INTERNAL MAINTENANCE SPECIFICATIONS

FORTRAN COMMON I/O LIBRARY

VERSION 2.0/4.0

M674

INTRODUCTION

The IMS assumes that the reader has a working knowledge of FORTRAN and knows the general form of the statements involved. For instance, in considering a statement such as PRINT n, L, it is assumed that the reader is familiar with the meanings normally attached to n and L.

Upon encountering an input or output statement, the compiler generates a calling sequence for use by the execution time routines. Format syntax only is checked at compiled time. format diagnostics are produced during execution. Each particular set of I/O statements, i.e., READ, WRITE, ENCODE, BUFFER IN, etc., use an individual execution time routine. These routines do their own processing within themselves and depend only on the record manager for the I/O. Information is passed to the record manager through macro calls STORE'ing directly into the FIT.

All information necessary for the completion of the task is generated by the compiler in the form of a parameter list (APLIST) and passed to the execution time routine. The APLIST is terminated by a zero word (APEND). This is a negative zero if control will be returned to the execution time routine for the completion of this statement. An example of this is the following statement: READ(1) I,A(I). The address of A(I) can be calculated only after I is read. INPB= is entered initially to read I, then returns to the calling routine which passes the address of A(I) in the APLIST for the restart call. If APEND is plus zero then this is a final call.

A table of source statements and associated object library routines used is provided later in this section. The form of the parameter list (APLIST) for each object routine is described in the section for that routine. The routines are described in alphabetical order.

SOURCE STATEMENTS AND ASSOCIATED PRIMARY OBJECT ROUTINES

The following table shows the entry points of primary object library routines used as a result of FORTRAN source statements.

Source Statement	Entry Points Used	Routine
READ (u,f) k READ f, k	INPCI.,INPCR.	INPC=
WRITE (u,f) k WRITE f, k PRINT f, k PUNCH f, k	OUTCI.,OUTCR.	OUTC=
READ (u) k READ (u)	INPBI., INPBR.	INPB=
WRITE (u) k WRITE (u)	OUTBI., OUTBR.	OUTB=
REWIND u	REWIND.	REWIND=
BACKSPACE u	BACKSP.	BACKSP=
ENDFILE u	ENDFIL.	ENDFIL=
CALL OPENMS (u,ix,l,p)	OPENMS.	OPENMS
CALL READMS (u,fwa,n,i)	READMS	READMS
CALL WRITMS (u,fwa,n,i)	WRITMS	WRITMS
CALL STINDX (u,ix,l)	STINDX	STINDX
A = UNIT(u)	UNIT	UNIT
B = EOF(i)	EOF	EOF
J = IOCHEC(u)	IOCHEC	IOCHEC
L = LENGTH(u)	LENGTH	LENGTH
BUFFER IN (u,p) (A,B)	BUFIN.	BUFIN=
BUFFER OUT (u,p) (A,B)	BUFOUT.	BUFOUT=

ENCODE (c,m,v) k	ENCODI., ENCODR.	ENCODE=
DECODE (c,m,v) k	DECODI., DECODR.	DECODE=
PROGRAM name(...)	Q8NTRY.	FORSYS=
PAUSE	PAUSE.	FORSYS=
CALL EXIT	EXIT	FORSYS=
STOP o	STOP.	FORSYS=
END	END.	FORSYS=

ABBREVIATIONS

APEND = terminator word of a parameter list

APLIST = parameter list

BOI = beginning of information

EOF = end-of-file

EOI = end-of-information

EOR = end-of-record

FIT = File Information Table

FWA = first word address

I/O = input/output

LCM = large core memory (CYBER 76)

LWA = last word address

SCM = small core memory

1.1 Common Parameter List Flag Bit Definitions

Three flag bits have been defined for modifying the context of 18- or 24-bit address fields passed through APLISTs to the common object time I/O routines. The flag bits are located in the parameter word that they modify, in bit positions 57 to 59. The address field subject to modification is normally right-justified in bits 0-17 or 0-23. Flag bit meaning is defined as follows:

Flag Status	Context
All flags = 0	The address field contains the direct address of the parameter being passed; the parameter resides in SCM.
LCM = 1	The address field contains the direct 24-bit parameter address; the parameter resides in LCM.
VAR = 1	The address field points to a variable.
FP = 1	The address field contains an ordinal and offset to a formal parameter. The form is 18/offset,6/ordinal.

1.2 APLIST Parameter Element Expansions

1. FIT pointer VFD 1/VAR, 1/FP, 40/0, 18/FITADR

   where FITADR = FWA of the FIT

2. LIST item VFD 1/LCM, 1/FP, 1/IND, 3/0, 6/TYPE, 18/NBREL, 6/0, 24/ITEM

   where ITEM = address of the I/O list item

   where TYPE = type of this list item. Possible values are:

   0 - reserved
   1 - logical
   2 - integer
   3 - real
   4 - double
   5 - complex

If IND = 1, NBREL contains the SCM address of the list element count.

If IND = 0, NBREL and the 6 bits after it = number of contiguous elements in the list item.

3. APEND VFD 60/END

where END = +0 to denote the end of an I/O list;

= -0 to denote the intermediate interruption in an I/O list. Subsequent list elements will be requested on the next call(s). (This can occur, for example, when a list element has function call as a subscript).

4. RECORD length VFD 42/0, 18/WDCNT

where WDCNT = number of words in a record.

5. FORMAT pointer VFD 1/LCM, 1/FP, 1/VAR, 33/0, 24/FMT

where FMT = vector to a FORMAT specification.

6. MODE pointer VFD 42/0,18/SCM address of word that contains the file mode.

7. BUFFWA VFD 1/LCM, 1/FP, 1/VAR, 33/0, 24/FWA

where FWA = vector to FWA of an I/O buffer area.

8. BUFLWA VFD 1/LCM, 1/FP, 1/VAR, 33/0, 24/LWA

where LWA = vector to LWA of an I/O buffer area.

9. STRING pointer VFD 1/LCM, 1/FP, 1/VAR, 33/0, 24/DATSTR

where DATSTR = vector to a data string for DECODE input or ENCODE output.

10. COUNT pointer VFD 1/LCM, 1/FP, 1/VAR, 33/0, 24/CNT

where CNT = vector to a word containing a character count.

11. TRACE VFD 12/LINECNT, 18/IDADR

where LINECNT = line number of source program statement that caused this call to be compiled, in binary. If LINECNT = 0, no line number will be printed in error traceback listings (to accommodate RUN compiler output).

IDADR = SCM address of a word containing the program unit identification for traceback. The ID word will contain the program unit name in bits 59-18, in left-justified display code with blank (55B) fill, and the address of the program unit entry point in bits 17-0. If the program unit has more than one entry point, the address will be that of the main entry point.

1.1 BACKSP=

1.2 PURPOSE

BACKSP= will backspace the logical unit specified one logical record.

1.3 USAGE

This routine is called in response to the FORTRAN statement BACKSPACE u, where u is the unit designator.

Entry Point: BACKSP.

Entry Conditions: X1 contains the FIT pointer.

Exit Conditions: Upon exit BACKSP= will have backspaced the file one logical record.

1.4 RESTRICTIONS and ERROR MESSAGES

none

1.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FWA of the FIT, GETFIT. is called and returns the address in X1.

b. If the file is positioned at BOI then control is returned to the calling routine.

c. The FORTRAN endfile bit in the FIT is cleared.

d. If the last operation on the file was a write, an ENDFILE is written and the file is backspaced over the ENDFILE.

e. The file is then backspaced over the user-s logical record using the record manager macro SKIPBL.

1.6 EXTERNAL CALLS

GETFIT. - locates the FWA of the FIT

2.1 BUFIN=

2.2 PURPOSE

BUFIN= initiates a read operation that reads one logical record from a specified file directly into the user-s buffer.

2.3 USAGE

This routine is called in response to the FORTRAN statement BUFFER IN (u,p) (A,B) where

u = Logical unit number
p = Recording mode
A = FWA of the block of data to be transmitted
B = LWA of the block of data to be transmitted

Entry Point: BUFIN.

Entry Conditions:

A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:

FIT pointer
Address of word containing the mode indicator
FWA of the block of data to be transmitted
LWA of the block of data to be transmitted

Exit Conditions: Upon exit a read request will have been made to the record manager.

2.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: Exit from BUFIN= does not guarantee that the read operation has begun. A call to UNIT must be made to check for the completion of the transfer of data.

Error messages:

	Error number
END OF FILE ENCOUNTERED, FILENAME - XXXXXX	55
WRITE FOLLOWED BY READ, FILENAME - XXXXXX	56

2.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FWA of the FIT, GETFIT. is called and returns the address in X1.

b. BUFIO is called to obtain and check the parameters from the APLIST.

c. If the FORTRAN endfile bit is set in the FIT, then issue error message number 55.

d. If the current file position is EOI, set the FORTRAN endfile bit in the FIT and return to calling routine.

e. If the last operation on the file was a write, then issue fatal error message number 56.

f. A request is made to the record manager to initiate the read operation and control is returned to the calling routine.

2.6 EXTERNAL CALLS

GETFIT. - Locates the FWA of the FIT.

BUFIO. - Obtains the parameters from the APLIST.

SYSERR. - Handles error processing

3.1 BUFIO=

3.2 PURPOSE

BUFIO= is a common routine for BUFIN= and BUFOUT=. It obtains the parameters from the APLIST and stores the necessary fields in the FIT.

3.3 USAGE

BUFIO= is called each time BUFIN= or BUFOUT= is called to obtain parameters from the APLIST.

Entry Point: BUFIO.

Entry Conditions:

X1 = FWA of the FIT
X2 = Open code
X3 = Record manager error exit
X4 = Record manager data exit
B7 = FWA of the APLIST

Exit Conditions: Upon exit all necessary fields will have been set in the FIT to complete the read or write.

3.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: none

Error Messages:

	Error number
AREA SPECIFICATION BAD - FWA .GT. LWA	58
AREA SPECIFICATION SPANS SCM/LCM	57

3.5 GENERAL DESCRIPTION of the ROUTINE

a. Store record manager error and data exit.

b. Store conversion mode in the FIT.

c. Open the file if it is not currently opened.

d. If the FWA and LWA are not in the same type of memory, then issue error message number 57.

e. If FWA is greater than LWA, then issue error message number 58.

f. Compute record length and store in the FIT.

g. Store FWA of the record area in the FIT.

3.6 EXTERNAL CALLS

SYSERR. - Handles error processing

4.1 BUFOUT=

4.2 PURPOSE

BUFOUT= initiates a write operation that writes one logical record to a specified file directly from the user-s buffer.

4.3 USAGE

This routine is called in response to the FORTRAN statement BUFFER OUT (u,p) (A,B) where

u = unit designator
p = recording mode
A = FWA of the block of data to be transmitted
B = LWA of the block of data to be transmitted

Entry Point: BUFOUT.

Entry Conditions:
A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:
FIT pointer
address of word containing the mode indicator
FWA of the block of data to be transmitted
LWA of the block of data to be transmitted

Exit Conditions: Upon exit a write request will have been made to the record manager.

4.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: Exit from BUFOUT= does not guarantee that the write operation has finished. A call to UNIT must be made to check for the completion of the transfer of data.

Error Messages: none

4.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FWA of the FIT, GETFIT. is called and returns the address in X1.

b. BUFIO. is called to obtain and check the parameters from the APLIST.

c. A request is made to the record manager to initiate the write operation and control is returned to the

calling routine.

4.6 EXTERNAL CALLS

GETFIT. - locates the FWA of the FIT.

BUFIO. - obtains the parameters from the APLIST.

SYSERR. - Handles error processing

5.1 CLOSMS

5.2 PURPOSE

CLOSMS links a FORTRAN caller to the common FORTRAN/COBOL random file utility to close a random file.

This routine is called in response to theFORTRAN statement

5.3 USAGE CALL CLOSMS (u) where u is the unit designator.

Entry Point: CLOSMS

Entry Conditions:

X1 = FIT pointer

Exit Conditions: The file will have been closed: the master index will not have been rewritten if the file was read only.

5.4 RESTRICTIONS and ERROR MESSAGES

none

5.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called to obtain the FWA of the FIT.

b. CLOS.RI is called to actually close the file.

c. If an error number is returned from CLOS.RI, then print the error message and terminate, else return to the calling routine.

5.6 EXTERNAL CALLS

GETFIT. - locates the FWA of the FIT.

CLOS.RI - does the actual closing of the file.

SYSERR. - Handles error processing

8.1 DECODE=

8.2 PURPOSE

DECODE= transfers, under format control, packed data from one core location into data items and arrays located elsewhere in core.

8.3 USAGE

This routine is called in response to the FORTRAN statement DECODE (c,m,v) L where

- c = Unsigned integer constant or a simple integer variable specifying the number of characters in the record.
- m = Statement number or variable identifier representing the format statement.
- v = Address of a variable identifier or an array identifier which is the starting location of the formatted record.
- L = List of variable locations.

Initial Entry Point: DECODI.

Entry Conditions:

- A1 = FWA of the APLIST
- X1 = Address of word containing the character count

APLIST format:

- Address of word containing the character count
- Address of word containing the format statement
- Address of word containing address of string
- List item
  - .
  - .
  - .
- APEND (zero word)

Restart Entry Point: DECODR.

Entry Conditions:

- A1 = FWA of the APLIST
- X1 = List item

APLIST format:

```
list item
   .
   .
   .
APEND (zero word)
```

Exit Conditions: If APEND is minus zero, data from the string will have been unpacked and moved to the DAT. buffer. If APEND is plus zero, the data will have been moved to the list items.

8.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: Character count must be less than 151 characters and greater than zero.

Error messages

	Error number
DECODE CHAR COUNT .GT. 150	66
DECODE CHAR COUNT .LE. ZERO	66

8.5 GENERAL DESCRIPTION of the ROUTINE

a. The character count is checked and if in error the appropriate error message is issued.

b. The FWA of the format and string are obtained from the APLIST and control is passed to KRAKER. for initialization.

c. The next list item is obtained from the APLIST and control is passed back to KRAKER. where the data item is formatted and moved to the DAT. buffer.

d. If the APLIST element is plus zero, a terminal call is made to KRAKER. If the APLIST element is minus zero, control is returned to the calling routine to obtain the next list item.

e. DECODE= is entered at DECODR. for a restart call. The logic then follows from step c.

8.6 EXTERNAL CALLS

KRAKER. - formats and stores data string in data items or arrays

SYSERR. - Handles error processing

9.1 ENCODE=

9.2 PURPOSE

ENCODE= moves data from data items or arrays in one area of core to another core area under control of a given format statement.

9.3 USAGE

Calls to ENCODE= are generated by the FORTRAN statement ENCODE (c,m,v) K where

c = Unsigned integer constant or simple integer variable (not subscripted) specifying the number of characters in the record.

m = Statement number or variable identifier representing the format statement.

v = Variable identifier or an array identifier which supplies the starting location of the bcd record.

K = list of variables.

Initial Entry Point: ENCODI.

Entry Conditions:
A1 = Address of the APLIST
X1 = Pointer to character count

APLIST format:
Pointer to character count
Pointer to format
Pointer to string
List item
.
.
.
APEND (zero word)

Restart Entry Point: ENCODR.

Entry Conditions:
A1 = Address of the APLIST
X1 = List item

APLIST format:

```
List item
   .
   .
   .
APEND (zero word)
```

Exit Conditions: If APEND is minus zero, then the data from the list items will have been packed in the DAT. buffer. If APEND is plus zero, then the data will have been moved from the DAT. buffer to the string address.

9.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: Character count must be less than 151 and greater than zero.

Error Messages:

	Error number
ENCODE CHAR COUNT .GT. 150	85
ENCODE CHAR COUNT .LE. ZERO	85

9.5 GENERAL DESCRIPTION of the ROUTINE

a. ENCODE= is entered the first time at ENCODI. where A0 is saved.

b. The character count is obtained from the first word of the APLIST. If the character count is out of bounds, the appropriate error message is issued.

c. The format pointer and string pointer are obtained from the APLIST.

d. The DAT. buffer is blanked.

e. Initialization parameters are passed to KODER. to prepare for moving coded data.

f. The next list item is obtained. If it is zero go to step h.

g. The data address and length are obtained from the list item and passed to KODER. The data is formatted in the DAT. buffer. Control is transferred to step f.

h. If the list item is plus zero, then a terminal call is made to KODER. where the data is moved from the DAT. buffer to the string area.

i. A0 is restored and control is returned to the calling program.

j. ENCODE= is entered at ENCODR. for restart calls and control is then transferred to step f.

9.6 EXTERNAL CALLS

KODER. - Encodes data items or arrays according to the given format statement.

SYSERR. - Handles error processing

10.1 ENDFIL=

10.2 PURPOSE

ENDFIL= will write an end-of-file indicator on a specified unit.

10.3 USAGE

This routine is called in response to the FORTRAN statement ENDFILE u, where u is the logical unit indicator.

Entry Points: ENDFIL.

Entry Conditions: X1 contains the FIT pointer.

Exit Conditions: An end-of-file will have been written on the specified unit.

10.4 RESTRICTIONS and ERROR MESSAGES

none

10.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FIT address, GETFIT. is called and returns the address in X1.

b. If the file is not opened, then the file is opened as a coded file.

c. An endfile is written by the record manager and control is returned to user.

10.6 EXTERNAL CALLS

GETFIT. - locates FWA of the FIT

11.1 EOF=

11.2 PURPOSE

EOF= checks the previous read operation to determine if an end-of-file (EOF) has been encountered.

11.3 USAGE

This routine is called in response to the FORTRAN function EOF (i) where i = the unit designator.

Entry Point: EOF

Entry Conditions: X1 contains the FIT pointer.

Exit Conditions: X6 contains zero if no EOF found, is not zero if an EOF found. The EOF condition is cleared if an EOF was encountered.

11.4 RESTRICTIONS and ERROR MESSAGES

none

11.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called and the FIT address is returned in X1.

b. X6 is set to zero to indicate no EOF.

c. If either the file is not opened or the FORTRAN endfile bit in the FIT is off, then control is returned to the calling routine.

d. The FORTRAN endfile bit in the FIT is cleared, X6 is set to 1.0, and control is returned to the calling routine.

11.6 EXTERNAL CALLS

GETFIT. - locates FIT address.

12.1 FORSYS=

12.2 PURPOSE

FORSYS= is a multiple entry routine which handles program initialization, error tracing, diagnostic printing, and termination of output buffers.

12.3.0 ENTRY POINTS

12.3.1 Q8NTRY.

Purpose: Initializes I/O buffer parameters.

Entry Conditions

A1 = FWA file vector, which contains one word per programmer declared file - 42/ 0Lfilename, 18/ FWA FIT. The list is terminated by a word containing the complement of the output file line limit.
A0 = SCM field length
X0 = LCM field length

Exit Conditions: The file vector has been transferred to RA+2 list, it is now terminated by a zero word. All FITS have been initialized. Any file substitutions specified on the control statement have been done, as well as any *PL=* parameter.

12.3.2 END.

Purpose: This entry point is called in response to the FORTRAN END statement. If not in (0,0) overlay control is returned to calling overlay, else a dayfile message END XXXXXX is issued, buffers are flushed and the program terminated normally.

Entry Conditions: X1 has address of entry point of program.

12.3.3 EXIT

Purpose: This entry point is called in response to the FORTRAN statement CALL EXIT to terminate a job with EXIT printed in the dayfile.

Entry Conditions: none

12.3.4 STOP.

Purpose: This entry point is called in response to the FORTRAN statement STOP with a message following.

12.3.5 ABNORM.

Purpose: This routine gains control from an execution time routine when a fatal error has been processed. An error message with traceback information is written to the dayfile, and the job terminated.

12.3.6 PACK.

Purpose: Packs up a line of characters in 1R format. The packed words are stored back into the area holding the unpacked characters. The first location preceding the unpacked area is destroyed.

Entry Conditions:
B1 = Number of characters to pack
A2 = Address of start of character string
X2 = First character

12.3.7 BURST.

Purpose: Unpack a line of input characters in 10H format to 1R format in the DAT. buffer. The word preceding the DAT. is destroyed.

Entry Conditions: X3 = total number of characters in record

12.3.8 IOERR.

Purpose: Prints the fatal record manager error number in the OUTPUT file. for file with error

12.3.9 SYSERR.

Purpose: Error tracing and diagnostic printing.

Entry Conditions:

X1 = Error number
X2 = Address of diagnostic message

Exit Conditions: SYSERR. will transfer to SYSTEM for non-standard error recovery, aborts the job, or returns to the calling routine, depending on the type of error being processed. if X1 is zero , then the message is printed and control returned. to the calling routine.

12.5.0 GENERAL DESCRIPTION of the ROUTINE

12.5.1 Q8NTRY.

a. Save the SCM and LCM field lengths in FLSCM. and FLLCM. respectively.

b. Fetch word from FILES. list, if negative go to step f. this list contains all the files declared on the program card and the addresses of the corresponding FIT-s.

c. Fetch word from RA+2 list. If this is zero no file substitution is requested, go to step d. If this is the print limit parameter, compute line limit and save in X5. If it is a file name save it in X7 so that it may be stored in the FIT.

d. If the first word of the FIT is zero, then store the name of the file. Store following fields in the FIT; SDS=YES, EO=AD, OF=N.

e. Decrease parameter count, store file name in RA+2, and fetch next name from FILES. list. If MODEL is GE to 75 do a SETFIT on the file.

f. After all file names have been processed store the OUTPUT file line limit and return to the calling routine.

12.5.2 END.

a. If in overlay other than (0,0) return to the calling overlay.

OR

b. Issue dayfile message END XXXXXX.

c. Call SYSEND. to terminate all output buffers.

d. Terminate the job.

12.5.3 EXIT

a. Issue dayfile message EXIT and transfer control to step c under END.

12.5.4 STOP

a. Move the word STOP followed by up to seven words of message to the DAT. buffer.

b. Write message in the DAT. buffer to the dayfile and transfer control to step c under END.

12.5.5 ABNORM.

a. The last error encountered is written to the dayfile with traceback information.

b. SYSEND. is called to terminate the output buffers.

c. The ABORT macro is used to terminate the job.

12.5.7 SYSERR.

a. The error number and address, and A0 are saved.

b. Locate the address of the FIT for the output file. Search the RA+2 list and the first word of each FIT. If the output file is not located, issue a fatal dayfile message.

c. Open the OUTPUT file if not currently open.

d. If the error number is zero, then no traceback information is generated.

e. If the error number is out of range, then set the number to the last error number.

f. If SYSTEM has been called then control is passed to SE7 in SYSTEM for extended error processing.

g. Print error message with traceback information. From entry point SYSERR. fetch the RJT word which called SYSERR. From this word fetch the TRACE. word, which has the routine name and entry point address, and line number from which the routine was called. This information is printed and the process is repeated using the RJT from the entry point word until the main program is reached or the maximum traceback is reached.

h. If SYSTEMC has been called then control is passed to SYSTEM at SE18.

i. If the error is non-fatal then exit through entry point SYSERR. else issue a fatal error message to the dayfile and abort the job.

14.1 GETFIT=

14.2 PURPOSE

GETFIT= is called whenever it is necessary to obtain the address of an FIT given the file name or tape number.

14.3 USAGE

A call to this routine is made from an object time routine when it is necessary to obtain the FIT address for a file.

Entry Point: GETFIT.

Entry Conditions: X1 contains the address of a word which contains the logical filename or logical unit number.

Exit Conditions: X1 contains the FIT address

14.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: none

Error Messages:

	Error Number
FILE NOT DECLARED	62

14.5 GENERAL DESCRIPTION of the ROUTINE

a. The word containing the filename/number is obtained.

b. If this is not a name the tape number is converted to display code and appended to TAPE.

c. The RA+2 list is searched for the filename. If the name is found the FIT address is placed in X1 and control is returned to the calling routine.

d. If the filename is not found FORSYS= is called to issue error message number 62.

14.6 EXTERNAL CALLS

none

15.1 INPB=

15.2 PURPOSE

INPB= transfers binary data from a file to variables in central memory. If no list is specified, the next logical record is skipped.

15.3 USAGE

This routine is called in response to the FORTRAN statement READ u, L where

L = List of variables to be read
u = Logical unit number

Initial Entry Point: INPBI.

Entry Conditions
A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:
FIT pointer
EOF address
list item
.
.
.
APEND (zero word)

Restart Entry Point: INPBR.

Entry Conditions:
A1 = FWA of the APLIST
X1 = First word of the APLIST

APLIST format:
list item
.
.
.
APEND (zero word)

Exit Conditions: The data from the file will have been transferred to the appropriate variables.

15.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: none

Error Messages:

	error number
END OF FILE ENCOUNTERED, FILENAME - XXXXXXX	65
LIST EXCEEDS DATA, FILENAME - XXXXXXX	88
WRITE FOLLOWED BY READ ON FILE XXXXXXX	89
PARITY ERROR READING (BINARY) FILE - XXXXXXX	90

15.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain FWA of the FIT, GETFIT. is called and returns the address in X1.

b. The EOF address from the APLIST is stored in cell END=.

c. Set flag to denote initial entry.

d. If the FORTRAN endfile bit (FPB) is set in the FIT then issue end-of-file encountered error message.

e. If the parity error bits are set in the FIT then issue a parity error message.

f. If the file position is EOI check if initial call. If yes set the FPB bit and return to the calling routine, else issue list exceeds data error.

g. If the file is not open then open the file.

h. If the current APLIST element is minus zero return to the calling routine. If it is plus zero skip the file to end of record if the file position is not EOR.

i. If the next APLIST element is plus zero set flag.

j. Obtain the data address and word count from the current APLIST element.

k. If initial call then skip to step p.

l. If the DAT. buffer is empty skip to step n.

m. Move the minimum of the word count and the current number of words in buffer to the variable.

n. If all data has been transferred to variable then return to step h.

o. If LEXDER flag is set then issue list exceeds data error message.

p. If the number of words needed is greater than the buffer size or if the next element is plus zero then read data directly into variable, obtain the next APLIST element, and return to step h.

q. Read a maximum of 150 words into the DAT. buffer.

r. If the number of words read exceeds or equals number needed, go to step m.

s. Set LEXDER flag and go to step m.

DATA EXIT FROM RECORD MANAGER

t. A list exceeds data error is issued if an EOF condition exists not on the initial call.

u. If it is the initial call then set the FPB bit in the FIT and return to the calling routine.

15.6 EXTERNAL CALLS

GETFIT. - locates FWA of the FIT.

SYSERR. - Handles error processing

16.1 INPC=

16.2 PURPOSE

INPC= is called to read a list of variables and arrays according to a given format.

16.3 USAGE

This routine is called in response to the FORTRAN statement READ(u,f) L where

u = Logical unit identifier
f = Format statement number
L = List of variables to be read

Initial Entry Point: INPCI.

Entry Conditions:
A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:
FIT pointer
FWA of the format
list item
.
.
.
APEND (zero word)

Restart Entry Point: INPCR.

Entry Conditions:
A1 = FWA of the APLIST
X1 = First word of APLIST

Exit Conditions: One FORTRAN logical record will have been transferred to the given storage locations.

16.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: The record cannot be longer than 150 characters.

Error Messages:	Error number
END-OF-FILE ENCOUNTERED, FILENAME - XXXXXXX	65
WRITE FOLLOWED BY READ ON FILE XXXXXXX	91
PARITY ERROR READING(CODED) FILE, FILENAME - XXXXXXX	92

16.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FIT address, GETFIT. is called and returns the address in X1.

b. If the FPB bit in the FIT is set then issue end of file encountered error message.

c. If the file position is EOI then set the FPB bit in the FIT, store minus zero in the first data list item, and return to calling routine.

d. If the file is not opened then open the file.

e. An initial call is made to KRAKER. One record is read and burst into one character per word in the DAT. buffer.

f. If the current list element is minus zero then control is returned to calling routine.

g. If the current list element is plus zero a terminal call is made to KRAKER. and control returned to calling routine.

h. The data address and word count are obtained from the current list item.

i. KRAKER. is called to format the data item and move it to the proper storage location. Control is transferred to step f.

CODE TRANSFERRED TO FROM KRAKER.

j. One logical record is read.

k. If a zero length record was read the DAT. buffer is blanked else the record is burst into one character per word in the DAT. buffer and control is returned to KRAKER.

16.6 EXTERNAL CALLS

BURST. - Entry point in FORSYS= which bursts a record into one character per word in the DAT. buffer.

GETFIT. - locates FIT address and returns it in X1.

KRAKER. - formats input data according to a given format.

SYSERR. - Handles error processing

17.1 IOCHEC

17.2 PURPOSE

IOCHEC tests the error status field in the FIT for a parity error on a specified unit.

17.3 USAGE

This routine is called in response to the function statement J = IOCHEC(u), where u is the logical unit, and J is the cell in which the result of the parity check is stored immediately after IOCHEC is exited.

Entry Points: IOCHEC

Entry Conditions: X1 contains the address of the filename.

Exit Conditions: X6 contains the parity error test indicator: 0, no error: -1,error.

17.4 RESTRICTIONS and ERROR MESSAGES

none

17.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called with X1 containing the filename or tape number. The FIT address is returned in X1.

b. The error status field of the FIT is checked for parity error status.

c. X6 is returned with zero if no parity error was detected else it is returned with the integer -1.

17.6 EXTERNAL CALLS

GETFIT. - obtains the FIT address

21.1 LENGTH

21.2 PURPOSE

LENGTH will return a count of central memory words transferred on the previous I/O request for a particular file or logical tape number.

21.3 USAGE

This routine is called in response to the FORTRAN external function L= LENGTH (J) where L is an integer cell in which the routine stores a count of the number of central memory words transferred from file J during the last I/O request.

Entry Points: LENGTH

Entry Conditions: X1 contains the address of the filename or of the tape number.

Exit Conditions: X6 contains number of words transferred to/from J.

21.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: Meaningful results are not guaranteed if a LENGTH function is requested on a buffer unit before requesting a UNIT function.

Errors: none

21.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called with the filename or tape number in X1. The FIT address is returned in X1.

b. The record length in characters is obtained from the FIT.

c. The character count is converted to word count and placed in X6. Control is returned to the calling program.

21.6 EXTERNAL CALLS

GETFIT. - obtains the FIT address.

24.1 OPENMS

24.2 PURPOSE

This routine links a FORTRAN program to the common FORTRAN/ COBOL random file utility to open a random file.

24.3 USAGE

This routine is called in response to the FORTRAN statement CALL OPENMS (u,ix,lngth,t) where

u = Unit designator
ix = First word address in central memory of the array which will contain the index
lngth = Length of index
t = Type of index
t = 0 for a numbered index
t = 1 for a name index

Entry Points: OPENMS

Entry Conditions:
A1 - FWA of the APLIST
X1 - FIT pointer

Exit Conditions: The random file will be opened.

24.4 RESTRICTIONS and ERROR MESSAGES

24.5 GENERAL DESCRIPTION of the ROUTINE

none

a. Store linkage in FORSYS= to jump to CLOS.RI when closing files.

b. GETFIT. is called with the filename or tape number in X1. The FIT address is returned in X1.

c. A return jump is made to the random utility routine to open the file.

d. If an error has occurred print the appropriate error message and abort. If no error has occurred then return to calling routine.

24.6 EXTERNAL CALLS

GETFIT. - obtains the FIT address.

25.1 OUTB=

25.2 PURPOSE

This routine transfers one logical record of binary information from storage to a file unit.

25.3 USAGE

This routine is called in response to the FORTRAN statement WRITE (i) L where

i = file unit number
L = list of variables

Initial Entry Point: OUTBI.

Entry Conditions:

A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:

```
FIT pointer
record length
list item
     .
     .
     .
APEND (zero word)
```

Restart Entry Point: OUTBR.

Entry Conditions:

A1 = FWA of the APLIST
X1 = List item

APLIST format:

```
list item
     .
     .
     .
APEND (zero word)
```

Exit Conditions: The logical record will have been transferred to the file unit.

25.4 RESTRICTIONS and ERRORS

Restrictions: none

Error Messages

	error number
PARITY ERROR ON LAST READ ON FILE XXXXXX	93

25.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FIT address, GETFIT= is called and returns the address in X1.

b. If the parity error bits are set in the FIT then issue a parity error message.

c. The total word count is computed for the write. This is done by executing the compiled code and summing the word count for the various APLIST. When a plus zero is encountered for the list element the sum is stored in the RL field of the FIT. The first list item is obtained from the APLIST.

d. If the current APLIST element is minus zero return to the calling routine. If it is plus zero flush any data that may be in the DAT. buffer and return to the calling routine.

e. Obtain the variable address and word count from the list element.

f. If this is a call with only one list item then write data from variable area and return to the calling routine.

g. If the data will not fit in the DAT. buffer then go to step h else move the data to the DAT. buffer, fetch the next list item and return to step d.

h. Write the data directly from the variable area if no data exists in the DAT. buffer, else write data from the DAT. buffer and reset buffer pointers.

i. Return to calling routine if terminal call.

j. If the word count is less than buffer size move data from variable area to DAT. buffer, update pointers, fetch next list item and go to step d, else write data directly from data area, fetch next list item and go to step d.

25.6 EXTERNAL CALLS:

GETFIT. - locates FIT address

SYSERR. - Handles error processing

26.1 OUTC=

26.2 PURPOSE

OUTC= transfers coded data from storage to a file unit according to a given format.

26.3 USAGE

This routine is called in response to the FORTRAN statement WRITE(u,f) L or PRINT f, L where

u = Logical unit number
f = Format statement to be read
L = List of variables to be written

Initial Entry Point: OUTCI.

Entry Conditions
A1 = FWA of the APLIST
X1 = FWA of the FIT

APLIST format:
FWA of the FIT
format pointer
list item
.
.
.
APEND (zero word)

Restart Entry Point: OUTCR.

Entry Conditions:
A1 = FWA of the FIT
X1 = List item

APLIST format:
list item
.
.
.
APEND (zero word)

Exit Conditions: Upon exit one FORTRAN logical record will have been written to the file unit.

26.4 RESTRICTIONS and ERROR MESSAGES

Restrictions: The maximum record size is 150 characters.

Error Messages:

	Error Number
OUTPUT FILE LINE LIMIT EXCEEDED	83
PARITY ERROR ON LAST READ, FILE XXXXXX	94

26.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FIT address, GETFIT. is called and returns the address in X1.

b. If the parity error bits are set in the FIT then issue a parity error message.

c. Open the file if it is not presently open.

d. Fill the DAT. buffer with blanks.

e. Make initialization call to KODER.

f. Fetch current APLIST element.

g. If the current list element is minus zero return to the calling routine.

h. If the current list element is plus zero a terminal call is made to KODER. and control is returned to the calling routine.

i. Obtain the variable address and the word count from the list element.

j. Call KODER. to format the data in the DAT. buffer and return to step f.

26.6 EXTERNAL CALLS

GETFIT. returns the FWA of the FIT in X1.

SYSERR. - Handles error processing

27.1 RANMS

27.2 PURPOSE

This routine is the FORTRAN random file processing utility. It is called from other object time routines to process the actual I/O.

27.3 USAGE

This routine is called from the following routines.

27.3.2 OPENMS

Entry Point: OPEN.RI

Purpose: This section of RANMS makes initial checks of the file and index. If there are no errors the file is opened.

Entry Conditions:
X1 = FWA of the FIT
X6 = FWA of the APLIST

APLIST Format:
FWA of CM buffer for the master index
Address of word containing CM buffer length
Address of word containing index type flag:
0 for number index, 1 for name index

Exit Conditions: The file will have been opened or an error code returned in X2.

27.3.3 READMS

Entry Point: READ.RI

Purpose: This section of RANMS reads a record from a random file.

Entry Conditions:
X1 = FWA of the FIT
X6 = FWA of the APLIST

APLIST Format:
FWA of data area in CM
Address of word containing index name/number key
Address of word containing record length

Exit Conditions: The record will have been read or an error number returned in X2.

27.3.5 WRITMS

Entry Point: WRIT.R1

Purpose: This section of RANMS writes a data record to a random file.

Entry Conditions:
X1 = FWA of the FIT
X6 = FWA of the APLIST

APLIST Format:
FWA of data to be written
Address of word containing record length
Address of word containing index name/number key
Address of word containing rewrite flag
(optional) Address of a word containing sub-index flag or 0

Exit Conditions: The record will have been written or an error number returned in X2.

27.4 RESTRICTIONS AND ERROR MESSAGES

Restrictions and errors are documented in the calling routines.

27.5 GENERAL DESCRIPTION of the ROUTINE

27.5.2 OPEN.RI

a. The file is open with file organization set to word-addressable.

b. If the record type in the FIT is not WA then a file control card was used and an error status is returned.

c. Read the index control word from the file.

d. Zero fill the area reserved for the master index.

e. If this is a new file then store appropriate fields in the FIT and return to calling routine.

f. If there is not enough room for the record then return to calling routine with error flagged.

g. Store appropriate fields in the FIT and read the master index.

h. If file index does not match parameter then return with error flagged, else return normally.

27.5.3 READ.RI

a. Call routine RFSUB2 to find file index entry corresponding to call.

b. If entry was not found return with error number in X2.

c. Unpack index control for parameters to record manager.

d. Read record from file and return to calling routine.

27.5.5 WRIT.RI

a. Set flag if this is a rewrite.

b. Set flag if there is a sub-index.

c. Call routine RFSUB2 to find file index entry corresponding to call.

d. If there is no room in index for new entry return with error number in X2.

e. If rewrite was requested then go to step h.

f. Write data record at the end of the file.

g. Update the pseudo EOI pointer and return to calling routine.

h. If index entry was not found return with error number in X2.

i. If there is room for the record then rewrite record as requested and return to the calling routine.

j. If write at EOI option is set then write the record at EOI and return, else return with error number in X2.

28.1 READMS

28.2 PURPOSE

READMS links a FORTRAN caller to the common FORTRAN/COBOL random file utility to read a record from a random file.

28.3 USAGE

This routine is called in response to the FORTRAN statement CALL READMS (u,fwa,n,k) where

u = Unit designator
fwa = FWA of the record
n = Record length
k = Index key

Entry Point: READMS

Entry Conditions:
A1 = FWA of the APLIST
X1 = FIT pointer

APLIST format:
FIT pointer
FWA of the record
Record length
Index key
APEND (zero word)

Exit Conditions: The record will have been transferred from the file to central memory.

28.4 RESTRICTIONS and ERROR MESSAGES

none

28.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called to locate the FWA of the FIT.

b. READ.RI is called to do the actual I/O.

c. If an error number is returned from READ.RI then print the error message and terminate, else return to the calling routine.

28.6 EXTERNAL CALLS

GFTFIT. - Locates the FWA of the FIT.

READ.RI - Does the actual transfer of data.

29.1 REWIND=

29.2 PURPOSE

This routine rewinds a FORTRAN file.

29.3 USAGE

This routine is called in response to the FORTRAN statement REWIND u, where u is the logical unit number.

Entry Point: REWIND.

Entry Conditions:
X1 = FIT pointer

Exit Conditions: Upon exit the file will have been rewound.

29.4 RESTRICTIONS and ERROR MESSAGES

none

29.5 GENERAL DESCRIPTION of the ROUTINE

a. If X1 does not contain the FIT address GETFIT. is called and returns the address in X1.

b. If the file is not open then set the open flag in the FIT to rewind and return to the calling routine.

c. If the last operation was a write then write an endfile indication on the file.

d. Clear the FORTRAN end-of-file bit in the FIT.

e. Rewind the file.

f. Set the open-close flag in the FIT to never open.

g. Return to the calling routine.

29.6 EXTERNAL CALLS

GETFIT. - locates FWA of the FIT.

30.1 STINDX

30.2 PURPOSE

STINDX selects a different array to be used as the current index to a random file.

30.3 USAGE

This routine is called in response to the FORTRAN statement CALL STINDX (u,ix,lngth,t) where

- u = Unit designator
- ix = FWA of array which will contain the index
- lngth = Length of index
- t = Type of index

Entry Point: STINDX

Entry Conditions:

- A1 = FWA of the APLIST
- X1 = FIT pointer

APLIST format:

- Unit designator
- FWA of array which will contain the index
- Index length
- Type of file

30.4 RESTRICTIONS and ERRORS

none

30.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called to locate FWA of the FIT.

b. STIN.RI is called to change the index.

c. If an error number is returned from STIN.RI then print the error message and terminate, else return to the calling routine.

30.6 EXTERNAL CALLS

GETFIT. - returns FWA of the FIT

STIN.RI - does the actual change of index

31.1 SYSTEM

31.2 PURPOSE

SYSTEM is a multiple entry routine which handles non-standard error processing.

31.3 ENTRY POINTS

31.3.1 SYSTEM

Purpose: This entry point is called to issue a supplied message with a supplied error number. If the error number is zero then all files for which the last operation was a write are to be flushed.

Entry Conditions:
X1 = Address of cell containing the error number
A1 = Address of the parameter list

APLIST format:
Address of cell containing the error number
Address of the error message

31.3.2 SYSTEMC

Purpose: Changes entry to SYSTEM-S error table according to arguments passed.

Entry Conditions:
A1 = Address of parameter list
X1 = Address of word containing error number

APLIST format:
Address of word containing error number
Address of list containing consecutive locations:
Fatal/non-fatal(fatal = 1, non-fatal = 0)
Print frequency
Print frequency increment (only significant if preceeding word = 0) special values: 0 - never list error, 1 - always list error, or x - list error only the first x times.
Print limit
Non-standard recovery address
Maximum traceback limit

If any word in the argument list is negative, the value already in table entry is not to be altered.

31.3.3 SYSLNK.

Purpose: Link SYSTEM to FORSYS=, thus for all calls to SYSERR. SYSTEM will be called for possible extended error processing.

Entry Conditions: none

31.3.4 SF7

Purpose: Control enters here from FORSYS= to update information in the error table kept in SYSTEM. This is not a true entry point but is one of the jumps plugged by SYSLNK.

Entry Conditions:
X2 = Address of message
A0 = Error number

31.3.5 SYS88

Purpose: SYS88 prints a summary of each error received during the execution of the job and the number of times it was received.

Entry Conditions: none

31.5.1 SYSTEM

a. Fetch the address of the error message.

b. Call SYSLNK. to link FORSYS= and SYSTEM together.

c. Store return address in entry word for SYSERR.

d. Fetch the error number, if nonzero jump to SYSERR+1 else call SYSEND. to end output buffers and return to the calling routine.

31.5.2 SYSTEMC

a. If the error number is out of bounds then return to the calling routine.

b. Set the appropriate fields in the error table according to the parameter list.

c. Call SYSLNK. to link FORSYS= and SYSTEM together and return to the calling routine.

31.5.3 SYSLNK.

a. Cells in SYSTEM which have branches to labels in SYSTEM are stored in FORSYS=.

31.5.4 SE7

a. Fetch the error table entry.

b. If the print limit is zero or if the print frequency and print frequency increment is zero then set flag not to print error message. If the print frequency does not equal print frequency increment then increment print frequency by one and set flag not to print, else clear the print frequency limit.

c. If the detection count is 4095 then go to step g.

d. If the message address is zero then call was made to bump count only, thus return to calling routine.

e. Transfer control to FORSYS= to print message if flag is set to print. Control is returned after printing.

f. If non-standard recovery is specified or if non-fatal error go to step h.

g. If the SKIP flag is zero then call SYS88 to print error summary and call SYSEND. to end output buffers. Issue fatal error message to dayfile and end the job, else return to the calling routine.

h. If no recovery address is specified then return to calling routine through SYSERR.

i. If fatal error with recovery has been specified store jump to abort in entry point of recovery routine and transfer control to entry point +1.

j. For a non-standard error with recovery specified the entry address of the routine in which the error was detected is stored into the entry point of the recovery routine. Control is transferred to the recovery routine.

31.5.5 SYS88

a. Print column headings.

b. Fetch error table entry. If detection count is zero then fetch next entry and repeat.

c. Print the error number and the total number of times it was detected.

32.1 UNIT

32.2 PURPOSE

UNIT returns the status of a buffered file.

32.3 USAGE

This routine is called in response to the FORTRAN statement function UNIT(i), where i is the unit designator.

Entry Point: UNIT

Entry Conditions: X1 contains the complement of either the logical file name or the logical unit number.

Exit Conditions: Upon exit UNIT will set X6
-1, unit ready, no error
+0, unit ready, EOF encountered
+1, unit ready, parity error encountered

32.4 RESTRICTIONS and ERROR MESSAGES

none

32.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called and returns the FIT address in X1.

b. X6 is set to 1.0.

c. Issue a record manager CHECK to ensure completion of the I/O.

d. If the last operation was not a write and the file position is EOF then clear the FPB bit in the FIT, set X6 to zero, and return.

e. If the parity bits in the FIT are not set then return, else set X6 to -1.0 and return.

32.6 EXTERNAL CALLS

GETFIT. - locates the FWA of the FIT.

33.1 WRITMS

33.2 PURPOSE

WRITMS links a FORTRAN caller to the common FORTRAN/COBOL random file utility to write a record to a random file.

33.3 USAGE

This routine is called in response to the FORTRAN statement CALL WRITMS (u,fwa,n,k) where

- u = Unit designator
- fwa = FWA where the record is to reside in CM
- n = Record length
- k = Index key
- A1 = FWA of the APLIST
- X1 = FIT pointer

APLIST format:

- Unit designator
- FWA of the record
- Record length
- Index key
- APEND (zero word)

Exit Conditions: The record will have been transferred from central memory to the file.

33.4 RESTRICTIONS and ERROR MESSAGES

none

33.5 GENERAL DESCRIPTION of the ROUTINE

a. GETFIT. is called to locate the FWA of the FIT.

b. WRIT.RI is called to do the actual I/O.

c. If an error number is returned from WRIT.RI then print the error message and terminate, else return to the calling routine.

33.6 EXTERNAL CALLS

GETFIT. - locates the FWA of the FIT

WRIT.RI - does the actual transfer of data