SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-024

02. Project Title:
PREFOR is designed to process code written in Fortran in which preFOR commands have been embedded. The output is standard Fortran thus increasing code portability..

Project No.
20-5702-723

03. SRN Title: PREFOR

04. Originator/Requester:
Thomas J. Ratchford

Date:
03/23/94

05. Summary of Actions

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

06. Persons Authorized Access

Name	RO/RW	A/C/D

07. Element Manager Approval: Date:

08. Remarks:

A copy of the software package PREFOR, 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: 03/23/94	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: PREFOR	
06. Software Title: PREFOR - A preprocessor used with TPA codes.		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: TPA Preprocessor

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

13. Narrative:

PREFOR - PREFOR is designed to process code written in Fortran in which preFOR commands have been embedded. The output is standard Fortran thus increasing code portability.

14. Computer Platform	15. Computer Operating System:	16. Programming Language(s):	17. Number of Source Program Statements:
CRAY/XMP	UNIX	FORTRAN	21,837 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:_____

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

CNWRA INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary Date			02. Summary prepared by (Name and phone)	03. Summary action
Yr.	Mo.	Day	R. Janetzke, x3318	New [x] Replacement [ ] Deletion [ ]
9 2	0 1	0 8	03. Software title: preFOR; A Pre-processor for FORTRAN files	Previous Internal Software ID
04. Software Date				
Yr.	Mo.	Day		

06. Short title: preFOR

07. Internal Software ID:

08. Software type	09. Processing Mode	10. APPLICATION AREA	
		General	Specific
[ ] Automated Data System	[ ] Interactive	[x] Computer Systems Support/Utility	configuration control
[x] Computer Program	[ ] Batch	[ ] Management/Business	
[ ] Subroutine/Module	[x] Combination	[ ] Scientific/Engineering	
		[ ] Process Control	
		[ ] Bibliographic/Textual	
		[ ] Other	

11. Submitting organization and address	12. Technical contact(s) and phone
SwRI	Ron Janetzke 512-522-3318

13. Narrative

preFOR is designed to process code written in FORTRAN in which preFOR commands have been embedded. The output of the processing will be a standard FORTRAN file that can be compiled like an other FORTRAN file. The preFOR commands allow the programmer significant flexibility in coding. preFOR commands include creation of code blocks that can be inserted at appropriate places. These insertions can be made conditional on use of a particular central processing unit (CPU), thus increasing code portability. In addition, the preFOR utility provides other features such as numbering of lines in the code, deletion of comments, and trimming of trailing characters in lines of code. preFOR is written in ANSI standard FORTRAN 77.

14. Keywords: FORTRAN, Pre-processor

15. Computer manufacturer and model	16. Computer operating system	17. Programming language(s)	18. Number of source program statements
DEC/VAX	VMS	FORTRAN 77	1147
19. Computer memory requirements	**20. Tape drives**	**21. Disk/Drum units**	**22. Terminals**
Virtual	0	0	0

23. Other operational requirements: None

24. Software availability			25. Documentation availability		
Available	Limited	In-house only	Available	Inadequate	In-house only
[x]	[ ]	[ ]	[x]	[ ]	[ ]

26. FOR SUBMITTING ORGANIZATION USE

This is a stand alone code used in respect to TPA Phase 2.
Dan 12/31/98

```
upcase.f.Z*  upcase.l.Z*  upstrg.f.Z*  upstrg.l.Z*  writer.f.Z*  writer.l.Z*
gemstone.10 ~/tpa/PREFOR/WKDIR => uncompress *
prefor.f.Z: No such file or directory
gemstone.11 ~/tpa/PREFOR/WKDIR => ls -l
total 290
-rwxrwxr-x   1 tjr1     tjr1          946 Jul  8  1993 Makefile*
-rwxrwxr-x   1 tjr1     tjr1         2977 Jul  8  1993 cname.f*
-rwxrwxr-x   1 tjr1     tjr1        13222 Jul  8  1993 cname.l*
-rwxrwxr-x   1 tjr1     tjr1       134102 Jul  8  1993 listing*
-rwxrwxr-x   1 tjr1     tjr1       748056 Jul  8  1993 prefor*
-rwxrwxr-x   1 jet      tjr1        23062 Jul  8  1993 prefor.f*
-rwxrwxr-x   1 tjr1     tjr1        64819 Jul  8  1993 prefor.l*
-rwxrwxr-x   1 tjr1     tjr1        99380 Jul  8  1993 prefor.m*
-rwxrwxr-x   1 tjr1     tjr1          950 Jul  8  1993 strail.f*
-rwxrwxr-x   1 tjr1     tjr1         8058 Jul  8  1993 strail.l*
-rwxrwxr-x   1 tjr1     tjr1         1804 Jul  8  1993 tabfix.f*
-rwxrwxr-x   1 tjr1     tjr1        11114 Jul  8  1993 tabfix.l*
-rwxrwxr-x   1 tjr1     tjr1          934 Jul  8  1993 upcase.f*
-rwxrwxr-x   1 tjr1     tjr1         5829 Jul  8  1993 upcase.l*
-rwxrwxr-x   1 tjr1     tjr1          782 Jul  8  1993 upstrg.f*
-rwxrwxr-x   1 tjr1     tjr1         7790 Jul  8  1993 upstrg.l*
-rwxrwxr-x   1 tjr1     tjr1         5783 Jul  8  1993 writer.f*
-rwxrwxr-x   1 tjr1     tjr1        22440 Jul  8  1993 writer.l*
gemstone.12 ~/tpa/PREFOR/WKDIR =>
                                                    134.20.1.1 11:02:33
```

PREFOR Fortran Program Static and Dynamic Analysis

March 10, 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 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 PREFOR program contains 7 Fortran routines.

There are no alternate entry points.

4. Common Block Irregularities

There are no common blocks in the PREFOR program.

5. Interface Irregularities

No Exceptions to report.

6. Local Variable Irregularities

Local variable exceptions are noted as follows:

Module	Variable	Exception
prefor	deckcd	Defined, Unused
writer	indent	UNUSED
writer	lcmind	UNUSED
writer	lcomnt	Undefined, Unused
writer	ltabin	UNUSED
writer	ltabrp	Undefined, Unused
writer	upcase	Undefined, Unused

7. Fortran Extensions

The following modules contain potential overlaps in character assignment statements:

prefor, tabfix, writer.

The following modules contain lowercase characters in their active Fortran:

prefor, cname, strail, tabfix, upcase, upstrg, writer.

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.

The performance data show that a high percentage of the overall execution time (99.894%) is spent in the first 3 routines listed. This is due primarily to the following (applies to some or all of the 3 routines):

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

2) a high overhead factor for calls to the routines (IFact > 1)

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

A detailed optimization analysis effort should focus on these 3 areas.

PERFORMANCE DATA FOR PREFOR

ROUTINE NAME	Time	%ExTime	%AccumT	%Vflops	IFact	MC/MR	IBFR
WRITER	5.802	68.034	68.034	0.00000	0.71	0.154	1.075
PREFOR	1.976	23.175	91.209	0.00000	0.00	0.315	0.743
STRAIL	0.741	8.685	99.894	0.00000	5.53	0.222	0.089
CNAME	0.007	0.081	99.975	0.00000	0.13	0.998	0.517
UPCASE	0.002	0.019	99.994	0.00000	1.10	1.708	0.846
UPSTRG	0.001	0.006	100.000	0.00000	0.00	3.728	1.455
Totals (All Traced Routines)	8.528	100.000	100.000	0.00000	1.99	0.202	0.912

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)

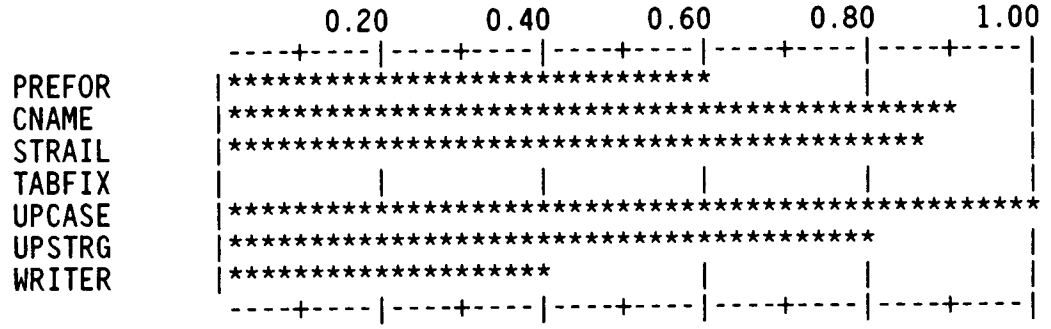
9. Coverage Analysis

A coverage analysis shows that the sample problem yielded a 58% segment coverage of PREFOR. 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.

Note that 1 routine has 0% coverage. This routine is not tested with the supplied sample problem.

One routine achieves 20%-39% coverage, 2 routines achieve 60%-79% coverage, 2 routines achieve 85%-90% coverage, and 1 routine achieves 100% coverage.

Module Name	Number of Segments in module	Number of Segments Executed	Percent Segment Coverage
PREFOR	123	74	60.2
CNAME	29	26	89.7
STRAIL	7	6	85.7
TABFIX	16	0	0.0
UPCASE	4	4	100.0
UPSTRG	5	4	80.0
WRITER	41	16	39.0
Totals	225	130	57.8



coverage = 0.	TABFIX	
0.20 <= coverage < 0.40	WRITER	
0.60 <= coverage < 0.80	PREFOR	UPSTRG
0.85 <= coverage < 0.90	CNAME	STRAIL
coverage = 1.00	UPCASE	

Program coverage for this run =0.58

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 moderately high values for some routines. This code may benefit from a restructuring effort aimed at reducing the number of unconditional GO TO and logical IF statements in these routines.

All routines show a good ratio of non-blank comments to source code.

McCabe's extended cyclomatic complexity (vg2), normalized per 100 lines of source code, indicates 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 PREFOR

Name	loc	sloc	cmnt	ncomt	ncomt /sloc	vg2 /sloc	cgoto	cgoto /sloc	ugoto	ugoto /sloc	bIF	bif /sloc	lIF	lif /sloc	Bhat
preFOR	657	241	354	300	124.5	35.3	0	0.0	23	9.5	28	11.6	16	6.6	4
CNAME	98	47	54	42	89.4	46.8	0	0.0	5	10.6	4	8.5	7	14.9	1
strail	42	14	22	15	107.1	35.7	0	0.0	2	14.3	0	0.0	2	14.3	0
tabfix	68	25	34	26	104.0	28.0	0	0.0	1	4.0	3	12.0	1	4.0	0
upcase	38	7	29	22	314.3	42.9	0	0.0	0	0.0	1	14.3	0	0.0	0
upstrg	39	10	25	16	160.0	30.0	0	0.0	1	10.0	0	0.0	1	10.0	0
WRITER	198	60	108	89	148.3	35.0	0	0.0	1	1.7	11	18.3	5	8.3	1

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