Eldon W

UNISYS

AN/UYK - 44

CPU REPERTOIRE

OCTAL FORMAT				HEXADECIMAL FORMAT			CODING FORMAT			SR1 BITS		
										11	10	9–8
o	f	a	m	OP	a	m		INSTRUCTION	OPERATION	C	OV	CC
00	0		01	00		1	ICP a	† Initiate CP Bit	Execute the CP Bit Subtest specified by $(R_a)_{7-0}$	–	–	–
00	0	0	02	00	0	2	SRM	† RAM BIT Sig to 78–7D	BIT Signature to CP Control Memory 78 thru 7D	–	–	–
00	0	0	03	00	0	3	LRM	† 78–7D to RAM BIT Sig	CP Control Memory 78 thru 7D to BIT Signature	–	–	–
00	0	0	04	00	0	4	IMP	† Initiate MP BIT	Execute maintenance panel subtest	–	–	–
00	0	0	05	00	0	5	IDS	† Initiate/Update Deadstick	Activate/reset the deadstick timer	–	–	–
00	0	0	06	00	a	6	RSCS a	† Read Semi Conductor Memory Status Register	SCM SR → R_a	–	–	–
00	0	0	07	00	0	7	EEC	† Enable Error Correct Logic	1 → SCM SR bit 2^4	–	–	–
00	0	0	10	00	0	8	DEC	† Disable Error Correct Logic	0 → SCM SR bit 2^4	–	–	–
00	0	0	11	00	0	9	SEL a	† Search Error Log		–	–	–
00	0	0	m	00	a	m	–	With m=0 A–F Illegal	Causes CP Instruction Fault Interrupt when executed	–	–	–
00	2	a	m	02	a	m	SPT a,y,m	Stack Put Top	(Y)→(R_a); (R_a)−1→Y	–	–	–
00	3	a	m	03	a	m	BL a,y,m	Byte Load	$(Y)_{byte}$→ R_a	0	0	X
01	0	a	m	04	a	m	LR a,m	Load (Register)	(R_m)→R_a	0	0	X
01	1	a	m	05	a	m	LI a,m	Load (Indirect)	(Y*)→R_a	0	0	X
01	2	a	m	06	a	m	LK a,y,m	Load (Constant)	Y→R_a	0	0	X
01	3	a	m	07	a	m	L a,y,m	Load	(Y)→R_a	0	0	X
02	0	a	00	08	a	0	PR a	Make Positive (Register)	If $(R_a)<0$,0−(R_a)→R_a If $(R_a)\geq 0$,(R_a) unchanged	X 0	X 0	X X
02	0	a	01	08	a	1	NR a	Make Negative (Register)	If $(R_a)\geq 0$,0−(R_a)→R_a If $(R_a)<0$,(R_a) unchanged	X 0	0 0	X X
02	0	a	02	08	a	2	RR a	Round (Register)	(R_a)+(R_{a+1})$_{15}$→R_a	X	X	X
02	0	a	04	08	a	4	TCR a	Two's Complement	0 − (R_a)→R_a	X	X	X
02	0	a	05	08	a	5	TCDR a	Two's Complement Double (Register)	0 − (R_a, R_{a+1}) → R_a,R_{a+1}	X	X	X
02	0	a	06	08	a	6	OCR a	One's Complement	FFFF−(R_a)→R_a	0	0	X
02	0	a	10	08	a	8	IROR a	Increase by 1 (Register)	(R_a)+1→R_a	X	X	X
02	0	a	11	08	a	9	DROR a	Decrease by 1 (Register)	(R_a)−1→R_a	X	X	X
02	0	a	12	08	a	A	IRTR a	Increase by 2 (Register)	(R_a)+2→R_a	X	X	X
02	0	a	13	08	a	B	DRTR a	Decrease by 2 (Register)	(R_a)−2→R_a	X	X	X
02	1	a	m	09	a	m	LDI a,m	Load Double (Indirect)	(Y*,Y*+1) → R_a,R_{a+1}	0	0	X
02	3	a	m	0B	a	m	LD a,y,m	Load Double (Index)	(Y,Y+1) → R_a,R_{a+1}	0	0	X
03	0	a	00	0C	a	0	ER a	Executive Return (Register)	If Class II interrupts enabled, (P)+1→R_a	0	0	X
03	0	a	01	0C	a	1	SSOR a	Store Status Register 1 (Register)	(SR1)→R_a	0	0	X
03	0	a	02	0C	a	2	SSTR a	Store Status Register 2 (Register)	(SR2)→R_a	0	0	X
03	0	a	03	0C	a	3	SCR a §	Store Real Time Clock Lower (Register)	$(RTC)_{15-0}$→R_a	0	0	X
03	0	a	04	0C	a	4	LPR a	Load P Register	(R_a)→P	–	–	–
03	0	a	05	0C	a	5	LSOR a	† Load Status Register 1 (Register)	(R_a)→SR1	–	–	–
03	0	a	06	0C	a	6	LSTR a	† Load Status Register 2 (Register)	(R_a)→SR2	–	–	–
03	0	a	07	0C	a	7	LCR a	§† Load Real Time Clock Lower (Register)	(R_a)→RTC_{15-0}	–	–	–
03	0	00	10	0C	0	8	ECR	§† Enable Real Time Clock Count and Interrupt	Enable RTC Count and Overflow Interrupt	–	–	–
03	0	00	11	0C	0	9	DCR	§† Disable Real Time Clock Count and Interrupt	Disable RTC Count and Overflow Interrupt	–	–	–
03	0	a	12	0C	a	A	LEM a	§† Load and Enable Monitor Clock and Interrupt	(R_a) → MC Register; Enable Count and Interrupt	–	–	–
03	0	00	13	0C	0	B	DM §	† Disable Monitor Clock Count	Disable MC Count and Interrupt	–	–	–
03	0	a	14	0C	a	C	LCRD a	§† Load Real Time Clock Double and Enable Count (Register)	(R_a,R_{a+1}) → RTC and Enable Count	–	–	–
03	0	a	15	0C	a	D	SCRD a	§† Store Real Time Clock Double (Register)	(RTC) → R_a,R_{a+1}	0	0	X
03	0	00	16	0C	0	E	ECIR	§† Enable Real Time Clock Overflow Interrupt	Enable RTC Overflow Interrupt	–	–	–
03	0	00	17	0C	0	F	DCIR	§† Disable Real Time Clock Overflow Interrupt	Disable RTC Overflow Interrupt	–	–	–
03	3	a	m	0F	a	m	LM a,y,m	Load Multiple	If m ≥ a; (Y...Y+m−a) →R_a...R_m	–	–	–
									If m < a; (Y...Y+m−a+16) →R_a...R_m	–	–	–
04	0	a	00	10	a	0	SQR a	# Square Root	$\sqrt{(R_a,R_{a+1})}$→R_a+1;Res.→R_a	0	X	X
04	0	a	01	10	a	1	RVR a	Reverse Register (Register)	Reverse order of bits in R_a	0	0	X
04	0	a	02	10	a	2	CNT a	Count Ones (Register)	Number of binary ones in R_a→R_{a+1}	–	–	–
04	0	a	03	10	a	3	SFR a	Scale Factor (Register)	Shift (R_a,R_{a+1}) left until $(R_a)_{15} \neq (R_a)_{14}$, zero fill; shift count → R_{a+1} [1]	–	–	–
04	0	a	04	10	a	4	SMC a	§ Store Monitor Clock	Monitor Clock → R_a	–	–	–
04	0	a	05	10	a	5	SQRT a	# Floating Point Square Root	$\sqrt{(R_a,R_{a+1})}$ → R_a,R_{a+1}	0	X	X
04	0	a	06	10	a	6	LCEP a	§† Load Clock Enable Periodic	(R_a) → RTC_{15-0} and enable interrupt; upon interrupt, (R_{a+1}) → RTC_{15-0}	–	–	–
04	0	a	10	10	a	8	IS	† Initialize System		0	0	0
04	0	a	11	10	a	9	IB	† Initialize Bus		–	–	–
04	2	a	m	12	a	m	QPT a,y,m	Queue Put Top	(Y)−(R_a),(R_a)→Y,if (Y)=0 then (R_a)→Y+1	–	–	–
04	3	a	m	13	a	m	BLX a,y,m	Byte Load and Index by 1	$(Y)_{byte}$→$R_{a\ 7-0}$, 0→$R_{a\ 15-8}$ (Rm)+1→R_m	0	0	X
05	0	a	m	14	a	m	SBR a,m	Set Bit (Register)	1→$(R_a)_m$	0	0	X
05	1	a	m	15	a	m	LXI a,m	Load and Index by 1 (Indirect)	(Y*)→R_a;(R_m)+1→R_m if a ≠ m	0	0	X
05	2	a	m	16	a	m	QPB a,y,m	Queue Put Bottom	(R_a)→(Y+1),(R_a)→Y+1, 0→(R_a)	–	–	–
05	3	a	m	17	a	m	LX a,y,m	Load and Index by 1 (Index)	(Y)→R_a;(R_m)+1→R_m	0	0	X

(1) Count=31 for all zeros or all ones
† *Privileged Instructions*
# *Math Pac Instructions*
§ *RTC or External clock required*

PX 14280B
NOVEMBER 1987

OCTAL FORMAT o	f	a	m	HEXADECIMAL FORMAT OP	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
06	0	a	m	18	a	m	ZBR a,m	0 → (R_a)$_m$	0	0	X
06	1	a	m	19	a	m	LDXI a,m	(Y*,Y*+1)→R_a,R_{a+1}; (R_m)+2→R_m	0	0	X
06	2	a	m	1A	a	m	SGT a,y,m	(Y)→R_a; if (Y) ≠ 0 then ((Y)) → Y and (P)+3→P; if (Y)=0, then (P)+2→P			
06	3	a	m	1B	a	m	LDX a,y,m	(Y,Y+1) → R_a,R_{a+1}; (R_m)+2→R_m	0	0	X
07	0	a	m	1C	a	m	CBR a,m	Compare Bit to Zero (Register)	(R_a)m:0	0	0	X
07	1	00	m	1D	0	m	† LPI m	Load Program Status Words (Indirect)	(Y*,Y*+1,Y*+2)→P, SR1, SR2	–	–	–
07	2	a	m	1E	a	m	QGT a,y,m	Queue Get Top	(Y)→R_a; if (Y) = 0, then (P)+2→P; if (Y) ≠ 0, then (P)+3→P, ((Y))→Y, and if ((Y))=0, then Y→Y+1	–	–	–
07	3	00	m	1F	0	m	LP y,m	† Load Program Status Words (Index)	(Y, Y+1, Y+2)→P, SR1, SR2	–	–	–
10	0	a	m	20	a	m	LRSR a,m	Logical Right Single 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 Single Shift (Constant)	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 (Index)	(R_a)$_{7-0}$→Y_{byte}	–	–	–
11	0	a	m	24	a	m	ARSR a,m	Algebraic Right Single 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)→Y*	–	–	–
11	2	a	m	26	a	m	ARS a,y,m	Algebraic Right Single Shift (Constant)	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 (Index)	(R_a)→Y	–	–	–
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	0	0	X
12	1	a	m	29	a	m	SDI a,m	Store Double (Indirect)	(R_a,R_{a+1})→Y*,Y*+1	–	–	–
12	2	a	m	2A	a	m	LRD a,y,m	Logical Right Double Shift (Constant)	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 (Index)	(R_a,R_{a+1})→Y, Y+1	–	–	–
13	0	a	m	2C	a	m	ARDR a,m	Algebraic Right Double Shift (Register)	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 (Constant)	Shift (R_a, R_{a+1}) right Y_{5-0} places, sign fill	0	0	X
13	3	a	m	2F	a	m	SM a,y,m	Store Multiple	If m ≥ a; (R_a...R_m) → Y...Y+m–a; If m < a; (R_a...R_m) → Y...Y+a+16	–	–	–
14	0	a	m	30	a	m	ALSR a,m	Algebraic Left Single 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 Single Shift (Constant)	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 and Index by 1 (Index)	(R_a)$_{7-0}$→Y_{byte}; (R_m)+1 → R_m	–	–	–
15	0	a	m	34	a	m	CLSR a,m	Circular Left Single Shift (Register)	Shift (R_a) left circularly (R_m)$_{5-0}$ places	0	0	X
15	1	a	m	35	a	m	SXI a,m	Store and Index by 1 (Indirect)	(R_a)→Y*; (R_m) + 1 → R_m	–	–	–
15	2	a	m	36	a	m	CLS a,y,m	Circular Left Single Shift (Constant)	Shift (R_a) left circularly Y_{5-0} places	0	0	X
15	3	a	m	37	a	m	SX a,y,m	Store and Index by 1 (Index)	(R_a) → Y; (R_m) + 1 → R_m	–	–	–
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	0	X	X
16	1	a	m	39	a	m	SDXI a,m	Store Double and Index by 2 (Indirect)	(R_a, R_{a+1}) → Y*, Y*+1; (R_m) + 2 → R_m	–	–	–
16	2	a	m	3A	a	m	ALD a,y,m	Algebraic Left Double Shift (Constant)	Shift (R_a, R_{a+1}) left Y_{5-0} places, zero fill	0	X	X
16	3	a	m	3B	a	m	SDX a,y,m	Store Double and Index by 2 (Index)	(R_a,R_{a+1}) → Y,Y+1; (R_m)+2 → R_m	–	–	–
17	0	a	m	3C	a	m	CLDR a,m	Circular Left Double Shift (Register)	Shift (R_a, R_{a+1}) left circularly (R_m)$_{5-0}$ places	0	0	X
17	1	00	m	3D	0	m	SZI m	Store Zero (Indirect)	0 → Y*	–	–	–
17	2	a	m	3E	a	m	CLD a,y,m	Circular Left Double Shift (Constant)	Shift (R_a,R_{a+1}) left circularly Y_{5-0} places	0	0	X
17	3	00	m	3F	0	m	SZ y,m	Store Zeros (Index)	0 → Y	–	–	–
20	0	a	m	40	a	m	SUR a,m	Subtract (Register)	(R_a) – (R_m) → R_a	X	X	X
20	1	a	m	41	a	m	SUI a,m	Subtract (Indirect)	(R_a) – (Y*) → R_a	X	X	X
20	2	a	m	42	a	m	SUK a,y,m	Subtract (Constant)	(R_a) – Y → R_a	X	X	X
20	3	a	m	43	a	m	SU a,y,m	Subtract (Index)	(R_a) – (Y) → 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}) → 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) → R_a,R_{a+1}	X	X	X
21	3	a	m	47	a	m	SUD a,y,m	Subtract Double (Index)	(R_a,R_{a+1}) – (Y,Y+1) → R_a,R_{a+1}	X	X	X
22	0	a	m	48	a	m	AR a,m	Add (Register)	(R_a) + (R_m) → R_a	X	X	X
22	1	a	m	49	a	m	AI a,m	Add (Indirect)	(R_a)+(Y*) → R_a	X	X	X
22	2	a	m	4A	a	m	AK a,y,m	Add (Constant)	(R_a)+Y→R_a	X	X	X
22	3	a	m	4B	a	m	A a,y,m	Add (Index)	(R_a)+(Y)→ 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}) → 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) → R_a,R_{a+1}	X	X	X
23	3	a	m	4F	a	m	AD a,y,m	Add Double (Index)	(R_a,R_{a+1}) + (Y,Y+1) → R_a,R_{a+1}	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 (Index)	(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})	X	X	X
25	1	a	m	55	a	m	CDI a,m	Compare Double (Indirect)	(R_a,R_{a+1}) : (Y*,Y*+1)	X	X	X
25	3	a	m	57	a	m	CD a,y,m	Compare Double (Index)	(R_a,R_{a+1}) : (Y,Y+1)	X	X	X
26	0	a	m	58	a	m	MR a,m	Multiply (Register)	(R_{a+1})*(R_m) → R_a,R_{a+1}	0	0	X
26	1	a	m	59	a	m	MI a,m	Multiply (Indirect)	(R_{a+1})*(Y*) → R_a,R_{a+1}	0	0	X
26	2	a	m	5A	a	m	MK a,y,m	Multiply (Constant)	(R_{a+1}) * Y → R_a,R_{a+1}	0	0	X

OCTAL FORMAT o	f	a	m	HEXADECIMAL FORMAT OP	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
26	3	a	m	5B	a	m	M a,y,m	Multiply (Index)	(R_{a+1})*(Y) → R_a,R_{a+1}	0	0	X
27	0	a	m	5C	a	m	DR a,m	Divide (Register)	(R_a,R_{a+1})/(R_m) →R_{a+1}; Rem. → R_a	0	X	X
27	1	a	m	5D	a	m	DI a,m	Divide (Indirect)	(R_a,R_{a+1})/(Y*) → R_{a+1}; Rem. → R_a	0	X	X
27	2	a	m	5E	a	m	DK a,y,m	Divide (Constant)	(R_a, R_{a+1})/Y → R_{a+1}; Rem. → R_a	0	X	X
27	3	a	m	5F	a	m	D a,y,m	Divide (Index)	(R_a,R_{a+1})/(Y) → R_{a+1} Rem. → R_a	0	X	X
30	0	a	m	60	a	m	ANDR a,m	AND (Register)	(R_a) ⋀ (R_m) → R_a	0	0	X
30	1	a	m	61	a	m	ANDI a,m	AND (Indirect)	(R_a) ⋀ (Y*) → R_a	0	0	X
30	2	a	m	62	a	m	ANDK a,y,m	AND (Constant)	(R_a) ⋀ Y→R_a	0	0	X
30	3	a	m	63	a	m	AND a,y,m	AND (Index)	(R_a) ⋀ (Y)→R_a	0	0	X
31	0	a	m	64	a	m	ORR a,m	OR (Register)	(R_a)V(R_m) → R_a	0	0	X
31	1	a	m	65	a	m	ORI a,m	OR (Indirect)	(R_a)V(Y*) → R_a	0	0	X
31	2	a	m	66	a	m	ORK a,y,m	OR (Constant)	(R_a)VY → R_a	0	0	X
31	3	a	m	67	a	m	OR a,y,m	OR (Index)	(R_a)V(Y) → R_a	0	0	X
32	0	a	m	68	a	m	XORR a,m	Exclusive OR (Register)	(R_a)⊻(R_m) → R_a	0	0	X
32	1	a	m	69	a	m	XORI a,m	Exclusive OR (Indirect)	(R_a)⊻(Y*) → R_a	0	0	X
32	2	a	m	6A	a	m	XORK a,y,m	Exclusive OR (Constant)	(R_a)⊻Y → R_a	0	0	X
32	3	a	m	6B	a	m	XOR a,y,m	Exclusive OR (Index)	(R_a)⊻(Y) → 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)$_n$→R_{a_n}	0	0	X
33	1	a	m	6D	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	6E	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	6F	a	m	MS a,y,m	Masked Substitute (Index)	If (R_a+1)$_n$ = 1, (Y)$_n$→R_{a_n}	0	0	X
34	0	a	m	70	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	71	a	m	CMI a,m	Compare Masked (Indirect)	(R_a) ⋀ (R_a+1) : (Y*) ⋀ (R_a+1)	0	0	X
34	2	a	m	72	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	73	a	m	CM a,y,m	Compare Masked (Index)	(R_a) ⋀ (R_a+1) : (Y) ⋀ (R_a+1)	0	0	X
35	0	00	00	74	0	0	IOCR	£† Input/Output Command	Execute I/O command instruction located in 60 and 61(2)	–	–	–
35	0	a	m	74	a	m	IOC a,y,m	£† Input/Output Command	Execute I/O command instruction in location Y (and Y+1 if a 2-word instruction)(2)	–	–	–
35	1	00	m	75	0	m	BFI m	Biased Fetch (Indirect)	Set CC upon (Y*), 1→$Y^*_{15,14}$	0	0	X
35	2	00	m	76	0	m	REX y,m	Remote Execute	Execute (Y)	X/0	X/0	X/0
35	3	00	m	77	0	m	BF y,m	Biased Fetch (Index)	Set CC upon (Y), 1→$Y_{15,14}$	0	0	X
36	0	a	m	78	a	m	CLR a,m	Compare Logical (Register)	(R_a)::(R_m)	X	X	X
36	1	a	m	79	a	m	CLI a,m	Compare Logical (Indirect)	(R_a)::(Y*)	X	X	X
36	2	a	m	7A	a	m	CLK a,y,m	Compare Logical (Constant)	(R_a)::Y	X	X	X
36	3	a	m	7B	a	m	CL a,y,m	Compare Logical (Index)	(R_a)::(Y)	X	X	X
37	0	a	00	7C	a	0	VF a	# Trigonometric Vector without correction	0→R_a (R_a)=y (R_{a+1})=x (R_{a+2})=0	$\frac{\sqrt{(R_{a+1})^2+(R_a)^2}}{.4DBA}$ →R_{a+1} arctan((R_a)/(R_{a+1}))→R_{a+2}			
37	0	a	01	7C	a	1	RF a	# Trigonometric Rotate without correction	(R_a)cos(R_{a+2})+(R_{a+1})sin(R_{a+2}) / .4DBA →R_a (R_a)=y (R_{a+1})=x (R_{a+2})=θ	(R_{a+1})cos(R_{a+2})–(R_a)sin(R_{a+2}) / .4DBA →R_{a+1} 0→R_{a+2}			
37	0	a	02	7C	a	2	VFP a	# Trigonometric Vector	0→R_a (R_a)=y (R_{a+1})=x (R_{a+2})=0	$\sqrt{(R_a)^2+(R_{a+1})^2}$ →R_{a+1} arctan(R_a)/(R_{a+1}))→R_{a+2}			
37	0	a	03	7C	a	3	RFP a	# Trigonometric Rotate	(R_a)cos(R_{a+2})+(R_{a+1})sin(R_{a+2})→R_a (R_{a+1})cos(R_{a+2})–(R_a)sin(R_{a+2})→R_{a+1} 0→R_{a+2}			
37	0	a	04	7C	a	4	VH a	# Hyperbolic Vector without correction	0→R_a (R_a)=y (R_{a+1})=x (R_{a+2})=0	$\frac{\sqrt{(R_{a+1})^2-(R_a)^2}}{1.1A8F}$ →R_{a+1} arctanh((R_a)/(R_{a+1}))→R_{a+2}			
37	0	a	05	7C	a	5	RH a	# Hyperbolic Rotate without correction	(R_a)cosh(R_{a+2})+(R_{a+1})sinh(R_{a+2}) / 1.1A8F →R_a (R_a)=y (R_{a+1})=x (R_{a+2})=v	(R_a)sinh(R_{a+2})+(R_{a+1})cosh(R_{a+2}) / 1.1A8F →R_{a+1} 0→R_{a+2}			
37	0	a	06	7C	a	6	VHP a	# Hyperbolic Vector	0→R_a (R_a)=y (R_{a+1})=x (R_{a+2})=0	$\sqrt{(R_{a+1})^2-(R_a)^2}$ →R_{a+1} arctanh(R_a)/(R_{a+1}))→R_{a+2}			
37	0	a	07	7C	a	7	RHP a	# Hyperbolic Rotate	(R_a)cosh(R_{a+2})+(R_{a+1})sinh(R_{a+2})→R_a (R_{a+1})cosh(R_{a+2})+(R_a)sinh(R_{a+2})→R_{a+1} 0→R_{a+2}			
37	0	a	10	7C	a	8	FC a,y	# Floating Point Compare	(R_a,R_{a+1}):(Y,Y+1)	0	0	X
37	0	a	11	7C	a	9	FXC a	# Fixed to Floating Point Conversion	(R_a)→Exp.:(R_{a+1})→Man.	X	X	X
37	0	a	12	7C	a	A	FLC a	# Floating Point to Fixed Single Conversion	Convert (R_a,R_{a+1}), Exp.→R_a Man→R_{a+1}	0	0	X
37	0	a	13	7C	a	B	NF a	# Floating Point Normalize	Normalize (R_a,R_{a+1})	X	X	X
37	0	a	16	7C	a	E	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	0	X	X
37	0	a	17	7C	a	F	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	0	0	X
37	1	a	00	7D	a	0	SIN a	# Floating Point Sine	SIN (R_a,R_{a+1}) → R_a,R_{a+1}	0	X	X
37	1	a	01	7D	a	1	COS a	# Floating Point Cosine	COS(R_a,R_{a+1}) → R_a,R_{a+1}	0	X	X
37	1	a	02	7D	a	2	TAN a	# Floating Point Tangent	TAN(R_a,R_{a+1}) → R_a,R_{a+1}	0	X	X
37	1	a	03	7D	a	3	ASIN a	# Floating Point Arcsine	ASIN(R_a,R_{a+1}) → R_a,R_{a+1}	0	X	X
37	1	a	04	7D	a	4	ACOS a	# Floating Point Arccosine	ACOS(R_a,R_{a+1}) → R_a,R_{a+1}	0	X	X

(2) The command instruction address is relative to page set 0.

£ IOC required

OCTAL FORMAT o	f	a	m	HEXADECIMAL FORMAT OP	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
37	1	a	05	7D	a	5	ATAN a	# Floating Point Arctangent	ATAN(R$_a$,R$_{a+1}$) → R$_a$,R$_{a+1}$	0	X	X
37	1	a	06	7D	a	6	EXP a	# Floating Point Exponential	EXP(R$_a$,R$_{a+1}$) → R$_a$,R$_{a+1}$	0	X	X
37	1	a	07	7D	a	7	ALOG a	# Floating Point Natural Log	ALOG(R$_a$,R$_{a+1}$) → R$_a$,R$_{a+1}$	0	X	X
40	0	00	m	80	0	m	JER m	Jump Equal	If (CC) indicates = or 0, (R$_m$) → P	–	–	–
40	0	01	m	80	1	m	JNER m	Jump Not Equal	If (CC) indicates ≠ or not 0, (R$_m$) → P	–	–	–
40	0	02	m	80	2	m	JGER m	Jump Greater or Equal	If (CC) indicates ≥ or +, (R$_m$)→P	–	–	–
40	0	03	m	80	3	m	JLSR m	Jump Less	If (CC) indicates < or –, (R$_m$) → P	–	–	–
40	0	04	m	80	4	m	JOR m	Jump Overflow	If overflow set, (R$_m$) → P	–	–	–
40	0	05	m	80	5	m	JCR m	Jump Carry	If carry set, (R$_m$)→P	–	–	–
40	0	06	m	80	6	m	JPTR m	Jump Power Out of Tolerance	If power out of tolerance, (R$_m$)→P	–	–	–
40	0	07	m	80	7	m	JBR m	Jump Bootstrap 2 Selected	If bootstrap 2 selected, (R$_m$)→P	–	–	–
40	0	10	m	80	8	m	JR m	Jump	(R$_m$)→P	–	–	–
40	0	11	m	80	9	m	JSR m	† Jump After Stop	Stop; upon restart, (R$_m$) → P	–	–	–
40	0	12	m	80	A	m	JKSR 1,m	† Jump After Stop Key 1 Set	If key 1 set, stop; (R$_m$)→P	–	–	–
40	0	13	m	80	B	m	JKSR 2,m	† Jump After Stop Key 2 Set	If key 2 set, stop; (R$_m$)→P	–	–	–
40	1		d	81		d	LJ d	Local Jump	(P)+d→P	–	–	–
40	1	0→	1	81	0→	1	NOP	No Operation (Software)	(P)+1→P	–	–	–
2(40	1	0→	1)	2(81	0→	1)	NOPD	No Operation Double (Software)	(P)+1→P, (P)+1→P	–	–	–
40	2	00	m	82	0	m	JE y,m	Jump Equal	If (CC) indicates = or 0, Y→P	–	–	–
40	2	01	m	82	1	m	JNE y,m	Jump Not Equal	If (CC) indicates ≠ or not 0, Y→P	–	–	–
40	2	02	m	82	2	m	JGE y,m	Jump Greater or Equal	If (CC) indicates ≥ or +, Y→P	–	–	–
40	2	03	m	82	3	m	JLS y,m	Jump Less	If (CC) indicates < or –, Y→P	–	–	–
40	2	04	m	82	4	m	JO y,m	Jump Overflow	If overflow set, Y→P	–	–	–
40	2	05	m	82	5	m	JC y,m	Jump Carry	If carry set, Y→P	–	–	–
40	2	06	m	82	6	m	JPT y,m	Jump Power Out of Tolerance	If power out of tolerance, Y→P	–	–	–
40	2	07	m	82	7	m	JB y,m	Jump Bootstrap 2 selected	If bootstrap 2 selected, Y → P	–	–	–
40	2	10	m	82	8	m	J y,m	Jump	Y→P	–	–	–
40	2	11	m	82	9	m	JS y,m	† Jump After Stop	Stop; upon restart Y→P	–	–	–
40	2	12	m	82	A	m	JKS 1,y,m	† Jump After Stop Key 1 Set	If key 1 set, stop; Y→P	–	–	–
40	2	13	m	82	B	m	JKS 2,y,m	† Jump After Stop Key 2 Set	If key 2 set, stop; Y→P	–	–	–
40	3	00	m	83	0	m	JE *y,m	Jump Equal	If (CC) indicates = or 0, (Y) → P	–	–	–
40	3	01	m	83	1	m	JNE *y,m	Jump Not Equal	If (CC) indicates ≠ or not 0, (Y)→P	–	–	–
40	3	02	m	83	2	m	JGE *y,m	Jump Greater or Equal	If (CC) indicates ≥ or +, (Y) → P	–	–	–
40	3	03	m	83	3	m	JLS *y,m	Jump Less	If (CC) indicates < or –, (Y) → P	–	–	–
40	3	04	m	83	4	m	JO *y,m	Jump Overflow	If overflow set, (Y) → P	–	–	–
40	3	05	m	83	5	m	JC *y,m	Jump Carry	If carry set, (Y) → P	–	–	–
40	3	06	m	83	6	m	JPT *y,m	Jump Power Out of Tolerance	If power out of tolerance, (Y)→P	–	–	–
40	3	07	m	83	7	m	JB *y,m	Jump Bootstrap 2 selected	If bootstrap 2 selected, (Y) → P	–	–	–
40	3	10	m	83	8	m	J *y,m	Jump	(Y)→P	–	–	–
40	3	11	m	83	9	m	JS *y,m	† Jump After Stop	Stop; upon restart, (Y) → P	–	–	–
40	3	12	m	83	A	m	JKS 1,*y,m	† Jump After Stop Key 1 Set	If key 1 set, stop; (Y)→P	–	–	–
40	3	13	m	83	B	m	JKS 2,*y,m	† Jump After Stop Key 2 Set	If key 2 set, stop; (Y)→P	–	–	–
41	0	a	m	84	a	m	XJR a,m	Index Jump	If (R$_a$) ≠ 0, (R$_a$) – 1 → R$_a$, (R$_m$) → P	–	–	–
41	1		d	85		d	LJI d	Local Jump Indirect	((P)+d)→P	–	–	–
41	2	a	m	86	a	m	XJ a,y,m	Index Jump	If (R$_a$) ≠ 0, (R$_a$) – 1 → R$_a$, Y→P	–	–	–
41	3	a	m	87	a	m	XJ a,*y,m	Index Jump	If (R$_a$) ≠ 0, (R$_a$) – 1 → R$_a$, (Y)→P	–	–	–
42	0	a	m	88	a	m	JLRR a,m	Jump, Link Register	(P) + 1 → R$_a$; (R$_m$) → P	–	–	–
42	2	a	m	8A	a	m	JLR a,y,m	Jump, Link Register	(P) + 2 → R$_a$; Y → P	–	–	–
42	3	a	m	8B	a	m	JLR a,*y,m	Jump, Link Register	(P) + 2 → R$_a$; (Y)→P	–	–	–
43	1		d	8D		d	LJLM d	Local Jump, Link Memory	(P) + 1 → (P) + d; (P) + d + 1 → P	–	–	–
43	2	00	m	8E	0	m	JLM y,m	Jump, Link Memory	(P) + 2 → Y; Y + 1 → P	–	–	–
43	3	00	m	8F	0	m	JLM *y,m	Jump, Link Memory	(P) + 2 → (Y); (Y) + 1 → P	–	–	–
44	0	a	m	90	a	m	JZR a,m	Jump Zero	If (R$_a$) = 0, (R$_m$) → P	–	–	–
44	1		d	91		d	LJE d	Local Jump Equal	If (CC) indicates = or 0, (P)+d→P	–	–	–
44	2	a	m	92	a	m	JZ a,y,m	Jump Zero	If (R$_a$) = 0, Y→P	–	–	–
44	3	a	m	93	a	m	JZ a,*y,m	Jump Zero	If (R$_a$) = 0, (Y)→P	–	–	–
45	0	a	m	94	a	m	JNZR a,m	Jump Not Zero	If (R$_a$)≠0, (R$_m$)→P	–	–	–
45	1		d	95		d	LJNE d	Local Jump Not Equal	If (CC) indicates ≠ or not 0, (P)+d→P	–	–	–
45	2	a	m	96	a	m	JNZ a,y,m	Jump Not Zero	If (R$_a$) ≠ 0, Y→P	–	–	–
45	3	a	m	97	a	m	JNZ a,*y,m	Jump Not Zero	If (R$_a$) ≠ 0, (Y)→P	–	–	–
45	0	a	m	98	a	m	JPR a,m	Jump Positive	If (R$_a$)≥0, (R$_m$)→P	–	–	–
46	1		d	99		d	LJGE d	Local Jump Greater or Equal	If (CC) indicates ≥ or +, (P)+d→P	–	–	–
46	2	a	m	9A	a	m	JP a,y,m	Jump Positive	If (R$_a$) ≥ 0, Y→P	–	–	–
46	3	a	m	9B	a	m	JP a,*y,m	Jump Positive	If (R$_a$)≥0, (Y)→P	–	–	–
47	0	a	m	9C	a	m	JNR a,m	Jump Negative	If (R$_a$) < 0, (R$_m$) → P	–	–	–
47	1		d	9D		d	LJLS d	Local Jump Less	If (CC) indicates < or –, (P)+d→P	–	–	–
47	2	a	m	9E	a	m	JN a,y,m	Jump Negative	If (R$_a$) < 0, Y→P	–	–	–
47	3	a	m	9F	a	m	JN a,*y,m	Jump Negative	If (R$_a$)<0, (Y)→P	–	–	–
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$_{a+3}$	0	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$_{a+3}$	0	X	X
50	3	a	m	A3	a	m	FSU a,y,m	# Floating Point Subtract (Index)	(R$_a$, R$_{a+1}$) – (Y, Y+1) → R$_a$, R$_{a+1}$; Res.→R$_{a+2}$, R$_{a+3}$	0	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$_{a+3}$	0	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$_{a+3}$	0	X	X

OCTAL FORMAT o	f	a	m	HEXADECIMAL FORMAT OP	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
51	3	a	m	A7	a	m	FA a,y,m	# Floating Point Add (Index)	(R$_a$, R$_{a+1}$) + (Y, Y+1) → R$_a$, R$_{a+1}$; Res.→R$_{a+2}$, R$_{a+3}$	0	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$_{a+3}$	0	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$_{a+3}$	0	X	X
52	3	a	m	AB	a	m	FM a,y,m	# Floating Point Multiply (Index)	(R$_a$, R$_{a+1}$)*(Y, Y+1) → R$_a$, R$_{a+1}$; Res.→R$_{a+2}$, R$_{a+3}$	0	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$_{a+3}$	0	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$_{a+3}$	0	X	X
53	3	a	m	AF	a	m	FD a,y,m	# Floating Point Divide (Index)	(R$_a$, R$_{a+1}$)/(Y,Y+1) → R$_a$, R$_{a+1}$; Rem.→R$_{a+2}$, R$_{a+3}$	0	X	X
54	0	a	m	B0	a	m	LARR a,m	‡† Load Address Register (Register)	• (R$_m$) → AR$_r$	–	–	–
54	1	a	m	B1	a	m	LARI a,m	‡† Load Address Register (Indirect)	• (Y*)→AR$_r$	–	–	–
54	3	a	m	B3	a	m	LARM a,y,m	‡† Load Address Register Multiple (Index)	• (Y, ..., Y+u)→AR$_r$,...,AR$_{r+u}$	–	–	–
55	0	a	m	B4	a	m	SARR a,m	‡ Store Address Register (Register)	• (AR$_r$)→R$_m$	–	–	–
55	1	a	m	B5	a	m	SARI a,m	‡ Store Address Register (Indirect)	• (AR$_r$)→Y*	–	–	–
55	3	a	m	B7	a	m	SARM a,y,m	‡ Store Address Register Multiple (Index)	• (AR$_r$,...,AR$_{r+u}$) → Y,..., Y+u	–	–	–
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$_{a+3}$	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$_{a+3}$	0	0	X
56	3	a	m	BB	a	m	MD a,y,m	# Multiply Double (Index)	(R$_a$, R$_{a+1}$)*(Y, Y+1) → R$_a$, R$_{a+1}$, R$_{a+2}$, R$_{a+3}$	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$_{a+1}$	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$_{a+1}$	0	X	X
57	3	a	m	BF	a	m	DD a,y,m	# Divide Double (Index)	(R$_a$, R$_{a+1}$, R$_{a+2}$, R$_{a+3}$)/ (Y, Y+1) → R$_{a+2}$, R$_{a+3}$; Rem. → R$_a$, R$_{a+1}$	0	X	X
60	0	a	m	C0	a	m	LLRS a,m	Logical Right Single Shift (Literal)	Shift (R$_a$) right m places, zero fill	0	0	X
60	1	a	m	C1	a	m	LARS a,m	Algebraic Right Single Shift (Literal)	Shift (R$_a$) right m places, sign fill	0	0	X
60	2	a	m	C2	a	m	LLRD a,m	Logical Right Double Shift (Literal)	Shift (R$_a$, R$_{a+1}$) right m places, zero fill	0	0	X
60	3	a	m	C3	a	m	LARD a,m	Algebraic Right Double Shift (Literal)	Shift (R$_a$, R$_{a+1}$) right m places, sign fill	0	0	X
61	0	a	m	C4	a	m	LALS a,m	Algebraic Left Single Shift (Literal)	Shift (R$_a$) left m places, zero fill	0	X	X
61	1	a	m	C5	a	m	LCLS a,m	Circular Left Single Shift (Literal)	Shift (R$_a$) left circularly m places	0	0	X
61	2	a	m	C6	a	m	LALD a,m	Algebraic Left Double Shift (Literal)	Shift (R$_a$, R$_{a+1}$) left m places, zero fill	0	X	X
61	3	a	m	C7	a	m	LCLD a,m	Circular Left Double Shift (Literal)	Shift (R$_a$, R$_{a+1}$) left circularly m places	0	0	X
62	0	a	m	C8	a	m	LSU a,m	Subtract (Literal)	(R$_a$) – m → R$_a$	X	X	X
62	1	a	m	C9	a	m	LSUD a,m	Subtract Double (Literal)	(R$_a$, R$_{a+1}$) – m → R$_a$, R$_{a+1}$	X	X	X
62	2	a	m	CA	a	m	LA a,m	Add (Literal)	(R$_a$)+m→R$_a$	X	X	X
62	3	a	m	CB	a	m	LAD a,m	Add Double (Literal)	(R$_a$, R$_{a+1}$) + m → R$_a$, R$_{a+1}$	X	X	X
63	0	a	m	CC	a	m	LL a,m	Load (Literal)	m → R$_a$	0	0	X
63	1	a	m	CD	a	m	LC a,m	Compare (Literal)	(R$_a$) : m	X	X	X
63	2	a	m	CE	a	m	LMUL a,m	Multiply (Literal)	(R$_{a+1}$) * m → R$_a$, R$_{a+1}$	0	0	X
63	3	a	m	CF	a	m	LDIV a,m	Divide (Literal)	(R$_a$, R$_{a+1}$)/m → R$_{a+1}$; Rem. → R$_a$	0	X	X
64	0	a	m	D0	a	m	LIR a,m	†Load Inter-Register	(R$_m$*) → R$_a$[3]	0	0	X
64	2	a	m	D2	a	m	LMR a,y,m	Load Multiple (Index)	(Y...Y+15–a) → R$_a$...R$_{15}$	–	–	–
64	3	a	m	D3	a	m	BSU a,y,m	Byte Subtract	(R$_a$) – (Y)$_{byte}$ → R$_a$	X	X	X
65	0	a	m	D4	a	m	SIR a,m	†Store Inter-Register	(R$_a$) → R$_m$[3]	0	0	X
65	2	a	m	D6	a	m	SMR a,y,m	Store Multiple (Index)	(R$_a$...R$_{15}$) → (Y...Y+15–a)	–	–	–
65	3	a	m	D7	a	m	BA a,y,m	Byte Add (Index)	(R$_a$)+(Y)$_{byte}$ → R$_a$	X	X	X
66	3	a	m	DB	a	m	BC a,y,m	Byte Compare (Index)	(R$_a$) : (Y)$_{byte}$	X	X	X
67	0	a	m	DC	a	m	UM1 a,m	User Macro (Software)	Reserved for user macro instructions	–	–	–
67	0	a	m	DC	a	m	UM2 a,m	User Macro (Software)	Reserved for user macro instructions	–	–	–
67	1	a	m	DD	a	m	UMI a,m	User Macro (Software)	Reserved for user macro instructions	–	–	–
67	2	a	m	DE	a	m	UMK a,y,m	User Macro (Software)	Reserved for user macro instructions	–	–	–
67	3	a	m	DF	a	m	BCX a,y,m	Byte Compare and Index by 1 (Index)	(R$_a$) : (Y)$_{byte}$; (R$_m$) + 1 → R$_m$	X	X	X
70	0	a	m	E0	a	m	LPAR a,m	‡† Load Physical Address (Register)	MAP (Y*, (R$_{m+1}$)) → Y**, Y** → R$_a$, R$_{a+1}$	0	0	X
70	1	a	m	E1	a	m	LPAI a,m	‡† Load Physical Address (Indirect)	MAP ((Y*), (R$_{m+1}$)) → Y**, Y** → R$_a$, R$_{a+1}$	0	0	X
70	2	a	m	E2	a	m	LPAK a,m	‡† Load Physical Address (Constant)	MAP (Y, (R$_{m+1}$)) → Y**, Y** → R$_a$, R$_{a+1}$	0	0	X
70	3	a	m	E3	a	m	LPA a,y,m	‡† Load Physical Address (Index)	MAP ((Y), (R$_{m+1}$)) → Y**, Y** → R$_a$, R$_{a+1}$	0	0	X
71	3	a	m	E7	a	m	LMAP a,y,m	‡† Load Mapped	MAP (Y, (R$_{m+1}$)) → Y**, (Y**) → R$_a$	0	0	X
72	3	a	m	EB	a	m	SMAP a,y,m	‡† Store Mapped	MAP (Y, (R$_{m+1}$)) → Y**, (R$_a$) → Y**	–	–	–
73	1	a	m	ED	a	m	LPLI a,m	‡† Load Physical Location (Indirect)	(R$_m$, R$_{m+1}$) → Y**, (Y**) → R$_a$	0	0	X
73	3	a	m	EF	a	m	LPL a,y,m	‡† ▪Load Physical Location (Index)	Y$_{R_m, R_{m+1}}$ → Y**, (Y**) → R$_a$	0	0	X
74	1	a	m	F1	a	m	SPLI a,m	‡† Store Physical Location (Indirect)	Y$_{R_m, R_{m+1}}$ → Y**, (R$_a$) → Y**	–	–	–
74	3	a	m	F3	a	m	SPL a,y,m	‡† ▪Store Physical Location (Index)	Y$_{R_m, R_{m+1}}$ → Y**, (R$_a$) → Y**	–	–	–

▪ *Special Indexing: Effective address = y + (R$_m$, R$_{m+1}$)*

• *r = (R$_a$)$_{7–0}$ = Word Designator*
u = (R$_a$)$_{15–8}$ = Count

(3) Rm is general register m of the general set not selected in bit 14 of Status Register 1*

‡ *MAE Card required*

INPUT OUTPUT INSTRUCTIONS

OCTAL FORMAT o	f	a	m	HEXADECIMAL FORMAT OP	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9-8 CC
							INPUT/OUTPUT INSTRUCTIONS – COMMAND/CHAIN INSTRUCTIONS				
70	0	00	00	E0	0	0	ACR 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; Set External Interrupt Enable (EIE) line	–	–	–
70	0	00	05	E0	0	5	ACR 5 CCR 0,5	Channel Control	Disable external interrupts channel a; all channels; Clear External Interrupt Enable (EIE) line	–	–	–
70	0	a	06	E0	a	6	CCR a,6	Enable Selected Interrupts	Enable Class III, Priority 2,3,4 interrupts, channels 0 to a–1	–	–	–
70	0	a	07	E0	a	7	CCR a,7	Disable Selected Interrupts	Disable Class III, Priority 2,3,4 interrupts, channels 0 to a–1	–	–	–
70	0	a	10	E0	a	8	CCR a,8	Channel Control	Master clear, channel a	–	–	–
70	0	a	11	E0	a	9	CCR a,9	Clear Input on Channel a				
70	0	a	12	E0	a	A	CCR a,A	Clear Output on Channel a				
70	0	a	14	E0	a	C	CCR a,C	Channel Control	Enable external interrupts, channel a; Set External Interrupt Enable (EIE) line	–	–	–
70	0	a	15	E0	a	D	CCR a,D	Channel Control	Disable external interrupts,channel a, Clear External Interrupt Enable (EIE) line	–	–	–
70	0	a	16	E0	a	E	CCR a,E	Channel Control	Enable Class III, Priority 2,3,4 interrupts, channel a	–	–	–
70	0	a	17	E0	a	F	CCR a,F	Channel Control	Disable Class III, Priority 2,3,4 interrupts, channel a	–	–	–
							INPUT/OUTPUT INSTRUCTIONS – COMMAND INSTRUCTIONS				
71	2	a	02	E6	a	2	ICK a,y	Initiate Input Chain	Y→IOCM$_2$, initiate input chain	–	–	–
71	2	a	06	E6	a	6	OCK a,y	Initiate Output Chain	Y→IOCM$_6$, initiate output chain	–	–	–
71	2	a	m	E6	a	m	WIMK a,y,m	Write Control Memory	Y→IOCM$_m$, channel a	–	–	–
71	2	a	m	E6	a	m	WCMK am,y	Write Control Memory				
71	3	a	m	E7	a	m	WIM a,y,m	Write Control Memory	(Y)→IOCM$_m$, channel a	–	–	–
72	3	a	m	EB	a	m	RIM a,y,m RCM am,y	Read Control Memory	Channel a, (IOCM$_m$) → Y	–	–	–
76	0	a	m	F8	a	m	SICR a,m	Serial Interface Control	Set or clear serial channel a discretes	–	–	–
76	3	a	m	FB	a	m	SST a,y,m	Store Serial Status	Channel a status bits per m → Y	–	–	–
							INPUT/OUTPUT INSTRUCTIONS – CHAIN INSTRUCTIONS				
70	2	a	m	E2	a	m	LCMI a,y,m	Load Control Memory				
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→IOCM$_m$	–	–	–
71	3	00	m	E7	0	m	LCM m,y	Load Control Memory	(Y)→IOCM$_m$	–	–	–
72	3	00	m	EB	0	m	SCM m,y	Store Control Memory	(IOCM$_m$) → Y	–	–	–
73	0	00	00	EC	0	0	HCR	Halt Chain	Halt chaining (chaining)	–	–	–
73	0	01	00	EC	1	0	IPR	Interrupt Processor	Generate chain interrupt (chaining)	–	–	–
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$_{15,14}$	–	–	–
73	3	02	00	EF	2	0	TF y	Test and Set Y	0 → Y$_{15,14}$ set condition			
73	3	04	m	EF	4	m	ZB y,m	Clear Bit	0 → Ym			
73	3	05	m	EF	5	m	SB y,m	Set Bit	1 → Ym			
73	3	07	m	EF	7	m	CB y,m	Compare Bit to Zero	Ym:0 set condition			
74	2	00	00	F2	0	0	SJC 0,y	Serial Jump (Unconditional)	Unconditional Y → CAP; clear flag	–	–	–
74	2	01	00	F2	1	0	SMJC 1,y	Serial Jump (Conditional)	Serial Jump if suppress flag not set. No jump for MIL-STD-1397 or NAT-STD-4153. (4)			
74	2	02	00	F2	2	0	SMJC 2,y	Serial Jump (Conditional)	Serial Jump if monitor flag set. No jump for MIL-STD-1397 or NAT-STD-4153. (4)			
74	2	04	00	F2	4	0	SJMC 4,y	Serial Jump (Conditional)	Jump if condition bit (bit 15) in I/O status word is set.			
74	2	10	00	F2	8	0	SJMC 8,y	Serial Jump (Conditional)	Y → CAP if Input Buffer is active			
74	2	11	00	F2	9	0	SJMC 9,y	Serial Jump (Conditional)	Y → CAP if Output Buffer is active			
74	2	12	00	F2	A	0	SJMC A,y	Serial Jump (Conditional)	Y → CAP if External Function Buffer is active. No jump for MIL-STD-188C, RS-232-C, or VACALES			
75	0	00	m	F4	0	m	SFSC m	Search for sync	Perform functions per m-designator	–	–	–
76	0	00	m	F8	0	m	CSIR m	Serial Interface Control	Set or clear serial channel discrete function	–	–	–
76	3	00	m	FB	0	m	CSST y,m	Store Serial Status	Serial status bit per m → Y	–	–	–
77	3	a	m	FF	a	m	IIC a,y,m	Built-In Test (BIT)	Execute the IOC BIT subtest specified by (Y)	–	–	–

(4) for MIL-STD-188C and RS-232-C flag is cleared during next character time; for VACALES, flag is cleared when next character is transferred to memory.

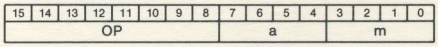
ASSIGNED MEMORY ADDRESSES

ADDRESS	ASSIGNMENT
0–3F C0–13F	NDRO MEMORY
48–5F	INTERRUPT PROCESSING
60–61 78–7D	COMMAND CELLS, IOC 0 BIT SIGNATURE
7F	AUTO START ENTRANCE (NORMAL)
80–BF	EXTERNAL INTERRUPT WORD STORAGE (IOC)

INSTRUCTION FORMATS

INSTRUCTION TYPE

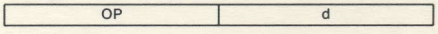
RL



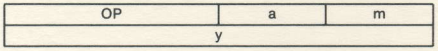
OP – 8-bit code specifying the operation; RL format only
a – General register designator
m – 4-bit literal constant

RR
RI, TYPE 2

RI, TYPE 1



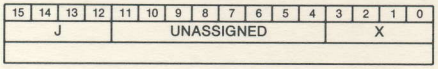
RK, RX



OP CODE – Code specifying the operation
a – General register or subfunction designator
m – General register or subfunction designator
d – Displacement value (two's complement)
y – Address or arithmetic constant

INDIRECT WORD FORMAT

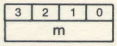
IW 1
IW 2



J-VALUE	OPERAND ADDRESS
0	IW 2
1	IW 2 + (Rx)
2	IW 2 + (Rm)
3	IW 2 + (Rm+1)
J-VALUE	OPERAND ADDRESS (CASCADED)
4	IW at IW 2
5	IW at IW 2 + (Rx)
6	IW at IW 2 + (Rm)
7	IW at IW 2 + (Rm+1)
10–17	Unassigned

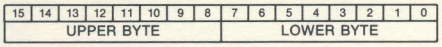
OPERAND FORMATS

Literal Format – 4-bit unsigned integer

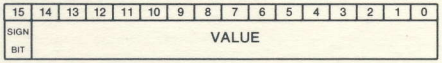


4-bit m-field of the RL format instructions

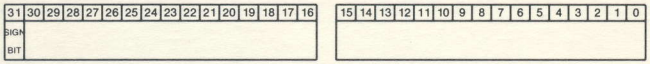
Byte Format – 8-bit unsigned integer



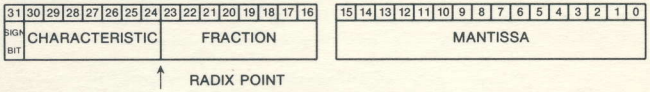
Single-Length Format



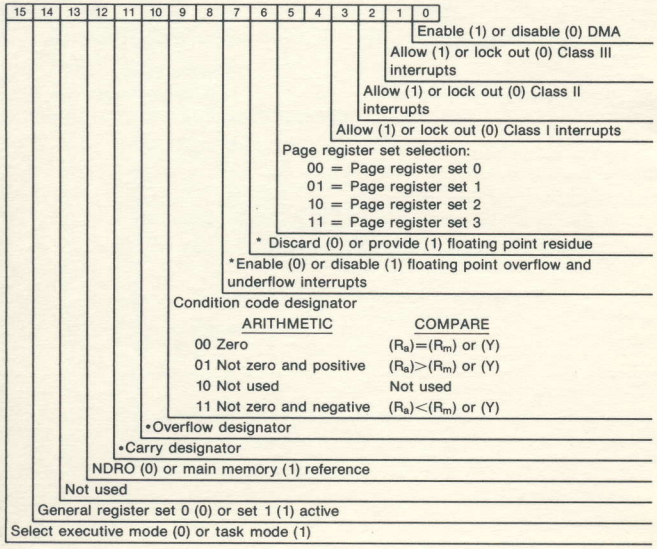
Double-Length Format Ra,Ra+1; Rm,Rm+1; y, y+1



Floating-Point Format (Ra), (Ra+1); (Rm), (Rm+1); (y), (y+1)



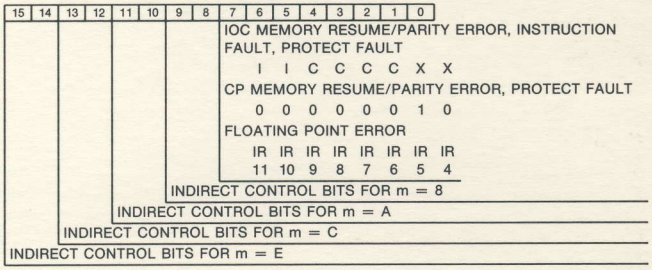
STATUS REGISTER 1 FORMAT



* *MATHPAC option only*

• *Bits 11 and 10 together form the floating point underflow or overflow designator, as follows:*
01 = Overflow
11 = Underflow

STATUS REGISTER 2 FORMAT



X INTERPRETATION

XX = 00 – INPUT CHAIN
01 – OUTPUT CHAIN
11 – I/O COMMAND

C INTERPRETATION

CCCC – CHANNEL NUMBER

I INTERPRETATION

II – IOC NUMBER

INDIRECT CONTROL BIT INTERPRETATION

00 – NORMAL ADDRESSING
01 – NORMAL ADDRESSING
10 – INDIRECT ADDRESSING (WORD AT y)
11 – INDIRECT ADDRESSING WITH INDEXING (WORD AT y + Rm)

OR

V |	0	1
0	0	1
1	1	1

XOR

V |	0	1
0	0	1
1	1	0

AND

A |	0	1
0	0	0
1	0	1

OPERAND FORMATION

FORMAT	DESCRIPTION
RR	Operand=(Rm)
R1, TYPE 1	Local Jump Address Y=(P)+d
R1, TYPE 2	Operand at Y*=(Rm)
RK	Operand Y=y+(Rm) if m≠0 Operand Y=y if m=0
RX Word	Operand at Y=y if m=0 Operand at Y=y+(Rm) if m≠0
RX Byte	Operand at Y upper if m=0 Operand at Y=(Rm)/2+y if m≠0 B=$(Rm)_0$
RL	Operand=m (an absolute literal)

MRC DISPLAY

DISPLAY CODE	INFORMATION DISPLAYED	
000	MRC State	AXX– RUN (Program Run) – = blank XFXX PWR (Power Fault) XXFX PROG (Instruction Fault) –XXS STOP – = blank s = STOP condition s = 0 Power up or Master Clear 1 Jump-stop 1 2 Jump-stop 2 3 Unconditional jump-stop 4 Stop key depression 5 Breakpoint stop 6 Opstep stop
001	Status Register 1	
002	Status Register 2	
003	Program Address Register	
004	Instruction Register	
005	Real-Time Clock Register Lower	
006	Real-Time Clock Register Upper	
007	Monitor Clock Register	
008	Relative Memory Address	
009	Relative Memory Data	
00A	Absolute Memory Addresses 16-21	
00B	Absolute Memory Addresses 0-15	
00C	Absolute Memory Data	
00D	Breakpoint Address	$0_2 = 0$ Disable instruction breakpoint $0_2 = 1$ Enable instruction breakpoint $1_2 = 0$ Disable write breakpoint $1_2 = 1$ Enable write breakpoint $2_2 = 0$ Disable read breakpoint $2_2 = 1$ Enable read breakpoint
00E	Breakpoint Mode	
00F	Operation Step Control	0 - CP run mode 1 - CP opstep mode 2 - IOC opstep mode
100-10F	General Register Set 0	$3-0_2$ = Register
110-11F	General Register Set 1	$3-0_2$ = Register
200-2FF	Page Registers, 00-3F	$7-6_2$ = Page register set $5-0_2$ = Page register
300-31F	P History Address/Code	300 = Address of most recent instruction to alter P 301 = Type of instruction that changed P
A00-AFF	IOC Control Memory	$4-7_2$ = Channel $3-0_2$ = Channel memory location
B00-BFF	IOC Channel Status	$4-7_2$ = Channel $3-0_2$ = Channel status location
C00-C0F	IOC Output Data	$3-0_2$ = Channel
D00	IOC Command Address	
D01	IOC Command Instruction	
D02	IOC Chain Instruction	
D03	IOC Translates	
DFF	IOC Select	
E00-E59	Test Parameters	
EEE	Test in Process	
F00-F05	Fault Signature	
FFF	Fault Code	

CORDIC FUNCTIONS (OPTIONAL MATHPAC INSTRUCTIONS)

HEXADECIMAL FORMAT	OCTAL FORMAT	CODING	FUNCTION	INPUT PARAMETERS			OUTPUT PARAMETERS		
OP a m	o f a m	FORMAT		R_k	R_{k+1}	R_{k+2}	$Y \rightarrow R_k$	$X \rightarrow R_{k+1}$	$W \rightarrow R_{k+2}$
7C a 0	37 0a 00	VF a	Trigonometric vector without correction	y	x	0	0	$X=\frac{R}{K}=\frac{\sqrt{x^2+y^2}}{K}$	$W=\theta=\tan^{-1}\frac{y}{x}$
7C a 1	37 0a 01	RF a	Trigonometric rotate without correction	y	x	θ	$Y=\frac{y\cos\theta+x\sin\theta}{K}$	$X=\frac{x\cos\theta-y\sin\theta}{K}$	0
7C a 2	37 0a 02	VFP a	Trigonometric vector	y	x	0	0	X3 $R=\sqrt{x^2+y^2}$	$W=\theta=\tan^{-1}\frac{y}{x}$
7C a 3	37 0a 03	RFP a	Trigonometric rotate	y	x	θ	$Y=y\cos\theta+x\sin\theta$	$X=x\cos\theta-y\sin\theta$	0
7C a 4	37 0a 04	VH a	Hyperbolic vector without correction	y	x	0	0	$X=\frac{\sqrt{x^2-y^2}}{K_1}$	$W=v=\tanh^{-1}\frac{y}{x}$
7C a 5	37 0a 05	RH a	Hyperbolic rotate without correction	y	x	v	$Y=\frac{y\cosh v+x\sinh v}{K_1}$	$X=\frac{x\cosh v+y\sinh v}{K_1}$	0
7C a 6	37 0a 06	VHP a	Hyperbolic vector	y	x	0	0	$X=\sqrt{x^2-y^2}$	$W=v=\tanh^{-1}\frac{y}{x}$
7C a 7	37 0a 07	RHP a	Hyperbolic rotate	y	x	v	$Y=y\cosh v+x\sinh v$	$X=x\cosh v+y\sinh v$	0
7C a 1	37 0a 01	RF a	Sin θ, COS θ	0	0.4DBA	θ	$Y=\sin\theta$	$X=\cos\theta$	0
7C a 6	37 0a 06	VHP a	$\text{Log}_e x$	x-1	x+1	0	0	$2\sqrt{x}$	$W=1/2\ \log_e x$ $=\tanh^{-1}\frac{x-1}{x+1}$
7C a 7	37 0a 07	RHP a	Exponential	1	1	v positive	$Y=e^x=\sinh v+\cosh v$	$X=e^x=\sinh v+\cosh v$	0
7C a 1	37 0a 01	RF a	Polar to Cartesian without correction	0	R	θ	$Y=\frac{R\sin\theta}{K}$	$X=\frac{R\cos\theta}{K}$	0
7C a 3	37 0a 03	RFP a	Polar to Cartesian	0	R	θ	$Y=R\sin\theta$	$X=R\cos\theta$	0
7C a 1	37 0a 01	RF a	Sin θ; cos θ	0	1	θ	$Y=\frac{\sin\theta}{K}$	$X=\frac{\cos\theta}{K}$	0

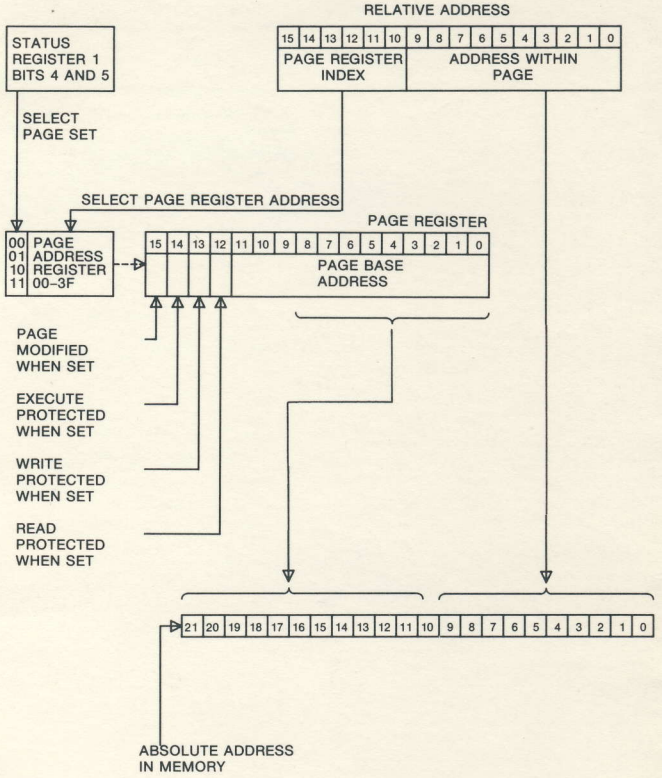
x,y Cartesian Coordinates
θ Angle of Rotation Trigonometric Mode
w Angle of Rotation Hyperbolic Mode
k 0.4DBA
l 1.1A8F

Bit 15 of all input parameters indicates sign 0 = positive, 1 = negative
Two's complement notation is used for negative values
The radix point for Registers R_k and R_{k+1} must be the same
The radix point for W = Constant in hyberbolic mode is between bit 2^{15} and 2^{14}

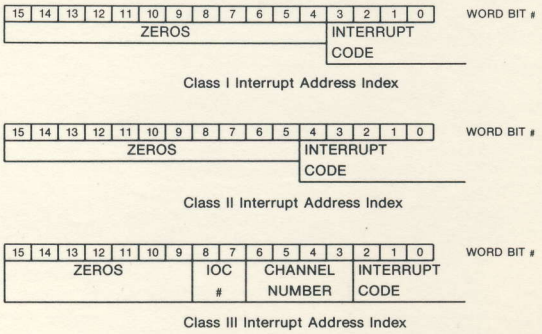
The maximum value for positive trigonometric coordinates x and y is 36F6 for m = 0,1 and 5A82 for m = 2,3
The maximum value for positive hyperbolic coordinates x and y is 35CD for m = 5 and 2D7C for m = 7

Angle θ is represented in Binary Angular Measurement (BAMS), Bit 2^{15} represents 180°. Each successive bit equal to one represents an angle one-half as large as its adjoining higher order bit. Least significant bit = .0054931° = 19.7" $y/x \leq .75$ for m = 4, 6 and $x \leq 76A6$ for m = 6.

MEMORY ADDRESS GENERATION



INTERRUPT ENTRANCE ADDRESS INDEX



INTERRUPT PRIORITY

CLASS	PRIORITY	INTERRUPT	BINARY INTERRUPT CODE	NOTES
I	1	Power Fault	0000	1
HARDWARE	2	IOC Memory Resume	0010	2
	3	IOC Memory Parity	0100	2
	4	CP Memory Resume	0010	2
	5	CP Memory Parity	0100	2
II	1	CP Instruction Fault	00000	1
SOFTWARE	2	IOC Instruction Fault (74)	00010	3
	3	IOC Instruction Fault	00010	3
	4	IOC Protect Fault	11000	2
	5	Floating Point	00100	4
	6	Executive Return	00110	4
	7	Executive Mode Fault	10000	1
	8	CP Protect Fault	11000	2
	9	RTC Overflow	01000	5
	10	Monitor Clock	01010	5
III	1	IOC Intercomputer Timeout	II CCCC 110	6
IOC AND MMIO	2	IOC External Interrupt/Discrete	II CCCC 000	6,7
	3	IOC Output Chain Interrupt	II CCCC 100	6
	4	IOC Input Chain Interrupt	II CCCC 010	6
	5	MMIO Discrete Interrupt	CC CCCC 110	8
	6	MMIO External Interrupt	CC CCCC 000	8
	7	MMIO Output Data Ready	CC CCCC 100	8
	8	MMIO Input Data Ready	CC CCCC 010	8

NOTES:

1 Cannot be locked out

2 Interrupt is lost if locked out

3 Interrupt action is not locked out within the IOC, but the interrupt is lost if locked out by the CP

4 No operation if locked out

5 One level of queuing

6 One level of queuing per channel

7 Discrete interrupt for MIL-STD-188C, VACALES, or RS-232-C Serial channels

8 Bits 3 through 8 define the MMIO channel number

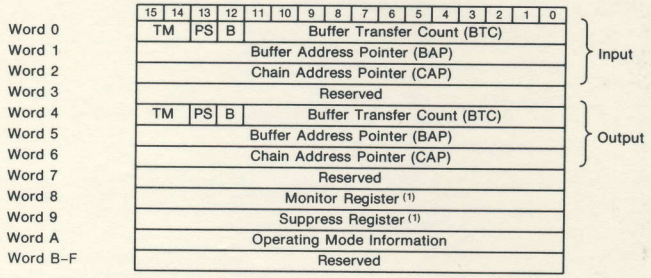
II–IOC Number

C –Channel Number

MAIN MEMORY ASSIGNMENTS FOR INTERRUPT HANDLING

FUNCTION	ADDRESS ASSIGNMENT TO CLASS I	II	III
Store the contents of P at address	58	50	48
Store the contents of SR1 at address	59	51	49
Store the contents of SR2 at address	5A	52	4A
Store the contents of RTC lower at address	5B	53	4B
Store the contents of RTC upper at address	5F	57	4F
Reload P with index plus the contents of address	5C	54	4C
Reload SR1 from address	5D	55	4D
Reload SR2 from address	5E	56	4E

I/O CONTROL MEMORY



TM = 00 – Abort the transfer. For input, continue accepting the input data, but do not write it into memory.

TM = 01 – Transfer 8-bit bytes.

TM = 10 – Transfer 16-bit words.

TM = 11 – Transfer 32-bit double words.

PS = 0 – Use page register set 0.

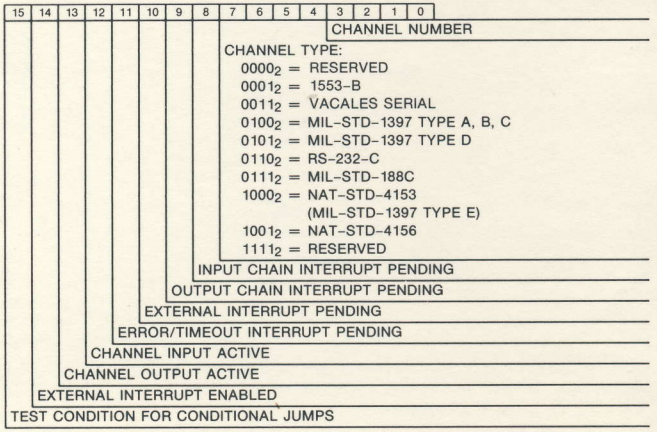
PS = 1 – Use page register set 2 if the channel number of the group is less than 8; otherwise use page register set 3.

B = 0 – Most significant byte will be used when performing 8-bit transfers.

B = 1 – Least significant byte will be used when performing 8-bit transfers. The B-bit changes state as each byte transfers.

(1) RS-232-C/MIL-STD-188C only

I/O STATUS WORD



STATUS WORD INTERPRETATION

WORD BIT	MIL-STD-188C FUNCTION	RS-232-C FUNCTION	MIL-STD-188C AND RS-232-C DESCRIPTION
2^0	PARITY ERROR	PARITY ERROR SERIAL CHANNEL DETECTS A PARITY ERROR ON AN INPUT WORD.	
2^1	OVERRUN	OVERRUN	SERIAL CHANNEL DOES NOT STORE AN INPUT WORD BEFORE ANOTHER IS TRANSMITTED.
2^2	BREAK	BREAK	SERIAL CHANNEL DOES NOT DETECT A STOP-BIT. (USED IN ASYNCHRONOUS MODE ONLY)
2^3	E ACTIVE	CLEAR TO SEND	LINE IS SET "ACTIVE" BY AN EXTERNAL EQUIPMENT.

MIL-STD-1397 PARALLEL OPERATING MODES

MODE REGISTER						MODE OF OPERATION
15 - 5	4	3	2	1	0	
	0	0	0	0	0	COMPUTER TO PERIPHERAL 16-BIT
	0	0	0	0	1	
	0	0	0	1	0	
	0	0	0	1	1	
	0	0	1	0	0	
	0	0	1	0	1	
	0	0	1	1	0	
	0	0	1	1	1	
	0	1	0	0	0	COMPUTER TO PERIPHERAL - 16-BIT
	0	1	0	0	1	COMPUTER TO COMPUTER - 16-BIT
	0	1	0	1	0	UNDEFINED
	0	1	0	1	1	TEST MODE - 16-BIT
	0	1	1	0	0	COMPUTER TO PERIPHERAL - 32-BIT
	0	1	1	0	1	COMPUTER TO COMPUTER - 32-BIT
	0	1	1	1	0	EXTERNALLY SPECIFIED ADDRESSING
	0	1	1	1	1	UNDEFINED TEST MODE - 32-BIT
	1	1	0	0	0	PERIPHERAL INPUT CHANNEL (PIC) - 16-BIT
	1	1	1	0	0	PERIPHERAL INPUT CHANNEL (PIC) - 32-BIT
RESERVED						

MIL-STD-188C AND RS-232-C OPERATING MODES

15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | REGISTER BITS INTERPRETED

IF BIT 3 = 0 (NO PARITY)
00 ⇒ 5-BIT CHARACTER
01 ⇒ 6-BIT CHARACTER
10 ⇒ 7-BIT CHARACTER
11 ⇒ 8-BIT CHARACTER

IF BIT 3 = 1 (INCLUDES PARITY)
00 ⇒ 6-BIT CHARACTER
01 ⇒ 7-BIT CHARACTER
10 ⇒ 8-BIT CHARACTER
11 ⇒ 9-BIT CHARACTER

0 ⇒ SELECT ODD PARITY
1 ⇒ SELECT EVEN PARITY

0 ⇒ DISABLE PARITY CHECKING
1 ⇒ ENABLE PARITY CHECKING

0 ⇒ ONE STOP-BIT ASYNCHRONOUS
1 ⇒ TWO STOP-BITS OUTPUT

0 ⇒ SYNCHRONOUS CHANNEL OPERATION(1)
1 ⇒ ASYNCHRONOUS CHANNEL OPERATION(1)

0 ⇒ RS-232-C OPERATION(1)
1 ⇒ MIL-STD-188C OPERATION(1)

ASYNCHRONOUS CLOCK SPEED SELECTION

00	RESERVED	10_8	9600 BAUD
01	RESERVED	11_8	4800 BAUD
02	50 BAUD	12_8	1800 BAUD
03	75 BAUD	13_8	1200 BAUD
04	134.5 BAUD	14_8	2400 BAUD
05	200 BAUD	15_8	300 BAUD
06	600 BAUD	16_8	150 BAUD
07	2400 BAUD	17_8	110 BAUD

MUST BE ZERO

RESERVED

(1) Set by hardware

VACALES OPERATING MODES

15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0

NOT USED

0 ⇒ SELECT ODD PARITY
1 ⇒ SELECT EVEN PARITY

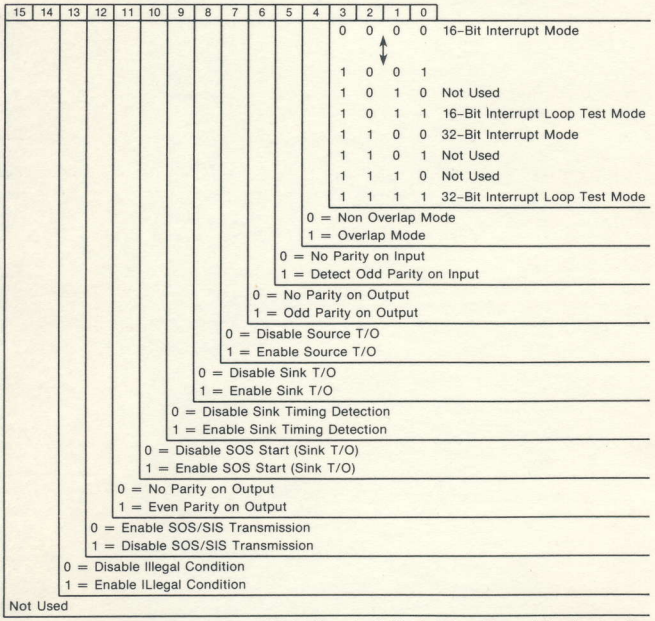
0 ⇒ DISABLE PARITY CHECKING
1 ⇒ ENABLE PARITY CHECKING

RESERVED

1 ⇒ VACALES 0 ⇒ NOT VACALES

0000 ⇒ 1-BIT CHARACTER
1111 ⇒ 16-BIT CHARACTER

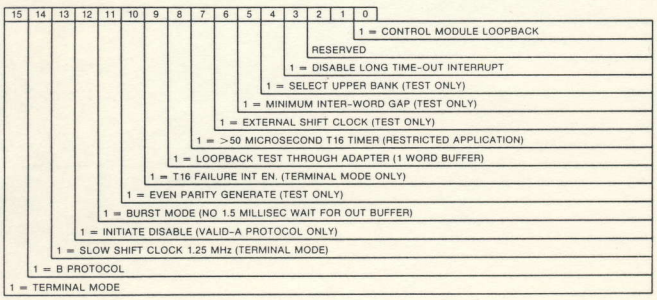
MIL-STD-1397 TYPE D AND NAT-STD-4153 (MIL-STD-1397 TYPE E) AND OPERATING MODES



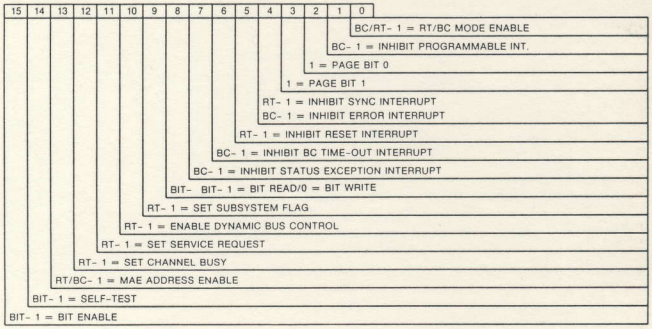
NOTE: All information transfers contain a 32-bit information field. For I/O and External Function transfers the number of valid data bits within this 32-bit field may be 8, 16 or 32. Selection is made by the Transfer Mode (TM) field in the Buffer Control Word (BCW) of the Initiate Transfer Instruction.

NOTE: For External Interrupt Transfers the 32-bit field may contain either 16 or 32 valid data bits. Selection is made by bits 0 through 3 (Mode Bits) of I/O Control Memory location 12_8 of the associated I/O channel.

NATO-STD-4156 SERIAL-OPERATING MODES



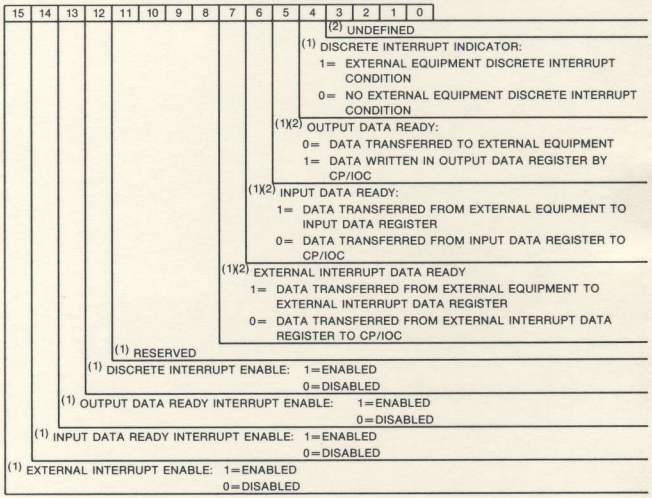
MIL-STD-1553B SERIAL-OPERATING MODES



MEMORY MAPPED INPUT/OUTPUT CONTROL AND STATUS REGISTER

SET BY CP/IOC, CLEARED BY MMIO WHEN BUS INITIALIZATION SIGNAL OCCURS

SET AND CLEARED BY MMIO WHEN CONDITION OCCURS; CLEARED BY MMIO WHEN BUS INITIALIZATION SIGNAL OCCURS



NOTES: (1) NOT MODIFIABLE BY EXTERNAL EQUIPMENT
(2) NOT MODIFIABLE BY CP OR IOC

MMIO MAIN MEMORY ADDRESS ASSIGNMENTS ARE LIMITED TO 0-8K. EACH MMIO CHANNEL REQUIRES FOUR CONSECUTIVE LOCATIONS. MEMORY ADDRESS ASSIGNMENTS ARE HARDWIRED PER USER DEFINITIONS. MMIO EXTERNAL/INTERRUPTS USE THE CLASS III INTERRUPT ENTRANCE ADDRESS.

MEMORY MAPPED INPUT/OUTPUT ASSIGNED ADDRESSES

ADDRESS	X	-	EXTERNAL INTERRUPT WORD
	x+1	-	INPUT DATA WORD
	x+2	-	OUTPUT DATA WORD
	x+3	-	MMIO CONTROL/STATUS WORD