati.for.* In all other subroutines *rand(0)* is used. The seed for the random number sequence is taken from the computer clock using *ltime()* which returns integer values for hour, minute and second.

9. In *manager.for, rap.for, and track.for* the VMS specific function *JNINT* was replaced by the UNIX generic function *NINT.* This function is the nearest integer function:

$$NINT(a) = int(a + [0.5 * sign(a)])$$

10. In *get_ bdb.for* and *get_ rap.for* the OPEN statements have been changed simply to reflect the different file systems used in DOS and UNIX. In *get_ terr.for* the method of accessing the file had to be changed. The piece of code below is the version of *get_ terr.for* for the LAHEY compiler under DOS:

```
byte = 1

map_file = 'c:\helscam\mapdata\adatsmap.rdc'

open(unit=TERRAIN_FILE,access='TRANSPARENT',form='UNFORMATTED',
   &   status='OLD',file=map_file)


do i=1,map_MAX_EASTING
```

```
      read(TERRAIN_FILE,REC=byte) (ie2(j),iv2(j),j=1,273)

C       Number of bytes in each record is 273 * 3 = 819.
C       Therefore increase record counter "byte" by 819.
      byte = byte + 819
```

11. The following code is from the UNIX version:

```
  byte = 1

  map_file = '/home/dlor/jthompso/helscam/adatsmap.rdc'

  open(unit=TERRAIN_FILE,access='DIRECT',RECL=819,form='UNFORMATTED',
     &    status='OLD',file=map_file)

  do i=1,map_MAX_EASTING

    read(TERRAIN_FILE,REC=byte) (ie2(j),iv2(j),j=1,273)

    byte = byte +  1
```

12. The OPEN statement in the DOS version has " access='TRANSPARENT' "; this is not a standard FORTRAN construct. Transparent denotes a random access, formatted or unformatted file with a record length of 1 byte and no header. Transparent files can behave like either formatted or unformatted files. In this case the file is accessed sequentially one record of length 819 bytes at a time. The UNIX code uses a standard DIRECT access with a record length of 819 bytes. In the DOS version the variable *byte* is incremented by 819 to indicate the start position of the next record whereas, since the record length is specified in the UNIX version, *byte* is incremented by one to indicate the next record to access. The difference in the file structures between the two operating systems can be seen in the statements where the character string *map_ file* is assigned a value.

13. Reading the terrain data, which is binary data, posed additional problems which required more than just changing the open statement. The order in which data - the bits and bytes - are arranged differ between VAX and Intel Computers on one hand and SPARC computers on the other. The bytes in a 32-bit integer, when read from address n, end up in the register as shown in Tables I and II. This difference is only apparent when trying to read binary data created on one architecture on a different architecture.

Table I

Bits and Bytes for Intel and VAX Computers

Byte n+3	Byte n+2	Byte n+1	Byte n
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00

Most significant Least significant

Table II

Bits and Bytes for 680x0 and SPARC Computers

Byte n	Byte n+1	Byte n+2	Byte n+3
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00

Most significant Least significant

14. There are filters which are available to correct the byte order on binary files It is also possible to write the binary file as an ASCII file on the source architecture, read it as an ASCII file on the destination architecture and then write it again as a binary file. This latter solution was used to correct the byte order on the terrain data.

15. In the original version of HELSCAM the binary data was input as integer*1. Integer*1 is a non-standard type definition which was replaced by either logical*1 or byte, which are synonymous, and are standard ANSI constructs. These occur in *environ.inc* and *get_ terr.for.*

16. The final change to make the code more standard was trivial. In *parameter.inc* and *rap.inc* there were more continuation statements than allowed by ANSI FORTRAN; these were broken up into separate statements.

17. The increased performance of current computers has allowed some modifications in the code for beam rider missiles. The VAX on which HELSCAM was originally run was very slow, and this had prompted the use of a variable step size for the Runge-Kutta algorithm used. The step size was increased in some circumstances, reducing the amount of computation required, but decreasing the accuracy of the result. The step size has now been made constant. The computer limitations also resulted in the use of a two dimensional algorithm for beam rider missiles. The beam rider algorithm has now been extended to three dimensions.

18. Another benefit of using a workstation to develop and debug software was revealed in the process of porting HELSCAM to the Sun. The FORTRAN compiler on the Sun is better than those previously employed to compile HELSCAM, and the floating-point environment (hardware, system software, and software libraries) on the Sun is an implementation of the IEEE Standard 754. This standard specifies:

- Hardware storage formats for IEEE double precision (64 bits) and IEEE single precision (32 bits).
- Accuracy requirements on basic floating-point operations: *add, subtract, multiply, divide, square root, remainder,* and *compare.* (These operations may not suffer more than one rounding error.)
- Accuracy requirements for conversions between formats.
- Five types of IEEE floating-point exceptions and when exceptions of these types should be indicated to the user. (The five types of floating-point exceptions are *invalid operation, division by zero, overflow, underflow,* and *inexact.*)
- Four rounding modes: These are round towards the nearest representable value, round towards zero, round towards $+\inf$, and round towards $-\inf$.
- Rounding-precision: For example, if a computer delivers results only in extended precision format, the user should nonetheless be able to specify that the result is to be rounded to a single precision with trailing zeros.

19. The Standard allows greater control over computation. It also supports exception handling, interval arithmetic, and diagnosis of anomalies. The standard also insures that results will be bitwise identical on any machines adhering to the standard regardless of architecture. For a description of these standards, and a discussion of other issues relating to numerical computations see Ref. 3. For a very detailed examination of floating point arithmetic and the IEEE standards Ref. 4 is highly recommended.

20. When HELSCAM was ported to the Sun, simulation runs of a scenario were done on both a PC and the Sun, for the purpose of comparison. After the runs on the Sun the following message was displayed:

```
Note: the following IEEE floating-point arithmetic exceptions
occurred and were never cleared; see ieee_flags(3M):
Operand; Inexact;  Underflow;
Sun's implementation of IEEE arithmetic is discussed in
the Numerical Computation Guide.
```

21. The *inexact* and *underflow* exceptions are not uncommon nor are they usually cause for alarm, but the *operand* which signifies an illegal operand was a problem. One of the weakest parts of the FORTRAN language is its inability to deal with exceptions. Thus to locate the source of this problem exception handling routines in the 'C' language were included in HELSCAM. These routines halt execution of the program at the point where a specified exception occurs. When the program is halted it is possible to check the current values of variables and to execute the code one step at a time. For a discussion of exceptions and exception handling see Ref. 3 or Ref. 4.

22. By using the exception handling capabilities of the Sun some limitations which existed within the HELSCAM simulation code were traced, and corrected. This highlights the benefits of using a platform which adheres to the IEEE standards and shows that the process of porting HELSCAM to the Sun was worth the effort.

The Future

23. HELSCAM can be a useful tool but there are steps which should be taken to ensure its continued use. First the code must be cleaned up and more fully documented; next the model must be fully validated and verified so that the user has confidence in the results which are produced; there are some areas which need to be updated; and finally some new capabilities need to be added to the model.

24. The combination of a lack of sufficient internal documentation and some of the programming style used, generally make the simulation core difficult to analyse and debug. In particular the use of *includes* and *common blocks* instead of formal parameter passing makes it difficult to ensure that no side effects occur, or to trace all places where variables are assigned values.

25. There are very few places in the simulation core of HELSCAM where any error checking has been implemented. For example it is possible to put in a flight path for a fixed wing aircraft which has the aircraft flying through terrain. This was easily restricted by checking to make sure that when the aircraft's altitude is calculated it is never zero or less. This type of error checking makes a program easier to use properly since both bugs and problems with the inputs which describe the systems and the scenarios are found more easily.

26. It would be unproductive to go through all of the code for HELSCAM making changes to the documentation and programming style or to add extensive documentation. If, however, some parts of the code are modified then error checking and documentation should be improved in those sections.

27. Verification is the process of determining that a model accurately represents the developer's concept and specifications. This process involves identifying and examining the assumptions underlying the model, examining the inputs and data bases within the model, and ensuring that the code performs all calculations correctly and in the intended fashion. There are many approaches to verification, the most common being structured walk-through techniques to determine if the model logic correctly performs intended functions, and sensitivity analyses. Unexpected sensitivity (or lack of sensitivity) may highlight a problem. (Ref. 5)

28. Validation is the process of determining whether or not the model accurately represents the real world, from the point of view of the intended use of the model. Validation includes each module as well as the model as a whole.

29. There was definitely some effort put into the verification of HELSCAM when it was first written. Unfortunately this work was not documented. Since an incorrect subscript in the code modelling the beam rider missile was found, it is clear that HELSCAM has not been fully tested. It is possible that there are other sections of code which do not perform their function correctly. For HELSCAM to be accepted and used in more studies, some effort must be expended to test each section of the code. One possible way to approach this would be to start by running simple scenarios which would use different sections of the code and then to make the scenarios progressively more complex.

30. At the same time the input parameters for the scenarios could be varied to perform sensitivity analysis and to see if there are any ranges of values where the code breaks down. This analysis would be very helpful to users since it can identify how sensitive the model outputs are to small variations in the input parameters. If the model is functioning as intended and outputs change dramatically with a small change to the value of a particular input, then great care must be taken to ensure the accuracy of that input.

31. It is not sufficient to verify that the code is correct; rather the algorithm from which the code is derived, and in some cases the data-bases used must also be checked. For example, one area which needs to be updated is the routines to model target acquisition (Ref. 6). The data used in these algorithms date back to 1989, and some of the values used were generated by an early version of the British Aerospace model, Oracle. There are four types of sensors represented in HELSCAM: eyeball; binocular; image intensifier; and thermal imager. Each system has been characterized by one particular sensor, i.e. the thermal imager is represented by the LR Trigat. The data base and systems represented must be updated and enlarged, especially for any study comparing the effects of using different target acquisition systems.

32. The beam rider missile algorithms are also somewhat of a problem. In simulations of the ADATS missile the miss distances were often unrealistic. This might be the result of some of the parameters used in the model being incorrect; however, this is typical of the types of problems which need to be investigated.

33. Some of the algorithms which were included in HELSCAM were very appropriate for the studies for which the model was originally intended, but are not adequate for other applications. For example the representation of C^3I within HELSCAM is too simplistic for many scenarios and should be replaced.

34. One of the concerns about any model like HELSCAM is that a certain set of parameters are required to describe a system within the model. Often the required parameters are not available, leaving the user uncertain as to the best way to deal with the situation. Thus, if an algorithm is very detailed, the required inputs may not be available. If the algorithm is too simple then the inputs may be readily available but the results may be too crude. There is no easy solution to this problem but if any changes are made in HELSCAM this concern should be kept in mind.

35. The previous discussion has centred on verification. Validation is a different problem. Some possible approaches in this area involve: checking results to see if they are reasonable; comparison with historical data; comparison with results from other models; or ideally where possible comparison with test data, engineering trials, field data or operational data.

36. The documentation, verification and validation as described above would take a great deal of time and effort. However when a problem is discovered while actually using a model it can be very time consuming (and frustrating) to "fix" the problem, and often the solution is just a patch. There is a steep learning curve in order for someone to reach the point where they could successfully modify or debug HELSCAM. A concerted effort to clear up all the problems at once would most likely actually save resources, and would result in enough "corporate" knowledge of HELSCAM to enable future modifications to be done at much less expense. It would also be beneficial to document all of the work which is done to verify and validate the model, and to document all of the modifications so that potential users would know what had been done.

37. There are some areas which have not yet been modelled within HELSCAM which will need to be included to make the simulation more versatile. There are no capabilities for HELSCAM to model missiles with seeker heads. There is presently no way to represent guided weapons fired from an air platform, or to show the effect on a guided weapon if the platform guiding it is destroyed.

38. This discussion has, so far, avoided any mention of COMSCAM. As a final comment it should be noted that some of the issues addressed above, such as documentation and parameter passing, may have been dealt with in COMSCAM. If the simulation core of COMSCAM can be separated from the graphical user interface of COMSCAM, and new input and output routines added, then it might be preferable to use the C++ code of COMSCAM in preference to the FORTRAN code of HELSCAM. It would seem impractical to pursue development of both models, but careful consideration must be given to each model before a decision is made.

Conclusion

39. Very little difficulty was encountered in porting HELSCAM to run under UNIX on a Sun Workstation. The major flaw discovered in HELSCAM was an example of a lack of consideration of the effects of floating point computations on mathematical calculations.

40. HELSCAM can prove to be a very useful tool for DLOR, if some time and effort are expended to fully debug, test, and update the simulation core of this model.

LIST OF REFERENCES

(1) Young, Dr. P.J., **HELSCAM V2.0 Development and Application Guide**, DLOR Research Note 92/6, December 1992.

(2) Thompson, J.M & Bowles, Maj A.B., **Future Air Defence Requirements**, ORA Project Report PR 681, August 1994. **SECRET**

(3) Sun Microsystems, Inc, **Numerical Computation Guide**, Mountainview CA, 1992.

(4) Goldberg, David, "What Every Computer Scientist Should Know About Floating-Point Arithmetic", **ACM Computing Surveys**, Vol. 23, No. 1, March 1991.

(5) Headquarters, Department of the Army, Washington, DC, **Army Model and Simulation Management Program**, Army Regulation 5-11, 10 June 1992.

(6) Thibault, Gaetan, Smith, Pete, & Chouinard Paul, **The HELSCAM Target Acquisition Algorithm**, DLOR Staff Note 89/5, July 1989.

Annex A HELSCAM's Makefile for the SUN

```
FFLAGS   = -O
OBJS = acq_info.o acquire.o altitude.o ammo.o bdb.o \
  beam_rk.o  beam_rk1.o col_info.o comm.o coor.o \
  dam_info.o damage.o end_sim.o fir_info.o \
  get_bdb.o get_rap.o get_terr.o get_var.o gun.o \
  initial.o loc_info.o location.o los.o manager.o \
  movement.o msl.o orientat.o p_area.o path.o \
  prop_rk.o prop_rk1.o range.o rap.o selectio.o \
  simadmin.o simulati.o track.o velocity.o \
  weapon.o

SRCS = acq_info.for acquire.for altitude.for ammo.for bdb.for \
  beam_rk.for beam_rk1.for col_info.for comm.for coor.for \
  dam_info.for damage.for end_sim.for fir_info.for \
  get_bdb.for get_rap.for get_terr.for get_var.for gun.for \
  initial.for loc_info.for location.for los.for manager.for \
  movement.for msl.for orientat.for p_area.for path.for \
  prop_rk.for prop_rk1.for range.for rap.for selectio.for \
  simadmin.for simulati.for track.for velocity.for \
                  weapon.for


acq_info.o: acq_info.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c acq_info.for

acquire.o: acquire.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c acquire.for

altitude.o: altitude.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c altitude.for

ammo.o: ammo.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c ammo.for

bdb.o: bdb.for admin.inc bdb.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c bdb.for
```

```
beam_rk.o: beam_rk.for admin.inc bdb.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c beam_rk.for

beam_rk1.o: beam_rk1.for
f77  $(FFLAGS) -c beam_rk1.for

col_info.o: col_info.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c col_info.for

comm.o: comm.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c comm.for

coor.o: coor.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS)  -c coor.for

dam_info.o: dam_info.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c dam_info.for

damage.o: damage.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c damage.for
end_sim.o: end_sim.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c end_sim.for

fir_info.o: fir_info.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c fir_info.for

get_bdb.o: get_bdb.for  bdb.inc paramete.inc
f77  $(FFLAGS) -c get_bdb.for

get_rap.o: get_rap.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc rap.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c get_rap.for

get_terr.o: get_terr.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c get_terr.for

get_var.o: get_var.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c get_var.for
```

```
gun.o: gun.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c gun.for

initial.o: initial.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c initial.for

loc_info.o: loc_info.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c loc_info.for

location.o: location.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c location.for

los.o: los.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c los.for

manager.o: manager.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS)  -c manager.for

movement.o: movement.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS)  -c movement.for

msl.o: msl.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c msl.for

orientat.o: orientat.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c orientat.for

p_area.o: p_area.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c p_area.for

path.o: path.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS)  -c path.for

prop_rk.o : prop_rk.for
f77  $(FFLAGS)  -c prop_rk.for

prop_rk1.o : prop_rk1.for
f77  $(FFLAGS)  -c prop_rk1.for
```

```
range.o: range.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c range.for

rap.o: rap.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc rap.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c rap.for

selectio.o: selectio.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c selectio.for

simadmin.o: simadmin.for
f77  $(FFLAGS) -c simadmin.for

simulati.o: simulati.for admin.inc bdb.inc common.inc environ.inc gensim.inc \
 paramete.inc policy.inc process.inc rap.inc scenario.inc state.inc \
 system.inc utility.inc
f77  $(FFLAGS) -c simulati.for

track.o: track.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c track.for

velocity.o: velocity.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77  $(FFLAGS) -c velocity.for

weapon.o: weapon.for admin.inc common.inc environ.inc gensim.inc paramete.inc \
 policy.inc process.inc scenario.inc state.inc system.inc \
 utility.inc
f77 $(FFLAGS) -c weapon.for

helscam: $(OBJS)
f77 $(FFLAGS) $(OBJS) -o helscam
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared e.g. Establishment Sponsoring a contractor's report, or tasking agency, are entered in Section 8).

DLOR
National Defence Headquarters
Ottawa, Ontario, K1A 0K2

2. SECURITY CLASSIFICATION
(overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title)

HELSCAM Modifications (U)

4. AUTHORS (last name, first name, middle initial)

Thompson, John M.

5. DATE OF PUBLICATION (month Year of Publication of document)

November 1994

6a. NO OF PAGES (total containing information. Include Annexes, Appendices, etc.)

19

6b. NO OF REFS (total cited in document)

6

7. DESCRIPTIVE NOTES (the category of document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

DLOR Research Note

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address).

DLOR

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

N/A

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written.)

N/A

10a. ORIGINATOR's document number (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

N/A

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

N/A

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors: further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

None

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the test is bilingual).

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified . If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

HELSCAM
IEEE Floating Point Standards

148903

NO. OF COPIES NOMBRE DE COPIES	COPY NO. COPIE N°	INFORMATION SCIENTIST'S INITIALS INITIALES DE L'AGENT D'INFORMATION SCIENTIFIQUE
1	1	JL
AQUISITION ROUTE ▶ FOURNI PAR	ORA	
DATE ▶	09 Jan 95	
DSIS ACCESSION NO. ▶ NUMÉRO DSIS		

DND 1158 (6-87)

National Defence / Défense nationale

PLEASE RETURN THIS DOCUMENT TO THE FOLLOWING ADDRESS:

DIRECTOR
SCIENTIFIC INFORMATION SERVICES
NATIONAL DEFENCE
HEADQUARTERS
OTTAWA, ONT. - CANADA K1A 0K2

PRIÈRE DE RETOURNER CE DOCUMENT À L'ADRESSE SUIVANTE:

DIRECTEUR
SERVICES D'INFORMATION SCIENTIFIQUES
QUARTIER GÉNÉRAL
DE LA DÉFENSE NATIONALE
OTTAWA, ONT. - CANADA K1A 0K2