Interner Bericht
DESY F1-82/01
August 1982

THE INPUT/OUTPUT SOFTWARE FOR THE 370/E EMULATOR

by

D. Notz

The Input/Output Software for the 370/E Emulator

by

D. Notz

Deutsches Electronen-Synchrotron DESY, Hamburg

June 1982

Abstract

The input/output programs of the 370/E emulator are described. The 370/E is connected via a control computer (NORD, TMS9900) to the IBM. We explain in detail the buffer handling and the required modifications to run IBM programs on the 370/E.

Contents

I. Introduction

The 370/E emulator is a processor which is able to process IBM 370 code. It was developed by H. Brafman and R. Fall at the Weizmann Institute, Rehovot, Israel. One can therefore run programs either on the IBM or on the emulator without recompiling or translating all programs. Programs which have been developed and tested on an IBM can be downloaded to the 370/E without any change. It is not necessary to translate the code as it is needed on the 168/E emulator. Before running a program on the 370/E one has to link it together with the 370/E input/output routine. This is done by the linkage editor.
Like the IBM the 370/E has a memory which contains data and instructions and which is a direct image of the IBM memory. Nearly all IBM instructions are implemented. Only commercial instructions which work on non binary representations of numbers are not implemented.
For the user the 370/E looks like a box, 60 cm long, 40 cm wide and 30 cm hight, with one input/output cable. Inside this box one can find up to 14 boards.

1) Control board
2) Integer board
3) Floating point mantissa board
4) Floating point exponent board
5) Multiply board
6) Interface board
7) - 14) Memory boards for one Mbyte

The speed of the 370/E is of the order of 60 % - 75 % of an IBM 370/168 depending on the program and the IBM model.

This paper explains in chapters I to III the general ideas how input/output is performed in FORTRAN programs by IBM and how the interface routines work on the 370/E. Chapters IV - VI describe in more detail these routines and chapter VII the buffer organization and tables. This part of the manual is useful for people who want to implement this system on the 370/E.

At DESY, the 370/E is connected via a NORD computer and the DESY online net to the IBM 370/168.

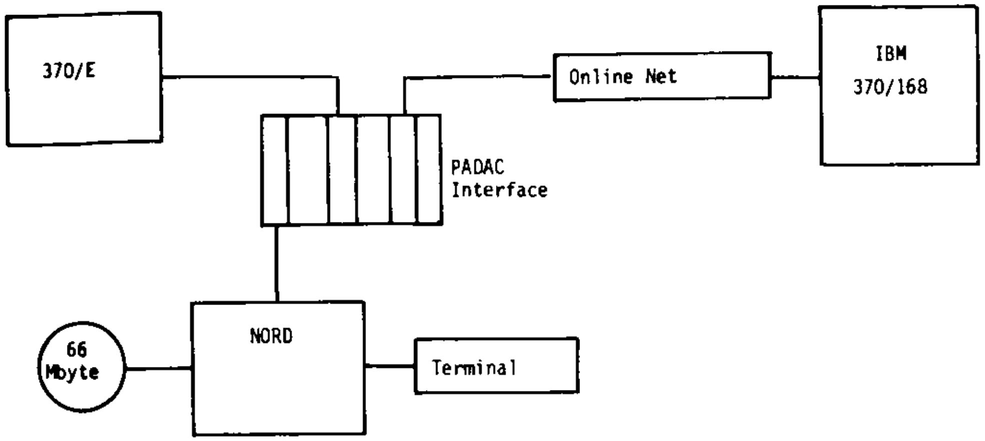


Fig. 1 The 370/E is connected via PADAC to a NORD10 or NORD100. The NORD has a connection to the IBM via a TMS microprocessor and an IBM 2701 unit with parallel data adapter.

The speed of the online net is of the order of 8 μsec/byte. In order to keep the dead time for experiments on the net to a low level one should avoid transfer rates above 1 transfer/sec where one can send ~20 kbytes per transfer. This transfer rate is sufficient to load programs from the IBM to the 370/E and to run CPU intensive programs with low input/output rates. In this environment the NORD only establishes the transfer of buffers between the 370/E and the IBM. It can therefore be replaced by a microprocessor like the TMS 9900 for PADAC (PADAC is the standard interface at DESY).

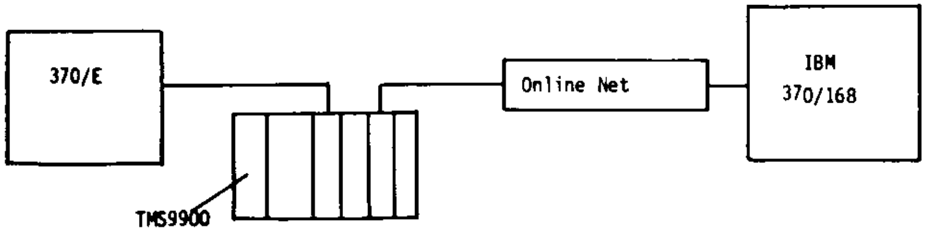


Fig. 2 Offline application. A Monte Carlo program is loaded from the IBM. Input/Output is done to the IBM discs.

In experiments the 370/E can get the input data from the online computer which must have a link to an IBM in order to get the programs.

From the programmers point of view all input/output of data can be performed by a FORTRAN READ/WRITE statement.
In this paper we describe the mechanisms to perform READ and WRITE and how information is exchanged between the 370/E and the IBM.

II.1 FORTRAN Input/Output at the IBM

It is our goal to run IBM programs on the 370/E without any changes. Any input and output should be done via READ/WRITE statements. For a better understanding of the following chapters we describe in this section how input/output is performed at the IBM.

Suppose you have a simple program like

```
   I = 0
   WRITE(6,2)I
 2 FORMAT(1X,'TEXT',I4)
   STOP
   END
```

The compiler then generates several calls to the input/output package IBCOM# :

```
   64 (IBCOM#)   to initialize the job
    4 (IBCOM#)   to initialize the write operation
    8 (IBCOM#)   to write I
   16 (IBCOM#)   to finish the write operation
   68 (IBCOM#)   to terminate the job
```

IBCOM# calls FIOCS# to request a buffer. This buffer is filled with the formatted information. FIOCS# requests via supervisor calls (SVCs) space in memory for the buffers and READ/WRITE operations from the supervisor (Fig. 3).

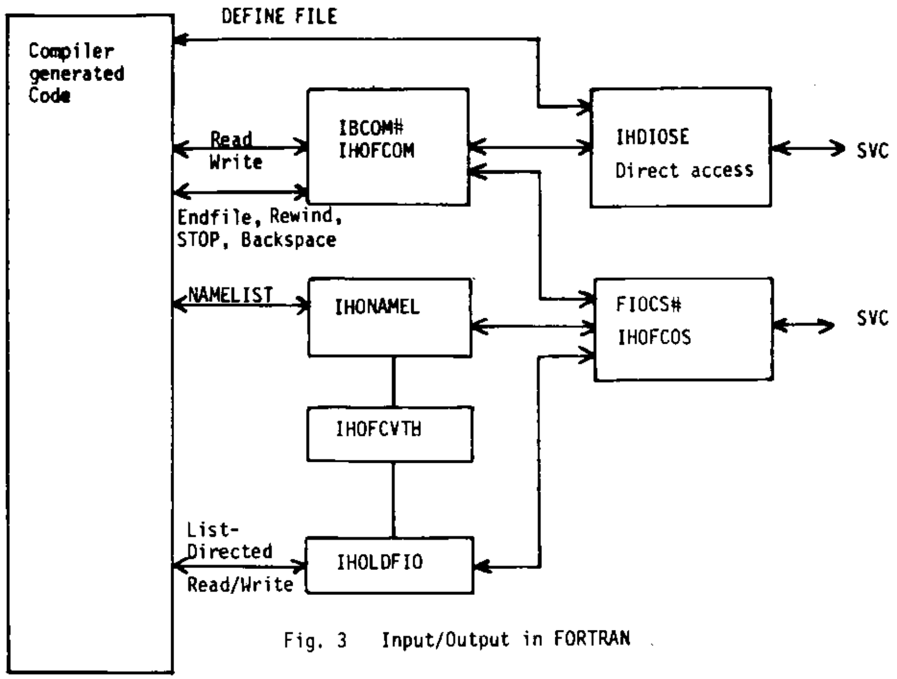


Fig. 3 Input/Output in FORTRAN

II.2 Input/Output at the 370/E

At the 370/E we use the same code as on the IBM. If the user wants to perform I/O via READ/WRITE the compiler generated code calls IBCOM#. IBCOM# is also called by the FORTRAN library if errors occur (like negative square roots). IBCOM# then calls FIOCS#. On the 370/E we use our FIOCS# to do the buffer handling. For each I/O unit a unit block and two buffers are created in COMMON/IHCBF2/. The size of the buffers and records are defined by the IHOUAC table. If a buffer is full for write it is sent to the IBM via the control computer. During this transfer a second buffer is filled. Instead of sending the data to the IBM the control computer can write the buffers also to its local discs. For a READ, a buffer is requested from the IBM. While the first reading is serviced a second buffer is filled by the IBM and sent to the 370/E.

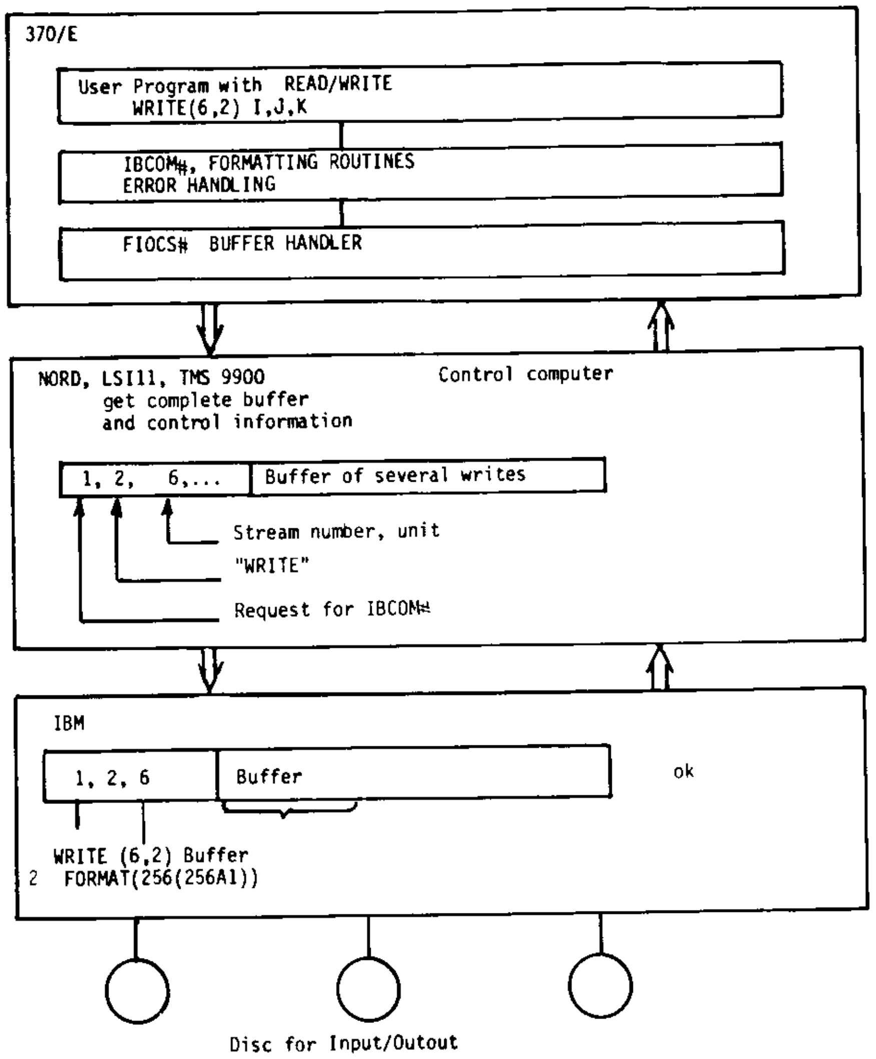


Fig. 4 A user's READ/WRITE results in filling a buffer which is then sent to the IBM. The control computer only transfers buffer to/from the IBM.

The IBM online program handles the input/output of the 370/E. Output is done directly via IBCOM# by WRITE operations if the records in the buffers are complete. For incomplete records the segments are collected at the IBM and written later on after the record is complete. The reading of data is more complicated. Due to the pipelining of double buffers in the 370/E and the IBM one has to know ahead how many words the user on the 370/E wants to read. Input is therefore done by special routines which access FIOCS# at the IBM.

The whole input/output procedure can be tested on the IBM if one uses two IBCOM#s where one IBCOM# and FIOCS# are renamed to CCOM and PIOCS#.

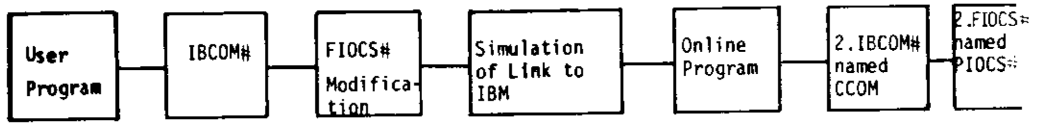


Fig. 5 One can test the input/output of the 370/E by simulating the IBM link and using 2 IBCOM#s within one job.

III.1 Submitting of JOBs to the 370/E from the IBM

In this chapter we describe how jobs can be processed on the 370/E. Jobs are prepared at the IBM by using an editor like TSO, NEWLIB or Wylbur. The following description is only correct if the 370/E is running at DESY but it can be easily modified for other installations.

1) The mainprogram must be written as a subroutine with the name STA370. This subroutine may have a STOP at the end instead of a RETURN statement. CALL DCBSET for each file. Compile the routine which may call other routines.

2) LINK the program. The first module which is loaded must be the system of the 370/E which then calls STA370.
Copy member SYST370E from TASS01.SOURCE and TASS01.LIBRARY into your library.

   If you link the program from the terminal under NEWLIB then define:

```
    MEMBER  = SYST370E
    LIBRARY = your library containing STA370
              other libraries
    MODULE NAME = any name, i.E.  E370TEMP
```

   If the program is linked in a batch job:

```
// EXEC FCL
      :
//LKED.SYSLIN DD DSN = TASS01.LIBRARY(SYST370E),DISP=SHR
// DD
// DD
      :
//LKED.SYSLMOD DD DSN = xxxxxx.yyyy (E370TEMP),DISP=SHR
                        \____ ____/          any name
                             V
                        your library
```

3) The linked system load module must be submitted to the job queue of the 370/E. One has to prepare a file which contains all the job control cards for the program and the data sets. The format is fixed.
   Example:

```
//F1BNOTOO JOB TIME=10
//STEPOO EXEC PGM=xxxxxx.yyyy(E370TEMP)
//LISTFILE DD DSN=xxxxxx.LIST. File name for listing
//FT08F001 DD DSN=xxxxxx.yyy.  File name for unit 8
//FT09F001 DD DSN=xxxxxx.yyy.  File name for unit 9.
```

   Assume the control cards are in file ZZZ.AAA

4) The job can than be submitted by
   a) CALL 'TASS01.LIBRARY(SUB370E)'
         type in name of file containing job control cards:
      ZZZ.AAA

```
b)  //JOBLIB DD DSN=TASS01.LIBRARY,DISP=SHR
    // EXEC PGM=SUBM370E
    //FT05F001 DD *
    ZZZ.AAA
```

5) If the job has finished on the 370/E you may inspect the printed results with

```
LIST  'xxxxxx.LIST'    or
PRINT 'xxxxxx.LIST'
```

III.2 The Program in the 370/E

In the following three sections we describe the structure of the programs in the 370/E, in the NORD and in the IBM.
The first part of the user program in the 370/E must contain the system E37SYS. E37SYS has a reference to the user's program STA370. Inside of E37SYS are on fixed locations pointers to the various programs and tables. These pointers are used by the control computer to do input/outout and to handle interrupts. The general structure of E37SYS is shown in Fig. 6.

When the program is started the registers are resetted and IBCOM# is initialized. Control is then passed to STA370. For an input/output operation information is exchanged via COMMON/CPLIST/ LISTPT,LISTPF. LISTTO contains addresses and lengths of blocks which are sent to the IBM via the control processor. LISTFR points to the addresses where the answer from the IBM should be written. After several WRITEs a buffer is full.
The 370/E generates a supervisor call SVC1 and stops:
The NORD reads LISTPT, the pointer to LISTTO. Then control information and buffers are transferred via DMA to the NORD. The NORD also reads LISTFR indicating where the answer from the IBM should be placed.
The processor is then restarted. It generates an SVC2 to wait for the answer of the IBM. The NORD then transfers the IBM buffer into the 370/E (Fig. 7). In the new interface SVC1 and SVC2 will be replaced by SIO.

If the processor stops due to an error condition (addressing error, divide check) the NORD places the actual address, the length of the last instruction, the condition code and an interrupt code at the PSW locations and starts the processor at the interrupt service routine. This routine saves the registers and makes a trace back. According to the option table IHOUATBL execution continues or terminates.

```
Address
(Hex)
 0   Reset registers
     jump to MAIN2
     Save area
78   COMMON/CPLIST/LISTPT      Pointer to LISTTO. (Buffer to IBM)
7C   LISTPF                    Pointer to LISTFR (Buffer from IBM)
80   Text 'SYST370E82/05/28
     VIHCBUF      Pointer to COMMON/IHCBUF
     VIHCBF2      Pointer to COMMON/IHCBF2/
                  for buffers and blocks
     VIHOUAC      Pointer to unit table for LRECL, BLKSIZE, IRECFM
     VIHOUAT      Pointer to option table for error handling (CALL ERRSET)

     PSWADD       Program status word.  Address.
     PSWLCC       Program status word. Length. Condition Code
     INTCODE      Interrupt code
     VINTSERV     Entry point for interrupt service routine
     REGSAV       Save area for registers
     FLTSAV       Save area for floating point registers

Interrupt service routine
MAIN2 set up IBCOM#
IHOUAC Unit table
FIOCS# Buffer handler interface routine
MVCOM  Service routines
```

Fig. 6 Layout of the system for the 370/E. The pointers are on fixed locations and are known by the control computer.

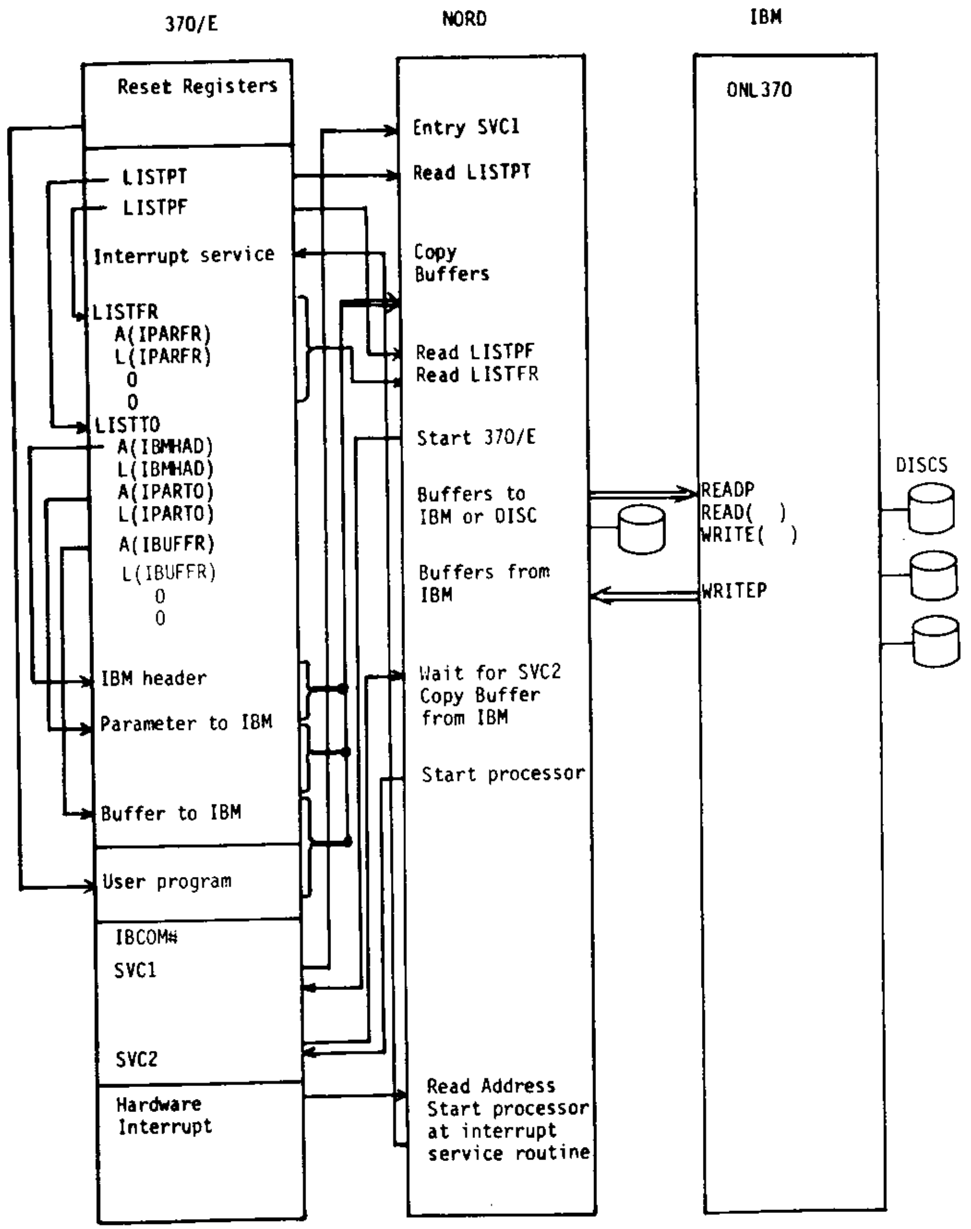


Fig. 7 Information exchange between 370/E, NORD and IBM. The 370/E generates a SVC1 if it wants to transfer a buffer and a SVC2 to wait for the answer.

III.3 The Program in the Control Computer (NORD)

There are several possibilities for a control computer for the 370/E. It could be a microprocessor (TMS 9900, NORD100/E emulator) without any discs, a mini-computer (LST11, PDP11, NORD10) with terminals and small discs or a large computer (VAX) with big discs. In all cases one needs a link to an IBM to transfer the linked load modules. Input/Output to files can be performed via an IBM if a fast link is available or to local discs or tapes of the control computer.
Due to this large variety the program in the control processor should be as small as possible. The program in the NORD10 at DESY is shown in Fig. 8

LAIBM loads the Online program at the IBM using the protocol for the DESY online net. JOBWT sends a message to the online program and waits for a job. If there is a job in the queue the files are allocated and the code is downloaded to the 370/E. The program then waits for interrupts of the 370/E in SVCWT.
For SVC1 the NORD reads the buffer addresses, transfers the buffer from the 370/E to the NORD and restarts the processor.
For SVC2 the 370/E is waiting for a buffer. The NORD knows from the previous SVC1 to which location the input buffer should be stored. After the buffer transfer the 370/E is restarted. A SVC5 indicates the end of a job. The NORD can close the files and ask for another job.
If the processor stops due to an error the NORD places the address and condition code into a fixed location of the 370/E and starts an interrupt service routine in the 370/E. This routine can then do a traceback and abort the program . This works under the assumption that this part of program is not destroyed. It is also possible that the NORD produces a DUMP of registers and memory at the IBM.

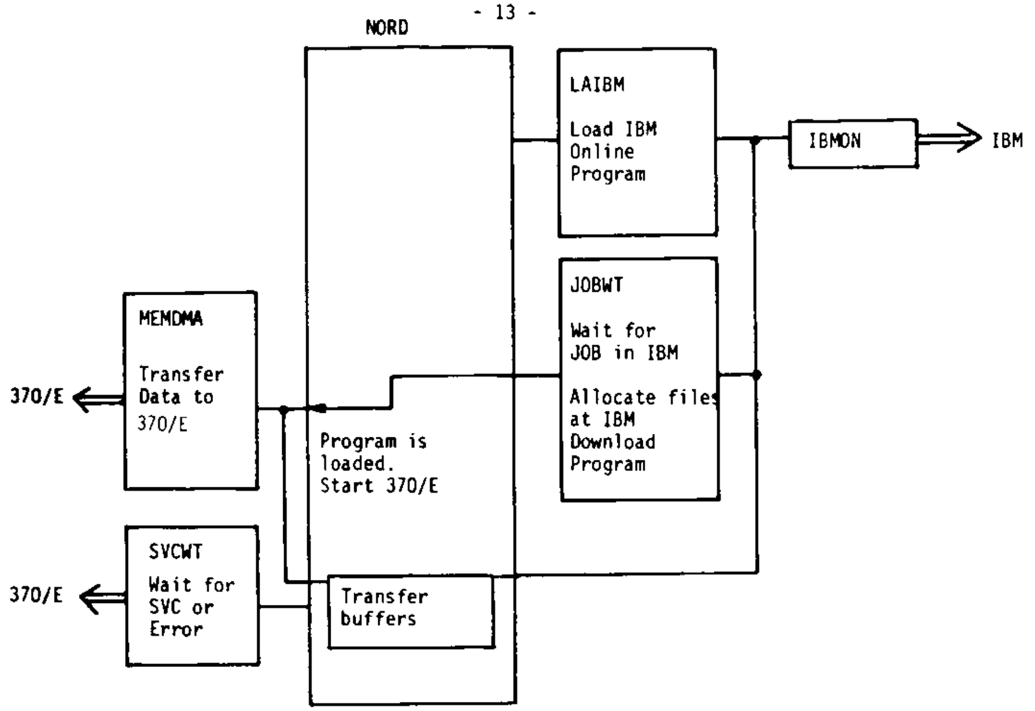


Fig. 8 Program in control computer. Load program and transfer buffers.

III.4. The Online Program in the IBM

The Online Program is loaded at the IBM by the supervisor for the online net. It is then started and waits for interrupts of the NORD.
The NORD sends a message and waits for a job. CKJOB is then called and reads the disc for the 370/E job queue. If there is a job waiting the NORD requests allocation of files. CKJOB allocates the files. The filenames are also taken from the job queue disc. The Online program knows which files are allocated and which dataset organisation is used for each file. Then the loadmodule is allocated and transferred to the 370/E via the NORD. As the online program knows the size of the program and the allocated files the unit table in the loadmodule can be updated. The 370/E therefore gets the information about files and can abort the job if an illegal file is referenced.
The IBM now waits for buffers and reads or writes them to the different files. At the end of a job all buffers are closed and the files are deallocated.

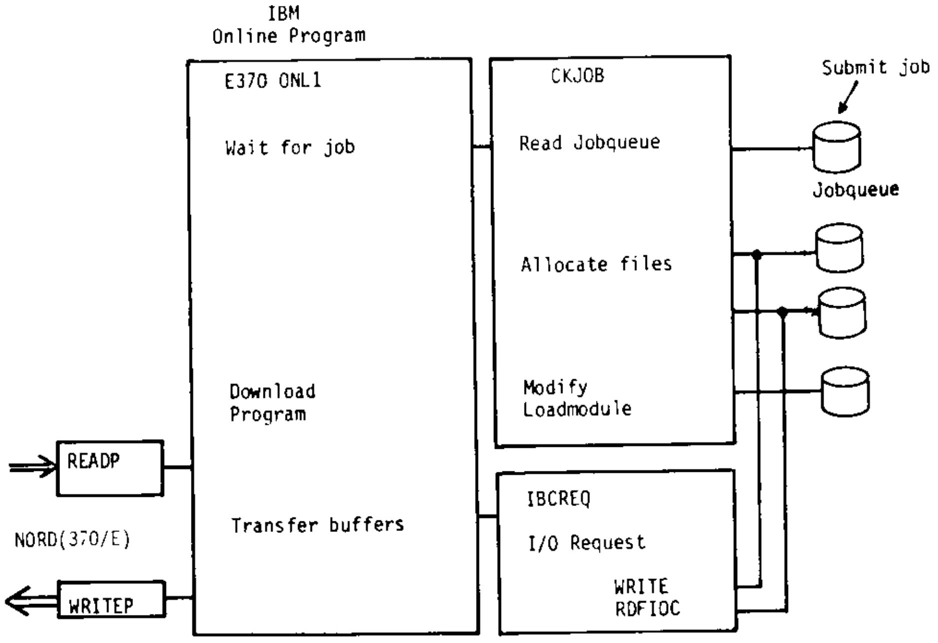


Fig. 9 IBM online program. Files are allocated, loadmodule downloaded and input/output performed.

IV. The Buffer Handling Routines in the 370/E

In Fig. 3 we have shown how input/output is organized in FORTRAN programs. In the following chapters the buffer routines are explained in more detail. The linkage between different routines is shown in Fig. 10.

The compiler generates several calls to IBCOM# depending on the READ/WRITE requests:

```
IBCOM# +  0:  Initialize READ with FORMAT
       +  4:      "      WRITE  "   "
       +  8:  Input/Output a variable
       + 12:  Input/Output an array
       + 16:  Finish current input/output operation
       + 20:  Initialize READ without FORMAT
       + 24:     "       WRITE   "       "
       + 28:  Input/Output a variable
       + 32:  Input/Output an array
       + 36:  Finish current input/output operation
       + 40:  Backspace
       + 44:  Rewind
       + 48:  End-of-file
       + 52:  STOP
       + 56:  PAUSE
       +
       + 64:  Initialize JOB
       + 68:  Terminate JOB
       +308:  Partial array handler
```

IBCOM# passes control to the system (GETMAIN, FREEMAIN, ABEND, DELETE, EXTRACT, LOAD, IHOSTAE, SPIE, STAE, WTOR) to interrupt the supervisor and to load service routines for an abnormal end. IBCOM# then calls FIOCS# for input/output buffer handling. For the 370/E FIOCS# is completely rewritten.

For each READ/WRITE IBCOM# passes to FIOCS# in Regester 2 a pointer to the data set reference number. In addition the following parameters are passed:

```
BALR   0,1      Jump to FIOCS#
DC     AL1(0)
DC     AL1(F0,FF,00,0F)
```

This information is then used to compute the address and length of the next buffer.

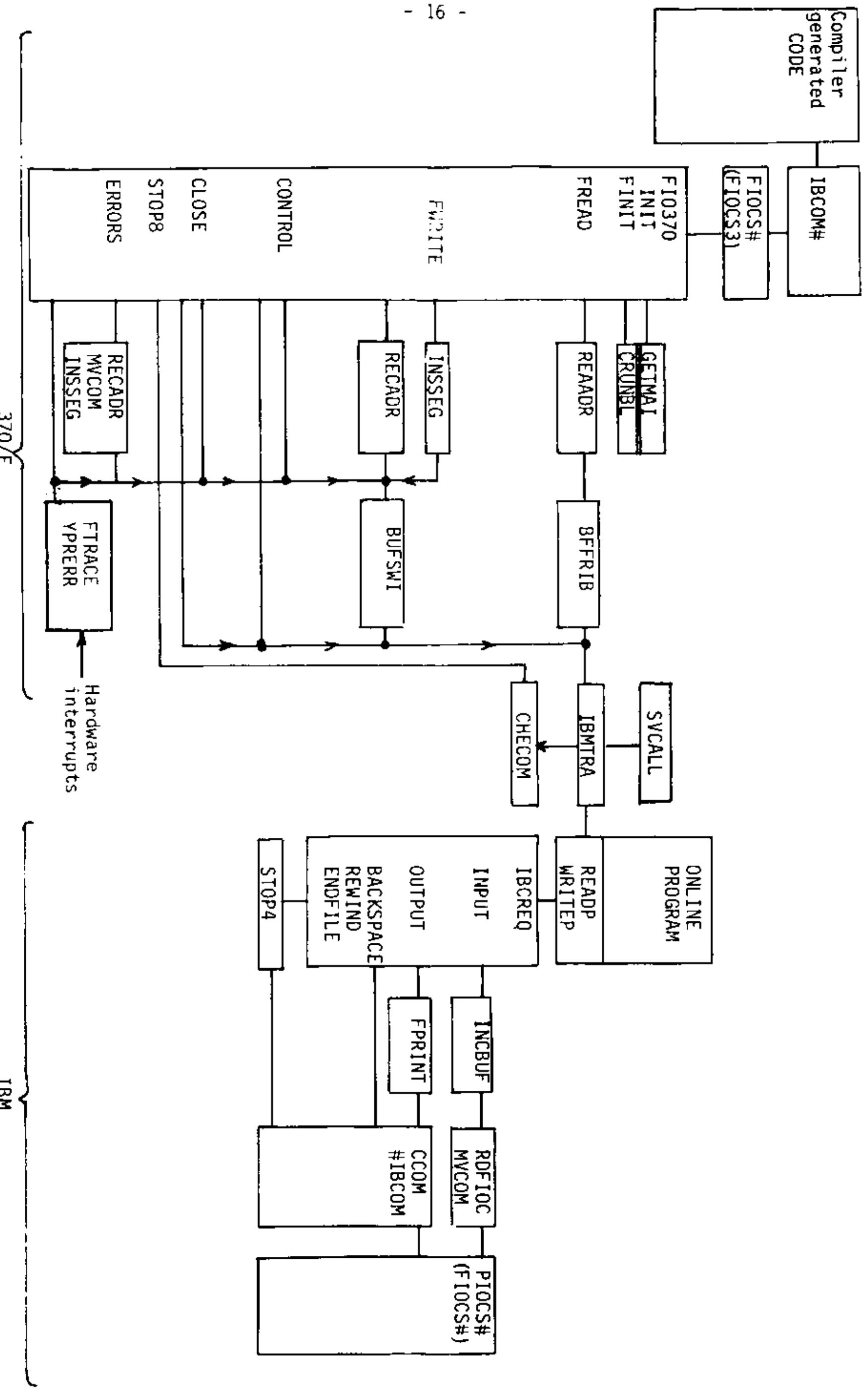


Fig. 10 Layout of programs for input/output. The 370/E- and the IBM-part can run as a single program at the IBM for testing.

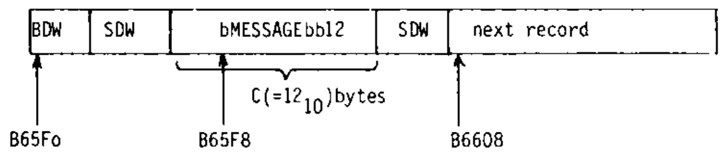
Example 1: Output with variable record format

```
     I = 12
     WRITE(6,2)I
   2 FORMAT(1X,7H MESSAGE, I4)
```

12 bytes = C_{16} bytes

Calls to FIOCS#	Register 2 before FIOCS#	Register 2 after FIOCS#	Register 3
Initialization 'FF' (Formatted Output)	Pointer to unit	0B65F8 (Address of Buffer)	85 (= 133 Bytes for one line)
WRITE	C (12 bytes are filled in buffer)	0B6608 (Address of next buffer)	85

BUFFER



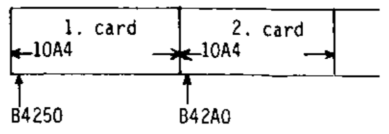
BDW = Block descriptor word
SDW = Segment descriptor word

Example 2 : Input with fixed record format

```
     DIMENSION CARD(20)
     READ(5,4) CARD
   4 FORMAT(10A4/10A4)
```

Calls to FIOCS#	Register 2 before FIOCS#	Register 2 after FIOCS#	Register 3
Initialization 'FO'	Pointer to Unit	B4250	50_{16} (= 80 bytes on card)
READ		B42A0	50

BUFFER for fixed records



For control information the parameters look as follows:

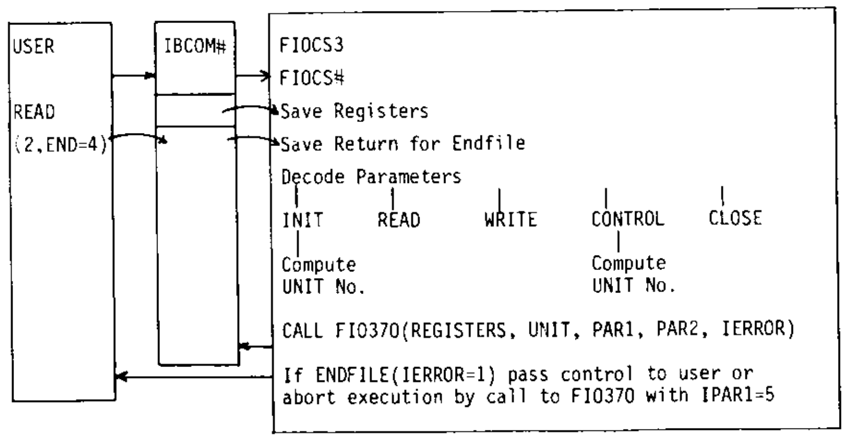
```
     BALR   0,1        Jump to FIOCS#
     DC     AL1(3)
     DC     AL1(0)     for BACKSPACE
               (1)     for REWIND
               (2)     for END-OF-FILE
```

To close all data sets we have at the end

```
     BALR   0,1        Jump to FIOCS#
     DC     AL1(4)
     DC     AL1(0)
```

IV.1 FIOCS3. The 370/E FIOCS# Interface Routine

FIOCS3 is an assembler routine which fulfills the linkage conventions of IBCOM#. It's entry points are FIOCS# and FIOCSBEP. All registers are saved in an internal save area SAVE1. The parameters are then decoded and control is passed to FIO370. FIO370 and all buffer handling routines are written in FORTRAN. The return addresses of IBCOM# for end-of-file condition (READ(1,END=4)B) are stored internally. If an endfile occurs the registers are restored and control is passed directly to the user's program. In case the user has not specified the END-parameter execution is terminated.



IV. 2 FIO370. The 370/E Buffer Handler

FIO370 is the steering routine for input/output buffer handling.

CALL FIO370(IREG, IUNIT, IPAR1, IPAR2, IRCODE)
DIMENSION IREG(11)

As we have seen above the main information between IBCOM# and FIOCS# is passed via registers. FIO370 has therefore access to registers 2 and 3:
IAD = IREG(2) = Register 2.

IUNIT is the current unit number

```
IPAR1 = 0  for initialization of an READ/WRITE operation
      = 1  for READ data
      = 2  for WRITE data
      = 3  for BACKSPACE, REWIND, EOF
      = 4  to  close all buffers
      = 5  for abnormal end.
```

IPAR2 = 'FO','FF','OO','OF' for formatted/unformatted input/output

```
IRCODE = return flag = 0  if there was no error
                       1  for end-of-file.
```

When FIO370 is called the first time it requests space in the buffer pool for the control words of the IBM answer. These words indicate on which unit the IBM has lastly processed and what error conditions occurred.

IPAR1 = 0

Initialize an input/output operation. If the unit was not used before a unit block (UB) and two buffers are created in the buffer pool COMMON/IHCBF2/. All buffers which are exchanged between the 370/E and the IBM are organized internally like records with variable format. If the actual organisation is fixed, unknown or formatted only complete records are placed into a block.
For input data sets REAADR is called to read the address and length for the next buffer. For the first READ of a unit two requests are sent to the IBM one after another to fill two blocks. After receiving of the first buffer control is passed to the user's program while the second buffer is filled at the IBM and transferred to the 370/E simultaneously. This method of double buffering minimizes dead time.
For output data sets RECADR is called to return the address and length of the next buffer. If a buffer is complete it is sent to the IBM while the second buffer is filled.

IPAR1 = 1

Entry for READ. Call REAADR for the next buffer address. For an end-of-file set the return code and pass control to the user's program.

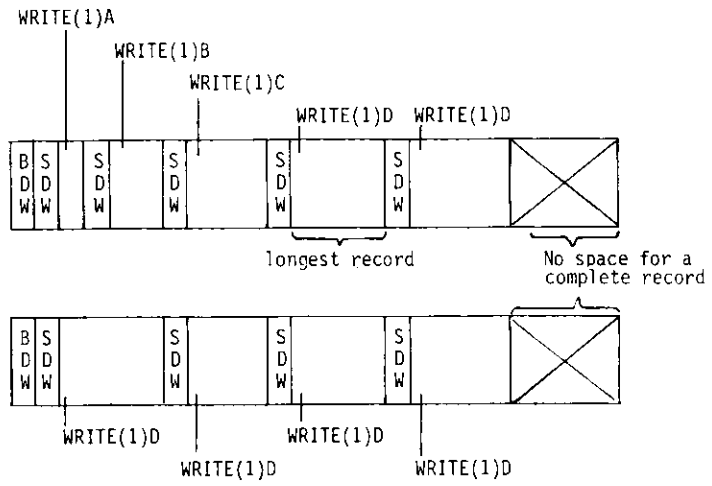
IPAR1 = 2

Entry for WRITE. Register 2 contains record length of the previous record. This record length is inserted in the segment descriptor word SDW of a data buffer.

RECADR is called to compute the address and max. length for the next buffer for WRITE. Spanned records with segments in different files are also marked by the SDW in the two righthanded bytes. This information is filled by IBCOM#. The record length LRECL of the unit assignment table is updated. LRECL contains the longest record length plus 4 for the SDW or BLOCKSIZE minus 4. This modification is necessary in order to send the segments of a logical record to the IBM without interleaving of segments of other units (see FPRINT). At the IBM incomplete logical records are collected in one local array and are written at once if the record is complete.

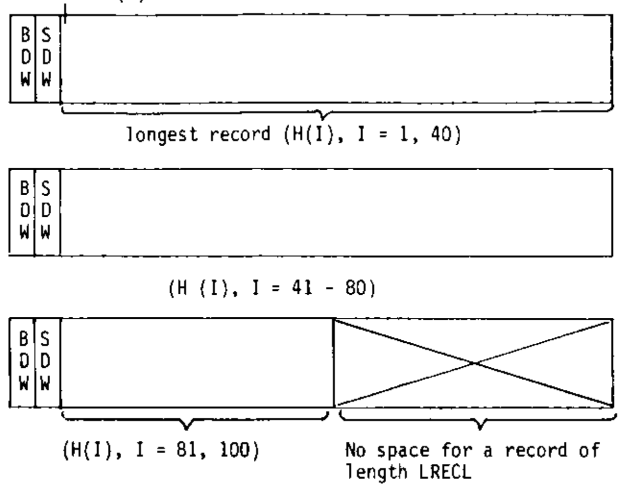
Example 3: Write short records

DIMENSION A(1), B(2), C(3), D(4)



Example 4: Write long records

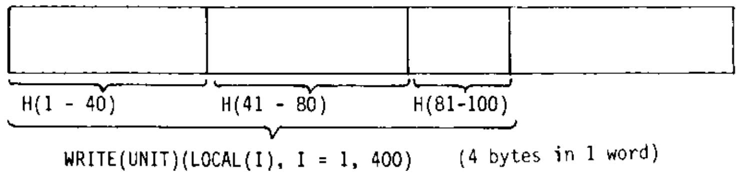
```
     DIMENSION H(100)
        WRITE(2)H
```



B D W
S D W
longest record (H(I), I = 1, 40)
(H (I), I = 41 - 80)
(H(I), I = 81, 100)
No space for a record of length LRECL

At the IBM spanned records are stored in array LOCAL and written after the last segment has reached the IBM

At IBM: LOGICAL * 1 LOCAL (32768)



IPAR1 = 3

Backspace, Rewind and end file are requested. If the unit is reserved for output the last buffer is sent to the IBM. Afterwards a control pattern is transferred to the IBM to do the BACKSPACE, REWIND or ENDFILE at the IBM.

For input data sets only a REWIND and ENDFILE control pattern is transferred to the IBM.
BACKSPACE for input files cause a lot of problems due to pipelining. Therefore BACKSPACE for input files is not allowed and an error message will be printed.

IPAR1 = 4

Close all buffers. The output buffers still containing some information are sent to the IBM. Afterwards a control pattern is passed to the IBM. This will cause a STOP 4 at the IBM after an answer has been sent to the 370/E. The 370/E also halts afterwards.

IPAR1 = 5

This entry point is used if an end-of-file occurs and no END parameter is specified in the READ statement. An error message is printed. Afterwards all output buffers will be sent to the IBM and execution terminates (ABEND).

Error Messages

FIO370 UNIT OUT OF RANGE if the unit number is less than 1 or greater than 99.
FIO370 SCC B37: No more buffer space available in COMMON/IHCBF2/.
FIO370 NO BACKSPACE: The user wants to backspace an input unit
FIO370 NO ENDFILE.STOP.UNIT xxx: No endfile exit specified in READ.

Service Routines for Buffer Handling and Input/Output

IV. 3. BFFRIB Buffer from IBM

This routine is called by REAADR and requests for input files the next buffer from the IBM. The control parameters to the IBM are:

```
     IPARTO(1) = 1 , IBCOM# request
     IPARTO(2) = 1 , READ
     IPARTO(3) = Unit number
     IPARTO(4) = ABYTE + BBYTE + CBYTE + DBYTE
                 ABYTE = ('FO','00')
     IPARTO(5) = BLKSIZE
     IPARTO(6) = RECFM + BUFNO + LRECL
```

Information to IBM
LISTTO(1,1) = address of the control parameters
Information from IBM
LISTFR(1,1) = address of the IBM answer
LISTFR(1,2) = address of the Buffer to which the IBM data are written

The routine does not wait for the IBM answer. Execution continues.

IV. 4. BUFSWI Switch Buffers and send them to IBM

This routine is called by RECADR if a buffer is full, by INSSEG for the last segment or by FI0370 to send the buffers to the IBM for closing or control (REWIND,BACKSPACE,ENDFILE).
The current output buffer is transferred to the IBM and the second output buffer is prepared for control parameters to IBM:

IPARTO(1)	= 1 , IBCOM# request
IPARTO(2)	= 2 , WRITE
IPARTO(3)	= Unit number
IPARTO(4)	= ABYTE + BBYTE + CBYTE + DBYTE ABYTE = ('FF','OF')
IPARTO(5)	= BLKSIZE
IPARTO(6)	= RECFM + BUFNO + LRECL

Information to IBM

LISTTO(1,1)	= address of the control parameters
LISTTO(1,2)	= address of the buffer to IBM

Information from IBM

LISTFR(1,1)	= address of IBM answer

The routine does not wait for the IBM answer. Execution continues.

IV. 5. CHECOM Wait for Buffer and Check the Answer

This routine is called by IBMTRA to finish the previous IBM transfer and by FI0370 if execution terminates.
We have seen in BFFRIB and BUFSWI that execution continues after an IBM transfer has been started. CHECOM generates a SVC2, waits until the transfer is finished and checks the answer for errors. In future a TIO will be generated.
The IBM waiting flag IBWAIT is zero if the data are already transferred and the answer was checked before. In this case CHECOM doesn't do anything.
When data are sent to the IBM the 370/E program tells the NORD on which locations the answer should be written. This information is used to transfer data back to the 370/E. The answer of the IBM also contains the unit number so that the error information can be placed into the unit block. In this way end-of-file conditions are detected by the corresponding READ statement (Fig. 11).

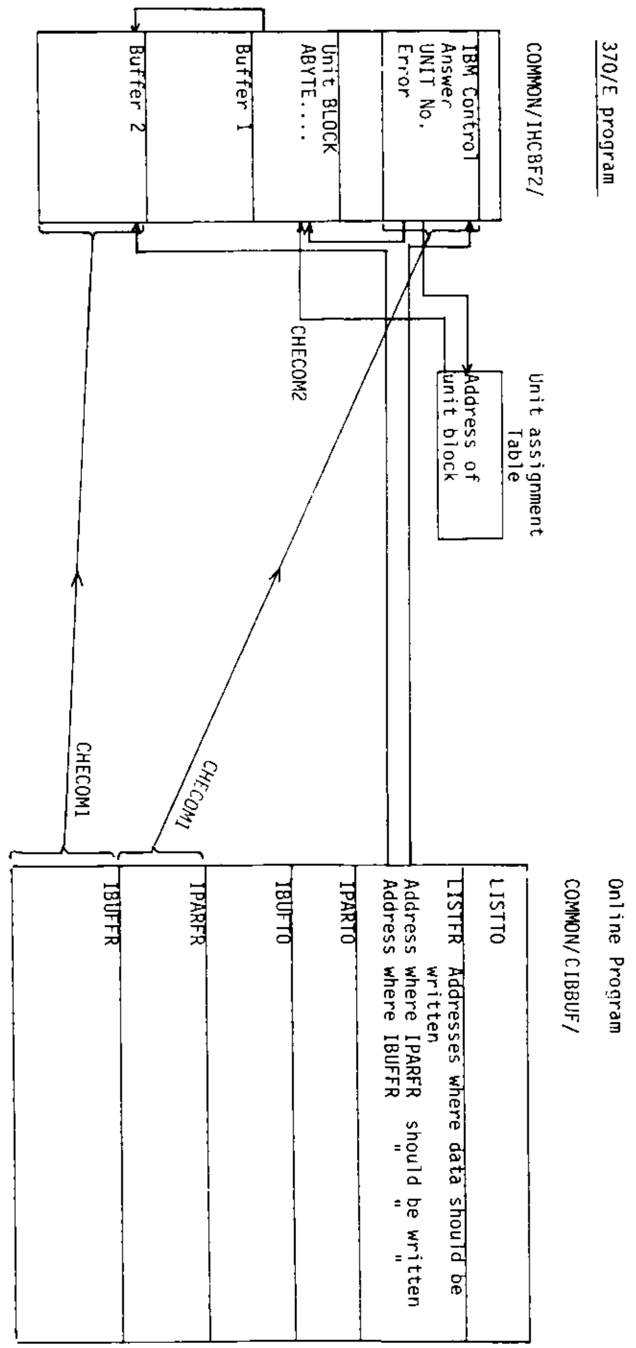


Fig. 11 Transfer of data by CHECOM between IBM online program and 370/E if both parts are running in one job.

IV. 6. CRUNBL Create Unit Block

This routine is called by FIO370 and creates a unit block. It reserves space in COMMON/IHCBF2/ for the unit block and space for two buffers of length BLKSIZE. The address of the unit block is inserted in the unit assignment table.

IV. 7. DCBSET Change data control block

See description of unit assignment table for further details.

```
DCBSET(IUNIT,IBLK,LRECL,IBUFNO,IRECFM)
IUNIT  = Unit number
IBLK   = Block size in bytes.
         For formatted or fixed records:  IBLK ≥ LRECLmax+4
         For all record formats        :  IBLK < 32767
                                              < IBM link limits
                                                (space in TMS9900)
                                              < Dual port memory size -30
LRECL  = Record length = 80  for cards. (For fixed records internally stored as
                       = 137 for line printer                     LRECL = 84)
                       = event length, ≤ IBLK-4
IBUFNO = 2
IRECFM = Record format. See unit assignment table
       = 90Z = 144 for FB  fixed blocked
       = 54Z =  84 for VBA variable blocked with ANS printer control
       = 58Z =  88 for VBS for event data
```

IV. 8. GETMAI Get space in main pool

GETMAI(LGBYTE,IADDR,IERR,IHIER) reserves space in COMMON/IHCBF2/.

```
LGBYTE  = No. of bytes being requested
IHIER   = 1  if buffer in dual port memory
        = 0  if buffer elsewhere
```

IADDR is the FORTRAN array address within IARBUF. The first word is not used because an address of 1 in the unit assignment table indicates a closed unit.

All requests are getting buffers on a double word boundary.

IV. 9. IADDR Compute Address

IADDR is a function and computes the address of a variable. It is part of the MVCOM control section.

```
Example:    DIMENSION A(200)
            starts at location B4F00₁₆

            IA = IADDR(A(2))
            IA = B4F04  is the address of A(2)
```

IV. 10. IBMTRA Transfer Data to the IBM

This routine prepares parameters for the interface and the control computer. The routine is called by BFFRIB,BFTOIB for buffers and by FIO370 for errors, closing and control requests. IBMTRA prepares a header block IBMHAD which is sent in front to the IBCOM# information.

```
CALL IBMTRA(LISTTO,LISTFR)
LISTTO(1,1) = Address of IPARTO
      (2,1) = Length of IPARTO = 6 words (32 bit)
      (1,2) = Address of buffer to IBM
      (2,2) = Length of buffer to IBM
      (1,3) = 0
      (2,3) = 0
LISTFR(1,1) = Address where IPARFR is written
      (2,1) = length of IPARFR = 10
      (1,2) = buffer from IBM
      (2,2) = length of buffer from IBM
      (1,3) = 0
      (2,3) = 0
```

The header information which is sent to the IBM contains:

```
IBMHAD(1) = No. of 32 bit words
      (2) = 821 identifier
      (3) = length of LISTTO = 4  (without the two zeros at the end)
      (4) = length of LISTFR = 4  (      "       "      "      )
       :  = LISTTO                (      "       "      "      )
            LISTFR                (      "       "      "      )
```

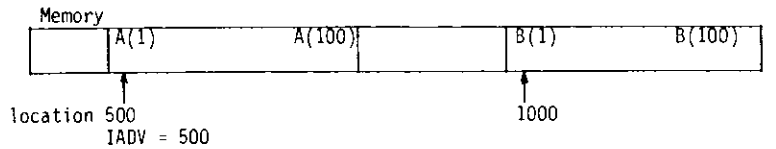
Before data are transferred to the IBM CHECOM is called to transfer the answer of the previous request from the IBM to the 370/E.

IV. 11. IDSTL Compute Distance between Variables

This function is used to transfer data from one array to another one. It is part of the control section MVCOM.

Example 5: Of the second array only the address is known.

```
      COMMON/C/IADV
      DIMENSION A(200)
C     Compute address of A
      IADV = IADDR(A(1))
       :
      SUBROUTINE TWO
      COMMON/C/IADV
      DIMENSION B(100)
C     Compute distance between A and B
      ID = IDSTL(B,IADV)/4
```



IADV = 500

ID = (500 - 1000)/4 = -500/4

```
C     B(1-125) points to A(1)
C     Copy data from B to A
      DO 2 I = 1,100
2     B(I+ID) = B(I)
```

IV. 12. INSSEG Insert Segment Word

INSSEG(IREG) inserts the segment descriptor word after a record is filled by IBCOM# and updates the record offset. The length of the segment is given by register 2 in IREG(2). If this segment is not the first segment of a logical record the buffer is sent to the IBM by BUFSWI to fulfill FPRINT conditions for multisegment records.

IV. 13. MVCOM Move Characters

This routine moves bytes from one array to another one.

```
     DIMENSION TO(100), FROM(100)
     CALL MVCOM(TO,FROM,400)
```

moves 400 bytes from array FROM to array TO. This routine is needed often because the source or destination address may not coincide with the full word boundary. If your output record is only 15 bytes long the following segment descriptor is not on a full word boundary. MVCOM is used to set the SDW in such a case.

IV. 14. REAADR Compute Address for READ

REAADR(IREG,IRCODE) is called by FIO370 and returns the address and the length of the next segment in registers 2 and 3. If a buffer is finished the third one is requested from the IBM while the second one was already filled and will be used now.

IV. 15. RECADR Compute Address for WRITE

RECADR(IREG,IERR) computes the address and the length for the next record. RECADR is called by FIO370. Full buffers are sent to the IBM by BUFSWI and the next buffer is filled.

IV. 16. FTRACE Interrupt Routine for Tracebeck

FTRACE is called for hardware interrupts by INTSRV or by FIO370 for input/output errors. It performs a traceback (divide check, overflow) and may abort the program if the error counter reaches its maximum. Via COMMON/CPLIST/ the program has access to the program status word PSW, the interrupt code and the registers. As FTRACE may be called also from FIO370 it may not do any WRITE or formatting. Input/output is therefore done by calling the buffer routines directly. After return the interrupt service routine gives a SVC6.

V. The Subroutines in the Program of the Control Computer

In the NORD we use features which are specific to NORD software and to the DESY installation. We therefore describe first the main routines and explain afterwards the special service routines. The layout of the NORD program is shown in Fig. 8. As the input/output to the IBM and to the 370/E is via DMA the program must be fixed in the NORD memory.

V. 1. JOBWT. Wait for JOB

JOBWT sends a control pattern IPARTO to the IBM program which passes control to CKJOB:

```
      IPARTO(1) = 2
      IPARTO(2) = 0 :  Wait for a job
      IPARTO(2) = 2 :  Allocate files at the IBM
      IPARTO(2) = 1 :  Load system load module
```

When the IBM has a job available for processing JOBWT asks for allocation of files. If all files are available the loadmodule is downloaded from the IBM and stored into the 370/E.
If a file allocation error occurs the next job is taken.

V. 2. LAIBM. Load the IBM module

LAIBM sends a message to the IBM·supervisor to load the correct program at the IBM.

```
       'CHANGE':  Unload the current program and call the supervisor
       'LOAD E370ONL1':  Load the online program for the 370/E
```

The loading of a program takes some time. Therefore several loops are needed to wait until the program at the IBM is ready for execution.

V. 3. MEMDMA. Transfer blocks to/from the 370/E Memory

The transfer of information between the NORD and the 370/E is controlled via MEMDMA. If less than 100 bytes are transferred we use programmed I/O. For larger blocks DMA is used. The routine takes the characteristics of the interface (like window register for LSI11) into account.

V. 4. SVCWT. Wait for SVC and Errors

Every 20 msec this routine checks the status of the processor. When the processor stops the SVC code and backplane registers are read out. In future this routine will measure the CPU time of the processor's program in order to interrupt processing in the case of dead loops.

V.5. INTHAD Interrupt Handling

If SVCWT detects an error (divide check, overflow, addressing) INTHAD is called. INTHAD has access to the backplane registers and the interrupt address of the 370/E. Using the contents of the backplane registers one can compute the type of interrupt. The address, the instruction length and the interrupt type are loaded into the 370/E processor and the 370/E is started at the start address of the interrupt service routine.
SVC6 indicates the return of the interrupt. Execution continues one instruction behind the interrupt address. If the interrupt routine is entered twice without the occurrence of a return (SVC6) of the previous interrupt the NORD aborts the job at the 370 and terminates execution.

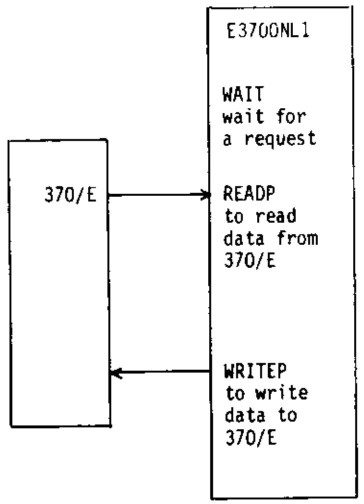
V.6. Service Routines in the NORD

CLOSE	Close a file
FIXC(ISEG,IPAGE)	Fix a contiguous segment starting on address IPAGE
HOLD(MS,1)	Wait MS·20 msec
HOLD(IS,2)	Wait IS sec
IBMON(1,LIST,$\genfrac{}{}{0pt}{}{5}{4}$,IER)	WRITE/READ to/from IBM.
	LIST contains addresses and length of buffers
IPEEK(IREG)	Read a register from the 370/E interface
IPOKE(IREG,IDAT)	Write into a register from the 370/E interface
MEMR/MEMW(IADR,DAT)	Read/Write to 370/E memory
OPEN	Open file to read unit number for terminal
POWER(IER)	Initialze 370/E
RESRV(IUNIT,$\genfrac{}{}{0pt}{}{0}{1}$,0)	Reserve a terminal for input/output
RELES(IUNIT,$\genfrac{}{}{0pt}{}{0}{1}$)	Release a terminal for input/output
SRUN(IADR)	Start 370/E at address IADR
UNFIX(ISEG)	Unfix a segment in the NORD
ZASEB	Convert ASCII/EBCDIC

VI. The Input/Output Routines in the IBM Online Program

The structure of the IBM Online Program is shown in Fig. 9. The program is started from the NORD via an interrupt to the IBM online supervisor. It then goes into a WAIT and waits for interrupts of the NORD.

The general structure of the online program can be recognized by this figure:



After the start of the program it waits for an interrupt of the 370/E in the routine WAIT.

CALL WAIT(PCD, &1, &2, &3, &4)

Relevant for us are returns 1, 2 and 3:

RETURN1:	The online program should finish by a STOP. If it does not finish it will be aborted. This exit is used to remove the job if there is no transfer between IBM and 370/E for half hour.
RETURN2:	A read request was given by the 370/E to transfer data to the IBM. (The IBM reads data from the 370/E). CALL READP afterwards.
RETURN3:	A write request was given. CALL WRITEP to transfer data to the 370/E.

Data are transferred from the 370/E to the IBM by

CALL READP(ARRAY, ICOUNT, IACT, &1, &2, &3, &4)

ARRAY	is the field where the data are written
ICOUNT	contains the maximum number of bytes to be transferred
IACT	contains the number of bytes which are actually transferred. IACT ≤ ICOUNT.IACT is always a multiple of 64 words of 16 bits.
RETURN1:	Time out
RETURN2:	No read request
RETURN3:	Transfer error
RETURN4:	Do a STOP

The answer of the IBM is sent to the 370/E by a

CALL WRITEP(ARRAY, ICOUNT)

Example 6:

```
    DIMENSION  TOIBM(100), FRIBM(50)
100 CALL WAIT(PCD, &500, &200, &300, &100)
200 CALL READP(TOIBM, 400, IACT, &100, &100, &100, &500)
    :
    Data from the 370/E are in TOIBM
    Put answer into FRIBM
    GOTO 100
300 CALL WRITEP(FRIBM, 200)
    GOTO 100
500 STOP
    END
```

VI. 1. ALCDYN. Allocate a data set

The two subroutines ALCDYN and DALDYN serve to allocate and unallocate data sets dynamically at execution time and may be called by FORTRAN programs.

CALL ALCDYN(DDN,DSN,UNIT,VOLUME,DISP,SPACE,DSORG,DCB,ERRORS)

1. DDN - character string of fixed length 8 - DDNAME
2. DSN - character string of fixed length 44 - Data set name
3. UNIT - character string of fixed length 8 - Unit name
4. VOLUME - character string of fixed length 6 - Volume name

5. DISP(3) - array of 3 character strings of fixed length 8 - disposition parameter for allocation.
   DISP(1): status specification: OLD | MOD | NEW | SHR
   DISP(2): normal disposition: KEEP | DELETE | CATLG | UNCATLG
   DISP(3): conditional disposition: KEEP | DELETE | CATLG | UNCATLG
6. SPACE(10) - array of 10 half words -
   SPACE(1): space unit specification:

= -1	space unit not defined (SPACE(2...4) will be ignored)
> 0	average block size
= -7360 or 'T'	tracks
= -15552 or 'C'	cylinders

   SPACE(2): primary allocation quantity
   SPACE(3): secondary allocation quantity
   SPACE(4): directory allocation quantity for PO data set
   SPACE(5): "release unused space" parameter (RLSE)

= 1	release unused space
= 0	do not release unused space

7. DSORG - half word -
   Data set organization specification for allocation:

   = - 1 not set
   = 512 or x'0200'PO (partitioned)
   = 16384 or x'4000'PS(physical sequential)
   Actual DSORG is always returned in this field.
8. DCB(22) - array of 22 half words -
   DCB parameters for allocation:

   DCB(1): to specify the record format give the sum of all relevant numbers from the following list:

8	standard fixed or spanned variable (S)
16	blocked (B)
64	variable (V)
128	fixed (F)
192	undefined (U)

   Examples: to specify RECFM=VB set DCB(1)=80, to specify RECFM=VBS set DCB(1)=88.

   DCB(2): BLKSIZE value
   DCB(3): LRECL value
   DCB(4): BUFNO value

VI. 2. CALLOC Interface for Allocate dataset

The routines to allocate a dataset needs many parameters. CALLOC gets the relevant information for a dataset from the job queue, unpacks the information and allocates a file. If the unit is reserved the old file is deallocated and a second allocate is carried out. Some information about file allocation is printed to the printing unit

```
        INTEGER*2 NER(3)
        LOGICAL*1 LOGIN(78)

        CALL CALLOC(LOGIN,NER,IUNIT)

INPUT: LOGIN = DESCRIPTOR OF UNIT. LOGICAL * 1. PACKED FORMAT
       LOGIN(1) = UNIT NUMBER
            (2) = SEQUENCE NUMBER (2 = FT01F002)
            (3) = DISP = 10 = OLD       )
                         20 = MOD       )
                         30 = NEW       )FIRST PARAMETER
                         40 = SHR       )
                       + 1 = KEEP        )
                       + 2 = DELETE      )
                       + 3 = CATLG       ) SECOND PARAMETER
                       + 4 = UNCATLG     )
             (4) = X'40' PHYSICAL SEQUENTIAL
                   X'02' PARTITIONED
           (5+6) = SPACE =-1,QUANTITY,'T'=-7360,'C'=-15552
           (7+8) = PRIMARY ALLOCATION
          (9+10) = SECUNDARY ALLOCATION
         (11+12) = DIRECTORY
         (13+14) = 0 OR 1(RLSE)
         (15-20) = VOLUME
         (21-28) = UNIT ='FAST' OR '        '
         (29-72) = DSNAME
         (73-74) = DCB BLKSIZE
         (75-76) = DBC LRECL
            (77) = RECFM = 8 (STANDARD OR SPANNED)+16(B)
                          + 64(V) + 128 (F) + 192 (U)
                 = 0, IF DCB FROM DATA SET
            (78) = --
        IUNIT    = PRINTING UNIT. IF ZERO NO PRINT OUT
```

VI. 3. CKJOB Wait for a Job and Check

CKJOB is called from the NORD. It checks the job queue if there is a job waiting for processing. The disc for the job queue is allocated and searched for a job. The job control information is then read and files will be allocated. Then the load module is read. The control information of the load module is used to compute the size of the program and to get the addresses where the segments of the program should be loaded into the 370/E. According to the allocated files the unit table is updated to inform the 370/E about the available files.

```
        INTEGER*4 IPARTO(6),IPARFR(10),IBUFFR(5000)
        CALL CKJOB(IPARTO,IPARFR,IBUFFR)
INPUT:  IPARTO(1) = 2  LOOK FOR JOBS AND ALLOCATE FILES
        IPARTO(2) = 0. WAIT FOR A JOB
                    1. LOAD SYSTEM LOAD MODULE TO THE 370/E
                    2. ALLOCATE FILES AND SEND RESULT TO 370/E
        IPARTO(3) = NO. OF FILE TO BE ALLOCATED
                  = 0 FOR PRESENT JOB. 1 FOR PREVIOUS JOB
OUTPUT:  IPARFR(1) = 2. ANSWER OF CKJOB
         IPARFR(2) = 0. THERE IS NO JOB. TRY LATER
                   = 1. THERE IS A JOB WAITING
                   = 2. SYSTEM LOAD MODULE IS SENT
                   = 3. END OF SYSTEM LOAD MODULE
                   = 4. FILE ALLOCATED
         IPARFR(3) = 0. NO ERROR OF ALLOCATING FILES
                   = IERALC(1) OF ALLOCATING ROUTINE
         IPARFR(4) = IERALC(2) OF ALLOCATING ROUTINE
         IPARFR(5) = LENGTH OF TEXT FILE (IN 32 BIT WORDS)
                   = JOB NUMBER
                   = FILE NUMBER FOR ALLOCATION
         IPARFR(6) = ADDRESS IN 370/E FOR TEXT FILE
                   = JOB TIME
                   = FILE ALLOCATION OK.(0). END OF ALLOCATION (1)
         IPARFR(7) = LENGTH OF TEXT FILE

         IBUFFR    = TEXT FILE FOR LOAD MODLE. IBUFFR(5000)
                   = NAME OF JOB
                   = NAME OF FILE

CALLING SEQUENCE: WAIT FOR JOB
                  ALLOCATE FILES
                  READ LOAD MODULE
```

VI. 4. CLJOB Close job and deallocate files

At the end of a user's program all files are closed and deallocated. Than the online program will be unloaded to release all buffers (STOP4)

VI. 5. CONDYN Connect files

If one wants to read two files like a single file one can connect these files like:

```
    //GO.FT01F001  DD  DSN =
    //             DD  DSN =
```

CONDYN can concatenate up to 20 files.

CALL   CALL CONDYN(NDDN,DDNS,IPERM,ERRORS)

1. NDDN - half word -
   number of files to be concatenated ($2 \leq$ NDDN ≤ 20)
2. DDNS - character string of fixed length 8*NDDN -
   array of ddnames of files to be concatenated; each ddname is to be 8 bytes long (eventually padded right with blanks).
3. IPERM - half word -
   requested type of concatenation:
   = 0 : non-permanent
   = 1 : permanent

NOTE: Permanently concatenated files cannot be de-concatenated dynamically; only dynamic un-allocation will be possible.

VI. 6. DALDYN Deallocate files

Unallocation of data sets by ddname at execution time. Dataset disposition at allocation time is executed at unallocation time and may be altered using subroutine DALDYN:

CALL   CALL DALDYN(DDN,OVDISP,ERRORS)

1. DDN - character string of fixed length 8 -
   ddname of file to be unallocated.
2. OVDISP - character string of fixed length 8 -
   overriding disposition for data set to be unallocated:
   OVDISP = KEEP | DELETE | CATLG ¦ UNCATLG

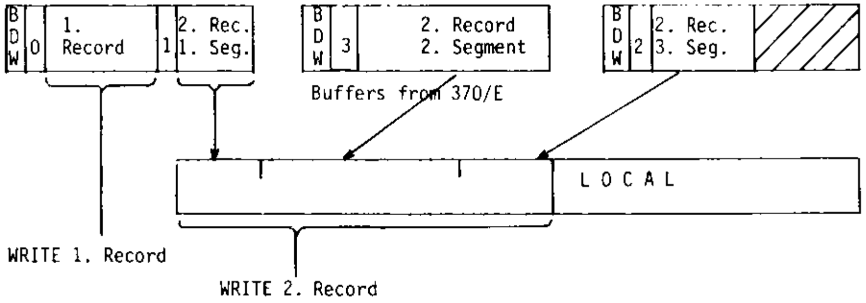
VI. 7. FPRINT Write Records at the IBM

FPRINT gets a complete block from the 370/E, searches for logical records and outputs them.

FPRINT(IUNIT,LBUF,IFORM)
UNIT = unit number
LBUF = Buffer from 370/E
IFORM = 0, write without format control
= 1, write with format control.

Stand alone records are written directly, spanned records are copied to a local array. A write is then given after the record is complete. Therefore spanned records must be sent to the IBM without interference of other units.

Example 7:



VI. 8. IBCREQ Handle IBCOM# Requests

IBCREQ(IPARTO,IBUFTO,IPARFR,IBUFFR)
All IBCOM# requests INPUT,OUTPUT,BACKSPACE,REWIND,ENDFILE are handled by this routine. The routine is called by the IBM online program E3700NL1.
IPARTO contains the parameters sent to the IBM,
IBUFTO is the buffer to the IBM for output
IPARFR contains the answer from the IBM and control information
IBUFFR is the buffer from the IBM to the 370/E for input.

```
    IPARTO(1) = 1  IBCOM# request
          (2) = 1  READ
              = 2  WRITE
              = 3  CONTROL
          (3) = unit number
          (4) = ABYTE         or = 1,  BACKSPACE
                                 = 2,  REWIND
                                 = 3,  ENDFILE
          (5) = BLKSIZE
          (6) = LRECL
    IPARFR(4) = 0, no error
              = 1, end-of-file.
```

VI. 9. INCBUF Fill Input Buffer

In order to avoid high dead time the IBM fills input buffers already before the user at the 370/E gives an actual READ command. For the very first READ statement the user program has to wait until the IBM has filled a buffer. But while execution continues at the 370/E the IBM fills the next buffer. Therefore the IBM program does not know, how many words the user at the 370/E would like to read and whether the format is fixed for cards or variable for events. The usual FORTRAN input techniques do not help us in this case. We therefore wrote extra input routines using FIOCS# at the IBM.

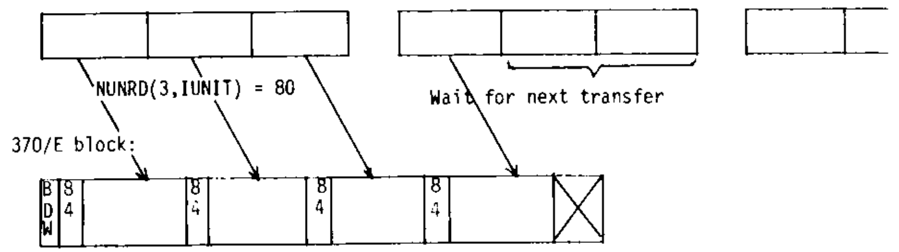
INCBUF(IUNIT,IFORM,IPARFR,LBUFFR)
is called by IBCREQ and fills a complete input buffer for IUNIT.

IUNIT	= Unit number
IFORM	= 0 for unformatted input = 1 for formatted input fixed records IFORM is taken from ABYTE which is sent in IPARTO
IPARFR	= Parameters from IBM
IPARFR(4)	= 1 for end-of-file
LBUFFR	= Buffer which is sent to the 370/E The first word in the buffer contains the block descriptor word and is used to determine the block size. This word is modified for end-of-file conditions in the last record.

The block which is transferred to the 370/E is organized as a variable record for all data set formats: Information like record length or format is taken from the unit assignment table for input datasets. If the input device has a fixed record format complete records are placed into a block.

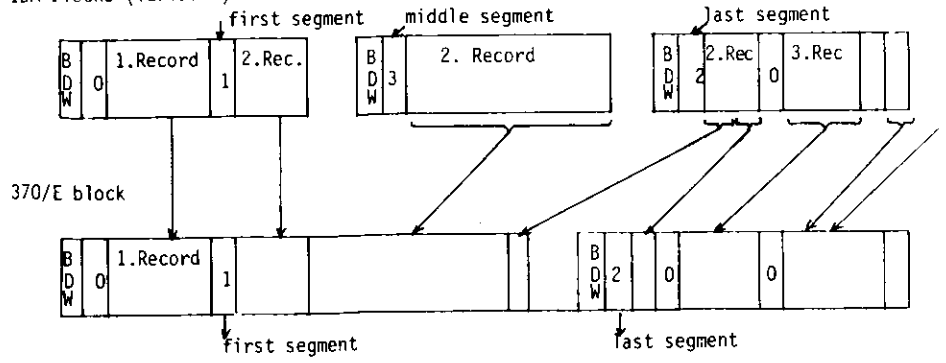
Example 8:

IBM blocks (fixed):



Variable records are copied to the 370/E buffer. Care must be taken for spanned records and the segment descriptors for the 370/E must be marked correctly.

IBM blocks (variable):



VI. 10. RDFIOC READ by FIOCS#

For input datasets we cannot use FORTRAN READs because the IBM does not know in advance how many words should be read. Read is therefore performed by FIOCS#.

RDFIOC(IUNIT,IFORM,NANS) requests the address and the length for the next segment of IUNIT. FIOCS# returns in registers 2 and 3 the address and the length. This information is stored into array NANS:

INTEGER*2 NANS

$\left.\begin{array}{l}\text{NANS(1) =}\\ \text{NANS(2) =}\end{array}\right\}$ Address of next segment

NANS(3) = length of segment

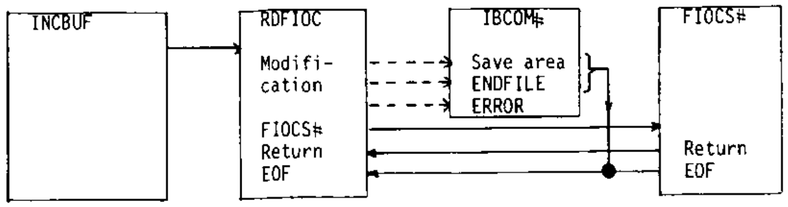
NANS(4) = 0. No error. Used for segment offset.
= -1. End-of-file

NANS(5) = Code from segment descriptor word = 0, 1, 2 or 3 for complete, first, last or middle segment

NANS(6) = Record format. Taken from the data control block of the unit block.

RDFIOC also modifies the ENDFILE and ERROR words and register save area in IBCOM# for proper return in case of end-of-file conditions. IHOUATBL contains the IBM unit assignment table and is needed to compute the address of the unit block in order to get the dataset format.

Endfile condition:



If the 370/E is connected to a non IBM computer RDFIOC must be modified. In this case RDFIOC must know the format of data for several units:

Example:

```
        SUBROUTINE RDFIOC(IUNIT,IFORM,NANS)
        INTEGER*2 NANS(6),NZ(2)
        COMMON/CRC/LENG,EVENT(1000)
        DIMENSION CARD(20)
        EQUIVALENCE (CARD(1),LENG)
        EQUIVALENCE (NZ(1),IZ)
    C   Address of "buffer"
        IZ = IADDR(CARD)
        NANS(1) = NZ(1)
        NANS(2) = NZ(2)
    C   No error
        NANS(4) = 0
    C   Complete record
        NANS(5) = 0
        IF(IUNIT.NE.5) GOTO 2
    C   CARDS.
        80 bytes per card
        NANS(3) = 80
    C   RECFM
        NANS(6) = 144
        READ(5,4,END = 8)CARD
    4   FORMAT(20A4)
        GOTO 99
    8   NANS(4) = -1
        GOTO 99
    2   CONTINUE
    C   Next unit
          :
```

VII. Buffer organization and Tables

VII. 1. Unit assignment table COMMON/IHOUAC/

The unit assignment table is organized like IHOUATBL. It has entries for 100 units. Of these only 99 units are allowed while unit number 100 is used to point to the area of the IBM answer.

The unit assignment table looks as follows:

```
COMMON/IHOUAC/NUPRES,NUMHUN,ISTDUN,IUNAS(4,100)
INTEGER*2 NUPRES,NUMHUN
NUPRES  =  Present unit number
NUMHUN  =  Maximum number of units * 16
ISTDUN  =  Standard units for ERROR,READ,PRINT,PUNCH
        =  Z06050607
IUNAS(1,..)  =  Address of unit block in COMMON/IHCBF2/.
                The address is the FORTRAN array address in IARBUF.
                IUNAS(1,...) = 2 if the unit block starts at IARBUF(2).
                IUNAS(1,...) = 1 if no unit block was created.
```

IUNAS(2,...) =

not used	IHOASYNC	BLKSIZE	

IUNAS(3,...) =

IRECFM	Buf no = 2	LRECLI	

The blocksize may be chosen independently of the blocksize of the data definition card (// DD). It depends on the memory space, the characteristics of the IBM link and the IBM online program. If it is too small one has many transfers between the 370/E and the IBM. For a 370/E with dual port memory for input/output the blocksize should not exceed the size of that memory minus 30.

IRECFM defines the record format

```
IRECFM = Z04 =   4  = ANS printer control
       = Z08 =   8  = spanned
       = Z10 =  16  = blocked
       = Z40 =  64  = variable
       = Z80 = 128  = fixed
       = ZC0 = 192  = undefined
```

Typical formats are 88 for VBS, 128 for F, 196 for UA or 84 for VBA. The buffers between 370/E and IBM are all organized as VB. For fixed, unknown or ANS records only complete records are placed into a block while spanned records may be used for variable records.

LRECLI is the record length. For fixed records it should have the size as specified on the IBM data definition card +4 (i.e. LRECLI = 84 = Z54 for cards).

LRECL should have the length of the longest record (i.e. LRECL = 137 for line printer) but should be less or equal to BLOCKSIZE-4. During execution LRECL is updated to the longest record.
Data control block definitions can be changed by a
CALL DCBSET(IUNIT,IBLK,LRECL,IBUFNO,IRECFM)
For fixed records LRECLI = LRECL + 4.

VII.2. The buffer pool area for unit blocks and buffers

All buffers and unit blocks are stored in

COMMON/IHCBUF/LGBYTE,LBUSED(3) /IHCBF2/IARBUF(10000)
Common for unit blocks and buffer space

LGBYTE	= length of IARBUF in bytes = 40000
LBUSED(1)	= No. of words used
IARBUF	= unit blocks, buffer space.

Space is allocated in IARBUF by a

```
     CALL GETMAI(LBYTE,IADDR,IERR,IHIER)
```

IHIER defines whether the buffers should reside in the dual port memory or in the normal memory. At present all buffers are located in one contiguous array and only LBUSED(1) is used.

VII.2.1 The control block of the IBM answer

The control block of the IBM is 10 words long and contains:

```
        IPARFR(1) = 1
        IPARFR(2) = 1, 2, 3 for READ, WRITE, CONTROL
        IPARFR(3) = IUNIT which was processed at the IBM
        IPARFR(4) = 0  no error
                    1  end-of-file
                    2  I/O error
                    3  FIOCS#(IBM) error.
```

Unit no. 100 of the unit assignment table points to this area.

VII. 2.2 The unit block (UB)

The unit block describes the position and status of the input/output buffers. The address in the unit assignment table points to the unit block. If the address is 1 the unit was not opened before and a unit block must be created.

A unit block has the following form

```
             IADDU = IUNAS(1,IUNIT)
             IARBUF(IADDU + 0) = ABYTE + BBYTE + CBYTE + DBYTE
```

IADDU + 0	ABYTE	BBYTE	CBYTE	DBYTE
+ 1	Address of buffer 1			
+ 2	Address of buffer 2			
+ 3	Address of current buffer			
+ 4	Record offset within buffer			
+ 5	IDECB = 0, 1, 2			
+ 6	No. of current buffer = 1, 2			
+ 7	0 Address of 1. Buffer to IBM			
+ 8	0 Address of 2. Buffer to IBM			

```
ABYTE = Parameter 2 of FIO370
      = 'FO','FF','OO','OF' for formatted/unformatted input/output
BBYTE = 1 for end-of-file
CBYTE = 1 for opened data set
DBYTE = 0
```

For a given unit two buffers for data are created. Within a buffer the offset pointer points to the last byte which is used. It is set to 4 for an empty buffer because 4 bytes are needed for the blockdescriptor.

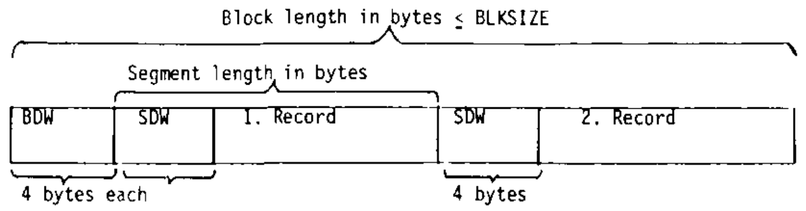
```
IDECB = 0, no buffer sent to IBM
        1, buffer 1 sent to IBM
        2, buffer 2 sent to IBM
```

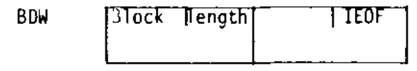
A unit block is created by

```
     CALL CRUNBL(ABYTE,IERROR)
```

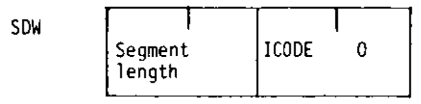
VII. 2.3 The unit buffers

For each unit two buffers are reserved in the buffer pool. Each buffer is organized as variable blocked records with block and segment descriptor words (BDW,SDW)





IEOF = 1 if this is the last block before an end-of-file occurred (Non IBM standard)



```
ICODE = 0, stand alone record
      = 1, first of a multisegment record
      = 2, last of a multisegment record
      = 3, neither first nor last segment.
```

The layout of buffers and pointers is shown in Fig. 12.

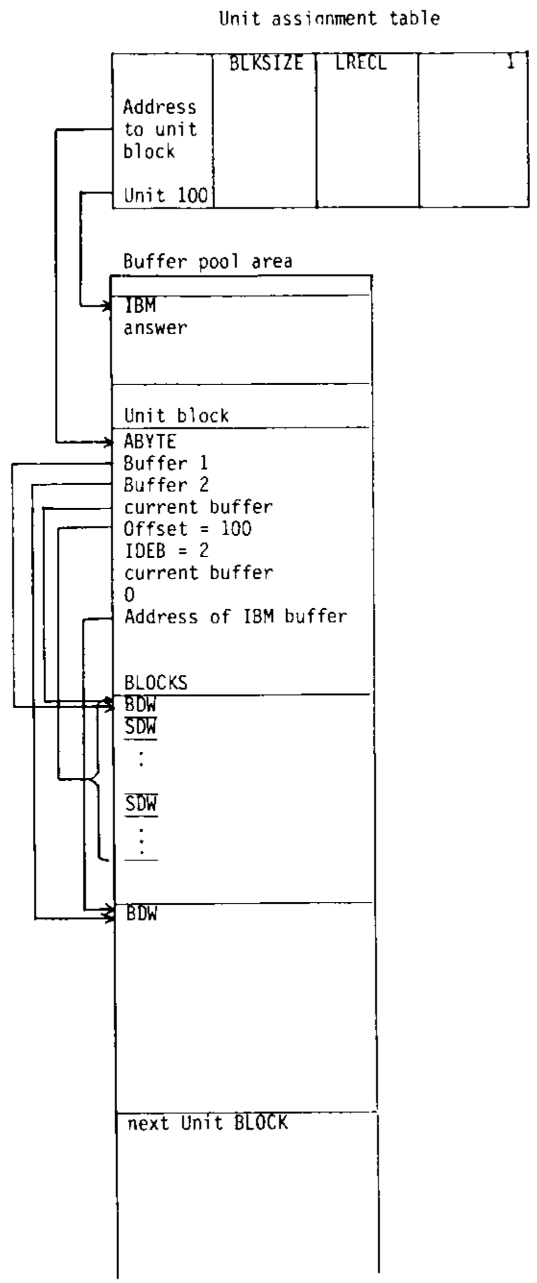


Fig. 12 Buffers and Pointers

VII.3. Unit Assignment Table for Input at the IBM

For input units a special table at the IBM is needed because input is not done via IBCOM# but via FIOCS#.

```
COMMON/IHCUNR/NUNRD(6,99)
INTEGER*2 NUNRD
```

NUNRD(1+2 ,IUNIT)	= Address of current segment
	= Register 2 from FIOCS#
NUNRD(3,IUNIT)	= length of current segment or record
NUNRD(4,IUNIT)	= 0. A read was successfull
	= -1, end-of-file
	< -1, FIOCS# error
	> 0. Segment offset pointer. The following words are not yet copied to the 370/E buffer
NUNRD(5,IUNIT)	= Segment descriptor word
	= 0, complete segment = 1, first segment = 2, last segment = 3, middle segment
NUNRD(6,IUNIT)	= Record format. Taken from IBM DCB in unit block.

VII.4. The COMMON/CIBUF/ of the Online Program

```
COMMON/CIBUF/IBMBUF(5000)
```

IBMBUF(1)	=	Number of 32 bit words including this word	
(2)	=	821 = code	
(3)	=	NUMTO = No. of different blocks sent to IBM	
(4)	=	NUMFR = " " " sent from IBM to 370/E	
(5)	=	Address of PARTO (Address in 370/E memory)	
(6)	=	length of PARTO	
(7)	=	Address of BUFTO (Address in 370/E memory)	
(8)	=	length of BUFTO	
(9)	=	Address of PARFR (Address in 370/E where data are written)	
(10)	=	length of PARFR	
(11)	=	PARTO(1)	length of PARTO = IBMBUF(6)
⋮			
(16)	=	PARTO(6)	
(17)	=	BUFTO(1)	length of BUFTO = IBMBUF(8)
⋮			
(116)	=	BUFTO(100)	
(117)	=	PARFR(1)	length of PARFR = IBMBUF(10)
⋮			
(126)	=	PARFR(10)	

VII.5. Job Control Information

The job control information i.e.

```
     //F1BNOTOO JOB TIME=1
     //STEPOO EXEC PGM=F1BNOT.TSOLIBL(J370)
     //LISTFILE DD DSN=F1BNOT.LIST3
     //FT03F001 DD DSN=F1BNOT.INP
```

is stored into

```
     INTEGER*4 NJOB(4)
     LOGICAL*1 LBUFFR(78,11)
```

NJOB(1)-(3)	=	'F1BNOT' = Job name
NJOB(4)	=	Time in minutes
LBUFFR(1,1)	=	File information for load module
LBUFFR(1,2)	=	" " " List file
LBUFFR(1,3)	=	" " " file 1
LBUFFR(1,4)	=	" " " file 2

For each file:

```
LOGIN(1) = UNIT NUMPER
     (2) = SEQUENCE NUMBEP (2 = FT01F002)
     (3) = DISP = 10 = OLD      )
                  20 = MOD      )
                  30 = NEW      )FIRST PARAMETEF
                  40 = SHR      )
                + 1 = KEEP        )
                + 2 = DELETE      )
                + 3 = CATLG       ) SECOND PARAMETEP
                + 4 = UNCATLG     )
      (4) = X'40' PHYSICAL SEQUENTIAL
            X'02' PARTITIONED
    (5+6) = SPACE =-1,CUANTITY,'T'=-7360,'C'=-15552
    (7+8) = PRIMARY ALLOCATION
   (9+10) = SECUNDARY ALLOCATION
  (11+12) = DIRECTORY
  (13+14) = 0 OR 1(RLSE)
  (15-20) = VOLUME
  (21-28) = UNIT ='FAST' OR '           '
  (29-72) = DSNAME
  (73-74) = DCB BLKSIZE
  (75-76) = DBC LRECL
     (77) = RECFM = 8 (STANDARD OR SPANNED)+16(B)
                  + 64(V) + 128 (F) + 192 (U)
          = 0, IF DCB FROM DATA SET
     (78) = --
```

VII.6. File for Job Queue

The file for the job queue is a direct organized data set (Fig. 13). The first record contains a counter indicating how many jobs are processed. This record is only modified by the online routine CKJOB. The second record indicates how many jobs are requested. All remaining records contain the job control information for each job. If the end of the disc is reached the first job files will be overwritten if they are already processed. Otherwise the queue is full.

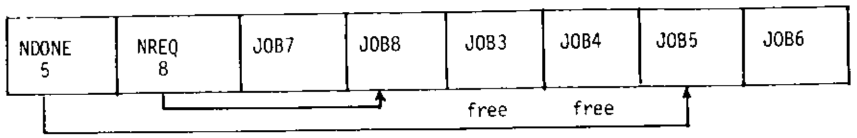


Fig. 13 Organization of job queue

Appendix

A 1. A Program to Test the 370/E

In order to test the processor a test program was written which generates per transfer 500 random numbers, ALOG, SQRT, ARSIN, SIN, COS, TAN, EXP and fills an array A(10,500). 20 000 bytes are then transferred from the IBM to the TMS 9900, the control processor of the online net (15 μsec/16 bit word). At the end of this transfer data are sent to the NORD (10 μsec/16 bit word). When data are transferred in the opposite direction from the NORD via the TMS9900 to the IBM the two transfers overlap for higher speed.
The NORD transfers data to the 370/E and starts it. In order to minimize time the 370/E requests from the IBM the next block with results by transferring 10 control words to the IBM (Fig. A1).
The 370/E is then started, the ten words are sent to the IBM and the IBM is activated. The NORD then waits for the end of the IBM transfer and the end of the 370/E computation.

A 2. A Program to Test Error Handling

In order to test the error handling of the 370/E we wrote a small program which does FORTRAN I/O and contains some errors. The program source code is shown in Fig. A2. It is compiled and linked at the IBM. The listing of the linkage editor is given in Fig. A3. All buffer handling routines and the system of the 370/E is loaded at the beginning. The job is then submitted and executed at the 370/E. Output is transferred to the IBM and printed on a LISTFILE (Fig. A4). Software errors like negative square roots are handled by the FORTRAN library and hardware interrupts like divide checks are controlled by an interrupt service routine. Execution might terminate for address errors or continue for divide checks.

A 3. Some Remarks

1) The 370/E can be connected to each computer. Input/output to tapes and line printers can be controlled by this computer as long as the buffer formats are treated correctly. Error messages are printed in the known IBM format. In order to develop and load programs a link to an IBM is needed.

2) There are some restrictions for programmers using assembler language. Due to pipelining it is not allowed to modify the following instruction and to execute it afterwards.

3) It might be possible to support also direct access I/O.

4) There is no access to system control blocks. It is complicated to transfer them from the IBM as the addresses of a job in the IBM and in the 370/E are different.

5) To connect the 370/E to IBM we use at DESY a 2701 unit with parallel data adapter. One might think of available interfaces like MODCOM, S/1 or NOVA to connect the 370/E to the IBM channel.

6) When one compares results of the 370/E and the IBM one must use the same COMPILERS. The same FORTRAN library but a different compiler can give different results:

$$R = \text{Z40170E9A}$$

$$\begin{aligned}\underbrace{\text{COS(R*2*3.141592)}}_{\text{This code is different in both compilers}} &= \text{Z40D81733 for IEKAA0}\\ &= \text{Z40D81731 for FORTRANQ}\end{aligned}$$

7) The processor was tested with the program described in A1. With the bit slice 2901A we found two wrongly calculated exponent functions per 4 Mill. numbers, with the 2901B no error was found for 40 Mill. numbers.

8) The new interface has the following features: If a SVC, a program interrupt or an external interrupt occurs the old program status word PSW is saved on locations 32, 40 or 24. The program branches to the address given by the new PSW defined in locations 96 , 104 or 88.
Input/output is performed in the following way: Like LISTTO and LISTFR a set of channel command words CCW describe the I/O commands, the addresses and lengths of buffers. The channel address word for channel zero in permanent storage location 72 points to the first CCW. The processor gives a start I/O operation and stops. The control processor has to read the channel device address (in ARH + ARL), sets the single cycle bit and clocks for CAW (in ARH + ARL), 1. CCW and 2. CCW (in DBH + DBL). If the channel is not available the control computer returns the channel status word CSW (in DBH and DBL) and sets the condition code. The processor is started again. The end of data transfer is signaled by an interrupt or tested by the test I/O command TIO.

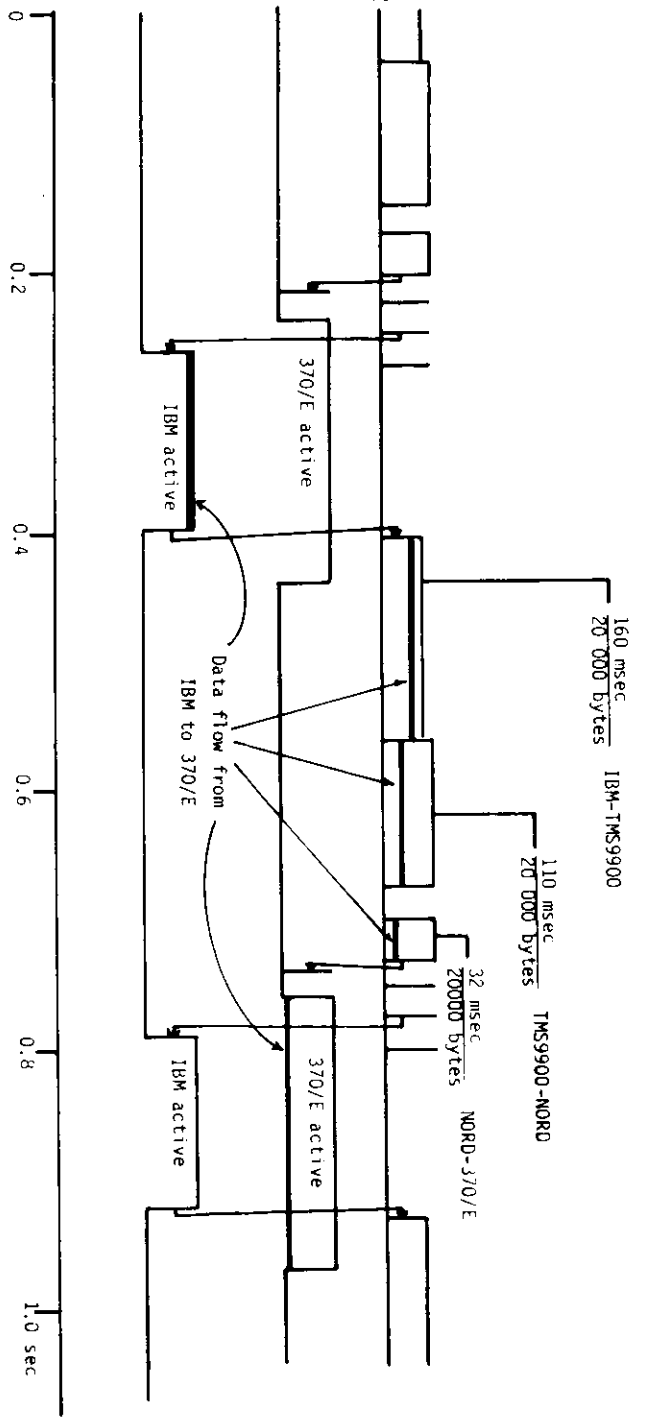


Fig.A1 Timing diagram for a test program

```
               C    02/07/82 207020626  MEMBER NAME  STA371   (TSOLIB)        FOR
ISN 0002             SUBROUTINE STA371
               C-----------------------------------------------------------------
ISN 0003             DIMENSION ADR(1)
               C                                 PRINT SIN  AND  COS
ISN 0004             PI=3.141592
ISN 0005             DO 2 I=1.180.10
ISN 0006             RAD=I*PI/180
ISN 0007             SN=SIN(RAD)
ISN 0008             CS=COS(RAD)
ISN 0009             WRITE(6.4)I.SN.CS
ISN 0010       4     FORMAT(1X.I4.4F10.4)
ISN 0011       2     CONTINUE
               C                                FLOAT DIVIDE
ISN 0012             A=3.
ISN 0013             B=0
ISN 0014             C=A/B
ISN 0015             WRITE(6.6)A.B.C
ISN 0016        6    FORMAT(1X.'AFTER DIVIDE CHECK'.3F10.5)
               C                                FIXED DIVIDE
ISN 0017             I=5
ISN 0018             J=0
ISN 0019             K=I/J
ISN 0020             WRITE(6.8)I.J.K
ISN 0021        8    FORMAT(1X.'AFTER FIXED DIVIDE'.3I6)
               C                                OVERFLOW
ISN 0022             EOV=1.E60
ISN 0023             EOVL=EOV*EOV
ISN 0024             WRITE(6.10)EOV.EOVL
ISN 0025       10    FORMAT(1X.'OVERFL'.2E15.7)
               C                                UNDERFLOW
ISN 0026             EUN=1.E-60
ISN 0027             EUNV=EUN*EUN
               C                               WRONG UNIT
               C     WRITE(0.12)
ISN 0028             WRITE(6.12)
ISN 0029       12    FORMAT(1X.'AFTER WRONG UNIT')
               C                               NEGATIVE SQRT
ISN 0030             AA=SQRT(-1.)
ISN 0031             SQ=SQRT(-2.)
ISN 0032             WRITE(6.14)AA.SQ
ISN 0033        14   FORMAT(1X.'AFTER NEGATIVE SQRT'.2F15.5)
               C                               ADDRESS EXCEPTION
ISN 0034             WRITE(6.16)
ISN 0035        16   FORMAT(1X.'BEFORE ADDRESS VIOLATION')
ISN 0036             II=130000
ISN 0037             AB=ADR(II)
ISN 0038             WRITE(6.20)
ISN 0039        20   FORMAT(1X.'FINISH')
ISN 0040             RETURN
ISN 0041             END
```

Fig. A2. FORTRAN source code

```
F64-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST,SIZE=(438K,24K)
          VARIABLE OPTIONS USED -  SIZE=(448512,24576)
IE#0000      SETSSI 18218306
IE#0000      INCLUDE NEWLIB1(SYST370E)


                                              MODULE MAP

  CONTROL SECTION                          ENTRY

    NAME       ORIGIN  LENGTH                NAME     LOCATION      NAME     LOCATION
  MAIN0            00      78
  CPLIST           78      A4
  MAIN2           120      84
  SVCALL          1D8      10
  INTSRV          1E8      30
  READP           218       2
  IMOUAC          220     658
  FIOCS#          878     1F0              FIOCSBEP     878

  MVCOM           A68      66
  IADDR           AC0       6
  IDSTL           AC8      14
  IREGIS          AE0       A
  BFFRIB          AF0     216
  BUFSWI          D08     28E
  CHECON          F98     198
  CRUNBL         1130     24C
  FI0370         1380     A3E
  IBMIRA         1DC0     3A8
  READDR         2168     286
  RECADR         2420     2DE
  FTRACE         2700     76C
  YPRERR         2E70     240
  GETMAI         30B0     180
  DCBSET         3230     1A0
  INSSEG         33D0     1A0
  IAND     *     3570      3C              IOR         357E

  IHJECOMH*      3580     E30              IBCOM#      35E4      FDIOCS#      36A0
                                           INTSWTCH    4328
  FIOAP#   *     43E0     61C
  IHOCOMH2*      4A00     9A5
                                           SEGDASD     4DD2
  IHOUATBL*      53A8     636
  STA370   *     59E0      EC
  IHOFCVTH*      5AD0     CFA              ADCON#      5AD0      FCVADUTP     5B3C
                                           FCVLOUTP    5B64      FCVZOUTP     586C
                                           FCVIOUTP    5B84      FCVEOUTP     588C
                                           FCVCOUTP    5BBC      ADCONI#      62CC
                                           ADCONO#     62E8      INT6SWCH     67C9
  IHOEFNTH*      67D0     600
                                           ARITH#      6700      ADJSWTCH     6D64
  IHOUOPT  *     6FD0     535
  COPSUB   *     7508     4D4
  IHOERRM  *     79E0     624              ERRMON      79E0      IHOERRE      79F8

  IHOFCONI*      8008     416              FQCONI#     8008

  IHOFCONO*      8420     6B6              FQCONO#     8420

  IHOFTEN  *     8CD8     220              FTEN#       8CD8

  IHOETRCH*      8EF8     2AE              IHOTRCH     8EF8      ERRTRA       8F00

  IHCBUF         91A8      10
  IHCBF2         91B8    9C40
  TSTCOM        12DF8       4
  CIBWAT        12E00       4

 ENTRY ADDRESS         00

 TOTAL LENGTH       12E08
****E370TEMP   NOW REPLACED IN DATA SET
AUTHORIZATION CODE IS          0.
```

Buffer routines and 370/E system

Space for buffers and pointers

Fig. A3. Listing of the linkage editor.

```
JOB          :FIBNOT
TIME         :    1 MIN
START TIME   :02/07/82 06.34.21
MODULE NAME:FIBNOT.TSOLIBL(E370TEMP)
LIST FILE    :FIBNOT.LIST370E

//GO.FT09F001 DD DSN=FIBNOT.COMVBS
// DISP=(SHR        KEEP      KEEP      ).DS-ORG= 16384
ERROR 0004 0410 0000
//GO.FT09F001 DD DSN=FIBNOT.COMVBS
// DISP=(SHR        KEEP      KEEP      ).DS-ORG= 16384
//GO.FT08F001 DD DSN=FIBNOT.COMFB
// DISP=(SHR        KEEP      KEEP      ).DS-ORG= 16384
ERROR 0004 0410 0000
//GO.FT08F001 DD DSN=FIBNOT.COMFB
// DISP=(SHR        KEEP      KEEP      ).DS-ORG= 16384
//GO.FT88F001 DD DSN=FIBNOT.TSOLIBL(E370TEMP)
// DISP=(SHR        KEEP      KEEP      ).DS-ORG=    612
    1 .1745E-01 .9998
   11 .1908      .9816
   21 .3584      .9336
   31 .5150      .8572
   41 .6561      .7547
   51 .7771      .6293
   61 .8746      .4848
   71 .9455      .3256
   81 .9877      .1564
   91 .9998     -.1745E-01
  101 .9816     -.1908
  111 .9336     -.3584
  121 .8572     -.5150
  131 .7547     -.6561
  141 .6293     -.7771
  151 .4848     -.8746
  161 .3256     -.9455
  171 .1564     -.9877
FLOATING POINT DIVIDE
PSW 00008A90 IL+CC 0000000A
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14     REG. 15     REG.  0     REG.  1
              YPRERR                    00113D06    00002E70    00000000    00002790
              FTRACE                    60000202    00002700    00000085    91119AD4
              STA371                    40005A90    00008800    400001AA    00000000
              STA370                    500001B6    000059E0    400001AA    400001AA
              MAIN2                     5000000C    00000120    00000000    00000005
AFTER DIVIDE CHECK3.0000       .0       3.0000

FIXED POINT DIVIDE
PSW 00008ACC IL+CC 0000000A
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14     REG. 15     REG.  0     REG.  1
              YPRERR                    00113D06    00002E70    00000000    00002790
              FTRACE                    60000202    00002700    00000000    00000005
              STA371                    40005A90    00008800    400001AA    00000000
              STA370                    500001B6    000059E0    400001AA    400001AA
              MAIN2                     5000000C    00000120    00000000    00000005
AFTER FIXED DIVIDE         5        0        5

EXPONENT OVERFLOW
PSW 0000BB04 IL+CC 00000001
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14     REG. 15     REG.  0     REG.  1
              YPRERR                    00113D06    00002E70    00000000    00002790
              FTRACE                    40000202    00002700    00000000    00000005
              STA371                    40005A90    00008800    400001AA    00000000
              STA370                    500001B6    000059E0    400001AA    400001AA
              MAIN2                     5000000C    00000120    00000000    00000005
OVERFL  0.1000000E+61   0.7458339E-34
AFTER WRONG UNIT
```

From IBM
Online Job
Allocate files

Hardware interrupts

Fig. A4. Output of the 370/E on LISTFILE at the IBM

Fig. A4. continued

Software interrupts

```
IHO251I SQRT ARG=-0.1000000E+01, LT ZERO
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14    REG. 15    REG. 0     REG. 1
           SQRT               0030      4000885A   000096A8   00000060   88000000
           STA371             0002      40005A90   00008800   400001AA   00000000
           STA370                       500001B6   000059E0   400001AA   400001AA
           MAIN2                        5000000C   00000120   00000000   00000005
STANDARD FIXUP TAKEN , EXECUTION CONTINUING
IHO251I SQRT ARG=-0.2000000E+01, LT ZERO
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14    REG. 15    REG. 0     REG. 1
           SQRT               0030      60008B6C   000096A8   00000060   90000000
           STA371             0002      40005A90   00008800   400001AA   00000000
           STA370                       500001B6   000059E0   400001AA   400001AA
           MAIN2                        5000000C   00000120   00000000   00000005
STANDARD FIXUP TAKEN , EXECUTION CONTINUING
AFTER NEGATIVE SQRT     1.0000          1.4142
BEFORE ADDRESS VIOLATION

ADDRESS EXCEPTION    0C5
PSW 00008B60 IL+CC 00000002
TRACEBACK  ROUTINE   CALLED FROM ISN    REG. 14    REG. 15    REG. 0     REG. 1
          YPRERR                        00113D06   00002E70   00000000   00002790
          FTRACE                        40000202   00002700   0A0088F6   00000041
          STA371                        40005A90   00008800   400001AA   00000000
          STA370                        500001B6   000059E0   400001AA   400001AA
          MAIN2                         5000000C   00000120   00000000   00000005
EXECUTION TERMINATED
```

Acknowledgement

I would like to thank H. Brafman, R. Fall and R. Yaari for the development of the 370/E processor. This paper was strongly influenced by J. Hart (RAL) and his investigation of IBCOM# and FIOCS# . The routines for dynamic allocation of files were written by P. Schilling (DESY, R2).