BOSS XVM USER'S MANUAL

DEC-XV-OBUAA-A-D

XVM Systems

digital

BOSS XVM USER'S MANUAL

DEC-XV-OBUAA-A-D

digital equipment corporation · maynard. massachusetts

First Printing, December 1975

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM		TYPESET-11

CONTENTS

CONTENTS (Cont'd)

CONTENTS (Cont'd)

FIGURES

TABLES

LIST OF ALL XVM MANUALS

The following is a list of all XVM manuals and their DEC numbers, including the latest version available. Within this manual, other XVM manuals are referenced by title only. Refer to this list for the DEC numbers of these referenced manuals.

BOSS XVM USER'S MANUAL	DEC-XV-OBUAA-A-D
CHAIN XVM/EXECUTE XVM UTILITY MANUAL	DEC-XV-UCHNA-A-D
DDT XVM UTILITY MANUAL	DEC-XV-UDDTA-A-D
EDIT/EDITVP/EDITVT XVM UTILITY MANUAL	DEC-XV-UETUA-A-D
8TRAN XVM UTILITY MANUAL	DEC-XV-UTRNA-A-D
FOCAL XVM LANGUAGE MANUAL	DEC-XV-LFLGA-A-D
FORTRAN IV XVM LANGUAGE MANUAL	DEC-XV-LF4MA-A-D
FORTRAN IV XVM OPERATING ENVIRONMENT MANUAL	DEC-XV-LF4EA-A-D
LINKING LOADER XVM UTILITY MANUAL	DEC-XV-ULLUA-A-D
MAC11 XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMLAA-A-D
MACRO XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMALA-A-D
MTDUMP XVM UTILITY MANUAL	DEC-XV-UMTUA-A-D
PATCH XVM UTILITY MANUAL	DEC-XV-UPUMA-A-D
PIP XVM UTILITY MANUAL	DEC-XV-UPPUA-A-D
SGEN XVM UTILITY MANUAL	DEC-XV-USUTA-A-D
SRCCOM XVM UTILITY MANUAL	DEC-XV-USRCA-A-D
UPDATE XVM UTILITY MANUAL	DEC-XV-UUPDA-A-D
VP15A XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVPAA-A-D
VT15 XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVTAA-A-D
XVM/DOS KEYBOARD COMMAND GUIDE	DEC-XV-ODKBA-A-D
XVM/DOS READER'S GUIDE AND MASTER INDEX	DEC-XV-ODGIA-A-D
XVM/DOS SYSTEM MANUAL	DEC-XV-ODSAA-A-D
XVM/DOS USERS MANUAL	DEC-XV-ODMAA-A-D
XVM/DOS V1A SYSTEM INSTALLATION GUIDE	DEC-XV-ODSIA-A-D
XVM/RSX SYSTEM MANUAL	DEC-XV-IRSMA-A-D
XVM UNICHANNEL SOFTWARE MANUAL	DEC-XV-XUSMA-A-D

PREFACE

This manual is the prime document for the BOSS XVM(BOSS) Batch Operating Software System and describes its features, concepts, language and operating procedures. The first four chapters provide a general description of the BOSS System components, both hardware and software, and fundamental system concepts. Brief descriptions of all system programs with applicable document numbers are contained in Chapter 4.

The remaining four chapters deal with the BOSS System at a more technical level. They are primarily concerned with the syntactical structure and operation of the batch commands. The job control language is presented in Chapter 6. The information in this chapter is given in a reference format with each command in alphabetical order. The information in these chapters is directed primarily to readers who are familiar with either the FORTRAN IV language or the DIGITAL XVM(XVM) assembly language, MACRO XVM(MACRO). FORTRAN I/0 considerations are specifically covered in the FORTRAN IV Operating Environment Manual. In addition, there are special BOSS commands for the Peripheral Interchange Program (PIP).

A quick reference summary of PIP command strings, operating procedures, error messages and string examples is provided in Appendix A.

Detailed information on the operations of the XVM/DOS Monitor and its file structure as well as interactive operating procedures for preparing user-created applications via XVM/DOS software are provided in the XVM/DOS Users Manual.

The XVM/DOS Keyboard Command Guide contains a chapter of reference information on BOSS XVM.

CHAPTER 1
INTRODUCTION

1.1 BATCH OPERATING SYSTEM

BOSS XVM(BOSS) is the DIGITAL XVM(XVM) Batch Operating System Software. It is a dedicated (stand-alone) Job-System Interface residing with the Disk Operating System XVM/DOS(DOS) in a "DUOPS" configuration (see Figure 1-1). BOSS is a general purpose scientific, commercial, and applications programming package designed for computational centers in private industry and universities.

BOSS is a comprehensive Batch system with a full range of source and run-time software facilities including languages and peripheral I/O Device Handlers. Its multi-job management includes Job Accounting and Line Editing Features. The System's card resident Command Language incorporates Peripheral Interchange, Editing, and Job Control functions.

Batch is a method of serial computer program processing which maximizes the number of programs that can be run in a given time. It optimizes cyclical non-interactive application tasks, since jobs can easily be prepared and desk-checked off-line.

1.2 DUAL OPERATING ENVIRONMENTS

The system required for a BOSS user contains both the standard XVM/DOS software, communicating with the DOS Monitor, and the batch oriented software, communicating with the BOSS Monitor. Files constructed by either operating system are compatible in both.

1.2.1 Disk System

The Disk Operating System is an interactive environment under console keyboard control. A DOS command "BOSS " is used to load the batch software.

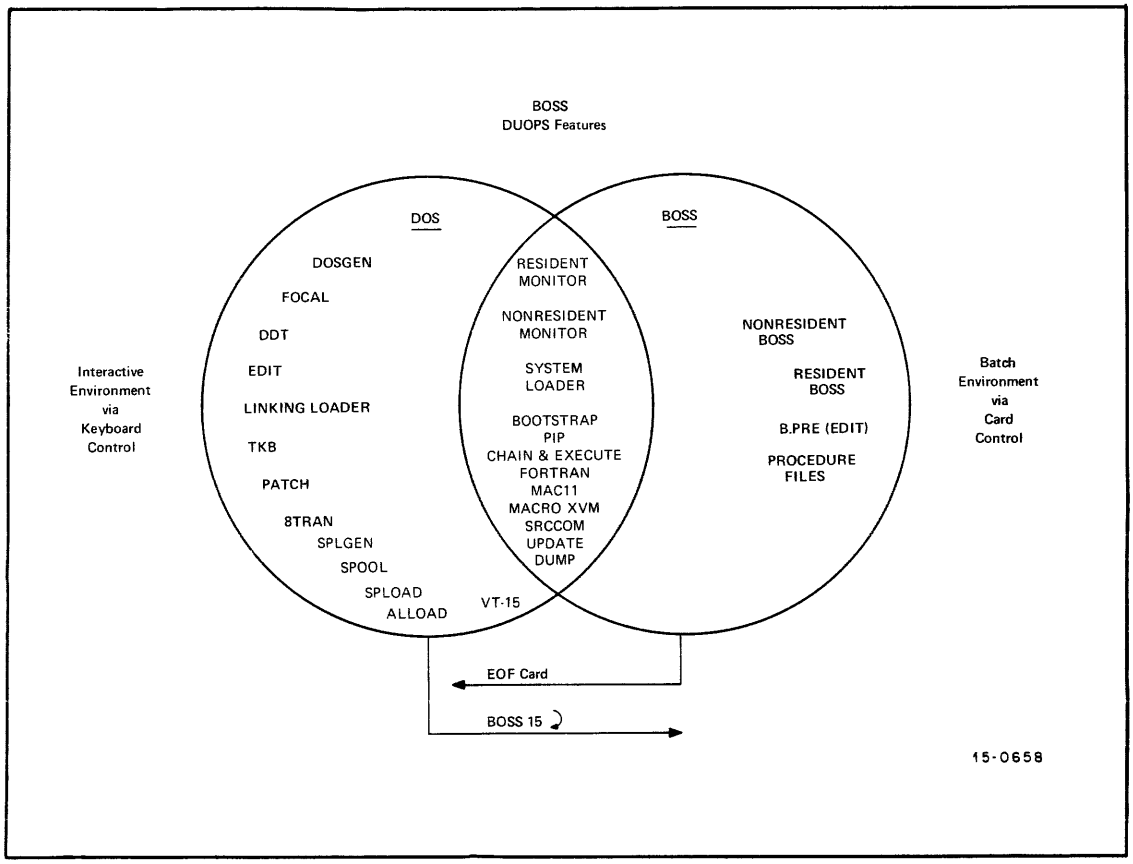


Figure 1-1 Dual Operating Environment

1.2.2 Batch System

Batch Operating System Software (BOSS) controls a batch environment exclusively through the use of BOSS command cards. An End-Of-File-Card is required at the end of a job deck to reenter DOS software. The program features of both environments are illustrated in Figure 1-1.

1.3 BOSS XVM

BOSS is a disk-resident batch operating system. It is a set of programs supervised by a Monitor. The Monitor tailors the system configuration according to the BOSS Command Language card, and then supervises operation of the system or user programs requested by control cards. The set of programs called BOSS XVM provides powerful I/O capabilities and programming tools that include:

Program Preparation
Compilation
Assembly
Debugging
Execution of User Programs and
Job Process Accounting

Chapter 4 presents the various programs available to the BOSS user. For detailed information about any of these programs, the user should refer to the appropriate manual.

1.3.1 System Features

BOSS offers all the I/O power available in DOS

- Substantial device independence
- I/O in a variety of data modes
- Powerful I/O macros that initialize the device, search for files, open and close files, and take care of all I/O transfers and device interrupts.
- A special file structure suitable for efficient use of disk storage space
- Dynamic disk storage allocation
- Dynamic buffer allocation
- A simplified file protection scheme for disk files

- Random access to physical disk records and, using FORTRAN routines, random access to logical disk records.
- The number of simultaneously open disk files limited only by available buffer space and the number of logical device slots assigned to the disk
- Spooled data to/from slow UNIBUS peripherals onto the RK05 system-resident disk to increase throughput.[1]

In addition, BOSS has added the following features:

- Job accounting facilities
- Line editing
- A procedure-file-driven command language which is:
  1. Open-ended
  2. User-modifiable
- The Batch Control Language resides on an 80 column card in a "free-flowing" format.

[1] UC15 systems only

CHAPTER 2
CARD CONVENTIONS

2.1 CARD PUNCHING

The basic operating unit within the BOSS batch job stream is the punched card. This card contains data to be interpreted as system control information, user source coding and run-time data. It is a standard eighty column card using the Hollerith Card Code format. Original data is transcribed (e.g., from accounting forms and program coding forms etc.) to punched cards from any of several types of "Key Punch" equipment.

2.1.1 Hollerith Card Format

The Hollerith card has 80 columns and 12 rows. The columns are numbered 1 through 80. Ten rows are identified by numbers ∅ through 9. The remaining two rows, "11" and "12", are not marked. Along with the "0" row, they make up the zone rows. Each column on the card can be used to record one numeric, alphabetic, or special character.

2.1.2 Hollerith Card Code

The Hollerith card is used to code alphanumeric characters into computer-usable data. An alphabetic character is represented by a numeric and a zone overpunch in one column. As shown in Figure 2-1, the twelve rows are used to represent 26 alphabetic characters, 10 decimal digits, and certain special symbols.

2.1.2.1 Digits - The digits are recorded by punching a single hole in the corresponding digit or zero position of the desired column.

2.1.2.2 Alphabetic Characters - A letter is a combination of one zone punch and one digit punch in the desired column. For example, A is 12-zone and digit-1 punches, N is 11-zone and digit-5 punches, and Z is 0-zone (zero) and digit-9 punches.

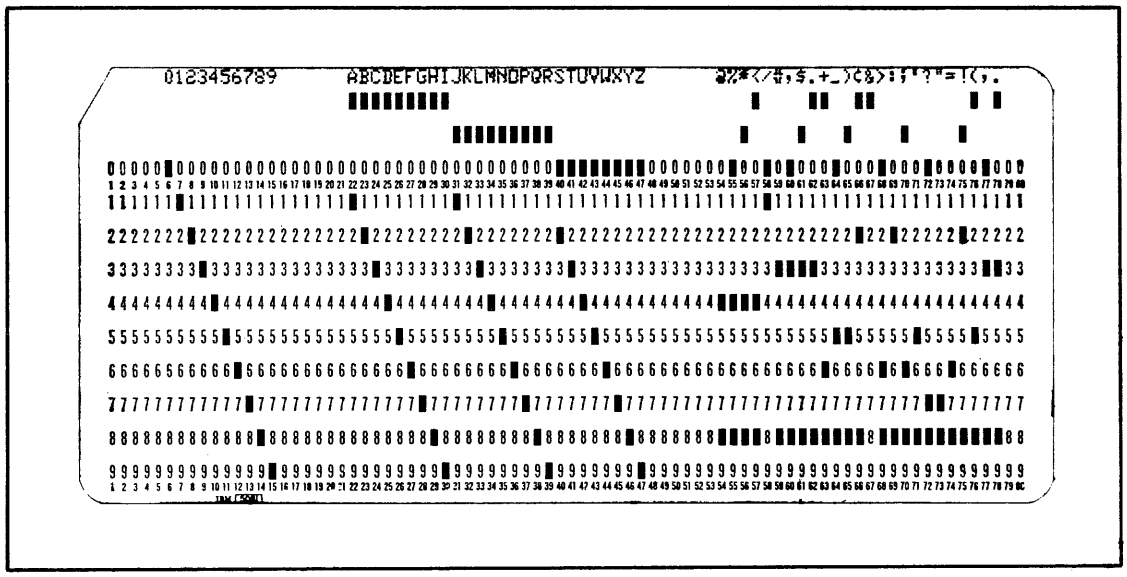


Figure 2-1
DEC Ø29 Card Code

2.1.2.3 Special Characters - A special character is one, two, or three holes in the desired column as shown in Figure 2-1. Punching two or three holes in one column for a letter or special character is generally automatic when the corresponding key is pressed. Numbers are represented in columns 6 through 15, letters in columns 22 through 46, and special characters in columns 54 through 78.

2.2 DIGITAL XVM ASCII/HOLLERITH CORRESPONDENCE

The BOSS Character Set represented in Table 2-1, shows the correspondence between the XVM 64 character graphic subset of ASCII and the DECØ29/Ø26 Hollerith codes. The Ø29 code, except as indicated by brackets [ ], is a subset of the standard Hollerith punched card code specified in ANSI standard X3.26-1970. Parentheses denote 1963 character set. The following special conditions should be noted.

1. ASCII codes ØØ-37 and 140-177 have no corresponding codes in the DECØ26 and Ø29 Hollerith sets, and, therefore, are not presented here.
2. ALT MODE is simulated by a 12-8-1 punch (multipunch A8).
3. End-of-File corresponds to a 12-11-Ø-1 punch (multipunch AØ-) in column 1.
4. Back-arrow (←) is simulated by: Ø-8-5 multipunch in Ø29 code; 8-2 multipunch in Ø26 code.

5. The card reader hardware supplies the binary equivalent of Hollerith code which, in turn is mapped into 7-bit ASCII by the Card Reader Handler.

6. For UC15 Systems, a special end of spooled deck card is necessary to force all the card images through the PIREX monitor buffers and onto the spooled area on the disk.

   The format of this card is an ALT Mode in column 1 and an ALT mode in column 2. (12-8-1, 12-8-1).

   One of these cards should follow in every deck (applies to only unichannel systems with card reader spooling enabled).

2.3 CARD COMMANDS

The card commands used to control one or more jobs in a user's input deck are shown in Figure 2-2. The reader should refer to Chapters 5 and 6 for information concerning the BOSS command language syntax and individual command structure.

TABLE 2-1

BOSS CHARACTER SET

ASCII		HOLLERITH		ASCII		HOLLERITH	
Char.	7-bit Code	DEC Ø29 Code	DEC Ø26 Code	Char.	7-bit Code	DEC Ø29 Code	DEC Ø26 Code
Space	40			@	100	8-4	8-4
!	41	[11-8-2]	12-8-7	A	101	12-1	12-1
"	42	8-7	0-8-5	B	102	12-2	12-2
#	43	8-3	0-8-6	C	103	12-3	12-3
$	44	11-8-3	11-8-3	D	104	12-4	12-4
%	45	0-8-4	0-8-7	E	105	12-5	12-5
&	46	12	11-8-7	F	106	12-6	12-6
'	47	8-5	8-6	G	107	12-7	12-7
(	50	12-8-5	0-8-4	H	110	12-8	12-8
)	51	11-8-5	12-8-4	I	111	12-9	12-9
*	52	11-8-4	11-8-4	J	112	11-1	11-1
+	53	12-8-6	12	K	113	11-2	11-2
'	54	0-8-3	0-8-3	L	114	11-3	11-3
-	55	11	11	M	115	11-4	11-4
.	56	12-8-3	12-8-3	N	116	11-5	11-5
/	57	0-1	0-1	O	117	11-6	11-6
0	60	0	0	P	120	11-7	11-7
1	61	1	1	Q	121	11-8	11-8
2	62	2	2	R	122	11-9	11-9
3	63	3	3	S	123	0-2	0-2
4	64	4	4	T	124	0-3	0-3
5	65	5	5	U	125	0-4	0-4
6	66	6	6	V	126	0-5	0-5
7	67	7	7	W	127	0-6	0-6
8	70	8	8	X	130	0-7	0-7
9	71	9	9	Y	131	0-8	0-8
:	72	8-2	11-8-2	Z	132	0-9	0-9
;	73	11-8-6	0-8-2	[	133	12-8-2	11-8-5
<	74	12-8-4	12-8-6	\	134	11-8-7	8-7
=	75	8-6	8-3	]	135	0-8-2	12-8-5
>	76	0-8-6	11-8-6	Λ(↑)	136	12-8-7	8-5
?	77	0-8-7	12-8-2	_(←)	137	0-8-5	8-2

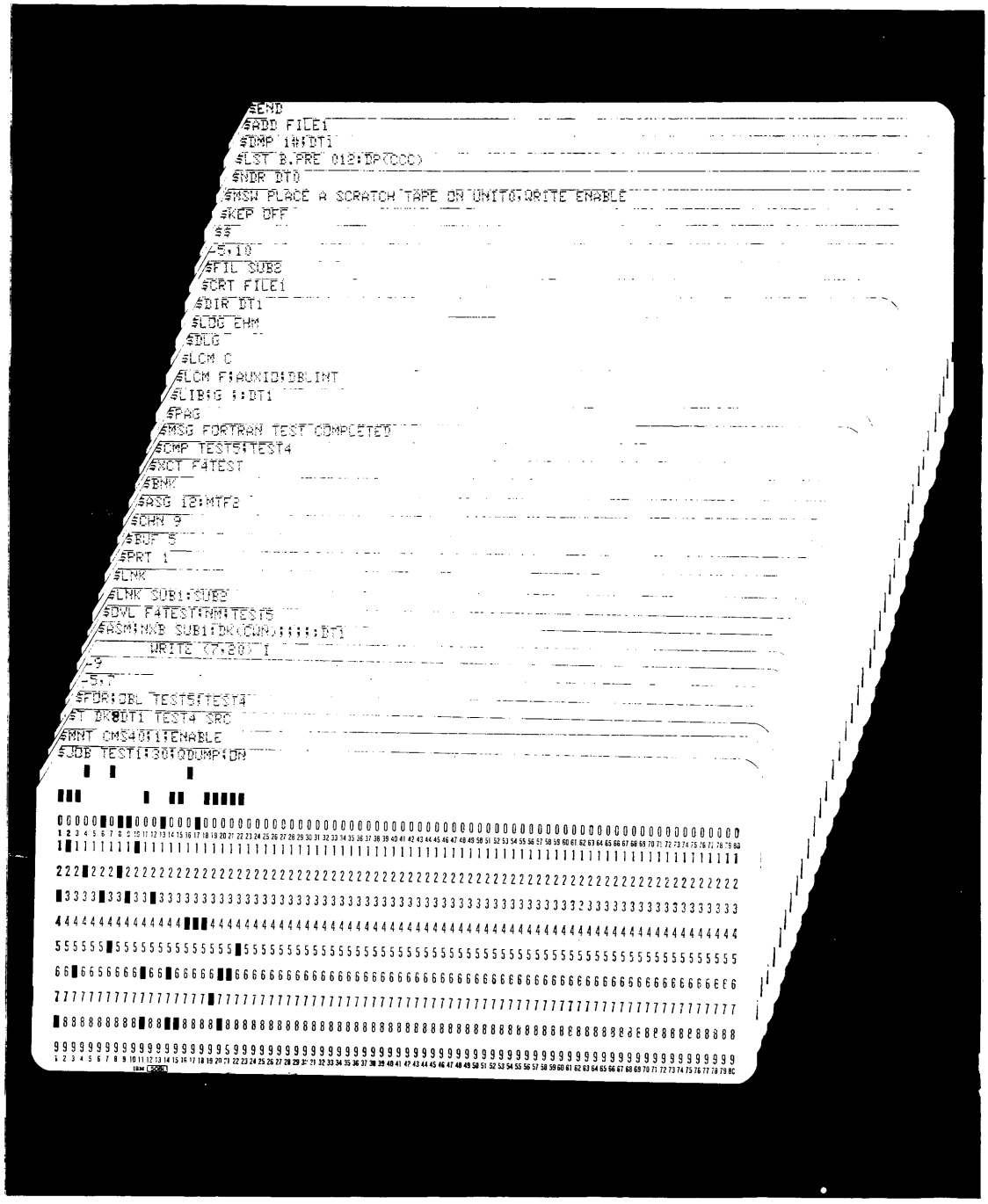


Figure 2-2 BOSS Card Commands

CHAPTER 3
SYSTEM HARDWARE ENVIRONMENT

3.1 INTRODUCTION

The Batch Operating System Software is defined within the limits of a specified DIGITAL XVM/PDP-15 hardware system configuration; i.e., central processor model, minimum and maximum core requirement, necessary features, and types and numbers of peripheral devices.

In order to run the BOSS XVM portion of the XVM/DOS system, the minimum hardware requirements include a card reader and a line printer.

3.2 MINIMUM HARDWARE REQUIREMENTS

For a DIGITAL XVM System: [9]

KP15 Central Processor
24,576 Words of 18 Bit Memory
LA36, LA30C, KSR33, KSR35 or VT05 Console Terminal[2]
PC15 High-Speed Paper Tape Reader/Punch
KE15 Extended Arithmetic Element
KW15 Real-Time Clock

XM-15 Memory System including: [9]

Wide Addressing
Memory Protection and Relocation
Instruction Prefetch
Automatic Priority Interrupt

TC15 DECtape Control with 1 TU56 Dual DECtape Transport[4]; or

TC59 Magtape Control with 1 TU10, TU20 or TU30 (7- or 9-track) Magtape Transport

RK15 Cartridge Disk System with RK05 Cartridge Disk Drive and 8,192 Words of 16-bit Core Memory[5]; or

RP15 Disk Pack Control with one RP02 or RP03[1] Disk Pack Drive; or

RF15 DECdisk Control with two RS09 DECdisk drives.

CR11 Card Reader or CR15 Card Reader or CR03B Card Reader.

LP11 Line Printer or LS11 Line Printer or LV11 Electro Static Printer; or

LP15 Line Printer

For a PDP-15 Computer System:

KP15 Central Processor

24,576 Words of 18-bit Core Memory

KSR35, KSR33, LA30C, LA36 or VT05 Console Terminal[2]

PC15 High-Speed Paper Tape Reader/Punch

KE15 Extended Arithmetic Element

KW15 Real-Time Clock

KA15 Automatic Priority Interrupt (required only for RK15 systems in certain configurations)[3]

TC15 DECtape control with one TU56 Dual-DECtape transport[4], or

TC59 Magtape Control with one TU10, TU20 or TU30 (7- or 9-track) Magtape Transport

RF15 DECdisk Control with two RS09 DECdisk drives; or

RP15 Disk Pack Control with one RP02/RP03[1] Disk Pack Drive; or

RK15 Cartridge Disk System with one RK05 Cartridge Disk Drive and 8,192 Words of 16-bit Core Memory[5]

CR15 Card Reader or CR11 Card Reader or CR03B Card Reader

DP15 Line Printer or LP11 Line Printer or LS11 Line Printer or

LV11 Electrostatic Printer

3.3 OPTIONAL HARDWARE SUPPORTED

For the DIGITAL XVM:[9]

CR15/CR11/CR03B Card Readers

Core Memory 16-bit (12K)[6]

Core Memory 18-bit (ME15, MF15) (32K to 128K)[8]

TC15/4 TU56 Dual DECtape or TC02/8 TU55 DECtape

RF15/8 RS09 Disk

RK15/8 RK05 Disk Cartridge[6]

RP15/8 RP02/RP03 Disk Packs[1]

FP15 Floating Point Processor

1 VT15/1 (VT04 or VT07)/1 VL04 Graphics Display

LP11/LP15/LS11/LV11[6] Line Printer

TC59/8 (TU10/TU20/TU30) Magtape

XY11/XY311[6] Plotter

System Hardware Environment

VP15A Storage Scope
LA36/LA30C/KSR33/ASR33/KSR35/ASR35/VT05/LK35[7,2] Terminal
1 VW01 Writing Tablet

For the PDP-15:

XM-15 Memory Processing Unit including:

- Wide Addressing
- Memory Protection and Relocation
- Instruction Prefetch
- Automatic Priority Interrupt

KA15 Automatic Priority Interrupt
CR03B/CR15/CR11[6] Card Reader
Core Memory 16-bit (12K)[6]
Core Memory 18-bit (MM15/MK15/ME15/MF15) (32K to 128K)[8]
TC15/4 TU56 or TC02/8 TU55 DECtape
RF15/8 RS09 Disk
RK15/8 RK05 Disk Cartridge[6]
RP15/8 (RP02/RP03) Disk Pack
FP15 Floating Point Processor
1 VT15/1 (VT04 or VT07)/1 VL04 Graphics Display
LP15/LP11/LS11/LV11[6] Line Printer
TC59/8 (TU10/TU20/TU30) Magtape
XY11/XY311[6] Plotter
VP15A Storage Scope
KSR33/KSR35/ASR33/ASR35/LA30C/LA36/VT05/LK35[7,2] Terminal
1 VW01 Writing Tablet

Notes:

1. The RP03 Disk Pack is supported as if it were an RP02 Disk Pack.
2. The LA36 and LA30C DECwriters and the VT05 Video Terminal are supported up to 300 baud.
3. API is required if the user has more than four PDP-11 options which need to interrupt the PDP-15.
4. The older style of DECtape can be used: TC02 Control with 2 TU55 single DECtape transports.
5. The RK15 includes within it an 8,196 word UNICHANNEL -15 peripheral processor. If the spooling software is to be used, the

requirement for 16-bit core memory is raised from 8,192 words to 12,288 words, if more than two devices are to be spooled.

6. Prerequisite hardware for devices like the CR11, LP11, LS11, LV11, XY11, and XY311, plus 16-bit core memory, is the RK15. The RK15 minimally contains a UNICHANNEL-15 with 8K words of 16-bit memory, an RK11E Cartridge Disk Control and 1 RK05 Cartridge Disk Drive. At any given time, only one line printer, one card reader, and one XY plotter in the combined XVM/PDP-11 or PDP-15/PDP-11 system is supported by the software. The LS11 or LV11 printer must be connected to the same vectors/priority as would the LP11.
7. The LK35 Graphic Keyboard is not a substitute for the console terminal; consequently, it also requires an LT15A Single Teletype Control. The paper tape facility on ASR Teletypes is not software supported. The LA30C and LA36 DECwriters and the VT05 Video Terminal can be run a baud rate of up to 300.
8. Memory configuration greater than 32K require the XM-15 options, and can utilize ME15 or MF15 memory only.
9. The XM-15 hardware option is incompatible with MM-15 memory, KA-15 automatic priority interrupt and KM/KT-15 memory protect and relocate. The XM-15 hardware functionally replaces these options.

CHAPTER 4
BOSS XVM SOFTWARE ENVIRONMENT

4.1 INTRODUCTION

This chapter gives information necessary for the user to evaluate system programs to be run under BOSS control. Each system program will assist the user in performing a particular task in the process of application design and implementation. Considerations for system modification can be found in the XVM/DOS System Manual.

4.2 SYSTEM SOFTWARE

The Batch Operating System's service routines perform four primary tasks for all user applications (see Figure 4-1).

1. Run-Time Aids - External routines from several libraries are available to the user. The libraries may contain either user-designed routines or those provided by DOS which can be implicitly or explicitly called.
2. General Utilities - BOSS provides facilities for efficient storage, flow, and retrieval of system and user data. There are also system features that provide file verification and data buffer allocation, file data manipulation, etc.
3. Program Preparation and Maintenance - There are system programs to aid user program preparation by modifying file text for source programs and programs to aid maintenance of object programs.
4. Language Assembly and Compilation - Programs are available to translate problem-oriented and procedure-oriented languages into machine language and to incorporate routines into complete, executable programs.

4.2.1 Choice of Languages

User source programs can be implemented at two levels depending on

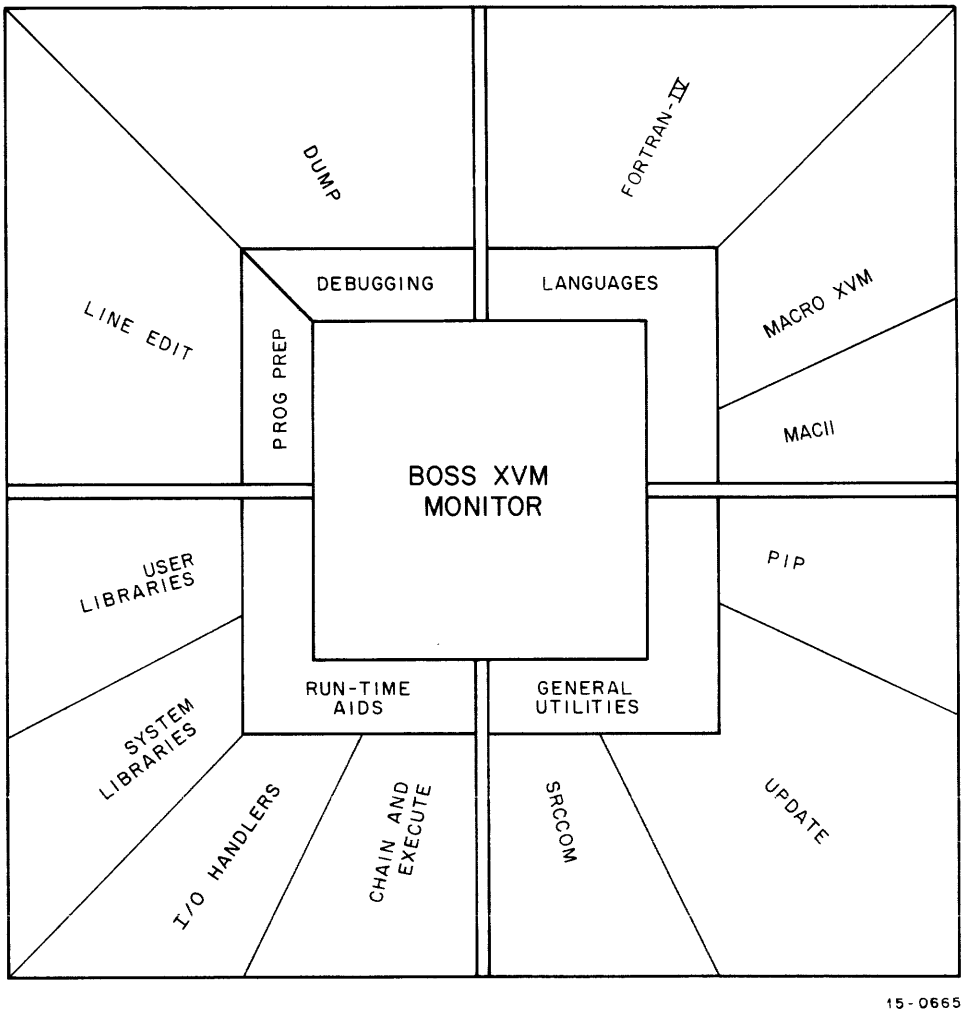


Figure 4-1 BOSS XVM Software System

those particular system features required for a given processing environment. BOSS supports the following levels of automation for object program preparation:

1. Compiler level language - FORTRAN IV (called via the $FOR card),
2. Assembly language - MACRO (called via the $ASM card).
3. PDP-11 Assembly language - MAC11 (called via the $MAC11 card).

4.2.1.1 FORTRAN IV Compiler - The XVM FORTRAN is a higher-level language system that accepts statements written in the FORTRAN IV language and produces a relocatable object program which can be loaded with either the Loader or with CHAIN/EXECUTE. The XVM FORTRAN IV is based on the language of USA Standard FORTRAN (X3.9-1966). The FORTRAN system consists of the FORTRAN IV compiler and an Object Time System (OTS).

Subroutines written in FORTRAN IV, or MACRO assembly language can be loaded with and called by FORTRAN IV main programs. Comprehensive source language diagnostics are produced during compilation. The system's Data-Directed Input-Output package permits input or output of ASCII data without reference to a FORMAT statement. The system can also perform memory-to-memory transfers (Encode/Decode), moving data from memory to the I/O Buffer, to memory.

BOSS supports two versions of the FORTRAN compiler. The non-floating point version, F4M, and its OTS, perform floating point operations by generating software calls to subroutines. The hardware floating point version, FPF4M, and its OTS support the optional FP-15 hardware by generating floating point instructions.

A FORTRAN IV program may be compiled and run in several different hardware environments. The FORTRAN programmer need not be too concerned with the details of his environment since the FORTRAN Object-Time System (OTS) will ensure that his source statements generate the appropriate computer instructions. For example, an arithmetic statement such as A=A*B will appear the same in any FORTRAN IV program. In the object program it may be transformed to a subroutine call or a floating point instruction, depending on the hardware configuration on which the program is produced. FORTRAN data-transmission statements automatically call a number of OTS subroutines which serve as an interface between the user program and the Monitor. These routines

may also be called from MACRO assembly language programs. Further, programs written in FORTRAN IV can be linked to programs or routines written in the MACRO assembly language.

For more information concerning this higher-level programming language, refer to the FORTRAN IV XVM Language Manual and the FORTRAN IV XVM Operating Environment Manual.

4.2.1.2 MACRO XVM Assembler - The MACRO XVM Assembler provides users with highly sophisticated macro generating and calling facilities within the context of a symbolic assembler. MACRO permits the programmer to use mnemonic symbols to represent instruction operation codes, locations, and numeric quantities. It simplifies the handling of recurring instruction sequences through its macro instruction feature. This feature permits the user to represent these instruction sequences with a single source statement. Some of the prominent features of MACRO include:

1. The ability to:
   (a) define macros
   (b) define macros within macros (nesting)
   (c) redefine macros (in or out of macro definitions)
   (d) call macros within macro definitions
   (e) have macros call themselves (recursion)
   (f) combine up to three input files for one assembly

2. Conditional assembly based on the computational results of symbols or expressions.

3. The ability to repeat a generated word of output.

4. Boolean expressions.

5. Optional octal, symbolic, and cross-reference listings.

6. Two forms of radix control (octal, decimal), and two text modes (ASCII and 6-bit trimmed ASCII).

7. Global symbols for easy linking of separately assembled programs.

8. Choice of output format: relocatable, absolute binary (checksummed), or full binary -- capable of being loaded via the hardware READIN switch.

9. Ability to utilize user-designed input/output macros.

10. A Table of Contents option containing the page numbers and text of all assembled .TITLE statements in the program.

11. The ability to combine up to three separate input files for one assembly.

12. The ability to use an external file of system macros.

Refer to the MACRO XVM Assembler Language Manual for a complete description of the language.

The standard object code produced by MACRO XVM is in a relocatable format which is acceptable to the CHAIN utility program. Relocatable programs that are assembled separately and use identical global symbols[1] where applicable, can also be combined by CHAIN into an executable program.

An output listing, showing both the programmer's source coding and the binary object program produced by MACRO is printed if desired. This listing includes all the symbols used by the programmer with their assigned values. If assembly errors are detected, erroneous lines are marked with specific letter error codes.

4.2.1.3 MAC11 Assembler[2] - This MACRO Assembler (MAC11) provides the user of the UNICHANNEL-15 system with the capability of assembling the full repertoire of the PDP-11/20 instruction set. Besides making the UC15 system self-sufficient it provides the users with highly sophisticated macro operating and calling facilities within the context of a symbolic assembler. Some notable features of MAC11 are:

1. The ability to:

   (a) define macros,

---

[1] Symbols which are referenced in one program and defined in another.

[2] Only for UC15 systems.

(b) Define macros within macros (nesting),
(c) Redefine macros (in or out of macro definition)
(d) call macros within macro definition
(e) provide alternate exit points from macros (particularly nested macros)
(f) pass arguments (numeric and non-numeric) and compute the number of arguments passed
(g) provide built in error reporting capability in a macro

2. Create automatic local labels.

3. Concatenate strings.

4. Generate indefinite and definite repeat blocks.

5. Conditional assembly based on the computational results of symbols or expressions.

6. Two forms of radix control (octal, decimal) and three text modes (ASCII, ASCIZ and 6-bit ASCII).

7. Ability to utilize user designed input/output macros.

8. Use local and non-local labels.

9. Provide table of contents containing the page numbers, text of all assembled .TITLE statements in the program and the line numbers on an output listing.

MAC11 permits the programmer to use mnemonic symbols to represent instruction operation codes, locations, and numeric quantities. It is essentially a comprehensive macro instruction generator. This generator permits easy handling of recursive instruction sequences, changing only the arguments.

The assembler facilitates the development of instructions called "macros" which, when used as a source statement, can cause a specific sequence of instructions to be generated in the object program. Refer to the MAC11-XVM Assembler Language Manual for a complete description of the language.

The only object code produced by MAC11 is in an absolute format on paper tape.

An output listing, showing both the programmer's source coding and the binary object program produced by MAC11, is printed if desired. This listing includes all the symbols used by the programmer with their assigned values. If assembly errors are detected, erroneous lines are marked with specific letter error codes.

4.2.2 XVM/DOS System Generator (SGEN XVM)

SGEN XVM (SGEN) is a DOS Utility program used to modify disk resident system files. It is not part of the BOSS run-time environment. It is available only in the DOS interactive mode. SGEN is provided as part of the general-purpose package. It enables the user to tailor his system and add to the supplied software in order to develop a resident software system unique to the installation or to his specific needs.

The System Manager calls the system generator program via the Monitor command "SGEN". When SGEN is loaded, it initiates an interactive question/answer sequence regarding the following system functions and parameters.

1. Options
2. Type of printer unit used
3. Required device handler designations (i.e., the standard I/O configuration the user wants for the system programs
4. Skip-Chain information - Priority Interrupt Skip Chain contents and order
5. Default assumptions
6. System device designation
7. .DAT slot assignments
8. Monitor Identification Code to designate the privileged access to the System Manager

9. Default buffers that are needed at any time during a user program,

10. Default Files Protection Code.

Careful planning is necessary to ensure that the most efficient system will be developed for the user's particular needs. For more information, refer to the SGEN XVM Utility Manual.

4.2.3 CHAIN XVM and EXECUTE XVM Programs

These programs permit the BOSS user to load and execute his FORTRAN IV and MACRO object programs. CHAIN XVM (CHAIN) relocates segments and constructs a system of core overlays in an easy and straightforward manner.

CHAIN also reserves portions of user core (called COMMON blocks) from one segment to another so that the program segments can communicate. The FORTRAN IV compiler and the MACRO assembler can reserve COMMON BLOCKS for future segmentation. This method of segmentation permits multiple overlays of executable code, constants, variables, arrays, and labeled COMMON blocks.

In addition, CHAIN organizes subroutines into units called LINKS, which may overlay each other. Several LINKS may overlay a larger LINK without overlaying each other. A LINK is loaded into core when a subroutine within the LINK is called via the $LNK card, and remains resident until overlayed. A LINK's core image is not recorded or "swapped out" when it is overlayed. The same image is brought into core each time a LINK is loaded. For maximum run-time efficiency, segments must be processed serially. CHAIN is called by the $MAP or $OVL command.

EXECUTE XVM (EXECUTE) is a control program which initiates loading of the absolute (XCT/XCU) files created by CHAIN and supervises the transfer of control from one segment chain to another. The $XCT command is used to execute the executable file previously created by use of the $MAP or $OVL commands.

The $XCT card initiates EXECUTE operation. EXECUTE is a loader-monitor that runs the program files created by CHAIN. See the CHAIN XVM/EXECUTE XVM Utility Manual for detailed instructions.

4.2.4 DUMP XVM Program

DUMP XVM (DUMP) gives the user the ability to output, on any device specified, core locations that have been preserved on disk via the $DMP and $QDP commands. For more information refer to the $DMP Section in Chapter 6.

4.2.5 Peripheral Interchange Program (PIP XVM)

PIP XVM (PIP) can transfer data files from any input device to any output device. It can also be used to:

1. Refresh file directories,
2. List file directory contents,
3. Delete, insert, segment, or combine files,
4. Perform code conversions,
5. Assign protection codes,
6. Transfer files, or
7. Copy the entire contents of disk and tape storage units.

It may also be used to update and allocate restricted disk storage surfaces. Refer to the PIP XVM Utility Manual for additional information.

4.2.6 BOSS Preprocessor (B.PRE)

The system program B.PRE is a "Line Editor" used to preprocess MACRO and FORTRAN source files. It can also be used for any disk resident ASCII file. The Line Editor permits the user to insert, delete, and substitute file lines. For more information, refer to the Line Editor Section in Chapter 5.

4.2.7 UPDATE XVM - Library Update Program

This system program gives the user the capability to examine, extract, and update the binary library files on mass storage devices. The

UPDATE XVM program is called via the $LIB card and must be followed by the $LCM command functions. For more information, refer to the UPDATE XVM Utility Manual.

4.2.8 SRCCOM XVM - Source Compare Program

The SRCCOM XVM program compares any two symbolic source programs (ASCII) and indicates their differences. It is called via the $CMP card. This program is useful for program identification and/or verification, proofing an edited program, comparison of old and new versions of the same program, etc. For more information, refer to the SRCCOM XVM Utility Manual.

4.3 BOSS XVM CONCEPTS AND FACILITIES

BOSS XVM (BOSS) is a general purpose batch processing system. It provides the user with the ability to use system programs and the disk file structure of DOS. The exceptions are those programs that work only in an interactive environment. The user makes use of the system facilities via a batch, card-oriented control language. BOSS permits a hands-off environment.

The typical installation that runs BOSS has a computer operator. The operator receives a job from a user in the form of a card deck. He runs the job, and returns to the user the input card deck and the line printer output with information pertaining to that run. The operator also has access to a tape bank. If a job requires the use of tape (either DECtape or magtape) the user can instruct the operator to mount the desired tape(s) through a special "message" control ($MNT) card which is part of the BOSS command language.

The general purpose facilities of BOSS include:

1. A comprehensive disk file structure with random access and protection.

2. A line editor for card oriented on-line file updating.

3. A fast overlay scheme.

4. User created library files.

5. An accounting capability.

6. A job timer.

The user can operate with only five control cards:

$JOB
$FOR
$MAP
$XCT
$END or EOF

added to his FORTRAN source deck and data cards. He can compile and execute one or more FORTRAN programs (one main program plus one or more subprograms). For additional power, the user has at his disposal many variations on each control card which allow him to make use of any DOS facility.

Another feature of BOSS gives the system programmer in charge of a BOSS installation the ability to modify the command language. He can accomplish this without having to interrupt operations. BOSS achieves this through its "procedure driven command language". Each BOSS command has an associated ASCII file which describes the action of the command in terms of DOS commands. This file is called the Procedure File. Procedure Files permit rapid modification of the BOSS command language to suit installation needs using simple editing procedures. The command name and the name of its Procedure File have to coincide to create the association between them.

4.3.1 BOSS File Structure

BOSS disk file structure and file operations are identical to DOS I/O and file operations. The Monitor makes the association between logical and physical devices via the Device Assignment Table[1]. At run time, the user assigns handlers those .DAT slots via the $ASG command, and the Monitor loads the handlers along with the user's program. Chap-

[1]The size of the .DAT is determined at system generation time, and may have as many as 76_{10} -- 15_8 negative and 77_8 positive -- entries.

ters 4 and 6 of the XVM/DOS User's Manual describe the characteristics of Monitor I/O operations for devices other than the disk.

Additional BOSS facilities are described in the following paragraphs.

4.3.2 Disk File Structure

The following is a description of the DOS file structure used with BOSS and the changes this structure makes to operations and procedures. Ordinarily, each disk user has a directory which points to named files. The disk has as many directories as users have cared to establish. The number of files associated with any one directory on a disk is limited only by the available space.

Since the disk may have a variable number of directories, the Monitor must know how to find each user's directory. It therefore maintains a Master File Directory (MFD) at a known location[1], and the Master File Directory points to each User File Directory (UFD). BOSS allows only those users who know a special code, called the Monitor Identification Code (MIC) and use the $MIC command, to have access to the MFD. Figure 4-2, Master File Directory, illustrates the organization of the MFD.

4.3.2.1 User Identification Codes (UIC) - The Monitor finds User File Directories by seeking associated User Identification Codes (UICs), which are all listed in the Master File Directory. The UIC is a three-character code that is necessary for all directory oriented I/O to Directory. (Non-directory oriented I/O to the disk, i.e., TRAN macros, use no directory references.) A programmer may operate under as many UICs as he wishes, provided all are unique and none is reserved[2] He may establish a new User File Directory by (1) logging in his new UIC to the Monitor via the $LOG command and (2) by issuing a $N DK command. This establishes a new User File Directory, or refreshes (wipes clean) an old directory under that UIC. (During user program operation, .ENTER will also create a new MFD entry and/or a UFD, if none exists.) Figure 4-3, User File Directory, illustrates the organization of a UFD.

---

[1] On the RF and RK disks, the first block of the MFD is located at block 1777.
On the RP disk, the first block of the MFD is located at block 47Ø4Ø.

[2] The following are reserved UIC's: @@@, ???, PAG, BNK, SYS, IOS, CTP, PER.

Word #	Contents	Description
Ø	777777	Dummy
1	nnnnnn	Bad Allocation Table's first block number, or 777777, if there is none
2	nnnnnn	SYSBLK's first block number, or -1, if there is none
3	$4_{Ø-2}$+SUBMAP	MFD entry size in bits Ø-2, plus the block number of the first submap
.	.	.
.	.	.
.	.	.
4N	.SIXBT	UIC for this UFD
4N+1	nnnnnn	Block number for the first block of this UFD or 777777, if no UFD exists (as after PIP's N DK)
4N+2	$P_{Ø}$+M	Protection code in bit Ø, plus the UFD entry size for each file
4N+3	spare	Unused at this writing
.	.	.
.	.	.
.	.	.
374[1]	nnnnnn	Spooler disk area size
375[1]	nnnnnn	Spooler disk area starting block
376	nnnnnn	Pointer to previous MFD block, or 777777 if none.
377	nnnnnn	Pointer to next MFD block, or 777777 if none.

Figure 4-2
Master File Directory

Word #	Contents	Description
.	.	.
.	.	.
.	.	.
8N	.SIXBT	Name of this file
8N+1	.SIXBT	and its
8N+2	.SIXBT	Extension
8N+3	$T_{Ø}$+blknum	Truncation code in bit Ø, plus the number of the first block of the file
8N+4	nnnnnn	Number of blocks in this file
8N+5	ribptr	Pointer to the first block of the Retrieval Information Block
8N+6	$P_{Ø-1}$+ribwrd	Protection code in bits Ø-1, plus the first word in ribptr used by the RIB--if the last block of the file has room for the RIB, the handlers will put it there, and load word 8N+6 accordingly.
8N+7	crdate	Date of file's creation -mmddyy (yy modulo 7Ø)
.	.	.
.	.	.
.	.	.
376	nnnnnn	Pointer to previous block, or 777777 if none
377	nnnnnn	Pointer to next UFD block, or 777777 if none

Figure 4-3
User File Directory

[1]Bits Ø-1 of word 374 are concatenated with bits Ø-1 of word 375 to produce a 4-bit checksum for the 16-bit fields of words 374 and 375. Checksum = word 374 (2-17) + word 375 (2-17) + 1.

4.3.2.2 The User File Directory Table (.UFDT) - The Monitor must have a way of knowing which User File Directory to reference when a program issues I/O commands to the disk. It makes the association between disk I/O commands and User File Directories by using a User File Directory Table (.UFDT). There are as many entries in the User File Directory Table as there are slots in the Device Assignment Table. Figure 4-4, Relationship Between the .DAT and the .UFDT, shows how the two compare. Disk I/O to a particular .DAT slot will affect files in the User File Directory named in the corresponding .UFDT slot. Unless modified during System Generation, the default value in each .UFDT slot is SCR, for Scratch (except in those slots reserved for the system, -2, -3, +5, +6 and -7). Whenever a user issues a LOGIN command to the Monitor, the Monitor automatically assigns the three-character code punched after $LOG to each .UFDT slot with the SCR default.

The user may modify the .UFDT via the $ASG command card. This allows program I/O to operate on files listed under a UFD that is not associated with the UIC current[1] to the system. If the operator has not changed the .UFDT via the $ASG command, the program may issue a .USER command to the appropriate .UFDT slot. The system will then enter the desired UIC into the proper .UFDT slot. The program can then operate on files listed by that User File Directory, according to the protection codes specified.

4.3.2.3 File Protection - BOSS offers a simplified form of file protection. Each User File Directory has a protection code (optionally specified in commands to PIP), and each file has a protection code (optionally specified in the .ENTER command, via the $PRT command, or a command to PIP). The protection codes are in effect only when a program tries to reference a file listed under a UIC other than the one currently logged in to the system. If a User File Directory is protected, then the protection codes for each file are in effect. If the User File Directory has been specified as unprotected, then no protection is provided for any file in the directory. There are three protection states possible for files in a protected User File Directory:

Protection code = 1	Unprotected, with the exception that the file may not be deleted and the number of blocks may not change, if the directory is protected.

[1] The "current" UIC is the one logged in to the Monitor.

Protection code = 2 Write protected, if directory protected.

Protection code = 3 Read/Write protected, if directory protected.

User File Directories may have one of two protection states:

Protection code = Ø Unprotected

Protection code = 1 Protected

The default protection code for User File Directories is always 1. The default protection code for files is established at system generation time. Users may change the file default protection code via the $PRT command to the Monitor. Whatever code is specified in a $PRT command will remain the default code until a $DLG (logout) or a new $PRT command is received. (See $PRT command in Chapter 6.)

The $N␣DK command to PIP allows the user to set the protection of the User File Directory, and the $R␣DK command to PIP allows the user to change that protection. The .ENTER I/O macro establishes a file's protection code. The only way to dynamically change a file's protection code is by recopying the file. Note that the disk handlers do not protect the user from himself.

File protection codes may be changed by using the RENAME COMMAND:

R DK FILE SRC←DK FILE SRC(#)

Where # is one of three possible protection codes for files in a protected UFD (see above).

4.4 ORGANIZATION OF SPECIFIC FILES ON DISK

Disk file blocks have forward and backward links, and upon receipt of a .CLOSE I/O macro, the disk handlers fill out a Retrieval Information Block (RIB). The RIB indicates which blocks the file occupies, and also associates the logical sequence of blocks in the file with the physical locations of the blocks on the disk. The disk handler uses the RIB to implement .RTRAN commands and to delete files. Figure 4-5, the Retrieval Information Block, illustrates a RIB.

After a user has created a disk file, he can access logical records sequentially via .READ commands, and physical records (blocks) of that file by referencing relative block numbers in the .RTRAN command.

DAT/UFDT Number	DAT Contents	UFDT Contents	Comment
+N	nondisk handler	UIC_1	This UIC is irrelevant
.	.	.	
.	.	.	
.	.	.	
+2	DKA	UIC_1	I/O to this slot will go to files in UIC_1
+1	DPA	UIC_2	I/O to this slot will go to files in UIC_2
.	.	.	
.	.	.	
.	.	.	
-15	NON	UIC_1	This UIC is irrelevant

The user has logged in under UIC_1. He has assigned some nondisk handler to DAT+N, the DECdisk to DAT+2, and the Disk Pack unit Ø to DAT+1, and NON (no handler) to DAT -15. In addition, he has changed the UFDT assignment for UFDT+1 to UIC_2. I/O to DAT+N will not reference the UIC, since UICs are relevant to disk I/O. I/O to DAT+2 will reference UIC_1, while I/O to DAT+1 will reference UIC_2. The UIC for DAT -15 is irrelevant, since no handler is assigned.

Figure 4-4 Relationship Between the DAT and the UFDT

WORD #	Contents	Description
Ø	nnnnnn	Total number of blocks described by this RIB. (The RIB need not start in word Ø of its block. See Figure 4-3, User File Directory.)
1	nnnnnn	First data block pointer
2		Second data block pointer
3		Third data block pointer
.	.	.
.	.	.
.	.	.
376	nnnnnn	Pointer to previous RIB block or -1 if no previous RIB block.
377	nnnnnn	Pointer to next RIB block or -1 if no next RIB block.

Figure 4-5 Retrieval Information Block

The .RTRAN commands require the file be opened with the .RAND command. The .RTRAN command allows the user to read or write any block in a file on a disk pack or disk cartridge and any word in a file on the DECdisk. The user is prohibited, however, from changing the total number of blocks in the file. Thus, insertion and deletion of blocks is only accomplished by recopying the file.

The FORTRAN IV OTS file routines allow random access at the logical record level (as opposed to the physical block level) for any file with fixed-length records. IOPS binary records may be as long as desired, when using these routines. For more information, see FORTRAN IV XVM Operating Environment Manual.

4.5 THE DISK HANDLERS

The disk handlers allow as many concurrently open input or output files as there are .DAT slots available to the user, and buffers available to the disk handler. The disk handlers operate under a dynamic buffer allocation scheme. Whenever the Monitor loads a system or user program, it allocates buffer space. This space is called the buffer pool. Whenever a program opens a disk file, the handler obtains a buffer from the buffer pool. It will return the buffer when the program closes the file.

The BOSS Monitor determines the amount of buffer space from two .SCOM registers and adds one for the RTF file. One register, .SCOM+27 contains the standard buffer size, as determined at system generation time. The other, .SCOM+26, contains the number of buffers to be allocated. The user can set the number of buffers needed for the next program by using the $BUF command. When the Non-resident Monitor returns, it will restore the number of buffers in .SCOM+26 to a value set at system generation time unless the user issued a $KEP command.

The buffers in the buffer pool are available to programs, as well as to disk handlers. A program may use .GTBUF and .GVBUF to obtain and return a buffer from the pool. The $BUF command card is used to temporarily change buffer default values for disk I/O and for the .GVBUF and .GTBUF requests. Whenever a program is using a buffer, however, it is unavailable to any other program.

Three handlers each are provided for operations with the RF15 DECdisk, and RP02 Disk Pack and the RK05 Disk Cartridge. Version for version,

these handlers are identical in their functions with these two exceptions:

a. While all three types of disks are block addressable for direct access operations (.TRAN and .RTRAN macros), the DECdisk in addition provides word addressability via .RTRAN macros.

b. The Disk Pack and Disk Cartridge have a unit structure while the DECdisk does not. This means that each DECdisk is treated as a single addressable unit regardless of the actual number of platters incorporated (up to 8), but separate Disk Pack and Disk Cartridge units are only separately addressable.

The handlers operate in all data modes (i.e., IOPS, Image, and Dump). A list of the I/O Macros which are acceptable to the various handler versions can be found in the XVM/DOS Users Manual.

4.6 XVM/DOS I/O MACROS

Table 4-1, Synopsis of XVM/DOS I/O Macros, is a summary of the I/O macros which the DOS Monitor recognizes. The disk random access commands are .RAND and .RTRAN. All commands are described in the XVM/DOS Users Manual.

4.7 ERROR PROCESSING

The Nonresident Monitor's INSTRUCT ERRORS printout (via $DOS I ERRORS) indicates the information given with each error message. In BOSS, all errors are terminal, except IOPS 4. This means that the operator may type CTRL R to recover from IOPS 4 error only. Error messages are output to the Line Printer and the Job Accounting File. On terminal errors the current job is aborted and the System searches for the next job ($JOB). Provisions to dump core via the $DMP, $JOB, or $QDP commands may be included in the job string.

4.8 BOSS XVM ACCOUNTING

BOSS has a very simple accounting mechanism. BOSS keeps an account record for each job in a random access file in the CTP User File Directory. CTP is protected, and can only be accessed after successful execution of a $MIC command.

TABLE 4-1

SYNOPSIS OF XVM/DOS I/O MACROS

Macro	Function
.CLEAR[1]	Initializes a directory on a directoried mass storage device[2]. All data on the device is lost and fresh bit maps and directories are written.
.CLOSE[1]	Terminates use of a file. In the case of output files on the disk, fills out the Retrieval Information Block (RIB) for later .RTRAN commands.
.DLETE[1]	Deletes a file from a directory on a mass storage device.
.ENTER	Primes a directoried mass storage device to accept an output file.
.FSTAT	Checks directory on a mass storage device for reference to the named file.
.INIT[1]	Initializes the device and device handler.
.MTAPE	Provides special commands for industry compatible magnetic tape.
.RAND	Opens a disk for random processing via .RTRAN macros.
.READ	Transfers a logical record, or the requested number of words, whichever is smaller, from the device to the user's I/O buffer.
.RENAM[1]	Renames a file in a directory of a mass storage device.
.RTRAN	Allows input and output to access to any block in a pre-existent file on the disk (any word if RF).
.SEEK	Checks for a named file in a directory on a mass storage device. If file is present, prepares it as an input file.
.TRAN	Gives independence from DOS-15 file structure by allowing input and output access to magtape, or to any block on disk or DECtape located by its physical block number.
.WAIT	Waits for I/O already started to complete, then continues.
.WAITR	Waits for I/O already started to complete, then continues. If I/O not done, passes control to specified address.
.WRITE	Transfers data from user's I/O buffer to device.

[1]At completion of these operations, the buffer is given back to the buffer pool, if disk I/O.
[2].CLEAR initializes the MFD directory when directed to a disk storage device.

The name of the accounting file is ACCNTG nnn; the first file has an extension of ØØ1. Each file is ten physical blocks long and contains enough information for 310 jobs, thirty-one per physical block. See Table 4-2. When the file ACCNTG ØØ1 is filled up, its extension is changed to ØØ2 and a new file ACCNTG ØØ1 is started. A third file makes the second become ACCNTG ØØ3 and so on. Therefore, the current accounting file always has the extension ØØ1, and the oldest, ØØ2. Every time a job ends, BOSS checks the status of the ACCNTG ØØ1 file. If it exists, BOSS checks to see if it is full. If not full, a new entry is made in ACCNTG ØØ1. If all extension numbers have been used (up to 777), BOSS prints this message to the operator on the teleprinter:

```
MAX NUMBER OF ACCOUNTING FILES REACHED
PLEASE PROCESS AND DELETE THEM
```

Every time the system manager processes an accounting file he should delete the file. The BOSS system only creates accounting files. It does not provide a program for processing them.

TABLE 4-2

ACCOUNTING FILE RECORD STORAGE

```
Blocks/File
Records/Block
Record = 8 words
       Where:  1 - 3 = ID code
                   4 = DATE mm dd yy
                   5 = START TIME hh mm ss
                   6 = FINISH TIME hh mm ss
                   7 = RUN TIME hh mm ss
                   8 = JOB TERMINATION STATUS
```

4.8.1 Accounting File Record

For each completed job, BOSS writes out an 8-word record to the accounting file. The records have the following format:

WORD #	CONTENT
1	Job I.D.
2	in
3	.SIXBT
4	Date, packed mmddyy
5	Start Time, in hhmmss
6	End Time, in hhmmss
7	Run Time, in hhmmss
8	Terminal Job Status Word

A word whose contents equal 777777_8 immediately follows the last job accounting record in each physical block of the accounting file.

CHAPTER 5
LANGUAGE SYNTAX

5.1 INTRODUCTION

The batch language, as implemented by BOSS XVM, provides user control over a group of programs, routines, and data placed together for the purpose of processing as a single operating unit. Control of this operation is accomplished through parameters carried on special system command cards. The rules governing each parameter within a particular command-string structure are called the "language syntax". This section will outline the language, define the syntax, and specify those command functions used to edit and manipulate data.

5.2 LANGUAGE COMMANDS

BOSS XVM has a flexible command language with all the capabilities of the XVM/DOS System. Each language command, with its functional description, is given in Table 5-1. A detailed description of each command is given in Chapter 8.

The BOSS XVM Command Language (BCL) has generally three types of commands as follows:

1. Special function commands,
2. Commands that call system programs,
3. Peripheral Interchange Program (PIP) commands.

Two of these command functions are illustrated in a simple BOSS XVM job string example shown in Figure 5-1.

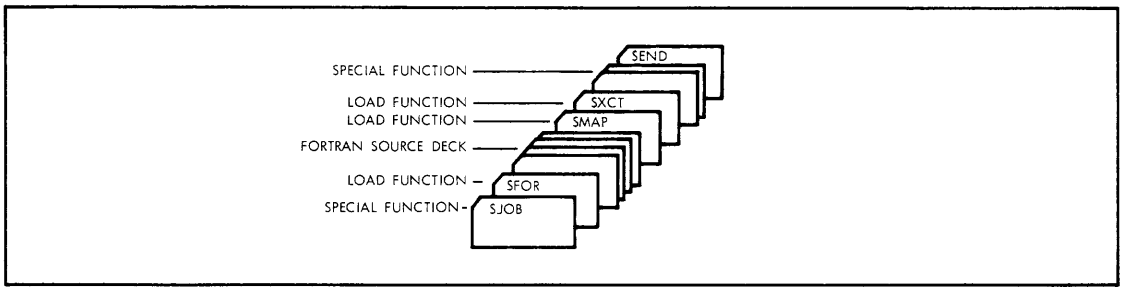


Figure 5-1
A BOSS XVM Job String Example

Table 5-1

BOSS Command Language

COMMANDS	FUNCTIONS
$	End-of-File Card for both Control and Data Files.
$$	End-of-ADDfile Card terminating a CRT file
$*	Continuation Card (Field continuation & Field Set addition).
-	Editor Command Card for Insertion, Deletion & Substitution.
$ADD	Add contents of file to input stream.
$ASG	Device correspondence, assign Device UIC to DAT.
$ASM	MACRO Assembler.
$B	Block copy (System Manager only).
$BNK	Enter BANK mode operation.
$BUF	Buffer Specification.
$C	Copy Mass Storage.
$CHN	Specify 7- or 9-Track magtape.
$CMP	SRCCOM, Source Compare Utility.
$CRT	Create ADD File.
$D	Delete file from user File Directory (UFD).
$DIR	List directory.
$DLG	LOGout UIC.
$DMP	Dump core upon terminal error.
$DOS	General Command string.
$END	End job.
$FIL	Create a file from cards.
$FOR	FORTRAN IV.
$I	Initialize, restore MFD.
$JOB	Begin a job.
$KEP	Retain device assignment.
$L	Initialize, restores MFD (System Manager only).
$LCM	Update Function.
$LIB	Update Command (Library Utility Program).
$LNK	Chain Link specification.
$LOG	LOGIN UIC.
$LST	List contents of file on line printer.
$MAC11	MAC11 assembler
$MAP	Chain Absolute File with/without overlays.
$MIC	LOGIN MIC (Not echoed).
$MNT	Tape mount message.
$MSG	Message to the operator.
$MSW	Message to the operator with Wait.
$N	New directory (UFD).
$NDR	Create new directory.
$OVL	Chain with overlays.
$PAG	Enter PAGE mode operation.
$PRT	Change file protection code.
$QDP	Dump Core on IOPS Error
$R	Rename file and/or change protection codes.
$RUN	FORTRAN compile, chain, and execute.
$S	Segment file into various members.
$T	Transfer files between I/O devices.
$U	Update bad allocation (BAT) and storage (SAT) tables.
$V	Verify files for Parity or Checksum.
$XCT	Execute.
$XVM	Enable or disable XVM mode for user programs.
(ALT) (ALT)	End of deck card for spooled operations.[1,2]
12-11-Ø-1	Terminate BOSS run and return to XVM/DOS card.[3]

[1] For systems with RKØ5 storage devices only.

[2] ALT is a 12-8-1 multipunch.

[3] 12-11-Ø-1 is a multipunch in column 1 of the card.

5.2.1 Special Function Commands

The special function commands provide those control parameters to be used in directing the operation of System software. These commands are as follows:

$ADD	$DOS	$MNT
$ASG	$END	$MSG
$BNK	$FIL	$MSW
$BUF	$JOB	$NDR
$CHN	$KEP	$PAG
$CRT	$LOG	$PRT
$DIR	$LST	$QDP
$DLG	$MIC	$XVM

5.2.2 System Program Load Commands

Commands that load System programs are as follows:

$ASM	$LCM	$MAP
$CMP	$LIB	$OVL
$DMP	$LNK	$RUN
$FOR	$MACII	$XCT

5.2.3 Peripheral Interchange Program Commands

The PIP - BOSS utility commands enable the system user to perform the following functions via card-resident commands to: interchange information, perform data verification and modification, establish, view and modify file directories. The basic PIP commands are as follows:

$B	$L	$T
$C	$N	$U
$D	$R	$V
$I	$S	

5.3 INTRODUCTION TO BOSS SYNTAX

The batch control language resides on eighty-column Hollerith punched cards in a BOSS syntax. The general command syntax, to be covered in this chapter, must be followed in order to run a job string without error.

In this syntax, all BOSS commands have a "$" sign in column one of the control card. All cards beginning with this character are assumed

by the system to be control cards. The general syntactical form of a command with its parameter string is:

$COMMAND;OPTIONS ARGUMENT-1;....;ARGUMENT-N

This command string is divided into specific fields which are delimited by field separators. The delimiters used are punctuation characters and block card columns.

The command field must contain one of the initial set of commands supplied with the BOSS System (see Table 5-1).

The option field can contain any legal BOSS parameter as given in a particular BOSS command format. Refer to Chapter 6 information concerning a specific command.

The argument field can be composed of three sub-fields as follows:

1. Filename and Extension,
2. Device, Handler and Unit Number,
3. User Identification Code.

These sub-fields comprise a "field set". The recurrence of a field set, presented above as ARGUMENT-N, should occur less than or equal to ten ($\leq$1Ø) times within the command string.

For each BOSS command, there corresponds a disk-resident ASCII file called a Procedure File. The Procedure File contains XVM/DOS commands. When BOSS executes the commands in the Procedure File, it carries out the DOS function specified by the BOSS control card. The Procedure File specifies where to substitute arguments from the control card. This internal process is illustrated in Figure 5-2 "Procedure File Concept". A more detailed discussion of this process will be covered in Chapter 8.

Procedure File expansion is very similar to a macro expansion, where:

1. The DOS commands are the assembler language,
2. The BOSS command name is the macro name,
3. The contents of the BOSS control card are the macro arguments, and
4. The Procedure File is the macro definition.

The expanded DOS commands are put in a disk file called the "Run Time" File, stored under the file name PRCFIL PRC. The BOSS Command Language is essentially an open-ended language since the user can add to it by defining his own Procedure Files.

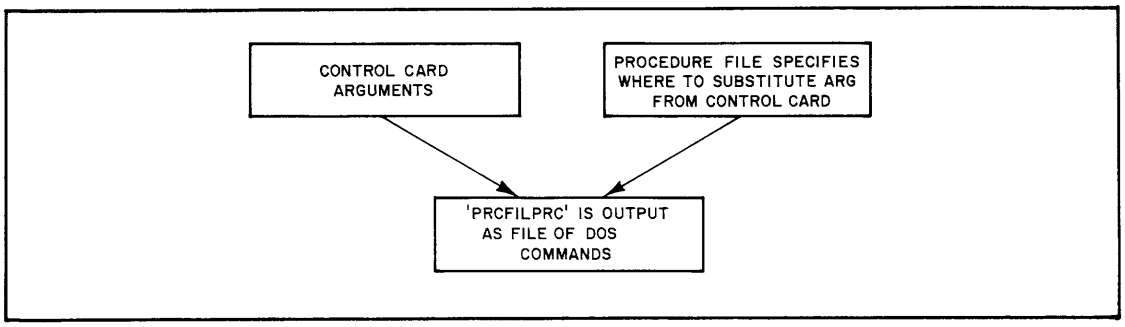


Figure 5-2 Procedure File Concept

5.4 CONTROL CARD GENERAL FORMAT

In order to ensure successful expansion of the information on control cards into a series of executable commands, all command strings must follow one of three formats:

1. Expanded Substitution,
2. Direct Substitution,
3. Peripheral Interchange (PIP). PIP commands via BOSS are identical to DOS PIP commands except for the "$" in column one of the card.

The rules for these formats can be found in the following sections of this chapter.

The following general symbology is used, for documentation purposes, to indicate to the user options within the command string:

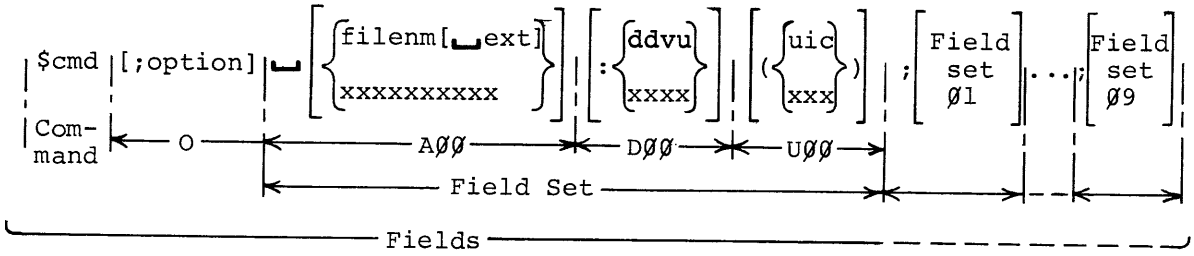
[] = Square Brackets - indicate optional quantities,
{} = Braces - indicate a choice of one

5.4.1 Expanded Substitution Card Format

The general command card format required for an Expanded Substitution File is organized with the following basic elements (see Figure 5-3).

5.4.1.1 Command String Field Delimiters - Command string field delimiters are found in Table 5-3.

Expanded Substitution Format:



Command Field - This field must be preceded by a dollar sign "$".

where:

Command	$	= Command Card. It must appear in column one. When used alone it acts as an End-of-File card.
	cmd	= Control Commands (3 alphabetic characters). Refer to Chapter 6.
	option	= One to six alphanumeric characters denoting specific options where required (refer to individual commands).
Field Set(s)	filenm	= A six character file-name plus a qualifying three character subfield, or
	ext	= A three character file-name extension (e.g., SRC, BIN, ABS, etc.).
	or { x - x	= A ten character field.
		Note: Extension subfield within "A" field is delimited by a blank (␣) if number of characters preceding are >Ø and ≤6; otherwise, go to next field.
	ddvu	= Device (dd), handler version (v), and unit number (u), or
	or { xxxx	= A four character field.
	uic	= User Identification Code, or
	or { xxx	= A three character field.
String Delimiters	;	= An Option field or the recurrence of Field Set [Ann:Dnn(Unn)]. All intervening Field Set semicolons are required prior to the last recurrence of a Field Set. A semicolon following the last Field Set indicates additional set(s) to follow on a Continuation Card.
	$	= Command field (contains any one command of the BOSS Command Language, see Table 5-1).
	␣	= Blank space (card column) and AØØ field delimiter,
	:	= "D" Field delimiter,
	()	= "U" Field delimiters.

Figure 5-3 Expanded Substitution Format

5.4.1.2 Command String Reserved Characters - In the command strings for both Expanded Substitution and Direct Substitution, certain characters must not be used within user supplied parameters, e.g., options, file-names, device-type or UIC). These reserved characters are given in Table 5-2.

Table 5-2

Special Control Card Reserved Characters

Type of Substitution	System Reserved Characters					
	[	&	:	;	(	)
Direct		S		X		
Expanded	√		X	X	X	X

where: X = Field Delimiters.
S = Special - starts next line in RTF.
√ Internal Terminator (code 33_8).
blank = Character may be used.

5.4.2 Fields

There are five basic fields within an expanded substitution command string:

1. Command Field
2. Option Field
3. Address Field (AØØ)
4. Device Field (DØØ)
5. User Identification Field (UØØ)

5.4.2.1 Command Field - The Command Field is a required field determined by a dollar ($) sign in column one of the card. In Figure 5-3, "cmd" indicates a user selection of one BOSS command (see Table 5-1). It must be 3 non-blank alphameric characters for the Expanded Substitution format.

5.4.2.2 Option Field

OPTIONS - One to six alphameric characters specifying options for the command. The first character of the Options Field must be in column 6. The Options Field is delimited by a "SEMICOLON" in column 5 and a "BLANK" in column 7,8,9,1Ø,11, or 12. The options as defined for specific commands can be found in Chapter 6.

5.4.2.3 Field Set

Field Set = [Ann:Dnn(Unn)]

A = Address = AØØ through AØ9
D = Device = DØØ through DØ9
U = UIC = UØØ through UØ9

This field has 3 subfields (described below). **The first Field Set is** delimited by a space on the left and either column 8Ø or semicolon on the right. FS(2)...FS(1Ø), are delimited by ";" on left and either ";" or end of card on the right. All blanks, with one exception (␣ext) **are ignored.**

The recurrence of a Field Set [Ann:Dnn(Unn)] is delimited by a semicolon ";". All intervening Field Set semicolons are required prior to the last recurrence of a Field Set. A semicolon following the last Field Set indicates additional set(s) to follow on a Continuation Card.

Address Field (AØØ)

Filenm ext - This optional field and subfield specifies a file name and the extension. The filenm ext field has two subfields, one of six characters and one of three characters. The first subfield is delimited either by the character count (6) or by a blank; the second subfield is by character count (9), by subfield character count (3), or by " : ". Only the first nine non-blank characters of this field are recognized by the Accounting File. All blanks are ignored in the field set. The filenm ext field contains the only exception to the rule.

A default file-name is provided for within some Procedure Files. The user should refer to the command string of a particular job command to find whether the system provides a default condition.

Device Field (DØØ)

ddvu – where: dd = device designation
v = device handler version
u = unit number (Ø - 8)

This field can be used to specify a device name and unit number. Only the first 4 non-blank characters in this field are recognized. The others are ignored. The **ddvu field is delimited by ":" and the** character count (4). The next occurrence of this field in a Procedure File would be DØØ, etc.

Unless otherwise stated by a specific command, the default assumption for this field is the System Device (e.g., DKA). The device designations acceptable by the System are as follows:

PP = Paper Tape Punch
PR = Paper Tape Reader
TT = Teletype
CDB = Card Reader
DTu = DECtape
MTu = Magtape
DK = DECdisk
DPu = Diskpack
RKu = Disk Cartridge
LP = Line Printer
VP = VP15A CRT display
XY = XY11 or XY11 Plotter

The device handler version has a default value of "A". Refer to the XVM/DOS Users Manual for more information concerning device handlers.

An example of a user supplied parameter for this field is as follows:

MTA3

User Identification Field (UØØ)

UIC – The UIC is used to specify User Identification Code. Only the first three non-blank characters in this field are recognized. The others are ignored. The UIC field is delimited by left and right parentheses, "(" and ")".

The BOSS Monitor finds disk User File Directories by seeking associated User Identification Codes (UICs), which are listed in the disk's Master File Directory. The UIC is necessary for all directory-oriented I/O to the disk. A programmer may identify himself ($LOG)

to the system with only one UIC at a time, but he may have as many UICs as he wishes, provided each is unique and none is reserved. Further, programs may simultaneously reference files under several different UIC's.

On DECtape, there is only one directory for the whole tape. On disk, there is a central directory, called the Master File Directory (MFD), but each user can have his own User File Directory (UFD). The MFD points to each UFD. Each UFD is named by a unique three character UIC. The User File Directory Table (UFDT) is part of the Resident Monitor associated with the Device Assignment Table. It indicates the User Identification Code (UIC) associated with every DAT slot (i.e., each logical device via $ASG). Disk I/O to a particular DAT slot will go to files in the UFD named by the corresponding UFDT slot.

5.4.3 Direct Substitution

```
$cmd  [AØØ] ; [AØ11;...;[A99]
```

where Ann = An open-ended field. Its recurrence is indicated by a semicolon prior to the field and its continuation by a following semicolon, prior to or in the last card column.

When processing a Direct Substitution Procedure File, BOSS places the fields on the control Card into the RTF just as they stand, with only leading spaces ignored. That is, BOSS does not necessarily expect to find file names, etc., as with normal substitution. Fields on the control cards are separated by semicolons (;), and are processed in a serial manner.

5.4.4 Peripheral Interchange (PIP) Command Formats

The PIP Utility command has two general formats. Both command strings given below have function delimiters unique to PIP and are not part of the BOSS General Format.

5.4.4.1 Single Device PIP Command Format

```
BOSS        OPERATION
COMMAND     COMMAND     DEVICE     FILE(S)  INVOLVED
CHARACTER   CHARACTER␣  MNEMONIC␣  filename ␣ Ext. ␣ (OPTIONAL SWITCHES)
```

Example:

```
$           V  ␣  DT1  ␣  FILEA,FILEB  ␣  (A)
```

Commands of this type specify PIP operations involving only a single device and a file or set of files. They specify the operations to be performed and the device involved. All switch options must be enclosed in parentheses.

5.4.4.2 Destination/Source Command Format

A. Destination Parameters

```
BOSS     OPERATION
COMMAND  COMMAND    DESTINATION DESTINATION FILE   OPTION
CHAR-    CHARACTER ␣ DEVICE ␣ (filename␣ ext)␣ SWITCHES
ACTER
```

B. Source Parameters

```
← SOURCE  SOURCE      FILE
  DEVICE␣ (filename␣ ext)
```

Example:

```
$T␣DK␣DESTFL␣SRC␣(G)←DT␣DATA␣SRV
```

Commands of this type are used to specify PIP operations which involve the transfer of data between two devices or device areas. Such commands consist of two sections separated by a back-arrow (←) delimiter.

[Destination ← Source]

5.4.5 Peripheral Interchange Commands (PIP)

BOSS XVM Peripheral Interchange Program commands are utility functions which enable the batch user to perform the following major operations via BOSS control cards.

1. Establish, view, and tailor the system file directories utilized by BOSS for systems management.
2. Initiate the interchange (transfer) of information between the disk and peripheral I/O devices.
3. Perform verification and modification procedures on information being transferred.
4. Establish, view, and modify file directories of file structured peripheral mass storage devices.
5. List and modify file directories of directory-oriented magnetic tapes.

BOSS PIP commands are identical to XVM/DOS PIP commands except for the "$" in column one. Under BOSS, the BOSS Monitor accepts PIP commands exclusively from cards, whereas under XVM/DOS PIP operates interactively via the teleprinter. These commands, along with their command string formats and examples, are summarized in Appendix A. Further discussion in this section deals primarily with PIP format information. A complete single source of information is not intended, but summary reference tables are included in Appendix A (Quick Reference Tables). A detailed coverage of the features provided by the utility program PIP XVM can be found in the PIP XVM Utility Manual.

In PIP commands, the primary operations are identified by two characters (see Table 5-3, PIP Primary Operations Summary) beginning in column one on an eighty column card. PIP commands may be executed alone or they may include one or more optional functions. For example, in the command"

```
$T␣DK←DT1␣FILEA  SRC
```

the letter "T" specifies the transfer operation, i.e., FILEA SRC is to be transferred from DECtape #1 (DT1) to the current User File Directory on the disk (DK). The back arrow (←), a multipunched character 0-8-5, is used to point to the receiving or destination argument field. These elements are formulated into a PIP command string by the user (see Table 5-4 Command String Delimiters). The PIP delimiters should not be confused with the general Procedure File syntax presented earlier.

Table 5-3

PIP Primary Operations Summary

The Primary: OPERATION	is identified by the COMMAND CHARACTER	and performs the FUNCTION
TRANSFER	$T	Transfer named data files between peripheral I/O devices.
VERIFY	$V	Check a named file for parity or checksum errors.
SEGMENT	$S	Divide a file into a specified number of segments (16 maximum) and store each segment as a separate named file.
LIST	$L	Provide listings of system directories.
NEW DIRECTORY	$N	Either clear an existing directory or, if one does not exist, establish a new one.
DELETE FILE	$D	Delete files from User File Directories.
RENAME	$R	Rename files and change protection codes for the file or the UFD in which it is listed.
COPY, Mass Storage	$C	Copy the contents of one mass storage medium onto another.
BLOCK COPY	$B	Copy the contents of one or more selected data storage blocks contained by one device onto another medium. Block copy to the disk may be performed only by the MIC user.
INITIALIZE	$I	Enable the system manager to clear all disk bit maps and restore the MFD to its original state. This command may be used only by the MIC user.
UPDATE	$U	Update the monitor's Bad Allocation Table (BAT) and Storage Allocation Table (SAT) whenever defective storage blocks are detected on the disk.

5.4.5.1 Command String Elements - The basic elements which comprise the PIP command strings are delimiters, mnemonics, and format requirements. These elements implement the major user capabilities offered by PIP. Sub-functions and parameters, which may be carried out within the context of major PIP operations, are specified in the "Optional Functions" paragraph.

A. Command Status Delimiters

Delimiters are flags which are set to separate elements of a command string. The delimiters used in PIP commands are listed and their uses described in the following table.

Table 5-4

Command String Delimiters

Delimiter:	Function	Example
␣ (space)	Separate major command string elements	T␣DK␣FILE...
← (back arrow)	Separate destination and source sections of a command string	dest. ← source T DK←DT1 FILEA SRC
: (colon)	a) In RENAME command, separates UFD name from specified UFD protection code	<JAN:Ø>
	b) Separate device mnemonic and filename	DT1:FILEA
, (comma)	a) Separate filenames within a command string list	DT1 FILEA,FILEB,FILEC
	b) Specify number of tapes or files involved in an operation when names are not needed	PR,,,(READ 3 tapes)
; (semicolon)	Separate filename and filename extension or data mode option	DT1;FILEA,FILEB SRC
<> (angle brackets)	Identify non-current UFD's	<JOE>
() (parentheses)	Identify option switches or specified protection codes	(A)
- (dash)	Separator to indicate a range of numbers	100-150

B. Optional Functions - PIP primary operations may be executed alone or they may include one or more "optional functions".

An optional function acts as a subset to one or more primary operations. It is used to specify parameters (such as data modes) and secondary operations (such as parity checks) which are to be carried out during the execution of the primary operation. An optional function is identified by an alphabetic letter enclosed in parentheses which is entered as a switch in a PIP command. Switch (A), which specifies that IOPS ASCII data is to be handled in the performance of the primary operation, and switch (Y), which indicates that a file segmentation operation is to be performed during the primary operation, are two examples of optional functions and their switches. The use of switch (A) in the following command:

```
$T  DK←DT1␣FILEA␣(A)
```

specifies that the data contained by FILEA is in IOPS ASCII form. All PIP optional functions are listed and described in Table 5-5.

Table 5-5 PIP Optional Functions

OPTION	COMMAND CHARACTER	FUNCTION
IOPS ASCII Data Mode	(A)	Specifies the type of data (i.e., format) handled by the primary PIP operation.
IOPS Binary Data Mode	(B)	
Image Alphanumeric Data Mode	(I)	
Dump Data Mode	(D)	
Image Binary Data Mode	(H)	
Bad Parity & Checksum Check & Correction	(G)	Outputs error messages and the lines containing errors detected. Corrective actions are permitted.
Tab to Space Conversion	(E)	Causes all tabs found in the data handled to be expanded into a series of spaces.

Table 5-5 PIP Optional Functions (Cont'd)

OPTION	COMMAND CHARACTER	FUNCTION
Convert Multiple Spaces to Tabs	(C)	Causes each group of two or more spaces encountered during the primary operation to be converted to a TAB.
Segment Files	(Y)	Indicates that the file being transferred is to be segmented.
Combine Files	(W)	Combines one or more separate files into a single file.
Form Feed	(F)	Causes a form-feed and a RETURN character to be inserted after the detection of each .EJECT statement or after every 55_{10} lines.
Delete Trailing Spaces	(T)	Causes all trailing spaces to be deleted from alphanumeric data being handled during the primary operation.
Delete Sequence Numbers	(Q)	Used for punched card input, this option causes all input sequence numbers to be deleted.
Reserve QAREA with New Directory	(Snn)	Used for DECtape devices only, this option initializes any existing directory or establishes a new directory, and causes a CTRL Q area to be allocated on the device. The size of the allocated area may be specified (i.e., nn) by the user.
New Directory	(N)	Performs the same function as the N primary operation. It either clears an existing directory or, if one does not exist, establishes a new one.
List MFD	(M)	Enables standard UIC users to obtain a listing of all unprotected UFD's contained by the device (disk). The MIC user will obtain a listing of all UFD's contained by the device.
List SYSBLK	(L)	Enables the user to obtain a listing of the system SYSBLK directory.
List UFD with Auxiliary Data	(P)	Causes file RIB data to be added to a UFD listing.
Delete UFD	(K)	Removes UFD entry from MFD.
Delete All Truncated Files	(X)	Causes all truncated files contained by the current or specified UFD to be deleted.

Table 5-5 PIP Optional Functions (Cont'd)

OPTION	COMMAND CHARACTER	FUNCTION
Vertical Forms	(V)	Translates the first character of each record to a special character which, when interpreted by the line printer handler will produce forms control functions. The translation will occur according to the FORTRAN conventions shown below. Character Found — Translated To — Meaning to LP Handler '1' — FF,14_8 — Skip to top of form '+' — DLE,20_8 — Overprint '0' — DC1,21_8 — Double Space ' ' (space) — LF,12_8 — Single Space anything else — LF,12_8 — Single Space

Table 5-6 Source and Object Code File Extension/BOSS Operation

COMMAND OPERATIONS	I/O CODE MODULES	
	INPUT AND SOURCE FILE EXTENSION	OUTPUT AND OBJECT FILE EXTENSION
1. Language Translation $FOR $ASM	Source to → Module SRC	Relocatable Object Module BIN
2. CHAIN Operation $MAP $OVL $LNK	Relocatable to → Module BIN	LOAD** Module Absolute XCU
3. Program Execution $XCT *	LOAD to → Module XCU, XCT	CORE for execution

* If load module file-name has 3 characters, then $XCT FIL can be written as $FIL, and "FIL" must not be the same as a legal BOSS command.

** Load module equals two files, one with an XCU extension, and the other with an XCT extension.

5.5 INPUT/OUTPUT CODE MODULES

The object code produced by FORTRAN IV ($FOR) and MACRO ($ASM) is relocatable binary (BIN file extension) which is made absolute (XCU file extension) by the CHAIN operations $MAP, and $OVL. A relocatable object program may be loaded into any part of memory. Programs in absolute forms can only be loaded into or executed from the locations assigned by the language translator.

The CHAIN program ($MAP or $OVL and $LNK) relocates object programs to create absolute executable core image files. In the process of creating these files, CHAIN joins programs by supplying definitions for global symbols (i.e., symbols which are referenced in one program and defined in another) thus providing a communication linkage between separately assembled programs.

CHAIN outputs two files, one consisting of the executable binary of the programs input to it (XCU file) and the other containing control information (XCT file).

The EXECUTE program ($XCT command) is used to run the user's executable binary files utilizing the control information contained in the XCT file. Detailed information on the use and features of CHAIN and EXECUTE is provided in the CHAIN XVM/EXECUTE XVM Utility Manual.

The above operations are outlined in Table 5-6 Source to Object Code File Extension/BOSS operation.

5.5.1 Specifying Data Modes[1]

Operations which involve the interchange of data require that the form of the data being handled (i.e., its data mode) be indicated in the initiating command string.

Data modes in BOSS are specified either by the filename extension of the file being transferred or by an equivalent PIP data mode option switch. The Data Mode indicators recognized by PIP are defined below. The Data Modes with their respective filename extensions and option switches are given in Table 5-7.

[1] Refer to the XVM/DOS Users Manual for detailed descriptions of data modes.

The available PIP data mode switches and their uses are:

a. (A) IOPS ASCII Switch - Files containing data in IOPS ASCII form require the use of the PIP (A) data mode switch. ASCII files are identified by the extensions:
   1) SRC. or
   2) a 3-digit numeric extension (e.g., 004).

b. (B) IOPS Binary Switch - Files containing data in IOPS Binary form must be handled using the (B) switch. The filename extension BIN is used to identify binary files.

c. (I) Image Alphanumeric Switch - The (I) switch is required during transfer of Image Alphanumeric Files and, as the name Image implies, maintains the File data in its exact form as read from the source file. The (I) switch must specifically be used when copies of paper tapes in either HRM or RIM hardware reader modes (MACRO .ABS or .FULL paper tape) are required.

d. (H) Image Binary Switch - Binary files (extension BIN) to be maintained in their exact form must be transferred using the (H) data mode switch.

e. (D) Dump Mode Switch - Files containing data in an absolute binary form (extension ABS) must be handled using the (D) mode switch.

The specific combinations of data mode switches and optional function switches which are permitted in each of the PIP primary operations are given in Appendix A.

Table 5-7

Data Modes and Data Mode Indicators

		DATA MODES				
		IOPS ASCII	IOPS BINARY	DUMP	IMAGE ALPHA-NUMERIC	IMAGE BINARY
FILENAME EXTENSIONS	SRC	✓				
	BIN		✓			
	ABS			✓		
	Numeric	✓				
OPTION SWITCHES	(A)	✓				
	(B)		✓			
	(D)			✓		
	(I)				✓	
	(H)					✓

5.6 LINE EDITOR

BOSS is equipped with a line editor to edit on-line files. The line editor is called by the $MAC11, $FIL, $FOR, and $ASM commands. The line editor operates on alphameric lines. Three functions are provided:

1. Insertion,
2. Deletion, and
3. Substitution.

New lines can be inserted following a specified line. One or more contiguous lines can be deleted; and one or more contiguous lines can be substituted for one or more lines (see Table 5-8 Line Editor). In a substitution procedure the number of lines deleted need not be equal to the number of lines inserted.

All line editor commands have a "-" (minus sign) in column one. Source lines to be inserted appear on the card as they will be inserted in the main storage file (no control character necessary).

Table 5-8

Line Editor

Editor Command Cards		
Command	Function	Edit Cards
$-L_1$	Insert - following numbered line L_1, insert source line card(s).	command card, followed by source line card(s)
$-L_1,L_2$	Delete - numbered line L_1, or L_1 through L_2	Command card only
$-L_1,L_2$	Substitute - delete L_1 or L_1 through L_2, substituting source line card(s). Reorder remaining line numbers in file.	command card, followed by source line card(s)

Where: 1. L = decimal line number of edited file.

$L_1 \leq L_2$

2. The Line Editor is called by the $MAC11, $FIL, $FOR, and $ASM commands. All Line Editor command cards must have a "-" in column one.

5.6.1 Insert

Insertion is specified by the command

```
-l
```

followed by the source lines to be inserted, one per card. Line one (Ll) is a decimal line number in the original source file after which the new source lines are inserted. The file is scanned through Ll. Insertion begins after Ll for as many lines as there are source line cards. For example, if FILEA has five source lines:

```
LINEl
LINE2
LINE3
LINE4
LINE5
```

And the following line edit command is issued:

```
-3
LINE A
LINE B
```

then FILEA will look like:

```
LINEl
LINE2
LINE3
LINEA
LINEB
LINE4
LINE5
```

5.6.2 Delete

Deletion is specified by the command

```
-l,-2
```

where Ll and L2 are decimal line numbers (Ll$\leq$L2) in the original source file. All lines from line Ll to line L2 inclusive are deleted.

For example, FILEA has 7 source lines:

```
LINEl
LINE2
LINE3
LINEA
LINEB
LINE4
LINE5
```

and the following line edit commands are issued:

```
-1,-2
-4,-6
```

Then FILEA will look like

```
LINE3
LINEA
LINEB
```

5.6.3 Substitute

Substitutions are specified with a deletion command followed by source cards. The range of lines specified in the deletion commands are deleted first; then the source lines following the deletion command are inserted in their place. For example, assume FILE A has the following three lines:

```
LINE1
LINE3
LINE5
```

and the following line-edit command is issued:

```
-1,2
LINEA
LINEB
LINEC
```

The final edited file will look like the following:

```
LINEA
LINEB
LINEC
LINE5
```

5.6.4 Editing Operation

The Line Editor uses two input DAT slots (the input file to be edited and the Card Reader) and one output DAT slot (the output file). The input and the output files may have the same name or different names, depending on the arguments used in the $MAC11, $FIL, $FOR, or $ASM. If the names are the same, the original file is lost, and only the edited file is kept. If the file names are different, then both the original file and the edited file are kept. The input file is read only once.

This requires that the line edit commands be ordered so that the line numbers are in ascending sequence. If two edit commands have line numbers out of sequence, an error message is printed. The input and output files are then closed, and the edit is stopped. In this way all the edits are made current up to but not including the one with the out-of-sequence line number.

5.7 BOSS PREPROCESSOR (B.PRE)

The preprocessor B.PRE is a BOSS XVM utility program. Its primary function is to preprocess MACRO MAC11, and FORTRAN IV (F4) Source Files for BOSS. It can be used generally for any file. This program is called by BOSS anytime an Edit (-) control card appears. It will then do the following:

1. If no data cards (i.e., cards without a "$" sign in column 1) follow the control card, B.PRE exits.
2. If data cards follow the Edit control card, B.PRE reads each data card and an associated ASCII file with their images. The file will be read from DAT -2.

An example of the Edit substitute preprocess for a FORTRAN compilation is shown below:

```
$JOB COPY;3Ø;CMS
$C DK←DTØ
$C DK←DT1
$END
$JOB TST1;3Ø;CMS
$FOR;B AEØ1ØØ
-15,15
     WRITE (6,2ØØ)
-24,24
1ØØ  WRITE (6,2Ø1)
-28,28
1Ø2  WRITE (6,2Ø2)
-32,32
1Ø4  WRITE (6,2Ø3)
-36,36
1Ø6  WRITE (6,2Ø4)
-4Ø,4Ø
11Ø  WRITE (6,2Ø5)
-45,45
```

When the line editor is used, the following rules should be followed:

1. **A disk must be used for input/output, or the following error message will be printed:**

   NON-DISK I/O DEVICE

   and an exit to the next control card will occur.

2. The same device/unit should be assigned to device-1 and device-2, (see Chapter 6, $MAC11, $FOR, and $ASM commands); if not the following message will be output and an exit to the next control card will occur.

   DIFFERENT DEVICES ASSIGNED TO .DAT-14 & -15

3. The same UFDT must be used for UIC-1 and UIC-2, otherwise the following error message will be printed, and an exit to the next control card occurs:

   DIFFERENT UIC ASSIGNED TO .DAT-14 & -15.

4. The user should use care to ensure that no line-edit commands are to be performed past the last line of the file, or errors will occur.

5.8 CONTINUATION CARD

The continuation card permits the user to continue command arguments via additional control cards. For Expanded Substitution this card provides for additional Field Sets. For Direct Substitution it provides for additional fields. The format for the continuation Card follows.

```
$*ccc[additional arguments]
```

where: $* = Continuation card.
ccc = Comment field (documentation only).
Additional arguments beginning in column 6.

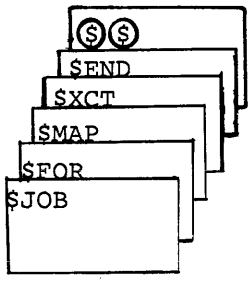
Continuation Cards are signaled by trailing punctuation in the expanded substitution and direct substitution. A semicolon followed by a carriage return or ALTMODE, neglecting spaces, means that there is a continuation card for the card being processed. If the card following this card is not a continuation card ($*), then an error occurred, and BOSS prints

ILLEGAL COMMAND CARD SEQUENCE

For direct substitution the appearance of the (;) does not terminate the field being processed.

5.9 END OF DECK CARD (FOR SPOOLED UC15 SYSTEMS ONLY)

The end of deck card enables the user to indicate end of job deck to the spooler. This card should be present at the end of a job deck as indicated below:



end of deck card ($$ card)

$ denotes an ALT.MODE

If the end of deck card is not present in a spooled UC15 system, the system will hang, if at least 5 cards are not present after the $END card. This is because card images are blocked, six cards to a block. These blocks are not written out onto the disk (spooled) unless the block is full or the end of deck card is read.

The end of deck card has no effect if spooling is not enabled since the card reader handler for CR11 throws it away when read; this card is not shown in job deck examples other than the one above.

CHAPTER 6
CONTROL COMMANDS

6.1 INTRODUCTION

Once the general card format is understood in terms of Expanded Substitution, Direct Substitution, and Field Continuation, the programmer can combine that knowledge with the Batch Command Language (BCL) syntax. The BCL commands are presented in a self-contained reference format to provide the user with the optimum information per command. For ease of reference, each command is in alphabetical order and not in a procedural sequence.

BCL provides the user with several levels of program and job control. Control functions are as follows:

Job Control,
System Program Load Control,
User Program Load Control,
File Control,
Add-File Control,
Continuation Control,
Message Control,
Line Editing Control,
File Interchange Control (PIP),
Emulation.

The individual BOSS commands required to implement these functions are given in Table 6-1 "Command Functions".

This section provides the user with the following information for each BOSS command as follows:

Function,
Format,
Argument,
Technical Note,
Example.

TABLE 6-1 Command Functions

PROGRAM CONTROL FUNCTIONS											
COMMANDS	Peripheral Interchange (PIP)	Line Editor	Job	Monitor	File	Add-File	Message	System Program Load	User Program Load	Emulation	No Procedure File
$					X						
$$						X					
$*				X							
-		X									
$ADD						X					X
$ASG				X							
$ASM		X						X			
$B	X										
$BNK				X							
$BUF				X							
$C	X										
$CHN				X							
$CMP								X			
$CRT						X					X
$D	X										
$DIR					X						
$DLG					X						
$DMP								X			
$DOS										X	
$END			X								X
$FIL		X			X						
$FOR		X						X			
$I	X										
$JOB			X								
$KEP				X							
$L	X										
$LCM								X			
$LIB								X			
$LNK									X		
$LOG					X						
$MAC11		X						X			
$MAP									X		
$MIC					X						
$MNT							X				
$MSG							X				
$NSW							X				
$N	X										
$NDR					X						
$OVL									X		
$PAG				X							
$PRT					X						
$QDP			X								
$R	X										
$RUN								X	X		
$S	X										
$T	X										
$U	X										
$V	X										
$XCT									X		
$XVM				X							

The command FUNCTION states the general command definition. A FORMAT is given which defines each argument in the command string. The command string indicates the maximum number of elements supported by the system. When any of these string elements is not required for a given implementation, its field delimiter (;) must remain in the string in the order in which it normally appears (refer to Chapter 6). Within the command string, each element is defined in detail under ARGUMENT(S). When the ARGUMENT is given in capitals, it must be implemented as shown: when in lower-case, the field content is supplied by the user. The user, when supplying arguments, should conform to the command requirements as specified in the argument definition and its associated TECHNICAL NOTE. EXAMPLE(S) are provided where variations in command string usage might be confusing.

All command string arguments are optional to the user, excluding message and literal-value elements. In addition to optional elements within the command, the user is provided with certain control parameters by "default". These default assignments are automatically assigned by the System. Some default values must be supplied by the user at System Generation time. BOSS supplied system defaults are noted in the ARGUMENT definition.

6.2 $ADD

FUNCTION: The $ADD card permits a BCL card file, created by $CRT, to be added to the input stream at a designated point.

FORMAT: $ADD␣file-name

ARGUMENTS: file-name = (Default - TMP ADD) The name assigned to the ADD file via $CRT.

TECHNICAL NOTES:

1. The $ADD command permits the BOSS user to specify a file of BOSS language commands to be inserted in the card input stream at a required point. All line images of the file specified by "file-name" are assumed to be BCL commands.
2. No data cards may appear in the ADD file.
3. $CRT, $ADD, and $JOB are illegal records in an ADD file.
4. ADD files are assumed to be on the system device, and under the current UIC.
5. All the lines in the ADD file will be executed before switching the input stream back to the card reader.
6. If the file name specified on the $ADD card does not correspond to a file, the following error message will be output:

   'NON-EXISTENT ADD FILE'

   and the next control card will be executed.
7. If, after an ADD file is opened, that ADD file is modified, the following error message will be output:

   DIFFERENT ADD FILE

   and the next control card will be executed.
8. Restriction-two $ADD cards in sequence will result in an IOPS11 error. It is always necessary to put any legal BOSS card other than $ADD, $JOB, or $SEND between two successive $ADD cards.

6.3 $ASG

FUNCTION: The $ASG card permits the temporary reassignment of the Monitor's Device Assignment Table (DAT). In addition, the corresponding slots of the User File Directory Table (UFDT) can be reassigned to UIC's other than current assignments. The DAT and UFDT default assignments are made at SGEN time.

FORMAT: $ASG␣dat-slot:device(uic)

ARGUMENTS:

dat-slot = This field contains the DAT slot reassignment. The assignment range is from -15 to +77 (octal). Slots reserved for exclusive System use are: -2, -3, -7, +5, and +6.

device = (Default - System Device) This four character field denotes the device type, Handler version, and unit number. The default for the Handler version = A and the unit number = Ø for MT or RK or DP or DT. Table 6-2 gives device types.

uic = (Default - Current UIC) This three character field denotes the User Identification Code (UIC) to associate with the corresponding DAT slot.

TECHNICAL NOTES:

1. The Device Assignment Table (DAT) slot structure is the same as in XVM DOS, except that in BOSS mode DAT slots +5 and +6 are permanently assigned to the card reader and line printer respectively. The user cannot modify them. Attempted use of these slots will result in the error condition:

   PERMANENT DAT SLOT

2. A $ASG card preceding some System program load commands is meaningless since the BOSS procedure File contains standard DAT assignments. DAT slot assignments are specified via the system program calling commands (e.g., $FOR, $ASM, $DMP, $LIB, $MAC11, and $LCM.)

3. Since the number of negative slots does not change (-15 is the lowest negative slot), the user need only be concerned with the number of positive slots available. $DOS SCOM or a $DOS REQUEST USER command will output sufficient system information to let the user determine the number of positive DAT slots available, and the system assignments.

Table 6-2

Device Type Codes for $ASG Command

DK	**DECdisk**
DP	**Disk Pack**
RK	**Disk Cartridge**
DT	**DECtape**
TT	**Teleprinter**
PP	**Paper Tape Punch**
PR	**Paper Tape Reader**
LP	**Line Printer**
CD	**Card Reader**
MT	**Magtape**
VP	VP15 GRT Display
XY	XY11/XY311 Plotter
LV	LV11 Raster Scan Plotter (optional)
DL	DL11 10/15 Communications Link (optional)
SY	System Disk

6.4 $ASM

FUNCTION: The $ASM card calls the PDP-15 MACRO Assembler and the Line Editor.

FORMAT:
```
$ASM;options␣file-name-1:device-1(uic-1)
          ;file-name-2:device-2(uic-2)
          ;file-name-3:device-3(uic-3)
          ;file-name-4:device-4(uic-4)
                    ;:device-5(uic-5)
                    ;:device-6(uic-6)

Line Edit Card/s, or
Source Card/s or
next BCL command
```

ARGUMENTS: options = (Default - B L) A six character option field. The options with their associated actions are given below:

Option		Action
A	=	Print symbols at end of PASS 2 in alphanumeric sequence.
B	=	Generate a binary file.
C	=	Program areas that fall between unsatisfied conditionals are not printed. It is not necessary to include the L option if this option is used.
E	=	This option enables the user to have any errors occurring during assembly printed on the Console Teleprinter in addition to device-5. The L or N switch should be used with the E option.
F	=	Process the MACRO definition file.
G	=	Print only the source line of a macro expansion. It is not necessary to type the L option.
H	=	Print user symbols after PASS 2. It is used in conjunction with A, V or S options.
I	=	Ignore .EJECT's. The pseudo-op is treated as a comment.
L	=	Generate a listing file on the requested output device.
N	=	Number each source line (decimal). If this option is used, it is not necessary to type the L option. If L and X are typed MACRO assumes that the N option was typed also.
O	=	Causes the assembler to omit the source extension and the linking loader code 33 from the binary file.
P	=	Before assembly begins, read assembly parameters from device-2. Only direct assignments may be used.
S	=	Same as selecting both A and V options.
T	=	Generate a table, during Pass-1, with the page number and the text of all assembled .TITLE statements in the program.

Option Action

V = Print symbols at end of PASS 2 in value sequence.
X = Load PASS 3 to perform the cross-referencing operation. Then continue assembling.
Z = Read MACRO definition during file PASS 1 and 2.

file-name-1 = Source File (Edit Output/New file-name)/or Macro input file-name
device-1(uic-1) = Standard format.

file-name-2 = Parameter File
device-2(uic-2) = Standard format.

file-name-3 = MACRO Definition File
device-3(uic-3) = Standard format.

file-name-4 = Edit Input (Old file-name)
device-4(uic-4) = Standard format.

device-5 = Print device
device-5 (uic-5) = Standard format.

device-6 = Binary output device (default is the system device)
device-6 (uic-6) = Standard format.

TECHNICAL NOTES:

1. The $ASM command provides the user with the symbolic programming capabilities of the MACRO XVM assembler which includes macro instruction generator.

2. When the user program has been successfully assembled, the output is a relocatable "object module" (BIN extension). This module can be further processed by using the $MAP or $OVL and $LNK commands. These commands will produce a "load module" (absolute file) with a file extension of XCT. The XCT file is actually two files, one with an XCT extension and the other with an XCU extension. The user can assemble for an absolute file (.ABS files cannot be chained). For further information see the CHAIN XVM/EXECUTE XVM Utility Manual. The load module must be executed using the $XCT command to run the user's object program.

3. If the line editor is to be used, the following rules should be followed:

   a. DECdisk, disk pack, or disk cartridge must be used for input/output, or the following error message will be printed:

      'NON-DISK I/O DEVICE'

      and an exit to the next control card will occur.

   b. The same device/unit should be assigned to device-1 and device-4. If not, the following message will be printed and an exit to the next control card will occur.

      'DIFFERENT DEVICES ASSIGNED TO .DAT-14 & -15'

c. The same .UFDT must be used for UIC-1 and UIC-4, otherwise the following error message will be printed, and an exit to the next control card will occur:

'DIFFERENT UIC ASSIGNED TO .DAT-14 & -15'

4. The $ASM command can be used to:

   a. Assemble a file already existing. The $ASM card must specify file-name-1, and the next control card should follow the $ASM command.

   b. Create a new file and assemble it. The $ASM card must specify file-name-1, and the "source" deck should follow this command.

   c. Assemble without creating a file. No file-name should be specified, and the "source" deck should follow the $ASM command.

   d. Edit an already existing file and assemble it. The file-name-1 must be specified, and line-edit commands should follow the $ASM command.

   e. Edit an already existing file, create a new "edited" file and assemble the newly created file. The original source file-name is specified in file-name-4. The new source file-name is specified in file-name-1, and the $ASM command should be followed by the line-edit commands.

   Depending on which function (see options above) the $ASM command specifies, the command can be followed by:

   A MACRO source deck,
   Line-Edit Commands,
   The next Control Card.

5. For additional information refer to the MACRO XVM Assembler Language Manual.

6.5 $BNK

FUNCTION: The $BNK card selects BANK Mode system operation (8K).

FORMAT: $BNK

TECHNICAL NOTES:

1. The default addressing mode is established at System Generation time by the System Manager.
2. In Bank Mode, Index Register usage is not permitted. User programs (including device handlers) are loaded within 8K memory banks using 13 bit address relocation. System library routines are loaded from the library contained in the BNK (uic) system directory. A $BNK card sets the system to operate in BANK mode. Conversely, a $PAG card sets the system to operate in PAGE Mode.

6.6 $BUF

FUNCTION: The $BUF card temporarily changes the default value for the number of buffers available.

FORMAT: $BUF␣number

ARGUMENTS: number = number (decimal radix) of buffers desired.

TECHNICAL NOTES:

1. This command temporarily changes the default value for the number of buffers available (in the Monitor's buffer pool) for disk I/O and for the .GVBUF and .GTBUF requests. The default value is restored by the System whenever the Non-Resident Monitor returns (after the succeeding system program or user program load). This value is set during system generation along with the actual size of the buffers to be allocated (the default values for systems as initially distributed are: Number of Buffers = 3, Buffer Size = 500_8. BOSS automatically adds one buffer to facilitate the Run Time File).
2. The user should exercise care when issuing this command by considering the trade-off of available core versus the convenience of a large number of buffers. The handlers for the three disk types always obtain the buffers required for their operation from the same common buffer pool. One of these buffers is required for each opened file. Terminal errors can result when an insufficient number of buffers is available. Alternatively, program loading errors can occur when the number of buffers allocated results in an insufficient amount of core for program loading. The user need not be concerned about buffer availability for system programs since each system program has its own default parameter for the maximum number of buffers required (e.g., MACRO has 4, EDITOR has 4.). If, however, he issues a $BUF command, the number on that card will take precedence over the default for a system program.

6.7 $CHN

FUNCTION: The $CHN card is used to specify 7-track or 9-track Magtape operation.

FORMAT: $CHN␣{7 / 9}

ARGUMENTS: 7 = 7-track Magtape
9 = 9-track Magtape

TECHNICAL NOTES:

1. The track count is set at System Generation time and can be modified by $CHN.
2. If 9-track is specified, density is assumed to be 800 BPI.

6.8 $CMP

FUNCTION: The $CMP card provides the user with a utility program which compares two source programs via the SRCCOM program.

FORMAT:

```
                  ⎧A⎫
$CMP; ⎨C⎬ ␣newfile-name:device-1(uic-1)
      ⎩M⎭
      ⎩Rn⎭
            ;oldfile-name:device-2(uic-2)
```

ARGUMENTS: Options - SRCCOM switch options are as follows:

A = Switch A is used to specify an abbreviated output format (Affected lines are not printed).

C = Used to ignore ASCII characters outside of the range 240-337 except for carriage return and horizontal tab characters.

M = Used in MACRO XVM comparisons to ignore comments and to perform space tab conversions.

Rn = Used to determine the number of consecutive lines that SRCCOM must rematch in order to detect an insertion or deletion (i.e., rematch factor). This switch is written as Rn, where: n is any digit from 1 to 9. The default value of n is 6.

newfile-name = This ten character field specifies the file-name of the new symbolic program to be compared to the symbolic program in the file specified by old file-name.
device-1 (uic-1 - The device which newfile is on Standard format.

oldfile-name = This ten character field specifies the file-name of the old symbolic program.
device-2(uic-2) = The device which oldfile is on Standard format.

TECHNICAL NOTES:

1. The $CMP command calls a utility program SRCCOM which compares two symbolic programs (SRC - written in IOPS ASCII) and indicates the differences between the compared programs. Output statements are made to the line printer.
2. When comparing programs which have had extensive modifications made in relatively small areas, the value of "n" in the R switch should be 6 or higher. Smaller values of "n" may result in erroneous SRCCOM reports.
3. For further information refer to the SRCCOM XVM Utility Manual.

EXAMPLES:

1. To compare MACRO program NEWFIL to MACRO program ORGFIL with M switch on:

```
$CMP;M␣NEWFIL␣SRC:DK(EEM);ORGFIL␣SRC:DK(EEM)
```

2. To compare NEWFIL to ORGFIL with all switches on:

```
$CMP;A,C,M,R5␣NEWFIL␣SRC;ORGFIL␣SRC
```

3. To compare two programs with no switches on:

```
$CMP␣NEWFIL;ORGFIL
```

6.9 $CRT

FUNCTION: The $CRT card is used to create $ADD files containing BOSS Control Language (BCL) images.

FORMAT: $CRT file-name

ARGUMENT: file-name = The name to be assigned to the ADD file. (Default - TMP ADD)

TECHNICAL NOTES:

1. The $CRT command is used to create ADD files. ADD files are files containing BCL images. All cards following the $CRT card will be written on the specified file until a "$$" card (dollar sign in columns one and two) is reached.
2. Illegal ADD file cards are: $CRT, $ADD, and $JOB. Also data cards must not be included in an ADD file. In a case of error, the following message is output:

   'ILLEGAL COMMAND FOR ADD FILE'

   and the job is terminated.
3. Restriction - two $CRT cards in sequence will result in an IOPS11 error. It is always necessary to put any legal BOSS card other than $CRT, $JOB or $SEND between two successive $CRT cards.

EXAMPLE:

```
$CRT␣TMPFL2
BCL cards
.  .
   .
$$
```

6.10 $DIR

FUNCTION: The $DIR card causes the directory of the device specified in the argument list to be output to the line printer.

FORMAT: $DIR␣device(uic)

ARGUMENTS:
device = (Default - System Device) This four character field specifies a directoried file device. Device type, handler version, and unit number can be specified. The default for the handler = A, and the unit number = Ø for DK, DP, RK, or DT.

uic = (Default - Current UIC) This three character field denotes the User Identification Code (UIC) to associate with the corresponding DAT slot. It specifies the UIC in case the device is DK (DECdisk) RK (Disk Cartridge) or DP (Disk Pack). It is ignored in all other cases.

EXAMPLES:

```
Command -
$DIR␣DK(DOE)
$DIR␣DT3
Job String -
$JOB␣TEST
$DIR␣DT1
$END
End-Of-File Card
```

6.11 $DLG

FUNCTION: The $DLG card causes the LOG-OUT of the current UIC.

FORMAT: $DLG

TECHNICAL NOTES:

1. The $DLG command causes the LOG-OUT of the current UIC, and the reinstating of the default UIC "SCR". No arguments are needed for the $DLG command.
2. In addition, the $DLG command causes all default device assignments to be reset. The $ASG command should be used to make new device assignments.

6.12 $DMP

FUNCTION: The $DMP card provides the user with the ability to output specified core locations from the CTRL Q area to any available listing device, or any physical block of a block oriented device.

FORMAT: $DMP␣command

ARGUMENTS:

command: (Default - ALL) This field specifies any legal DUMP command as specified in XVM/DOS. The DUMP commands are:

All = The ALL command will list the contents of the CTRL Q area from location 5 to the highest memory location of device-1 (DAT -14) upon device-2 (DAT -12). Highest memory location is defined as:

1. ↑Q area on disk devices: The highest memory location dumped to the ↑Q area as determined by appropriate .SCOM locations in the image on the ↑Q area.
2. ↑Q area on DECtape: The current memory size, as determined by appropriate .SCOM locations in core. To dump other DECtape ↑Q area sizes use the ALL (nn) option described in note 4.

a - b = List the CTRL Q area between absolute address a and b (in octal) of device-1 upon device-2.

Where "a" is greater than or equal to 5 and "b" is less than or equal to:

1. ↑Q area on disk: The highest memory location (defined above)

or

2. ↑Q area on DECtape: 377777_8 (131071_{10}).

nnn# = List the contents of block nnn (octal of device-1 upon device-2. For DECtape, the command nnn# may be followed by a minus (-); block nnn will be read in the reverse direction.

TECHNICAL NOTES:

1. The $DMP command provides the user with the ability to output, on any available listing device, specified core locations which have been stored on a bulk storage device using the $QDP (CTRL Q) command. The QDUMP argument of the $JOB card is an unconditional (ALL) dump. $QDP is initiated via a terminal error and is conditional on the command placement within the job deck.
2. If device-2 (listing output) is a directoried device, the listings output will go to a file named MEMORY DMP.
3. If device-1 is assigned to any device other than DECtape, Disk Pack or DECdisk, BOSS will exit the job string to the next control card. In the special case when the system device is Disk Pack, and device-1 is assigned to a DECdisk, an exit to the next control card will also occur.
4. The "ALL(nn)" command will specify that the DECtape ↑Q area size (nn=24, 32, 40, 48, ... 120, 128) is different than that of the current system memory size. This option is valid only if device-1 (.DAT-14) is assigned to DECtape.
5. The first block number and size of the CTRL Q area can be determined by listing SYSBLK with BOSPIP:

$L␣LP←DK␣(L)

6.13 $DOS

FUNCTION: The $DOS card is a general procedure file for implementing XVM/DOS command strings.

FORMAT: $DOS␣dos-command-string

ARGUMENT: dos-command-string = (default - Alt Mode) This field is used to specify any legal XVM/DOS command string.

TECHNICAL NOTE:

The $DOS command is a "direct substitution" procedure file which can be of general use for giving XVM/DOS command strings.

EXAMPLE:

```
$JOB␣TEST
$DOS␣REQUEST
$DOS␣I␣ERRORS
$DOS␣SCOM
$END
(Dos End-Of-File Card)
```

Where:

1. REQUEST - The REQUEST command causes a typeout of the I/O Device Handlers currently associated with the slots of the Monitor's Device Assignment Table (DAT).
2. INSTRUCT - The INSTRUCT command causes a listing of either the keyboard commands (INSTRUCT) or system errors (INSTRUCT ERRORS).
3. SCOM - The SCOM command causes the typeout of XVM/DOS system information. The information includes: (a) available I/O device handlers, (b) system default parameters, (c) important core addresses, and (d) the Priority Interrupt System's I/O Device Skip Chain order.

   NOTE: For more information on DOS commands refer to the XVM/DOS User's Manual.

6.14 $END

FUNCTION: The $END card marks the end of a Job.

FORMAT: $END

TECHNICAL NOTES:

1. When a Job is closed with a $END command, a message is output on the line printer. The message specifies in minutes the time taken for the Job to complete, the time of day of job initiation and termination, and the date.
2. After a $END card, BOSS will look for a $JOB card. Therefore a $END card should be followed only by a $JOB card, or an End-Of-File card.

6.15 $FIL

FUNCTION: The $FIL card is used to create a file or to edit an existing file.

FORMAT: $FIL␣new-file:device(uic-1);old-file:device-2(uic-2)

ARGUMENTS:

new-file = (Default - FILTMP SRC) This ten character field specifies the source file-name and extension of the file to be created.

device-1 = (Default - System Device) This four character field contains the device on which to create or edit the file. This must be a directoried device.

uic-1 = (Default - Current UIC) This three character field contains the name of the UFD to use when the device assignment is disk.

old-file = (Default - FILTMP SRC) This ten character field specifies the original file-name to be edited if the desired edited file-name "new-file" is different.

device-2 = (Default - System Device) This four character field contains the device on which the original file is stored. This must be a directoried device, the same as device-1.

uic-2 = (Default - System Device) This three character field contains the name of the UFD to use when the device assignment is disk. UIC-2 should be the same as UIC-1.

TECHNICAL NOTES:

1. If the line editor is to be used, the following rules should be followed:
   a. A disk must be used for input/output, or the following error message will be printed:

      'NON-DISK I/O DEVICE'

      and an exit to the next control card will occur.

b. The same device/unit should be assigned to device-1 and device-2; if not, the following message will be output and an exit to the next control card will occur:

'DIFFERENT DEVICES ASSIGNED TO .DAT-14 & -15'

c. The same uic must be used for UIC-1 and UIC-2; otherwise the following error message will be printed, and an exit to the next control card will occur:

'DIFFERENT UIC ASSIGNED TO .DAT-14 & -15.'

2. The $FIL command is used to create a file, or to edit an existing file on a directoried device. To create a new file, the file-name should be specified in "new-file" and the source deck should follow the $FIL command.
3. To edit an existing file, the file-name should be specified in "new-file" and the line-edit commands should follow the $FIL card.
4. To edit an existing file and create a new edited file while leaving the original untouched, the original source file-name should be specified in "old-file". The new source file-name should be specified in "new-file". The line-edit commands should follow the $FIL command.

EXAMPLES:

Create -

```
$FIL␣NEWFIL␣SRC
Source Deck
.
.
BCL card of $
```

Edit -

```
$FIL␣OLDFIL␣SRC
-5,1Ø


BCL card or $
```

6.16 $FOR

FUNCTION: The $FOR card calls FORTRAN IV (F4M or FPF4M) Compiler and the Line Editor.

FORMAT:
```
$FOR;options␣file-name-1:device-1(uic-1)
          ;file-name-2:device-2(uic-2)
                    ;:device-3(uic-3)

Line Edit Card/s, or
Source Card/s, or
BCL card
```

ARGUMENTS:

options - (Default - B, L) A six character option field. The FORTRAN options follow:
- B = Binary output
- L = Source listing
- O = Object listing
- S = Symbol map

file-name-1 = (Default - FILTMP) This ten character field specifies the SOURCE file-name. The source extension is assumed to be SRC.
device-1(uic-1) - Standard format.

file-name-2 = (Default - file-name-1 or FILTMP) A ten character field used to specify the name of the original source file to be edited. This name is required to edit an existing file, create a new "edited' file and compile the newly created file.
device-2(uic-2) = Standard format.

device-3 = (Default - Line Printer) This field is used to specify the desired output listing device (DAT = -12).

device-4 = (Default - System Disk) This field is used to specify the desired binary output device.

uic-4 = (Default - Current UIC) This field is used if device-4 specifies disk as the output device.

TECHNICAL NOTES:

1. If the line editor is to be used, the following rules should be followed:
   a. A disk must be used for input/output, or the following error message will be printed:

'NON-DISK I/O DEVICE'

and an exit to the next control card will occur.

b. The same device/unit should be assigned to device-1 and device-2; if not, the following message will be output and an exit to the next control card will occur:

'DIFFERENT DEVICES ASSIGNED TO .DAT-14 & -15'

c. The same uic must be used for UIC-1 and UIC-2, otherwise the following error message will be printed, and an exit to the next control card will occur:

'DIFFERENT UIC ASSIGNED TO .DAT-14 & -15'

2. The $FOR command can be used to;

   a. Compile a file already existing. The $FOR card must specify file-name-1, and the next control card should follow the $FOR command.

   b. Create a new file and compile it. The $FOR card must specify file-name-1, and the "source" deck should follow this command.

   c. Compile without creating a file. No file-name should be specified, and the "source" deck should follow the $FOR command.

   d. Edit an already existing file and compile it. The file-name-1 must be specified, and the line-edit commands should follow the $FOR command.

   e. Edit an already existing file, create a new "edited" file and compile the newly created file. The original source file-name is specified in file-name-2. The new source file-name is specified in file-name-1, and the $FOR command should be followed by the line-edit commands.

   Depending on which function (a, b, c, d, or e above) the $FOR command specifies, the command can be followed by:

   A FORTRAN Source Deck,
   Line-Edit Commands,
   The next Control Card.

3. A source deck makes up a FORTRAN-IV source program. This program is a sequence of symbolic statements translated by the FORTRAN compiler into an object program module. The basic unit of expression, in a FORTRAN source program, is a statement.

   A FORTRAN statement consists of a command portion which characterizes its function and may in addition require arguments. The form in which statements and comments are entered is governed by a 72-character line on a Standard FORTRAN coding form. The columns on the coding sheet correspond to card columns, and one character per column.

4. When the user program has been successfully compiled, the output is a relocatable "object module" (BIN extension). This module can be further processed by using the $MAP or $OVL and $LNK commands. These commands will produce a "load module" (absolute file) with a file extension of XCT. The XCT file is actually two files, one with an XCT extension and the other with an XCU extension. For further information see the CHAIN XVM/EXECUTE XVM Utility Manual. The load module must be executed using the $XCT command to run the user's program.

5. For further FORTRAN information refer to:

   FORTRAN IV XVM Language Manual, and FORTRAN IV, XVM Operating Environment Manual.

EXAMPLES:

A. 1. $FOR␣FILE␣SRC:DT3
   Next BCL Command
   2. $FOR;OB␣FILE
   Next BCL Command

B. 1. $FOR␣FILE
   Source Deck
   2. $FOR;OB␣FILE:DT5
   Source Deck

C. 1. $FOR
   Source Deck
   2. $FOR;OS
   Source Deck

D. 1. $FOR␣FILE
   Line-Edit Commands
   2. $FOR;OB␣FILE:DT5
   Line-Edit Commands

E. 1. $FOR␣FILEN:DK(MGC);FILEO:DK(MGC)
   Line Edit Commands
   2. $FOR␣FILE:DK(MGD);FILE1:DK(MGC);:DTØ
   Line Edit Commands

6.17 $JOB

FUNCTION: The $JOB card marks the beginning of a user's job, and establishes general control parameters.

FORMAT: $JOB␣id;time-limit;uic;QDUMP;{ON / OFF}

ARGUMENTS:

id = Identification field consisting of one to nine alphanumeric characters. This field is used to identify the job and all messages pertaining to it, including operator communication.

time-limit = (Default "1") - This ten character field specifies the maximum running time in minutes for the current job. If the time expires, the JOB is aborted with the message;

TIME ESTIMATE EXCEEDED

If the limit specified is "Ø" no limit exists, and the program can run indeterminantly.

uic = (Default "SCR") A three letter field for the current User Identification Code (UIC) which may be supplied by the user for each job. A UIC is part of the file protection scheme.

QDUMP = If an unconditional automatic core dump is desired on an IOPS error, this field must contain "QDUMP". A listing of the Control Q area is output on the Line Printer from the System device. The Control Q area size is established at System Generation (SGEN) time.

ON/OFF = (Default = OFF) If the KEEP function is desired, "ON" should be specified. This command instructs the System to retain or reset device/file-directory assignments after completion of the current program.

TECHNICAL NOTES:

1. All jobs must begin with a $JOB card.
2. For use in the Accounting File, the ID parameter is limited to nine characters.
3. $JOB will always cause the SCR uic to be cleared.

3. When command arguments are used, all intervening punctuation must be given whether or not the respective argument is utilized.

EXAMPLES:

```
$JOB
$JOB␣TEST-1SRC;;EEM
$JOB␣;2;IDU
$JOB␣;;EEM
$JOB␣ID;10;UIC
$JOB␣ID;1Ø;UIC:QDUMP;ON
$JOB␣ID;1Ø;EHM
```

6.18 $KEP

FUNCTION: The $KEP card instructs the BOSS Monitor to retain DAT/UFDT slot assignments as revised via $ASG.

FORMAT: $KEP␣{ON / OFF}

ARGUMENTS: ON = $KEP␣ON retains the device assignments.

OFF = $KEP␣OFF resets the device assignments.

TECHNICAL NOTES:

1. The OFF command permits $ASG device assignments to revert to those established by System Generation.
2. This command should be used primarily with the user's object module since BOSS automatically overrides $ASG, and $KEP when loading System programs requiring standard device assignments.

6.19 $LCM

FUNCTION: The $LCM card specifies a library file update function. It is a supplementary command to the $LIB command.

FORMAT: $LCM␣function;program-name-1;program-name-2

ARGUMENTS: function - (Default - CLOSE with a Carriage Return) This field specifies the UPDATE function to be performed. Either the abbreviated form of the function or the entire function name can be specified. The UPDATE functions are:

C = CLOSE name-2 This function is the normal terminator for any UPDATE operation. When used with the U option, it causes all of the remaining programs in the primary input file to be copied onto the output file starting at the current position of the file. When used with the N and L options, the C function permits normal output file closing operations and initiates the output listing file. If a file name is not specified with this command the output file is given the name originally specified in the last $LIB file-name-1 command.

D = DELETE name-1, or name-1 through name-2 This command causes the deletion from the specified file of a named routine or a range of routines. This is specified by name-1 of the first routine and name-2 as the last routine of the range. This function may be used only when the $LIB "U" option has been specified.

E = END This function causes the output file to be positioned at the end of the final routine of the original file. All remaining programs in the original file, beginning at its current position, are copied into the output file.

F = FREE name-1 This function frees or extracts a named file from the library file named in the "G" option of the $LIB command. The extracted program is transferred to the output device as a file of the same name.

I = INSERT name-1, or name-1 after -name-2. This function instructs UPDATE to insert programs contained on the secondary input (device-3) into the original file (at any point) as it

is copied onto the output file (device-2). On completion of an insert operation, the output file is positioned at a point immediately after the inserted program. If one name is used in the command, the named program is inserted into the file at the current position. If two names are specified, the name-1 program is inserted into the file immediately after the program specified by name-2.

R = REPLACE name-1, or name-1 with name-2 The R function causes programs in the original file to be replaced by new programs on the secondary input device (device-3). All programs in the original file, up to the element or routine to be replaced, are copied onto the output device (device-2). The new program(s) is then copied onto the output file from a secondary input. Upon completion, the output file is positioned immediately after the replaced routine. If one program name is specified, UPDATE replaces the name-1 program with a new program having the same name. If two programs are specified, name-1 is replaced by name-2.

program-name-1 = This ten character field specifies the name of a program to be acted on. It is used as specified by a particular $LCM function and corresponds to the $LIB command string.

program-name-2 = (Default - program-name-1) This name is used as specified by a particular $LCM function and corresponds to the $LIB command string. When a CLOSE function is to be performed with a file-name, the desired file-name must be in the program-name-2 field.

TECHNICAL NOTES:

1. The $LCM command specifies a library file update function. It is a supplementary command to the $LIB command. The $LCM command must follow either a $LIB card or another $LCM card.
2. Only one update function can be specified on the $LCM card. Refer to the following Table 6-3 "Legal Option/Command Combinations".
3. The END function is a convenient method of positioning the file to be modified. This will permit new routines from the secondary input to be appended to the file via the INSERT function.
4. Any number of $LCM cards can follow a $LIB card.
5, For more information, refer to the UPDATE XVM Utility Manual.

TABLE 6-3

LEGAL OPTION/COMMAND COMBINATIONS

$LIB Options*	$LCM Command Functions						
	INSERT	DELETE	REPLACE	FREE	END	CLOSE	KILL
U[L][S]	X	X	X	-	X	X	X
N[L][S]	X	-	-	-	-	X	X
G[L][S]	-	-	-	X	X	X	X
L	-	-	-	-	X	X	X

*The S option cannot be used alone.

Where: G = Get via FREE
L = List Element & Size
N = New Library File
S = Remove Symbol Table
U = Update via Deletion/Replacement

EXAMPLES: $LCM␣R;AUX1Ø

$LCM␣C;;.LIBR1

6.20 $LIB

FUNCTION: The $LIB card is used to call the Utility program UPDATE. It is used in conjunction with the $LCM card to specify the update activity.

FORMAT:
```
$LIB;options␣file-name-1:device-1(uic-1)
                          ;:device-2(uic-2)
                          ;:device-3(uic-3)
                          ;:device-4(uic-4)
```

ARGUMENTS:

options - (Default - LSU) A six character option field. The UPDATE options follow:

- G = Get via FREE This option primes UPDATE to accept the FREE manipulative command which allows individual programs to be extracted from libraries as separate files.
- L = List Element & Size This option lists the contents of the new or updated library output to device-2 (DAT-15) on the device assigned by device-4 (DAT-12). If no other options are selected, a listing of the named library residing on the primary input, device-1 (DAT-14) can be obtained.
- N = New Library File This option permits a new library file to be created from programs residing on device-3 (DAT -10). The new library is output to device-2 (DAT -15).
- S = Remove Symbol Table The S option causes UPDATE to remove the local symbol table from each program placed in the output file (device-2).
- U = Update via Deletion/Replacement This option permits the updating of an existing library file by the deletion, insertion and replacement of the programs within it (see $LCM command). The file to be updated resides on the device assigned to device-1 (DAT -14); programs to be inserted must be on device-3 (DAT -1Ø); and the update file is output to device-2 (DAT -15).

file-name-1 = (Default - .LIBR BIN) This ten character field specifies the name of the library file to be updated.

device-1 = (default - System Device) This field is used to specify the device on which file-name-1 is present (DAT -14).

uic-1 = (Default - Current UIC) This field is used to specify the UIC when device-1 is disk.

device-2 = (Default - device-1 or System Device) This field is used to specify the library output device (DAT -15).

uic-2 = (Default - uic-1 or Current UIC) This field is used to specify the UIC when device-2 is disk.

device-3 = (Default - System Device) This field is used to specify the secondary input device (DAT -1Ø).

uic-3 = (Default - Current UIC) This field is used to specify the UIC when device-3 is disk.

device-4 = (Default - Line Printer) This field is used to specify the listing device (DAT -12).

uic-4 = (Default - Current UIC) This field specifies the UIC when device-4 is disk.

TECHNICAL NOTES:

1. Every $LIB command must be followed by at least one $LCM command.
2. The S option causes UPDATE to remove the local symbol table from each program placed in the output file. This process shortens the length of a file by as much as 40% thus conserving storage space and reducing library search time during loading. Non-library programs will occupy less core since there will be no symbol table to load.
3. The $LIB command indicates the UPDATE operation and the name of the file to be modified or created. To manipulate the contents of named file, the user must specify action commands via the $LCM command. Legal option/command combinations are given in Table 6-3. For more information refer to the UPDATE Utility Manual.

6.21 $LNK

FUNCTION: The $LNK card is used to describe a LINK of the overlay structure created by CHAIN.

FORMAT: $LNK string.

ARGUMENT: string = (Default - ALT MODE) LINK-name = definition
The overlay structure is described in terms of LINK names on the $LNK card followed by the definition. Each LINK definition requires one card.

TECHNICAL NOTES:

1. The $LNK command is used to describe a LINK of the overlay structure being created by CHAIN. Each LINK command describes one LINK or overlay definition.
2. Each LINK definition requires a $LNK command, so that normally a sequence of $LNK commands would be used. The $LNK commands must follow the $OVL command, or another $LNK command.
3. Only one LINK may be specified per card. The default LINK definition is an ALT MODE; therefore, the last $LNK command should have no arguments.
4. When a LINK is to consist of only one external component, the name of the file containing the external component may be used as a LINK name. When a LINK is to consist of more than one external component, the LINK must be named and defined.
5. Each LINK definition requires one card of command input consisting of the LINK's name followed by an equal sign (=) followed by the LINK definition. The last definition card should be followed by a $LNK only card. A LINK definition is a list of the names of files which contain the relocatable binary units that comprise the LINK components. The individual file names listed are separated by commas (,); the two types of LINK components which may be used (external and internal) are separated within the definition by a slash (/). All external LINK component names must be listed before (to the left of) the slash separator; all internal LINK components must be listed after (to the right of) the slash. External LINK components are accepted only from files with names which match the external component name (i.e., GLOBAL symbol definition).

Example: $LNK␣ABC=SUB1,SUB2/SUB3,SUB4

In the above example, SUB1 and SUB2 are external components of LINK ABC, and SUB3 and SUB4 are internal components of LINK ABC.

6. Rules for defining a LINK:

   1) A LINK may not be a component of another LINK.

   2) The names of the components of a LINK may not be used as LINK names.[1]

   3) A file name used in the resident code description cannot be used in a LINK definition.

   4) A file name preceding a slash may be used only once.

   5) A file name following a slash may be used in other LINK definitions (following a slash).

7. OVERLAY STRUCTURE DESCRIPTION - An overlay structure is described using the names of defined LINKS, or the names of files containing LINK components and the operators colon (:) and comma (,), under the following set of rules:

   1) A line is an independent statement processed from left to right.

   2) A colon is read "is overlayed by".[2]

   3) A comma is read "and".

      Example:

      SUB1:SUB2

      SUB2:SUB3,SUB4

      is interpreted as -- SUB1 is overlayed by (uses the same core as) SUB2, SUB2 is overlayed by SUB3 and SUB4, but SUB3 and SUB4 do not overlay each other.

   4) A colon operator may not be used in a line after a comma has been used. This restriction prevents the following ambiguity:

      SUB2:SUB3,SUB4:SUB5

      The above line is rejected by CHAIN because it is not clear whether SUB5 overlays SUB3 or SUB4 or both. All four of the following examples are acceptable:

SUB2:SUB3,SUB4 SUB4:SUB5	SUB5 uses the same core as SUB4 but not the same core as SUB3.
SUB2:SUB3,SUB4 SUB3:SUB5	SUB5 uses the same core as SUB3 but not the same core as SUB4.

---

[1]When a LINK consists of only one component, the component's file name may be used as the LINK name in the "overlay structure description", but not in a LINK definition; i.e., it is not necessary to define a single component LINK but, if defined, the LINK name cannot be the component name.

[2]A loading order is not implied; just core mapping.

SUB2:SUB5:SUB3,SUB4	SUB5 uses the same core as SUB3 and SUB4. SUB3 and SUB4 are loaded individually (if non-resident) as called.
LINK=SUB3,SUB4 SUB2:LINK:SUB5	SUB5 uses the same core as SUB3. and SUB4. Both SUB3 and SUB4 are loaded (if non-resident) whenever either is called.

5) A LINK name may appear only once preceding a colon and only once following another colon.

6) If a LINK name is used twice, it must be used following a colon before being used preceding another colon.

7) Several LINKS overlaying each other may be defined in one statement.

SUB1:SUB2:SUB3,SUB4[1]

NOTE

This is a short method of defining the same overlay structure as in the first example, under rule 3.

Rules 5 and 6, although they may appear restrictive, do not limit the user's description of an overlay structure, but do prevent multiple description of the position of a LINK in an overlay structure. A LINK may be both overlayed and overlaying, and it may not be possible or convenient to describe both conditions by using the LINK name only once, as follows:

SUB1:SUB2:SUB3	SUB2 is overlaying SUB1 and is overlaying by SUB3.

Therefore, when a LINK is both overlaying and overlayed, its LINK name may be used twice, but the LINK(s) overlayed by it must be described before the LINK(s) by which it is overlayed.

Example:	SUB2:SUB3,SUB4	SUB3 overlays SUB2
	SUB3:SUB5	SUB3 is overlayed by SUB5

NOTE

The description of an overlay structure only defines a desired core mapping; i.e., stating that SUB1 is overlayed by SUB2 means that both are to be relocated to the same core and cannot co-reside, but does not imply that SUB1 must be called before SUB2. There is no imposed order in which routines must be called.

[1]A loading order is not implied; just core mapping.

nor is there restriction of the routines callable by any routine.

8. For further information, refer to the CHAIN XVM/EXECUTE XVM Utility Manual.

Examples:

```
$OVL␣XCTFIL;PGR;MAIN
$LNK␣LINK1=LNK1/DUMMY
$LNK␣LINK2=LNK2/DUMMY
$LNK␣LINK1:LINK2:LINK3
$LNK␣LNK4:LNK5
$LNK
```

6.22 $LOG

FUNCTION: The $LOG card specifies a new current User Identification Code (UIC).

FORMAT: $LOG␣uic

ARGUMENT: uic = (Default - SCR) This three character field must contain a three letter code for the current UIC. The UIC is part of the file protection scheme.

TECHNICAL NOTES:

1. The $LOG command specifies a new current UIC. This function is also performed by the $JOB command; but the $LOG command allows the user to change the current UIC during a job.
2. This command permits the user to enter his user UIC into the system in order to do directoried disk I/O. The Monitor sets the slots of the UFDT to the three-character code after LOG. All input/output operations to the disk are directed to the UFD associated with the UIC last entered by this command, unless a program has subsequently done a .USER, or a $ASG command is used.
3. Each LOG command issued enters a new UIC into the system and automatically deletes the one entered previously (see also $DLG). A UIC must consist of exactly three alphanumeric characters in any combination except "@@@", "???", "PAG", "BNK", "SYS", "PER", "IOS", and "CTP".

EXAMPLES:

```
$LOG␣FIL
$LOG␣EEM
```

6.23 $LST

FUNCTION: The $LST card is used to output the contents of an ASCII file on the line printer.

FORMAT: $LST␣file-name:device(uic)

ARGUMENTS:

file-name = (Default - FILTMP SRC) This ten character field specifies the source name and extension of the file to be listed.

device = (Default - System Device) This four character field specifies the device type, and handler version of the device on which the file to be listed is present.

uic = (Default - Current UIC) This three character field contains the name of the UFD the file is on. It is required when the file device is a disk.

EXAMPLE: $LST␣OPRTFL␣SRC:DPA1(EEM)

6.24 $MAC11

FUNCTION: The $MAC11 card calls the PDP-11 MAC11 assembler and the line editor.

FORMAT: $MAC11; options filename-1: Device-1 (uic-1); filename-2: Device-2 (uic-2); Device-3.

Line Edit Cards, or Source Cards, or next BCL command.

ARGUMENTS: Options - (Default -B,L) A six character option. The options with their associated actions are given below:

B = Generate binary paper tape
L = Generate a listing on the requested output device

filename-1 = Source file (edit output/new filename)

device-1 (uic-1) = Standard format

filename-2 = Edit input (old filename)

device-2 (uic-2) = Standard format

device-3 = Listing device - must not be directoried

TECHNICAL NOTES:

1. The $MAC11 command provides the user with the sumbolic programming capabilities of the MAC11 assembler which includes a macro instruction generator.

2. If the line editor is to be used, the following rules should be followed:

   a. DECdisk, disk pack, or disk cartridge must be used for input/output, or the following error message will be printed:

      'NON-DISK I/O DEVICE'

      and an exit to the next control card will occur.

   b. The same device/unit should be assigned to device-1 and device-2. If not, the following message will be printed and an exit to the next control card will occur.

      'DIFFERENT DEVICES ASSIGNED TO .DAT -14 and -15'

c. The same .UFDT must be used for UIC-1 and UIC-2, otherwise, the following error message will be printed and an exit to the next control card will occur:

'DIFFERENT UIC ASSIGNED TO .DAT -14 and -15'

3. The $MAC11 command can be used to:

a. Assemble a file already existing. The $MAC11 card must specify filename-1, and the next control card should follow the $MAC11 command.

b. Create a new file and assemble it. The $MAC11 card must specify filename-1, and the "source" deck should follow this command.

c. Assemble without creating a file. No filename should be specified and the "source" deck should follow the $MAC11 command.

d. Edit an already existing file and assemble it. The filename-1 must be specified and line-edit commands should follow the $MAC11 command.

e. Edit an already existing file, create a new "edited" file and assemble the newly created file. The original source filename is specified in filename-2. The new source filename is specified in filename-1, and the $MAC11 command should be followed by the line-edit commands.

Depending upon which function (see options above), the $MAC11 command specifies, the command can be followed by:

A MAC11 source deck,
Line-Edit commands,
The next Control Card

6.25 $MAP

FUNCTION: The $MAP card is used to create an executable absolute file without overlays (to CHAIN resident code only).

FORMAT: $MAP␣file-name-1;option;file-name-2

ARGUMENTS:

file-name-1 = (Default - TMPXCT) This field is used to specify the executable file name of the file to be created by CHAIN. It is a 1 to 6 character file-name.

options = This field specifies CHAIN options. See the CHAIN XVM/EXECUTE XVM Utility Manual for the options and their description.

file-name-2 = (Default - FILTMP) This field consists of a file-names separated by commas. This list is passed to CHAIN as the response to CHAIN's DEFINE RESIDENT CODE" inquiry.

TECHNICAL NOTES:

1. The $MAP command is used to CHAIN resident code only. It is a "Direct Substitution" command used to create an executable absolute file without overlays.

2. The $MAP, and $OVL commands produce a "load module" (absolute file) with a file extension of XCT. This module must be executed using the $XCT command to run the user's object program.

3. CHAIN requests a list of resident routines. All routines listed and any library routines they may call remain in core throughout a run. The first resident routine is the program to which initial control of the execution of the system is given. All other routines in the system are either subroutines or co-routines. Resident subroutines may be called by any routine; therefore, the names assigned to resident routines must be unique throughout the system.

4. See the CHAIN XVM/EXECUTE XVM Utility Manual for detailed information.

   Example:

   $MAP␣EXECUT;;MAIN,SUB1,SUB2,SUB3

5. More than one option can be specified. Options should be separated by commas.

   Example:

   $MAP␣EXECUT;SZ,PGR;MAIN,SUB1,SUB2

6.26 $MIC

FUNCTION: The $MIC card is used to LOGIN the Monitor Identification Code (MIC). This is a privileged command used to lift all file protection.

FORMAT: $MIC␣code

ARGUMENT: code = (Designated via SGEN)

TECHNICAL NOTES:

1. The $MIC is a three letter code used to lift all file protection and to allow the system maintainer to access all disk files. If the correct code is used with the $MIC command, all files can be accessed. When an incorrect code is used, a $DLG (LOGOUT) function is performed and a job abort is entered into the Accounting File.
2. This command provides the user with unrestricted supervisory access to all files contained in the various directories on the disk. The MIC of each system, as initially supplied to the user, is "SYS". As with $LOG, a $MIC command is deleted from the system using the $DLG or a new LOG command.
3. The MIC user is usually the system owner. He can easily change the MIC code via system generation to maintain code integrity.
4. The $MIC command is not echoed.

6.27 $MNT

FUNCTION: The $MNT card instructs the operator to mount a specified Tape on a specified Unit and to switch the WRITE function to LOCK or ENABLE.

FORMAT: $MNT; {D / M} ␣tape-number;drive-number; {LOCK / ENABLE}

ARGUMENTS:

D = (Default -DECtape) The one character option field specifies DECtape when it contains "D".

M = The one character option field specifies Magtape when it contains "M".

tape-number = This $\leq 10_{10}$ character message field identifies the "reel" to be mounted.

drive-number = This message field identifies the physical drive/unit (Ø-8) to mount the tape on.

LOCK = (Default - LOCK) Tells the operator to WRITE LOCK the physical device.

ENABLE = Tells the operator to WRITE ENABLE the physical device.

TECHNICAL NOTE:

1. The $MNT command instructs the operator to mount a tape reel. At the end of the message, the computer waits for the operator to mount the tape and then to type CTRL P to resume operation. The run-time-clock is suspended during this period, not the time-of-day clock.

EXAMPLES:

```
$MNT;M␣CMS4Ø;1;ENABLE
$MNT;D␣TID-REV-4;3;
```

6.28 $MSG

FUNCTION: The $MSG card permits messages to the operator.

FORMAT: $MSG␣message

ARGUMENT: message = Any text (≤ 72 characters/card).

TECHNICAL NOTES:

1. The $MSG command provides a way for a user of BOSS to convey messages to the operator.
2. Message information beyond the usable characters/card should be added via additional $MSG cards.
3. User messages are output to the teleprinter.
4. If a message and "wait" operation is desired the user should use a $MSW card.
5. The syntax for this command is determined by Direct Substitution File format.

6.29 $MSW

FUNCTION: The $MSW card permits messages to the operator to be followed by a wait condition.

FORMAT: $MSW␣message
(CTRL P at console teleprinter)

ARGUMENT: message = Any text (≤ 72 characters/card).

(↑P) = The computer waits for a CTRL P from the console teleprinter to resume operation.

TECHNICAL NOTES:

1. The $MSW command is like the $MSG command; except, at the end of the message, the computer waits for a CTRL P from the console teleprinter to resume operation.
2. User messages are output to the teleprinter.
3. Message information beyond the usable characters/card should be added via original $MSG cards concluding with a $MSW card.
4. The syntax for this command is determined by Direct Substitution File format.
5. The run-time clock is suspended from the time the $MSW command is issued until the operator answers with a CTRL P.

6.30 $NDR

FUNCTION: The $NDR card is used to refresh a file directory.

FORMAT: $NDR␣device(uic)

ARGUMENTS: device = (Default - System Device) This four character field denotes the device type, handler version, and unit number. Thd default for the handler = A, and the unit number = Ø for MT or DP or RK or DT or DK.

uic = (Default - Current UIC) This three character field denotes the User Identification Code (UIC) to associate with the corresponding device.

TECHNICAL NOTES:

1. The $NDR command causes the directory on the unit specified to be refreshed, or renewed; consequently all files on that unit are lost. The $NDR command and the $DIR command follow the same syntax. See PIP commands: $N, $D, and $R for additional directory utilities.

2. Care must be taken to ensure that needed files are not lost.

3. UFD's set up or initialized using this command are automatically assigned a default protection code which does not override, but enables the protect function of individual files (see $PCD command). In a protected directory, only the user logged in under the UIC of the UFD (or the MIC) may write into the UFD.

EXAMPLE:

```
$NDR␣DK(EEM)
$NDR␣MTA5
```

6.31 $OVL

FUNCTION: The $OVL card is used to create an executable absolute file with overlays. A $LNK command should follow the $OVL command.

FORMAT: $OVL␣file-name-1; option:file-name-2

ARGUMENTS:

file-name-1 = (Default - TMPXCT) This field is used to specify the executable file name of the file to be created by CHAIN. It is a 1 to 6 character file-name.

options = (Default - SZ) This field specifies CHAIN options. See the CHAIN XVM/EXECUTE XVM Utility Manual for the options and their descriptions.

file-name-2 = (Default - FILTMP) This field consists of a list of file-names separated by commas. This list is passed to CHAIN as the response to CHAIN's "DEFINE RESIDENT CODE" inquiry.

TECHNICAL NOTES:

1. The $OVL command is a "Direct Substitution" command used to create an executable absolute file with overlays.

2. When a user's program has been successfully assembled or compiled, the output is a relocatable "object module" (BIN extension). This module can be further processed by using the $OVL and $LNK commands. These commands will produce a "load module" (absolute file) with a file extension of XCT. The XCT file is actually two files, one with an XCT extension and the other with an XCU extension. The load module must be executed using the $XCT command to run the user's object program.

3. With $OVL, the user generates (via CHAIN) a system of overlays a resident main program which may include resident subprograms, a resident blank COMMON storage area, and a set of subroutines which overlay each other at the user's request. Subroutines are organized via the $LNK command into units called LINKS which may overlay each other. Several LINKS may overlay as larger LINK without overlaying each other. A LINK is called and it remains resident until overlaid. A LINK's core image is not recorded or "swapped out" when it is overlaid. The same image is brought into core each time a LINK is loaded.

4. See the CHAIN XVM/EXECUTE XVM Utility Manual for detailed information.

EXAMPLE: See $MAP comments.

6.32 $PAG

FUNCTION: The $PAG card selects PAGE Mode system operation (4K)

FORMAT: $PAG

TECHNICAL NOTES:

1. The default addressing mode is established at System Generation time by the user.
2. In PAGE Mode, relocatable system programs and user programs (including device handlers) are loaded within 4K memory pages, and Index Register usage is permitted. A $PAG card sets the system to operate in PAGE Mode. Conversely, a $BNK card sets the system to operate in BANK Mode.

6.33 $PRT

FUNCTION: The $PRT card is used to alter the default value of the file protection code.

FORMAT: $PRT␣ {1 / 2 / 3}

ARGUMENTS: File Protection Codes:

Code	Function	Description
1	= Unprotected	- Unprotected files can be either read from or written into by any user. "Writing into" a file implies the use of a random-access MACRO .RTRAN. The file can be completely altered, but it cannot be deleted from a protected UFD, and it's length may not be changed.
2	= Write Protected	- Write protected files can be read by any user.
3	= Read/Write Protected	- Read/Write protected files cannot be read or written.

TECHNICAL NOTES:

1. The various levels of file protection provided are effective only when the UFD in which the file resides is not associated with the UIC currently in effect, and the UFD itself is protected. Three file protection codes are provided, as described above.
2. The $PRT command is used to alter the default value of the file protection code, set at System Generation or via the $N command. The default file protection codes set by this command remain in effect until another $PRT command is given or until the user issues a $DLG (which resets the protection code to the system's default value). The default value of file protection code for the system, as initially supplied, is two (2).

EXAMPLE: $PRT␣2

6.34 $QDP

FUNCTION: The $QDP card initiates a core dump (control Q area) when a terminal error (IOPS) is detected. The occurrence of the condition is based on the placement of the card within the deck.

FORMAT: $QDP

TECHNICAL NOTES:

1. The $QDP command instructs the monitor to automatically give a core dump listing when a terminal IOPS error is detected. This command interrupts the program currently running, enters the contents of core into the QAREA on the System device, then lists this file on the line printer.
2. The control Q area size is established at System Generation. The Control Q function dumps the minimum of the ↑Q area size and the current system memory size.

6.35 $RUN

FUNCTION: The $RUN card takes a file that is located on the disk, does a Fortran compile of it, and begins to execute the program. This is an efficient combination of the $FOR, $MAP and $XCT cards.

FORMAT: $RUN file-name-1 = Device-1(uic-1)

ARGUMENTS:

File-name-1 = This is a Fortran source coded program located on Device-1.

Device-1 = (Default-System Device) This field is used to specify the desired source file input device.

uic-1 = (Default-current uic) This field is used if device-3 specifies disk as the input device.

TECHNICAL NOTES:

1. Once a program has been compiled and runs, the only other command needed to rerun the same program is $XCT.

First Run	Next Run	
$JOB TEST;BLR	$JOB TEXT;BLR	THIS IS MUCH
$RUN TEST	$XCT TEST	FASTER FOR THE SECOND RUN
DATA	DATA	
$END	$END	

2. The system device is utilized for all intermediate files. The resulting binary and XCT/XCU files are resident there.

6.36 $XCT

FUNCTION: The $XCT card is used to execute an executable absolute file previously created by $MAP or $OVL commands.

FORMAT: $XCT file-name:device(uic)

ARGUMENTS:

file-name = (Default - TMPXCT) This ten character field specifies the execute file name of the user's load module.

device = (default - System Device) This four character field denotes the device type, handler version, and unit number of the device the execute file is on.

uic = (Default - Current UIC) This three character field denotes the name of the UFD the file is on. It is only required when the file device is a disk.

TECHNICAL NOTE:

The $MAP and $OVL commands produce a "load module" (absolute file) with a file extension of XCT. This module must be executed using the $XCT command to run the user's object program.

This command should not be confused with the BOSS End-of-File ($) function. The $ only card is used to tell BOSS to run the Run-Time-File as it stands. The RTF is composed of BCL cards.

6.37 $XVM

FUNCTION: The $XVM card enables and disables XVM mode.

FORMAT: $XVM [ON / OFF]

ARGUMENTS: ON = Enables XVM mode for user programs
OFF = Disables XVM mode for user programs.

TECHNICAL NOTE:

1. This card should be used only on a system with the XM-15 hardware option.

CHAPTER 7
OPERATING PROCEDURES

7.1 INTRODUCTION

This chapter provides general operating procedures and considerations to assist users in operating in the BOSS XVM environment. Procedures for system startup and system generation are not included here. The discussions and examples in this section assume a properly configured and running system. The reader should be familiar with BOSS command cards, the keyboard commands, and the different types of I/O device handlers and their versions.

Specific operating procedures for the system programs (i.e., command strings, options, functions, error message, etc.) are provided in the various language and utility program manuals provided for XVM/DOS.

In general, keyboard operations at initial startup consists of issuing commands to set up system operation prior to calling system and user programs via BOSS card commands.

7.2 LOAD BOSS XVM

To load BOSS, load the XVM/DOS Monitor first, via the bootstrap. Place the proper bootstrap (DECdisk or Disk Cartridge or Disk Pack) in the Paper Tape Reader. Set Address to X7637 (X = highest bank 5 or 7). Press STOP and RESET, then READIN. To restart the bootstrap, set ADDRESS to X7646, and press STOP and RESET, then START. The Monitor responds with XVM/DOS Vnxnnn and requests a date (mm[/]dd[/]yy). After the Monitor replies with a "$", the user should enter "TIME␣hhmm ", then "BOSS " at the keyboard to call the Batch Monitor. If spooling is desired on a UC15 system, it should be enabled prior to calling "BOSS ". This is done as follows:

```
XVM/DOS Vnxnnn
$SPOOL                          (Load and execute the SPOOL program)
SPOOL XVM Vnxnnn
     RK UNIT # (0) 1            (Specify the unit to spool on.)
     BEGIN ? (Y) Y              (Initiate spooling)
     SPOOLING ENABLED
XVM/DOS Vnxnnn
```

Spooling is controlled only via XVM/DOS to protect users from harming each other.

Hereafter, all occurrences of card reader/line printer will refer to the actual device on non-UC15 systems and to the virtual device on spooled UC15 systems.

7.3 BOSS XVM OPERATION

On initialization, the batch monitor will be given control and immediately start processing jobs from the card reader. The card reader should be loaded with cards and the line printer with paper.

If no cards are in the card reader, the message:

IOPS4 CD NOT READY

will be output on the console teleprinter. The computer will then enter a "wait" state. To resume operation, the card reader must be loaded and "readied". If the card reader is a DIGITAL XVM(XVM) device, CTRL R must be pressed on the console teleprinter to resume operation. If it is a UNIBUS device, operation will resume automatically, once the device is readied, without requiring the keyboard entry of CTRL R. This distinction holds as well between XVM and UNIBUS line printers.

From this point on, system operation is automatic. The operator must resupply the card reader with jobs and make sure the line printer has paper. When an IOPS4 occurs during a user's program execution, BOSS's Time-Out feature[1] is operative. If the operator fails to resume operation within the time limit set via the $JOB command, the current job will be aborted. The following error message will be printed at both the teleprinter and line printer:

TIME ESTIMATE EXCEEDED

An aborted job will cause the System to process the next job in the job string without processing the current job. The aborted job, when corrected, can be reinserted at the end of the job string for processing.

For each $JOB card, a record of each job processed will be output to the teleprinter in the following format:

LOG JOB job i.d. BEGIN TIME
(Marks beginning of job, no operation action required)

When the card reader reads a card with an illegal punch, operation is suspended, and the message will be output on the console teleprinter. This error will occur on any code not specific to DEC026 or DEC029 card code. The operator can repunch the card in question and insert it into the card stream, or discard it. By typing CTRL R on the

[1] BOSS allows a run time control specification on the $JOB card. Should a job exceed the run time indicated (default is one minute), BOSS will skip through to the next $JOB card.

control teleprinter, if this is a XVM card reader, he will cause the operation to resume.

Jobs can be stacked in the card reader, and will be started and cleared automatically. Each job will be accompanied by line printer output, so the operator can monitor job processing.

If a card command is in error (does not correspond to a Procedure File), BOSS assumes that the command is the name of an absolute file (ABS) to be loaded. It then tries to load that file. If it doesn't exist, BOSS outputs the message:

ILLEGAL COMMAND

7.4 OPERATOR MESSAGES

Messages to the operator can be output on the console teleprinter. These messages are of two general categories:

1. Log messages (implementation via $MSG, $MSW, $JOB and $MNT commands)
2. Monitor error messages system generated

7.4.1 LOG Messages

LOG messages start with the word

LOG...msg...

Its purpose is to instruct the operator and to request special action (i.e., tape mounting message via $MNT command, etc.).

A variation of the LOG message (via $MSG) is the LOGW message. The LOGW message (via $MSW) begins with the word:

LOG...msg...

LOGW messages require operator action. At the end of the message, operation will be suspended. The teleprinter bell will be rung to signal the operator and ↑P will be output on the control teleprinter. The operator translates into action the instructions in the LOGW message, and at completion resumes operation by keying in CTRL P on the control teleprinter. From the time ↑P is output on the control teleprinter to the time the operator types CTRL P, job timing is suspended. The real-time clock continues to run. The system will keep track of the time of the day, but the time spent waiting for the operator will not be charged to the current job.

Two standard LOG and LOGW messages are as follows:

1. LOG JOB job i.d. BEGIN TIME
   Marks the beginning of a job; no operator action required.
2. LOGW MOUNT {D/M}-TAPE tape i.d. ON DRIVE drive #{LOCK/ENABLE}
   Instructs the operator to mount a specified tape (M for Magtape, D for DECtape) on a specific unit. The operator must type CTRL P to resume operation.

Additional standard messages with their particular System/Job responses are shown in Table 7-1.

TABLE 7-1

BOSS ERROR MESSAGES

Line Printer Messages

DIFFERENT ADD FILE[2]
END OF RUN TIME FILE REACHED BY USER[1]
ILLEGAL COMMAND[2]
ILLEGAL COMMAND CARD SEQUENCE[2]
ILLEGAL COMMAND FOR ADD FILE[1]
ILLEGAL .GET BY USER[1]
ILLEGAL PROC FILE[2]
LINE TOO LONG[2]
LOAD ERROR-ILLEGAL .DAT SLOT[1]
LOAD ERROR-INPUT DATA ERROR[1]
LOAD ERROR-MEMORY OVERFLOW[1]
LOAD ERROR-NONEXISTENT SYSTEM PROGRAM[1]
LOAD ERROR-PROGRAM >4K[1]
LOAD ERROR-UNRESOLVED GLOBALS[1]
NON-EXISTENT ADD FILE[2]
OPERATOR ABORT[1]
PROC FILE READ ERROR[2]
RUN TIME FILE TOO LONG[1]

TERMINAL ERROR nn {Standard message / Expanded message}[1]
Standard = ␣CAL address
Expanded = ,CALaddress,DEVu,message,CALfunction, UIC,filename

TERMINAL ERRORS = IOPS ERRORS (see Table 7-2)

TIME ESTIMATE EXCEEDED[1]
USER DID A .PUT (core dump on LP)[1]

---

[1]Go to next JOB (Job Abort)
[2]Go to next Control Card.
Note: Job abort conditions are recorded in Accounting File.

Console Teleprinter Message

MAX NUMBER OF ACCOUNTING FILES REACHED
PLEASE PROCESS AND DELETE THEM

Note: Jobs are not aborted but succeeding Accounting File entries are truncated.

7.4.2 IOPS Error Messages

The IOPS messages always indicate an error condition but no action is required by the operator except for IOPS4 messages. IOPS4 messages indicate that an I/O device is not ready; for example, a DECtape drive is not "write enabled" and should be. Then an IOPS4 message will be output to the operator. At this point, the operator can

write enable the DECtape drive and then type CTRL R to resume operation. The two standard IOPS4 messages for the Card Reader (CD) are as follows:

1. IOPS4 CD NOT READY

   Indicates device status condition (e.g., light check, stacker full, hopper empty, etc.) and operation suspended. CTRL R to resume operation.

2. IOPS4 CD ILLEGAL PUNCH

   Card reader attempted to read card with illegal punch (a non DEC 026 or 029 code). The operator should initiate corrective action and type CTRL R to resume operation.

The IOPS4 LP (Line Printer) message indicates an off-line condition or that the printer is out of paper. The System will remain in a wait status until these conditions are corrected.

IOPS error messages other than IOPS4 are output to the line printer as a TERMINAL ERROR message. These are essentially XVN/DOS IOPS messages, in a batch environment, which abort the current job. The message format is as follows:

TERMINAL ERROR__nn__{Standard message / Expanded message}

Standard = ␣ CALadr

Expanded = ,CALadr,DEVu,CALfcn,UIC,filename

TERMINAL ERRORS = IOPS ERRORS

where nn = error code number (see Table 7-2)

7.5 ABORT OPERATION

At any point, the operator can issue an unsolicited command to abort the current job. This is done by typing CTRL T. It is important to note that this action will abort the current job and the System will search for the next job in the input stream. The message:

OPERATOR ABORT

is output on the line printer, and also recorded on the accounting file.

TABLE 7-2
TERMINAL ERRORS (IOPS)

Code	Function
Ø	Illegal CAL function code
1	CAL* illegal
2	.DAT slot error
3	Illegal Interrupt
4	I/O device not ready
5	Setup CAL issued with no skip chain entry
6	Illegal handler function
7	Illegal Data Mode or attempt to change from non-file oriented to file oriented I/O.
1Ø	File still active
11	.SEEK/.ENTER/.RAND/.INIT not executed
12	Terminal Device Error
13	File not found
14	Directory full
15	Device full
16	Output buffer overflow
17	Too many files for handler
2Ø	Disk hardware failure
21	Illegal block number
22	Two output files on one tape unit
23	Illegal Word Pair Count on attempt to transmit more than 73_{10} characters
24	Illegal Unit Number
25	Negative or Ø character count (IOPS ASCII write) x or y Increment too large (2**14) (Binary Write)
27	Illegal write type
3Ø	API software level not serviced
31	Nonexistent memory reference
32	Memory Protect violation
33	Memory Parity error
34	Power fail with no user-defined save routine
35	Attempt to recover from error caused an error.
37	Print line overflow
4Ø	Header label error (Magtape)
41	Directory format error - (Magtape)
42	Accessibility Map overflow (Magtape)
43	Directory recording error (Magtape)
44	Logical EOT found (Magtape)
45	Long input record (Magtape)
46	Attempt to delete system (SYS) file
47	Illegal Horizontal TAB (line printer)
51	Illegal User File Directory
55	No buffers available or no TCB available
61	Parity error in Directory, Bit Map, or RIB
63	Protected User File Directory
64	Protected File
65	Unrecoverable Magtape error or unrecoverable telecommunications error
66	Relative Block not within file
67	Illegal DECdisk word transfer starting address or count or I/O buffer above 32K
7Ø	Buffer size too small
71	Empty UFD
72	Input Parity or Write Check error
73	Null file name
74	Disk System File Structure degradation
75	Disk System File Structure degradation
76	Disk System File Structure degradation
77	Undersized or nonexistent CTRL Q Area

Note: For more information, see XVM/DOS Users Manual

To leave batch mode and go back to single user interactive mode, an End-Of-File (EOF) card (12-11-Ø-1 punches in column 1, or all rows punched in column 1) must be the terminating point in the job string. A CTRL C will unconditionally abort BOSS XVM and return control to the XVM/DOS monitor.

7.6 EXAMPLES OF BATCH OPERATION

In general, batch operation consists of adding BCL commands to one or more user source decks to set up serial system operation via card input. This usually involves at least the $JOB and $END control cards. The BOSS operating environment is illustrated in Figure 7-1. The basic I/O configuration for both System Operation and program execution is shown. The .DAT slots used in this figure are given as information only; the Procedure File automatically refreshes the Device Assignment Table with these slots for each system program load. Both the Procedure File and Run-Time File are co-resident on the system device (disk). Device assignments for user-program execution must be made via the $ASG command prior to the $XCT command.

The following sections present three examples of BOSS controlled job streams. Each example is the result of an actual Job execution.

1. Program Execution

   Example 1 illustrates a simple Job stream to load and execute a program residing on DECtape.

2. Program Control Functions

   Example 2 adds to the command functions used in Example 1 and, in addition, illustrates the use of the ASSIGN and Buffer commands.

3. CHAINTEST Example

   Example 3 is a more complex job stream. It illustrates assembly of a main source program and individual subroutines. The object routines are then linked together in an Overlay structure for execution.

Refer to Figure 7-2 for the "Console Log" of the above examples.

7.6.1 Program Execution Example

Figure 7-3 shows the "Program Execution Job Deck" for this example. The identification for this job is TEST5. The job card also specifies an automatic time-out after 20 minutes. This job string asks for a DECtape to be mounted on unit 1 and then loads and executes ($XCT) an execute file from that tape. A message to the operator will be output on the teleprinter as follows:

```
LOGW MOUNT D-TAPE# TEST1 ON DRIVE# 1 - WRITE LOCK
```

Then CTRL P must be depressed on the keyboard to resume operation.

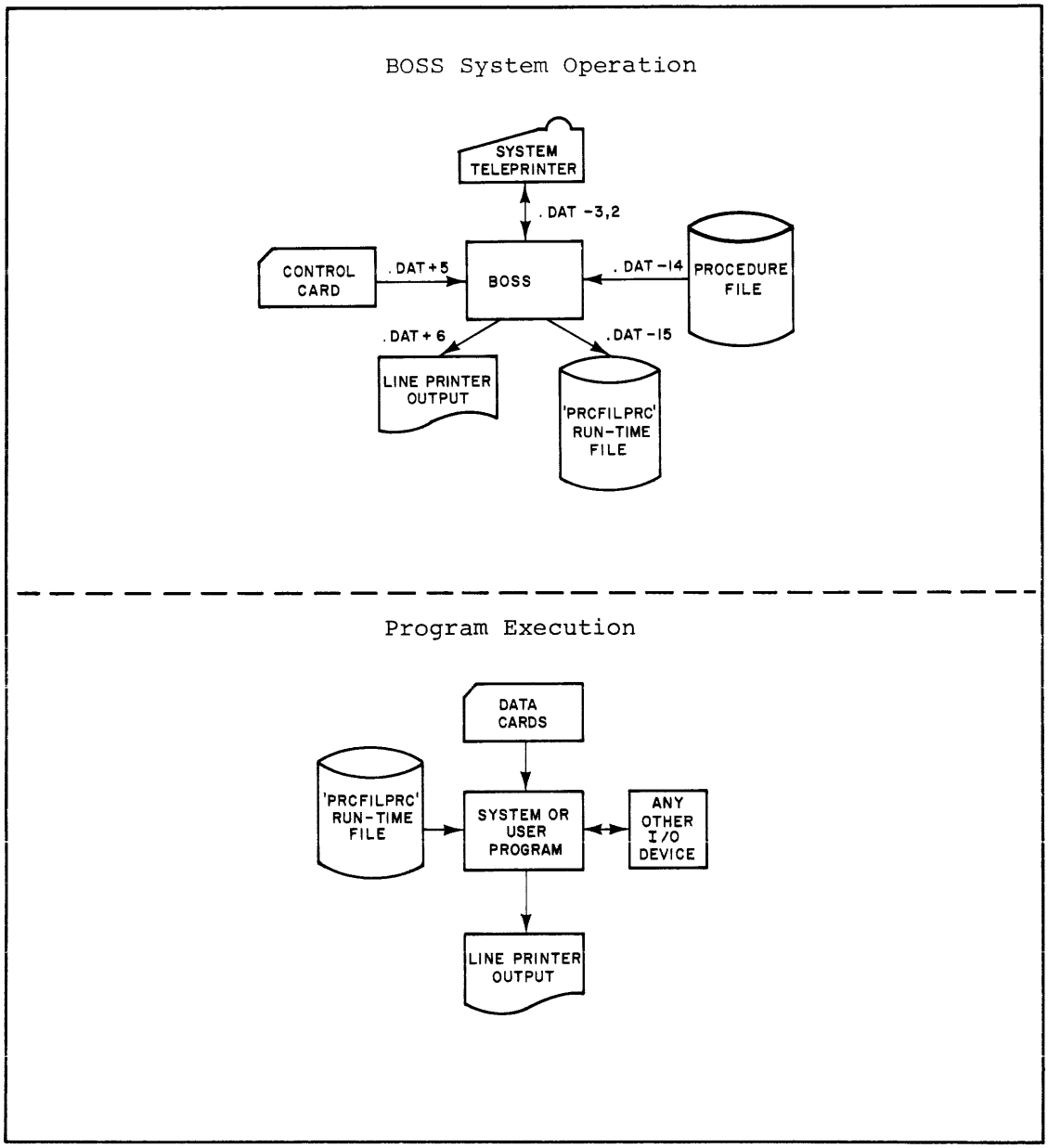


Figure 7-1 Operating Environments

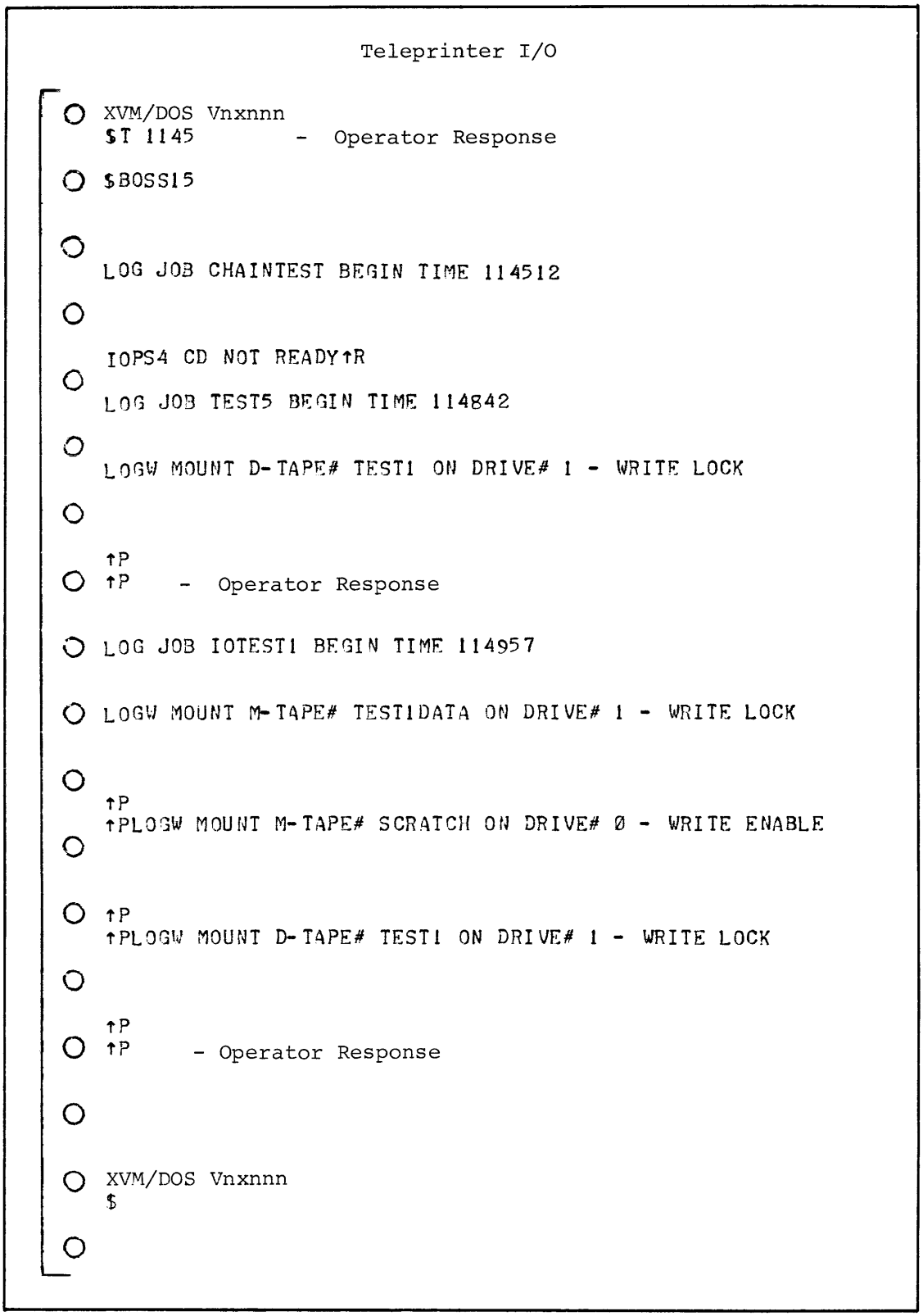


Figure 7-2 Console Log for Program Examples

Figure 7-4 shows the "Program Execution Job Listing" for this example. Job listings include the System identification

XVM BATCH OPERATING SYSTEM Vnxnnn

and a program message :

STOP ØØØØØØ

The listing also presents Accounting File information as indicated by the five columnar headings.

7.6.2 Program Control Function Example

Figure 7-5 illustrates a job deck containing several program control functions. The job identification is IOTEST1. A time-out function of 20 minutes is indicated. The job is logged in under the User Identification Code "EHM".

Two tapes are requested by the mount=tape $MNT commands.
An input tape "TEST1DATA" must be mounted on magtape unit 1 which must be in a write LOCK protect status. These requirements for the operator were in the form of a teleprinter message as follows:

LOGW MOUNT M-TAPE# TEST1DATA ON DRIVE# 1 - WRITE LOCK

A work tape "SCRATCH" was requested by the second $MNT card which generated the following teleprinter message:

LOGW MOUNT M-TAPE# SCRATCH ON DRIVE# Ø - WRITE ENABLE

The third $MNT card requested a program tape "TEST1" via the following message:

LOGW MOUNT D-TAPE# TEST1 ON DRIVE# 1 - WRITE LOCK

Note that the default assumption for the device type is a DECtape and not magtape.

The job requests the above tapes be mounted and then assigns ($ASG) six .DAT slots to specific devices in order to perform the user's

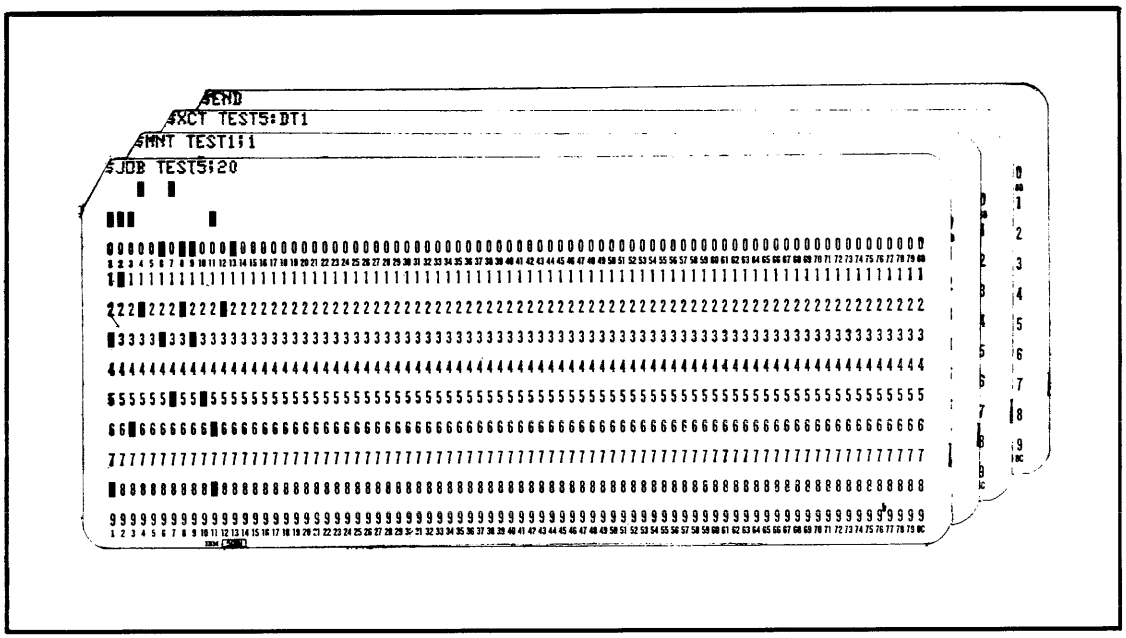


Figure 7-3 Program Execution

```
                                Job Listing

$JOB TEST5;20 - Job Card


XVM BATCH OPERATING SYSTEM Vnxnnn - System Acknowledgement
$MNT TEST1;1
$XCT TEST5:DT1
STOP  000000        - Program Output
$END
                Note:  This job asks for a DT to be mounted, then
                loads and executes a program from this tape.

                              Accounting File

JOB I.D.          DATE              START TIME        END TIME          RUN TIME
TEST5             NOV 25 '75        00:07:55          00:08:27          00:00:32
```

Figure 7-4 Program Executing Job Listing

program I/O. Buffers are assigned by the $BUF command for the total number of buffers required. Included in the number given in the $BUF␣7 card is a buffer for the program tape TEST1 indicated on the $XCT card. The number 7 does not include one additional buffer which is automatically assigned by the system for the Run Time File.

The TEST1 program is executed via the $XCT card. This program sorts card images from DECtape, magtape, and DECdisk. The new order is listed on the Line Printer as illustrated in Figure 7-6.

Figure 7-6 illustrates the output listing for IOTEST1. All job control commands are echoed along with the program output listing and the Accounting File entries for IOTEST1.

7.6.3 CHAINTEST Job Example

The job deck shown in Figure 7-7 includes only those BOSS-15 commands required to control the assembly of those link segments necessary for an overlay structure and then to execute the CHAINTEST program. The reader should refer to Figure 7-8 for individual source programs.

This example illustrates the following job conditions:

1. $JOB command control.
2. MACRO Assembly of:
   1 main program "MAIN"
   6 subroutines, LNK1-5 and "DUMMY"
3. Create an OVERLAY structure via $LNK cards and a load module via the $OVL card.
4. Execute the load module via the $XCT card.

A core map and a running log of subprograms is illustrated in Figure 7-8 following the $XCT MAIN line. The final output for this job is the Accounting File for CHAINTEST.

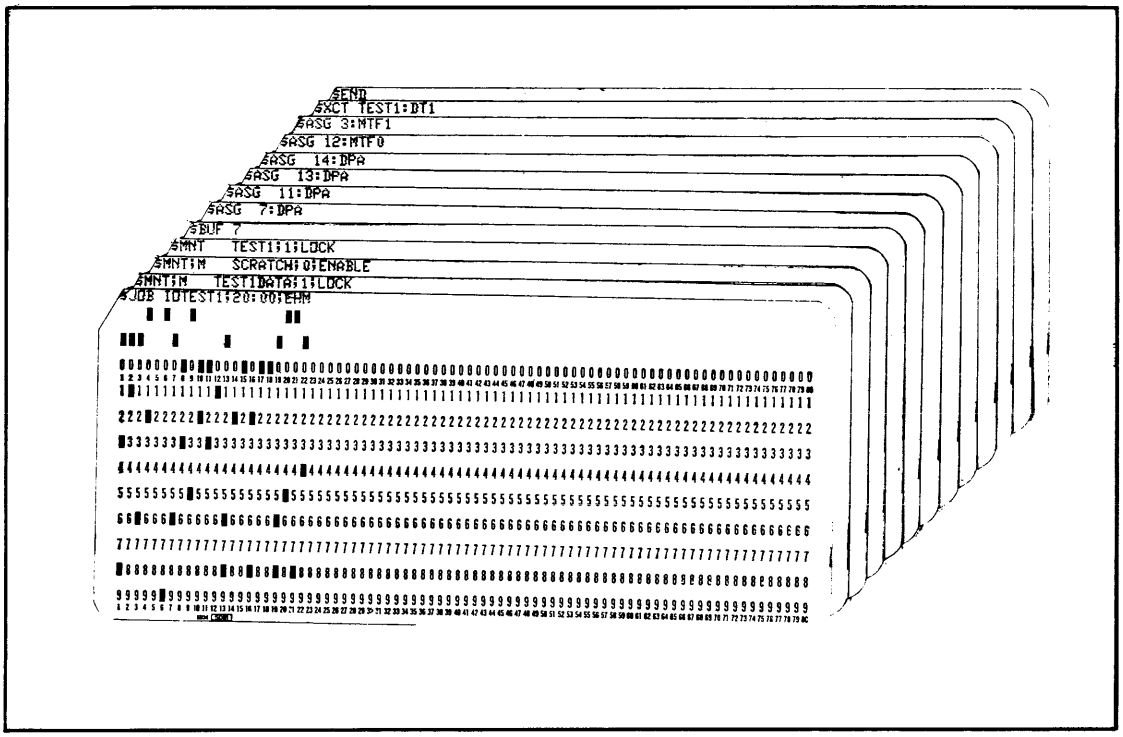


Figure 7-5 Job Deck Illustrating Program Control Functions

```
$JOB IOTEST1;20;00;EHM


XVM BATCH OPERATING SYSTEM Vnxnnn
$MNT;M   TEST1DATA;1;LOCK
$MNT;M   SCRATCH;0;ENABLE
$MNT   TEST1;1;LOCK
$BUF 7
$ASG  7:DPA
$ASG  11:DPA
$ASG  13:DPA
$ASG  14:DPA
$ASG 12:MTF0
$ASG 3:MTF1
$XCT TEST1:DT1
```

Figure 7-6 Print Listing Illustrating Program Control Functions

```
0.187231 N6S0,GE7ZOS*3Y(7IG0PQ A      RANDOM 3EST DATA      CARD    2
0.625965 6ILJR5 87G8X* Z)*02   B      RANDOM TEST DATA      CARD    3
0.281472  GKUS8,XRE8,05K73,R35 C      RANDOM TEST DATA      CARD    4
0.856929 P**HI)HJDBSNG5D,XH8FM D      RANDOM TEST DATA      CARD    5
0.930417 S30L,5Z2E4NO0-#,K*6PM E      RANDOM TEST DATA      CARD    6
0.955076 0S,R8PS3NX7CTNSCJDB , F      RANDOM TEST DATA      CARD    7
0.259010 +JLF,E,FGCV3*VU R4(2Y G      RANDOM TEST DATA      CARD    8
0.045381 ,(Y-QCX,940*B S,RTY6S H      RANDOM TEST DATA      CARD    9
0.392315 4A5HRG$J(V28GPAZUFJ0+ I      RANDOM TEST DATA      CARD   10
0.252906 PPU WJS(,1(Eg(SHSO4T* J      RANDOM TEST DATA      CARD   11
0.455305 MJIY8-I 1(EIX7CX,WIOZ K      RANDOM TEST DATA      CARD   12
0.702627 2 *5U) *+V*2( +M5*(L8 L      RANDOM TEST DATA      CARD   13
0.572985 C*HN52E089P586*B,+G B M      RANDOM TEST DATA      CARD   14
0.519516   LE3O1 ISF2 2BLYNTO0 N      RANDOM TEST DATA      CARD   15
0.870357 SJA4EL C585V TEK(MV97 O      RANDOM TEST DATA      CARD   16
0.828608 $S)RUY2SMX*J 9)JQ2.,+ P      RANDOM TEST DATA      CARD   17
0.472396 4LD41AL2,XV6 A(LN*X5* Q      RANDOM TEST DATA      CARD   18
0.754875 1QK5J4TGW)10 0(NZ*P60 R      RANDOM TEST DATA      CARD   19
0.504135 HG9WWBKPT8FV 2ZIZB4*A S      RANDOM TEST DATA      CARD   20
0.423322 Y 2CO+SSH,Z. JI3UYTLG T      RANDOM TEST DATA      CARD   21

0.707022 05ONI 4I9DI. NEUD WXF N      RANDOM TEST DATA      CARD  783
0.557847 V81PHU098CKG 25)*VAU, O      RANDOM TEST DATA      CARD  784
0.016083 UD 0 J51(I*S U S)N**R P      RANDOM TEST DATA      CARD  785
0.159886 7U4P4J+N(6AR JS=QGCQ* Q      RANDOM TEST DATA      CARD  786
0.942381 GBN*,PWPB249 L*WE4SSF R      RANDOM TEST DATA      CARD  787
0.191625 KPOHZK,AW)6G ,ES2KJ1D S      RANDOM TEST DATA      CARD  788
0.610828 D5SLSXV18V2  *L+ JWU7 T      RANDOM TEST DATA      CARD  789
0.778069 UL*)L2*CA*02 XH*A,+LX U      RANDOM TEST DATA      CARD  790
0.146458 3FUS8S UL*LI VRXABV8Z V      RANDOM TEST DATA      CARD  791
0.044160 5M*7EWTZ$9B* V*B *NV0 W      RANDOM TEST DATA      CARD  792
0.674306 GX S6YTC =)Y D56YP9** X      RANDOM TEST DATA      CARD  793
0.114963 OY67W +0R1XB ,(Y*S Q5 Y      RANDOM TEST DATA      CARD  794
0.319315 A=8RE3FYQ+ E SXW*H6BQ Z      RANDOM TEST DATA      CARD  795
0.115451 9GX ,JVM=ZON 1JT 5MBJ 0      RANDOM TEST DATA      CARD  796
0.206518 3Z XSMY 63*B  KAT66OE 1      RANDOM TEST DATA      CARD  797
0.170628 QM+7SN6,*9*7  TS(UZ5Y 2      RANDOM TEST DATA      CARD  798
0.460921 PVHQ*2=1RX)W XMU1 D(0 3      RANDOM TEST DATA      CARD  799
0.405499 XA**4S*J4OWU HSA6PHQ  4      RANDOM TEST DATA      CARD  800
 .OTS   11
$END


JOB I.D.        DATE              START TIME      END TIME        RUN TIME
IOTEST1         NOV 25 '75        23:19:21        23:23:25        00:05:57
```

Figure 7-6 (Cont.) Print Listing Illustrating Program Control Functions

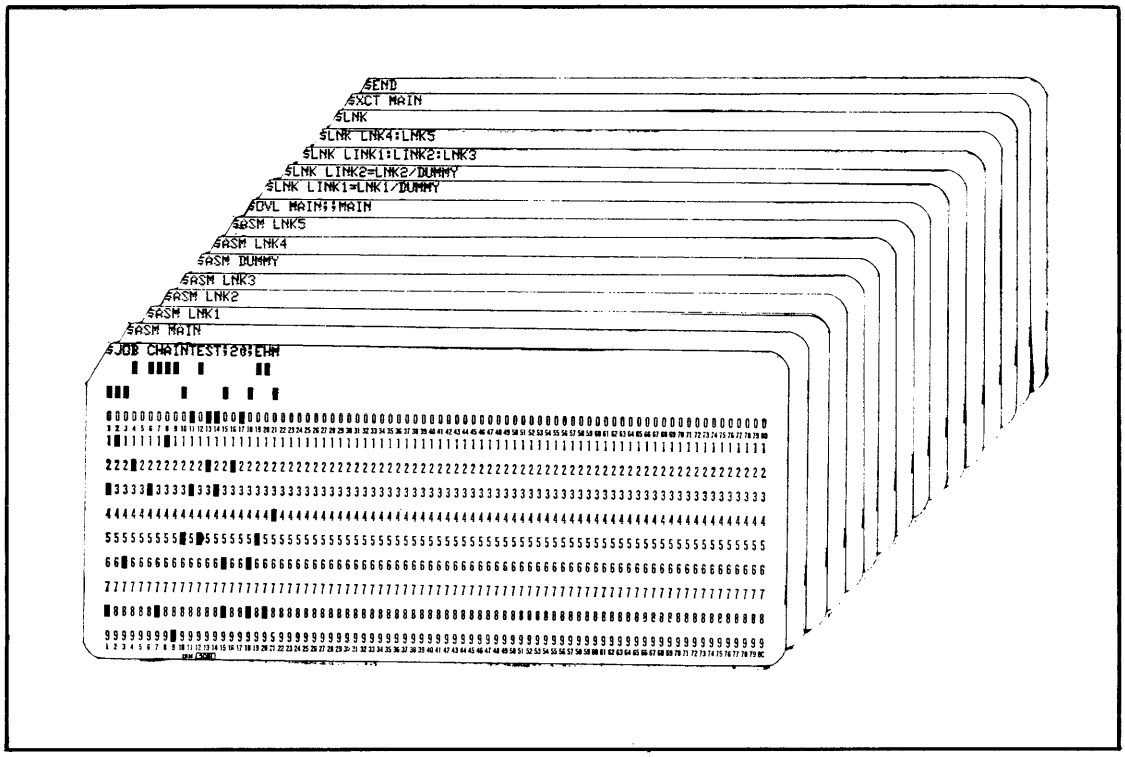


Figure 7-7 CHAINTEST Job Deck (without source cards)

Job Listing

```
$JOB CHAINTEST;20;EHM          - Job Card
                                 Job stream includes:
                                       1. Assembly ($ASM)
                                          One Main Program
                                          Six Subroutines
                                       2. Load Module with
                                          OVERLAY structure
                                          ($OVL and $LNK)
                                       3. Execute Load Module
                                          Core Map
                                          Program Output
XVM BATCH OPERATING SYSTEM Vnxnnn
$ASM MAIN                              - Assembly
 END OF PASS 1
```

Figure 7-8 Listing, CHAINTEST Example

Job Listing (continued)

```
PAGE   1      MAIN   SRC

                                        .IODEV 6
                                        .GLOBL WRITE,LNK1,LNK2,LNK3,RETURN
    00000 R                     MAIN    .INIT 6,1,0
    00000 R 001006 A *G                  CAL+1*1000 6&777
    00001 R 000001 A *G                  1
    00002 R 000000 A *G                  0+0
    00003 R 000000 A *G                  0
    00004 R 100027 R                    JMS WRITE
    00005 R 000013 R                    MAINLY
    00006 R 120043 E                    JMS* LNK1
    00007 R 120044 E                    JMS* LNK2
    00010 R 120045 E                    JMS* LNK3
                                        .EXIT
    00011 R 000000 A *G                  CAL
    00012 R 000015 A *G                  15
    00013 R 006002 A            MAINLY  006002
    00014 R 000000 A                    0
    00015 R 466031 A                    .ASCII /MAIN PROGRAM RUNNING/<15>
    00016 R 147100 A
    00017 R 502451 A
    00020 R 743644 A
    00021 R 406324 A
    00022 R 051252 A
    00023 R 472351 A
    00024 R 147216 A
    00025 R 064000 A
    00026 R 000000 A
    00027 R 000000 A            WRITE   0
    00030 R 220027 R                    LAC* WRITE
    00031 R 040034 R                    DAC BUFF
            000034 R            BUFF=.+2
                                        .WRITE 6,2,BUFF,36
    00032 R 002006 A *G                  CAL+2*1000 6&777
    00033 R 000011 A *G                  11
    00034 R 000034 R *G                  BUFF
                     *G                  .DEC
    00035 R 777734 A *G                  -36
                                        .WAIT 6
    00036 R 000006 A *G                  CAL 6&777
    00037 R 000012 A *G                  12
    00040 R 440027 R                     ISZ WRITE
    00041 R 620027 R                    JMP* WRITE
    00042 R 000000 A            RETURN  0
            000000 R                    .END MAIN
    00043 R 000043 E *E
    00044 R 000044 E *E
    00045 R 000045 E *E
SIZE=00046      NO    ERROR LINES
                SIZE=00046      NO ERROR LINES
```

Figure 7-8 (Cont.) Listing, CHAINTEST Example

Job Listing (continued)

```
$ASM LNK1
 END OF PASS 1

PAGE   1      LNK1   SRC

                                    .GLOBL LNK1,LNK4,WRITE
    00000 R 000000 A    LNK1    0
    00001 R 120016 E            JMS* WRITE
    00002 R 000005 R            LNK1LV
    00003 R 120015 E            JMS* LNK4
    00004 R 620000 R            JMP* LNK1
    00005 R 004002 A    LNK1LV  004002
    00006 R 000000 A            0
    00007 R 462231 A            .ASCII /LINK1 RUNNING/<15>
    00010 R 645542 A
    00011 R 202452 A
    00012 R 547234 A
    00013 R 446350 A
    00014 R 706400 A
            000000 R            .END LNK1
    00015 R 000015 E *E
    00016 R 000016 E *E
SIZE=00017      NO    ERROR LINES
                SIZE=00017      NO ERROR LINES




$ASM LNK2
 END OF PASS 1

PAGE   1      LNK2   SRC

                                    .GLOBL LNK2,LNK4,WRITE
    00000 R 000000 A    LNK2    0
    00001 R 120016 E            JMS* WRITE
    00002 R 000005 R            LNK2LV
    00003 R 120015 E            JMS* LNK4
    00004 R 620000 R            JMP* LNK2
    00005 R 004002 A    LNK2LV  004002
    00006 R 000000 A            0
    00007 R 462231 A            .ASCII /LINK 2 RUNNING/<15>
    00010 R 645500 A
    00011 R 311012 A
    00012 R 252634 A
    00013 R 472231 A
    00014 R 643432 A
            000000 R            .END LNK2
    00015 R 000015 E *E
    00016 R 000016 E *E
SIZE=00017      NO    ERROR LINES
                SIZE=00017      NO ERROR LINES
```

Figure 7-8 (Cont.) Listing, CHAINTEST Example

Job Listing (continued)

```
$ASM LNK3
 END OF PASS 1


PAGE   1      LNK3   SRC

                                    .GLOBL LNK3,LNK4,WRITE
    00000 R 000000 A     LNK3   0
    00001 R 120016 E            JMS* WRITE
    00002 R 000005 R            LNK3LV
    00003 R 120015 E            JMS* LNK4
    00004 R 620000 R            JMP* LNK3
    00005 R 004002 A     LNK3LV 004002
    00006 R 000000 A            0
    00007 R 462231 A            .ASCII /LINK 3 RUNNING/<15>
    00010 R 645500 A
    00011 R 315012 A
    00012 R 252634 A
    00013 R 472231 A
    00014 R 643432 A
            000000 R            .END LNK3
    00015 R 000015 E *E
    00016 R 000016 E *E
SIZE=00017     NO    ERROR LINES
                SIZE=00017     NO ERROR LINES
```

```
$ASM DUMMY
 END OF PASS 1

PAGE   1      DUMMY  SRC

                                    .GLOBL DUMMY
    00000 R 000000 A     DUMMY  0
    00001 R 620000 R            JMP* DUMMY
            000000 A            .END
SIZE=00002     NO    ERROR LINES
                SIZE=00002     NO ERROR LINES
```

Figure 7-8 (Cont.) Listing, CHAINTEST Example

Job Listing (continued)

```
$ASM LNK4
 END OF PASS 1


PAGE   1     LNK4   SRC

                                        .GLOBL LNK4,LNK5,RETURN,WRITE
    00000 R 000000 A     LNK4    0
    00001 R 200000 R             LAC LNK4
    00002 R 060017 E             DAC* RETURN
    00003 R 120020 E             JMS* WRITE
    00004 R 000006 R             LNK4LV
    00005 R 120016 E             JMS* LNK5
    00006 R 004002 A     LNK4LV  004002
    00007 R 000000 A             0
    00010 R 462231 A             .ASCII /LINK 4 CALLED/<15>
    00011 R 645500 A
    00012 R 321010 A
    00013 R 340630 A
    00014 R 462130 A
    00015 R 406400 A
            000000 A             .END
    00016 R 000016 E *E
    00017 R 000017 E *E
    00020 R 000020 E *E
SIZE=00021      NO    ERROR LINES
                SIZE=00021      NO ERROR LINES









$ASM LNK5
 END OF PASS 1


PAGE   1     LNK5   SRC

                                        .GLOBL LNK5,RETURN
    00000 R 000000 A     LNK5    0
    00001 R 220004 E             LAC* RETURN
    00002 R 040000 R             DAC LNK5
    00003 R 620000 R             JMP* LNK5
            000000 A             .END
    00004 R 000004 E *E
SIZE=00005      NO    ERROR LINES
                SIZE=00005      NO ERROR LINES
```

Figure 7-8 (Cont.) Listing, CHAINTEST Example

```
$OVL MAIN;/MAIN              - Create OVERLAY
$LNK LINK1=LNK1/DUMMY
$LNK LINK2=LNK2/DUMMY
$LNK LINK1:LINK2:LNK3
$LNK LNK4:LNK5
$LNK
$XCT MAIN                    - Execute Load Module

XCT FILE                     - Begin Core Map

MAIN

OPTIONS & PARAMETERS

SZ

RESIDENT CODE

MAIN

LINKS & STRUCTURE

LINK1=LNK1/DUMMY

LINK2=LNK2/DUMMY

LINK1:LINK2:LNK3

LNK4:LNK5

LINK TABLE
       57546-57636 00071

RESIDENT CODE
MAIN   57500-57545 00046

LINK **
LINK1

LNK1   57461-57477 00017
DUMMY  57457-57460 00002

LINK **
LINK2

LNK2   57461-57477 00017
DUMMY  57457-57460 00002

LINK **
LNK3

LNK3   57461-57477 00017

LINK **
LNK4

LNK4   57436-57456 00021

LINK **
LNK5

LNK5   57452-57456 00005

CORE REQ'D
       57436-57636 00201
```

Figure 7-8 (Cont.) Listing, CHAINTEST Example

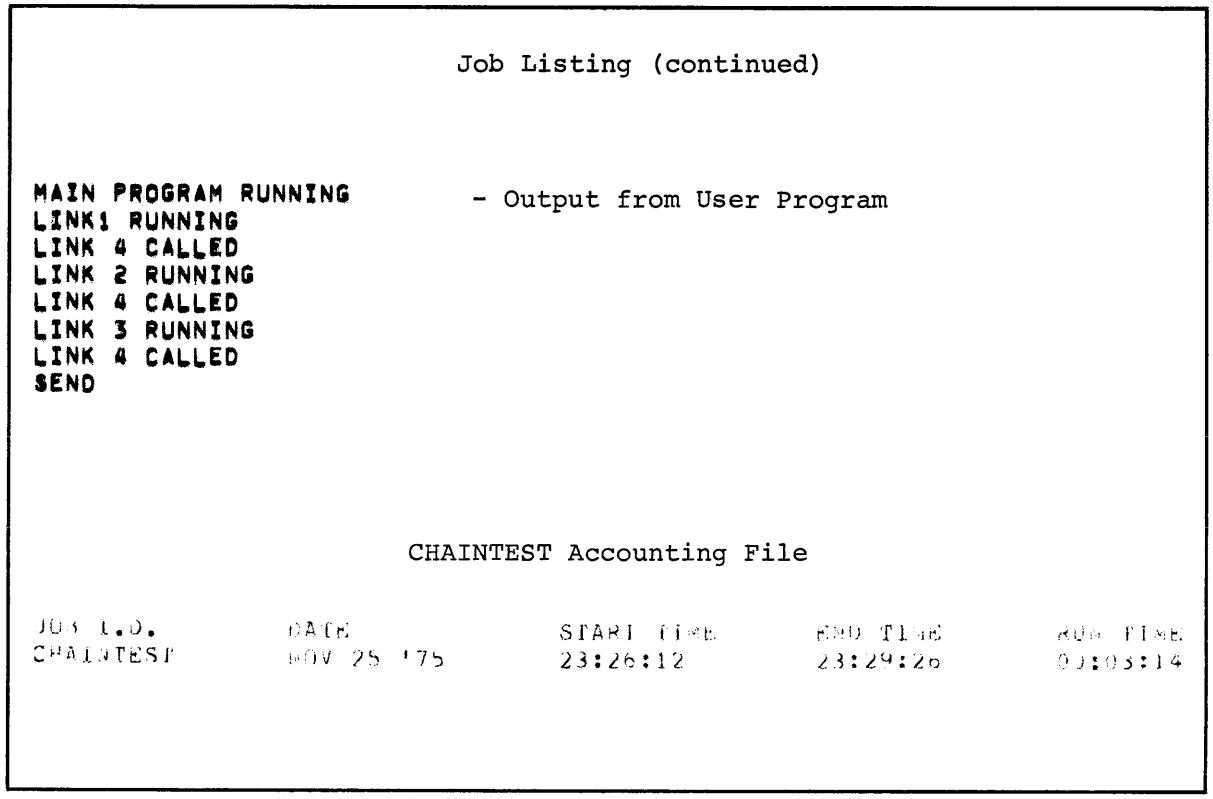


Figure 7-8 (Cont.) Listing, CHAINTEST Example

CHAPTER 8
RUN TIME FILE

Most System programs run by BOSS are identical to those run by XVM/DOS. Exceptions are the Resident and Nonresident Monitors, and B.PRE, the preprocessor. The XVM/DOS System Manual describes those programs in detail. BOSS expands the information on Control Cards into a series of commands in the format used by the XVM/DOS system programs. Nonresident BOSS does this command expansion, and sorts the expanded commands in a disk file, known as the Run Time File (PRCFIL PRC) (see Figure 8-1). Since XVM/DOS programs communicate with an operator at a teleprinter, BOSS feeds the expanded commands to the programs via .DAT slots assigned to the teleprinter. BOSS attaches DAT -2 to the Run Time File, and, unless instructed otherwise, directs teleprinter output to the Line Printer. Macro programs can initiate I/O to the teleprinter by setting bit 4 in .SCOM+52 and proceeding with macros directed to TAA (see Table 8-1). FORTRAN programs can do teleprinter I/O by calling TTON before any teleprinter I/O and calling TTOFF to resume line printer I/O to DAT slots assigned to TAA. (Program Termination causes an automatic TTOFF.)

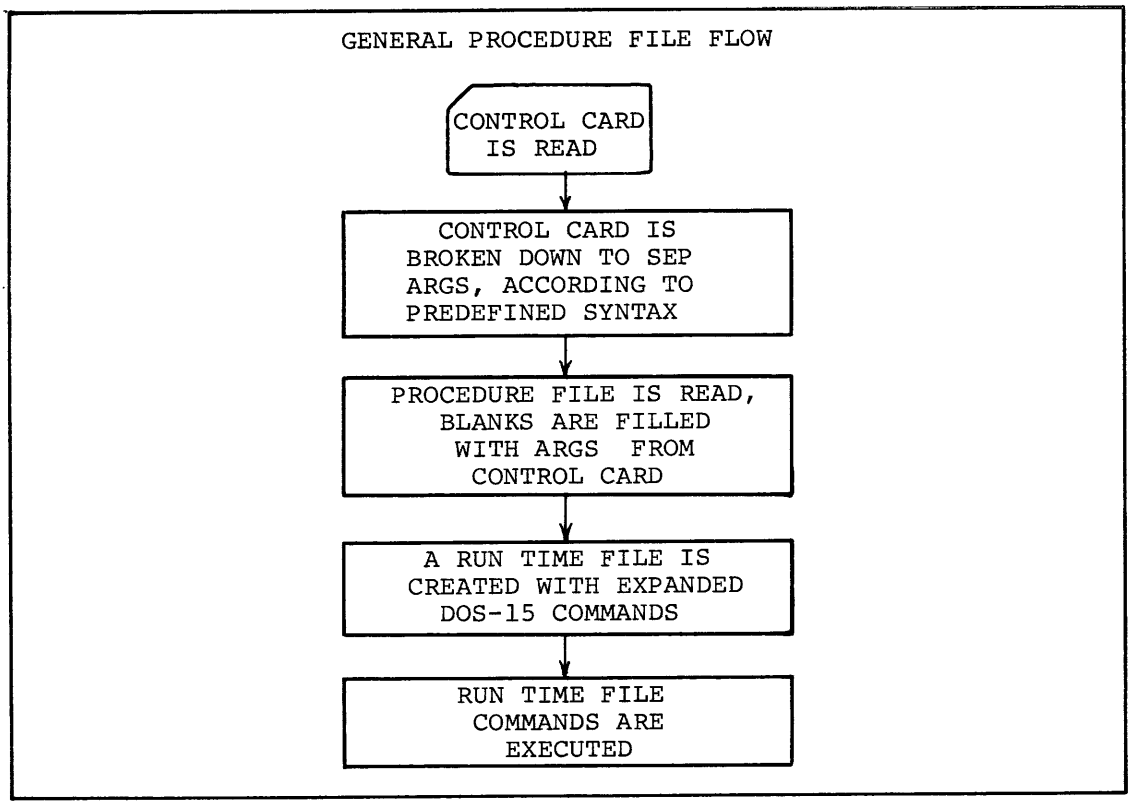


Figure 8-1 General Procedure File Flow

Table 8-1

.SCOM Table Entries

.SCOM = location 100_8	(For complete table see XVM/DOS System Reference Manual).
.SCOM+Ø thru +33	(See XVM/DOS System Reference Manual)
.SCOM+34	Elapsed time in seconds
.SCOM+35 thru +51	(See XVM/DOS System Reference Manual
.SCOM+52	BOSS Mode Bit Register: Bit Ø = 1 BOSS Mode. 1 = 1 Control Card Read by user, 5/7 ASCII image saved in first block of NRBOSS. 2 = 1 Resident BOSS reached "EOF" on run time estimate. 3 = 1 User exceeded time estimate. 4 = 1 I/O CAL to go to TTY. 5 = 1 Terminal IOPS error by user. 6 = 1 QDUMP to be given to user on IOPS error. 7 = 1 Operator abort (Control T). 8 = 1 Job active. 9 = 1 Exit from BOSS-15 Mode. 10 = 1 User tried to do a .PUT. Core will be dumped and a listing given on LP. 11 = 1 User tried to do a .GET. 12 -13 Not defined. 14 -16 .SYSLD error number. 17 = 1 Job abort.
.SCOM+53 thru +74	(See SVM/DOS System Reference Manual)
.SCOM+75	Bit Ø - 8 # of lines in RTF. 9 - 17 # of lines READ in RTF.
.SCOM+76 thru +77	(See XVM/DOS System Reference Manual)

8.1 PROCEDURE FILES

For each BOSS command there is a corresponding disk-resident ASCII file, called a Procedure File. The Procedure File contains XVM/DOS commands. When XVM/DOS (under BOSS) executes the commands in the Procedure File, it carries out the function specified by the BOSS control card. The XVM/DOS commands in the Procedure Files contain fields (for instance, a file name) that are ordinarily filled in by the XVM/DOS operator at the teleprinter. Nonresident BOSS fills them in with text strings from the control card. These fields are called, "Variable Fields". Before executing the XVM/DOS commands contained in the Procedure File, all the variable fields have to be resolved. This process is very similar to a macro expansion. The expanded XVM/DOS commands are put in a Disk File, called the "Run Time File (PRCFIL PRC)". The RTF can contain the expansion of one or more Procedure Files, up to 777_8 IOPS ASCII records.

BOSS expands Procedure Files strictly on a text string, character basis. It has no knowledge of the intrinsic function of each BOSS control card, except for $JOB and $END, $CRT, and $ADD ($END, $MNT, and $ADD have no Procedure Files).

8.1.1 Procedure File Format

File Control Record

FORMAT: [nnn] [comment]

n = decimal number (Ø - 4) indicating control option

Table 8-2

Procedure File Option Classes

File Options	Option Class I	Option Class II	Option Class III
Option Code	0,3	1,2	4

Options can be combined in any permutation but cannot include options from the same class.

where:

Ø = Expanded substitution (default - without 3).

1 = Open-ended file (default - without 2).

2 = Closed-ended file (leading spaces ignored).

3 = Direct substitution (leading spaces ignored).

4 = Test Mode - echoes RTF lines on line printer.

NOTE: Ø, 3 and 1, 2 are illegal combinations.

In order to ensure successful expansion, all Procedure Files must follow a strict format. The first record of the Procedure File must be a control record, with parameter information. The first record may also contain comments, because BOSS interprets only pertinent information, and ignores the rest. The numbers Ø, 1, 2, 3, and 4 specify different options. All other characters are ignored. The option digits can appear in any order, and anywhere on the record. The option specified by each number is given below:

Ø - Expanded Substitution (default, if "3" not given explicitly)

This option specifies that the Procedure File is to be expanded according to the normal rules of substitution, which are given below.

1 - Open Ended File (default, if "2" not given explicitly)

This option instructs the Non-Resident BOSS Monitor to leave the Run Time File open after expanding the current Procedure File. BOSS then searches for the next control card.

2 - Closed End File

This option instructs Non-Resident BOSS to close the Run Time File after expanding the current Procedure File, and to execute the XVM/DOS commands in the Run Time File. Procedure Files corresponding to commands that may possibly be followed by "Data Cards" should be of Type 2.

3 - Direct Substitution

This option indicates that BOSS should not expand the Procedure File according to normal rules. Refer to paragraph 8.1.2 for information on Direct Substitution.

4 - Test Mode

This option indicates that BOSS should echo the Procedure File expansion on the Line Printer. This allows a check on the Procedure File.

The following combinations are illegal:

Ø and 3

1 and 2

If BOSS finds an illegal option combination, it will print,

ILLEGAL PROC FILE

and search for the next control card.

8.1.2 Source Control Records

$L = [K\ [V]^n]^m$ or $[[V]^n K]^m$

where:

L = Source Control Record (line) and succeeding records starting at the second line of the Procedure File.

K = Constant.

V = Variable in the following format:

@XNN(K or V)@

where: @ = Variable Record delimiter.

X = O, A, D, U, denoting procedure fields:

O = Option field delimited from BCL Command Field by a **semicolon.**

A = filename text:

1 - delimited from BCL command or Option field by a blank.

2 - delimited in all other cases by a semicolon.

D = Device and unit BCL delimiter = a colon.

U = uic BCL delimiter = (.

NN = Decimal digits ØØ - 99 denoting a specific field in sequence.

Expanded Substitution - includes the Option Field (O) and:

AØØ - AØ9
DØØ - DØ9
UØØ - UØ9

Direct Substitution - AØØ - A99 starting at ØØ sequential order. A field can NOT appear more than once. (Exceptions are DlØ - Dl7)

Reserved field numbers are:
Dll = System Device,
Dl2 = Current uic,
Dl3 = Carriage Return,
Dl4 = Altmode.

() = Default Record delimiters.

n & m = Repeat functions (Number of characters/ record not to exceed line length).

BOSS uses all other records in the Procedure File as macro definition records. Records after the first one are all Macro Definition Records. For each such record, a record will be written in the Run Time File. Each Macro Definition Record has the same format. Two types of fields are used: K-fields and V-fields. K-fields specify constant character strings that will be written into the Run Time File exactly as they

appear in the Procedure File. V-fields specify variable character strings to be replaced by specified strings on the Control Cards. Each Macro Definition Line of a Procedure File can contain any number of K- and V-fields, in any combination. V-fields are delimited at @-signs. K-fields are delimited by adjacent V-fields, or the end or beginning of the record. Since there are only two types of fields, only one need have delimiters. Two adjacent V-fields, however, require two adjacent @-signs and parenthesis.

K-fields may be any string of legal IOPS ASCII characters, except the @-sign.

V-fields

A V-field has the following format:

$$V = @\left\{\begin{matrix} A\emptyset n \\ U\emptyset n \\ Dnn \\ O \end{matrix}\right\} (@[\left\{\begin{matrix} \text{V-field} \\ \text{K-field} \end{matrix}\right\}]@)@$$

The two @-signs delimit the field. The first part of the field (A, D, U or O) is a card-position identifier, and must be present. It identifies the position on the current Control Card of the character string to be substituted in the Run Time File. The legal combinations are:

```
AØØ,AØ1,....AØ9
UØØ,UØ1,....UØ9
DØØ,DØ1,....DØ9,D1Ø,...D17
O
```

With the exception of D1Ø through D17, each of the above position identifiers corresponds to a unique character string of the Control Card, according to the following scheme:

```
$CMD:O AØØ:DØØ(UØØ);AØ1:DØ1(UØ1);....;AØ9:DØ9(UØ9)
```

The D1Ø...D17 position identifiers do not correspond to character strings found on the Control Card, but rather to character strings defined by BOSS. Thus,

D1Ø - Unused
D11 - .SIXBT representation of the System Device Code (DK or DP)
D12 - Current Logged-in UIC
D13 - .SIXBT representation of Carriage RETURN
D14 - .SIXBT representation of ALT MODE
D15 - Unused
D16 - Unused
D17 - Unused

The parentheses in a V-field must be present. They are used to specify a default string. The default string is used in case the string on the Control Card specified by the position identifier is null. A set of parentheses must be included, even if the default string is null. The default string itself can be a variable, resulting in nested variables.

8.1.3 Direct Substitution

When processing a Direct Substitution Procedure File, BOSS places the fields on the Control Card into the Run Time File just as they stand, with only leading spaces ignored. That is, BOSS does not necessarily expect to find file names, and so on, as with normal substitution. Fields on the Control Cards are separated by semi-colons (;), and are processed in a serial manner. The ampersand (&) is used for a special purpose. It causes the current record being composed for the Run Time File to be terminated with a Carriage RETURN, and written out, and a new record started. This is so that the limit of seventy-five characters per line will not be exceeded.

There are only two legal field types within the Procedure File. They are as follows:

1. AØØ through A99
2. D1Ø through D17 (System Defined)

In making up Direct Substitution Procedure Files, the following rules must be followed:

1. The first line must contain a three (3). This declares the file to be direct substitution.
2. The "A" fields must appear in sequential order, starting at AØØ. Each "A" field can be used only once within the Procedure File.

3. The "D" fields can only be "DlØ" through "Dl7". They can be used any number of times, in any order.
4. Variable expressions must follow the standard V-field format, as in expanded substitution.

8.1.4 Example of Procedure File

The following example shows a typical Direct Substitution Procedure File, the Control Cards used to call it and the resulting lines produced in the Run Time File (RTF).

Procedure File - MAP PRC

```
1;3 DIRECT SUB FILE FOR CHAIN OPTION AND RES CODE ONLY
CHAIN
@AØØ(TMPXCT)@@Dl4()@
@AØ1(SZ)@@Dl4()@
@AØ2(FILTMP)@@Dl4()@
@Dl4()@
```

Control cards as they appear:

```
$MAP  TESTl;SZ,BNK;MAIN,SUB
```

Run Time File Lines:

```
CHAIN<CR>
TESTl<ALTMODE>
SZ,BNK<ALTMODE>
MAIN,SUB<ALTMODE>
<ALTMODE>
<ALTMODE>
```

NOTE: Dl4 =<ALTMODE> is an ALTMODE.

8.2 SUMMARY - DIRECT SUBSTITUTION CONTROL CARDS

1. Field Delimiters:

   SEMI-COLON(;) ONLY

2. Field Names:

   "A" FIELDS ONLY, AND IN SEQUENTIAL ORDER.

   Note: The appearance of an option field on the Control Card will cause an "ILLEGAL COMMAND" error to be printed on the Line Printer.

3. Card Continuation:
   A Semi-Colon (;) followed by a Carriage RETURN or ALTMODE with intervening spaces ignored. The field is not terminated.

4. Special Characters:
   The ampersand(&) writes out the Run Time File line with a Carriage RETURN and starts a new Run Time File line.
5. Procedure File
   First Line:
   A 3 must appear on this line.
6. Fields:
   AØØ thru A99 in sequential order starting at AØØ. Each field can be used only once.
   DlØ thru D17 in any order and can be used any number of times.

TABLE 8-3

FILE I/O STATUS

Procedure File	Procedure File Types			
	1 Open End	2 Close End	3 Direct Substitution	Ø Expanded Substitution
$ASG	X			X
$ASM		X		X
$BNK[1]		X		
$BUF	X			X
$CHN	X			X
$CMP	X			X
$DIR	X			X
$DLG		X		X
$DMP	X			X
$DOS	X			X
$FIL		X		X
$FOR		X		X
$JOB		X		X
$KEP	X			X
$LCM	X		X	
$LIB	X			X
$LNK	X		X	
$LOG		X		X
$LST		X		X
$MAC		X		X
$MAP	X		X	
$MIC		X		X
$MNT	X			X
$MSG	X		X	
$MSW	X		X	
$NDR	X			X
$OVL	X		X	
$PAG[1]		X		
$PRT	X			X
$QDP[1]	X			
$RUN		X		X
$XCT		X		X
$XVM		X		

[1]These Procedure Files do not fall under the category 3 or Ø because there is no argument field, therefore no substitution takes place.

Definitions of Procedure File Types

1 Open End Type - The current run-time file remains open so that other Procedure Files can be expanded into the run-time file.

2 Close End Type - The current run-time file is closed and control is passed to the Non-Resident Monitor.

3 Direct Substitution - The various fields on the Control Cards are put directly into the current run-time file as they appear on the card, only the " ; " field delimiter is omitted.

Ø Expanded Substitution - The various fields on the Control Card are expanded per the procedure file into the current run-time file.

APPENDIX A
QUICK REFERENCE TABLES

A.1 INTRODUCTION

This Appendix contains tabularized data intended as an aid to recall for users who are familiar with the contents of the PIP XVM Utility Manual. Tables are supplied which describe the Primary PIP Operations, PIP Optional operations, optional functions permitted within each primary operation, and the PIP command structure, plus a series of tables describing specific operations which may be carried out using PIP facilities. In some cases, the tables presented in this section are duplicates of those contained in the PIP XVM Utility Manual; this redundancy is necessary to make this manual a complete single source of information.

A.2 OPTIONS VERSUS PRIMARY OPERATIONS

The matrix Table A-1 illustrates the optional PIP function switches permitted in each of the PIP primary operations.

A.3 PIP COMMAND STRING FORMAT CHARTS

Figure A-1 illustrates the general format of the Destination/Source command string and the general format of the Single Device command string.

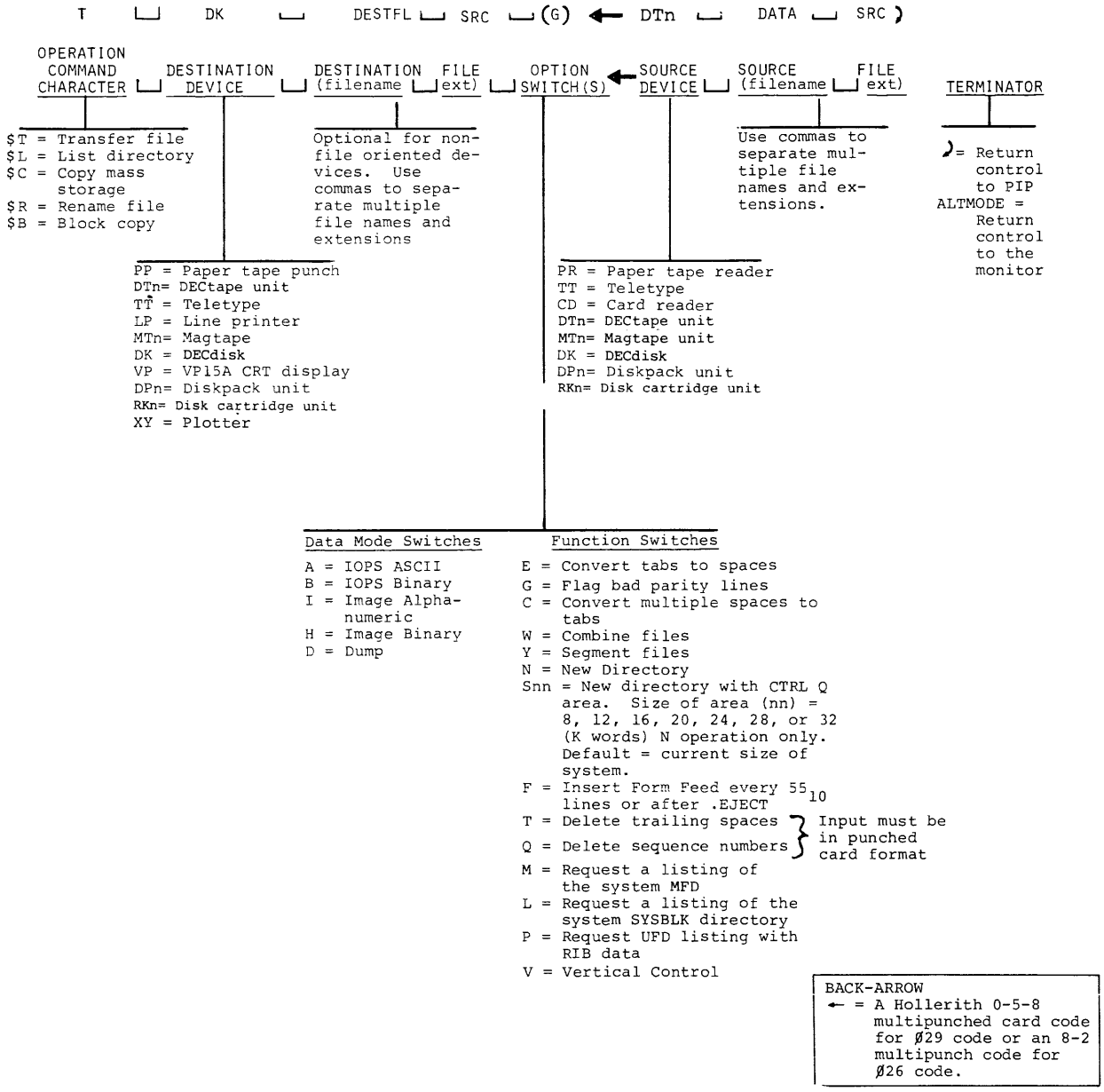


Figure A-1

PIP Command String Formats

A.4 REFERENCE TABLES

The PIP Tables are organized according to the following functional areas:

Table	Functions Described
A-1	Options, available options versus primary operations.
A-2	Directory Operations, how to set up, list, modify, and operate with disk (UFD) and DECtape directories.
A-3	List Operations, how to obtain printouts of directory and file information.
A-4	File Modification Operations, how to modify and manipulate files and file contents.
A-5	File Transfer Operations, how to transfer files between system storage devices.
A-6	Copy Operations, how to copy device contents, large groups of files or data blocks from system mass storage devices.
A-7	PIP Error Messages, how to interpret error conditions.

No attempt is made to describe all possible operation/switch combinations or applications. However, the user is provided with sufficient information to meet his reference requirements.

TABLE A-1

AVAILABLE OPTIONS VERSUS PRIMARY OPERATIONS

SWITCHES	TRANSFER T	VERIFY V	SEGMENT S	LIST L	NEW DIRECTORY N	DELETE FILE D	RENAME FILE R	COPY TAPE C	BLOCK COPY B	INITIALIZE I[2]	UPDATE U	DESCRIPTIONS
(A)	x	x										IOPS ASCII Data Mode
(B)	x	x										IOPS Binary Data Mode
(C)	x											Space to Tab Conversion
(D)	x											Dump Mode
(E)	x											Tab to Space Conversion
(F)	x											Insert Form Feed
(G)	x											Correct Bad Parity
(H)[1]	x							x				Image Binary Mode
(I)	x											Image Alphanumeric Mode
(N)	x							x	x			New Directory
(Snn)	x				x			x	x			New Dir. with CTRL Q area
(T)	x											Delete Trailing Spaces
(Q)	x											Delete Sequence Nos. (cards)
(W)	x											Combine Files
(Y)[3]	x											Segment Files
(K)					x							Delete Current UFD
(M)				x								List Unprotected UFDs
(X)						x						Delete Truncated Files
(P)				x								List Current UFD & RIB data
(L)				x								List SYSBLK
(V)	x											Vertical Forms Control

[1] Legal only with DECtape or Disk Pack or Disk Cartridge copy operations.

[2] Legal only when the current password is the MIC.

[3] Transfer, T, commands which include a Y option switch must be preceded by an S, segment command string.

TABLE A-2

DIRECTORY OPERATIONS

PROCESS	USE OPERATIONS	OPTION	EXAMPLE
1) Initialize the existing UFD or create a new UFD for the current UIC	N, New	None	$N␣DK
2) Initialize the existing UFD or create a new UFD for the current UIC during a T transfer	T, Transfer	(N),(NA), (NB	$T␣DK␣DESTFL←DT1␣SOURCE␣(NA)
3) Initialize or create a UFD for the current UIC with a specified UFD protection code	N with code specified written	(cd)	$N␣DK␣(Ø)
4) Delete a file from the current UFD	D, Delete	None	$D␣DK␣DESTFL␣BIN Note: Filename extension is required.
5) Rename a file in the current UFD and keep old protection code	R, Rename	None	$R␣DK␣NEWNAM␣BIN←DK␣OLDNAM␣BIN Note: Filename extension is required.
6) Rename a file in the current UFD and set new protection code	R, Rename	(cd)	$R␣DK␣NEWNAM␣BIN←DK␣OLDNAM␣BIN␣(3) Note: Protection code is specified within parentheses.
7) Change protection code of the current UFD	R, Rename	None	$R␣DK␣<JOE:Ø>
8) Initialize the directory of a non-mass storage device	N, New Directory		$N␣DT1
9) Initialize mass storage device directory and set up a QAREA OF:			
a) System core size		(S)	$N␣DT1␣(S)
b) Specified size		(Snn)	$N␣DT1␣(S32)

TABLE A-2 (Cont.)
Directory Operations

PROCESS	USE OPERATIONS	OPTION	EXAMPLE
10) List contents of directories:	L		
a) MFD	L, List	(M)	$L␣TT←DK␣(M)
b) UFD	L, List	None	$L␣TT←DK
c) DECtape	L, List	None	$L␣TT←DT
11) List contents of unprotected, noncurrent UFD	L, List	<UIC>	$L␣TT←DK␣<JOE>

TABLE A-3

LIST OPERATIONS

PROCESS	USE OPERATIONS	OPTION	EXAMPLE
1) List the system MFD	L, List	(M)	$L␣LP←DK␣(M)
2) List the UFD of the current UIC	L, List	None	$L␣TT←DK
3) List the contents of an unprotected non-current UFD	L, List	<UIC>	$L␣TT←DK␣<JOE>
4) List the system SYSBLK	L, List	(L)	$L␣TT←DK␣(L)
5) List current UFD with File Protection Codes and RIB information	L, List	(P)	$L␣TT←DK␣(P)
6) List directories of Mass Storage devices	L, List	None	$L␣TT←DT1
7) List a selected file entry from the current UFD or device directory	L, List	None	$L␣ TT←DK ␣ FILNAM␣BIN
8) List a selected file entry from an unprotected noncurrent UFD	L, List	None	$L ␣ TT←DK ␣ FILNAM ␣ SRC ␣ <JOE>
9) List all file entries having the same filename extension from			
a) the current UFD or device directory, or	L, List	None	$L␣TT←DK ␣ #SRC
b) an unprotected noncurrent UFD	L, List	<UIC>	$L ␣ TT←DK ␣ #BIN ␣ <JOE>

TABLE A-4

FILE MODIFICATION OPERATIONS

PROCESS	USE OPERATION	OPTION	EXAMPLE
1) Rename a file	R, Rename	None	$R␣DK␣NEWNAM␣BIN←DK␣OLDNAM␣BIN
2) Rename and set the protection code for a file	R, Rename	<UIC>	$R␣DK␣NEWNAM␣BIN←DK␣OLDNAM␣BIN␣(3)
3) Delete a file:			
a) from current UFD	D, Delete	None	$D␣DK␣FILEA␣BIN
b) from unprotected UFD	D, Delete	<UIC>	$D␣DK␣<JOE>␣FILEA␣BIN
c) from mass storage device directory	D, Delete	None	$D␣DT1␣FILEA␣BIN
4) Delete all truncated files from a UFD which is:			
a) current	D, Delete	(X)	$D␣DK␣(X)
b) specified	D, Delete	<UIC>	$D␣DK␣<ABC>␣(X)
5) Convert multiple spaces to tabs	T, Transfer	(C), (CA)	$T␣LP␣DESTFL←DT1␣SOURCE␣(CA)
6) Convert tabs to spaces	T, Transfer	(E), (EA)	$T␣LP␣DESTFL←DT1␣SOURCE␣(EA)
7) Detect and correct File Parity and Check-sum errors	T, Transfer	(G), (GA)	$T␣DK␣DESTFL←DT1␣SOURCE␣(GA)
8) Delete Trailing Spaces from file contents	T, Transfer	(T), (TA)	$T␣DK␣DESTFL←DT1␣SOURCE␣(TA)
9) Delete sequence from file (card input or card format data files only)	T, Transfer	(Q), (QA)	$T␣DT1␣DESTFL␣SRC␣(Q)←CD
10) Combine separate files into one file	T, Transfer	(W)	$T␣DT1␣LIBR␣(WB)←DT2␣FILA␣BIN,FILB␣BIN

TABLE A-4 (Cont)
File Modification Operations

PROCESS	USE OPERATION	OPTION	EXAMPLE
11) Segment a file into 2 to 16 separate:			
a) files	S, Segment +	(YA)	$S␣TAGA,TAGB,TAGC
	T, Transfer		$T␣DT1␣FILA,FILB,FILC,FILD␣(YA)←DT2␣SOURCE
b) papertapes	S, Segment +	(YA)	$S␣TAGA,TAGB,TAGC
	T, Transfer		$T␣PP␣,,,␣(YA)←DT1␣SOURCE
12) Verify a file for parity and/or checksum errors	V, Verify	(A), (B)	$V␣DK␣FILEA␣(A)

Table A-5
File Transfer Operations

PROCESS	USE OPERATION	OPTION	EXAMPLE
1) Transfer a file to the current UFD	T, Transfer	(A) or (B)	$T␣DK␣DESTFL←DT1␣SOURCE␣(A)
2) Transfer a file to a selected noncurrent UFD when protection code permits	T, Transfer	(A) or (B) <UIC>	$T␣DK␣<JOE>DESTFL←DT1␣SOURCE␣(A)
3) Transfer a file from a current UFD to a specified noncurrent unprotected UFD	T, Transfer	(A) or (B) <UIC>	$T␣DK␣<JOE>DESTFL←DK␣SOURCE␣(A)
4) Transfer file and:			
a) Convert spaces to tabs	T, Transfer	(C) or (CA) only	$T␣TT←DT1␣SOURCE␣(CA)
b) Convert tabs to spaces	T, Transfer	(E) or (EA) only	$T␣TT←DT1␣SOURCE␣(EA)
c) Enable parity or checksum error to be detected	T, Transfer	(G) or (GA) only	$T␣DT1␣←DT2␣SOURCE␣(GA)
d) Initialize the directory of a mass storage device	T, Transfer	(N) or (NA) or (NB)	$T␣DT1␣DESTFL←DT2␣SOURCE␣(NA)
e) Initialize or create a UFD for the current UIC	T, Transfer	(N) or(NA) or (NB)	$T␣DK␣DESTFL←DT1␣SOURCE␣(NA)
f) Combine two or more files into one file	T, Transfer	(W) or (WA) or (WB)	$T␣DK␣DESTFL←DT2␣A, B, C␣(WA)

TABLE A-6

COPY OPERATIONS

PROCESS	USE OPERATION	OPTION	EXAMPLE
1) Copy the contents of a specific device storage block	B, Block copy	None	$B␣DT1␣5Ø,51←DT2␣1Ø1,1Ø2
2) Copy the contents of a specific series (i.e., range) of device storage blocks	B, Block copy	None	$B␣DT1␣5Ø,55←DT2␣1ØØ,1Ø5
3) Copy the contents of one or more blocks and:			
a) initialize the directory of the destination device	B, Block copy	(N)	$B␣DT1␣5Ø-55␣(N)←DT2␣1ØØ-1Ø5
b) initialize the directory and reserve a ↑QSAVE area equal to core size	B, Block copy	(S)	$B␣DT1␣5,6,7␣(S)←DT2␣5Ø,51,52
c) initialize directory and specify size of ↑QSAVE area to be reserved	B, Block copy	(Snn)	$B␣DT1␣5,6,7␣(S32)←DT2␣5Ø,51,52
4) Add (copy) the files on one device to those of a second device	C, Copy	None	$C␣DK←DT1
5) Copy and add files from one device to another device and:			
a) initialize the directory of the destination device	C, Copy	(N)	$C␣DK␣(N)←DT1
b) initialize the destination device directory and reserve a core-sized ↑QAREA	C, Copy	(S)	$C␣DT1␣(S)←DK

TABLE A-6 (Cont.)
Copy Operations

PROCESS	USE OPERATION	OPTION	EXAMPLE
c) initialize the directory and specify size of ↑QAREA	C, Copy	(Snn)	$C␣DT1␣(S32)←DK (S) is not permitted for disk
6) Copy entire contents of a device onto another device. The contents of the source device to replace completely that of the destination device.	C, Copy	(H)	$C␣DT1␣(H)←DT2
7) Copy a paper tape	T, Transfer	(I)	$T␣PP␣(I)←PR

TABLE A-7 PIP Error Messages

Printed Message	Interpretation
ILL TERMINATOR	Command improperly terminated: only RETURN or ALT MODE are legal.
DEVICE NOT IN +.DAT	Return to the monitor and use the A (Assign) command to assign the device involved to a positive .DAT slot.
DEV ILL FOR OPTION FUNCTION	Change function or device.
FILE NAME TOO LONG	Filenames are restricted to a maximum of 6 characters plus a 3-character extension: the user has exceeded this limit.
ILL SWITCH	An optional switch entered in the command is not permitted in the primary operation being performed.
SWITCH CONFLICT	**The combination of optional switches in the command string is illegal.**
NEED DATA MODE	User must enter the proper data mode switch.
CHKSUM ERR	An error has occurred in the transfer of data; retry transfer.
PARITY ERR	File being processed contains a parity error. During T, transfer operation, this message may be accompanied by a printout of the line which contains the error. The user must Correct the indicated line (Use (G) switch in a Transfer, T, operation or the Editor).
S FUNCTION NOT DONE	A segmentation operation requires the entry of an S command followed by a T command. This message indicates the S command was not entered.
TOO FEW FILES TOO MANY FILES	These error messages indicate that the number of files on the destination side of the command does not match the number given on the source side.
INPUT LINE TOO LONG	Input commands are limited to one physical line of 72 characters; the user has attempted to exceed this limitation.
READ-COMP ERR ON BLK:XXXXX	During H mode or Block (B) copy operations, PIP compares the newly written data blocks against the original blocks and outputs this message if they are different.
FILE ALREADY PRESENT	**File under the 'new file name and extension', in a 'R' function, is already present on the device.**

TABLE A-7 (Cont.)
PIP Error Messages

Printed Message	Interpretation
ILL BLK #	Improper block number (i.e., too large, or negative) specified in command.
STRINGS 1-16 ACCEPTED	If a Segment, S, operation divides a file into more than 16 segments, PIP will load segments 1-15 to their respective destination files. All remaining segments are put into the 16th file and this message is printed.
ILL FUNCTION FOR UIC	The operation specified is not permitted at the level of the current user. For example, a standard user (UIC) cannot employ the (R) switch in a copy to system device operation.
UIC NOT IN MFD	Indicates that a File Directory (UFD) has not been established for the current user. The user should employ the PIP, N, command to set up a UFD under his identification code (i.e., UIC).
ILL UIC	User has given an improper code; the UIC must be a 3-character code within angle brackets (i.e., <XXX>).
NEED BLK #	No block number was given in an UPDATE command; enter required octal number.
P VIOLATION	Current user has attempted an operation which violates the established protection code of a UFD or a file.
ILL P CODE	Specified directory code is illegal (i.e., something other than Ø or 1) or the directory just read has an illegal protection code).
SYSBLK NOT ON DEV	Device specified is not the system device.
DISK FULL	Device has no further storage available.
ILL CNT	This error message is printed only in DECtape Directory listings. When output, it will appear immediately after the directory line "SYSTEM BLKS" (for example: Ø SYSTEM BLKS ILL CNT). This message indicates that an illegal number of system blocks have been detected. The user, on detection of this error message, should immediately attempt to transfer any files contained by the DECtape involved onto another DECtape. The files should be transferred one at a time. The faulty DECtape should be initialized to clear the error condition.
ILL UFD ENTRY SIZE or ILL MFD ENTRY SIZE	The size of an entry (i.e., filename or UFD) in the directory involved is illegal. This error indicates that the system is faulty.

TABLE A-7 (Cont.)
PIP Error Messages

Printed Message	Interpretation
NULL FILE NAME ILL	A filename must be specified; a Null-name is not acceptable.
ILL CMD STRUCTURE	The command string entered was not properly ordered or structured.
FILE STRUCTURE CONTAM-INATED	The file structure on the device (i.e., disk) is faulty; this error indicates that the system is faulty.
COMMAND STRING TOO LONG	The command string entered exceeds 72 characters.
TOO MANY FILES OR BLOCKS	The limit of 28_{10} was exceeded.
ILL FUNCTION	The function specified in the command string entered (i.e., first character) is not a legal PIP function.
ILL DEV/UNIT	Illegal device mnemonic specified in command string.
WARNING: BLOCK OCCUPIED	Message output during update operations; it indicates that the user has not deleted the file which contains the bad block (# given by system). The user should delete the file involved and reissue the update operation.
FILE NOT ON INPUT DEVICE	There is no file under the specified name on the directoried input device.

INDEX

BOSS XVM
User's Manual
DEC-XV-OBUAA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

digital

digital equipment corporation

PRINTED IN U.S.A.