APPENDIX A

INSTRUCTIONS AND MNEMONICS

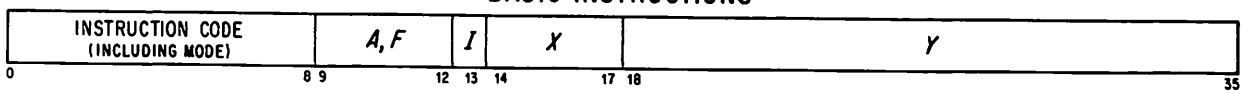
The drawing on the next two pages shows the formats of the various types of words used by the processor (instructions, pointers, operands, etc). The illustration on page A-4 shows the derivation of the instruction mnemonics. Next are two tables that list all instruction mnemonics and their octal codes both numerically and alphabetically. When two mnemonics are given for the same octal code, the first is the preferred form, but the assembler does recognize the second. For completeness, the tables include the MUUOs (indicated by an asterisk) that are recognized by MACRO for communication with the DECsystem-10 Time Sharing Monitor. A double dagger (‡) indicates a KI10 instruction code that is unassigned in the KA10.

Note that 247 is unassigned in the KI10, but 247 and 257 execute as no-ops in the KA10.

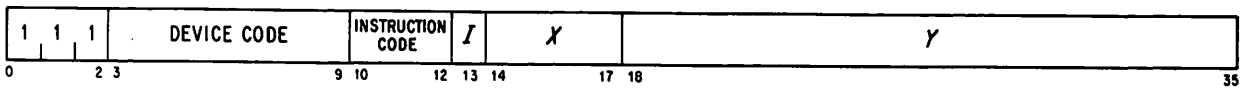
In-out device codes are included only in the alphabetic listing and are indicated by a dagger (†). Following the tables is a chart that lists the devices with their mnemonic and octal codes and DEC type numbers for both PDP-10 and PDP-6. A device mnemonic ending in the numeral 2 is the recommended form for the second of a given device, but such codes are not recognized by MACRO – they must be defined by the user.

Beginning on page A-13 is a list of all instructions showing their actions in symbolic form. On page A-23 is a table of the positive and negative powers of 2.

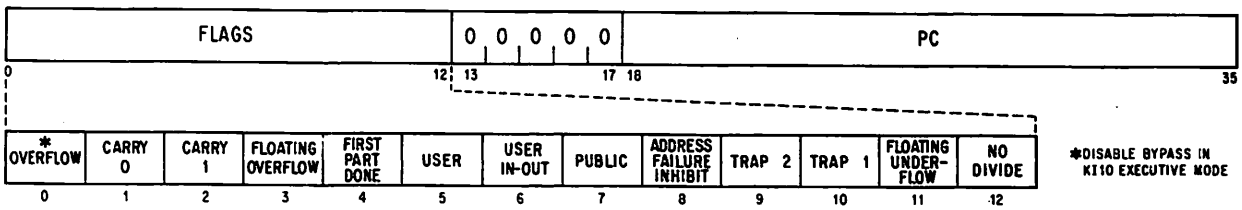
BASIC INSTRUCTIONS



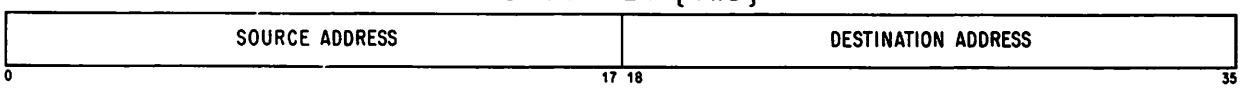
IN-OUT INSTRUCTIONS



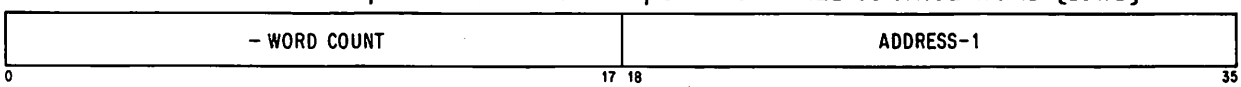
PC WORD



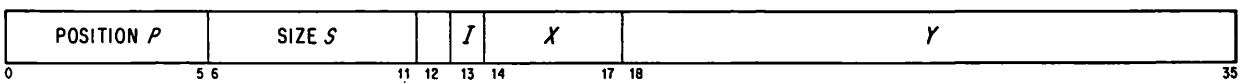
BLT POINTER {XWD}



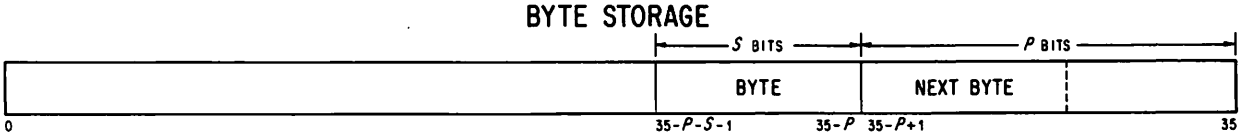
BLKI / BLKO POINTER, PUSHDOWN POINTER, DATA CHANNEL CONTROL WORD {IOWD}



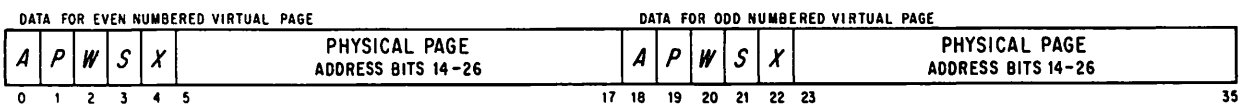
BYTE POINTER



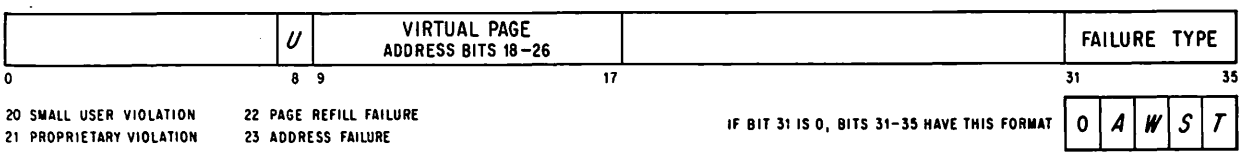
BYTE STORAGE



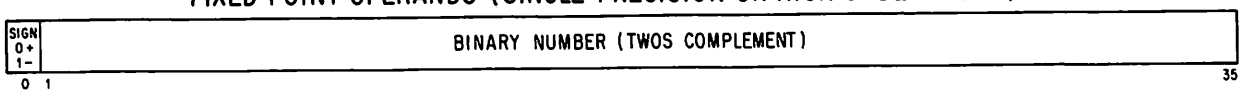
PAGE MAP WORD



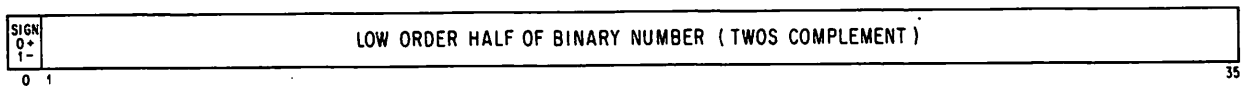
PAGE FAIL WORD



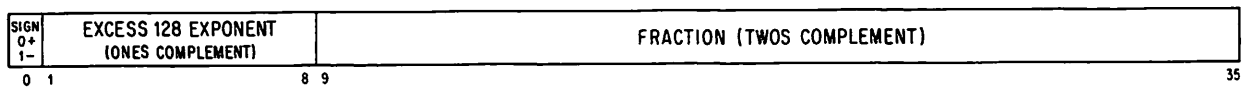
FIXED POINT OPERANDS (SINGLE PRECISION OR HIGH ORDER WORD)



LOW ORDER WORD IN DOUBLE LENGTH FIXED POINT OPERANDS

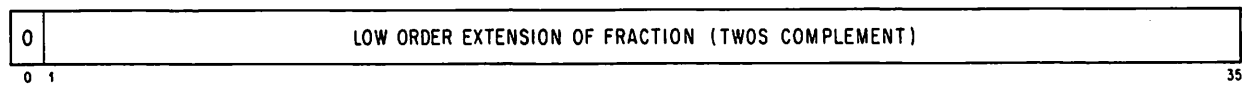


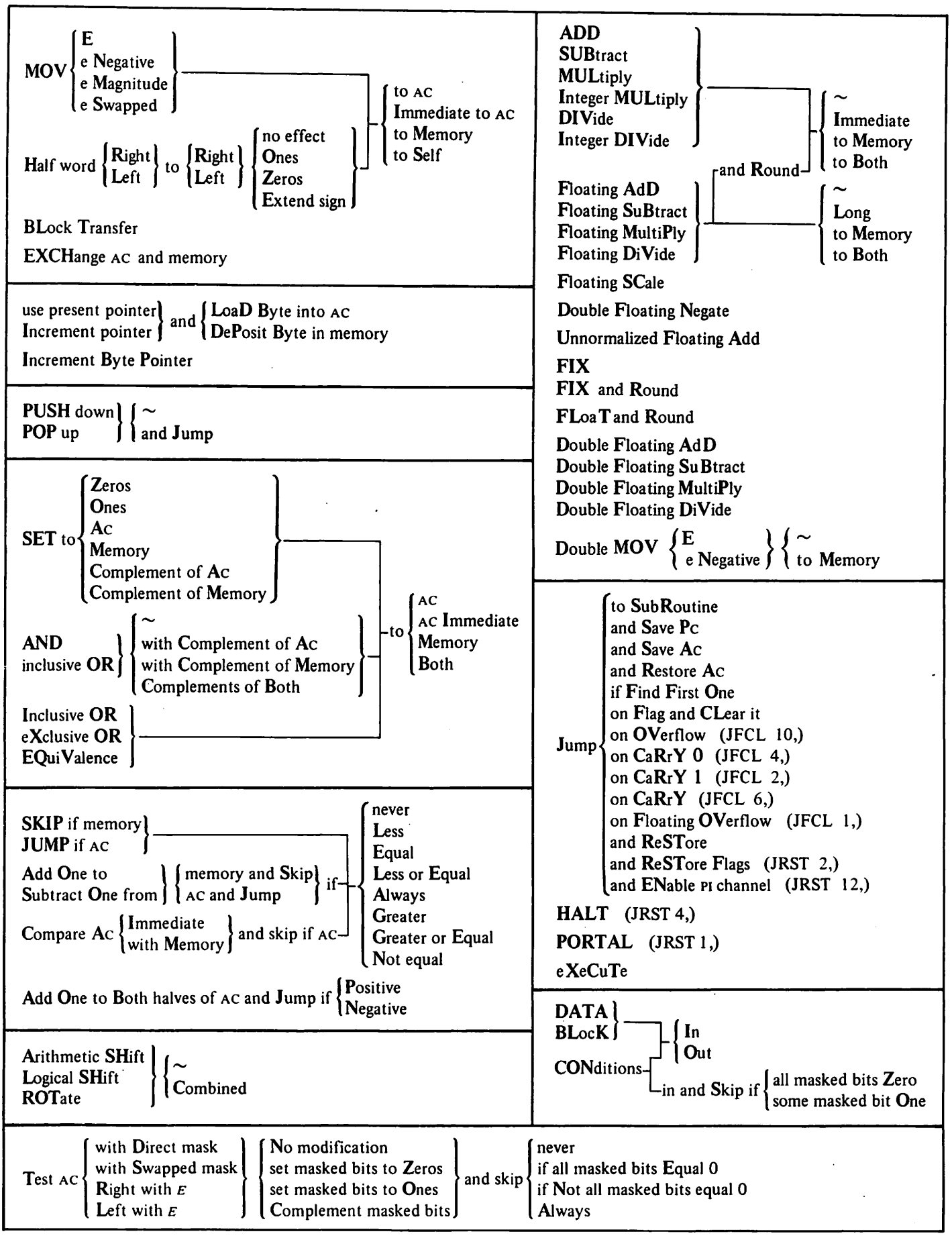
FLOATING POINT OPERANDS (SINGLE PRECISION OR HIGH ORDER WORD)



LOW ORDER WORD IN SOFTWARE DOUBLE LENGTH FLOATING POINT OPERANDS

LOW ORDER WORD IN HARDWARE DOUBLE LENGTH FLOATING POINT OPERANDS




MOV
E
e Negative
e Magnitude
e Swapped
to AC
Immediate to AC
to Memory
to Self
Half word
Right
Left
to
Right
Left
no effect
Ones
Zeros
Extend sign
BLock Transfer
EXCHange AC and memory
use present pointer
Increment pointer
and
LoaD Byte into AC
DePosit Byte in memory
Increment Byte Pointer
PUSH down
POP up
~
and Jump
SET to
Zeros
Ones
Ac
Memory
Complement of Ac
Complement of Memory
AND
inclusive OR
~
with Complement of Ac
with Complement of Memory
Complements of Both
to
AC
AC Immediate
Memory
Both
Inclusive OR
eXclusive OR
EQuiValence
SKIP if memory
JUMP if AC
Add One to
Subtract One from
memory and Skip
AC and Jump
if
never
Less
Equal
Less or Equal
Always
Greater
Greater or Equal
Not equal
Compare Ac
Immediate
with Memory
and skip if AC
Add One to Both halves of AC and Jump if
Positive
Negative
Arithmetic SHift
Logical SHift
ROTate
~
Combined
ADD
SUBtract
MULtiply
Integer MULtiply
DIVide
Integer DIVide
and Round
~
Immediate
to Memory
to Both
Floating AdD
Floating SuBtract
Floating MultiPly
Floating DiVide
and Round
~
Long
to Memory
to Both
Floating SCale
Double Floating Negate
Unnormalized Floating Add
FIX
FIX and Round
FLoaT and Round
Double Floating AdD
Double Floating SuBtract
Double Floating MultiPly
Double Floating DiVide
Double MOV
E
e Negative
~
to Memory
Jump
to SubRoutine
and Save Pc
and Save Ac
and Restore Ac
if Find First One
on Flag and CLear it
on OVerflow (JFCL 10,)
on CaRrY 0 (JFCL 4,)
on CaRrY 1 (JFCL 2,)
on CaRrY (JFCL 6,)
on Floating OVerflow (JFCL 1,)
and ReSTore
and ReSTore Flags (JRST 2,)
and ENable PI channel (JRST 12,)
HALT (JRST 4,)
PORTAL (JRST 1,)
eXeCuTe
DATA
BLocK
In
Out
CONditions
in and Skip if
all masked bits Zero
some masked bit One
Test AC
with Direct mask
with Swapped mask
Right with E
Left with E
No modification
set masked bits to Zeros
set masked bits to Ones
Complement masked bits
and skip
never
if all masked bits Equal 0
if Not all masked bits equal 0
Always

INSTRUCTION MNEMONICS

NUMERIC LISTING

000	ILLEGAL	106		162	FMPM
001		107		163	FMPB
:	LUUO'S	110	‡DFAD	164	FMPR
:		111	‡DFSB	165	FMPRI
037		112	‡DFMP	166	FMPRM
040	*CALL	113	‡DFDV	167	FMPRB
041	*INIT	114		170	FDV
042		115		171	FDVL
043	RESERVED FOR SPECIAL MONITORS	116		172	FDVM
044		117		173	FDVB
045		120	‡DMOVE	174	FDVR
046		121	‡DMOVN	175	FDVRI
047	*CALLI	122	‡FIX	176	FDVRM
050	*OPEN	123		177	FDVRB
051	*TTCALL	124	‡DMOVEM	200	MOVE
052		125	‡DMOVNM	201	MOVEI
053	RESERVED FOR DEC	126	‡FIXR	202	MOVEM
054		127	‡FLTR	203	MOVES
055	*RENAME	130	UFA	204	MOVS
056	*IN	131	DFN	205	MOVSI
057	*OUT	132	FSC	206	MOVSM
060	*SETSTS	133	IBP	207	MOVSS
061	*STATO	134	ILDB	210	MOVN
062	*STATUS	135	LDB	211	MOVNI
062	*GETSTS	136	IDPB	212	MOVNM
063	*STATZ	137	DPB	213	MOVNS
064	*INBUF	140	FAD	214	MOVM
065	*OUTBUF	141	FADL	215	MOVMI
066	*INPUT	142	FADM	216	MOVMM
067	*OUTPUT	143	FADB	217	MOVMS
070	*CLOSE	144	FADR	220	IMUL
071	*RELEAS	145	FADRI	221	IMULI
072	*MTAPE	146	FADRM	222	IMULM
073	*UGETF	147	FADRB	223	IMULB
074	*USETI	150	FSB	224	MUL
075	*USETO	151	FSBL	225	MULI
076	*LOOKUP	152	FSBM	226	MULM
077	*ENTER	153	FSBB	227	MULB
100	*UJEN	154	FSBR	230	IDIV
101		155	FSBRI	231	IDIVI
102		156	FSBRM	232	IDIVM
103		157	FSBRB	233	IDIVB
104		160	FMP	234	DIV
105		161	FMPL	235	DIVI

236	DIVM	306	CAIN	367	SOJG
237	DIVB	307	CAIG	370	SOS
240	ASH	310	CAM	371	SOSL
241	ROT	311	CAML	372	SOSE
242	LSH	312	CAME	373	SOSLE
243	JFFO	313	CAMLE	374	SOSA
244	ASHC	314	CAMA	375	SOSGE
245	ROTC	315	CAMGE	376	SOSN
246	LSHC	316	CAMN	377	SOSG
247		317	CAMG	400	SETZ
250	EXCH	320	JUMP	400	CLEAR
251	BLT	321	JUMPL	401	SETZI
252	AOBJP	322	JUMPE	401	CLEARI
253	AOBJN	323	JUMPLE	402	SETZM
254	JRST	324	JUMPA	402	CLEARM
25404	PORTAL	325	JUMPGE	403	SETZB
25410	JRSTF	326	JUMPN	403	CLEARB
25420	HALT	327	JUMPG	404	AND
25450	JEN	330	SKIP	405	ANDI
255	JFCL	331	SKIPL	406	ANDM
25504	JFOV	332	SKIPE	407	ANDB
25510	JCRY1	333	SKIPLE	410	ANDCA
25520	JCRY0	334	SKIPA	411	ANDCAI
25530	JCRY	335	SKIPGE	412	ANDCAM
25540	JOV	336	SKIPN	413	ANDCAB
256	XCT	337	SKIPG	414	SETM
257	‡MAP	340	AOJ	415	SETMI
260	PUSHJ	341	AOJL	416	SETMM
261	PUSH	342	AOJE	417	SETMB
262	POP	343	AOJLE	420	ANDCM
263	POPJ	344	AOJA	421	ANDCMI
264	JSR	345	AOJGE	422	ANDCMM
265	JSP	346	AOJN	423	ANDCMB
266	JSA	347	AOJG	424	SETA
267	JRA	350	AOS	425	SETAI
270	ADD	351	AOSL	426	SETAM
271	ADDI	352	AOSE	427	SETAB
272	ADDM	353	AOSLE	430	XOR
273	ADDB	354	AOSA	431	XORI
274	SUB	355	AOSGE	432	XORM
275	SUBI	356	AOSN	433	XORB
276	SUBM	357	AOSG	434	IOR
277	SUBB	360	SOJ	434	OR
300	CAI	361	SOJL	435	IORI
301	CAIL	362	SOJE	435	ORI
302	CAIE	363	SOJLE	436	IORM
303	CAILE	364	SOJA	436	ORM
304	CAIA	365	SOJGE	437	IORB
305	CAIGE	366	SOJN	437	ORB

440	ANDCB	521	HLLOI	602	TRNE
441	ANDCBI	522	HLLOM	603	TLNE
442	ANDCBM	523	HLLOS	604	TRNA
443	ANDCBB	524	HRLO	605	TLNA
444	EQV	525	HRLOI	606	TRNN
445	EQVI	526	HRLOM	607	TLNN
446	EQVM	527	HRLOS	610	TDN
447	EQVB	530	HLLE	611	TSN
450	SETCA	531	HLLEI	612	TDNE
451	SETCAI	532	HLLEM	613	TSNE
452	SETCAM	533	HLLES	614	TDNA
453	SETCAB	534	HRLE	615	TSNA
454	ORCA	535	HRLEI	616	TDNN
455	ORCAI	536	HRLEM	617	TSNN
456	ORCAM	537	HRLES	620	TRZ
457	ORCAB	540	HRR	621	TLZ
460	SETCM	541	HRRI	622	TRZE
461	SETCMI	542	HRRM	623	TLZE
462	SETCMM	543	HRRS	624	TRZA
463	SETCMB	544	HLR	625	TLZA
464	ORCM	545	HLRI	626	TRZN
465	ORCMI	546	HLRM	627	TLZN
466	ORCMM	547	HLRS	630	TDZ
467	ORCMB	550	HRRZ	631	TSZ
470	ORCB	551	HRRZI	632	TDZE
471	ORCBI	552	HRRZM	633	TSZE
472	ORCBM	553	HRRZS	634	TDZA
473	ORCBB	554	HLRZ	635	TSZA
474	SETO	555	HLRZI	636	TDZN
475	SETOI	556	HLRZM	637	TSZN
476	SETOM	557	HLRZS	640	TRC
477	SETOB	560	HRRO	641	TLC
500	HLL	561	HRROI	642	TRCE
501	HLLI	562	HRROM	643	TLCE
502	HLLM	563	HRROS	644	TRCA
503	HLLS	564	HLRO	645	TLCA
504	HRL	565	HLROI	646	TRCN
505	HRLI	566	HLROM	647	TLCN
506	HRLM	567	HLROS	650	TDC
507	HRLS	570	HRRE	651	TSC
510	HLLZ	571	HRREI	652	TDCE
511	HLLZI	572	HRREM	653	TSCE
512	HLLZM	573	HRRES	654	TDCA
513	HLLZS	574	HLRE	655	TSCA
514	HRLZ	575	HLREI	656	TDCN
515	HRLZI	576	HLREM	657	TSCN
516	HRLZM	577	HLRES	660	TRO
517	HRLZS	600	TRN	661	TLO
520	HLLO	601	TLN	662	TROE

663	TLOE	673	TSOE	70010	BLKO
664	TROA	674	TDOA	70014	DATAO
665	TLOA	675	TSOA	70020	CONO
666	TRON	676	TDON	70024	CONI
667	TLON	677	TSON	70030	CONSZ
670	TDO	70000	BLKI	70034	CONSO
671	TSO	70004	DATAI		
672	TDOE	70004	RSW		

INSTRUCTION MNEMONICS

ALPHABETIC LISTING

†ADC	024	AOSA	354	†CDP	110
ADD	270	AOSE	352	†CDR	114
ADDB	273	AOSG	357	CLEAR	400
ADDI	271	AOSGE	355	CLEARB	403
ADDM	272	AOSL	351	CLEARI	401
AND	404	AOSLE	353	CLEARM	402
ANDB	407	AOSN	356	†CLK	070
ANDCA	410	†APR	000	*CLOSE	070
ANDCAB	413	ASH	240	CONI	70024
ANDCAI	411	ASHC	244	CONO	70020
ANDCAM	412	BLKI	70000	CONSO	70034
ANDCB	440	BLKO	70010	CONSZ	70030
ANDCBB	443	BLT	251	†CPA	000
ANDCBI	441	CAI	300	†CR	150
ANDCBM	442	CAIA	304	DATAI	70004
ANDCM	420	CAIE	302	DATAO	70014
ANDCMB	423	CAIG	307	†DC	200
ANDCMI	421	CAIGE	305	†DCSA	300
ANDCMM	422	CAIL	301	†DCSB	304
ANDI	405	CAILE	303	‡DFAD	110
ANDM	406	CAIN	306	‡DFDV	113
AOBJN	253	*CALL	040	‡DFMP	112
AOBJP	252	*CALLI	047	DFN	131
AOJ	340	CAM	310	‡DFSB	111
AOJA	344	CAMA	314	†DIS	130
AOJE	342	CAME	312	DIV	234
AOJG	347	CAMG	317	DIVB	237
AOJGE	345	CAMGE	315	DIVI	235
AOJL	341	CAML	311	DIVM	236
AOJLE	343	CAMLE	313	†DLB	060
AOJN	346	CAMN	316	†DLC	064
AOS	350	†CCI	014	†DLS	240

JRST	254	ORCAM	456	SETOM	476
JRSTF	25410	ORCB	470	*SETSTS	060
JSA	266	ORCBB	473	SETZ	400
JSP	265	ORCBI	471	SETZB	403
JSR	264	ORCBM	472	SETZI	401
JUMP	320	ORCM	464	SETZM	402
JUMPA	324	ORCMB	467	SKIP	330
JUMPE	322	ORCMI	465	SKIPA	334
JUMPG	327	ORCMM	466	SKIPE	332
JUMPGE	325	ORI	435	SKIPG	337
JUMPL	321	ORM	436	SKIPGE	335
JUMPLE	323	*OUT	057	SKIPL	331
JUMPN	326	*OUTBUF	065	SKIPLE	333
LDB	135	*OUTPUT	067	SKIPN	336
*LOOKUP	076	†PAG	010	SOJ	360
†LPT	124	†PI	004	SOJA	364
LSH	242	†PLT	140	SOJE	362
LSHC	246	POP	262	SOJG	367
‡MAP	257	POPJ	263	SOJGE	365
MOVE	200	PORTAL	25404	SOJL	361
MOVEI	201	†PTP	100	SOJLE	363
MOVEM	202	†PTR	104	SOJN	366
MOVES	203	PUSH	261	SOS	370
MOVM	214	PUSHJ	260	SOSA	374
MOVMI	215	*RELEAS	071	SOSE	372
MOVMM	216	*RENAME	055	SOSG	377
MOVMS	217	†RMC	270	SOSGE	375
MOVN	210	ROT	241	SOSL	371
MOVNI	211	ROTC	245	SOSLE	373
MOVNM	212	RSW	70004	SOSN	376
MOVNS	213	SETA	424	*STATO	061
MOVS	204	SETAB	427	*STATUS	062
MOVSI	205	SETAI	425	*STATZ	063
MOVSM	206	SETAM	426	SUB	274
MOVSS	207	SETCA	450	SUBB	277
*MTAPE	072	SETCAB	453	SUBI	275
†MTC	220	SETCAI	451	SUBM	276
†MTM	230	SETCAM	452	TDC	650
†MTS	224	SETCM	460	TDCA	654
MUL	224	SETCMB	463	TDCE	652
MULB	227	SETCMI	461	TDCN	656
MULI	225	SETCMM	462	TDN	610
MULM	226	SETM	414	TDNA	614
*OPEN	050	SETMB	417	TDNE	612
OR	434	SETMI	415	TDNN	616
ORB	437	SETMM	416	TDO	670
ORCA	454	SETO	474	TDOA	674
ORCAB	457	SETOB	477	TDOE	672
ORCAI	455	SETOI	475	TDON	676

FIRST OCTAL DIGIT \ SECOND AND THIRD OCTAL DIGITS	00	04	10	14	20	24	30	34	40	44	50	54	60	64	70	74
0	6,10 APR CPA CENTRAL PROCESSOR	6,10 PI PRIORITY INTERRUPT	10 PAG* KI10 PAGING	10 DA10 CCI PDP-8,9 INTERFACE	10 DA10 CCI2 PDP-8,9 INTERFACE	10 AD10 ADC ANALOG-DIGITAL CONVERTER	10 AD10 ADC2 ANALOG-DIGITAL CONVERTER						10 DLB PDP-11	DL10 DLC DATA LINK	10 DK10 CLK REAL TIME CLOCK	10 DK10 CLK2 REAL TIME CLOCK
1	6 761 10 PTP PAPER TAPE PUNCH	6 760 10 PTR PAPER TAPE READER	10 CP10 CDP CARD PUNCH	6 461 CDR CARD READER	6 626 10 TTY CONSOLE TELETYPE	6 646 10 LP10 LPT LINE PRINTER	6,10 340 10 VP10 DIS DISPLAY	6,10 340 10 VP10 DIS2 DISPLAY	10 XY10 PLT PLOTTER	10 XY10 PLT2 PLOTTER	10 CR10 CR CARD READER	10 CR10 CR2 CARD READER	10 DLB2* PDP-11	DL10 DLC2 DATA LINK	10 RC10 DSK DISK/DRUM	10 RC10 DSK2 DISK/DRUM
2	6 136 DC DATA CONTROL	6 136 DC2 DATA CONTROL	6 551 UTC UTS DECTAPE		6 MTC MTS MTM† MAGNETIC TAPE		516	6 646 10 LP10 LPT2† LINE PRINTER	10 DC10 DLS DATA LINE SCANNER	10 DC10 DLS2 DATA LINE SCANNER	10 RP10 DPC DISK PACK SYSTEM	10 RP10 DPC2 DISK PACK SYSTEM	10 RP10 DPC3 DISK PASK SYSTEM	10 RP10 DPC4 DISK PACK SYSTEM	10 RH10 RMC* DATA CONTROL	10 RH10 RMC2 DATA CONTROL
3	6 DCSA	630 DCSB DATA COMMUNICATION			10 DTC	TD10 DTS DECTAPE	10 DTC2	TD10 DTS2 DECTAPE	10 TMC	TM10 TMS MAGNETIC TAPE	10 TMC2	TM10 TMS2 MAGNETIC TAPE				
4													10 DSS SINGLE SYNCHRONOUS LINE UNIT	DS10 DSI	10 DSS2 SINGLE SYNCHRONOUS LINE UNIT	DS10 DSI2
5																
6																
7																

CODES IN THIS SECTION RESERVED FOR USER SPECIAL DEVICES

KI10 UNRESTRICTED CODES RESERVED FOR USERS

KI10 UNRESTRICTED CODES RESERVED FOR DEC

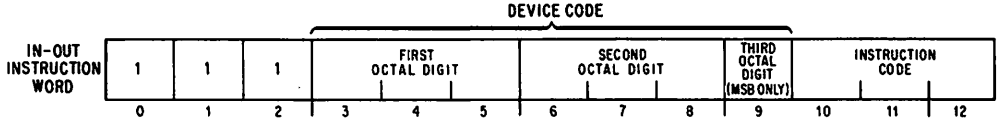
*IN THE PDP-6 THESE CODES ARE USED FOR OTHER DEVICES

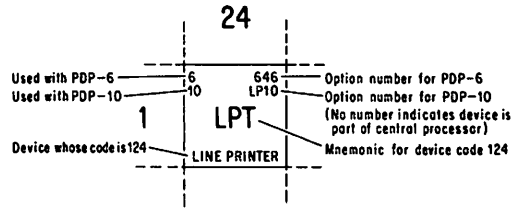
†FOR A THIRD LINE PRINTER USE CODE 230

010 DRUM PROCESSOR

160 PDP-7,8 INTERFACE

270 DISK FILE (DF)





DEVICE MNEMONICS

ALGEBRAIC REPRESENTATION

The remaining pages of this Appendix list, in symbolic form, the actual operations performed by the instructions. The grouping, as given below, differs slightly from that used in Chapter 2.

The terminology and notation used also vary somewhat from that in the body of the manual, as follows.

AC — The accumulator address in bits 9–12 of the instruction word (represented by *A* in the instruction descriptions).

AC+1 — The address one greater than AC, except that AC+1 is 0 if AC is 17.

E — The result of the effective address calculation. E is eighteen bits when used as an address, half word operand, mask or output conditions, but is a signed 9-bit quantity when used as a scale factor or a shift number.

E+1 — The address one greater than E, except that E+1 is 0 if E is 777777.

PC — The 18-bit program counter.

(X) — The word contained in register X.

$(X)_L$ — The left half of (X).

$(X)_R$ — The right half of (X).

$(X)_S$ — The word contained in X with its left and right halves swapped.

A_n — The value of bit n of the quantity A.

A,B — A 36-bit word with the 18-bit quantity A in its left half and the 18-bit quantity B in its right half (either A or B may be 0).

(X,Y) — The contents of registers X and Y concatenated into a double word operand.

$((X))$ — The word contained in the register addressed by (X), *ie* addressed by the word in register X.

$A \rightarrow B$ — The quantity A replaces the quantity B (A and B may be half words, full words or double words). *Eg*

$$(AC) + (E) \rightarrow (AC)$$

means the word in accumulator AC plus the word in memory location E replaces the word in AC.

(AC) (E) — The word in AC and the word in E.

$\wedge \vee \forall \sim$ — The Boolean operators AND, inclusive OR, exclusive OR, and complement (logical negation).

+ − × ÷ || The arithmetic operators for addition, negation or subtraction, multiplication, division, and absolute value (magnitude).

Square brackets are used occasionally for grouping. With respect to the values of their terms, the equations for a given instruction are in chronological order; *eg* in the pair of equations

$$(AC) + 1 \rightarrow (AC)$$
$$If\ (AC) = 0:\ E \rightarrow (PC)$$

the quantity tested in the second equation is the word in AC after it has been incremented by one.

Boolean

SETZ	400	0 → (AC)	SETO	474	777777777777 → (AC)
SETZI	401	0 → (AC)	SETOI	475	777777777777 → (AC)
SETZM	402	0 → (E)	SETOM	476	777777777777 → (E)
SETZB	403	0 → (AC) (E)	SETOB	477	777777777777 → (AC) (E)
SETA	424	(AC) → (AC) [*no-op*]	SETCA	450	~ (AC) → (AC)
SETAI	425	(AC) → (AC) [*no-op*]	SETCAI	451	~ (AC) → (AC)
SETAM	426	(AC) → (E)	SETCAM	452	~ (AC) → (E)
SETAB	427	(AC) → (E)	SETCAB	453	~ (AC) → (AC) (E)
SETM	414	(E) → (AC)	SETCM	460	~ (E) → (AC)
SETMI	415	0,E → (AC)	SETCMI	461	~ [0,E] → (AC)
SETMM	416	(E) → (E) [*no-op*]	SETCMM	462	~ (E) → (E)
SETMB	417	(E) → (AC) (E)	SETCMB	463	~ (E) → (AC) (E)
AND	404	(AC) ∧ (E) → (AC)	ANDCA	410	~ (AC) ∧ (E) → (AC)
ANDI	405	(AC) ∧ 0,E → (AC)	ANDCAI	411	~ (AC) ∧ 0,E → (AC)
ANDM	406	(AC) ∧ (E) → (E)	ANDCAM	412	~ (AC) ∧ (E) → (E)
ANDB	407	(AC) ∧ (E) → (AC) (E)	ANDCAB	413	~ (AC) ∧ (E) → (AC) (E)
ANDCM	420	(AC) ∧ ~ (E) → (AC)	ANDCB	440	~ (AC) ∧ ~ (E) → (AC)
ANDCMI	421	(AC) ∧ ~ [0,E] → (AC)	ANDCBI	441	~ (AC) ∧ ~ [0,E] → (AC)
ANDCMM	422	(AC) ∧ ~ (E) → (E)	ANDCBM	442	~ (AC) ∧ ~ (E) → (E)
ANDCMB	423	(AC) ∧ ~ (E) → (AC) (E)	ANDCBB	443	~ (AC) ∧ ~ (E) → (AC) (E)
IOR	434	(AC) ∨ (E) → (AC)	ORCA	454	~ (AC) ∨ (E) → (AC)
IORI	435	(AC) ∨ 0,E → (AC)	ORCAI	455	~ (AC) ∨ 0,E → (AC)
IORM	436	(AC) ∨ (E) → (E)	ORCAM	456	~ (AC) ∨ (E) → (E)
IORB	437	(AC) ∨ (E) → (AC) (E)	ORCAB	457	~ (AC) ∨ (E) → (AC) (E)
ORCM	464	(AC) ∨ ~ (E) → (AC)	ORCB	470	~ (AC) ∨ ~ (E) → (AC)
ORCMI	465	(AC) ∨ ~ [0,E] → (AC)	ORCBI	471	~ (AC) ∨ ~ [0,E] → (AC)
ORCMM	466	(AC) ∨ ~ (E) → (E)	ORCBM	472	~ (AC) ∨ ~ (E) → (E)
ORCMB	467	(AC) ∨ ~ (E) → (AC) (E)	ORCBB	473	~ (AC) ∨ ~ (E) → (AC) (E)
XOR	430	(AC) ∀ (E) → (AC)	EQV	444	~ [(AC) ∀ (E)] → (AC)
XORI	431	(AC) ∀ 0,E → (AC)	EQVI	445	~ [(AC) ∀ 0,E] → (AC)
XORM	432	(AC) ∀ (E) → (E)	EQVM	446	~ [(AC) ∀ (E)] → (E)
XORB	433	(AC) ∀ (E) → (AC) (E)	EQVB	447	~ [(AC) ∀ (E)] → (AC) (E)

Byte Manipulation

IBP	133	*Operations on* (E) [*see page 2-16*] *If* P − S ⩾ 0: P − S → P *If* P − S < 0: Y + 1 → Y 36 − S → P
LDB	135	BYTE IN ((E)) → (AC) [*see page 2-16*]
DPB	137	BYTE IN (AC) → BYTE IN ((E)) [*see page 2-16*]
ILDB	134	IBP *and* LDB
IDPB	136	IBP *and* DPB

Fixed Point Arithmetic

ADD	270	(AC) + (E) → (AC)	SUB	274	(AC) − (E) → (AC)
ADDI	271	(AC) + 0,E → (AC)	SUBI	275	(AC) − 0,E → (AC)
ADDM	272	(AC) + (E) → (E)	SUBM	276	(AC) − (E) → (E)
ADDB	273	(AC) + (E) → (AC) (E)	SUBB	277	(AC) − (E) → (AC) (E)
IMUL	220	(AC) × (E) → (AC)*	MUL	224	(AC) × (E) → (AC,AC+1)
IMULI	221	(AC) × 0,E → (AC)*	MULI	225	(AC) × 0,E → (AC,AC+1)
IMULM	222	(AC) × (E) → (E)*	MULM	226	(AC) × (E) → (E)†
IMULB	223	(AC) × (E) → (AC) (E)*	MULB	227	(AC) × (E) → (AC,AC+1) (E)
IDIV	230	(AC) ÷ (E) → (AC) REMAINDER → (AC+1)	DIV	234	(AC,AC+1) ÷ (E) → (AC) REMAINDER → (AC+1)
IDIVI	231	(AC) ÷ 0,E → (AC) REMAINDER → (AC+1)	DIVI	235	(AC,AC+1) ÷ 0,E → (AC) REMAINDER → (AC+1)
IDIVM	232	(AC) ÷ (E) → (E)	DIVM	236	(AC,AC+1) ÷ (E) → (E)
IDIVB	233	(AC) ÷ (E) → (AC) (E) REMAINDER → (AC+1)	DIVB	237	(AC,AC+1) ÷ (E) → (AC) (E) REMAINDER → (AC+1)

*The high order word of the product is discarded.
†The low order word of the product is discarded.

Floating Point Arithmetic

FAD	140	(AC) + (E) → (AC)	FADR	144	(AC) + (E) → (AC)
FADL	141	(AC) + (E) → (AC,AC+1)	FADRI	145	(AC) + E,0 → (AC)
FADM	142	(AC) + (E) → (E)	FADRM	146	(AC) + (E) → (E)
FADB	143	(AC) + (E) → (AC) (E)	FADRB	147	(AC) + (E) → (AC) (E)
FSB	150	(AC) − (E) → (AC)	FSBR	154	(AC) − (E) → (AC)
FSBL	151	(AC) − (E) → (AC,AC+1)	FSBRI	155	(AC) − E,0 → (AC)
FSBM	152	(AC) − (E) → (E)	FSBRM	156	(AC) − (E) → (E)
FSBB	153	(AC) − (E) → (AC) (E)	FSBRB	157	(AC) − (E) → (AC) (E)

FMP	160	(AC) X (E) → (AC)	FMPR	164	(AC) X (E) → (AC)
FMPL	161	(AC) X (E) → (AC,AC+1)	FMPRI	165	(AC) X E,0 → (AC)
FMPM	162	(AC) X (E) → (E)	FMPRM	166	(AC) X (E) → (E)
FMPB	163	(AC) X (E) → (AC) (E)	FMPRB	167	(AC) X (E) → (AC) (E)
FDV	170	(AC) ÷ (E) → (AC)	FDVR	174	(AC) ÷ (E) → (AC)
FDVL	171	(AC) ÷ (E) → (AC) REMAINDER → (AC+1)	FDVRI	175	(AC) ÷ E,0 → (AC)
FDVM	172	(AC) ÷ (E) → (E)	FDVRM	176	(AC) ÷ (E) → (E)
FDVB	173	(AC) ÷ (E) → (AC) (E)	FDVRB	177	(AC) ÷ (E) → (AC) (E)

		UFA	130	(AC) + (E) → (AC+1) *without normalization*	
		DFN	131	− (AC,E) → (AC,E)	
		FSC	132	(AC) X 2^E → (AC)	
		FLTR	127	(E) *floated, rounded* → (AC)	
FIX	122	(E) *fixed* → (AC)	FIXR	126	(E) *fixed, rounded* → (AC)

DFAD	110	(AC,AC+1) + (E,E+1) → (AC,AC+1)	
DFSB	111	(AC,AC+1) − (E,E+1) → (AC,AC+1)	
DFMP	112	(AC,AC+1) X (E,E+1) → (AC,AC+1)	
DFDV	113	(AC,AC+1) ÷ (E,E+1) → (AC,AC+1)	

DMOVE	120	(E,E+1) → (AC,AC+1)	DMOVEM	124	(AC,AC+1) → (E,E+1)
DMOVN	121	− (E,E+1) → (AC,AC+1)	DMOVNM	125	− (AC,AC+1) → (E,E+1)

Full Word Data Transmission

EXCH	250	(AC) ↔ (E)
BLT	251	*Move* E − $(AC)_R$ + 1 *words starting with* $((AC)_L)$ → $((AC)_R)$ [*see page 2-10*]

MOVE	200	(E) → (AC)	MOVS	204	$(E)_S$ → (AC)
MOVEI	201	0,E → (AC)	MOVSI	205	E,0 → (AC)
MOVEM	202	(AC) → (E)	MOVSM	206	$(AC)_S$ → (E)
MOVES	203	*If* AC ≠ 0: (E) → (AC)	MOVSS	207	$(E)_S$ → (E) *If* AC ≠ 0: (E) → (AC)
MOVN	210	− (E) → (AC)	MOVM	214	|(E)| → (AC)
MOVNI	211	− [0,E] → (AC)	MOVMI	215	0,E → (AC)
MOVNM	212	− (AC) → (E)	MOVMM	216	|(AC)| → (E)
MOVNS	213	− (E) → (E) *If* AC ≠ 0: (E) → (AC)	MOVMS	217	|(E)| → (E) *If* AC ≠ 0: (E) → (AC)

Half Word Data Transmission

HLL	500	$(E)_L \rightarrow (AC)_L$	HLLZ	510	$(E)_L,0 \rightarrow (AC)$
HLLI	501	$0 \rightarrow (AC)_L$	HLLZI	511	$0 \rightarrow (AC)$
HLLM	502	$(AC)_L \rightarrow (E)_L$	HLLZM	512	$(AC)_L,0 \rightarrow (E)$
HLLS	503	*If* AC ≠ 0: $(E) \rightarrow (AC)$	HLLZS	513	$0 \rightarrow (E)_R$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HLLO	520	$(E)_L,777777 \rightarrow (AC)$	HLLE	530	$(E)_L,[(E)_0 \times 777777] \rightarrow (AC)$
HLLOI	521	$0,777777 \rightarrow (AC)$	HLLEI	531	$0 \rightarrow (AC)$
HLLOM	522	$(AC)_L,777777 \rightarrow (E)$	HLLEM	532	$(AC)_L,[(AC)_0 \times 777777] \rightarrow (E)$
HLLOS	523	$777777 \rightarrow (E)_R$ *If* AC ≠ 0: $(E) \rightarrow (AC)$	HLLES	533	$(E)_0 \times 777777 \rightarrow (E)_R$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HLR	544	$(E)_L \rightarrow (AC)_R$	HLRZ	554	$0,(E)_L \rightarrow (AC)$
HLRI	545	$0 \rightarrow (AC)_R$	HLRZI	555	$0 \rightarrow (AC)$
HLRM	546	$(AC)_L \rightarrow (E)_R$	HLRZM	556	$0,(AC)_L \rightarrow (E)$
HLRS	547	$(E)_L \rightarrow (E)_R$ *If* AC ≠ 0: $(E) \rightarrow (AC)$	HLRZS	557	$0,(E)_L \rightarrow (E)$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HLRO	564	$777777,(E)_L \rightarrow (AC)$	HLRE	574	$[(E)_0 \times 777777],(E)_L \rightarrow (AC)$
HLROI	565	$777777,0 \rightarrow (AC)$	HLREI	575	$0 \rightarrow (AC)$
HLROM	566	$777777,(AC)_L \rightarrow (E)$	HLREM	576	$[(AC)_0 \times 777777],(AC)_L \rightarrow (E)$
HLROS	567	$777777,(E)_L \rightarrow (E)$ *If* AC ≠ 0: $(E) \rightarrow (AC)$	HLRES	577	$[(E)_0 \times 777777],(E)_L \rightarrow (E)$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HRR	540	$(E)_R \rightarrow (AC)_R$	HRRZ	550	$0,(E)_R \rightarrow (AC)$
HRRI	541	$E \rightarrow (AC)_R$	HRRZI	551	$0,E \rightarrow (AC)$
HRRM	542	$(AC)_R \rightarrow (E)_R$	HRRZM	552	$0,(AC)_R \rightarrow (E)$
HRRS	543	*If* AC ≠ 0: $(E) \rightarrow (AC)$	HRRZS	553	$0 \rightarrow (E)_L$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HRRO	560	$777777,(E)_R \rightarrow (AC)$	HRRE	570	$[(E)_{18} \times 777777],(E)_R \rightarrow (AC)$
HRROI	561	$777777,E \rightarrow (AC)$	HRREI	571	$[E_{18} \times 777777],E \rightarrow (AC)$
HRROM	562	$777777,(AC)_R \rightarrow (E)$	HRREM	572	$[(AC)_{18} \times 777777],(AC)_R \rightarrow (E)$
HRROS	563	$777777 \rightarrow (E)_L$ *If* AC ≠ 0: $(E) \rightarrow (AC)$	HRRES	573	$(E)_{18} \times 777777 \rightarrow (E)_L$ *If* AC ≠ 0: $(E) \rightarrow (AC)$
HRL	504	$(E)_R \rightarrow (AC)_L$	HRLZ	514	$(E)_R,0 \rightarrow (AC)$
HRLI	505	$E \rightarrow (AC)_L$	HRLZI	515	$E,0 \rightarrow (AC)$
HRLM	506	$(AC)_R \rightarrow (E)_L$	HRLZM	516	$(AC)_R,0 \rightarrow (E)$
HRLS	507	$(E)_R \rightarrow (E)_L$ *If* AC ≠ 0: $(E) \rightarrow (AC)$	HRLZS	517	$(E)_R,0 \rightarrow (E)$ *If* AC ≠ 0: $(E) \rightarrow (AC)$

HRLO	524	$(E)_R$,777777 → (AC)	HRLE	534	$(E)_R$,[$(E)_{18}$ X 777777] → (AC)
HRLOI	525	E,777777 → (AC)	HRLEI	535	E,[E_{18} X 777777] → (AC)
HRLOM	526	$(AC)_R$,777777 → (E)	HRLEM	536	$(AC)_R$,[$(AC)_{18}$ X 777777] → (E)
HRLOS	527	$(E)_R$,777777 → (E) *If* AC ≠ 0: (E) → (AC)	HRLES	537	$(E)_R$,[$(E)_{18}$ X 777777] → (E) *If* AC ≠ 0: (E) → (AC)

In-out

CONO	70020	E → COMMAND	CONSZ	70030	*If* $STATUS_R$ ∧ E = 0: *skip*
CONI	70024	STATUS → (E)	CONSO	70034	*If* $STATUS_R$ ∧ E ≠ 0: *skip*
DATAO	70014	(E) → DATA	DATAI	70004	DATA → (E)

BLKO	70010	(E) + 1000001 → (E)*	$((E)_R)$ → DATA [*see page 2-83*]
BLKI	70000	(E) + 1000001 → (E)*	DATA → $((E)_R)$ [*see page 2-83*]

Program Control

JSR	264	FLAGS,(PC) → (E)	E + 1 → (PC)	
JSP	265	FLAGS,(PC) → (AC)	E → (PC)	
JRST	254	E → (PC)	[*If* AC ≠ 0, *see page 2-63*]	
JSA	266	(AC) → (E)	E,(PC) → (AC)	E + 1 → (PC)
JRA	267	E → (PC)	$((AC)_L)$ → (AC)	
JFCL	255	*If* AC ∧ FLAGS ≠ 0:	E → (PC)	~ AC ∧ FLAGS → FLAGS
XCT	256	*Execute* (E)		
JFFO	243	*If* (AC) = 0: 0 → (AC + 1) *If* (AC) ≠ 0: E → (PC) [*see page 2-61*]		
MAP	257	PHYSICAL MAP DATA → (AC)		

Pushdown List

PUSH	261	(AC) + 1000001 → (AC)*	(E) → $((AC)_R)$	
POP	262	$((AC)_R)$ → (E)	(AC) − 1000001 → (AC)*	
PUSHJ	260	(AC) + 1000001 → (AC)*	FLAGS,(PC) → $((AC)_R)$	E → (PC)
POPJ	263	$((AC)_R)_R$ → (PC)	(AC) − 1000001 → (AC)*	

Shift and Rotate

ASH	240	(AC) X 2^E → (AC)	ASHC	244	(AC,AC+1) X 2^E → (AC,AC+1)
ROT	241	*Rotate* (AC) E *places*	ROTC	245	*Rotate* (AC,AC+1) E *places*
LSH	242	*Shift* (AC) E *places*	LSHC	246	*Shift* (AC,AC+1) E *places*

*In the KI10, 1 is added to or subtracted from each half separately.

Arithmetic Testing

AOBJP	252	(AC) + 1000001 → (AC)*	*If* (AC) ⩾ 0: E → (PC)
AOBJN	253	(AC) + 1000001 → (AC)*	*If* (AC) < 0: E → (PC)

CAI	300	*No-op*	CAM	310	*No-op*
CAIL	301	*If* (AC) < E: *skip*	CAML	311	*If* (AC) < (E): *skip*
CAIE	302	*If* (AC) = E: *skip*	CAME	312	*If* (AC) = (E): *skip*
CAILE	303	*If* (AC) ⩽ E: *skip*	CAMLE	313	*If* (AC) ⩽ (E): *skip*
CAIA	304	*Skip*	CAMA	314	*Skip*
CAIGE	305	*If* (AC) ⩾ E: *skip*	CAMGE	315	*If* (AC) ⩾ (E): *skip*
CAIN	306	*If* (AC) ≠ E: *skip*	CAMN	316	*If* (AC) ≠ (E): *skip*
CAIG	307	*If* (AC) > E: *skip*	CAMG	317	*If* (AC) > (E): *skip*
JUMP	320	*No-op*	SKIP	330	*If* AC ≠ 0: (E) → (AC)
JUMPL	321	*If* (AC) < 0: E → (PC)	SKIPL	331	*If* AC ≠ 0: (E) → (AC) *If* (E) < 0: *skip*
JUMPE	322	*If* (AC) = 0: E → (PC)	SKIPE	332	*If* AC ≠ 0: (E) → (AC) *If* (E) = 0: *skip*
JUMPLE	323	*If* (AC) ⩽ 0: E → (PC)	SKIPLE	333	*If* AC ≠ 0: (E) → (AC) *If* (E) ⩽ 0: *skip*
JUMPA	324	E → (PC)	SKIPA	334	*If* AC ≠ 0: (E) → (AC) *Skip*
JUMPGE	325	*If* (AC) ⩾ 0: E → (PC)	SKIPGE	335	*If* AC ≠ 0: (E) → (AC) *If* (E) ⩾ 0: *skip*
JUMPN	326	*If* (AC) ≠ 0: E → (PC)	SKIPN	336	*If* AC ≠ 0: (E) → (AC) *If* (E) ≠ 0: *skip*
JUMPG	327	*If* (AC) > 0: E → (PC)	SKIPG	337	*If* AC ≠ 0: (E) → (AC) *If* (E) > 0: *skip*
AOJ	340	(AC) + 1 → (AC)	SOJ	360	(AC) − 1 → (AC)
AOJL	341	(AC) + 1 → (AC) *If* (AC) < 0: E → (PC)	SOJL	361	(AC) − 1 → (AC) *If* (AC) < 0: E → (PC)
AOJE	342	(AC) + 1 → (AC) *If* (AC) = 0: E → (PC)	SOJE	362	(AC) − 1 → (AC) *If* (AC) = 0: E → (PC)
AOJLE	343	(AC) + 1 → (AC) *If* (AC) ⩽ 0: E → (PC)	SOJLE	363	(AC) − 1 → (AC) *If* (AC) ⩽ 0: E → (PC)
AOJA	344	(AC) + 1 → (AC) E → (PC)	SOJA	364	(AC) − 1 → (AC) E → (PC)
AOJGE	345	(AC) + 1 → (AC) *If* (AC) ⩾ 0: E → (PC)	SOJGE	365	(AC) − 1 → (AC) *If* (AC) ⩾ 0: E → (PC)

*In the KI10, 1 is added to or subtracted from each half separately.

AOJN	346	(AC) + 1 → (AC) *If* (AC) ≠ 0: E → (PC)	SOJN	366	(AC) − 1 → (AC) *If* (AC) ≠ 0: E → (PC)
AOJG	347	(AC) + 1 → (AC) *If* (AC) > 0: E → (PC)	SOJG	367	(AC) − 1 → (AC) *If* (AC) > 0: E → (PC)
AOS	350	(E) + 1 → (E) *If* (AC) ≠ 0: (E) → (AC)	SOS	370	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC)
AOSL	351	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) < 0: *skip*	SOSL	371	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) < 0: *skip*
AOSE	352	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) = 0: *skip*	SOSE	372	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) = 0: *skip*
AOSLE	353	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ⩽ 0: *skip*	SOSLE	373	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ⩽ 0: *skip*
AOSA	354	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *Skip*	SOSA	374	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *Skip*
AOSGE	355	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ⩾ 0: *skip*	SOSGE	375	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ⩾ 0: *skip*
AOSN	356	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ≠ 0: *skip*	SOSN	376	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) ≠ 0: *skip*
AOSG	357	(E) + 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) > 0: *skip*	SOSG	377	(E) − 1 → (E) *If* AC ≠ 0: (E) → (AC) *If* (E) > 0: *skip*

Logical Testing and Modification

TLN	601	*No-op*	TRN	600	*No-op*	
TLNE	603	*If* $(AC)_L \wedge E = 0$: *skip*	TRNE	602	*If* $(AC)_R \wedge E = 0$: *skip*	
TLNA	605	*Skip*	TRNA	604	*Skip*	
TLNN	607	*If* $(AC)_L \wedge E \neq 0$: *skip*	TRNN	606	*If* $(AC)_R \wedge E \neq 0$: *skip*	
TLZ	621	$(AC)_L \wedge \sim E \rightarrow (AC)_L$	TRZ	620	$(AC)_R \wedge \sim E \rightarrow (AC)_R$	
TLZE	623	*If* $(AC)_L \wedge E = 0$: *skip* $(AC)_L \wedge \sim E \rightarrow (AC)_L$	TRZE	622	*If* $(AC)_R \wedge E = 0$: *skip* $(AC)_R \wedge \sim E \rightarrow (AC)_R$	
TLZA	625	$(AC)_L \wedge \sim E \rightarrow (AC)_L$ *skip*	TRZA	624	$(AC)_R \wedge \sim E \rightarrow (AC)_R$	*skip*
TLZN	627	*If* $(AC)_L \wedge E \neq 0$: *skip* $(AC)_L \wedge \sim E \rightarrow (AC)_L$	TRZN	626	*If* $(AC)_R \wedge E \neq 0$: *skip* $(AC)_R \wedge \sim E \rightarrow (AC)_R$	

TLC	641	$(AC)_L \forall E \rightarrow (AC)_L$	TRC	640	$(AC)_R \forall E \rightarrow (AC)_R$
TLCE	643	*If* $(AC)_L \wedge E = 0$: *skip* $(AC)_L \forall E \rightarrow (AC)_L$	TRCE	642	*If* $(AC)_R \wedge E = 0$: *skip* $(AC)_R \forall E \rightarrow (AC)_R$
TLCA	645	$(AC)_L \forall E \rightarrow (AC)_L$ *skip*	TRCA	644	$(AC)_R \forall E \rightarrow (AC)_R$ *skip*
TLCN	647	*If* $(AC)_L \wedge E \neq 0$: *skip* $(AC)_L \forall E \rightarrow (AC)_L$	TRCN	646	*If* $(AC)_R \wedge E \neq 0$: *skip* $(AC)_R \forall E \rightarrow (AC)_R$
TLO	661	$(AC)_L \vee E \rightarrow (AC)_L$	TRO	660	$(AC)_R \vee E \rightarrow (AC)_R$
TLOE	663	*If* $(AC)_L \wedge E = 0$: *skip* $(AC)_L \vee E \rightarrow (AC)_L$	TROE	662	*If* $(AC)_R \wedge E = 0$: *skip* $(AC)_R \vee E \rightarrow (AC)_R$
TLOA	665	$(AC)_L \vee E \rightarrow (AC)_L$ *skip*	TROA	664	$(AC)_R \vee E \rightarrow (AC)_R$ *skip*
TLON	667	*If* $(AC)_L \wedge E \neq 0$: *skip* $(AC)_L \vee E \rightarrow (AC)_L$	TRON	666	*If* $(AC)_R \wedge E \neq 0$: *skip* $(AC)_R \vee E \rightarrow (AC)_R$
TDN	610	*No-op*	TSN	611	*No-op*
TDNE	612	*If* $(AC) \wedge (E) = 0$: *skip*	TSNE	613	*If* $(AC) \wedge (E)_S = 0$: *skip*
TDNA	614	*Skip*	TSNA	615	*Skip*
TDNN	616	*If* $(AC) \wedge (E) \neq 0$: *skip*	TSNN	617	*If* $(AC) \wedge (E)_S \neq 0$: *skip*
TDZ	630	$(AC) \wedge \sim (E) \rightarrow (AC)$	TSZ	631	$(AC) \wedge \sim (E)_S \rightarrow (AC)$
TDZE	632	*If* $(AC) \wedge (E) = 0$: *skip* $(AC) \wedge \sim (E) \rightarrow (AC)$	TSZE	633	*If* $(AC) \wedge (E)_S = 0$: *skip* $(AC) \wedge \sim (E)_S \rightarrow (AC)$
TDZA	634	$(AC) \wedge \sim (E) \rightarrow (AC)$ *skip*	TSZA	635	$(AC) \wedge \sim (E)_S \rightarrow (AC)$ *skip*
TDZN	636	*If* $(AC) \wedge (E) \neq 0$: *skip* $(AC) \wedge \sim (E) \rightarrow (AC)$	TSZN	637	*If* $(AC) \wedge (E)_S \neq 0$: *skip* $(AC) \wedge \sim (E)_S \rightarrow (AC)$
TDC	650	$(AC) \forall (E) \rightarrow (AC)$	TSC	651	$(AC) \forall (E)_S \rightarrow (AC)$
TDCE	652	*If* $(AC) \wedge (E) = 0$: *skip* $(AC) \forall (E) \rightarrow (AC)$	TSCE	653	*If* $(AC) \wedge (E)_S = 0$: *skip* $(AC) \forall (E)_S \rightarrow (AC)$
TDCA	654	$(AC) \forall (E) \rightarrow (AC)$ *skip*	TSCA	655	$(AC) \forall (E)_S \rightarrow (AC)$ *skip*
TDCN	656	*If* $(AC) \wedge (E) \neq 0$: *skip* $(AC) \forall (E) \rightarrow (AC)$	TSCN	657	*If* $(AC) \wedge (E)_S \neq 0$: *skip* $(AC) \forall (E)_S \rightarrow (AC)$
TDO	670	$(AC) \vee (E) \rightarrow (AC)$	TSO	671	$(AC) \vee (E)_S \rightarrow (AC)$
TDOE	672	*If* $(AC) \wedge (E) = 0$: *skip* $(AC) \vee (E) \rightarrow (AC)$	TSOE	673	*If* $(AC) \wedge (E)_S = 0$: *skip* $(AC) \vee (E)_S \rightarrow (AC)$
TDOA	674	$(AC) \vee (E) \rightarrow (AC)$ *skip*	TSOA	675	$(AC) \vee (E)_S \rightarrow (AC)$ *skip*
TDON	676	*If* $(AC) \wedge (E) \neq 0$: *skip* $(AC) \vee (E) \rightarrow (AC)$	TSON	677	*If* $(AC) \wedge (E)_S \neq 0$: *skip* $(AC) \vee (E)_S \rightarrow (AC)$

POWERS OF TWO

2^N	N	2^{-N}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 068 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

APPENDIX B

INPUT-OUTPUT CODES

The table beginning on the next page lists the complete 1968 ASCII code (ANSI X3.4-1968). The software handles the full character set, and for a program that does not handle lower case, it translates input codes 140–174 into the corresponding upper case codes (100–134) and translates both 175 and 176 into 033, escape. The actual character sets available on different terminals vary greatly, but usually a terminal without lower case will accept lower case codes, printing the corresponding upper case character. The definitions of the control codes are those given by ASCII; most control codes, however, have no effect on the console terminal, and the definitions bear no necessary relation to the use of the codes in conjunction with the DECsystem–10 software. Brackets enclose earlier definitions of control codes (mostly 1963 ASCII). The table includes bit 8 as an even parity bit, the form generally used for paper tape and asynchronous operations; odd parity is generally used for magnetic tape and synchronous operations.

Output codes are simply passed on to the terminal as they are, with the expectation that the terminal will ignore irrevelant control codes, and that a terminal that lacks lower case will print the corresponding upper case. A terminal that fails to live up to these assumptions will generally not operate satisfactorily with the DECsystem–10 software.

With all line printers, ten control characters are used for format control, and the interface also recognizes null for fill and delete for selecting hidden characters. The 64-character print set includes the figures and upper case; lower case is added for the 96-character set (with the smaller print set, giving a lower case code prints the upper case character). The larger print set includes a character hidden under delete, a feature that is optional on the LP10D and H. The printable characters are generally those defined by ASCII, with little if any variation. The 128-character printer uses the entire set of 7-bit codes for printable characters, with characters hidden under the ten control codes that affect the printer and also under null and delete.

The first two pages of the table of card codes [*pages B-8 to B-12*] list the column punches required to represent characters in the ASCII card code. When reading cards, the software translates the column punch into the octal code shown; when punching cards, it produces the listed column punch when given the corresponding code. There are also a few control hole patterns that the software responds to but does not translate. The next page lists two earlier DEC card codes that have only the figure and upper case character subset, plus a few control punches. The remaining pages of the table show the relationship among the early DEC card codes, the corresponding characters in the ASCII set, and several IBM card punches. Each column punch is produced by a single key on any keypunch for which a character is listed, the character being that which is printed at the top of the card.

ASCII CODE

Even Parity Bit	7-Bit Decimal	7-Bit Octal	Character	Remarks
0	000	000	NUL	Null, tape feed. Control shift P.
1	001	001	SOH	Start of heading [SOM, start of message]. Control A.
1	002	002	STX	Start of text [EOA, end of address]. Control B.
0	003	003	ETX	End of text [EOM, end of message]. Control C.
1	004	004	EOT	End of transmission; shuts off TWX machines and disconnects some data sets. Control D.
0	005	005	ENQ	Enquiry [WRU, "Who are you?"]. Triggers identification ("Here is . . .") at remote station if so equipped. Control E.
0	006	006	ACK	Acknowledge [RU, "Are you . . . ?"]. Control F.
1	007	007	BEL	Rings the bell. Control G.
1	008	010	BS	Backspace. Control H.
0	009	011	HT	Horizontal tab. Control I.
0	010	012	LF	Line feed. Control J.
1	011	013	VT	Vertical tab. Control K.
0	012	014	FF	Form feed to top of next page. Control L.
1	013	015	CR	Carriage return to beginning of line. Control M.
1	014	016	SO	Shift out; change character set or change ribbon color to red. Control N.
0	015	017	SI	Shift in; return to standard character set or color. Control O.
1	016	020	DLE	Data link escape [DCO]. Control P.
0	017	021	DC1	Device control 1, turns transmitter (reader) on. Control Q (X ON).
0	018	022	DC2	Device control 2, turns punch or auxiliary on. Control R (TAPE, AUX ON).
1	019	023	DC3	Device control 3, turns transmitter (reader) off. Control S (X OFF).
0	020	024	DC4	Device control 4 (stop), turns punch or auxiliary off. Control T (~~TAPE~~, AUX OFF).
1	021	025	NAK	Negative acknowledge [ERR, error]. Control U.
1	022	026	SYN	Synchronous idle [SYNC]. Control V.
0	023	027	ETB	End of transmission block [LEM, logical end of medium]. Control W.
0	024	030	CAN	Cancel [S_0]. Control X.
1	025	031	EM	End of medium [S_1]. Control Y.
1	026	032	SUB	Substitute [S_2]. Control Z.
0	027	033	ESC	Escape, prefix [S_3]. Control shift K.
1	028	034	FS	File separator [S_4]. Control shift L.
0	029	035	GS	Group separator [S_5]. Control shift M.
0	030	036	RS	Record separator [S_6]. Control shift N.
1	031	037	US	Unit separator [S_7]. Control shift O.

Figures				Upper Case				Lower Case			
Even Parity Bit	7-Bit Decimal	7-Bit Octal	Character	Even Parity Bit	7-Bit Decimal	7-Bit Octal	Character	Even Parity Bit	7-Bit Decimal	7-Bit Octal	Character[10]
1	032	040	SP[1]	1	064	100	@[4]	0	096	140	`[11]
0	033	041	!	0	065	101	A	1	097	141	a
0	034	042	"	0	066	102	B	1	098	142	b
1	035	043	#[2]	1	067	103	C	0	099	143	c
0	036	044	$	0	068	104	D	1	100	144	d
1	037	045	%	1	069	105	E	0	101	145	e
1	038	046	&	1	070	106	F	0	102	146	f
0	039	047	'[3]	0	071	107	G	1	103	147	g
0	040	050	(	0	072	110	H	1	104	150	h
1	041	051	)	1	073	111	I	0	105	151	i
1	042	052	*	1	074	112	J	0	106	152	j
0	043	053	+	0	075	113	K	1	107	153	k
1	044	054	,	1	076	114	L	0	108	154	l
0	045	055	−	0	077	115	M	1	109	155	m
0	046	056	.	0	078	116	N	1	110	156	n
1	047	057	/	1	079	117	O	0	111	157	o
0	048	060	Ø	0	080	120	P	1	112	160	p
1	049	061	1	1	081	121	Q	0	113	161	q
1	050	062	2	1	082	122	R	0	114	162	r
0	051	063	3	0	083	123	S	1	115	163	s
1	052	064	4	1	084	124	T	0	116	164	t
0	053	065	5	0	085	125	U	1	117	165	u
0	054	066	6	0	086	126	V	1	118	166.	v
1	055	067	7	1	087	127	W	0	119	167	w
1	056	070	8	1	088	130	X	0	120	170	x
0	057	071	9	0	089	131	Y	1	121	171	y
0	058	072	:	0	090	132	Z	1	122	172	z
1	059	073	;	1	091	133	[[5]	0	123	173	{
0	060	074	<	0	092	134	\[6]	1	124	174	¦[12]
1	061	075	=	1	093	135	][7]	0	125	175	}[13]
1	062	076	>	1	094	136	^[8]	0	126	176	~[14]
0	063	077	?	0	095	137	_[9]	1	127	177	DEL[15]

[1]Space.

[2]£ on some (non-DEC) units.

[3]Accent acute or apostrophe – ' before 1965, but used until recently on DEC units.

[4]` 1965–67, but never on DEC units.

[5]Shift K.

[6]~ 1965–67, but never on DEC units. Shift L.

[7]Shift M.

[8]Circumflex – ↑ before 1965, but used until recently on DEC units.

[9]Underscore – ← before 1965, but used until recently on DEC units.

[10]Codes 140–173 first defined in 1965. For a full ASCII character set the Monitor accepts codes 140–176 as lower case. For a character set that lacks lower case, the Monitor translates input codes 140–174 into the corresponding upper case codes (100–134) and translates both 175 and 176 into 033, escape. Early versions of the Monitor used 175 as the escape code and translated both 176 and 033 to it.

[11]Accent grave – @ 1965–67, but never on DEC units.

[12]Control character ACK before 1965; ¬ 1965–67, but never on DEC units. Vertical bar may or may not have gap depending on font design, but generally does not on DEC units.

[13]Unassigned control character (usually ALT MODE) before 1965. Code generated by ALT MODE key on most DEC units.

[14]Control character ESC before 1965; | 1965–67, but never on DEC units. Code generated by ESC key on some DEC units.

[15]Delete, rub out (not part of lower case set).

LINE PRINTER CODE: LP10A, B, C, D, E
Basic Character Set

Control				Figures				Upper Case			
Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character
09	009	011	HT	20	032	040	SP	40	064	100	@
0A	010	012	LF	21	033	041	!	41	065	101	A
0B	011	013	VT	22	034	042	"	42	066	102	B
0C	012	014	FF	23	035	043	#	43	067	103	C
0D	013	015	CR	24	036	044	$	44	068	104	D
				25	037	045	%	45	069	105	E
10	016	020	DLE	26	038	046	&	46	070	106	F
11	017	021	DC1	27	039	047	'	47	071	107	G
12	018	022	DC2	28	040	050	(	48	072	110	H
13	019	023	DC3	29	041	051	)	49	073	111	I
14	020	024	DC4	2A	042	052	*	4A	074	112	J
				2B	043	053	+	4B	075	113	K
00	000	000	NUL	2C	044	054	,	4C	076	114	L
7F	127	177	DEL	2D	045	055	-	4D	077	115	M
				2E	046	056	.	4E	078	116	N
				2F	047	057	/	4F	079	117	O
				30	048	060	0	50	080	120	P
				31	049	061	1	51	081	121	Q
				32	050	062	2	52	082	122	R
				33	051	063	3	53	083	123	S
				34	052	064	4	54	084	124	T
				35	053	065	5	55	085	125	U
				36	054	066	6	56	086	126	V
				37	055	067	7	57	087	127	W
				38	056	070	8	58	088	130	X
				39	057	071	9	59	089	131	Y
				3A	058	072	:	5A	090	132	Z
				3B	059	073	;	5B	091	133	[
				3C	060	074	<	5C	092	134	\
				3D	061	075	=	5D	093	135	]
				3E	062	076	>	5E	094	136	↑
				3F	063	077	?	5F	095	137	←

Additional Characters – 95, 96 and 128 Character Sets

LP10D, E				LP10E			
Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character
60	096	140	`	7F/00	127/000	177/000	· *NUL*
61	097	141	a	01	001	001	↓ *SOH*
62	098	143	b	02	002	002	α *STX*
63	099	143	c	03	003	003	β *ETX*
64	100	144	d	04	004	004	Λ *EOT*
65	101	145	e	05	005	005	¬ *ENQ*
66	102	146	f	06	006	006	ϵ *ACK*
67	103	147	g	07	007	007	π *BEL*
68	104	150	h	08	008	010	λ *BS*
69	105	151	i	7F/09	127/009	177/011	γ *HT*
6A	106	152	j	7F/0A	127/010	177/012	δ *LF*
6B	107	153	k	7F/0B	127/011	177/013	∫ *VT*
6C	108	154	l	7F/0C	127/012	177/014	± *FF*
6D	109	155	m	7F/0D	127/013	177/015	⊕ *CR*
6E	110	156	n	0E	014	016	∞ *SO*
6F	111	157	o	0F	015	017	∂ *SI*
70	112	160	p	7F/10	127/016	177/020	⊂ *DLE*
71	113	161	q	7F/11	127/017	177/021	⊃ *DC1*
72	114	162	r	7F/12	127/018	177/022	∩ *DC2*
73	115	163	s	7F/13	127/019	177/023	∪ *DC3*
74	116	164	t	7F/14	127/020	177/024	∀ *DC4*
75	117	165	u	15	021	025	∃ *NAK*
76	118	166	v	16	022	026	⊗ *SYN*
77	119	167	w	17	023	027	↔ *ETB*
78	120	170	x	18	024	030	^ *CAN*
79	121	171	y	19	025	031	→ *EM*
7A	122	172	z	1A	026	032	_ *SUB*
7B	123	173	{	1B	027	033	≠ *ESC*
7C	124	174	|	1C	028	034	≤ *FS*
7D	125	175	}	1D	029	035	≥ *GS*
7E	126	176	~	1E	030	036	≡ *RS*
7F/7F	127/127	177/177	¬ *DEL*	1F	031	037	∨ *US*

Code pairs indicate hidden characters. For characters after the 95th, corresponding ASCII control characters are given in italics to facilitate generating codes at a keyboard.

LINE PRINTER CODE: LP10F and H
Basic Character Set

Control				Figures				Upper Case			
Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character
09	009	011	HT	20	032	040	SP	40	064	100	@
0A	010	012	LF	21	033	041	!	41	065	101	A
0B	011	013	VT	22	034	042	"	42	066	102	B
0C	012	014	FF	23	035	043	#	43	067	103	C
0D	013	015	CR	24	036	044	$	44	068	104	D
				25	037	045	%	45	069	105	E
10	016	020	DLE	26	038	046	&	46	070	106	F
11	017	021	DC1	27	039	047	'	47	071	107	G
12	018	022	DC2	28	040	050	(	48	072	110	H
13	019	023	DC3	29	041	051	)	49	073	111	I
14	020	024	DC4	2A	042	052	*	4A	074	112	J
				2B	043	053	+	4B	075	113	K
00	000	000	NUL	2C	044	054	,	4C	076	114	L
7F	127	177	DEL	2D	045	055	−	4D	077	115	M
				2E	046	056	.	4E	078	116	N
				2F	047	057	/	4F	079	117	O
				30	048	060	0 [Ø]	50	080	120	P
				31	049	061	1	51	081	121	Q
				32	050	062	2	52	082	122	R
				33	051	063	3	53	083	123	S
				34	052	064	4	54	084	124	T
				35	053	065	5	55	085	125	U
				36	054	066	6	56	086	126	V
				37	055	067	7	57	087	127	W
				38	056	070	8	58	088	130	X
				39	057	071	9	59	089	131	Y
				3A	058	072	:	5A	090	132	Z [Ƶ]
				3B	059	073	;	5B	091	133	[
				3C	060	074	<	5C	092	134	\
				3D	061	075	=	5D	093	135	]
				3E	062	076	>	5E	094	136	^
				3F	063	077	?	5F	095	137	_

Table gives EDP character set of LP10FE and HE. Brackets enclose substitutions for scientific set, LP10FF and HF.

Additional Characters – LP10H

Hex	Decimal	Octal	Character	Hex	Decimal	Octal	Character
60	096	140	`	70	112	160	p
61	097	141	a	71	113	161	q
62	098	142	b	72	114	162	r
63	099	143	c	73	115	163	s
64	100	144	d	74	116	164	t
65	101	145	e	75	117	165	u
66	102	146	f	76	118	166	v
67	103	147	g	77	119	167	w
68	104	150	h	78	120	170	x
69	105	151	i	79	121	171	y
6A	106	152	j	7A	122	172	z
6B	107	153	k	7B	123	173	{
6C	108	154	l	7C	124	174	|
6D	109	155	m	7D	125	175	}
6E	110	156	n	7E	126	176	~
6F	111	157	o	7F/7F	127/127	177/177	← *DEL*

Character with code 177 is hidden under delete.

ASCII CARD CODE

Octal	Character	Column Punch	Octal	Character	Column Punch	Octal	Character	Column Punch	Octal	Character	Column Punch
000	NUL	12 0 9 8 1	040	SP	*None*	100	@	8 4	140	`	8 1
001	SOH	12 9 1	041	!	12 8 7	101	A	12 1	141	a	12 0 1
002	STX	12 9 2	042	"	8 7	102	B	12 2	142	b	12 0 2
003	ETX	12 9 3	043	#	8 3	103	C	12 3	143	c	12 0 3
004	EOT	9 7	044	$	11 8 3	104	D	12 4	144	d	12 0 4
005	ENQ	0 9 8 5	045	%	0 8 4	105	E	12 5	145	e	12 0 5
006	ACK	0 9 8 6	046	&	12	106	F	12 6	146	f	12 0 6
007	BEL	0 9 8 7	047	'	8 5	107	G	12 7	147	g	12 0 7
010	BS	11 9 6	050	(	12 8 5	110	H	12 8	150	h	12 0 8
011	HT	12 9 6	051	)	11 8 5	111	I	12 9	151	i	12 0 9
012	LF	0 9 5	052	*	11 8 4	112	J	11 1	152	j	12 11 1
013	VT	12 9 8 3	053	+	12 8 6	113	K	11 2	153	k	12 11 2
014	FF	12 9 8 4	054	,	0 8 3	114	L	11 3	154	l	12 11 3
015	CR	12 9 8 5	055	-	11	115	M	11 4	155	m	12 11 4
016	SO	12 9 8 6	056	.	12 8 3	116	N	11 5	156	n	12 11 5
017	SI	12 9 8 7	057	/	0 1	117	O	11 6	157	o	12 11 6
020	DLE	12 11 9 8 1	060	0	0	120	P	11 7	160	p	12 11 7
021	DC1	11 9 1	061	1	1	121	Q	11 8	161	q	12 11 8
022	DC2	11 9 2	062	2	2	122	R	11 9	162	r	12 11 9
023	DC3	11 9 3	063	3	3	123	S	0 2	163	s	11 0 2
024	DC4	9 8 4	064	4	4	124	T	0 3	164	t	11 0 3
025	NAK	9 8 5	065	5	5	125	U	0 4	165	u	11 0 4
026	SYN	9 2	066	6	6	126	V	0 5	166	v	11 0 5
027	ETB	0 9 6	067	7	7	127	W	0 6	167	w	11 0 6
030	CAN	11 9 8	070	8	8	130	X	0 7	170	x	11 0 7
031	EM	11 9 8 1	071	9	9	131	Y	0 8	171	y	11 0 8
032	SUB	9 8 7	072	:	8 2	132	Z	0 9	172	z	11 0 9
033	ESC	0 9 7	073	;	11 8 6	133	[	12 8 2	173	{	12 0
034	FS	11 9 8 4	074	<	12 8 4	134	\	0 8 2	174	¦	12 11
035	GS	11 9 8 5	075	=	8 6	135	]	11 8 2	175	}	11 0
036	RS	11 9 8 6	076	>	0 8 6	136	^	11 8 7	176	~	11 0 1
037	US	11 9 8 7	077	?	0 8 7	137	_	0 8 5	177	DEL	12 9 7

When reading or punching cards, the software translates between the octal codes and column punches listed here. The software also recognizes the following control punches.

Binary	7 9
Mode Switch	12 0 2 4 6 8
End of File	12 11 0 1 6 7 8 9

Column Punch	Character	Column Punch	Character	Column Punch	Character	Column Punch	Character
None	SP	11 8	Q	12 0 4	d	11 8 3	$
12	&	11 9	R	12 0 5	e	11 8 4	*
11	–	0 1	/	12 0 6	f	11 8 5	)
0	0	0 2	S	12 0 7	g	11 8 6	;
1	1	0 3	T	12 0 8	h	11 8 7	^
2	2	0 4	U	12 0 9	i	0 9 5	LF
3	3	0 5	V	12 9 1	SOH	0 9 6	ETB
4	4	0 6	W	12 9 2	STX	0 9 7	ESC
5	5	0 7	X	12 9 3	ETX	0 8 2	\
6	6	0 8	Y	12 9 5	HT	0 8 3	,
7	7	0 9	Z	12 9 7	DEL	0 8 4	%
8	8	9 2	SYN	12 8 2	[	0 8 5	_
9	9	9 7	EOT	12 8 3	.	0 8 6	>
12 11	¦	8 1	`	12 8 4	<	0 8 7	?
12 0	{	8 2	:	12 8 5	(	9 8 4	DC4
12 1	A	8 3	#	12 8 6	+	9 8 5	NAK
12 2	B	8 4	@	12 8 7	!	9 8 7	SUB
12 3	C	8 5	'	11 0 1	~	12 9 8 3	VT
12 4	D	8 6	=	11 0 2	s	12 9 8 4	FF
12 5	E	8 7	"	11 0 3	t	12 9 8 5	CR
12 6	F	12 11 1	j	11 0 4	u	12 9 8 6	SO
12 7	G	12 11 2	k	11 0 5	v	12 9 8 7	SI
12 8	H	12 11 3	l	11 0 6	w	11 9 8 1	EM
12 9	I	12 11 4	m	11 0 7	x	11 9 8 4	FS
11 0	}	12 11 5	n	11 0 8	y	11 9 8 5	GS
11 1	J	12 11 6	o	11 0 9	z	11 9 8 6	RS
11 2	K	12 11 7	p	11 9 1	DC1	11 9 8 7	US
11 3	L	12 11 8	q	11 9 2	DC2	0 9 8 5	ENQ
11 4	M	12 11 9	r	11 9 3	DC3	0 9 8 6	ACK
11 5	N	12 0 1	a	11 9 6	BS	0 9 8 7	BEL
11 6	O	12 0 2	b	11 9 8	CAN	12 11 9 8 1	DLE
11 7	P	12 0 3	c	11 8 2	]	12 0 9 8 1	NUL

7 9	*Binary*
12 0 2 4 6 8	*Mode Switch*
12 11 0 1 6 7 8 9	*End of File*

EARLY DEC CARD CODES

Character	7-Bit Octal	DEC 029	DEC 026	Character	7-Bit Octal	DEC 029	DEC 026
Space	040	*None*	*None*	@	100	8 4	8 4
!	041	11 8 2*	12 8 7	A	101	12 1	12 1
"	042	8 7	0 8 5	B	102	12 2	12 2
#	043	8 3	0 8 6	C	103	12 3	12 3
$	044	11 8 3	11 8 3	D	104	12 4	12 4
%	045	0 8 4	0 8 7	E	105	12 5	12 5
&	046	12	11 8 7	F	106	12 6	12 6
'	047	8 5	8 6	G	107	12 7	12 7
(	050	12 8 5	0 8 4	H	110	12 8	12 8
)	051	11 8 5	12 8 4	I	111	12 9	12 9
*	052	11 8 4	11 8 4	J	112	11 1	11 1
+	053	12 8 6	12	K	113	11 2	11 2
,	054	0 8 3	0 8 3	L	114	11 3	11 3
−	055	11	11	M	115	11 4	11 4
.	056	12 8 3	12 8 3	N	116	11 5	11 5
/	057	0 1	0 1	O	117	11 6	11 6
0	060	0	0	P	120	11 7	11 7
1	061	1	1	Q	121	11 8	11 8
2	062	2	2	R	122	11 9	11 9
3	063	3	3	S	123	0 2	0 2
4	064	4	4	T	124	0 3	0 3
5	065	5	5	U	125	0 4	0 4
6	066	6	6	V	126	0 5	0 5
7	067	7	7	W	127	0 6	0 6
8	070	8	8	X	130	0 7	0 7
9	071	9	9	Y	131	0 8	0 8
:	072	8 2	11 8 2 *or* 11 0†	Z	132	0 9	0 9
;	073	11 8 6	0 8 2	[	133	12 8 2	11 8 5
<	074	12 8 4	12 8 6	\	134	11 8 7*	8 7
=	075	8 6	8 3	]	135	0 8 2*	12 8 5
>	076	0 8 6	11 8 6	^	136	12 8 7*	8 5
?	077	0 8 7	12 8 2 *or* 12 0†	_	137	0 8 5	8 2

Binary	7 9
Mode Switch	12 0 2 4 6 8
End of File	12 11 0 1 6 7 8 9

†The Monitor accepts either punch for input but outputs only the triple punch.

These two DEC card codes provide a representation for the figure and upper case character subset. DEC 029 is not available in all programs, but it is almost identical to the ASCII subset, differing only in the four column punches indicated by asterisks as follows:

DEC 029	11 8 2	11 8 7	0 8 2	12 8 7
ASCII	12 8 7	0 8 2	11 8 2	11 8 7

The next two pages show the relationship among the various character sets for the column punches listed

Column Punch	Character	Column Punch	Character
None	*Space*	12 9	I
0	0	11 1	J
1	1	11 2	K
2	2	11 3	L
3	3	11 4	M
4	4	11 5	N
5	5	11 6	O
6	6	11 7	P
7	7	11 8	Q
8	8	11 9	R
9	9	0 1	/
12 1	A	0 2	S
12 2	B	0 3	T
12 3	C	0 4	U
12 4	D	0 5	V
12 5	E	0 6	W
12 6	F	0 7	X
12 7	G	0 8	Y
12 8	H	0 9	Z

Column Punch	026 Data Processing	026 Fortran	029	DEC 026	DEC 029	ASCII
12	&	+	&	+	&	&
11	–	–	–	–	–	–
12 0				?		
11 0				:		
8 2			:	_	:	:
8 3	#	=	#	=	#	#
8 4	@	-	@	@	@	@
8 5			'	^	'	'
8 6			=	'	=	=
8 7			"	\	"	"
12 8 2			¢	?	[	[
12 8 3			.	.	.	.

above, and where they exist, the corresponding single-key punch configurations and printed characters for several IBM key punches.

Column Punch	026 Data Processing	026 Fortran	029	DEC 026	DEC 029	ASCII
12 8 4	¤	)	<	)	<	<
12 8 5			(	]	(	(
12 8 6			+	<	+	+
12 8 7			|	!	^	!
11 8 2			!	:	!	]
11 8 3	$	$	$	$	$	$
11 8 4	*	*	*	*	*	*
11 8 5			)	[	)	)
11 8 6			;	>	;	;
11 8 7			¬	&	\	^
0 8 2			*See note*	;	]	\
0 8 3	,	,	,	,	,	,
0 8 4	%	(	%	(	%	%
0 8 5			←	"	_	_
0 8 6			>	#	>	>
0 8 7			?	%	?	?
7 9				*Binary*	*Binary*	EOT
12 0 2 4 6 8				*Mode Switch*	*Mode Switch*	
12 11 0 1 6 7 8 9				*End of File*	*End of File*	

NOTE: There is a single key for the 0 8 2 punch on the 029 but printing is suppressed.