OCTAL FORMAT o f a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
00 0 - -	—	Diagnostic return	If diagnostic jump set $R_{17} \rightarrow \mu P$	– NA –		
00 3 a m	BL a,y,m	Byte load	$(Y)_{byte} \rightarrow R_a$	0	0	X
01 0 a m	LR a,m	Load (Register)	$(Rm), \rightarrow, R_a$	0	0	X
01 1 a m	LI a,m	Load (Indirect)	$(Y^*) \rightarrow R_a$	0	0	X
01 2 a m	LK a,y,m	Load (Constant)	$Y \rightarrow R_a$	0	0	X
01 3 a m	L a,y,m	Load	$(Y) \rightarrow R_a$	0	0	X
02 0 a 00	PR a	Make positive	If $(R_a) < 0$, $(R_a)' \rightarrow R_a$	X	X	X
02 0 a 01	NR a	Make negative	If $(R_a) > 0$, $(R_a)' \rightarrow R_a$	X	0	X
02 0 a 02	RR a	Round	If $(R_a) \geq 0$, $(R_a) + (R_{a+1})_{15} \rightarrow R_a$ ③ If $(R_a) < 0$, $(R_a) - (R_{a+1})_{15} \rightarrow R_a$	X	X	X
02 0 a 04	TCR a	Two's Complement	$(R_a)' \rightarrow R_a$	X	X	X
02 0 a 05	TCDR a	Two's Complement Double	$(R_a, R_{a+1})' \rightarrow R_a, R_{a+1}$	X	X	X
02 0 a 06	OCR a	One's Complement	(R_a) bit-by-bit complement $\rightarrow R_a$	0	0	X
02 0 a 10	IROR a	Increase R_a by 1	$(R_a) + 1 \rightarrow R_a$	X	X	X
02 0 a 11	DROR a	Decrease R_a by 1	$(R_a) - 1 \rightarrow R_a$	X	X	X
02 0 a 12	IRTR a	Increase R_a by 2	$(R_a) + 2 \rightarrow R_a$	X	X	X
02 0 a 13	DRTR a	Decrease R_a by 2	$(R_a) - 2 \rightarrow R_a$	X	X	X
02 1 a m	LDI a,m	Load Double (Indirect)	$(Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$	0	0	X
02 3 a m	LD a,y,m	Load Double	$(Y, Y+1) \rightarrow R_a, R_{a+1}$	0	0	X
03 0 a 00	ER a	Executive Return	Generate interrupt; $(P)+1 \rightarrow R_a$	0	0	X
03 0 a 01	SSOR a	Store SR1	$(SR1) \rightarrow R_a$	0	0	X
03 0 a 02	SSTR a	Store SR2	$(SR2) \rightarrow R_a$	0	0	X
03 0 a 03	SCR a	Store Clock	$(RTC\ register)_{15-0} \rightarrow R_a$	0	0	X
03 0 a 04	LPR a	Load P	$(R_a) \rightarrow P$	– NA –		
03 0 a 05	LSOR a	Load SR1	$(R_a) \rightarrow SR1$	– NA –		
03 0 a 06	LSTR a	Load SR2	$(R_a) \rightarrow SR2$	– NA –		
03 0 a 07	LCR a	Load RTC lower	$(R_a) \rightarrow RTC\ register_{15-0}$;	– NA –		
03 0 00 10	ECR	Enable Clock	Enable RTC register	– NA –		
03 0 00 11	DCR	Disable Clock	Disable RTC register	– NA –		
03 0 a 12	LEM a	Load and Enable Mon. clock	$(R_a) \rightarrow$ Mon. Clock reg.; enable countdown	– NA –		
03 0 00 13	DM	Disable Monitor clock	Disable Mon. clock register	– NA –		
03 0 a 14	LCRD a	Load and enable Clock Double	$(R_a, R_{a+1}) \rightarrow$ RTC; enable count up	– NA –		
03 0 a 15	SCRD a	Store Clock Double	(RTC Register) $\rightarrow R_a, R_{a+1}$	0	0	X
03 0 00 16	ECIR	Enable Clock Interrupt	Enable RTC overflow interrupt	– NA –		
03 0 00 17	DCIR	Disable Clock Interrupt	Disable RTC overflow interrupt	– NA –		
03 3 a m	LM a,y,m	Load multiple	$(Y \ldots Y+m-a) \rightarrow R_a \ldots R_m$	– NA –		
# 04 0 a 00	SQR a	Square Root	$\surd (R_a, R_{a+1}) \rightarrow R_{a+1}$; Rem. $\rightarrow R_a$	0	X	X
04 0 a 01	RVR a	Reverse Register	Reverse (R_a)	0	0	X
04 0 a 02	CNT a	Count Ones	Number of binary ones in $R_a \rightarrow R_{a+1}$	– NA –		
04 0 a 03	SFR a	Scale Factor	Shift (R_a, R_{a+1}) left until $(R_a)_{15} \neq (R_a)_{14}$; shift count $\rightarrow R_{a+2}$ ①	– NA –		
04 3 a m	BLX a,y,m	Byte Load and index by 1	$(Y)_{byte} \rightarrow R_a$; $(Rm)+1 \rightarrow Rm$ ②	0	0	X
05 0 a m	SBR a,m	Set Bit	$1 \rightarrow (R_a)_m$	0	0	X
05 1 a m	LXI a,m	Load and index by 1 (Indirect)	$(Y^*) \rightarrow R_a$; $(R_m)+1 \rightarrow R_m$ ②	0	0	X
05 3 a m	LX a,y,m	Load and index by 1	$(Y) \rightarrow R_a$; $(R_m)+1 \rightarrow R_m$ ②	0	0	X
06 0 a m	ZBR a,m	Zero Bit	$0 \rightarrow (R_a)_m$	0	0	X
06 1 a m	LDXI a,m	Load Double Index by 2 (Indirect)	$(Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$; ② $(R_m)+2 \rightarrow R_m$	0	0	X
06 3 a m	LDX a,y,m	Load Double, index by 2	$(Y, Y+1) \rightarrow R_a, R_{a+1}$; $(R_m)+2 \rightarrow R_m$ ②			X
07 0 a m	CBR a,m	Compare Bit	Test bit m of R_a for zero	0	0	X
07 1 00 m	LPI m	Load PSW (Indirect)	$(Y^*, Y^*+1, Y^*+2) \rightarrow P, SR1, SR2$	– NA –		
07 3 00 m	LP y,m	Load PSW	$(Y, Y+1, Y+2) \rightarrow P, SR1, SR2$	– NA –		
10 0 a m	LRSR a,m	Logical Right Shift (Register)	Shift (R_a) right $(R_m)_{5-0}$ places, zero fill	0	0	X
10 2 a m	LRS a,y,m	Logical Right Shift	Shift (R_a) right Y_{5-0} places, zero fill	0	0	X
10 3 a m	BS a,y,m	Byte Store	$(R_a)_{7-0} \rightarrow Y_{byte}$	–NA–		
11 0 a m	ARSR a,m	Algebraic Right Shift (Register)	Shift (R_a) right $(R_m)_{5-0}$ places, sign fill	0	0	X
11 1 a m	SI a,m	Store (Indirect)	$(R_a) \rightarrow Y^*$	– NA –		
11 2 a m	ARS a,y,m	Algebraic Right Shift	Shift (R_a) right Y_{5-0} places, sign fill	0	0	X
11 3 a m	S a,y,m	Store	$(R_a) \rightarrow Y$	–NA–		
12 0 a m	LRDR a,m	Logical Right Double shift	Shift (R_a, R_{a+1}) right $(R_m)_{5-0}$ places, zero fill	0	0	X
12 1 a m	SDI a,m	Store Double (Indirect)	$(R_a, R_{a+1}) \rightarrow Y^*, Y^*+1$	– NA –		
12 2 a m	LRD a,y,m	Logical Right Double shift (Register)	Shift (R_a, R_{a+1}) right Y_{5-0} places, zero fill	0	0	X
12 3 a m	SD a,y,m	Store Double	$(R_a, R_{a+1}) \rightarrow Y, Y+1$	– NA –		

Optional Math Pac Instructions ①Count = 31 for all zeros or all ones. ② if $a \neq m$ ③ a must be even

OCTAL FORMAT o f a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
13 0 a m	ARDR a,m	Algebraic Right Double shift	Shift (R_a, R_{a+1}) right (R_m)$_{5-0}$ places, sign fill	0	0	X
13 2 a m	ARD a,y,m	Algebraic Right Double shift	Shift (R_a, R_{a+1}) right Y_{5-0} places, sign fill	0	0	X
13 3 a m	SM a,y,m	Store Multiple	(R_a ... R_m) → Y ... Y+m−a	–	NA	–
14 0 a m	ALSR a,m	Algebraic Left shift (Register)	Shift (R_a) left (R_m)$_{5-0}$ places, zero fill	0	X	X
14 2 a m	ALS a,y,m	Algebraic Left shift	Shift (R_a) left Y_{5-0} places, zero fill	0	X	X
14 3 a m	BSX a,y,m	Byte Store, index by 1	(R_a)$_{7-0}$ → Y_{byte}; (R_m)+1 → R_m	–	NA	–
15 0 a m	CLSR a,m	Circular Left shift (Register)	Shift (R_a) circularly left (R_m)$_{5-0}$ places	0	0	X
15 1 a m	SXI a,m	Store index by 1 (Indirect)	(R_a) → Y*; (R_m)+1 → R_m	–	NA	–
15 2 a m	CLS a,y,m	Circular Left shift	Shift (R_a) circularly left Y_{5-0} places	0	0	X
15 3 a m	SX a,y,m	Store, index by 1	(R_a) → Y; (R_m)+1 → R_m	–	NA	–
16 0 a m	ALDR a,m	Algebraic Left Double shift (Register)	Shift (R_a, R_{a+1}) left (R_m)$_{5-0}$ places, zero fill	0	X	X
16 1 a m	SDXI a,m	Store Double index by 2 (Indirect)	(R_a, R_{a+1}) → Y*, Y*+1; (R_m)+2 → R_m ②	–	NA	–
16 2 a m	ALD a,y,m	Algebraic Left Double shift	Shift (R_a, R_{a+1}) left Y_{5-0} places zero fill	0	X	X
16 3 a m	SDX a,y,m	Store Double, index by 2	(R_a, R_{a+1}) → (Y, Y+1); (R_m)+2 → R_m ②	–	NA	–
17 0 a m	CLDR a,m	Circular Left Double shift (Register)	Shift (R_a, R_{a+1}) circularly left (R_m)$_{5-0}$ places	0	0	X
17 1 00 m	SZI m	Store Zeros (Indirect)	0 → Y*	–	NA	–
17 2 a m	CLD a,y,m	Circular Left Double shift	Shift (R_a, R_{a+1}) circularly left Y_{5-0} places	0	0	X
17 3 00 m	SZ y,m	Store Zeros	0 → Y	–	NA	–
20 0 a m	SUR a,m	Subtract (Register)	(R_a) − (R_m) → R_a	X	X	X
20 1 a m	SUI a,m	Subtract (Indirect)	(R_a) − (Y*) → R_a	X	X	X
20 2 a m	SUK a,y,m	Subtract (Constant)	(R_a) − Y → R_a	X	X	X
20 3 a m	SU a,y,m	Subtract	(R_a) − (Y) → R_a	X	X	X
21 0 a m	SUDR a,m	Subtract Double (Register)	(R_a, R_{a+1}) − (R_m, R_{m+1}) → R_a, R_{a+1}	X	X	X
21 1 a m	SUDI a,m	Subtract Double (Indirect)	(R_a, R_{a+1}) − (Y*, Y*+1) → R_a, R_{a+1}	X	X	X
21 3 a m	SUD a,y,m	Subtract Double	(R_a, R_{a+1}) − (Y, Y+1) → R_a, R_{a+1}	X	X	X
22 0 a m	AR a,m	Add (Register)	(R_a) + (R_m) → R_a	X	X	X
22 1 a m	AI a,m	Add (Indirect)	(R_a) + (Y*) → R_a	X	X	X
22 2 a m	AK a,y,m	Add (Constant)	(R_a) + Y → R_a	X	X	X
22 3 a m	A a,y,m	Add	(R_a) + (Y) → R_a	X	X	X
23 0 a m	ADR a,m	Add Double (Register)	(R_a, R_{a+1}) + (R_m, R_{m+1}) → R_a, R_{a+1}	X	X	X
23 1 a m	ADI a,m	Add Double (Indirect)	(R_a, R_{a+1}) + (Y*, Y*+1) → R_a, R_{a+1}	X	X	X
23 3 a m	AD a,y,m	Add Double	(R_a, R_{a+1}) + (Y, Y+1) → R_a, R_{a+1}	X	X	X
24 0 a m	CR a,m	Compare (Register)	(R_a): (R_m)	X	X	X
24 1 a m	CI a,m	Compare (Indirect)	(R_a): (Y*)	X	X	X
24 2 a m	CK a,y,m	Compare (Constant)	(R_a): Y	X	X	X
24 3 a m	C a,y,m	Compare	(R_a): (Y)	X	X	X
25 0 a m	CDR a,m	Compare Double (Register)	(R_a, R_{a+1}): (R_m, R_{m+1})	X	X	X
25 1 a m	CDI a,m	Compare Double (Indirect)	(R_a, R_{a+1}): (Y*, Y*+1)	X	X	X
25 3 a m	CD a,y,m	Compare Double	(R_a, R_{a+1}): (Y, Y+1)	X	X	X
26 0 a m	MR a,m	Multiply (Register)	(R_{a+1}) · (R_m) → R_a, R_{a+1}	0	0	X
26 1 a m	MI a,m	Multiply (Indirect)	(R_{a+1}) · (Y*) → R_a, R_{a+1}	0	0	X
26 2 a m	MK a,y,m	Multiply (Constant)	(R_{a+1}) · Y → R_a, R_{a+1}	0	0	X
26 3 a m	M a,y,m	Multiply	(R_{a+1}) · (Y) → R_a, R_{a+1}	0	0	X
27 0 a m	DR a,m	Divide (Register)	(R_a, R_{a+1}) / (R_m) → R_{a+1}; remainder → R_a	0	X	X
27 1 a m	DI a,m	Divide (Indirect)	(R_a, R_{a+1}) / (Y*) → R_{a+1}; remainder → R_a	0	X	X
27 2 a m	DK a,y,m	Divide (Constant)	(R_a, R_{a+1}) / Y → R_{a+1}; remainder → R_a	0	X	X
27 3 a m	D a,y,m	Divide	(R_a, R_{a+1}) / (Y) → R_{a+1}; remainder → R_a	0	X	X
30 0 a m	ANDR a,m	AND (Register)	(R_a) ⊙ (R_m) → R_a	0	0	X
30 0 0 0	NOP	No Operation (software)	(R_0) ⊙ (R_0) → R_0	0	0	X
30 0 0 0	NOPD	No Operation Double (software)	(R_0) ⊙ (R_0) → R_0; (R_0) ⊙ (R_0) → R_0 Two instructions generated	0	0	X
30 1 a m	ANDI a,m	AND (Indirect)	(R_a) ⊙ (Y*) → R_a	0	0	X
30 2 a m	ANDK a,y,m	AND (Constant)	(R_a) ⊙ Y → R_a	0	0	X
30 3 a m	AND a,y,m	AND	(R_a) ⊙ (Y) → R_a	0	0	X
31 0 a m	ORR a,m	OR (Register)	(R_a) ⊕ (R_m) → R_a	0	0	X
31 1 a m	ORI a,m	OR (Indirect)	(R_a) ⊕ (Y*) → R_a	0	0	X
31 2 a m	ORK a,y,m	OR (Constant)	(R_a) ⊕ Y → R_a	0	0	X
31 3 a m	OR a,y,m	OR	(R_a) ⊕ (Y) → R_a	0	0	X
32 0 a m	XORR a,m	Exclusive OR (Register)	(R_a) ⊕̄ (R_m) → R_a	0	0	X
32 1 a m	XORI a,m	Exclusive OR (Indirect)	(R_a) ⊕̄ (Y*) → R_a	0	0	X
32 2 a m	XORK a,y,m	Exclusive OR (Constant)	(R_a) ⊕̄ Y → R_a	0	0	X
32 3 1 m	XOR a,y,m	Exclusive OR	(R_a) ⊕̄ (Y) → R_a	0	0	X

② if a ≠ m

OCTAL FORMAT o f a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
33 0 a m	MSR a,m	Masked Substitute (Register)	If (R_{a+1})$_n$=1; (R_m)$_n$ → R_{a_n}	0	0	X
33 1 a m	MSI a,m	Masked Substitute (Indirect)	If (R_{a+1})$_n$=1; (Y*)$_n$ → R_{a_n}	0	0	X
33 2 a m	MSK a,y,m	Masked Substitute (Constant)	If (R_{a+1})$_n$=1; Y_n → R_{a_n}	0	0	X
33 3 a m	MS a,y,m	Masked Substitute	If (R_{a+1})$_n$=1; (Y)$_n$ → R_{a_n}	0	0	X
34 0 a m	CMR a,m	Compare Masked (Register)	[(R_a) ⊙ (R_{a+1})]: [(R_m) ⊙ (R_{a+1})]	0	0	X
34 1 a m	CMI a,m	Compare Masked (Indirect)	[(R_a) ⊙ (R_{a+1})]: [(Y*) ⊙ (R_{a+1})]	0	0	X
34 2 a m	CMK a,y,m	Compare Masked (Constant)	[(R_a) ⊙ (R_{a+1})]: [Y ⊙ (R_{a+1})]	0	0	X
34 3 a m	CM a,y,m	Compare Masked	[(R_a) ⊙ (R_{a+1})] : [(Y) ⊙ (R_{a+1})]	0	0	X
35 0 00 00	IOCR	Input/Output Command	Execute (0140); 0 → 0140$_{15,14}$	–	NA	–
35 1 00 m	BFI m	Biased Fetch (Indirect)	(Y*) → CC; 1 → Y*$_{15,14}$	0	0	X
35 2 00 m	REX y,m	Remote Execute	Execute (Y); (P) + 2 → P	–	NA	–
35 3 00 m	BF y,m	Biased Fetch	(Y)$_{15}$ → CC; 1 → Y$_{15,14}$	0	0	X
#37 0 a m	See page 6	Trig & Hyperbolic		–	NA	–
40 0 00 m	JER m	Jump Equal	If CC indicates = or 0; (R_m) → P	–	NA	–
40 0 01 m	JNER m	Jump Not Equal	If CC indicates ≠ or not 0; (R_m) → P	–	NA	–
40 0 02 m	JGER m	Jump Greater or Equal	If CC indicates ≥ or +; (R_m) → P	–	NA	–
40 0 03 m	JLSR m	Jump Less	If CC indicates < or −; (R_m) → P	–	NA	–
40 0 04 m	JOR m	Jump Overflow	If overflow set: (R_m) → P	–	NA	–
40 0 05 m	JCR m	Jump Carry	If carry set: (R_m) → P	–	NA	–
40 0 06 m	JPTR m	Jump Power out of Tolerance	If power out of tolerance: (R_m) → P	–	NA	–
40 0 07 m	JBR m	Jump Bootstrap 2 selected	If bootstrap 2 selected: (R_m) → P	–	NA	–
40 0 10 m	JR m	Jump	(R_m) → P	–	NA	–
40 0 11 m	JSR m	Jump after Stop	Stop; (R_m) → P	–	NA	–
40 0 12 m	JKSR 1,m	Jump. If Key set−Stop, then jump (Register)	If key 1 set , stop; (R_m) → P	–	NA	–
40 0 13 m	JKSR 2,m	Jump. If Key set−Stop, then jump (Register)	If key 2 set , stop; (R_m) → P	–	NA	–
40 1 d	LJ xD	Local Jump	(P) + xD → P	–	NA	–
40 2 00 m	JE y,m	Jump Equal	If CC indicates = or 0; Y → P	–	NA	–
40 2 01 m	JNE y,m	Jump Not Equal	If CC indicates ≠ or not 0; Y → P	–	NA	–
40 2 02 m	JGE y,m	Jump Greater than or Equal	If CC indicates ≥ or +; Y → P	–	NA	–
40 2 03 m	JLS y,m	Jump Less	If CC indicates < or −; Y → P	–	NA	–
40 2 04 m	JO y,m	Jump on Overflow	If overflow set : Y → P	–	NA	–
40 2 05 m	JC y,m	Jump on Carry	If carry set , Y → P	–	NA	–
40 2 06 m	JPT y,m	Jump if Power out of Tolerance	If power out of tolerance; Y → P	–	NA	–
40 2 07 m	JB y,m	Jump if Bootstrap 2 selected	If bootstrap 2 selected; Y → P	–	NA	–
40 2 10 m	J y,m	Jump	Y → P	–	NA	–
40 2 11 m	JS y,m	Jump after Stop	Stop; Y → P	–	NA	–
40 2 12 m	JKS 1,y,m	Jump. If Key set−Stop, then jump	If key 1 set , stop; Y → P	–	NA	–
40 2 13 m	JKS 2,y,m	Jump. If Key set−Stop, then jump	If key 2 set , stop; Y → P	–	NA	–
40 3 00 m	JE *y,m	Jump Equal	If CC indicates = or 0; (Y) → P	–	NA	–
40 3 01 m	JNE *y,m	Jump Not Equal	If CC indicates ≠ or not 0; (Y) → P	–	NA	–
40 3 02 m	JGE *y,m	Jump Greater or Equal	If CC indicates ≥ or +; (Y) → P	–	NA	–
40 3 03 m	JLS *y,m	Jump Less	If CC indicates < or −; (Y) → P	–	NA	–
40 3 04 m	JO *y,m	Jump on Overflow	If overflow set; (Y) → P	–	NA	–
40 3 05 m	JC *y,m	Jump on Carry	If carry set; (Y) → P	–	NA	–
40 3 06 m	JPT *y,m	Jump if Power out of Tolerance	If power out of tolerance; (Y) → P	–	NA	–
40 3 07 m	JB *y,m	Jump if Bootstrap 2 selected	If bootstrap 2 selected; (Y) → P	–	NA	–
40 3 10 m	J *y,m	Jump	(Y) → P	–	NA	–
40 3 11 m	JS *y,m	Jump After Stop	Stop; (Y) → P	–	NA	–
40 3 12 m	JKS 1,*y,m	Jump. If Key set−Stop, then jump	If key 1 set, stop; (Y) → P	–	NA	–
40 3 13 m	JKS 2,*y,m	Jump. If Key set−Stop, then jump	If key 2 set, stop; (Y) → P	–	NA	–
41 0 a m	XJR a,m	Index Jump Register	If (R_a) ≠ 0; (R_a) − 1 → R_a, (R_m) → P	–	NA	–
41 1 d	LJI xD	Local Jump (Indirect)	[(P) + xD] → P	–	NA	–
41 2 a m	XJ a,y,m	Index Jump	If (R_a) ≠ 0; (R_a) − 1 → R_a; Y → P	–	NA	–
41 3 a m	XJ a,*y,m	Index Jump	If (R_a) ≠ 0; (R_a) − 1 → R_a; (Y) → P	–	NA	–
42 0 a m	JLRR a,m	Jump, Link Register (Register)	(P) + 1 → R_a; (R_m) → P	–	NA	–
42 2 a m	JLR a,y,m	Jump, Link Register	(P) + 2 → R_a; Y → P	–	NA	–
42 3 a m	JLR a,*y,m	Jump, Link Register	(P) + 2 → R_a; (Y) → P	–	NA	–
43 1 d	LJLM xD	Local Jump, Link Memory	(P) + 1 → (P) + xD; (P) + xD + 1 → P	–	NA	–
43 2 00 m	JLM y,m	Jump, Link Memory	(P) + 2 → Y; Y + 1 → P	–	NA	–
43 3 00 m	JLM *y,m	Jump, Link Memory	(P) + 2 → (Y); (Y) + 1 → P	–	NA	–
44 0 a m	JZR a,m	Jump Zero (Register)	If (R_a) = 0; (R_m) → P	–	NA	–
44 1 d	LJE xD	Local Jump Equal	If CC indicates = or 0; (P) + xD → P	–	NA	–
44 2 a m	JZ a,y,m	Jump Zero	If (R_a) = 0; Y → P	–	NA	–
44 3 a m	JZ a,*y,m	Jump Zero	If (R_a) = 0; (Y) → P	–	NA	–

Optional Math Pac Instruction

OCTAL FORMAT o f a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
45 0 a m	JNZR a,m	Jump Not Zero (Register)	If $(R_a) \neq 0$; $(R_m) \rightarrow P$	–	NA	–
45 1 d	LJNE xD	Local Jump Not Equal	If CC indicates $\neq$ or not 0; (P) + xD $\rightarrow$ P	–	NA	–
45 2 a m	JNZ a,y,m	Jump Not Zero	If $(R_a) \neq 0$; Y $\rightarrow$ P	–	NA	–
45 3 a m	JNZ a,*y,m	Jump Not Zero	If $(R_a) \neq 0$; (Y) $\rightarrow$ P	–	NA	–
46 0 a m	JPR a,m	Jump Positive (Register)	If $(R_a) \geq 0$; $(R_m) \rightarrow P$	–	NA	–
46 1 d	LJGE xD	Local Jump Greater or Equal	If CC indicates $\geq$ or +; (P) + xD $\rightarrow$ P	–	NA	–
46 2 a m	JP a,y,m	Jump Positive	If $(R_a) \geq 0$; Y $\rightarrow$ P	–	NA	–
46 3 a m	JP a,*y,m	Jump Positive	If $(R_a) \geq 0$; (Y) $\rightarrow$ P	–	NA	–
47 0 a m	JNR a,m	Jump Negative (Register)	If $(R_a) < 0$; $(R_m) \rightarrow P$	–	NA	–
47 1 d	LJLS xD	Local Jump Less	If CC indicates $<$ or – ; (P) + xD $\rightarrow$ P	–	NA	–
47 2 a m	JN a,y,m	Jump Negative	If $(R_a) < 0$; Y $\rightarrow$ P	–	NA	–
47 3 a m	JN a,*y,m	Jump Negative	If $(R_a) < 0$; (Y) $\rightarrow$ P	–	NA	–
# 50 0 a m	FSUR a,m	Floating point subtract (Register)	$(R_a, R_{a+1}) - (R_m, R_{m+1}) \rightarrow R_a$, R_{a+1}; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 50 1 a m	FSUI a,m	Floating point Subtract (Indirect)	$(R_a, R_{a+1}) - (Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 50 3 a m	FSU a,y,m	Floating point Subtract	$(R_a, R_{a+1}) - (Y, Y+1) \rightarrow R_a, R_{a+1}$; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 51 0 a m	FAR a,m	Floating point Add (Register)	$(R_a, R_{a+1}) + (R_m, R_{m+1}) \rightarrow R_a$, R_{a+1}; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 51 1 a m	FAI a,m	Floating point Add (Indirect)	$(R_a, R_{a+1}) + (Y^*, Y^*+1) \rightarrow R_a$, R_{a+1}; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 51 3 a m	FA a,y,m	Floating point Add	$(R_a, R_{a+1}) + (Y, Y+1) \rightarrow R_a, R_{a+1}$; Res. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 52 0 a m	FMR a,m	Floating point Multiply (Register)	$(R_a, R_{a+1}) \cdot (R_m, R_{m+1}) \rightarrow R_a$, $R_{a+1}, R_{a+2}, R_{a+3}$	0	X	X
# 52 1 a m	FMI a,m	Floating point Multiply (Indirect)	$(R_a, R_{a+1}) \cdot (Y^*, Y^*+1) \rightarrow R_a$, $R_{a+1}, R_{a+2}, R_{a+3}$	0	X	X
# 52 3 a m	FM a,y,m	Floating point Multiply	$(R_a, R_{a+1}) \cdot (Y, Y+1) \rightarrow R_a$, $R_{a+1}, R_{a+2}, R_{a+3}$	0	X	X
# 53 0 a m	FDR a,m	Floating point Divide (Register)	$(R_a, R_{a+1}) / (R_m, R_{m+1}) \rightarrow R_a$, R_{a+1}; Rem. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 53 1 a m	FDI a,m	Floating point Divide (Indirect)	$(R_a, R_{a+1}) / (Y^*, Y^*+1) \rightarrow R_a$, R_{a+1}; Rem. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
# 53 3 a m	FD a,y,m	Floating point Divide	$(R_a, R_{a+1}) / (Y, Y+1) \rightarrow R_a, R_{a+1}$; Rem. $\rightarrow R_{a+2}, R_{a+3}$	0	X	X
54 0 a m	LARR a,m	Load Address Register (Register)	$(R_m) \rightarrow AR_r$ SEE LEGEND	–	NA	–
54 1 a m	LARI a,m	Load Address Register (Indirect)	$(Y^*) \rightarrow AR_r$	–	NA	–
54 3 a m	LARM a,y,m	Load Address Register Multiple	$(Y, \ldots Y+u) \rightarrow AR_r \ldots AR_{r+u}$	–	NA	–
55 0 a m	SARR a,m	Store Address Register (Register)	$(AR_r) \rightarrow R_m$	–	NA	–
55 1 a m	SARI a,m	Store Address Register (Indirect)	$(AR_a) \rightarrow Y^*$	–	NA	–
55 3 a m	SARM a,y,m	Store Address Register Multiple	$(AR_r, \ldots AR_{r+u}) \rightarrow Y, \ldots Y+u$	–	NA	–
# 56 0 a m	MDR a,m	Multiply Double (Register)	$(R_a, R_{a+1}) \cdot (R_m, R_{m+1}) \rightarrow R_a$, $R_{a+1}, R_{a+2}, R_{a+3}$	0	0	X
# 56 1 a m	MDI a,m	Multiply Double (Indirect)	$(R_a, R_{a+1}) \cdot (Y^*, Y^*+1) \rightarrow R_a$, $R_{a+1}, R_{a+2}, R_{a+3}$	0	0	X
# 56 3 a m	MD a,y,m	Multiply Double	$(R_a, R_{a+1}) \cdot (Y, Y+1) \rightarrow R_a, R_{a+1}$, R_{a+2}, R_{a+3}	0	0	X
# 57 0 a m	DDR a,m	Divide Double (Register)	$(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (R_m, R_{m+1}) \rightarrow R_{a+2}, R_{a+3}$; Rem. $\rightarrow R_a, R_{a+1}$	0	X	X
# 57 1 a m	DDI a,m	Divide Double (Indirect)	$(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (Y^*, Y^*+1) \rightarrow R_{a+2}, R_{a+3}$; Rem. $\rightarrow R_a, R_{a+1}$	0	X	X
# 57 3 a m	DD a,y,m	Divide Double	$(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (Y, Y+1) \rightarrow R_{a+2}, R_{a+3}$; Rem. $\rightarrow R_a, R_{a+1}$	0	X	X
60 0 a m	LLRS a,m	Literal Logical Right Shift	Shift (R_a) right m places, zero fill	0	0	X
60 1 a m	LARS a,m	Literal Algebraic Right Shift	Shift (R_a) right m places, sign fill	0	0	X
60 2 a m	LLRD a,m	Literal Logical Right Double shift	Shift (R_a, R_{a+1}) right m places, zero fill	0	0	X
60 3 a m	LARD a,m	Literal Algebraic Right Double shift	Shift (R_a, R_{a+1}) right m places, sign fill	0	0	X
61 0 a m	LALS a,m	Literal Algebraic Left Shift	Shift (R_a) left m places, zero fill	0	X	X
61 1 a m	LCLS a,m	Literal Circular Left Shift	Shift (R_a) left circular m places	0	0	X
61 2 a m	LALD a,m	Literal Algebraic Left Double shift	Shift (R_a, R_{a+1}) left m places, zero fill	0	X	X
61 3 a m	LCLD a,m	Literal Circular Left Double shift	Shift (R_a, R_{a+1}) left circular m places	0	Q	X

Optional Math Pac Instructions

OCTAL FORMAT o f a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
62 0 a m	LSU a,m	Literal Subtract	$(R_a) - m \rightarrow R_a$	X	X	X
62 1 a m	LSUD a,m	Literal Subtract Double	$(R_a, R_{a+1}) - m \rightarrow R_a, R_{a+1}$	X	X	X
62 2 a m	LA a,m	Literal Add	$(R_a) + m \rightarrow R_a$	X	X	X
62 3 a m	LAD a,m	Literal Add Double	$(R_a, R_{a+1}) + m \rightarrow R_a, R_{a+1}$	X	X	X
63 0 a m	LL a,m	Literal Load	$m \rightarrow R_a$	0	0	X
63 1 a m	LC a,m	Literal Compare	(R_a) : m	X	X	X
63 2 a m	LMUL a,m	Literal Multiply	$(R_{a+1}) \cdot m \rightarrow R_a, R_{a+1}$	0	0	X
63 3 a m	LDIV a,m	Literal Divide	$(R_a, R_{a+1}) / m \rightarrow R_{a+1}$; remainder $\rightarrow R_a$	0	X	X
64 3 a m	BSU a,y,m	Byte Subtract	$(R_a) - (Y)_{byte} \rightarrow R_a$	X	X	X
65 3 a m	BA a,y,m	Byte Add	$(R_a) + (Y)_{byte} \rightarrow R_a$	X	X	X
66 3 a m	BC a,y,m	Byte Compare	$(R_a) : (Y)_{byte}$	X	X	X
67 0 a m	UM1 a,m UM2 a,m	User Macro	Reserved for User Macro	–	NA	–
67 1 a m	UMI a,m	User Macro	Reserved for User Macro	–	NA	–
67 2 a m	UMK a,y,m	User Macro	Reserved for User Macro	–	NA	–
67 3 a m	BCX a,y,m	Byte Compare and Index By 1	$(R_a) : (Y)_{byte}$; $(R_m) + 1 \rightarrow R_m$	X	X	X

COMMAND/CHAIN INSTRUCTION

70 0 00 00	ACR m	Channel Control	Master clear all channels
70 0 00 04	ACR m CCR 0,m	Channel Control	Enable external interrupts, all channels
70 0 00 05	ACR m CCR 0,m	Channel Control	Disable external interrupts, all channels
70 0 00 06	ACR m CCR 0,m	Channel Control	Enable Class III, Priority 2, 3, 4 interrupts
70 0 00 07	ACR m CCR 0,m	Channel Control	Disable Class III, Priority 2, 3, 4 interrupts
70 0 a 10	CCR a,m	Channel Control	Master clear chan. a
70 0 a 14	CCR a,m	Channel Control	Enable chan. a interrupts
70 0 a 15	CCR a,m	Channel Control	Disable chan. a interrupts
70 0 a 16	CCR a,m	Channel Control	Enable chan. a Class III, Priority 2, 3, 4 interrupts
70 0 a 17	CCR a,m	Channel Control	Disable chan. a Class III, Priority 2, 3, 4 interrupts

COMMAND INSTRUCTION

71 2 a 02	ICK a,y	Initiate Input Chain	Y $\rightarrow$ Channel a Chain Pointer; initiate input chain
71 2 a 06	OCK a,y	Initiate Output Chain	Y $\rightarrow$ Channel a Chain Pointer; initiate output chain
71 3 a m	WIM a,y,m WCM am, y	Write Control Memory	(Y) $\rightarrow$ Chan. a CM_m (See Figure 6)
72 3 a m	RIM a,y,m RCM am,y	Read Control Memory	Chan. a $(CM_m) \rightarrow$ Y (See Figure 6)
76 0 a m	SICR a,m	Serial Interface Control	Set or clear chan. a discrete function per Table 3
76 3 a m	SST a,y	Store Serial Status	Channel a Serial Status bits $\rightarrow$ Y (See Table 4)

CHAIN INSTRUCTION

70 3 00 00	IO 0,y	Input Data	(Y, Y+1) $\rightarrow$ BCW, BAP; initiate transfer
70 3 01 00	IO 1,y	Output Data	(Y, Y+1) $\rightarrow$ BCW, BAP; initiate transfer
70 3 02 00	IO 2,y	External Function	(Y, Y+1) $\rightarrow$ BCW, BAP; initiate transfer
70 3 03 00	IO 3,y	Force External Function	(Y, Y+1) $\rightarrow$ BCW, BAP; initiate transfer
71 2 00 m	LCMK m,y	Load Control Memory	Y $\rightarrow CM_m$ (See Figure 6)
71 3 00 m	LCM m,y	Load Control Memory	(Y) $\rightarrow CM_m$ (See Figure 6)
72 3 00 m	SCM m,y	Store Control Memory	$(CM_m) \rightarrow$ Y (See Figure 6)
73 0 00 00	HCR	Halt Chain	Halt chaining
73 0 01 00	IPR	Interrupt Processor	Generate chain interrupt
73 3 00 00	ZF y	Zero Flag	$0 \rightarrow Y_{15,14}$
73 3 01 00	SF y	Set Flag	$1 \rightarrow Y_{15,14}$
74 2 00 00	SJMC 0,y	Serial Jump on Met Condition	Unconditional Y $\rightarrow$ CAP ; clear flag
74 2 01 00	SJMC 1,y	Serial Jump on Met Condition	If suppress flag not set, Y $\rightarrow$ CAP ; clear flag
74 2 02 00	SJMC 2,y	Serial Jump on Met Condition	If monitor flag set, Y $\rightarrow$ CAP; clear flag
75 0 00 m	SFSC m	Search For Sync	Perform function(s) assigned to m-bits per Figure 7
76 0 00 m	CSIR m	Serial Interface Control	Set or clear discrete function per Table 3
76 3 00 00	CSST y	Store Serial Status	Serial Status bits $\rightarrow$ Y; See Table 4

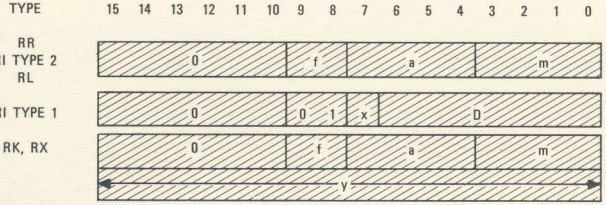
#TRIGONOMETRIC AND HYPERBOLIC FUNCTIONS
(Operation Code 37)

x&y Cartesian coordinates. Radix point assumed between bits 15 and 14
θ Angle of rotation Trignonometric mode (BAMS) Bit 14 = 90°
v Angle of rotation Hyperbolic mode Radix point assumed between bits 15 and 14
K 0.46672_8
K_1 1.15217_8

o f a m	CODING FORMAT	FUNCTION	INPUT PARAMETERS			OUTPUT RESULTS		
			R_a	R_{a+1}	R_{a+2}	$Y \rightarrow R_a$	$X \rightarrow R_{a+1}$	$W \rightarrow R_{a+2}$
37 0 a 00	VF a	Trigonometric vector	y	x	0	0	$X = \frac{R}{K}=\sqrt{x^2+y^2}$ / K	$W = \theta = \tan^{-1} \frac{y}{x}$
37 0 a 01	RF a	Trigonometric rotate	y	x	θ	$Y = \frac{y \cos \theta + x \sin \theta}{K}$	$X = \frac{x \cos \theta - y \sin \theta}{K}$	0
37 0 a 02	VFP a	Trig. vector with prescale	y	x	0	0	$X = R = \sqrt{x^2+y^2}$	$W = \theta = \tan^{-1} \frac{y}{x}$
37 0 a 03	RFP a	Trig. rotate with prescale	y	x	θ	$Y = y \cos \theta + x \sin \theta$	$X = x \cos \theta - y \sin \theta$	0
37 0 a 04	VH a	Hyperbolic vector	y	x	0	0	$X = \frac{\sqrt{x^2-y^2}}{K_1}$	$W = v = \tanh^{-1} \frac{y}{x}$
37 0 a 05	RH a	Hyperbolic rotate	y	x	v	$Y = \frac{y \cosh v + x \sinh v}{K_1}$	$X = \frac{x \cosh v + y \sinh v}{K_1}$	0
37 0 a 06	VHP a	Hyp. vector with postscale	y	x	0	0	$X = \sqrt{x^2-y^2}$	$W = v = \tanh^{-1} \frac{y}{x}$
37 0 a 07	RHP a	Hyp. rotate with postscale	y	x	v	$Y = y \cosh v + x \sinh v$	$X = x \cosh v + y \sinh v$	0
37 0 a 01	RF a	Sin θ, COS θ without prescale	0	0.46672_8	θ	$Y = \sin \theta$	$X = \cos \theta$	0
37 0 a 06	VHP a	$\text{Log}_e x$	x-1	x+1	0	0	2x	$W = 1/2 \log_e x$ $= \tanh^{-1} \frac{x+1}{x-1}$
37 0 a 07	RHP a	Exponential	1	1	v positive	$Y = e^v = \sinh v + \cosh v$	$X = e^v = \sinh v + \cosh v$	0
37 0 a 01	RF a	Polar to Cartesian without prescale	0	R	θ	$Y = \frac{R \sin \theta}{K}$	$X = \frac{R \cos \theta}{K}$	0
37 0 a 03	RFP a	Polar to Cartesian with prescale	0	R	θ	$Y = R \sin \theta$	$X = R \cos \theta$	0
37 0 a 01	RF a	Sin θ; cos θ	0	1	θ	$Y = \frac{\sin \theta}{K}$	$X = \frac{\cos \theta}{K}$	0

Optional Math Pac Instructions

SPERRY UNIVAC
DEFENSE SYSTEMS



DEFINITION OF FIELDS

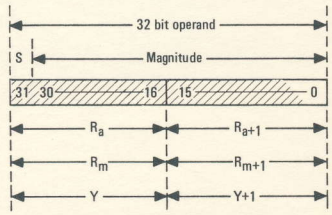
- 0 Operation (Function) Code
- f Format Designator
  - 00 ⇒ Format RR, Register to Register or RL-1 Format
  - 01 ⇒ Format RI, Register Indirect Memory or RL-2 Format
  - 10 ⇒ Format RK, Register-Literal Constant or RL-3 Format
  - 11 ⇒ Format RX, Register-Indexed Address, Constant or RL-4 Format
- a General Register or Subfunction Designator
- m General Register or Subfunction Designator
  - 4-bit Unsigned Literal Constant in RL Format
- xD Signed Deviation Value (Two's Complement)
- y Address or Arithmetic Constant

Figure 1. Instruction Word Format

LEGEND

- B Byte pointer, 0 → Upper, 1 → Lower
- C Carry
- CC Condition Code
- OV Overflow
- IW Indirect Word
- j Designator Field in IW
- x General Register Designator in IW1
- y Contents of Second Instruction Word or IW2
- Y Effective Operand Address or Constant
- Y* Effective Operand Address in Rm
- TM I/O Transfer Mode
  - 00 → Abort Transfer
  - 01 → 8-bit Byte Transfer
  - 10 → 16-bit Word Transfer
  - 11 → 32-bit Dual Word Transfer
- BWC Buffer Word Count
- BAP Buffer Address Pointer
- CM Control Memory Word
- CAP Chain Address Pointer
- RTC Real-Time Clock
- ( ) Contents of register or address
- r $(R_a)_{5\text{-}0}$
- u $(R_a)_{13\text{-}8}$

FORMAT	OPERAND FORMATION
RR	Operand = (R_m)
RI-1	Local Jump Address Y = (P) + xD
RI-2	Operand at Y* = (R_m)
RK	Operand Y = y + (R_m) if m ≠ 0 Operand Y = y if m = 0
RX Word	Operand at Y = y if m = 0 Operand at Y = y + (R_m) if m ≠ 0,10,12,14,16 Operand at indirect address if m = 10,12,14,16
RX Byte	Operand at Y upper if m = 0 Operand at Y = $(R_m)/2$ + y if m ≠ 0,10,12,14, 16; B = $(R_m)_0$ Operand at indirect address if m = 10,12,14,16
RL	Operand = m (an absolute literal)



Double Length Operands

OR ⊕	0	1
0	0	1
1	1	1

XOR ⊕	0	1
0	0	1
1	1	0

AND ⊙	0	1
0	0	0
1	0	1

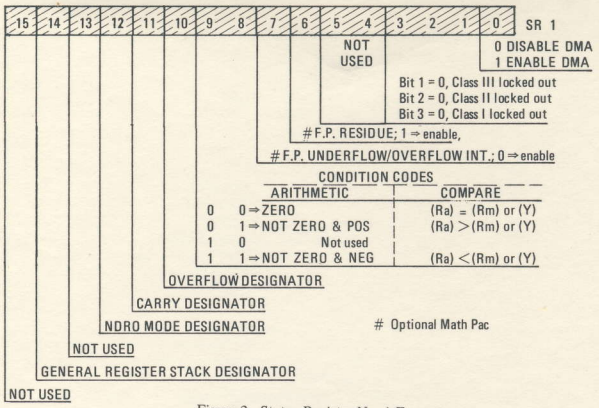


Figure 2. Status Register No. 1 Format

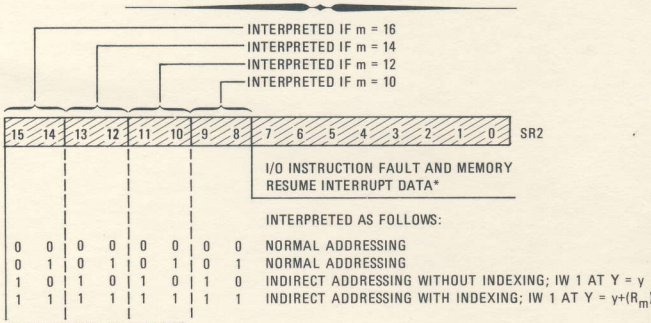
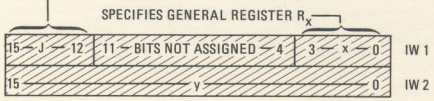


Figure 3. Status Register No. 2 Format

OCTAL J-VALUE	OPERAND/IW1, LOCATION
0	WORD AT Y = (IW2) BYTE AT UPPER HALF OF Y = (IW2)
1	WORD AT Y = (IW2) + (R_x) BYTE AT Y = (IW2) + (R_x) */2
2	WORD AT Y = (IW2) + (R_m) BYTE AT Y = (IW2) + (R_m)*/2
3	WORD AT Y = (IW2) + (R_{m+1}) BYTE AT Y = (IW2) + (R_{m+1}) */2
4	NEXT IW 1 AT ADDRESS Y = (IW2)
5	NEXT IW 1 AT ADDRESS Y = (IW2) + (R_x)
6	NEXT IW 1 AT ADDRESS Y = (IW2) + (R_m)
7	NEXT IW 1 AT ADDRESS Y = (IW2) + (R_{m+1})
10-17	NOT ASSIGNED



* B = LSB of register

Figure 4. Indirect Addressing

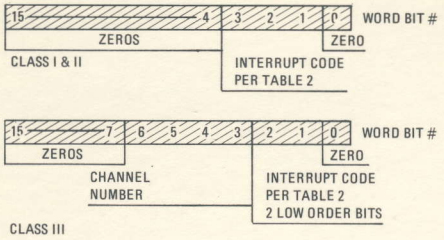


Figure 5. Interrupt Entrance Address Index

TABLE 1. ASSIGNED MEMORY ADDRESS

Function	Address Assignment to Class III	II	I
Store P addresses	110	120	130
Store SR #1 addresses	111	121	131
Store SR #2 addresses	112	122	132
Store RTC lower addresses	113	123	133
P Reload addresses	114	124	134
SR #1 Reload addresses	115	125	135
SR #2 Reload addresses	116	126	136
Store RTC upper addresses	117	127	137
I/O Command cells	140-141		
Auto start entrance	177		
External interrupt word storage	200-217		
NDRO	00-77, 300-477		

TABLE 2. INTERRUPT PRIORITY

Class	Priority Within Class	Interrupt	Binary Interrupt Code Generated
Class I, Hardware Errors	1	Power Fault	000
	2	Memory Resume	001
Class II, Software Interrupts	1	CP Instruction Fault	000
	2	I/O Instruction Fault	001
	3	#F.P. Overflow/Underflow Interrupt	010
	4	Executive Return Instruction	011
	5	RTC Overflow	100
	6	Monitor Clock	101
Class III, IOC Interrupts	1	Intercomputer Time-Out	11
	2	External Interrupt or Discrete Interrupt ①	00
	3	Output Chain Interrupt	10
	4	Input Chain Interrupt	01

① Serial MIL-STD-188C vacales, or EIA-STD-RS 232C Channels # Optional Math Pac function

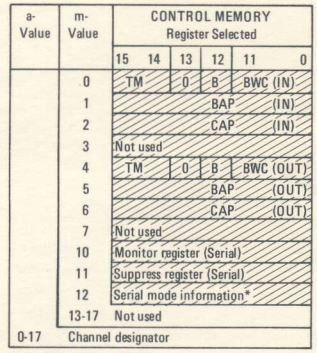


Figure 6. I/O Control Memory

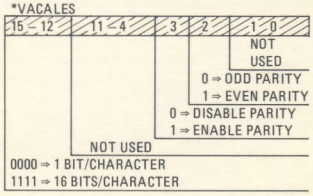


Figure 6. I/O Control Memory

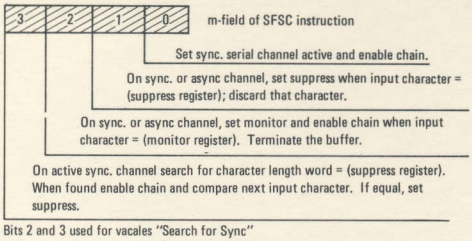


Figure 7. SFSC Operations

BITS	MIL-STD-188	RS-232	VACALES
0 - 7	ALWAYS ONES	ALWAYS ONES	ALWAYS ONES
8	1 ⇒ B DISCRETE TURNED ON	1 ⇒ RING INDICATOR ON	1 ⇒ B DISCRETE TURNED ON
9	1 ⇒ C DISCRETE TURNED OFF	1 ⇒ RECEIVED LINE SIGNAL DETECTOR OFF	1 ⇒ CARRIER DETECT TURNED OFF
10	1 ⇒ I DISCRETE TURNED ON	ALWAYS ONE	1 ⇒ ALARM INDICATE TURNED ON
11	ALWAYS ONE	ALWAYS ONE	1 ⇒ SYNC ERROR TURNED ON
12	ALWAYS ONE	ALWAYS ONE	1 ⇒ TRANSMIT FULL ON TURNED OFF
13 - 15	ALWAYS ONES	ALWAYS ONES	ALWAYS ONES

FIGURE 8. SERIAL CHANNEL INTERRUPT WORD FORMAT

TABLE 3. SERIAL I/O DISCRETE FUNCTIONS

Octal m-Value	Function	MIL-STD-188C/VACALES Discrete	Line Designator (188C)	Line Designator (Vacales)	EIA-STD-RS232 Discrete	Line Designator
0	Set	Loop test (internal)	–	–	Loop test (internal)	–
1	Clear	Loop test (internal)	–	–	Loop test (internal)	–
2	Clear	Not used	–	–	Spare	–
3	Set	Not used	–	–	Spare	–
4	Clear	Control Line 6	J	J1	Spare	–
5	Set	Control Line 6	J	J1	Spare	–
6	Clear	Control Line 5	H	TRAN. PREP	Enable Ring Indicator	CE
7	Set	Control Line 5	H	TRAN. PREP	Enable Ring Indicator	CE
10	Clear	Control Line 4	G	G1	Request to Send	CA
11	Set	Control Line 4	G	G1	Request to Send	CA
12	Clear	Control Line 3	F	F1	New Sync	CH
13	Set	Control Line 3	F	F1	New Sync	CH
14	Clear	Control Line 2	D	D1	Data Terminal Ready	CD
15	Set	Control Line 2	D	D1	Data Terminal Ready	CD
16	Clear	Control Line 1	A	LOOP BACK	Loop Test (external)	
17	Set	Control Line 1	A	LOOP BACK	Loop Test (external)	

TABLE 4. SERIAL I/O STATUS INTERPRETATION

Word Bit #	MIL-STD-188 Function	EIA-STD-RS232 Function	VACALES FUNCTION
2^0	Parity Error	Parity Error	–
2^1	Overrun	Overrun	Overrun
2^2	Break	Break	Parity Error
2^3	E Active	Clear to Send	Sync Error