MEMORANDUM
RM-6248-PR
MARCH 1970

JOOSTRAN: An Interactive JOSS Dialect for Writing and Debugging FORTRAN Programs

W. R. Graham and D. C. MacNeilage

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC RELEASE AND SALE; ITS DISTRIBUTION IS UNLIMITED.
MEMORANDUM
RM-6248-PR
MARCH 1970

JOSTRAN: An Interactive JOSS Dialect for Writing and Debugging FORTRAN Programs
W. R. Graham and D. C. MacNeilage

This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0015—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, HQ USAF.

Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of the United States Air Force.

DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.
PREFACE AND SUMMARY

JOSS was developed under contract to the Air Force, and is currently being used by approximately twenty DoD departments, The RAND Corporation, and numerous private commercial enterprises around the country. JOSTRAN is a JOSS dialect developed to expedite the construction of FORTRAN Programs. FORTRAN is the principal language used in performing numerical calculations with digital computers.

Writing and debugging a program in FORTRAN, which is executed by batch processing, is an experience that makes heavy demands upon the programmer's time and patience, and sometimes upon his sanity. In this Memorandum, a procedure is introduced which simplifies considerably the generation of FORTRAN programs.

Access to JOSS or a similar system is required. The program is first written and debugged in a specified FORTRAN-compatible dialect of JOSS called "JOSTRAN." JOSTRAN allows the user to take advantage of the interactive, list processing facilities of JOSS while writing and debugging the program, and at the same time minimizes the problems of ultimately translating the program into well-written FORTRAN. The completed and debugged JOSTRAN program is translated into FORTRAN in a straightforward manner, test calculations in the two languages being used as a final check of the translation.

Although this process has proven itself to be of considerable benefit in reducing FORTRAN program development time, the authors believe that JOSTRAN should be only an intermediate step in the evolution of a more versatile and powerful JOSS.
ACKNOWLEDGMENT

The authors wish to express their appreciation to A. C. Smith for the several useful additions and changes he suggested during the preparation of this Memorandum.
CONTENTS

PREFACE AND SUMMARY ........................................ iii
ACKNOWLEDGMENT .................................................. v

Section
I. INTRODUCTION .................................................. 1
II. JOSS->FORTRAN .............................................. 2
III. JOSTRAN ...................................................... 3
IV. RULES AND CONVENTIONS OF JOSTRAN ....................... 4
    Jostran Rules for the Fortran Programmer .................. 6
V. CONCLUSION ................................................... 8
I. INTRODUCTION

FORTRAN is today the most universally available and used language for programming computers to perform numerical calculations. In all but a few computing facilities, FORTRAN programs are executed by "batch processing," a somewhat primitive process that separates the completion of the programming effort from an inspection of its results by a period usually ranging between an hour and a week. Its drawbacks from the user's point of view are considerable.

JOSS* is a computing utility developed at RAND under Air Force funding and presently used at RAND and about 20 organizations within the Department of Defense. JOSS is an interactive system in which programming is performed through a dialogue between the user, at a remote terminal of the utility, and a central computer. JOSS language programs are executed by "list processing"; each statement is interpreted at execution time.

The FORTRAN language and the language of JOSS are greatly different; the differences become more apparent as one's skill with them increases. The JOSS language permits a flexibility and subtilty of expression that are difficult to imitate in FORTRAN. However, what FORTRAN lacks in subtlety it makes up in computing speed. Larger calculations can be performed using FORTRAN if one has the need or leisure to tolerate batch processing.

*JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program. For a description of the JOSS system and language, see C. L. Baker, JOSS: Introduction to a Helpful Assistant, The RAND Corporation, RM-5058-PR, July 1966.
II. JOSS\textasteriskaddition{\textdagger}FORTRAN

A programming methodology has been devised and used for the last two years which combines the major benefits of JOSS and FORTRAN. A program destined to be written in FORTRAN is first written and debugged on the JOSS system, using of course the JOSS language. The advantages gained by doing this over initial FORTRAN programming are:

(1) On-line editing.
(2) Instantaneous syntax checking.
(3) Very fast calculation turnaround (instantaneous, for most purposes).
(4) Expeditious debugging features resulting from the list processing: arbitrary program interruption, interrogation, and modification with the ability to resume execution at the point of interruption.

When the program has been written and debugged in JOSS, small test problems can be run for later reference. Then the finished JOSS program is translated into FORTRAN; all that remains to be done is the relatively straightforward task of making the FORTRAN program follow the logic of the JOSS program, and reproduce the results of the test problems.*

III. JOOSTRAN

The problem with a haphazard implementation of the above scheme is that there are several aspects of well-written JOSS that translate into rather awkward FORTRAN. However, the FORTRAN program can be made easy to construct from the JOSS version and reasonably intelligible and efficient if certain restrictions are placed upon the JOSS program and certain conventions are followed. The dialect of JOSS that results we call "JOOSTRAN."
IV. RULES AND CONVENTIONS OF JOSTRAN

A. Subscripted Variables; Dimensioned Arrays:

1. Use no more than three indices.
   For example:
   Do: \( x(i), y(i,j), z(i,j,k) \)
   Not: \( d(i,j,k,1,...) \)

2. Be consistent in the number of indices used for each variable.
   For example:
   If: 1.1 Set \( a(i,j) = \ldots \)
   Do: 2.2 Set \( a(m,n) = \ldots \)
   Not: 2.2 Set \( a(k) = \ldots \)

3. Pack all arrays densely.
   For example:
   Do: Use \( a(1), a(2), a(3), \ldots \)
   Not: Just \( a(2), a(4), a(6), \ldots \)

4. Index all arrays from 1 with increments of +1.
   For example:
   Do: Use \( a(1), a(2), a(3) \)
   Not: \( a(-1), a(0), a(1) \)

5. Keep a list of array names and dimensions.
   For example:
   Dimensioning List: \( a(\alpha), b(\beta,\alpha), \ldots \)
   where \( \alpha, \beta, \ldots \) are positive integers known at the start of program execution.

B. Variable Names:

1. Use the FORTRAN variable naming convention whenever possible.
   For example:
   Denote integer variables by the upper and lower case letters:
   I, J, K, L, M, N, i, j, k, l, m, n.
   Denote floating point variables by the remaining letters.
2. Otherwise, keep a list.
   For example:
   
   Integer List: L, i, S, t, . . .
   Floating Point List: F, 1, 0, a, T . . .

C. Do Loops:

1. Start and increment with unsigned integers or nonsubscripted
   integer variables greater than zero.
   For example:
   
   Do: Do step (part) n for I = 1(1)11.
   Do: Do step (part) n for I = i(j)k.
       where i, j, k greater than zero.
   Not: Do step (part) n for I = -1(1)1.
   Not: Do step (part) n for I = .1(.1)1.

2. Keep a table of the calling Do commands, and the steps and
   parts called.
   For example, for all:
   
   (Step) \( \psi \) Do part \( \phi \)
   (Step) \( \psi \) Do step \( \phi \)
   Form the table:

<table>
<thead>
<tr>
<th>Step</th>
<th>Location of Do Command</th>
<th>Location Called by Do Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>( \psi ) Do part ( \phi )</td>
<td>8</td>
</tr>
<tr>
<td>3.24</td>
<td>( \psi ) Do step ( \phi )</td>
<td>2.4</td>
</tr>
<tr>
<td>4.267</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>9.1</td>
<td></td>
<td>2.4, 5.43</td>
</tr>
</tbody>
</table>

   (Step 9.1 is an assigned Go To with the variable
taking on the values 2.4 and 5.43)

D. Form and Step Numbering:

   For example:
   
   Form 1, Form 92, . . .
2. Steps: Use 1.xxx to 99.xxx, where \( x = 1, \ldots, 9 \).

For example:

1.1 Set \( z = 0 \).

82.093 Set \( w = 1 \).

E. Data I/O Handling:

1. FORTRAN disk and tape units may be simulated by using JOSS file retrieval and storage commands.

For example, use item storage for simulating logical records on FORTRAN units:

Recall item 1 (T5LR1)

File \( x,y,z \) as item 2 (T5LR2)

correspond to using FORTRAN tape unit 5; logical records 1 and 2.

Recall item 11 (D6LR1)

File \( p,u,t \) as item 12 (D6LR2)

correspond to using FORTRAN disk file 6; logical records 1 and 2.

Using JOOSTRAN will make the translation from JOSS to FORTRAN a tedious but usually uncomplicated problem when the translation rules presented below are followed:

**JOOSTRAN RULES FOR THE FORTRAN PROGRAMMER**

A. Variable Names: The use of mnemonics for FORTRAN variables would require a variable translation table. To avoid the need for a table and still have an unambiguous translation, the following is proposed:

1. Represent JOSS single letter upper- and lower-case names in FORTRAN by the same single letter with appropriate suffixes \( U, L, F \) (for upper case, lower case, and introduced FORTRAN variable) and when required by the integer-floating point variable convention by the prefixes \( I \) and \( R \) (for integer and real).
For example:

<table>
<thead>
<tr>
<th>JOSS</th>
<th>FORTRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Variables:</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>AL</td>
</tr>
<tr>
<td>A</td>
<td>AU</td>
</tr>
<tr>
<td>i</td>
<td>RIL</td>
</tr>
<tr>
<td>Integer Variables:</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>IU</td>
</tr>
<tr>
<td>p</td>
<td>IPL</td>
</tr>
<tr>
<td>Introduced Variable:</td>
<td></td>
</tr>
<tr>
<td>JF</td>
<td></td>
</tr>
<tr>
<td>XF</td>
<td></td>
</tr>
</tbody>
</table>

2. In some instances, a JOSS dimension may be used not to represent a FORTRAN dimension but only to obtain more variables. In this case, the JOSS dimension may be removed in translation to the FORTRAN variable.

For example:

<table>
<thead>
<tr>
<th>JOSS</th>
<th>FORTRAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Variables:</td>
<td></td>
</tr>
<tr>
<td>b(1), b(2), b(3), ...</td>
<td>BL1, BL2, BL3, ...</td>
</tr>
<tr>
<td>B(1,1), B(1,2), B(1,3), ...</td>
<td>BU1C1, BU1C2, BU1C3, ...</td>
</tr>
<tr>
<td>j(1,1), j(1,2), j(1,3), ...</td>
<td>RJL1(1), RJL1(2), RJL1(3), ...</td>
</tr>
<tr>
<td>Integer Variables:</td>
<td></td>
</tr>
<tr>
<td>J(1,1,1), J(2,1,1), ...</td>
<td>JU1(1,1), JU2(1,1), ...</td>
</tr>
<tr>
<td>q(1), q(2), q(3), ...</td>
<td>IQL1, IQL2, IQL3, ...</td>
</tr>
</tbody>
</table>
V. CONCLUSION

The sequential use of JOSS and FORTRAN expedited through JOSTRAN is a union based on necessity rather than elegance. To those who regard the complete JOSS language more as a beautiful medium of expression than as a practical tool, the authors offer their sympathy. We believe that JOSTRAN should be only an intermediate step in the evolution of a more versatile and powerful JOSS.