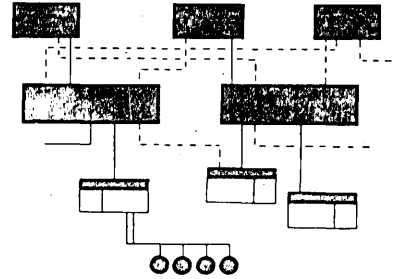


TIBS:
600-60
- 96
- 183
- 223

GE-600 Series Fortran IV Input / Output Library

SYSTEM
SUPPORT
INFORMATION



ABSTRACT

This manual describes the implementation details of the FORTRAN IV Input/Output Routines available for use with all configurations of the COMPATIBLES/600.

GENERAL  ELECTRIC

| | | |
|--|--|-------------------|
| GENERAL ELECTRIC INFORMATION SYSTEMS DIVISION COMPUTER EQUIPMENT DEPARTMENT | GE-600 SERIES TECHNICAL INFORMATION BULLETIN | DATE Oct. 1968 |
| | | NO. 600-223 |
| SUBJECT: Changes to FXEM and FSLEW | | REF. CPB-1137 |

The FXEM and FSLEW revisions contained in this TIB include features implemented in GECOS-III System Development Letter 1. The LHSF writeup was issued December 1965 as TIB 600-60. It is reissued here as change pages for convenience, since in its earlier form it was not part of the manual.

Replace old pages in GE-625/635 FORTRAN IV Input/Output Library System Support Information, CPB-1137, with attached new pages as follows:

| <u>Remove</u> | <u>Replace</u> | <u>Insert</u> |
|-----------------|-----------------|---------------|
| iii/iv 31-34 | iii/iv 31-34 | |
| | | 34.1-34.5 |
| 35/36 43/44 | 35/36 43/44 | |
| | | 44.1-44.3 |
| 45/46 | 45/46 | |

Vertical bars in the margins of these new pages indicate changes or additions to the existing text. This new information will be included in the next edition of the manual.

Place this sheet in the front of your manual to show that the contents of this TIB have been incorporated.

TIBs that currently apply to CPB-1137 are:

600-60 (superseded by this TIB)
 600-96
 600-183
 600-223

| | | |
|--|--|--------------------|
| GENERAL ELECTRIC INFORMATION SYSTEMS DIVISION COMPUTER EQUIPMENT DEPARTMENT | GE-600 SERIES TECHNICAL INFORMATION BULLETIN | DATE March 1968 |
| | | NO. 600-183 |
| SUBJECT: Revisions to the FORTRAN IV Binary I/O Interface Routine | | REF. CPB-1137 |

Please remove the following pages from your GE-625/635 FORTRAN IV Input/Output Library SSI manual and replace them with the attached revised pages of the same number. This TIB includes new features implemented in Systems Development Letter 12.

iii/iv
 9/10 ~~21/22~~
 23/24
 33/34

In addition, insert the following attached pages which are new:

24.1/blank
 34.1/34.2
 45/46
 47/blank

It is suggested that you add this page to the front of your manual to show that this TIB has been entered and that the following TIB's are now in effect:

TIB 600-60
 TIB 600-96
 TIB 600-183

TIB Disposition

This information will be incorporated into the next addition of the manual.

GENERAL  ELECTRIC

COMPUTER DEPARTMENT

GE-600 SERIES

TECHNICAL INFORMATION BULLETIN

DATE
April 1966

NO.
600-96

SUBJECT:

Corrections to GE-625/635 FORTRAN IV I/O Library


REF.
CPB-1137

INSTRUCTIONS

The attached pages replace pages 43 and 44 of the GE-625/635 FORTRAN IV Input/Output Library SSI, CPB-1137.

TIB DISPOSITION

The revised pages will appear in the next edition of GE-625/635 FORTRAN IV Input/Output Library SSI, CPB-1137.

| | | |
|--|--|-------------------|
| GENERAL  ELECTRIC COMPUTER DEPARTMENT | GE-600 SERIES TECHNICAL INFORMATION BULLETIN | DATE Dec. 1965 |
| | | NO. 600-60 |
| SUBJECT: Restart of a Link Job From H* Tape (FORTRAN IV Input/ Output Library) | | REF. CPB-1137 |

The attached pages describe in detail the procedures used in reloading a saved H* tape file, which has been created by a previous run. (Procedures used to create an H* file are described in GE-625/635 General Loader, CPB-1008. To load the subsequent links, use the LLINK or LINK subroutine as described in the GE-625/635 FORTRAN IV Input/Output Library, CPB-1137.) The attached pages should be inserted in the GE-625/635 FORTRAN IV Input/Output Library, CPB-1137.

Subsequent issues of CPB-1137 will include this information.

230
346
576

.LHSF - .LHSNF -- RESTART OF LINK JOB FROM H* TAPE

I. PURPOSE

To reload a program from an H* file (tape) which was generated in a previous GELOAD activity. The H* file was generated by having a \$ TAPE H* control card (see GE-625/635 General Loader CPB-1008) at execution time.

II. METHOD

The H* file generated by GELOAD contains a link identified as /////// which is the main or common subprogram of the job. If the FCB option was in effect during loading (generation of H* file), a second link identified as ///////1 containing all file control blocks generated by GELOAD will also be present on H*. This subroutine searches the H* file for these identifiers (///////1 is optional), restores them, and enters the main subprogram at the entry location specified during the GELOAD activity.

III. USAGE

1. This program is called directly from the subroutine library and requires no other subprograms.
2. The entire job could be set up as follows:

```

$ SNUMB
$ IDENT
$ USE .LHSF
$ ENTRY .LHSF
$ EXECUTE
$ LIMITS
$ TAPE H*,...
$ DATA AB',... (Optional)
.
.
.
$ ENDJOB
***EOF

```

If the NOFCB option was in effect when GELOAD generated the H* file, an entry card of the form:

```
$ ENTRY .LHSNF
```

must replace the card following the \$ USE card.

IV. RESTRICTIONS

1. A \$ LOWLOAD card (see CPB-1008) must be included if the H* file was generated under this option.
2. The same memory limits must be requested as were when the H* file was generated.
3. Use of one of the set up subroutines must have been made when the H* file was generated. Entry to the main link is made through those subroutines for purposes of initialization of fault vectors.



**GE-600 SERIES
FORTRAN IV INPUT / OUTPUT
LIBRARY**

May 1965

GENERAL  ELECTRIC

COMPUTER DEPARTMENT

CONTENTS

| | <u>Program No.</u> | <u>Page</u> |
|--|--------------------|-------------|
| INTRODUCTION | | 1 |
| DEBUG--Object Time Debug Processor | E1.001 | 3 |
| DUMP--Memory Dump | E1.002 | 5 |
| EXIT--Job Termination | E1.003 | 7 |
| FBIO--Short List Binary I/O Interface | E1.004 | 8 |
| FBST--Backspace Record | E1.005 | 9 |
| FCLOSE--File Closing | E1.006 | 11 |
| FEFT--Rewind and End File Processor | E1.007 | 12 |
| FEOF--End-of-File Processor | E1.008 | 13 |
| FLDO--Double Precision Powers of Ten | E1.009 | 14 |
| FLGEOF--Initialization of End-of-File Processing | E1.010 | 15 |
| FLGERR--Initialization of Data Error Processing | E1.011 | 16 |
| FOPEN--File Opening | E1.012 | 17 |
| DVCHK--Exponent Overflow and Divide Check Tests | E1.013 | 20 |
| FRWB--Binary I/O Interface | E1.014 | 21 |
| FRWD--BCD I/O Interface by Format Control | E1.015 | 23 |
| FSIO--Short List BCD I/O Interface | E1.016 | 25 |
| FSLIO--Short List I/O Processor | E1.017 | 26 |
| SLITE--Sense Light Simulator | E1.018 | 27 |
| SSWTCH--Sense Switch Test | E1.019 | 28 |
| FVFI--NAMELIST Input | E1.020 | 29 |

| | <u>Program No.</u> | <u>Page</u> |
|--|--------------------|-------------|
| FVFO--NAMELIST and Debug Output | E1.021 | 31 |
| FXEM--Execution Error Monitor | E1.022 | 32 |
| LINK--Restore Links During Execution | E1.023 | 36 |
| STORE--Access Half of a Double Precision or Complex Word | E1.024 | 38 |
| SETBUF--Define a Buffer(s) for a Specified File Control Block | E1.025 | 39 |
| SETFCB--Define File Control Block | E1.026 | 40 |
| SETLGT--Define Logical File Table | E1.027 | 41 |
| SETUP--Pre-Execution Initializer | E1.028 | 42 |
| FSLEW--Carriage Control Simulator | E1.029 | 44 |
| LHSF--Restore Link - H* | E1.030 | 44.3 |
| FINC--BCD Internal Conversion Interface | E1.031 | 46 |

INTRODUCTION

Individuals interested in details of the FORTRAN Input/Output subprograms will find the following description of their general mode of operation useful. It is assumed that the reader is familiar with the manner in which the General Loader (GELOAD) processes the GECOS and FFILE file control cards.

The Input/Output library is dependent upon the fact that execution has started with the initialization subprogram .SETU. for one of its functions is to initialize fault vector cell 25₈ to contain the address of the "logical file--file control block" table. The user can also accomplish this initialization by calling SETLGT in the case where he has created his own table. The library also depends upon the limits of unused memory being expressed in fault vector cell 37₈ (always done by GELOAD).

A call to any I/O library subprogram from the FORTRAN language contains, as one of the arguments, the logical file expressed as an integer. This integer is placed in character position 5 of cell .FBAD. (defined in subprogram FOPEN) by the called I/O subprogram. The called subprogram then calls FOPEN which searches the "logical file--file control block" table defined below:

1. Fault vector cell 25₈ is of the form:

```
ZERO  TAB,0
```

2. The actual table has the form:

```
ZERO  ENDTAB,0
TAB  VFD  18/FCB1,6/LGU1,6/LGU2,6/LGU3
      VFD  18/FCB2,6/LGU4,6/LGU5,6/LGU6
      .
      .
      .
ZERO  0,0
      .
      .
      .
ENDTAB ZERO  0,0
```

where:

TAB-1 contains the address of the last available location in the table

FCB1 contains the address of cell LOCSYM of file control block 1

LGU1, LGU2, LGU3 are the FORTRAN logical files which reference file control block 1. Missing files are filled in with zero.

If more than three logical files reference the same file control block, the FCB1 pointer and the additional logical files may occur at any other place in the table.

There are as many entries in the table as needed to express the various file control blocks and logical files referencing them. After the last entry in the table, zeros fill out the table.

FOPEN places the address of the file control block for the referenced file in bits 0-18 of cell .FBAD., but does not destroy the logical file in bits 30-35 of that cell. FOPEN then proceeds to open the file and return. *number?*

The subprogram which initiated the call to FOPEN now has the information necessary to perform calls to the proper GEFRC subprograms. In the case of an output file, bits 30-35 of cell .FBAD. are used as the report code of the output record. Thus, if many logical files are connected to SYSOUT, they will be separated automatically at printing time according to the file code originally specified in the FORTRAN calling sequence.

File control blocks

the .UNON.

*no longer appear
in output listing.*

DEBUG--OBJECT TIME DEBUG PROCESSOR

I. PURPOSE

To decide whether or not to produce debug output based on the contents of the IF and FOR clauses.

II. METHOD

DEBUG assumes the General Loader (GELOAD) has placed a debug table in memory. DEBUG is entered by a DRL instruction. The location of the DRL instruction is checked to see if it was inserted by GELOAD (a legal debug request), or if it was originally present in the interrupted program. If it was a debug request, the IF and FOR clauses are examined in the order in which they were specified. If they are satisfied, DEBUG writes information describing the particular interrupt it is interpreting. It then calls FVFO for the list output, using the NAMELIST table supplied in the debug table by GELOAD. The instruction replaced by the DRL instruction is then executed and control is returned to the next instruction in the interrupted program. If the DRL instruction was not inserted by GELOAD, it is ignored and control is returned to the interrupted program. For a more detailed description of the DEBUG feature consult the GE-635 General Loader Reference Manual, CPB-1008.

III. USAGE

1. Calling Sequence--entered by a DRL instruction.
2. DEBUG uses 285 memory locations.
3. No error conditions.

IV. RESTRICTIONS

The subprograms FRWD, FOPEN, and FVFO must be in memory.

DUMP--MEMORY DUMP

I. PURPOSE

To dump all or designated areas of memory in a selected format.

II. METHOD

An appropriate NAMELIST table is created using the parameters specified in the calling sequence. DUMP calls FVFO for the actual NAMELIST output processing. The panel is dumped followed by the blocks of memory requested. If DUMP was called, execution is terminated by a call to EXIT. If PDUMP was called, the panel is restored and control is returned to the calling program.

III. USAGE

1. Calling Sequence - CALL DUMP (A₁, B₁, F₁, ..., A_n, B_n, F_n)
CALL PDUMP (A₁, B₁, F₁, ..., A_n, B_n, F_n)

where A_i and B_i are arguments that indicate the limits of memory to be dumped. Either A or B may represent the upper or lower limits.

F_i is an integer indicating the dump format desired:

F_i = 0 octal

F_i = 1 integer

F_i = 2 real

$F_i = 3$ double precision

$F_i = 4$ complex

$F_i = 5$ logical

If the last F_i is omitted, it is assumed to be zero.

If no arguments are given, all of memory is dumped
in octal.

2. DUMP uses 160 memory locations.

IV. RESTRICTIONS

The subprograms EXIT and FVFO must be in memory.

EXIT--JOB TERMINATION

I. PURPOSE

To purge all buffers and terminate current activity.

II. METHOD

EXIT transforms the logical file table created by the General Loader (GELoad) into a file designator word list for closing all files. It calls CLOSE which purges the buffers and writes an end of file on an output file and notes the closing of an input file. Execution is terminated by a MME GEFINI.

III. USAGE

1. Calling Sequence - CALL EXIT
or CALL .FEXIT

EXIT and .FEXIT are equivalent.

2. EXIT uses 30 memory locations.
3. No error conditions.

IV. RESTRICTIONS

The subprogram CLOSE must be in memory.

EXIT exists in the library as a subset of FOPEN.

FBIO--SHORT LIST BINARY I/O INTERFACE

I. PURPOSE

To call FSLIO for short list binary I/O.

II. METHOD

FBIO consists of four calling sequences for FSLIO.

III. USAGE

1. Calling Sequence - CALL .FBLO.(A,M) for single-precision
binary output,
CALL .FBDO.(A,M) for double-precision
binary output,
CALL .FBLI.(A,M) for single-precision
binary input,
CALL .FBDI.(A,M) for double-precision
binary input,

where A = location of array, and M = location of number
of elements.

2. FBIO uses 20 memory locations.
3. No error conditions.

IV. RESTRICTIONS

The subprograms FRWB and FSLIO must be in memory.

FBST--BACKSPACE RECORD

I. PURPOSE

To backspace one logical record on a file.

II. METHOD

The buffer for the file is inspected and the number of logical records in the buffer is obtained. The contents of the buffer are written on the file if it is an output file. The file is physically backspaced and the buffer re-initialized. If the original number of logical records in the buffer was N, N-1 logical records are skipped by successive calls to GET. By using this technique, which always involves physical backspacing, FBST remains more independent of the General File and Record Control program (GEFRC).

so that there is something to backspace over & read into buffer. It would require that another backspace be given after these reads. It also means that the file is closed as output & reopened as input.

III. USAGE

1. Calling Sequence - CALL .FBST.(N)

where N is the logical file to be backspaced.

2. FBST uses 240 memory locations.

3. The error conditions are:

FXEM error #47 if logical file requested was connected either to SYSOUT or to a physical device other than magnetic tape. FXEM terminates execution.

FXEM error #49 if an erroneous end of file appears.

FXEM terminates execution.

IV. RESTRICTIONS

1. The subprograms GET, BSREC, EXIT, FOPEN, and FXEM must be in memory.
2. If more than one logical file refers to the same physical file, a request to backspace the physical file results in pointing to the last previous logical record, regardless of the logical file to which it is connected.

FCLOSE--FILE CLOSING

I. PURPOSE

To close a file and release the buffer assigned to that file. The buffer is released only if it is standard size (320 words).

II. METHOD

FCLOSE opens the file in its previous mode and calls the General File and Record Control (GEFRC) subprogram CLOSE to close the file without rewind. FCLOSE examines and releases any buffers of standard size assigned to the file. It places their memory address in a table of available buffers (location .FBFTB in the FOPEN subprogram) for possible reassignment to a newly opened file.

III. USAGE

1. Calling Sequence - CALL FCLOSE(U)
where U is the logical file number.
2. FCLOSE uses 47 memory locations.

IV. RESTRICTIONS

1. If more than one logical file refers to one physical file, the physical file must be closed using FCLOSE only once.
2. The subprograms CLOSE and FOPEN must be in memory.

FEFT--REWIND AND ENDFILE PROCESSOR

I. PURPOSE

To rewind input or output files or write an end of file on output files.

II. METHOD

FEFT records the desired rewind or write end-of-file option in a file designator word. FEFT calls subprogram CLOSE which writes an end of file (if the file designated is an output file) followed by a rewind (if requested).

III. USAGE

1. Calling Sequences - CALL .FEFT.(N) compiled for the
FORTRAN statement ENDFILE N
CALL .FRWT.(N) compiled for the
FORTRAN statement REWIND N
where N is the logical file desired.

2. FEFT uses 70 memory locations.

3. The error condition is:

FXEM error #35 if there is an attempt to rewind or write end of file on files I* or P*. Execution is continued with normal return to caller.

IV. RESTRICTIONS

The subprograms FOPEN, OPEN, CLOSE, FXEM must be in memory.

FEOF--END-OF-FILE PROCESSOR

I. PURPOSE

To write an end-of-file message and either terminate execution or return to the calling program.

II. METHOD

If a library subroutine detects an end of file on an input file, it calls FEOF. FEOF places the file label in a message and calls FXEM to print the message. FEOF either terminates execution, or if the user has provided for end-of-file condition by having previously called FLGEOF for the current file, FEOF returns to the calling program indicating an end of file was encountered.

III. USAGE

1. Calling Sequence - CALL .FEOF.

It is assumed that the address of the proper file control block is in the FOPEN subprogram cell .FBAD.

2. FEOF uses 35 words.

3. FXEM error #34 is always used to write the end-of-file message.

IV. RESTRICTIONS

The subprograms FOPEN and FXEM must be in memory.

F1D0--DOUBLE PRECISION POWERS OF TEN

I. PURPOSE

To store a table of double-precision powers of ten for quick reference by all decimal radix conversion routines.

II. METHOD

.F1D0. + 2*n is the location of DEC 1.0Dn = 10^n , for
n = -38, -37, ..., -1, 0, 1, ..., 37, 38.

III. USAGE

1. .F1D0. is the only SYMDEF symbol.
2. F1D0 uses 156 memory locations.
3. No error conditions; no executable instructions.

IV. RESTRICTIONS

None.

FLGEOF--INITIALIZATION OF END-OF-FILE PROCESSING

I. PURPOSE

To provide a signal to FEOF requesting a return to the calling subprogram if an end-of-file condition occurs.

II. METHOD

The address of the variable to be used for end-of-file processing is placed in word -15 of the file control block. The value of the variable is set to zero. (INITIAL VALUE)

III. USAGE

1. Calling Sequence - CALL FLGEOF (U,V)

where U is the logical file number

V is the address of variable used to indicate

an end-of-file condition. $C(V) = 1$ when EOF encountered
 $= 0$ " " not encountered

2. FLGEOF uses 18 memory locations.

IV. RESTRICTIONS

1. If more than one logical file refers to one physical file, the same variable must be used for all logical files referring to that physical file.

2. The subprogram FOPEN must be in memory.

This is bad
If the variable is an integer and later used in a computed GO TO, there is no "zero" branch in the list and we would have to do this:

```
CALL FLGEOF (5, I)
I = I + 1
GO TO (10, 20), I
10 READ (, -)
```

or else it requires the variable to be declared logical to be used
 $IF (I) GO TO 20$ $\left\{ \begin{array}{l} 1 = \text{true} \\ 0 = \text{false} \end{array} \right.$

FLGERR--INITIALIZATION OF DATA ERROR PROCESSING

I. PURPOSE

To provide a variable for indicating the occurrence of erroneous data. (parity?)

II. METHOD

FLGERR places the address of the variable in word -16 of the file control block. The value of the variable is set to zero.

III. USAGE

1. Calling Sequence - CALL FLGERR (U,V)

where U is the logical file number

V is the address of the variable used to indicate an input data error.

*CV) = 1 + error
- 0 = no error*

2. FLGERR uses 18 memory locations.

IV. RESTRICTIONS

1. If more than one logical file refers to one physical file, the same variable must be used for all logical files referring to that physical file.
2. The subprogram FOPEN must be in memory.

FOPEN--FILE OPENING

I. PURPOSE

To select and assure that the physical file associated with the desired logical file is open.

II. METHOD

FOPEN is divided into three phases:

1. Locating the physical file,
2. Assigning buffers to the file,
3. Assuring the file is open.

FOPEN examines the logical file table for a logical file identical to the one in cell .FBAD. If one is found, FOPEN places the address of the file control block associated with that logical file in cell .FBAD. The file control block is examined to determine if buffers are required that have not been previously assigned. If buffer assignment is necessary, the table .FBFTB is examined to see if any buffers have been released that were previously assigned to another file; these are used first. If none exist, the buffers are assigned from the available unused memory. The file control block is again examined to see if the file is open. If it is not open, the General File and Record Control (GEFRC) subprogram OPEN is called for the proper file control block. Control is returned to the calling program.

III. USAGE

1. Calling Sequences -

- a. CALL .FGTFB to obtain the address of the file control block in the address field of cell .FBAD. only.
- b. CALL .FOPEN (S) to obtain the information described in CALL .FGTFB and to assure that the logical file is open.

S indicates the mode in which the file is to be opened:

S = -1 Open the file in its previous mode.

S = odd Open the file as output.

S = even Open the file as input.

In both calling sequences it is assumed that upon entry to FOPEN the logical file referenced is contained in character position 5 ^(bits 30-35) of cell .FBAD.

- c. The table of standard length reusable buffers (321 words) is defined as .FBFTB.
- d. An equivalent entry to .FOPEN is .FXOP. which is used by the FXEM subprogram to prevent destroying the calling sequences in case of recursive entry.

2. FOPEN uses 210 memory locations.

3. The error conditions are:

FXEM error #37 if the logical file requested is not present in the logical unit table; FXEM terminates execution.

FXEM error #38 if there is not enough memory available for buffer assignment; FXEM terminates execution.

FXEM error #56 if there is an attempt to read file P* (SYSOUT); FXEM terminates execution.

FXEM error #54 if there is an attempt to write on file I* (SYSIN); FXEM terminates execution.

Abort code Q2 if no logical file table exists.

Abort code Q3 if logical file 06 does not exist in the logical unit table; a message from FXEM cannot be written.

IV. RESTRICTIONS

The subprograms OPEN, SETIN, SETOUT, and FXEM must be in memory.

DVCHK--EXPONENT OVERFLOW AND DIVIDE CHECK TESTS

I. PURPOSE

To test the General Comprehensive Operating Supervisor (GECOS) Fault Status Word for a previous exponent register overflow or divide check.

II. METHOD

DVCHK tests the required bit in the Fault Status Word, and sets an integer variable to 1 if ON, or 2 if OFF. DVCHK always exits with the required bit turned OFF (zero).

III. USAGE

1. Calling Sequence - CALL DVCHK(J) for divide check,
CALL OVERFL(J) for exponent register
overflow,

where J = location of integer variable.

2. DVCHK uses 20 memory locations.

3. No error conditions.

IV. RESTRICTIONS

DVCHK assumes normal GECOS recovery for exponent register overflow and divide check.

WRITE (fwd) VL

I. PURPOSE

To process binary I/O lists, using standard General File and Record Control (GEFRC) variable-length records.

II. METHOD

FRWB groups binary logical records into physical records of the size specified in the file control block. If the size of the logical record exceeds the physical buffer size, or if there is insufficient space remaining in the physical buffer for the logical record, the logical record is partitioned into segments spanning as many physical records as required.

us 318
wrto)

The record control word for each segment of the logical record contains a media code 3. The last segment is followed by a 1-word record whose record control word contains a media code 1. All logical records which are completely contained within a physical record have a media code 1 in the record control word.

III. USAGE

1. Calling Sequence - CALL .FRDB.(ARG) compiled for READ (n),
CALL .FWRB.(ARG) compiled for WRITE (n),
CALL .FRLR.(ARG) compiled for end-of-
input list,

CALL .FWLR.(ARG) compiled for end-of-
output list,

TSX1 .FBDT. } for double-precision
STAQ address, tag } input list items,

TSX1 .FBLT. } for all other input
STA address, tag } list items,

| | | |
|-------------------|---|--|
| LDAQ address, tag | } | for double-precision output list items, |
| TSX1 .FBDT. | | |
| LDA address, tag | } | for all other output list items, |
| TSX1 .FBLT. | | |

where ARG = location of DEC n.

2. FRWB uses 239 memory locations.
3. The error conditions are:

FXEM Error #34 - illegal end-of-file mark.

FRWB calls FEOF for error recovery.

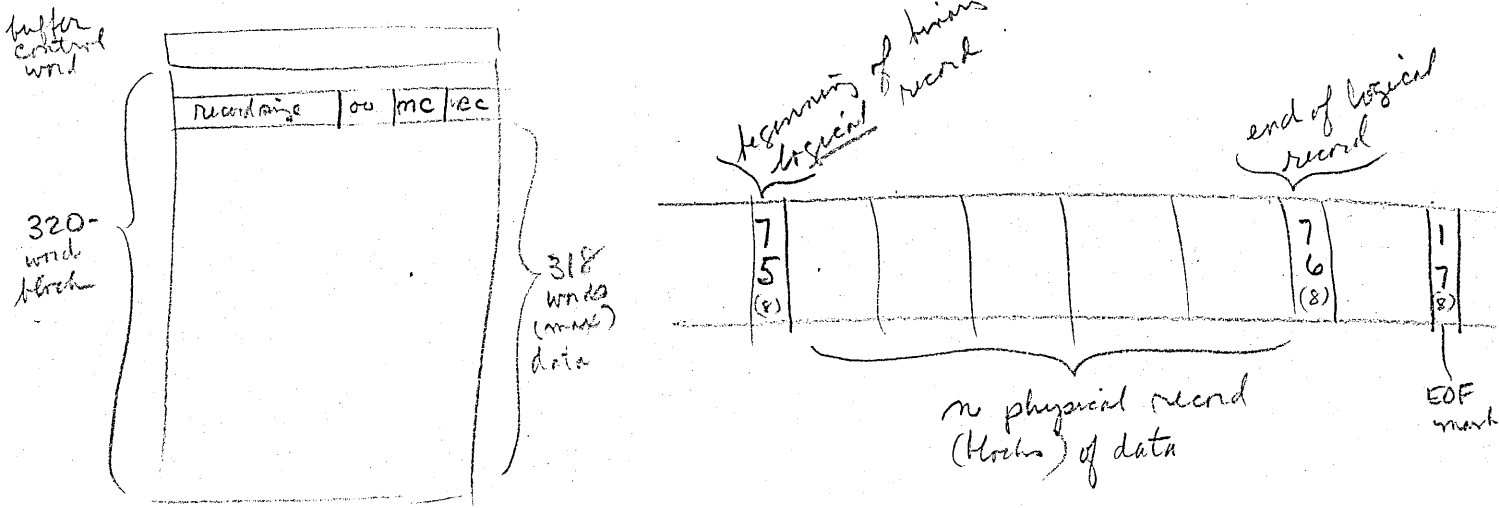
If an end-of-file other than 17_8 , 75_8 , or 76_8 is detected, the activity will be aborted (CB-Abort) unless an error return has been provided.

FXEM Error #40 - list exceeds logical record length.

All remaining list items are set to zero.

IV. RESTRICTIONS

The subprograms FEOF, FXEM, FOPEN, GET, and WTREC must be in memory.



FRWD--BCD I/O INTERFACE BY FORMAT CONTROL

I. PURPOSE

To process a list for BCD I/O or for internal conversion, in parallel with a FORMAT statement, using standard General File and Record Control (GEFRC) variable-length records.

II. METHOD

Each list item corresponds to a FORMAT specification. Each BCD record is a card image (80 columns) or a print line image (up to 132 columns, including carriage control).

III. USAGE

1. Calling Sequence - CALL .FRDD.(ARG,LF) compiled for

READ (n,f),

CALL .FWRD. (ARG,LF) compiled for

WRITE (n,f),

CALL .FRCD.(ARG,LF) compiled for

READ f, [n=41] *n is file code*

CALL .FPRN.(ARG,LF) compiled for

PRINT f, [n=42]

CALL .FPUN.(ARG,LF) compiled for

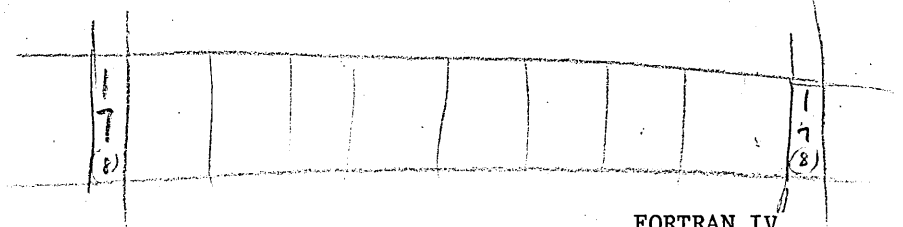
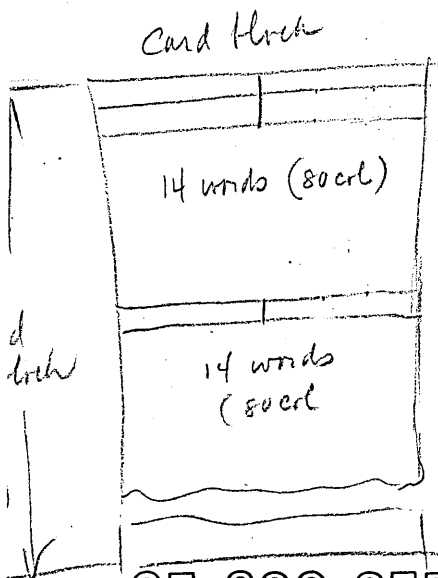
PUNCH f, [n=43]

CALL .FRTN. compiled for end-of-

input list,

CALL .FFIL. compiled for end-of-

output list,



GE-600 SERIES

FORTRAN IV
I/O LIBRARY

TSX1 .FCNV.
STAQ address, tag } for double-precision input list items,

TSX1 .FCNV.
STA address, tag } for all other input list items,

LDAQ address, tag }
TSX1 .FCNV. } for double-precision output list items,

LDA address, tag }
TSX1 .FCNV. } for all other output list items,

where ARG = location of ARG XX,

XX = location of DEC n,

and LF = location of FORMAT statement f.

2. Calling sequences for internal conversion.

a) Binary-to-decimal conversion

CALL .BDCNV(BUF,FORM,WDA,LINES)

LDAQ (data) }
TSX1 .FCNV. } for double precision items

LDA (data) }
TSX1 .FCNV. } for single precision items

CALL .FFIL. for end of output list

where:

BUF location of a buffer in which to store the resulting BCD record

FORM location of the format controlling the conversion

WDA location of an array in which to store the number of words in each of the resulting lines.

LINES location in which to store the number of lines.

b) Decimal-to-binary conversion

```
CALL .DBCNV(BUF,FORM,WDA,LINES)
TSX1 .FCNV. }
STAQ (data) } for double precision items
TSX1 .FCNV. }
STA (data) } for single precision items
CALL .FRTN. for end of input list
```

where:

BUF location of the BCD record to be converted
FORM location of the format controlling the conversion
WDA location of an array containing the number of words in each line in the BCD record
LINES location containing the number of lines

3. FRWD uses 1146 memory locations.
4. The error conditions are:

FXEM Error #31 - illegal FORMAT statement,
FORMAT scan proceeds as for end of FORMAT.

FXEM Error #32 - illegal character in data or bad format.
Data scan treats illegal character as zero.

FXEM Error #34 - illegal end of file mark.
FRWD calls FEOF for error recovery.

FXEM Error #57 - illegal character for L conversion.
Data scan treats illegal character as space.

*must be
.T. or .F. or
.TRUE. or .FALSE.*

IV. RESTRICTIONS

The subprograms FIDO, FEOF, FOPEN, FSLEW, FXEM, GET, and WTREC must be in memory.

FSIO--SHORT LIST BCD I/O INTERFACE

I. PURPOSE

To call FSLIO for short list BCD I/O.

II. METHOD

FSIO consists of four calling sequences for FSLIO.

III. USAGE

1. Calling Sequence -CALL .FSLO.(A,M) for single-precision
BCD output,
CALL .FSDO.(A,M) for double-precision
BCD output,
CALL .FSLI.(A,M) for single-precision
BCD input,
CALL .FSDI.(A,M) for double-precision
BCD input,

where A = location of array, and M = location of number
of elements.

2. FSIO uses 20 memory locations.
3. No error conditions.

IV. RESTRICTIONS

The subprograms FRWD and FSLIO must be in memory.

FSLIO--SHORT LIST I/O PROCESSOR

I. PURPOSE

To provide list processing for a nonsubscripted array, in conjunction with subprogram FBIO or FSIO.

II. METHOD

FSLIO initializes the I/O loop, and processes the entire array, starting with the first element.

III. USAGE

1. The calling sequence stores C(X2) in .FSLII and points C(X2) to a three-word parameter vector. The first two words, an even-odd pair, specify the I/O calling skeleton for the loop. The third word specifies the precision (1 or 2) and the state (0 for output, 1 for input) in 18-bit fields.

Calling Sequence -

| <u>Output</u> | | <u>Input</u> | |
|---------------|----------|--------------|----------|
| E STX2 | .FSLII | E STX2 | .FSLII |
| TSX2 | .FSLIO,I | TSX2 | .FSLIO,I |
| E LDr | ** ,2 | E TSX1 | c |
| TSX1 | c | STr | ** ,2 |
| ZERO | p ,0 | ZERO | p ,1 |

Single precision: r = A and p = 1.

Double precision: r = AQ and p = 2.

c = .FCNV., .FBDT., or .FBLT..

2. FSLIO uses 24 memory locations.
3. No error conditions.

IV. RESTRICTIONS

FSLIO restores C(X2) = C(.FSLII)₀₋₁₇ and exits to the caller of FBIO and FSIO. Furthermore, FSLIO uses the input parameters of FBIO or FSIO.

SLITE--SENSE LIGHT SIMULATOR

I. PURPOSE

To simulate the setting and testing of sense lights.

II. METHOD

Bits 1-35 of .FLITE correspond to Sense Lights 1-35, respectively, with 0 denoting OFF and 1 denoting ON.

III. USAGE

1. Calling Sequence -CALL SLITE(ZERO) to clear Sense

Lights 1-35,

CALL SLITE(I) to turn ON Sense

Light i,

CALL SLITET(I,J) to test and turn

OFF Sense Light i,

where ZERO = location of integer 0,

I = location of integer i,

and J = location of an integer variable to be set
to 1 if Sense Light i was ON, or 2 if it
was OFF.

Note that .FLITE is a SYMDEF symbol.

2. SLITE uses 56 memory locations.

3. The error condition is:

FXEM Error #51 if i is not 0-35, or if i is 0 in SLITET.

Sense Light i is ignored if setting, or is declared
OFF if testing.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

SSWTCH--SENSE SWITCH TEST

I. PURPOSE

To test the General Comprehensive Operating Supervisor (GECOS) switch word for the status of a sense switch.

II. METHOD

Bits 6-11 of the GECOS switch word correspond to Sense Switches 1-6, respectively, with 0 denoting OFF and 1 denoting ON.

III. USAGE

1. Calling Sequence - CALL SSWTCH(I,J) to test Sense Switch i,
where I = location of integer i
J = location of an integer variable to be set to 1 if Sense Switch i is ON, or 2 if it is OFF.
2. SSWTCH uses 42 memory locations.
3. The error condition is:
FXEM Error #53 if i is not 1-6.
Then Sense Switch i is declared to be OFF.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

FVFI--NAMELIST INPUT

I. PURPOSE

To process NAMELIST input using the General File and Record Control (GEFRC) variable-length records.

II. METHOD

FVFI scans the input file for the proper NAMELIST name. When the name is found, FVFI scans the record for variables, confirms that the variables are included in the NAMELIST, and stores the input value according to the type specified in the NAMELIST table.

III. USAGE

1. Calling Sequence - CALL .FVFI.(ARG,LF)

where ARG = location of ARG XX

XX = location of DEC n (logical unit)

LF = location of NAMELIST table

2. FVFI uses 670 memory locations.
3. In the error conditions described below, execution is continued only if the user has previously initialized FLGERR, which causes a normal return when bad data is encountered.

FXEM Error #42 - illegal heading card.

FVFI continues as for end of data.

FXEM Error # 43 - illegal variable name.

FVFI continues as for end of data.

FXEM Error # 44 - illegal subscript or array size exceeded.

FVFI continues as for end of data.

FXEM Error #45 - illegal character after right parenthesis.

Data scan assumes comma between right parenthesis and next character.

FXEM Error #46 - illegal character in data.

Data scan treats illegal character as zero.

FXEM Error #48 - illegal logical constant.

Data scan treats illegal constant as .FALSE.

FXEM Error #52 - illegal Hollerith field.

FVFI continues as for end of data.

IV. RESTRICTIONS

The subprograms FIDO, FEOF, FOPEN, FXEM, and GET must be in memory.

FVFO--NAMELIST AND DEBUG OUTPUT

I. PURPOSE

To process NAMELIST, DEBUG, DUMP, and PDUMP output, using standard General File and Record Control (GEFRC) variable-length records.

II. METHOD

FVFO scans a NAMELIST table, and prints the current value of each NAMELIST variable in the format specified by its entry in the NAMELIST table.

III. USAGE

1. Calling Sequence - CALL .FVFO.(ARG,LF) for NAMELIST

output, *WRITE (i, namelist name) list*

CALL .FVDO.(ARG,LF) for DEBUG, DUMP,

PDUMP output,

where ARG = location of ARG XX

XX = location of DEC n (logical unit)

LF = location of NAMELIST table.

2. FVFO uses 510 words.

3. No error conditions.

IV. RESTRICTIONS

The subprograms F1DO, FOPEN, FXEM, and WTREC must be in memory.

FXEM -- EXECUTION ERROR MONITOR

I. PURPOSE

1. To print a trace of subroutine calls.
2. To print execution error messages.
3. To terminate execution with a Q6 abort or do one of the following:

Continue with execution of the program

Transfer to an alternate error routine

4. To allow the user to determine if an error has been processed by
FXEM.

II. METHOD

1. Error linkage for tracing calls is generated by the General Macro Assembler (GMAP). Tracing stops when the address of the CALL instruction in the error linkage word is zero, or when the number of traces exceeds a given constant.

The error trace prints in reverse order. It includes the name of each calling routine, identifying number of the CALL instruction, absolute location of the CALL instruction, and the calling arguments (up to 5).

2. The functions of this routine are optional. The options are controlled by the following switch word pairs:

. FXSW1--termination

. FXSW2--message printing

. FXSW3--alternate error returns

Each of the bits 1-71 in a switch word pair corresponds to an error code.

3. Special processing applies to error code 55. When this error is encountered, the following message is written:

ILLEGAL VALUE FOR COMPUTED GO TO AT ID NUMBER XXXXX

4. The error code is always stored in the location FXCODE in FXEM.
Since this is a SYMDEF, it may be accessed by a FORTRAN or GMAP program.
5. The error code is also stored indirectly through a pointer defined in FXEM. This pointer may be set by calls to ANYERR. If this pointer has been initialized to contain the address of a variable in the user's program via a call to ANYERR, the variable will also contain the error code, expressed as an integer, upon return to the calling subprogram.
6. FXOPT is an entry to .FXEM. which, for a given error code, sets the corresponding bits in .FXSW1, .FXSW2, and .FXSW3 to the low-order bit of the second, third, and fourth arguments. (See "Usage," item 5.) The first argument is the error code. When a call is made to .FXEM., the error code is used to shift each switch word pair and set the options accordingly.
7. FXALT stores the location of its argument in location FXALT1 in .FXEM. (See "Usage," item 6.) If the alternate error return option is used, index register 1 and the indicator register are restored; and a transfer is made to FXALT1 indirect. Thus, if the alternate return is a subprogram, the RETURN statement transfers to the location following the call to .FXEM. If no alternate has been supplied, a Q5 abort occurs.

8. A divide check, an overflow, or an underflow transfers to .FXEM. via the program fault vector. (For a description of the fault vector, see GE-625/635 Comprehensive Operating Supervisor (GECOS-III), CPB-1518, pp. 151-53, or for GECOS-II see GE-625/635 Comprehensive Operating Supervisor, CPB-1195, pp. 131-33. See also "SETUP," p. 42 in this manual.) .FXEM. writes the error message and loads the proper values into the EAQ-registers. The normal return is RET 6 (divide check) or RET 8 (overflow and underflow). If an alternate return is requested, the indicators and index register 1 are loaded from the fault vector, so that a RETURN statement in the alternate routine will transfer to the location immediately following the one that generated the fault.
9. FXEM is the entry provided for error conditions detected by the user's program. Error codes 61-66 are reserved for users. The statement

```
CALL FXEM (NCODE, MSG, N)
```

prints an error trace and N words of the message in the array MSG. MSG must be an array containing Hollerith information. If either MSG or N is omitted or is zero, no message is printed. If N is greater than 20 words, only 20 words are printed.

III. USAGE

1. Calling Sequence - CALL . FXEM. (X, Y)

Where X = Address of error description controls

Y = Address of tally word which is used to indicate card column found in error. Y is optional and is used only if a card image is to be printed.

2. Instruction Sequence at X

X ZERO A, B

ZERO C, D

ZERO E, F

A = Address of card image to be printed (or zero when card image is not to be printed).

B = Error code expressed as an integer (n) in the range $1 \leq n \leq 71$

C = Address of message 1.

D = Word count of message 1.

E = Address of message 2.

F = Word count of message 2.

3. CALL ANYERR (V)

— sets switch word to allow programmer to regain control instead of terminating execution with an error code

The statement CALL ANYERR (V)--where V is a variable into which the ^{integer} error code is to be stored for the user--sets the pointer in FXEM and initializes V at zero. The value of V changes to contain the error code of detected errors if FXEM is called.

*Repetitive calls to ANYERR will redefine V and reset it to zero.
(The error code is also stored in FXCODE, a symbol in FXEM)*

4. Switch Word Pairs

. FXSW1 (termination)--Figure 1 shows the standard bit settings and the names of the routines that use the corresponding error codes. The meaning of the bit settings is as follows:

- 1 Continue execution
- 0 Terminate with a Q6 abort

Termination may be overridden by the corresponding bit of . FXSW3 (see below).

. FXSW2 (message printing and trace)--The meaning of the bit settings is as follows:

- 1 Suppress printing
- 0 Print

This pair is initialized to zero. Settings may be changed by program call to FXOPT.

. FXSW3 (alternate error return)--The meaning of the bit settings is as follows:

- 1 Use alternate error return (overrides termination option set in . FXSW1)
- 0 Use normal return

This pair is initialized to zero. Settings may be changed by program call to FXOPT.

5. CALL FXOPT (NCODE, I1, I2, I3)

FXOPT is an entry to . FXEM which may be called to alter the standard switch word settings. In the statement CALL FXOPT (NCODE, I1, I2, I3), NCODE is an error code; and I1, I2, and I3 provide the settings for the corresponding bits in the three switch word pairs.

Examples:

- 1. CALL FXOPT (32, 0, 1, 0)
- 2. CALL FXOPT (32, 1, 0, 0)
- 3. CALL FXOPT (32, 0, 0, 1)

FXOPT
FXOP2
FXOP3

Error code 32 denotes an illegal character in input data.

Example 1 causes a Q6 abort when the error occurs, and no message

Set switch word independent
LDAQ
STAR
calls .FXEM
272-bit switches
.FXSW m

| | | |
|--------------|-------------------------------------|-------------------------|
| | <i>set</i> | <i>reset</i> |
| <i>FXOPT</i> | <i>Set sw. to terminate program</i> | <i>continue program</i> |
| <i>FXOP2</i> | <i>suppress diagnostic</i> | <i>print diagnostic</i> |
| <i>FXOP3</i> | <i>optional return</i> | <i>normal return</i> |

.FXSW1 > (word pair)

| . FXSW1 | | | . FXSW1 + 1 | | |
|-----------------------------|-----------------------|---------|-----------------------------|-----------------------|---------|
| Bit Position/ Error Code | Routine Using Code | Setting | Bit Position/ Error Code | Routine Using Code | Setting |
| 1 | XP1 | 1 | 36 | (Not Used) | 0 |
| 2 | XP1 | 1 | 37 | EXIT | 0 |
| 3 | XP2 | 1 | 38 | . OPEN. | 0 |
| 4 | XP2 | 1 | 39 | (Not Used) | 0 |
| 5 | XP3 | 1 | 40 | FRWB | 1 |
| 6 | XP3 | 1 | 41 | (Not Used) | 0 |
| 7 | XP3 | 1 | 42 | FVFI | 1 |
| 8 | FXPF | 1 | 43 | FVFI | 1 |
| 9 | FLOG | 1 | 44 | FVFI | 1 |
| 10 | FLOG | 1 | 45 | FVFI | 1 |
| 11 | FATN | 1 | 46 | FVFI | 1 |
| 12 | FSCN | 1 | 47 | FBST | 0 |
| 13 | FSQR | 1 | 48 | FVFI | 1 |
| 14 | FDX1 | 1 | 49 | FBST | 0 |
| 15 | FDX1 | 1 | 50 | (Not Used) | 0 |
| 16 | FDX2 | 1 | 51 | SLITE | 1 |
| 17 | FDX2 | 1 | 52 | FVFI | 1 |
| 18 | FDX2 | 1 | 53 | SSWTCH | 1 |
| 19 | FDXP | 1 | 54 | EXIT | 0 |
| 20 | FDLG | 1 | 55 | FXEM | 0 |
| 21 | FDLG | 1 | 56 | EXIT | 0 |
| 22 | FDSQ | 1 | 57 | FRWD | 1 |
| 23 | FDSC | 1 | 58 | (Not Used) | 0 |
| 24 | FDAT | 1 | 59 | (Not Used) | 0 |
| 25 | FCAS | 1 | 60 | (Not Used) | 0 |
| 26 | FCXP | 1 | 61 | } User Codes | 0 |
| 27 | FCXP | 1 | 62 | | 0 |
| 28 | FCLG | 1 | 63 | | 0 |
| 29 | FCSC | 1 | 64 | | 0 |
| 30 | FCSC | 1 | 65 | | 0 |
| 31 | FRWD | 1 | 66 | | 0 |
| 32 | FRWD | 1 | 67 | EUNDER | 1 |
| 33 | (Not Used) | 0 | 68 | OVER | 1 |
| 34 | FEOF | 1 | 69 | EOVER | 1 |
| 35 | FEFI | 1 | 70 | FXDVCK | 1 |
| | | | 71 | FLDVCK | 1 |

Figure 1. . FXSW1, Switch Word Pair Controlling Termination Option

or trace is printed.

Example 2 causes execution to continue after message and trace are printed.

Example 3 indicates that return is to an alternate error routine after trace and message are printed, since the alternate return option takes precedence over termination.

6. CALL FXALT (SR)

FXALT is an entry to .FXEM which may be called to set the alternate error return location. The statement CALL FXALT (SR) communicates to FXEM the name, SR, of the alternate error routine. An EXTERNAL SR must be included in the calling routine. If the alternate return option for an error code is indicated but no call to FXALT has been made, a Q5 abort follows when the error occurs. A RETURN statement in the alternate routine continues execution at the instruction immediately following the one where the error occurred.

The statement CALL FXALT (\$N) designates statement N in the calling program as the alternate error return.

Note: If the same error also occurs in the alternate error routine, an interminable loop results.

7. Overflows and Divide Check

The fault processor processes divide check, overflow, exponent overflow, and exponent underflow faults. A message is output on file 06 stating the type of fault and the memory location at which the fault occurred. Execution continues in the normal manner, although the EAQ-registers may have been reset as depicted in the following table.

| FAULT | EAQ-REGISTERS |
|--------------------|------------------------------|
| Exponent overflow | Largest floating-point value |
| Divide check (FP) | Largest floating-point value |
| Exponent underflow | Floating-point zero |
| Overflow (1) | No change |
| Divide check (1) | No change |

To have another value returned in the EAQ-registers after a divide check, CALL FXDVCK (R,M) should be executed prior to the occurrence of the fault. The statement causes the value of R to be returned in the EAQ-registers after a real divide check and the value of M to be returned in the Q-register after an integer divide check. The first argument must be double precision. The second argument may be omitted.

8. CALL FXEM (NCODE, MSG, N)

A FORTRAN-callable entry has been added to FXEM so that it may be called when the program detects an error condition.

The statement CALL FXEM (NCODE, MSG, N) causes the printing of (1) an error trace and (2) the Hollerith message contained on the MSG array. The number of words N to be printed must be within the limits $0 < N \leq 20$. If only the first argument is given, only the trace is printed.

9. User Error Codes

Error codes 61 - 66 are reserved for programmer usage.

IV. RESTRICTIONS

1. When the error code is greater than the maximum value specified for error codes, the message ERROR CODE XX GREATER THAN MAX is written and execution is terminated.
2. When Y = Address of Tally Word, the address portion of Y-1 must contain the word count of the card image.
3. The subprograms FOPEN, WTREC, and EXIT must be in memory.

LINK--RESTORE LINKS DURING EXECUTION

I. PURPOSE

To restore the Link specified in the calling sequence to the exact position it had at load time. The procedure to follow after restoring the Link depends on the entry used:

LINK - Restore the Link and transfer to its entry point as specified at load time.

LLINK - Restore the Link and return to the statement or instruction following the CALL to this subroutine.

If DEBUG is requested at load time in any or all of the Links, these subroutines join together the respective Debug Tables enabling the user to take snap dumps of any links in memory at the time of his request.

II. METHOD

LINK assumes the General Loader has generated a file (file code H*) containing the user's program segmented into Links as specified by \$ LINK control cards.

Both entries to this subroutine use the GERSTR function of the General Comprehension Operating Supervisor (GECOS). After restoring a Link, tests are made to determine if the Link contained a Debug Table. If so, the address of this table is chained to existing tables. If none exists in memory at this time, the address is placed in the DRL cell (cell 13 of the Fault Vector). Debug Tables, corresponding to Links which are overlaid in the process, are deleted from the chain.

III. USAGE

1. Calling Sequence - CALL LINK (LINKID)
CALL LLINK (LINKID)

where LINKID is the location of the Link Identifier specified as a literal.

2. LINK uses 70 memory locations.
3. No error conditions.

IV. RESTRICTIONS

An H* file generated by GESAVE must be present containing the Links.

STORE--ACCESS HALF OF A DOUBLE PRECISION OR COMPLEX WORD

I. PURPOSE

To provide a method of accessing only half of a double-precision or complex number.

II. METHOD

A call is generated by SIFT for PART or STORE if a variable or element of an array was referenced in FORTRAN II without an I or D in column one. At execution time, PART will return with the address of the proper half and STORE will store the desired value in the desired part of the double precision or complex number.

III. Usage

1. Calling Sequence - $X = \text{PART}(Y, Y(I), 10)$

$\text{CALL STORE}(Y, Y(I), 10, T)$

The first three parameters are the same.

First parameter = array name

Second parameter = array element, desired part of
double precision or complex

How?

Third parameter = array size

The fourth parameter for STORE is the value to be stored in the second parameter.

2. STORE uses 30 memory locations.
3. No error conditions.

IV. Restrictions

None

SETBUF--DEFINE A BUFFER(S) FOR A SPECIFIED FILE CONTROL BLOCK

I. PURPOSE

To allow the user to assign space in memory for use as an input/output buffer.

II. METHOD

Subprogram SETBUF searches the logical file table for the specified file and its associated file control block. It then attaches the buffers defined to the file control block. No check is made to verify that buffers are of sufficient size; this is the users responsibility.

III. USAGE

1. Calling Sequence - CALL SETBUF (I,A)
 or CALL SETBUF (I,A,B)

where I is the logical file designator

 A is the location of the first buffer

 B is the location of the second buffer if necessary

2. SETBUF uses 28 memory locations.

IV. RESTRICTIONS

The subprogram FOPEN must be in memory.

The size of the total buffer must be one location greater than the area to be used for actual storage of records.

Therefore, a standard size buffer is 321 words long.

SETFCB--DEFINE FILE CONTROL BLOCK

I. PURPOSE

To allow the user to define a file control block for use by the FORTRAN I/O library subprograms.

II. METHOD

The subprogram SETFCB searches the previously defined logical file table for an open space to insert the reference to the file control block. It accepts the file control block address and appends character positions 3, 4, 5 to the various logical file codes referring to this file control block. It makes as many entries as necessary in the "logical file - file control block" table.

III. USAGE

1. Calling Sequence - CALL SETFCB (A, I, J, ...)

where

A is the location of LOCSYM in the user created file control block.

I, J, ... are the logical files that refer to this file control block.

2. SETFCB uses 62 memory locations.

3. The error conditions are:

Abort code Q2 if no logical file table exists.

Abort code Q1 if there is no space available in the logical file table for inserting desired file control block.

IV. RESTRICTIONS

None.

SETLGT--DEFINE LOGICAL FILE TABLE

I. PURPOSE

To allow the user to define a logical unit table for use by the FORTRAN I/O library subprograms.

II. METHOD

The subprogram SETLGT accepts the array specified by the user as the logical unit table. It changes its first location to be a pointer to the last usable position of the array and places the address of the array +1 in fault vector location 25₈.

III. USAGE

1. Calling Sequence - CALL SETLGT (A,I)

where

A is the location of the logical unit table to be used.

I is the number of cells in the table A.

2. SETLGT uses 14 memory locations.

IV. RESTRICTIONS

SETLGT must be called before any Input/Output is requested. SETLGT is called only when the user wishes to suppress the logical file table generated by GELOAD and place the table in his own portion of memory. The user should use the NOFCB option on the \$OPTION GELOAD control card.

SETUP--PRE-EXECUTION INITIALIZER

I. PURPOSE

To perform installation standard procedures prior to execution of the user program.

II. METHOD

SETUP clears unused memory, sets it to either the constant specified in the \$OPTION GELOAD control card or to zeros if the constant is not defined. SETUP places the address of the "logical file - file control block" table in fault vector location 25_g. In a link job with debug requested in link 0, it places a transfer to the debug table in fault vector location 15_g. It places the entry point and a bit indicating a low-load job in fault vector location 24_g. It then calls the subprogram .FLTTPR to initialize for fault processing. Currently there is a secondary SYMDEF .FLTTPR imbedded in subprogram FXEM to satisfy this SYMREF in a FORTRAN execution. This routine places transfers to fault processing routines in fault vector locations 7_g and 11_g. These are also imbedded in subprogram FXEM. There also exists on the library a separate subprogram with SYMDEF .FLTTPR which places returns in fault vector locations 7_g and 11_g to ignore faults. This subprogram will be used if the job uses no FORTRAN library subprograms. Either .FLTTPR returns to SETUP which zeros out

all index registers and performs a TSX1 to the real entry point of the user's program. The SETUP subprogram is written in such a way that it can be easily changed by an installation to perform their own desired fault processing, accounting techniques, etc.

III. USAGE

1. SETUP is entered by GELOAD only; the entry point is defined as .SETU. . GELOAD assumes a five position storage block from cell .SETU.-1 to cell .SETU.-5 inclusive.

| <u>Cell</u> | <u>Definition</u> |
|-------------|---|
| .SETU.-5 | upper half contains lowest address of memory used by program and LABELED COMMON region lower half contains highest address of memory used in BLANK COMMON region |
| .SETU.-4 | logical unit table pointer in address field |
| .SETU.-3 | upper half = lowest cell used by program lower half \neq 0 = address of pointer to debug subroutine in link 0 |
| .SETU.-2 | memory reset constant |
| .SETU.-1 | upper half = entry point address lower half \neq 0 indicates low-load job |

2. SETUP uses 40 memory locations.
3. No error conditions.

IV. RESTRICTIONS

If the user requires initialization of the cells specified in SETUP, he must either use this subprogram or supply his own subprogram to perform the initialization.

FSLEW--CARRIAGE CONTROL SIMULATOR

I. PURPOSE

To format FORTRAN-generated print lines for the General Electric carriage-positioning-after-printing printer (PRT201).

II. METHOD

FSLEW is called by the FORTRAN I/O routine FRWD, BCD I/O Interface by Format Control (see p. 23). Control is passed to FSLEW after each print line has been prepared according to the format specification.

Recognized carriage control characters are 0, 1, +, and \backslash . FSLEW looks to see if the first character of the prepared print line is one of these characters.

If the first character is not a recognized carriage control character, FSLEW assumes the normal case: single space carriage positioning (\backslash). It therefore appends one word of single space slew information to the current print line.

If the first character of the prepared print line is a recognized carriage control character, FSLEW proceeds as follows:

␣ - Single Space - This is the normal case, as explained above.

The procedure is the same.

+ - Space Suppress - The single space slew information that was appended to the previous print line is replaced by space suppress slew information.

Note: For space suppression control, two conditions are assumed: (1) that the last record written on the current file is the line on which overprinting is desired and (2) that this line is currently in the buffer (so that its slew information can be changed). Both conditions must be satisfied for proper operation.

1 - Eject Before Printing - A 1-word print line (that is, a 1-word record) consisting of slew-to-top-of-page information is generated. This causes a slew to top of page to follow immediately after the single space resulting from the information appended to the previous line.

0 - Double Space - A 1-word print line (that is, a 1-word record) consisting of single space slew information is generated. This information and the single space information appended to the previous print line result in a double space operation.

Besides taking the actions described above for the recognized carriage control characters, FSLEW sets any such character to a blank (␣), if it is not already a blank.

The NOSLEW option on the \$ FFILE card causes bit 23 of FCB word -6 to be set to 1.

FSLEW recognizes this option and changes the output files written by FSLEW as follows:

1. The addition of a slew word at the end of a data record is inhibited.
2. The generation of 1-word print lines containing only slew information is inhibited.
3. The substitution of a blank character for the carriage control character (first character of data record) is inhibited.
4. A media code of zero is stored in the record control word in place of media code 3.
5. The "blank line" records generated by consecutive slashes in FORMAT statements are represented by 1-word records consisting only of blanks. (In the absence of the NOSLEW option, consecutive slashes in FORMAT statements cause 1-word records containing slew characters for single-line slews to be generated.)

III. USAGE

1. Calling Sequence - CALL . FSLEW (PL) where PL is the location of the print line. It is assumed that location . FBAD. in subprogram FOPEN contains the address of the file control block for the output file, and that word +1 of the file control block contains the size of the print line.
2. FSLEW uses 66 memory locations.

IV. RESTRICTIONS

The subprograms WTREC, FXEM, and FOPEN must be in memory.

LHSF--RESTORE LINK - H*

I. PURPOSE

To reload a program from an H* file (tape) which was generated in a previous GELOAD activity. The H* file was generated by having a \$ TAPE H* control card (see GE-625/635 General Loader, CPB-1008) at execution time.

II. METHOD

The H* file generated by GELOAD contains a link identified as /////// which is the main or common subprogram of the job. If the FCB option was in effect during loading (generation of H* file), a second link identified as ///////1 containing all file control blocks generated by GELOAD will also be present on H*. This subroutine searches the H* file for these identifiers (/////1 is optional), restores them, and enters the main subprogram at the entry location specified during the GELOAD activity.

III. USAGE

1. This program is called directly from the subroutine library and requires no other subprograms.
2. The entire job could be set up as follows:

```
$ SNUMB
$ IDENT
$ USE      .LHSF
$ ENTRY    .LHSF
$ EXECUTE
$ LIMITS
$ TAPE     H*,....
$ DATA    AB   (Optional)
. . .
. . .
$ ENDJOB
***EOF
```

If the NOFCB option was in effect when GELOAD generated the H* file, an entry card of the form:

```
$ ENTRY    .LHSNF
```

must replace the card following the \$ USE card.

IV. RESTRICTIONS

1. A \$ LOWLOAD card (see CPB-1008) must be included if the H* file was generated under this option.
2. The same memory limits must be requested as were in effect when the H* file was generated.
3. One of the setup subroutines must have been used when the H* file was generated. Entry to the main link is made through those subroutines for purposes of initialization of fault vectors.

FINC - BCD INTERNAL CONVERSION INTERFACE

I. PURPOSE

To provide the interface with .BDCNV or .DBCNV for the internal conversion of an array.

II. METHOD

.FINC sets up and executes the required calling sequences.

III. USAGE

1. Calling sequences

a) Binary-to-decimal conversion

CALL FOCNVS(LIST,N,BUF,FORM,WDA,LINES)

for an array of single precision numbers

CALL FOCNVD(LIST,N,BUF,FORM,WDA,LINES)

for an array of double precision numbers

where:

| | |
|-------|---|
| LIST | location of the array to be converted |
| N | number of words in the array |
| BUF | location of a buffer in which to store the resulting BCD record |
| FORM | location of the format controlling conversion |
| WDA | location of an array in which to store the number of words in each of the resulting lines |
| LINES | location in which to store the number of lines. |

b) Decimal-to-binary conversion

CALL FICNVS(LIST,N,BUF,FORM,WDA,LINES)

for an array of single precision items

CALL FICNVD(LIST,N,BUF,FORM,WDA,LINES)

for an array of double precision items

where:

LIST location of an array in which to store the binary numbers

N number of fields to be converted

BUF location of the input array containing the BCD record

FORM location of the format controlling conversion

WDA location of an array containing the number of words in each line of the BCD record.

LINES location containing the number of lines in the BCD record.

2. FINC uses 73 memory locations

3. No error conditions

IV. RESTRICTIONS

The subprogram FRWD must be in memory.



Progress Is Our Most Important Product

GENERAL  ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA