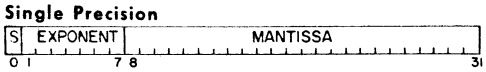
ASCII CHARACTER CODES (cont.)

Decimal	7-bit Octal	Character	To Produce On TTY Cntrl	Mod 33,35 Shift	Char	Even Parity 8-bit Code
70	106	F			F	306
71	107	G			G	107
72	110	H			H	110
73	111	I			I	311
74	112	J			J	312
75	113	K			K	113
76	114	L			L	314
77	115	M			M	115
78	116	N			N	116
79	117	O			O	317
80	120	P			P	120
81	121	Q			Q	321
82	122	R			R	322
83	123	S			S	123
84	124	T			T	324
85	125	U			U	125
86	126	V			V	126
87	127	W			W	327
88	130	X			X	330
89	131	Y			Y	131
90	132	Z			Z	132
91	133	[		✓	K	333
92	134	\		✓	L	134
93	135	]		✓	M	335
94	136	^		✓	N	336
95	137	←		✓	O	137
96	140	`				140
97	141	a				341
98	142	b				342
99	143	c				143
100	144	d				344
101	145	e				145
102	146	f				146
103	147	g				347
104	150	h				350
105	151	i				151
106	152	j				152
107	153	k				353
108	154	l				154
109	155	m				355
110	156	n				356
111	157	o				157
112	160	p				360
113	161	q				161
114	162	r				162
115	163	s				363
116	164	t				164
117	165	u				365
118	166	v				366
119	167	w				167
120	170	x				170
121	171	y				371
122	172	z				372
123	173	{				173
124	174	|				374
125	175	}				175
126	176	~				176
127	177	DEL			rubout	377

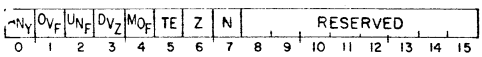
FLOATING POINT NUMBER FORMAT

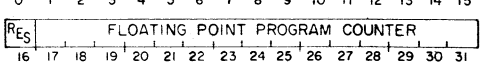
Single Precision



Double Precision

FLOATING POINT STATUS REGISTER





ECLIPSE LINE COMPUTERS Instruction Reference Card

POWER OF 2 TABLE

2^n	n
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 488	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

OCTAL – DECIMAL CONVERSION

To find the decimal number, locate the octal number and its decimal equivalent for each position. Add these to obtain the decimal number. To find the octal number, locate the next lower decimal number and its octal equivalent. Each difference is used to obtain the next octal number until the entire number is developed.

	8^5	8^4	8^3	8^2	8^1	8^0
0	0	0	0	0	0	0
1	32,768	4,096	512	64	8	1
2	65,536	8,192	1,024	128	16	2
3	98,304	12,288	1,536	192	24	3
4	131,072	16,384	2,048	256	32	4
5	163,840	20,480	2,560	320	40	5
6	196,608	24,576	3,072	384	48	6
7	229,376	28,672	3,584	448	56	7

DataGeneral
Engineering Publications Department
Southboro, Massachusetts 01772

©Data General Corporation 1974
All rights reserved
ECLIPSE is a trademark of Data General Corporation

015–000038–00

I/O DEVICE CODES

5

OCTAL DEVICE CODE	MNEMONIC	PRIORITY MASK BIT	DEVICE NAME
00	----	--	Unused
01	WCS	--	Writeable control store
02	ERCC	--	Error checking and correction
03	MAP	--	Memory allocation and protection
04			
05			
06	MCAT	12	Multiprocessor adapter transmitter
07	MCAR	12	Multiprocessor adapter receiver
10	TTI	14	TTY input
11	TTO	15	TTY output
12	PTR	11	Paper tape reader
13	PTP	13	Paper tape punch
14	RTC	13	Real-time clock
15	PLT	12	Incremental plotter
16	CDR	10	Card reader
17	LPT	12	Line printer
20	DSK	9	Fixed head disc
21	ADCV	8	A/D converter
22	MTA	10	Magnetic tape
23	DACV	--	D/A converter
24	DCM	0	Data communications multiplexor
25			
26			
27			
30	QTY	14	Asynchronous hardware multiplexor
30	SLA	14	Synchronous line adapter
31[2]	IBM1	13	IBM 360/370 interface
32	IBM2		
33	DKP	7	Moving head disc
34	CAS	10	Cassette tape
34[2]	MX1	11	Multiline asynchronous controller
35	MX2		
36	IPB	6	Interprocessor bus--half duplex
37	IVT	6	IPB watchdog timer
40	DPI	8	IPB full duplex input
41	DPO	8	IPB full duplex output
40[3]	SCR	8	Synchronous communication receiver
41[4]	SCT	8	Synchronous communication transmitter
42	DIO	7	Digital I/O
43	DIOT	6	Digital I/O timer
44	MXM	12	Modem control for MX1/MX2
45			
46	MCAT1	12	Second multiprocessor transmitter
47	MCAR1	12	Second multiprocessor receiver
50	TTI1	14	Second TTY input
51	TTO1	15	Second TTY output
52	PTR1	11	Second paper tape reader
53	PTP1	13	Second paper tape punch
54	RTC1	13	Second real-time clock
55	PLT1	12	Second incremental plotter
56	CDR1	10	Second card reader
57	LPT1	12	Second line printer
60	DSK1	9	Second fixed head disc
61			
62	MTA1	10	Second magnetic tape
63			
64			
65			
66			
67			
70	QTY1	14	Second asynchronous hardware multiplexor
70	SLA1	14	Second synchronous line adapter
71[2]		13	Second IBM 360/370 interface
72			
73	DKP1	7	Second moving head disc
74	CAS1	10	Second cassette tape
74[2]		11	Second multiline asynchronous controller
75			
76			
77	CPU	--	Central processor and console functions

[2] Code returned by INTA and used by VCT
[3] Can be set up with any unused even device code equal to 40 or above
[4] Can be set up with any unused odd device code equal to 41 or above

ASCII CHARACTER CODES

6

Decimal	7-bit Octal	Character	To Produce On TTY Cntrl	To Produce On TTY Mod Shift	To Produce On TTY 33,35 Char	Even Parity 8-bit Code
0	000	NUL	✓	✓	P	000
1	001	SOH	✓		A	201
2	002	STX	✓		B	202
3	003	ETX	✓		C	003
4	004	EOT	✓		D	204
5	005	ENQ	✓		E	005
6	006	ACK	✓		F	006
7	007	BEL	✓		G	207
8	010	BS	✓		H	210
9	011	HT	✓		I	011
10	012	NL			line feed	012
			✓		J	012
			✓		line feed	212[1]
11	013	VT	✓		K	213
12	014	FF	✓		L	014
13	015	CR			return	215
			✓		M	215
			✓		return	015[1]
14	016	SO	✓		N	216
15	017	SI	✓		O	017
16	020	DLE	✓		P	220
17	021	DC1	✓		Q	021
18	022	DC2	✓		R	022
19	023	DC3	✓		S	223
20	024	DC4	✓		T	024
21	025	NAK	✓		U	225
22	026	SYN	✓		V	226
23	027	ETB	✓		W	027
24	030	CAN	✓		X	030
25	031	EM	✓		Y	231
26	032	SUB	✓		Z	232
27	033	ESC			esc	033
			✓	✓	K	033
28	034	FS	✓	✓	L	234
29	035	GS	✓	✓	M	035
30	036	RS	✓	✓	N	036
31	037	US	✓	✓	O	237
32	040	SP			space	240
33	041	!		✓	1	041
34	042	"		✓	2	042
35	043	#		✓	3	243
36	044	$		✓	4	044
37	045	%		✓	5	245
38	046	&		✓	6	246
39	047	'		✓	7	047
40	050	(		✓	8	050
41	051	)		✓	9	251
42	052	*		✓	:	252
43	053	+		✓	;	053
44	054	,			,	
45	055	-			-	055
46	056	.			.	056
47	057	/			/	257
48	060	0			0	060
49	061	1			1	261
50	062	2			2	262
51	063	3			3	063
52	064	4			4	264
53	065	5			5	065
54	066	6			6	066
55	067	7			7	267
56	070	8			8	270
57	071	9			9	071
58	072	:			:	072
59	073	;			;	273
60	074	<		✓	,	074
61	075	=		✓	-	275
62	076	>		✓	.	276
63	077	?		✓	/	077
64	100	@		✓	P	300
65	101	A			A	101
66	102	B			B	102
67	103	C			C	303
68	104	D			D	104
69	105	E			E	305

[1] On even parity TTY's, these codes are odd parity.

CODING CONVENTIONS

Symbols listed in upper case are to be coded exactly as shown.

The symbols <>, and ==== are not coded. They act only to indicate how an assembly language instruction may be written. Their general definition is given below:

<> Indicates optional operands. The operand enclosed in the brackets (e.g. <#>) may be coded or not, depending on whether or not the associated option is desired.

==== Indicates specific substitution is required. Substitute the indicated accumulator, address, name, or number.

ABBREVIATIONS

AC = Accumulator	FPAC = Floating Point Accumulator
ACS = Source Accumulator	FACS = Floating Point Source AC
ACD = Destination Accumulator	FACD = Floating Point Destination AC
N = Digit In the Range 0-3	n = Digit In the Range 1-4
	i = 16-bit Signed or Unsigned Integer

In instructions that utilize an effective address, the following coding conventions are used: The indirect bit is set by coding the symbol ↑ somewhere in the operand string. The index bits are set by coding a comma followed by one of the digits 0-3 as the last operand of the operand string. If no index is coded, the bits are set to 00. The displacement id coded as a signed number in the current assembler radix. If no sign is coded, the number is assumed to be positive.

The standard instructions that use the two accumulator-multiple operation format have several options that can be obtained by coding the optional operands. The characters to be coded are given below with their results.

CLASS ABBREVIATION	CODED CHARACTER	RESULT BITS	OPERATION
c	(omitted)	00	Do not initialize the carry bit.
	Z	01	Initialize the carry bit to 0.
	O	10	Initialize the carry bit to 1.
	C	11	Initialize the carry bit to the complement of its present value
sh	(omitted)	00	Leave the result of the operation unaffected.
	L	01	Combine the carry and the 16-bit result into a 17-bit number and rotate it one bit left.
	R	10	Combine the carry and the 16-bit result into a 17-bit number and rotate it one bit right.
	S	11	Exchange the two 8-bit halves of the 16-bit result without affecting the carry.
#	(omitted)	0	Load the result of the shift operation into ACD.
	#	1	Do not load the result of the shift operation into ACD.

The following operands initiate operations that test the result of the shift operation. If the condition is true, the next sequential instruction is skipped.

skip	(omitted)	000	Never skip.
	SKP	001	Always skip.
	SZC	010	Skip if carry = 0.
	SNC	011	Skip if carry ≠ 0.
	SZR	100	Skip if result = 0.
	SNR	101	Skip if result ≠ 0.
	SEZ	110	Skip if either carry or result = 0.
	SBN	111	Skip if both carry and result ≠ 0.

NOTE: Instructions that specify both "no load" and "no skip" will not operate in the expected manner. These bit combinations are used for other instructions in the instruction set.

On this card, the "base value" for an instruction is given along with its name and how to code it. This base value is the 6-digit octal number that represents how the instruction would be assembled if all its options were omitted and all its operands were 0. For example, the base value for ADD is 103000. This represents an ADD 0,0 instruction.

UNSIGNED INTEGER COMPARISONS

SUB#	acs,acd,SZR	Skip if contents of ACS = contents of ACD
SUB#	acs,acd,SNR	Skip if contents of ACS ≠ contents of ACD
ADCZ#	acs,acd,SNC	Skip if contents of ACS < contents of ACD
SUBZ#	acs,acd,SNC	Skip if contents of ACS ≤ contents of ACD
SUBZ#	acs,acd,SZC	Skip if contents of ACS > contents of ACD
ADCZ#	acs,acd,SZC	Skip if contents of ACS ≥ contents of ACD

STANDARD INSTRUCTION SET

INSTRUCTION NAME	BASE VALUE	MNEMONIC AND OPERANDS
Add	103000	ADD<c><sh><#> acs,acd<,skip>
Add Complement	102000	ADC<c><sh><#> acs,acd<,skip>
Add Immediate	100010	ADI n,ac
AND	103400	AND<c><sh><#> acs,acd<,skip>
AND Immediate	143770	ANDI i,ac
AND with Complemented Source	100610	ANC acs,acd
Block Add and Move	113710	BAM
Block Move	133710	BLM
Compare to Limits	102370	CLM acs,acd
Complement	100000	COM<c><sh><#> acs,acd<,skip>
Count Bits	102610	COB acs,acd
Decimal Add	100210	DAD acs,acd
Decimal Subtract	100310	DSB acs,acd
Decrement and Skip if Zero	014000	DSZ <↑>displacement<,index>
Dispatch	142710	DSPA ac,<↑>displacement<,index>
Double Hex Shift Left	101610	DHXL n,ac
Double Hex Shift Right	101710	DHXR n,ac
Double Logical Shift	101310	DLSH acs,acd
Enter WCS	100070	XOP1 acs,acd,entry no.
Exchange Accumulators	100710	XCH acs,acd
Exclusive OR	100510	XOR acs,acd
Exclusive OR Immediate	123770	XORI i,ac
Execute	123370	XCT ac
Extended Add Immediate	163770	ADDI i,ac
Ext. Decrement and Skip if Zero	116070	EDSZ <↑>displacement<,index>
Ext. Increment and Skip if Zero	112070	EISZ <↑>displacement<,index>
Extended Jump	102070	EJMP <↑>displacement<,index>
Extended Jump to Subroutine	106070	EJSR <↑>displacement<,index>
Extended Load Accumulator	122070	ELDA ac,<↑>displacement<,index>
Extended Operation	100030	XOP acs,acd,operation no.
Extended Store Accumulator	142070	ESTA ac,<↑>displacement<,index>
Halve	143370	HLV ac
Hex Shift Left	101410	HXL n,ac
Hex Shift Right	101510	HXR n,ac
Inclusive OR	100410	IOR acs,acd
Inclusive OR Immediate	103770	IORI i,ac
Increment	101400	INC<c><sh><#> acs,acd<,skip>
Increment and Skip if Zero	010000	ISZ <↑>displacement<,index>
Jump	000000	JMP <↑>displacement<,index>
Jump to Subroutine	004000	JSR <↑>displacement<,index>
Load Accumulator	020000	LDA ac,<↑>displacement<,index>
Load Byte	102710	LDB acs,acd
Load Effective Address-all modes	162070	ELEF ac,<↑>displacement<,index>
Load Effective Address-user mode	060000	LEF ac,<↑>displacement<,index>
Load Map	113410	LMP
Locate and Reset Lead Bit	102510	LRB acs,acd
Locate Lead Bit	102410	LOB acs,acd
Logical Shift	101210	LSH acs,acd
Modify Stack Pointer	103370	MSP ac
Move	101000	MOV<c><sh><#> acs,acd<,skip>
Negate	100400	NEG<c><sh><#> acs,acd<,skip>
Pop Block	107710	POPB
Pop Multiple Accumulators	103210	POP acs,acd
Pop PC and Jump	117710	POPJ
Push Jump	102270	PUSHJ
Push Multiple Accumulators	103110	PSH acs,acd
Push Return Address	103710	PSHR
Restore	167710	RSTR
Return	127710	RTN
Save	163710	SAVE i
Set Bit to One	102010	BTO acs,acd
Set Bit to Zero	102110	BTZ acs,acd
Sign Extend and Divide	137710	DIVX
Signed Divide	157710	DIVS
Signed Multiply	147710	MULS
Skip if ACS > ACD	101010	SGT acs,acd
Skip if ACS ≥ ACD	101110	SGE acs,acd
Skip on Non-zero Bit	102770	SNB acs,acd
Skip on Zero Bit	102210	SZB acs,acd
Skip on Zero Bit and Set to One	102310	SZBO acs,acd
Store Accumulator	040000	STA ac,<↑>displacement<,index>
Store Byte	103010	STB acs,acd
Subtract	102100	SUB<c><sh><#> acs,acd<,skip>
Subtract Immediate	100110	SBI n,ac
System Call	103510	SYC acs,acd
Unsigned Divide	153710	DIV
Unsigned Multiply	143710	MUL

FLOATING POINT INSTRUCTION SET

INSTRUCTION NAME	BASE VALUE	MNEMONIC AND OPERANDS
Absolute Value	143050	FAB fpac
Add Double (FPAC)	100150	FAD facs,facd
Add Double (Memory)	101150	FAMD facd,<@>displacement<,index>
Add Single (FPAC)	100050	FAS facs,facd
Add Single (Memory)	101050	FAMS facd,<@>displacement<,index>
Clear Errors	153350	FCLE
Compare Floating Point	103450	FCMP facs,facd
Divide Double (FPAC)	100750	FDD facs,facd
Divide Double (Memory)	101750	FDMD facd,<@>displacement<,index>
Divide Single (FPAC)	100650	FDS facs,facd
Divide Single (Memory)	101650	FDMS facd,<@>displacement<,index>
Fix to AC	102650	FFAS ac,fpac
Fix to Memory	102750	FFMD fpac,<@>displacement<,index>
Float from AC	102450	FLAS ac,fpac
Float from Memory	102550	FLMD fpac,<@>displacement<,index>
Halve	163150	FHLV fpac
Load Double	102150	FLDD fpac,<@>displacement<,index>
Load Exponent	123150	FEXP fpac
Load Single	102050	FLDS fpac,<@>displacement<,index>
Load Status	123350	FLST <@>displacement<,index>
Move Floating Point	103550	FMOV facs,facd
Multiply Double (FPAC)	100550	FMD facs,facd
Multiply Double (Memory)	101550	FMMD facd,<@>displacement<,index>
Multiply Single (FPAC)	100450	FMS facs,facd
Multiply Single (Memory)	101450	FMMS facd,<@>displacement<,index>
Negate	163050	FNEG fpac
No Skip	103250	FNS
Normalize	103050	FNOM fpac
Pop Floating Point State	167350	FPOP
Push Floating Point State	163350	FPSH
Read High Word	123050	FRH fpac
Scale	103150	FSCAL fpac
Skip Always	107250	FSA
Skip on EQ Zero	113250	FSEQ
Skip on GE Zero	127250	FSGE
Skip on GT Zero	137250	FSGT
Skip on LE Zero	133250	FSLE
Skip on LT Zero	123250	FSLT
Skip on NE Zero	117250	FSNE
Skip on No DVZ	147250	FSND
Skip on No Error	177250	FSNER
Skip on No MOF	143250	FSNM
Skip on No OVF	163250	FSNO
Skip on No OVF and No DVZ	167250	FSNOD
Skip on No UNF	153250	FSNU
Skip on No UNF and No DVZ	157250	FSNUD
Skip on No UNF and No OVF	173250	FSNUO
Store Double	102350	FSTD fpac,<@>displacement<,index>
Store Single	102250	FSTS fpac,<@>displacement<,index>
Store Status	103350	FSST <@>displacement<,index>
Subtract Double (FPAC)	100350	FSD facs,facd
Subtract Double (Memory)	101350	FSMD facd,<@>displacement<,index>
Subtract Single (FPAC)	100250	FSS facs,facd
Subtract Single (Memory)	101250	FSMS facd,<@>displacement<,index>
Trap Disable	147350	FTD
Trap Enable	143350	FTE

I/O INSTRUCTION SET

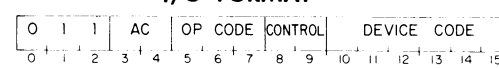
The instructions that use the I/O format can set bits 8-9 by coding optional operands, The characters to be coded are given below with their results.

CLASS ABBREVIATION	CODED CHARACTER	RESULT BITS	OPERATION
f	(omitted)	00	Do not affect the BUSY and DONE flags
	S	01	Set the BUSY flag to 1 and set the DONE flag to 0--starts the device.
	C	10	Set the BUSY flag to 0 and set the DONE flag to 0--idles the device.
	P	11	Pulse the I/O bus control line--the effect, if any, depends on the device.
t	BN	00	If the BUSY flag ≠ 0, the next sequential instruction is skipped.
	BZ	01	If the BUSY flag = 0, the next sequential instruction is skipped.
	DN	10	If the DONE flag ≠ 0, the next sequential instruction is skipped.
	DZ	11	If the DONE flag = 0, the next sequential instruction is skipped.

INSTRUCTION NAME	OP CODE	MNEMONIC AND OPERANDS
Data In A	1	DIA<f> ac,device
Data In B	3	DIB<f> ac,device
Data In C	5	DIC<f> ac,device
Data Out A	2	DOA<f> ac,device
Data Out B	4	DOB<f> ac,device
Data Out C	6	DOC<f> ac,device
I/O Skip	7	SKP<t> device[5]
No I/O Transfer	0	NIO<f> device[5]

[5] For this instruction, the AC field (bits 3-4), is set to zero.

I/O FORMAT



RESERVED STORAGE LOCATIONS

LOCATION (octal)	NAME	CONTENTS
0	I/O RETURN	Return address from I/O interrupt.
1	I/O HANDLER	Address of I/O interrupt handler.
2	SC HANDLER	Address of SCL instruction handler.
3	PF HANDLER	Address of protection fault handler.
4	VCT STACK PTR	Address of beginning of VCT stack.
5	CURRENT MASK	Current interrupt priority mask
6	VCT STACK LMT	Point in VCT stack where overflow occurs.
7	VCT STACK FLT	Address of VCT stack fault handler.
20-27	AUTO-INC	Auto-incrementing locations.
30-37	AUTO-DEC	Auto-decrementing locations.
40	STACK POINTER	Address of the last word placed on the stack.
41	FRAME POINTER	Address of start of current frame minus one.
42	STACK LIMIT	Point in stack where stack overflow occurs.
43	STACK FAULT	Address of stack fault handler.
44	XOP ORIGIN	Address of the beginning of the XOP table.
45	FP FAULT	Address of floating point fault handler.
46		Reserved for future use.
47		Reserved for future use.

CPU INSTRUCTIONS

INSTRUCTION NAME	MNEMONIC	I/O INSTRUCTION	BASE VALUE
Halt	HALTA ac	DOC ac,CPU	063077
Interrupt Acknowledge	INTA ac	DIB ac,CPU	061477
Interrupt Disable	INTDS	NIOC CPU	060277
Interrupt Enable	INTEN	NIOS CPU	060177
I/O Reset	IORST	DICC 0,CPU	062677
Mask Out	MSKO ac	DOB ac,CPU	062077
Read Switches	READS ac	DIA ac,CPU	060477
Vector	VCT <@>displacement	DIBP 0,CPU	061777