CRAY RESEARCH, INC.

CRAY X-MP CAL REFERENCE CARD

SQ-0067

CAL CONTROL STATEMENT

```
CAL,CPU=type,I=idn,L=ldn,B=bdn,E=edn,ABORT,DEBUG,options,
    LIST=name,S=sdn,SYM=sym,T=bst,X=xdn.
```

CPU	Omitted	Machine currently executing CAL
	CPU=*type*	Specify CRAY-1 or CRAY-XMP
I	Omitted	Source on $IN
	I=*idn*	Source on *idn*
L	Omitted	List output on $OUT
	L=0	No list output
	L=*ldn*	List output on *ldn*
B	Omitted	Binary on $BLD
	B=0	No binary
	B=*bdn*	Binary on *bdn*
E	Omitted	No error listing
	E	Error list on $OUT
	E=*edn*	Error list on *edn* unless *edn*=*ldn*, then *ldn*
ABORT	Omitted	Do not abort
	ABORT	Abort on fatal error during assembly
DEBUG	Omitted	Write binary record on fatal error and set fatal error flag
	DEBUG	Write binary record with fatal error flag clear
options:	See *options* under CAL control statement in CAL Reference Manual (*options* overrides the LIST pseudo.)	
LIST	Omitted	LIST pseudos with a null (empty) location field processed
	LIST	All LIST pseudos processed
	LIST=*name*	LIST pseudo instructions with a location field matching name processed
S	Omitted	$SYSTXT
	S=0	No system text
	S=*sdn*	System text on *sdn*
SYM	Omitted	No symbol table
	SYM	Symbol table on dataset holding binary load data
	SYM=*sym*	Symbol table on *sym*
T	Omitted	No binary system text written
	T=0	No binary system text written
	T	Binary dataset written to $BST
	T=*bst*	Binary system text written to *bst*
X	Omitted	No global cross-reference records written
	X=0	No global cross-reference records written
	X	Global cross-reference records written to $XRF
	X=*xdn*	Global cross-reference records written to *xdn*

INSTRUCTIONS

CRAY X-MP	CAL	UNIT	DESCRIPTION
000000	ERR	-	Error exit
††0010*ijk*	CA,A*j* *Ak*	-	Set the channel (A*j*) current address to(A*k*) and begin the I/O sequence
††0011*ijk*	CL,A*j* *Ak*	-	Set the channel (A*j*) limit address to (A*k*)
††0012*ij*0	CI,A*j*	-	Clear channel (A*j*) interrupt flag; clear device master-clear (output channel).
††0012*ij*1	MC,A*j*	-	Clear channel (A*j*) interrupt flag; set device master-clear (output channel); clear device ready-held (input channel).
††0013*ij*0	XA A*j*	-	Enter XA register with (A*j*)
††0014*ij*0	RT S*j*	-	Enter RTC register with (S*j*)
††001401	IP 1	-	Set interprocessor interrupt
††001402	IP 0	-	Clear interprocessor interrupt
††001403	CLN 0	-	Enter CLN register with 0
††001413	CLN 1	-	Enter CLN register with 1
††001423	CLN 2	-	Enter CLN register with 2
††001433	CLN 3	-	Enter CLN register with 3
††0014*ij*4	PCI S*j*	-	Enter II register with (S*j*)
††001405	CCI	-	Clear PCI request
††001406	ECI	-	Enable PCI request
††001407	DCI	-	Disable PCI request
00200*k*	VL A*k*	-	Transmit (A*k*) to VL register
†002000	VL 1	-	Transmit 1 to VL register
002100	EFI	-	Enable interrupt on floating-point error
002200	DFI	-	Disable interrupt on floating-point error
002300	ERI	-	Enable operand range interrupts
002400	DRI	-	Disable operand range interrupts
002500	DBM	-	Disable bidirectional memory transfers
002600	EBM	-	Enable bidirectional memory transfers
002700	CMR	-	Complete memory references
0030*ij*0	VM S*j*	-	Transmit (S*j*) to VM register
†003000	VM 0	-	Clear VM register
0034*jk*	SM*jk* 1,TS	-	Test & set semaphore *jk* in SM
0036*jk*	SM*jk* 0	-	Clear semaphore *jk* in SM
0037*jk*	SM*jk* 1	-	Set semaphore *jk* in SM
004000	EX	-	Normal exit
0050*jk*	J B*jk*	-	Jump to (B*jk*)
006*ijkm*	J *exp*	-	Jump to *exp*
007*ijkm*	R *exp*	-	Return jump to *exp*; set B00 to P.
010*ijkm*	JAZ *exp*	-	Branch to *exp* if (A0)=0
011*ijkm*	JAN *exp*	-	Branch to *exp* if (A0)≠0
012*ijkm*	JAP *exp*	-	Branch to *exp* if (A0) positive; 0 is positive.
013*ijkm*	JAM *exp*	-	Branch to *exp* if (A0) negative
014*ijkm*	JSZ *exp*	-	Branch to *exp* if (S0)=0
015*ijkm*	JSN *exp*	-	Branch to *exp* if (S0)≠0
016*ijkm*	JSP *exp*	-	Branch to *exp* if (S0) positive; 0 is positive.
017*ijkm*	JSM *exp*	-	Branch to *exp* if (S0) negative
†††020*ijkm*	A*i* *exp*	-	Transmit *exp*=*jkm* to A*i*
†††021*ijkm*	A*i* *exp*	-	Transmit *exp*=ones complement of *jkm* to A*i*
†††022*ijk*	A*i* *exp*	-	Transmit *exp*=*jk* to A*i*
023*ij*0	A*i* S*j*	-	Transmit (S*j*) to A*i*
023*i*01	A*i* VL	-	Transmit (VL) to A*i*
024*ijk*	A*i* B*jk*	-	Transmit (B*jk*) to A*i*
025*ijk*	B*jk* A*i*	-	Transmit (A*i*) to B*jk*
026*ij*0	A*i* PS*j*	Pop/LZ	Population count of (S*j*) to A*i*
026*ij*1	A*i* QS*j*	Pop/LZ	Population count parity of (S*j*) to A*i*
026*i*,*j*7	A*i* SB*j*	-	Transmit (SB*j*) to A*i*

CRAY X-MP	CAL	UNIT	DESCRIPTION
†13hi00 0	,Ah Si	Memory	Store (Si) to (Ah)
140*ijk*	Vi Sj&Vk	V Logical	Logical products of (S*j*) and (V*k*) to V*i*
141*ijk*	Vi Vj&Vk	V Logical	Logical products of (V*j*) and (V*k*) to V*i*
142*ijk*	Vi Sj!Vk	V Logical	Logical sums of (S*j*) and (V*k*) to V*i*
†142i0k	Vi Vk	V Logical	Transmit (V*k*) to V*i*
143*ijk*	Vi Vj!Vk	V Logical	Logical sums of (V*j*) and (V*k*) to V*i*
144*ijk*	Vi Sj\Vk	V Logical	Logical differences of (S*j*) and (V*k*) to V*i*
145*ijk*	Vi Vj\Vk	V Logical	Logical differences of (V*j*) and (V*k*) to V*i*
†145*iii*	Vi 0	V Logical	Clear V*i*
146*ijk*	Vi Sj!Vk&VM	V Logical	Transmit (S*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*.
†146i0k	Vi #VM&Vk	V Logical	Vector merge of (V*k*) and 0 to V*i*
147*ijk*	Vi Vj!Vk&VM	V Logical	Transmit (V*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*.
150*ijk*	Vi Vj<Ak	V Shift	Shift (V*j*) left (A*k*) places to V*i*
†150*ij*0	Vi Vj<1	V Shift	Shift (V*j*) left one place to V*i*
151*ijk*	Vi Vj>Ak	V Shift	Shift (V*j*) right (A*k*) places to V*i*
†151*ij*0	Vi Vj>1	V Shift	Shift (V*j*) right one place to V*i*
152*ijk*	Vi Vj,Vj<Ak	V Shift	Double shift (V*j*) left (A*k*) places to V*i*
†152*ij*0	Vi Vj,Vj<1	V Shift	Double shift (V*j*) left one place to V*i*
153*ijk*	Vi Vj,Vj>Ak	V Shift	Double shift (V*j*) right (A*k*) places to V*i*
†153*ij*0	Vi Vj,Vj>1	V Shift	Double shift (V*j*) right one place to V*i*
154*ijk*	Vi Sj+Vk	V Int Add	Integer sums of (S*j*) and (V*k*) to V*i*
155*ijk*	Vi Vj+Vk	V Int Add	Integer sums of (V*j*) and (V*k*) to V*i*
156*ijk*	Vi Sj-Vk	V Int Add	Integer differences of (S*j*) and (V*k*) to V*i*
†156i0k	Vi -Vk	V Int Add	Transmit negative of (V*k*) to V*i*
157*ijk*	Vi Vj-Vk	V Int Add	Integer differences of (V*j*) and (V*k*) to V*i*
160*ijk*	Vi Sj*FVk	Fp Mult	Floating-point products of (S*j*) and (V*k*) to V*i*
161*ijk*	Vi Vj*FVk	Fp Mult	Floating-point products of (V*j*) and (V*k*) to V*i*
162*ijk*	Vi Sj*HVk	Fp Mult	Half-precision rounded floating-point products of (S*j*) and (V*k*) to V*i*
163*ijk*	Vi Vj*HVk	Fp Mult	Half-precision rounded floating-point products of (V*j*) and (V*k*) to V*i*
164*ijk*	Vi Sj*RVk	Fp Mult	Rounded floating-point products of (S*j*) and (V*k*) to V*i*
165*ijk*	Vi Vj*RVk	Fp Mult	Rounded floating-point products of (V*j*) and (V*k*) to V*i*
166*ijk*	Vi Sj*IVk	Fp Mult	2-floating-point products of (S*j*) and (V*k*) to V*i*
167*ijk*	Vi Vj*IVk	Fp Mult	2-floating-point products of (V*j*) and (V*k*) to V*i*
170*ijk*	Vi Sj+FVk	Fp Add	Floating-point sums of (S*j*) and (V*k*) to V*i*
†170i0k	Vi +FVk	Fp Add	Normalize (V*k*) to V*i*
171*ijk*	Vi Vj+FVk	Fp Add	Floating-point sums of (V*j*) and (V*k*) to V*i*
172*ijk*	Vi Sj-FVk	Fp Add	Floating-point differences of (S*j*) and (V*k*) to V*i*
†172i0k	Vi -FVk	Fp Add	Transmit normalized negatives of (V*k*) to V*i*
173*ijk*	Vi Vj-FVk	Fp Add	Floating-point differences of (V*j*) and (V*k*) to V*i*
174*ij*0	Vi /HVj	Fp Rcpl	Floating-point reciprocal approximations of (V*j*) to V*i*
174*ij*1	Vi PVj	V Pop	Population counts of (V*j*) to V*i*
174*ij*2	Vi QVj	V Pop	Population count parities of (V*j*) to V*i*
1750*j*0	VM Vj,Z	V Logical	VM=1 where (V*j*)=0

PSEUDO INSTRUCTIONS

PROGRAM CONTROL

```
IDENT    name
END
ABS
COMMENT  'string'
```

MICROS

```
name  MICRO   'string',exp1,exp2
name  MICRO   'string',exp1
name  MICRO   'string'
name  OCTMIC  exp,count
name  DECMIC  exp,count
```

DATA DEFINITION

```
symbol  CON   exp1,exp2,...,expn
symbol  BSSZ  exp
symbol  DATA  data1,data2,...,datan
symbol  VWD   n1/exp1,n2/exp2,...,nm/expm
        REP   ct,exs,inc,bex
```

LISTING CONTROL

```
name  LIST      op1,op2,...,opn
      LIST      *
      SPACE     count
      EJECT
      TITLE     'string'
      SUBTITLE  'string'
name  TEXT      'string'
      ENDTEXT
```

options	
ON	OFF
XRF	NXRF
XNS	NXNS
DUP	NDUP
MAC	NMAC
MIF	NMIF
MIC	NMIC
LIS	NLIS
WEM	NWEM
WTXT	NTXT
WRP	NWRP
WMR	NWMR

ERROR CONTROL

```
code  ERROR
code  ERRIF  exp1,op,exp2

op:   LT, LE, GT, GE, EQ, or NE
code: See Fatal or Warning Errors
```

LOADER LINKAGE

```
ENTRY   symbol1,symbol2,...,symboln
EXT     sym1,sym2,...,symn
MODULE  modtype
START   symbol
```

CODE DUPLICATION

```
dupname  DUP     times
         DUP     times,count
dupname  ECHO    e1=(list1),e2=(list2),...,en=(listn)
dupname  ENDDUP
dupname  STOPDUP
```

SYMBOL DEFINITION

```
symbol  =       exp,attribute
symbol  SET     exp,attribute
symbol  MICSIZE name

attribute:  P, W, V, or blank
```

CONDITIONAL ASSEMBLY

```
ifname  IFA     attribute,exp
        IFA     attribute,exp,count
ifname  IFE     exp1,op,exp2
        IFE     exp1,op,exp2,count
ifname  IFC     'string1',op,'string2'
        IFC     'string1',op,'string2',count
ifname  SKIP    count
ifname  ENDIF
ifname  ELSE
```

attribute	
PA	parcel
WA	word
VAL	value
EXT	external
REL	relocatable
ABS	absolute
COM	common
DEF	defined
SET	SET-defined
REG	register
MIC	micro

(# can precede attribute)

BLOCK CONTROL

```
        BLOCK   name
        COMMON  name
        ORG     exp
symbol  BSS     exp
        LOC     exp
        BITW    exp
        BITP    exp
symbol  ALIGN
```

MODE CONTROL

```
BASE  O, D, M, or *
QUAL  qualification
QUAL  *
QUAL
```

MACRO DEFINITION

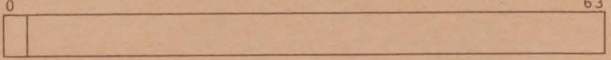
```
      MACRO
lfp   name   p1,p2,...,pn,e1=d1,e2=d2,...,em=dm
      LOCAL  sym1,...,symn
        .
        .    (body of definition)
name  ENDM

name1 OPSYN name2
```

OPDEF DEFINITION

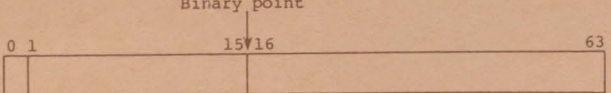
```
name  OPDEF
lfp   synres  synop
      LOCAL   sym1,...,symn
        .
        .     (body of definition)
name  ENDM
```

DATA FORMATS

Twos Complement Integer (24 bits)



Twos Complement Integer (64 bits)



Signed Magnitude Floating-point (64 bits)