MEMORANDUM
RM-5220-PR
AUGUST 1967

THE JOSS PRIMER

S. L. Marks and G. W. Armerding

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5220-PR
AUGUST 1967

THE JOSS PRIMER

S. L. Marks and G. W. Armerding

This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0045—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 Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

Published by The RAND Corporation

PREFACE

JOSS† is an on-line, time-shared computing service developed at The RAND Corporation to provide for the solution of numerical problems. Computational service is made available to the JOSS user through a personal IBM Selectric typewriter console--a mobile unit that is connected through an office outlet to a central high-speed computer. An auxiliary control box activates the console and indicates by visual and audible signals who controls the typewriter--the user for input or JOSS for output. The user instructs JOSS in imperative English sentences. He can specify procedures in arithmetic, algebra, trigonometry, and logic, and describe formats for typed results. JOSS will type an error message when the user violates a convention, helping him correct the error and resume processing. This dialogue between the user and JOSS makes the computer accessible to users of varying experience in computer programming.

The JOSS Primer is designed to introduce JOSS to the beginning user by means of examples that can be followed by a reader without programming experience. He is advised to study each example to understand the JOSS features illustrated, even though the mathematics may be unfamiliar. He first reads the entire Primer and then, seated at a JOSS console, follows the instructions step-by-step, duplicating examples, trying variations, and observing results. Emphasis is on the ease with which JOSS can be learned and used.

At the end of his last console session, the reader should be ready to solve simple problems, and to extend his knowledge by experience and further reading. To encourage the beginner to continue his use of JOSS, the Primer concludes with lists of JOSS commands and functions and suggested reading in the JOSS literature. *The JOSS User's Reference Manual*, in preparation as a companion document to the Primer, provides a source of information for the more experienced JOSS user. The Reference Manual

†JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

is an indexed presentation of all JOSS features from the user's point of view.

This memorandum is a part of Air Force Project RAND's continuing study of man-machine interaction, with the eventual goal of bringing the full power of an information processor directly to the user.

CONTENTS

I. INTRODUCTION

JOSS: A HELPFUL ASSISTANT

JOSS,[†] the JOHNNIAC Open Shop System, provides a personalized computing service to users ranging from those with no computer experience to professional programmers.

The JOHNNIAC computer, on which the experimental version of JOSS was implemented, was built in 1953 at The RAND Corporation and was named for the mathematician John von Neumann. Since its retirement in 1966, JOHNNIAC has been on display at the Los Angeles County Museum. The first version of JOSS was designed by J. C. Shaw, and went into operation in 1963, using a system of consoles designed by T. O. Ellis and M. R. Davis. The current version of JOSS operates on a Digital Equipment Corporation PDP-6 computer, and was implemented by C. L. Baker, G. E. Bryan, I. D. Greenwald, and J. W. Smith.

The user interacts with JOSS through a mobile console connected by telephone lines to the computer. The console consists of a modified IBM Selectric typewriter and a small control box. The user communicates with JOSS in imperative English sentences, which enable him to specify computational procedures in arithmetic, algebra, trigonometry, and logic.

When the user has control of the typewriter, a green light on the control box is on and typing is in green; when JOSS has control, a red light is on and typing is in black. Each time JOSS returns control to the user, a beep is heard. When the user violates a JOSS convention, JOSS types a message informing him of the violation. In most cases, it is easy for the user to correct his error and continue.

Of the many JOSS consoles connected to the PDP-6 computer, several may be in operation at one time, although an individual user is rarely

[†]JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

aware of this multiple use. The ability to serve many users at once, coupled with the capability of recognizing the user's errors and assisting him to correct them, makes JOSS a truly "open shop" system, easy to learn, and accessible to those without computer programming experience.

THE PURPOSE OF THE JOSS PRIMER

The goal of the JOSS Primer is to familiarize the beginner with JOSS to the point where he will be able to solve simple problems, and to prepare him to extend his knowledge of JOSS.

The Primer presents the basic elements of JOSS in terms of step-by-step instructions to the reader to type commands, to observe results, and to try variations of the examples. The beginner need not have mathematical training, nor must he be a skilled typist. If his interest does not extend beyond ordinary arithmetic, the examples from algebra and trigonometry need not be stressed. However, because all the examples have been chosen to illustrate specific JOSS features, the user may wish to follow each example even though the mathematics used is not familiar.

At the end of the Primer, there is a list of JOSS commands and functions as well as suggested reading from the JOSS literature.

SUGGESTIONS FOR USING THE JOSS PRIMER

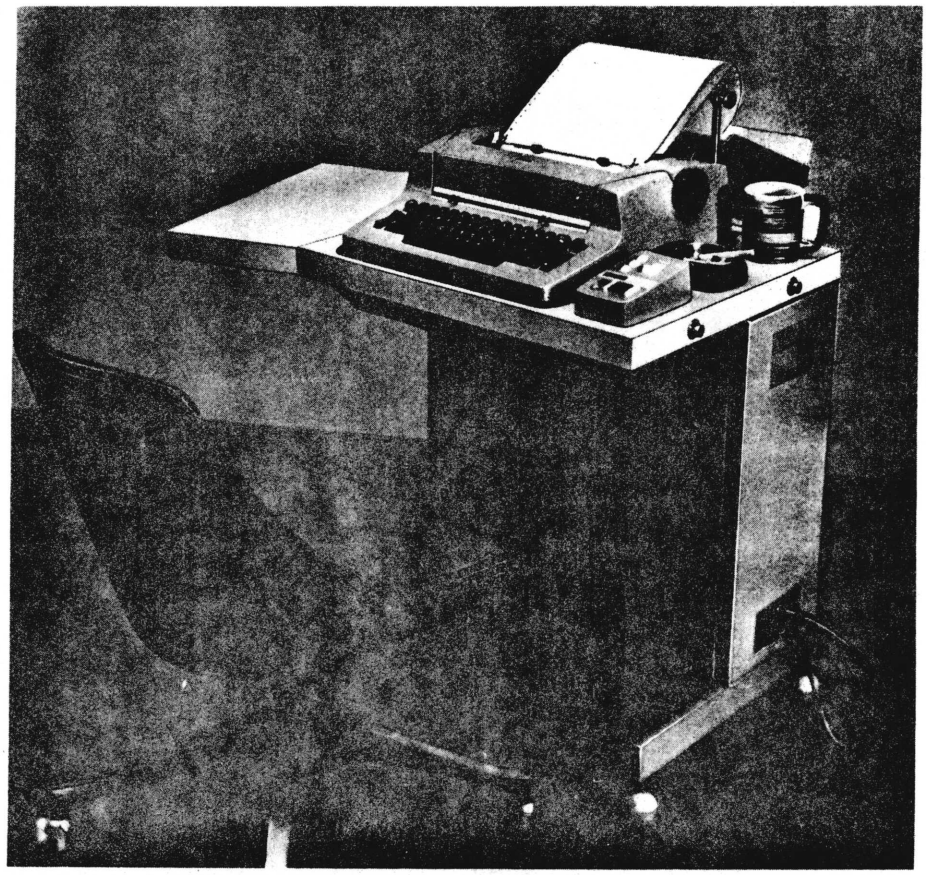
The discussion to follow is divided into three sections: Sec. II, "The JOSS Console," discusses the characteristics of the console and its use; Sec. III, "Arithmetic, Accuracy, and Answers," demonstrates the JOSS arithmetic operations and introduces the method of arranging lines of typed output; and Sec. IV, "Computing with JOSS," illustrates the computational features of JOSS, enabling the beginner to solve simple problems.

The reader is advised to study the entire Primer before approaching a

JOSS console. He should concentrate his reading time on Sec. IV. The initial reading should be followed by two sessions at a JOSS console:

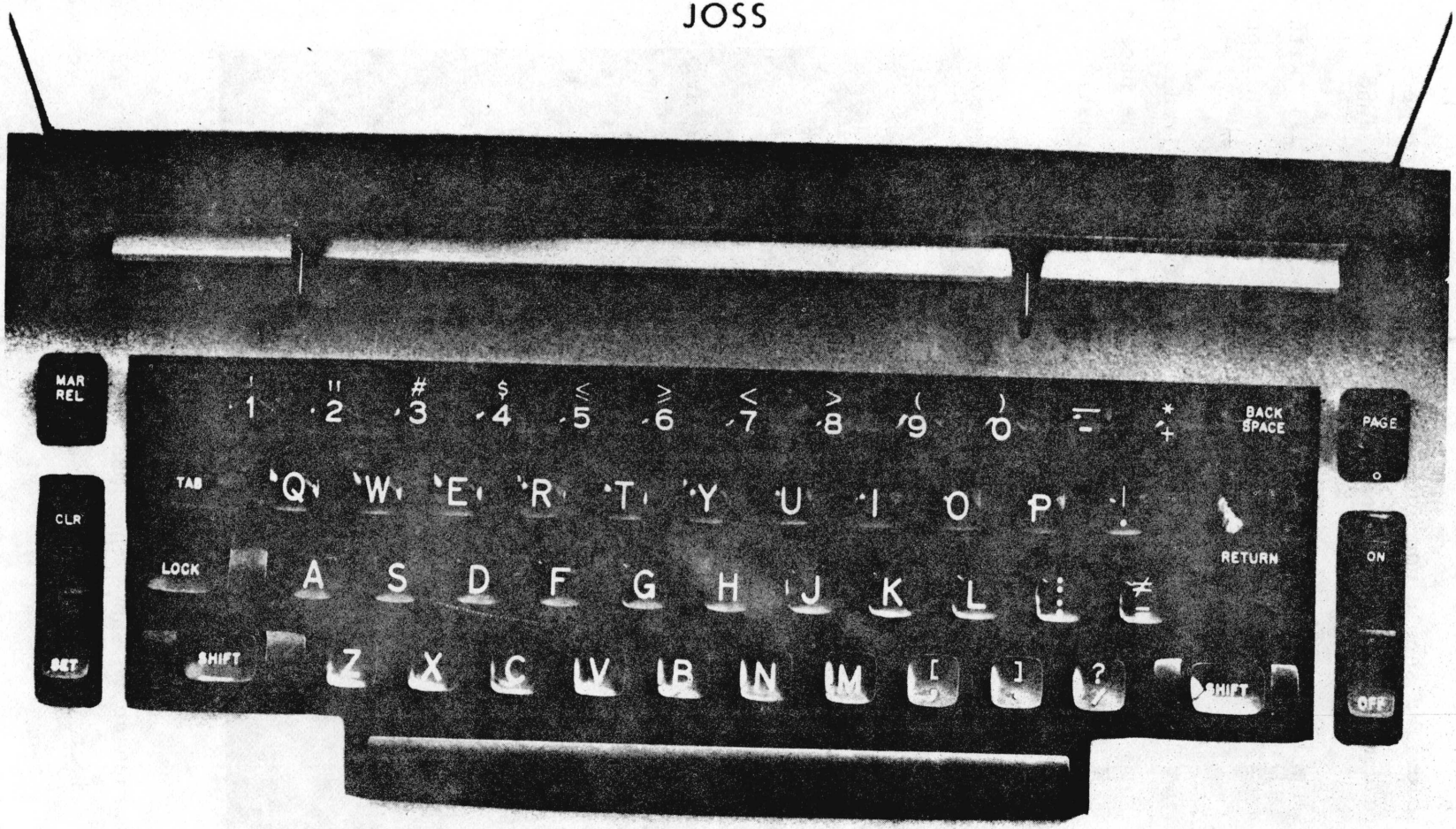
Session 1. Follow the directions of Secs. II and III, with emphasis on *doing* rather than *studying*. You should move quickly through this material.

Session 2. Begin with Sec. III if a review is desired. Then follow the directions of Sec. IV; expect the pace to be slower than for the previous two sections.



The photograph above shows the JOSS console, which consists of a modified IBM Selectric typewriter and a control box. At first, you will operate the typewriter separately, and then will use the control box to connect the console to the computer. At that point you will begin to "converse" with JOSS.

JOSS


MAR REL
CLR
SET
1 2 3 4 5 6 7 8 9 0
BACK SPACE
PAGE
TAB
Q W E R T Y U I O P
RETURN
LOCK
A S D F G H J K L
ON
SHIFT
Z X C V B N M
SHIFT
OFF

II. THE JOSS CONSOLE

Notice the dark blue rocker switch at the right side of the keyboard; press it ON. Now press the large RETURN key on the keyboard, returning the type ball to the left margin. Type the following characters, noticing similarities and differences, and pressing the RETURN key as indicated:

The numeral one (1) and the lower-case letter el (l).
The numeral zero (0) and the letter oh (O).
(JOSS distinguishes between these numerals and letters.)
The period (.) and the centered dot (•).
(These similar-appearing characters have distinct uses in JOSS.)
The plus (+) and the hyphen (-).
The underscore (_), the slash (/), and the vertical bar (|).
The asterisk (*).
The pair of parentheses (()) and the pair of brackets ([]).
The signs denoting less than or equal (≤), greater than or equal (≥), less than (<), greater than (>), equal (=), and not equal (≠).
The space (or number) sign (#), the dollar sign ($), and the double quotation mark (").
The colon (:), the semicolon (;), the comma (,), and the question mark (?).

```
1lOO.•+-_/|*()[]≤≥<>=≠#$":;,?
```

Press RETURN.
The upper- and lower-case letters.

```
QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm
```

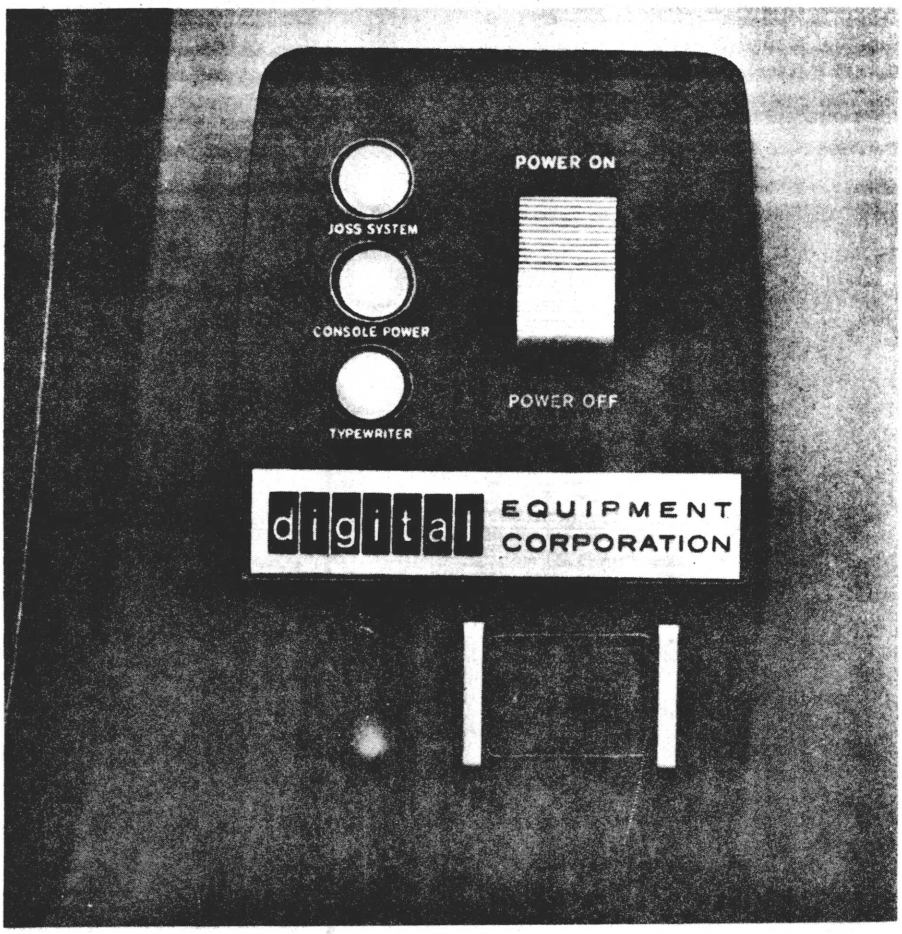
Press RETURN.
The digits.

```
1234567890
```

Press RETURN.

The SHIFT, TAB, and BACK SPACE operate as on a standard typewriter.

Press the PAGE key. Nothing happens because the PAGE key is operative only when you have linked the console to JOSS, and you are about to do this.



Press the white rocker switch on the control box toward POWER ON.

1. The red light will come on.
2. The three white lights (JOSS SYSTEM, CONSOLE POWER, TYPEWRITER) will come on.
3. JOSS will type two lines in black.
4. A soft beep will sound as the red light goes off and the green light comes on.

JOSS has greeted you with

```
JOSS at your service.
Initials please:
```

If everything seems to happen at once and you would like to try it again, press the switch to POWER OFF. Now back to POWER ON, and the above cycle is repeated. The green light and the beep are the signals that it is your turn. You may take as long as you like to respond, since JOSS is occupied with other consoles during this *green state* of your console.

Type your initials. Notice that while you are typing in green, the green light stays on, indicating that you, the user, are in control of the typewriter.

```
JOSS at your service.
Initials please: slm
```

Press the RETURN key to return control to JOSS. The red light is on as JOSS, typing in black, requests your project number. Suppose you type the number *7*.

```
Project number: 7
```

Press the RETURN key and notice the error message from JOSS. This time try the number *6*.

```
Please use a legitimate number: 6
```

Press the RETURN key.

```
If you are learning use "1" :
```

JOSS has identified you as a learner and instructed you to use the learner's identification number *1*. You may use *1* as a project number while you are learning JOSS, but later you should use a legitimate RPN (RAND Project Number).

Instead of typing the number one, try typing the lower-case letter el, and then press RETURN. JOSS repeats its message. As soon as you type a legitimate project number, JOSS will eject to a new page, type a heading, and space down for you to begin. Now type the number one as requested and press RETURN.

```
          If you are learning use "1" : 1
-------------------------------------------------------------------

          8:32  7/20/67   #21  slm  1          [1]
```

The time appears as the first item in the heading at the top of the JOSS page, followed by the date, the number of the communication line connecting your console to JOSS, your initials and project number, and the page number. Occasionally, messages from JOSS (such as Recess 1530-1800 PDT) also will appear in the heading to the right of the page number.

When you use the Primer at a JOSS console

1. Type what you see in green, observing the capitalization and punctuation.
2. At the end of each line of typing, return control to JOSS by pressing the RETURN key; then listen for the beep before you continue.
3. Whenever you have control of the typewriter, use the RETURN key as often as you wish to space a line.

JOSS ignores a line that begins or ends with an asterisk. In the Primer text, a row of green asterisks has been used to indicate logical breakpoints in the discussion.

****

```
          Type users.
                users:  12
```

You have learned the number of users currently sharing JOSS. More important, you have used a basic JOSS verb, *Type*. Your instruction to JOSS is an imperative English sentence, beginning with a capitalized verb and ending with a period. When you pressed the RETURN key, the line was transmitted to JOSS.

```
Type time.
        time:  0832
```

Another version of the *Type* command has produced the current 24-hour Pacific time. This command will now be used to illustrate two ways to correct typing errors:

```
Type tiime.
Eh?
```

The JOSS message *Eh?* usually indicates an error in spelling, spacing, punctuation, or capitalization. One method of error correction is to type over an incorrect character, either with the correct one or with # (which JOSS interprets as a space). To illustrate, type the command once more, repeating the spelling error, but this time don't press RETURN. Instead, backspace to the second *i* in *tiime*, then overtype *me.#* and press RETURN.

```
Type tiime#
        time:  0833
```

A common typing error is not beginning a command with a capital letter; another is forgetting to end a command with a period.

```
type time*
```

The asterisk at the right caused JOSS to ignore the incorrect line.

In summary, you can correct typing errors as follows:

- If you get the error message *Eh?*, you can type the line again, correctly.
- If you discover an error while typing, you can (1) backspace and overtype each wrong character with the correct character or #, finish typing if necessary, and press RETURN; or (2) end what you have typed with an asterisk, press RETURN, and type the line correctly.

III. ARITHMETIC, ACCURACY, AND ANSWERS

The *Type* commands used most often in JOSS are the ones that evaluate expressions and type the results. The simplest evaluations consist of combinations of numeric operations: addition, subtraction, multiplication, division, exponentiation (raising a number to a power), and absolute value (the magnitude of a number).

```
Type 2+2.
         2+2 =        4
```

JOSS has typed the value of the expression that follows the verb *Type*.

The hyphen is the JOSS symbol for subtraction; the period is used as the decimal point as well as punctuation.

```
Type .035-1.2.
    .035-1.2 =       -1.165
```

The centered dot is the JOSS symbol for multiplication; the slash, for division. The asterisk is not only used for deletion, as shown previously, but also for exponentiation. A pair of vertical bars indicates the absolute value of the enclosed expression.

```
Type 7•3.
         7•3 =       21
Type 4/2.
         4/2 =        2
Type 2*3.
         2*3 =        8
Type |4-7|.
       |4-7| =        3
```

A single *Type* command can instruct JOSS to perform several evaluations:

```
Type 3+2, 3+1/2, (3+1)/2, 1/3+2, 1/(3+2).
         3+2 =        5
       3+1/2 =        3.5
     (3+1)/2 =        2
       1/3+2 =        2.33333333
     1/(3+2) =         .2
```

Parentheses and brackets, in matching pairs, are used in JOSS to group expressions.

```
Type 8/4•2, 8/4/2, 8/(4/2), _, 8•4/2.
      8/4•2 =        4
      8/4/2 =        1
    8/(4/2) =        4

      8•4/2 =       16
```

Additions and/or subtractions in a series are executed from left to right, as are multiplications and/or divisions in a series. However, if uncertain how JOSS will perform a particular combination of numeric operations, you may insert parentheses or brackets to clarify the desired order. For example, *8/4/2* may be written *(8/4)/2*. The underscore in the *Type* command above produced a blank line in the typed output.

```
Type _, 2*3*2, 2*(3*2), (2*3)*2.

      2*3*2 =       64
    2*(3*2) =      512
    (2*3)*2 =       64
```

Exponentiation in a series is performed from left to right; parentheses are used to change this order. To improve readability, you may insert blank lines and space between expressions.

```
Type 5(3 - 4.6).
Eh?
```

JOSS is objecting because the centered dot was not used to indicate multiplication.

```
Type 5•(3 - 4.6).
 5•(3 - 4.6) =       -8
```

To summarize the numeric operations in one expression and to illustrate the use of parentheses,

```
Type _, |2 + 8/4•(4 - 7) - 2*3•4|, _, |2 + (8/4)•(4 - 7) - (2*3)•4|.

|2 + 8/4•(4 - 7) - 2*3•4| = 36

|2 + (8/4)•(4 - 7) - (2*3)•4| = 36
```

All the values that JOSS has typed so far have been in a notation called *fixed point*. When the magnitude of a value is large (greater than or

equal to 10^6) or small (less than .001), JOSS types the value in *scientific notation.* Two values that will illustrate scientific notation are (1) the number of seconds in a year, and (2) the volume of a box whose dimensions are .16, .1, and .05.

```
Type _, 365•24•60•60, _, (.16)•(.1)•(.05).

365•24•60•60 =          3.1536•10*7

(.16)•(.1)•(.05) =      8•10*(-4)
```

The number of seconds in a year can be read as 3.1536 times 10 raised to the 7th power, or 31536000. The volume of the box is 8 times 10 raised to the -4th power, or .0008. To illustrate a negative value typed in scientific notation,

```
Type .0001 - .0002.
.0001 - .0002 =       -1•10*(-4)
```

All JOSS values have nine significant digits. The following example illustrates how JOSS rounds values after the ninth significant digit:

```
Type _, 1/3, 3•(1/3), 3•1/3.

      1/3 =          .333333333
  3•(1/3) =          .999999999
    3•1/3 =        1
```

****

The JOSS *form* provides a way for you to describe the desired arrangement of typed output on a page. Each form consists of two lines. The first line identifies the form by an integer, and begins with a capital letter and ends with a colon. The second line describes the desired arrangement of output. Form 1 below describes two numeric fields, separated by four spaces. The underscore specifies typed output in fixed-point notation, each underscore corresponding to one decimal position. The period locates where JOSS will type the decimal point. The first field described in form 1 is to contain one integer digit; the second field is to contain two integer digits and three fraction digits.

```
Form 1:
_    __._____

Type 2 + 2, .035 - 1.2 in form 1.
4    -1.165
```

This *Type* command uses the phrase *in form 1*, instructing JOSS to type the values of the two expressions, arranged as described in form 1. Notice that form 1 allows for two integer digits in the second field, and that JOSS used one of these positions for typing the minus sign.

Underscores in a field description correspond to positions of fixed-point notation; within the fixed-point field, the period locates where JOSS will type the decimal point.

Periods comprising a field description correspond to positions of scientific notation.

Fourteen periods in each field description of form 2 below provide positions for the maximum number of significant digits, which is nine.

```
Form 2:
..............    ..............

Type 1/3, -3•(1/3) in form 2.
 3.33333333-01    -9.99999999-01
```

The fourteen periods also provide positions for a minus sign or a blank, a decimal point, and a signed two-digit power of 10. *-9.99999999-01* can be read as *-.999999999*. Reducing the number of periods in a field description will reduce the number of fraction digits JOSS will type in that field.

The minimum number of periods needed to describe a field of scientific notation is seven.

```
Form 3:
.......    .......

Type 1/3, -3•(1/3) in form 3.
 3.3-01    -1.0 00
```

To meet the requirements of form 3, JOSS rounded the values typed in each field.

To look at all the forms currently defined,

```
Type all forms.

Form 1:
_    __.___

Form 2:
.............    .............

Form 3:
.......    .......
```

To delete specific forms, the JOSS verb *Delete* is used.

```
Delete form 2, form 3.
```

Satisfy yourself that JOSS has deleted forms 2 and 3:

```
Type all forms.

Form 1:
_    __.___
```

Specific forms can be typed.

```
Type form 1.
_    __.___
```

Notice that when you instruct JOSS to *Type all forms*, each form is preceded by its identification. However, JOSS types a specific form without typing its identification. This fact will prove useful in Sec. IV when you will want JOSS to type a heading.

In summary:

- Section III has introduced (1) the *Type* commands that instruct JOSS to evaluate numeric expressions and type the resulting values, and (2) the form, which enables the user to describe the arrangement of typed output on a page.
- JOSS carries all values to nine significant digits.
- The form can specify whether values should be typed in fixed-point or scientific notation; the underscore corresponds to a position of fixed-point notation with the period locating where JOSS types the

decimal point; the period corresponds to a position of scientific notation, and the number of periods ranges from seven to fourteen. JOSS rounds values as necessary to satisfy the requirements of a form.

- The phrase *in form* x, where x represents an integer identification, can be appended to a *Type* command.

Section IV will introduce other JOSS commands to extend your skill in using JOSS as a computing aide.

IV. COMPUTING WITH JOSS

Each of the commands introduced in Sec. III is called a direct command, because as soon as it is transmitted to JOSS, JOSS obeys the command but keeps no record of it. If the user wants JOSS to obey a direct command a second time, he must type the command again.

The following example of a direct command types text within quotation marks:

```
Type "JOSS is an easy-to-use computing aide.".
JOSS is an easy-to-use computing aide.
```

A JOSS command is called an indirect command if it is preceded by a label, called a step number. JOSS does not obey an indirect command, or a sequence of indirect commands, until instructed to do so by an initial direct command. For example:

```
1.1 Type "JOSS is an easy-to-use computing aide.".

Do step 1.1.
JOSS is an easy-to-use computing aide.
```

The JOSS verb *Do*, used here in a direct command, instructs JOSS to perform the indirect command, step 1.1. Repeat the direct command *Do step 1.1* to prove that JOSS has stored step 1.1.

You can use a *Type* command to look at step 1.1.

```
Type step 1.1.
1.1 Type "JOSS is an easy-to-use computing aide.".
```

You can delete step 1.1 with a *Delete* command:

```
Delete step 1.1.
```

You can look at all the steps currently stored:

```
Type all steps.
```

There are no steps for JOSS to type, since you instructed JOSS to delete the only one stored.

Steps are grouped into parts; the integer portion of a step number is called the part number.

```
1.1 Type "JOSS is an".
1.2 Type "easy-to-use".
1.3 Type "computing aide.".
```

These three steps comprise part 1. The part number groups steps into a part (as the steps of part 1), and the step number orders the steps within a part (as steps 1.1, 1.2, and 1.3).

Another *Do* command instructs JOSS to obey part 1; JOSS will perform the steps of part 1 in sequence:

```
Do part 1.
JOSS is an
easy-to-use
computing aide.
```

Steps can be inserted in a part by assigning them appropriate step numbers.

```
1.15 Type "easy-to-learn,".

Type part 1.
1.1 Type "JOSS is an".
1.15 Type "easy-to-learn,".
1.2 Type "easy-to-use".
1.3 Type "computing aide.".
```

JOSS inserted step 1.15 between step 1.1 (or step 1.10) and step 1.2 (or step 1.20). The command *Type part 1* instructs JOSS to type the new part 1.

```
Do part 1.
JOSS is an
easy-to-learn,
easy-to-use
computing aide.
```

The *Do* command instructs JOSS to perform the new part 1.

Further modify part 1:

```
1.3 Do part 2.
1.4 Type "system.".
```

And add a part 2:

```
2.1 Type "open".
2.2 Type "shop".
```

Another *Type* command will type all the parts stored by JOSS:

```
Type all parts.

1.1 Type "JOSS is an".
1.15 Type "easy-to-learn,".
1.2 Type "easy-to-use".
1.3 Do part 2.
1.4 Type "system.".

2.1 Type "open".
2.2 Type "shop".
```

By retyping step 1.3, you instructed JOSS to replace the old step 1.3; and you added step 1.4 and part 2. The new step 1.3 is a *Do* command, instructing JOSS to perform part 2. A *Do* command can be either direct or indirect.

```
Do part 1.
JOSS is an
easy-to-learn,
easy-to-use
open
shop
system.
```

If the *Do* command is direct (*Do part 1.*), JOSS returns to the user. If the *Do* command is indirect (*1.3 Do part 2.*), JOSS returns, after performing the *Do*, to the step that follows the indirect *Do* command (step 1.4).

The *To* command is used only as an indirect command. To see how *To* differs from *Do*, replace step 1.3:

```
1.3 To part 2.

Do part 1.
JOSS is an
easy-to-learn,
easy-to-use
open
shop
```

In performing this new step 1.3, JOSS executed the steps of part 2, but did not return to step 1.4, *Type "system.".* The *To* command instructs JOSS to move to the sequence of steps of part 2.

It is not necessary to begin with part 1.

```
Do part 2.
open
shop
```

JOSS will remind you that the *To* command is meaningful only as an indirect command:

```
To part 2.
Don't give this command directly.
```

In summary:

- Direct commands are typed, transmitted to JOSS, obeyed, but not saved. Indirect commands, those preceded by a step number, are typed and transmitted to JOSS; JOSS stores indirect commands but does not obey them until instructed to do so by an initial direct command.
- Step numbers order the steps within a part; the integer portion of a step number is called the part number and groups the steps into a part. Parts and steps can be typed, deleted, added, or replaced.
- The *Do* command may be either direct or indirect; after carrying out a direct *Do* command, JOSS returns to the user; after obeying an indirect *Do* command, JOSS continues with the step that follows the indirect *Do* command.
- The *To* command may be used only indirectly and instructs JOSS to move to a new sequence of commands.

****

JOSS assigns values to letters in various ways. One way is with the *Set* command, which assigns to the letter on the left of the equals sign the value of the expression on the right.

```
Set a = 3.
Set M = a + 2/3.
Type _, a, M.

          a =        3
          M =        3.66666667
```

The first *Set* command assigned the value 3 to the letter *a*. JOSS used this value of *a* to evaluate the expression *a + 2/3* and then assigned the value of the expression to the letter *M*.

```
Set a = 10.
Type _, a, M.

          a =       10
          M =        3.66666667
```

This *Set* command instructed JOSS to assign the value 10 to the letter *a*. However, the letter *M* has the same value as before, because JOSS has not been instructed to evaluate *M* for the new value of *a*.

```
Delete all parts.
1.1 Set M = a + 2/3.
1.2 Type _, a, M.
```

Step 1.1 assigns a value to *M* for a current value of *a*. Step 1.2 spaces a line and types the current value of *a* and the corresponding value of *M*.

Instruct JOSS to perform these steps for a specific value of *a*, say, 5.

```
Do part 1 for a = 5.

          a =        5
          M =        5.66666667
```

A *Do* command can instruct JOSS to perform a part for either a single value of a letter, as above, or a range of values of a letter:

```
Do part 1 for a = 1, 1.5, 2, 2.5, 3.

          a =        1
          M =        1.66666667

          a =        1.5
          M =        2.16666667
```

```
a =        2
M =        2.66666667

a =        2.5
M =        3.16666667

a =        3
M =        3.66666667
```

The JOSS form, introduced in Sec. III, can be used to type several values on a single line. For reference, type first:

```
Type part 1.
1.1 Set M = a + 2/3.
1.2 Type _, a, M.
```

Then replace step 1.2:

```
1.2 Type a, M in form 1.
```

Define form 1 to describe fixed-point fields for a and M, separated by four spaces. Provide for an integer position and a fraction position for a, and an integer position and eight fraction positions for M.

```
Form 1:
_._    _._________
```

Repeat the previous direct *Do* command:

```
Do part 1 for a = 1, 1.5, 2, 2.5, 3.
1.0    1.66666667
1.5    2.16666667
2.0    2.66666667
2.5    3.16666667
3.0    3.66666667
```

The range of values of a in this *Do* command happens to be small. Another way to express this same range, and a convenient one when the range is large, is as follows:

```
Do part 1 for a = 1(.5)3.
1.0    1.66666667
1.5    2.16666667
2.0    2.66666667
2.5    3.16666667
3.0    3.66666667
```

The values of a begin at 1, increase in steps of .5, and end at 3.

Suppose the values of a range from -2 to 0 at steps of .5.

```
Do part 1 for a = -2(.5)0.
Error at step 1.2:  I can't express value in your form.
```

To find the error, type the current values of a and M:

```
Type a, M.
          a =        -2
          M =        -1.33333333
```

As defined, form 1 allows only one integer position each for a and M. This is sufficient as long as each letter has a positive value less than 10. To allow a position for JOSS to type the minus sign in each field, redefine form 1:

```
Form 1:
_._    _._______
```

After attempting to perform step 1.2, JOSS typed an error message and returned control to you. To instruct JOSS to resume with step 1.2, type the direct command *Go*.

```
Go.
-2.0    -1.33333333
-1.5     -.83333333
-1.0     -.33333333
 -.5      .16666667
  .0      .66666667
```

A heading will make this table of values more readable. For ease in designing the heading, type form 1 as a guide and then define form 2.

```
Type form 1.
_._    _._______
Form 2:
  a         M
```

Add part 2 to type the heading described in form 2 and to *Do* part 1 for the range of values of a.

```
2.1 Type form 2, _.
2.2 Do part 1 for a = -2(.5)0.
```

Step 2.1 instructs JOSS to type the definition of form 2 (without typing its identification) and space a line. Step 2.2 instructs JOSS to perform part 1 for each value of *a* in the range.

Type parts 1 and 2 for reference:

```
Type part 1, part 2.
1.1 Set M = a + 2/3.
1.2 Type a, M in form 1.

2.1 Type form 2, _.
2.2 Do part 1 for a = -2(.5)0.
```

A direct *Do* command will initiate the procedure:

```
Do part 2.
  a          M

-2.0     -1.33333333
-1.5      -.83333333
-1.0      -.33333333
 -.5       .16666667
  .0       .66666667
```

JOSS can type a line containing both text and values. Suppose you want to identify a set of typed output by case number. Define a form to include the word *Case* and a description of a numeric field with two integer positions for the case number:

```
Form 3:
Case __
```

Add the following steps to the beginning of part 2:

```
2.01 Page.
2.02 Type c in form 3.
2.03 Line.
```

Steps 2.01, 2.02, and 2.03 instruct JOSS to eject to a new page, to type the current value for case number arranged as described in form 3, and to space a line.

Suppose the case number is *1*.

```
Do part 2 for c = 1.
```

```
        9:02  7/20/67   #13  slm  1          [4]



        Case  1

          a          M

        -2.0     -1.33333333
        -1.5      -.83333333
        -1.0      -.33333333
         -.5       .16666667
          .0       .66666667
```

In summary:

- The *Set* command, used directly or indirectly, has assigned to a letter the value of an expression.
- A *for* phrase, specifying the range of a letter, can be appended to a *Do* command. The range of the letter can be expressed as a list of specific values, and as a beginning value, an increment, and a final value.
- When JOSS cannot express a value in a form you have defined, it is helpful to type the current values of pertinent letters, and correct the form definition as necessary. To continue the computation, type *Go*.
- JOSS can type a heading for a table of values when instructed to type a specific form. A form can combine text with a description of the arrangement of values.

****

Letters that have subscripts, such as b_1 and b_2, are typed in JOSS as *b(1)* and *b(2)*.

```
        Delete all.
```

Assign values to b_1 and b_2:

```
/ Set b(1) = 16.
  Set b(2) = 1/2.
```

All the assigned values of a subscripted letter can be typed with one command:

```
Type b.
        b(1) =       16
        b(2) =         .5
```

Or specific values can be typed:

```
Type b(1), b(2).
        b(1) =       16
        b(2) =         .5
```

Suppose $b_{1,1}$ is to be assigned the value 34.

```
Set b(1,1) = 34.
```

Then type all the assigned values of b as before:

```
Type b.
      b(1,1) =       34
```

JOSS typed only the value most recently assigned, and not the values of b_1 and b_2 that were previously assigned. Instruct JOSS to type the value of b_2:

```
Type b(2).
b(2) = ???
```

JOSS no longer has the value of b_2 stored. The command *Set b(1,1) = 34* not only assigned a value to $b_{1,1}$ but told JOSS that the letter b is now defined to have two subscripts; thus, the values of b with one subscript are deleted.

```
Delete all.
```

The *Demand* command is used indirectly to assist the user in assigning values to letters:

```
1.1 Demand x.
```

In the performance of step 1.1, JOSS will type the letter x, and wait for you to type a value.

```
Do step 1.1.
x = 9
```

The *Demand* command is most useful when several values are to be assigned:

```
1.1 Demand b(i).
Do part 1 for i = 1, 2.
b(1) = 16
b(2) = 1/2
```

The range of i in the *Do* command has been expressed as a list of specific values.

```
1.1 Demand c(i).
Do part 1 for i = 1(1)4.
c(1) = 3.5
c(2) = -.43
c(3) = 6
c(4) = .053
```

And above, the range of i in the *Do* command has been expressed as a beginning value, an increment, and an end value.

Now look at all the values that JOSS has stored:

```
Type all values.

     i =      4
     x =      9

  b(1) =     16
  b(2) =     .5

  c(1) =     3.5
  c(2) =     -.43
  c(3) =     6
  c(4) =     .053
```

Consider the sum of the values of c_i for $i = 1, 2, 3,$ and 4. Mathematically, the sum can be written as $\sum_{i=1}^{4} c_i$. One way to calculate the sum in JOSS is as follows:

```
Type c(1) + c(2) + c(3) + c(4).
c(1) + c(2) + c(3) + c(4) = 9.123
```

Another way is to use the JOSS *sum* function:

```
Type sum[c(1), c(2), c(3), c(4)].
sum[c(1), c(2), c(3), c(4)] = 9.123
```

The sum function is especially convenient when there are many values to be added. It is then used as follows:

```
Type sum[i = 1(1)4: c(i)].
sum[i = 1(1)4: c(i)] = 9.123
```

The colon inside the brackets separates the range of values of i (on the left) and the expression whose values are to be summed over the range.

In summary:

- To type a subscripted letter, type the letter followed by a pair of parentheses enclosing the subscript, or subscripts.
- The indirect *Demand* command assists the user in assigning a value to a letter and is most helpful when there are many values to be assigned.
- The sum function instructs JOSS to add the values given as the argument for the function. The argument may be a list of values or an expression to be evaluated for each value of an index.

****

```
Set n = 5.
Type _, n, prod[k = 1(1)n: k].

          n =        5
prod[k = 1(1)n: k] = 120
```

In this use of the *prod* function, JOSS is instructed to evaluate the product of the values of k in the range *1(1)n*. Since n is assigned the value 5, the product can be evaluated by JOSS.

The product above is dependent for its value on the value of n. JOSS permits you to define a formula that expresses this dependence.

```
Let P(n) = prod[k = 1(1)n: k].
```

The JOSS verb *Let* instructs JOSS to define a formula. JOSS will refer to the definition whenever the formula name is used in an expression. In the above example, you have defined *P(n)* to be *factorial n*.

```
Type _, P(1), P(2), P(5), P(20).

        P(1) =         1
        P(2) =         2
        P(5) =       120
       P(20) =         2.432902•10*18

Type P(0).
Error in formula P:  Illegal set of values for iteration.
```

The JOSS error message reminds you that when $n = 0$, the range of k in the formula P is meaningless. A *conditional command* can be used to include the case of *factorial 0* (which is defined to equal 1).

The following steps will evaluate and type *P(n)* for any positive n, and also for $n = 0$.

```
1.1 To part 2 if n = 0.
1.2 Type _, n, P(n).

2.1 Line.
2.2 Type "Special case:  P(0) = 1".
```

Step 1.1 contains the conditional clause *if* $n = 0$. In performing step 1.1, JOSS will first examine this condition; if the condition is satisfied (that is, if the current value of n is zero), JOSS will obey the imperative *To part 2*. If the condition fails (if n is not zero), JOSS will move to step 1.2, the step following the conditional step.

```
Do part 1 for n = 5, 0.

          n =         5
       P(n) =       120

Special case:  P(0) = 1
```

If step 1.1 were a *Do* command instead of a *To* command, JOSS would obey step 1.2 regardless of the value of n. For, if $n = 0$, the condition is satisfied, JOSS performs part 2, and returns to the step following the *Do* command, step 1.2. If n is not zero, the condition fails and JOSS moves at once to step 1.2.

To improve the appearance of the typed output, modify parts 1 and 2:

```
1.2 Type n, P(n) in form 1.
2.1 Type n, 1 in form 1.

Delete step 2.2.
```

Now look at parts 1 and 2:

```
Type all parts.

1.1 To part 2 if n = 0.
1.2 Type n, P(n) in form 1.

2.1 Type n, 1 in form 1.
```

Allow one integer position for n and three for $P(n)$ in form 1:

```
Form 1:

-   ---

Do part 1 for n = 0(1)7.
0      1
1      1
2      2
3      6
4     24
5    120
6    720
Error at step 1.2:  I can't express value in your form.
```

To find the error, instruct JOSS to type the current values of n and $P(n)$:

```
Type n, P(n).
          n =       7
       P(n) =    5040
```

Form 1 must be redefined for a four-digit $P(n)$:

```
Form 1:

-   ----
```

JOSS is waiting to resume at step 1.2.

```
Go.
7    5040
```

To illustrate further the use of a conditional command, two JOSS functions are introduced. The *ip* function instructs JOSS to use the integer

part of a value, and the *fp* function instructs JOSS to use the fraction part of a value:

```
Type _, ip(3.64), fp(3.64).

    ip(3.64) =        3
    fp(3.64) =         .64
```

If *fp(n)* is not zero, that is, if *n* has a fraction part, then *n* is not an integer. This fact can be used to add to the example of *factorial n* the condition that *n* be an integer. Modify part 1, add a part 3, and then look at parts 1, 2, and 3:

```
1.15 To part 3 if fp(n) ≠ 0.
3.1 Type n in form 2.

Type all parts.

1.1 To part 2 if n = 0.
1.15 To part 3 if fp(n) ≠ 0.
1.2 Type n, P(n) in form 1.

2.1 Type n, 1 in form 1.

3.1 Type n in form 2.
```

Define form 2, allowing one integer and two fraction positions for *n*:

```
Form 2:
non-integer n = _.__
```

Add part 4 in order to type a heading and *Do* part 1 for a range of values of *n*:

```
4.1 Type form 3, _.
4.2 Do part 1 for n = 0(1)3, 3.64, 4.
```

Type form 1 as a guide and then define form 3:

```
Type form 1.

Form 3:
n    P(n)
```

Notice that the range of values of *n* in step 4.2 combines both ways of expressing a range.

```
Do part 4.
n    P(n)

0       1
1       1
2       2
3       6
non-integer n = 3.64
4      24
```

Steps 1.1 and 1.15 are conditional *To* commands. If the condition of step 1.1 is satisfied, JOSS moves to part 2; if not, JOSS moves to step 1.15. If the condition of step 1.15 is satisfied, JOSS moves to part 3; if not, JOSS moves to step 1.2. Part 4 types the heading and controls the iteration over the range of n. The direct *Do* command initiated the process.

In summary:

- These last examples have introduced the functions *prod*, *ip*, and *fp*; and the *Let* command, which defines a formula.
- A conditional clause was appended to a *To* command (and can be appended to any JOSS command).

****

Three important JOSS functions are *sqrt*, where *sqrt(x)* is the square root of x; *log*, where *log(x)* is the natural logarithm of x; and *exp*, where *exp(x)* is e^x, the exponential of x. Consider a part 1 that illustrates the range of their arguments.

```
Delete all.

1.1 Type x, sqrt(x), log(x), exp(x).

Do part 1 for x = -10.
Error at step 1.1:  I have a negative argument for sqrt.

Set x = 0.
Go.
Error at step 1.1:  I have an argument ≤ 0 for log.
```

```
Set x = 5.
Go.
          x =         5
     sqrt(x) =         2.23606798
      log(x) =         1.60943791
      exp(x) =       148.413159
```

The argument of the square root must be positive or zero; for the logarithm, the argument must be positive.

The following steps instruct JOSS to type a table of x, *sqrt(x)*, *log(x)*, *exp(x)*:

```
1.1 Type form 1.
1.2 Line.
1.3 Type x, sqrt(x), log(x), exp(x) in form 2.
1.4 Set x = x + 20.
1.5 To step 1.3.
```

Form 2 below describes four fields separated by four spaces. The first three fields are to be in fixed-point notation; field 1 is to have three integer positions; field 2, two integer positions and eight fraction positions; field 3, one integer position and eight fraction positions. Field 4 is to be in scientific notation with the maximum number of significant digits.

```
Form 2:
___ __._______   _._______   ............
Form 1:
 x       sqrt(x)        log(x)          exp(x)
```

Step 1.4 instructs JOSS to assign to x the current value of x increased by 20; step 1.5 returns JOSS to step 1.3 to calculate table entries in an endless iteration.

Before typing a *Do* command (which not only will initiate the program but also will set x to an initial value), find the INTERRUPT button on the control box. When the second entry of the table is being typed (when $x = 30$), press the INTERRUPT button. The INTERRUPT light will come on at once, but a few lines may type before JOSS types a message and halts. (JOSS is bringing you up-to-date on calculations already performed.)

```
Do part 1 for x = 10.
 x       sqrt(x)        log(x)          exp(x)

 10     3.16227766    2.30258509     2.20264658 04
 30     5.47722558    3.40119738     1.06864746 13
 50     7.07106781    3.91202301     5.18470553 21
 70     8.36660027    4.24849524     2.51543867 30
I'm at step 1.4.
```

The INTERRUPT button may be used at any time to change from *red state* (JOSS in control) to *green state* (you in control). In this case, the purpose of the INTERRUPT might be to change the size of the increment of x. Change step 1.4 to increment x by 50 instead of 20; then type *Go* to continue:

```
1.4 Set x = x + 50.
Go.
120    10.95445120    4.78749174     1.30418088 52
170    13.03840480    5.13579844     6.76179382 73
220    14.83239700    5.39362755     3.50579098 95
Error at step 1.3:  I have an overflow.
```

JOSS is waiting at step 1.3, presumably with a value of *exp(x)* greater than the maximum value that JOSS can handle. You might explore this maximum value by trying the following: Assign x the value 220. Replace step 1.4 with a step that increments x by a value smaller than 20, say, 5. Then type *Go* to continue.

Once an appropriate range for x is decided upon, a simpler program to compute the above table is as follows:

```
Delete part 1.
1.1 Type form 1.
1.2 Line.
1.3 Do part 2 for x = 10(20)220.

2.1 Type x, sqrt(x), log(x), exp(x) in form 2.
```

In summary:

- The *sqrt* function computes the square root of an argument greater than or equal to zero. The *log* function computes the natural logarithm of an argument greater than zero. The *exp* function computes

the exponential of an argument, such as e^x, where the maximum value of x that JOSS allows is 100•log(10), or 230.258509.

- JOSS types an error message if a restriction on an argument is violated and indicates at what step the violation occurred. Type *Go* to continue at that step after the error is corrected.
- The INTERRUPT button is used to halt the computation. The INTERRUPT light comes on at once, but JOSS may continue typing for a few lines. Type *Go* to continue from the step indicated by JOSS.

****

Other useful functions are available in JOSS.

```
Type min(10, 1, 100).
min(10, 1, 100) =     1
```

The *min* function selects the lowest value in its argument list.

```
Type min[17.5, 1/2, -5, sqrt(2)].
min[17.5, 1/2, -5, sqrt(2)] = -5
```

The selection of the lowest value may involve the evaluation of an expression, such as *sqrt(2)*. The companion *max* function selects the highest value in its argument list.

```
Type max(3*2, 9.2, 9•2).
max(3*2, 9.2, 9•2) = 18
```

Instead of finding the minimum value in a list of expressions, the *min* function can calculate the minimum value that an expression assumes for a series of values of an index:

```
Type min[x = 1(.1)5: x*2 - 5•x + 6].
min[x = 1(.1)5: x*2 - 5•x + 6] = -.25
```

JOSS evaluates the expression *x*2 - 5•x + 6* for each value of x in the range *x = 1(.1)5*, and finds that -.25 is the minimum value the expression assumes for these values of x.

Two useful trigonometric functions in JOSS are *sin* and *cos*, which evaluate the sine and cosine of an angle in radians:

```
Type _, sin(0), cos(0), sin(3.14159265), cos(3.14159265).

      sin(0) =         0
      cos(0) =         1
sin(3.14159265) =      3.8•10*(-9)
cos(3.14159265) =     -1
```

The sine of 3.14159265 radians (the value of π to nine significant digits) equals 3.8 times 10 raised to the -9th power, or .0000000038; the fact that the sine is not exactly zero is due in part to the inexactness of the input value of π, and in part to JOSS's method for computing the sine.

To compute the sine or cosine of an angle given in degrees, rather than radians, assign *d* the value $\pi/180$, the factor for converting degrees to radians:

```
Set d = 3.14159265/180.
Type d.
           d =          .0174532925
```

Instruct JOSS to type the sine and cosine of 30 degrees:

```
Type _, sin(30•d), cos(30•d).

   sin(30•d) =          .499999999
   cos(30•d) =          .866025404
```

****

The *Stop* command, used indirectly, halts a computational procedure. It is used when a temporary halt is desired in a sequence of steps, perhaps to allow time to define a formula or a form, or to set tab stops. *Go* is typed to continue.

```
Delete all.
1.1 Type "After the stop message, type Go.".
1.2 Stop.
1.3 Type "JOSS is easy to learn and easy to use.".

Do part 1.
After the stop message, type Go.
Stopped by step 1.2.
Go.
JOSS is easy to learn and easy to use.
```

Press the PAGE key. After the page ejects, tear off the record of your conversation with JOSS. Then press the white rocker switch on the control box to POWER OFF.

****

At this point in your acquaintance with JOSS, you have familiarized yourself with the look and the sound; with the answers, accuracy, and speed; and with the power to compute and to communicate. You are ready to extend your knowledge of JOSS according to your needs.

The remaining pages of the Primer include a list of JOSS commands and functions. You have used most of the JOSS verbs: *Type*, *Delete*, *Set*, *Let*, *Do*, *To*, *Demand*, *Go*, *Stop*, and the line-spacing commands *Page* and *Line*. The command list contains uses of these familiar verbs you will want to try.

Some commands will be new to you and are not self-explanatory; the use of some functions may not be clear from their definitions alone. You may want to read about them, along with other JOSS features not covered in the Primer. Following the list of commands and functions, there is a list of documents selected from RAND publications on JOSS, with comments to help you plan your reading. Depending on your particular interests, you can learn the use of the *Done* command, or how to solve an engineering problem, or why JOSS designers chose the laconic *Eh?* as an error message.

If you need assistance in using JOSS, consult the JOSS Representative in your department.

However, another way to learn if "such-and-such" will work in JOSS is to try it and see.

L = letter
S = subscripted letter
P = proposition
F = formula

int = expression with integer value
num = expression with numerical value
rng = range of values of a letter, such as 1(.1)3,3.64,4

- An expression, e.g., 3+sqrt(2), has a numerical value. A proposition, e.g., a≤b, has a truth value (true or false).
- An if *P* clause can be appended to any command except a short Set.
- A conditional expression may be used wherever an expression is allowed; e.g., define f(x) by a conditional expression: Let f(x)=(x<0:0;0≤x<1:x*2;1) defines f(x) to be 0 for x<0, x*2 for 0≤x<1, and otherwise 1.

JOSS COMMAND LIST

TYPE COMMANDS

Type *var*.
Type *S*.
Type *S(int,int)*.
Type *L*.
Type "any text".
Type *P*.
Type form *int*.
Type step *num*.
Type part *int*.
Type formula *F*.
Type *F(num)*.
Type *F(P)*.
Type all steps.
Type all parts.
Type all formulas.
Type all forms.
Type all values.
Type all.
Type time.
Type timer.
Type size.
Type users.
Type item-list.

- Several individual Type commands may be combined in one, except for Type "any text", and Type item-list.
- in form *int* may be appended to Type commands that specify individual values.

SET COMMANDS

Set *L=num*.
Set *L=P*.
Set *S(int,int)=num*.
Set *S(int,int)=P*.

SHORT SET COMMANDS

L=num
L=P
S(int,int)=num
S(int,int)=P

DELETE COMMANDS

Delete *L*.
Delete *S*.
Delete *S(int,int)*.
Delete form *int*.
Delete step *num*.
Delete part *int*.
Delete formula *F*.
Delete all steps.
Delete all parts.
Delete all formulas.
Delete all forms.
Delete all values.
Delete all.

- Several individual Delete commands may be combined in one, such as Delete *L*, form *int*, all parts.

LET COMMANDS

Let *F=num*.
Let *F=P*.
Let *F(L)=num*.
Let *F(L)=P*.

DO COMMANDS

Do step *num*.
Do step *num*, *int* times.
Do step *num* for *L=rng*.
Do part *int*.
Do part *int*, *int* times.
Do part *int* for *L=rng*.

TO COMMANDS

To step *num*.
To part *int*.

DEMAND COMMANDS

Demand *L*.
Demand *S(int,int)*.
Demand *L* as "any text".
Demand *S(int,int)* as "any text".

FILE COMMANDS

Use file *int (ident)*.
File ... as item *int (code)*.
Recall item *int (code)*.
Discard item *int (code)*.

- file *int* and *ident* are assigned by the Computer Sciences Department.
- item *int* and *code* are assigned by the user at time of filing. 1≤*int*≤25. *code*, if used, ≤5 letters and numbers.
- "..." stands for any combination of elements that can be deleted by a Delete command.

SINGLE-WORD COMMANDS

Page.
Line.
Go.
Stop.
Done.
Quit.
Cancel.

PARENTHETICAL COMMANDS

Cancel or any Do command may be enclosed in parentheses.

SPECIAL COMMANDS

Reset timer.
Let *S* be sparse.

JOSS FUNCTION LIST

ip(x) = integer portion of x
fp(x) = fraction portion of x
dp(x) = digit portion of x
xp(x) = exponent portion of x
sgn(x) = -1 if x<0, 0 if x=0, +1 if x>0

sqrt(x) = square root of x for x≥0
log(x) = natural logarithm of x for x>0
exp(x) = e^x

sin(x) = sine of x, x in radians, |x|<100
cos(x) = cosine of x, x in radians, |x|<100
arg(x,y) = angle in radians formed by the positive x-axis and the line from the origin to (x,y)

sum(x,y,z) or sum[i=*rng:num*] = sum of values in argument list
prod(x,y,z) or prod[i=*rng:num*] = product of values in argument list

min(x,y,z) or min[i=*rng:num*] = minimum of values in argument list
max(x,y,z) or max[i=*rng:num*] = maximum of values in argument list

conj(p,q,r) or conj[i=*rng:P*] = true if and only if *P* is true for all i
disj(p,q,r) or disj[i=*rng:P*] = true if any *P* is true

first[i=*rng:P*] = first i for which *P* is true

tv(*P*) = the truth value of *P*; 0 if *P* is false, 1 if *P* is true

- *num* or *P* is evaluated for each value of *i* in the range *rng*; *i* usually appears in *num* or *P*.

Appendix B

SUGGESTED JOSS READING LIST

Shaw, J. C., *JOSS: A Designer's View of an Experimental On-line Computing System*, The RAND Corporation, P-2922, August 1964, 33 pp. An introduction to JOSS by the originator of the system, with discussion of certain design considerations. Most of the material is applicable to the current system and of interest to the computer novice or programmer.

Baker, C. L., *JOSS: Introduction to a Helpful Assistant*, The RAND Corporation, RM-5058-PR, July 1966, 50 pp. Based on a paper delivered before a data-processing conference. Photographs and examples illustrate basic JOSS features. The memorandum is designed to show JOSS as an easily used computing aide.

Gimble, E. P., *JOSS: Problem Solving for Engineers*, The RAND Corporation, RM-5322-PR, May 1967, 85 pp. First appeared in August 1966 under the title *Problem-Solving with JOSS*, by E. P. Gimble, engineer at McClellan Air Force Base, California. The present version was prepared by C. L. Baker of RAND and presents a comprehensive survey of the JOSS language, with examples and a list of commands and functions.

Bryan, G. E., and J. W. Smith, *JOSS Language (Aperçu and Précis, Pocket Précis, Poster Précis)*, The RAND Corporation, RM-5377-PR, August 1967. Brief reference summaries of the JOSS language in three formats: book-size (23 pp.), pocket-size (19 pp.), and poster-size (1 p.).

Bryan, G. E., and E. W. Paxson, *The JOSS Notebook*, The RAND Corporation, RM-5367-PR, August 1967, 138 pp. Notes on JOSS usage based on a mathematician's efforts to learn the system. Heavy use is made of annotated examples from JOSS.

Bryan, G. E., *JOSS: Introduction to the System Implementation*, The RAND Corporation, P-3486, November 1966, 19 pp. An overview of the JOSS system, including discussion of the concept, capabilities, hardware, software, language, and usage of JOSS, together with some programming details on the supervisor.

In preparation

Marks, S. L., *The JOSS User's Reference Manual*, The RAND Corporation, RM-5219-PR. An indexed presentation of all JOSS features from the user's point of view. The Reference Manual is a companion document to the Primer and intended for the user with some JOSS experience.