SPERRY UNIVAC
COMPUTER SYSTEMS

ATTN: CHARLIE GIBBS

00681
CAV208M45541 UP 8815-A

SPERRY UNIVAC UAS
SUITE 906
1177 WEST HASTINGS ST
VANCOUVER BC V6E 2K3
CAV

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

FORTRAN IV

Programmer Reference

UP-8814-B

This Library Memo announces the release and availability of Updating Package B to "SPERRY UNIVAC Operating System/3 (OS/3) FORTRAN IV Programmer Reference", UP-8814-B.

This update incorporates minor changes and corrections for release 8.0 and describes the // PARAM ERRFIL parameter for using the error file processor with FORTRAN IV.

Copies of Updating Package B are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry Univac representative. To receive only the updating package, order UP-8814-B. To receive the complete manual, order UP-8814.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists B00, B15, 28U, and 29U (Package B to UP-8814, 33 pages plus Memo)	Library Memo for UP-8814-B
		RELEASE DATE: September, 1982

PAGE STATUS SUMMARY

ISSUE: Update B – UP-8814
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	B
Preface	1	Orig.
Contents	1 thru 4 5 6 thru 8	Orig. B Orig.
PART 1	Title Page	Orig.
1	1 2 thru 9	B Orig.
2	1 thru 7	Orig.
PART 2	Title Page	Orig.
3	1 thru 6	Orig.
4	1 thru 7	Orig.
5	1 thru 28	Orig.
6	1 thru 8	Orig.
7	1 thru 23 24, 25 26	Orig. B Orig.
8	1 thru 3	Orig.
PART 3	Title Page	Orig.
9	1 2 3 4 5	Orig. B Orig. B A
10	1 thru 4	Orig.
11	1, 2 3 thru 5 6 7 8 thru 10 11 12 thru 16 17 thru 19 20, 21 22	B Orig. B A Orig. A Orig. A B A
11 (cont)	23 thru 33	Orig.
12	1 thru 4	Orig.
PART 4	Title Page	Orig.
Appendix A	1 thru 8	Orig.
Appendix B	1 thru 5	Orig.
Appendix C	1 2 2a 3 thru 5 6 thru 16	A B A B Orig.
Appendix D	1 thru 15	Orig.
Appendix E	1 thru 12	Orig.
Appendix F	1 thru 6	Orig.
Index	1 2, 3 4 thru 9	Orig. B Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

PART 3. COMPILE, EXECUTE, AND DEBUG PROCEDURES

9. COMPILATION

10. DEBUGGING

11. CONSOLIDATED DATA MANAGEMENT (CDM) EXECUTION ENVIRONMENT CONFIGURATION

1. Introduction

1.1. SCOPE

This manual describes FORTRAN IV operating with SPERRY UNIVAC Operating System/3 (OS/3) and a consolidated data management (CDM) environment. The components of this operational environment include: ←

- a compiler that transforms programs written in an extended American National Standard FORTRAN language into a form suitable for execution;
- a library of input/output (I/O) and data formatting routines; and
- a library of commonly used mathematical functions and service routines.

The FORTRAN IV compiler accepts programs written in the FORTRAN language and produces an object module that is suitable input to the linkage editor. Source programs may reside in the control stream or in a source program library. A job control procedure is provided to call the compiler, allocate scratch files, etc.

The output of the compiler must be processed by the linkage editor; during this processing, mathematical and I/O routines are taken from the FORTRAN library and included in the executable program. User-defined procedures, if they are required, also are included during the link edit. These latter procedures may be coded in FORTRAN or in some other language, such as COBOL, assembly, etc.

The output of the linkage editor is a load module that may consist of several overlay phases. During the execution of the object program, the overlay phases may be loaded by specific calls by FORTRAN statements, or they may be loaded automatically by referencing a routine in an overlay that is not currently in main storage. The load module will accept and produce ASCII files.

During compilation, the compiler produces the following listings:

1. A listing of the source program. For each diagnostic, the source statement is marked at the character for which the diagnostic is produced.
2. An error listing that contains the diagnostic messages and associated severity codes. (See Appendix D.)
3. A main storage map showing the addresses allocated to the variables and arrays in the program. An alphabetical and address sorted listing is optionally available.

Any of these listings can be suppressed by user options.

The FORTRAN IV compiler is self-initializing, and up to 100 FORTRAN source programs may be processed by one call on the compiler by job control. If a FORTRAN source statement follows an END statement in the source input file, it is assumed that another program is to be processed, and the compiler reinitializes itself.

1.1.1. Compatibility

The FORTRAN IV language includes the American National Standard FORTRAN X3.9-1966 and the IBM System/360/370 DOS FORTRAN IV languages as subsets. Programs that conform to either of these specifications are accepted without change. FORTRAN IV is also highly compatible with SPERRY UNIVAC Series 70 FORTRAN.

1.1.2. Extensions

The FORTRAN IV language provides many extensions to *American National Standard FORTRAN, X3.9—1966.* These extensions are:

- Subscript expressions may be integer or real arithmetic expressions (2.4.1).
- Arithmetic assignment statements can be used to assign complex values to integer and real variables, or integer and real values to complex variables (3.3.1).
- A literal message is permitted with the STOP and PAUSE statements (4.9 and 4.10).
- An executable END statement is provided (4.11).
- The inclusion of statement labels (preceded by the & character) in the list of actual arguments in a subroutine call to be referenced by a RETURN statement is permitted. Thus, the subroutine can transfer control back to designated statements in the calling program (5.4.2.1).
- The ENTRY statement permits entry into a function or subroutine subprogram at points other than the beginning of the subprogram (5.4.3).
- Standard library routines are available: OVERFL, DVCHK, ERROR, ERROR1, SLITE, SLITET, SSWTCH, DUMP, PDUMP, EXIT, FETCH, LOAD, and OPSYS (5.6.3).
- Arrays may have a maximum of seven dimensions (6.2.1).
- Dimension declarator subscripts are permitted in common storage (6.2.1).
- Optional length specifications for logical, integer, complex, and real variables and arrays can be declared (6.4.1).
- An IMPLICIT statement is provided for user-defined implicit typing of symbolic names in a program unit (6.4.2).
- End-of-file and error recovery are provided in READ statements (7.3.1.1).
- The applicability of the G field descriptor is extended to cover integer and logical data (7.3.3.1.6).
- Z and T format codes are provided (7.3.3.1.9 and 7.3.3.1.12).
- Special I/O formats and statements are provided for direct access storage devices (7.4).

where:

u

Is a file identifier, an integer constant specifying a file, or a unit reference number.

r

Is an integer constant specifying the number of records in the file.

m

Is an integer constant specifying the maximum size of a record in the file in terms of characters (bytes), main storage locations (bytes), or main storage units (words), depending on the specification for x.

x

Is one of three possible code letters to indicate an option of format control:

L, to transfer either formatted or unformatted data, where the specification form determines the number of bytes;

E, to transfer formatted data, where the specification for m determines the number of bytes;

U, to transfer unformatted data, where the specification of m designates main storage units.

v

Is the associated variable for the file, which must be an unsubscripted integer*4 variable. After execution of a READ or WRITE statement, the variable is assigned a value in the range ($1 \leqslant v \leqslant r$) indicating the sequential position of the next record in the file; after execution of a FIND statement, it is assigned a value indicating the position of the desired record. It is not defined (i.e., set to a value) by the DEFINE FILE statement.

Description:

A DEFINE FILE statement is executable, and it dynamically describes one or more files that may be referenced during program execution. At the start of execution of a FORTRAN program, all direct access units are considered to be undefined, and no READ, WRITE, or FIND references are permitted. When a DEFINE FILE is executed, the characteristics of one or more units are registered with the FORTRAN system, and the units are made available for use. Thereafter, further definitions of previously defined units are ignored.

The associated variable v should not be passed indiscriminately between subprograms or used for purposes other than a file pointer, since the compiler has no syntactic clues as to its usage when the DEFINE FILE statement is absent in a subprogram. When an associated variable must be transmitted to a subprogram, it should be passed in COMMON storage or, less preferably, associated with a dummy argument called by name.

To calculate the record size in storage units (when using the u specification for parameter x): determine the total number of bytes required for all the items of the I/O list, and divide this by 4. If the quotient is not an integer, round it to the next highest integer. There is no restriction on the transmission of multiple records by FORMAT/list interaction, but unformatted lists cannot specify more than one disk record.

Example:

```
1      7
       DEFINE FILE 3(100,120,L,FILE3),
      15(98,80,U,FILE5)
```

File 3 is composed of 100 records, the maximum size of which is 120 bytes. L indicates that the record size is specified in bytes. If the I/O statement contains a reference to a format, 120 bytes of formatted data are transferred; if unformatted data is transferred, File 5 contains 98 records, each 80 bytes in length.

7.4.2. Disk READ Statement

Format:

```
READ (u'p,a,ERR=label1,END=label2)k
```

where:

u
: Is a file identifier represented by an integer constant or variable followed by an apostrophe.

p
: Is an integer expression designating the position of the record in the file, which should be in the range ($1 \leq p \leq r$), where r is the number of records in the file.

a
: Is an optional label of a FORMAT statement, an array name, or the character asterisk.

ERR=label$_1$
: Optionally specifies the label of a statement to which control is to be transferred when an error condition occurs.

END=label$_2$
: Optionally specifies the label of a statement to which control is to be transferred when an ENDFILE record is encountered, or when p is outside the file limits.

k
: Is an I/O list.

Example:

```
1    5 7
       INTEGER FILE3/1/
       DEFINE FILE 3(100,512,L,FILE3)
        .
        .
        .
       READ (3' FILE3,87,ERR=110) A,B,(C(I),I=1,30)
    87 FORMAT (32F16.4)
```

The first record in file 3 is transferred to main storage when the READ statement is first executed. Each subsequent execution of the READ statement transfers the next record in the file to main storage, unless the associated variable FILE3 is explicitly redefined. The descriptor 32F16.4 indicates that each unit of data consists of 16 bytes and 32 such units of data are to be transferred. Thus, the 512 bytes (16x32) of the record are transferred to main storage.

The slash in a FORMAT specification can control the starting point of data transfer in a file. If the FORMAT statement in the example were:

```
FORMAT(//32F16.4)
```

the first execution of the READ statement would transfer the third record in the file; the second execution would transfer the sixth record.

7.4.3. Disk WRITE Statement

Format:

```
WRITE (u'p,a) k
```

where:

u
: Is a file identifier represented by an integer constant or variable followed by an apostrophe.

p
: Is an integer expression designating the position of the record in the file.

a
: Is an optional FORMAT statement label, an array name, an integer variable to which the statement label

k
: of a FORMAT statement has been assigned, or the character asterisk.
Is an I/O list.

Example:

```
1    5 7
       DEFINE FILE 4(150,36,L,FILE4)
         .
         .
         .
       LOGICAL L
       DOUBLE PRECISION D
         .
         .
         .
       FILE4 = 2
         .
         .
         .
       WRITE (4' FILE4+1,2) I,R,D,L
     2 FORMAT (I8, F12.2, D15.5, L1)
```

Thirty-six bytes (8 + 12 + 15 + 1) are transferred from storage to the third record in the file. The format specification indicates the number of bytes for the integer, real, double precision, and logical values transferred. If the WRITE statement does not specify a format label, an unformatted WRITE statement is executed. In this case, 20 bytes are transferred.

Variable Name	Type	Number of Bytes
I	Integer	4
R	Real	4
D	Double precision	8
L	Logical	4
		20 Total

7.4.4. Disk FIND Statement

Format:

```
FIND (u'p)
```

where:

u
: Is a file identifier represented by an integer constant or variable and followed by an apostrophe.

p
: Is an integer expression designating the position of a record in the file.

Description:

The FIND statement can decrease the time required to execute an object program requiring records from disk. This statement positions the access arms to a disk address specified by a file identifier and a record position. During the time the arms are being positioned, execution of the object program can continue. After positioning, a READ statement accessing the record addressed in the FIND statement may be executed, and the record is transferred to main storage; thus, data transfer is completed more quickly when the arms are pre-positioned to a required track address prior to the execution of a READ statement. The FIND statement is never logically required in a program.

Example:

```
1       7
        FIND (4' 20)
          .
          .
          .
        READ (4' 20) A, B, C
```

This example shows the relationship between a READ statement and a FIND statement. While the access arms are being positioned, the statements between the FIND statement and the READ statement are executed.

9. Compilation

9.1. GENERAL

The FORTRAN IV compiler accepts source programs from either a card file or a disk file. Card files are entered directly into the card reader along with the appropriate job control stream. Disk files are built by the system librarian or from a workstation terminal with the system editor and its job control stream from a disk (filed job control stream) or from the card reader.

Appendix C contains compilation examples.

9.2. FORTRAN IV COMPILERS

The FORTRAN IV compilers are named FOR4 and FOR4L; both require one work file allocated in the job control stream. FOR4 requires 10800_{16} (X'10800') bytes of main storage plus space for the prologue; FOR4L requires 19000_{16} (X'19000') bytes of main storage plus space for the prologue. When FOR4 is executed, FOR4L will automatically be loaded if sufficient main storage was allocated for the job. No other use is made of additional storage. FOR4L contains significantly larger tables and workfile I/O buffers.

See Appendix C for examples of compilations.

9.3. PARAMETER STATEMENT FORMAT

Parameter statements for the compiler appear as punched cards in the job control stream.

Format:

```
1          10     16
//      △PARAM△    n1=d1,n2=d2,...
```

The // sequence must be in columns 1 and 2; columns 73 through 80 are not used. One or more blanks are required before and after PARAM, and one or more blanks are permitted after a comma. Each argument consists of a name (n), and equal sign, and a compiler directive (d). An argument may not contain embedded blanks. Multiple PARAM statements are permitted, but continuation is not. An argument may not continue on another card. For an explanation of statement conventions that apply to this section, refer to 1.4.

9.3.1. Compiler Arguments

A list of arguments provided by the FORTRAN IV compiler follows. Descriptions of the arguments follow the list.

Format:

LABEL	△OPERATION△	OPERAND
//	△PARAM△	OUT=filename,MAP=(S,A,L), LIN=filename,LST=option,OPT=(S,N,X,C,T), ERRFIL=module-name/lfdname IN=module-name/filename

Input Argument:

IN=module-name/filename
Specifies compilation of source programs residing in disk files.

Module-name is a one to eight alphanumeric character identifier indicating the name of a source module to be compiled. Filename is a one to eight alphanumeric character identifier indicating the name of a file in which the module resides. If /filename is not specifed, a default name is assumed and can be described via the LIN argument.

The occurence of an IN argument signals the end of the scanning for other PARAMs. Arguments following an IN argument on a given // PARAM card are ignored. Subsequent // PARAM statements may contain only IN arguments to allow for stacked compilations (see 9.5).

Output argument:

OUT=filename
Specifies the file in which the compiler is to place object modules.

A one to eight alphanumeric character identifier is specified by filename. If OUT is not specified, the compiler places all object modules in the temporary scratch file YRUN.

Map Argument:

MAP=(S,A,L)
Specifies the type of maps produced by the compiler. One or all options may be chosen. The options include:

S
Specifies object summary information, including module size and external subroutines called.

A
Specifies an alphabetical listing of the addresses assigned to variables, arrays, and statement labels.

L
Specifies a listing of the addresses assigned to variables, arrays, and statement labels in order by the storage locations assigned.

When a MAP argument is specified, it supercedes the maps selected by the LST argument. Also, when a MAP argument is specified, it is not necessary to specify LST=M.

Library Input Argument:

LIN=filename
Specifies the name of the default file in which the source modules reside.

A one to eight alphanumeric character identifier is specified by file name. If LIN is not specified, the compiler assumes the default filename of LIB1. This argument is used in conjunction with the IN argument.

Listing Argument:

LST=option
Specifies the quanity of listings produced by the compiler.

One option may be chosen. The options include:

N
Specifies an abbreviated listing consisting of only the compiler identification, parameters, and diagnostics.

S
Specifies, in addition to the N listing, the source code listing.

M
Specifies, in addition to the S listing, an object summary and a storage map showing the addresses assigned to variables and arrays. (Can be superseded by the MAP argument.)

If no LST PARAM is specified, the S option is assumed.

Options Argument:

OPT=(S, N, C, T)
Specifies compilation options.

One or all options may be chosen. The options include:

S
Specifies that statement numbers will be inserted into the generated code as an aid to debugging. When S is specified, the size of the object program and its execution time can increase significantly.

N
Specifies that no object program is to be generated. The program units are merely compiled and cannot be executed.

X
Specifies compilation of all cards with the character X in column 1. If this option is not specified, these cards will be treated as comments.

C
Specifies all references to array elements are to be checked to determine if they are outside the declared limits of the array.

T

Specifies that tracing of executed labels is requested. The compiler generates a special subroutine call at every label. A TRACE ON statement must occur in the program to activate tracing.

If only one OPT argument is specified, the parentheses are optional.

Error File Argument:

ERRFIL=module-name/lfdname

Specifies that an error file is created. This file allows the programmer to correct source codes using the error file processor (EFP) instead of the output listing.

Module-name is a one to eight alphanumeric character identifier indicating the name of the module for an OS/3 MIRAM source library file. Lfdname is a one to eight alphanumeric character identifier indicating the LFD name of an OS/3 MIRAM source library file.

If no ERRFIL parameter is specified, the error file is not written.

9.4. STACKED COMPILATION

The FORTRAN IV compiler is capable of processing up to 100 source program units during a single execution. When the source programs are on punched cards, one or more units may be placed between the /$ and /* data set delimiters. The data set is proceeded by compilation // PARAM statements. All FORTRAN IV compiler parameters are global and apply to all programs compiled. When a parameter is to be changed, the job control stream should be organized into two or more FORTRAN IV compilations, each containing the required parameters. For example:

```
1    5 7
// WORK1
// EXEC FOR4
// PARAM
/$
.
.              (one or more program units)
.
/*
// WORK1
// EXEC FOR4
// PARAM
/$
.
.              (one or more program units)
.
/*
```

When the source programs are on disk files, the programs are identified by using a librarian module name. A source module consists of one or more FORTRAN program units. The IN compiler parameter is used to identify source files to the compiler. Note that once FORTRAN IV encounters an IN parameter, no parameters, except other IN parameters, may occur.

When FORTRAN IV is doing a stacked compilation, the work file usage is cumulative. Thus, it may be desirable to do multiple executions of the compiler to reduce the work file requirements.

11. Consolidated Data Management (CDM) Execution Environment Configuration

11.1. CDM RELATIONSHIP

This section describes the interface between FORTRAN IV and consolidated data management (CDM), including: ←

- the relationship between unit numbers and external files;
- the kinds of devices supported;
- performance considerations such as record blocking and buffering; and
- system defaults (assumptions made by the system when specific directions are not provided).

Default actions taken when various errors are detected during program execution and how these defaults are changed to suit application requirements are also described. An example of a complete execution environment is given in 11.4.

The FORTRAN IV execution environment is consistent with the CDM of device independence. This means that substitution of similarly constructed files is possible without recompiling or reassembling. More discussion about device independence is presented in 11.3. ←

FORTRAN IV supports the following device classes:

- Disk and diskette
- Tape
- Workstation terminal
- Unit record (card reader, card punch, printer, and control stream input).

FORTRAN IV accesses these devices via the standard data management interfaces (CDIB and RIB structures). This interface is described in the OS/3 common data management concepts user guide, UP-8825 (current version).

Before a FORTRAN IV program can be executed, a group of I/O subroutines must be incorporated to support the FORTRAN I/O statements and provide an interface to data management. These I/O subroutines are individually called by the FORTRAN IV compiler and automatically placed in the executable program by the linkage editor. In the executable program, the control module is the center of the entire I/O scheme, because it contains the following:

- A unit table consisting of one entry for each unit number specified in the user program. Each entry contains FORTRAN control data and the information needed to connect the unit number to the actual device.
- A work area for record processing
- A buffer needed to support the REREAD feature

An executable program may only contain one I/O control module. This module may be supplied either by FORTRAN IV or configured by the user. When supplied by FORTRAN IV, the configuration allows for standard unit numbers used to reference any data management devices (11.2). When supplied by the user, he may configure his own set of unit definitions by using the FORTRAN IV unit definition procedure (UNIT). The user-supplied unit numbers are associated with the physical device via a device assignment set (DVC-LFD) at program execution.

11.2. CDM SUPPLIED CONFIGURATIONS

The following configurations are supplied for general use in simple applications. The unit numbers selected are industry standard. FORTRAN II I/O support is also included in these configurations.

Control Module FL$IO

Unit	Device
1	80 byte records; lfdname of FORT1; data can be reread; standard label if tape; can be cards in control stream if // LFD FORT1 is missing
3	Diagnostic device; lfdname of PRNTR; record size of 121; must be output device
5	Equivalent to unit 1
6	Equivalent to unit 3
29	Used to reread data from unit 1
READ	FORTRAN II READ statement (equivalent to unit 1)
PRINT	FORTRAN II PRINT statement (equivalent to unit 3)

Control Module FL$IO1

Unit	Device
1	80 byte records; lfdname of FORT1; data can be reread; standard label if tape; can be cards in control stream if // LFD FORT1 is missing
2	80 byte records; lfdname of FORT2
3	Diagnostic device; lfdname of PRNTR; record size of 121; must be output device

Table 11—1. FORTRAN IV Devices and Arguments

Argument	Disk	Tape	Unit record	Workstation	Equivalent	Reread
FAUE			*			
FBFSZ		*				
FBKNO		*				
FCHAR			*			
FCKPTREC		*				
FCLRW		*				
FCRDERR			*			
FDEVICE	*	*	*	*	*	*
FDIAGNOS	*	*	*	*		
FERROPT		*				
FEQUIV					*	
FFILABL	*					
FFILEID	*	*	*	*		
FIOOPT				*		
FLINCNTL				*		
FNUMBUF		*	*	*		
FOPRW		*				
FOPTION	*	*	*	*		
FRECFORM	*	*				
FRECSIZE	*	*	*	*		
FREREAD	*	*	*	*		
FSCREND				*		
FSPOOLIN	*	*	*	*		
FTRANS		*	*			
FTYPEFLE	*	*	*	*		
FUNIT	*	*	*	*	*	*
FVERIFY	*					

NOTE:

Only arguments applicable to the device type are recognized. All other arguments are ignored. If the record size or record format arguments are not compatible, they produce an OPEN error and terminate the job step.

If necessary, a unit can be defined as device dependent. This is done via a device type parameter on the UNIT definition procedure. Device dependency is discouraged because any device change requires reassembling a new UNIT definition. The UNIT definition procedure is called via an assembly language source module with the form.

```
1         10
-------------------------
name      START
          unit definition
          unit definition
                .
                .
                .
          unit definition_n
          unit termination
          error definition
          END
```

Each element of the preceding assembly module is discussed in 11.3.1 through 11.3.5.

11.3.1. START Statement

The START statement, a subprogram declarator statement required by the assembler, is the first statement of the configuration definition.

Format:

1	10
name	START

A 1- to 8-character symbolic name used to reference the control module on a linkage editor INCLUDE statement is specified by name. START is coded as shown.

11.3.2. UNIT Definition Procedure (UNIT)

Each file definition consists of a call on the FORTRAN unit definition procedure (UNIT), with optional arguments specifying characteristics of the file. Each argument consists of a file attribute name, an equal sign, and a particular characteristic of the file being defined. If an argument is not required it is omitted and the comma is deleted.

→ Consolidated data management (CDM) supplies the default file attributes whenever the target device is known (FDEVICE argument) and FORTRAN IV accepts these defaults. Attributes not relating to the assigned device are ignored.

When defining the file attributes with the UNIT definition procedure, the following syntactical differences between FORTRAN and assembly language should be remembered:

- In the assembler, the statement character is required for line 1 through (n-1) in column 72. whereas in FORTRAN it is required in lines 2 through n in column 6.
- No embedded blanks are permitted, and all continuation lines must start in column 16, as is illustrated in the subsequent examples.

11.3.2.1. Unit Record Definition

The following devices are considered unit record devices: reader, punch, and printer. The unit record devices are defined by using the UNIT procedure call in the paragraph. The arguments may appear anywhere in the UNIT definition procedure while FUNIT is the only required argument. Following the format, descriptions of the UNIT arguments and a UNIT example are presented.

FTYPEFLE=INPUT
Specifies an input file. If this argument is omitted and FUNIT=READ is specified, INPUT is assumed. INPUT should be specified if the disk is to be read but never written.

FTYPEFLE=OUTPUT
Specifies an output file. If this argument is omitted and FUNIT=PUNCH or PRINT is specified, OUTPUT is assumed. OUTPUT should be specified if the disk is to be written but not read.

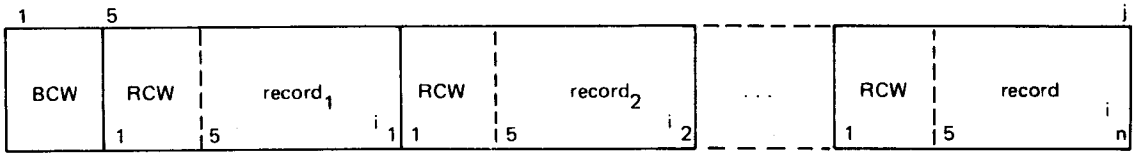
Buffer Size Argument:

FBFSZ=k
Specifies the size of the input/output area used in processing the file records. The size must be a positive value greater than or equal to the record size. System overhead is reduced if the buffer size is an integer multiple of the record size.

Record Format Argument:

FRECFORM=VARUNB
Specifies that the records are variable and unblocked.

Variable-length unblocked records



where:

i
Specifies record size

j
Specifies block size

BCW
Specifies a data management block control word

RCW
Specifies a data management record control word

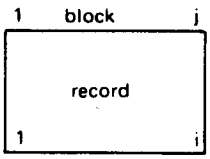
The FORMAT statement (7.3.3) may not specify a record greater than i–4. For unformatted input/output records, no size limitation exists because large FORTRAN IV records are automatically segmented into multiple data management records via the record control words identifying that start, middle, and the end segment of the I/O list.

The block and record control words are controlled by FORTRAN IV and data management and are not accessible with the FORTRAN IV language. The FBFSZ and FRECSIZE arguments are interpreted as maximums; shorter records are accepted and are generated, if possible, to save space on the external file and reduce channel contention for main storage access.

FRECFORM=FIXUNB

Specifies that the records are fixed and unblocked.

Fixed-length unblocked records



where:

i

Specifies FRECSIZE argument

j

Specifies FBFSZ argument

The block size (j) and the record size (i) must be equal. The FORMAT statement may not require more than i character positions. In an unformatted I/O list, no more than i bytes may be required for a record. In other words, when a FORMAT statement processes a record, it cannot request more than "recsize" bytes of data.

Record Size Argument:

FRECSIZE=k

Specifies the logical record size (in bytes). When accessing an existing file, the value is compared to the record size specification of that file. Any incompatibilities produce OPEN errors. If FRECSIZE and FRECFORM are omitted from the UNIT definition procedure, the file is processed using the physical file information. If the file has never been written to, a 255-byte logical record size and fixed, unblocked records are assumed.

Optional Units Argument:

FOPTION=YES

Specifies an optional unit, a unit not always required during program execution. When specified and the file is not allocated by job control, WRITE statements are ignored and the first READ reference causes an end-of-file condition. A unit need not be declared as optional if the program logic does not reference the unit.

Write Verification Argument:

FVERIFY=YES

Specifies that all WRITE statements cause the data to be automatically read back to ensure proper recording on the disk surface.

This increased reliability necessarily causes some performance degradation.

Diagnostic Messages Argument:

FDIAGNOS=YES
Specifies the current disk unit as the diagnostic device. If FRECSIZE is specified, its value must be 101 or greater. Debugging information may also be written to this device. If multiple diagnostic devices are specified, the FORTRAN system will post messages to the first diagnostic unit encountered. This argument is not available for input files. If omitted, diagnostics are transmitted to the system log and either the system console or the initiating workstation terminal.

Spooled Card Input File Argument:

FSPOOLIN=YES
or
FGETJCS=YES
Specifies that this unit will default to a spooled card input file via a GETCS when the lfdname declared in the FFILEID is not found.

The spoolin feature can be applied to any device but cannot exceed 128 bytes. If the record size is omitted, 80 bytes are assumed.

Reread Argument:

FREREAD=YES
Specifies this unit is to participate in the reread feature (7.3.4). The reread unit consists of a single buffer where each formatted input record is transferred. To conserve processor time, this data movement is inhibited unless specified.

Example:

```
1         10                                                        72
-------------------------------------------------------------------
          UNIT   FUNIT=35,                                          X
                 FDEVICE=DISK,                                      X
                 FFILEID=PAYROL,                                    X
                 FTYPEFLE=INPUT
```

This UNIT procedure call specifies a disk file (FDEVICE=DISK). The unit number is 35 (FUNIT=35), the file name is PAYROL, and it is an input file.

The following defaults are assumed by CDM if the file has never been written to:

- The record format (FRECFORM argument) is fixed and unblocked.
- The record size (FRECSIZE argument) is 255 bytes.

11.3.2.4. Workstation Unit Definition

The workstation unit is defined by using the UNIT definition procedure presented in this paragraph. The workstation terminal supports single line input and output (similar to a card reader) and full screen input and output provided by the screen format services. The arguments may appear anywhere in the UNIT definition procedure but FUNIT is the only required argument. Following the format, descriptions of the UNIT arguments and a UNIT example are presented.

Format.

```
1         10        16
          UNIT  [FDEVICE=WORKSTN] FUNIT={k     }
                                        {READ  }
                                        {PUNCH }
                                        {PRINT }

          [FFILEID={filename                    }] [FSCREND={WRAP    }]
                   {FORTk;if FUNIT=k            }          {SCROLL  }
                   {READER;if FUNIT=READ        }          {NEWPAGE }
                   {PUNCH;if FUNIT=PUNCH        }
                   {PRNTR;if FUNIT=PRINT        }

          [FTYPEFLE={WORK;                      }]
                    {INPUT;if FUNIT=READ        }
                    {OUTPUT;if FUNIT=PUNCH      }
                    {                or         }
                    {               PRINT       }

          [FIOOPT=YES] [FLINCNTL=YES] [FNUMBUF={1}]
                                              {2}

          [FOPTION=YES] [FRECSIZE=k] [REREAD=YES]

          [FSPOOLIN=YES] [FDAIGNOS=YES]
                 or
          [FGETJCS=YES ]
```

Device Identification Argument:

FDEVICE=WORKSTN
Specifies that this is a workstation terminal device.

NOTE:

The use of the FDEVICE parameter should be avoided since this negates device independence.

Unit Identifier Argument:

FUNIT=k
Specifies the unit identifier whose value is a unique integer constant in the range from $1 \leq k \leq 99$.

FUNIT=READ
Specifies READ as the unit identifier.

FUNIT=PUNCH
Specifies PUNCH as the unit identifier.

FUNIT=PRINT
Specifies PRINT as the unit identifier.

Appendix C. FORTRAN Sample Job Streams

C.1. JOB CONTROL PROCEDURE

The FOR4 procedure call statement generates the necessary job control statements to compile a FORTRAN IV program. Optionally, the procedure call statement can generate job control statements that specify the following:

- input – source library;
- output – object library;
- PARAM control statements defining the compiler processing logic; and
- automatically link and/or execute the program.

The input may be embedded data cards (/$, source deck, /*) immediately after the FOR4 procedure call, or a module in any library as defined by the IN parameter. This results in the appropriate DVC–LFD control statement sequence with an LFD name, LIB1, and the PARAM control statement, PARAM IN=module-name/LIB1, unless the PARAM LIN statement is specified.

The object code may be written in YRUN by default, but a specific output library can be specified by the OUT parameter. This results in the appropriate DVC–LFD control statement sequence with an LFD name, OUTFPUT, and the PARAM control statement, PARAM OUT=OUTFPUT.

The ALTLOD parameter generates the necessary DVC–LFD control statement with an LFD name, ALTLOD, and the appropriate EXEC control statement to load and execute the FORTRAN compiler from a private library other than YLOD. The format for the job control procedure is presented on the following page.

Format:

```
//[symbol] {FOR4  } [,PRNTR= { N                              } ] [,IN= { (vol-ser-no,label) } ]
           {FOR4L }          { (( lun ) [,vol-ser-no] )        }        { (RES)              }
           {FOR4LG}          { ((  N  )                )       }        { (RES,label)        }
                             { (( 20  )                )       }        { (RUN,label)        }

         [,OUT= { (vol-ser-no,label) } ] [,SCR1= { vol-ser-no } ]
                { (RES,label)        }           { RES        }
                { (RUN,label)        }
                { (*,label)          }
                { (RUN,$Y$RUN)       }

         [,ALTLOD= { (vol-ser-no,label) } ] [,OPT=(S,N,X,C,T)]
                   { (RES,label)        }
                   { (RUN,label)        }
                   { (*,label)          }
                   { (RES,$Y$RUN)       }

         [,LIN=filename] [,LST=option] [,MAP=(S,A,L)] [,SIZE= { L } ]
                                                              { S }

         [,ERRFIL=(vol-ser-no,label,module-name)]
```

Label:

symbol

Specifies the 1- to 6-character source module name; only used when the IN parameter is used.

Operation:

FOR4

This form of the procedure call statement is used to compile a FORTRAN IV source program.

FOR4L

This form of the procedure call statement is used to compile a FORTRAN IV source program and link-edit the object modules (see Note 1).

FOR4LG

This form of the procedure call statement is used to compile a FORTRAN IV source program, link-edit the object modules, and execute the load module (see Notes 1, 2, and 3).

NOTES:

1. *Linkage control cards or program data are not allowed with this form of the procedure call statement.*
2. *The FOR4LG procedure call statement cannot be used when operating with the shared code data management feature. Instead, use the FOR4L procedure call statement and provide a separate EXEC statement to execute the load module.*
3. *Device assignment sets must be specified prior to the jproc.*

Keyword parameter PRNTR:

```
PRNTR={N                              }
      {({lun}[,vol-ser-no])           }
      { {N  }                         }
      { {20 }                         }
```

Specifies the logical unit number of the printer, and, optionally, the destination-id (vol-ser-no). If a printer device assignment set is not to be generated, the value N is coded, and the printer device assignment set must be manually inserted in the control stream.

Keyword Parameter SCR1:

```
SCR1={vol-ser-no}
     {RES       }
```

Specifies the volume serial number of the work file labeled $SCR1. If omitted, the work file is assumed to be on the SYSRES device.

Keyword Parameter ALTLOD:

This parameter specifies the location of the alternate load library. If omitted, the compiler is loaded from YRUN.

ALTLOD=(vol-ser-no,label)
: Specifies the volume serial number (vol-ser-no) and file identifier (label) of an alternate load library that contains the FORTRAN IV compiler.

ALTLOD=(RES,label)
: Specifies that the alternate load library is located on the job's SYSRES device, in the file identified by the file identifier (label).

ALTLOD=(RUN,label)
: Specifies that the alternate load library is located on the job's YRUN file with the file identifier (label) specified by the user.

ALTLOD=(*,label)
: Specifies that the alternate load library is located on a catalog file identified by the file identifier (label).

Keyword Parameter OPT:

OPT=(S, N, X, C, T)
: Specifies one or all of the following compilation options.

S

Specifies that statement numbers will be inserted into the generated code as an aid to debugging. When S is specified, the size of the object program and its execution time can increase significantly.

N

Specifies that no object program is to be generated. The program units are merely compiled and cannot be executed.

X

Specifies compilation of all cards with the character X in column 1. If this option is not specified, these cards will be treated as comments.

C

Specifies all references to array elements are to be checked to see if they are outside the declared limits of the array.

T

Specifies that tracing of executed labels is requested. The compiler generates a special subroutine call at every label. A TRACE ON must occur in the program to activate tracing.

If only one OPT argument is specified, the parentheses are optional.

Keyword Parameter LIN:

LIN=filename

Specifies the name of the default filename in which the source modules reside.

A 1- to 8-alphanumeric-character identifier is specified by filename. If the LIN parameter is not specified, the compiler assumes the default filename of LIB1. This parameter is used in conjunction with the IN parameter.

Keyword Parameter LST:

LST=option

Specifies the quantity of listings produced by the compiler. One of the following options may be chosen.

N

Specifies an abbreviated listing consisting of only the compiler identification, parameters, and diagnostics.

S

Specifies, in addition to the N listing, the source code listing.

M

Specifies, in addition to the S listing, an object summary and a storage map showing the addresses assigned to variables and arrays. (Can be superseded by the MAP parameter.)

If no LST param is specified, the S option is assumed.

Keyword Parameter MAP:

MAP=(S, A, L)

Specifies the type of maps produced by the compiler. One or all of the following options may be chosen.

S

Specifies object summary information, including module size and external subroutines called.

A

Specifies an alphabetical listing of the addresses assigned to variables, arrays, and statement labels.

L

Specifies a listing of the addresses assigned to variables, arrays, and statement labels in order by the storage locations assigned.

When a MAP argument is specified, it supersedes the maps selected by the LST parameter. Also, when a MAP argument is specified, it is not necessary to specify LST=M.

Keyword Parameter SIZE:

SIZE={L / S}

Specifies the size of the FORTRAN IV compiler to be used.

L

Specifies the large version.

S

Specifies the small version.

If omitted, S is assumed.

Keyword Parameter ERRFIL:

This parameter specifies that error diagnostic messages are written to a file that is accessed by the error file processor. When you specify this parameter, error records are created for every error generated by the compiler.

ERRFIL=(vol-ser-no,label,module-name)

The vol-ser-no specifies the volume serial number of the file. The label specifies the file identifier (name of the file that the module is placed into). The module name is the name of the module that is referenced by the error file processor.

If omitted, the error file is not created.

Example 1a:

The following example illustrates the use of the FOR4 procedure call statement in its basic form:

```
      1         10     16
1.  // JOB FRTRN1A
2.  // FOR4
3.  /$
4.  .
5.  .   source deck
6.  .
7.  /*
```

Line	Explanation
1	Indicates that the name of the job is FRTRN1A.
2	Indicates the name of the procedure being called (FOR4). No keyword parameters specifying special options for this compile are used.
3	Indicates start of data.
4-6	Represents the source deck to be compiled.
7	Indicates end of data.

Example 1b:

The basic form generates the following control stream:

```
     1       10    16                                                72
1.  // JOB FRTRN1B
2.  // DVC 2Ø    // LFD PRNTR
3.  // DVC RES
4.  // EXT ST,C,3,CYL,1
5.  // LBL $SCR1  // LFD $SCR1
6.  // EXEC FOR4
7.  /$
8.  .)
9.  .}source deck
1Ø. .)
11. /*
```

Line	Explanation
1	Indicates that the name of the job is FRTRN1B.
2	Indicates the default logical unit number and LFD name of the printer.
3-5	Indicates that the work file needed for compiling is, by default, on the SYSRES device, has both a file-label and LFD name of $SCR1, and uses the sequential access technique; that allocation is contiguous; that three cylinders are allocated for the secondary increment; and that one cylinder is allocated for the first extent.
6	Loads the FORTRAN IV compiler from YLOD.
7	Indicates start of data.
8–10	Represents the source deck to be compiled.
11	Indicates end of data.

Example 2a:

The following example illustrates the use of a FOR4 procedure call statement that defines all the keyword parameters:

```
1.  // JOB FRTRN2A
2.  //PROGNM FOR4 PRNTR=21,IN=(DSC1,U$SCR),                              X
3.  //1           OUT=(DSC2,U$OBJ),LST=S,SIZE=L,                        X
4.  //2           SCR1=DSC2,ALTLOD=(DSC3,ALTLODLIB)
5.  /&
```

Line	Explanation
1	Indicates that the name of the job is FRTRN2A.
2	Indicates the name of the procedure being called (FOR4). The source module name is PROGNM. The logical unit number of the printer is 21, and the input file has a volume serial number of DSC1, with a file-label of U$SCR.

Index

D

F

G

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

long line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD