SOFTWARE RELEASE NOTICE

01. SRN Number: RDCO-SRN-104

02. Project Title: FISSP-CLOUD - Fission Product Inventory, Release, Transport, and Dose Calculation	Project No. 20-5702-622
03. SRN Title: FISSP-CLOUD	
04. Originator/Requestor: Budhi Sagar	Date: 01/22/96

05. Summary of Actions

- ☐ Release of new software
- ☐ Release of modified software:
  - ☐ Enhancements made
  - ☐ Corrections made
- ☐ Change of access software
- ■ Software Retirement

06. Persons Authorized Access

Name	RO/RW	A/C/D
N/A		

07. Element Manager Approval: [signature] Date: 1/30/96

08. Remarks:

Not considered important to regulatory reviews in revised FY96 OPS Plans.

SOFTWARE SUMMARY FORM

01.Summary Date: 06/28/94	02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083	03. Summary Action: New
04. Software Date: 8/28/94	05. Short Title: FISSP-CLOUD	
06. Software Title: FISSP-CLOUD - Fission Product Inventory, Release, Transport, and Dose Calculation.		07. Internal Software ID: NONE

08. Software Type:	09.Processing Mode:	10. APPLICATION AREA
☐ Automated Data System	☐ Interactive	A. General:
■ Computer Program	☐ Batch	☐ Scientific/Engineering ☐ Auxiliary Analyses
☐ Subroutine/Module	■ Combination	☐ Total System PA
		☐ Subsystem PA ■ Other
		b. Specific:

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA, SwRI, San Antonio, Texas	H. Karimi, (210) 522-5253

13. Narrative:

FISSP-CLOUD - A system of linked codes. FISSP calculates the fission product inventory in a U^{235} fueled reactor for a specified power level. In CLOUD, the released nuclides are allowed to drift and deffuse in three dimensions as determined by the appropriate coefficients in the Sutton diffusion equation.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	19,244 lines of code
18. Computer Memory Requirements: UNKNOWN	19. Tape Drives: NONE	20. Disk/Drum Units: N/A	21. Graphics: UNKNOWN

22. Other Operational Requirements

NONE

23. Software Availability:	24. Documentation Availability:
■ Available ☐ Limited ☐ In-House ONLY	■ Available ☐ Inadequate ☐ In-House ONLY

25. Submission Package Status:

Acceptance Criteria: Met ■ Not Met ☐ Software QA Assessment: Successful ■ Unsuccessful ☐

Code Custodian:_____ Date:_____

CLOUD CRAY LISTING

```
-rw-r-----   1 tjr1     tjr1         1207 Jun 24 14:03 Makefile
-rw-r-----   1 tjr1     tjr1        22384 Jun 24 14:03 cloud.F
-rw-r-----   1 tjr1     tjr1        54448 Jun 24 14:03 cloud.SRC
-rw-r-----   1 tjr1     tjr1            0 Jun 28 13:04 cloud.dir
-rw-r-----   1 tjr1     tjr1        22384 Jun 24 14:03 cloud.src
-rw-r-----   1 tjr1     tjr1        32562 Jun 24 14:03 prob.in
-rw-r-----   1 tjr1     tjr1          275 Jun 24 14:03 x.cloud.covr
-rw-r-----   1 tjr1     tjr1          259 Jun 24 14:03 x.cloud.test
```

6/28/94 TJR

FISSP CRAY LISTING

```
-rw-r-----   1 tjr1     tjr1         1207 Jun 27 12:36 Makefile
-rw-r-----   1 tjr1     tjr1        35316 Jun 27 12:36 f01
-rw-r-----   1 tjr1     tjr1          324 Jun 27 12:36 f02
-rw-r-----   1 tjr1     tjr1        85779 Jun 27 12:36 f03
-rw-r-----   1 tjr1     tjr1        54432 Jun 27 12:36 f04
-rw-r-----   1 tjr1     tjr1          324 Jun 27 12:36 f05
-rw-r-----   1 tjr1     tjr1        32562 Jun 27 12:36 f06
-rw-r-----   1 tjr1     tjr1         5265 Jun 27 12:36 f07
-rw-r-----   1 tjr1     tjr1      1427904 Jun 27 12:36 f08
-rw-r-----   1 tjr1     tjr1       273226 Jun 27 12:36 f09
-rw-r-----   1 tjr1     tjr1        21250 Jun 27 12:36 fissp.F
-rw-r-----   1 tjr1     tjr1        35144 Jun 27 12:36 fissp.SRC
-rw-r-----   1 tjr1     tjr1            0 Jun 28 13:08 fissp.dir
-rw-r-----   1 tjr1     tjr1        21250 Jun 27 12:36 fissp.src
```

6/28/94 TJR

FISSP Fortran Program Static and Dynamic Analysis

DRAFT

June 6, 1994

Earl S. Marwil
John E. Tolli
Scientific Computing Unit
Idaho National Engineering Laboratory

1. Introduction

This analysis was performed on the Cray version of the software as converted from an IBM/PC version provided by Southwest Research Institute (SwRI).

One sample problem was used along with the source code. The program was analyzed using the Craft (Cross Reference Analysis of Fortran) tool, FORWARN, the Fortran 77 analyzer, and PC-Metric. These tools provide static analysis, coverage analysis, and complexity analysis.

2. References

[1] N.H. Marshall and E.S. Marwil, Cross Reference Analysis of Fortran (CRAFT), EG&G-CATT-9198, EG&G Idaho, Inc., July 1991.
[2] Fortran 77 Analyzer User's Manual, National Bureau of Standards, NBS GCR 81-359, 1981
[3] FORWARN User's Guide, Quibus Enterprises, Inc., July 1991.
[4] PC-Metric User's Guide, SET Laboratories, Inc., 1987.

3. Functions

The FISSP program contains 1 Fortran routine.

4. Common Block Irregularities

There are no common blocks in the FISSP program.

5. Interface Irregularities

Not applicable.

6. Local Variable Irregularities

No exceptions to report.

7. Fortran Extensions

The FISSP program statement has arguments.

There are "REAL*n" statements in the program.

8. Optimization

The following table summarizes the performance data gathered from execution of the sample problem. Only those routines exercised by the sample problem are shown (see "Coverage Analysis" for a list of routines not exercised by the sample problem, i.e., coverage = 0%). The table lists all program modules in descending order according to CPU time. To optimize code execution time, emphasis should be placed on those modules which appear highest in the listing.

As the performance data show, there is a low percentage of floating point operations which are performed in vector mode (%Vflops is small).

A detailed optimization analysis effort should focus on this area.

PERFORMANCE DATA FOR FISSP

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
FISSP	12.032	100.000	100.000	57.17473	0.00	0.818	0.807
Totals (All Traced Routines)	12.032	100.000	100.000	57.17473	0.00	0.818	0.807

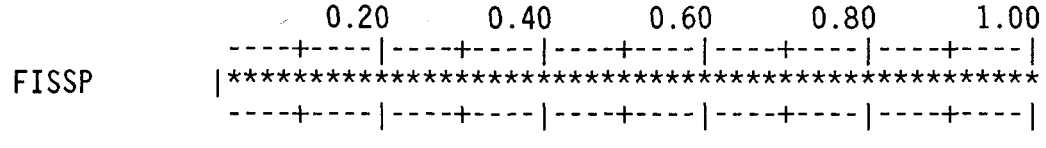
Key:

- %AccumT = accumulated percentage of total CPU time
- %ExTime = percentage of total CPU time
- %Vflops = percentage of floating point operations due to vector floating point operations
- IBFR = Instruction Buffer Fetch Rate (megafetches/sec)
- IFact = Inline Factor (total calls to routine / average time spent in routine for each call)
- MC = number of memory conflicts
- MR = number of memory references
- Time = total CPU time (sec)

9. Coverage Analysis

A coverage analysis shows that the sample problem yielded a 87% segment coverage of FISSP. Sample problems provided with simulation programs typically achieve only 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
FISSP	193	167	86.5
Totals	193	167	86.5



```
0.85 <= coverage < 0.90      FISSP

  Program coverage for this run =0.87
```

10. Complexity Analysis

Some key metrics are the number of executable statements (sloc), the number of non-blank comments (ncomt), McCabe's extended cyclomatic complexity (vg2), the number of branching statements (cgoto, ugoto, bIF, and lIF), and Halstead's predicted number of errors in (re)writing the code (bhat). Measures are normalized per 100 executable statements for ease of comparison and are listed in the table below.

The branching measures for this code (ugoto/sloc, lif/sloc) indicate moderate/low values. This code appears to be fairly well structured.

The program shows a fair ratio of non-blank comments to source code.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates a high value. Generally, the routines with the highest complexity are those most likely to have defects. As a guideline, normalized measures of 15 or greater should be considered complex. A software maintenance program should focus on those routines with the highest measures.

Complexity Report by Subprogram for FISSP

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
FISSP	436	234	170	142	60.7	25.6	0	0.0	22	9.4	0	0.0	8	3.4	5

Legend of Metrics in Report

loc -- lines of code
sloc -- number of executable statements
cmnt -- total number of commnts
ncomt -- number of non-blank COMMENT statements
100*ncomt/sloc -- percent, nonblank comments to number of executable statements
100*vg2/sloc -- percent, extended complexity of number of executable statements
cgoto -- number of COMPUTED GO TO statements
100*cgoto/sloc -- percent, computed GOTO's to number of executable statements
ugoto -- number of UNCONDITIONAL GO TO statements
100*ugoto/sloc -- percent, unconditional GOTO's to number of executable statements
bIF -- number of BLOCK IF statements
100*bif/sloc -- percent, Block IF statements to number of executable statements
lIF -- number of LOGICAL IF statements
100*lif/sloc -- percent, logical IF statements to number of executable statements
Bhat -- Halstead's predicted number of errors in writing code

CLOUD Fortran Program Static and Dynamic Analysis

DRAFT

June 7, 1994

Earl S. Marwil
John E. Tolli
Scientific Computing Unit
Idaho National Engineering Laboratory

1. Introduction

This analysis was performed on the Cray version of the software converted from an IBM/PC version as provided by Southwest Research Institute (SwRI).

One sample problem was used along with the source code. The program was analyzed using the Craft (Cross Reference Analysis of Fortran) tool, FORWARN, the Fortran 77 analyzer, and PC-Metric. These tools provide static analysis, coverage analysis, and complexity analysis.

2. References

[1] N.H. Marshall and E.S. Marwil, Cross Reference Analysis of Fortran (CRAFT), EG&G-CATT-9198, EG&G Idaho, Inc., July 1991.
[2] Fortran 77 Analyzer User's Manual, National Bureau of Standards, NBS GCR 81-359, 1981
[3] FORWARN User's Guide, Quibus Enterprises, Inc., July 1991.
[4] PC-Metric User's Guide, SET Laboratories, Inc., 1987.

3. Functions

The CLOUD program contains 1 Fortran routine.

4. Common Block Irregularities

There are no common blocks in the CLOUD program.

5. Interface Irregularities

Not applicable.

6. Local Variable Irregularities

Local variable exceptions are noted as follows:

Module	Variable	Exception
cloud	term2	Defined, Unused

7. Fortran Extensions

The CLOUD program statement has arguments.

There are "REAL*n" statements in the program.

8. Optimization

The following table summarizes the performance data gathered from execution of the sample problem. Only those routines exercised by the sample problem are shown (see "Coverage Analysis" for a list of routines not exercised by the sample problem, i.e., coverage = 0%). The table lists all program modules in descending order according to CPU time. To optimize code execution time, emphasis should be placed on those modules which appear highest in the listing.

In order to obtain meaningful statistics for performance evaluation, the program should execute for a reasonable amount of time. Note that the execution time for this sample problem is short (< 10 sec) and that the resulting statistics may therefore not accurately reflect program performance for more typical (possibly longer) runs.

As the performance data show, there is a low percentage of floating point operations which are performed in vector mode (%Vflops is small).

A detailed optimization analysis effort should focus on this area.

PERFORMANCE DATA FOR CLOUD

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
CLOUD	7.596	100.000	100.000	1.18914	0.00	0.201	0.609
Totals (All Traced Routines)	7.596	100.000	100.000	1.18914	0.00	0.201	0.609

Key:

- %AccumT = accumulated percentage of total CPU time
- %ExTime = percentage of total CPU time
- %Vflops = percentage of floating point operations due to vector floating point operations
- IBFR = Instruction Buffer Fetch Rate (megafetches/sec)
- IFact = Inline Factor (total calls to routine / average time spent in routine for each call)
- MC = number of memory conflicts
- MR = number of memory references
- Time = total CPU time (sec)

9. Coverage Analysis

A coverage analysis shows that the sample problem yielded a 91% segment coverage of CLOUD. Sample problems provided with simulation programs typically achieve only 35% to 50% coverage. A statement of software quality cannot be made for routines that have low coverage, i.e., large portions of the code are untested.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
CLOUD	282	256	90.8

```
Totals        282          256          90.8


                      0.20       0.40       0.60       0.80       1.00
                    ----+----|----+----|----+----|----+----|----+----|
          CLOUD     |*********************************************************
                    ----+----|----+----|----+----|----+----|----+----|
```

```
0.90 <= coverage < 0.95       CLOUD
```

```
Program coverage for this run =0.91
```

10. Complexity Analysis

Some key metrics are the number of executable statements (sloc), the number of non-blank comments (ncomt), McCabe's extended cyclomatic complexity (vg2), the number of branching statements (cgoto, ugoto, bIF, and lIF), and Halstead's predicted number of errors in (re)writing the code (bhat). Measures are normalized per 100 executable statements for ease of comparison and are listed in the table below.

The branching measures for this code (ugoto/sloc, lif/sloc) indicate moderate/low values. This code appears to be fairly well structured.

The program shows a low ratio of non-blank comments to source code. This code may benefit from more internal documentation.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates a high value. Generally, the routines with the highest complexity are those most likely to have defects. As a guideline, normalized measures of 15 or greater should be considered complex. A software maintenance program should focus on those routines with the highest measures.

Complexity Report by Subprogram for CLOUD

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
CLOUD	436	234	170	142	60.7	25.6	0	0.0	22	9.4	0	0.0	8	3.4	5

Legend of Metrics in Report

loc -- lines of code
sloc -- number of executable statements
cmnt -- total number of commnts
ncomt -- number of non-blank COMMENT statements
100*ncomt/sloc -- percent, nonblank comments to number of executable statements
100*vg2/sloc -- percent, extended complexity of number of executable statements
cgoto -- number of COMPUTED GO TO statements
100*cgoto/sloc -- percent, computed GOTO's to number of executable statements
ugoto -- number of UNCONDITIONAL GO TO statements
100*ugoto/sloc -- percent, unconditional GOTO's to number of executable statements
bIF -- number of BLOCK IF statements
100*bif/sloc -- percent, Block IF statements to number of executable statements
lIF -- number of LOGICAL IF statements
100*lif/sloc -- percent, logical IF statements to number of executable statements
Bhat -- Halstead's predicted number of errors in writing code