AAEC/TM567

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

AEREREAD - A REREAD, INCORE READ/WRITE AND AUTOMATIC PRINTER CARRIAGE OVERFLOW ROUTINE FOR IBM 360 OS FORTRAN USERS

by

R.E. DAVIDS

September 1970

ISBN 0 642 99384 X

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AEREREAD - A REREAD, INCORE READ/WRITE AND AUTOMATIC PRINTER CARRIAGE OVERFLOW ROUTINE FOR IBM 360 OS FORTRAN USERS

by

R. E. DAVIDS

ABSTRACT

This routine permits the same data to be read by the IBM 360 OS Fortran user more than once. It also enables input and output transfers to be made to or from a specified Fortran array in similar manner to the transfers made by the ENCODE and DECODE statements of other Fortran systems. An automatic printer carriage overflow feature is enabled on finding the end of a page on an on-line printer. The routine is based on a REREAD routine published in the IBM Installation Newsletter.

National Library of Australia card number and ISBN 0 642 99384 X

CONTENTS

1. INTRODUCTION

A rereading capability whereby one record of data could be read more than once under different formats was available on the IBM 7040 computer and was widely used, particularly in data editing and variable data programs. An automatic printer carriage overflow (or page-skip) function was also available on the 7040, where it had been programmed into the 1401 resident IOCP program.

When the A.A.E.C. changed to an IBM 360 computer a routine was needed to perform these functions. The original idea for the method of operation of this routine was obtained from a routine published in the IBM Installation Newsletter, the latest version of which appeared in the December 1968 issue.

The AEREREAD routine has gone through many modifications since it first came into use early in 1967 and has now attained a certain measure of Release independence. This report corresponds to the version which works under Release 17 of the IBM 360 Operating System. It has only been tested on a PCP system, although there is no reason to doubt that it would work under systems with MFT and MVT. The printer carriage overflow section may, however, require some modification.

2. METHOD OF OPERATION

Input and output for the users of the full (non-basic) Fortran system under the IBM 360 OS are controlled by the Fortran library routine IHCFCOMH (or IHCECOMH if the Extended Error Message facility is specified). The alias for both these routines is IBCOM#. This routine calls routine IHCFIOSH (or IHCEFIOS) to do input and output. The alias for these two routines is FIOCS#.

The AEREREAD routine intercepts all calls between IBCOM# and FIOCS# once it has been activated and simulates returns from FIOCS# to IBCOM# for all incore operations. The printer carriage overflow is effected by simulating the OS PRTOV macro each time a line is printed.

2.1 Activating the AEREREAD Routine

The interception of calls is initiated by means of a call to the entry point REREAD. This routine changes the address constant for FIOCS# in the IBCOM# routine to an address within the AEREREAD routine. The letters RR are also written on the printer, starting in column 130, to indicate to the user that REREAD is in use and to ensure that a data control block for the printer on unit FT03F001 has been opened. The address of this block is then found by searching the data extent block queue and the task input output table via the communication vector table and the task control block, and is stored within the AEREREAD routine. Subsequent calls to REREAD are ignored. REREAD is implicitly called if PRNTOV is called. However, for all other functions it must be called first somewhere at the beginning of the program. In overlay programs AEREREAD must be in the same segment as IBCOM# and FIOCS#, generally the root segment.

2.2 Operation

Once the routine has been activated it monitors all input, output and control requests between IBCOM# and FIOCS#. The various operations available are integrated and depend on switches set by the SETBUF, PRNTOV and PRNTOF entry points described in Section 3. The rereading capability is always enabled once REREAD has been called, and cannot be disabled.

There are five types of call from IBCOM# to FIOCS#:

(a) Initialisation (code = 0)
(b) READ operation (code = 1)
(c) WRITE operation (code = 2)
(d) Control operation (code = 3)
(e) Close data sets (code = 4)

The code is the first byte of the argument list, which is pointed to by register zero on entry to AEREREAD.

Two types of return can be made from FIOCS# to IBCOM#:

(f) Normal
(g) Error.

AEREREAD can handle the calls in two ways:

(i) The call can be passed to FIOCS#, with original arguments and return address (return is to IBCOM#).
(ii) The call can be passed to FIOCS#, with simulated arguments and a return to AEREREAD, from which a return to IBCOM# is then made.

Hence interception can be made both on the way into and out of FIOCS#.

During the initialisation call the argument list originally supplied to IBCOM# by the problem program is also available and is pointed to by register 2. It is important to distinguish clearly between these two argument lists since both are used by AEREREAD. The are referred to in this report as the IBCOM# and FIOCS# argument lists respectively.

When it gets control, AEREREAD immediately passes on all calls to FIOCS# by method (i) above, except the initialisation and WRITE types. The initialisation call is checked for the unit number (IBCOM# argument list), for whether it is formatted or unformatted, and for whether it is input or output (FIOCS# argument list).

All normal input operations are passed on to FIOCS# by method (ii) above. On return the contents of registers 2 and 3, which contain respectively the buffer address and length, are saved in AEREREAD. For normal output operations the output unit number is saved for checking later during a WRITE operation.

If input is requested from unit 99 (the unit can be specified in Fortran as either an integer constant or variable) then the request is not passed on to FIOCS# but registers 2 and 3 are restored to what they were after the last READ, and return to IBCOM# is effected.

If the incore READ/WRITE switch is on (set by SETBUF) then both input and and output initialising calls from IBCOM# are checked to see if they correspond to the unit that has been nominated by the programmer for incore operations. If the nominated unit was used then registers 2 and 3 are set to the in-core area indicated by the programmer. No checking is done to see if the size of the array indicated by the programmer is exceeded or not. If more data than can fit in the buffer are transferred, then Fortran decides it needs a new record. The extra information then goes into the same area in core and, in the case of a WRITE operation, over-writes what was initially written. In the case of a READ operation, the same information is obtained more than once.

With a WRITE operation the saved unit from the last output initialisation call is checked. If the printer carriage overflow switch is on and if the unit is 3 or 6, the printer overflow mask is set for channel-12 overflow testing in the selected printer data control block. Control is then passed to FIOCS# by method (i) above. The standard printer unit in use at the A.A.E.C. is FT03F001, but this can easily be changed by altering two statements in the program. The control program then looks after the skipping operation once a channel-12 overflow has been detected.

AEREREAD tests that the unit is a 1403 or 1404 printer, before setting the overflow mask. If printed output is on tape the system does not work, but it is a simple matter to institute a line count procedure and insert a USASI control character into the first byte of the record. The line count must then, of course, be reset every time the user does his own page skip, and the first character needs to be continually monitored. Also the buffer address must be saved during initialisation and the requirement for finding the printer DCB address no longer exists. This second method is better for systems that use spooling.

An error return from FIOCS# is passed straight back to IBCOM which then prints the appropriate error message, so that a record which gave an error return cannot be reread.

3. USER'S GUIDE

3.1 Initiation

Insert a

CALL REREAD (no arguments)

or

CALL PRNTOV (0)

statement at the beginning of the program, or anywhere logically before AEREREAD functions are required.

3.2 Rereading

Use either

```
READ (99,n) list
```

or

```
READ (N,n) list
```

where N = 99

n = format statement number or array name, and

list = list of variables as described in the Fortran manual.

Notes

(a) The same record can be reread an unlimited number of times.

(b) END= and ERR= exits may be specified but could never occur from incore operations.

(c) Although REREAD does work for unformatted READ statements there is little point in its use and trouble may occur with spanned records (see Fortran Programmer's Guide for description of spanning). This also applies to incore READ/WRITE statements.

Example

In this example a card is read in, and then reread in a different mode depending on a code in column one.

```
      READ (1,88) IA,IB,IC
88 FORMAT (3I1)
      IF (IA) 10,20,30
10 READ (99,87) A,B,C
87 FORMAT (3X,3E15.7)
      GO TO 100
20 READ (99,86) CA,CB
86 FORMAT (20A4)
      .
      .
      .
30 etc.
```

3.3 Printer Carriage Overflow Skipping

To activate use

```
CALL PRNTOV (0)
```

The zero argument is optional at present but will be required when the line-count version (see Section 3.3.1) is introduced.

To deactivate use

CALL PRNTOF

Notes

(a) Page skipping commences on the first channel 12 detected on the printer after activation.

(b) Unlimited activation and deactivation is allowed.

(c) Attempting to activate the activated state or to deactivate the deactivated state has no effect.

3.3.1 Line count version (not standard at A.A.E.C.)

To activate with this version

CALL PRNTOV (U,N)

where N = number of lines per page desired, and

U = Fortran logical unit on which this monitoring is desired. Only one unit may be activated at a time. U = 0 indicates that the standard version (see above) is to be used.

3.4 Incore READ/WRITE

To activate the facility include

CALL SETBUF (N,IA,IL)

where N = Fortran logical unit to be used to indicate incore operations,

IA = name of array to or from which incore operation will take place, and

IL = length of area available in bytes.

Notes

(a) If N is positive the activation switch is set OFF after each use. If N is negative the activation switch remains in effect or set ON, until it is set OFF by N turning positive.

(b) Only one Fortran logical unit may be activated at any one time. Setting a new unit cancels the last one. No DD statement is required for the incore unit. If one is provided then incore operations take place while the switch remains activated for that unit and will do input/output on other occasions it is used. If a unit is used before it is activated and no DD card has been supplied then a standard Fortran error message indicates this.

(c) The array IA must be suitably dimensioned to accommodate the largest operation used. IA may be a subscripted variable if it is desired that the operation should start in the middle of an array.

(d) IL will be four times the array dimension for single precision or INTEGER*4 variables, or eight times the area dimension for double precision or REAL*8 variables. The buffer lengths may of course be indicated to be less than array size but should never be more than the array size.

Bring the facility into operation by using any of the following:

```
READ (N,n) list
READ (l,n) list
WRITE (N,n) list
WRITE (l,n) list
```

where l = integer constant representing unit activated by SETBUF,
N = integer variable equal to unit activated by SETBUF,
n = format statement number (constant or array name), and
list = Fortran list of variables (optional).

Notes

(a) END= and ERR= exits may be specified but will be ignored.

(b) The buffer is not cleared to blanks for each WRITE statement, so that single characters may be changed using the T format.

(c) After each use the activation switch is set OFF if the unit specified was positive. The switch remains set ON if the unit specified was negative.

(d) The unit number may be between 1 and 99. Zero is not a valid unit. If 99 is specified then the standard REREAD function is overwritten, but it may be restored in the normal way (see (c) above). Unit 99 cannot be redesignated for rereading by using the CALL REREAD statement.

(e) The note (c) in Section 3.2, regarding unformatted input/output, applies here.

Example 1 Use of I/O Conversion Routines

```
C     CONVERT INTERNAL FLOATING VARIABLE PQ TO A
C     CHARACTER STRING IN ARRAY IZ
      DIMENSION IZ (4)
      PQ = 16.342
      CALL SETBUF (11,IZ,16)
      WRITE (11,99) PQ
   99 FORMAT (F16.3)
      .
      .
```

```
      STOP
      END
```

Example 2 Byte Manipulation

```
C     CHANGE LETTER F TO NUMBER 8
      DIMENSION NP(3)
      DATA NP/'ABCDEFGHIJKL'/
      CALL SETBUF (22,NP,12)
      WRITE (22,99)
   99 FORMAT (T6,'8')
      .
      .
      STOP
      END
```

Example 3 Setting of the Activation Switch

```
      DIMENSION EX(3)
C     CALL SETBUF (-73,EX,12)
      ACTIVATION SWITCH NOW SET ON FOR UNIT 73
      READ (73,89) I
   99 FORMAT (I9)
      READ (73,88) Z
   88 FORMAT (F6.4)
C     ACTIVATION SWITCH STILL ON FOR UNIT 73
      CALL SETBUF (25,EX,12)
C     ACTIVATION SWITCH ON FOR UNIT 25
      READ (25,89) I
C     ACTIVATION SWITCH SET OFF
      END
```

Example 4 General Example Incorporating Main Features of AEREAD

This example uses the incore WRITE to generate a variable format definition which is then written incore and printed. The result is that, starting off with one asterisk in column 2, each line printed contains one more asterisk than the previous one.

```
1  DIMENSION IA(20),IB(4)
2  DATA IA/20*'    '/
3  CALL REREAD
5  DO 10 I=2,80
```

```
    6 N=33
    7 CALL SETBUF(8,IB,16)
    8 WRITE(8,999)I
  999 FORMAT('(T',I2,',1H*)')
    9 CALL SETBUF(N,IA,80)
   11 WRITE(N,IB)
   12 WRITE(3,998)IA
  998 FORMAT(1X,20A4)
   10 CONTINUE
   13 STOP
   14 END
```

Notes (numbers refer to statement numbers in example)

1. IA is output area. IB is area that will hold a variable generated format.
7. This sets up incore operations on array IB for unit 8.
8. A new format statement is generated incore in array IB.
9. This initialises an incore operation on array IA using this time a random variable unit 33 (for the sake of the example only).
11. The next line incore is written to array IA.
12. This prints out IA to the printer (statements 9 and 11 are really redundant and 12 could have been rewritten as WRITE (3,IB)).

Example 5 Conversion of CDC Fortran REREAD Feature

To simulate

```
      ENCODE (4,99,V) B
   99 FORMAT (F4.2)
```

use

```
      CALL SETBUF (33,V,4)
      WRITE (33,99) B
   99 FORMAT (F4.2)
```

3.5 Storage Requirements

The routine that has only the standard printer carriage overflow function uses about 600 bytes of core storage (260 hex) and the 'CSECT' AAEPSW uses 4 bytes of core storage. If it were desired to activate more than one incore unit at once then more storage space would be needed.

4. PRINT SWITCH

A facility is provided whereby the Fortran programmer can check whether a

line has been printed. If the printer carriage overflow has been activated and an output record written on the specified printer unit, and if the device is a printer, then a word in the special 'CSECT' AAEPSW is set to zero. The Fortran programmer can check this value using labelled COMMON.

Example

```
      SUBROUTINE CHECK (*)
      COMMON/AAEPSW/IA
      IF (IA) 10,20,10
   20 IA = 1
      RETURN 1
   10 RETURN
      END
```

This routine would check whether a line was printed anywhere in the program since the last call to the routine and whether the device being used was an on-line printer or not. The option is used by a trace routine PRINTT (available at the A.A.E.C.) which prints many trace calls on one line by suppressing the printer advance. When this routine finds that the program has also printed a line then it prints the next trace on a new line to prevent overwriting of the program's output line.

5. RELEASE INDEPENDENCE

In so far as AEREREAD relies on a calling sequence and register usage which is internal to the IBM subroutine library (and also non-standard) it is not Release-independent. This calling sequence was changed when the extended error handling package was introduced in Release 15/16 because an error return address was introduced in the argument list to FIOCS#. However it seems unlikely that the calling sequence will be changed again in the near future. The actual calling sequences are included as comments in the listing of the program in Appendix 1.

Provided the calling sequences and internal register usages do not change, the present version will be Release-independent. It already copes effectively with the two versions of IBCOM# and FIOCS# currently available (that is, with and without the extended error handling package).

The method of overwriting the FIOCS# address in IBCOM#, by modifying one of the instructions at the beginning of IBCOM, executing it and then restoring it, was copied from the REREAD routine published in the IBM Installation Newsletter.

6. CONCLUSION

The AEREREAD routine should provide a powerful data editing tool for Fortran programmers, particularly when used in conjunction with the SCAN free-input routines (Bennett and Pollard 1967). It is also useful when converting Fortran programs

that were written for other Fortran compilers which allow the use of the ENCODE and DECODE statements, such as the CDC Fortran compilers. The Share Fortran Project has suggested the inclusion of a Fortran character-type variable and this may be implemented in the IBM Fortran Compilers. It is already available in the University of Waterloo's Fortran Compiler. If a variable were coded for the unit in the incore READ/WRITE statements then the conversion of Fortran source program could be easily accomplished by removing the CALL SETBUF cards or providing a dummy SETBUF routine and equivalencing the first two arguments to SETBUF (that is, inserting EQUIVALENCE (N,IA)).

Users of this routine should nevertheless bear in mind that it is a non-standard facility and may let them down when it is most needed, for example when it is required to run the program on another machine, or under a different operating system. Programmers who would like to be able to run their programs anywhere should use strictly standard Fortran as defined by the U.S.A.S.I. standard.

7. ACKNOWLEDGEMENTS

I am indebted to the author (unknown) of the REREAD routine published in the IBM Installation Newsletter and to Mr. R. G. J. Wood of IBM, Australia, who first implemented it at the A.A.E.C.

8. REFERENCES

Bennett, N. W. and Pollard, J. P. (1967). - SCAN - A free input subroutine for the IBM/360. AAEC/TM399.

IBM Field Support Centre, Australia and New Zealand Region (1968). - Installation Newsletter, Issue 68-24.

IBM Systems Reference Library (1968). - System/360 Fortran IV Language, Form C28-6515-7.

IBM Systems Reference Library (1968). - System/360 Operating System, Fortran IV (G and H) Programmer's Guide, Form C28-6817-0.

United States of America Standards Institute (1966). - American Standard Fortran, X39-1966.

APPENDIX 1

SOURCE LISTING OF AEREREAD


```
RERD     TITLE   'AAEC REREAD SUBROUTINE'
********* RELEASE INDEPENDENT VERSION AFTER R.17   5 MAY 69  R.E.D.
*              A SOLUTION TO THE FORTRAN REREAD PROBLEM USING DSRN99
*              TO TRIGGER THE INCORE MOVE.  A ONE-TIME CALL TO REREAD
*              CAUSES IBCOM TO INSPECT ALL FURTHER I/O CALLS AND SAVE
*              THE POINTERS TO THE BUFFERS OF DATA FOR A  READ
*                    REGARDLESS OF THE PHYSICAL UNIT INVOLVED.
*              A SUBSEQUENT          READ(99,FORMAT) LIST
*              WILL CAUSE THE SAME DATA FROM THE LAST FORMAT READ TO
*              BE TRANSMITTED TO THE 'LIST' BY THE CONVERSION CALLED
*              FOR BY 'FORMAT'.
*
*        QUERIES  TO ROB DAVIDS, A.A.E.C., LUCAS HEIGHTS, N.S.W.
*
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* CALLING SEQUENCE FROM FORTRAN TO IBCOM#
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        L        L,=V(IBCOM#)
*        CNOP     0,4
*        BAL      R,0(L)                FRDWF
*              OR
*        BAL      R,4(L)                FWRWF
*        DC       XL0.4'PI',XL0.4'UI',AL3(UNIT)
*        DC       AL1'FI',AL3(FORMAT)
*        DC       AL4(EOFADD)           OPTIONAL
*        DC       AL4(ERRADD)           OPTIONAL
*                 WHERE PI = X'0' IF NEITHER EOF NOR ERR,
*                            X'1' IF EOF ONLY,
*                            X'2' IF ERR ONLY,
*                            X'3' IF BOTH EOF AND ERR.
*                 UI = X'0' IF UNIT IS AN INTEGER CONSTANT,
*                      X'1' IF UNIT IS A VARIABLE NAME,
*                      X'4' IF UNIT IS A STANDARD SYSTEMS UNIT.
*                 FI = X'00' IF FORMAT IS A STATEMENT LABEL,
*                      X'01' IF FORMAT IS AN ARRAY NAME.
*
*        L        L,=V(IBCOM#)
*        BAL      R,8(L)                FIOLF
*        DC       XL1'L',XL0.4'T',XL0.4'X',XL0.4'B',XL1.4'D'
*                 WHERE LENGTH(L) = SIZE(IN BYTES) OF THE ITEM,
*                 TYPE(T) = LOGICAL/INTEGER/REAL/COMPLEX/LITERAL,
*                 AND X, B, D ARE THE INDEX, BASE, DISPLACEMENT
*                 WHICH SPECIFY THE ADDRESS OF THE ITEM.
*
*        L        L,=V(IBCOM#)
*        BAL      R,12(L)               FIOAF
*        DC       XL1'SPAN',AL3(ADDRESS)
*        DC       XL1'L',XL0.4'T',XL2.4'ELEMENTS'
*                 WHERE SPAN REFERS TO STRUCTURED ARRAYS,
*                 ADDRESS = BEGINNING LOCATION OF THE ARRAY,
*                 LENGTH(L) = SIZE(IN BYTES) OF EACH ITEM,
*                 TYPE(T) = LOGICAL/INTEGER/REAL/COMPLEX/LITERAL,
*                 ELEMENTS = NUMBER OF ITEMS IN THE ARRAY.
```

```
*
*         L         L,=V(IBCOM#)
*         BAL       R,16(L)                 FENDF
*
          SPACE     3
* ERROR CONDITIONS
*         INVALID CHARACTER IN FORMAT STATEMENT.
*         ATTEMPT TO READ OR WRITE PAST END OF RECORD.
*         FOR OTHERS, SEE FIOCS ROUTINE.
*
*         L         L,=V(IBCOM#)
*         CNOP      0,4
*         BAL       R,20(L)                 FRDNF
*               CR
*         BAL       R,24(L)                 FWRNF
*         DC        XL0.4'PI',XL0.4'UI',AL3(UNIT)
*         DC        AL4(EOFADD)             OPTIONAL
*         DC        AL4(ERRADD)             OPTIONAL
*                   WHERE PI = X'0' IF NEITHER EOF NOR ERR,
*                              X'1' IF EOF ONLY,
*                              X'2' IF ERR ONLY,
*                              X'3' IF BOTH EOF AND ERR.
*                   UI = X'0' IF UNIT IS AN INTEGER CONSTANT,
*                        X'1' IF UNIT IS A VARIABLE NAME,
*                        X'4' IF UNIT IS A STANDARD SYSTEMS UNIT.
*
*         L         L,=V(IBCOM#)
*         BAL       R,28(L)                 FIOLN
*         DC        XL1'LENGTH',XL0.4'0',XL0.4'X',XL0.4'B',XL1.4'D'
*                   WHERE LENGTH = SIZE (IN BYTES) OF THE ITEM,
*                   AND X, B, D ARE THE INDEX, BASE, DISPLACEMENT
*                   WHICH SPECIFY THE ADDRESS OF THE ITEM.
*
*         L         L,=V(IBCOM#)
*         BAL       R,32(L)                 FIOAN
*         DC        XL1'SPAN',AL3(ADDRESS)
*         DC        XL1'LENGTH',AL3(ELEMENTS)
*                   WHERE SPAN REFERS TO STRUCTURED ARRAYS,
*                   ADDRESS = BEGINNING LOCATION OF THE ARRAY,
*                   LENGTH(L) = SIZE(IN BYTES) OF EACH ITEM,
*                   ELEMENTS = NUMBER OF ITEMS IN THE ARRAY.
*
*         L         L,=V(IBCOM#)
*         BAL       R,36(L)                 FENDN
*
          SPACE     3
* ERROR CONDITIONS
*         INPUT LIST LONGER THAN LOGICAL RECORD.
*         ATTEMPT TO OUTPUT MORE THAN 255 RECORDS IN A LOGICAL RECORD.
*         FOR OTHERS, SEE FIOCS ROUTINE.
          SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* FIOCS# CALLING SEQUENCE FROM IBCOM#
*ENTRY POINTS--ONE ENTRY POINT WITH PARAMETERS AS FOLLOWS-
*
```

```
*     INITIALIZATION
*          LA     GRX,DSRN                    POINTER TO DATA SET REF NO
*          L      L,=V(FIOCS#)
*          BALR   R,L
*          DC     AL1(0)
*          DC     AL1(FQUALS)
*          DC     A(ERREXT)        ERROR EXIT ADDRESS
*                 WHERE DSRN IS OF THE FORM XL1'FLAG',AL3(UNIT)
*                 AND FQUALS = X'F0' FOR FORMATTED INPUT
*                              X'FF' FOR FORMATTED OUTPUT
*                              X'00' FOR NON-FORMATTED INPUT
*                              X'0F' FOR NON-FORMATTED OUTPUT
*
*     READ OPERATION
*          L      L,=V(FIOCS#)
*          BALR   R,L
*          DC     AL1(1)
*          DC     AL1(0)
*          DC     A(ERREXT)        ERROR EXIT ADDRESS
*
*     WRITE OPERATION
*          L      GRX,RECLEN
*          L      L,=V(FIOCS#)
*          BALR   R,L
*          DC     AL1(2)
*          DC     AL1(0)
*          DC     A(ERREXT)        ERROR EXIT ADDRESS
*                 WHERE RECLEN = NUMBER OF BYTES OF OUTPUT
*
*
*     CONTROL OPERATION
*          LA     GRX,DSRN
*          L      L,=V(FIOCS#)
*          BALR   R,L
*          DC     AL1(3)
*          DC     AL1(CQUALS)
*          DC     A(ERREXT)        ERROR EXIT ADDRESS
*                 WHERE DSRN IS OF THE FORM XL1'FLAG',AL3(UNIT)
*                 AND CQUALS = X'00' FOR BACKSPACE BLOCK
*                              X'01' FOR REWIND
*                              X'02' FOR WRITE END OF FILE MARK
*
*     CLOSE DATA SETS
*          L      L,=V(FIOCS#)
*          BALR   R,L
*          DC     AL1(4)
*          DC     AL1(0)
*          DC     A(ERREXT)        ERROR EXIT ADDRESS
*
*INPUT--ALL INPUT PARAMETERS ARE FROM THE MODULE IHCFCOMH,
*    AND ARE CONTAINED IN THE REGISTERS 0, 2, OR 3.
*
*OUTPUT--POINTERS TO BUFFERS, AND BUFFER LENGTHS, ARE GIVEN TO EITHER
*    FCOMH, THE CALLING ROUTINE, OR TO THE CONTROL PROGRAM.
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
          SPACE 3
          ENTRY REREAD
          EXTRN IBCOM#
          EXTRN FIOCS#
AEREREAD  CSECT
* REGISTER DEFINITIONS
R         EQU   0                          RETURN REG. USED IN CUR. IBCOM
L         EQU   1                          LINKAGE REG. USED IN CUR. IBCOM
GRX       EQU   2                          FIRST ARGUMENT
GRY       EQU   3                          SECOND ARGUMENT
*         ROUTINE SETBUF SPECIFIES A BUFFER TO BE USED BY A LOGICAL
*         FORTRAN UNIT FOR EITHER FORMATTED OR UNFORMATTED INPUT OR
*                   OUTPUT.  ON THE NEXT USE BY FORTRAN OF THAT LOGICAL
*                   UNIT:-
*                        ON INPUT:
*                   THE INPUT RECORD WILL BE OBTAINED FROM THE BUFFER
*                   SPECIFIED.
*                        ON OUTPUT:
*                   THE OUTPUT INFORMATION WILL BE PUT IN THE AREA IN
*                   CORE SPECIFIED AS A BUFFER AND NO I/O WILL OCCUR.
*                   CALL SETBUF (IU,IA,IL)
*                   IU   :     LOGICAL UNIT REQUIRED, MAY BE AN INTEGER
*                              CONSTANT OR VARIABLE
*                              IF IU IS NEGATIVE BUFFER REMAINS SET
*                   IA   :     AREA TO BE USED
*                   IL   :     LENGTH OF BUFFER IN CHARACTERS
*         ERROR CONDITIONS:
*                   EOF AND ERR CONDITIONS MAY BE SPECIFIED IN THE
*                   READ AND WRITE STATEMENTS CONCERNED BUT SUCH EXITS
*                   WILL NOT OCCUR FROM THE IN-CORE OPERATIONS.
          ENTRY SETBUF
          USING *,15
SETBUF    SAVE  (14,4),,*
          LM    2,4,0(1)           GET ARGUMENT ADDRESSES INTO REG 2-4
          ST    3,RER2A            SAVE AREA ADDRESS
          L     2,0(2)             GET UNIT ADDRESS
          LTR   2,2                TEST UNIT VALUE
          BC    11,SB1             BRANCH IF UNIT NOT NEGATIVE
          LPR   2,2                MAKE POSITIVE
          O     2,=X'01000000'                SET SWITCH CONTINUOUS SETBUF
SB1       ST    2,RER2U            SAVE UNIT
          L     4,0(4)             GET LENGTH ADDRESS
          ST    4,RER2L            SAVE LENGTH
          MVI   RER1+1,0           ACTIVATE SWITCH
          RETURN (2,4),T           BACK TO FORTRAN
          ENTRY  PRNTOV,PRNTOF
          USING *,15
PRNTOF    MVI   RER9+1,X'F0'       DEACTIVATE PRINT SWITCH
          BCR   15,14
          USING *,15
PRNTOV    MVI   RER9+1,X'00'       ACTIVATE PAGE SKIP SWITCH
          BALR  15,0
          USING *,4
REREAD    SAVE  (14,12),,*
          LR    4,15                        REG 4 NEW BASE REGISTER
```

```
RER10A   BC     0,RER13                 FIRST ENTRY SWITCH
         OI     RER10A+1,X'F0'            SET SWITCH FOR FIRST ENTRY
         LM     15,1,ADIBCOM            IBCOM, FIOCS AND REREAD ADRR
         MVI    74(15),X'50'            MODIFY INSTRUC IN IBCOM TO ST
         EX     0,74(15)                ST REREAD ADDR IN IBCOM
         MVI    74(15),X'58'            RESTORE INSTRUCTION TO LOAD
         BAL    14,4(15)                OPEN PRINTER BY WRITING
         DC     A(3)                    FT LOGICAL UNIT 3, INTEGER FRM
         DC     X'01'                   VARIABLE ARRAY TYPE FORMAT
         DC     AL3(MSG)                FORMAT STATEMENT
         L      15,ADIBCOM              TERMINATE
         BAL    14,16(15)
*        THIS SECTION FINDS THE ADDRESS OF PRINTER DCB ON FT03F001
         L      5,16             ADDRESS OF CVT TABLE
         L      6,4(5)                  TCB ADDRESS ADDRESS
         L      6,4(6)                  CURRENT TCB ADDRESS
         LM     7,8,8(6)                ADDR OF DEB & TIOT
RER12    L      9,24(7)                 DCB ADDRESS
         LH     10,40(9)                TIOT OFFSET
         L      7,4(7)                  ADDRESS OF NEXT DEB
         LA     10,4(8,10)              DDNAME ADDR IN TIOT
         CLC    0(8,10),RER12U          IS IT?
         BNE    RER12                   LOOK UP DEB QUEUE
*                                       MUST FIND SINCE JUST CREATED
         ST     9,RER9A                 SAVE PRINTER DCB ADDRESS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
RER13    RETURN (14,12),T                BACK FROM CALL REREAD
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 3
****             ENTRY HERE FROM IBCOM WHEN REREAD HAS BEEN ACTIVATED
         USING *,1
RER      STM    0,15,RER1S       SAVE REG 4-15
         DROP   1
         LR     11,1             REG 11 IS NEW BASE REG
         USING  RER,11
         LR     5,2              POINTER TO ARG LIST FOR IBCOM
         LR     4,0              POINTER TO ARG LIST FOR FIOCS
         CLI    0(4),X'00'       IS IT INIT OPERATION ?
         BNE    RER7
         TM     0(2),X'05'       CHECK IF UNIT IS INTEGER CONSTANT
         BZ     RER1             YES
         TM     0(2),X'04'       CHECK FOR STANDARD SYSTEMS UNIT
         BO     RER4             YES - GO CHECK IF  INPUT
*                STANDARD SYSTEMS UNITS ARE: READ 1,A
*                                            PRINT 99,
*                                            PRINT 99,
         L      5,0(5)           MUST BE VARIABLE UNIT
*                UNIT ADDRESS NOW IN REG 5
RER1     BC     15,RER4          SETBUF ACTIVATION SWITCH
         CLC    1(3,5),RER2U+1   CHECK LOGICAL UNIT VS SETBUF UNIT
         BNE    RER4             NOT EQUAL GO CHECK IF INPUT
         TM     RER2U,1                   CONTINUOUS SETBUF ?
         BO     RER11                     DON'T RESET SETBUF ACTIVATION SW
         MVI    RER1+1,X'F0'     DEACTIVATE SETBUF SWITCH
RER11    TM     1(4),X'0F'       IS IT AN OUTPUT OPERATION?
```

```
RER2      L      2,RER2A          LOAD AREA ADDR SET BY SETBUF
          L      3,RER2L          LOAD LENGTH ADDR SET BY SETBUF
TOIBCOM   LA     1,6(4)           PUT RETURN ADDRESS IN REG 1    R.17 **
IBEX      LM     4,15,RER1S+16    RESTORE  REGS
          BCR    15,1             RETURN TO IBCOM AS IF FROM FIOCS
RER3      MVI    RER8+1,X'F0'     ACTIVATE WRITE TO BUFFER SWITCH
          B      RER2
RER4      TM     1(4),X'0F'       IS IT AN INPUT OPERATION ?
          BO     RER6             NO
          CLC    1(3,5),RER4N+1   IS UNIT = 99 ?
          BE     RER5             YES
          L      1,ADFIOCS         IOCS ADDRESS TO REG1
          MVC    RER4A(2),0(4)    PROVIDE ARG FOR FIOCS
          BALR   0,1              GO TO FIOCS
RER4A     DC     X'0000'          ARG FOR FIOCS
RER41     BC     15,ERROR         ERROR RETURN                    R.17**
          ST     2,RER4B          STORE BUFFER ADDR FOR REREAD
          ST     3,RER4L          STORE BUFFER LENGTH FOR REREAD
          B      TOIBCOM          BACK TO IBCOM
ERROR     LA     1,2(4)                  ERROR EXIT TO IBCOM R.17
          B      IBEX                                        R.17
RER5      L      2,RER4B          RESTORE REG2 & 3 TO WHAT THEY WERE
          L      3,RER4L          ..AFTER THE LAST READ
          B      TOIBCOM
RER6      L      5,0(5)           LOAD UNIT
          ST     5,RER6A          SAVE WRITE UNIT
          MVI    RER8+1,X'00'     DEACTIVATE WRITE TO BUFFER SWITCH
          B      TOFIOCS          CONTINUE ON TO FIOCS
RER7      CLI    0(4),X'02'       IS IT A WRITE OPERATION
          BNE    TOFIOCS          NO
RER8      BC     0,TOIBCOM        WRITE-TO-BUFFER SWITCH
RER9      BC     15,TOFIOCS        PAGE SKIP SWITCH - INITIALLY DISABLED
          TM     RER6A+3,X'02'    UNIT 3 OR 6
          BZ     TOFIOCS          NOT 2
          TM     RER6A+3,X'05'
          BC     11,TOFIOCS       NOT  4 OR 1  (2ND HALF OF TEST)
          L      1,RER9A          ADDRESS OF UNIT BLOCK FOR PRINTER
          TM     17(1),X'48'      IS DEVICE A PRINTER
          BC     14,TOFIOCS       NO  IGNORE
          OI     16(1),16         SET BIT ON IN DCP
          SR     1,1              ZERO REG 1
          L      8,=V(AAEPSW)     ADDRESS OF PRINTSWITCH
          ST     1,0(8)           SET TO ZERO IF LINE WAS PRINTED
TOFIOCS   L      1,ADFIOCS
          LM     4,15,RER1S+16    RESTORE REGISTERS
          BCR    15,1             OF TO FIOCS
RER1S     DC     16F'0'           SAVE AREA
ADIBCOM     DC   A(IBCOM#)                DO NOT ..
ADFIOCS     DC   A(FIOCS#)                ... CHANGE ORDER THESE 3 ADCONS
RERAD     DC     A(RER)           REREAD ENTRY ADDRESS .. LOADED WITH LM
RER2U     DC     F'0'             UNIT SAVED FROM LAST CALL SETBUF
RER2A     DC     F'0'             BUFFER ADDR FROM CALL SETBUF
RER2L     DC     F'0'             BUFFER LENGTH FROM SETBUF
RER4B     DC     F'0'             BUFFER ADDR SAVED FROM LAST READ
RER4L     DC     F'0'             BUFFER LENGTH SAVED FROM LAST READ
```

```
RER4N    DC    F'99'          REREAD UNIT
RER6A    DC    F'0'           LAST WRITE UNIT SAVE AREA
RER9A    DC    A(RER4N-57)              PRINTER DCB ADDR; INIT INOCUOUS
RER12U   DC    C'FT03F001'              PRINTER DDNAME AT A.A.E.C.
MSG      DC    C'(T130,2HRR)'           FORMAT STATEMENT TO OPEN PTR
AAEPSW   CSECT
         DS    1F                       SET TO ZERO IF LINE WAS PRINTED
* THIS CAN BE TESTED FROM FORTRAN BY USING
*      COMMON /AAEPSW/ IVAR
*         THIS IS USED BY PRINTT TO DECIDE NEW LINE
         END
```

APPENDIX 2

FLOWCHART OF REREAD, SETBUF AND PRINTSKIP FUNCTIONS

THIS ROUTINE INTERCEPTS CALLS BETWEEN IBCOM & FIOCS

IN THIS CHART FIOCS = IHCFIOSH & IBCOM = IHCFCOMH