MEMORANDUM
RM-5257-PR
FEBRUARY 1967

JOSS: DISC FILE SYSTEM

I. D. Greenwald

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5257-PR
FEBRUARY 1967

JOSS: DISC FILE SYSTEM

I. D. Greenwald

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.

DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

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

Published by The RAND Corporation

PREFACE

JOSS* is a multi-user, single-server computing system that provides for the solution of numerical problems. The system consists of a central computer containing the JOSS program and a number of typewriter consoles connected to the computer via telephone lines. The JOSS filing system provides the individual user with a service that will save him the need for retyping frequently used programs and/or data.

This Memorandum describes the JOSS filing system, and is intended primarily as program documentation for maintenance personnel, particularly those who may need to correct or modify the file-handling routines. The report may have interest for others, however, since it includes the rationale for more critical decisions and techniques.

This Memorandum is a part of The RAND Corporation's continuing program of research in computer sciences under U.S. Air Force Project RAND.

The JOSS system was originally implemented on the JOHNNIAC computer in 1963 by J. C. Shaw; the present expanded version is implemented on the Digital Equipment Corporation PDP-6 computer.

---

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

SUMMARY

The JOSS filing system provides the individual user with a service that will save him the need for retyping frequently used programs and/or data. A secondary purpose is to provide the capability for limited program chaining. This Memorandum presents program documentation for JOSS maintenance personnel, particularly those who may need to correct or modify the file-handling routines.

The opening section describes the filing system's rationale, which implies low user demand for file action in terms of total session time; therefore, considerable concern is given the amount of primary storage occupied by the file-service routines and associated core buffers. The following section outlines the salient hardware characteristics; and Sec. III briefly discusses the logical organization of information on the file. Section IV contains a description of the interactions of the Monitor, Interpreter, and Disc Processor as viewed by the latter. An interesting coding technique utilized by the Disc Processor is analyzed in Sec. V. With certain provisions, this specialized technique is valuable for "straight line" coding of discontinuous processes.

The Memorandum concludes with detailed program flows for both the disc service routines (Sec. VI) and the Disc Processor (Sec. VII). An Appendix presents user-directed instructions for using the JOSS filing capabilities.

CONTENTS

I. PHILOSOPHY OF THE JOSS FILING SYSTEM

The rationale behind the JOSS* filing system is to provide the individual user with a service that will save him the need for retyping frequently used programs and/or data. (A by-product of this service is his ability to use the files as back-up against system failure.) A secondary purpose is to provide the capability for limited program chaining. Since this rationale implies low user demand for file action in terms of total session time, there is little concern with the time to service such demands. On the other hand, there is considerable concern with the amount of primary storage that the file-service routines and associated core buffers occupy.

The core buffers are desirable for three reasons:

1) No requirement to freeze a user in core during file action; i.e., freedom to swap him to drum as total system activity requires.
2) The flexibility of being able to file in other than "core dump" format, e.g., typewriter output format. Intermediate buffers are necessary, because of insufficient time between words from the disc to process such formats.
3) The manner in which hardware relocation is used precludes direct file input to a user core block without impairing system efficiency.

Providing complete service for one user's file request before proceeding to another makes possible confinement to one core buffer (128 words), i.e., if the disc actions are not time-shared with other disc actions. This is consistent with the goal of minimizing space at the expense of time.

---

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

II. HARDWARE

The Data Products Corporation Model 5022 DISCfILE has 45,056 records of 128 words each, organized as follows:

128 words per record;
44 records per position;
64 positions per disc;
16 discs per file.

We shall refer to the 44 records within one position as a track.

That portion of the disc which has been allocated* is organized into linked available record space,† except for the first track. The first track contains a Master Control Record (Fig. 1) and 43 Directory Records (Fig. 2).

---

*An off-line maintenance program is used to allocate tracks on the disc to JOSS use. By keeping allocated tracks at a level consistent with expected use, one can considerably reduce the time for dumping to backup magnetic tape.

†Normally linked records contain the record number of the next in the right-hand half of the first word. The last record on a chain contains a zero pointer.

WORD		
0	# of available records	pointer to first
1	# of active directory entries	# of allocated tracks
2	pointer to first available directory entry	# of allocated records
3	ID of tape from last reload	
4	date from tape	date of reload
5	Available	
127		

Fig. 1--Master Control Record

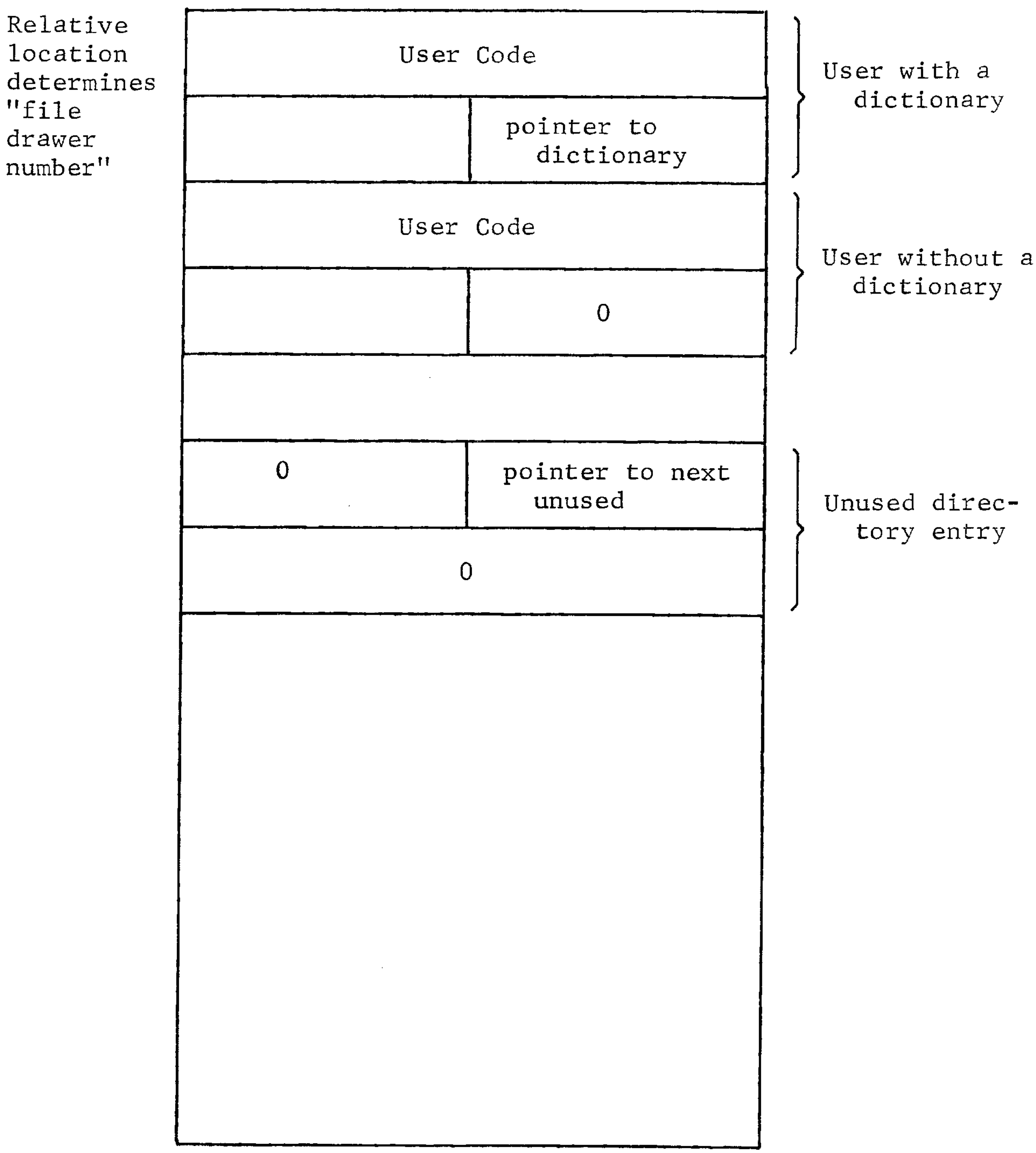

Fig. 2--A Directory Record

III. FILE ORGANIZATION

From the standpoint of the JOSS user, permanent storage* is organized into file drawers called files. The user requests a file by filling out a file request form. The file code which he supplies is entered (off-line) into the disc directory and the relevant file number is returned to him. (The Appendix contains excerpted instructions for the use of files.)

The first time the user requests that something be filed, a user dictionary (Fig. 3) is created, taking the first record on the available space list. This dictionary can be purged only by the off-line maintenance program.

The actual filing is done one record at a time, using available space as required. When the last record has been written, the user dictionary and the master control record are updated and rewritten.

---

*Many JOSS users are unaware of the existence of a disc file.

WORD			
0	User File #		Dictionary Header
1	# of items	# of records	
2	User Code		
3+5·(item #-1)	# of records	pointer to first	Item Entry
+1	date written	date last used	
+2	SPARE	item type	
+3	Item Code		
+4	Project Number		
Up to 25 numbered items			

Fig. 3--A Dictionary Record

IV. PROGRAM INTERFACES

The filing program is organized into two major components:

1) The Disc Input/Output Routines;
2) The JOSS Disc Processor.

The Input/Output Routines accept requests for reading, writing, or checking (after write) one record. After initiating a disc seek, the routines exit to the caller. From then on, the I/O routines are interrupt driven: completion of the seek, read, write, or check causes an interrupt at which time error detection and correction is done. Finally, the I/O routines set a "logical interrupt" signal that tells the monitor to re-enter the Disc Processor. A flag is set to indicate whether or not an unrecoverable disc error has been discovered.

The Processor interfaces with both the Monitor and the Interpreter. For the most part, the Processor sets a "logical interrupt" flag for the Monitor whenever information is to be conveyed to the Interpreter. This information consists of a set of communication cells (Table 1) and an I/O buffer.

Since the Processor is coded as a one-entry point routine, the communication cell ACTION also contains a bit to indicate whether or not this is the first call for the given action or is a resumption of the action subsequent to a logical interrupt. The Interpreter sets this bit when an action is initiated; the Processor resets it upon entry.

Table 1

COMMUNICATION CELLS

1	FILE:	User file number.
2	KEY:	User file code.
3	PROG:	User item number.
4	NAME:	User item code.
5	RPN:	RAND Project number (for accounting).
6	ACTION:	1 = RECALL 2 = FILE 3 = DISCARD 4 = TYPE ITEM-LIST 5 = USE (Bit 17 is set to 1 on initial ACTION request)
7	TYPE:	Interpreter code to allow for several file constructs. Supplied by the Interpreter when filing; supplied by the Processor when recalling.
8	FLAG:	Supplied by Interpreter when filing: zero if this is not last buffer load, non-zero if this is last buffer load. Supplied by Processor when recalling: number of records.

When a user requires disc service, the Interpreter requests the disc from the Monitor. When this user reaches the top of the disc queue, the Monitor attaches the disc to the user and re-enters the Interpreter. Upon completion of the required service, the Interpreter informs the Monitor that the disc may be released.

Upon attachment of the disc to a user, and through completion of the service, the Disc Processor interfaces with the Monitor and Interpreter via the 128-word buffer (BFR) and a set of communication cells.

The main interface with the Monitor is through the cell DISC.S., which is used to signal "logical interrupts" from the Processor. There are three such signals:

1) The attention of the Interpreter is required.
2) A discard has been completed. The attention of the Interpreter is required. In addition, the Monitor does accounting for the number of record/days of disc usage. This information is supplied by the Processor via a set of cells labeled DISC.D.
3) Disc I/O is complete. Re-enter the Disc Processor.

In addition, DISC.S=1 may also signal the Monitor that a "skulk" record is in the buffer. At midnight on the last day of each month, the Monitor requests the Processor to read all active user dictionaries in order to record them on tape for off-line accounting. Since the Monitor knows that this activity is going on, no ambiguity results from the dual use of the same signal.

One further interface with the Monitor is via the cell D.TIME, which is used to time out the disc in the event of failure. The disc service modules set this cell for a one-second time-out. The Monitor modifies and checks

D.TIME and re-enters the Processor for an error signal whenever time runs out (D.TIME=0).

When the Monitor signals the Interpreter that the disc has been attached to the user, the Interpreter supplies the items indicated in Table 1 to the Disc Processor and requests the Monitor to initiate Processor action. The Monitor clears the logical interrupt cell DISC.S and calls the Processor. The latter inspects Bit 17 of the ACTION cell to determine whether this is an initial request or a continuing action. If the former, Bit 17 is reset for the next entry.

The Processor reports back to the Interpreter by filling in the communication cell RESULT (see Table 2), and then setting a logical interrupt in DISC.S. The RESULT codes are fairly self-explanatory: 1 and 2 deal with filing; 3 and 4 with recalls; 6 with typing item-lists; 7, 9, 13, and 14 treat identification checks; 8 is applicable to all ACTIONS except USE or FILE; 10 and 11 relate to disc space (user files are currently limited to 100 records each); 12 indicates hardware failure (in the event of an unrecoverable disc error, the Processor will inhibit further disc access); 15 indicates a required DISCARD before a FILE for an existing item; 100 and 200 are codes supplied to the Monitor during the monthly "skulk."

The Disc Processor is transparent to the information being transmitted in the buffer. That is, it executes the tasks of reading, writing, and deleting user records on the disc; it never inspects the information content of these records. Hence, the real ACTIONS are read, write, delete; the words RECALL, FILE, and DISCARD were used in

Table 2

DISC SUBPROCESSOR SIGNALS TO INTERPRETER

RESULT	SIGNALS
1	Please fill the buffer for filing.
2	Filing has been completed.
3	The buffer contains a recall record.
4	The buffer contains the last recall record.
5	Discard has been completed.
6	User dictionary is in input buffer.
7	The user has supplied an illegal file number.
8	The user has no items.
9	The user file KEY doesn't check.
10	The user has used too much disc.
11	The disc is full.
12	Unrecoverable disc error.
13	The user item NAME doesn't check.
14	USE is OK.
15	User is trying to FILE over an in-use item.
100	User dictionary in buffer.
200	No more user dictionaries.

the foregoing discussion for clarity of exposition. As a result, the TYPE code (see Table 1) does indeed supply "open-ended" capability for formatting user information.

V. CODING TECHNIQUE

The heart of the coding of the Disc Processor and the Disc I/O routines is based on a technique described to the author in 1964 by Bob McClure of Texas Instruments, Inc. McClure was concerned with generalized techniques of program structuring, especially as they related to binding re-entrant subroutines with multiple entry points. The specialized use in the Processor is concerned with multiple entry-point subroutines in which the subroutine itself determines its next entry point. This specialized technique is valuable for dealing with discontinuous processes by "straight line" coding, provided that re-entrant capability is not required.

The coding is simple. It requires the existence of a "Jump and Set Return" instruction and indirect addressing capability or the ability to simulate these (e.g., by using index registers). The JSR stores the contents of the program counter at the specified address and replaces the contents of the program counter with the specified address + 1. (Those familiar with the IBM 7040/44 will recognize this as a "TSL.") Thus:

$$\alpha \ : \ \text{JSR } \beta$$

results in cell β containing $\alpha + 1$ and control transferring to $\beta + 1$.

If we now define a single exit point from the subroutine

EXIT : 0

Exit to main program ,

and a single entry point

```
ENTRY : JUMP to BEGIN if initial request
        Jump indirect EXIT        ,
```

we can "straight line" code the retrieval of a program stored on disc:

```
BEGIN : Reset initial request bit
        Set address of appropriate directory record
        Initiate read
        JSR EXIT
        Process record
        Set address of user dictionary
        Initiate read
        JSR EXIT
        Set address of first information record
LOOP  : Initiate read
        JSR EXIT
        Process record
        Jump to END if last record
        Set RESULT=3
        Set DISC.S=1
        JSR EXIT
        Set address of next information record
        Jump to LOOP
END   : Set RESULT=4
        Set DISC.S=1
        JSR EXIT
        HALT (should not be re-entered)
```

The multiple entries to the routine are accomplished by having the I/O trap routines set a signal for the Monitor to jump to ENTRY.

Further aesthetic appeal and space-saving may be achieved by combining the functions "Initiate read" and "JSR EXIT" into a subroutine, SR, which itself is called by a JSR:

```
SR : 0
     Initiate read
     JSR EXIT
     Jump indirect SR
```

We may then define a macro, READ, to be the instruction "JSR SR" and a second macro, SIGNAL N, to be the sequence:

```
Set RESULT=N
Set DISC.S=1
JSR EXIT
```

to obtain the appearance of completely sequential code.

```
BEGIN :  Set directory address
         READ
         Process
         Set dictionary address
         READ
         Set info record address
LOOP :   READ
         Process
         Jump to END on last record
         SIGNAL 3
         Set info record address
         Jump to LOOP
END :    SIGNAL 4
         HALT
```

In coding the Disc Processor, it was decided to make its entry a JSR as well:

```
ENTRY :  0
         Jump to BEGIN if initial request
         Jump indirect EXIT
```

and, hence:

```
EXIT :   0
         Jump indirect ENTRY
```

VI. DISC SERVICE ROUTINES

DATA CELLS

A.SV: \
B.SV: } Space for saving high-speed registers A and B.

D1: Disc address; filled by external world before calling the service routines.

D2: Place to which BLKI/Ø points; set from D3 by subroutine E2 (Seek)

D3: XWD - ↑D128,BFR-1

D5: Number of tries; set to 3 by subroutine E1 (housekeep), modified and tested by subroutine E11 (test results)

D6: Error type : Preset to 2 (no error) by E2 modified by E10 (file interrupt) in the event of error. Tested by E11.

D7: Status of disc

0 = idle

1 = busy (set by E2 and E5; tested and reset by E10)

2 = unrecoverable disc error (set by E4, tested by external world)

-1 = successful completion (set by RD, WR, or RC; tested by external world)

D10: Copy of D1; set by E1, used by E2.

EXIT POINTS

LEAVE: This exit is used when subsequent re-entry is expected. Transfer to this exit is via a JSR.

LV1: This exit is used when re-entry is not expected. Transfer to this exit is via a JRST.

Exit is accomplished by JEN @E10.

ENTRY POINTS

D.RD: Read a disc record into BFR.

D.WR: Write a disc record from BFR.

D.RC: Read compare a disc record with BFR.

(NOTE: In all cases the disc address is assumed to be in cell D1. Transfer to all entry points is via JSR.)

SUBROUTINES (all entered via JSR)

E1: Housekeeping:

Force an interrupt on channel 5

Set proper return in E10

Set D5 for 3 tries (in the event of errors)

E2: Seek:

Set timer (D.TIME) for 1 second time-out

Set channel 5 interrupt for E10

Set "disc busy" in D7

Preset success in D6

Initialize seek

Exit via LEAVE

.

.

On re-entry:

Call E11 to check results

Unrecoverable error -- Call E4

Retry -- Loop back

Success -- Exit

E4: Unrecoverable error:

Clear file

Set unrecoverable error in D7

E4.E: Set logical interrupt (3) in DISC.S

Exit via LV1

E5: Disc I/O Setup:

Set timer (D.TIME) for 1 second

Set "disc busy" in D7

Store Data channel CWD in cell 42

Set interrupts

Enable interrupts

E6: Data Control Interrupt

Shut off data control

End disc file if reading

E7: Read

Call E5 with Read CWD

Select interrupt conditions (error, done)

Start I/O

E8: Write

Call E5 with Write CWD

Select interrupt conditions (error, done)

Start I/O

E10: File Interrupt

Call E4 if D7 $\neq$ busy

Reset D7

Mark error type in D6 if error

Clear errors, set D7, and exit if command incomplete

Clear file, reset timer, and re-enter via LEAVE if command complete

E11: Test Results

Exit to caller +3 if D6 = success

Decrement D5 and exit to caller +2 if > 0

Exit to caller +1

FLOW (Let α stand for RD or WR or RC)

D.α: Call E1 (housekeep)

D.α1: Call E2 (seek)

Call E7, E8, E9 (read, write, read compare)

Exit via LEAVE

(Re-enter)

Call E11 (check results)

D.α2: Call E4 on unrecoverable error

To D.α1 on retry

D.α3: Set D7 = -1 on successful end

Re-enter Disc processor via E4.E

VII. DISC PROCESSOR

ENTRY, EXIT, AND ERRORS

Entry is always via JSR to DISC.C

Exit is via JSR to EXIT

Errors:

UI: Unrecoverable error (disables further use of files by setting bit 21 of cell SWITCH) then goes to ERROR.

ERROR: Resets timer and sets RESULT = 12 (disc error), then exits via F1.4 (F1.4 stores away RESULT, sets logical interrupt and calls EXIT)

SUBROUTINES (all called via JSR)

P16 Convert record number to Disc address (record # in register A)

If record number is less than zero or greater than N.SIZE go to ERROR

Convert to disc address noting that there are 44 records per position and 64 positions per disc.

Leave result in register A and cell D1

P20 Read a record

Call D.RD

Call EXIT

.

.

On re-entry,

Check D7 for success (-1):

Exit if yes

Go to UI if no

P21 Write a record

Call D.WR

Call EXIT

.

.

On re-entry,

Go to UI if error (D7 $\neq$ -1)

Call D.RC

Call EXIT

.

.

On re-entry,

Go to UI if error (D7 $\neq$ -1)

Otherwise exit

In what follows, the words "Read" and "Write" should be interpreted as calls on P20 and P21, respectively.

T2: Set up a user dictionary in BFR

BFR	File #
+1	0
+2	File key
+3	0
	0
+127	0

T3: Place list on available space; Input:

Cell N = record # of first record of list

J = available space count

K = pointer to first available

Note that the master control record is not updated by T3.

The list will be placed on the top of available space.

It is assumed the list is terminated by a zero pointer.

T4: Update the Master Control Record; Input

Cell J = # of available records

K = pointer to first available

Read the MCR, update, Write MCR

PROCESSOR FLOW

DISC.C: To ERROR if timer has run out

To DISC.I if initial call

To ERROR if disc is disabled or KEY has changed

Re-enter Processor via EXIT

DISC.I: To ERROR if disc is disabled

Compute DATE

To F1

F1: Save KEY for later checks

Read Master Control Record

Number of allocated records → N.SIZE

To ACCTNG if ACTION = 100

Set J = # available records

K = pointer to 1st

L = # of directory entries

To F2 if user FILE # is legal

F1.3: Set "illegal file number"

F1.4: Store in RESULT

Set D7=0 (disc idle)

Set logical interrupt in DISC.S

Leave via EXIT

To ERROR if re-entered

F2: Read directory record for this FILE

To F2A if KEY doesn't check

To F2.1A if ACTION $\neq$ open

Set "open is OK"

To F1.4

F2A: Set "user KEY no good"

To F1.4

F2.1A: To F2.3 if user has a dictionary

To F2.1 if ACTION = write

Set "no programs"

To F1.4

F2.1: To F2.2 if no space on disc (decrement J)

Set K in pointer to user dictionary and in Q

Write dictionary record

Convert Q to disc address via P16

Read the record

Place pointer to next in K

Set up dictionary via T2

Write dictionary

To F2.5

F2.2: Set "insufficient space"

To F1.4

F2.3: Save pointer to dictionary in Q

Convert to disc address via P16 and read

F2.4: To F2.5 if ACTION ≠ print dictionary

Set "dictionary in core"

To F1.4

F2.5: Place dictionary item entry in P(1)P+4

HALT on illegal ACTION

To F2.6 on READ

To F2.7 on WRITE

To F2.6A if empty item

Call F10 (delete)

Set DISC.S for deletion signal

Set date in P+1 for accounting

Set "Deletion complete"

To F1.4

F2.6: To F2.6A if empty item

Call F11 (read)

Set "last record in core"

To F1.4

F2.6A: Set "program name no good"

To F1.4

F2.7: Call F12 (write)

Set "writing complete"

To F1.4

F10: (Delete)

To F2.6A if item code $\neq$ NAME

Clear first word of item entry

Decrement number of programs and number of records

Convert Q to disc address via P16

Write dictionary

Place records on available space via T3

Update master control via T4

Exit

F11: (Read)

To F2.6A if item code $\neq$ NAME

Set DATE into date last used in item entry

Set communication cells from item entry:

FLAG = number of records

TYPE = item type

Convert Q to disc address via P16

Write dictionary

F11.1: Get pointer to next information
record from P
Convert to disc address via P16
Read the record into BFR
To F11.2 if not last record (pointer $\neq$ 0)
Exit

F11.2: Place pointer in P
Set "next record in core" in RESULT
Set disc idle (D7 = 0)
Set logical interrupt (DISC.S = 1)
Leave via EXIT
To F11.1 on re-entry

F12: (Write)
To F12.5 if item is not empty
To F12.2 if user has not used up SIZE

F12.A: Set "too much disc"
To F1.4

F12.1A: Set "insufficient space"
To F1.4

F12.2: Set pseudo item entry in P:
\# records = 0
pointer = K
TYPE, DATE, NAME, RPN

F12.3: Decrement available records (J);
  to F12.1A if zero
 Increment # records in P
 Increment total user space; to
  F12.A if > SIZE
 Read record K into BFR
 Place pointer in K
 Set "please fill buffer" in RESULT
 Set disc idle and logical interrupt
 Leave via EXIT

 On re-entry:
 If last record (FLAG = 0), set pointer
  in BFR = 0
 Write the record
 To F12.3 if FLAG ≠ 0

F12.4: Read user dictionary (record # in Q)
 Increment # of programs and # of records
 Place P in item entry
 Write the dictionary
 Update master control record via T4
 Exit

F12.5: Set "delete before writing"
 To F1.4

```
ACCTNG:     Set J = 1    (Index to directory record)
    A0:     Set K = 0    (Index to entry in directory)
    A2:     Read record J
    A1:     To A4 if entry K has a dictionary
            Increment K
            To A1 if K ≤ 63
    A3:     Increment J
            To A0 if J ≤ 43
            Set "no more records"
            To F1.4
    A4:     Read record pointed to by entry K (user dictionary)
            Set "User dictionary in BFR" in RESULT
            Set disc idle and logical interrupt
            Leave via EXIT

            On re-entry:
            Increment K
            To A2 if K ≤ 63
            To A3
```

Appendix

INSTRUCTIONS FOR USE OF JOSS FILES

JOSS filing space is organized into numbered _files_ each of which is compartmentalized in turn into twenty-five numbered _items_.

To use this feature of JOSS, each user must first request assignment to a personal file. To help protect against inadvertent misuse, each file will be given a five-character code (supplied by the user) when assigned. Thereafter, each reference to this file must include both the file number and the assigned code, in the proper format; e.g., "file 98765 (G9999)."

As each of the twenty-five items in a file is used, it may be given a user-supplied code of from one to five characters; the format of an item reference is thus the same as the format of the file reference. This device gives additional protection and provides bookkeeping information.

The JOSS sentences for utilizing file space are of four types:

1) Use file _n_ (_code_).

The number _n_ is the file number assigned the user and _code_ is the designated code name. (Note the mandatory space after _n_.) After checking that the _code_ correctly corresponds to the file number of an assigned file, JOSS's response indicates that all item references will be to items in this file. This command may be given at any time and remains in force until another "Use" is given or the console is turned off. An error message will result if the user references an unassigned file.

Example: Use file 9999 (Z9990).

2) File list as item n (code).

The number n refers to the item number (1,2,...25), and code is the code name associated with the item. (The use of a code here is optional.) The information to be filed, list, may be almost anything appearing in a "Type" statement: "all," "all parts," "all forms," "all values," "all formulas," "part k," "step j," "form k," "x," etc. If item n is already in use, JOSS will respond with an appropriate message.

Examples: File all as item 17 (prog).
File a,b,c,d as item 11 (data).
File step 1.1 as item 23.

Once an item has been filed, it will remain so until discarded.

3) Discard item n (code).

After checking that the code correctly corresponds to the item, item n is discarded and the freed space made available.

4) Recall item n (code).

After checking that the code correctly corresponds to the item, item n is recalled from the file exactly as if it were being typed at the user's console. Parts, steps, forms, formulas, and values will be added to the current user program, replacing already existing program steps, forms, values, etc., where required.

5) Type item-list.

Types out the item number, item code, date of filing, RPN, and space occupied for each used item in the file.

In all cases (Use, File, Discard, Recall, Type item-list), JOSS will respond with a message when the requested action has been completed if the action was requested in direct mode. The length of time for this response will vary from two seconds to eight minutes, depending on the number of users waiting for file action and the sizes of the items being handled. A user should anticipate long delays when JOSS is turned on, at noon, and just prior to recess.

JOSS BIBLIOGRAPHY

PUBLICATIONS OF CURRENT INTEREST

Baker, C. L., JOSS: Introduction to a Helpful Assistant, The RAND Corporation, RM-5058-PR, July 1966 (AD 636993).

-----, JOSS: Console Design, The RAND Corporation, RM-5218-PR, February 1967.

Bryan, G. E., JOSS: Introduction to the System Implementation, The RAND Corporation, P-3486, December 1966; also published by The Digital Equipment Computer Users Society, DECUS Proceedings, Fall 1966.

-----, JOSS: User Scheduling and Resource Allocation, The RAND Corporation, RM-5216-PR, January 1967.

Greenwald, I. D., JOSS: Arithmetic and Function Evaluation Routines, The RAND Corporation, RM-5028-PR, September 1966.

-----, JOSS: Console Service Routines (The Distributor), The RAND Corporation, RM-5044-PR, September 1966.

PUBLICATIONS OF HISTORICAL INTEREST

Baker, C. L., JOSS: Scenario of a Filmed Report, The RAND Corporation, RM-4162-PR, June 1964 (AD 602074).

"The JOSS System: Time-Sharing at RAND," Datamation, Vol. 10, No. 11, November 1964, pp. 32-36. (This article is based on RM-4162-PR above.)

Shaw, J. C., JOSS: A Designer's View of an Experimental On-Line Computing System, The RAND Corporation, P-2922, August 1964 (AD 603972); also published in AFIPS Conference Proceedings (1964 FJCC), Vol. 26, Spartan Books, Baltimore, Maryland, 1964, pp. 455-464.

-----, JOSS: Conversations with the Johnniac Open-Shop System, The RAND Corporation, P-3146, May 1965 (AD 615604).

-----, JOSS: Examples of the Use of an Experimental On-Line Computing Service, The RAND Corporation, P-3131, April 1965 (AD 614992).

-----, JOSS: Experience with an Experimental Computing Service for Users at Remote Typewriter Consoles, The RAND Corporation, P-3149, May 1965 (AD 615943).