SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-028

02. Project Title:

AIRCOM - A Module for Iterative Performance Assessment Phase 2.

Project No. 20-5702-723

03. SRN Title: AIRCOM

04. Originator/Requester: Thomas J. Ratchford

Date: 04/22/94

05. Summary of Actions

- ■ Release of new code admitted to CM System (R. Baca)
- ☐ Release of modified code:
  - ☐ Enhancements made
  - ☐ Corrections made
- ■ Change of access code (R. Janetzke)

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: Date:

08. Remarks:

A copy of the software package AIRCOM, Ver. 1.1 was retained by the Principle Investigator for use in the CNWRA work center; therefore, a new release may not be necessary.

SOFTWARE SUMMARY FORM

01. Summary Date: 04/22/94	02. Summary prepared by(Name and Phone) T.J. Ratchford 522-3083	03. Summary Action: New
04. Software Date: 8/15/93	05. Short Title: AIRCOM	
06. Software Title: AIRCOM - A Module for Iterative Performance Assessment Phase 2.		07. Internal Software ID: NONE

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

11. Submitting Organization and Address:	12. Technical Contact(s) and Phone:
CNWRA, SwRI, San Antonio, Texas	R. Janetzke, (210) 522-3318

13. Narrative:

AIRCOM - Aircom was designed to be a preprocessor for the DITTY module that runs under the TPA code. AIRCOM oraganizes DITTY release input file by nuclide.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	6,143 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: Thomas J Ratchford Date: 4/22/94

CNWRA Form TOP-4-1 (08/93)

```
gemstone.1 ~ => cd ~/tpa/AIRCOM/WKDIR
gemstone.2 ~/tpa/AIRCOM/WKDIR => ls -l
total 69
-rwxrwx---   1 tjr1     tjr1       131072 Apr 22 12:08 AIRCOM.TAR*
-rwxrwx---   1 tjr1     tjr1          932 Apr 22 12:06 Makefile*
-rwxrwx---   1 tjr1     tjr1          359 Apr 22 12:06 TPA_AIR.AGD*
-rwxrwx---   1 tjr1     tjr1        12288 Apr 22 12:06 aircom.F*
-rwxrwx---   1 tjr1     tjr1        48069 Apr 22 12:06 aircom.f*
-rwxrwx---   1 tjr1     tjr1          635 Apr 22 12:06 aircom.inp*
-rwxrwx---   1 tjr1     tjr1        23372 Apr 22 12:06 aircom.pre*
-rwxrwx---   1 tjr1     tjr1         6371 Apr 22 12:06 c14air.dat*
-rwxrwx---   1 tjr1     tjr1          510 Apr 22 12:06 ckname.F*
-rwxrwx---   1 tjr1     tjr1         2748 Apr 22 12:06 dr2air.dat*
-rwxrwx---   1 tjr1     tjr1          644 Apr 22 12:06 factor.F*
-rwxrwx---   1 tjr1     tjr1          666 Apr 22 12:06 fnuc.F*
-rwxrwx---   1 tjr1     tjr1         8983 Apr 22 12:06 opnfil.F*
-rwxrwx---   1 tjr1     tjr1          291 Apr 22 12:06 valnuc.F*
-rwxrwx---   1 tjr1     tjr1         2638 Apr 22 12:06 volair.dat*
-rwxrwx---   1 tjr1     tjr1          235 Apr 22 12:06 x.aircom.cov*
-rwxrwx---   1 tjr1     tjr1          247 Apr 22 12:06 x.aircom.tes*
gemstone.3 ~/tpa/AIRCOM/WKDIR =>                    134.20.1.1 14:03:32
```

4/22/98

AIRCOM Fortran Program
Static and Dynamic Analysis

June 28, 1993

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 provided by Southwest Research Institute (SwRI).

One sample problem was supplied 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

There are 6 entry points with no alternate entry points. There are no unreferenced subroutines or functions.

4. Common Block Irregularities

There are no common blocks.

5. Interface Irregularities

Argument usage is consistent.

6. Local Variable Irregularities

Parameter usage is consistent.

Integer parameter "airerr" is declared but unused in "aircom".

The local variables "nnucar", "title", and "vecnum" are defined but unused in "aircom".

7. Fortran Extensions

All routines contain some lower case alphabetic characters in their active Fortran.

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 longer runs.

The performance data show that a high percentage of the overall execution time (88.298%) is spent in the first routine listed (AIRCOM). This is due primarily to the following:

1) a low percentage of floating point operations which are performed in vector mode (%Vflops is small)

2) a high rate of instruction buffer fetches (IBFR > 1)

A detailed optimization analysis effort should focus on these 2 areas.

PERFORMANCE DATA FOR AIRCOM

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
AIRCOM	0.061	87.618	87.618	10.36721	0.00	0.241	1.049
OPNFIL	0.004	6.020	93.638	0.00000	0.00	0.418	0.450
FNUC	0.002	3.036	96.674	0.00000	0.02	0.380	0.178
VALNUC	0.001	1.718	98.392	0.00000	0.01	0.136	0.189
FACTOR	0.001	1.326	99.718	0.00000	0.02	0.129	0.340
CKNAME	0.000	0.282	100.000	0.00000	0.09	0.897	1.208
Totals (All Traced Routines)	0.069	100.000	100.000	7.12713	0.01	0.255	0.963

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)

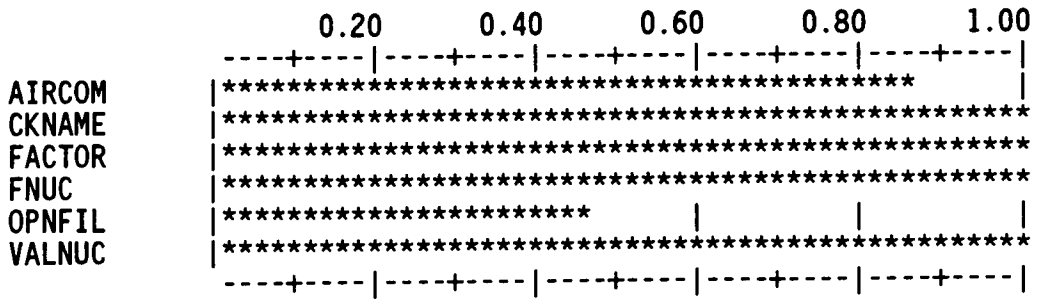
9. Coverage Analysis

One sample problem was supplied. A coverage analysis shows that this problem yielded an 82% segment coverage of AIRCOM. Sample problems provided with simulation programs typically achieve 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.

One routine achieves 40%-59% coverage, 1 routine achieves 85%-89% coverage, and 4 routines achieve 100% coverage.

The following table shows the percent coverage for each routine.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
AIRCOM	134	115	85.8
CKNAME	5	5	100.0
FACTOR	8	8	100.0
FNUC	6	6	100.0
OPNFIL	28	13	46.4
VALNUC	6	6	100.0
Totals	187	153	81.8




```
0.40 <= coverage < 0.60      OPNFIL

0.85 <= coverage < 0.90      AIRCOM

     coverage = 1.00         CKNAME    FACTOR    FNUC      VALNUC

 Program coverage for this run =0.82
```

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 indicate few unconditional GO TO statements and logical IFs for most program modules. This code appears to be well structured.

Most routines have a good ratio of non-blank comments to source code. The exception is "valnuc" which has no comments.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates relatively high values. 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 Aircom

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bif bIF	bif /sloc	lif lIF	lif /sloc	Bhat
aircom	346	256	75	67	26.2	27.0	0	0.0	5	2.0	32	12.5	12	4.7	4
ckname	16	9	4	2	22.2	44.4	0	0.0	0	0.0	1	11.1	0	0.0	0
factor	24	14	8	5	35.7	28.6	0	0.0	1	7.1	1	7.1	1	7.1	0
fnuc	23	10	10	6	60.0	30.0	0	0.0	1	10.0	1	10.0	0	0.0	0
opnfil	208	59	131	118	200.0	27.1	0	0.0	0	0.0	8	13.6	0	0.0	0
valnuc	13	10	0	0	0.0	30.0	0	0.0	1	10.0	1	10.0	0	0.0	0

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