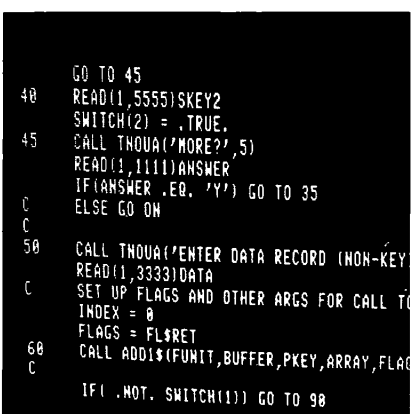
Prime Computer, Inc.

Fortran
Revision 18



FDR3338

The Programmer's Companion is a new series of pocket-size quick-reference guides to Prime software products

Published by Prime Computer, Incorporated
500 Old Connecticut Path, Framingham, MA 01701

Produced by Prime Computer Technical Publications Department, 500 Old Connecticut Path, Framingham, MA 01701

The information contained in this document reflects the software as of Revision 18 1 and is subject to change without notice Prime Computer Inc assumes no responsibility for errors that may appear in this document

First Printing July 1978
Second Printing Revision February 1979
Third Printing Revision January 1980
Fourth Printing Revision March 1982

Credits

Research and copy
Anthony Lewis

Design
William I Agush

Production
D Christine Benders

Typesetting
JL Associates, Berkeley Typographers

Printing and binding
Commonwealth Publishing

CONTENTS

TYPOGRAPHIC CONVENTIONS

abbreviation of PRIMOS commands

The minimum required abbreviation of PRIMOS commands is shown in rust-colored letters Only internal commands can be abbreviated

braces { }

Of a group of words or parameters contained within braces at least one *must* appear in a command or statement

comma

Where a comma appears in a FORTRAN statement it is *required*

lowercase

A parameter whose legal value is to be selected by the user is printed in lowercase letters

parentheses ( )

Parentheses where they appear are a *required* literal part of the command or statement syntax

square brackets [ ]

A word or parameter enclosed in square brackets is *optional*

PRIMOS CONCEPTS

binary file

A translation of a source file generated by a language translator (FTN COBOL RPG) Such files are in the required format for the loaders Also called **object files**

byte

8 bits 1 ASCII character

directory

A special type of file containing a list of names of files or other directories along with information on their characteristics and location A directory may be the MFD, a UFD or a sub UFD Directories with names in the MFD are UFDs all others are sub UFDs

file

An organized collection of information stored on a disk (or a peripheral storage medium such as tape) Each disk file has an identifying label called a **filename.**

filename

The name of a file or directory Filenames may have up to 32 characters The first character *must not* be numeric (0-9) Filenames can be composed of *only* the following characters

A-Z, 0-9, _ # $ & - * /

file-unit, PRIMOS

A number between 1 and 127 (177) assigned as a pseudonym to an active file by PRIMOS This number may be given in place of a filename in certain commands, such as CLOSE PRIMOS level internal commands require octal values

PRIMOS assigned units	Octal	Decimal
INPUT	1	1
LISTING	2	2
BINARY	3	3
COMINPUT	6	6
SEG's Loadmap	13	11
COMOUTPUT	177	127

mode, addressing

An addressing scheme The mode determines the construction of the instructions generated by the compiler

LOAD

Prime's linking loader for 32R and 64R modes See the Loading and Debugging Programmer's Companion

pathname

A multi part name which uniquely specifies a particular file (or directory) within a file system tree A pathname (also called a treename) gives a path from the disk volume, through directory and subdirectories, to a particular file or directory Its format is

$$\left\{\begin{array}{c}\text{<volume>}\\ \text{<ldisk>}\\ \text{<*>}\end{array}\right\}\ \text{directory [password]}$$

$$\left[\ \text{subdirectory [password]}\ \right] > \left\{\begin{array}{l}\text{filename}\\ \text{subdirectory [password]}\end{array}\right\}$$

<volume> is the disk name, <ldisk> is the logical disk number, <*> represents the current volume Note the angle brackets are required

PRIMOS

Prime's family of single and multi-user disk operating systems

SEG

Prime's linking loader for 64V mode See the Loading and Debugging Programmer's Companion

segment

A 65,536 word block of addressing space

source file

An ASCII text program file consisting of text, program statements, comments etc

FORTRAN CONCEPTS

array

An ordered multidimensional set of data (In FORTRAN, an array may have 1 to 7 subscripts)

characters, legal

The following characters are allowed in Prime FORTRAN IV

The 26 letters A-Z
The 10 digits 0-9
These 12 special characters = ' + - * / ( ) , $
Blank (i e , space)

COMMON

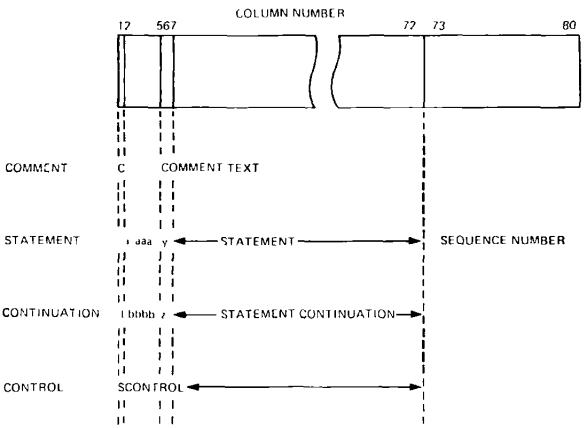
A named area in memory shared among program modules loaded into the same runfile

data type

See DATA TYPE section

line format

Each program line is a string of 1 to 72 characters Each character position in the line is called a column, numbered from left to right starting with 1 The following is a schematic of a program line



aaaaa	STATEMENT LABEL (OPTIONAL)
bbbbb	BLANKS
v	BLANK OR ZERO
z	ANY CHARACTER EXCEPT BLANK OR ZERO

Note bbbbb may be a statement number but control cannot be transferred to it

operands

Operands are those elements which are manipulated by the program They are constants parameters, variables, arrays, and address constants

operators, arithmetic

**	Exponentiation	+	Addition
-	Unary minus	-	Subtraction
*	Multiplication	=	Equality or replacement
/	Division		

operators, logical

AND	Logical intersection
NOT	Logical negation
OR	Logical union (non-exclusive OR)

operators relational

LT	Less than
LE	Less than or equal to
EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to

operators priority of

FORTRAN evaluates operators within expressions in the following order

**	Exponentiation
-	Unary minus
* or /	Multiplication or division
+ or -	Addition or subtraction
LT LE EQ NE GT GE	Relational operators
NOT	Logical negation
AND	Logical intersection
OR	Logical union

At equal level of operators priority evaluation proceeds from left to right Expressions within parentheses are evaluated before operations outside the parentheses are performed

parameters

Parameters are named constants and can be in any data mode

unit number FORTRAN

The following is a list of default FORTRAN/PRIMOS device correspondences

FORTRAN unit number	PRIMOS device
1	User terminal (reading and writing)
2	Paper tape reader/punch
3	Parallel interface card reader
4	Serial line printer
5 20	file unit 1 16
21 24	9 track magnetic tape unit 0 3
25 28	7 track magnetic tape unit 0 3
29 139	file unit 17 127

variables, FORTRAN

Variable names have from 1 to 6 characters Character 1 *must* be alphabetic characters 2-6 (if any) must be alphanumeric If no mode is specified, variables beginning with the letters I, J, K L M, N are INTEGER, variables beginning with either A-H or O Z are REAL

PROGRAM COMPOSITION

Each program (or subroutine or external function) consists of a number of program lines Program lines are grouped and ordered by type of statement as shown below Comments, TRACE and LIST control statements can be used anywhere in the program The END statement *must* be the last statement of a program, nothing may follow END except FUNCTION or SUBROUTINE of another subprogram The last logical statement of a main program should be CALL EXIT for an orderly return to PRIMOS command level

Header statement if required

FUNCTION SUBROUTINE, BLOCK DATA

Storage and Specification Statements

COMMON, DIMENSION EQUIVALENCE, SAVE, EXTERNAL, COMPLEX DOUBLE PRECISION, INTEGER, INTEGER*2, INTEGER*4, LOGICAL, REAL, REAL*4, REAL*8 IMPLICIT PARAMETER

DATA Statements

Statement Function Definitions

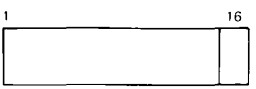
Executable Statements

Arithmetic and logical assignments	
Control Statements	GOTO, ASSIGN IF DO, CONTINUE, PAUSE, STOP, RETURN
Input/Output Statements	READ, WRITE, PRINT, FORMAT REWIND, BACKSPACE, END FILE
Subroutines	CALL subname [(arg-1, , arg-n)]

END Statement

PRIME MEMORY FORMATS AND FORTRAN DATA TYPES

LOGICAL



Equivalent to INTEGER*2 values of 0 and 1 respectively
All other values *illegal* for LOGICAL variables

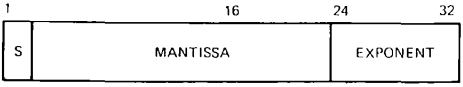
INTEGER*2

INTEGER*2 are in 2's complement with a range from -32768 to 32767 octal '100000 and '077777 respectively -0=0 -(-32768)=-32768 Integer division truncates

INTEGER*4

INTEGER*4 are in 2's complement with a range from -2147483648 to 2147483647, octal (word-1, word-2) '100000, 000000 and '077777, '177777 respectively -0=0 -(-2147483648)=-2147483648 Integer division truncates

REAL*4 (REAL)



BIT 1 SIGN BIT
BITS 2 24 MANTISSA
BITS 25 32 EXPONENT

Mantissa and sign are treated as a 2's complement number Exponent is an unsigned, excess 128, binary exponent

Number=mantissa*2**(exponent-128) (zero is mantissa=exponent=0)

$-1 \leqslant mantissa < -1/2$
$1/2 \leqslant mantissa < 1$
$0 \leqslant exponent \leqslant 255$

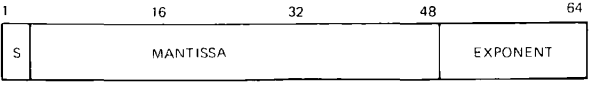
Value range is −1*2**127 to (1-ϵ)*2**127, octal value (word-1, word 2) '100000, '000377† to '077777, '177777 respectively

Values *closest* to zero are
(-1/2+ϵ)*2**-128 ('137777, '177400) and 1/2*2**-128 ('040000, '000000)†

Effective precision is between 22 and 23 bits

†-These numbers cause exponent overflow if negated due to the asymmetry of 2's complement notation

REAL*8 (DOUBLE PRECISION)



BIT 1 SIGN BIT
BITS 2 48 MANTISSA
BITS 49 64 EXPONENT

Mantissa and sign are treated as a 2s complement number The exponent is a signed, excess 128, binary exponent

Number=mantissa*2**(exponent-128) (zero is mantissa=exponent=0)

$-1 \leq$ mantissa $< -1/2$
$1/2 \leq$ mantissa < 1
$-32768 \leq$ exponent < 32767

Value range is $-1*2**32639$ to $(1-\epsilon)*2**32639$, octal value (word 1 word 2 word 3, word-4) '100000, '000000 '000000,'077777† to '077777,'177777,'177777,'077777

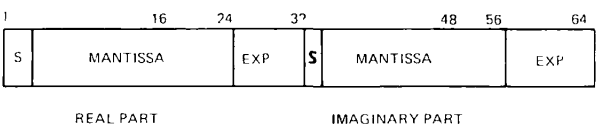
Values closest to zero are

$(-1/2+\epsilon)*2**-32896$ ('137777 177777 '177777 '100000) and $1/2*2**-32896$ (040000 000000 000000 100000)†

Effective precision is between 46 and 47 bits

†-These numbers cause exponent overflow if negated due to the asymmetry of 2's complement notation

COMPLEX



Two REAL*4 numbers representing the real and imaginary parts

CHARACTERS

The 8-bit marking variety (parity bit *always on*) of ASCII is used for standard internal and external character code PRIME's code set is effectively a 128 character code set (ASCII spacing representation (parity bit *always off*) can be entered into the system but *most* system software will fail to recognize the characters as their terminal printing equivalent)

The parity bit in the left-hand character corresponding to the sign bit equals 1, (i.e., negative number) Characters packed into numeric items will always be negative numbers if accessed numerically If the data item is not completely filled (e.g. A2 format into a REAL*4 item), the item will be right padded with blanks (ASCII '240)

The positions of the exponents for REAL and DOUBLE PRECISION items preclude sorting character data as

REAL items, but sorting is legitimate in integer items. EQUAL comparisons in REAL items are valid It is recommended that integer data types be used for character representation

FORTRAN LANGUAGE STATEMENTS

A B *Assignment*

Assigns a value to a variable

Data mode rules

FROM B (right hand side)	TO A (left hand side) I*2	I*4	R*4	R*8	C
I*2	A	S A	FI A	DF A	FL AtR
I*4	I A	A	FI RA	DF A	FI AtR
R*4	Fx A	Fx A	A	DE A	AtR
R*8	Fx A	Fx A	DI RA	A	X
C	Ix AR	Ix AR	AR	X	A

A	Assign	Transmit resulting value without change
AR	Assign Real Part	
AtR	Assign to Real Part (Imaginary Part=0)	
RA	Real Assign	Transmit as much precision of the most significant part of the resulting value as Real datum can obtain
DE	DP Evaluate	Evaluate, then DP float
FL	Float	Transform value to Real datum form
DF	DP Float	Transform value to Double-precision form
Fx	Fix	Truncate fractional part and transform integral part to integer
T	Truncate	Take 16 low-order bits and store in short integer datum

S	Sign-Extend	Pad 16 high order bits with 0s or 1s if short integer is positive or negative respectively
X		Not permitted

ASSIGN k TO i *Control*

Assigns a statement number k to a variable i

BACKSPACE u *Device Control*

Repositions FORTRAN unit u so that the preceding record is now the next record (Magnetic tape only)

BLOCK DATA *Header*

Labels a block data subprogram

CALL subroutine [(arguments)] *External*

Calls the specified **subroutine** with an optional list of **arguments** to be passed and returned

COMMON /X1/A1/ /Xn/An *Storage*

Defines COMMON blocks Each **A** is a non-empty list of variable names or array names and each **X** is a COMMON block name or is empty (blank COMMON) The format // (with *no* characters except blanks between the slashes) may be used to denote blank COMMON

[statement-number] CONTINUE *Control*

Transfers control to the next logical executable statement

DATA k1/d1/, kn/dn/ *Data Initialization*

Initializes variables or array elements **k** to the values **d** at load time

DECODE (c,f,a[, ERR-b]) list *Coding*

Formatted DECODE Converts the first **c** characters in array from ASCII data into the **list** elements according to the specified format **f** **ERR-b** transfers control to statement number **b** if a FORMAT/DATA mismatch occurs

DECODE (c,*,a[, ERR-b]) list *Coding*

List-directed DECODE Permits inputting/decoding of data from free-format input devices (such as a terminal) **ERR-b** transfers control to statement number **b** if a FORMAT/DATA mismatch occurs

DIMENSION V1(I1),...Vn(In) *Storage*

Declares the name of the array, the number of subscripts (IJ=J1, J7), and the maximum value of the subscripts

DO n i=m1, m2 [,m3] *Control*

Executes statements up to and including the statement label **n** **m1**, **m2** and **m3** are positive integers (constants parameters or variables *only* — *no* array elements or expressions) with m2≥m1 i is an integer variable which assumes the values m1, m1+m3, m1+2*m3, etc m1 is the initial value, m2 is the limit value and m3 the increment If m3 is not specified, the default value 1 is used

ENCODE (c,f,a) list *Coding*

Converts the elements of **list** into ASCII data according to format **f** and stores the first **c** characters in array **a**

END *Control*

The final statement of program subroutine, or external function

ENDFILE u *Device Control*

Writes an end-of-file record on FORTRAN unit **u** indicating the end of a sequential file for magnetic tape Closes a disk file on FORTRAN unit **u**

EQUIVALENCE (k11, k12, k13...), (k21, k22, k23...) *Storage*

Overlays single variables onto each other, entire arrays onto each other, elements of an array onto single variables, etc

EXTERNAL V1, Vn *External*

Permits the name of an external function, **V**, to be passed in a subroutine call or function reference

statement-number FORMAT (df1 dfn) *Format*

Defines format fields by **statement-number.**

d	Format delimiter
/	Proceed to next record
,	Remain within current record

Throughout description of FORMAT statements **m** is used to represent the scale factor and **n** the repeat group

Results of formats in Input statements

mPnFw.d	*Floating*
mPnEw.d	*Exponential*
mPnGw.d	*General*
mPnDw.d	*Double precision*

Scale factor Internal value formed by dividing the input number by 10**m (no effect if input number has D or E representation)

Input numbers may be represented as integers mixed integers or scaled numbers (with exponents) Leading blanks are treated as zeroes imbedded and trailing blanks are ignored The implied decimal point is placed to the left of the first d digits counting from the right (if no decimal point in the input number) A decimal point in the input number overrides the positional decimal point The decimal exponent (D or E) and the exponent value are a unit, both must be included or omitted All numbers are assumed positive unless a minus sign is present

wX *Space†*

Skips **w** columns in the input data (negative w backspaces to reload record)

Tw *Tab*

Tabs to column **w** in the input record

wHc1c2 cw *Hollerith*

Not used

nAw *ASCII*

Stores ASCII characters in Integer Real Complex or Double-precision variables If input is greater than storage available in variables, only the left-most characters are stored

nLw *Logical*

Stores true/false in internal representation based upon first non-space character in the input data (all others ignored) If T—set to +1, if F—set to 0, if anything else—set to 0 and set error flag

nIw *Integer*

Stores input numbers in integers If no sign is present, a plus sign is assumed A sign or blank is counted as one character position No decimal points are allowed

If there are more numbers than the field width **w** only the left-most w characters are stored

B 'string' *Business*

Not used

†-No repeat count is allowed associated with the format specifier itself but the format specifier may be included in a group repetition

Results of formats in Output statements

mPnFw.d *Floating*

Prints Real or Double-precision Numbers as mixed ouput (no exponent) with as many significant figures as the data type allows **w** is the total field width and must allow one position for a decimal point and one for a minus sign (if negative numbers are to be printed) **d** is the number of decimal places (right of decimal point) Numbers are right justified Leading zeroes are inserted for numbers less than 1 trailing zeroes are used to fill the decimal places if necessary Only minus signs are printed If total field width is too small, the number is truncated and a **$** is printed if positive, an = if negative If the decimal section is too small, the number is rounded Scale the output number is multiplied by 10**m

mPnEw d *Exponential*

Prints Real or Double-precision numbers as a number with a magnitude between 0 1 and 0 9999999 times an exponent The field width **w** must allow for a minus sign (if one is to be printed) a decimal point and three or four positions for the exponent representation The number **d** sets the number of places to the right of the decimal point —the maximum is seven

The representation with magnitude less than 1 may be overridden using scale factors (**m**) Before output conversion the fractional part of output number is multiplied by 10**m exponent decreased by m

Exponent Value		Representation	Width
wxyz	9999 to 1000	=wxy	4
xyz	999 to 100	-xyz	4
yz	99 to 10	E yz	4
z	9 to 9	E-z or E z	4
yz	10 to 99	E yz	4
xyz	100 to 999	+xyz	4
wxyz	1000 to 9999	$wxy	4

mPnGw d *General*

Prints Real or Double-precision numbers in F or E format according to the magnitude of the number and the decimal place specifier **d**

Range	Equivalent format
0.1 to 1.0	F(w-4) d,4X
1.0 to 10.0	F(w-4) (d-1) 4X
•	•
•	•
•	•
10**(d-2) to 10**(d-1)	F(w-4) 1 4X
10**(d-1) to 10**d	F(w-4) 0 4X
Outside Range	Ew d

The scale factor m is effective *only* in E format range then works same as E format

mPnDw d *Double precision*

Prints Double-precision numbers only in an exponential format similar to the E format except that the letter D is used instead of E and that **d** has a maximum value of 14 Scale factor in same as E format

wX *Space*†

Writes **w** spaces into the output record (negative w back-spaces for replacing)

Tw

Positions output pointer to column **w** in the output record Back tabbing is permitted

wHc1c2 cw *Hollerith*†

Prints the string **c1c2 cw**

Does not require an item in the output list

Need not be followed by a delimiter

nAw *ASCII*

Prints Integer, Real, Complex or Double-precision variables as ASCII characters **w** is number of characters per variable or array name Output is right justified and padded with spaces

nLw *Logical*

Prints logical variables +1 prints as T 0 prints as F Output is right justified and padded with spaces If **w**<1, there is no output

nIw *Integer*

Prints contents of integer (short or long) variables or array names as a string of integers (no decimal points) If string is longer than field width **w** then number is right truncated and preceded by a $ if positive and = if negative Minus signs are printed but not plus signs

B 'string' *Business†*

Prints templated numerical output for business purposes

Features include Fixed and floating signs, trailing signs plus sign suppression, trailing minus change to CR fixed and floating $ field filling leading zero suppression insertion of commas Length of string determines field width, if number is greater than field width output is printed as string of asterisks

String Symbol	Usage
+	Fixed sign
+ +	Floating sign
-	Fixed sign plus sign suppression
- -	Floating sign plus sign suppression
$	Fixed currency sign
$ $	Floating currency sign
Z	Print digits 1-9, replace leading zeroes with blanks
#	Print digits 0-9
.	Position of decimal point
,	Position of comma
CR	Trailing blank (positive) or CR (Negative)
*	Fill field with asterisks

† No repeat count is allowed with the format specifier itself but the format specifier may be included in a group repetition

FULL LIST *Compile Runtime*

Causes a listing of subsequent source code with a symbolic listing Overridden by compiler parameters

[mode] FUNCTION name (arguments) *Header*

Defines the **name** of the function in the calling program i e the variable that returns the value calculated by the function

GO TO k *Control*

Unconditional Transfers control to statement **k**

GO TO i [(k1 kn)] *Control*

Assigned Transfers control to statement **i** i must have been previously assigned a value

GO TO (kl kn) i

Computed Transfers control to statement **kj** when integer expression **i=j** If the value of i lies outside the range 1 to n then control passes to the next executable statement after the computed GO TO

IF (logical-expression) statement *Control*

The **logical-expression** may be TRUE or FALSE **statement** is any valid executable statement except a DO or a logical IF If logical expression is true the statement is executed if false control passes to the next executable statement

IF (arithmetic-expression) k1 k2 k3 *Control*

The **arithmetic-expression** has either an integer real or double precision value If arithmetic-expression 0< (negative) control is passed to statement **k1** if =0 (exactly) control is passed to statement **k2**, and if >0 (positive) control is passed to statement **k3**

IMPLICIT mode-1 (list-1) mode-n (list-n)

Specification

Overrides the language convention for default data typing by first letter of variable name

INSERT *Compiler/Runtime*

See **$INSERT**

LIST *Compiler/Runtime*

Causes a listing of subsequent source code with no symbolic listing. Overridden by compiler parameters.

mode V1,...Vn *Specification*

Overrides the implicit mode assignments of symbol names done either by IMPLICIT or language default.

NO LIST *Compile/Runtime*

Causes a cessation of subsequent source code listing and of symbolic listing. Overridden by compiler parameters.

PARAMETER (V1-C1, . . . Vn=Cn) *Specification*

Sets parameter values. The **V**'s are variables (arrays not allowed) and the **C**'s are constants or constant-valued expressions of the same mode as the corresponding variables. *Prime's FORTRAN compiler will also accept the list without the enclosing parentheses.*

PAUSE [n] *Control*

Halts the program. Prints ****PA n (R-identity) or ****PAUSE n (V-identity) at the terminal. **n** is an optional five-digit decimal number and is printed in octal representation.

PRINT f [,list] *I/O*

Prints the **list** of elements at the user's terminal in format **f**. Equivalent to WRITE (1,f) [list].

READ (u,f[, END=a] [, ERR=b]) list *I/O*

Formatted READ. Reads data on FORTRAN unit **u** into the variables/array names specification **list**, according to format **f**. **END=a** transfers control to statement number a if an end-of-file condition is encountered. **ERR=b** transfers control to statement number b if a device or format error is encountered.

READ (u [, END=a] [, ERR=b]) list *I/O*

Binary READ. Reads data on FORTRAN unit **u** into the variables/array names specification **list**. **END=a** transfers

control to statement number a if an end-of-file condition is encountered **ERR=b** transfers control to statement number b if a device error is encountered

READ (u,*[, END=a] [, ERR=b]) list *I/O*

List-directed READ Converts input data from the free-format device into the list items according to the data type **END-a** transfers control to statement number a if an end-of-file condition is encountered **ERR b** transfers control to statement number b if a device or format error is encountered

READ (u'r[,f] [,ERR=b]) list *I/O*

or

READ (u[,f], REC=r [,ERR=b]) list *I/O*

Direct Access READ Reads data from record **r** of file opened on unit **u** into **list** of variables according to format **f** If the format is omitted a binary READ is performed **ERR=b** transfers control to statement number b if a device or format error is encountered

RETURN *Control*

Returns to the main program from a subroutine or external function

REWIND u *Device Control*

Repositions FORTRAN unit **u** to its initial point *Does not* close or truncate disk file

SAVE V1, Vn *Storage*

Assigns local storage in the linkage frame to specified variables or array names *(Meaningful in 64V mode only ignored in other modes)*

STOP [n] *Control*

Halts the program Prints ****ST n (R-identity) or ****STOP n (V-identity) at the terminal and returns control to PRIMOS level **n** is an optional decimal number of up to five digits and is printed in octal representation

SUBROUTINE name (arguments) *Header*

Defines a program as a callable subroutine

TRACE V1, . . Vn *Compile/Runtime*

Item TRACE Prints the value of the variable at each point in the program where the variable is modified

TRACE n *Compile/Runtime*

Area TRACE Prints the value of the variables used in statement **n** during execution of the code between the area TRACE statement and statement n

WRITE (u,f [,ERR=b]) list *I/O*

Formatted WRITE Writes out data in **list** to FORTRAN unit **u** according to format **f** **ERR=b** transfers control to statement number b if a device error is encountered

WRITE (u [,ERR=b]) list *I/O*

Binary WRITE Writes out all data in the **list** into a record in binary format **ERR=b** transfers control to statement number b if a device error is encountered

WRITE (u'r [,f] [,ERR-b) list *I/O*

or

WRITE (u [,f], REC=r [,ERR=b]) list *I/O*

Direct Access WRITE Writes data from record **r** of file opened on unit **u**, into **list** of variables, according to format **f** If the format is omitted, a binary WRITE is performed **ERR=b** transfers control to statement number b if a device error is encountered

$INSERT pathname *Compile/Runtime*

Inserts the file **pathname** into the program at compilation time

FORTRAN IV COMPILER

FTN { **pathname [options]** / **[options] -INPUT pathname [options]** }

Invokes the FORTRAN IV compiler The source program file should be input-filename FTN For more information, refer to the FORTRAN Reference Guide
Options (• indicates Prime-supplied defaults)
Specify Input/Output Devices

BINARY { **pathname** / **NO** / • **YES** }

Specifies binary (object) file Default name input-filename BIN

INPUT pathname

Specifies source file

-LISTING { **pathname** / • **NO** / **YES** / **TTY** / **SPOOL** }

Specifies listing file Default name input filename LIST

-SOURCE pathname

Same as INPUT

Enable Listings Cross References

	-ERRLIST	Print error-only listing
•	**-ERRTTY**	Print error messages at user terminal
	-EXPLIST	Print listing including assembler-like output
•	**-LIST**	Print source program and error listing
	-NOERRTTY	Suppress error messages to terminal
•	**-NOTRACE**	Suppress global trace
•	**-NOXREF**	Suppress cross reference listing
	-TRACE	Enable global trace
	-XREFL	Print full cross reference listing
	-XREFS	Print partial cross reference listing (referenced variables only)

Memory Usage

	-BIG	Handle arrays spanning segment boundaries (64V only)
	-DEBASE	Conserve loader base areas
	-DYNM	Enable dynamic allocation of local storage (64V only)
•	**-NOBIG**	No arrays spanning segment boundaries
	-PBECB	Generate code to load ECBs into procedure frame
•	**-SAVE**	Static allocation of local storage
•	**-32R**	Generate code to run in 32R mode
	-64R	Generate code to run in 64R mode
	-64V	Generate code to run in 64V mode

Operations

	-DCLVAR	Flag undeclared variables
•	**-FP**	Generate floating point skip instructions
	-FRN	Round REAL*4 numbers moved from registers to main storage
	-INTL	INTEGER default is INTEGER*4
•	**-INTS**	INTEGER default is INTEGER*2
•	**-NODCLVAR**	*Do not* flag undeclared variables
	-NOFP	Suppress generation of floating-point skip instructions
•	**-NOFRN**	Do not round REAL*4 numbers moved from registers to main storage
	-SPO	Special library compilation

Optimization

	-OPT	Optimize DO loops which do not contain GO TO statements
•	**-STDOPT**	Perform standard optimization
	-UNCOPT	Optimize DO loops unconditionally

Debugging

	-DEBUG	Generate code for full debugger (DBG) functionality (64V mode only)
•	**-NODEBUG**	Do not generate code for debugger
	-PROD	Generate code for partial debugger (DBG) functionality (64V mode only)

COMPILER ERROR MESSAGES

ARG LIST REQUIRED

Argument list *not* specified in FUNCTION statement

ARRAY NAME REQUIRED

Something other than an array name appeared in a position where *only* an array name is allowed

ARRAY NESTING OVFLO

Use of arrays as subscripts in other arrays exceeds allowable nesting limit (32)

ARRAY/BLOCK OVERFLOW

Array/Block exceeds space allocated to user

CHAR STRING SIZE

A character string was not terminated, or a string in a DATA statement was longer than the associated variable list

COMMON NAME ILL

Illegal use of a name already declared in COMMON

COMPILER OVERFLOW

Insufficient memory to compile program

CONFLICTING DECLARN

Name(s) declared as more than one data mode

CONSTANT REQUIRED

A name appeared where *only* a constant or parameter is allowed

CONSTANT TOO LARGE

Constant exponent excessive for data type

DATA MODE ERROR

Illegal mode mixing in expression expression mode not of required type, or constant in DATA statement is of different mode than associated name in variable list

DEBUG TURNS OFF OPT

Both the -DEBUG and -OPT (or -UNCOPT) options were selected Compilation will proceed as if the optimization option was not included

DIVISION BY ZERO

Attempt to divide by a zero constant

END/REC PROHIBITED

The END=statement-number cannot be used in a direct access READ or WRITE statement

EXCESS CONSTANTS

Number of constants in DATA statement exceed variables for storing them

EXCESS SUBSCRIPTS

Too many subscripts in EQUIVALENCE or DATA list item

FUNCT VAL UNDEFINED

The function name was not assigned a value in FUNCTION subprogram

GBL MDE/IMPL CNFLCT

Implicit statement and global mode specification may *not* be used in the same program unit

ILL. CONSTANT EXPR.

Variables found in a PARAMETER statement

ILL. DO TERMINATION

Improper DO loop nesting, or an illegal statement terminating a DO loop

ILL. EQUIVALENCE

EQUIVALENCE group violates EQUIVALENCE rules or specifies an impossible equivalencing

ILL. LOGICAL IF

A logical IF contained *within* a logical IF, or a DO statement contained *within* a logical IF

ILL. OVER 64K COMMON

A COMMON area exceeds 64K words of user memory

ILL. STMT NO. REF

Reference to a specification statement number

ILL. UNARY OP USAGE

Improper use of an operator in an expression

ILL. USE OF ARG

SUBROUTINE or FUNCTION statement used in COMMON, EQUIVALENCE or DATA statement

ILL USE OF CLMN. 6

Continuation line found without a continuation or statement line preceding it

ILL USE OF STMT

Statement illegal in context of program

INCONSISTENT USAGE

The use of the name listed in the error message conflicts with earlier usage This message also will be generated at the END statement in a SUBROUTINE subprogram if the name is used within the subprogram

INTEGER REQUIRED

A non-integer name or constant appeared where only an integer name or constant is allowed

INTERNAL ERROR

Some combination of source code statements has generated an unresolvable error The programmer should never see this error

MULT DEF STMT NO

The statement number of the current line has already been defined

NAME REQUIRED

A constant appeared where only a name is allowed

NO DEBUG IN R MODE (warning)

The -DEBUG (or -PROD) option was included for compilation in a mode other than 64V Compilation will proceed as if the debugging option was not included

NO END STMT

The last statement in the source was not an END statement

NO PATH TO STMT

The current statement does not have a statement number and the previous statement was an unconditional transfer of control Also generated at the end of a program unit for labelled statements, if control cannot reach these statements

NONCOMMON DATA

A BLOCK DATA subprogram initialized data not defined in COMMON, or contained executable statements

PAREN NESTING>31

Nesting of parentheses (syntactical, array, or function reference) in expressions may not exceed 31

PARENTHESIS MISSING

Incorrect parenthesis used in an implied DO loop in an I/O statement

PROG SIZE OVERFLOW

Program too large for allocated user space

SAVE ITEM ILLEGAL

Improper item in SAVE statement

STMT NAME SPELLING

A statement name was recognized by its first four characters, but the *remaining* spelling was incorrect

STMT NO MISSING

A FORMAT statement appeared without a statement number

SUBPGM/ARR NAME ILL

Illegal use of subprogram or array name

SUBPROGRAM NAME ILL

Illegal use of subprogram name

SYMBOLIC SUBSCR ILL

Illegal usage of symbolic subscript, in specification statement

SYNTAX ERROR

General syntax error, context usually shows offending character(s)

TOO FEW SUBSCRIPTS

Number of subscripts used in an array is fewer than the number originally declared in a DIMENSION or mode specification statement

UNDECLARED VARIABLE

The listed variable did not appear in a specification statement (occurs only if -DCLVAR option is used)

UNDEFINED STMT NO.

The listed statement number was not defined in the subprogram The listed line number is the line number of the last reference to the statement number

UNRECOGNIZED STMT

The compiler could not identify the statement

WARNING — DEBUG TURNS OFF OPT

Both the -DEBUG and -OPT (or -UNCOPT) options were selected Compilation will proceed as if the optimization option had not been included

WARNING — NO RETURN OR STOP

Either the STOP (main program) or RETURN (subroutine) statement at the end of the program was omitted This message will occur in a subroutine if there is no RETURN statement immediately preceding the END statement regardless of the presence of other RETURNs in the subroutine

WARNING — name — NEVER GIVEN A VALUE

The local variable **name** never had a value assigned to it at any point in the program

WARNING — name — PARAMETER IS BETTER

The variable **name** was initialized in a DATA statement and remains constant in the program It would be more efficient to assign a value with the PARAMETER statement

WARNING — name — VARIABLE NOT USED

The variable **name** was declared but not used in the program Such variables are not accessible when using the high-level language debugger (DBG)

FORTRAN MATHEMATICAL LIBRARY

Data modes

C	COMPLEX
I	INTEGER (either short or long)
I*2	INTEGER*2 (short)
I*4	INTEGER*4 (long)
R*4	REAL*4 (REAL)
R*8	REAL*8 (DOUBLE PRECISION)

Prefixes

n	Any number of arguments
2	Two arguments
3	Three arguments

Data Mode

Name	Argument(s) passed	Value returned	Description
ABS	**R*4**	**R*4**	Absolute value
AIMAG	**C**	**R*4**	Converts imaginary part of argument to REAL*4
AINT	**R*4**	**R*4**	Truncates to whole number
ALOG	**R*4**	**R*4**	Natural logarithm
ALOG10	**R*4**	**R*4**	Base-10 logarithm
AMAX0	**nI**	**R*4**	Maximum value of list
AMAX1	**nR*4**	**R*4**	Maximum value of list
AMIN0	**nI**	**R*4**	Minimum value of list
AMIN1	**nR*4**	**R*4**	Minimum value of list
AMOD	**2R*4**	**R*4**	Remainder when first is divided by second
AND	**nI**	**I**	Logical AND of arguments
ATAN	**R*4**	**R*4**	Arctangent (principal value)
ATAN2	**2R*4**	**R*4**	Arctangent (principal value) of first divided by second
CABS	**C**	**R*4**	Absolute value
CCOS	**C**	**C**	Cosine
CEXP	**C**	**C**	Exponential
CLOG	**C**	**C**	Natural logarithm
CMPLX	**2R*4**	**C**	Converts 2 REAL*4 numbers to 1 COMPLEX number
CONJG	**C**	**C**	Complex conjugate
COS	**R*4**	**R*4**	Cosine
CSIN	**C**	**C**	Sine
CSQRT	**C**	**C**	Square root
DABS	**R*8**	**R*8**	Absolute value
DATAN	**R*8**	**R*8**	Arctangent (principal value)

DATAN	**2R*8**	**R*8**	Arctangent (principal value) of first divided by second
DBLE	**R*4**	**R*8**	Converts REAL*4 to REAL*8
DCOS	**R*8**	**R*8**	Cosine
DEXP	**R*8**	**R*8**	Exponential
DIM	**2R*4**	**R*4**	Positive difference
DINT	**R*8**	**R*8**	Truncates to whole number
DLOG	**R*8**	**R*8**	Natural logarithm
DLOG2	**R*8**	**R*8**	Base-2 logarithm
DLOG10	**R*8**	**R*8**	Base-10 logarithm
DMAX1	**nR*8**	**R*8**	Maximum value of list
DMIN1	**nR*8**	**R*8**	Minimum value of list
DMOD	**2R*8**	**R*8**	Remainder when first is divided by second
DSIGN	**2R*8**	**R*8**	Magnitude of first, sign of second
DSIN	**R*8**	**R*8**	Sine
DSQRT	**R*8**	**R*8**	Square root
EXP	**R*4**	**R*4**	Exponential
FLOAT	**I**	**R*4**	Converts INTEGER to REAL*4
IABS	**I**	**I**	Absolute value
IDIM	**2I**	**I**	Positive difference
IDINT	**R*8**	**I**	Converts REAL*8 to INTEGER
IFIX	**R*4**	**I**	Converts REAL*4 to INTEGER
INT	**R*4**	**I**	Converts REAL*4 to INTEGER
INTL	**I**	**I*4**	Converts INTEGER to INTEGER*4
INTS	**I**	**I*2**	Converts INTEGER to INTEGER*2
IRND	**I**	**I**	Integer random number generator >0 Initialize and return argument =0 Return random number <0 Initialize and return random number
ISIGN	**2I**	**I**	Magnitude of first, sign of second
LOC	**operand**	**I*2**	Location of operand (R-mode)
LOC	**operand**	**I*4**	Location of operand (V-mode)
LS	**2I**	**I**	Left shift
LT	**2I**	**I**	Left truncate
MAX0	**nI**	**I**	Maximum value of list
MAX1	**nR*4**	**I**	Maximum value of list
MIN0	**nI**	**I**	Minimum value of list
MIN1	**nR*4**	**I**	Minimum value of list

MOD	**2I**	**I**	Remainder when first is divided by second
NOT	**I**	**I**	Logical NOT
OR	**2nI**	**I**	Logical OR
REAL	**C**	**R*4**	Converts real part of argument to REAL*4
RND	**I**	**R*4**	Initializes real random number generator >0 Initialize and return FLOATed argument =0 Return random number <0 Initialize and return random number
RS	**2I**	**I**	Right shift
RT	**2I**	**I**	Right truncate
SHFT	**2I**	**I**	Right/left shift
	3I	**I**	Two shift operations
SIGN	**2R*4**	**R*4**	Magnitude of first, sign of second
SIN	**R*4**	**R*4**	Sine
SNGL	**R*8**	**R*4**	Converts REAL*8 to REAL*4
SQRT	**R*4**	**R*4**	Square root
TANH	**R*4**	**R*4**	Hyperbolic tangent
XOR	**nI**	**I**	Logical XOR (exclusive OR)

LOAD MAP OPTIONS — SEG

0 or omitted	Full map
1	Extent map only
2	Extent map and base areas
3	Undefined symbols address order
4	Full map (same as 0)
5	System programmer's map
6	Undefined symbols alphabetic order
7	Full map sorted alphabetically
10	All symbols address order
11	All symbols, alphabetical order

LOAD MAP OPTIONS — LOAD

0 or omitted	Full map
1	Load state
2	Load state and base areas
3	Undefined symbols address order
4	Full map (same as 0)
5	System programmer's map
6	Undefined symbols alphabetical order
7	Full map all symbols alphabetical order
10	Special symbol map for PSD (must be written to file)

ASCII CHARACTER SET

The standard character set used by Prime is the ANSI, ASCII 7-bit set

PRIME USAGE

Prime hardware and software uses standard ASCII for communications with devices The following points are particularly important to Prime usage

- Output Parity is normally transmitted as a zero (space) unless the device requires otherwise, in which case software will compute transmitted parity Some controllers may have hardware to assist in parity generations
- Input Parity is ignored by hardware and by standard software Input drivers are responsible for making the parity bit suit the host software requirements Some controllers may assist in parity error detection
- The Prime internal standard for the parity bit is one, i e , '200 is ORed to the octal value

KEYBOARD INPUT

In the Editor, non-printing characters may be entered into text with the logical escape character ^ and the octal value The character is interpreted by output devices according to their hardware

Example Typing ^207 will enter one character into the text

CTRL-P	('220)	is interpreted as a BREAK
.CR.	('215)	is interpreted as a newline (NL)
"	('242)	is interpreted as a character erase
?	('277)	is interpreted as a line kill
\	('334)	is interpreted as a logical tab (Editor)

Printer control

First character of each ASCII output record controls number of vertical spaces to be printed before printing the new line For space, 0 -, 1 and + the control character is not printed (Used with -FTN option of the spooler)

Space	1 line
0	2 lines
-	3 lines
1	Form feed
+	No advance (overprint)
Other	1 line

ASCII Character Set (Nonprinting)

Octal Value	ASCII Character	Comments/Prime Usage	Control Characters
200	**NULL**	Null character — filler	^@
201	**SOH**	Start of header (communications)	^A
202	**STX**	Start of text (communications)	^B
203	**ETX**	End of text (communications)	^C
204	**EOT**	End of transmission (communications)	^D
205	**ENQ**	End of I D (communications)	^E
206	**ACK**	Acknowledge affirmative (communications)	^F
207	**BEL**	Audible alarm (bell)	^G
210	**BS**	Back space one position (carriage control)	^H
211	**HT**	Physical horizontal tab	^I
212	**LF**	Line feed ignored as terminal input	^J
213	**VT**	Physical vertical tab (carriage control)	^K
214	**FF**	Form feed (carriage control)	^L
215	**CR**	Carriage return (carriage control) *(1)*	^M
216	**SO**	RRS — red ribbon shift	^N
217	**SI**	BRS — black ribbon shift	^O
220	**DLE**	RCP — relative copy *(2)*	^P
221	**DC1**	RHT — relative horizontal tab *(3)*	^Q
222	**DC2**	HLF — half line feed forward (carriage control)	R
223	**DC3**	RVT — relative vertical tab *(4)*	^S
224	**DC4**	HLR — half line feed reverse (carriage control)	^T
225	**NAK**	Negative acknowledgement (communications)	^U
226	**SYN**	Synchronicity (communications)	^V
227	**ETB**	End of transmission block (communications)	^W
230	**CAN**	Cancel	^X
231	**EM**	End of Medium	^Y
232	**SUB**	Substitute	^Z
233	**ESC**	Escape	^[
234	**FS**	File separator	^\
235	**GS**	Group separator	^]
236	**RS**	Record separator	^^
237	**US**	Unit separator	^_

Notes

1 Interpreted as NL at the terminal
2 BREAK at terminal Relative copy in file next byte specifies number of bytes to copy from corresponding position of preceding line
3 Next byte specifies number of spaces to insert
4 Next byte specifies number of line feeds to insert

Conforms to ANSI X3.4-1968

The parity bit ('200) has been added for Prime usage

Nonprinting characters (^C) can be entered at most terminals by typing the (CONTROL) key and the C character key simultaneously

ASCII Character Set (Printing)

Octal Value	ASCII Character	Octal Value	ASCII Character	Octal Value	ASCII Character
240	.SP. *(1)*	300	@	340	` *(9)*
241	!	301	A	341	a
242	" *(2)*	302	B	342	b
243	# *(3)*	303	C	343	c
244	$	304	D	344	d
245	%	305	E	345	e
246	&	306	F	346	f
247	' *(4)*	307	G	347	g
250	(	310	H	350	h
251	)	311	I	351	i
252	*	312	J	352	j
253	+	313	K	353	k
254	, *(5)*	314	L	354	l
255	-	315	M	355	m
256	.	316	N	356	n
257	/	317	O	357	o
260	0	320	P	360	p
261	1	321	Q	361	q
262	2	322	R	362	r
263	3	323	S	363	s
264	4	324	T	364	t
265	5	325	U	365	u
266	6	326	V	366	v
267	7	327	W	367	w
270	8	330	X	370	x
271	9	331	Y	371	y
272	:	332	Z	372	z
273	;	333	[	373	[
274	<	334	\	374	|
275	=	335	]	375	]
276	>	336	^ *(7)*	376	~ *(10)*
277	? *(6)*	337	– *(8)*	377	DEL *(11)*

Notes

1 Space forward one position
2 Terminal usage — erase previous character
3 £ in British use
4 Apostrophe/single quote
5 Comma
6 Terminal usage — kill line
7 1963 standard ↑; terminal use — logical escape
8 1963 standard ←
9 Grave
10 1963 standard ESC
11 Rubout — ignored

Conforms to ANSI X3.4-1968
1963 variances are noted

The parity bit ('200) has been added for Prime usage.

POWERS OF 2

2^n	n	2^{-n}
1	0	1
2	1	.5
4	2	.25
8	3	.125
16	4	.0625
32	5	.03125
64	6	.015625
128	7	.0078125
256	8	.00390625
512	9	.001953125
1024	10	.0009765625
2048	11	.00048828125
4096	12	.000244140625
8192	13	.0001220703125
16384	14	.00006103515625
32768	15	.000030517578125
65536	16	.0000152587890625
131072	17	.00000762939453125
262144	18	.00000381469726562 5
524288	19	.000001907348632812 5
1048576	20	.0000009536743164062 5
2097152	21	.0000004768371582031 25
4194304	22	.0000002384185791015625
8388608	23	.0000001192092895507812 5
16777216	24	.0000000596046447753906 25
33554432	25	.0000000298023223876953125
67108864	26	.0000000149011611938476562 5
134217728	27	.0000000074505805969238281 25
268435456	28	.0000000037252902984619140 625
536870912	29	.0000000018626451492309570 3125
1073741824	30	.0000000009313225746154785 15625
2147483648	31	.0000000004656612873077392 57812 5
4294967296	32	.0000000002328306436538696 28906 25