JOSS: INTRODUCTION TO THE SYSTEM IMPLEMENTATION

G. E. Bryan

November 1966

P-3486

Presented at the Fall Symposium of Digital Equipment Computer Users Society (DECUS), Lawrence Radiation Laboratory, Berkeley, California, November 4 & 5, 1966.

JOSS:* INTRODUCTION TO THE SYSTEM IMPLEMENTATION

G. E. Bryan
The RAND Corporation
Santa Monica, California

Abstract

JOSS is a time-shared computer system that provides for the solution of numerical problems via an easily learned language at remote typewriter consoles. The PDP-6 hardware used to implement JOSS consists of 32,000 words of 1.75μsec core memory, a 1-million-word 4μsec drum, a 6-million-word discfile, and various peripheral devices. A special data relocation mode for memory references has been added to facilitate interpretation of JOSS programs. The JOSS consoles, built around a Selectric I/O typewriter, were specially manufactured to RAND specifications. Features include full duplex signaling, line parity checking, a page eject mechanism, and several buttons and lights to control and report console status. The stand-alone JOSS software consists of the JOSS language interpreter and its arithmetic subroutines, a monitor for user scheduling and resource allocation, and I/O routines for the disc, drum, consoles, and other peripheral devices. JOSS service is currently available to nearly 500 users through 34 consoles, six of which are remote to RAND operating over both private and dataphone lines.

The JOSS System

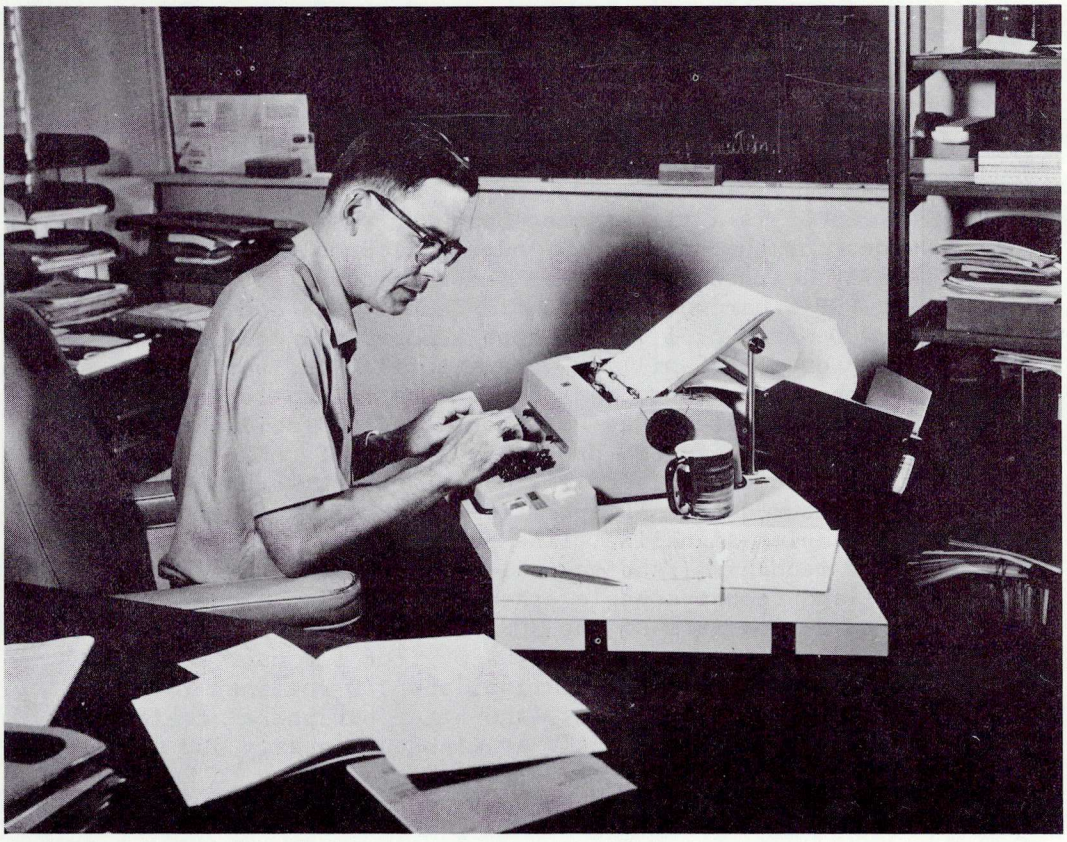
JOSS is a computer system that allows a user** direct interaction with a powerful computer through a familiar device (typewriter) and in a familiar language (arithmetic or algebra). The machine that houses JOSS is dedicated exclusively to that task 24 hours a day, 7 days a week. No background tasks are performed. In contrast with project MAC, SDC, and other "general purpose" time-sharing systems, JOSS has been designed for the casual user and applications programmer rather than for the systems programmer.

In order to make such a service available to many people at an economic price, the system is time shared; that is, simultaneous and noninterfering service is supplied to a number of users at their individual typewriter consoles. The primary advantage of JOSS is its ability to provide fast solutions to reasonably complex problems with a minimum of administrative delay. The user must specify all data relevant to describing his problem and the algorithm for its solution, but need only provide a minimum of detail regarding how his problem is to be solved on the available hardware. The JOSS user has at his command a machine of about the power of a

---

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

**Usually scientists and engineers, but also secretaries and kids.



4K 704, with the additional bonus of a language interpreter.

JOSS is a problem-solving tool that the user can apply to small- and medium-size problems with a minimum investment on his part in learning its use.

History

Work was first started on JOSS in 1960. The system was implemented on the JOHNNIAC computer (now retired) by J. C. Shaw, to whom goes the bulk of the credit for both design and construction. The system was partially operational in early 1963 and fully operational with eight consoles in January 1964--no small accomplishment considering the 4000 word memory and doddering years of JOHNNIAC.

However, JOSS was impressive enough on the few days JOHNNIAC felt well to substantiate the acquisition of a new computer and the creation of a well-supported project to build a second JOSS.* As one user quipped, "It's better than beer--we're hooked."

Introduction of the new JOSS in formal operation took place in mid-February 1966, although selected users had been contributing to system debugging since its first coherent words in early November 1965. Implementation on a large modern computer gave the new JOSS about an order-of-magnitude more capability than its predecessor--30 times as fast, 5 times the storage per user, 4 times as many

---

*The name JOSS still stands for JOHNNIAC Open Shop System in spite of the fact that JOHNNIAC now resides in the Los Angeles County Museum. It has been suggested that JOSS should now be interpreted as JOSS Open Shop System.

consoles, 50 percent faster consoles, room for several powerful new language features, and, in addition, spare capacity. We believe that well over 100 consoles can be handled within the present configuration without service degradation.

Scope and Intent

JOSS is commonly characterized as a tool for the solution of small numerical problems--and so it is. But the word "small" would be better rendered as "not large." To say that JOSS is a good desk calculator is a substantial understatement, although it is often used effectively for that purpose. A list of the limitations of JOSS is perhaps more instructive than one of its capabilities. As a data retrieval system, it is poor; no provision exists for handling large files of information; it can't tackle very large problems (by today's standards); the 40-page FORTRAN code is unfeasible; and the compact but long-running program, say, 2 hours on a 7094, although possible, would be extremely tedious--perhaps as much as 60 hours.

Together with other so-called, time-sharing systems, JOSS enjoys the substantial advantages of the interactive environment. The user is able to approach his console with perhaps only a partially formed idea of his problem and to come away in a few minutes or hours with the answer. This method is estimated to be about ten times faster than the usual problem-inception-to-problem-solution approach to a computer. It is successful enough that many problems that weren't worth the effort before are now being solved.

JOSS differs from the general-purpose interactive time-shared systems in that its operation is simple and its goals are limited. What little information the casual user does have to remember about the system's operation can usually be brought back to mind by experimenting at the console without recourse to a manual of operation or to the help of a system "expert." For this ease of use, JOSS gives up many general-purpose features, but retains a large complement of casual users. As Willis Ware has said, "For a certain class of problems, at least, the programmer as the middleman between the problem and the machine is no longer needed."

Hardware

The essential hardware components of the PDP-6 computer system used for JOSS are outlined in Fig. 1.

The arithmetic processor, a word-organized multiaccumulator-index register machine, is provided with 32,000 words of high-speed core memory in two independently accessible 16,000-word boxes. JOSS uses one of the boxes, the low-addressed one, for the JOSS software, and the other to hold the user programs and data.

The processor contains a relocation register whose contents are added to memory references if certain conditions are met. The RAND PDP-6 has been modified so that the appearance of bit 20 in the address satisfies the requirement. The contents of the relocation register are set to the base address of the locations in memory that contain the user's program. All user references, as signaled by bit 20, are modified by the hardware to refer to the correct current user location. The contents of the relocation register represent, therefore, the context that determines the user of the moment. Since bit 20 corresponds to a real address of 32,768, both the size of the JOSS system code and the size of individual users are limited to a maximum of 32,767 locations. In effect, the RAND relocation mode provides a second level of indexing.

Because the capacity of core memory is not sufficient to hold data for all possible users simultaneously, the magnetic drum is used to store data for some of the users during those times when interpretation of their data is not required. Drum transfers are controlled by the input/output processor through independent ports to the memories. Thus, transfers of user

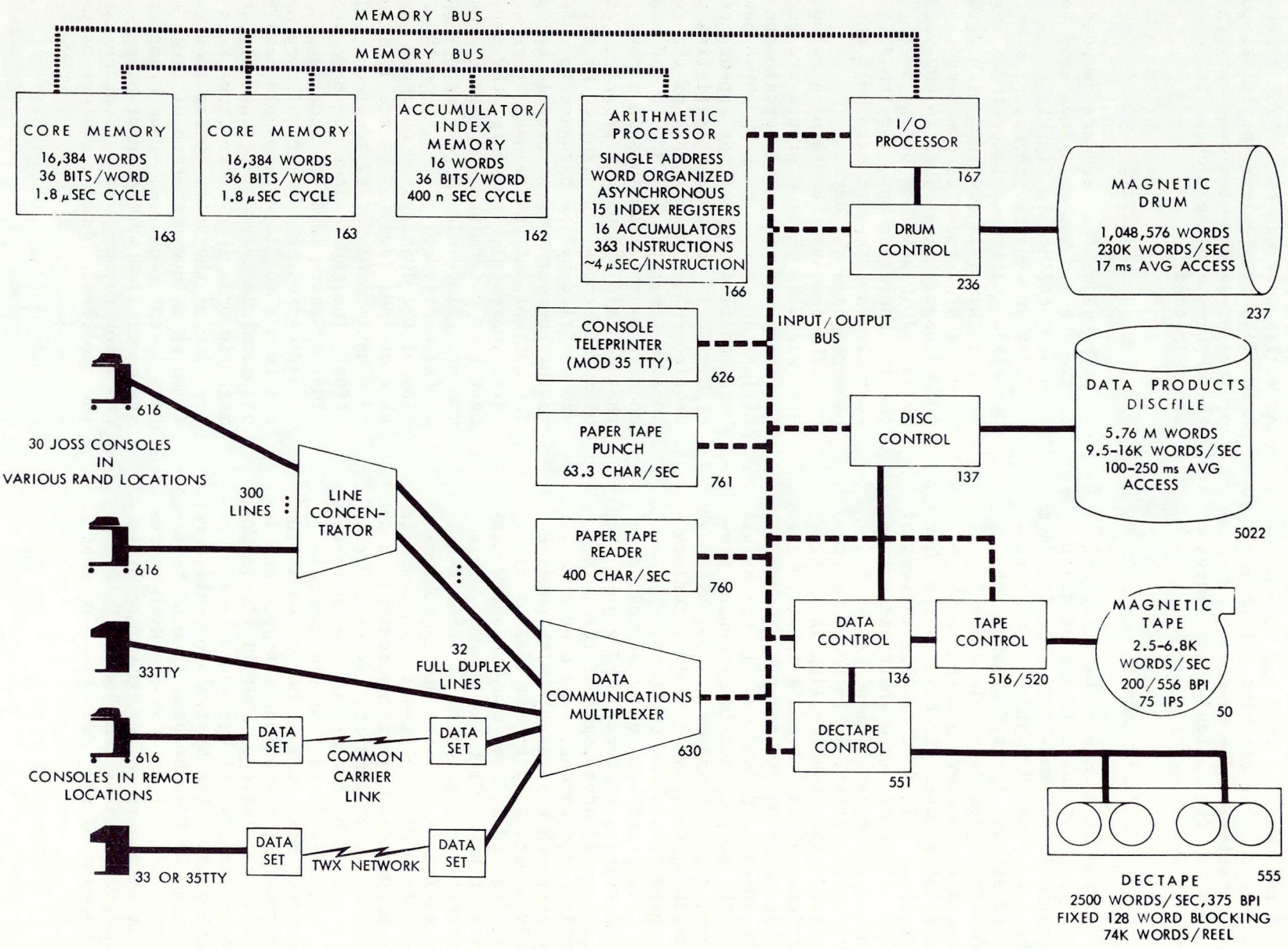


Fig. 1—JOSS PDP-6 system

data between the drum and core are accomplished independently of the arithmetic processor. The attention of the arithmetic processor is needed only to initialize the I/O processor for the transfer and to take action after its completion. Memory cycles are taken by the I/O processor as needed to service the drum. These cycles interleave with those taken by the arithmetic processor in the interpretation of JOSS users' programs.

Logging of information descriptive of the gross system operation is done on the console teleprinter. The information is printed each minute and includes the number of present users, the number of users computing, the amount of computing accomplished, the total lines transmitted to and received from users, and various errors.

The data communications multiplexer scans lines connected to the JOSS consoles and reports via a machine interrupt to the arithmetic processor when a character has been received or the transmission of an output character has been completed. All communication with the consoles passes through the multiplexer and down the I/O bus to the arithmetic processor.

In addition to the local JOSS console lines, the multiplexer has timing and other special gear necessary to interface with dataphones connected to remote JOSS terminals as well as with local TTY's and TTY's on the TWX network.

The data control handles the transfer of information to both the discfile and the magnetic tape. Because it cannot transfer data to more than one device at a time, usage must be shared.

The discfile provides long-term storage for users' programs and data. Its capacity of 200 million bits is sufficient for many thousands of user programs. Access time to individual records on the disc is generally about 200 ms, but long programs, queued use of the disc by several users, and computing commitments may extend an individual file or recall action to several minutes. Normally, however, a disc action takes about a second.

The IBM-compatible tape unit is used to collect accounting records and statistical information, which are processed in the general RAND accounting system on another computer. The discfile is periodially dumped onto tape for backup purposes, using an off-line program not contained in the regular JOSS software.

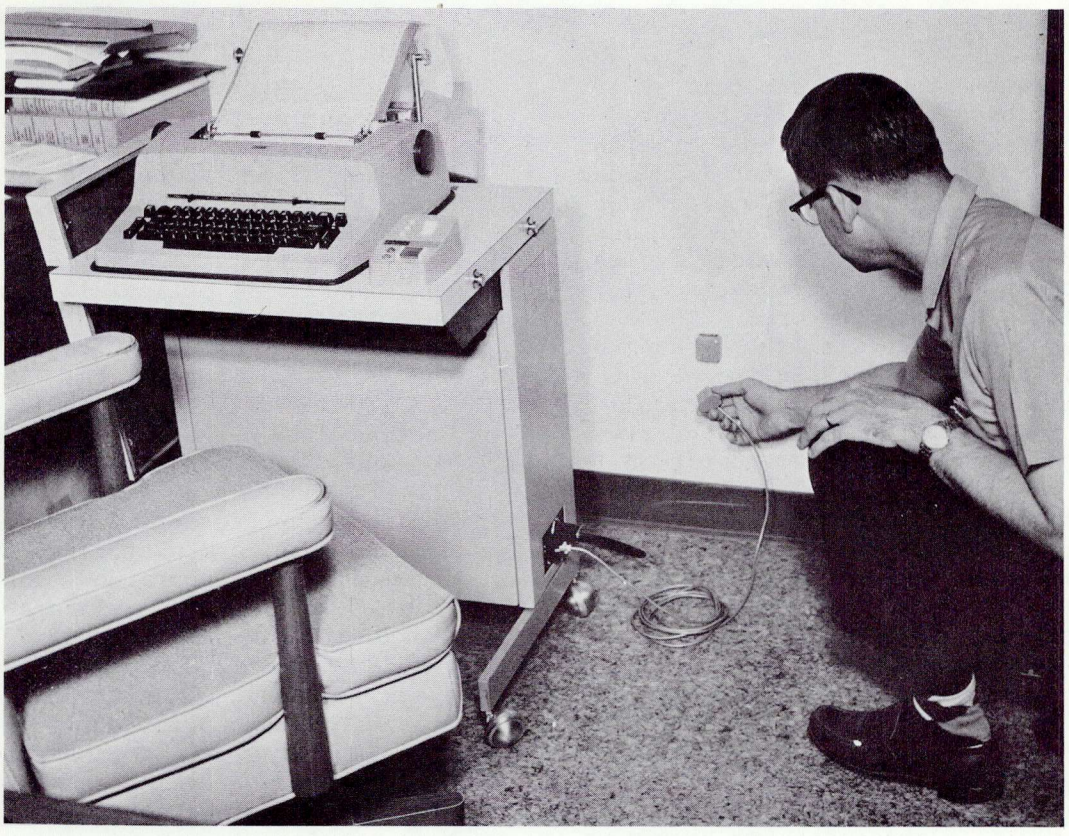
The JOSS system includes four Dectape drives that are used in system support. Operating binaries for the DEC-supplied time-sharing system and its subprocessors (assembler, editor, file manipulator) used in JOSS development are contained on Dectapes, as well as symbolic, relocatable, and absolute copies of the JOSS software. The IBM-compatible tape is used for assembly listings during DEC time-sharing system operation.

Paper tape reader and paper tape punch are used for loader and maintenance program input and for paper tape copying.

The line concentrator is built around Strowger line-finder stepping switches in much the same way as an ordinary PBX or CDO. It serves to concentrate 300 lines from user offices to 40 multiplexer inputs. The presence of a JOSS console at the plug in the user's office is detected by the concentrator and it establishes a connection to the computer. It is through this device that "plug in" computer power is provided in the individual user's office.

In addition to the JOSS console described below, up to eight input lines may be devoted to teletype operation. Currently, two model 33 TTY's are in use at RAND and two lines are available on the TWX's network.

The JOSS console was specially built to RAND specifications by the Digital Equipment Corporation. The IBM Selectric I/O writer includes a pin-feed platen, a two-color computer controllable ribbon shift, a special character set adapted to algebraic languages, and a specially built form-feed mechanism for moving the paper to page top.

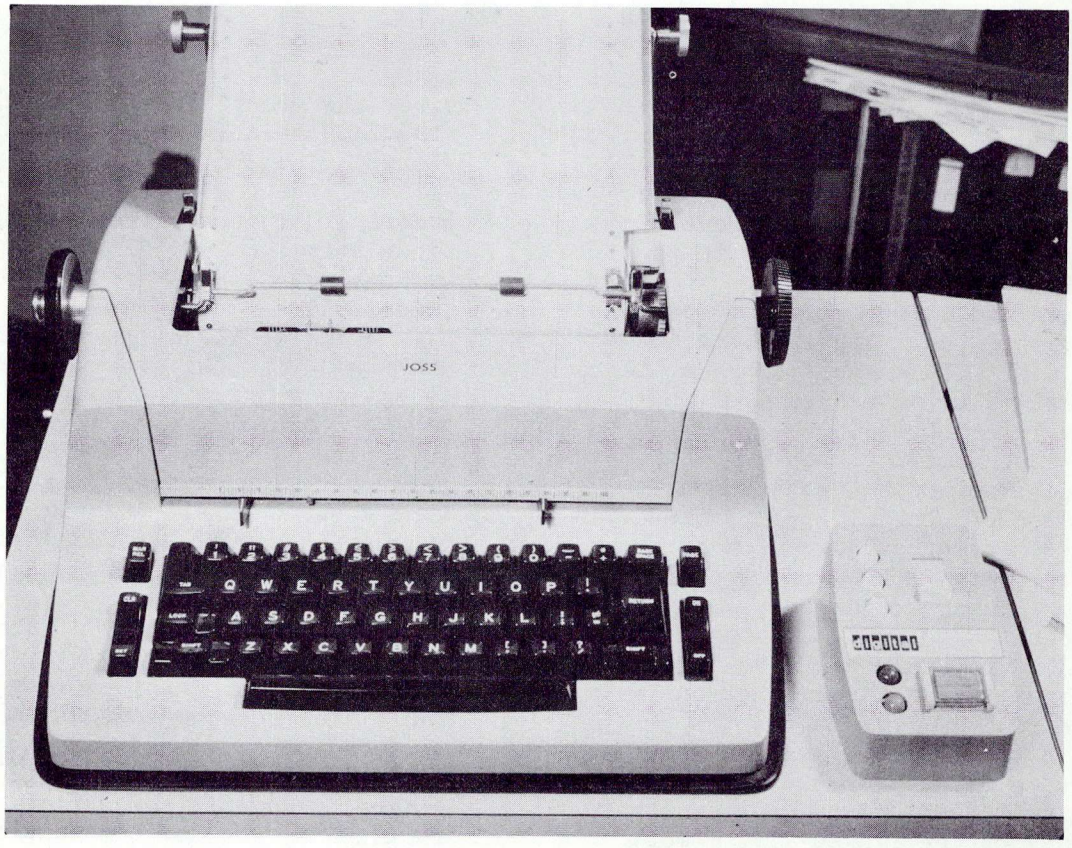


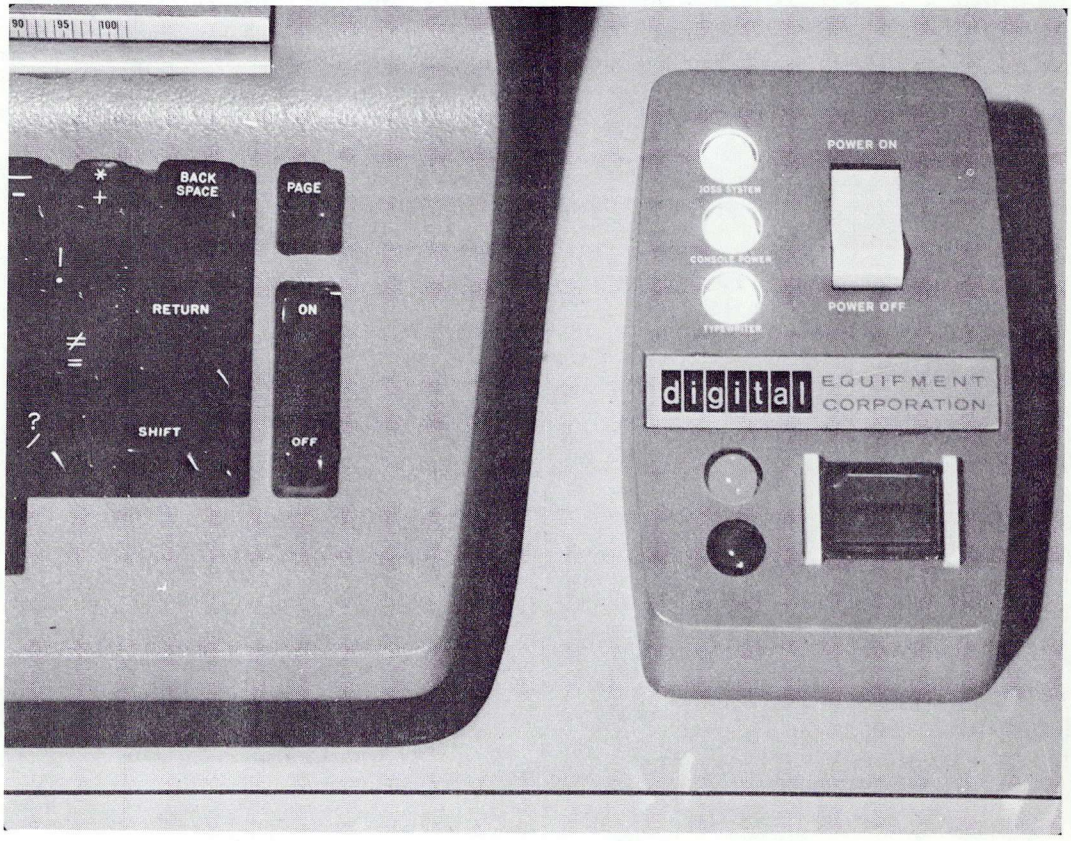
The cabinetry that holds the typewriter and, in its base, all the associated electronics is mounted on casters for mobility and includes a four-way detachable side table that provides a convenient work surface on the left or right side of the console at the user's option.

A console control box at the right side of the typewriter contains two control keys and three white status lights. The power ON/OFF key in addition to applying or removing power at the console sends a unique signal to the computer reporting logical ON and OFF for the console. An interrupt button provides a signal to request return of control of the console to the user during those times when the computer has control.

The status lights report that (1) the JOSS system is on, (2) the console has power on, (3) the typewriter is ready to print, (4) the computer controls the typewriter (red light), and (5) the user controls the typewriter (green light). During times when the computer is in control (red light on), the keyboard is locked and the ribbon color is shifted to black. When control of the typewriter is returned to the user, a beep tone sounds, the green light turns on, the keyboard unlocks, and the ribbon color shifts to green.

The console electronics contained in the base of the cabinetry control the console operation and send and receive signals to the computer over a full duplex line in an 8-bit, 11-unit start-stop code. The 67-ms signaling time per character is designed to keep the typewriter operating at its full capacity of 15 characters per second. Six bits are sent for each character (including up and down shift characters), one bit is used to indicate control information (request console status or console status report), and one bit is used for a parity check.


JOSS


BACK SPACE
PAGE
RETURN
ON
SHIFT
OFF
POWER ON
JOSS SYSTEM
CONSOLE POWER
TYPEWRITER
POWER OFF
digital EQUIPMENT CORPORATION

The JOSS Language

The language provided for JOSS users is simple and easy to learn with relatively few rules governing correct use. In many ways it is similar to other algebraic languages in wide use on every variety of computing machine. The language has been specially adapted to convenient, direct use by an active user at a typewriter console.

Most familiar statement types exist: Replacement (assignment), transfer of control, input, output, and formatting are executed by the verbs: Set, To, Demand, Type, and Form. The conditional if clause may be appended to any JOSS statement. The JOSS Do statement acts more like a subroutine call than the similarly named FORTRAN statement.

Significantly, some statement types do not appear. Declarations such as DIMENSION are unnecessary, because of the linked-list memory assignment in the user's block, and modes (e.g., REAL, INTEGER) are handled implicitly.

Whether a statement is to be interpreted immediately or stored for future execution is indicated implicitly by prefixing statements to be stored with a "step label," which gives the proper location of the new statement relative to others already stored. Thus, a step labeled 1.25 will be inserted after step 1.2 and before 1.28; it will replace any previous step labeled 1.25. Statements without step labels are interpreted immediately.

A collection of steps with step numbers having the same integer part is called a "part." Thus, all steps labeled one-point-something constitute part 1. A Do statement causes interpretation of a part as if it were a subroutine. Example 1 illustrates the use of six common JOSS verbs, the conditional if clause, direct and indirect program statements, and the ordering of statements by step number.

Certain of the JOSS language facilities deserve special mention because they are less frequently found in the common algebraic languages. The verb Let defines a formula or rule for computation. It may have up to ten parameters. The functions sum and prod allow direct expression of the mathematical operations for summation and product over a specified range of values. Use of these and similar functions (max and min) eliminates many program loops and aids in the compact expression of the desired computation.

Example 1--Sample JOSS Program

```
1.1 Demand x(i).
1.2 Set s=s+x(i).
1.3 Type i,x(i),s in form 1.
1.35 Set i=i+1.
1.4 To step 1.1 if i<4.

Form 1:
i = _  x(i) = ___.__  sum = ___.__

Set s=0.
Set i=1.
Do part 1.
        x(1) = 97.45
i = 1   x(i) =  97.45  sum =  97.45
        x(2) = -67.98
i = 2   x(i) = -67.98  sum =  29.47
        x(3) = 3.47
i = 3   x(i) =  3.47   sum =  32.94
```

Conditional expressions, which may be used wherever expressions are valid and which use colons and semicolons to denote the if...then... if...then...if...then...otherwise... notation, again contribute to compact notation of complex choices and discontinuous functions. For example, see P(x) in Example 3 below.

The first function allows the user to find the first value of an index that satisfied a given proposition.

The compact expression achieved by these features is shown in Examples 2-4, which give formulas for polynomial root finding, prime number determination, and two-point Gaussian integration.

As shown in Example 2, roots of the polynomial p(x) are found by Newton's method expressed in formula i(x), where q(x) is the approximate derivative of p(x). The formula r(x) recursively improves the root until it is sufficiently close to zero. The program at step 1 prints the three roots as found by setting approximate starting values through the for clause of the controlling Do statement.

The formula P(x) given in Example 3 has value true or false depending on whether x is prime or not. The first part, P(x), filters out certain special cases (e.g., negative numbers). The second formula tests for primeness by finding the first exact divisor (fractional part = 0) of x and reporting

Example 2--Root Finding

```
1 Type r(x),p(r(x)) in form 1.

Form 1:
x = __._____      p(x) =  _._____

        i(x):  x-p(x)/q(x)
        p(x):  x*3-10•x*2-6•x+10
        q(x):  [p(x+d)-p(x)]/d
        r(x):  [ |p(x)|<10*(-6): x; r(i(x))]

             d =          1•10*(-4)
Do step 1 for x=-5,1,10.
x = -1.24670      p(x) =    .00000
x =   .76528      p(x) =    .00000
x = 10.48142      p(x) =    .00000
```

Example 3--Prime Number Determination

```
1 Type x if P(x).


        P(x):  [x≤0:false; x≤3:true; p(x)]
        p(x):  first[ i=2,1(2)ip[sqrt(x)],x: fp(x/i)=0 and i≠1] = x
Do step 1 for x=0(1)12,100001(2)100101.
           x =         1
           x =         2
           x =         3
           x =         5
           x =         7
           x =        11
           x =    100003
           x =    100019
           x =    100043
           x =    100049
           x =    100057
           x =    100069
```

"true" if that divisor is x itself. Only divisors up to the integer part of the square root of x are tested. The typing program at step 1 and the controlling Do statement test the formulas and give some output.

The function I in Example 4 integrates the function f over the range a,b in n intervals.

Example 4--Gaussian Integration

```
      I(f):  h/2•sum(i=1(1)n: sum[j=1(1)m: f(x(i,j))])
        h:   (b-a)/n
    x(i,j):  a+h/2•[t(j)+2•i-1]

        m =          2
        n =         30
a=0
b=1
t(1)=-1/sqrt(3).
t(2)=-t(1)
Type I(exp),exp(1)-1.
     I(exp) =          1.71828184
  exp(1)-1 =           1.71828183
Let R(x)=(1+x*2)/(1+x*4).
Type I(R),sqrt(2)•arg(1,1).
      I(R) =          1.11072073
sqrt(2)•arg(1,1) =    1.11072073
```

JOSS arithmetic is carried out by an interpretive package of routines that operates on numbers carried in scientific notation--an integer magnitude and a decimal exponent. Primary advantages of this notation are exact I/O number conversion and the restriction of repeating fractions to those familiar in the decimal system.

Users may save programs, data, forms, and formulas on the discfile and retrieve them from the file using the verbs File, Discard, and Recall. Items stored on the disc are in symbolic form. The file operation behaves as if the user were typing on the disc and the recall operation acts as if the disc were typing on the user's program space. This means that the user's current core contents are only changed as implied by the contents of the disc. Statements replace current statements of the same number, and new values are assigned if variables defined on the disc were previously defined in core. The user may reference the files with his program to accomplish a limited form of chaining.

There are a number of features normally included in computing systems that JOSS does not have: (1) There is no way for a user to handle high-volume I/O, which precludes the use of large tape files. (2) The interpretive mode of JOSS operation (even down to the arithmetic) limits the speed of operation. Thus, very long, detailed calculations are impractical. (3) The maximum amount of core available to individual use is limited to 4K, making very large programs or programs with large data bases infeasible. Finally, (4) JOSS operates only on numbers, which rules out generalized symbol manipulation programs. All of these limitations were imposed because their inclusion was considered incompatible with a high-speed, highly interactive computing service for a large number of casual users.

JOSS Software

The JOSS operating software is divided into five principal parts: the interpreter, its arithmetic and

function subroutines, the monitor (which also contains the drum, tape, and TTY console I/O routines), the distributor (JOSS console and TTY I/O), and the disc routines. All of this code is permanently resident in lower memory. Its total size is about 16,000 memory locations, and includes all I/O buffers for the consoles and several thousand words devoted to the gathering of performance statistics.

The interpreter, along with the arithmetic subroutines, is the part of JOSS that examines users' commands and computes answers in response to them. Users' programs are analyzed character by character by the interpreter, as the name implies, to produce the indicated results. No compilation of user programs is done. User commands and data are carried in linked lists within variable-size user blocks. The entire user block must be in core during interpretation, and the RAND relocation hardware requires that this block must always occupy a contiguous area of memory. User blocks have a minimum size of 1000 words and increase in increments of 1000 to a system-imposed limit of 4000 words.

The distributor and disc routines are trap time I/O packages that control transmission of data to and from the user consoles and the discfile. Both communicate with the monitor about the I/O activity that they control through signals, which are software analogs of machine interrupts. These signals are referred to as logical traps or logical interrupts.

The monitor acts as a scheduling, resource-allocating, and synchronizing device, deciding when, and ensuring that, all data and hardware necessary for a particular action are simultaneously available. To carry out this process, the monitor maintains a series of queues of users in various activity states and of data for hardware devices. Signals from the other software components and a time-interrupt signal control both the changing of users from state to state and through these states the monitor's scheduling of the tasks of the system.

Also included in the monitor code are trap time routines to handle tape, drum, TTY console I/O, real-time clock interrupts; routines to gather and display performance statistics about overall JOSS operation; routines to provide accounting information to be input to the RAND computer time-charging mechanism; and routines that act as processors on the level of the interpreter to supply the log-on procedures of receiving initials, project number, department name, and the final log-off process.

Figure 2 shows the interrelations of the JOSS software, including the data buffers that interface certain of the components. Control and data paths are shown, although it is sometimes difficult to classify a particular path as control or data.

The disc routines and the distributor, as well as the portions of the monitor that deal with the drum, tape, and TTY console I/O, are trap time routines; that is, they operate only in response to hardware interrupts from the I/O devices that they control. Brief exceptions to this rule occur when the routines are entered to initialize an I/O action. These trap time routines generally signal completion of activity through logical traps that are interpreted by the monitor in its main processing loop.

Top-level control of the machine is shared between the monitor and the interpreter, with the bulk of time spent in the interpreter when users are requesting computation. The monitor regains control at least every 200 ms through the trap time setting of the signal COMEBACK, the logical interrupt signal for the interpreter. During its control periods, the monitor adjusts user states according to the current logical signals, initializes I/O actions as may be appropriate, and determines, from the user states, which user should receive the attention of the interpreter next.

The logical control signals that pass between the various software components of JOSS are summarized in

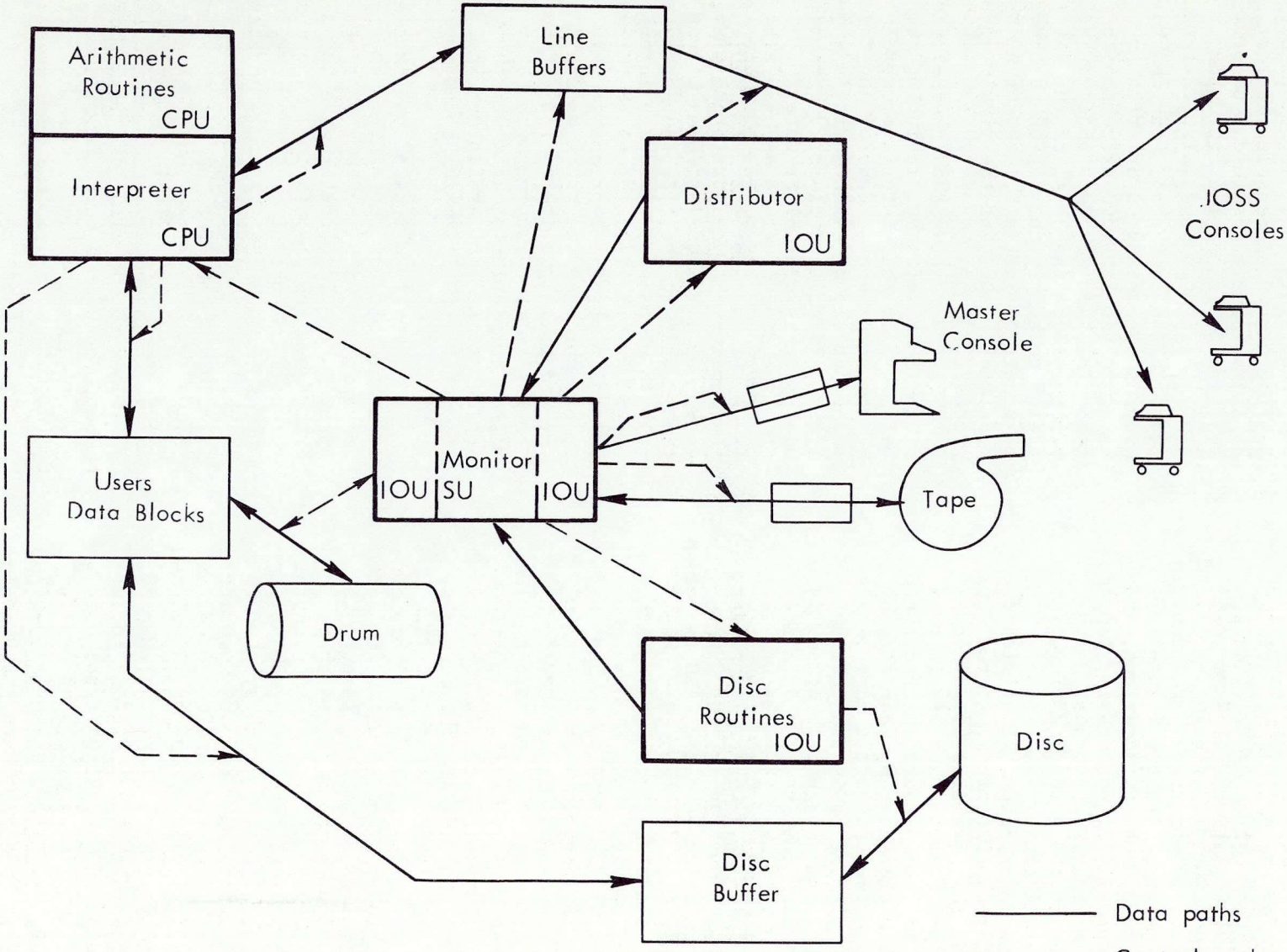


Fig. 2—Interrelations of JOSS software

Fig. 3. Most of the signals are indicated by the setting of certain communication cells. Some signals, however, particularly the ones that start I/O action, are routine calls.

User State Transitions

During JOSS operation, users are sequenced through the monitor states in response to signals from the consoles, signals from the interpreter (in response to the executing program), and a time signal generated by the hardware.

A diagram of the transitions between states is presented in Fig. 4. Although the figure gives an excellent impression of the overall operation of the monitor, it is by no means complete. Reference to the actual code must be made if the exact details of state transitions are required.

Several subcycles are identifiable in the figure. At the top left of the diagram are the states that relate to console turn on and turn off; at the top center, the states applying to program input and short computations; at the lower left, the typewriter output limited sequence; at the lower center, the compute limited sequence; at the lower right, the disc activity sequence; and at the upper right, the drum transfer sequence, used in certain cases when more core blocks are needed by a user.

The states are broadly divided into two priority groups: a high-priority group, shown with heavy shading, and a low-priority group. Users whose states are in the high group require interpreter service, while those in the low group do not. Signals from the consoles or from the user's program cause changes from one state to another.

The ON and OFF sequence provides mechanisms for calling the separate software processors that provide the user's initial core block and that handle the initial salutation, including receipt and checking of the user's initials, project number, and department, and the writing of the accounting record at turn-off time. At turn on, the user is placed in QM state if the system limit on the number of users has been reached. Under most operating conditions, this limit is set higher than the number of consoles so that the limit is never reached. Before the installation of the drum, the limit was lowered to a value that would keep all active users in high-speed core. When the user turns his console off, his state is changed to TOF, except when a disc action is in progress. In the latter case, the OFF signal is flagged in the user's status word, and his state remains unchanged through completion of the disc action. At that time, the change to TOF state, production of final accounting records, and the reenabling of the console occurs.

During periods when a user is typing program steps and doing short computations via direct commands, he cycles through the states in the Input and Short Computation Sequence. During typing the console is green and the user's state is GR. As soon as the carrier return is pressed, his state changes to RC and subsequently to CU for interpretation of the line just typed. If the line is accepted without comment, the state returns directly to GR while error messages or response lines return to GR via DSU during the time that the typewriter is printing the output line. Attachment of a buffer to each green user is required for receiving his input. In the event that none are free, the user's state is changed to ABG until a buffer becomes available.

The Output Limited Sequence controls those users whose programs produce output faster than can be printed at the console. One of the system parameters is called the choke number. When the number of lines of output ready for printing at the console equals this number, the user's state is changed to CK and further computation ceases. When the number of output lines is reduced to the value of the unchoke number, the user's state is changed to UC and computation is resumed. If, during this sequence, the user presses the interrupt button, his state is changed to RIB and computation is resumed for response to the interrupt request.

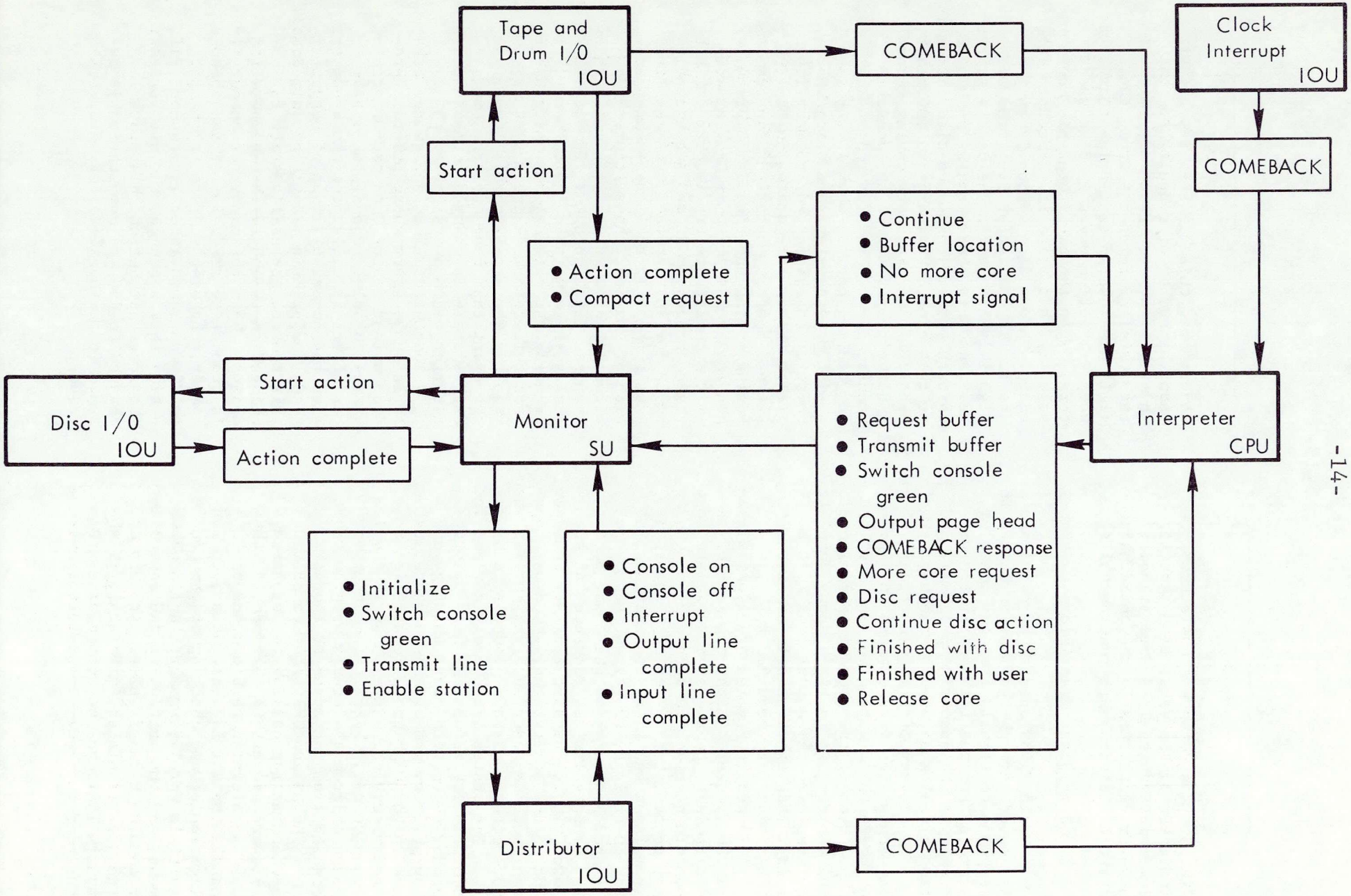


Fig. 3—Summary of logical control signals

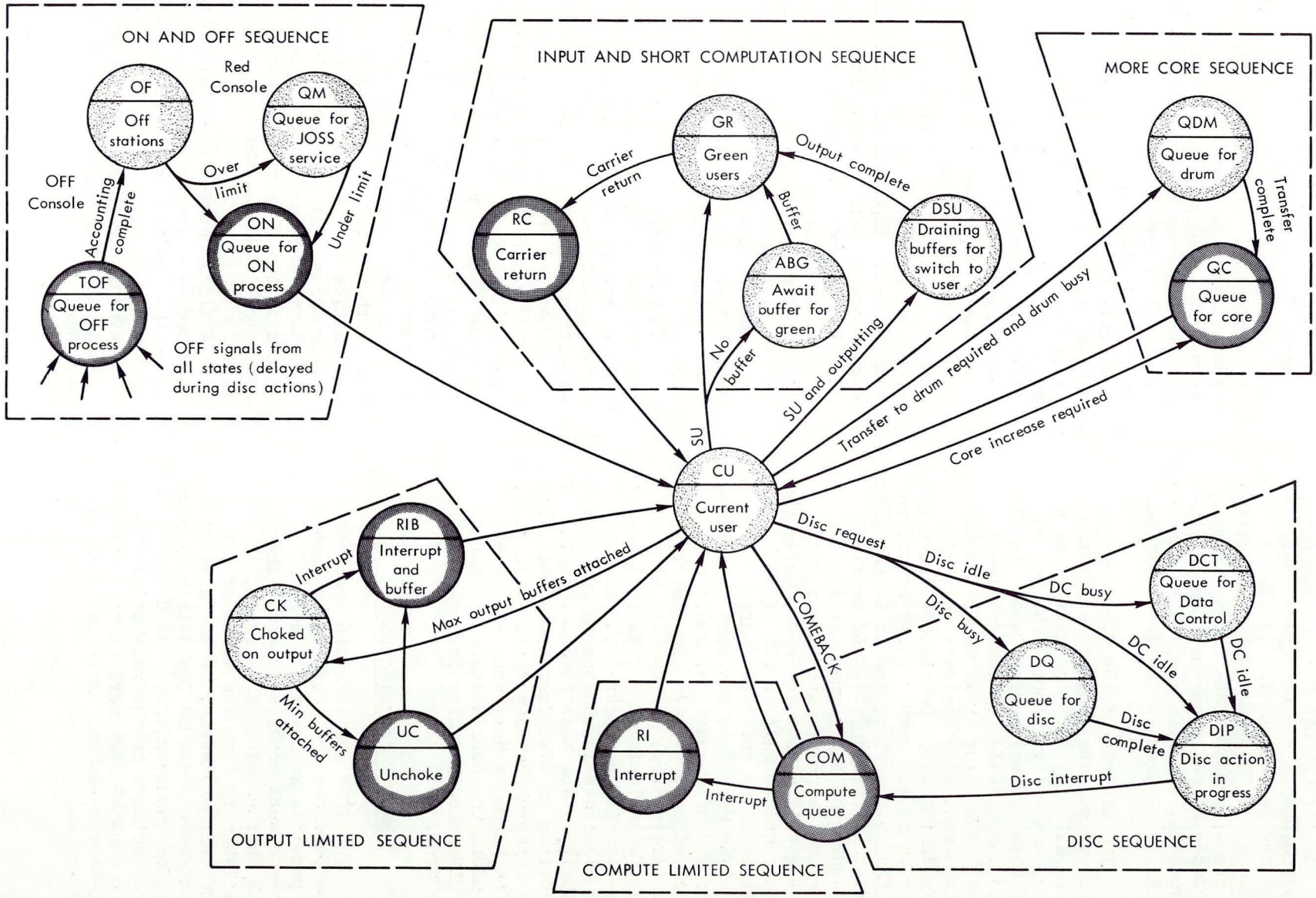


Fig.4—JOSS user state transitions

In the Compute Limited Sequence, those users in the COM queue share the computer in 200-ms time slices or "quanta," as controlled by the setting of the signal COMEBACK through the hardware clock interrupt. During these sequences, the user's state is changed to RI if he presses the interrupt button.

The purpose of the Disc Sequence is to queue users for access to the disc, allowing only one user on at a time, and to synchronize the use of the data control device, which is also used for transfer of information to tape. The major cycle is CU-DIP-COM with actual disc activity occurring only during DIP. During COM the disking user waits his turn for the attention of the interpreter, and in CU the disc buffer is either filled or emptied depending on the command in process. Other users requesting disc service during this time wait in DQ, while the disc routines wait in DCT if the tape unit is using the data control.

The More Core Sequence is employed when a user requests an additional block of core storage (implicitly through the interpreter) and no idle blocks are available in core. In this case, the user is transferred to the drum and his state is changed to QC, a high-priority state that forces the swap algorithm to make core space available and restore the user to the main core. If the drum is busy when the request is made, the user's state is changed to QDM where he waits for transfer to the drum.

Main Processing Loop

The JOSS monitor main processing loop provides for action on a number of possible asynchronous events. These events or signals are either such that no machine interrupt is available to flag the event or are such that processing at the time of occurrence would be impossible, improper, or inconvenient. The deferral of these actions to the main processing loop ensures that certain routines need not be trap-protected or coded as pure procedures.

Figure 5 outlines the flow of control for the major functions acted on in the main processing loop. There are two main paths, JOSS active (processing a user's request) and JOSS idle. The idle path has a minor branch not shown on the figure that distinguishes idle passes when the drum is busy. This indicates that a user currently on the drum has requested service and no other activity could be performed in overlap with the drum transfer. Usually this happens when there are many users but a very light computing load.

The interpreter releases control to the monitor whenever any of the monitor-provided facilities are desired and also when the signal COMEBACK is set. COMEBACK is set by the occurrence of monitor-distributor signals and every 200 ms by a real-time clock interrupt. Thus, when the interpreter is processing long computations, the monitor regains control in order to service signals from other consoles and to time-share the use of the machine among compute-bound programs.

Console signals are produced for the monitor by the distributor. These signals are translated into state changes, which will later control the selection of system tasks.

The disc interrupt signal is set by the on-line disc routines when a disc buffer is either filled during an input action or emptied during the filing of user information on the disc. The state of the user with disc action in progress is changed to compute (COM) so that the interpreter can proceed with the filling or draining of the next buffer load of information. This ensures that the user's block will be in-core during the necessary points in the disc transfer without requiring it during the entire transfer.

A final mode of processing is provided by a monitor flag that indicates the monthly production of disc accounting records. This disc-to-tape operation is synchronized through logical traps from the disc and tape routines.

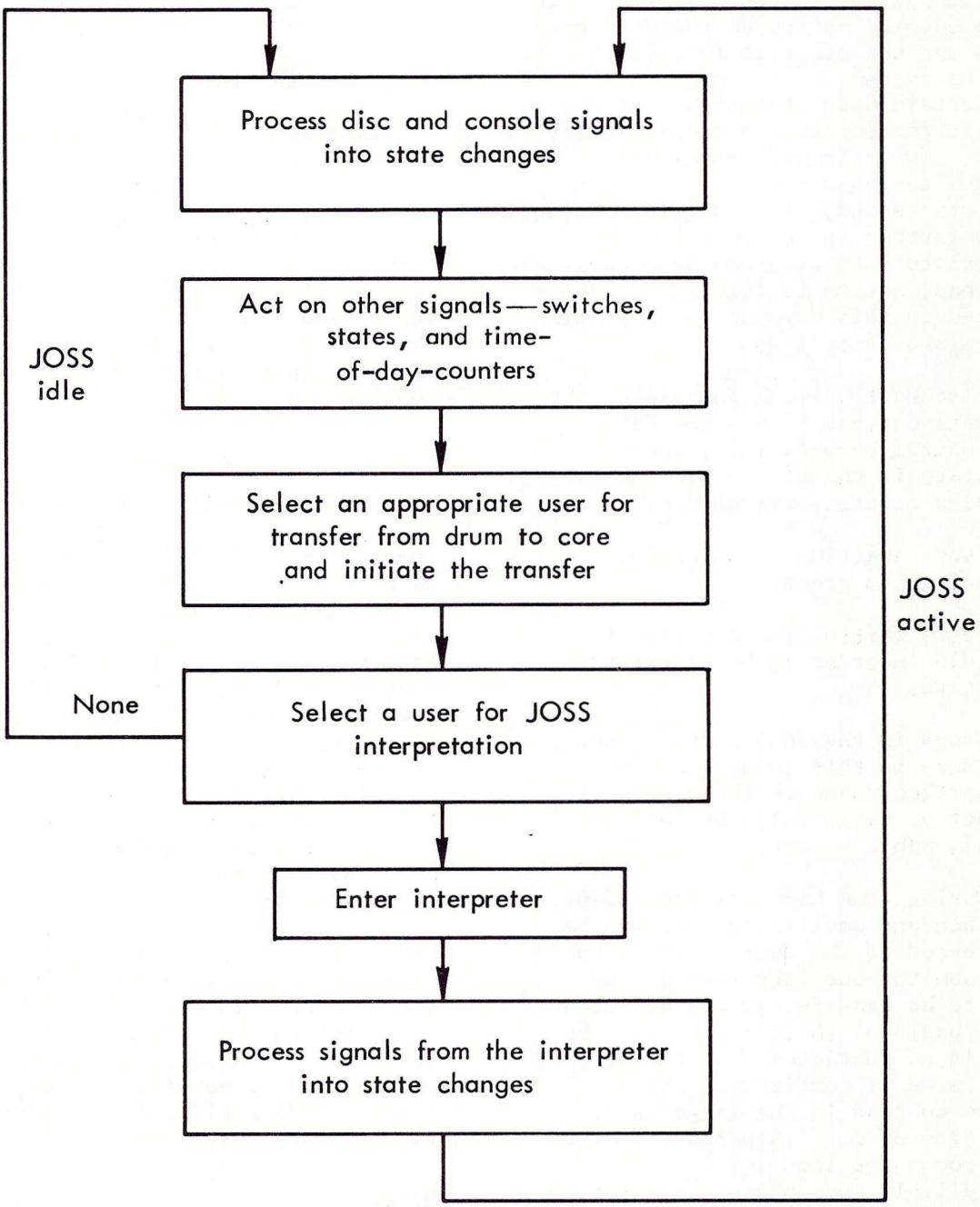


Fig.5—JOSS monitor main processing loop

The miscellaneous functions performed in the main processing loop can be divided into four broad categories: switch examination, dormant queue service, core compaction, and periodic functions.

Two special switches are monitored: one to enable the system shutdown procedure and the other to forcefully shut down the system.

Certain user states are entered to await the occurrence of particular events. The main processing loop examines the queues corresponding to these states and, if a user is waiting, checks further to determine if the appropriate condition has been obtained. If it has, action is taken. Queues serviced in this way and the resulting actions are as follows:

DCT Disc user waiting for use of the data control. When the data control becomes idle, the user state is changed to DIP and the disc routines are entered.

ABG Users awaiting a buffer for switch to green.

QDM Users waiting for the drum to idle in order to be transferred there.

QM Users in the JOSS service queue. Users in this queue are given service whenever the number of active users falls below the allowable number.

During some kinds of drum swaps, more than one small-size user may be transferred to the drum in order to make room for one large user. The users to be transferred are selected on the basis of their priority. Thus, there is no guarantee that the core space freed is contiguous, which it must be to read in the large user. Compaction of core is performed at the main processing loop level to bring all available core blocks adjacent to one another at a time when they are not in use by the interpreter. The waiting transfer from the drum to core is then initiated.

Major time interval incrementing--the counting of minutes, hours, days, months, and years--is accomplished from the main processing loop. Functions performed in this part of the MPL include initiating log reports each minute, accumulating statistics on the minute and hour, and initiating accounting for the disc each month.

Next in line in the MPL is the selection of an appropriate user to be transferred from the drum to core. This action is only performed if a transfer is not already in progress. Of course, only the high-priority users are candidates for transfer into core.

Finally, an in-core user is selected and the interpreter entered. The main processing loop is completed when the interpreter returns to the monitor.

JOSS Usage

The statistics given below are intended to describe briefly the extent to which JOSS is typically used, as of the fall of 1966. It should be noted that usage patterns are still changing fairly rapidly. Improvements that have had a substantial impact on usage include that from JOHNNIAC to the PDP-6, which increased computational speed by a factor of thirty, and the later additions of the drum and discfile, which gave users five times more core space and the ability to file programs and data. Intermediate system changes of smaller magnitude, as well as the natural growth of user interest, will, no doubt, keep the usage patterns changing on the positive side, as is usual throughout the computing industry.

With the exception of six hours per week of scheduled maintenance and occasional (or possibly frequent) unscheduled down time, JOSS operates 24 hours per day, 7 days per week.

Usage

- About 500 users, some casual, some addicted.

- About 300 different users each month.
- Computation is divided into 150 sessions each day or about 3000 each month.
- The users are served by 34 consoles, of which 2 are on the TWX's network and 5 are in remote locations (3 on the East Coast). The remainder are located in RAND.
- Usage is highest during mid-morning and mid-afternoon, normally peaking at 25 concurrent users. The average number of simultaneous users for a 24-hour period is 6; average in prime shift is about 13.

Computing

- Actual computing--as contrasted to idling when waiting for work--is 130 hours per month, which is 18 percent of total time or 74 percent of prime shift.
- Approximately 3,000,000 JOSS statements are processed every 24 hours in an average time of 5.5 ms per statement.

Sessions

- A typical console session lasts 45 minutes; 10 percent are less than 2 minutes; and 10 percent greater than 2 hours.
- Computing time during a session averages 2 minutes or about 5 percent of session time, but 60 percent use less than 10 seconds and 1 percent compute more than 1 hour.
- On the average during each session two items are retrieved from the files, one is discarded, and one placed in the file.
- Users input 2 lines per minute on the average and output 6, which totals 90 input lines and 270 output lines during the typical session.
- Mean interaction (that is, time for an individual user between carrier returns) is 30 seconds and is distributed approximately exponentially with 40 percent less than 6 seconds and 1 percent greater than 5 minutes.
- Average character rates per user are 1 per second input and 3 per second output.
- User programs average 320 JOSS cells or 1000 machine words. Twenty percent use less than 10 JOSS cells and 7 percent more than 1000.
- The session time distribution of size is as follows:

<1K words	58%
1K-2K	30%
2K-3K	5%
3K-4K	7%

The early usage of JOSS as implemented on the PDP-6 confirmed it to be an effective computational tool. The average computation per session of 20,000 statements indicates that complex problems are being solved, but the hardware and implementation are such that considerable computational capacity remains. This additional capacity appears sufficient to support between 100 and 200 consoles at current usage rates.