I. Plumb

RT-11 SYSTEM
REFERENCE CARD

digital

CONTENTS

CONTENTS (Cont)

SYSCOM LOCATIONS

40,41 Starting adrs of job
42,43 Initial value of Stack Pointer
44,45 Job status Word

Bit	Meaning
15	USR swap bit (unused in F/B)
14*	Upper-/lower-case bit
13*	Reenter bit
12*	TT: special mode bit
9	Overlay bit
8	Chain bit
7*	Error halt bit
6*	Inhibit TT: I/O suspend bit

*Indicates bit set by user

46,47 USR load adrs; normally 0
50,51 High adrs; initially set by Linker, may be modified via .SETTOP
52 EMT error code; must be addressed as a byte
54,55 Beginning of resident Monitor; must never be altered
56 Character requiring fill (7-bit ASCII)
57 Filler count (if 56 and 57 = 0, no fillers are required)

PERMANENT DEVICE NAMES

CR Card Reader
CTn Cassette (n=0-1)
DK Default Storage Device (Disk or DECtape)
DKn Extra units above unit 0 (n=1-7)
DPn RP02 Disk (n=0-7)
DSn RJS03/4 Disks (n=0-7)
DTn DECtape (n=0-7)
DXn RX01 Floppy Disk (n=0,1)
LP Line Printer
MMn TJU16 Magtape (n=0-7)
MTn Magtape (n=0-7)
PP High-Speed Paper Tape Punch
PR High-Speed Paper Tape Reader
RF RF11 Fixed-Head Disk Drive
RKn RK Disk Cartridge (n=0-7)
SY System Device
SYn Unit (n) of same device type as that from which the system was bootstrapped
TT Console Terminal

ASSUMED FILE EXTENSIONS

.BAD	File with bad blocks
.BAK	Editor backup file
.BAS	BASIC source file
.BAT	BATCH command file
.CTL	BATCH control file (generated by BATCH)
.CTT	BATCH internal temporary file
.DAT	BASIC or FORTRAN data file
.DIR	Directory listing file
.DMP	DUMP output file
.FOR	FORTRAN IV source file
.LDA	Absolute binary file
.LLD	Library listing file
.LOG	BATCH log file
.LST	MACRO listing file
.MAC	MACRO/EXPAND source file
.MAP	Load map file
.OBJ	Relocatable binary file, library file
.PAL	EXPAND output file; ASEMBL input file
.REL	Relocatable image file for foreground program
.SAV	Memory image or SAVE file for background program
.SOU	Temporary source file (generated by BATCH)
.SYS	System files and handlers

SPECIAL FUNCTION KEY COMMANDS

CTRL A	Display next page of output; used only after CTRL S (used only with GT ON)
CTRL B	Direct all keyboard input to background job (until CTRL F)
CTRL C	Terminate execution; if background, return to Monitor command level
CTRL E	Display all I/O on screen and console terminal simultaneously (with GT ON command only)
CTRL F	Direct all keyboard input to foreground job (until CTRL B)
CTRL O	Inhibit printing on console terminal
CTRL Q	Resume console output; used only after CTRL S
CTRL S	Temporarily suspend terminal output until CTRL A or CTRL Q
CTRL U	Delete current line being entered
CTRL X	Delete entire command string (EDIT only)

SPECIAL FUNCTION KEY COMMANDS (continued)

CTRL Z	Terminate TT: input
RUBOUT	Delete last character typed on present line
CARRIAGE RETURN	Execute command line (except EDIT)
ALTMODE	Terminate text string; two ALTMODEs cause command execution (EDIT only)

KEYBOARD MONITOR

Commands

.ASS { IGN }

Clear all user-defined assignments

.ASS { IGN } dev:udev

Assign dev a user-defined 1-3 character name (udev)

.ASS { IGN } :udev

Deassigns logical name udev

.B { loc }

Set relocation base (for E and D) to indicated loc; if loc is omitted, resets base to zero

.CLO { SE }

Close all currently open files in background

.DAT { E } dd-mmm-yy

Enter date (day-month-year)

.D loc=val1{,val2,...valn}

Deposit specified values (val1,...) starting at specified location

.E locm {-locn}

Prints contents of specified locations in octal on console terminal

.FRU { N } dev:filnam.ext {/N:n}{/S:n}{/P}

Load and start indicated file as foreground job

/N:n	allocate n extra words to the job
/S:n	allocate n words for stack space
/P	load but do not start job; print bottom address

Commands (continued)

.GET dev:filnam.ext

Load specified background program from indicated dev into memory

.GT OFF

Clear display and resume console terminal output (after .GT ON)

.GT ON { /L:n }{ /T:n }

Enable display processor as Monitor terminal output device

/L:n specifies the number (n) of lines of text to display (12" screen – 1<=n<=37; 17" screen – 1<=n<=50)

/T:n indicates top position (n) of scroll display (12" screen – 1<=n<=1350; 17" screen – 1<=n<=1750)

.IN { ITIALIZE }

Initialize system parameters

.LOA { D } dev { =x } { ,dev=x,. . . }

Make device handler resident for use with specified job where x is the partition (B or F) which is assigned ownership of the device. If x is omitted, the device is public.

.R filnam.ext

Load and start indicated background program from SY:

.RE { ENTER }

Start program at reentry address

.RSU { ME }

Resume execution of suspended foreground job

.RU { N } dev:filnam.ext

Load and start indicated background program from dev specified

.SAV { E } dev:filnam.ext { parameters }

Save areas of memory (indicated by parameter list) into file on dev specified; parameter list format is:

addr1-addr2, addr3, ..., addrn

Commands (continued)

.STA { RT }{ address }

Begin execution at specified address; if no address, the contents of location 40 are assumed

.SUS { PEND }

Stop execution of foreground job

.TIM { E }{ hh:mm:ss }

Enter time of day (hours:minutes:seconds); print current time of day

.UNL { OAD } dev { ,dev,... }

Make specified dev non-resident; UNL FG reclaims the space used by the foreground job

.SET dev { NO } option { =value }

Change dev characteristics and certain system configuration parameters.

LP WIDTH=n	Sets line printer width to n
LP CR	Passes carriage returns to printer
LP NOCR	Does not pass carriage returns to printer
LP FORM0	Issues form feed before a request to print block zero
LP NOFORM0	Turns off FORM0 mode
LP HANG	Waits for operator action if line printer not ready
LP NOHANG	Generates error when line printer not ready
LP LC	Passes lower case characters to printer
LP NOLC	Translates lower case to upper case before printing
CR CODE=n	Modifies card reader handler input code; n is 26 or 29
CR CRLF	Appends CRLF to each card image
CR NOCRLF	Turns off CRLF mode
CR IMAGE	Stores each card column as 12-bit binary number, 1 column per word

SET Command (continued)

CR NOIMAGE	Translates normally, packing card code into ASCII data, one column per byte
CR TRIM	Removes trailing blanks from card images
CR NOTRIM	Transfers full 80 characters
CR HANG	Waits if card reader not ready
CR NOHANG	Generates error if card reader not ready
CT RAW	Sets CT handler to perform read after write to verify data
CT NORAW	Turns off read after write

The following options control the console terminal. They are available only in Foreground/Background mode; they are not permanent and must be reissued when Monitor is rebooted.

TTY TAB	Executes hardware tabs on TT:
TTY NOTAB	Simulates tab stops every 8 positions
TTY FORM	Executes hardware form feeds
TTY NOFORM	Simulates form feeds by printing 8 line feeds
TTY PAGE	Enables CTRL S and CTRL Q
TTY NOPAGE	Turns off PAGE mode
TTY CRLF	Provides carriage return at right margin
TTY NOCRLF	Turns off CRLF mode
TTY WIDTH=n	Sets console width to n
TTY FB	Sets CTRL B and CTRL F to cause console terminal context switch
TTY NOFB	Turns off FB mode; treats CTRL F and CTRL B as control characters
TTY SCOPE	Allows RUBOUT to erase character from the scope
TTY NOSCOPE	Causes RUBOUT to echo \ and character, then delete the character

SET Command (continued)

The following are available in both Monitors but must be reissued when Monitor is rebooted.

TTY COPY	Enables auto-print mode of copier on VT 50 terminal
TTY NOCOPY	Disables auto-print mode of copier on VT 50 terminal
TTY HOLD	Enables auto hold mode on VT50 terminal
TTY NOHOLD	Disables auto hold mode on VT50 terminal
USR SWAP	Permits USR to be swapped
USR NOSWAP	Fixes USR as resident and prohibits swapping

KMON Error Messages

?ADDR?	Address out of range in E or D command
?DAT?	Illegal DATE specified; no DATE entered
?ER RD OVLY?	An overlay in MONITR.SYS is unreadable
F?	CTRL F typed and no foreground job exists (F/B only)
?F ACTIVE?	FRUN or UNLOAD used when a foreground job exists and is active (F/B only)
?FIL NOT FND?	File specified in R, RUN, GET, or FRUN command not found
?FILE?	No file named where one expected
?HANDLR?	Close attempted with no handler in memory; file cannot be closed
?ILL CMD?	Illegal Monitor command or command line too long. May indicate a command which is illegal when a Foreground job is active.

KMON Error Messages (continued)

?ILL DEV?	Illegal or nonexistent device, dev=F when the single-job monitor running, or ASSIGN attempted on DK or SY
?NO CLOCK?	No KW11L clock available for TIME command
?NO FG?	SUSPEND, RSUME, or UNLOAD FG command given, but no foreground job in memory (F/B only)
?OVR COR?	Attempt to R, RUN, GET, or FRUN a file that is too big
?PARAMS?	Bad SAVE parameters
?REL FIL I/O ER?	Program requested not REL file or hard error encountered trying to read file (F/B only)
?SV FIL I/O ER?	I/O error on .SAV file in SAVE (output) or R, RUN, or GET (input) command
?SY I/O ER?	I/O error on system dev
?TIM?	Illegal TIME command argument

Fatal Monitor Errors

?M-BAD FETCH	Error while reading in device handler from SY, illegal load address, or handler was not loaded
?M-DIR IO ERR	Error on directory I/O
?M-DIR OVFLO	Directory overflow
?M-DIR UNSAFE	Error while USR was updating a device directory (F/B only)
?M-FP TRAP	Floating-point exception trap occurred when no .SFPA exception routine active
?M-ILL ADDR	Address specified in a Monitor call is odd or is not within job's limits (F/B only)
?M-ILL CHAN	Channel number too large
?M-ILL EMT	Illegal EMT executed
?M-ILL USR	USR called illegally
?M-NO DEV	No device handler in memory

KEYBOARD MONITOR (continued)

Fatal Monitor Errors (continued)

?M-OVLY ERROR	Unsuccessful attempt to read overlay segment on channel 17
?M-SWP ERR	Error detected reading/writing scratch blocks on system device
?M-TRAP TO 4 ?M-TRAP TO 10	Job referenced illegal memory, or illegal instruction was used; printed PC indicates where failure occurred

Bootstrap Errors

?B—I/O ERROR	An I/O error occurred during system boot
?B—NO MONITR.SYS	No monitor exists on volume being booted
?B—NOT ENOUGH CORE	Not enough memory for the system being booted
?B—NO BOOT ON VOLUME	No bootstrap has been written on the volume

GENERAL COMMAND STRING FORMAT

dev:filnam.ext[n],. . .=dev:filnam.ext/s:value,. . .

where: dev = 3-character device name

filnam.ext = filename and extension (up to 3 output files and 6 input files may be specified)

[n] = length (decimal) of output files (optional)

/s:value = switch letter with optional octal or 3 alphabetic character value

/s!value = switch letter with optional decimal value

CSI ERROR MESSAGES

?ILL CMD?	Syntax error
?FIL NOT FND?	Input file not found
?DEV FUL?	Output file will not fit
?ILL DEV?	Device specified does not exist

Command Arguments

n	Any integer in range ± 16383
0	Refers to beginning of line
/	Refers to end of text in current text buffer
=	Represents -n where n = length of last text argument used (with J,D,C commands only)

Commands

n = argument, filespec = dev:filenam.ext, $ = ALTMODE, ESC
text = text string

A	Advance pointer (by lines)
B	Move pointer to beginning of text buffer
nCtext$	Change to indicated text
nD	Delete specified characters
EBfilespec$	Open file for editing, creating a back-up (.BAK) copy
EC	Display text on terminal (after ED)
ED	Display text on screen, commands on terminal; must be first command
EF	Close current output file
EL	Echo characters in upper- and lower-case
EM	Execute macro command string
ERfilespec$	Open file for input
EU	Echo characters in upper-case only (after EL)
EV	Print editor version number
EWfilespec$	Open new file for output
EX	Terminate editing, output remainder of input file, return to Monitor
nFtext$	Search entire input file for n^{th} following occurrence of text string
nGtext$	Search current text buffer for n^{th} following occurrence of text string
Itext$	Insert text into text buffer
nJ	Move pointer (by characters)
nK	Delete specified number of lines from text buffer

Commands (continued)

nL	List specified number of lines on terminal
Mxcommand stringx$	Insert command string into MACRO buffer where x is delimiter not used in command string; Mxx deletes the MACRO buffer; 0M clears MACRO buffer
nN	Output current text buffer; read next page of input and repeat n-1 times
nPtext$	Search entire input file for nth occurrence of text string without producing output
R	Read next page of text into text buffer
nS	Save n following lines of text in save buffer
U	Copy contents of save buffer into text buffer; 0U deletes save buffer
V	Print current line on terminal
nW	Write n following lines of text to output file
nXtext$	Replace lines of text with indicated text

Immediate Mode Commands

ALTMODE	Enter immediate mode with 2 successive ALTMODEs; return to command mode with 1 ALTMODE
CTRL D	Move cursor forward 1 character
CTRL G	Move cursor to beginning of previous line
CTRL N	Move cursor to beginning of next line
CTRL V	Move cursor back 1 character
RUBOUT	Delete character immediately preceding cursor
Any character except CTRL/C or those above	Insert character as text immediately before cursor

PERIPHERAL INTERCHANGE PROGRAM—PIP

Switches

/A	Copy file in ASCII mode
/B	Copy file in formatted binary mode
/C	Include only files with current date (with wild card and another switch)
/D	Delete file from specified dev
/E	List entire dev directory (include empty spaces)
/F	List short dev directory (filenames only)
/G	Ignore input errors during file transfer
/I	Copy file in image mode (default)
/K	Type absolute block numbers (octal) of bad blocks
/L	List entire directory (no empties listed)
/M:n	Indicate file position (n) for cassette or mag-tape operations
/N:n	Used with /Z to indicate number of directory segments (n) to allocate
/O	Bootstrap specified dev (DT0, RKn, RF, DPn, DSn, DXn)
/Q	Request individual confirmation of each file in a wild card operation
/R	Rename input file to output filename
/S	Compress input device to output device
/T	Extend number of blocks allocated for a file
/U	Copy indicated bootstrap file into blocks 0 and 2 of specified dev
/V	Print version number of PIP in use
/W	Include in directory listing the absolute starting block and extra directory words for each file on dev (with /E or /L)
/X	Copy files individually (without combining)
/Y	Used when accessing system (.SYS and .BAD) files
/Z {:n}	Zero directory of specified dev (n = number of extra words in each directory entry)

LINKER-LINK

Switches

/A	Alphabetize load map entries
/B:n	Specify bottom adrs (n) to be used by program (illegal with /R)
/C	Allow additional lines of command string input
/F	Use default FORTRAN library (FORLIB.OBJ) on system device (SY)

LINKER-LINK (continued)

Switches (continued)

/I	Include global symbols to be searched from library
/L	Produce output file in LDA format (illegal with /R)
/M { :n }	Allow console specification of user's stack address (n=6-digit unsigned octal number or a global address name)
/O:n	Program will be an overlay structure; n specifies the overlay region to which module is assigned
/R	Produce output in REL format; REL files can be linked in either single job or F/B but only can be run in foreground under F/B
/S	Allow maximum memory space for Linker symbol table
/T { :n }	Allow console specification of program starting (transfer) address (n = 6-digit unsigned octal number or a global address name)

ON-LINE DEBUGGING TECHNIQUE — ODT

Commands

RETURN	Close open loc; accept next command
LINE FEED	Close current loc; open next sequential loc
↑ or ^	Open previous loc
← or _	Index contents of opened location by contents of PC and open resulting loc
>	Use contents of open loc as relative branch and open referenced loc
<	Return to sequence prior to @, > or ←, and open succeeding loc
@	Use contents of open loc as absolute adrs and open referenced loc
r/	Open word at loc r
/	Reopen last opened loc
r\	Open byte at loc r
\	Reopen last opened loc as byte
!	Print adrs of opened loc relative to relocation reg whose contents are closest
n!	Print adrs of opened loc relative to relocation reg n
$n/	Open general reg n

Commands (continued)

$B/	Open first word of breakpoint table
$C/	Open constant reg
$F/	Open format reg
$M/	Open first mask reg
$P/	Open priority reg
$R/	Open first relocation reg
$S/	Open status reg
r;nA	Print n bytes in ASCII format starting at loc r, then allow input of n bytes from terminal
;B	Remove all breakpoints
r;nB	Set breakpoint n at loc r
;nB	Remove breakpoint n
r;C	Store r in constant reg
r;E	Search for instructions which reference effective adrs r
;F	Fill memory words with contents of constant reg
r;G	Start program at loc r
;I	Fill memory bytes with low-order 8 bits of constant reg
r;O	Calculate offset from current loc to r
;P	Proceed from breakpoint
k;P	Proceed from breakpoint; stop after encountering the breakpoint k times
;R	Set relocation reg to -1
;nR	Set relocation reg n to -1
r;nR	Set relocation reg n to value of r (default n=0)
R	Subtract relocation reg whose contents are closest to but < = contents of open loc from contents of open loc and print result
nR	Subtract contents of relocation reg n from contents of opened word and print result
;S	Disable single-instruction mode
;nS	Enable single-instruction mode; disable breakpoints
r;W	Search for words with bit patterns matching r
X	Perform a Radix-50 unpack of opened loc; permit storage of new Radix-50 binary number

LIBRARIAN – LIBR

Switches

/C	Continue command on next line
/D	Delete module from library file
/G	Delete entry point (global or .CSECT) from library directory
/R	Replace module in library file
/U	Insert and replace (update) module in library file

If no switch is specified, an insert is assumed.

DUMP

Switches

/B	Output octal bytes
/E:n	End output at block n
/G	Ignore input errors
/N	Suppress ASCII output
/O:n	Output only block number n
/S:n	Start output with block n
/W	Output octal words
/X	Output RAD5Ø characters

SRCCOM

Switches

/B	Compare blank lines
/C	Ignore comments and spacing
/F	Do not send form feeds to output file
/H	Print list of available switches
/L {:n}	Specify number of lines (n < = 31Ø octal) that determines a match (default n=3)
/S	Ignore spaces and tabs

MACRO/CREF

Listing Control Switches

/L:arg	Enable listing arg
/N:arg	Disable listing arg

Arg is one of the following:

BEX	Binary extensions
BIN	Generated binary code
CND	Unsatisfied conditions and all .IF and .ENDC statements
COM	Comments

Listing Control Switches (continued)

LD	Listing directives having no arguments
LOC	Location counter
MC	Macro calls and repeat range expansions
MD	Macro definitions and repeat range expansions
ME	Macro expansions
MEB	Macro expansion binary code
SEQ	Source line sequence numbers
SRC	Source code
SYM	Symbol table
TOC	Table of contents
TTM	Listing output format in terminal mode
/L	Ignore .LIST and .NLIST directives without arguments and list entire program
/N	Produce only symbol table and error listings

Function Control Switches

/D:arg	Disable (.DSABL) certain functions in source input files
/E:arg	Enable (.ENABL) certain functions in source input files
	Arg is one of the following:
ABS	Absolute binary output
AMA	Assemble all relative addresses as absolute addresses
CDR	Treat source columns => 73 as comments
FPT	Floating point truncation
LC	Accept lower case ASCII input
LSB	Local symbol block
PNC	Binary output

CREF Switches

/C	Equivalent to /C:S:M:E
/C:C	Cross reference control sections
/C:E	Cross reference error codes
/C:M	Cross reference MACRO symbolic names
/C:P	Cross reference permanent symbols
/C:R	Cross reference register symbols
/C:S	Cross reference user-defined symbols

Assembly Error Codes

A	Address in instruction incorrect or relocation error

Assembly Error Codes (continued)

B	Boundary error; loc counter updated by +1
D	Doubly-defined symbol referenced
E	End directive not found (.END generated)
I	Illegal character; character ignored
L	Line buffer overflow (>132 characters)
M	Multiple label definition
N	Number missing decimal point
O	Opcode error; directive out of context
P	Phase error
Q	Questionable instruction syntax
R	Register illegally used or referenced
T	Truncation error
U	Undefined symbol (zero value assigned)
Z	Instruction incompatible on various PDP-11 models

Error Messages

?BAD SWITCH?	Specified switch not recognized
?INSUFFICIENT CORE?	Too many symbols; divide into separately assembled subprograms
?NO INPUT FILE?	Input file not specified
?OUTPUT DEVICE FULL?	No room to write output; try PIP to compress device
?TOO MANY OUTPUT FILES?	More than two output files specified
?C-CHAIN-ONLY-CUSP?	Cannot invoke CREF without MACRO
?C-CRF FILE ERROR?	I/O error while accessing DK:CREF.TMP
?C-DEVICE?	Invalid device specified to CREF by MACRO (system error)
?C-LST FILE ERROR?	I/O error while writing cross-ref table to list file

.CDFN .area,.addr,.num

Define new I/O channel (1 channel = 5 words)

.CHAIN

Allow background program to transfer control to another program which is specified in loc 500-511 (RAD50) without operator intervention; words 512-777 are saved

.CHCOPY .area,.chan,.ochan

Open channel, logically connecting it to a channel on another job open for either input or output (F/B only)

.CLOSE .chan

Terminate activity on specified channel and free it for use in another operation

.CMKT .area,.id,.time

Cancel one or more outstanding mark time requests (F/B only) where id is the .MRKT id and time is a pointer to two words where monitor is to store number of ticks remaining in request

.CNTXSW .area,.addrlist

Specify locations to be included in context switch (F/B only), addrlist is terminated with a 0 word. Valid locations are: 2-476, user job area, 160000-177776

.CSIGEN .devspc,.defext,.cstring

Call CSI in general mode and load handlers in .devspc. If .cstring is 0, get command string from terminal

.CSISPC .outspc,.defext,.cstring

Call CSI in special mode and return file specifications in .outspc. If .cstring is 0, get string from terminal

.CSTAT .area,.chan,.addr

Furnish 6 information words about channel (F/B only)

word 1	channel status
word 2	file start block
word 3	file length
word 4	highest block written
word 5	device unit number
word 6	device name (RAD50)

.DATE

Move current date word into R0

bits 14-10	month (1-12.)
bits 9-5	day (1-31.)
bits 4-0	year -72. (decimal)

.DELETE .area,.chan,.dblk,.count

Delete named file from indicated device

.DEVICE .area,.addr

Set up list of addresses to be loaded with specified values upon program termination (F/B only)

.DSTATUS .cblk,.devnam

Provide information about device characteristics

word 1	device status
word 2	handler size
word 3	handler entry point
word 4	device size

.ENTER .area,.chan,.dblk,.length,.count

Allocate space on specified device and create tentative entry for named file

.EXIT

Terminate user program and return control to Monitor; if R0=0, perform INIT; if R0≠0 do not perform INIT

.FETCH .coradd,.devnam

Load device handler into memory from system device

.GTIM .area,.addr

Return current time of day in ticks past midnight

.GTJB .area,.addr

Pass certain job parameters back to user program

word 1	job number (0=B; 2=F)
word 2	high memory limit
word 3	low memory limit
word 4	start of I/O channel space
words 5-8	reserved

.HERR

Disable error interception and allow Monitor to detect and act on fatal errors

.HRESET

Reset channels, release device handlers, and stop all I/O transfers in progress

.INTEN .priority { ,.pic }

Notify Monitor that interrupt occurred and set processor priority to correct level. The second parameter is non-blank to generate position-independent code

.LOCK

Lock USR in memory

.LOOKUP .area,.chan,.dblk,.count

Associate specified channel with device and/or file name

.MRKT .area,.time,.crtn,.id

Schedule completion routine to be entered after specified time interval (number of ticks) (F/B only)

.MWAIT

Suspend execution until all messages are transmitted or received (F/B only)

.PRINT .addr

Output an ASCII string to terminal

.PROTECT .area,.addr

Used to obtain exclusive control of vector in range 0–476 (F/B only)

.PURGE .chan

Deactivate channel without taking any other action

.QSET .addr,.qleng

Enlarge I/O queue for Monitor; 1 queue element = 7 words

.RCTRLO

Enable terminal printing

.RCVD .area,.buff,.wcnt

Post request to receive message and continue execution (F/B only)

.RCVDC .area,.buff,.wcnt,.crtn

Post request to receive message and enter completion routine when message received (F/B only)

.RCVDW .area,.buff,.wcnt

Post request to receive message and wait until received (F/B only)

.READ .area,.chan,.buff,.wcnt,.blk

Initiate transfer of words from specified channel to memory and continue execution

.READC .area,.chan,.buff,.wcnt,.crtn,.blk

Initiate transfer from channel to memory; continue executing user program; enter specified routine when transfer complete

.READW .area,.chan,.buff,.wcnt,.blk

Transfer words from specified channel to memory; return control to user program when complete

.REGDEF

Define general registers R0-R5, SP, PC

.RELEAS .devnam

Remove device handler from memory

.RENAME .area,.chan,.dblk

Change file name

.REOPEN .area,.chan,.cblk

Reassociate channel with file on which a SAVESTATUS was performed

.RSUM

Resume job after it was suspended (F/B only)

.SAVESTATUS .area,.chan,.cblk

Store 5 words (containing data concerning file definition) into memory; free channel for use

word 1	channel status
word 2	start block of file
word 3	length of file
word 4	reserved
word 5	even byte = I/O count odd byte = device unit number

.SDAT .area,.buff,.wcnt

Initiate message transfer; return control to user program immediately (F/B only)

.SDATC .area,.buff,.wcnt,.crtn

Initiate message transfer; transfer control to specified routine when message is received (F/B only)

.SDATW .area,.buff,.wcnt

Initiate message transfer; return control to user program when message received (F/B only)

.SERR

Inhibit fatal errors from aborting job

.SETTOP .addr

Request additional memory for program and return highest address available in R0

.SFPA .area,.addr
Set user interrupt for floating point processor exceptions

.SPFUN .area,.chan,.code,.buff,.wcnt,.blk,.crtn
Provide special device functions to magtape, cassette, and floppy disk

.SPND
Suspend running job (F/B only)

.SRESET
Reset certain memory areas, dismiss device handlers, purge currently open files, reset to 16 channels, reset I/O queue to one element

.SYNCH .area
Cause the rest of the interrupt service routine to run as a completion routine

.TLOCK
Attempt to gain ownership of USR; if unsuccessful, return control with C bit set (F/B only)

.TRPSET .area,.addr
Allow user job to intercept traps to 4 and 10

.TTINR
Input character from terminal, if possible

.TTOUTR
Output character to terminal, if possible

.TTYIN .char
Input character from terminal and wait until done

.TTYOUT .char
Output character to terminal and wait until done

.TWAIT .area,.time
Suspend running job for specified amount of time (number of ticks) (F/B only)

.UNLOCK
Release USR from memory

. . V1 . .
Enable macro expansions to occur in Version 1 format

. . V2 . .
Enable macro expansions to occur in Version 2 format

.WAIT .chan
Suspend program execution until channel I/O complete

FORMAT OF PROGRAMMED REQUESTS (cont)

.WRITC .area,.chan,.buff,.wcnt,.crtn,.blk
Transfer words from memory to specified channel; when complete, pass control to specified routine

.WRITE .area,.chan,.buff,.wcnt,.blk
Initiate transfer from memory to channel; return control to user program immediately

.WRITW .area,.chan,.buff,.wcnt,.blk
Transfer words from memory to channel; when complete, return control to user program

BASIC/RT11

Statements

CALL "function name" [(argument list)]
Calls assembly language user functions from BASIC program

CHAIN "file descriptor" [LINE number]
Terminates execution of user program, loads and executes specified program starting at line number, if included

CLOSE $\left[\left\{\begin{matrix}\text{VFn}\\ \#n\end{matrix}\right\}\right]$
Closes logical file specified; if no file specified, closes all open files

DATA data list
Used in conjunction with READ to input data into executing program

DEF FN letter (argument)=expression
Defines user function to be used in program

DIM variable(n), variable(n,m), variable$(n), variable$(n,m)
Reserves space for lists and tables according to subscripts specified after variable name

END
Placed at physical end of program to terminate program execution

FOR variable = expr1 TO expr2 [STEP expr3]
Sets up loop to be executed specified number of times (expr = expression)

Statements (continued)

GOSUB line number
Transfers control to specified line of subroutine

GO TO line number
Unconditionally transfers control to specified line in program

IF expression rel.op expression { THEN / GO TO } line number
Conditionally transfers control to specified line of program

IF END #n { THEN / GO TO } line number
Tests for end file on sequential input file #n

INPUT var list
Inputs data from terminal to specified variables

INPUT #expression: var list
Inputs data from particular input device, as specified in OPEN statement, to specified variables

[LET] variable=expression
Used to assign value to specified variable

[LET] VFn(i)=expression
Used to set value of virtual memory file element

NEXT variable
Placed at end of FOR loop to return control to FOR statement

OPEN

OPEN file [FOR { INPUT / OUTPUT }] [(b)] AS FILE #n [DOUBLE BUF]

Opens sequential file for input or output as specified

OPEN file [FOR { INPUT / OUTPUT }] [(b)] AS FILE VFnx (dimension)=string length

Opens virtual memory file for input or output

OVERLAY "file descriptor"
Overlays or merges program currently in memory with specified file, and continues execution

PRINT expression list
Outputs data to terminal

Statements (continued)

PRINT #expression: expression list
Outputs to particular output device, as specified in OPEN statement

RANDOMIZE
Causes random number generator to calculate different random numbers every time program is run

READ variable list
Assigns values listed in DATA statement to specified variables

REM comment
Inserts explanatory comments into BASIC program

RESTORE
Resets data block pointer so same data can be used again

RESTORE #n
Rewinds input sequential file #n to beginning

RETURN
Returns program control to statement following last GOSUB statement

STOP
Used at the logical end of the program to terminate execution

Key Commands

ALTMODE (ESC)
Deletes entire current line

CTRL C
Interrupts execution of command or program and returns control to RT-11 Monitor

CTRL O
Stops output to terminal and returns BASIC to READY message when program or command execution is completed

CTRL U
Deletes entire current line

(SHIFT/O)
Deletes last character typed

RUBOUT
Deletes last character typed

Edit and Control Commands

CLEAR
Sets array and string buffers to nulls and zeroes

LIST
Lists user program currently in memory on terminal

LIST line number

LIST -line number

LIST line number- [END]

LIST line number-line number
Lists specified program line(s) on terminal

LISTNH
Lists user program currently in memory on terminal (no header)

LISTNH line number

LISTNH -line number

LISTNH line number- [END]

LISTNH line number-line number
Lists lines associated with specified numbers (no header)

NEW "file descriptor"
Does SCRatch and sets current program name to one specified

OLD "file descriptor"
Does a SCRatch and inputs program from specified file

RENAME "file descriptor"
Changes current program name to one specified

REPLACE "file descriptor"
Replaces specified file with current program

RUN
Executes program in memory

RUNNH
Executes program in memory (no header)

SAVE "file descriptor"
Outputs program in memory as specified file

SCRatch
Erases entire storage area

Math Functions

ABS(x)
Returns absolute value of x

ATN(x)
Returns arctangent of x as angle in radians in range $+\pi/2$ or $-\pi/2$

BIN(x$)
Computes integer value of string of 1's and 0's

COS(x)
Returns cosine of x radians

EXP(x)
Returns value of e↑x where e=2.71828

INT(x)
Returns greatest integer less than or equal to x

LOG(x)
Returns natural logarithm of x

OCT(x$)
Computes integer value from string of blanks and digits from 0 to 7

RND(x)
Returns random number between 0 and 1

SGN(x)
Returns value indicating sign of x

SIN(x)
Returns sine of x radians

SQR(x)
Returns square root of x

TAB(x)
Causes terminal type head to tab to column number x

String Functions

ASC(x$)
Returns as decimal number 7-bit internal code for 1-character string (x$)

CHR$(x)
Generates 1-character string having ASCII value of x

DAT$
Returns current date in format 07-May-73

String Functions (continued)

LEN(x$)
Returns number of characters in string (x$)

POS(x$,y$,z)
Searches for and returns position of first occurrence of y$ in x$ starting with z^{th} position

SEG$(x$,y,z)
Returns string of characters in positions y through z in x$

STR$(x)
Returns string which represents numeric value of x

TRM$(x$)
Returns x$ without trailing blanks

VAL(x$)
Returns number represented by string (x$)

Error Messages

?ARG AT LINE xxxxx	ARGUMENT ERROR
?ATL AT LINE xxxxx	ARRAYS TOO LARGE
?BDR AT LINE xxxxx	BAD DATA READ
?BRT	BAD DATA-RETYPE FROM ERROR
?BSO AT LINE xxxxx	BUFFER STORAGE OVERFLOW
?DCE AT LINE xxxxx	DEVICE CHANNEL ERROR
?DFO AT LINE xxxxx	DISPLAY FILE OVERFLOW
?DNR	DEVICE NOT READY
?DV0 AT LINE xxxxx	DIVISION BY 0
?ETC AT LINE xxxxx	EXPRESSION TOO COMPLEX
?FDE AT LINE xxxxx	FILE DATA ERROR
?FIO AT LINE xxxxx	FILE I/O ERROR
?FNF AT LINE xxxxx	FILE NOT FOUND
?FNO AT LINE xxxxx	FILE NOT OPEN
?FTS AT LINE xxxxx	FILE TOO SHORT
?FWN AT LINE xxxxx	FOR WITHOUT NEXT
?GND AT LINE xxxxx	GOSUBS NESTED TOO DEEPLY
?IDF AT LINE xxxxx	ILLEGAL DEF
?IDM AT LINE xxxxx	ILLEGAL DIM
?ILN	ILLEGAL NOW
?ILR AT LINE xxxxx	ILLEGAL READ
?INS AT LINE xxxxx	IMPROPERLY NESTED SUBPICTURES
?LTL	LINE TOO LONG
?NBF AT LINE xxxxx	NEXT BEFORE FOR
?NER AT LINE xxxxx	NOT ENOUGH ROOM

BASIC/RT 11 (continued)

Error Messages (continued)

?NPR	NO PROGRAM
?NSM AT LINE xxxxx	NUMBERS AND STRINGS MIXED
?OOD AT LINE xxxxx	OUT OF DATA
?OVF AT LINE xxxxx	OVERFLOW
?PTB	PROGRAM TOO BIG
?PWF AT LINE xxxxx	POWER FAIL
?RBG AT LINE xxxxx	RETURN BEFORE GOSUB
?RPL	FILE SAVED ALREADY EXISTS
?SOB AT LINE xxxxx	SUBSCRIPT OUT OF BOUNDS
?SSO AT LINE xxxxx	STRING STORAGE OVERFLOW
?STL AT LINE xxxxx	STRING TOO LONG
?SYN AT LINE xxxxx	SYNTAX ERROR
?TLT	LINE TOO LONG TO TRANSLATE
?UFN AT LINE xxxxx	UNDEFINED FUNCTION
?ULN AT LINE xxxxx	UNDEFINED LINE NUMBER
?WLO AT LINE xxxxx	WRITE LOCKOUT
?↑ER AT LINE xxxxx	↑ ERROR

FORTRAN IV

Statements

Arithmetic or Logical Assignment

a = b

Assigns value of expression b to variable a

Arithmetic Statement Function

tname (a1,...)=x

Creates user-defined function having variables a1,... as dummy arguments

ACCEPT f, list

Reads input from logical unit 5 (default TT:); f is format statement label, and list is optional data list

ASSIGN n TO ivar

Assigns statement number n to integer variable ivar

BACKSPACE u

Currently open file on logical unit number u is backspaced one record

BLOCK DATA

Specifies subprogram which follows as BLOCK DATA subprogram

Statements (continued)

CALL {name / name (arg1, arg2,...,)}

Calls SUBROUTINE subprogram with name specified, passing actual arguments (arg) to replace dummy arguments in SUBROUTINE definition

COMMON/name1/var1,...,/name2/var,var

Reserves one or more blocks of storage space under name specified to contain variables (var) associated with that block name

CONTINUE

Causes no processing

DATA var1, var2,.../val1,val2,.../

Causes elements in list of values (val) to be initially stored in corresponding elements of list of variable names

DECODE (c,f,v) list

Changes elements in list of variables from ASCII into desired internal format (c is number of characters, f is format specifier, v is array name)

DEFINE FILE u (m,n,U,ivar),...

Defines record structure of direct access file

DIMENSION s1, s2,...,sk

Reserves storage space for specified arrays

DO n ivar = e1,e2,e3

1. Sets integer variable (ivar) = to the expression (e1)
2. Executes statements through statement number n
3. Increments ivar = ivar+e3
4. If e3 > 0 and ivar < = e2, or e3 < 0 and ivar > = e2, goes back to 2 above, else falls through to statement following statement n

END

Delimits a program unit

ENCODE (c,f,v) list

Changes elements in list of variables into ASCII format (c is number of characters, f is format specifier, v is array name)

Statements (continued)

END FILE u

Closes file currently open on logical unit number u

EQUIVALENCE (var1, var2,...), (varj,vark,...)

Assigns same storage location to each variable within set of parentheses

EXTERNAL name1, name2,...

Informs system that names specified are those of FUNCTION or SUBROUTINE subprograms

FIND (u'r)

Positions direct access file on logical unit number u to record r and sets associated variable to record number r

FORMAT (field specification,...)

Describes format in which one or more records are to be transmitted

FUNCTION name

FUNCTION name (var1, var2,...)

type FUNCTION name (var1, var2,...)

Begins FUNCTION subprogram, indicating program name and any dummy variable names (var); optional type specification can be included

Unconditional GOTO

GOTO n

Transfers control to statement number n

Computed GOTO

GOTO (k1, k2,...kn), e

Transfers control to statement number ki where i = value of expression e. If e < 1 or e > n, no transfer takes place.

Assigned GOTO

GOTO ivar

GOTO ivar, (k1, k2,...kn)

Transfers control to statement most recently associated with ivar by an ASSIGN statement

Arithmetic IF

IF (expression) n1,n2,n3

Transfers control to statement number n depending upon value of the expression

Logical IF

IF (expression) statement

Executes statement if logical expression tests true

Statements (continued)

IMPLICIT type (a1,b2-b3,...),...

Elements a and b represent single (or a range of) letter(s) whose presence as the initial letter of a variable specifies variable to be of that type, if that variable is not explicitly given a type

PAUSE

PAUSE display

Suspends program execution and prints octal constant, decimal constant, or alphanumeric literal display, if one is specified; resume execution by typing carriage return

PRINT f, list

Writes output on logical unit 6 (default LP:); f is format statement label, list is optional data list

Formatted READ

READ(u,f)

READ(u,f) list

READ f,list

Reads at least one logical record from device u (default 1) according to format specification f and assigns values to the variables in the list

Unformatted READ

READ(u)

READ(u) list

Reads one logical record from device u, assigning values to variables in list

Direct Access READ

READ(u'r) list

Reads from logical unit number u, record number r, and assigns values to variables in list

Transfer of Control on Error

END=n

ERR=m

END=n,ERR=m

Optional elements in READ statement list allowing control transfer on end of file conditions or error

RETURN

Returns control to calling program from current subprogram

REWIND u

Repositions logical unit number u to beginning of currently opened file

Statements (continued)

STOP
STOP display
Terminates program execution and prints octal constant, decimal constant, or alphanumeric literal display, if one is specified

SUBROUTINE name
SUBROUTINE name (var1,var2,...)
Begins a SUBROUTINE subprogram, indicating program name and any dummy variable names (var)

TYPE f,list
Writes output on logical unit 7 (default TT:); f is format statement label and list is optional data list

Type Declarations
type var1,var2,...vark
Variable names (var) are assigned specified data type in program unit; type = REAL, INTEGER, DOUBLE PRECISION, LOGICAL, or COMPLEX. An optional byte length may be given after type or after any variable name (e.g., INTEGER * 2).

Formatted WRITE
WRITE (u,f)
WRITE (u,f) list
Causes one or more logical records containing values of variables in list to be written onto logical unit u according to format specification f

Unformatted WRITE
WRITE (u)
WRITE (u) list
Causes one or more logical records containing values of variables in list to be written onto logical unit u

Direct Access WRITE
WRITE (u'r) list
Causes one logical record containing values of variables in list to be written onto record r of logical unit number u

Statements (continued)

Transfer of Control on Error
END=n
ERR=m
END=n,ERR=m

Optional elements in WRITE statement list allowing control transfer on end-of-file conditions or error

Library Functions

(arg type → result type in all cases)

ABS(X)	Real absolute value	Real	→	Real
IABS(I)	Integer absolute value	Integer	→	Integer
DABS(X)	Double precision absolute value	Double	→	Double
CABS(Z)	Complex to Real, absolute value where $Z=(x,y)$ $CABS(Z)=(x^2+y^2)^{1/2}$	Complex	→	Real
FLOAT(I)	Integer to Real conversion	Integer	→	Real
IFIX(X)	Real to Integer conversion IFIX(X) is equivalent to INT(X)	Real	→	Integer
SNGL(X)	Double to Real conversion	Double	→	Real
DBLE(X)	Real to Double conversion	Real	→	Double
REAL(Z)	Complex to Real conversion, obtain real part	Complex	→	Real
AIMAG(Z)	Complex to Real conversion, obtain imaginary part	Complex	→	Real
CMPLX(X,Y)	Real to Complex conversion CMPLX(X,Y)=X+i*Y	Real	→	Complex

Truncation functions return the sign of the argument * largest integer $\leqslant$ | arg |

AINT(X)	Real to Real truncation	Real	→	Real
INT(X)	Real to Integer truncation	Real	→	Integer
IDINT(X)	Double to Integer truncation	Double	→	Integer

Remainder functions return the remainder when the first argument is divided by the second

AMOD(X,Y)	Real remainder	Real	→	Real
MOD(I,J)	Integer remainder	Integer	→	Integer
DMOD(X,Y)	Double precision remainder	Double	→	Double

Maximum value functions return the largest value from among the argument list; $\geqslant 2$ arguments

AMAX0(I,J,...)	Real maximum from Integer list	Integer	→	Real
AMAX1(X,Y,...)	Real maximum from Real list	Real	→	Real
MAX0(I,J,...)	Integer maximum from Integer list	Integer	→	Integer
MAX1(X,Y,...)	Integer maximum from Real list	Real	→	Integer
DMAX1(X,Y,...)	Double maximum from Double list	Double	→	Double

Library Functions (continued)

Minimum value functions return the smallest value from among the argument list; ⩾ 2 arguments

AMIN0(I,J,...)	Real minimum of Integer list	Integer	→	Real
AMIN1(X,Y,...)	Real minimum of Real list	Real	→	Real
MIN0(I,J,...)	Integer minimum of Integer list	Integer	→	Integer
MIN1(X,Y,...)	Integer minimum of Real list	Real	→	Integer
DMIN1(X,Y,...)	Double minimum of Double list	Double	→	Double

The transfer of sign functions return (sign of the second argument) * (absolute value of first argument)

SIGN(X,Y)	Real transfer of sign	Real	→	Real
ISIGN(I,J)	Integer transfer of sign	Integer	→	Integer
DSIGN(X,Y)	Double precision transfer of sign	Double	→	Double

Positive difference functions return the first argument minus the minimum of the two arguments

DIM(X,Y)	Real positive difference	Real	→	Real
IDIM(I,J)	Integer positive difference	Integer	→	Integer

Exponential functions return the value of e raised to the argument power

EXP(X)	e^X	Real	→	Real
DEXP(X)	e^X	Double	→	Double
CEXP(Z)	e^Z	Complex	→	Complex
ALOG(X)	Returns $\log_e(X)$	Real	→	Real
ALOG10(X)	Returns $\log_{10}(X)$	Real	→	Real
DLOG(X)	Returns $\log_e(X)$	Double	→	Double
DLOG10(X)	Returns $\log_{10}(X)$	Double	→	Double
CLOG(Z)	Returns $\log_e$ of complex argument	Complex	→	Complex
SQRT(X)	Square root of Real argument	Real	→	Real
DSQRT(X)	Square root of Double precision argument	Double	→	Double
CSQRT(Z)	Square root of Complex argument	Complex	→	Complex
SIN(X)	Real sine	Real	→	Real
DSIN(X)	Double precision sine	Double	→	Double
CSIN(Z)	Complex sine	Complex	→	Complex
COS(X)	Real cosine	Real	→	Real
DCOS(X)	Double precision cosine	Double	→	Double
CCOS(Z)	Complex cosine	Complex	→	Complex
TANH(X)	Hyperbolic tangent	Real	→	Real
ATAN(X)	Real arc tangent	Real	→	Real
DATAN(X)	Double precision arc tangent	Double	→	Double
ATAN2(X,Y)	Real arc tangent of (X/Y)	Real	→	Real
DATAN2(X,Y)	Double precision arc tangent of (X/Y)	Double	→	Double
CONJG(Z)	Complex conjugate, if Z=X+i*Y COMJG(Z)=Z-i+y	Complex	→	Complex
RAN(I,J)	Returns a random number of uniform distribution over the range 0 to 1. I and J must be integer variables and should be set initially to 0. Resetting I and J to 0 regenerates the random number sequence. Alternate starting values for I and J will generate different random number sequences.	Integer	→	Real

System Subroutines

CALL ASSIGN (lun,name,icnt,mode,control,numbuf)
Allows the association of device and/or filename information with a logical unit number

CALL DATE (array)
Returns the current date in 'array'

CALL EXIT
Equivalent to the STOP statement without message

CALL IDATE (mon,iday,iyear)
Returns three integer values representing the current month, day, and year

RAN (i_1,i_2)
A function call to the random number generator

CALL RANDU (i_1,i_2,x)
A subroutine call to the random number generator

CALL SETERR (number,ncount)
Allows user to specify the disposition of OTS detected errors

CALL USEREX (name)
Pass control to routine 'name' as part of program termination

Logical Device Assignments

Logical Unit Number	Default Device
1	SY:
2	DK:
3	DK:
4	DK:
5	TT: (input)
6	LP:
7	TT: (output)
8	PR:
9	PP:

Switches

/A Include compilation statistics

/D Compile lines with D in column one

/E Read a full 80 columns of each record

/H Print a list of compiler switches

/L :n Specify listing options

n is any one or sum of the following
- 0 list diagnostics only
- 1 list source program only
- 2 list storage map
- 4 list generated code

/N:m Specify maximum number of logical units that may be concurrently open (m=octal constant between 1 and 17)

/O Include options-in-effect

/P Disable the global optimizer

/R:m Specify maximum record size allowed (m=octal constant between 4 and 7777)

/S Suppress Internal Statement Numbers

/T Allocate two words for default length of integer variables

/U Disable USR swapping at run-time

/V Disable vectoring of arrays

/W Enable compiler warning diagnostics

Compiler Error Diagnostics

B	Columns 1-5 of a continuation line are not blank; columns ignored and compilation continues
C	Illegal continuation; continued line ignored and compilation continues
E	Missing END sentinel; an END sentinel is provided
H	Hollerith string or quoted literal string too long; statement aborted
I	Non-FORTRAN character used; character ignored and compilation continues
K	Illegal statement label definition; illegal statement label ignored and compilation continues
L	Line too long; line truncated to 80 characters and compilation continues
M	Multiply defined label; label ignored
P	Statement contains unbalanced parentheses; statement aborted
S	Syntax error; statement aborted
U	Statement not legal FORTRAN statement; statement aborted

Fatal Compiler Error Diagnostics

C	Too many constant subscripts employed in statement; simplify statement
O	Unrecoverable error occurred while Compiler was writing object file (.OBJ); rectify hardware problem or allow more space for output
P	Optimizer push down overflow; simplify complex statements
R	Unrecoverable hardware error occurred while Compiler was reading source file; rectify hardware problem
S	Subexpression stack overflow; simplify complex statements
T	Memory Overflow; break up program into subprograms or compile on larger machine

Fatal Compiler Error Diagnostics (continued)

W	Unrecoverable error occurred while Compiler was writing listing file; rectify hardware problem or allow more space for listing file
Y	Code generation stack overflow; simplify complex statements
Z	Compiler error; report to DIGITAL

Compiler-Secondary Phase-Errors

These error messages are self-explanatory when printed and are not included here. Refer to Appendix G of the *RT-11 System Reference Manual* for detailed definitions.

OTS Error Diagnostics

0	FATAL	NON-FORTRAN ERROR CALL
1	FATAL	INTEGER OVERFLOW
2	FATAL	INTEGER ZERO DIVIDE
3	FATAL	COMPILER GENERATED ERROR
4	WARNING	COMPUTED GO TO OUT OF RANGE
5	COUNT:3	INPUT CONVERSION ERROR
6	IGNORE	OUTPUT CONVERSION ERROR
10	COUNT:3	FLOATING OVERFLOW
11	IGNORE	FLOATING UNDERFLOW
12	FATAL	FLOATING ZERO DIVIDE
13	COUNT:3	SQRT OF NEGATIVE NUMBER
14	FATAL	UNDEFINED EXPONENTIATION OPERATION
15	FATAL	LOG OF NEGATIVE NUMBER
16	FATAL	WRONG NUMBER OF ARGUMENTS
20	FATAL	INVALID CHANNEL NUMBER
21	FATAL	NO AVAILABLE CHANNELS
22	FATAL	INPUT RECORD TOO LONG
23	FATAL	HARDWARE I/O ERROR
24	FATAL	ATTEMPT TO READ/WRITE PAST END OF FILE
25	FATAL	ATTEMPT TO READ AFTER WRITE
26	FATAL	RECURSIVE I/O NOT ALLOWED
27	FATAL	ATTEMPT TO USE DEVICE NOT IN SYSTEM
28	FATAL	OPEN FAILED FOR FILE
29	FATAL	NO ROOM FOR DEVICE HANDLER

OTS Error Diagnostics (continued)

30	FATAL	NO ROOM FOR BUFFERS
31	FATAL	NO AVAILABLE RT-11 CHANNEL
32	FATAL	FMTD–UNFMTD–RANDOM I/O TO SAME FILE
33	FATAL	ATTEMPT TO READ PAST END OF RECORD
34	FATAL	UNFMTD I/O TO TTY OR LPT
35	FATAL	ATTEMPT TO OUTPUT TO READ ONLY FILE
36	FATAL	BAD FILE SPECIFICATION STRING
37	FATAL	RANDOM ACCESS READ/WRITE BEFORE DEFINE FILE
38	FATAL	RANDOM I/O NOT ALLOWED ON TTY OR LPT
39	FATAL	RECORD LARGER THAN RECORD SIZE IN DEFINE FILE
40	FATAL	REQUEST FOR A BLOCK LARGER THAN 65535
41	FATAL	DEFINE FILE ATTEMPTED ON AN OPEN UNIT
42	FATAL	MEMORY OVERFLOW COMPILING OBJECT TIME FORMAT
43	FATAL	SYNTAX ERROR IN OBJECT TIME FORMAT
44	FATAL	2ND RECORD REQUEST IN ENCODE/ DECODE
45	FATAL	INCOMPATIBLE VARIABLE AND FORMAT TYPES
46	FATAL	INFINITE FORMAT LOOP
47	FATAL	ATTEMPT TO STORE OUTSIDE PARTITION
59	WARNING	USR NOT LOCKED
60	FATAL	STACK OVERFLOWED
61	FATAL	ILLEGAL MEMORY REFERENCE
62	FATAL	FORTRAN START FAIL
63	FATAL	ILLEGAL INSTRUCTION

FORMAT OF SYSLIB CALLS

File-Oriented Operations

CLOSEC – CALL CLOSEC (chan)
Closes the specified channel

File-Oriented Operations (continued)

IDELET – i = IDELET (chan,dblk[,count])
Deletes the file from the specified device

IENTER – i = IENTER (chan,dblk,length,[count])
Creates a new file for output

IRENAM – i = IRENAM (chan,dblk)
Changes the name of the indicated file to a new name

LOOKUP – i = LOOKUP (chan,dblk[,count])
Opens an existing file for I/O via the specified channel

Data Transfer Functions

IRCVD – i = IRCVD (buff,wcnt)

IRCVDC – i = IRCVDC (buff,wcnt,crtn)

IRCVDF – i = IRCVDF (buff,wcnt,area,crtn)

IRCVDW – i = IRCVDW (buff,wcnt)
Receives data; allows a job to read messages or data sent by another job in an F/B environment; the four modes correspond to the IREAD, IREADC, IREADF, and IREADW modes (F/B only)

IREAD – i = IREAD (wcnt,buff,blk,chan)
Transfers data via the specified channel to a memory buffer and returns control to the user program when the transfer request is entered in the I/O queue; no special action is taken upon completion of I/O

IREADC – i = IREADC (wcnt,buff,blk,chan,crtn)
Transfers data via the specified channel to a memory buffer and returns control to the user program when the transfer request is entered in the I/O queue; upon completion of the read, the specified assembly language routine is entered as an asynchronous completion routine

Data Transfer Functions (continued)

IREADF – i = IREADF (wcnt,buff,blk,chan,area,crtn)
Transfers data via the specified channel to a memory buffer and returns control to the user program when the transfer request is entered in the I/O queue; upon completion of the read, the specified FORTRAN subroutine is entered as an asynchronous completion routine

IREADW – i = IREADW (wcnt,buff,blk,chan)
Transfers data via the specified channel to a memory buffer and returns control to the program only after the transfer is complete

ISDAT – i = ISDAT (buff,wcnt)

ISDATC – i = ISDATC (buff,wcnt,crtn)

ISDATF – i = ISDATF (buff,wcnt,area,crtn)

ISDATW – i = ISDATW (buff,wcnt)
Allows the user to send messages or data to the other job in an F/B environment; the four modes correspond to the IWRITC, IWRITE, IWRITF, and IWRITW modes (F/B only)

ITTINR – i = ITTINR ()
Inputs one character from the console keyboard

ITTOUR – i = ITTOUR (char)
Transfers one character to the console terminal

IWAIT – i = IWAIT (chan)
Waits for completion of all I/O on a specified channel (commonly used with the IREAD and IWRITE functions)

IWRITC – i = IWRITC (wcnt,buff,blk,chan,crtn)
Transfers data via the specified channel to a device and returns control to the user program when the transfer request is entered in the I/O queue; upon completion of the write, the specified assembly language routine is entered as an asynchronous completion routine

Data Transfer Functions (continued)

IWRITE – i = IWRITE (wcnt,buff,blk,chan)
Transfers data via the specified channel to a device and returns control to the user program when the transfer request is entered in the I/O queue; no special action is taken upon completion of the I/O

IWRITF – i = IWRITF (wcnt,buff,blk,chan,area,crtn)
Transfers data via the specified channel to a device and returns control to the user program when the transfer request is entered in the I/O queue; upon completion of the write, the specified FORTRAN subroutine is entered as an asynchronous completion routine

IWRITW – i = IWRITW (wcnt,buff,blk,chan)
Transfers data via the specified channel to a device and returns control to the user program only after the transfer is complete

MWAIT – CALL MWAIT
Waits for messages to be processed (F/B only)

PRINT – CALL PRINT (string)
Outputs an ASCII string to the terminal

Channel-Oriented Operations

ICDFN – i = ICDFN (num)
Defines additional channels for doing I/O

ICHCPY – i = ICHCPY (chan,ochan)
Allows access to files currently open in the other job's environment (F/B only)

ICSTAT – i = ICSTAT (chan,addr)
Returns the status of a specified channel (F/B only)

IFREEC – i = IFREEC (chan)
Returns the specified RT-11 channel to the available pool of channels

Channel-Oriented Operations (continued)

IGETC – i = IGETC ()
Allocates an RT-11 channel and marks it in use to the FORTRAN I/O system

ILUN – i = ILUN (lun)
Returns the RT-11 channel number with which the FORTRAN logical unit is associated

IREOPN – i = IREOPN (chan,cblk)
Restores the parameters stored via an ISAVES function and reopens the channel for I/O

PURGE – CALL PURGE (chan)
Clears out a channel

Device And File Specifications

IASIGN – i = IASIGN (lun,idev[,ifilex[,isize[,itype]]])
Sets information in the FORTRAN logical unit table

ICSI – i = ICSI (outspc,defext,[cstring],[switch],n)
Calls the RT-11 CSI in special mode to decode file specifications and switches

Timer Support Operations

CVTTIM – CALL CVTTIM (time,hrs,min,sec,tick)
Converts a 2-word internal format time to hours, minutes, seconds, and ticks

GTIM – CALL GTIM (time)
Gets time of day

ICMKT – i = ICMKT (id,time)
Cancels an unexpired mark time request (F/B only)

ISCHED – i = ISCHED (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be entered at the specified time of day as an asynchronous completion routine (F/B only)

Timer Support Operations (continued)

ISLEEP – i = ISLEEP (hrs,min,sec,tick)
Suspends mainline execution of the running job for a specified amount of time, completion routines continue to run (F/B only)

ITIMER – i = ITIMER (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be entered as an asynchronous completion routine when the time interval specified has elapsed (F/B only)

ITWAIT – i = ITWAIT (time)
Suspends the running job for a specified amount of time; completion routines continue to run (F/B only)

IUNTIL – i = IUNTIL (hrs,min,sec,tick)
Suspends the mainline execution of the running job until a specified time-of-day; completion routines continue to run (F/B only)

JTIME – CALL JTIME (hrs,min,sec,tick,time)
Converts hours, minutes, seconds, and ticks into 2-word internal format time

MRKT – i = MRKT (id,crtn,time)
Marks time, i.e., schedules asynchronous routine to be entered after a specified interval (F/B only)

SECNDS – a = SECNDS (atime)
Returns the current system time in seconds past midnight minus a specified time

TIMASC – CALL TIMASC (time,strng)
Converts a specified 2-word internal format time into an 8-character ASCII string

TIME – CALL TIME (strng)
Returns the current system time-of-day as an 8-character ASCII string

RT-11 Services

CHAIN – CALL CHAIN (dblk,var,wcnt)
Chains to another program (in the background job only)

DEVICE – CALL DEVICE (ilist)
Specifies actions to be taken on normal or abnormal program termination

GTJB – CALL GTJB (addr)
Returns the parameters of this job

IDSTAT – i = IDSTAT (devnam,cblk)
Returns the status of the specified device

IFETCH – i = IFETCH (devnam)
Loads device handlers into memory

IQSET – i = IQSET (qleng)
Expands the size of the RT-11 monitor queue from the free space managed by the FORTRAN system

ISPFN – i = ISPFN (code,chan[,wcnt,buff,blk])

ISPFNC – i = ISPFNC (code,chan,wcnt,buff,blk,crtn)

ISPFNF – i = ISPFNF (code,chan,wcnt,buff,blk,area, crtn)

ISPFNW – i = ISPFNW (code,chan[,wcnt,buff,blk])
Performs special functions on the magtape, cassette, and floppy units; the four modes correspond to the IWRITE, IWRITC, IWRITF, and IWRITW modes

ITLOCK – i = ITLOCK ()
Indicates whether the USR is currently in use by another job and performs a LOCK if possible (F/B only)

LOCK – CALL LOCK
Makes the RT-11 monitor USR permanently resident until an UNLOCK function is executed

RT-11 Services (continued)

RCHAIN – CALL RCHAIN (flag,var,wcnt)
Allows a program to access variables passed across a chain

RCTRLO – CALL RCTRLO
Enables output to the terminal by cancelling the effect of a previously typed CTRL O, if any

RESUME – CALL RESUME
Causes the mainline execution of a job to resume where it was suspended with a SUSPND function call (F/B only)

SUSPND – CALL SUSPND
Suspends mainline execution of the running job; completion routines continue to execute (F/B only)

UNLOCK – CALL UNLOCK
Releases the USR if a LOCK was performed; the user program is swapped in if required

AJFLT – a = AJFLT (jsrc)
Converts specified INTEGER*4 value to REAL*4 and returns result as function value

DJFLT – d = DJFLT (jsrc)
Converts specified INTEGER*4 value to REAL*8 and returns result as function value

IAJFLT – i = IAJFLT (jsrc,ares)
Converts specified INTEGER*4 value to REAL*4 and stores the result

IDJFLT – i = IDJFLT (jsrc,dres)
Converts specified INTEGER*4 value to REAL*8 and stores the result

IJCVT – i = IJCVT (jsrc[,ires])
Converts specified INTEGER*4 value to INTEGER*2

*INTEGER*4 Support Functions (continued)*

JADD – i = JADD (jopr1,jopr2,jres)
Computes the sum of two INTEGER*4 values

JAFIX – i = JAFIX (asrc,jres)
Converts a REAL*4 value to INTEGER*4

JCMP – i = JCMP (jopr1,jopr2)
Compares two INTEGER*4 values and returns an INTEGER*2 value which reflects the signed comparison result

JDFIX – i = JDFIX (dsrc,jres)
Converts a REAL*8 value to INTEGER*4

JDIV – i = JDIV (jopr1,jopr2,jres[,jrem])
Computes the quotient of two INTEGER*4 values

JICVT – i = JICVT (isrc,jres)
Converts an INTEGER*2 value to INTEGER*4

JJCVT – CALL JJCVT (jsrc)
Converts 2-word internal time formats to INTEGER*4 format, and vice versa

JMOV – i = JMOV (jsrc,jdest)
Assigns an INTEGER*4 value to a variable

JMUL – i = JMUL (jopr1,jopr2,jres)
Computes the product of two INTEGER*4 values

JSUB – i = JSUB (jopr1,jopr2,jres)
Computes the difference between two INTEGER*4 values

Character String Functions

CONCAT – CALL CONCAT (a,b,out[,len[,err]])
Concatenates two variable length strings

Character String Functions (continued)

GETSTR – CALL GETSTR (lun,out,len[,err])
Reads a character string from a specified FORTRAN logical unit

INDEX – CALL INDEX (a, pattrn , [i] , m) or
m = INDEX (a, pattrn [,i])
Returns the location in one string of the first occurrence of another string

INSERT – CALL INSERT (in,out,i[,m])
Inserts a string at a specified position in another string

LEN – i = LEN (a)
Returns the number of characters in string a

PUTSTR – CALL PUTSTR (lun,in,char)
Writes a variable-length character string on a specified FORTRAN logical unit

REPEAT – CALL REPEAT (in,out,i[,len[,err]])
Concatenates a specified string with itself to provide an indicated number of copies and stores the resultant string

SCOMP – CALL SCOMP (a,b,i) or i = ISCOMP (a,b)
Compares two character strings

SCOPY – CALL SCOPY (in,out[,len[,err]])
Copies a character string from one array to another

STRPAD – CALL STRPAD (a,i[,err])
Pads a variable-length string on the right with blanks to create a new string of a specified length

SUBSTR – CALL SUBSTR (in,out,i[,len])
Copies a substring from a specified string

TRANSL – CALL TRANSL (in,out,r[,p])
Replaces one string with another after performing character modification

Character String Functions (continued)

TRIM – CALL TRIM (a)
Removes trailing blanks from a character string

VERIFY – CALL VERIFY (a,b,i) or i = IVERIF (a,b)
Indicates whether characters in one string appear in another

RAD50 Conversion Operations

IRAD50 – n = IRAD50 (icnt,input,output)
Converts ASCII characters to RAD50, returning the number of characters converted

R50ASC – CALL R50ASC (icnt,input,output)
Converts RAD50 characters to ASCII

RAD50 – a = RAD50 (input)
Converts six ASCII characters, returning a REAL*4 result which is the 2-word RAD50 value

Miscellaneous Services

IADDR – i = IADDR (var)
Obtains the memory address of a specified entity

INTSET – i = INTSET (vect,pri,id,crtn)
Establishes a specified FORTRAN subroutine as an interrupt service routine at a specified priority

IPEEK – i = IPEEK (iaddr)
Returns the value of a word located at a specified absolute memory address

IPOKE – CALL IPOKE (iaddr,ivalue)
Stores an integer value in an absolute memory location

ISPY – i = ISPY (ioff)
Returns the integer value of the word located at a specified offset from the RT-11 resident monitor

Commands

ASK var
: Request input from current input device

ASK "text", var
: Output text and store input as a variable (var)

COMMENT
: Ignore remainder of line

DO n.m
: Execute line n.m; return to command following DO command

DO n
: Execute all group n lines; return to command following DO command upon completion or upon encountering a RETURN

DO ALL
: Execute entire program as a subroutine

DO var
: Execute the line or group of lines defined by the variable (var)

ERASE
: Erase symbol table

ERASE n.m
: Erase line n.m

ERASE n
: Erase all group n lines

ERASE ALL
: Erase entire program; clear all variables

ERASE TEXT
: Erase text only; do not erase symbol table

Commands (continued)

FOR I = X,Y,Z; (commands)

Used to implement loops; the variable I is set equal to the value of X. The commands are executed following which I is incremented by the value of Y; the commands are again executed and I again incremented, and so on, until the value of I exceeds the value of Z

GO

Start program at lowest numbered line number

GO n.m

Transfer control to line n.m

GO n

Transfer control to lowest numbered statement in group n

GO?

Start at lowest numbered line and trace entire program until another ? or an error is encountered, or until program completion

IF (X), L1,L2,L3

If the value of X < 0, transfer control to line number L1; if X = 0, transfer control to line number L2; if X > 0, transfer control to line number L3. X may be a defined variable, value, or expression. If a line number is not specified, proceed to next sequential command

KILL

Stop all I/O devices. Error code ?09 is printed

LIBRARY INPUT 1, TEST/T

Attempt to open old file SY:TEST.FCL as ASCII file number 1. If file does not exist, FOCAL returns an error message

LIBRARY OPEN 1, TEST/T

Attempt to open old file SY:TEST.FCL as ASCII file number 1. If file does not yet exist, it is created

Commands (continued)

LIBRARY MAKE 1, TEST/T
Create new file called SY:TEST.FCL as ASCII file number 1

LIBRARY CLOSE 1
Terminate all activity with file number 1

LIBRARY TYPE 1, ⟨type args⟩
Type output to file 1

LIBRARY WRITE 1, ⟨write arg⟩
Write output to file 1

LIBRARY ASK 1, ⟨ask args⟩
Read ASK input from file 1

LIBRARY RUN TEST ⟨args⟩
ERASE all text and variables, read in FOCAL program saved in file SY:TEST.FCL, and begin execution at the line or group specified. (If args are left out, the first statement in the program is assumed)

LIBRARY GET TEST
Read in file SY:TEST.FCL and merge it with the current program (must be terminated by a carriage return)

LIBRARY NEXT TEST ⟨arg⟩
ERASE all text, leaving variables intact; "RUN" program saved in file "TEST.FCL." Execution continues at the start of the program or at the line/group number specified in arg

LIBRARY SAVE TEST
Program text is saved in file called SY:TEST.FCL. If another file by that name already exists, it is deleted

LIBRARY DELETE TEST
The RT-11 file SY:TEST.FCL is deleted

Commands (continued)

MODIFY n.m

Enable editing of line n.m. Editing commands are:

CTRL L	next occurrence
CTRL G	new search character
RUBOUT	delete last character
CTRL U BACKARROW	delete back to line feed
⟨CR⟩	delete remainder of line
⟨LF⟩	preserve rest of line

OPERATE

Force all pending output to currently selected output device; current output device is not altered

OPERATE T

Select terminal printer

OPERATE K

Select terminal keyboard for input

OPERATE P

Select high-speed paper tape punch for output

OPERATE R

Select high-speed paper tape reader for input

OPERATE RP

Select both high-speed reader and punch for I/O

OPERATE TK

Select both terminal keyboard and printer for I/O

OPERATE L

Select line printer for output

QUIT

Return control to user (command code)

Commands (continued)

RETURN
Terminate DO subroutines, returning to original sequence

SET var=exp
Perform arithmetic assignment. The variable on the left side of the "=" is set equal to the value of the expression on the right

TYPE exp
Evaluate expression and type "=" followed by result in current output format

TYPE exp,exp
Compute each expression and type the resultant values

TYPE "text"
Type text (may be followed by ! to generate ⟨CR⟩ ⟨LF⟩ or # to generate only ⟨CR⟩)

TYPE $
Type symbol table. Must be terminated by ⟨CR⟩ only

WRITE
Type out entire program

WRITE ALL
Same as WRITE

WRITE n
Type out all group n lines

WRITE n.m
Type out line n.m

XECUTE FUNC (#,)
Call functions (FUNC(#,arg)) without need for a dummy SET statement

Functions

FABS(exp)
Returns absolute (positive) value of expression

FADC(ch)
Provides access to A/D channels

FCHR(arg)
Accepts and/or prints ASCII codes

FCLK()
Returns value of the time elapsed

FCOS(ang)
Calculates cosine of a specified angle in radians

FERR(n)
FERR(n.m)
Intercepts a FOCAL error and performs a DO to the line (n.m) or group (n)

FEXP(arg)
Exponential function

FINT(vec,n,prior,CSR addr,mask)
Logically connects a line or group of user program with a device and its interrupt vector

FITR(exp)
Provides the integer part of a number

FLN(arg)
Natural logarithm function

FLOG(arg)
Base ten logarithm

FPRM(para,val)
Alters FOCAL internal parameters

Functions (continued)

FQUE(cnt,n,interval,delay,prior)
Schedules a line or group of user program to be performed a specified number of times at regular time intervals at a specified software priority

FRAN()
Generates a random value between −1 and 1

FSBR(n,arg)
Calls program group specified as a subroutine

FSGN(arg)
Sign function

FSIN(ang)
Calculates sign of the specified angle in radians

FSQT(exp)
Computes square root of expression

FX(func,UNIBUS-addr,data)
Controls additional device options or non-standard peripherals or references memory storage

Error Messages

(i) Internal FOCAL error
(o) Operational error
(r) A run-time error

?00 Manual restart from location 0 or by CTRL C (r)

?01 Illegal line number

?02 Illegal variable or function name

?03 Unmatching parentheses

?04 Illegal command

?05 Nonexistent line number

Error Messages (continued)

?06 Nonexistent group or line number in DO

?07 Illegal format in SET or FOR

?08 Double or missing operators in expression

?09 Stack overflow, nonexistent device, or bad address specification

?10 Memory filled by text or command line too long (o)

?11 Memory filled by variables or no room for variables (o)

?12 Exponent range greater than E+38 (o)

?13 Disallowed bus address in "FX" (o)

?14 Division by zero attempted (r)

?15 Attempt to exponentiate to a negative power or power too large (r)

?16 Too many characters in input data (r)

?17 Square root of negative number (r)

?18 Input buffer overflow

?19 Subscript out of range (r)

?20 Invalid argument to function call (o)

?21 Bad argument to function call (o)

?22 Unable to perform specified interrupt linkage or unable to schedule desired routine (o) (r)

?23 Symbol table shuffle error (i)

?24 Memory allocation error

FOCAL-11 (continued)

Error Messages (continued)

?25 Internal memory error (i)

?26 Illegal RELM request (o)

?27 General I/O error (r)

?28 Insufficient resources (r)

?29 File number out of range (o)

?30 Illegal format code (o)

?31 File specification syntax error (o)

?32 Fatal write error encountered (unopened channel, etc.) (o)

?33 Attempt to read (or write) past EOF (o)

?34 File was not found (r)

?35 Illegal library command encountered while performing a LIBRARY GET, RUN, or NEXT (o)

?36 Internal virtual file error (i)

?37 Illegal floating point call (i)

?38 Integer overflow error (range outside +32,767 to -32,768) (r)

?39 Interrupt linkage error (i)

BATCH COMMAND LANGUAGE

Command Field Switches

/BAN Print header of job on the log file

/NOBAN Do not print a job header

/CRE Produce a cross reference listing during assembly

Command Field Switches (continued)

/NOCRE	Do not produce a cross reference listing
/DEL	Delete input files after operation is complete
/NODEL	Do not delete input files after operation is complete
/DOL	Following data may have a $ in the first character position of a line; reading of the data is terminated by $JOB, $SEQUENCE, $EOD, $EOJ or by physical end-of-file
/NODOL	Following data may not have a $ in the first character position ($ in first character position signifies a BATCH control command)
/LIB	Include the default library in the link operation
/NOLIB	Do not include the default library in the link operation
/LIS	Produce a temporary listing file on the listing device or write data images on the log device
/NOLIS	Do not produce a temporary listing file
/MAP	Produce a temporary linkage map on the listing device
/NOMAP	Do not produce a linkage map
/OBJ	Produce a temporary object file as output of compilation or assembly or include temporary object files in link
/NOOBJ	Do not produce a temporary object file or do not include temporary object files in link

Command Field Switches (continued)

/RT11	Set BATCH to operate in RT-11 mode
/NORT11	Do not set BATCH to operate in RT-11 mode
/RUN	Link and execute programs compiled since the last "link-and-go" operation or start of job
/NORUN	Do not link and execute the program after performing the specified command
/TIM	Write time of day to the log file when commands are executed
/NOTIM	Do not write time of day to log file
/UNI	Check for unique spelling of switches and keynames
/NOUNI	Do not check for unique spelling
/WAI	Pause to wait for operator action; type ⟨CR⟩ to continue
/NOWAI	Do not pause for operator action
/WRI	Indicate that the operator is to WRITE-ENABLE a specified device or volume
/NOWRI	Indicate that no writes are allowed or that the specified volume is read-only; the operator is informed

Specification Field Switches

/BAS	BASIC source file
/EXE	Indicates runnable program image file to be created as result of a link operation
/FOR	FORTRAN source file

Specification Field Switches (continued)

/INP	Input file
/LIB	Library file to be included in link operation (prior to default library)
/LIS	Listing file
/LOG	Indicates that the device is a logical device name
/MAC	MACRO source file
/MAP	Linker map file
/OBJ	Object file (output of assembly or compilation)
/OUT	Output file
/PHY	Indicates physical device name
/SOU	Indicates source file
/VID	Volume identification

Commands

$BASIC [/switch] [dev:filnam.ext/sw] [!comments]
Compiles a BASIC source program; /switch may be /RUN, /NORUN, /LIS, /NOLIS; /sw may be /BAS, /SOU, /INP

$CALL dev:filnam.ext [!comments]
Transfers control to another BATCH file, executes that BATCH file, returns to calling BATCH stream

$CHAIN dev:filnam.ext [!comments]
Relinquishes control to another BATCH file

$COPY[/switch] dev:filnam.ext/OUTPUT-dev:filnam1.ext[/INPUT] [!comments]
Copies files; /switch may be /DEL, /NODEL

Commands (continued)

$CREATE[/switch] dev:filnam.ext [!comments]
Creates new files from data included in BATCH stream; /switch may be /DOL, /NODOL, /LIS, /NOLIS

$DATA[/switch] [!comments]
Indicates the start of data; /switch may be /DOL, /NODOL, /LIS, /NOLIS

$DELETE dev:filnam1.ext[,dev:filnam2.ext,...]–
[!comments]
Deletes files

$DIRECTORY [dev:filnam.ext/LIST] [dev:filnam.ext]–
[/INPUT] [!comments]
Provides a directory of the specified device

$DISMOUNT[/switch] ldn:[/LOGICAL] [!comments]
Signals the operator to dismount a volume from a device; deassigns logical device name; /switch may be /WAI, /NOWAI

$EOD [!comments]
Indicates the end of data

$EOJ [!comments]
Indicates the end of a job

$FORTRAN[/switch] [filnam1.ext/sw]–
[filnam2.ext/OBJECT] [filnam3.ext/LIST]–
[filnam4.ext/EXECUTE] [filnam5.ext/MAP]–
[filnam6.ext/LIBRARY] [!comments]
Compiles a FORTRAN source program; /switch may be /RUN, /NORUN, /OBJ, /NOOBJ, /LIS, /NOLIS, /MAP, /NOMAP, /DOL, /NODOL; /sw may be /FOR, /SOU, /INP

$JOB[/switch] [/switch2]...[/switchn] [!comments]
Indicates the start of a job; /switch may be /BAN, /NOBAN, /LIS, /NOLIS, /RT11, /NORT11, /TIM, /NOTIM, /UNI, /NOUNI

Commands (continued)

$LIBRARY mylib [!comments] or $LIBRARY mylib +-
FORLIB [!comments]
Specifies libraries that are to be used in linkage operations

$LINK[/switch] [filnam1.ext/OBJECT]-
[filnam2.ext/LIBRARY] [filnam3.ext/MAP]-
[filnam4.ext/EXECUTE] [!comments]
Links modules for execution; /switch may be /LIB, /NOLIB, /MAP, /NOMAP, /OBJ, /NOOBJ, /RUN, /NORUN

$MACRO[/switch] [filnam1.ext/sw]-
[filnam2.ext/OBJECT] [filnam3.ext/LIST]-
[filnam4.ext/MAP] [filnam5.ext/LIBRARY]-
[filnam6.ext/EXECUTE] [!comments]
Assembles MACRO Source program; /switch may be /RUN, /NORUN, /OBJ, /NOOBJ, /LIS, /NOLIS, /CRE, /NOCRE, /MAP, /NOMAP, /DOL, /NODOL, /LIB, /NOLIB; /sw may be /MAC, /SOU, /INP

$MESSAGE[/switch] message [!comments]
Issues a message to the operator; /switch may be /WAI, /NOWAI

$MOUNT[/switch] dev:[/PHYSICAL][/VID=x]-
[ldn:/LOGICAL] [!comments]
Signals the operator to mount a volume on a device; optionally assigns a logical device name; /switch may be /WAI, /NOWAI, /WRI, /NOWRI

$PRINT[/switch] dev:filnam.ext[/INPUT]-
[,dev:filnam2.ext,...] [!comments]
Prints files on device LST:; /switch may be /DEL, /NODEL

$RT11 [!comments]
Specifies that the following lines are RT-11 mode command lines; $ causes exit from RT-11 mode

Commands (continued)

$RUN dev:filnam.ext [!comments]
Causes a program to execute

$SEQUENCE id [!comments]
Assigns an arbitrary identification number to a job

RT-11 Mode Control Commands

(# means space or tab)

#NOTTY	Do not write terminal I/O to the log file
#TTYIN	Write only terminal input to the log file
#TTYIO	Write terminal I/O to the log file
#TTYOUT	Write only terminal output to the log file (default)
'text'	Command to BATCH run-time handler; text can be:

CTY	accept input from console terminal
FF	output current log buffer
NL	insert new line in BATCH stream
x	insert contents of variable x
"message"	direct message to console terminal

Operating Switches

/N Compile but do not execute

/T:n n=0 sets the /NOTIME switch as default on $JOB;
n=1 sets /TIME as default

Operating Switches (continued)

/U Detach BATCH run-time handler from RT-11 monitor for subsequent removal

/V Print BATCH compiler version number

/X Indicate that input is a precompiled BATCH program

Operator Directives

\ A Change input source to be console terminal

\ B Change input source to be BATCH stream

\ C Send following characters to log device

\ D Consider following characters as user data

\ E Send following characters to RT-11 monitor

\ F Force the output of the current log block; must be followed immediately by another directive or BATCH will terminate with FE

\ Hn Help function to change the logging mode where n specifies the following:

0 Log only .TTYOUT and .PRINT

1 Log .TTYOUT, .PRINT, and .TTYIN

2 Do not log .TTYOUT, .PRINT, and .TTYIN

3 Log only .TTYIN

Error Messages

Most BATCH error messages are self-explanatory; those that are 2-letter codes are included here; refer to the *RT-11 System Reference Manual,* Chapter 12, for detailed explanations of all error messages.

BATCH COMMAND LANGUAGE (continued)

Error Messages (continued)

BC Bad code found in the control file by the BATCH handler

FE A Forced End occurred due to an \F⟨CR⟩ directive in the control file or typed at the terminal

IO Input or Output error occurred; log file may be full

LU Lock Up occurred in the BATCH handler because it could not find a free channel

BOOTSTRAPS

UNIT 0 Bootstraps

Address	DECtape	Contents RK11 Disk	RF11 Disk
001000	012700	012700	012700
001002	177344	177406	177466
001004	012710	012710	005010
001006	177400	177400	005040
001010	012740	012740	012740
001012	004002	000005	177400
001014	005710	105710	012740
001016	100376	100376	000005
001020	012710	005007	105710
001022	000003		100376
001024	105710		005007
001026	100376		
001030	012710		
001032	000005		
001034	105710		
001036	100376		
001040	005007		

Unit 0 Bootstraps (continued)

Address	RX01 Disk	Contents RP02/RP03 Disk	RJS03/4 Disk
001000	012702	012705	012705
001002	1002n7*	176716	172044
001004	012701	012715	012745
001006	177170	177400	177400
001010	130211	012745	012745
001012	001776	000005	000071
001014	112703	105715	032715
001016	000007	100376	100200
001020	010100	005007	001775
001022	010220		100762
001024	000402		005007
001026	012710		
001030	000001		
001032	006203		
001034	103402		
001036	112711	*n = 4 for unit 0	
001040	111023	6 for unit 1	
001042	030211		
001044	001776		
001046	100756		
001050	103766		
001052	105711		
001054	100771		
001056	005000		
001060	022710		
001062	000240		
001064	001347		
001066	122702		
001070	000247		
001072	005500		
001074	005007		

Unit n Bootstrap RK11 Disk

Address	Contents		
001000	012700	xxxxxx=020000	unit 1
001002	177406	040000	unit 2
001004	012760	060000	unit 3
001006	xxxxxx	100000	unit 4
001010	000004	120000	unit 5
001012	012700	140000	unit 6
001014	177406	160000	unit 7
001016	012710		
001020	177400		
001022	012740		
001024	000005		
001026	105710		
001030	100376		
001032	005007		

1. Enter 001000 in switch register.
2. Press LOAD ADRS.
3. Deposit the appropriate bootstrap from the preceding tables.
4. Enter 001000 in switch register and examine each location for correctness.
5. Ensure that device is ready.
6. Enter 001000 in switch register.
7. Press LOAD ADRS and START.

BM792-YB Hardware Bootstrap

173100		LOAD ADRS
177462	(RF11 Disk)	START
177406	(RK11,RK05 Disk)	
177344	(DECtape)	

MR11-DB Hardware Bootstrap

773100	(RF11 Disk)	LOAD ADRS
773110	(RK11,RK05 Disk)	START
773120	(DECtape)	

NOTES