

" . . ."ASCH STRING (FIXED)	9-36
" . . ."ASCH STRING (FIXED)	9-37
/ RECORD TERMINATOR	9-41
ACCEPT STATEMENTS	8-4
ACCEPT-SIDPLAY	9-47
ARITHMETIC EXPRESSIONS	3-1
ARITHMETIC IF	5-4
ARRAY DECLARATORS	4-1
ARRAYS	2-11
ASSIGNED GO TO	5-2
ASSIGNMENT STATEMENTS	3-9
AUXILIARY INPUT/OUTPUT STATEMENTS	8-11
AW LEFTMOST ASCH CHARACTERS	9-26
BASIC EXTERNAL FUNCTIONS	7-6
BLOCK DATA SUBPROGRAMS	4-22
BLOCK DATA SUBPROGRAMS	6-7
BREAK STATEMENTS	5-13
CALL STATEMENTS	5-14
CHANGING STANDARD ATTRIBUTES OF FILES	10-2
CHARACTER CONSTANTS	2-8
CHARACTER EXPRESSIONS	3-8
CHARACTER FORMAT	2-4
CHARACTER SET	1-2
CHARACTER VARIABLES AND ARRAYS IN COMMON BLOCKS	4-7
COMMENT LINES	1-3
COMMON STATEMENTS	4-4
COMPILER SUBSYSTEM COMMANDS	11-1
COMPLEX CONSTANTS	2-7
COMPLEX FORMAT	2-3
COMPOSITE NUMBERS	2-9
COMPUTED GO TO	5-1
CONSTANTS	2-5
CONTINUE STATEMENTS	5-13
CONTROL COMMAND	11-2
CONTROL RECORDS	1-3
CONTROL STATEMENTS	5-1
CORE-TO-CORE CONVERSION	9-47
CORRESPONDENCE OF COMMON BLOCKS	4-5
DATA ELEMENTS	2-1
DATA STATEMENTS	4-18
DATA STORAGE FORMATS	2-1
DIMENSION STATEMENTS	4-3
DIRECT INTRINSIC CALLS	10-2
DISC INPUT/OUTPUT	9-2
DISPLAY STATEMENTS	8-7
DO STATEMENTS	5-4
DOUBLE PRECISION REAL CONSTANTS	2-6
DOUBLE PRECISION REAL FORMAT	2-3
DUMMY AND ACTUAL ARGUMENT CHARACTERISTICS	2-15
DW.D DOUBLE PRECISION NUMBERS	9-7
EDIT COMMAND	11-6
EDIT DESCRIPTORS	9-35
EDIT SPECIFICATIONS	9-35
END LINES	6-2
ENTERING AND EXITING DO LOOPS	5-8
EQUIVALENCE BETWEEN ARRAYS OF DIFFERENT DIMENSIONS	4-11
EQUIVALENCE IN COMMON BLOCKS	4-14
EQUIVALENCE IN DATA STATEMENTS	4-20
EQUIVALENCE OF ARRAY ELEMENTS	4-10




EQUIVALENC OF DIFFERENT TYPES	4-9
ERROR AND WARNING MESSAGES	D-1
ERROR MESSAGES	D-1
ERROR POSITION INDICATION	D-1
EW.D REAL NUMBERS	9-9
EXPRESSIONS	3-1
EXPRESSIONS AND ASSIGNMENT STATEMENTS	3-1
EXTERNAL STATEMENTS	4-18
FIELD DESCRIPTORS	9-3
FIXED FIELD FORMAT	1-3
FORMAT SPECIFICATIONS	9-3
FORMAT STATEMENTS	8-11
FORMAT STATEMENTS	9-1
FORTRAN FILE FACILITY	10-1
FORTRAN/3000 AND ANSI STANDARD FORTRAN	8-1
FORTRAN/3020 AND PREVIOUS VERSIONS OF HP FORTRAN	C-1
FREE FIELD FORMAT	1-4
FREE-FIELD CONTROL CHARACTERS	9-44
FREE-FIELD INPUT	9-44
FREE-FIELD INPUT/OUTPUT	9-43
FREE-FIELD OUTPUT	9-46
FUNCTION REFERENCES	2-12
FUNCTION SUBPROGRAM REFERENCE	2-13
FUNCTION SUBPROGRAMS	6-5
FUNCTION SUBPROGRAMS	7-6
FUNCTIONS	7-1
FW.D REAL NUMBERS	9-11
GO TO STATEMENTS	5-1
GW.D REAL NUMBERS	9-13
IF COMMAND	11-5
IF STATEMENTS	5-3
IMPLICIT STATEMENT	4-17
INPUT/OUTPUT STATEMENTS	8-1
INTEGER CONSTANTS	2-5
INTEGER FORMAT	2-1
INTRINSIC FUNCTIONS	2-15
INTRINSIC FUNCTIONS	7-1
INTRODUCTION	1-1
IW INTEGER NUMBERS	9-20
LABEL ASSIGNMENT STATEMENTS	3-10
LINES	1-2
LOGICAL CONSTANTS	2-7
LOGICAL EXPRESSIONS	3-4
LOGICAL FORMAT	2-1
LOGICAL IF	5-4
LW LOGICAL (BOOLEAN) VALUES	9-24
MAIN PROGRAMS	6-3
MAIN PROGRAMS AND SUBPROGRAMS	6-1
MW.D REAL NUMBERS	9-16
NESTING	9-42
NESTING DO LOOPS	5-6
NH ASCH STRING (VARIABLE)	9-38
NON-FORTRAN LANGUAGE SUBPROGRAMS	6-8
NON-FURTRAN PROGRAM UNITS	A-1
NUMBER RANGES	2-17
NW.D REAL NUMBERS	9-18
NX ASCH BLANKS	9-39
OW OCTAL INTEGER NUMBERS	9-22
PAGE COMMAND	11-4

PROGRAM ELEMENTS	1-2
PROGRAM FORMAT	1-3
RANGE AND EXECUTION OF DO LOOPS	5-5
READ OR WRITE STATEMENTS	9-2
READ STATEMENTS	8-1
REAL (FLOATING POINT) CONSTANTS	2-6
REAL (FLOATING POINT) FORMAT	2-2
REPEAT SPECIFICATIONS--EDIT DESCRIPTORS	9-42
REPEAT SPECIFICATIONS--FIELD DESCRIPTORS	9-35
RETURN STATEMENTS	5-15
RULES FOR INPUT	9-4
RW RIGHTMOST ASCH CHARACTERS	9-28
S STRINGS OF ASCH CHARACTERS	9-30
SCALE FACTOR	9-32
SET COMMAND	11-5
SIMPLE VARIABLES	2-11
SPECIFICATION INTERRELATIONSHIPS	9-42
SPECIFICATIONS STATEMENTS	4-1
SPL/3000 PROGRAMS	A-1
STANDARD INPUT AND LIST FILES	10-1
STATEMENT FUNCTION REFERENCE	2-13
STATEMENT FUNCTIONS	4-22
STATEMENT FUNCTIONS	7-5
STATEMENT LABELS	1-3
STATEMENT ORDER IN PROGRAM UNITS	6-1
STATEMENTS	1-2
SUBROUTINE SUBPROGRAMS	6-3
SUBSCRIPTS	2-11
SYSTEM INTRINSICS	A-2
THE FORMATTER	9-1
TITLE COMMAND	11-5
TN POSITION (TABULATE) DATA	9-40
TRACE COMMAND	11-8
TYPE STATEMENTS	4-16
UNCONDITIONAL GO TO	5-1
UNFORMATTED (BINARY) TRANSFER	9-48
UNLIMITED GROUPS	9-43
VARIABLES	2-11
WARNING MESSAGES	D-1
WRITE STATEMENT EXECUTION	8-6
WRITE STATEMENTS	8-5

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HEWLETT  PACKARD

DATA SYSTEMS • 11000 Wolfe Road, Cupertino, California 95014, Telephone 408-257-7000

FROM Dick Slegt

DATE March 21, 1973

TO Software Training Dev.

SUBJECT:

The attached FORTRAN index comes to us courtesy
of Doug Mecham.

DS:lj

S/3000 FORTRAN ALPHABETICAL LIST OF KEY TERMS

"aaa", 'aaa', nH, %mC	character constant	2-8
ACCEPT		8-4
ACCEPT/DISPLAY	terminal dialog	9-47
ACCEPT statements		8-4
allocate local storage; initialize	DATA statements	4-18
alphanumeric characters, leftmost	Aw	9-26
alphanumeric characters, rightmost	Rw	9-28
arguments		2-15
arithmetic condition		5-4
arithmetic expressions	numeric equations	3-1
arithmetic IF		5-4
arithmetic operators	+ - * / ** partial-word designator	3-1
array declarators	fixed array	4-1
	variable array	4-1
array variables	any array type	2-11
array type	array variables	2-11
arrays		2-11
assigned branch		5-2
assignment statements	equations, any expression/ designator	3-9
ASSIGN statement	assign label value to integer item	3-10
Auxiliary input/output statements		8-11
Aw	alphanumeric characters, leftmost	9-26
Aw leftmost ASCH characters		9-26



BASIC external functions		7-6
BLOCK data	global data allocation and initialization	6-7
block data subprograms		4-22
break statements		5-13
byte	character storage forms	2-4
CALL	subprogram reference	5-14
call statements		5-14
calling SPL program units	SPL programs	A-1
changing standard attributes of files		10-2
character constant	"aaa", 'aaa', nH, %mC	2-8
character constants		2-8
character, data type		4-16
character expressions		3-8
character equations		3-8
character format		2-4
character operator	substring designator	3-8
character set	0-9 A-Z $\beta = + - * / ()$, . \$ " ' [] # % @ :	1-2
character storage forms	8 bits, byte	2-4
character string	nH	9-38
character string	"..."	9-36
character string	'...'	9-37
character string input/output		9-38
character variables and arrays in common blocks		4-7
comment lines	C in column 1 or #	1-3
comment statement		4-4

compilation specification		11-2
compiler action for errors		D-7
compiler commands	compiler control	11-1
compiler control	control records	1-3
compiler control	compiler commands	11-1
compiler predefined functions	intrinsic functions	7-1
compiler-system interface	system command structure	---
complex constant	(real, real)	2-7
complex data type		4-16
complex format		2-3
complex storage forms	2 real numbers	2-3
composite number	bit pattern; %[number of bits/ value]	2-9
computational processes	functions	7-1
computational processes	statement functions	4-22
computational subprogram unit	function	6-5
computed branch		5-1
computed GO TO		5-1
conditional branch	IF statement	5-3
constants		2-5
continue statements	loop termination	5-13
control command		11-2
control records		1-3
control statements		5-1
\$CONTROL	compilation specification	11-2

core-to-core conversion		9-47
correspondence of common blocks		4-5
data area size specification	dimension statement	4-3
data declaration	specification statements	4-1
data elements		2-1
data input	input rules	9-4
data statements	allocate local storage; initialize local storage	4-18
data storage formats		2-1
data storage types	data type statement	4-16
decimal number	integer constant $\pm n$	2-5
definite branch	GO TO statement	5-1
dimension statement	data area size specification	4-3
direct intrinsic calls		10-2
disc input/output		9-2
display statements	terminal output	8-7
DO statements	iteration: looping	5-4
double precision formatter	Dw.d	9-7
double precision constant $\pm n.nD\pm e$	floating point	2-6
double precision data type		4-16
double precision real constants		2-6
double precision real format		2-3
double precision storage	sign +9 bits exponent +38 bits fraction	2-3
Dummy and actual argument characteristics		2-15
Dw.d	double precision, formatter	9-7
Dw.D double precision numbers		9-7



\$EDIT	statement sequence control	11-6
edit command		11-6
edit descriptions	format line control	9-35
edit specifications		9-35
END line	program unit termination	6-2
entering and exiting DO loops		5-8
entry points	not implemented	---
equations	assignment statements	3-9
equivalence	storage overlays	4-9
equivalence between arrays of different dimensions		4-11
equivalence in common blocks		4-14
equivalence in data statements		4-20
equivalence of array elements		4-18
equivalence statement		4-9
error and warning messages		D-1
error code		D-7
error messages		D-1
error position indication		D-1
EW.D real numbers	real with exponent, formatter	9-9
expression statements		3-1
expressions		3-1
expressions and assignment statements		3-1
exiting loops		5-8
external library functions	library defined functions	7-6
external statement	external subprogram reference	3-1/4-18
external subprogram reference	external statement	4-18

field descriptors		4-18
file attributes change		10-2
file facility		10-3
fixed field format		1-3
fixed point, commas		9-18
fixed point, monetary form, \$ & commas; formatter		9-16
floating point	double precision constant $\pm n.nD\pm e$	2-6
floating point	real (constant) $\pm n.nE\pm e$	2-6
format	formatted input/output	8-11
format conversions	input/output format control	9-3
format field descriptors	forms of input/output specification	9-3
format line control	edit descriptions	9-35
format nesting	format specification grouping	9-42
format specification grouping	format nesting	9-42
format specifications		9-3
format specification interrelationship		9-42
format statement	form of format statement statements	9-1
format statements		8-11/9-1
format subprograms	globally defined functions	7-6
format unlimited grouping	unlimited groups in formatter	9-42
formatted input/output	format	8-11
formatter	formatting input/output	9-1
formatting input/output	formatter	9-1
form of format statement statements	format statement	9-1
FORTTRAN file facility		10-1

FORTRAN/3000 and ANSI standard FORTRAN		B-1
free field control characters		9-44
free-field format		1-4
free-field input/output	free field input/output	9-43/9-44
free-field output		9-46
function	computational subprogram unit	6-5
function references	referenced computational processes	2-12
functions	computational processes	7-1
function subprogram references	reference external process	2-13
function subprograms		6-5/7-6
Fw.d	real without exponent, formatter	9-11
Global data area identification	COMMON statement	4-4
globally defined functions	format subprograms	7-6
global subprogram unit	subroutine	6-3
GO TO statement	definite branch	5-1
Gw.d	real with/without exponent, formatter	9-13
\$IF	selective statement compilation specifications	11-5
IF command		11-5
IF statement	conditional branch	5-3
IMPLICIT statement	variable type specification	4-17
implied DO loop	implied iteration/looping	8-11
implied looping	implied DO loop	8-11
initialization	block data	6-7
initialize local storage	Data statements	4-18
input/output format control	format conversions	9-3
input data	read	8-1

input/output statements		8-1
input rules	data input	9-4
integer constant $\pm n$	decimal number	2-5
integer data type		4-16
integer format		2-1
integer storage format	sign + 15 bits magnitude	2-1
intrinsic calls		10-2
intrinsic functions	compiler predefined functions	7-1/2-15
intrinsic references	reference of system defined routines	2-15
introduction		1-1
iteration: looping	DO statement	5-4
iteration looping	implied DO loop	8-11
Iw	decimal integer/integer format	9-20
Iw integer numbers		9-20
Label assignment statements		3-10
Label value	assign statement	3-10
library defined functions	external library functions	7-6
library programs	library routines	---
library routines	library programs	---
lines	1...80 characters	1-2
local program defined functions	statement functions	7-5
logical	16 bits	2-1
logical condition		5-4
logical constant	%m; TRUE, FALSE	2-7
logical equations	logical expressions	3-4
logical data type		4-16

logical expressions	logical equations	3-4
logical value, formatter	Lw	9-24
logical format		2-1
logical IF		5-4
loop termination	CONTINUE	5-13
Lw	logical (Boolean) value, formatter	9-24
Lw logical (Boolean) values	Lw	9-24
main programs		6-1
main programs and subprograms		6-1
main program unit	program	6-3
mixed mode expressions		3-9
MPE intrinsics	system intrinsics	---
Mw.d	fixed point, monetary form, \$ & commas; formatter	9-16
Mw.d real numbers		9-16
nesting		9-42
nesting DO loops		5-6
nesting loops		5-6
nH	character string/input/output	9-38
non-FORTRAN program units		A-1
non-FORTRAN program units using SPL	program units	6-8
number ranges	numeric size for each data type	2-17
numeric equations	arithmetic expressions	3-1
numeric size for each data type	number ranges	2-17
nPf	scale (exponent) factor, formatter	9-32
Nw.d real numbers	fixed point, formatter	9-18
Nx ASCH blanks		9-39

octal constant	%m octal number	2-6
octal integer, formatter	Ow	9-22
output data	write	8-5
Ow	octal integer, formatter	9-22
\$PAGE	page eject + page heading, compiler	11-4
page command		11-4
page eject + page heading compiler	\$PAGE	11-4
page heading; compiler		11-5
partial-word designators	arithmetic operators	3-1
PAUSE	suspend program	5-13
%m; TRUE, FALSE	logical constant	2-7
position (tabulate) data		9-40
program	main program unit	6-3
program elements		1-1
program format		1-3
program operation monitor	\$TRACE	11-8
program unit termination	END line	6-2
program units	non-FORTRAN program units using SPL	6-8
r	repeat specification, formatter	9-42
range and execution of DO loops		5-5
read	input data	8-1
read statements		8-1
read or write statements		9-2
referenced computational processes		2-12
reference external process		2-13
real data type		4-16
real (constant)	±n.nE±e floating point	2-6
real (floating point) constants		2-6

real (floating point) format		2-2
real (storage format)	sign + 9 bits exponent + 22 bits fraction	2-2
real without exponent, formatter		9-11
reference local process	statement function references	2-13
record terminator, formatter	/	9-41
repeat specification--edit descriptors		9-42
repeat specification--field descriptors		9-35
return statements	transfer control from subprogram	5-15
rules for input		9-4
rightmost, formatter	Rw	9-28
Rw	alphanumeric characters, rightmost, formatter	9-28
S	strings of characters, formatter	9-30
S strings of ASCH characters		9-28
scale (exponent) factor		9-32
selective statement compilation specs	\$IF	11-5
selective statement compilation control	\$SET	11-5
\$SET	selective statement compilation control	11-5
SET command		11-5
simple variables	any data type	2-11
slash	record terminator, formatter	9-41
specification interrelationships		9-42
specification statements	data declaration	4-1
specifications statements		4-1
SPL programs	calling SPL program units	A-1

SPL/3000 programs		A-1
standard files (FTNaaa)	standard input and list files	10-1
standard input and list files	standard files (FTNaaa)	10-1
statement categories	statement order in program units	6-1
statement labels	1...99999	1-2/1-3
statement functions	local program defined functions	7-5
statement functions	computational processes	4-22
statement function references	reference local process	2-13
statement order in program units	statement categories	6-1
statements	72 characters/line; up to 20 continuous lines	1-2
statement selection for compilation		11-5
statement sequence control	\$EDIT	11-6
storage overlays	equivalence	4-9
strings of characters, formatter	S	9-30
subprograms		6-1
subroutine	global/local subprogram unit	6-3
subroutine subprograms		6-3
subscripts	maximum = 255	2-11
suspend program	PAUSE	5-13
system command structure	compiler-system interface	---
system intrinsics	using system intrinsics	A-2
system intrinsics	MPE intrinsics	3-9

The formatter		9 1
terminal dialog	accept/display	9-47
terminal output	display	8-7
\$TITLE	page heading;compiler	11-5
title command		11-5
Tn	position (tabulate) data, for matter	9-40
Tn position (tabulate) data		9-40
\$TRACE	program operation monitor	11-8
Trace command		11-8
transfer of control	control statements	5-1
TRUE	logical constant	2-7
transfer control from subprogram	return statements	5-15
type statements		4-16
unconditional branch		5-1
unconditional GO TO		5-1
unformatted (binary) transfer		9-48
unlimited groups		9-43
variable array		4-1
variables	name \leq 15 characters	2-11
warning messages		D-1
write	output data	8-5
write statement execution		8-6
write statements		8-5
"..."	character string	9-36
'...'	character string	9-37

COMMANDS ALPHA LISTING

ABORT	3-19	\$NEWPASS	4-3
BASIC	4-5	\$NEWPASS	5-6
BASIC	5-22	\$NULL	4-2
BREAK	8-47	\$NULL	5-5
BUILD	5-24	\$OLD PASS	4-4
BYE	3-20	\$OLDPASS	5-7
CONTINUE	3-23	\$STDIN	4-2
DATA	3-21	\$STDIN	5-5
EDITOR	4-14	\$STDINX	4-2
EOD	3-21	\$STDINX	5-5
EOJ	3-23	\$STDLIST	4-2
EOJ	3-13	\$STDLIST	5-5
ESL	8-47	(CONTROL H)	8-47
FILE	5-10	(CONTROL Q)	8-47
FILE	5-11	(CONTROL X)	8-47
FILENAMES	5-28	(CONTROL Y)	8-47
FORMALDESIGNATOR	5-6	*FORMALDESIGNATOR	5-6
FORTRAN	4-7	-ADDRL	7-19
FORTRAN	5-22	-ADDSL	7-22
FREERIN	9-6	-AUXUSL	7-9
GETRIN	9-2	-BUILDRL	7-18
HELLO	3-16	-BUILDSL	7-21
JOB	3-10	-BUILDUSL	7-3
LISTF	5-26	-CEASE	7-6
PREP	4-9	-COPY	7-9
PTAPE	8-57	-EXIT	7-2
PURGE	5-26	-HIDE	7-23
RENAME	5-36	-LISTRL	7-20
RESET	5-22	-LISTSL	7-22
RESTORE	5-34	-LISTUSL	7-10
RESUME	3-19	-NEWSEG	7-8
RUN	4-12	-PREPARE	7-12
SAVE	5-25	-PURGERBM	7-7
SDM	4-15	-PURGERL	7-19
SEGMENTER	4-15	-PURGESL	7-22
SEGMENTER	7-2	-REVEAL	7-23
SHOWTIME	8-3	-RL	7-19
SPEED	8-48	-SL	7-21
SPL	4-7	-USE	7-5
SPL	5-22	-USL	7-3
STAR	4-14		
STORE	5-29		
TELL	8-2		
TELLOP	8-3		

INTRINSIC ALPHA LISTING

ACTIVATE	11-10		GETJCW	8-40
ADJUSTUSLF	8-43		GETLOCRIN	9-6
ALTUSEG	12-7		GETORIGN	11-24
ARITHMETICTRAP	8-30		GETPRIORITY	11-22
ARITRAP	8-24		GETPRIVMODE	14-3
ASCII	8-5		GETPROCID	11-25
BINARY	8-4		GETUSERMODE	14-4
CAUSEBREAK	8-38		INIUSLF	8-43
CHECKDEV	8-41		KILL	11-13
CHRUNUS	8-11		LIBRARYTRAP	8-31
COMMAND	8-22		LOADPROC	7-24
CONTROLYTRAP	8-35		LOCKGLORIN	9-3
CREATE	11-7		LOCKLOCRIN	9-7
DASCII	8-6		MAIL	11-16
DBINARY	8-5		MYCOMMAND	8-18
DLSIZE	8-36		PRINT	8-9
DMOVIN	12-4		PRINTOP	8-10
DMOVOUT	12-6		PROCINFO	11-27
EXPANDUSLF	8-44		PROCTIME	8-11
FATHER	11-24		PTAPE	8-58
FHECK	6-31		QUIT	8-39
FCLUSE	6-13		READ	8-7
FCONTROL	6-35		RECEIVEMAIL	11-19
FCONTROL	8-54	BRFAK	RESETCONTROL	8-29
FCONTROL	8-50	CHANGE SPEED	SEARCH	8-15
FCONTROL	8-53	ECHO	SENDMAIL	11-17
FCONTROL	8-58	ENABLE TERMINAL I/O TIMER	SETJCW	8-40
FCONTROL	8-54	ESCAPE	SUSPEND	11-12
FCONTROL	8-60	LINE-TERMINATION CHARACTER	SWITCHDB	14-5
FCONTROL	8-55	PARITY-CHECKING	SYSTEMTRAP	8-34
FCONTROL	8-59	READ TERMINAL I/O TIMER	TERMINATE	8-39
FCONTROL	8-56	TAPE-MODE	TERMINATE	11-13
FGETINFO	6-27		TIMER	8-10
FLOCK	9-10		UNLOADPROC	7-25
FOPEN	6-4		UNLOCKGLORIN	9-4
FPOINT	6-26		UNLOCKLOCRIN	9-8
FREAD	6-15		WHO	8-12
FREADDIR	6-16		XARITRAP	8-24
FREADSEAK	6-18		XCONTRAP	8-29
FREUSEG	12-3		XLIBTRAP	8-26
FRELUCRIN	9-9		XSYSTRAP	8-27
FRELATE	6-41		ZSIZE	8-37
FRENAME	6-40			
FSETMODE	6-38			
FSPACE	6-25			
FUNLOCK	9-11			
FUPDATE	6-24			
FWRITE	6-19			
FWRITEDIR	6-22			
GETUSEG	12-2			

MPE COMMANDS

MPE MULTIPROGRAMMING EXECUTIVE OPERATING
SYSTEM 03000-90005

MSC MULTIPROGRAMMING EXECUTIVE SYSTEM
MANAGER/SUPERVISOR CAPABILITIES
03000-90038

MCO MULTIPROGRAMMING EXECUTIVE CONSOLE
OPERATOR'S GUIDE
03000-90006

ABORT	MPE	3-19
ABORT	MSC	3-19
ALLOCATE	MSC	7-19
ALLOW	MSC	7-5
ALTACCT	MSC	6-7
ALTGROUP	MSC	6-11
ALTUSER	MSC	6-15
BAR	MSC	7-5
BASIC	MPE	4-5
BASIC	MPE	5-22
BUILD	MPE	5-24
BYE	MPE	3-20
BYE	MSC	3-20
CONTINUE	MPE	3-23
CONTINUE	MSC	3-23
DATA	MPE	3-21
DATA	MSC	3-21
DEALLOCATE	MSC	7-20
EDITOR	MPE	4-14
EOD	MPE	3-21
EOD	MSC	3-13
EOJ	MPE	3-23
EOJ	MPE	3-13
EOJ	MSC	3-13
FILE	MPE	5-10
FILE	MPE	5-11
FILENAMES	MPE	5-28
FORMALDESIGNATOR	MPE	5-6
FORTRAN	MPE	4-7
FORTRAN	MPE	5-22
FREPTN	MPE	9-6
GETRIN	MPE	9-2
HELLO	MPE	3-16
HELLO	MSC	3-16
JOB	MPE	3-10
JOB	MSC	3-10
LISTACCT	MSC	6-8
LISTF	MPE	5-26

LISTGROUP	MSC	3-5
LISTGROUP	MSC	6-12
LISTUSER	MSC	6-16
NEWACCT	MSC	6-4
NEWGROUP	MSC	6-10
NEWUSER	MSC	6-14
PREP	MPE	4-9
PTAPE	MPE	8-57
PURGE	MPE	5-26
PURGEACCT	MSC	6-9
PURGEGROUP	MSC	6-13
PURGEUSER	MSC	6-17
PUTJOB	MSC	7-7
QUANTUM	MSC	7-6
RENAME	MPE	5-36
RESET	MPE	5-22
RESTORE	MPE	5-34
RESTORE	MSC	H-6
RESUME	MPE	3-19
RESUME	MSC	3-19
RUN	MPE	4-12
RUN	MSC	3-5
RUN	MSC	7-9
SAVE	MPE	5-25
SDM	MPE	4-15
SEGMENTER	MPE	4-15
SEGMENTER	MPE	7-2
SHOWQ	MSC	3-5
SHOWQ	MSC	7-8
SHOWTIME	MPE	8-3
SPEED	MPE	8-48
SPL	MPE	4-7
SPL	MPE	5-22
STAR	MPE	4-14
STORE	MPE	5-29
STORE	MSC	H-1
SYSDUMP	ECO	2-6
SYSDUMP	MSC	4-11
TELL	MPE	8-2
TELL	MSC	7-21
TELLOP	MPE	8-3

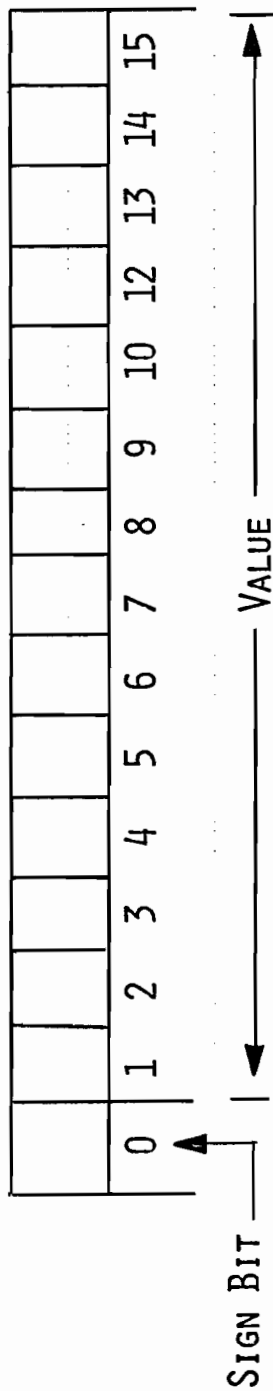
\$NEWPASS	MPE	4-3
\$NEWPASS	MPE	5-6
\$NULL	MPE	4-2
\$NULL	MPE	5-5
\$OLD PASS	MPE	4-4
\$OLDPASS	MPE	5-7
\$STDIN	MPE	4-2
\$STDIN	MPE	5-5
\$STDINX	MPE	4-2
\$STDINX	MPE	5-5
\$STDLIST	MPE	4-2
\$STDLIST	MPE	5-5
(BPEAK)	MPE	8-47
(CONTROL H)	MPE	8-47
(CONTROL Q)	MPE	8-47
(CONTROL X)	MPE	8-47
(CONTROL Y)	MPE	8-47
(ESC)	MPE	8-47
*FORMALDESIGNATOR	MPE	5-6
-ADDRL	MPE	7-19
-ADDSL	MPE	7-22
-AUXUSL	MPE	7-9
-BUILDRL	MPE	7-18
-BUILDUSL	MPE	7-21
-BUILDUUSL	MPE	7-3
-CEASE	MPE	7-6
-COPY	MPE	7-9
-EXIT	MPE	7-2
-HIDE	MPE	7-23
-LISTRL	MPE	7-20
-LISTSL	MPE	7-22
-LISTUSL	MPE	7-10
-NEWSLG	MPE	7-8
-PREPARE	MPE	7-12
-PURGERBM	MPE	7-7
-PURGERL	MPE	7-19
-PURGESL	MPE	7-22
-REVEAL	MPE	7-23
-RL	MPE	7-19
-SL	MPE	7-21
-USE	MPE	7-5
-USL	MPE	7-3
=ABORTJOB	ECO	3-1
=DISPLAYJOB	ECO	4-2
=DOWN	ECO	5-1
=REPLY	ECO	6-3
=SESSION	ECO	8-1
=SHOWQ	ECO	7-1
=SHUTDOWN	ECO	2-6
=SHUTDOWN	MSC	5-15
=TELL	ECO	6-1
=UP	ECO	5-1
=WARN	ECO	6-1
=WHATJOB	ECO	4-1

HP 3000

**fortran
introduction**

DATA TYPES

INTEGER



REAL

S	E								F						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 1

F															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 2

Represents Zero (0) in Memory

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 2

Represents One (1) in Memory



OCTAL CONSTANTS

INTEGER	REAL
%	%
±%	±%
±%"	±%"
±%'	±%'

%4777	%3775R
+%605	%"ABCD"R
-%17	-%"BC"R

%177777
%"AB"
-% 'A'

VARIABLES

15 CHARACTER MAXIMUM

I - N INTEGER

ALL OTHERS REAL

OVERRIDE BY

TYPE

INTEGER

REAL

DOUBLE PRECISION

COMPLEX

LOGICAL

CHARACTER

IMPLICIT

TYPE (LETTER,...,LETTER),...

LETTER

SINGLE A,X,Y

RANGE R-W

ARRAYS

MAX OF 255 DIMENSIONS

DECLARED IN

TYPE

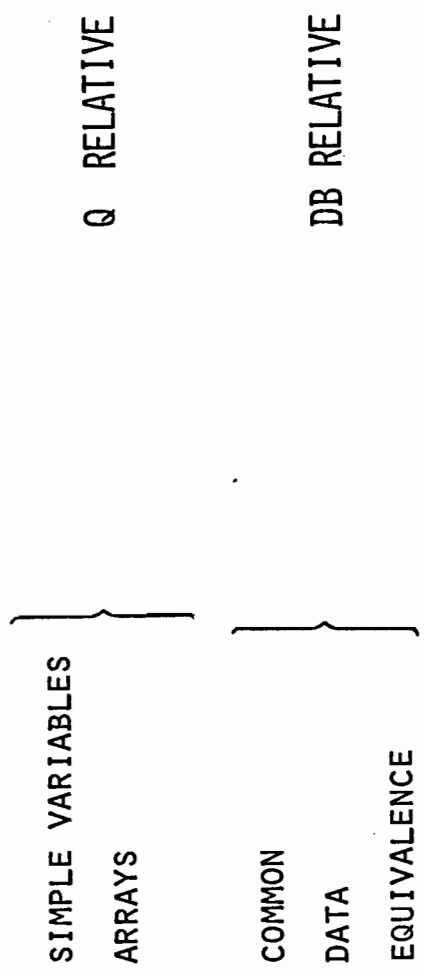
COMMON

DIMENSION

SUBSCRIPTS

EXPRESSION TRUNCATED TO INTEGER

STORAGE



DL

DB

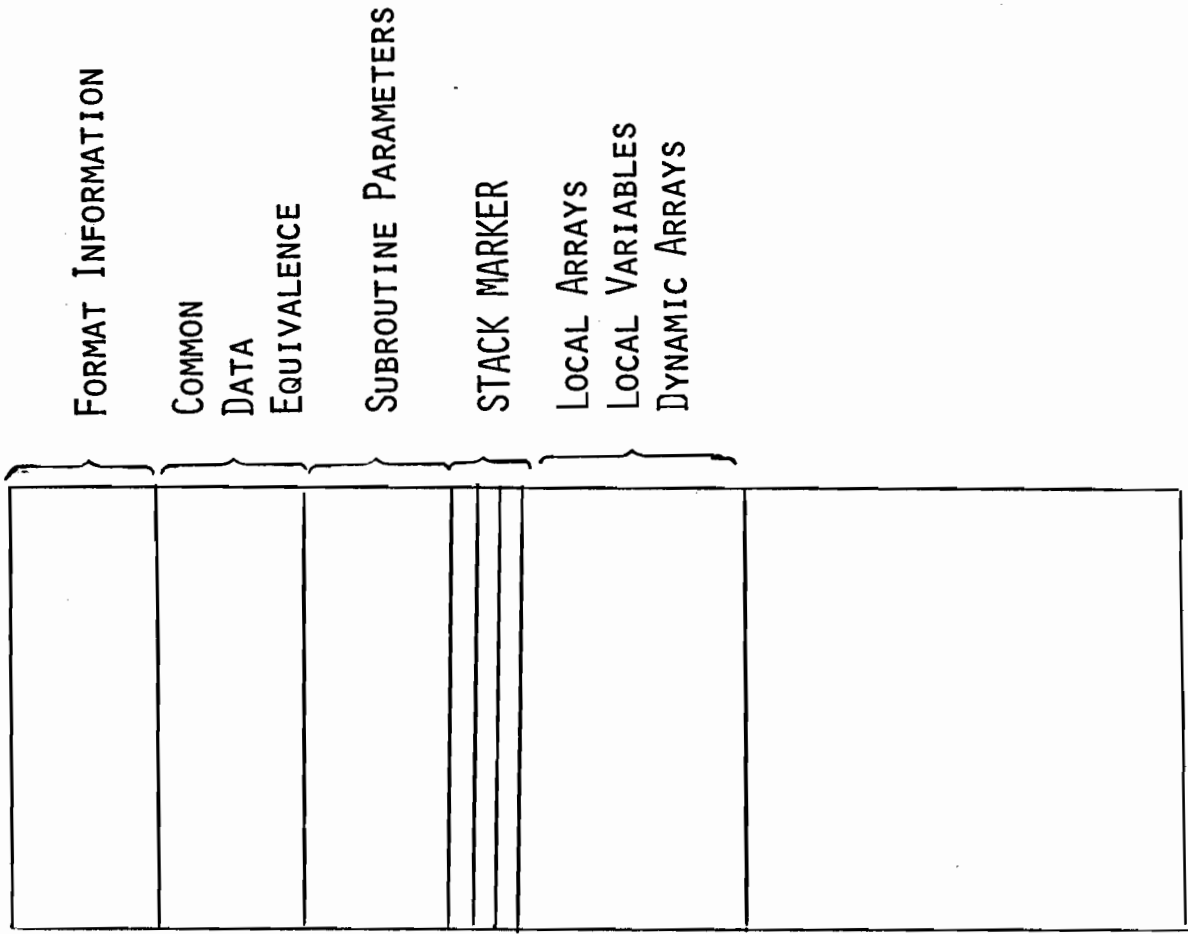
Q1

Q2

Q3

S

Z



FORMAT INFORMATION

COMMON

DATA

EQUIVALENCE

SUBROUTINE PARAMETERS

STACK MARKER

LOCAL ARRAYS

LOCAL VARIABLES

DYNAMIC ARRAYS

COMMON

COMMON /BLOCKNAME/DATAELEMENT, ...,DATAELEMENT/BLOCKNAME/DATAELEMENT...

A6B

BLOCKDATA SUBROUTINE

BLOCKDATA [NAME]

COMMON

IMPLICIT

TYPE

DIMENSION

EQUIVALENCE

DATA

END

ONLY STATEMENTS ALLOWED

HIERARCHY

OPERATORS

**

*/

+-

TYPE

INTEGER

REAL

DOUBLE

COMPLEX

INPUT FORMAT

STATEMENT
1 - 5 LABEL - STATEMENT NUMBER
6 CONTINUATION
7 - 72 STATEMENT
73 - 80 SEQUENCE
1 - C COMMENT
CONTROL RECORD
1 \$
2 - 72 CONTROL OPTIONS
73 - 80 SEQUENCE
& TO CONTINUE ON NEXT CARD
NEXT CARD MUST HAVE \$



I/O OVERVIEW

UNIT NUMBERS

5 CARDREADER OR TERMINAL

6 LINEPRINTER OR TERMINAL

INTEGER VARIABLES

\$CONTROL FILE = NUMBER

READ

WRITE

DISPLAY (PRINT)

ACCEPT

FORMATS:

X, H, '...', "..." , I,E,F

I INPUT FW.Ø

G, T,...

FILE NAMES

WRITE (19,10) A

READ (5,20) B

WRITE (6,30) C

FREE FIELD INPUT

TERMINAL ORIENTATED

\$CONTROL FREE

SEQUENCE FIELD MAX 99999 CORRESPONDS TO 73-80 OF CARD

FIRST ON CARD OR BLANK.

LABEL OR STATEMENT

COMMENT

& CONTINUATION

\$CONTROL LABEL,MAP

FORTRAN EXECUTION

:FORTRAN [TEXT] [USL] [LIST] [MASTER] [NEW]
DEFAULT \$STDIN \$NEWPASS \$STDLIST \$NULL \$NULL
:PREP \$OLDPASS, \$NEWPASS
:RUN \$OLDPASS
:SAVE \$OLDPASS,-----

FORTRAN EXECUTION

:FORTPREP [TEXT] [,PROG] [,LIST] [,MASTER] [,NEW]

DEFAULT \$STDIN \$NEWPASS \$STDLIST \$NULL \$NULL

:RUN \$OLDPASS

:SAVE \$OLDPASS,-----

FORTRAN EXECUTION

:FORTGO [TEXT] [,LIST] [,MASTER] [,NEW]
DEFAULT \$STDIN \$STDLIST \$NULL \$NULL

:SAVE \$OLDPASS,-----

HP 3000

symbol trace

SYMBOL TRACE

PROGRAM IDENTIFIERS

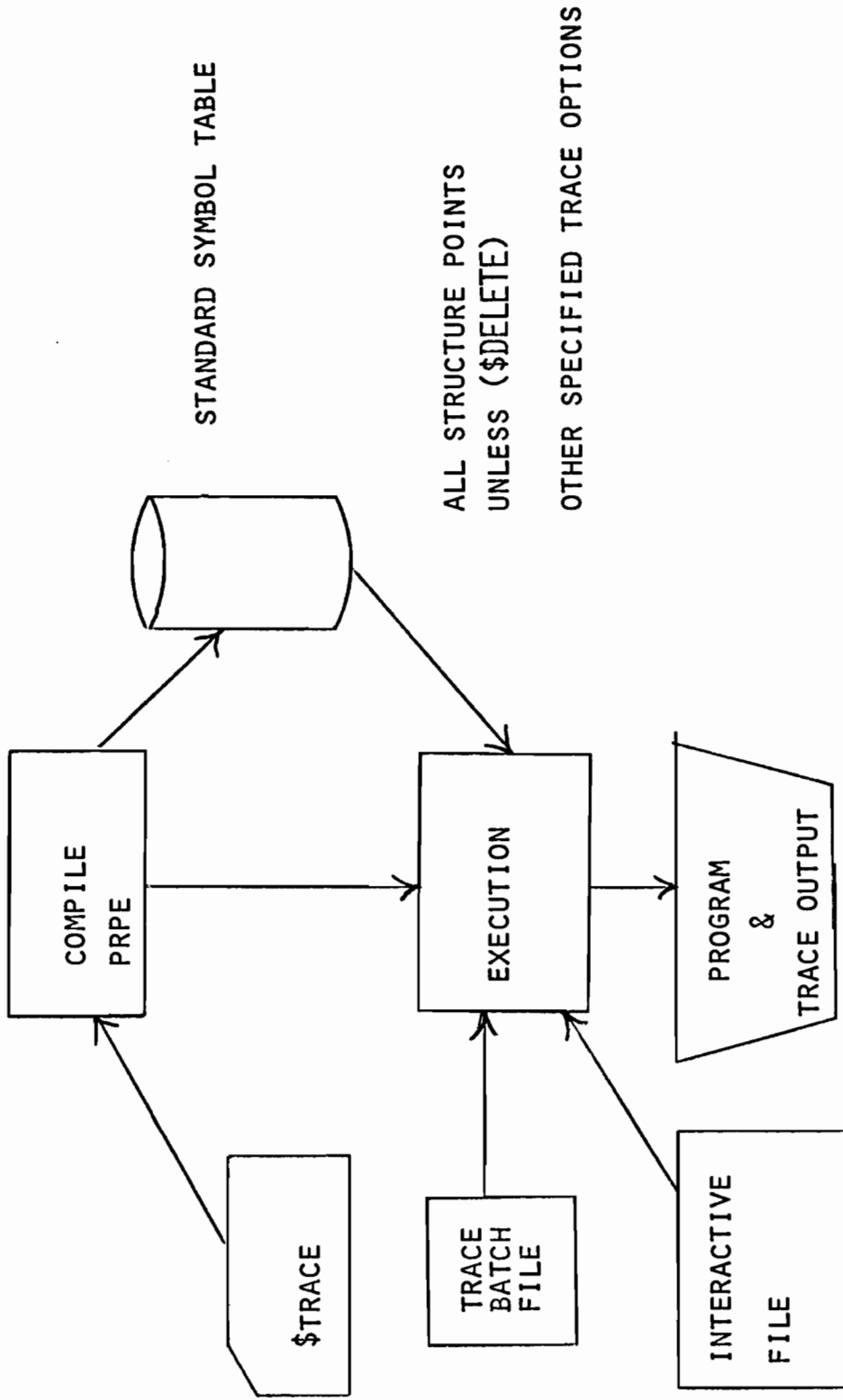
TRACE VARIABLE	FTN
ARRAY	SIMPLE VARIABLE
LABEL	ARRAY
ROUTINE	STATEMENT LABEL
	FUNCTION SUBPROGRAM
	SUBROUTINE SUBPROGRAM
	STATEMENT FUNCTION

STRUCTURE POINTS

CALL
RETURN
ENTER
EXIT

SPECIFY

VARIABLES
ARRAYS
SUBPROGRAMS
LABELS
CONDITIONS



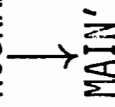
TRACE OPERATION

\$TRACE CONTROL RECORDS

WHERE

HOW

\$TRACE PROGRAM UNIT NAME; IDENTIFIER,...



SUBPROGRAM NAME

BLANK IF NOT FIRST

SAME AS PREVIOUS

- ↓ VARIABLE
- ARRAY
- \$DELETE (1st)
- LABEL
- ROUTINE

TRACE BATCH FILE

\$TRACESTART

[ABORT]



PRINT-HALT TABLE OVERFLOW

OR

ERROR IN PRINT-HALT PARAGRAPH

● PARAGRAPH STRUCTURE

PARAGRAPH-TYPE NAME

SENTENCES

•

SENTENCES

NULL RECORD TERMINATOR

\$TRACEEND

PRINT [PROGRAM NAME]

	C2 ↓ VALUE CONDITION	C3 ↓ LABEL-LABEL	C4 ↓ USE
VARIABLE ↓ VARIABLE NAME			
EQUAL	RELOP ↓ =		@ INTEGER
NOT EQUAL	<>		↓ VARIABLE OR CONSTANT
GREATER THAN	>	* -LABEL OR	
LESS THAN	<	LABEL-*	
GREATER THAN OR EQUAL	>=		
LESS THAN OR EQUAL	<=		

C4

C3

C2

C1

ARRAY

SUBSCRIPT VALUE CONDITION

ARRAY NAME

(*RELOP INTEGER)

VARIABLE OR

CONSTANT

LABEL C4

\$LABEL C3 C4

(ALL STATEMENT NUMBER
IN PROGRAM UNIT)

ROUTINE P1 C3 C4

STRUCTURE POINTS
FOR ROUTINE NAMED

↓
()

INCLUDE PARAMETER VALUES

\$FORM P1 C3 C4

ALL STRUCTURE POINTS

PRINT }
HALT }

ARRAY C1 C2 C3 C4

OR

VARIABLE C2 C3 C4

OR

LABEL C3 C4

OR

\$LABEL C3 C4

OR

ROUTINE P1 C3 C4

OR

\$FORM P1 C3 C4

C1 SUBSCRIPT VALUE CONDITION (*RELOP INTEGER)

C2 IDENTIFIER VALUE CONDITION RELOP PRIMARY

C3 LABEL CONDITION LABEL - LABEL *

C4 USE CONDITIONS @INTEGER

P1 ARGUMENTS ()

INTERACTIVE TRACE

HELLO TRACE

BATCH FILE =

RETURN OR BATCH FILE NAME

MODE = $\left. \begin{array}{l} (N) \\ (R) \end{array} \right\}$

NORMAL

RESTRICTED

*

BYE TRACE

*SET [PROGRAM NAME]

IDENTIFIER = VALUE PRINTED, OPTION CAN / CHANGE VALUE

ARRAY(10), NUMBER WANTED

*-SET MAIN'

VALUE = 36 / 34

VALUE = 34

ARRAY (1), 10 = -10 1000 55 778 932
 20 30 40 50 60

STRINGS ""=""

ON ERROR DETECTION RETURNS*

* GO [LABEL]

* DROPALL

TO DELETE ENTIRE PRINT/HALT TABLE

*DROP [PROGRAM]

\$ALL

TO DELETE ALL SENTENCES FOR THIS PROGRAM UNIT

*DROP [PROGRAM]

IDENTIFIER

;

TO DELETE SENTENCES RELATED TO IDENTIFIER IN THIS PROGRAM UNIT

SINGLE LETTER ABBREVIATIONS

NAMES IN ALPHANUMERIC ORDER

FIRST ENCOUNTERED ENTRY WITH SPECIFIED LETTER IS USED.

*CHECK M
MAIN'

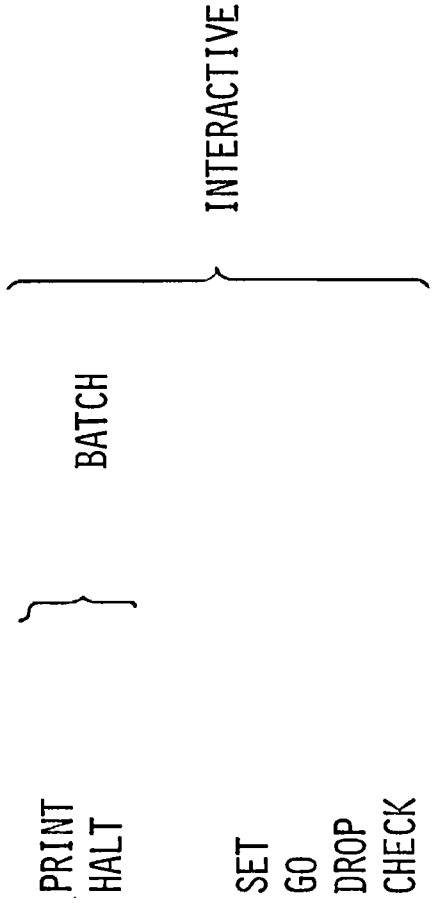
*CHECK S
SUB1

*CHECK PROGRAM

*CHECK
SUB1

S = SAM
J = JAKE
I = IN

TRACE PARAGRAPHS



HP 3000

fortran
type character

CHARACTER DATA

CHARACTER FORMAT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

CHARACTER CONSTANTS

"ABC""XYZ" 'AB"CD' "AB'CD"

%101C

HOLLERITH = ASCII = STRING

19HBLANKS ARE INCLUDED BLANKS ARE INCLUDED "BLANKS ARE INCLUDED"
7HAB"CDFG AB"CDFG "AD""CDFG"

T Y P E

CHARACTER C1,C2*2

CHARACTER *5 Z,X(10),Q*2

CHARACTER *5 P(6,3),R*3(5)

CHARACTER CFUNCT*4,F*3(5,2)



CHARACTER ASSIGNMENTS & EXPRESSIONS

CHARACTER A * 5, B

B = "B"

A = B

A = "C"

A = "BCDE"

B = A

(A.LE.B)

SUBSTRING DESIGNATORS

NAME [FIRST CHARACTER:NUMBER OF CHARACTERS]

CHARACTER*5 A,X

I=2

A= "ABCDE"

X= A[2:3] = BC~~DE~~

X= A[4] = DE~~ABC~~

X= A [1:1] = B~~CDE~~

X [3:1] = A[2:1] =

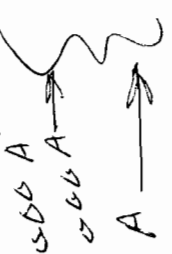
CHARACTER FORMATS

INPUT LEADING SPACES

SAVED →

LOST →

S - FIELD LENGTH FROM DECLARATION OF THAT VARIABLE



RW - RIGHT JUSTIFIED IN FIELD



HP 3000 FORTRAN SUBPROGRAMS

STATEMENT FUNCTIONS

FUNCTION SUBROUTINES

SUBROUTINES

FORTRAN INTRINSIC FUNCTIONS

BASIC EXTERNAL FUNCTIONS

USER

COMPILED

STATEMENT FUNCTIONS

LOCAL

ARGUMENTS

CONSTANTS

SIMPLE VARIABLES

ARRAY ELEMENTS

FUNCTION REFERENCES

EXPRESSIONS ABOVE ONLY

$F(A,B,C) = A*B + C$

FUNCTIONS AND SUBROUTINES

~~GLOBAL~~

ARGUMENTS

VARIABLE NAME

ARRAY NAME

ARRAY ELEMENT

HOLLERITH CONSTANTS

EXPRESSIONS

ARITHMETIC OR LOGICAL EXPRESSION

BOUNDED BY BACK SLASHES (\)

SUBROUTINE

STATEMENT # REF.

SUBROUTINE NAME (PARM, PARM, ..., PARM, * , ... , *)

OPTIONAL
MULTIPLE
RETURNS

OPTIONAL

RETURN point

A

CALL NAME(PARM, PARM, ..., \$LABEL, ..., \$LABEL)

B

OPTIONAL

MULTIPLE RETURNS

OPTIONAL

POSITION of RETURN program

RETURN N

OPTIONAL MULTIPLE RETURNS

FUNCTIONS

TYPE FUNCTION NAME (PARAM, PARAM, ..., PARAM)



OPTIONAL

MUST

HAVE

1 PARAMETER



FORTRAN INTRINSICS FUNCTIONS

TABLE 7-1

<i>INT.</i>	INUM	}	<i>CHAR STRING TO #'S</i>
<i>REAL</i>	RNUM		
<i>DOUBLE PRE.</i>	DNUM		
	STR	CHAR	<i>TO STRING</i>

ALSO

MAX

MIN

TYPE CHANGE

INTRINSIC FUNCTION

INDEX (A1,A2)

↑ EXACT MATCH FROM LEFT
↓ TOTAL STRING

C1 123456789
ABCDEFghi
↑

C2 DEFG
1234

↓ C2 COMPARE AGAINST C1

- I = INDEX (C1,C2) I=4
- I = INDEX (C1[4],C2) I=1
- I = INDEX (C1[2],C2) I = 3
- I = INDEX (C1[6],C2) I=0

BASIC EXTERNAL FUNCTIONS

PASSABLE

TABLE 7-2

I.E. LOGARITHMIC
TRIGONOMETRIC

LOCAL ARRAYS
 DYNAMIC ARRAY (N, I)
 ARE XYZ (N, I)
 SUBROUTINE ARRAY
 DIMENSION COMMON
 SAVE TO USE
 TRM

GO TO

GO TO LABEL

COMPUTED

1 ← 2 → 3 →

GO TO (LABEL, LABEL, ..., LABEL), INDEX EXPRESSION

ASSIGNED

ASSIGN LABEL TO VARIABLE

GO TO VARIABLE, (LABEL, LABEL, ..., LABEL)

NOT USED

\$CONTROL BOUNDS

dynamic bounds
 violation →

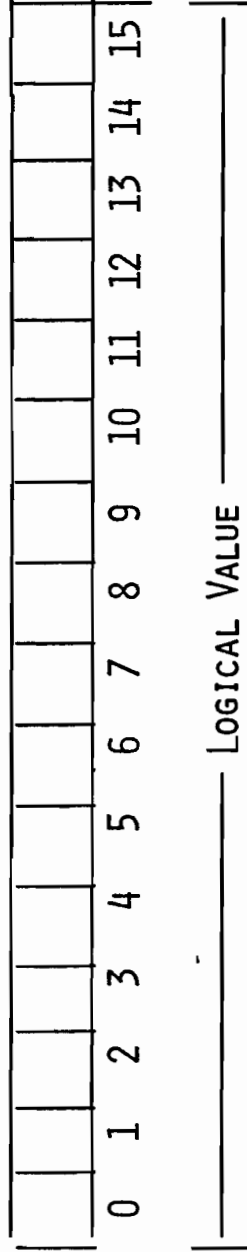
Check subscripts
 but take much time
 to process

HP 3000

**fortran
logical
capabilities**

BIT MANIPULATION

LOGICAL FORMAT



LOGICAL CONSTANTS

.TRUE. = %177777

.FALSE. = %000000

TEST BIT 15 ONLY = WHEN TEST PERFORMED

%177777L

%"AB"L

%1006L

LOGICAL OPERATORS

16 BIT LOGICAL

.NOT. COMPLEMENT
.AND. AND
.XOR. EXCLUSIVE OR
.OR. INCLUSIVE OR

← HIGH ORDER
ALSO

COMPOSITE NUMBERS

% [BIT PATTERN]

LETTER

- BLANK
L
R
D

- INTEGER
LOGICAL
REAL
DOUBLE

Handwritten: REAL → LOGICAL → INTEGER

Handwritten: A1/N1, A2/N2, ..., AN/NN

% [4/15, 6/13, 2/1 R]

Handwritten: 4/15, 6/13, 2/1

Handwritten: LOGICAL

PARTIAL WORD DESIGNATORS

PRIMARY [FIRST BIT :NUMBER OF BITS]

↑
1W
R
C

161111

Bit 0 1 2 3

LOGICAL A,B,C

A = %1400L

B = A[6:2]

C = .TRUE.

C[3:2] = .FALSE.

B = A [1:1] → wrap around
a partial word is
PART of
WORD. C5

FORMATS

LW LOGICAL NOT ON F AND RIGHT JUSTIFIED

OW OCTAL

\$EDIT

*overrides actual
card sequence*

SEQNUM =

INC=

NOSEQ

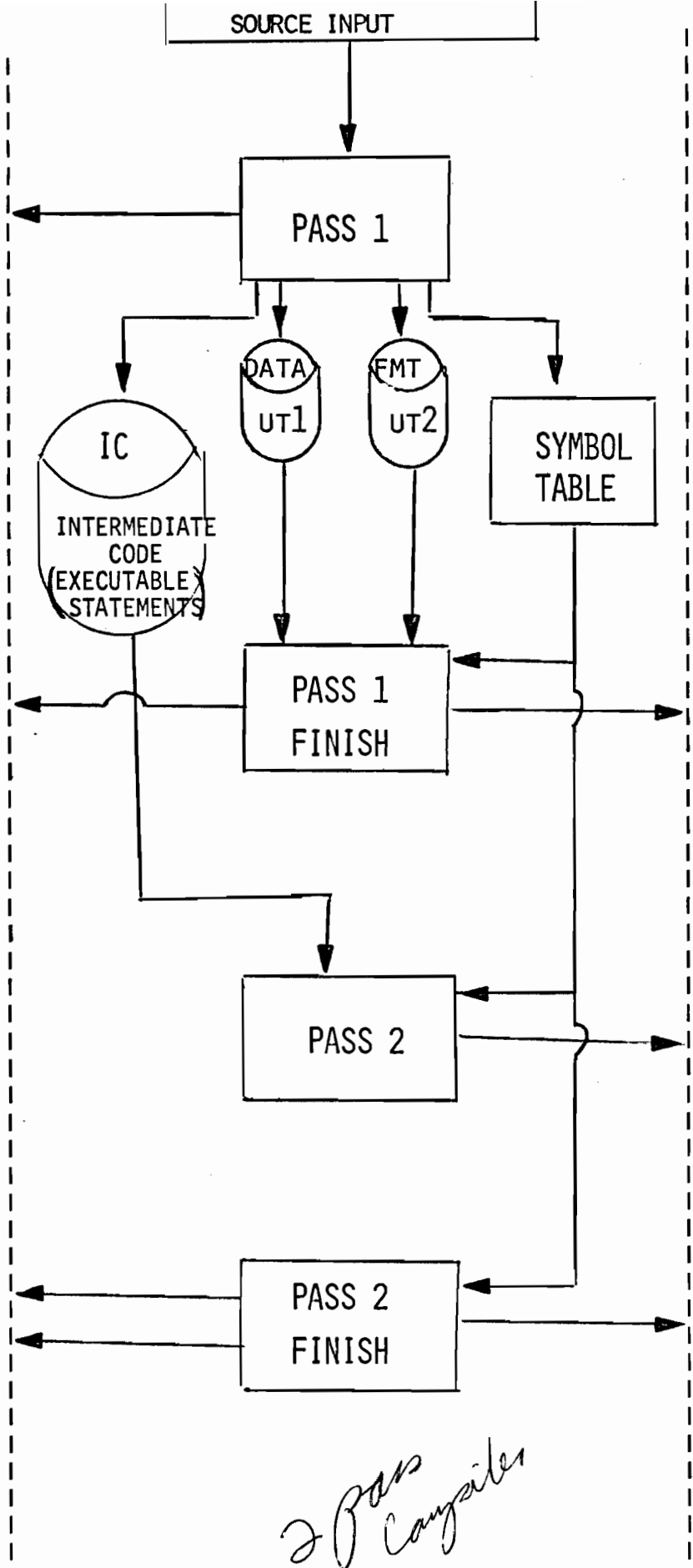
\$EDIT SEQNUM = 1000, INC = 1000

LISTINGS
SOURCE

USL
FILE

SMAP

LABEL MAP
CODE



HEADERS
DATA
FORMAT-DB
TRACE
CODE
FORMAT-DB
ALLOCATION
CODE
EXECUTABLE
STATEMENTS
HEADERS
COMMON
PCAL
FLUT
ENTRIES
PROG UNIT
SEGMENT

2 Pass Compiler

HP 3000

**fortran
i/o**

READ (control part)
WRITE

element,element,...,element

optional
can be implied DO

(unit,format,label,label)

optional

END = statement label

ERR = statement label

optional

statement label
array name
character variable or array
* free field

file (sequential access)

file @ record (direct access)

expression (not complex)
converted to integer

integer constant or variable (1-99)

FORMATS

Dw.d	
Ew.d	
Fw.d	
Gw.d	(F or E)
Mw.d	monetary
Nw.d	numeric
Iw	input default Fw.0
Ow	Octal
Lw	Logical
Aw	left justified
Rw	right justified
Sw	string
P	scale factor.

"..." skip on input
'...' printed on output
nH
nX
Tn tabulation of next position for output.
. / end current record.

*Object
Time
Formats*

FREE FIELD CONTROL

Data item delimiter space, comma, any ASCII character not part of the data item

Record terminator	/
sign of data	(+,-)
fraction subfield	. (Period)
exponent subfield	(E,+,-,D)
Octal	%
character string	"..."
comment	<<...>>

C O M P A N Y X Y Z

DATE	ACCOUNT 630	ACCOUNT 635	ACCOUNT 640	ACCOUNT 645	DAILY TOTAL
8/ 7/72	\$10.48	\$20.97	\$279.82	\$225.27	\$536.54
8/ 8/72	15.01	99.57	534.02	972.65	1,621.25
8/ 9/72	1.53	91.29	939.65	763.93	1,796.40
8/10/72	2.01	90.70	343.95	648.09	1,084.75
8/11/72	1.65	22.37	526.66	151.79	702.47
8/14/72	91.65	46.57	191.74	248.30	578.26
8/15/72	69.18	25.60	396.15	493.40	984.33
8/16/72	14.19	85.39	995.05	320.81	1,415.44
8/17/72	62.59	30.99	241.30	306.80	641.68
8/18/72	36.21	89.20	483.60	196.55	805.56

TOTAL \$304.50 \$602.65 \$4,931.94 \$4,327.59
 GRAND TOTAL \$10,166.68

D5A

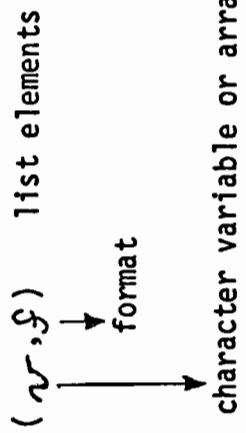
```

1 DIMENSION ACCT(4,50),TOTAL(50),COLTOTAL(4)
2 INTEGER DATE(3,50),TODAY(3),PAGE
3 DATA TOTAL /50*0/, COLTOTAL/4*0/, PAGE/1/, GRANDTOTAL/0/
4 FORMAT(3I3/(3I3,4F7.2))
5 20 FORMAT(' DATE ',2(I2,'/'),I2,T126,'PAGE',T131,I2// T47
6 *C O M P A N Y X Y Z//
7 *T10,'DATE',T26,4('ACCOUNT',I3X),T106,'DAILY TOTAL',
8 */T28,'630',T48,'635',T68,'640',T88,'645'//T8
9 *2(I2,'/'),I2,T24,4(M12.2,8X),T104,M16.2/
10 *(T8,2(I2,'/'),I2,T24,4(N12.2,8X),T104,M16.2))
11 30 FORMAT('//T7,'TOTAL',T21,4(M15.2,5X)
12 *//T84,'GRAND TOTAL',T101,M19.2)
13 READ(5,10,END=99)TODAY,((DATE(I,J),I=1,3),(ACCT(I,J),I=1,4),
14 *J=1,50)
15 99 J=J-1
16 DO 100 I=1,J
17 DO 100 K=1,4
18 COLTOTAL(K) = COLTOTAL(K) + ACCT(K,I)
19 TOTAL(I) = TOTAL(I) + ACCT(K,I)
20 WRITE(6,20)TODAY,PAGE,((DATE(I,K),I=1,3),(ACCT(I,K),I=1,4),
21 *TOTAL(K),K=1,4)
22 DO 200 I=1,4
23 GRANDTOTAL = GRANDTOTAL + COLTOTAL(I)
24 WRITE(6,30)COLTOTAL,GRANDTOTAL
25 STOP
26 END

```

CORE TO CORE CONVERSION

READ
WRITE



FORMATTER

FORTRAN LOGICAL UNIT TABLE

DB-1 → I_{fa}

where

I_{fa} is a positive integer to specify the FLUT byte displacement from DB.

The FLUT is written:

DB + I _{fa} →	U ₁	F ₁
	U ₂	F ₂
	.	.
	.	.
	U _n	F _n
	255	

← The terminal entry (required)

where I_{fa} is defined above and

U₁ . . . U_n = the UNIT numbers (integers in the range [1,99]) in the left byte of each entry, to be specified in Formatter initialization calls

F₁ . . . F_n = 0, when the FLUT is prepared

The last U entry must be 255 to signal the end of the FLUT

NOMINAL FORTRAN/3000 PARAMETERS

The following parameters can be superseded with an MPE/3000 :FILE command.

FILEDESIGNATOR	FTN <i>dd</i> , where <i>dd</i> is the UNIT number in the FLUT (for example, FTN03).
FOPTIONS	All 16 bits are cleared for the following file options: Domain Specifications (bits 15 and 14): NEW BINARY file (bit 13) Default File Designator (bits 12, 11, 10): 000 ¹ Record Format (bits 9 and 8): VARIABLE Carriage Control (bit 7): none ² Nolabel Option (bit 6): no, use standard label (Bits 5 through 0 are spares.)
AOPTIONS	The bits are set to a pattern for the following access options: Access (bit 15 clear): SEQUENTIAL Input Possible (bit 14 set): YES Output Possible (bit 13 set): YES Update (bit 12 clear): NO Append (bit 11 clear): NO Inhibit (bit 10 set): NO Exclusive Access (bit 9 clear; bit 8 set): all modes Retain (bit 7 set): YES Dynamic Locking (bit 6 clear): NO (Bits 5 through 0 are spares.)
RECSIZE	System default value: 128 words.
DEVICE	System default: DISC.
FORMMSG	None.
RECMODE	Clear (800 FPI--magnetic tape, packed binary--card reader).
BLOCKFACTOR	System default value: 1.
NUMBUFFERS	System default: 1.
FILESIZE	System default: (see <i>HP 3000 Multiprogramming Executive Operating System (HP 03000-90005)</i>)
NUMEXTENTS	System default: (see <i>HP 3000 Multiprogramming Executive Operating System (HP 03000-90005)</i>)
INITIALLOC	System default value: 0.
FILECODE	System default: 0.

*Lock Access
UNDER
PROGRAM
control*

¹ Except for FTN05: 100 for \$STDIN, and FTN06: 001 for \$STDLIST.

² Except for FTN06, yes (set).

INTEGER = FNUM (UNIT)

TO FIND FILE NUMBER ASSIGNED TO FORTRAN
LOGICAL UNIT NUMBER.

UNIT = FORTRAN LOGICAL UNIT NUMBER.

CALL FSET (UNIT, NEW, OLD)

TO CHANGE FILE NUMBER ASSIGNED
TO FORTRAN LOGICAL UNIT NUMBER.
UNIT = FORTRAN LOGICAL UNIT NUMBER.
NEW = FILE NUMBER TO BE ASSIGNED
OLD = PREVIOUS FILE NUMBER

AUXILIARY I/O STATEMENTS

REWIND FILE

BACKSPACE FILE

ENDFILE FILE

DOUBLE PRECISION

S	E										F				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 1

F															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Word 2

F															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

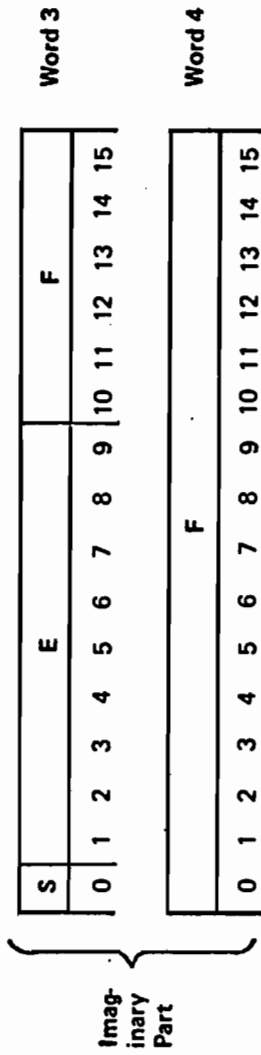
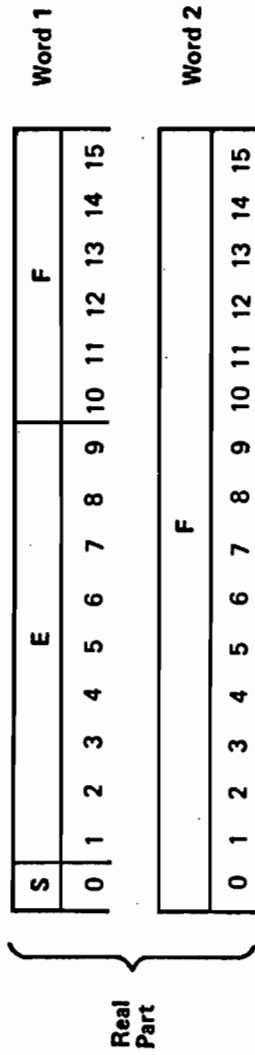
Word 3

Dw.D

±% "ABCDEF" D

% 67542 D

COMPLEX



(.....)

2 Fw,D CMPLX (.....)
 2 Ew,D REAL ()
 AIMAG ()

LOGICAL
 .EQ.
 .NE.

USING MPE 3000 INTRINSICS

- LANGUAGE SPL

- DATA TYPES

- DECLARATIONS

- CONDITION CODES

DATA TYPES

SPL

REAL

LONG

BYTE

LOGICAL

INTEGER

DOUBLE INTEGER

FORTRAN

REAL

DOUBLE PRECISION

CHARACTER

LOGICAL

INTEGER

(USE REAL OR TWO ELEMENT INTEGERS ARRAYS)



PROCEDURE PRINTOP (MESSAGE, LENGTH, CONTROL);
 VALUE LENGTH, CONTROL;
 ARRAY MESSAGE;
 INTEGER LENGTH, CONTROL;
 OPTION EXTERNAL;

NOT TYPE
 DEFAULT TO
 LOGICAL →

CHECK
 SIZE
 NOT
 TYPE
 OF
 DATA

FORTRAN
 PASSES
 ARGUMENTS
 BY REFERENCE

FORTTRAN USAGE

ADDRESS
 WORD
 REF →

ADDRESS
 BYTE REF

CHARACTER MESSAGE *21
 LOGICAL LMESSAGE (11)
 EQUIVALENCE (MESSAGE, LMESSAGE)
 DATA MESSAGE/"TOLD TO THE OPERATOR"/
 CALL PRINTOP(LMESSAGE, \ *21\, \0\)
 IF (.CC.) 10, 20, 20
 CONTINUE

MINUS ⇒ BYTES

SOFT OK HAND
 ERROR ERROR

SPL
 PASSES
 BY VALUE

BACKSLASHES ⇒ PASS BY VALUE

WHO

MPE 8-12

B.T

PROCEDURE WHO(MODE, CAPABILITY, LATTER, USERN, GROUPN, ACCTN, HOMEN, TERM)
8 9 10 11 12 13 14 15

Bit map

LOGICAL MODE, TERMN;

DOUBLE CAPABILITY, LATTER;

BYTE ARRAY USERN, GROUPN, ACCTN, HOMEN;

each 8 BYTES LONG

OPTION VARIABLE, EXTERNAL;

FORTRAN USAGE

LOGICAL MODE, TERMN, VAR

INTEGER CAP(2), LATTR(2)

CHARACTER *8 UN,GN,AN,HN

DATA VAR/.TRUE./

*Bit map
passed in
by value*

CALL WHO(MODE, CAP, LATTR, UN, GN, AN, HN, TERMN, \VAR)

PARTIAL WHO

LOGICAL MODE, VAR

INTEGER CAP(2)

DATA VAR /%[2/3, 6/0] L/

CALL WHO(MODE,CAP, 0,0,0,0,0,0, \VAR\)

GETJCW

MPE 8-40

LOGICAL PROCEDURE
OPTION EXTERNAL;

GETJCW;

FORTRAN USEAGE

SUBROUTINE FORTGETJCW (JCW)

LOGICAL JCW

JCW = GETJCW(I)

END

LOGICAL JCW

CALL FORTGETJCW (JCW)

no subroutine;

Don't

rewrite

stack size

SETJCW

MPE 8-40

PROCEDURE SETJCW(WORD);

VALUE WORD:

LOGICAL WORD:

OPTION EXTERNAL;

FORTRAN USAGE

CALL SETJCW (\25\)

WHEN USING MPE INTRINSICS

- CHECK DATA TYPES
- VALUE
- OPTION VARIABLE
- FUNCTION WITHOUT ARGUMENTS
- CONDITION CODE SETTINGS
- OPTION PRIVILEGED

April 11, 1973

Richard Sleight
Bill O'Shaunessy
HEWLETT PACKARD

11000 Wolfe Road
Cupertino, CA 95014

408-257-7000

Carolyn Morris
HEWLETT PACKARD

11000 Wolfe Road
Cupertino, CA 95014

408-257-7000 (2079)

The purpose of this note is to clarify certain areas involved in the use of the system intrinsic through FORTRAN. There are six areas which deserve special considerations:

1. Data types
2. Value parameters
3. Condition codes
4. Option variable
5. Functions without parameters
6. Obtaining RUN parameters

The following examples illustrate FORTRAN techniques in dealing with these six areas.

① DATA TYPES

The following data types are available in FORTRAN, with an SPL correspondence.

<u>FORTRAN</u>	<u>SPL</u>
REAL	REAL
DOUBLE PRECISION	LONG
CHARACTER	BYTE
LOGICAL	LOGICAL
INTEGER	INTEGER
(use REAL or two element INTEGER arrays)	DOUBLE INTEGER

To maintain precision, it is sometimes necessary to generate a procedure or routine to handle type conversion. These are two examples:

DOUBLE INTEGER to DOUBLE PRECISION

A conversion of a two element INTEGER array or a REAL variable containing an SPL double integer, to DOUBLE PRECISION.

In SPL:

```
LONG PROCEDURE DIDP (DI);
DOUBLE DI;
BEGIN ASSEMBLE (ADD 53);
    DIDP: = LONG (DI);
END
```

In FORTRAN:

```
DOUBLE PRECISION DIDP, VAL
:
:
VAL = DIDP (R)
:
:
```

DOUBLE PRECISION to DOUBLE INTEGER

A conversion of a DOUBLE PRECISION variable to a two element INTEGER array or REAL variable containing an SPL double integer (SPL DOUBLE).

In SPL:

```
DOUBLE PROCEDURE DPDI (DP);
    LONG DP;
BEGIN
DPDI; = FIXR [REAL (DP)]
END
```

In FORTRAN:

```
DOUBLE PRECISION DP
:
:
R = DPDI (DP)
```

Note: The SPL functions, FIXR and FIXT are application dependent. Two functions, DPDIR and DPDIT, may be needed.



②. VALUE PARAMETERS

Whenever a parameter is specified by value, the value or variable must be included in back slashes, e.g., \VAR\or\25\. This parameter must also be of correct length according to the parameter type.

③. CONDITION CODES

The condition code is checked by an arithmetic IF statement with a special argument, .cc.. Branching occurs to the appropriate statement label on the conditions < , =, or > .

Note: If the procedure is a function and the returned value is to an array element, the condition code will not be valid because of the intermediate instruction to handle the subscripts. This invalidity also applies if the value is used in an arithmetic expression.

The following example is of the intrinsic PRINTOP. This intrinsic called through FORTRAN uses value parameters. The program also illustrates the condition code check.

PRINTOP MPE Page 8-10

```
PROCEDURE PRINTOP (Message, Length, Control);  
VALUE Length, Control  
ARRAY Message:  
  
INTEGER Length, Control;  
OPTION External;
```

FORTRAN USAGE

```
CHARACTER MESSAGE*21  
LOGICAL LMESSAGE(11)  
EQUIVALENCE(LMESSAGE,MESSAGE)  
DATA MESSAGE/"TOLD TO THE OPERATOR"/  
10 CALL PRINTOP(MESSAGE,\-21\,\0\  
IF(.CC.) 10,20,20  
20 CONTINUE  
STOP  
END
```

Note: Corresponding variables should agree in length. This can be accomplished by the use of EQUIVALENCE.

EQUIVALENCE LMESSAGE,MESSAGE

④. OPTION VARIABLE

When OPTION VARIABLE is specified by an intrinsic, an extra logical value argument is appended to the complete argument list. This variable serves as a "bit map" with each unique bit representing a variable mentioned in the argument list. The argument bits are mapped in a right to left correspondence (bit 15 to bit 0). Thus, in the example below, variable TERMN is represented by bit 15. Each "on" bit (bit = 1) indicates a required parameter. While SPL handles its own bit mapping, FORTRAN must be told which parameters are required by the intrinsic. If more than sixteen arguments are specified by an intrinsic, then two or more logical value parameters are needed for mapping. The last parameter represents the right most argument.

The following example uses WHO to illustrate the OPTION VARIABLE.

WHO

```
PROCEDURE WHO (Mode, Capability, Latter, Usern, Groupn, Acctn,
              Homen, Termn)
LOGICAL Mode, Termn;
DOUBLE Capability, Latter;
BYTE ARRAY Usern, Groupn, Acctn, Homen;
OPTION VARIABLE, EXTERNAL;
```

FORTRAN USEAGE

```
LOGICAL MODE,TERMN,VAR
INTEGER CAP(2),LATTR(2)
CHARACTER *8 UN,GN,AN,HN
DATA VAR/.TRUE./
CALL WHO(MODE,CAP,LATTR,UN,GN,AN,HN,TERMN,\VAR\)
STOP
END
```


There are two useful FORTRAN functions for use with the file systems.

FNUM

This function supplies the file number value for use in FCONTROL, FCLOSE, etc. This value is set at FOPEN time. This value must be typed INTEGER.

FSET

A subroutine that allows the file number returned by FOPEN to be used in a FORTRAN program. FSET supplies the formatter with this FOPEN value in order that FORTRAN READ and WRITE may be used on the file.

One application is to use FOPEN to specify a file formal designator, other than FTNxx and to pass that value to the formatter, using FSET.

Note: When using system intrinsic functions, the name must be declared as the correct type in the FORTRAN program.

⑥ OBTAINING RUN PARAMETERS

To obtain a parameter passed by the RUN command (;PARAM=) or CREATE intrinsic use the following SPL procedure:

```
$CONTROL SUBPROGRAM
```

```

$CONTROL SUBPROGRAM
BEGIN
PROCEDURE GETPARAM(PARAM);
LOGICAL PARAM;
BEGIN
  LOGICAL A=@+0;
  LOGICAL POINTER B;
  @B := @A - INTEGER(A) - 4;
  PARAM := B;
END;
END.
```

FORTRAN utilization

```
$CONTROL FREE  
CALL GETPARM(I)  
DISPLAY I  
STOP  
END
```

Note: This must be called from the main program only.

WHEN USING MPE INTRINSICS

- * Check Data Types
- * Value
- * Option Variable
- * Function Without Arguments
- * Condition Code Settings

Reference: Systems Programming Languages (03000-90002)
Page 3-34 VARIABLE

The following examples illustrate extensive use of system
intrinsic.

PAGE 1 HEWLETT-PACKARD 32201A.02.0 EDIT/3000 FRI, FEB 23, 1973, 9:13 A

```

1  $CONTROL LABEL,MAP,CODE
2  CHARACTER*4 YEAR,DAY,HOURS,MIN,SEC,TSEC
3  CALL DATIME(YEAR,DAY,HOURS,MIN,SEC,TSEC)
4  WRITE(6,1) YEAR,DAY[2:3],HOURS[3:2],MIN[3:2],TSEC[4:1]
5  1  FORMAT(' YEAR      DAY      HOUR  MINUTES  SECONDS',/,
6  >  '  A5,'      ',A3,'      ',A2,'      ',A2,'      ',A2,',',A:
7  STOP
8  END
9  C
10 C  SUPROUTINE DATIME RETURNS THE YEAR,DAY,HOUR,MINUTES,
11 C  SECONDS,AND TENTHS OF SECONDS. ALL VALUES ARE IN
12 C  CHARACTER*4 TYPE VARIABLES, RIGHT JUSTIFIED
13 C
14  SUBROUTINE DATIME(YEAR,DAY,HOURS,MIN,SEC,TSEC)
15  CHARACTER*4 YEAR,DAY,HOURS,MIN,SEC,TSEC
16  DOUBLE PRECISION CHRONOS,RTIME
17  INTEGER I(3)
18  EQUIVALENCE (RTIME,I)
19  RTIME=CHRONOS(NON)
20  YEAR=STR(I(1)[0:7],4)
21  YEAR[1:2]='19'
22  DAY=STR(I(1)[7:9],4)
23  HOURS=STR(I(2)[0:8],4)
24  MIN =STR(I(2)[8:8],4)
25  SEC =STR(I(3)[0:8],4)
26  TSEC =STR(I(3)[8:8],4)
27  RETURN
28  END

```

YEAR	DAY	HOUR	MINUTES	SECONDS
1973	54	9	14	38.1

```
SCONTROL LABEL,MAP,CODE,FILE=1
  CHARACTER*255 BUFFER(2)
  INTEGER FNUM
  IFN=FNUM(1)
  CALL FGETINFO(\IFN\,0,0,0,IREC,0,0,0,0,0,0,
>0,0,0,0,0,0,0,0,0,\%10\,\%100000\ )
  IF(IREC.LT.0) IREC=-IREC/2
  1  READ(1,END=7,ERR=131) (BUFFER(I9\)[1:IREC],I=1,2)
  2  WRITE(2,END=132,ERR=132) (BUFFER(I)[1:IREC],I=1,2)
  GO TO 1
  7  DISPLAY 'NORMAL COMPLETION OF COPY OCCURRED.'
  STOP
  131 DISPLAY 'ERROR OCCURRED INON INPUT FILE.'
  STOP
  132 DISPLAY 'ERROR OCCURRED ON OUTPUT FILE.'
  STOP
  END
```

PAGE 1 HEWLETT-PACKARD 32201A.02.0 FDIT/3000 THU, FEB 22, 1973, 10:45

```
1 CONTROL MAP, CODE, LABEL
2 DOUBLE PRECISION ITIME, DTIME
3 ITIME=DTIME(-1.000)
4 C
5 C BODY OF PROGRAM
6 C
7 ITIME=DTIME(ITIME)
8 STOP
9 END
10 C DTIME IS A FUNCTION SUBPROGRAM THAT IS USED TO
11 C DETERMINE THE C.P.U. TIME USED BY A PROCESS.
12 C IT ALWAYS RETURNS AS ITS FUNCTION VALUE THE
13 C NUMBER OF MILLISECDS THAT HAVE ELAPSED SINCE
14 C THE BEGINNING OF THE PROCESS. ITS ARGUMENT IS
15 C A DOUBLE PRECISION VARIABLE: IF THIS
16 C ARGUMENT IS GREATER THAN OR EQUAL TO ZERO, THE
17 C FUNCTION DISPLAYS THE DIFFERENCE BETWEEN THE
18 C CURRENT C.P.U. TIME USED AND THE ARGUMENT.
19 C EXAMPLE OF USE:
20 C AT BEGINNING OF PROGRAM INSERT:
21 C DOUBLE PRECISION DTIME, ITIME
22 C AT PLACE WHERE TIMING IS TO BEGIN INSERT:
23 C ITIME=DTIME(-1.000)
24 C BODY OF PROGRAM
25 C AT PLACE WHERE TIMING IS TO BE PRINTED INSERT:
26 C ITIME=DTIME(ITIME) PRINTS TIME SINCE LAST CALL
27 C OR
28 C ITIME=DTIME(0.000) PRINTS TIME SINCE START
29 C
30 DOUBLE PRECISION FUNCTION DTIME(DT)
31 DOUBLE PRECISION DELTAT, DT
32 INTEGER IT(2)
33 REAL RT(1)
34 EQUIVALENCE (IT, RT)
35 RT(1)=PROCTIME(NON)
36 RIT2=FLOAT(IT(2))
37 IF (RIT2.LT.0.0) RIT2=65536.0+RIT2
38 DTIME=DBLE(FLOAT(IT(1)))*65536.000 + DBLE(RIT2)
39 IF (DT.LT.0.000) RETURN
40 DELTAT=DTIME-DT
41 DISPLAY 'CPU TIME USED: ', DELTAT, ' MILLISECS.'
42 RETURN
43 END
```



```

1  SCCONTROL MAP, CODE, LIST, LABEL
2  CHARACTER*80 DBUFFER, BUFFER
3  LOGICAL LBUFFER(40), LDBUFFER(40)
4  CHARACTER*10 ABM, TMC, TM(9)
5  LOGICAL ABML(5), TMCL(5), TML(5,9)
6  EQUIVALENCE (ABML, ARM), (TMCL, TMC), (TML, TM),
7  > (LBUFFER, BUFFER), (LDBUFFER, DBUFFER)
8  INTEGER DSCFN, FNUM, RECNUM
9  DATA ARM/'CHECKABORT'/,
10 > TM(1)/'CDREADERR '/,
11 > TM(2)/'FREADDIRER'/,
12 > TM(3)/'DSKFOPENE9'/,
13 > TM(4)/'FWRTDIRERR'/,
14 > TM(5)/'FREADERR '/,
15 > TM(6)/'FUPDATEERR'/,
16 > TM(7)/'FSPACEERR '/,
17 > TM(8)/'FREADR2ERR'/,
18 > TM(9)/'FUPDATEZER'/,
19 > TMC/'DSFCLOSER'/
20 33 BUFFER='FILE FTN01=RECS,OLD;ACC=UPDATE
21 BUFFER(34:1)=*15C.
22 DBUFFER='FILE FTN02=SASD,OLD;ACC=UPDATE
23 DBUFFER(34:1)=*15C
24 CALL COMMAND(BUFFER, IERR, IPARM)
25 CALL COMMAND(DBUFFER, IERR, IPARM)
26 34 READ(1, END=1301, ERR=1301) (LBUFFER(I), I=1,36)
27 37 IF (BUFFER(1:2).EQ.'?') GO TO 98
28 DISPLAY BUFFER
29 READ(BUFFER,39) RECNUM
30 39 FORMAT(I4)
31 RECNUM=RECNUM-1
32 READ(2*RECNUM, ERR=1302, END=1302) (LDBUFFER(I), I=1,36)
33 44 DBUFFER(29:5)=BUFFER(29:5)
34 45 WRITE(2*RECNUM, ERR=1304, END=1304) (LDBUFFER(I), I=1,36)
35 49 READ(2, ERR=1305, END=1305) (LDBUFFER(I), I=1,36)
36 53 DBUFFER(29:5)='REC+1'
37 531 DSCFN=FNUM(2)
38 54 CALL FUPDATE(\DSCFN, LDBUFFER, \36\ )
39 55 IF (.CC.) 1306,56,1306
40 56 CALL FCONTROL(\DSCFN, \2\,0)
41 57 IF (.CC.) 1300,59,58
42 58 READ(2, END=1308, ERR=1308) (LDBUFFER(I), I=1,36)
43 62 DBUFFER(29:5)='REC+2'
44 63 CALL FUPDATE(\DSCFN, LDBUFFER, \36\ )
45 64 IF (.CC.) 1309,65,1309
46 65 CALL FCONTROL(\DSCFN, \2\,0)
47 66 IF (.CC.) 1300,67,67
48 67 CALL FSPACE(\DSCFN, \1\ )
49 68 IF (.CC.) 1307,69,1307
50 69 READ(2, END=1308, ERR=1308) (LDBUFFER(I), I=1,36)
51 73 DBUFFER(34:25)='REC+4 ==USED FSPACE=== '
52 74 CALL FUPDATE(\DSCFN, LDBUFFER, \36\ )
53 75 IF (.CC.) 1309,76,1309
54 76 CALL FCONTROL(\DSCFN, \2\,0)
55 77 IF (.CC.) 1300,78,78

```

PAGE 2 HEWLETT-PACKARD 32201A.02.0 EDIT/3000 TUE, FEB 20, 1973, 3:04

```
56      78  GO TO 34
57      1300 CALL PRINTOP(ABML,\-10\,0)
58      80  GO TO 98
59      1301 CALL PRINTOP(TML(1,1),\-10\,0)
60      DISPLAY BUFFER
61      GO TO 98
62      1302 CALL PRINTOP(TML(1,2),\-10\,0)
63      84  GO TO 98
64      1303 CALL PRINTOP(TML(1,3),\-10\,0)
65      86  GO TO 98
66      1304 CALL PRINTOP(TML(1,4),\-10\,0)
67      88  GO TO 98
68      1305 CALL PRINTOP(TML(1,5),\-10\,0)
69      90  GO TO 98
70      1306 CALL PRINTOP(TML(1,6),\-10\,0)
71      92  GO TO 98
72      1307 CALL PRINTOP(TML(1,7),\-10\,0)
73      94  GO TO 98
74      1308 CALL PRINTOP(TML(1,8),\-10\,0)
75      96  GO TO 98
76      1309 CALL PRINTOP(TML(1,9),\-10\,0)
77      98  CONTINUE
78      99  CALL FPOINT(\DSCFN,\0\,\49\)
79      100 READ(2,END=1308,ERR=1308) (LDRUFFR(I),I=1,36)
80      102 DBUFFR(34:25)='THIS IS THE MIDDLE RECORD'
81      103 CALL FUPDATE(\DSCFN,LDRUFFR,\36\)
82      104 IF(.CC.) 1309,108,1309
83      108 STOP
84      END
```

```
CONTROL,MAP,CODE,LIST,LABEL
CHARACTER*80 DBUFFER,BUFFER
LOGICAL LBUFFER(40),LDBUFFER(40)
CHARACTER*6 CARDFD,DSCFD,LISTFD
CHARACTER*10 ABM,TMC,TM(9)
LOGICAL ABML(5),TMCL(5),TML(5,9)
EQUIVALENCE (ABML,ABM),(TMCL,TMC),(TML,TM),
>(LBUFFER,BUFFER),(LDBUFFER,DBUFFER),(DRECNM,RRECNM)
INTEGER PRFN,CFN,DSCFN,Q,X,DCFOPT,DCAOPT,PRFOPT
>,PRAOPT,RECNUM,DRECNM(2),FOPEN,BINARY
DATA CARDFD/1RECS 1/,
> DSCFD/1SASD 1/,
> LISTFD/1LIST 1/,
> ABM/1CHECKABORT 1/,
> TM(1)/1CDREADERR 1/,
> TM(2)/1FPEADDIRER 1/,
> TM(3)/1DSKFOPENE9 1/,
> TM(4)/1FWRDIRERR 1/,
> TM(5)/1FREADERR 1/,
> TM(6)/1FUPDATEERR 1/,
> TM(7)/1FSPACEERR 1/,
> TM(8)/1FREADR2ERR 1/,
> TM(9)/1FUPDATEZER 1/,
> TMC/1DSCCLOSER 1/,
> DCFOPT/5/,
> DCAOPT/5/,
> PRFOPT/12/,
> PRAOPT/3/
PRFN=FOPEN(LISTFD,\PRFOPT\,\PRAOPT\,
>0,0,0,0,0,0,0,0,0,0,0,0,\%016000\ )
CFN=FOPEN(CARDFD,\1\,\0\,
>0,0,0,0,0,0,0,0,0,0,0,0,\%016000\ )
DSCFN=FOPEN(DSCFD,\DCFOPT\,\DCAOPT\,
>0,0,0,0,0,0,0,0,0,0,0,0,\%016000\ )
IF(DSCFN.EQ.0) GOTO 1303
DRECNM(1)=0
34 Q=FREAD(\CDFN\,LBUFFER,\-80\ )
IF(.CC.) 1301,37,1301
37 IF(BUFFER(1:2).EQ.'1*') GO TO 98
RECNUM=BINARY(BUFFER,\4\ ) - 1
DRECNM(2)=RECNUM
CALL FREADDIR(\DSCFN\,LDBUFFER,\36\,\RRECNM\ )
IF(.CC.) 1302,42,1302
42 CALL FCONTROL(\DSCFN\,\2\,0)
43 IF(.CC.) 1300,44,44
44 DBUFFER(29:5)=BUFFER(29:5)
45 CALL FWRITEDIR(\DSCFN\,LDBUFFER,\36\,\RRECNM\ )
46 IF(.CC.) 1304,47,1304
47 CALL FCONTROL(\DSCFN\,\2\,0)
48 IE(.CC.) 1300,49,49
49 Q=FREAD(\DSCFN\,LDBUFFER,\36\ )
50 IF(.CC.) 1305,51,1305
51 CALL FCONTROL(\DSCFN\,\2\,0)
52 IF(.CC.) 1300,53,53
53 DBUFFER(29:5)=1REC+1
54 CALL FUPDATE(\DSCFN\,LDBUFFER,\36\ )
55 IF(.CC.) 1306,56,1306
```

PAGE .0002

```
56 CALL FCONTROL(\DSCFN,\2\,0)
57 IF(.CC.) 1300,58,58
58 Q=FREAD(\DSCFN,LDBUFFR,\36\
59 IF(.CC.) 1308,60,1308
60 CALL FCONTROL(\DSCFN,\2\,0)
61 IF(.CC.) 1300,62,62
62 DRUFFR(29:5)='REC+2'
63 CALL FUPDATE(\DSCFN,LDBUFFR,\36\
64 IF(.CC.) 1309,65,1309
65 CALL FCONTROL(\DSCFN,\2\,0)
66 IF(.CC.) 1300,67,67
67 CALL FSPACE(\DSCFN,\1\
68 IF(.CC.) 1307,69,1307
69 Q=FREAD(\DSCFN,LDBUFFR,\36\
70 IF(.CC.) 1305,71,1305
71 CALL FCONTROL(\DSCFN,\2\,0)
72 IF(.CC.) 1300,73,73
73 DRUFFR(34:25)='REC+4 ==USED FSPACE===='
74 CALL FUPDATE(\DSCFN,LDBUFFR,\36\
75 IF(.CC.) 1309,76,1309
76 CALL FCONTROL(\DSCFN,\2\,0)
77 IF(.CC.) 1300,78,78
78 GO TO 34
1300 CALL PRINTOP(ABML,\-10\,0)
80 GO TO 98
1301 CALL PRINTOP(TML(1,1),\-10\,0)
GO TO 98
1302 CALL PRINTOP(TML(1,2),\-10\,0)
84 GO TO 98
1303 CALL PRINTOP(TML(1,3),\-10\,0)
86 GO TO 98
1304 CALL PRINTOP(TML(1,4),\-10\,0)
88 GO TO 98
1305 CALL PRINTOP(TML(1,5),\-10\,0)
90 GO TO 98
1306 CALL PRINTOP(TML(1,6),\-10\,0)
92 GO TO 98
1307 CALL PRINTOP(TML(1,7),\-10\,0)
94 GO TO 98
1308 CALL PRINTOP(TML(1,8),\-10\,0)
96 GO TO 98
1309 CALL PRINTOP(TML(1,9),\-10\,0)
98 CONTINUE
99 CALL FPOINT(\DSCFN,\0\,\49\
100 Q=FREAD(\DSCFN,LDBUFFR,\36\
101 IF(.CC.) 1308,102,1308
102 DRUFFR(34:25)='THIS IS THE MIDDLE RECORD'
103 CALL FUPDATE(\DSCFN,LDBUFFR,\36\
104 IF(.CC.) 1309,105,1309
105 CALL FCLOSE(\DSCFN,\0\,\0\
106 IF(.CC.) 1310,108,1310
1310 CALL PRINTOP(TMCL,\-10\,0)
108 STOP
END
```

HP 3000 MPE

INTRINSICS



3000/MPE INTRINSICS

- MUST BE DECLARED IF A VALUE RETURNED
- SOME INTRINSICS RETURN A VALUE
- INTRINSIC CALL IN PROGRAM BODY
- PARENS ENCLOSE PARAMETERS
- PARAMETERS ARE POSITIONAL
- COMMAS SEPARATE PARAMETERS
- SOME PARAMETERS CAN BE PASSED BY VALUE
- CONDITION CODES WILL BE SET



FOPEN

- ESTABLISHES RELATIONSHIP
BETWEEN PROGRAM AND FILE
- ALLOCATES FILE DEVICE
- ALLOCATES NEW DISK FILE EXTENTS
- ESTABLISHES ACCESS AVAILABILITY
- PERFORMS SECURITY CHECK
- PROCESSES FILE LABELS
- CONSTRUCTS MPE CONTROL BLOCKS
- RETURNS A FILE NUMBER TO BE USED BY
OTHER INTRINSICS

FOPEN PARAMETERS

- FORMAL DESIGNATOR ^{XXXX}
- FOPTIONS
- AOPTIONS
- RECORD SIZE
- DEVICE ^{PUT XXXX}
(PASS BY REF ONLY)
- FORMS MESSAGE
END WITH.
- USER LABELS
- BLOCKING FACTOR
- NUMBER BUFFERS
- FILE SIZE
- NUMBER EXTENTS
- INIT EXTENTS
- FILE CODE
- BIT MAP (FORTRAN ONLY)

ANY NAME HAVE AN INTRINSIC
WITH OPTION VARIABLE \Rightarrow BITMAP NEEDED

FOPEN

FORMAL DESIGNATOR

CHAR-STRING = FORTRAN

○ ADDRESS OF BYTE ARRAY CONTAINING PROGRAM

FILE NAME *SPL*

○ NAME STARTS WITH LETTER

○ NAME ENDS WITH BLANK

○ CAN BE OMITTED IF FOPTIONS

FOPEN
FOPTIONS

<u>BITS</u>	<u>MEANING</u>
14-15	00 = NEW FILE 01 = OLD PERMANENT 10 = OLD TEMPORARY 11 = OLD FILE
13	IF = 1 ASCII FILES (NEW)
10-12	DEFAULT FILE DESIGNATOR
8-9	00 = FIXED LENGTH RECORDS 01 = VARIABLE LENGTH 10 = UNDEFINED
7	IF = 1 CARRIAGE CNTL CHAR EXPECTED
6	IF = 1 UNLABELED TAPE
5	IF = 1 IGNORE :FILE
0-4	RESERVED

FOPEN

DEFAULT FILE DESIGNATOR

(ACTUAL FILE DESIGNATOR)

FOPTION

10-12 000	SAME AS FORMAL FILE DESIGNATOR
001	\$STDLIST
010	\$NEWPASS
011	\$OLDPASS
100	\$STDIN
101	\$STDINX
110	\$NULL

FOPEN
AOPTIONS

<u>BITS</u>	<u>MEANING</u>
12-15	0000 READ ACCESS ONLY
	0001 WRITE ACCESS ONLY DELETE PREVIOUS DATA
	0010 WRITE ACCESS ONLY
	0011 APPEND ACCESS
	0100 ANY ACCESS EXCEPT UPDATE
	0101 ALL FILE INTRINSICS ALLOWED
11	IF = 1 ALLOW MULTI-RECORD I/O
10	IF = 1 ALLOW FILE LOCKING (RIN)
8-9	00 DEFAULTS TO ACCESS TYPE
	01 EXCLUSIVE ACCESS
	10 SEMI-EXCLUSIVE ACCESS
	11 SHARE ACCESS
7	IF = 1 INHIBIT BUFFERING
0-6	RESERVED

FILE INFORMATION
SOURCE HIERARCHY



- FILE LABEL FOR OLD FILES
- FILE COMMAND
- FOPEN INTRINSIC
- SYSTEM DEFAULT VALUES

*A options
ARE ALTERABLE*

*F options
ARE FIXED*

*FILE COMMANDS
LAST TOTAL
JOB or SESSION
UNLESS
RESET @*

PROGRAM DATA INPUT INTRINSIC

FREAD

- READS NEXT SEQUENTIAL RECORD
- READS WORDS OR BYTES
- RETURNS COUNT OF B/W TRANSFERRED
- LOGICAL POINTER INCREMENTED

FILE II
ICT = FREAD (\IFN\, BUFADR, \-80\).

PROGRAM DATA OUTPUT INTRINSIC

FWRITE

- WRITES NEXT SEQUENTIAL RECORD
- WRITES WORDS OR BYTES
- ALLOWS FOR CARRIAGE CONTROL
- LOGICAL POINTER INCREMENTED

*MUST SPECIFY
BECAUSE NO
BIT MAP*

CALL FWRITE (\IOFN\,BUFADR,-132\, \O\)

FCLOSE PARAMETERS

FILE NUMBER

FILE DISPOSITION

0 = NO CHANGE

1 = PERMANENT FILE

2 = TEMPORARY FILE (REWIND)

3 = TEMPORARY FILE

4 = DELETE FILE FROM SYSTEM

SECURITY CODE

CALL FCLOSE(\IFN\,1,0)

*DON'T CHECK
FOR DUPLICATE
FILE NAMES
UNTIL FCLOSE
TIME!!!*

DIRECT ACCESS INTRINSIC

FREADDIR

- FILE NUMBER
- BUFFER ADDRESS
- TRANSFER COUNT
- RECORD NUMBER *DOUBLE word*

CALL FREADDIR(\IFN\,BUFR,\-40\,*0*\IREC\)

DIRECT ACCESS INTRINSIC

FWRITEDIR

- FILE NUMBER
- BUFFER ADDRESS
- TRANSFER COUNT
- RECORD NUMBER

CALL FWRITEDIR(\IOFN\,BUFFER,\128\,\0\,\63\)

*Records #
0 (zero) = 1 RECORD*

DIRECT ACCESS INTRINSIC

FUPDATE

- FILENUMBER
- TARGET
- TCOUNT

*WRITES TO LOGICAL
RECORD - 1.*

CALL FUPDATE(\IOFN\,BUFFER,\100\)

LOGICAL POINTER INTRINSICS

FSPACE

*Tape or
disc.*

- FILE NUMBER
- DISPLACEMENT

*FIXED
LENGTH*

CALL FSPACE(\IFN\,-25\)

FPOINT

- FILE NUMBER
- RECORD NUMBER

DOUBLE WORD

*DISC ONLY
FIXED
LENGTH*

CALL FPOINT (\IFN\,0\,IPTR\)

CALL FPOINT (\IFN\,0\,100\)

FILE CONTROL INTRINSIC

FCONTROL

○ FILE NUMBER

○ CONTROL CODE

1 = LINE CONTROL

2 = COMPLETE I/O

3 = READ HARDWARE STATUS WORD

4 = SET TIME-OUT INTERVAL

5 = REWIND FILE

6 = WRITE END OF FILE

7 = SPACE FORWARD TO TAPE MARK

8 = SPACE BACKWARD TO TAPE MARK

9 = REWIND AND UNLOAD TAPE FILE

*FORCE PHYSICAL I/O
NO BUFFERING*



○ PARAM

*PRINTER control
space*

CALL FCONTROL(\PRFILE\, 1, PRCTL)

CALL FCONTROL(\TPFILE\, 7, DUMMY)

FILE STATUS INTRINSIC

FGETINFO

- FILE NUMBER
- FILE NAME
- FOPTIONS
- AOPTIONS
- RECORD SIZE
- DEVICE TYPE
- LOGICAL DEVICE NUMBER
- DEVICE HARDWARE ADDRESS
- DISC FILE CODE

FILE STATUS INTRINSIC

FGETINFO

- CURRENT RECORD POINTER
- EOF RECORD NUMBER
- FILE LIMIT
- COUNT OF RECORDS ACCESSED
- COUNT OF PHYSICAL I/O
- DISC EXTENT SIZE
- NUMBER OF USER LABELS
- FILE CREATOR NAME
- DISC LABEL ADDRESS

CALL FGETINFO (\IFN\,0,0,0,I,REC,0,0,0,
>IFCD,0,0,EOF,0,0,0,0,0,IBLK,
>0,0,0,0,0,0,\%10\,\%105050\)

*2 word
BIT*
19

INTRINSIC CONDITION CODE

FCHECK

- FILE NUMBER
- ERROR CODE *LOWER order 8 BITS*
- WORDS LEFT OVER DUE TO ERROR
- RELATIVE BLOCK IN ERROR
- COUNT OF LOGICAL RECORDS IN ERROR BLOCK

*IF=0
CK LAST
FILE OPENED*

CALL FCHECK(\IFN\,ERRCOD, LOVRWD, RECBLK, RECCT)

ACCESS-MODE OPTIONS

FSETMODE

- PREVENT AUTOMATIC ERROR RECOVERY
- VERIFY WRITE AS COMPLETE
- INHIBIT CR/LF FOR TERMINAL INPUT

CALL FSETMODE(\IFN\,4\)

DIRECT ACCESS OPTIMIZING INTRINSIC

FREADSEEK

- FILE NUMBER
- RECORD NUMBER

LOOK AHEAD READ INTO SYSTEM BUFFER

MORE THAN AN "OFF-LINE SEEK"

CALL FREADSEEK(\IOFN\, \O\, \IREC\)

FILE RENAME INTRINSIC

FRENAME

- FILE NUMBER
- NEW FILE NAME

- WILL RENAME AN OPENED FILE
- USUALLY A NEW FILE

CALL FRENAME(\IFN\, INM STRG)

HP 3000 STAR

■ STATISTICAL

● ANALYSIS

■ ROUTINES

IN SESSION

:STAR [LISTFILE
*LISTFILE]

STAR COMMAND INTERPRETER PROMPT

IN BATCH

:STAR [LISTFILE
*LISTFILE] [,NOLIST]

INCLUDE # ON CARD

> MODULE PROMPT



DATA

[r,0]

, BETWEEN VARIABLES, COLUMNS

; BETWEEN OBSERVATIONS, ROWS

TO EXIT USE TERM AFTER A ? PROMPT

(IF NO DATA ON FILE STAR IS TERMINATED)

SAVE

YES
NO

VARIABLES

OBSERVATIONS

	NUMBER OF BIRTHS	NUMBER OF DEATHS	POPULATION	NUMBER OF CARS
CITY #1	1,739	1,821	111,739	25,769
CITY #2	86	119	6,753	1,531
CITY #3	821	976	60,871	18,789
CITY #4	1,373	1,431	90,831	22,561
CITY #5	817	926	53,101	12,321

VARIABLES = 4

OBSERVATION = 5

SAVE

[FILENAME]

[T
,P]

PRINT

FILE

[FILENAME
*FILENAME]

SAVE

ELEMSTAT

$[n_1, n_2, \dots, n_m]$
ALL

MEAN

STANDARD ERROR $\equiv \frac{\sigma}{\sqrt{N}}$

STANDARD DEVIATION

VARIANCE

KURTOSIS

SKEWNESS

RANGE

MINIMUM

MAXIMUM

$\left[\begin{array}{l} T_1, N \\ T_7, N^5, N \end{array} \right]$

TRANSFORM

$$T_1(n) = 1/n$$

$$T_2(n) = e^n$$

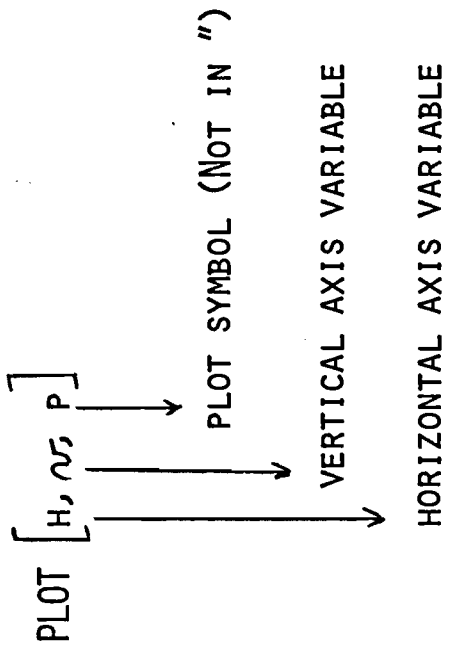
$$T_3(n) = \log_e n$$

$$T_4(n) = \log_{10} n$$

$$T_5(n) = \sqrt{\quad}$$

$$T_6(n) = \text{ROUND}(n) = \text{SIGN}(n) \text{ TRUN}(n + 0.5)$$

$$T_7(n) = n * 10^N \text{ WHERE } N \text{ IS AN INTEGER}$$



SCATTER DIAGRAM

CORRELATE

REGRESSION

DEPENDENT

INDEPENDENT

REGRESSION REPORT

RESIDUALS (YES OR NO)

MORE (YES OR NO)



FREQUENCY [N]

HOW MANY INTERVALS (1-100)

ENTER INTERVAL BOUNDARY POINTS (I + 1)

HISTOGRAM (YES OR NO)

HISTOGRAM [N]

SAME AS FREQUENCY

EDIT

COMMAND

COPY FILENAME

SAVE

AROW N

DROW [R, [,R2]]

LROW [R, [,R2]]

ROBS [R,C]

RRROW [R, [,R2]]

TERM

(DELETE HIGHER NUMBER FIRST)

FINISH

SAVE

1 STAR

ARE YOU AN EXPERIENCED STAR USER?

≥ NO

WHAT IS YOUR DATA FILE?

≥ { * TO CREATE A NEW DATA FILE
FILE NAME TO USE AN EXISTING FILE

STAR MODULES

- CORRELATE** Calculates product-moment correlation coefficients, mean, and standard deviation of all variables (columns).
- DATA** This command allows a user to input a data matrix through the terminal. Must specify the number of variables (columns), and number of observations (rows). The data is input separated by commas, and each row terminated with a semicolon.
- EDIT** Editor module
- AROW R** Add row(s) to end of data matrix,
 - COPY** file name Copies current data file onto a new file. Recomputes file size:
(# observations)*(# variables)*2(words/value)*2(extra space for expansion).
 - DROW R[,R1]** Delete row R or rows R to R1.
 - LROW R[,R1]** List row R or rows R to R1.
 - ROBS R,C** Replace observation R(row) for variable C(column).
 - RRROW R[,R1]** Replaces row R, or rows R to R1, for all variables.
 - TERM** Terminates edit module.
- ELEMSTAT** This calculates the following elementary statistics on each specified variable (column): mean, standard error, standard deviation, variance, kurtosis, skewness, range, minimum and maximum.
- FILE** This command reads the specified file to obtain a current data matrix.
- FINISH** Terminates STAR and returns control to MPE. It does not terminate the Session. STAR may be run again during the same session accessing the temporary files. FINISH is only valid in the executive module which uses the prompt character #.
- FREQUENCY** Lists the frequency distribution of a variable (column). Must specify the number of ranges desired, and the R+1 values limiting each range.
- HELP** Gives a short explanation of the previous question.
- HISTOGRAM** Prints a bar graph for a variable (column). Must specify the number of intervals, and the boundary point values.
- PLOT** Prints an X,Y plot of any two variables. The first specified variable is on the X axis.
- PRINT** Prints the entire data matrix.
- PROMPT** The executive module uses the prompt character #. The sub-modules use the prompt character >.

STAR MODULES

- REGRESSION Calculates multiple linear regression coefficients and related statistics.
- SAVE This command sets a flag which indicates the necessity to copy the current data matrix to the permanent file upon termination of STAR. Note, the actual file writing activity does not take place immediately. Actual writing to the file can be forced by SAVE, then DATA with a minimum input 1,1;1, then FILE.
- :STAR [list file [,NOLIST]]
The normal list device running under STAR is the terminal. Output may be directed to another device by using the optional list file. During this mode, the prompt characters and questions can be suppressed by using the optional ,NOLIST. Notice, STAR is valid in MPE under the prompt character :.
- TERM This command is only valid under sub-module control when the prompt character is >. It transfers control from the sub-module back to the executive module.
- TRANSFORM Performs mathematical manipulation on the specified variable.

$$T_1(x) = 1/x$$

$$T_2(x) = e^x$$

$$T_3(x) = \text{Log}_e x$$

$$T_4(x) = \text{Log}_{10} x$$

$$T_5(x) = \sqrt{x}$$

$$T_6(x) = \text{round}(x) = \text{sign}(x) [\text{trunc}(|x| + 0.5)]$$

$$T_7(x) = x * 10^n \text{ where } n \text{ is an integer}$$

TERM Terminate TRANSFORM module

The required input format is: transform number, variable number. 2,3 would produce the exponential for variable (column) three. Notice, for transform 7 the powers of ten integer must be supplied. The word "all" may replace an individual variable number. After executing the transform pair it prompts "for additional transform execution".

DEFINITIONS

Ref: Handbook Prob. & Statistics
QA 276 B44
The Chem Rubber Pub

QA 273 B925
Handbook . . .
Burlington & May

KURTOSIS

Measure of the sample concentration around the mean as compared with the normal distribution. Values > 0 indicate more heavily concentrated (sharper peak).

MAXIMUM

The maximum is the largest numerical value in the sample.

MEAN

The mean of a group is the numerical average $\bar{X} = \frac{1}{N} (X_1 + X_2 + \dots + X_N)$.

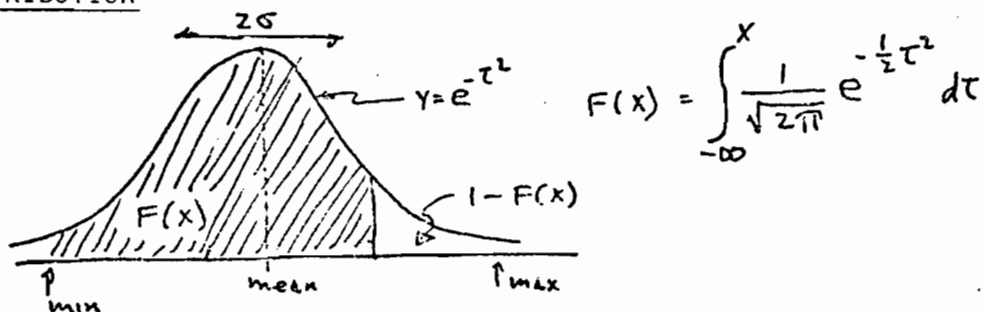
MEDIAN

Median is the $\frac{N+1}{2}$ value where the sample is arranged in ascending order of magnitude, and N is odd. For N even the Median is usually taken as the average of the two middle values.

MINIMUM

The minimum is the smallest numerical value in the sample.

NORMAL DISTRIBUTION



- The Normal Distribution follows the "bell shaped curve". With a symmetrical distribution Skewness is zero. This standard curve includes 68% within $\pm\sigma$, 95% $\pm 2\sigma$, and 99% $\pm 3\sigma$ (Kurtosis = 0). A finite sample will have a maximum and minimum, and will deviate from this ideal.

RANGE

The range is the difference between the largest and smallest values in the sample.

REGRESSION

The linear regression coefficients are a measure of the functional relationship between two or more variables.

SKEWNESS

Measure of the change from the normal Distribution curve left or right. The sample has a considerable number of extreme cases on one side. SKEW $\neq 0$ indicates skew to the right side.

STANDARD DEVIATION

$$S_j = \sqrt{\sum_{i=1}^n (x_{ij} - \bar{J}_j)^2 / (n-1)}$$

The Standard Deviation provides a measure of the distribution of a sample around the mean. The smaller this sample spread the smaller the Standard Deviation. ↖ around the mean.

VARIANCE

The Variance is the square of the Standard Deviation.

exercise

MODULE: USER TECHNIQUES

TOPIC REF: CHARACTER

APPLICABLE REF: _____

PURPOSE Negative D0 variables and character capabilities practice.

GIVE N

CHARACTER INPUT*80

READ(5,10) INPUT

10 FORMAT (S)

Write a program segment to find the last nonblank character.

DO 20 I = 80, 1, -1

IF (INPUT [I:1], .NE., " ") GO TO 25

20 CONTINUE

25 DISPLAY I

INST.

NAME

exercise

PURPOSE To use the 16 bit logical expression extention of HP 3000 FORTRAN

GIVEN

LOGICAL X,Y,Z,A,B,C,D,E

DATA X,Y,Z/146235L,7157L,2347L/

DETERMINE THE VALUES OF A,B,C,D,E

A = .NOT. X

B = Y .AND. Z

C = .TRUE. .XOR. X

D = X .OR. Y

E = X .AND. .NOT. Y .XOR. Z .AND. .NOT. X .OR. Y

INST.

NAME

exercise

PURPOSE To use the composite number extension of HP 3000 FORTRAN

GIVEN

LOGICAL A,B

DETERMINE VALUES OF A,B,C,D,I,J

$$A = \%[3/3, 3/12, 5/27, 5/0]L$$

$$B = \%[7/64, 4/20, 3/1]L$$

$$C = \%[7/23, 0/44, 2/3]R$$

$$D = \%[18/3714563, 5/6]R$$

$$I = \%[5/1, 6/19]$$

$$J = \%[3/5, 4/5]$$

INST.

NAME

exercise

PURPOSE To use the partial word designator extension of HP 3000 FORTRAN

GIVEN

IMPLICIT LOGICAL A-H

DATA A/W [3/1, 4/17, 5/10] U/, B/7134725/, I/10542/

DETERMINE VALUES OF EACH

B = A [9]

C = A [4:6]

E = D [15:7]

J = D [8]

F = I [3:4]

K = I [14]

INST.

NAME

MODULE: USER TECHNIQUES FORTRANTOPIC REF: INTRODUCTION

APPLICABLE REF: _____

laboratory project

OBJECTIVE Use the EDITOR to input a FORTRAN source program then compile and run that program.

GIVEN: A listing of the program.

Provide a listing with your name for the program name and a copy of output to the instructor.

```

$CONTROL FREE
PROGRAM GCD
1 ACCEPT N,K
IF(N.EQ.0) GO TO 15
IF (N.EQ.15) IF (K.EQ.0) PAUSE 10
MMM = N
MM=K
13 MR=N-K*(N/K)
IF (MR.EQ.0) GO TO 12
11 N = K
K = MR
GO TO 13
12 WRITE(6,14) MMM,MM,K
14 FORMAT(' GCD OF 'I8' AND 'I8' IS 'I8' .')
GO TO 1
15 END

```

INST.

NAME

laboratory project

MODULE: USER TECHNIQUES FORTRAN

TOPIC REF: SYMBOL TRACE

APPLICABLE REF: _____

OBJECTIVE To utilize SYMBOL TRACE in a given program, using PRINT, HALT, SET, GO, DROP, AND CHECK paragraphs.

GIVEN:

File

TRACELAB

an ASCII file of the source program

TRACEIN

an ASCII input file for this program



Procedure:

1. a. Modify program TRACELAB to trace
main programs
array IN, IOUT
variable STOPNOW
subprogram REVER
array IN, IOUT
variables I,J,K

Save the program in your own file name

- b. Create a batch trace file tracing the structure in both the main program and subroutine REVER. Also to return control to trace at statement label 20 of the main program.



INST.

NAME

2. Compile and prep finding the initial stack size.
3. Save the program file.
4. Run the program with a stack about twice as large as indicated in the prep PMAP.

Give batch file name (kept above), normal mode.

continue program execution

5. At the next halt add tracing of statement labels in both the main program and subroutine REVER
6. Completely clean the print halt table.

Return control to trace at the encounter of statement label 20 of the main program and at statement label 40 of subroutine REVER

Continue execution once then determine the current value of I, J and K then change the value of J to 1

continue execution

When control is returned to trace discontinue any tracing in subroutine REVER, continue execution.

7. Have control returned to trace at statement label 20 of the main program.

Continue execution.

transfer control to statement label 5 of the main program.

Continue execution.

Why second line?

Discontinue trace control at statement label 20 of the main program.

Trace subroutine REVER and its parameters from the main program

8. Discontinue tracing subroutine REVER from the main program. Trace the variable J in subroutine REVER when J exceeds 9, transfer control to trace when J exceeds 11.

Continue execution twice.

9. Discontinue tracing J in subroutine REVER.

Trace array IOUT in subroutine REVER when the subscript is less than 6.

Continue execution.

10. Discontinue tracing IOUT in subroutine REVER.

Trace the value of J between statement labels 40 and 50 in subroutine REVER

11. Print the value of J between labels 40 and 50 in subroutine REVER only after the fourth encounter.

Continue execution

Did you remember to stop the printing in step 11?
If not try step 12 again after stopping the trace on
J from step 11.

12. Check the meaning of letter M
13. Terminate the program by setting STOPNOW to 1.

Continue execution.

laboratory project

OBJECTIVE To utilize the character manipulation extentions of HP 3000 FORTRAN

GIVEN:

FILE DESCRIPTION:

filename CHARLAB
disk
Binary file
sequential
unblocked

RECORD DESCRIPTION:

last name	16 characters	} left justified
first name	16 characters	
middle initial	1 character	
code	integer	

SAMPLE OUTPUT:

MR. RICHARD A. TALAMO
 MS. DONNA MELVIN
 MISS SHARON B. TILDEN
 MRS. SUSAN J. CLARK
 DR. JOHN KREBREKS

CODE MEANING:

<u>Value</u>	<u>Title</u>
1	MR.
2	MS.
3	MISS
4	MRS.
5	DR.

INST.

NAME

PROCEDURE: AT THE FINISH OF THIS LAB GIVE THE INSTRUCTOR
A LISTING OF FORMATED NAMES USING CHARLAB AS INPUT,
ALSO A COMPILE COPY OF THE PROGRAM.

laboratory project

OBJECTIVE To utilize the logical and bit manipulation extentions
of HP 3000 FORTRAN

Using a 13 element logical array and the logical constant %6666L.

Fill the first element of the array with (%15) in the most significant bits.

Fill the rest of the array by right shifting 1bit from the previous element.

Compare all bit positions of 4 bit size, no wrap around, of the constant and each array value for a match with octal %15. Report the number of occurrences in integer format and the array in octal format.

At the end of this lab give the instructor a listing of the program and its output.



INST.

NAME

laboratory project

OBJECTIVE Utilize FORTRAN I/O and file commands to read an ASCII list file and create a Binary disk file

Create a DISK file to be used with STAR input is STARF

CREATED BY FORMAT (3(3X,E15.5,','),3X,E15.5,',';')

STAR FILE FORMAT STATISTICAL ANALYSIS ROUTINES PAGE 1-4 / 1-6

INST.

NAME

laboratory project

OBJECTIVE To use the STAR subsystem DATA, ELEMSTAT, PLOT, TRANS-
FORM, EDIT, COPY, PRINT, and SAVE modules.

1. Initiate a star session
2. Indicate that you have no file and want the data module.
3. Input a 2 variable 10 observation data set.
4. Run elementary statistics on current data.
5. Initiate file module use the assigned file.
6. Run elementary statistics on all variables.
7. Plot variable 1 vs variable 2, 3, 4.
8. Transform variable 2 by 1/X.
9. Plot variable 1 vs variable 2.
10. Find and replace bad data point in variable 4.
11. Copy to a new file.
12. Save in a permanent file the current data file.
13. Produce a frequency and Histogram for variable 3 use 6 intervals, Boundaries are:

0.00
 0.09
 0.27
 0.44
 0.62
 0.79
 1.00

Provide the instructor with a listing of steps 6, 7, 9, 10, 13 output or values via PRINT.

INST.

NAME

MODULE: USER TECHNIQUES

TOPIC REF: FINAL LAB

APPLICABLE REF: _____

laboratory project

OBJECTIVE To utilize HP 3000 subsystems to solve a major programming problem

This lab is to be used to finish any incomplete labs, then:

Program and run one of the following problems:

1. Solve the ~~te~~xt analysis program attached.
2. Solve the STAR file manipulation project attached.
3. A program of your own approved by the instructor.

INST.

NAME

TEXT ANALYSIS PROGRAMGIVEN

1. TEXTANLY.PUB.UT CARD IMAGES ON DISK.
2. STATE TABLE
3. TEXT ATTRIBUTES
 - A. PARAGRAPHS ARE INDENTED
 - B. SENTENCES END WITH .? :OR;
 - C. IGNOR SENTENCES OF ONLY ONE WORD
 - D. MAXIMUM WORD LENGTH OF 24 CHARACTERS
 - E. MAXIMUM OF 200 DIFFERENT WORDS;

USING

1. S FORMAT FOR INPUT AND OUTPUT OF CHARACTER STRINGS
2. COMPUTED GO TO IMPLEMENTATION OF STATE TABLE
3. FORTRAN FUNCTION INDEX

REQUIRED:

WRITE, COMPILE AND RUN A PROGRAM WHICH WILL PRODUCE

A NEATLY FORMATED REPORT GIVING:

1. LETTER FREQUENCY
2. TOTAL NUMBER OF LETTERS
3. TOTAL NUMBER OF WORDS
4. AVERAGE NUMBER OF WORDS PER SENTENCE
5. NUMBER OF PARAGRAPHS
6. AVERAGE NUMBER OF SENTENCES PER PARAGRAPH

OPTIONAL

1. REPORT THE WORD FREQUENCY AND AVERAGE WORD LENGTH.
2. SORT THE LETTER FREQUENCY BY NUMBER OF OCCURRENCES AND LIST.
3. SORT THE WORD FREQUENCY BY NUMBER OF OCCURRENCES, AND ALPHABETICALLY LIST EACH.
4. INCLUDED IN THE WORD FREQUENCY LISTINGS THE PARAGRAPH NUMBER AND SENTENCE NUMBER OF FIRST OCCURRENCE.

USEFULL SUBROUTINES

ADDLETTER

ADD LETTER TO BUILDING WORD

ADDWORD

ADD WORD TO LIST OF WORDS

NEXTPLACE

FIND NEXT TRANSFER ITEM NUMBER

STATE TABLE

	NOT OTHER	;;? .	ALPHA	*	END CARD	
1 START	11	11	3	14	14	
2 END WORD	10	9	4	10	5	
3 FIRST WORD OF SENTENCE BUILDING	2	12	3	2	2	
4 WORD BUILDING	2	8	4	2	2	
5 NEW CARD	10	9	4	13	14	
6 NEW CARD LOOK FOR START OF SENTENCE OR P	7	11	3	13	14	
7 NEW P	11	11	3	11	14	
8 END SENTENCE AND WORD	16	16	3	16	6	
9 END SENTENCE	16	16	3	16	6	
10 IGNOR LOOK START NEXT WORD	10	10	4	10	5	
11 IGNOR LOOK START NEXT SENTENCE	11	11	3	11	15	
12 DELETE WORD	11	11	3	11	11	
13 NORMAL END	—	—	—	—	—	
14 ERROR	—	—	—	—	—	
15 NEW CARD LOOK START OF SENTENCE	11	11	3	13	15	
16 IGNOR LOOK FOR START OF SENTENCE OR P	16	16	3	16	6	

USER TECHNIQUES
Project

An Interactive STAR-file Manipulator

Implement the following commands or a set of commands with similar capability.

1. MAKESTAR(Result, OPND,V,O,Format)

This command will read the ASCII file, OPND, under the given FORTRAN Format and convert it to a binary STAR file. V & O are integer number that designate the number of variables and observations respectively of the OPND file. A suggested record length for the result is 128 words. The number of "data groups" can be calculated by your program.

2. TAKE(Result, OPND,V,O)

Take V variables and O observations out of STAR file, OPND, and places them in the STAR-file, Result. V & O are integer numbers.

(Positive V or O could mean the first V variables or first O observations and negative V or O could mean the last V variables or last O observations. Another syntax even more general would be to let V and O be ranges eg..., 1/5, 11/17) would be variables 1 through 5 in observations 11 through 17.)

3. JOINV (Result, OPND1, OPND2)

Join Variables.

Join the variables from the OPND2 file to the variables in the OPND1 file and place the result in the Result file. The OPND1 file and OPND2 file must have the same number of observations. For example, if OPND1 has 5 variables and OPND2 has 3 variables and both have 50 observations then the result file will have 50 observations and 8 variables.

4. JOINO (Result, OPND1, OPND2)

This command is similar to JOINV. JOINO joins the observations in OPND2 to the observations in OPND1 and produces the Result file. The OPND1 file and OPND2 file must have the same number of variables. Eg. OPND1 has 24 observations and OPND2 has 15 observations and both have 6 variables. The Result file of the JOINO command will have 6 variables and 39 observations.

5: SPECS (OPND)

"SPECS" displays the specifications of the OPND file. Specifications displayed should include number of variables, number of observations, and number of Datagroups. Display more if you wish.

Feel free to change any of the above suggested commands or to implement more commands.