OCTAL FORMAT				HEXIDECIMAL FORMAT				CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
o	f	a	m	OP	a	m							
00	0	–	–	00	–	–		–	Diagnostic return	If diagnostic jump set $R_{17} \rightarrow \mu P$	–	NC	–
00	3	a	m	03	a	m		BL a,y,m	Byte load	(Y) byte $\rightarrow$ Ra_{7-0}; 0 $\rightarrow$ Ra_{15-8}	0	0	X
01	0	a	m	04	a	m		LR a,m	Load (Register)	$(Rm), \rightarrow R_a$	0	0	X
01	1	a	m	05	a	m		LI a,m	Load (Indirect)	$(Y^*) \rightarrow R_a$	0	0	X
01	2	a	m	06	a	m		LK a,y,m	Load (Constant)	$Y \rightarrow R_a$	0	0	X
01	3	a	m	07	a	m		L a,y,m	Load	$(Y) \rightarrow R_a$	0	0	X
02	0	a	00	08	a	0		PR a	Make positive	If $(R_a) < 0$, $(R_a)' \rightarrow R_a$	X	X	X
02	0	a	01	08	a	1		NR a	Make negative	If $(R_a) > 0$, $(R_a)' \rightarrow R_a$	X	0	X
02	0	a	02	08	a	2		RR a	Round	$(R_a) + (R_{a+1})_{15} \rightarrow R_a$ ③	X	X	X
02	0	a	04	08	a	4		TCR a	Two's Complement	$(R_a)' \rightarrow R_a$	X	X	X
02	0	a	05	08	a	5		TCDR a	Two's Complement Double	$(R_a, R_{a+1})' \rightarrow R_a, R_{a+1}$	X	X	X
02	0	a	06	08	a	6		OCR a	One's Complement	(R_a) bit-by-bit complement $\rightarrow R_a$	0	0	X
02	0	a	10	08	a	8		IROR a	Increase R_a by 1	$(R_a) + 1 \rightarrow R_a$	X	X	X
02	0	a	11	08	a	9		DROR a	Decrease R_a by 1	$(R_a) - 1 \rightarrow R_a$	X	X	X
02	0	a	12	08	a	A		IRTR a	Increase R_a by 2	$(R_a) + 2 \rightarrow R_a$	X	X	X
02	0	a	13	08	a	B		DRTR a	Decrease R_a by 2	$(R_a) - 2 \rightarrow R_a$	X	X	X
02	1	a	m	09	a	m		LDI a,m	Load Double (Indirect)	$(Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$ ③	0	0	X
02	3	a	m	0B	a	m		LD a,y,m	Load Double	$(Y, Y+1) \rightarrow R_a, R_{a+1}$ ③	0	0	X
03	0	a	00	0C	a	0		ER a	Executive Return	Generate interrupt; $(P)+1 \rightarrow R_a$ ⑥	0	0	X
03	0	a	01	0C	a	1		SSOR a	Store SR1	$(SR1) \rightarrow R_a$	0	0	X
03	0	a	02	0C	a	2		SSTR a	Store SR2	$(SR2) \rightarrow R_a$	0	0	X
03	0	a	03	0C	a	3		SCR a	Store Clock	$(RTC\ register)_{15-0} \rightarrow R_a$	0	0	X
03	0	a	04	0C	a	4		LPR a	Load P	$(R_a) \rightarrow P$	–	NC	–
03	0	a	05	0C	a	5		LSOR a	Load SR1	$(R_a) \rightarrow SR1$	–	NA	–
03	0	a	06	0C	a	6		LSTR a	Load SR2	$(R_a) \rightarrow SR2$	–	NC	–
03	0	a	07	0C	a	7		LCR a	Load RTC lower	$(R_a) \rightarrow RTC\ register_{15-0}$;	–	NC	–
03	0	00	10	0C	0	8		ECR	Enable Clock	Enable RTC reg. (countup and interrupt)	–	NC	–
03	0	00	11	0C	0	9		DCR	Disable Clock	Disable RTC reg. (countup and interrupt)	–	NC	–
03	0	a	12	0C	a	A		LEM a	Load and Enable Mon. clock	$(R_a) \rightarrow$ Mon. clock reg.; enable countdown and interrupt	–	NC	–
03	0	00	13	0C	0	B		DM	Disable Monitor clock	Disable Mon. clock reg. (countdown and interrupt) ③	–	NC	–
03	0	a	14	0C	a	C		LCRD a	Load and enable Clock Double	$(R_a, R_{a+1}) \rightarrow RTC$; enable countup only	–	NC	–
03	0	a	15	0C	a	D		SCRD a	Store Clock Double	(RTC Register) $\rightarrow R_a, R_{a+1}$ ③ ⑤	0	0	X
03	0	00	16	0C	0	E		ECIR	Enable Clock Interrupt	Enable RTC overflow interrupt	–	NC	–
03	0	00	17	0C	0	F		DCIR	Disable Clock Interrupt	Disable RTC overflow interrupt	–	NC	–
03	3	a	m	0F	a	m		LM a,y,m	Load multiple	$(Y_{...}\ Y+m-a) \rightarrow R_{a_{...}}\ R_m$	–	NC	–
# 04	0	a	00	10	a	0		SQR a	Square Root	$\sqrt{(R_a, R_{a+1})} \rightarrow R_{a+1}$; Rem. $\rightarrow R_a$ ③	0	X	X
04	0	a	01	10	a	1		RVR a	Reverse Register	Reverse (R_a)	0	0	X
04	0	a	02	10	a	2		CNT a	Count Ones	Number of binary ones in $R_a \rightarrow R_{a+1}$	–	NC	–
04	0	a	03	10	a	3		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	–	NC	–
04	3	a	m	13	a	m		BLX a,y,m	Byte Load and index by 1	(Y) byte $\rightarrow R_a$; $(R_m)+1 \rightarrow R_m$ ②	0	0	X
05	0	a	m	14	a	m		SBR a,m	Set Bit	$1 \rightarrow (R_a)_m$	0	0	X
05	1	a	m	15	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	17	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	18	a	m		ZBR a,m	Zero Bit	$0 \rightarrow (R_a)_m$	0	0	X
06	1	a	m	19	a	m		LDXI a,m	Load Double Index by 2 (Indirect)	$(Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$; ② ③ ④ $(R_m)+2 \rightarrow R_m$	0	0	X
06	3	a	m	1B	a	m		LDX a,y,m	Load Double, index by 2	$(Y, Y+1) \rightarrow R_a, R_{a+1}$; $(R_m)+2 \rightarrow R_m$ ④ ③ ②	0	0	X
07	0	a	m	1C	a	m		CBR a,m	Compare Bit	Test bit m of R_a for zero	0	0	X
07	1	00	m	1D	0	m		LPI m	Load PSW (Indirect)	$(Y^*, Y^*+1, Y^*+2) \rightarrow P, SR1, SR2$; enable power fault interrupt	–	NA	–
07	3	00	m	1F	0	m		LP y,m	Load PSW	(Y, Y+1, Y+2) → P, SR1, SR2; enable power fault interrupt	–	NA	–
10	0	a	m	20	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	22	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	23	a	m		BS a,y,m	Byte Store	$(R_a)_{7-0} \rightarrow Y_{byte}$	–	NC	–
11	0	a	m	24	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	25	a	m		SI a,m	Store (Indirect)	$(R_a) \rightarrow Y^*$	–	NC	–
11	2	a	m	26	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	27	a	m		S a,y,m	Store	$(R_a) \rightarrow Y$	–	NC	–
12	0	a	m	28	a	m		LRDR a,m	Logical Right Double shift (Register)	Shift (R_a, R_{a+1}) right $(R_m)_{5-0}$ places, zero fill ③	0	0	X
12	1	a	m	29	a	m		SDI a,m	Store Double (Indirect)	$(R_a, R_{a+1}) \rightarrow Y^*, Y^*+1$ ③	–	NC	–
12	2	a	m	2A	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	2B	a	m		SD a,y,m	Store Double	$(R_a, R_{a+1}) \rightarrow Y, Y+1$ ③	–	NC	–

Optional Math Pac Instructions ① Count = 31 for all zeros or all ones. ② if a ≠ m ③ a,m,y must be even
④ if a+1 ≠ m ⑤ cc set on Ra+1 only ⑥ if Class II interrupts enabled

OCTAL FORMAT o f a m	HEXIDECIMAL FORMAT OP a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
13 0 a m	2C 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	2E a m	ARD a,y,m	Algebraic Right Double shift	Shift (R_a, R_{a+1}) right Y_{5-0} places, sign fill ③	0	0	X
13 3 a m	2F a m	SM a,y,m	Store Multiple	$(R_a \ldots R_m) \rightarrow Y \ldots Y+m-a$	– NC –		
14 0 a m	30 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	32 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	33 a m	BSX a,y,m	Byte Store, index by 1	$(R_a)_{7-0} \rightarrow Y_{byte}$; $(R_m)+1 \rightarrow R_m$	– NC –		
15 0 a m	34 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	35 a m	SXI a,m	Store index by 1 (Indirect)	$(R_a) \rightarrow Y^*$; $(R_m)+1 \rightarrow R_m$	– NC –		
15 2 a m	36 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	37 a m	SX a,y,m	Store, index by 1	$(R_a) \rightarrow Y$; $(R_m)+1 \rightarrow R_m$	– NC –		
16 0 a m	38 a m	ALDR a,m	Algebraic Left Double shift (Register)	Shift (R_a, R_{a+1}) left $(R_m)_{5-0}$ places, zero fill ③	0	X	X
16 1 a m	39 a m	SDXI a,m	Store Double index by 2 (Indirect)	$(R_a, R_{a+1}) \rightarrow Y^*, Y^*+1$; $(R_m)+2 \rightarrow R_m$ ②③	– NC –		
16 2 a m	3A a m	ALD a,y,m	Algebraic Left Double shift	Shift (R_a, R_{a+1}) left Y_{5-0} places zero fill ③	0	X	X
16 3 a m	3B a m	SDX a,y,m	Store Double, index by 2	$(R_a, R_{a+1}) \rightarrow (Y, Y+1)$; $(R_m)+2 \rightarrow R_m$ ② ③	– NC –		
17 0 a m	3C a m	CLDR a,m	Circular Left Double shift (Register)	Shift (R_a, R_{a+1}) circularly left $(R_m)_{5-0}$ places ③	0	0	X
17 1 00 m	3D 0 m	SZI m	Store Zeros (Indirect)	$0 \rightarrow Y^*$	– NC –		
17 2 a m	3E a m	CLD a,y,m	Circular Left Double shift	Shift (R_a, R_{a+1}) circularly left Y_{5-0} places ③	0	0	X
17 3 00 m	3F 0 m	SZ y,m	Store Zeros	$0 \rightarrow Y$	– NC –		
20 0 a m	40 a m	SUR a,m	Subtract (Register)	$(R_a) - (R_m) \rightarrow R_a$	X	X	X
20 1 a m	41 a m	SUI a,m	Subtract (Indirect)	$(R_a) - (Y^*) \rightarrow R_a$	X	X	X
20 2 a m	42 a m	SUK a,y,m	Subtract (Constant)	$(R_a) - Y \rightarrow R_a$	X	X	X
20 3 a m	43 a m	SU a,y,m	Subtract	$(R_a) - (Y) \rightarrow R_a$	X	X	X
21 0 a m	44 a m	SUDR a,m	Subtract Double (Register)	$(R_a, R_{a+1}) - (R_m, R_{m+1}) \rightarrow R_a, R_{a+1}$	X	X	X
21 1 a m	45 a m	SUDI a,m	Subtract Double (Indirect)	$(R_a, R_{a+1}) - (Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$	X	X	X
21 3 a m	47 a m	SUD a,y,m	Subtract Double	$(R_a, R_{a+1}) - (Y, Y+1) \rightarrow R_a, R_{a+1}$ ③	X	X	X
22 0 a m	48 a m	AR a,m	Add (Register)	$(R_a) + (R_m) \rightarrow R_a$	X	X	X
22 1 a m	49 a m	AI a,m	Add (Indirect)	$(R_a) + (Y^*) \rightarrow R_a$	X	X	X
22 2 a m	4A a m	AK a,y,m	Add (Constant)	$(R_a) + Y \rightarrow R_a$	X	X	X
22 3 a m	4B a m	A a,y,m	Add	$(R_a) + (Y) \rightarrow R_a$	X	X	X
23 0 a m	4C a m	ADR a,m	Add Double (Register)	$(R_a, R_{a+1}) + (R_m, R_{m+1}) \rightarrow R_a, R_{a+1}$	X	X	X
23 1 a m	4D a m	ADI a,m	Add Double (Indirect)	$(R_a, R_{a+1}) + (Y^*, Y^*+1) \rightarrow R_a, R_{a+1}$ ③	X	X	X
23 3 a m	4F a m	AD a,y,m	Add Double	$(R_a, R_{a+1}) + (Y, Y+1) \rightarrow R_a, R_{a+1}$ ③	X	X	X
24 0 a m	50 a m	CR a,m	Compare (Register)	(R_a): (R_m)	X	X	X
24 1 a m	51 a m	CI a,m	Compare (Indirect)	(R_a): (Y^*)	X	X	X
24 2 a m	52 a m	CK a,y,m	Compare (Constant)	(R_a): Y	X	X	X
24 3 a m	53 a m	C a,y,m	Compare	(R_a): (Y)	X	X	X
25 0 a m	54 a m	CDR a,m	Compare Double (Register)	(R_a, R_{a+1}): (R_m, R_{m+1}) ③	X	X	X
25 1 a m	55 a m	CDI a,m	Compare Double (Indirect)	(R_a, R_{a+1}): (Y^*, Y^*+1) ③	X	X	X
25 3 a m	57 a m	CD a,y,m	Compare Double	(R_a, R_{a+1}): (Y, Y+1) ③	X	X	X
26 0 a m	58 a m	MR a,m	Multiply (Register)	$(R_{a+1}) \cdot (R_m) \rightarrow R_a, R_{a+1}$ ③	0	0	X
26 1 a m	59 a m	MI a,m	Multiply (Indirect)	$(R_{a+1}) \cdot (Y^*) \rightarrow R_a, R_{a+1}$ ③	0	0	X
26 2 a m	5A a m	MK a,y,m	Multiply (Constant)	$(R_{a+1}) \cdot Y \rightarrow R_a, R_{a+1}$ ③	0	0	X
26 3 a m	5B a m	M a,y,m	Multiply	$(R_{a+1}) \cdot (Y) \rightarrow R_a, R_{a+1}$ ③	0	0	X
27 0 a m	5C a m	DR a,m	Divide (Register)	$(R_a, R_{a+1}) / (R_m) \rightarrow R_{a+1}$; remainder $\rightarrow R_a$ ③	X	X	X
27 1 a m	5D a m	DI a,m	Divide (Indirect)	$(R_a, R_{a+1}) / (Y^*) \rightarrow R_{a+1}$; remainder $\rightarrow R_a$	X	X	X
27 2 a m	5E a m	DK a,y,m	Divide (Constant)	$(R_a, R_{a+1}) / Y \rightarrow R_{a+1}$; remainder $\rightarrow R_a$ ③	X	X	X
27 3 a m	5F a m	D a,y,m	Divide	$(R_a, R_{a+1}) / (Y) \rightarrow R_{a+1}$; remainder $\rightarrow R_a$ ③	X	X	X
30 0 a m	60 a m	ANDR a,m	AND (Register)	$(R_a) \wedge (R_m) \rightarrow R_a$	0	0	X
30 1 a m	61 a m	ANDI a,m	AND (Indirect)	$(R_a) \wedge (Y^*) \rightarrow R_a$	0	0	X
30 2 a m	62 a m	ANDK a,y,m	AND (Constant)	$(R_a) \wedge Y \rightarrow R_a$	0	0	X
30 3 a m	63 a m	AND a,y,m	AND	$(R_a) \wedge (Y) \rightarrow R_a$	0	0	X
31 0 a m	64 a m	ORR a,m	OR (Register)	$(R_a) \vee (R_m) \rightarrow R_a$	0	0	X
31 1 a m	65 a m	ORI a,m	OR (Indirect)	$(R_a) \vee (Y^*) \rightarrow R_a$	0	0	X
31 2 a m	66 a m	ORK a,y,m	OR (Constant)	$(R_a) \vee Y \rightarrow R_a$	0	0	X
31 3 a m	67 a m	OR a,y,m	OR	$(R_a) \vee (Y) \rightarrow R_a$	0	0	X
32 0 a m	68 a m	XORR a,m	Exclusive OR (Register)	$(R_a) \forall (R_m) \rightarrow R_a$	0	0	X
32 1 a m	69 a m	XORI a,m	Exclusive OR (Indirect)	$(R_a) \forall (Y^*) \rightarrow R_a$	0	0	X
32 2 a m	6A a m	XORK a,y,m	Exclusive OR (Constant)	$(R_a) \forall Y \rightarrow R_a$	0	0	X
32 3 a m	6B a m	XOR a,y,m	Exclusive OR	$(R_a) \forall (Y) \rightarrow R_a$	0	0	X
33 0 a m	6C a m	MSR a,m	Masked Substitute (Register)	If $(R_{a+1})_n = 1$; $(R_m)_n \rightarrow R_{a_n}$ ③	0	0	X
33 1 a m	6D a m	MSI a,m	Masked Substitute (Indirect)	If $(R_{a+1})_n = 1$; $(Y^*)_n \rightarrow R_{a_n}$ ③	0	0	X
33 2 a m	6E a m	MSK a,y,m	Masked Substitute (Constant)	If $(R_{a+1})_n = 1$; $Y_n \rightarrow R_{a_n}$ ③	0	0	X
33 3 a m	6F a m	MS a,y,m	Masked Substitute	If $(R_{a+1})_n = 1$; $(Y)_n \rightarrow R_{a_n}$ ③	0	0	X
34 0 a m	70 a m	CMR a,m	Compare Masked (Register)	$[(R_a) \wedge (R_{a+1})]$: $[(R_m) \wedge (R_{a+1})]$ ③	0	0	X

② if a ≠ m ③ a,m,y must be even

OCTAL FORMAT o f a m	HEXIDECIMAL FORMAT OP a m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
34 1 a m	71 a m	CMI a,m	Compare Masked (Indirect)	$[(R_a) \wedge (R_{a+1})]$: $[(Y^*) \wedge (R_{a+1})]$ ③	0	0	X
34 2 a m	72 a m	CMK a,y,m	Compare Masked (Constant)	$[(R_a) \wedge (R_{a+1})]$: $[Y \wedge (R_{a+1})]$ ③	0	0	X
34 3 a m	73 a m	CM a,y,m	Compare Masked	$[(R_a) \wedge (R_{a+1})]$: $[(Y) \wedge (R_{a+1})]$ ③	0	0	X
35 0 00 00	74 0 0	IOCR	Input/Output Command	Execute (0140); $0 \rightarrow 0140_{15,14}$	– NC –		
35 1 00 m	75 0 m	BFI m	Biased Fetch (Indirect)	$(Y^*) \rightarrow CC$; $1 \rightarrow Y^*_{15,14}$	0	0	X
35 2 00 m	76 0 m	REX y,m	Remote Execute	Execute (Y); (P) + 2 → P	– NA –		
35 3 00 m	77 0 m	BF y,m	Biased Fetch	$(Y) \rightarrow CC$; $1 \rightarrow Y_{15,14}$	0	0	X
#37 0 a m	7C a 0	See page 6	Trig & Hyperbolic		– NC –		
#37 0 a 010	7C a 10	FC a,y	Floating Point Compare	(Ra, Ra+1): (Y, Y+1) ⑦ ⑧	0	0	X
#37 0 a 011	7C a 11	FXC a	Fixed to Floating Point Conversion	Form normalized Floating Point number in Ra, Ra+1, from the binary exponent in R_a and integer mantissa in R_{a+1} (2's complement)	X	X	X
#37 0 a 012	7C a 12	FLC a	Floating Point to Fixed Conversion	Unpack Floating Point number in R_a, R_{a+1} into binary exponent in R_a and integer mantissa into R_{a+1}	0	0	X
#37 0 a 013	7C a 13	NF a	Floating Point Normalize	Normalize the Floating Point number in R_a and R_{a+1}.	X	X	X
#37 0 a 016	7C a 16	QAL a,y	Algebraic Left Quadruple Shift	Shift $(R_a, R_{a+1}, R_{a+2}, R_{a+3})$ Left Y_{5-0} places, zero fill ⑦	0	X	X
#37 0 a 017	7C a 17	QAR a,y	Algebraic Right Quadruple Shift	Shift $(R_a, R_{a+1}, R_{a+2}, R_{a+3})$ Right Y_{5-0} places, sign fill ⑦	0	0	X
40 0 00 m	80 0 m	JER m	Jump Equal	If CC indicates = or 0; $(R_m) \rightarrow P$	– NC –		
40 0 01 m	80 1 m	JNER m	Jump Not Equal	If CC indicates ≠ or not 0; $(R_m) \rightarrow P$	– NC –		
40 0 02 m	80 2 m	JGER m	Jump Greater or Equal	If CC indicates ≥ or +; $(R_m) \rightarrow P$	– NC –		
40 0 03 m	80 3 m	JLSR m	Jump Less	If CC indicates < or –; $(R_m) \rightarrow P$	– NC –		
40 0 04 m	80 4 m	JOR m	Jump Overflow	If overflow set: $(R_m) \rightarrow P$	– NC –		
40 0 05 m	80 5 m	JCR m	Jump Carry	If carry set: $(R_m) \rightarrow P$	– NC –		
40 0 06 m	80 6 m	JPTR m	Jump Power out of Tolerance	If power out of tolerance: $(R_m) \rightarrow P$	– NC –		
40 0 07 m	80 7 m	JBR m	Jump Bootstrap 2 selected	If bootstrap 2 selected: $(R_m) \rightarrow P$	– NC –		
40 0 10 m	80 8 m	JR m	Jump	$(R_m) \rightarrow P$	– NC –		
40 0 11 m	80 9 m	JSR m	Jump after Stop	Stop; $(R_m) \rightarrow P$	– NC –		
40 0 12 m	80 A m	JKSR 1,m	Jump. If Key set–Stop, then jump (Register)	If key 1 set , stop; $(R_m) \rightarrow P$	– NC –		
40 0 13 m	80 B m	JKSR 2,m	Jump. If Key set–Stop, then jump (Register)	If key 2 set , stop; $(R_m) \rightarrow P$	– NC –		
40 1 d	81 d	LJ xD	Local Jump	(P) + D → P	– NC –		
40 2 00 m	82 0 m	JE y,m	Jump Equal	If CC indicates = or 0; Y → P	– NC –		
40 2 01 m	82 1 m	JNE y,m	Jump Not Equal	If CC indicates ≠ or not 0; Y → P	– NC –		
40 2 02 m	82 2 m	JGE y,m	Jump Greater than or Equal	If CC indicates ≥ or +; Y → P	– NC –		
40 2 03 m	82 3 m	JLS y,m	Jump Less	If CC indicates < or –; Y → P	– NC –		
40 2 04 m	82 4 m	JO y,m	Jump on Overflow	If overflow set: Y → P	– NC –		
40 2 05 m	82 5 m	JC y,m	Jump on Carry	If carry set, Y → P	– NC –		
40 2 06 m	82 6 m	JPT y,m	Jump if Power out of Tolerance	If power out of tolerance; Y → P	– NC –		
40 2 07 m	82 7 m	JB y,m	Jump if Bootstrap 2 selected	If bootstrap 2 selected; Y → P	– NC –		
40 2 10 m	82 8 m	J y,m	Jump	Y → P	– NC –		
40 2 11 m	82 9 m	JS y,m	Jump after Stop	Stop; Y → P	– NC –		
40 2 12 m	82 A m	JKS 1,y,m	Jump. If Key set–Stop, then jump	If key 1 set , stop; Y → P	– NC –		
40 2 13 m	82 B m	JKS 2,y,m	Jump. If Key set–Stop, then jump	If key 2 set , stop; Y → P	– NC –		
40 3 00 m	83 0 m	JE *y,m	Jump Equal	If CC indicates = or 0; (Y) → P	– NC –		
40 3 01 m	83 1 m	JNE *y,m	Jump Not Equal	If CC indicates ≠ or not 0; (Y) → P	– NC –		
40 3 02 m	83 2 m	JGE *y,m	Jump Greater or Equal	If CC indicates ≥ or +; (Y) → P	– NC –		
40 3 03 m	83 3 m	JLS *y,m	Jump Less	If CC indicates < or –; (Y) → P	– NC –		
40 3 04 m	83 4 m	JO *y,m	Jump on Overflow	If overflow set; (Y) → P	– NC –		
40 3 05 m	83 5 m	JC *y,m	Jump on Carry	If carry set; (Y) → P	– NC –		
40 3 06 m	83 6 m	JPT *y,m	Jump if Power out of Tolerance	If power out of tolerance; (Y) → P	– NC –		
40 3 07 m	83 7 m	JB *y,m	Jump if Bootstrap 2 selected	If bootstrap 2 selected; (Y) → P	– NC –		
40 3 10 m	83 8 m	J *y,m	Jump	(Y) → P	– NC –		
40 3 11 m	83 9 m	JS *y,m	Jump After Stop	Stop; (Y) → P	– NC –		
40 3 12 m	83 A m	JKS 1,*y,m	Jump. If Key set–Stop, then jump	If key 1 set, stop; (Y) → P	– NC –		
40 3 13 m	83 B m	JKS 2,*y,m	Jump. If Key set–Stop, then jump	If key 2 set, stop; (Y) → P	– NC –		
41 0 a m	84 a m	XJR a,m	Index Jump Register	If $(R_a) \neq 0$; $(R_a) - 1 \rightarrow R_a$, $(R_m) \rightarrow P$	– NC –		
41 1 d	85 d	LJI xD	Local Jump (Indirect)	[(P) + D] → P	– NC –		
41 2 a m	86 a m	XJ a,y,m	Index Jump	If $(R_a) \neq 0$; $(R_a) - 1 \rightarrow R_a$; Y → P	– NC –		
41 3 a m	87 a m	XJ a,*y,m	Index Jump	If $(R_a) \neq 0$; $(R_a) - 1 \rightarrow R_a$; (Y) → P	– NC –		

Optional Math Pac Instruction

③ a,m,y must be even ⑦ cannot be executed via execute remote ⑧ operands must be normalized

OCTAL FORMAT o	f	a	m	HEXIDECIMAL FORMAT OP	a	m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
42	0	a	m	88	a	m	JLRR a,m	Jump, Link Register (Register)	(P) + 1 → R_a; (R_m) → P	–	NC	–
42	2	a	m	8A	a	m	JLR a,y,m	Jump, Link Register	(P) + 2 → R_a; Y → P	–	NC	–
42	3	a	m	8B	a	m	JLR a,*y,m	Jump, Link Register	(P) + 2 → R_a; (Y) → P	–	NC	–
43	1		d	8D		d	LJLM xD	Local Jump, Link Memory	(P) + 1 → (P) + D; (P) + D + 1 → P	–	NC	–
43	2	00	m	8E	0	m	JLM y,m	Jump, Link Memory	(P) + 2 → Y; Y + 1 → P	–	NC	–
43	3	00	m	8F	0	m	JLM *y,m	Jump, Link Memory	(P) + 2 → (Y); (Y) + 1 → P	–	NC	–
44	0	a	m	90	a	m	JZR a,m	Jump Zero (Register)	If (R_a) = 0; (R_m) → P	–	NC	–
44	1		d	91		d	LJE xD	Local Jump Equal	If CC indicates = or 0; (P) + D → P	–	NC	–
44	2	a	m	92	a	m	JZ a,y,m	Jump Zero	If (R_a) = 0; Y → P	–	NC	–
44	3	a	m	93	a	m	JZ a,*y,m	Jump Zero	If (R_a) = 0; (Y) → P	–	NC	–
45	0	a	m	94	a	m	JNZR a,m	Jump Not Zero (Register)	If (R_a) ≠ 0; (R_m) → P	–	NC	–
45	1		d	95		d	LJNE xD	Local Jump Not Equal	If CC indicates ≠ or not 0; (P) + D → P	–	NC	–
45	2	a	m	96	a	m	JNZ a,y,m	Jump Not Zero	If (R_a) ≠ 0; Y → P	–	NC	–
45	3	a	m	97	a	m	JNZ a,*y,m	Jump Not Zero	If (R_a) ≠ 0; (Y) → P	–	NC	–
46	0	a	m	98	a	m	JPR a,m	Jump Positive (Register)	If (R_a) ≥ 0; (R_m) → P	–	NC	–
46	1		d	99		d	LJGE xD	Local Jump Greater or Equal	If CC indicates ≥ or +; (P) + D → P	–	NC	–
46	2	a	m	9A	a	m	JP a,y,m	Jump Positive	If (R_a) ≥ 0; Y → P	–	NC	–
46	3	a	m	9B	a	m	JP a,*y,m	Jump Positive	If (R_a) ≥ 0; (Y) → P	–	NC	–
47	0	a	m	9C	a	m	JNR a,m	Jump Negative (Register)	If (R_a) < 0; (R_m) → P	–	NC	–
47	1		d	9D		d	LJLS xD	Local Jump Less	If CC indicates < or – ; (P) + D → P	–	NC	–
47	2	a	m	9E	a	m	JN a,y,m	Jump Negative	If (R_a) < 0; Y → P	–	NC	–
47	3	a	m	9F	a	m	JN a,*y,m	Jump Negative	If (R_a) < 0; (Y) → P	–	NC	–
# 50	0	a	m	A0	a	m	FSUR a,m	Floating point subtract (Register)	(R_a, R_{a+1}) – (R_m, R_{m+1}) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 50	1	a	m	A1	a	m	FSUI a,m	Floating point Subtract (Indirect)	(R_a, R_{a+1}) – (Y*, Y*+1) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 50	3	a	m	A3	a	m	FSU a,y,m	Floating point Subtract	(R_a, R_{a+1}) – (Y, Y+1) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 51	0	a	m	A4	a	m	FAR a,m	Floating point Add (Register)	(R_a, R_{a+1}) + (R_m, R_{m+1}) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 51	1	a	m	A5	a	m	FAI a,m	Floating point Add (Indirect)	(R_a, R_{a+1}) + (Y*, Y*+1) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3} ③	X	X	X
# 51	3	a	m	A7	a	m	FA a,y,m	Floating point Add	(R_a, R_{a+1}) + (Y, Y+1) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 52	0	a	m	A8	a	m	FMR a,m	Floating point Multiply (Register)	(R_a, R_{a+1}) · (R_m, R_{m+1}) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 52	1	a	m	A9	a	m	FMI a,m	Floating point Multiply (Indirect)	(R_a, R_{a+1}) · (Y*, Y*+1) → R_a, R_{a+1}; Res. → R_{a+2}, R_{a+3}	X	X	X
# 52	3	a	m	AB	a	m	FM a,y,m	Floating point Multiply	(R_a, R_{a+1}) · (Y, Y+1) → R_a, R_{a+1}, Res. → R_{a+2}, R_{a+3}	X	X	X
# 53	0	a	m	AC	a	m	FDR a,m	Floating point Divide (Register)	(R_a, R_{a+1}) / (R_m, R_{m+1}) → R_a, R_{a+1}; Rem. → R_{a+2}, R_{a+3}	X	X	X
# 53	1	a	m	AD	a	m	FDI a,m	Floating point Divide (Indirect)	(R_a, R_{a+1}) / (Y*, Y*+1) → R_a, R_{a+1}; Rem. → R_{a+2}, R_{a+3}	X	X	X
# 53	3	a	m	AF	a	m	FD a,y,m	Floating point Divide	(R_a, R_{a+1}) / (Y, Y+1) → R_a, R_{a+1}; Rem. → R_{a+2}, R_{a+3}	X	X	X
54	0	a	m	B0	a	m	LARR a,m	Load Address Register (Register)	(R_m) → AR_r SEE LEGEND	–	NC	–
54	1	a	m	B1	a	m	LARI a,m	Load Address Register (Indirect)	(Y*) → AR_r	–	NC	–
54	3	a	m	B3	a	m	LARM a,y,m	Load Address Register Multiple	(Y, ... Y + u) → AR_r . . . . AR_{r+u}	–	NC	–
55	0	a	m	B4	a	m	SARR a,m	Store Address Register (Register)	(AR_r) → R_m	–	NC	–
55	1	a	m	B5	a	m	SARI a,m	Store Address Register (Indirect)	(AR_r) → Y*	–	NC	–
55	3	a	m	B7	a	m	SARM a,y,m	Store Address Register Multiple	(AR_r, . . . AR_{r+u}) → Y, . . . Y + u	–	NC	–
# 56	0	a	m	B8	a	m	MDR a,m	Multiply Double (Register)	(R_a, R_{a+1}) · (R_m, R_{m+1}) → R_a, R_{a+1}, R_{a+2}, R_{a+3} ③	0	0	X
# 56	1	a	m	B9	a	m	MDI a,m	Multiply Double (Indirect)	(R_a, R_{a+1}) · (Y*, Y*+1) → R_a, R_{a+1}, R_{a+2}, R_{a+3} ③	0	0	X
# 56	3	a	m	BB	a	m	MD a,y,m	Multiply Double	(R_a, R_{a+1}) · (Y, Y+1) → R_a, R_{a+1}, R_{a+2}, R_{a+3} ③	0	0	X
# 57	0	a	m	BC	a	m	DDR a,m	Divide Double (Register)	(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (R_m, R_{m+1}) → R_{a+2}, R_{a+3}; Rem. → R_a, R_{a+1} ③	0	X	X
# 57	1	a	m	BD	a	m	DDI a,m	Divide Double (Indirect)	(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (Y*, Y*+1) → R_{a+2}, R_{a+3}; Rem. → R_a, R_{a+1} ③	0	X	X
# 57	3	a	m	BF	a	m	DD a,y,m	Divide Double	(R_a, R_{a+1}, R_{a+2}, R_{a+3}) / (Y, Y+1) → R_{a+2}, R_{a+3}; Rem. → R_{a+1} ③	0	X	X
60	0	a	m	C0	a	m	LLRS a,m	Literal Logical Right Shift	Shift (R_a) right m places, zero fill	0	0	X
60	1	a	m	C1	a	m	LARS a,m	Literal Algebraic Right Shift	Shift (R_a) right m places, sign fill	0	0	X
60	2	a	m	C2	a	m	LLRD a,m	Literal Logical Right Double shift	Shift (R_a, R_{a+1}) right m places, zero fill ③	0	0	X

Optional Math Pac Instructions ③ a,m,y must be even

OCTAL FORMAT o	f	a	m	HEXIDECIMAL FORMAT OP	a	m	CODING FORMAT	INSTRUCTION	OPERATION	C	OV	CC
60	3	a	m	C3	a	m	LARD a,m	Literal Algebraic Right Double shift	Shift (R_a, R_{a+1}) right m places, sign fill ③	0	0	X
61	0	a	m	C4	a	m	LALS a,m	Literal Algebraic Left Shift	Shift (R_a) left m places, zero fill	0	X	X
61	1	a	m	C5	a	m	LCLS a,m	Literal Circular Left Shift	Shift (R_a) left circular m places	0	0	X
61	2	a	m	C6	a	m	LALD a,m	Literal Algebraic Left Double shift	Shift (R_a, R_{a+1}) left m places, zero fill ③	0	X	X
61	3	a	m	C7	a	m	LCLD a,m	Literal Circular Left Double shift	Shift (R_a, R_{a+1}) left circular m places ③	0	0	X
62	0	a	m	C8	a	m	LSU a,m	Literal Subtract	(R_a)–m → R_a	X	X	X
62	1	a	m	C9	a	m	LSUD a,m	Literal Subtract Double	(R_a, R_{a+1})–m → R_a, R_{a+1} ③	X	X	X
62	2	a	m	CA	a	m	LA a,m	Literal Add	(R_a) + m → R_a	X	X	X
62	3	a	m	CB	a	m	LAD a,m	Literal Add Double	(R_a, R_{a+1}) + m → R_a, R_{a+1} ③	X	X	X
63	0	a	m	CC	a	m	LL a,m	Literal Load	m → R_a	0	0	X
63	1	a	m	CD	a	m	LC a,m	Literal Compare	(R_a) : m	X	X	X
63	2	a	m	CE	a	m	LMUL a,m	Literal Multiply	(R_{a+1}) · m → R_a, R_{a+1} ③	0	0	X
63	3	a	m	CF	a	m	LDIV a,m	Literal Divide	(R_a, R_{a+1}) / m → R_{a+1}; ③ remainder → R_a	0	X	X
64	3	a	m	D3	a	m	BSU a,y,m	Byte Subtract	(R_a) – (Y) byte → R_a	X	X	X
65	3	a	m	D7	a	m	BA a,y,m	Byte Add	(R_a) + (Y) byte → R_a	X	X	X
66	3	a	m	DB	a	m	BC a,y,m	Byte Compare	(R_a) : (Y) byte	X	X	X
67	0	a	m	DC	a	m	UM1 a,m UM2 a,m	User Macro – CP	Reserved for User Macro		–NA–	
67	1	a	m	DD	a	m	UMI a,m	User Macro – CP	Reserved for User Macro		–NA–	
67	2	a	m	DE	a	m	UMK a,y,m	User Macro – CP	Reserved for User Macro		–NA–	
67	3	a	m	DF	a	m	BCX a,y,m	Byte Compare and Index By 1	(R_a) : (Y) byte ; (R_m) + 1 → R_m	X	X	X

COMMAND/CHAIN INSTRUCTION

70	0	00	00	E0	0	0	ACR 0 CCR 0,0	Channel Control	Master clear all channels			
70	0	00	04	E0	0	4	ACR 4 CCR 0,4	Channel Control	Enable external interrupts, all channels			
70	0	00	05	E0	0	5	ACR 5 CCR 0,5	Channel Control	Disable external interrupts, all channels			
70	0	00	06	E0	0	6	ACR 6 CCR 0,6	Channel Control	Enable Class III, Priority 2, 3, 4 interrupts			
70	0	00	07	E0	0	7	ACR 7 CCR 0,7	Channel Control	Disable Class III, Priority 2, 3, 4 interrupts			
70	0	a	10	E0	a	8	CCR a,10	Channel Control	Master clear chan. a			
70	0	a	14	E0	a	C	CCR a,14	Channel Control	Enable chan. a external interrupts			
70	0	a	15	E0	a	D	CCR a,15	Channel Control	Disable chan. a external interrupts			
70	0	a	16	E0	a	E	CCR a,16	Channel Control	Enable chan. a Class III, Priority 2, 3, 4 interrupts			
70	0	a	17	E0	a	F	CCR a,17	Channel Control	Disable chan. a Class III, Priority 2, 3, 4 interrupts			
72	0	a	m					User Macro – I/O	Reserved for User Macro			
72	1	a	m					User Macro – I/O	Reserved for User Macro			

COMMAND INSTRUCTION

71	2	a	02	E6	a	2	ICK a,y	Initiate Input Chain	Y → Channel a Chain Pointer; initiate input chain			
71	2	a	06	E6	a	6	OCK a,y	Initiate Output Chain	Y → Channel a Chain Pointer; initiate output chain			
71	3	a	m	E7	a	m	WIM a,y,m WCM am,y	Write Control Memory	(Y) → Chan. a CM_m (See Figure 6)			
72	3	a	m	EB	a	m	RIM a,y,m RCM am,y	Read Control Memory	Chan. a (CM_m) → Y (See Figure 6)			
76	0	a	m	F8	a	m	SICR a,m	Serial Interface Control	Set or clear chan. a discrete function per Table 3			
76	3	a	m	FB	a	m	SST a,y	Store Serial Status	Channel a Serial Satus bits → Y (See Table 4)			

CHAIN INSTRUCTION

70	3	00	00	E3	0	0	IO 0,y	Input Data	(Y, Y+1) → BCW, BAP; initiate transfer			
70	3	01	00	E3	1	0	IO 1,y	Output Data	(Y, Y+1) → BCW, BAP; initiate transfer			
70	3	02	00	E3	2	0	IO 2,y	External Function	(Y, Y+1) → BCW, BAP; initiate transfer			
70	3	03	00	E3	3	0	IO 3,y	Force External Function	(Y, Y+1) → BCW, BAP; initiate transfer			
71	2	00	m	E6	0	m	LCMK m,y	Load Control Memory	Y → CM_m (See Figure 6) { initiate input chain, m = 2; initiate output chain, m = 6			
71	3	00	m	E7	0	m	LCM m,y	Load Control Memory	(Y) → CM_m (See Figure 6)			
72	3	00	m	EB	0	m	SCM m,y	Store Control Memory	(CM_m) → Y (See Figure 6)			
73	0	00	00	EC	0	0	HCR	Halt Chain	Halt chaining			
73	0	01	00	EC	1	0	IPR	Interrupt Processor	Generate chain interrupt			
73	3	00	00	EF	0	0	ZF y	Zero Flag	0 → $Y_{15,14}$			
73	3	01	00	EF	1	0	SF y	Set Flag	1 → $Y_{15,14}$			
74	2	00	00	F2	0	0	SJMC 0,y	Serial Jump on Met Condition	Unconditional Y → CAP			
74	2	01	00	F2	1	0	SJMC 1,y	Serial Jump on Met Condition	If suppress flag not set, Y → CAP			
74	2	02	00	F2	2	0	SJMC 2,y	Serial Jump on Met Condition	If monitor flag set, Y → CAP			
75	0	00	m	F4	0	m	SFSC m	Search For Sync	Perform function(s) assigned to m-bits per Figure 7			
76	0	00	m	F8	0	m	CSIR m	Serial Interface Control	Set or clear discrete function per Table 3			
76	3	00	m	FB	0	m	CSST y	Store Serial Status	Serial Status bits → Y; See Table 4			

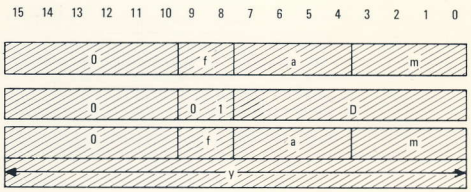
③ a,m,y must be even

TRIGONOMETRIC AND HYPERBOLIC FUNCTIONS
(Operation Code 37)

x&y Cartesian coordinates. Radix point assumed to be the same
θ Angle of rotation Trignonometric mode (BAMS) Bit 15 = 180°
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	$X = \frac{R}{K}\sqrt{x^2+y^2}$	$W = \theta = \tan^{-1}\frac{y}{x}$
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	$X = \sqrt{x^2 - y^2}$ 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	0	0.46672_8	θ	$Y = \sin\theta$	$X = \cos\theta$	0
37 0 a 03	RFP a	Sin θ; Cos θ	0	1	θ	$Y = \sin\theta$	$X = \cos\theta$	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 06	VHP a	$\text{Log}_e x$	x−1	x+1	0	0	$2\sqrt{x}$	$W = 1/2\ \log_e x$ $= \tanh^{-1}\frac{x-1}{x+1}$
37 0 a 07	RHP a	Exponential	1	1	v	$Y = e^v = \cosh v + \sinh v$	$X = e^v = \cosh v + \sinh v$	0

Optional Math Pac Instructions



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
- D Signed Deviation Value (Two's Complement)
- y Address or Arithmetic Constant

Figure 1. Instruction Word Format

LEGEND

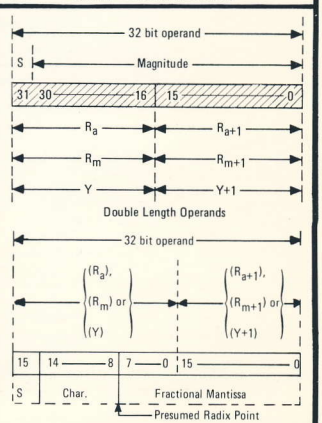
- B Byte pointer, 0 → Upper, 1 → Lower
- C Carry
- CC Condition Code
- OV Overflow
- IW Indirect Word
- j Designator Field in IW
- x General Register Designator in IW1
- y Contents of Second Instruction Word or IW2
- Y Effective Operand Address or Constant
- Y* Effective Operand Address in Rm
- TM I/O Transfer Mode
  - 00 → Abort Input 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 $(R_a)_{13\text{-}8}$
- ' 2's Complement
- : Compare

OR		
V	0	1
0	0	1
1	1	1

XOR		
⊻	0	1
0	0	1
1	1	0

AND		
Λ	0	1
0	0	0
1	0	1

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)



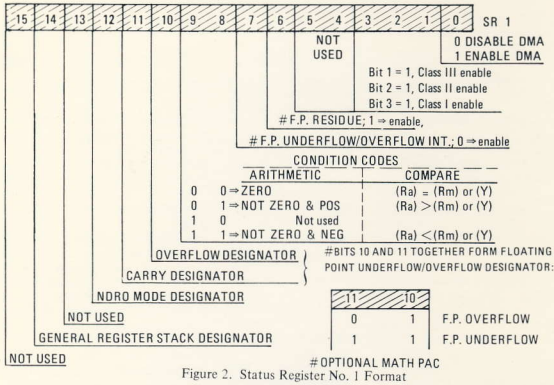


Figure 2. Status Register No. 1 Format

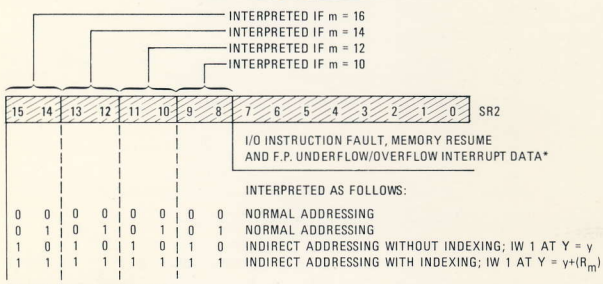
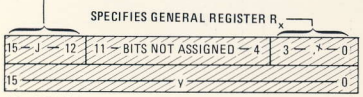


Figure 3. Status Register No. 2 Format

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

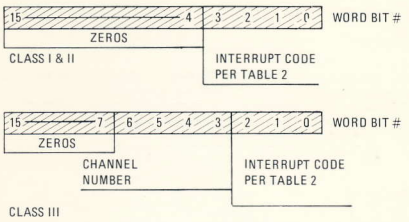


Figure 5. Interrupt Entrance Address Index

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	0000
	2	Memory Resume	0010
Class II, Software Interrupts	1	CP Instruction Fault	0000
	2	I/O Instruction Fault	0010
	3	#F.P. Overflow/Underflow Interrupt	0100
	4	Executive Return Instruction	0110
	5	RTC Overflow	1000
	6	Monitor Clock	1010
Class III, IOC Interrupts	1	Intercomputer Time-Out	110
	2	External Interrupt or Discrete Interrupt ①	000
	3	Output Chain Interrupt	100
	4	Input Chain Interrupt	010

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

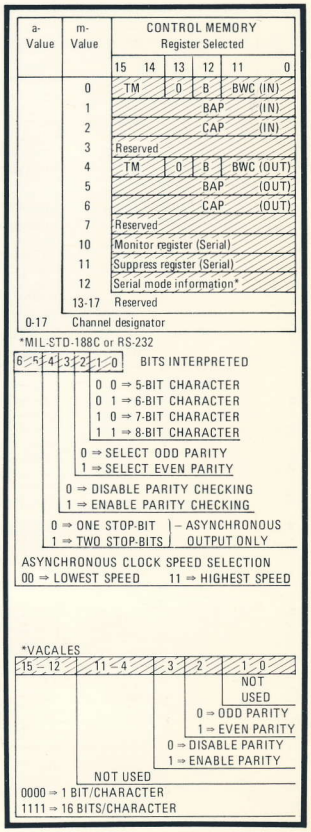


Figure 6. I/O Control Memory

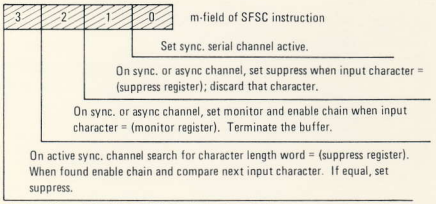


Figure 7. SFSC Operations

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	1 ⇒ I DISCRETE TURNED ON	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	NoOp	Not used	–	–	Spare	–
3	NoOp	Not used	–	–	Spare	–
4	Set	Control Line 6	J	J	J (non-std.)	–
5	Clear	Control Line 6	J	J	J (non-std.)	–
6	Set	Control Line 5	H	TRAN. PREP	Disable Ring Indicator Interrupt (internal)	–
7	Clear	Control Line 5	H	TRAN. PREP	Enable Ring Indicator Interrupt (internal)	–
10	Clear	Control Line 4	G	G	Request to Send	CA
11	Set	Control Line 4	G	G	Request to Send	CA
12	Clear	Control Line 3	F	F	New Sync	CH
13	Set	Control Line 3	F	F	New Sync	CH
14	Clear	Control Line 2	D	D	Data Terminal Ready	CD
15	Set	Control Line 2	D	D	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