William R. Ogden

CCU-03
January 20, 1966
Reprinted: July 12, 1966

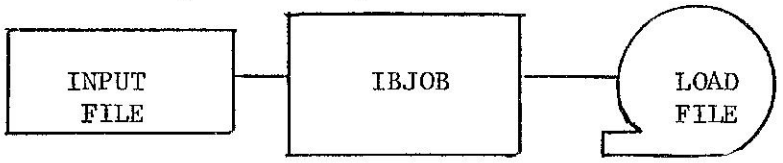
NOTES FOR 7040 FORTRAN USERS

FORTRAN is a problem-oriented algebraic language similar to ALGOL. FORTRAN IV is available at CIT on an IBM 7040 system. This system is primarily used for sponsored research; the turnaround for unsponsored work is unpredictable and often quite long. For this reason it is recommended that, at present, FORTRAN not be used for new programs.

TABLE OF CONTENTS

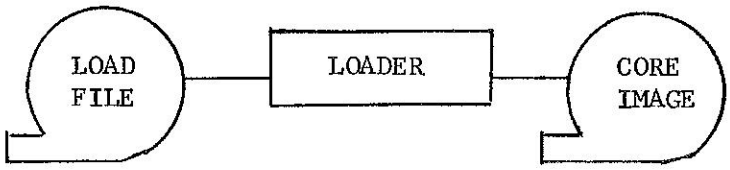
1. A SHORT INTRODUCTION TO THE OPERATING SYSTEM

The 7040 runs under the control of the IBSYS monitor. The most important part of this monitor is the IBJOB processor. The IBJOB can be pictured as having two main phases:

1. 

(TAPE UNIT)

IBJOB reads cards from the system input file (Card reader on the Carnegie Tech System). FORTRAN or MAP programs are translated into binary form and placed on a load file. Binary decks are stacked directly onto the load file. IBJOB continues reading input decks and stacking them on the load file (after translation if necessary) until a $ENTRY card is found.

2. 

The $ENTRY card causes the loader, another part of IBJOB, to take control. The loader reads the load file, relocates the decks on it, and takes care of any cross references between decks.

For example, if one deck contains the statement:

CALL SUMSQ

Where SUMSQ is another program (subroutine), the loader will look for the deck SUMSQ and inform the calling program of its exact location in memory. If the necessary subroutine cannot be found among the decks on the load file, the loader will look for it in the system library.

Note that the input decks have priority over the subroutine library. The user can write routines having the same name as library routines (e.g., SIN, COS, SQRT); these will then be used instead of the library programs.

If the loader cannot find a necessary subroutine in either the load file or the library, it will print an error message such as:

UNDEFINED CONTROL SECTION name

and terminate the job.

Note that the loader does not know nor care whether the program originated as FORTRAN, MAP, or a binary deck. As the loader processes the input decks they are combined into one program and placed on a core image file. This file is then read into memory and program execution begins.

A typical input deck might be:

```
Columns:  1       8       16
          $JOB            SAMPLE PROGRAM NR 1
          $ID                   010          CC01W002  NAME, DEPT
          $IBJOB          NODECK
          $IBFTC MAIN
                            .
                            .
                            .    (FORTRAN program)
                            .
                        END
          $IBFTC SUB1
                            .
                            .
                            .    (FORTRAN subroutine)
                            .
                        END
                .
                .
                .       (binary deck for another subroutine.  The binary
                .        deck contains all necessary control cards)
                .
                .
          $ENTRY MAIN
                    .
                    .   (data cards (if any))
                    .
                    .
          $IBSYS
```

The $JOB card must be the first card of any job. It clears I/O buffers, resets switches, and generally resets the system for the next job. Any comments in columns 16-45 will appear at the top of each page of printed output during the job.

The $ID card must follow the $JOB card; it calls the CIT accounting routine. Columns 21-23 contains the maximum time in minutes for the job. Columns 33-40 have the same usage number, and 42-65 have the user's name, department, etc. These card columns are the same as those for a G-20 job card.

The $IBJOB card is next; it calls the IBJOB processor described above. Starting in column 16 a number of options may be punched. These may be given in any order and are separated with commas; no blanks are permitted between options. Four of the options are of interest to the FORTRAN user:

DECK or NODECK	This controls the punching of a binary deck for any source language decks in this run. The system assumes DECK unless told differently.
GO or NOGO	This determines whether the program will be executed after any necessary compilation. The system assumes GO.
SOURCE or NOSOURCE	If all programs for this job are in the form of binary decks, the NOSOURCE option may be used to speed up the loading process. The system assumes SOURCE.
NOMAP or MAP	The MAP option will produce a directory giving the location of programs in memory. This is useful for debugging purposes. The system assumes NOMAP.

The $JOB, $ID, and $IBJOB cards are necessary for all jobs.

A $IBFTC card may contain a 6-character program name starting in column 8. If the FORTRAN deck is a subroutine it must contain the subroutine name in column 8.

The main program is usually the first (and, if there are no subroutines, the only) deck. If there are subroutines, they are placed after the main deck. A FORTRAN subroutine must be preceded by a $IBFTC card. A subroutine which has already been punched into a binary deck includes all the necessary control cards as part of the binary deck. Such a deck is placed with the other subroutines. The order of the subroutines is not important.

The $ENTRY card signals the system that all programs and subroutines have been read and that the program should now be executed (unless the $IBJOB card has a NOGO option). Column 16 may contain a deck name. If it does, the deck named will be given control when execution begins. If column 16 is blank, the first deck in the job will be given control. This is why the main program is usually placed first. A $ENTRY card is necessary if the job is

to be executed.

Data cards (if any) follow the $ENTRY card. No blank cards are needed unless the program uses them.

The $IBSYS card indicates the end of the data and returns control to the system monitor.

Several examples of complete decks follow:

```
$JOB              EXAMPLE ONE
$ID                     010          G495AA01
$IBJOB            NOSOURCE
                  .
                  .   (binary deck from previous compilation)
                  .
$ENTRY
$IBSYS               (the program does not need data)

      * * * * * * * * * * * * * * * * * * * *

$JOB              EXAMPLE TWO
$ID                     007          S205BB99     JONES,    EE
$IBJOB            NODECK
$IBFTC PGR2
                  DIMENSION A(20)
                  .
                  .        (FORTRAN program)
                  .
                  END
$ENTRY
                  .
                  .          DATA
                  .
$IBSYS * * * * * * * * * * * * * * * * * * * *

$JOB              EXAMPLE THREE
$ID                     015          S206LJ01     CHEM
$IBJOB
                  .
                  .        (binary deck)
                  .
$IBFTC LSQ
                  .
                  .        (FORTRAN Language subroutine)
                  .
$IBMAP ACTNH
                  .
                  .         (MAP language subroutine)
                  .
                  END
      .
      .           (binary deck obtained from previous compilation)
      .
$ENTRY
$IBSYS
```

2. COMPILATION TIMING

The FORTRAN compiler is rather slow. This is due in part to its complexity and in part to the very slow tape units used with the 7040. A very rough estimate might be 1.5 minutes for one (or each) 75 card program. Even for very short programs the compilation and loading time will be at least 1 minute. For this reason the estimated maximum time stated on the $ID card should make allowances for compilation as well as execution time. At present the $ID card time is total time on the machine and extra time (for example, 3 minutes) should be allowed for the operator to correct machine problems, change tapes, etc.

3. FORTRAN IV INPUT/OUTPUT

FORTRAN IV uses input/output statements of the form:

```
WRITE (N, FMT) list     or
READ (N,  FMT) list
```

where N is a unit number and FMT is the statement number of a format statement. The unit assignments are as follows:

0, 1, 2, 3, 4 are tape units
5 is the system input unit (card reader)
6 is the system output unit (printer)
7 is the system punch unit (card punch)

More detail on unit assignments is shown in Table 2 at the end of this paper.

Example:

```
      READ (5. 37) A,B,C
   37 FORMAT (1X,2F10.0,E16.5)
```

This would read a card with values for A, B, and C.

The card punch is available to users. It is rather slow (250 cards/minute) and its use should be avoided if possible. The printer operates at 600 lines per minute.

If it is necessary to use tape units for intermediate storage during a program, it is more efficient to use a binary mode I/O statement. This is done by writing, for example,

```
WRITE (3) A,B,C
```

and later,

```
READ  (3) A,B,C
```

where the 3 refers to tape unit 3 (see unit assignments on preceding page). This type of statement causes the data to be written and read in 7040 internal form. Since it represents only temporary storage, there is no need to transform it to the neat form used when a format statement is specified. This type of statement should never be used with the reader, printer or punch (units 5, 6, or 7).

If a program writes a large number of tape records (say > 100), it is often possible to greatly decrease running time by redefining the FORTRAN files with a $FILE card. See the user consultant for details.

4. OFF LINE OUTPUT

Sometimes the computer operators switch the output to tape units for later printing and punching by a smaller computer (IBM 1401). This is called off line output. The system automatically takes care of switching unit assignments. That is, if a program says

```
WRITE (6,37) A,B,C
```

the system would write on tape rather than on the printer. This tape would be listed later.

This off line output does not affect the user (indeed, he is not aware of it) except for the time involved. The 7040 can write tape much faster than it can print or punch. Thus a program which does only a little computing and much printing will run faster if the output is off line.

If a program is going to print or punch an unusually large amount of output, the user should place a note on his envelope, advising the operator to use off line output, if possible. A "large amount" is hard to define; perhaps 2000 punched cards or 5000 printed lines would be a rough dividing line. The user consultant can help with this problem.

5. FORTRAN ERROR MESSAGES

FORTRAN makes use of the IBMAP assembler and error messages are produced by both FORTRAN and MAP. A sample listing is shown here:

```
ISN                SOURCE STATEMENT
 0 $IBFTC      ILLUS
 1             DIMENSION Y(10), X(10,10)
 2             DO 1 I=1,10
 3      1      X(I,J)=Y(I)+1
 5             IF (I .EQ. 1) A=A+1
10             IF (I .EQ. 1) A=A+I
13             READ(0) A, ((X(I,J),I=1,10,B,Y,J=1,10)
24             GO TO 8
25             STOP
26             END
```

the following error messages were produced:

```
2  STATEMENT 19  ERROR 1232  ISN-12   ILLEGAL MIXED MODE
0  STATEMENT 10  ERROR 1327  ISN-25   STATEMENT SHOULD HAVE A STATEMENT
                 NUMBER BECAUSE OF PRECEDING GO TO
2  STATEMENT 84  ERROR  26   8S   IS AN UNDEFINED SYMBOL
   HIGHEST SEVERITY WAS 2.  EXECUTION DELETED
```

The first number in each error message line is the severity code; 0 and 1 are warnings only, 2 or 4 cause the job to be terminated. Next appears STATEMENT XX; this is the statement number of the MAP translation and is of no interest to the FORTRAN user. ERROR XXXX identifies the error message; this can be ignored. If the sequence ISN-XX appears, XX refers to the column of Internal Sequence Numbers to the left of the program listing. The gaps in these numbers are caused by internal expansion of the program. The ISN-XX number identifies the statement in error and the err message which follows is usually quite helpful.

Some error message are from the MAP assembler and do not give an ISN number. Often the MAP error messages are not very helpful, but the most common one, given above, is useful. When FORTRAN statement numbers are translated into MAP the letter S is added to each statement number.

Thus the missing symbol 8S means that statement number 8 is missing. Looking over the program we see a GO TO 8, but there is no statement 8. Usually errors referred to by MAP error messages can be cleared up by correcting all errors found by the FORTRAN compiler (i.e., those with ISN numbers).

EXECUTION ERRORS

Errors during execution take two forms: 1. library subroutine errors, or 2. machine errors.

1. The library subroutines produce such messages as:

SQRT (-X) INVALID ARGUMENT

-B**C WHERE C IS REAL

ATAN1(0,0) INVALID ARGUMENTS

etc

After the message is printed, the subroutine returns a conventional answer and execution continues. For example, the square root routine uses the absolute value, etc.

Sometimes a floating point trap occurs. This is caused by the computation of a real number too large ($>10^{38}$) or too small ($<10^{-38}$) to be stored in the machine. (The number zero is an exception, and does not cause such a trap). The word trap means that the machine hardware recognizes an error condition and interrupts the program to execute a special routine. If the error was an underflow (number $<10^{-38}$) the floating point trap routine, FPT, will make the answer zero and allow the program to continue. If the error was an overflow, FPT will print the message OVERFLOW and replace the number with the maximum allowed number. After a set number (8) of overflows, the monitor will terminate the program.

2. Machine errors. These are errors that are non-computational in nature and are detected by the machine hardware, rather than by a system subroutine. Two of these are very common:

ILLEGAL INSTRUCTION. This causes the program to be terminated with the above message. This error is caused by the program attempting to execute something that is not a valid instruction; e.g. data.

MEMORY PROTECT VIOLATION. This also causes the program to be terminated with a message. The section of the memory which contains the monitor is protected from changes by the user's program, which will not reference this part of memory unless something goes wrong.

Both of these errors are almost always caused by array subscripts going out of range. FORTRAN makes no check on subscripts before using them. For example:

```
DIMENSION  (A(50)
    :
I=80
    :
A(I)=12.7
    :
```

the subscript I is out of range. This will probably cause another part of the program to be altered, and possibly cause an instruction trap (i.e., interruption because of an illegal instruction). The only other common source of these errors is an incorrect index in a computed GO TO statement; e.g.,

```
GO TO (5,7,15,8),J
```

if J is other than 1, 2, 3, or 4 it is incorrect and will probably cause an error.

FORTRAN sometimes produces messages such as:

FORTRAN HAS IGNORED ERROR n AND TAKEN THE OPTIONAL RETURN TO EXECUTION

These error codes (n) are defined in Table 1 at the end of this paper.

6. FORTRAN IV provides the following library subroutines:

Y=EXP(X)	$Y= e^X$	where X and Y are real
Y=SQRT(X)	$Y= \sqrt{X}$	
Y=ALOG(X)	$Y=\log_e X$	
Y=ALOG10 (X)	$Y=\log_{10} X$	
Y=SIN(X)		
Y=COS(X)		
Y=TAN(X)	trig functions have arguments in radians	
Y=COTAN(X)		
Y=ATAN(X)	Y=Arctan (X)	Y in radians
Y=ATAN2(X,Z)	Y=Arctan (X/Z)	
Y=ARSIN(X)	Y=Arcsin (X)	
Y=ARCOS(X)	Y=Arcos(X)	
Y=SINH(X)		
Y=COSH(X)		
Y=TANH(X)		
Y=ERF(X)	Y=Error function (X)	
Y=GAMMA(X)	Y=Gamma (X)	
Y=ALGAMMA(X)	Y=log Gamma (X)	

The above routines produce an accuracy of 8 significant digits. A set of double precision routines are also provided for the above functions.

The following complex subroutines are provided:
X and Y here represent complex variables

Y=CSQRT(X)	$Y= \sqrt{X}$	
Y=CEXP(X)	$Y=e^X$	
Y=CLOG(X)	$Y=\log_e X$	
Y=CSIN(X)	Y=sin X	
Y=CCOS(X)	Y=cos X	
R=CABS(X)	$R=\lvert X\rvert$	where R is real

In general the above functions produce 8 significant digits for both real and imaginary parts.

Thirty-one built in functions are provided, including the following common ones:

X=ABS(Y)	absolute value of real Y
J=IABS(K)	absolute value of integer K
Y=AMAX1(X1,X2,X3,...XN)	Y=largest value of the real arguments
J=MAXO(K1,K2,.....KN)	J=largest value of the integer arguments

AMIN1 and MIN0 are included for minimum values

Y=REAL(Z)	Y= real part of complex Z
Z=CMPLX(X,Y)	Z complex = X +iY X and Y are real
Z=CONJG(C)	Z= conjugate of C Z and C are complex

7. USEFUL REFERENCES FOR FORTRAN ARE THE FOLLOWING:

FORTRAN IV	by McCracken; this is an excellent text for learning FORTRAN.
IBM	7040 Programmers' Guide; form C28-6318
IBM	7040 FORTRAN IV Language; form C28-6329
IBM	7040 Mathematical subroutines
CIT	User Manual

All of these are available in the bookstore.

The programmer's guide lists many control cards not mentioned above. The general FORTRAN user will not need these, and, in any event, the following should never be used without prior consultation with the computing center staff:

$ATTACH	$CBEND	$TIME
$DATE	$DETACH	$UNITS
$ENDEDIT	$IBCBC	$UNLIST
$IBEDT	$JEDIT	
$LABEL	$LIST	
$STOP	$SWITCH	

Table 1

FORTRAN ERROR RETURNS[1]

Note: Ω = largest number in the machine

ERROR CODE	FORTRAN EXPRESSION	ERROR CONDITION	OPTIONAL RETURN
1	Y=Z**R	Z = R = 0	Y = 0
2	Y=Z**R	Z = 0 and R < 0	Y = 0
3	Y=TAN(X)		
	Y=COTAN(X)	$\lvert X \rvert \geq 2^{20}$	Y = 0
4	Y=TAX(X)	$X = \frac{K\pi}{2}$, K is odd integer	$Y = -\Omega$
	Y=COTAN(X)	$X = K\pi$, K is any integer	$Y = -\Omega$
5	Y=SINH(X)		
	Y=COSH(X)	X > 88.029692	$Y = -\Omega$
7	Y=Z**R	Z < 0 and R $\neq$ 0	$Y = \lvert X \rvert^{R}$
8	Y=EXP(X)	X > 88.029692	$Y = \Omega$
9	Y=ATAN2(Z,R)	(Z,R) = (0,0)	Y = 0
10	Y=ALOG(X)		
	Y=ALOG10(X)	X = 0	$Y = -\Omega$
11		X < 0	$Y = \log\lvert X \rvert$
12	Y = SIN(X)		
	Y = COS(X)	$\lvert X \rvert \geq 2^{25}$	Y = 0
13	Y=ARSIM(X)		
	Y=ARCOS(X)	$\lvert X \rvert > 1$	Y = 0
14	Y=SQRT(X)	X < 0	$Y = \lvert X \rvert^{1/2}$

[1] For further information see:
IBM 7040/7044 Operating System (16/32K)
Subroutine Library
(FORTRAN IV Mathematical Subroutines)
Form C28-6806-1
Also IBM 7040/7044 Operating System (16/32K)
System Programmers Guide
Form C28-6339-3

ERROR CODE	FORTRAN EXPRESSION	ERROR CONDITION	OPTIONAL RETURN
	Let Complex X = R + iZ		
15	Y=CEXP(X)	$R > 88.029692$	$Y=\Omega(\cos Z+i \sin Z)$
16		$Z \geq 2^{25}$	Y = 0 + 0i
17	Y=CLOG(X)	X = 0 + 0i	$Y = -\Omega + 0i$
18	Y=CSIN(X)		
	Y|CCOS(X)	$\|Z\| > 88.029692$	see manual
19		$\|R\| \geq 2^{25}$	Y = 0 + 0i
			Y = -0 or -0i
20	Y=GAMMA(X)	$2^{-127} \geq X$ or	
	Y=ALGAMA(X)	$X \geq 34.843$	$Y = \Omega$
21		$X \leq 0$ or $X \geq 1.54926(2^{120})$	$Y = \Omega$
22	Y=DATANZ(RZ)	(RZ) = (0,0)	Y = 0
24	Y = DEXP(X)	$X > 88.029692$	$Y = \Omega$
25	Y = DLOG(X)		
	Y = DLOG10(X)	X = 0	$T = -\Omega$
26		$X < 0$	$Y = \log\|X\|$
27	Y = DSIN(X)		
	Y = DCOS(X)	$\| X \| > 2^{50} \pi$	Y = 0
28	Y = DSQRT(X)	$X < 0$	$Y = \|X\|^{1/2}$

ERROR CODE	ERROR CONDITION	OPTIONAL RETURN
31	Excessive floating point traps	To XIT
32	Variable unit not defined	To XIT
33	Attempt to backspace past beginning of file	To XIT
34	Attempt to write file back on system input, output, or punch unit	Ignore the operation
35	Attempt to rewind the system input output, or punch unit	Ignore the operation
36	Attempt to write on system input unit	To XIT
37	Attempt to read on system output file	To XIT

ERROR CODE	ERROR CONDITION	OPTIONAL RETURN
38	File mark reading	Read next file
39	Interval word count ≠ IOCS count	Process record read
40	Input list exceeds FORTRAN record length	Set remaining list items to zero
41	Output line has overflowed 1401 limit	Treat as end of format
42	Invalid input character	Treat character as zero
43	Invalid input character	Treat character as blank
44	Invalid input character	Treat character as zero
45	Invalid character in variable FORMAT	Treat character as end of format
46	I > 4	Take no action
47	I = 0, I > 4	Set J = 2
48	I = 0, I > 6	Set J = 2
49	Block sequence error	Process the record read
50	Check sum error	Process the record read
51	Block sequence and check sum errors	Process the record read
52	Permanent read redundancy	Process the record read
53	Attempt to write on unopened file	Return to IOBS for job termination
54	Buffer overflow	Write as much of the record as will fit in buffer
55	Error in IOBS Type 2 or 3 record control word	Set the word count to actual number of words remaining in buffer.
56	Unexpected mode change	Process the record that was read
59	Attempt to backspace the system output or punch file	Ignore the operation
60	Illegal GOTO	Take first branch

Table 2

IBM 7040 UNIT ASSIGNMENTS

SYMBOLIC UNIT	PHYSICAL UNIT	CHANNEL	UNIT NUMBER	FORTRAN UNIT
S.SLB1	T	C	8	
S.SLB2	T	C	9	
S.SIN1	RD	3	1	5
S.SOU1	PR	3	1	6
S.SPP1	PU	3	1	7
S.SCK1	T	C	2	
S.SU00	T	C	1	0
S.SU01	T	B	1	1
S.SU02	T	B	2	2
S.SU03	T	C	2	3
S.SU04	T	C	3	4
S.SU05	T	C	4	8
S.SU06	T	C	5	9
S.SU07	T	C	6	
S.SU08	T	C	7	
S.SU09	T	B	3	
S.SU10	T	B	4	10
S.SU11	T	B	5	11
S.SU12	T	B	6	12
S.SU13	T	B	7	

Note: Logical units 8-12 do not appear in the IBM manuals, but have been added to our system for the users' benefit.

T - TAPE

RD - CARD READER

PR - PRINTER

PU - PUNCH