27
11-28-80
7#C to NTIS
25-avg

SAND80-2181
Unlimited Release

#1

AJAX User's Guide

MASTER

M. Dale Fimple

Prepared by Sandia National Laboratories, Albuquerque, New Mexico-87185
and Livermore, California 94550 for the United States Department
of Energy under Contract DE-AC04-76DP00789

Printed October 1980

Sandia National Laboratories

R136

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

SAND80-2181
 Unlimited Release
 Printed October 1980

A		JJJ		A	X	X
A A		J		A A	X	X
A A		J		A A	X	X
AAAAAAA		J		AAAAAAA	X	
A A		J		A A	X	X
A A	J	J		A A	X	X
A A	JJJJJJ			A A	X	X

USER'S GUIDE

M. Dale Fimple
 Sandia National Laboratories
 Division 1723
 Albuquerque, New Mexico

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ABSTRACT:

AJAX is a computer program which processes programs written in DIGITAL FORTRAN-IV PLUS language, and produces new versions of these programs. The most important feature of a program produced by AJAX is that the numbered statements in the program will be renumbered such that they will be monotonically increasing from start to finish. Other formatting features are available as options. AJAX is written in VAX FORTRAN-IV PLUS language and runs on version 1.6 of the VMS operating system.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

PREFACE

During the last fifteen to twenty years a number of computer programs have been published which do a satisfactory job of cleaning up old FORTRAN programs, primarily by renumbering the statements, so that referenced statement numbers are easy to locate in the listing of the program. Most of these programs also eliminate statement numbers which are not referenced.

Probably the best known of these programs is TIDY, which was written by HARRY M. MURPHY, JR. of the AIR FORCE WEAPONS LABORATORY at KIRTLAND AIR FORCE BASE in ALBUQUERQUE, N.M. TIDY was written for the large scale machines built by CONTROL DATA CORPORATION. TIDY was made public in a report "TIDY, A COMPUTER CODE FOR RENUMBERING AND EDITING FORTRAN SOURCE PROGRAMS", TECHNICAL REPORT NO. AFWL-TR-66-96, OCTOBER 1966. This program has been adapted for use on most of the large scale computers in use since that time.

The current author found TIDY to be a very satisfactory and valuable program while working on the CDC-6600, the IBM-7090, and the UNIVAC 1108. However, when confronted with conversion to the mini-computers available in 1975, it was decided that it would be more advisable to write a new program.

AJAX was originally written in 1975 to run on the DATA GENERAL NOVA-3 computer, and processed programs written in DATA GENERAL FORTRAN-IV language. In 1976 AJAX was converted to run on the DATA GENERAL ECLIPSE computer and accept programs written in the DATA GENERAL FORTRAN-V language as input. The ECLIPSE version is still in use, but is undocumented. AJAX was converted to the VAX/VMS system in the summer of 1979. AJAX has been corrected, modified, and expanded when appropriate during the last year, and it is presumed that this process will continue for some time. It is anticipated that AJAX will be up-dated to handle DIGITAL EQUIPMENT CORPORATION'S VAX/VMS native mode FORTRAN 77 language when this compiler is released.

The current author has been heavily influenced in this area of programming by the author of TIDY as well as others, (particularly Hans Andreson of IBM and Duane Jensen of DIKEWOOD CORP.) In particular, a subroutine used in AJAX, to analyse FORTRAN statements, is a direct descendant of a routine called CSCAN which was written by Duane Jensen (deceased) of DIKEWOOD CORP. It is my impression that this routine was part of a FORTRAN compiler written for the IBM-7040 at Kirtland Air Force Base in the early 1960's. This routine has been modified and expanded by the present author, to the extent that he is completely responsible for any defects in the current version of this routine.

TABLE OF CONTENTS.

1.0	USE.	4
1.1	INPUT.	4
1.2	OUTPUT (WITH EXAMPLES).	5
1.2.1	STANDARD FORTRAN OUTPUT.	6
1.2.2	USERS INPUT FORMAT.	7
1.2.3	BLOCK STRUCTURED OUTPUT.	8
1.3	DIGITAL CONTROL LANGUAGE (DCL).	9
1.4	OPTIONS.	10
1.4.1	BASE	10
1.4.2	INCREMENT	10
1.4.3	D LINE or DLINE	10
1.4.4	OUTPUT	10
1.4.5	INDENTATION	11
1.4.6	INSERT	11
1.5	DEFAULT OPTIONS.	12
2.0	ERROR CONDITIONS.	13
2.1	PASS1	13
2.1.1	ERROR INFORMATION FROM AJAX.	13
2.1.2	FILE STATUS.	13
2.2	PASS2.	13
2.2.1	ERROR INFORMATION FROM AJAX.	13
2.2.2	FILE STATUS.	13
3.0	GENERATING THE EXECUTABLE ELEMENT.	14
4.0	KNOWN DEFICIENCIES.	15
4.1	EMBEDDED COMMENTS.	15
4.1.1	EXAMPLE 1.	15
4.1.2	EXAMPLE 2.	15
4.2	DANGLING CODE.	15

1.0 USE.

1.1 INPUT.

AJAX accepts as input either of the following:

a) A file containing one or more FORTRAN program modules.

b) A file containing the "names" of one or more files as described in (a) above.

It is assumed that programs to be processed by AJAX have been compiled without diagnostics by the VAX/VMS FORTRAN-IV PLUS compiler, distributed with Version 1.0 - 1.6 of VAX/VMS.

1.2 OUTPUT (WITH EXAMPLES).

AJAX produces one or more files, (corresponding to the input), containing new versions of the input files. The output version will usually be a more readable version of the old program(s). The output program will always have the referenced statements re-numbered so that they are monotonically increasing from the beginning to the end of the program module. The base statement number and the increment between adjacent statement numbers is selectable by the user. At the option of the user, the output program may be re-formatted in several available formats.

In order to demonstrate the several output formats the following rather typical FORTRAN function has been constructed strictly for demonstration purposes. This program is written in VAX FORTRAN-IV PLUS and the indentation is typical of many such programs. The statement numbers have been deliberately scrambled and there is one which is superfluous, since it is never referenced in the program. The program compiles without diagnostics.

```

      INTEGER FUNCTION FIND(A,B,N)
      LOGICAL*1 A,B(N)
300    IF(A.LT.B(1))THEN
          FIND=(-1)
      ELSE IF(A.GT.B(N))THEN
          FIND=0
      ELSE
          DO 33 I=1,N
              IF(A.EQ.B(I))THEN
                  FIND=I
                  GO TO 8
              ENDIF
33      CONTINUE
      8      ENDIF
          RETURN
      END

```

1.2.1 STANDARD FORTRAN OUTPUT. -

When the user selects this mode of output, the new version of his program(s) will be formatted in the historically conventional FORTRAN format. All leading tabs, and blanks will be removed, giving a program with no indentation properties. The program will have the general appearance of a card oriented program. Needless to say, using this mode of output will destroy any existing attempt at indentation by the user.

This output option would convert the demonstration program to the following:

```

      INTEGER FUNCTION FIND(A,B,N)
      LOGICAL*1 A,B(N)
      IF(A.LT.B(1))THEN
      FIND=(-1)
      ELSE IF(A.GT.B(N))THEN
      FIND=0
      ELSE
      DO 100 I=1,N
      IF(A.EQ.B(I))THEN
      FIND=I
      GO TO 110
      ENDIF
100    CONTINUE
      ENDIF
110    RETURN
      END

```

1.2.2 USERS INPUT FORMAT. -

This is the default output mode in AJAX, and results in a program which is formatted in approximately the same manner as the input. Specifically, all leading tabs are retained in each statement, and breaks between adjacent continuation statements usually are not changed. The intent, in this case, is to simply renumber the statements and leave everything else, as much as possible, the way the programmer wrote it.

The demonstration program, when processed with this option yields the following program:

```

      INTEGER FUNCTION FIND(A,B,N)
      LOGICAL*1 A,B(N)
      IF(A.LT.B(1))THEN
          FIND=(-1)
      ELSE IF(A.GT.B(N))THEN
          FIND=0
      ELSE
          DO 100 I=1,N
              IF(A.EQ.B(I))THEN
                  FIND=I
                  GO TO 110
              ENDIF
          CONTINUE
100    ENDIF
110    RETURN
      END

```

1.2.3 BLOCK STRUCTURED OUTPUT. -

This output mode represents an attempt to present the program with a maximum ammount of block structure consistent with the available language constructs. The "DO LOOP" is indented in a manner similar to the "IF", "THEN", "ELSE" construction.

With the use of this option, the demonstration program comes out as follows:

```

      INTEGER FUNCTION FIND(A,B,N)
      LOGICAL*1 A,B(N)
      IF(A.LT.B(1))THEN
        FIND=(-1)
      ELSE IF(A.GT.B(N))THEN
        FIND=0
      ELSE
        DO 100 I=1,N
          IF(A.EQ.B(I))THEN
            FIND=I
            GO TO 110
          ENDIF
100      CONTINUE
      ENDIF
110      RETURN
      END

```

It should be noted that the number of indentation spaces for each level of indentation is an option, and may vary from 0 to 8. When 8 is used, tabs are inserted instead of groups of blanks. The example above uses the default value, which is four spaces per indentation level.

1.3 DIGITAL CONTROL LANGUAGE (DCL).

Two lines of DCL are required to execute AJAX:

```
$ AJAX:==$device:[directory]AJAX.EXE          (1)
```

```
$ AJAX/opt1/opt2/---/optn      pname          (2)
```

The first command line need be executed only one time per session and would probably be included in the user's LOGIN.COM file. The "device" and "directory" will be different for each user installation.

The second line contains the actual command to execute AJAX. OPT1,OPT2,---OPTN are defined below. The name of the program to be processed is "pname" as in the usual DCL command.

pname may take any of the following forms;

```
NAME
NAME.TYP
@NAME
@NAME.TYP
```

Where .TYP is a file type such as .FOR, .COM, .FTN etc.

In the absence of the @, NAME refers directly to the file containing the FORTRAN program(s) to be processed, and in the absence of .TYP, the suffix .FOR is implied. When @ is present, NAME refers to a file containing the names of several program files to be processed by AJAX. When the @ is present and .TYP is not specified, .COM is implied. In all of the cases above, version numbers are not allowed, since AJAX always processes the highest version present (n) and creates a new file with version equal to (n+1).

1.4 OPTIONS.

In specifying options, the first three characters are sufficient to specify an option name. The options presently available are:

1.4.1 BASE -

/BASE=nnn

nnn will be the number of the first numbered statement in the output program. The default value is 100.

1.4.2 INCREMENT -

/INCREMENT=nn

nn will be the difference between the nth and the (n+1)th numbered statements. The default value is 10.

1.4.3 D_LINE or DLINE -

/D_LINE or /DLINE

Debug lines (statements with D in column 1) are to be retained in the new program. To delete debug statements use the option, /NOD LINE or /NODLINE. The default is /D LINE. NOTE THAT IN THIS CASE THE "NO" PRECEDING D LINE OR DLINE SIMPLY NEGATES THE FOLLOWING OPTION. THE "NO" MUST BE FOLLOWED BY AT LEAST THREE CHARACTERS OF THE OPTION, EXAMPLE; "NOD_L" OR "NODLI". "NOD" WILL NOT WORK!

1.4.4 OUTPUT -

/OUTPUT=aaaaaa

aaaaaa is the format for the output program. Acceptable values for aaaaaa are:

STANDARD for the output option gives an output program with no indentation of statements as in old fashioned card oriented FORTRAN.

INPUT for the output option will give an output program formatted approximately the same as the input program. However, most redundant or extraneous blanks will be removed.

BLOCK for the output format will produce a program which is block structured, with respect to BLOCK IF and DO LOOP statements.

The default output format is INPUT.

1.4.5 INDENTATION -

/INDENTATION=nn

This option is meaningful only if the option /OUTPUT=BLOCK is also present. nn is the number of blanks inserted before each statement for each level of indentation. The level of indentation depends on the basic structure of the input program. nn may be any number from 0 to 8. If nn=0, then the output format will be the same as STANDARD (see above). If nn = 8, one tab will be inserted instead of 8 blanks. The default for INDENT is 4.

1.4.6 INSERT -

/INSERT=timer-routine

timer-routine is a user supplied subroutine name.

The use of the INSERT= timer-routine option will result in the following changes in the user's program.

a) Immediately before the first executable statement of the program, the following statement will be inserted;

```
CALL timer-routine('pname',1)
```

b) Immediately before all "RETURN" statements, the following statement will be inserted.

```
CALL timer-routine('pname',2)
```

AJAX will correctly handle both of the following situations:

```

1)
becomes      100      RETURN

              100      CALL timer-routine('pname',2)
              RETURN

2)
becomes      100      IF(A.EQ.B)RETURN

              100      IF(A.EQ.B)THEN
              CALL timer-routine('pname',2)
              RETURN
              ENDIF

```

The INSERT option was put in AJAX as an aid in improving the execution time of AJAX. It is documented here because it is felt that it may be of use to other programmers in the same manner. The timer-routine (aptly named TIMER) included as a part of AJAX may be used, since it is written in such a way as to be independent of AJAX. If TIMER is used, then the subroutines TIMEX and AJSORT must also be linked with the executable program. TIMEX is the "error exit" routine for TIMER, and AJSORT is a SHELL sort routine. When the TIMER routine included in the AJAX package is used, the following call must be executed at the end of a run, to obtain the run time statistics:

```
CALL timer-routine('pname',3)
```

1.5 DEFAULT OPTIONS.

The default values for the options are the same as those obtained from execution of the following DCL command line.

```
$ AJAX/BASE=100/INCREMENT=10/D_LINE/FORM=INPUT PNAME.FOR
```


2.0 ERROR CONDITIONS.

In most cases of error in processing, AJAX will crash, and produce the following results:

2.1 PASS1

2.1.1 ERROR INFORMATION FROM AJAX. -

If in pass1, some information concerning the cause of the crash will appear on the terminal and will also appear in a file named PNAME.JAX where PNAME is the name of the file being processed.

2.1.2 FILE STATUS. -

If in pass1 the input file will remain the highest cycle or version number with the given name.

2.2 PASS2.

2.2.1 ERROR INFORMATION FROM AJAX. -

If in pass2, the error information will appear only on the terminal.

2.2.2 FILE STATUS. -

If in pass2, AJAX will be in the process of creating the (n+1) version of the input file, and will leave this file in the user's directory. This file must be deleted before proceeding with any other action.

3.0 GENERATING THE EXECUTABLE ELEMENT.

AJAX will be distributed on a tape produced by the BCK/RST utility. The internal tape label is BONAMI. This tape should be loaded into an appropriate mass storage directory, with a set of commands similar to the following:

```
$ ALLOCATE      MTAO:
$ MOUNT        MTAO:  BONAMI
$ ASSIGN       MTAO:  TAPE:
$ MCR RST *.* /SL=MTAO:AJAX
$ DEASSIGN     TAPE
$ DISMOUNT     MTAO:
$ DEALLOCATE   MTAO:
```

The COMPILATION, LIBRARY BUILD, and the LINK operations are accomplished by executing the following command:

```
$ @AJAX.SSG
```

The command file, AJAX.SSG, executes several other command files, which are also included on the distribution tape. At the conclusion of the generation run, all the listing and object files will be in the directory. If desired, these may be deleted.

4.0 KNOWN DEFICIENCIES.

It is likely that the last bug in AJAX has not been found. To believe otherwise would be naive. The author welcomes the submission of programs which cause AJAX to fail. There are at least two known problems in the use of AJAX, for which no immediate solution is available.

4.1 EMBEDDED COMMENTS.

The following examples of programming were not anticipated in the design of AJAX, and modification of AJAX to handle them is not trivial. Some thought has been given to the problem, however, and it is expected that a future version of AJAX will be capable of handling them properly.

4.1.1 EXAMPLE 1. -

```

      DIMENSION A(3)
      DATA A/ 1.0,           ! A(1)=1.0
1         2.0,           ! A(2)=2.0
      2         3.0/       ! A(3)=3.0

```

4.1.2 EXAMPLE 2. -

```

      GOOD      ERROR
      GO TO(    100,    110,
C          1      ENDFILE
          120    ), ISTAT

```

4.2 DANGLING CODE.

While testing AJAX on as many programs available from as many sources as possible, this phenomenon was encountered fairly often. The programmer inserts code which he probably intends to make use of at a later date. The code starts with a numbered statement, but the statement is not referenced in the program module. Many FORTRAN compilers would detect the fact that the statement is never referenced and will never be reached. The VAX FORTRAN-IV PLUS compiler does not even issue a warning that the situation exists. If this type of program is processed by AJAX, the statement number that is never referenced will be deleted, leaving the statement "dangling". Subsequently, when the program is

compiled with VAX FORTRAN-IV PLUS, a warning message is issued to the effect that there is no path to the statement.

For the following two reasons, no attempt will be made to modify AJAX to take care of this situation:

a) AJAX does not attempt to analyze the program sufficiently to detect that the code can not be reached.

b) On several occasions, this trait of AJAX has been the means of solving difficult debugging problems. If a statement, or group of statements, can not be reached, the programmer should be aware of this.

DISTRIBUTION:

1120 T. L. Pace
1253 T. D. Sullivan
1523 B. T. Fox (5)
1700 W. C. Myre
1710 V. E. Blake
1715 W. D. Olson
1715 P. H. Arnold
1715 R. K. Cover
1715 C. C. Newcom
1720 C. H. Mauney
1721 J. W. Kane
1723 J. J. Baremore (15)
1723 M. D. Fimple (30)
1725 J. H. Davis
1727 V. K. Smith
1729 J. D. Williams
1730 J. D. Kennedy
1750 J. E. Stiegler
1760 J. Jacobs
2113 M. E. Daniel
2424 G. Carli (5)
2424 D. K. Robbins
2611 R. K. Vokes
2644 D. M. Darsey (5)
3141 T. L. Werner (5)
3151 W. L. Garner (3)for DOE/TIC (Unlimited Release)
3154-3 R. P. Campbell (25) DOE/TIC
5530 W. Hermann
5530 D. B. Saylor
8266 E. A. Aas

John Browning
AFWL/NTYV
KAFB
Albuquerque, N. M. 87117

Carlos Fernandez
AFWL/ADD
KAFB
Albuquerque, N. M. 87115

Randall G. Rusho
AFWL/ARLO
KAFB
Albuquerque, N. M. 86117

DISTRIBUTION: (continued)

Tom Benion
MS: 970
LASL

Dale Sparks
MS: 970
LASL

Tom Stup
MS: 828
LASL