UNIVAC
OS/4
Information Interchange Standards

Programmer Reference

HAS "A" + "B"

SPERRY UNIVAC COMPUTER SYSTEMS

7902 Rev. 1

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

Other trademarks of the Sperry Rand Corporation include:

FASTRAND

UNISCOPE

UNISERVO

PAGEWRITER

MATED-FILM

PRINTED IN U.S.A.

UNIVAC OS/4
Information Interchange
Standards
Programmer Reference
UP-7902 Rev.1

December 1973

UPDATING PACKAGE B

File pages as specified below

SECTION	DESTROY FORMER PAGES NUMBERED	FILE NEW PAGES NUMBERED
Cover and Disclaimer	†	†
Page Status Summary	PSS-1 Rev.1	PSS-1 Rev.2
Acknowledgment	Acknowledgment-1	Acknowledgment-1 Rev.1
Contents	Contents-3	Contents-3 Rev.1
Section 1	1 and 2 3 Rev.1 and 4	1 Rev.1 and 2 Rev.1 3 Rev.2 and 4
Section 2	1 and 2 3 and 4 5	1 Rev.1 and 2 3 Rev.1 and 4 5 Rev.1
Section 3	5 and 6 7 and 8 9 and 10 15 and 16 21 and 22 23 and 24 25 and 26	5 Rev.1 and 6 Rev.1 7 Rev.1 and 8 Rev.1 9 Rev.1 and 10 15 and 16 Rev.1 21 and 22 Rev.1 23 Rev.1 and 24 Rev.1 25 Rev.1 and 26

† Destroy old cover and file new cover

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

UNIVAC 9400 System
DOS Information Interchange Standards
Programmer Reference,
UP-7902 Rev. 1

December 1972

UPDATING PACKAGE A

File pages as specified below

SECTION	DESTROY FORMER PAGES NUMBERED	FILE NEW PAGES NUMBERED
Page Status Summary	PSS-1	PSS-1 Rev. 1
Contents	1 and 2	1 and 2 Rev. 1
Section 1	3 and 4	3 Rev. 1 and 4
Section 3	17 and 18	17 Rev. 1 and 18 Rev. 1
	N. A.	18a**

**This is a new page

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

PAGE STATUS SUMMARY

ISSUE: Update B to UP-7902 Rev. 1

The following table lists the status of each page in this document and indicates the update package (if applicable).

Section	Page Number	Page Status	Update Package
Cover/Disclaimer	1	Rev. 1	B
PSS	1	Rev. 2	B
Acknowledgment	1	Rev. 1	B
Contents	1 2 3	Orig. Rev. 1 Rev. 1	 A B
1	1 2 3 4 thru 12	Rev. 1 Rev. 1 Rev. 2 Orig.	B B B
2	1 2 3 4 5	Rev. 1 Orig. Rev. 1 Orig. Rev. 1	B B B
3	1 thru 4 5 thru 9 10 thru 15 16 17, 18 18a 19 thru 21 22 thru 25 26	Orig Rev. 1 Orig. Rev. 1 Rev. 1 Orig. Orig. Rev. 1 Orig.	 B B A A B
User Comment Sheet			
Total 57 pages including covers			

Section	Page Number	Page Status	Update Package

ACKNOWLEDGMENT

The UNIVAC 9400 System provides full ASCII support as defined in the following standards:

- The American National Standard Code for Information Interchange (ASCII), X3.4–1968.
- The American National Standard Perforated Tape Code for Information Interchange, X3.6–1965.
- The American National Standard Recorded Magnetic Tape for Information Interchange, 800 CPI, NRZI, X3.22–1967.
- The American National Standard Magnetic Tape Labels for Information Interchange, X3.27–1969.
- The American National Standard Hollerith Punched Card Code, X3.26–1970. ←

Federal Information Processing Standards Publication 7 (FIPS PUB 7, Supplement to FIPS PUBS 1, 2, and 3) provides further details concerning the implementation and applicability of these media standards. The standards listed above have been approved by ANSI and have been adopted by the Federal Government as Federal Information Processing Standards (FIPS).

CONTENTS

3. ANSI STANDARDS SYSTEM SUPPORT

FIGURES

TABLES

1. INTRODUCTION

1.1. GENERAL

This document describes hardware and software support for the use of information interchange standards with the UNIVAC OS/4 Operating System (OS/4). This document also provides information concerning the hardware and software component support and descriptions of the available commands, instructions, and statements that provide information interchange.

Use of this document assumes a knowledge of the following:

- *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version)
- *UNIVAC 9700 System OS/4 FORTRAN Supplementary Reference, UP-7991* (current version)
- *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version)
- *UNIVAC 9400 System Supervisor Programmer Reference, UP-7689* (current version)
- *UNIVAC 9400/9480 System Operations Handbook Operator Reference, UP-7871* (current version)
- *UNIVAC 9400 System Data Management System Programmer Reference, UP–7629* (current version)
- *UNIVAC OS/4 Data Utility Routine Programmer Reference, UP-7849* (current version)
- *UNIVAC 9400 System FORTRAN Supplementary Reference, UP-7693* (current version)
- *UNIVAC 9700 System OS/4 Assembler Programmer Reference, UP-7935* (current version)
- *UNIVAC OS/4 COBOL Supplementary Reference, UP-7709* (current version)
- *UNIVAC OS/4 Report Program Generator Programmer Reference, UP-7707* (current version)
- *UNIVAC OS/4 Sort/Merge Programmer Reference, UP-7664* (current version)
- *UNIVAC OS/4 Linkage Editor Programmer Reference, UP-7703* (current version)
- *UNIVAC OS/4 Utility and Service Routines Programmer Reference, UP-7713* (current version)

This document is primarily concerned with programming support for the standards approved by the American National Standards Institute (ANSI).

1.2. SYSTEM REQUIREMENTS

The minimum system supporting information interchange requires 49K bytes of main storage. This excludes COBOL support which requires a minimum system of 65K bytes (extended COBOL). Because of increased main storage requirements, the 32K minimum system is unable to support information interchange. Information interchange is supported under the disc operating environment only; tape processors (tape linkers, etc.) do not provide ASCII support either.

ASCII system performance may be slightly degraded by introduction of data translations from one code to the other. The amount of degradation is contingent upon the volume of data input from the card reader and the volume of incoming and outgoing operator console messages. As these I/O devices are relatively slow the impact of additional software translations is negligible.

With the exception of main storage requirements, the hardware configurations reamin the same. The UNIVAC 0768–02 Printer is capable of printing 94 characters (plus space); this includes the full printable graphic set of ASCII characters. Early UNIVAC 9400 System central processor units have been updated by UNIVAC System Field Change Order (FCO) Number 190 to be ASCII-compatible with UNPK and EDIT hardware instructions. Also, UNIVAC System FCO Number 371 must be applied to the hardware to correct a discrepancy in signs generated by the central processor when the system is in ASCII mode. (See your Univac customer engineer to ensure that FCO Numbers 190 and 371 are in effect.) ASCII character compatibility for the console and printer are available in UNIVAC 9400 Systems with serial numbers from 205 up.

UNIVAC OS/4 software remains in Extended Binary Coded Decimal Interchange Code (EBCDIC). Modifications of certain software routines have been made where necessary to provide ASCII sensitivity.

Except for COMREG, which may be either ASCII or EBCDIC, the following are maintained in EBCDIC throughout the system:

- header record fields
- EXTRN/ENTRY records
- field job streams
- file control blocks (FCB)
- physical unit blocks (PUB)
- system information block (SIB)
- job preamble

Both EBCDIC mode and ASCII mode jobs can run concurrently. Table 1–1 lists the codes available for operating certain peripheral devices.

All UNIVAC OS/4 software must be run with EBCDIC files; that is, ASCII must not be declared for printer files or tape files unless explicity specified in this document (for example, by the magnetic tape preparation routine; see 3.9.1).

PERIPHERAL DEVICES	CODES
Card reader	EBCDIC. Software translation to ASCII is provided.*
Card punch	UNIVAC code. Software translation is provided.
Printer**	Any binary - graphic correspondence.
Optical document reader	ASCII or EBCDIC
Communications devices	ASCII or device-dependent codes
Paper tape	ASCII
Magnetic tape	ASCII or EBCDIC
Disc	EBCDIC. No ASCII standards have been adopted; however, ASCII data may be recorded in the current disc environment.

**For data management users, an optional ASCII hardware translate feature is provided with the UNIVAC 0716-02 Card Reader. Incorporation of the UNIVAC 0716–02 Card Reader is usually utilized to enter EBCDIC data into the system. Exceptions to this rule occur when a pure ASCII hardware translate feature is wired into the equipment. With the combination ASCII/EBCDIC hardware translate feature, both data bases are supported. ASCII data is entered only by data management and physical I/O users.*

***Printer device is the UNIVAC 0768–00,01 Printer, available with 63 printable characters, or the UNIVAC 0768–02,03 Printer, available with 94 printable characters.*

Table 1–1. Codes Used with System Peripherals

1.3. CODE DEFINITIONS AND CHARTS

Tables 1–2 and 1–3 provide the code charts for ASCII and EBCDIC, respectiveiy. The following code definitions are relevant to the discussion of character codes presented in this document and shown in Table 1–2.

ASCII – is the name given to the 128-character, seven-bit code, defined in ANSI X3.4–1968, which is assigned to the lower seven bits of an eight-bit code configuration. The bit configuration is read from the high order bit (b_8) to the low order bit (b_1) with b_8 equal to 0.

EBCDIC – is the name given to the 256-character, eight-bit code used as the system code by the UNIVAC 9400 System. The bit configuration is read from the high order bit (b_0) to the low order bit (b_7).

Character translation from ASCII to EBCDIC and from EBCDIC to ASCII is based on the eight-bit code correspondence shown in Table 1–4. Following translation, any character which does not appear as one of the characters listed in Table 1–2 should be restricted from being included on a data interchange tape.

Character data entered through the UNIVAC 0716-02 Card Reader conforms to the same character correspondence. Any Hollerith punched card code with an internal binary representation (when entered in main storage) that does not appear with the ASCII characters listed in Table 1–2 should not be used when creating data interchange tapes.

BIT POSITIONS 4, 3, 2, 1	BIT POSITIONS 7, 6, 5							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	! ①	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[②	k	{
1100	FF	FS	,	<	L	\ ④	l	¦ ⑥
1101	CR	GS	–	=	M	] ②	m	}
1110	SO	RS	.	>	N	^ ①	n	~
1111	SI	US	/	?	O	—	o	DEL

③ ④

⑤

NOTES:

① The following optional graphics can be substituted in the character set:

¬ for ^

¦ for !

② The graphics shown apply to the UNIVAC 0768 Printer Subsystem with drum feature C1344-01 installed and to the console printer shipped with UNIVAC 9400 Systems with serial numbers 205 and up. For UNIVAC 0768 Printer Subsystems without the C1344-01 feature, and for console printers with serial numbers below 205, substitute the following graphics in the charts:

¢ for [

! for]

③ The set of 63 graphic characters (SP is not a printable graphic).

④ Graphics available by use of the type 0768-02 printer, which prints a 94-character set (DEL is not a printable graphic).

⑤ The set of 94 graphic characters.

⑥ For printers other than the type 0768-02 printer, the following substitution is made:

\ for ¦

Table 1–2. ASCII Chart

BIT POSITIONS 4, 5, 6, 7	BIT POSITIONS 0, 1, 2, 3															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE	DS ①		SP	&	–						{④	}④	\④	0
0001	SOH	DC1	SOS ①				/		a	j	~		A	J		1
0010	STX	DC2	FS ①	SYN					b	k	s		B	K	S	2
0011	ETX	DC3							c	l	t		C	L	T	3
0100									d	m	u		D	M	U	4
0101	HT		LF						e	n	v		E	N	V	5
0110		BS	ETB						f	o	w		F	O	W	6
0111	DEL		ESC	EOT					g	p	x		G	P	X	7
1000		CAN							h	q	y		H	Q	Y	8
1001		EM						④	i	r	z		I	R	Z	9
1010					[②	]②	: ⑤	:								
1011	VT				·	$	,	#								
1100	FF	FS		DC4	<	*	%	@								
1101	CR	GS	ENQ	NAK	(	)	—	'								
1110	SO	RS	ACK		+	;	>	=								
1111	SI	US	BEL	SUB	! ③	^ ③	?	"								

④

Table 1–3. EBCDIC Chart (Part 1 of 2)

NOTES:

① DS, SOS, and FS are the control characters for the EDIT instruction (see 2.2.10) and have been assigned for ASCII mode processing so as not to conflict with the corresponding character positions previously assigned in the EBCDIC chart. As these characters are now outside the range as defined in ANSI X3.4-1968, they must not appear in external storage media, such as ANSI standard tapes. This presents no difficulty due to the nature of the EDIT instruction.

② The graphics shown apply to the UNIVAC 0768 Printer Subsystem with drum feature C1344-01 installed and to the console printer shipped with UNIVAC 9400 Systems with serial numbers 205 and up. For UNIVAC 0768 Printer Subsystems without the C1344-01 feature, and for console printers with serial numbers below 205, substitute the following graphics in the charts:

¢ for [

! for]

③ The following optional graphics can be substituted in the character set:

¬ for ^

¦ for !

④ Graphics (including the lowercase alphabet) available by use of the type 0768-02 printer, which prints a 94-character set

⑤ For printers other than the type 0768-02 printer, the following substitution is made:

\ for ¦

Table 1–3. EBCDIC Chart (Part 2 of 2)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
00	0	NUL		12-0-9-8-1	00	0	
01	1	SOH		12-9-1	01	1	
02	2	STX		12-9-2	02	2	
03	3	ETX		12-9-3	03	3	
04	4	EOT		9-7	37	55	
05	5	ENQ		0-9-8-5	2D	45	
06	6	ACK		0-9-8-6	2E	46	
07	7	BEL		0-9-8-7	2F	47	
08	8	BS		11-9-6	16	22	
09	9	HT		12-9-5	05	5	
0A	10	LF		0-9-5	25	37	
0B	11	VT		12-9-8-3	0B	11	
0C	12	FF		12-9-8-4	0C	12	
0D	13	CR		12-9-8-5	0D	13	
0E	14	SO		12-9-8-6	0E	14	
0F	15	SI		12-9-8-7	0F	15	
10	16	DLE		12-11-9-8-1	10	16	
11	17	DC1		11-9-1	11	17	
12	18	DC2		11-9-2	12	18	
13	19	DC3		11-9-3	13	19	
14	20	DC4		9-8-4	3C	60	
15	21	NAK		9-8-5	3D	61	
16	22	SYN		9-2	32	50	
17	23	ETB		0-9-6	26	38	
18	24	CAN		11-9-8	18	24	
19	25	EM		11-9-8-1	19	25	
1A	26	SUB		9-8-7	3F	63	
1B	27	ESC		0-9-7	27	39	
1C	28	FS		11-9-8-4	1C	28	
1D	29	GS		11-9-8-5	1D	29	
1E	30	RS		11-9-8-6	1E	30	
1F	31	US		11-9-8-7	1F	31	
20	32	SP, SPACE			40	64	
21	33	Exclamation point	!	12-8-7	4F	79	
22	34	Quotation mark, dieresis	"	8-7	7F	127	
23	35	Number sign, pound sign	#	8-3	7B	123	
24	36	Dollar sign	$	11-8-3	5B	91	
25	37	Percent sign	%	0-8-4	6C	108	
26	38	Ampersand	&	12	50	80	
27	39	Apostrophe, acute accent	'	8-5	7D	125	
28	40	Opening parenthesis	(	12-8-5	4D	77	
29	41	Closing parenthesis	)	11-8-5	5D	93	
2A	42	Asterisk	*	11-8-4	5C	92	
2B	43	Plus sign	+	12-8-6	4E	78	
2C	44	Comma, cedilla	,	0-8-3	6B	107	
2D	45	Minus sign, hyphen	-	11	60	96	
2E	46	Period, decimal point	.	12-8-3	4B	75	
2F	47	Slash, virgule, solidus	/	0-1	61	97	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 1 of 6)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
30	48	0, zero	0	0	F0	240	
31	49	1, one	1	1	F1	241	
32	50	2, two	2	2	F2	242	
33	51	3, three	3	3	F3	243	
34	52	4, four	4	4	F4	244	
35	53	5, five	5	5	F5	245	
36	54	6, six	6	6	F6	246	
37	55	7, seven	7	7	F7	247	
38	56	8, eight	8	8	F8	248	
39	57	9, nine	9	9	F9	249	
3A	58	Colon	:	8-2	7A	122	
3B	59	Semicolon	;	11-8-6	5E	94	
3C	60	Less than	<	12-8-4	4C	76	
3D	61	Equal sign	=	8-6	7E	126	
3E	62	Greater than	>	0-8-6	6E	110	
3F	63	Question mark	?	0-8-7	6F	111	
40	64	Commercial at symbol	@	8-4	7C	124	
41	65	Capital A	A	12-1	C1	193	+1
42	66	Capital B	B	12-2	C2	194	+2
43	67	Capital C	C	12-3	C3	195	+3
44	68	Capital D	D	12-4	C4	196	+4
45	79	Capital E	E	12-5	C5	197	+5
46	70	Capital F	F	12-6	C6	198	+6
47	71	Capital G	G	12-7	C7	199	+7
48	72	Capital H	H	12-8	C8	200	+8
49	73	Capital I	I	12-9	C9	201	+9
4A	74	Capital J	J	11-1	D1	209	–1
4B	75	Capital K	K	11-2	D2	210	–2
4C	76	Capital L	L	11-3	D3	211	–3
4D	77	Capital M	M	11-4	D4	212	–4
4E	78	Capital N	N	11-5	D5	213	–5
4F	79	Capital O	O	11-6	D6	214	–6
50	80	Capital P	P	11-7	D7	215	–7
51	81	Capital Q	Q	11-8	D8	216	–8
52	82	Capital R	R	11-9	D9	217	–9
53	83	Capital S	S	0-2	E2	226	
54	84	Capital T	T	0-3	E3	227	
55	85	Capital U	U	0-4	E4	228	
56	86	Capital V	V	0-5	E5	229	
57	87	Capital W	W	0-6	E6	230	
58	88	Capital X	X	0-7	E7	231	
59	89	Capital Y	Y	0-8	E8	232	
5A	90	Capital Z	Z	0-9	E9	233	
5B	91	Opening bracket	[	12-8-2	4A	74	
5C	92	Reverse slash	\	0-8-2	E0	224	
5D	93	Closing bracket	]	11-8-2	5A	90	
5E	94	Circumflex	^	11-8-7	5F	95	
5F	95	Underline	—	0-8-5	6D	109	
60	96	Grave accent	`	8-1	79	121	
61	97	Lowercase a	a	12-0-1	81	129	
62	98	Lowercase b	b	12-0-2	82	130	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 2 of 6)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
63	99	Lowercase c	c	12-0-3	83	131	
64	100	Lowercase d	d	12-0-4	84	132	
65	101	Lowercase e	e	12-0-5	85	133	
66	102	Lowercase f	f	12-0-6	86	134	
67	103	Lowercase g	g	12-0-7	87	135	
68	104	Lowercase h	h	12-0-8	88	136	
69	105	Lowercase i	i	12-0-9	89	137	
6A	106	Lowercase j	j	12-11-1	91	145	
6B	107	Lowercase k	k	12-11-2	92	146	
6C	108	Lowercase l	l	12-11-3	93	147	
6D	109	Lowercase m	m	12-11-4	94	148	
6E	110	Lowercase n	n	12-11-5	95	149	
6F	111	Lowercase o	o	12-11-6	96	150	
70	112	Lowercase p	p	12-11-7	97	151	
71	113	Lowercase q	q	12-11-8	98	152	
72	114	Lowercase r	r	12-11-9	99	153	
73	115	Lowercase s	s	11-0-2	A2	162	
74	116	Lowercase t	t	11-0-3	A3	163	
75	117	Lowercase u	u	11-0-4	A4	164	
76	118	Lowercase v	v	11-0-5	A5	165	
77	119	Lowercase w	w	11-0-6	A6	166	
78	120	Lowercase x	x	11-0-7	A7	167	
79	121	Lowercase y	y	11-0-8	A8	168	
7A	122	Lowercase z	z	11-0-9	A9	169	
7B	123	Opening brace	{	12-0	C0	192	+0
7C	124	Vertical line	|	12-11	6A	106	
7D	125	Closing brace	}	11-0	D0	208	–0
7E	126	Overline, tilde	~	11-0-1	A1	161	
7F	127	DEL, delete		12-9-7	07	7	
80	128			11-0-9-8-1	20	32	
81	129			0-9-1	21	33	
82	130			0-9-2	22	34	
83	131			0-9-3	23	35	
84	132			0-9-4	24	36	
85	133			11-9-5	15	21	
86	134			12-9-6	06	6	
87	135			11-9-7	17	23	
88	136			0-9-8	28	40	
89	137			0-9-8-1	29	41	
8A	138			0-9-8-2	2A	42	
8B	139			0-9-8-3	2B	43	
8C	140			0-9-8-4	2C	44	
8D	141			12-9-8-1	09	9	
8E	142			12-9-8-2	0A	10	
8F	143			11-9-8-3	1B	27	
90	144			12-11-0-9-8-1	30	48	
91	145			9-1	31	49	
92	146			11-9-8-2	1A	26	
93	147			9-3	33	51	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 3 of 6)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
94	148			9-4	34	52	
95	149			9-5	35	53	
96	150			9-6	36	54	
97	151			12-9-8	08	8	
98	152			9-8	38	56	
99	153			9-8-1	39	57	
9A	154			9-8-2	3A	58	
9B	155			9-8-3	3B	59	
9C	156			12-9-4	04	4	
9D	157			11-9-4	14	20	
9E	158			9-8-6	3E	62	
9F	159			11-0-9-1	E1	225	
A0	160			12-0-9-1	41	65	
A1	161			12-0-9-2	42	66	
A2	162			12-0-9-3	43	67	
A3	163			12-0-9-4	44	68	
A4	164			12-0-9-5	45	69	
A5	165			12-0-9-6	46	70	
A6	166			12-0-9-7	47	71	
A7	167			12-0-9-8	48	72	
A8	168			12-8-1	49	73	
A9	169			12-11-9-1	51	81	
AA	170			12-11-9-2	52	82	
AB	171			12-11-9-3	53	83	
AC	172			12-11-9-4	54	84	
AD	173			12-11-9-5	55	85	
AE	174			12-11-9-6	56	86	
AF	175			12-11-9-7	57	87	
B0	176			12-11-9-8	58	88	
B1	177			11-8-1	59	89	
B2	178			11-0-9-2	62	98	
B3	179			11-0-9-3	63	99	
B4	180			11-0-9-4	64	100	
B5	181			11-0-9-5	65	101	
B6	182			11-0-9-6	66	102	
B7	183			11-0-9-7	67	103	
B8	184			11-0-9-8	68	104	
B9	185			0-8-1	69	105	
BA	186			12-11-0	70	112	
BB	187			12-11-0-9-1	71	113	
BC	188			12-11-0-9-2	72	114	
BD	189			12-11-0-9-3	73	115	
BE	190			12-11-0-9-4	74	116	
BF	191			12-11-0-9-5	75	117	
C0	192			12-11-0-9-6	76	118	
C1	193			12-11-0-9-7	77	119	
C2	194			12-11-0-9-8	78	120	
C3	195			12-0-8-1	80	128	
C4	196			12-0-8-2	8A	138	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 4 of 6)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
C5	197			12-0-8-3	8B	139	
C6	198			12-0-8-4	8C	140	
C7	199			12-0-8-5	8D	141	
C8	200			12-0-8-6	8E	142	
C9	201			12-0-8-7	8F	143	
CA	202			12-11-8-1	90	144	
CB	203			12-11-8-2	9A	154	
CC	204			12-11-8-3	9B	155	
CD	205			12-11-8-4	9C	156	
CE	206			12-11-8-5	9D	157	
CF	207			12-11-8-6	9E	158	
D0	208			12-11-8-7	9F	159	
D1	209			11-0-8-1	A0	160	
D2	210			11-0-8-2	AA	170	
D3	211			11-0-8-3	AB	171	
D4	212			11-0-8-4	AC	172	
D5	213			11-0-8-5	AD	173	
D6	214			11-0-8-6	AE	174	
D7	215			11-0-8-7	AF	175	
D8	216			12-11-0-8-1	B0	176	
D9	217			12-11-0-1	B1	177	
DA	218			12-11-0-2	B2	178	
DB	219			12-11-0-3	B3	179	
DC	220			12-11-0-4	B4	180	
DD	221			12-11-0-5	B5	181	
DE	222			12-11-0-6	B6	182	
DF	223			12-11-0-7	B7	183	
E0	224			12-11-0-8	B8	184	
E1	225			12-11-0-9	B9	185	
E2	226			12-11-0-8-2	BA	186	
E3	227			12-11-0-8-3	BB	187	
E4	228			12-11-0-8-4	BC	188	
E5	229			12-11-0-8-5	BD	189	
E6	230			12-11-0-8-6	BE	190	
E7	231			12-11-0-8-7	BF	191	
E8	232			12-0-9-8-2	CA	202	
E9	233			12-0-9-8-3	CB	203	
EA	234			12-0-9-8-4	CC	204	
EB	235			12-0-9-8-5	CD	205	
EC	236			12-0-9-8-6	CE	206	
ED	237			12-0-9-8-7	CF	207	
EE	238			12-11-9-8-2	DA	218	
EF	239			12-11-9-8-3	DB	219	
F0	240			12-11-9-8-4	DC	220	
F1	241			12-11-9-8-5	DD	221	
F2	242			12-11-9-8-6	DE	222	
F3	243			12-11-9-8-7	DF	223	
F4	244			11-0-9-8-2	EA	234	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 5 of 6)

ASCII-8		CHARACTER NAME	SYMBOL	CARD PUNCHES	EBCDIC-8		
HEX.	DEC.				HEX.	DEC.	SIGNED NUMBERS
F5	245			11-0-9-8-3	EB	235	
F6	246			11-0-9-8-4	EC	236	
F7	247			11-0-9-8-5	ED	237	
F8	248			11-0-9-8-6	EE	238	
F9	249			11-0-9-8-7	EF	239	
FA	250			12-11-0-9-8-2	FA	250	
FB	251			12-11-0-9-8-3	FB	251	
FC	252			12-11-0-9-8-4	FC	252	
FD	253			12-11-0-9-8-5	FD	253	
FE	254			12-11-0-9-8-6	FE	254	
FF	255	E0, Eight ones		12-11-0-9-8-7	FF	255	

Table 1–4. Correspondence Between ASCII-8, EBCDIC-8, and Punched Card Code (Part 6 of 6)

2. PROGRAMMING

2.1. GENERAL

The UNIVAC OS/4 Operating System (OS/4) provides software support for the ANSI standards. The programming approach is to maintain the current EBCDIC operating system software base while providing an internal ASCII processing capability for the user. This approach satisfies the objectives established in FIPS PUB 7. Where an interface exists between the user's ASCII base and the EBCDIC based operating system, the minimal translations are transparent to the user.

A program running in ASCII mode is processed with the program status word (PSW) set to ASCII (PSW bit 12 = 1); this engages the processor in the ASCII state for special treatment of the UNPK and EDIT hardware instructions. Card images retrieved by the GETCS macro instruction are translated into the ASCII character set. Operator console messages and responses are also translated for an ASCII object program. ASCII files may be defined to the system by way of standard job control statements. Data management, upon dynamic recognition of an ASCII file, provides mode sensitive processing. This processing includes the handling of ANSI standard labels for magnetic tape and translating where necessary for reader and punch activities. Print files of ASCII characters, having been defined by job control statements, are satisfied by the system providing the appropriate load code.

ASCII object programs are generated by a language processor other than the assembler through use of special parameters introduced by a PARAM job control statement. This causes ASCII literals to be generated instead of the current EBCDIC literals. Compatible ASCII EDIT masks are generated; object time subroutines are also sensitive to the program mode. Translation facilities are provided with the language processors.

When a program generated in ASCII mode is loaded into main storage, it engages the ASCII processing mode for that particular job.

The user can capitalize on the dual code potential of the system by running both EBCDIC and ASCII based programs concurrently. The mode of one job does not interfere with the processing of another job operating in the opposite mode.

2.2. USER CONSIDERATIONS

Programming considerations which the user is concerned with when programming support for ANSI standards are discussed in this section. Software components of OS/4 which support ANSI standards are discussed in Section 3.

2.2.1. ASCII Overhead

The ASCII overhead in the supervisor is optional and can be excluded from the user environment during system generation (see 3.2.1).

2.2.2. ASCII Object Code

When ASCII object code is needed, one must assemble or compile the source program declaring the ASCII mode. Once in object code, the ASCII program defines that ASCII processing is called for every time that it is loaded into main storage for execution.

2.2.3. Tape Translation

The data utility package will include a provision for translating data management format tapes from one mode to the other.

2.2.4. Tape Files and Code Sensitivity

The mixture within a given job step of ASCII and EBCDIC files is permitted. For example, data management is code sensitive during the time the OPEN and CLOSE transient routines process user tape files. Data management recognizes the file mode which was defined through job control. Following the standard label processing, tape blocks are transferred directly to and from without data translation, regardless of the indigenous mode of the program or the tape file's data base. It is the responsibility of the user to invoke whatever data translations that are required when processing a file whose mode is different from that of the object code. For example, the need for data translation may first be realized during user label processing of a complementary mode tape file.

Tape volume checking is also enhanced to recognize ASCII labels in addition to the current volume confirmation which is performed for EBCDIC tape volumes.

2.2.5. Data Translation

Translation procedures and high level language facilities are provided for data translation from ASCII to EBCDIC and EBCDIC to ASCII. These features are discussed in 3.2.2.

2.2.6. Printer File Sharing

Users sharing the printer with more than one print file within a job step can do so only if they maintain the same mode (either EBCDIC or ASCII). Otherwise, the last file to be defined causes the current load code to be issued. For this reason, the practice of sharing printers is not recommended.

2.2.7. FORTRAN and Report Program Generator (RPG) Mixed Files

Because of the data-sensitive nature of the I/O processing inherent to RPG compiler output, processing both ASCII and EBCDIC tapes during the same job step is prohibited. Within FORTRAN, a facility is provided for processing tapes comprised of data consisting of a different code from that of the object program (for example, processing EBCDIC files in an ASCII program).

2.2.8. Collating Sequence

Note that the collating sequence of the ASCII character set differs vastly from the collating sequence of the EBCDIC character set. The appropriate collating sequence is ensured by presenting the desired character set when using the sort/merge program. Caution must be exercised when processing different mode tapes. Comparison for magnitude of two character strings yields different results when in an ASCII base rather than an EBCDIC base.

2.2.9. PACK and UNPK Instructions

Caution must be exercised when dealing with decimal strings in an ASCII environment rather than the EBCDIC environment. PACK and UNPK hardware instructions reverse the juxtapostion of the numeric and sign digits in the least significant byte of a decimal string; therefore, additional sign processing is introduced when considering decimal strings in the ASCII environment. ←

In the EBCDIC environment, when an unsigned decimal string such as 622 is entered into main storage, it appears internally in unpacked format as F6F2F2. When packed, it becomes 622F, which is interpreted by the hardware as a positive integer (+622). Because of the differences between ASCII and EBCDIC, the same processing does not result in an ASCII based system. If the same unsigned decimal string (622) were processed in an ASCII based system, it would appear internally in unpacked format as 363232. When packed, it becomes 6223. It is unlikely that the 3 in the sign position would be interpreted by the hardware as a positive sign. Therefore, it is of paramount importance to ensure that the proper sign is forced using this field in a compare or arithmetic instruction.

The signs generated by the central processor in ASCII mode have been modified as follows:

1100 +

1101 –

Thus, the sign conventions are the same regardless of ASCII or EBCDIC mode. The UNPK instruction has been modified to fill the zone portion of the byte with a 3 when in ASCII mode. For example, 622C in unpacked format is 3632C2, which requires additional editing (the sign must be changed to 3 zone to retain all positions of the unpacked field in ASCII). The UNPK instruction continues to fill the zone portion of the byte with an F when in EBCDIC mode.

For additional information concerning the PACK and UNPK instructions, refer to the *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version) or the *UNIVAC 9700 System OS/4 Assembler Programmer Reference, UP-7935* (current version). ←

2.2.10. EDIT Control Characters

EDIT control characters differ in hexadecimal representation between ASCII and EBCDIC modes as follows:

CONTROL CHARACTER	EBCDIC	ASCII
DS (digit select)	20	80
SOS (significant start)	21	81
FS (field separator)	22	82

NOTE: At present, these control characters have no printable graphics.

The EDIT control characters are recognized according to the hexadecimal values recorded above for the ASCII and EBCDIC modes.

For additional information concerning the EDIT instruction and EDIT control characters, refer to the *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version) or the *UNIVAC 9700 System OS/4 Assembler Programmer Reference, UP-7935* (current version). ←

2.2.11. Assembling ASCII Modules

Conceivably an executable program that is comprised of many parts may be in part sensitive to ASCII mode. Consider the case of an assembled program whose many modules have been bound together using the linkage editor. It is possible that not all of the modules are ASCII-mode sensitive. That is, if a module does not execute calls on the supervisor macro instructions, GETCS or OPR, and does not process UNPK or EDIT hardware instruction, the module is mode insensitive. Therefore, it may execute similarly in either code base, A user thus considers assembling, in ASCII mode, only those modules which are mode sensitive. This capability reduces the effort when converting EBCDIC programs to ASCII and permits the user to maintain both versions of a given program. Caution must be exercised, however, when processing in this fashion. For example, character strings should be closely scrutinized when they are passed to or from a module which purports to be mode insensitive; it would be possible to erroneously refer to character data which might have been generated using the complementary mode.

2.2.12. Numeric Overpunch

Although overpunching of numeric fields is a user practice of long standing, eliminating this practice from information interchange standards would also eliminate ambiguous results. Therefore, data should be constrained to a stream of ASCII characters; numeric data will be expected to be provided with a discrete sign character.

The processing of separate signed data is managed internally by the UNIVAC 9400 System compilers. Assembler users must address the problem presented by separate sign characters within their own code. In order to process numeric data with a separate sign character, the sign character must be interpreted and a resultant sign F, C (+), or D (-) must be jammed into the sign field portion of the numeric field. The standard procedure for processing numeric string data (leading or trailing) is:

1. Test sign character for minus (—); if minus, generate a hexadecimal sign of D.

2. All other sign representations are interpreted as positive (+); a hexadecimal sign of C results.

The external representation of the numeric string +123 would be the appropriate Hollerith punched card code for card data; X'2B313233' if resident on an ASCII tape.

The internal ASCII representation of the numeric string +123 depends on the particular language processor and the point in time when the string is accessed. The string would be represented as X'2B313233' and would be packed and edited for proper sign before being presented in any decimal arithmetic. The resulting decimal constant would be X'123C'.

In a program where mixed mode data exists, EBCDIC data with conventional overpunched signs is not restricted. It is the user's responsibility to differentiate between the ASCII data and the EBCDIC data and their respective processing requirements.

The alternate sign convention defined for the UNIVAC 9400 System has no relationship to PSW bit 12=1. The central processor recognizes the alternate signs A (+) and B (-) regardless of the processor mode. With the incorporation of system FCO numbers 190 and 371 (see 1.2), the signs generated by the processor will be C (+) and D (-) regardless of ASCII or EBCDIC mode. The UNPK and EDIT instructions, however, are mode sensitive and react according to the PSW bit 12 setting.

2.2.13. Relocatable Loader

The supervisor generated with ASCII sensitivity must be resident whenever an attempt is made to load an ASCII program for execution; otherwise, an error message will result.

ASCII programs are required to reside in a relocatable load library; programs residing in the absolute execution area will not activate the ASCII state when loaded.

2.3. MISCELLANEOUS CONSIDERATIONS

The current printing devices print a 63-character plus space subset of the ASCII code. Unprintable characters continue to be processed as is done currently. The full 94-character graphic set of ASCII is supported by the UNIVAC 0768–02, 03 printer. Either hexadecimal constants are specified within the assembler, or mismatches will occur.

Since data may be output in any form by the user, no validity checks are performed by the system. Ensuring that no special binary forms or overpunch sign data are output to compromise a standard interchange tape is responsibility of the user.

3. ANSI STANDARDS SYSTEM SUPPORT

3.1. GENERAL

This section reflects functional changes in the individual software areas which are included to support information interchange standards. Whenever an object program in ASCII mode communicates with the EBCDIC based system and literal values are passed, they are translated by the software routine called. Those software areas in which ASCII system support is provided are:

- Supervisor
- Data Management
- Data Utilities
- Job Control
- Language Processors
- Utility and Service Routines

3.2. SUPERVISOR

When an ASCII program is introduced into the system for execution, the supervisor must be generated for ASCII support and must be resident. The same is true for an EBCDIC program which accesses an ASCII data file or calls on ASCII translation facilities (see 3.2.2). Otherwise, an error message occurs and the job is terminated.

Macro instructions and operator commands which are components of the supervisor and are used to implement and control a program in ASCII mode are described in the following paragraphs.

3.2.1. ASCII Supervisor Generation

When generating the supervisor to include ASCII support, the SYSTEM macro instruction is used to describe to the system certain facilities which the supervisor must provide. To include the code which provides the ASCII/EBCDIC capability, the following keyword parameters are specified.

LABEL	↹ OPERATION ↹	OPERAND
	SYSTEM	[,ASCII] ...

The above format shows only the parameters required to include the ASCII/EBCDIC capability in the supervisor.

The ASCII to EBCDIC and EBCDIC to ASCII translation facilities are resident in the supervisor (see 3.2.2). Depending on the nature of the call, either resident SVC code will be activated, or a call will be made on a transient routine.

Card images which are retrieved through use of the supervisor macro instruction GETCS are translated from EBCDIC to ASCII when routed to an ASCII calling program.

The edited alphanumeric margin, which is provided for the abnormal termination dump (ABEND), will represent the internal ASCII literals. When EBCDIC and ASCII literals are resident in main storage within a single problem program, only the ASCII characters are edited for the margin.

3.2.2. Translation Macro Instructions

Two macro instructions have been designed for translating data from ASCII to EBCDIC and from EBCDIC to ASCII. Both macro instructions generate calls on SVC code. If an ASCII supervisor is not resident, the translation macro is ignored and control is returned to the calling program. By issuing a TASV macro instruction (see 3.2.3) prior to a translation macro, this condition is prevented from occurring.

Character translation is performed in the area which contains the characters to be translated. Thus, the original characters are replaced by the translated characters.

Due to critical timing considerations, translation requests for character strings which exceed 141 characters result in processing by a transient routine. Requests for character strings less than 141 characters are handled by the SVC code.

3.2.2.1. ASCII TO EBCDIC (AETRAN)

The ASCII-to-EBCDIC-translate macro instruction (AETRAN) translates the 128 ASCII characters to the corresponding EBCDIC characters, plus the 128 additional characters listed in Table 1–4.

LABEL	OPERATION	OPERAND
	AETRAN	{string-addr / (1)}, {string-length / (0)}

Positional Parameter 1

string-addr — the symbolic address of the character string to be translated.

(1) — indicates that register 1 has been loaded with the address of the character string to be translated.

NOTE: If register 1 contains binary zeros on input, the address of the ASCII to EBCDIC translate table in the supervisor will be returned in register 1 upon output.

Positional Parameter 2

string-length — an integer indicating the length (in bytes) of the character string to be translated.

(0) — indicates that register 0 has been loaded with the number of bytes to be translated.

Examples:

```
LABEL      OPERATION      OPERAND
1          10     16
           AETRAN  FILE,(0)
           AETRAN  (1),128
```

3.2.2.2. EBCDIC TO ASCII (EATRAN)

The EBCDIC-to-ASCII-translate macro instruction (EATRAN) translates the EBCDIC characters to the corresponding ASCII characters, plus the 128 additional characters listed in Table 1–4.

LABEL	OPERATION	OPERAND
	EATRAN	{string-addr / (1)}, {string-length / (0)}

Positional Parameter 1

string-addr — the symbolic address of the character string to be translated.

(1) — indicates that register 1 has been loaded with the address of the character string to be translated.

NOTE: If register 1 contains binary zeros on input, the address of the EBCDIC to ASCII translate table in the supervisor will be returned in register 1 upon output.

Positional Parameter 2

string-length — an integer indicating the length (in bytes) of the character string to be translated.

(0) — indicates that register 0 has been loaded with the number of bytes to be translated

Examples:

```
LABEL      OPERATION      OPERAND
1          10     16
           EATRAN  FILEA,128
           EATRAN  (1),(0)
```

3.2.3. TASV Macro Instruction

The TASV (test-for-ASCII-supervisor) macro instruction is available to test whether an ASCII supervisor is resident in main storage. The format of the TASV macro instruction is:

LABEL	OPERATION	OPERAND
	TASV	{symbol / (15)}

Positional Parameter 1

symbol — specifies the label of the instruction to which the program branches if an ASCII supervisor is not found resident.

(15) — specifies register 15, which contains the address of the instruction to which the program branches if an ASCII supervisor is not found resident.

NOTE: If no operand is supplied, control is returned to the user at the next instruction coded in-line.

Example:

```
LABEL      OPERATION      OPERAND
1          10             16
           TASV           ERREXIT
           TASV           (15)
```

Operational Considerations:

The following console message will be produced before exiting via the error path:

LP01 ASCII EXEC NOT RESIDENT

The test for the ASCII supervisor is achieved by interrogating the systems information block (SIB) as follows:

```
LABEL      OPERATION      OPERAND
1          10             16
           GETADR         SIB
           TM             SB$CHR+2(R1$),JB$ASCII
           BZ             ERROR
```

3.2.4. Test-for-ASCII-Problem-Program Mode

The following instructions may be used if the user desires to determine the current mode of his problem program.

```
LABEL      OPERATION      OPERAND
1          10         16
           GETADR   JCB
           TM       JB$PSW+1(R1$),X'08'
           BO       ASCII path
```

3.2.5. The Communication Area

The communication area (COMREG) within the problem program's preamble can be conditioned for ASCII mode by specifying ASC as positional parameter 3 in the SET COMREG statement. In this way, messages which are processed through use of the supervisor macro instructions GETCOM and PUTCOM may be consistent with the mode of the calling program.

The following format for the SET COMREG statement is used only when conditioning the communication area for ASCII mode. For additional information concerning the SET COMREG statement, see the *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version).

The format of the SET COMREG statement when character constants are specified in ASCII is:

// SET COMREG, character-string, ASC

Positional Parameter 3

ASC — indicates that the character string specified by positional parameter 2 is to be stored in the communication region of the job preamble using ASCII character representation.

3.2.6. Operator Communications

ASCII system support which affects operator communications is described in the following paragraphs. All other operator communications are described in the *UNIVAC 9400 System Supervisor Programmer Reference, UP-7689* (current version) and the *UNIVAC 9400/9480 System Operations Handbook Operator Reference, UP-7871* (current version).

3.2.6.1. ALTER COMMAND

When program alterations which are character-ALTER commands (C'ccc...') are entered from the operator's console, the job number must be specified as positional parameter 1 when dealing with an ASCII program. Unless the job number parameter is specified, character-ALTER commands are applied using EBCDIC characters. The processing of hexadecimal alterations is not affected.

3.2.6.2. DISPLAY COMMAND

When printing at the system's console selected areas of main storage which are within the boundaries of an ASCII program, the job number must be specified as positional parameter 1 if character representation is requested (CLn). Otherwise, EBCDIC character constants are assumed; if ASCII characters are incorrectly displayed, the results will be garbled and incoherent. When accessing EBCDIC characters within an ASCII area, positional parameter 1 should not be specified. The easiest method for displaying mixed EBCDIC and ASCII character data is to request hexadecimal representation. Note that current rules regarding positional parameter 1 still apply when EBCDIC is displayed.

3.2.6.3. OPR MACRO INSTRUCTION

Console messages and replies which are transmitted by way of the OPR supervisor macro are translated to EBCDIC for output and to ASCII upon input when routed to or from an ASCII calling program.

3.3. DATA MANAGEMENT

Data management facilities are provided to recognize a file which is defined by job control to be ASCII (see 3.5) and interact accordingly. The following paragraphs reveal the activities involved for each peripheral device.

NOTE: Regardless of the mode of the file or of the program, all filenames defined by a define-the-file (DTF) declarative macro instruction are in EBCDIC after the OPEN macro instruction is called. When ASCII filenames are recognized, the OPEN macro instruction causes the ASCII fields within the DTF definition to be translated to EBCDIC (these fields may exist in EBCDIC originally).

For detailed information concerning data management facilities, refer to *UNIVAC OS/4 Data Management System Programmer Reference, UP-7629* (current version).

3.3.1. Magnetic Tape

Facilities are provided for creating and processing ANSI standard labels. Optional labels which conform to the ANSI standard are passed to the user if not processed directly. The ANSI standard label processing is provided in addition to the current EBCDIC label processing (see Table 3–1). ASCII tapes may be unlabeled, but will not be accepted if nonstandard labels are presented.

LABEL TYPE	CODE BASE	REMARKS
Unlabeled	EBCDIC or ASCII	
Nonstandard Label	EBCDIC	Not permitted for ASCII code tapes.
Standard Label	EBCDIC	See Note 2.
ANSI Standard Label	ASCII	Not supported for EBCDIC.

NOTES:

1. The above tapes are supported by data management system.
2. Standard as defined in *UNIVAC OS/4 Data Management System Programmer Reference, UP-7629* (current version).

Table 3–1. Classification of Tape Labels Supported by Data Management

Table 3–2 illustrates the handling of the ANSI standard labels supported by the operating system. Table 3–3 illustrates the handling of the ANSI standard labels supported by the user.

NAME	IDENTIFIER AND NO.	ACTION BY SYSTEM	
		INPUT	OUTPUT
Initial volume label	VOL1* (all others prohibited)	Read and process	Construct
File header label(s)	HDR1*	Read and process	Construct
	HDR2 to HDR9	Bypass	Bypass
End of Volume Label(s)	EOV1*	Read and process	Construct
	EOV2 to EOV9	Bypass	Bypass
End of File Label(s)	EOF1*	Read and process	Construct
	EOF2 to EOF9	Bypass	Bypass

*These labels are required; the others are optional.

Table 3–2. ANSI Standard Tape Labels Supported by the Operating System

NAME	IDENTIFIER AND NO.	ACTION BY SYSTEM	
		INPUT	OUTPUT
User volume header label(s)	UVL1 to UVL9	Bypass	Bypass
User file header label(s)	UHLa ①	Read and pass to user ②	Write if requested ②
User end of file label(s)	UTLa ①	Read and pass to user ②	Write if requested ②

NOTES:

① a = any alphanumeric character.

② No action by the system unless user label processing is defined; use of these labels is optional.

Table 3–3. User Supported ANSI Standard Tape Labels

The standard label describes four record formats: fixed-length records (F-type), variable-length records specified by a decimal length (D-type), variable-length records specified by a binary length (V-type), and undefined records (U-type). V-type records are not preferred and are only acceptable upon agreement between the interchange parties; therefore, they are not supported in OS/4 for ASCII files. ASCII tape data files which specify variable-length records are interpreted as D-type records.

Variable-length record processing remains static, as far as the user is concerned. When ASCII variable records (D-type) are processed by data management, they are converted to V-type format for internal processing. Therefore, if the length fields are currently being defined by existing programs, there will be no apparent change since the decimal fields are converted from binary upon output.

- Fixed-Length Records

  All records in the file are of the same length. No indication of the length is required within the file.

- Variable-Length Records

  When the records of a file are not all the same length, the length is recorded in the first four character positions of the record as four ASCII numeric characters.

  NOTE: Fixed- or variable-length records may be blocked or unblocked.

- Undefined Records

  If records do not meet the definitions of variable or fixed-length records, they are undefined. The use of this format requires the agreement of interchange parties.

Checkpoint records, according to the label standard, are extraneous and considered nonstandard. Therefore, the introduction of checkpoint records on a standard tape is assumed upon agreement of the interchange parties. The format, composition, and processing of checkpoint records will remain unchanged.

3.3.1.1. FEATURES REQUIRING AGREEMENT OF THE INTERCHANGE PARTIES

The following optional features requiring agreement of the interchange parties may be included with magnetic tape handling of ASCII files.

- Block Sequence Indicator (BSI)

  A one-digit revolving counter (that is, 1, 2,..., 8, 9, 0, 1, 2 ....) may be inserted as the first character of each block as a precaution against hardware malfunction that would cause a block to be skipped or, in effect, read twice.

- Buffer Offset (BUFOFF)

  At the option of the user, each data block may be preceded by a buffer offset field (from 1 to 99 characters in length) which contains additional information. If the user specifies variable-length records, data management inserts the block length in a buffer offset of four character on output and checks this length on input against the length of the block read in. Any other use of the buffer offset is ignored and bypassed on input and is not written on output.

- Block Padding

  Computer systems that read and write tapes in multiples of a given word size are permitted to pad each block to a word multiple by adding circumflex characters (5/14).

  – Any padding present on input is stripped off by data management.

  – No padding is supplied by data management on output.

  – All padding must be circumflex characters (5/14) and may occur only at the end of a block.

- Block Length

  The standard establishes the maximum block length at 2048 characters. Larger block lengths may be used if interchange parties agree.

  If the length-checking feature is to be used with variable records, the block length must be representable with four characters or less; that is, block length must be less than 9999.

Figures 3–1 and 3–2 illustrate the ASCII tape block structure for input and output, respectively.

3.3.1.2. DYNAMIC MODE SPECIFICATION

The mode of each data management file is determined dynamically during execution of the OPEN macro instruction by means of the control stream definition for that file. To process the file as an ASCII file, the following LFD statement must be specified in the control stream:

```
// LFD    filename,,,,ASC
```

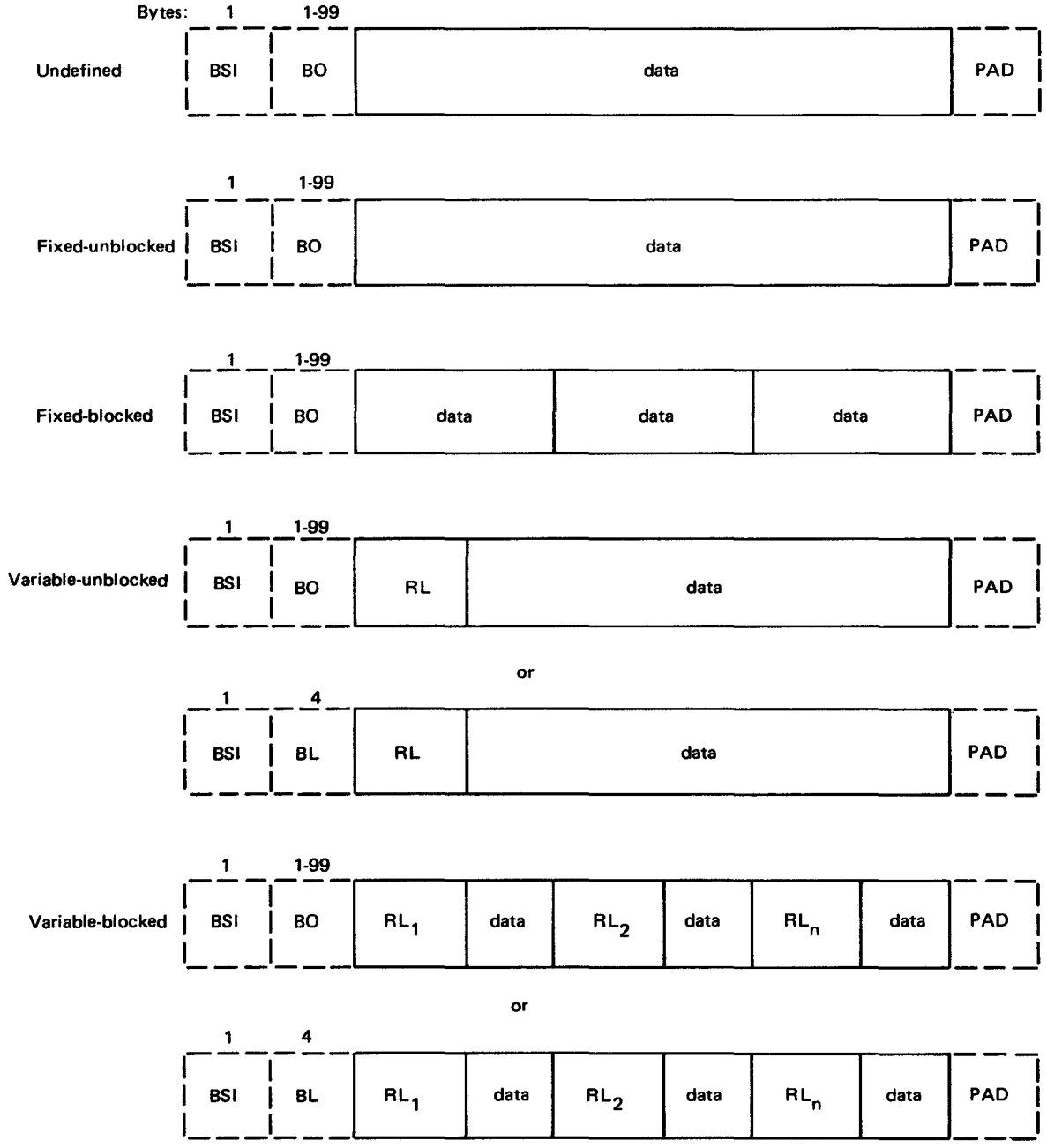
If different modes are specified by the LFD statement and the DTFMT macro instruction, the necessary changes in the DTF will be made when the OPEN macro instruction is executed to ensure proper processing of the file. A message is always printed on the console when the mode specified by the LFD statement and the mode specified by the DTFMT macro instruction differ.

Note that if the DTFMT macro instruction specifies the fixed-length, unblocked record format and the ASCII=YES keyword parameter, and the LFD statement does not specify the ASC parameter, then the RCSZ=n parameter must be specified in the DTFMT macro instruction so that the OPEN macro instruction can properly set the block size of the file. The reason for specifying RCSZ=n (for input files) is that the BKSZ specification for the EBCDIC file does not include an allowance for padding or buffer offset (see 3.3.1.1) that may have been included in the definition of the ASCII file.

3.3.1.3. DECLARATIVE MACRO INSTRUCTION DTFMT

The declarative macro instruction DTFMT is required for both input and output files. The following format of the DTFMT macro instruction shows the keyword parameters applicable to defining an ASCII tape file only. All other keyword parameters for the DTFMT macro instruction are described in the *UNIVAC OS/4 Data Management System Programmer Reference, UP-7629* (current version). A description of each keyword parameter for defining an ASCII tape file follows the format.

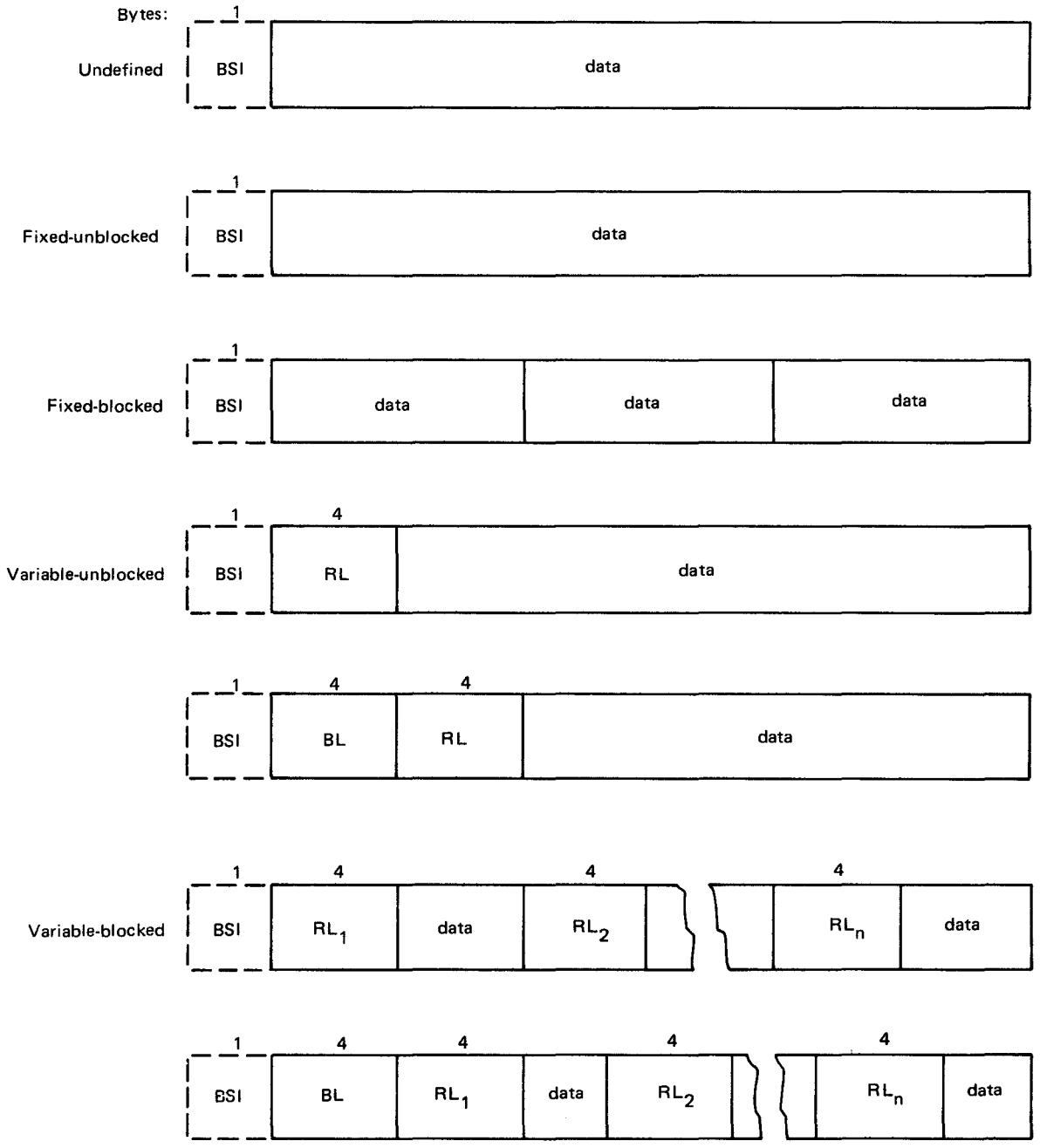
LABEL	♭ OPERATION ♭	OPERAND
filename	DTFMT	[,ASCII=YES] [,BUFOFF={n | 0}] [,LENCHK=YES]



NOTES:

1. Dashed lines indicate optional features which require agreement of the interchange parties.
2. BSI = Block sequence indicator
3. PAD = Block padding
4. BO = Buffer offset
5. BL = Block length (four decimal characters required)
6. RL = Record length (four decimal characters required)
7. RL_n = Record length of n^{th} record

Figure 3–1. ASCII Block Structure, Input



NOTES:

1. Dashed lines indicate optional features which require agreement of the interchange parties.
2. BSI = Block sequence indicator
3. BL = Block length (four decimal characters required)
4. RL = Record length (four decimal characters required)
5. RL_n = Record length of n^{th} record

Figure 3–2. ASCII Block Structure, Output

- ASCII Declaration

  To specify that ASCII processing and block structure (see Tables 3–3 and 3–4) are to be used for this file, include the following keyword parameter:

  ASCII=YES

- Buffer Offset

  If a buffer offset is to be used with ASCII tape files, the following keyword parameter is specified:

  BUFOFF=$\left\{ \begin{matrix} n \\ 0 \end{matrix} \right\}$

  where:

  n – specifies the length in characters of the buffer offset (1 to 99)

  0 – specifies for input files that no buffer offset is to be inserted.

  For input files, the buffer offset is bypassed and ignored by data management, except for variable records when the buffer offset contains the block length. In this case data management will check block length if the user specifies LENCHK=YES and BUFOFF=4 parameters. The buffer offset field is never passed to the user.

  If BUFOFF=4 for variable-length records in output files, data management supplies the block length to the buffer offset. Buffer offset is not allowed for output in any other instance.

- Length Check

  This keyword parameter is used for input files to verify the block length in the buffer offset. The format is:

  LENCHK=YES

  This specifications is ineffective unless BUFOFF=4 and RCFM=VARBLK or VARUNB.

  NOTE: For output files, the block length is placed in the buffer offset field if BUFOFF=4.

The following describes the effect of ASCII tape files on keyword parameters which have been previously described in the data management manual (see 3.3) for the DTFMT declarative macro instruction.

- Block Numbering

  The BKNO=YES keyword parameter is used to specify that the block sequence indicator (see 3.3.1.1) is to be inserted in front of each block.

- Block Size

  The block size keyword parameter (BKSZ=n) specifies maximum block size. Maximum block size must include any buffer offset or padding.

- Tape Labels

  For ASCII tape files, the tape may have standard labels or may be unlabeled. The formats are:

  – FLBL=STD

    Standard labels VOL1, EOV1, HDR1 and EOF1 are processed as specified in ANSI X3.27–1969.

NOTE: The HDR2 label is not supported for ASCII tape files.

– FLBL=NO

This keyword parameter is used if the tape is unlabeled.

- Current Record Pointer and Work Area Processing

If data management is to bypass the buffer offset for input files, either IORG=(r) or WORK=YES must be specified.

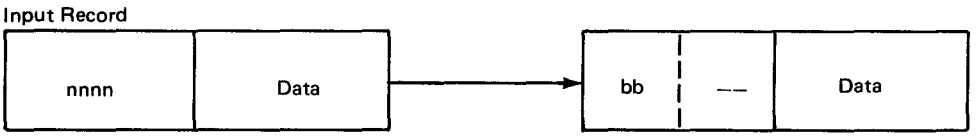
- Record Format

For ASCII tape files, this keyword parameter specifies that D-type records are to be processed. The options for ASCII tape files are coded as follows:

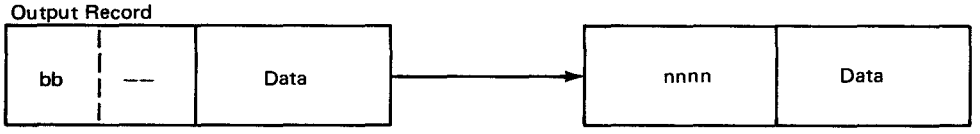
– RCFM=VARBLK Variable length, blocked

– RCFM=VARUNB Variable length, unblocked

For input files, each record begins with a four-byte field which specifies the length of the record as four decimal characters (nnnn). This field is converted to a two-byte binary value (bb) and stored in the most significant two bytes of the field.



For output files, the user supplies the record length as a two-byte binary value (bb) in the most significant two bytes of the record. Data management converts this value to four decimal characters (nnnn).



NOTE: If the user wants the block length for the variable-length record format to be processed by data management, either RCFM=VARBLK or RCFM=VARUNB, and BUFOFF=4 (and for input files, LENCHK=YES) must be specified.

V-type records are not supported for ASCII tape files.

- Special Label Handling

The LBAD=symbol keyword parameter specifies the address of the user routine which processes header or trailer labels. EBCDIC information is exchanged between data management and the user during processing of LBAD. For ASCII, this data is treated as hexadecimal constants. Note that if the characters O, E, F, or V are passed to indicate the end of the OPEN transient routine, the end-of-file condition, or the end-of-volume condition, these characters become X'D6', X'C5', X'C6', or X'E5', respectively.

Note also that no check of user-written labels is made to ensure conformance with ANSI X3.27–1969.

Example:

```
LABEL      OPERATION      OPERAND                                    72
1          10             16
* ASCII TAPE FILE DEFINITION - SAMPLE
XAMPLE     DTFMT BKNO=YES,                                          X
                 BKSZ=1600,                                         X
                 IOAI=AREABASE,                                     X
                 EOFA=ENDRUN,                                       X
                 ASCII=YES,                                         X
                 BUFOFF=4,                                          X
                 LENCHK=YES,                                        X
                 FLBL=STD,                                          X
                 RCFM=VARBLK,                                       X
                 LBAD=ERRO
```

3.3.2. Card Reader, Punch, and Printer

The data management card reader package provides translation to the ASCII character set when card data is input to an ASCII file. The translation to ASCII shall be through a software translate for all card readers except the UNIVAC 0716-02 Card Reader. If on this reader, ASCII or dual hardware translate feature is included, the translation to ASCII will be provided by the hardware. This results in an appreciable time savings.

The data management card punch package accepts ASCII character strings when processing an ASCII file. The punched output code remains the same.

The following format of the DTFCD declarative macro instruction shows the keyword parameter necessary to provide translation to the ASCII character set. A description of the keyword parameter follows the format.

LABEL	OPERATION	OPERAND
filename	DTFCD	[,ASCII=YES]

ASCII Conversion

This keyword parameter is used to specify internal processing in ASCII.

ASCII=YES

When using the UNIVAC Type 0711 Card Reader, 600 card-per-minute (cpm), this keyword parameter must be specified if it is desired to translate the card data into ASCII code.

When using the UNIVAC Type 0604 Row Punch Subsystem, this keyword parameter must be specified if internal processing is in ASCII and punched card output is desired. ASCII is first translated to EBCDIC and then punched in compressed code.

On the printer subsystems, this keyword parameter should be specified if internal processing is in ASCII. This specification indicates that an ASCII load code is expected.

The MODE=STD keyword parameter must be specified when ASCII=YES is specified.

If the ASCII conversion is not desired, the MODE keyword parameter will determine the translate (EBCDIC or binary).

Examples:

```
LABEL     OPERATION     OPERAND                                          72
1         10            16
* ASCII   READER  FILE DEFINITION - SAMPLE                              X
*                                                                       X
CARDIN    DTFCD  IOAI=AREAI,                                            X
                 BKSZ=80,                                               X
                 EOFA=LASTCD,                                           X
                 RCFM=FIXUNB,                                           X
                 TYPE=INPUT,                                            X
                 WORK=YES,                                              X
                 ASCII=YES
```

```
LABEL     OPERATION     OPERAND                                          72
1         10            16
* ASCII   PRINTER FILE DEFINITION - SAMPLE                              X
*                                                                       X
PARTLST   DTFPR  IOAI=OUTI,                                             X
                 BKSZ=132,                                              X
                 RCFM=UNDEF,                                            X
                 CTLCHR=YES,                                            X
                 PRAD=2,                                                X
                 PRTOV=YES,                                             X
                 RCSZ=(10),                                             X
                 ASCII=YES
```

3.3.3. Optical Document Reader (ODR)

The ODR currently conforms to interchange standards when the ASCII optional feature is requested from the manufacturer instead of the EBCDIC mode.

3.3.4. Paper Tape

The paper tape reader/punch may be used to process ASCII by proper wiring of the program connector.

3.4. DATA UTILITIES

The data utility routine provides the capability to translate to data management format tapes from ASCII to EBCDIC and EBCDIC to ASCII. For information describing the data utility routine, see *UNIVAC OS/4 Data Utility Routine Programmer Reference, UP-7849* (current version).

To extend the data utilities to include the processing of ASCII files, add the following two PARAM statements:

// PARAM ASCIN=YES[,LENCHK,nn]

// PARAM ASCOUT=YES[,LENCHK,nn]

where: The first PARAM statement is used to indicate the input file is ASCII, and the second latter PARAM statement is used if the output file is ASCII. Otherwise, EBCDIC is assumed.

LENCHK — indicates a length check on variable records of magnetic tape files.

nn — corresponds to a two-digit decimal value for the buffer offset length (maximum value of 99).

Translation of a file from EBCDIC to ASCII, or vice versa, is indicated by defining either the output file as ASCII or the input file as ASCII, respectively.

3.5. JOB CONTROL

Job control provides for definition of ASCII mode files. A file is defined to be ASCII by the following LFD statement:

// LFD filename,,,,ASC

Positional Parameter 5

ASC — specifies that the file being defined is ASCII mode.

if blank — the file being defined is assumed to be EBCDIC mode.

NOTE: Positional parameters 1 through 4 are specified as described in the *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version).

When an ASCII printer file is defined, job control invokes the ASCII load code in place of an EBCDIC load code when the printer is allocated to the job (see Table 3–4). Subsequent printer orders are expected to present only ASCII character strings. Any characters which are extraneous to the ASCII set result in printer mismatches unless the proper load code is initiated prior to the print order. When the printer mode changes for the job step, the printer file must be redefined to job control in order to initiate the proper load code. There is an exception; see data management printer specification (3.3) for load codes. If an ASCII file is to be defined to the system, the ASCII supervisor must be resident or the job step terminates with an error message.

Character strings which represent volume serial numbers (VOL statement) and logical file names (LFD statements, DTF names, PIOCB names, and so on) must begin with alphanumeric characters only. No special characters are permitted in the leading position; otherwise, invalid data translations which lead to erroneous processing result (see 3.2.5).

CHARACTER:	&	Z	K	J	Q	X	V	W
CODE:	26	5A	4B	4A	51	58	56	57
	Y	P	G	B	U	M	C	D
	59	50	47	42	55	4D	43	44
	L	F	H	S	R	O	A	N
	4C	46	43	53	52	4F	41	4E
	I	T	E	.	,	–	0	1
	49	54	45	2E	2C	2D	30	31
	2	3	4	5	6	7	8	9
	32	33	34	35	36	37	38	39
	*	/	+	$	(	)	=	'
	2A	2F	2B	24	28	29	3D	27
	>	<	;	:	[	|	¬	—
	3E	3C	3B	3A	5B	21	5E	5F
	"	]	?	\	%	#	@	NP
	22	5D	3F	7C	25	23	40	20

a. ASCII – 64 Bytes

CHARACTER:	&	Z	K	J	Q	X	V	W
CODE:	50	E9	D2	D1	D8	E7	E5	E6
	Y	P	G	B	U	M	C	D
	E8	D7	C7	C2	E4	D4	C3	C4
	L	F	H	S	R	O	A	N
	D3	C6	C8	E2	D9	D6	C1	DE
	I	T	E	.	,	–	0	1
	C9	E3	C5	4B	6B	60	F0	F1
	2	3	4	5	6	7	8	9
	F2	F3	F4	F5	F6	F7	F8	F9
	*	/	+	$	(	)	=	'
	5C	61	4E	5B	4D	5D	7E	7D
	>	<	;	:	[	|	¬	—
	6E	3C	5E	7A	4A	4F	5F	6D
	"	]	?	\	%	#	@	NP
	7F	5A	6F	6A	6C	7B	7C	40

b. EBCDIC – 64 Bytes

Table 3–4. Job Control Load Code Tables (Part 1 of 2)

CHARACTER:	(	)	'	E	T	A	O	R	C	I	S	L	D	P	N	U	H	M	F	Y	B	W	V	K
CODE:	28	29	27	45	54	41	4F	52	43	49	53	4C	44	50	4E	55	48	4D	46	59	42	57	56	4B
	X	J	Z	G	Q	+	=	>	<	$	[	]	&	__	{	}	.	,	0	1	2	3	4	5
	58	4A	5A	47	51	2B	3D	3E	3C	24	5B	5D	26	5F	7B	7D	2E	2C	30	31	32	33	34	35
	6	7	8	9	*	–	\	|	~	\	#	@	^	%	"	?	!	q	g	z	j	x	k	v
	36	37	38	39	2A	2D	60	7C	7E	5C	23	40	5E	25	22	3F	21	71	67	7A	6A	78	6B	76
	w	b	y	f	m	h	u	n	P	d	l	s	i	c	r	o	a	t	e	/	:	;	.	,
	77	62	79	66	6D	68	75	6E	70	64	6C	73	69	63	72	6F	61	74	65	2F	3A	38	2E	20
	0	1	2	3	4	5	6	7	8	9	*	–	NP											
	30	31	32	33	34	35	36	37	38	39	2A	2D	20											

c. ASCII – 109 Bytes

CHARACTER:	(	)	'	E	T	A	O	R	C	I	S	L	D	P	N	U	H	M	F	Y	B	W	V	K
CODE:	4D	5D	7D	C5	E3	C1	D6	D9	C3	C9	E2	D3	C4	D7	D5	E4	C8	D4	C6	E8	C2	E6	E5	D2
	X	J	Z	G	Q	+	=	>	<	$	[	]	&	__	{	}	.	,	0	1	2	3	4	5
	E7	D1	E9	C7	D8	4E	7E	6E	4C	5B	4A	5A	50	6D	C0	D0	4B	6B	F0	F1	F2	F3	F4	F5
	6	7	8	9	*	–	\	|	~	\	#	@	^	%	"	?	!	q	g	z	j	x	k	v
	F6	F7	F8	F9	5C	60	79	6A	A1	E0	7B	7C	5F	6C	7F	6F	4F	98	87	A9	91	A7	92	A5
	w	b	y	f	m	h	u	n	p	d	l	s	i	c	r	o	a	t	e	/	:	;	.	,
	A6	82	A8	86	94	88	A4	95	97	84	93	A2	89	83	99	96	81	A3	85	61	7A	5E	4B	6B
	0	1	2	3	4	5	6	7	8	9	*	–	NP											
	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	5C	60	40											

d. EBCDIC – 109 Bytes

Table 3–4. Job Control Load Code Tables (Part 2 of 2)

3.6. LANGUAGE PROCESSORS

Facilities are provided to generate ASCII object code for the disc assembler (DASM), COBOL, FORTRAN, and the Report Program Generator (RPG). The language processors, excluding COBOL, call for ASCII translations when they are requested to generate an ASCII object program; therefore, they require the supervisor, generated for ASCII support, to be resident during their execution. Otherwise, the following error message results:

LP01 ASCII EXEC NOT RESIDENT

where ASCII EXEC refers to the supervisor generated with ASCII sensitivity (see 3.2.1). This ASCII supervisor is also required when executing the ASCII object programs.

3.6.1. Assembler

The disc assembler (DASM) generates ASCII object code upon request. Character data constants (C-type) generate ASCII characters when in ASCII mode. Zoned decimal data constants (Z-type) generate the appropriate zones. Hexadecimal and other constants are not affected by the ASCII specification (for example, X'20').

Assembler directives are available to change the mode of the assembly from EBCDIC to ASCII and from ASCII to EBCDIC. The assembled output is declared an ASCII program if the mode is declared as ASCII at any time during the assembly. As a result, care must be exercised throughout the entire program to recognize the ASCII sensitivity (see 2.2.9). The mode of the object code is recorded at the end of the assembly listing.

- ASCII Directive

  The ASCII directive is used to define ASCII constant generation immediately following the directive, up to the recognition of the next mode directive.

- EBCDC Directive

  The EBCDC directive is used to define EBCDIC constant generation immediately following the directive, up to the recognition of the next mode directive.

  If no mode directive is specified, EBCDIC constants are generated.

- Literal Constants

  Literal constants are generated according to the mode under which they are referenced — not according to the mode for the region in which they are generated.

For additional information concerning assembler directives, literal constants, and so on, refer to the *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version).

Example:

LABEL	OPERATION	OPERAND	COMMENTS
	ASCII		
	L	2,=C'CAT'	
	EBCDC		
	L	3,=C'DOG'	
		X'43',X'41',X'54'	(HEX REPRESENTATION OF ASCII CHARACTER CONSTANT CAT)
		X'C4',X'D6',X'C7'	(HEX REPRESENTATION OF EBCDIC CHARACTER CONSTANT DOG)

3.6.2. FORTRAN

The FORTRAN disc compiler (DFOR) provides the facility to generate ASCII object code upon request. The compiler supplies subroutines to translate data from EBCDIC to ASCII and vice versa, provides mode sensitivity in library routines, and handles ANSI standard tape labels.

The compiler also has the facility of generating an ASCII module instead of the standard EBCDIC module. This is accomplished by including a PARAM statement in the control stream. The format of the PARAM statement is:

// PARAM ASC= $\left\{\begin{matrix}(Y)\\(N)\end{matrix}\right\}$

where:

(Y) — specifies that an ASCII module is to be generated.

(N) — specifies that an ASCII module is not to be generated; an EBCDIC module will be generated.

If this PARAM statement is omitted, the object program will be generated in the EBCDIC mode.

NOTE: All generated object modules to be linked by the linkage editor must be in only one mode.

For a full description of FORTRAN statements and the control stream, refer to the *UNIVAC 9400 System FORTRAN Supplementary Reference, UP-7693* (current version).

3.6.2.1. TRANSLATE SUBROUTINES

Subroutines are available to the user to translate data from EBCDIC to ASCII and vice versa. If an input buffer is employed at FORTRAN execution time, it is referenced by the symbolic name FI$BUF. This symbol must be declared by an ENTRY statement in the principal I/O subroutine. When FI$BUF is specified as a parameter to the FORTRAN translate subroutine, an EXTERNAL statement must be used to create the necessary linkages so that the correct address will be generated. This is a special usage of the EXTERNAL statement; therefore, FI$BUF is a unique symbolic name.

- ASCII-to-EBCDIC Translation

  The ASCII-to-EBCDIC-translate subroutine (FT$AE) translates the 128 ASCII characters to the corresponding EBCDIC characters. The CALL statement is issued by the problem program to establish direct linkage with the subroutine. The format is:

LABEL	OPERATION	OPERAND
unused	CALL	FT$AE,($a_1$,$a_2$)

Positional Parameter 1

FT$AE is the symbolic address of the entry point in the ASCII-to-EBCDIC-translate subroutine.

Positional Parameter 2

a_1 represents a FORTRAN symbolic variable name, array element name, array name, or specifically FI$BUF.

a_2 represents a FORTRAN integer expression, the value of which specifies the length (in bytes) of the character string to be translated.

- EBCDIC-to-ASCII Translation

The EBCDIC-to-ASCII-translate subroutine (FT$EA) translates the EBCDIC character to the corresponding ASCII character. If no correspondence exists, the EBCDIC bit pattern is translated to the ASCII SUB character. The CALL statement is issued by the problem program to establish direct linkage with the subroutine. The format is:

LABEL	ↂ OPERATION ↂ	OPERAND
unused	CALL	FT$EA,($a_1$,$a_2$)

Positional Parameter 1

FT$EA is the symbolic address of the entry point in the EBCDIC-to-ASCII-translate subroutine.

Positional Parameter 2

a_1 represents a FORTRAN symbolic variable name, array element name, array name, or specifically FI$BUF.

a_2 represents a FORTRAN integer expression, the value of which specifies the length (in bytes) of the character string to be translated.

3.6.2.2. COMPLEMENTARY MODE TAPE FILES

Complementary mode tape files are allowed on input only. A tape is considered in a complementary mode when the mode of the file is opposite that of the mode of the object program. Output of complementary mode files is not allowed since there is no acceptable method of interrupting the I/O process to give control to the user for translation. Therefore, the problem program should be generated in the same mode as that required for the output files.

On input tape files, the mechanism to be used involves current features of the UNIVAC FORTRAN compiler. The user program should issue a READ statement with no list. If the READ is formatted, it is sufficient to specify one numeric type field descriptor in the FORMAT statement to ensure correct termination of the read request. In both the formatted and unformatted cases, the READ must not require more than one physical tape record to be satisfied (e.g., no slashes should be given in a FORMAT statement associated with this READ statement).

Upon return from the READ, the user calls the appropriate translate subroutine (3.6.2.2) using FI$BUF as positional parameter 1. The input buffer has a length of 255 bytes; therefore, 255 should be the maximum value specified by positional parameter 2 when translating the input buffer.

After translation, the buffer may be reread through a formatted or unformatted READ on unit number 29 with the normal list of variable names, etc. This READ, however, is also restricted to require only one physical tape record.

Example:

```
LABEL    OPERATION    OPERAND
1        10       16
         EXTERNAL     FI$BUF
         READ         (7,3)
3        FORMAT       (I1)
         L=255
         CALL         FT$AE(FI$BUF,L)
         READ         (29,4)    (RVAR(K),K=1,6)
4        FORMAT       (3HXYZ,6G20.7)
```

3.6.2.3. TAPE FORMATS

ANSI standard labels are checked for an input file in the ASCII mode of operation and are generated for output requests by the object program. Unformatted and NAMELIST I/O requests from FORTRAN are not acceptable for interchange tapes, but interchangeability is guaranteed for FORTRAN I/O under format control.

The format for all FORTRAN-generated ASCII data blocks is:

CHARACTER POSITION	FIELD	NAME	LENGTH
1	1	Block sequence indicator in decimal	1
2 – 4	2	Block length in decimal	4
5 – 9	3	Record length in decimal	4
10 – 264	4	Data	1 – 255
265 – 267	5	Block padding (circumflex characters)	3

3.6.2.4. OTHER PERIPHERAL DEVICES

ASCII files must be declared to the system by means of the ASCII positional parameter on the LFD job control statement.

Depending on the mode of a file assigned to either the printer or the card punch, the system handles records for the object program in the declared mode. Card input is sensitive to the mode of the object program.

Console replies are translated to ASCII when the FORTRAN object program is in the ASCII mode. Console messages are transmitted without restriction when in ASCII mode.

Any file may be declared an ASCII file except those files assigned to disc. Data may be stored in ASCII on the disc at the user's discretion, however.

3.6.3. COBOL

When the ASCII PARAM statement is specified in the control stream, the COBOL compiler produces an object program which assumes that the contents of any data item, with an implied or explicit USAGE IS DISPLAY, is represented in ASCII. The format of the PARAM statement is:

```
// PARAM OUT=A
```

Although an EBCDIC or ASCII object program may be generated by the compiler, compilation is always performed in EBCDIC mode, as is the source program input and all listable output. The following discussion applies to programs produced under control of the ASCII option.

When ASCII mode is selected, compiler-created object programs are sensitive at the hardware instruction level to the ASCII character set.

The value associated with the figurative constant HIGH-VALUE is hexadecimal 7F. QUOTE, ZERO, and SPACE take on the corresponding ASCII value. LOW-VALUE remains hexadecimal 00.

Values associated with WORKING-STORAGE items whose USAGE is DISPLAY are ASCII characters in the object program. The allocation of values for COMPUTATIONAL data is not affected by the selection of ASCII mode. These values are generated in the corresponding COMPUTATIONAL format. When COMPUTATIONAL items are moved or compared with DISPLAY items, they are automatically converted to the ASCII character values.

The ASCII character set does not facilitate the use of the traditional overpunch sign convention. However, UNIVAC OS/4 implementation of ASCII does permit overpunching (see *UNIVAC OS/4 COBOL Supplementary Reference, UP-7709* (current version). An ASCII signed numeric DISPLAY item should have an S in its PICTURE and be associated with a SIGN IS SEPARATE CHARACTER clause. Signed numeric items used in conjunction with arithmetic operations are automatically converted to signed COMPUTATIONAL format. When items with separate signs are converted to packed decimal format, the following sign convention prevails: hexadecimal C for plus and hexadecimal D for minus.

Signed numeric values or procedure division literals must be specified in the source language with the sign character as the left-most character, regardless of the separate sign option.

ASCII files must be declared to the compiler by the APPLY ASCII ON filename clause. ASCII files must be declared to the system by the ASCII positional parameter on the LFD job control statement. Any file may be classified as an ASCII file except those files assigned to DISC. An object program mix of ASCII and non-ASCII files is permitted.

Regardless of the mode specified for files assigned to TAPE, user labels and data records are presented to the object program in their external character code representation. No translation of input or output records (or labels) is performed by the system. The RECORDING MODE IS D clause may be specified for ASCII tape files which contain variable-length records.

An option within the APPLY ASCII ON file-name clause allows specification of a buffer offset for any tape input file or for the activation of the block length check feature on tape files with variable length records.

If a file assigned to the printer is declared as being an ASCII file, the device will accept only ASCII records. Multiple print files assigned to the same device with different mode specification are not permitted.

Depending on the mode specified for a file assigned to a card reader, the system will present input records to the object program in either ASCII or EBCDIC. Note that multiple files assigned to the same card reader device with different mode specifications are not permitted.

Depending on the mode specified for a file assigned to a card punch, the system will accept output records from the object program in either ASCII or EBCDIC. Note that multiple files assigned to the same card punch device with different mode specifications are not permitted.

Procedure division numeric literals and nonnumeric literals used in conjunction with items whose USAGE IS DISPLAY are allocated within the object program as ASCII constants. Literals associated with COMPUTATIONAL data items are allocated in the corresponding COMPUTATIONAL format.

The results of an IF statement associated with an item whose USAGE IS DISPLAY is based upon the assumption that the contents of the item is represented in ASCII.

DISPLAYs upon SYSLST and output from TRACE/EXHIBIT statements require that ASCII be specified on the LFD control card for SYSLST. If, at the time this output is being directed to SYSLST, a user print file assigned to the same physical device is OPEN but was not declared as an ASCII file, the device will not properly duplicate the debug output records.

Since the console assumes the mode of the object program, only ASCII data should be DISPLAYed upon SYSCONSOLE. When ACCEPTing data from SYSCONSOLE, the object program will receive ASCII data.

Data ACCEPTed from the control stream will also be in ASCII. This includes PARAM statements.

SORTing is controlled entirely by the SORT KEY specifications and the values contained within the key fields. SORT KEY specifications whose category is other than numeric are processed according to hexadecimal value. Numeric keys are treated algebraically regardless of the item's operational sign location.

The TRANSFORM statement may be used to convert DISPLAY items from one character representation to another. A special format of the TRANSFORM statement facilitates translation from ASCII to EBCDIC or from EBCDIC to ASCII via compiler-created translate tables. To specify conversion the reserved words ASCII and EBCDIC must be used with the words FROM and TO. If the TRANSFORM statement contains nonnumeric literals, transformation across code bases is not possible.

The language available for ASCII support (SIGN, MODE D, APPLY ASCII, and TRANSFORM) is also available to the COBOL user when the ASCII PARAM option is not specified. This permits processing of ASCII data files in an EBCDIC object program.

For a full description of COBOL statements and the job control stream, refer to *UNIVAC OS/4 COBOL Supplementary Reference, UP-7709* (current version). ←

3.6.4. Report Program Generator (RPG)

The disc RPG provides the facility to generate ASCII object code for source programs compiled in OS/4 mode. ← ASCII constants, literals, and edit words are generated within the object output. The use of overpunches on numeric fields to provide signed characters is not accepted. Leading and trailing signs are supported. The RPG object programs cannot accept tape files which are different from the mode of the program (for example, EBCDIC program and ASCII tape files).

- File Description Specifications

  If tape block numbering is desired, a B must be specified in column 28 of the File Description form.

  Tape buffer offset may be defined for ASCII input files only by specifying the length in number of bytes in columns 35 through 38 of the File Description form (right justified in the field). Tape buffer offset definition for output files is ignored. Input files defining tape buffer offset must include the length of the offset in block size specifications.

- Input or Output Specification

An ASCII numeric item that is signed must be specified as a separate signed character. An R or L character must be specified for right or left separate sign fields in column 43 of the input specification or column 44 of the output specification. Note that output sign separation is restricted to nonedited numeric fields.

- Control Stream

The disc RPG generates an ASCII object program when the following PARAM statement is specified in the control stream:

```
// PARAM    ASC=Y
```

where:

ASC=Y — specifies that an ASCII program is to be generated.

- Printing ASCII Line Counter Tape or Disc Report Files

The utility program UTRPG is available for printing tape and disc report files created by the use of the line counter feature of RPG. A maximum of eight tape and/or disc files, one report to each file, may be printed in one run. The files are printed one at a time, in their entirety. All outuput is directed to a single printer. The message

DM4 did filename LFD CHANGED MODE

is typed on the console.

ASCII input tape files must have ASCII standard labels, created by using the magnetic tape preparation routine, UTPREP. Line counter tape files must not have block numbering. Tape filenames on LFD statements are TREPT1,,,,ASC through TREPT8,,,,ASC; disc filenames are DREPT1,,,,ASC through DREPT8,,,,ASC.

Output requires an LFD job control statement with the specification PRNTR,,,,ASC for the print file.

- ASCII RPG Halt Indicator Processing

The halt indicator processing module, which is elected by placing the letter D in column 8 of the RPG source header card, requires that COMREG settings be chosen from among the permissible EBCDIC characters designated in *UNIVAC OS/4 Report Program Generator Programmer Reference, UP-7707* (current version). If the ASC parameter is included in the SET COMREG statement, the characters must be specified in hexadecimal. The following two SET statements are correct and equivalent:

```
// SET      COMREG,X'C4',ASC
// SET      COMREG,C'D'
```

If any of the first ten COMREG bytes is set to D (X'C4'), an ASCII print file must be defined in the control stream as LFD PRNTR,,,,ASC.

For a full description of RPG specifications and control stream, refer to the *UNIVAC OS/4 Report Program Generator Programmer Reference, UP-7707* (current version).

3.7. SORT/MERGE

The sort/merge program recognizes two new key field format codes. These format codes, which are supported for both ASCII and EBCDIC collating sequences, are in addition to the format codes already defined in *UNIVAC OS/4 System Sort/Merge Programmer Reference, UP-7664* (current version). ←

The two additional format codes are specified by the form-i specification of the FIELD keyword parameter. They are two-character alphabetic codes defined as follows:

ZL – zoned decimal with leading sign

ZT – zoned decimal with trailing sign

The binary format codes for the FIELD parameter are X'05' for leading sign and X'06' for trailing sign. These formats have been provided to support numeric data with separate sign characters (+ or –). If the discrete sign character is not determined to be minus (–), it is assumed to be positive (+). The length of these key fields must not exceed 17 bytes, including the sign byte if zoned leading or trailing fields are specified. ←

The sorting of character data is independent of the program and file mode since the collating sequence is defined by the binary representation of the sort keys. That is, if ASCII character data is presented in the key, the sort/merge routine will adapt the ASCII collating sequence. This is done by applying the compare-logical instruction (CLC) which reacts on the binary magnitude of the character string. Thus, the collating sequence may be changed by translating the key fields whenever desirable.

3.8. LINKAGE EDITOR

The disc linkage editor has been enhanced to define ASCII load modules. Phases which contain both ASCII and EBCDIC or just ASCII object modules are defined to be ASCII phases; the ASCII definition is carried over to all of the other phases which comprise the load module. UNIVAC software, which is anticipated to be linked into user phases such as data management modules, is coded in a manner which leaves each module mode insensitive. For a full description of the linkage editor, refer to the *UNIVAC OS/4 Linkage Editor Programmer Reference, UP-7703* (current version) ←

3.9. UTILITY AND SERVICE ROUTINES

Utility and service routines provided for ASCII support are the magnetic tape preparation routine and the tape-to-print dump routines. System support for these routines is described in the following paragraphs. For detailed information concerning the tape prep and tape print routines, see *UNIVAC OS/4 Utility and Service Routines Programmer Reference, UP-7713* (current version). ←

3.9.1. Magnetic Tape Preparation Routine

The magnetic tape preparation routine (UTPREP) initializes tapes in standard label format which conform to the ANSI label specifications. To initialize a tape using the ASCII characters (ANSI standard labels), specify the LFD statement in the control stream as follows:

```
// LFD    TAPEOT0n,,,,ASC
```

where n represents any decimal number 1 through 9.

If the ASCII positional parameter is omitted, the tape is initialized using EBCDIC characters.

When block numbers are requested, the one-byte block-sequence indicator (see 3.3.1.1) results for ASCII files.

NOTE: Tapes used for system software functions must be prepped with EBCDIC standard labels; otherwise, tape label checking will detect tape label errors and will result in program termination.

When creating both ASCII and EBCDIC files in the same job step, only one tape may be specified (positional parameter 3) on every PARAM statement included in that job step. Furthermore, the numbe; of PARAM statements that are in a job step must be the same as the number of logical files that are defined (i.e., the number of LFD TAPEOT statements).

3.9.2. Tape-to-Print Dump Routine

The tape-to-print dump routine (UTTPPR) produces a listing of an entire tape, or of specified portions of a tape in the form of ASCII characters when ASCII mode is declared to the routine and alphanumeric representation is requested. To define ASCII tape files to UTTPPR, specify the LFD statement in the job control stream as follows:

```
// LFD    TAPEIN,,,,ASC
```

If the ASC positional parameter is omitted, the listing produced is assumed to be in EBCDIC characters.

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

CUT

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT