AN/UYK

AN/UYK-44(V)

DATA PROCESSING SET

REFERENCE

GUIDE

SPERRY

NAVAL SEA SYSTEMS COMMAND

PX 14152

5/83

AN/UYK-44(V)

DATA PROCESSING SET

REFERENCE

GUIDE

SPERRY

SYMBOL CONVENTION

o	The operation code of an instruction.
f	The format code designator of an instruction or a subfunction designator.
a	The a-designator of an instruction.
m	The m-designator of an instruction.
am	The combined a- and m-designators used to denote an I/O controller control memory address.
j	The j-designator of an indirect word.
x	The x-designator of an indirect word.
y	The 16-bit operand of a 32-bit instruction.
d	The two's complement number from bits 7-0 of an RI Type 1 instruction, with bit 7 extended to fill.
R_a	The general register designated by a.
R_m	The general register designated by m.
R_m*	The general register designated by m in the inactive set of general registers.
()	The contents of. For example, (R_a) refers to the contents of the general register designated by a.
→	Is transmitted to. For example, $(R_m) \rightarrow R_a$ means the contents of R_m is transmitted to R_a.
+	Addition.
−	Subtraction.
*	Multiplication.
/	Division.
⊙	Logical AND (Logical Product).
⊕	Logical OR (Logical Sum).
⊘	Logical Exclusive OR (Logical Difference).
:	Arithmetic comparison.
::	Logical comparison.
P	The program address register.
Y*	The effective operand address contained in R_m. R_0 may be used for R_m.
Y**	A 32-bit physical address operand.
Y	Effective operand of 32-bit instructions except for RX format jump instruction whose effective operand is (Y).
	Y=y+(R_m), for all but R_0.
	Y=y, for R_0 only.
	Y=(P)+d, sign extended to 16 bits.
$Y_{Rm,Rm+1}$	The RX address formed using the 32-bit value of (Rm,Rm+1) in place of (Rm) as the index.
⌝	Executive mode instruction.
#	Optional MATH PAC instruction.
*	r = bits 0-7 of R_a, u = bits 8-15 of R_a.

CPU REPERTOIRE

HEXADECIMAL FORMAT OP	a	m	OCTAL FORMAT o	f	a	m	CODING FORMAT	INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9-8 CC
00		1	00	0			01 ICP a	Initiate CP Bit	Execute the CP Bit Subtest specified by $(R_a)_{7-0}$	–	–	–
00	0	2	00	0	0	0	02 SRM	RAM BIT Sig to 7B-7D	BIT Signature to CP Control Memory 7B thru 7D	–	–	–
00	0	3	00	0	0	0	03 LRM	7B-7D to RAM BIT Sig	CP Control Memory 7B thru 7D to BIT Signature	–	–	–
00	0	4	00	0	0	0	04 IMP	Initiate MP BIT	Execute maintenance panel subtest	–	–	–
00	0	5	00	0	0	0	05 IDS	Initiate/Update Deadstick	Activate/reset the deadstick timer	–	–	–
02	a	m	00	2	a	m	SPT a,y,m	Stack Put Top	(Y)→(R_a); (R_a)→Y	–	–	–
03	a	m	00	3	a	m	BL a,y,m	Byte Load	$(Y)_{byte} \rightarrow R_a$	0	0	X
04	a	m	01	0	a	m	LR a,m	Load (Register)	$(R_m) \rightarrow R_a$	0	0	X
05	a	m	01	1	a	m	LI a,m	Load (Indirect)	$(Y^*) \rightarrow R_a$	0	0	X
06	a	m	01	2	a	m	LK a,y,m	Load (Constant)	$Y \rightarrow R_a$	0	0	X
07	a	m	01	3	a	m	L a,y,m	Load	$(Y) \rightarrow R_a$	0	0	X
08	a	0	02	0	a	00	PR a	Make Positive (Register)	If $(R_a)<0$, $0-(R_a) \rightarrow R_a$ If $(R_a) \geq 0$, (R_a) unchanged	X 0	X 0	X X
08	a	1	02	0	a	01	NR a	Make Negative (Register)	If $(R_a) \geq 0$, $0-(R_a) \rightarrow R_a$ If $(R_a)<0$, (R_a) unchanged	X 0	0 0	X X
08	a	2	02	0	a	02	RR a	Round (Register)	$(R_a)+(R_{a\oplus1})_{15} \rightarrow R_a$	X	X	X
08	a	4	02	0	a	04	TCR a	Two's Complement	$0-(R_a) \rightarrow R_a$	X	X	X
08	a	5	02	0	a	05	TCDR a	Two's Complement Double (Register)	$0-(R_a, R_{a\oplus1}) \rightarrow R_a, R_{a\oplus1}$	X	X	X
08	a	6	02	0	a	06	OCR a	One's Complement	$\overline{FFFF}\oplus(R_a) \rightarrow R_a$	0	0	X
08	a	8	02	0	a	10	IROR a	Increase by 1 (Register)	$(R_a)+1 \rightarrow R_a$	X	X	X
08	a	9	02	0	a	11	DROR a	Decrease by 1 (Register)	$(R_a)-1 \rightarrow R_a$	X	X	X
08	a	A	02	0	a	12	IRTR a	Increase by 2 (Register)	$(R_a)+2 \rightarrow R_a$	X	X	X
08	a	B	02	0	a	13	DRTR a	Decrease by 2 (Register)	$(R_a)-2 \rightarrow R_a$	X	X	X
08	a	D	02	0	a	15	EBIT a	⌉ Execute BIT	$(R15) \rightarrow up$	–	–	–
09	a	m	02	1	a	m	LDI a,m	Load Double (Indirect)	$(Y^*,Y^*\oplus1) \rightarrow R_a, R_{a\oplus1}$	0	0	X
0B	a	m	02	3	a	m	LD a,y,m	Load Double (Index)	$(Y,Y\oplus1) \rightarrow R_a, R_{a\oplus1}$	0	0	X
0C	a	0	03	0	a	00	ER a	Executive Return (Register)	If Class II interrupts enabled, $(P)+1 \rightarrow R_a$	0	0	X
0C	a	1	03	0	a	01	SSOR a	Store Status Register 1 (Register)	$(SR1) \rightarrow R_a$	0	0	X
0C	a	2	03	0	a	02	SSTR a	Store Status Register 2 (Register)	$(SR2) \rightarrow R_a$	0	0	X
0C	a	3	03	0	a	03	SCR a	Store Real Time Clock Lower (Register)	$(RTC)_{15-0} \rightarrow R_a$	0	0	X
0C	a	4	03	0	a	04	LPR a	Load P Register	$(R_a) \rightarrow P$	–	–	–
0C	a	5	03	0	a	05	LSOR a	⌉ Load Status Register 1 (Register)	$(R_a) \rightarrow SR1$	–	–	–
0C	a	6	03	0	a	06	LSTR a	⌉ Load Status Register 2 (Register)	$(R_a) \rightarrow SR2$	–	–	–
0C	a	7	03	0	a	07	LCR a	⌉ Load Real Time Clock Lower (Register)	$(R_a) \rightarrow RTC_{15-0}$	–	–	–
0C	0	8	03	0	00	10	ECR	⌉ Enable Real Time Clock Count and Interrupt	Enable RTC Count and Overflow Interrupt	–	–	–
0C	0	9	03	0	00	11	DCR	⌉ Disable Real Time Clock Count and Interrupt	Disable RTC Count and Overflow Interrupt	–	–	–
0C	a	A	03	0	a	12	LEM a	⌉ Load and Enable Monitor Clock and Interrupt	$(R_a) \rightarrow$ MC Register; Enable Count and Interrupt	–	–	–
0C	0	B	03	0	00	13	DM	⌉ Disable Monitor Clock Count	Disable MC Count and Interrupt	–	–	–
0C	a	C	03	0	a	14	LCRD a	⌉ Load Real Time Clock Double and Enable Count (Register)	$(R_a, R_{a\oplus1}) \rightarrow$ RTC and Enable Count	–	–	–
0C	a	D	03	0	a	15	SCRD a	Store Real Time Clock Double (Register)	$(RTC) \rightarrow R_a, R_{a\oplus1}$	0	0	X
0C	0	E	03	0	00	16	ECIR	⌉ Enable Real Time Clock Overflow Interrupt	Enable RTC Overflow Interrupt	–	–	–
0C	0	F	03	0	00	17	DCIR	⌉ Disable Real Time Clock Overflow Interrupt	Disable RTC Overflow Interrupt	–	–	–
0F	a	m	03	3	a	m	LM a,y,m	Load Multiple	If m ≥ a; $(Y...Y+m-a) \rightarrow R_a...R_m$ If m < a; $(Y...Y+m-a+16) \rightarrow R_a...R_{15}$	–	–	–
10	a	0	04	0	a	00	SQR a	# Square Root	$\sqrt{(R_a, R_{a\oplus1})} \rightarrow R_a$; Res.→$R_a$	0	X	X
10	a	1	04	0	a	01	RVR a	Reverse Register (Register)	Reverse order of bits in R_a	0	0	X
10	a	2	04	0	a	02	CNT a	Count Ones (Register)	Number of binary ones in $R_a \rightarrow R_{a+1}$	–	–	–
10	a	3	04	0	a	03	SFR a	Scale Factor (P⁻¹ister)	Shift $(R_a, R_{a\oplus1})$ left until $(R_a)_{15} \neq (R_a)_{14}$, zero fill; shift count →$R_{a\oplus1}$+1[1]	–	–	–
10	a	4	04	0	a	04	SMC a	Store Monitor C k	Monitor Clock → R_a	–	–	–
10	a	5	04	0	a	05	SQRT a	# Floating Point S· re Root	$\sqrt{(R_a, R_{a\oplus1})} \rightarrow R_a, R_{a+1}$	0	X	X
10	a	6	04	0	a	06	LCEP a	⌉ Load Clock Enat Periodic	$(R_a) \rightarrow RTC_{15-0}$ and enable interrupt; upon interrupt, $(R_{a+1}) \rightarrow RTC_{15-0}$	–	–	–
10	a	8	04	0	a	10	IS	⌉ Initialize System		0	0	0
10	a	9	04	0	a	11	IB	⌉ Initialize Bus		–	–	–
12	a	m	04	2	a	m	QPT a,y,m	Queue Put Top	(Y)→(R_a),(R_a)→Y,if (Y)=0 then (R_a)→Y⊕1	–	–	–
13	a	m	04	3	a	m	BLX a,y,m	Byte Load and Index by 1	$(Y)_{byte} \rightarrow R_{a\ 7-0}$; $0 \rightarrow R_{a\ 15-8}$ $(Rm)+1 \rightarrow R_m$	0	0	X
14	a	m	05	0	a	m	SBR a,m	Set Bit (Register)	$1 \rightarrow (R_a)_m$	0	0	X
15	a	m	05	1	a	m	LXI a,m	Load and Index by 1 (Indirect)	$(Y^*) \rightarrow R_a$;$(R_m)+1 \rightarrow R_m$ if a ≠ m	0	0	X
16	a	m	05	2	a	m	QPB a,y,m	Queue Put Bottom	$(R_a) \rightarrow (Y\oplus1)$,$(Ra) \rightarrow Y\oplus1$, $0 \rightarrow (Ra)$	–	–	–
17	a	m	05	3	a	m	LX a,y,m	Load and Index by 1 (Index)	$(Y) \rightarrow R_a$;$(R_m)+1 \rightarrow R_m$	0	0	X
18	a	m	06	0	a	m	ZBR a,m	Zero Bit (Register)	$0 \rightarrow (R_a)_m$	0	0	X
19	a	m	06	1	a	m	LDXI a,m	Load Double Index by 2 (Indirect)	$(Y^*, Y^*\oplus1) \rightarrow R_a, R_{a\oplus1}$; $(R_m)+2 \rightarrow R_m$	0	0	X
1A	a	m	06	2	a	m	SGT a,y,m	Stack Get Top	$(Y) \rightarrow R_a$; if (Y) ≠ 0 then ((Y)) → Y and (P)+3→P; if (Y)=0, then (P)+2→P	–	–	–
1B	a	m	06	3	a	m	LDX a,y,m	Load Double and Index by 2 (Index)	$(Y,Y\oplus1) \rightarrow R_a, R_{a\oplus1}$; $(R_m)+2 \rightarrow R_m$	0	0	X
1C	a	m	07	0	a	m	CBR a,m	Compare Bit to Zero (Register)	$(R_a)m$:0	0	0	X
1D	0	m	07	1	00	m	LPI m	⌉ Load Program Status Words (Indirect)	$(Y^*, Y^*+1, Y^*+2) \rightarrow P$, SR1, SR2	–	–	–
1E	a	m	07	2	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	–	–	–
1F	0	m	07	3	00	m	LP y,m	⌉ Load Program Status Words (Index)	(Y, Y+1, Y+2)→P, SR1, SR2	–	–	–
20	a	m	10	0	a	m	LRSR a,m	Logical Right Single Shift (Register)	Shift (R_a) right $(R_m)_{5-0}$ places, zero fill	0	0	X
22	a	m	10	2	a	m	LRS a,y,m	Logical Right Single Shift (Constant)	Shift (R_a) right Y_{5-0} places, zero fill	0	0	X
23	a	m	10	3	a	m	BS a,y,m	Byte Store (Index)	$(R_a)_{7-0} \rightarrow Y_{byte}$	–	–	–
24	a	m	11	0	a	m	ARSR a,m	Algebraic Right Single Shift (Register)	Shift (R_a) right $(R_m)_{5-0}$ places, sign fill	0	0	X
25	a	m	11	1	a	m	SI a,m	Store (Indirect)	$(R_a) \rightarrow Y^*$	–	–	–

(1) Count=32 for all zeros; 31 for all ones

CPU REPERTOIRE (CONT.)

HEXADECIMAL FORMAT			OCTAL FORMAT			CODING FORMAT					SR1 BITS 11	10	9–8
OP	a	m	o	f	a	m	INSTRUCTION	OPERATION	C	OV	CC
26	a	m	11	2	a	m	ARS a,y,m	Algebraic Right Single Shift (Constant)	Shift (R_a) right Y_{5-0} places, sign fill	0	0	X
27	a	m	11	3	a	m	S a,y,m	Store (Index)	$(R_a) \rightarrow Y$	–	–	–
28	a	m	12	0	a	m	LRDR a,m	Logical Right Double Shift (Register)	Shift $(R_a,R_{a\oplus1})$ right $(R_m)_{5-0}$ places, zero fill	0	0	X
29	a	m	12	1	a	m	SDI a,m	Store Double (Indirect)	$(R_a,R_{a\oplus1}) \rightarrow Y^*,Y^*\oplus1$	–	–	–
2A	a	m	12	2	a	m	LRD a,y,m	Logical Right Double Shift (Constant)	Shift $(R_a,R_{a\oplus1})$ right Y_{5-0} places, zero fill	0	0	X
2B	a	m	12	3	a	m	SD a,y,m	Store Double (Index)	$(R_a,R_{a\oplus1}) \rightarrow Y, Y\oplus1$	–	–	–
2C	a	m	13	0	a	m	ARDR a,m	Algebraic Right Double Shift (Register)	Shift $(R_a,R_{a\oplus1})$ right $(R_m)_{5-0}$ places, sign fill	0	0	X
2E	a	m	13	2	a	m	ARD a,y,m	Algebraic Right Double Shift (Constant)	Shift $(R_a,R_{a\oplus1})$ right Y_{5-0} places, sign fill	0	0	X
2F	a	m	13	3	a	m	SM a,y,m	Store Multiple	If m ≥ a; $(R_a...R_m) \rightarrow Y...Y+m-a$; If m < a; $(R_a...R_m) \rightarrow Y...Y+m-a+16$	–	–	–
30	a	m	14	0	a	m	ALSR a,m	Algebraic Left Single Shift (Register)	Shift (R_a) left $(R_m)_{5-0}$ places, zero fill	0	X	X
32	a	m	14	2	a	m	ALS a,y,m	Algebraic Left Single Shift (Constant)	Shift (R_a) left Y_{5-0} places, zero fill	0	X	X
33	a	m	14	3	a	m	BSX a,y,m	Byte Store and Index by 1 (Index)	$(R_a)_{7-0} \rightarrow Y_{byte}$; $(R_m)+1 \rightarrow R_m$	–	–	–
34	a	m	15	0	a	m	CLSR a,m	Circular Left Single Shift (Register)	Shift (R_a) left circularly $(R_m)_{5-0}$ places	0	0	X
35	a	m	15	1	a	m	SXI a,m	Store and Index by 1 (Indirect)	$(R_a) \rightarrow Y^*$; $(R_m) + 1 \rightarrow R_m$	–	–	–
36	a	m	15	2	a	m	CLS a,y,m	Circular Left Single Shift (Constant)	Shift (R_a) left circularly Y_{5-0} places	0	0	X
37	a	m	15	3	a	m	SX a,y,m	Store and Index by 1 (Index)	$(R_a) \rightarrow Y$; $(R_m) + 1 \rightarrow R_m$	–	–	–
38	a	m	16	0	a	m	ALDR a,m	Algebraic Left Double Shift (Register)	Shift $(R_a,R_{a\oplus1})$ left $(R_m)_{5-0}$ places, zero fill	0	X	X
39	a	m	16	1	a	m	SDXI a,m	Store Double and Index by 2 (Indirect)	$(R_a,R_{a\oplus1}) \rightarrow Y^*, Y^*\oplus1$; $(R_m) + 2 \rightarrow R_m$	–	–	–
3A	a	m	16	2	a	m	ALD a,y,m	Algebraic Left Double Shift (Constant)	Shift $(R_a,R_{a\oplus1})$ left Y_{5-0} places, zero fill	0	X	X
3B	a	m	16	3	a	m	SDX a,y,m	Store Double and Index by 2 (Index)	$(R_a,R_{a\oplus1}) \rightarrow Y,Y\oplus1$; $(R_m)+2 \rightarrow R_m$	–	–	–
3C	a	m	17	0	a	m	CLDR a,m	Circular Left Double Shift (Register)	Shift $(R_a,R_{a\oplus1})$ left circularly $(R_m)_{5-0}$ places	0	0	X
3D	0	m	17	1	00	m	SZI m	Store Zero (Indirect)	$0 \rightarrow Y^*$	–	–	–
3E	a	m	17	2	a	m	CLD a,y,m	Circular Left Double Shift (Constant)	Shift $(R_a,R_{a\oplus1})$ left circularly Y_{5-0} places	0	0	X
3F	0	m	17	3	00	m	SZ y,m	Store Zeros (Index)	$0 \rightarrow Y$	–	–	–
40	a	m	20	0	a	m	SUR a,m	Subtract (Register)	$(R_a) - (R_m) \rightarrow R_a$	X	X	X
41	a	m	20	1	a	m	SUI a,m	Subtract (Indirect)	$(R_a) - (Y^*) \rightarrow R_a$	X	X	X
42	a	m	20	2	a	m	SUK a,y,m	Subtract (Constant)	$(R_a) - Y \rightarrow R_a$	X	X	X
43	a	m	20	3	a	m	SU a,y,m	Subtract (Index)	$(R_a) - (Y) \rightarrow R_a$	X	X	X
44	a	m	21	0	a	m	SUDR a,m	Subtract Double (Register)	$(R_a,R_{a\oplus1}) - (R_m,R_{m\oplus1}) \rightarrow R_a,R_{a\oplus1}$	X	X	X
45	a	m	21	1	a	m	SUDI a,m	Subtract Double (Indirect)	$(R_a,R_{a\oplus1}) - (Y^*,Y^*\oplus1) \rightarrow R_a,R_{a\oplus1}$	X	X	X
47	a	m	21	3	a	m	SUD a,y,m	Subtract Double (Index)	$(R_a,R_{a\oplus1}) - (Y,Y\oplus1) \rightarrow R_a,R_{a\oplus1}$	X	X	X
48	a	m	22	0	a	m	AR a,m	Add (Register)	$(R_a) + (R_m) \rightarrow R_a$	X	X	X
49	a	m	22	1	a	m	AI a,m	Add (Indirect)	$(R_a)+(Y^*) \rightarrow R_a$	X	X	X
4A	a	m	22	2	a	m	AK a,y,m	Add (Constant)	$(R_a)+Y \rightarrow R_a$	X	X	X
4B	a	m	22	3	a	m	A a,y,m	Add (Index)	$(R_a)+(Y) \rightarrow R_a$	X	X	X
4C	a	m	23	0	a	m	ADR a,m	Add Double (Register)	$(R_a,R_{a\oplus1}) + (R_m,R_{m\oplus1}) \rightarrow R_a,R_{a\oplus1}$	X	X	X
4D	a	m	23	1	a	m	ADI a,m	Add Double (Indirect)	$(R_a,R_{a\oplus1}) + (Y^*,Y^*\oplus1) \rightarrow R_a,R_{a\oplus1}$	X	X	X
4F	a	m	23	3	a	m	AD a,y,m	Add Double (Index)	$(R_a,R_{a\oplus1}) + (Y,Y\oplus1) \rightarrow R_a,R_{a\oplus1}$	X	X	X
50	a	m	24	0	a	m	CR a,m	Compare (Register)	$(R_a) : (R_m)$	X	X	X
51	a	m	24	1	a	m	CI a,m	Compare (Indirect)	$(R_a) : (Y^*)$	X	X	X
52	a	m	24	2	a	m	CK a,y,m	Compare (Constant)	$(R_a) : Y$	X	X	X
53	a	m	24	3	a	m	C a,y,m	Compare (Index)	$(R_a) : (Y)$	X	X	X
54	a	m	25	0	a	m	CDR a,m	Compare Double (Register)	$(R_a,R_{a\oplus1}) : (R_m,R_{m\oplus1})$	X	X	X
55	a	m	25	1	a	m	CDI a,m	Compare Double (Indirect)	$(R_a,R_{a\oplus1}) : (Y^*,Y^*\oplus1)$	X	X	X
57	a	m	25	3	a	m	CD a,y,m	Compare Double (Index)	$(R_a,R_{a\oplus1}) : (Y,Y\oplus1)$	X	X	X
58	a	m	26	0	a	m	MR a,m	Multiply (Register)	$(R_{a\oplus1})*(R_m) \rightarrow R_a,R_{a\oplus1}$	0	0	X
59	a	m	26	1	a	m	MI a,m	Multiply (Indirect)	$(R_{a\oplus1})*(Y^*) \rightarrow R_a,R_{a\oplus1}$	0	0	X
5A	a	m	26	2	a	m	MK a,y,m	Multiply (Constant)	$(R_{a+1})*Y \rightarrow R_a,R_{a\oplus1}$	0	0	X
5B	a	m	26	3	a	m	M a,y,m	Multiply (Index)	$(R_{a\oplus1})*(Y) \rightarrow R_a,R_{a\oplus1}$	0	0	X
5C	a	m	27	0	a	m	DR a,m	Divide (Register)	$(R_a,R_{a\oplus1})/(R_m) \rightarrow R_{a\oplus1}$; Rem. $\rightarrow R_a$	0	X	X
5D	a	m	27	1	a	m	DI a,m	Divide (Indirect)	$(R_a,R_{a\oplus1})/(Y^*) \rightarrow R_{a+1}$; Rem. $\rightarrow R_a$	0	X	X
5E	a	m	27	2	a	m	DK a,y,m	Divide (Constant)	$(R_a,\ R_{a\oplus1})/Y \rightarrow R_{a\oplus1}$; Rem. $\rightarrow R_a$	0	X	X
5F	a	m	27	3	a	m	D a,y,m	Divide (Index)	$(R_a,R_{a\oplus1})/(Y) \rightarrow R_{a\oplus1}$ Rem. $\rightarrow R_a$	0	X	X
60	a	m	30	0	a	m	ANDR a,m	AND (Register)	$(R_a) \odot (R_m) \rightarrow R_a$	0	0	X
61	a	m	30	1	a	m	ANDI a,m	AND (Indirect)	$(R_a) \odot (Y^*) \rightarrow R_a$	0	0	X
62	a	m	30	2	a	m	ANDK a,y,m	AND (Constant)	$(R_a) \odot Y \rightarrow R_a$	0	0	X
63	a	m	30	3	a	m	AND a,y,m	AND (Index)	$(R_a) \odot (Y) \rightarrow R_a$	0	0	X
64	a	m	31	0	a	m	ORR a,m	OR (Register)	$(R_a) \otimes (R_m) \rightarrow R_a$	0	0	X
65	a	m	31	1	a	m	ORI a,m	OR (Indirect)	$(R_a) \otimes (Y^*) \rightarrow R_a$	0	0	X
66	a	m	31	2	a	m	ORK a,y,m	OR (Constant)	$(R_a) \otimes Y \rightarrow R_a$	0	0	X
67	a	m	31	3	a	m	OR a,y,m	OR (Index)	$(R_a) \otimes (Y) \rightarrow R_a$	0	0	X
68	a	m	32	0	a	m	XORR a,m	Exclusive OR (Register)	$(R_a) \oplus (R_m) \rightarrow R_a$	0	0	X
69	a	m	32	1	a	m	XORI a,m	Exclusive OR (Indirect)	$(R_a) \oplus (Y^*) \rightarrow R_a$	0	0	X
6A	a	m	32	2	a	m	XORK a,y,m	Exclusive OR (Constant)	$(R_a) \oplus Y \rightarrow R_a$	0	0	X
6B	a	m	32	3	a	m	XOR a,y,m	Exclusive OR (Index)	$(R_a) \oplus (Y) \rightarrow R_a$	0	0	X
6C	a	m	33	0	a	m	MSR a,m	Masked Substitute (Register)	If $(R_{a\oplus1})_n = 1$, $(R_m)_n \rightarrow R_{an}$	0	0	X
6D	a	m	33	1	a	m	MSI a,m	Masked Substitute (Indirect)	If $(R_{a\oplus1})_n = 1$, $(Y^*)_n \rightarrow R_{an}$	0	0	X
6E	a	m	33	2	a	m	MSK a,y,m	Masked Substitute (Constant)	If $(R_{a\oplus1})_n = 1$, $Y_n \rightarrow R_{an}$	0	0	X
6F	a	m	33	3	a	m	MS a,y,m	Masked Substitute (Index)	If $(R_{a\oplus1})_n = 1$, $(Y)_n \rightarrow R_{an}$	0	0	X
70	a	m	34	0	a	m	CMR a,m	Compare Masked (Register)	$(R_a) \odot (R_{a\oplus1}) : (R_m \odot R_{a\oplus1})$	0	0	X
71	a	m	34	1	a	m	CMI a,m	Compare Masked (Indirect)	$(R_a) \odot (R_{a\oplus1}) : (Y^* \odot (R_{a\oplus1})$	0	0	X
72	a	m	34	2	a	m	CMK a,y,m	Compare Masked (Constant)	$(R_a) \odot (R_{a\oplus1}) : Y \odot (R_{a\oplus1})$	0	0	X
73	a	m	34	3	a	m	CM a,y,m	Compare Masked (Index)	$(R_a) \odot (R_{a\oplus1}) : (Y) \odot (R_{a\oplus1})$	0	0	X
74	0	0	35	0	00	00	IOCR	Input/Output Command	Execute I/O command instruction located in 60 and 61(2)	–	–	–
74	a	m	35	0	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)	–	–	–
75	0	m	35	1	00	m	BFI m	Biased Fetch (Indirect)	Set CC upon (Y*), $1 \rightarrow Y^*_{15,14}$	0	0	X
76	0	m	35	2	00	m	RCX y,m	Remote Execute	Execute (Y)	X/0	X/0	X/0

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

CPU REPERTOIRE (CONT.)

HEXADECIMAL FORMAT			OCTAL FORMAT			CODING FORMAT					SR1 BITS 11	10	9–8
OP	a	m	o	f	a	m	INSTRUCTION	OPERATION	C	OV	CC
77	0	m	35	3	00	m	BF y,m	Biased Fetch (Index)	Set CC upon (Y), $1 \rightarrow Y_{15,14}$	0	0	X
78	a	m	36	0	a	m	CLR a,m	Compare Logical (Register)	$(R_a) : (R_m)$	X	X	X
79	a	m	36	1	a	m	CLI a,m	Compare Logical (Indirect)	$(R_a) : (Y^*)$	X	X	X
7A	a	m	36	2	a	m	CLK a,y,m	Compare Logical (Constant)	$(R_a) : Y$	X	X	X
7B	a	m	36	3	a	m	CL a,y,m	Compare Logical (Index)	$(R_a) : (Y)$	X	X	X
7C	a	0	37	0	a	00	VF a	# Trigonometric Vector without correction	$0 \rightarrow R_a$			
									$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=0$	$\frac{\sqrt{(R_{a+1})^2+(R_a)^2}}{.4DBA} \rightarrow R_{a+1}$ $\arctan((R_a)/(R_{a+1})) \rightarrow R_{a+2}$			
7C	a	1	37	0	a	01	RF a	# Trigonometric Rotate without correction	$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=\Theta$	$\frac{(R_a)\cos(R_{a+2})+(R_{a+1})\sin(R_{a+2})}{.4DBA} \rightarrow R_a$ $\frac{(R_{a+1})\cos(R_{a+2})-(R_a)\sin(R_{a+2})}{.4DBA} \rightarrow R_{a+1}$ $0 \rightarrow R_{a+2}$			
7C	a	2	37	0	a	02	VFP a	# Trigonometric Vector	$0 \rightarrow R_a$ $(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=0$	$\sqrt{(R_a)^2+(R_{a+1})^2} \rightarrow R_{a+1}$ $\arctan((R_a)/(R_{a+1})) \rightarrow R_{a+2}$			
7C	a	3	37	0	a	03	RFP a	# Trigonometric Rotate	$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=\Theta$	$(R_a)\cos(R_{a+2})+(R_{a+1})\sin(R_{a+2}) \rightarrow R_a$ $(R_{a+1})\cos(R_{a+2})-(R_a)\sin(R_{a+2}) \rightarrow R_{a+1}$ $0 \rightarrow R_{a+2}$			
7C	a	4	37	0	a	04	VH a	# Hyperbolic Vector without correction	$0 \rightarrow R_a$			
									$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=0$	$\frac{\sqrt{(R_a+1)^2-(R_a)^2}}{1.1A8F} \rightarrow R_{a+1}$ $\operatorname{arctanh}((R_a)/(R_{a+1})) \rightarrow R_{a+2}$			
7C	a	5	37	0	a	05	RH a	# Hyperbolic Rotate without correction	$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=v$	$\frac{(R_a)\cosh(R_{a+2})+(R_{a+1})\sinh(R_{a+2})}{1.1A8F} \rightarrow R_a$ $\frac{(R_a)\sinh(R_{a+2})+(R_{a+1})\cosh(R_{a+2})}{1.1A8F} \rightarrow R_{a+1}$ $0 \rightarrow R_{a+2}$			
7C	a	6	37	0	a	06	VHP a	# Hyperbolic Vector	$0 \rightarrow R_a$ $(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=0$	$\sqrt{(R_a+1)^2-(R_a)^2} \rightarrow R_{a+1}$ $\operatorname{arctanh}(R_a)/(R_{a+1})) \rightarrow R_{a+2}$			
7C	a	7	37	0	a	07	RHP a	# Hyperbolic Rotate	$(R_a)=y$ $(R_{a+1})=x$ $(R_{a+2})=v$	$(R_a)\cosh(R_{a+2})+(R_{a+1})\sinh(R_{a+2}) \rightarrow R_a$ $(R_{a+1})\cosh(R_{a+2})+(R_a)\sinh(R_{a+2}) \rightarrow R_{a+1}$ $0 \rightarrow R_{a+2}$			
7C	a	8	37	0	a	10	FC a,y	# Floating Point Compare	$(R_a,R_{a+1}) : (Y,Y+1)$		0	0	X
7C	a	9	37	0	a	11	FXC a	# Fixed to Floating Point Conversion	$(R_a) \rightarrow Exp.:(R_{a+1}) \rightarrow Man.$		X	X	X
7C	a	A	37	0	a	12	FLC a	# Floating Point to Fixed Single Conversion	Convert (R_a,R_{a+1}), Exp. $\rightarrow R_a$ Man $\rightarrow R_{a+1}$		0	0	X
7C	a	B	37	0	a	13	NF a	# Floating Point Normalize	Normalize (R_a,R_{a+1})		X	X	X
7C	a	E	37	0	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		0	X	X
7C	a	F	37	0	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		0	0	X
7D	a	0	37	1	a	00	SIN a	# Floating Point Sine	$SIN\ (R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	1	37	1	a	01	COS a	# Floating Point Cosine	$COS(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	2	37	1	a	02	TAN a	# Floating Point Tangent	$TAN(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	3	37	1	a	03	ASIN a	# Floating Point Arcsine	$ASIN(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	4	37	1	a	04	ACOS a	# Floating Point Arccosine	$ACOS(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	5	37	1	a	05	ATAN a	# Floating Point Arctangent	$ATAN(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	6	37	1	a	06	EXP a	# Floating Point Exponential	$EXP(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
7D	a	7	37	1	a	07	ALOG a	# Floating Point Natural Log	$ALOG(R_a,R_{a\oplus1}) \rightarrow R_a+R_{a+1}$		0	X	X
80	0	m	40	0	00	m	JER m	Jump Equal	If (CC) indicates = or 0, $(R_m) \rightarrow P$		–	–	–
81	1	m	40	0	01	m	JNER m	Jump Not Equal	If (CC) indicates ≠ or not 0, $(R_m) \rightarrow P$		–	–	–
80	2	m	40	0	02	m	JGER m	Jump Greater or Equal	If (CC) indicates ≥ or +, $(R_m) \rightarrow P$		–	–	–
80	3	m	40	0	03	m	JLSR m	Jump Less	If (CC) indicates < or –, $(R_m) \rightarrow P$		–	–	–
80	4	m	40	0	04	m	JOR m	Jump Overflow	If overflow set, $(R_m) \rightarrow P$		–	–	–
80	5	m	40	0	05	m	JCR m	Jump Carry	If carry set, $(R_m) \rightarrow P$		–	–	–
80	6	m	40	0	06	m	JPTR m	Jump Power Out of Tolerance	If power out of tolerance, $(R_m) \rightarrow P$		–	–	–
80	7	m	40	0	07	m	JBR m	Jump Bootstrap 2 Selected	If bootstrap 2 selected, $(R_m) \rightarrow P$		–	–	–
80	8	m	40	0	10	m	JR m	Jump	$(R_m) \rightarrow P$		–	–	–
80	9	m	40	0	11	m	JSR m	Jump After Stop	Stop; upon restart, $(R_m) \rightarrow P$		–	–	–
80	A	m	40	0	12	m	JKSR 1,m	Jump After Stop Key 1 Set	If key 1 set, stop; $(R_m) \rightarrow P$		–	–	–
80	B	m	40	0	13	m	JKSR 2,m	Jump After Stop Key 2 Set	If key 2 set, stop; $(R_m) \rightarrow P$		–	–	–
81		d	40	1		d	LJ d	Local Jump	$(P)+d \rightarrow P$		–	–	–
81	0→	1	40	1	0→	1	NOP	No Operation (Software)	$(P)+1 \rightarrow P$		–	–	–
2(81	0→	1)	2(40	1	0→	1)	NOPD	No Operation Double (Software)	$(P)+1 \rightarrow P$, $(P)+1 \rightarrow P$		–	–	–
82	0	m	40	2	00	m	JE y,m	Jump Equal	If (CC) indicates = or 0, $Y \rightarrow P$		–	–	–
82	1	m	40	2	01	m	JNE y,m	Jump Not Equal	If (CC) indicates ≠ or not 0, $Y \rightarrow P$		–	–	–
82	2	m	40	2	02	m	JGE y,m	Jump Greater or Equal	If (CC) indicates ≥ or +, $Y \rightarrow P$		–	–	–
82	3	m	40	2	03	m	JLS y,m	Jump Less	If (CC) indicates < or –, $Y \rightarrow P$		–	–	–
82	4	m	40	2	04	m	JO y,m	Jump Overflow	If overflow set, $Y \rightarrow P$		–	–	–
82	5	m	40	2	05	m	JC y,m	Jump Carry	If carry set, $Y \rightarrow P$		–	–	–
82	6	m	40	2	06	m	JPT y,m	Jump Power Out of Tolerance	If power out of tolerance, $Y \rightarrow P$		–	–	–
82	7	m	40	2	07	m	JB y,m	Jump Bootstrap 2 selected	If bootstrap 2 selected, $Y \rightarrow P$		–	–	–
82	8	m	40	2	10	m	J y,m	Jump	$Y \rightarrow P$		–	–	–
82	9	m	40	2	11	m	JSR y,m	Jump After Stop	Stop; upon restart $Y \rightarrow P$		–	–	–
82	A	m	40	2	12	m	JKS 1,y,m	Jump After Stop Key 1 Set	If key 1 set, stop; $Y \rightarrow P$		–	–	–
82	B	m	40	2	13	m	JKS 2,y,m	Jump After Stop Key 2 Set	If key 2 set, stop; $Y \rightarrow P$		–	–	–
83	0	m	40	3	00	m	JE *y,m	Jump Equal	If (CC) indicates = or 0, $(Y) \rightarrow P$		–	–	–
83	1	m	40	3	01	m	JNE *y,m	Jump Not Equal	If (CC) indicates ≠ or not 0, $(Y) \rightarrow P$		–	–	–
83	2	m	40	3	02	m	JGE *y,m	Jump Greater or Equal	If (CC) indicates ≥ or +, $(Y) \rightarrow P$		–	–	–
83	3	m	40	3	03	m	JLS *y,m	Jump Less	If (CC) indicates < or –, $(Y) \rightarrow P$		–	–	–
83	4	m	40	3	04	m	JO *y,m	Jump Overflow	If overflow set, $(Y) \rightarrow P$		–	–	–
83	5	m	40	3	05	m	JC *y,m	Jump Carry	If carry set, $(Y) \rightarrow P$		–	–	–
83	6	m	40	3	06	m	JPT *y,m	Jump Power Out of Tolerance	If power out of tolerance, $(Y) \rightarrow P$		–	–	–
83	7	m	40	3	07	m	JB *y,m	Jump Bootstrap 2 selected	If bootstrap 2 selected, $(Y) \rightarrow P$		–	–	–
83	8	m	40	3	10	m	J *y,m	Jump	$(Y) \rightarrow P$		–	–	–
83	9	m	40	3	11	m	JS *y,m	Jump After Stop	Stop; upon restart, $(Y) \rightarrow P$		–	–	–
83	A	m	40	3	12	m	JKS 1,*y,m	Jump After Stop Key 1 Set	If key 1 set, stop; $(Y) \rightarrow P$		–	–	–

CPU REPERTOIRE (CONT.)

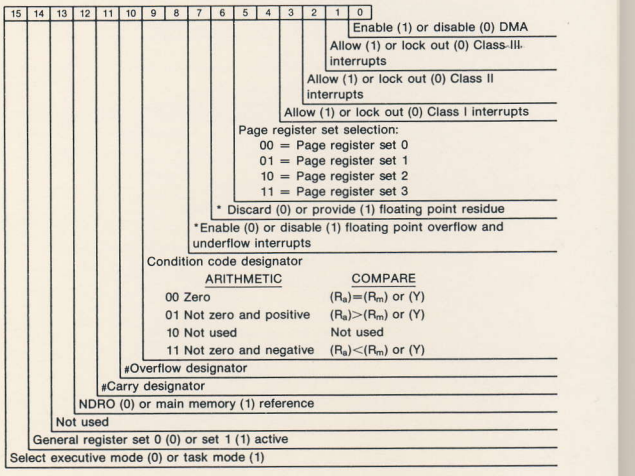
HEXADECIMAL FORMAT OP	a	m	OCTAL FORMAT o	f	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
83	B	m	40	3	13	m	JKS 2,*y,m] Jump After Stop Key 2 Set	If key 2 set, stop; (Y)→P	–	–	–
84	a	m	41	0	a	m	XJR a,m Index Jump	If (R_a) ≠ 0, (R_a) – 1 → R_a, (R_m) → P	–	–	–
85		d	41	1		d	LJI d Local Jump Indirect	((P)+d)→P	–	–	–
86	a	m	41	2	a	m	XJ a,y,m Index Jump	If (R_a) ≠ 0, (R_a) – 1 → R_a, Y→P	–	–	–
87	a	m	41	3	a	m	XJ a,*y,m Index Jump	If (R_a) ≠ 0, (R_a) – 1 → R_a, (Y)→P	–	–	–
88	a	m	42	0	a	m	JLRR a,m Jump, Link Register	(P) + 1 → R_a; (R_m) → P	–	–	–
8A	a	m	42	2	a	m	JLR a,y,m Jump, Link Register	(P) + 2 → R_a; Y → P	–	–	–
8B	a	m	42	3	a	m	JLR a,*y,m Jump, Link Register	(P) + 2 → R_a; (Y)→P	–	–	–
8D		d	43	1		d	LJLM d Local Jump, Link Memory	(P) + 1 → (P) + d; (P) + d + 1 → P	–	–	–
8E	0	m	43	2	00	m	JLM y,m Jump, Link Memory	(P) + 2 → Y; Y + 1 → P	–	–	–
8F	0	m	43	3	00	m	JLM *y,m Jump, Link Memory	(P) + 2 → (Y); (Y) + 1 → P	–	–	–
90	a	m	44	0	a	m	JZR a,m Jump Zero	If (R_a) = 0, (R_m) → P	–	–	–
91		d	44	1		d	LJE d Local Jump Equal	If (CC) indicates = or 0, (P)+d→P	–	–	–
92	a	m	44	2	a	m	JZ a,y,m Jump Zero	If (R_a) = 0, Y→P	–	–	–
93	a	m	44	3	a	m	JZ a,*y,m Jump Zero	If (R_a) = 0, (Y)→P	–	–	–
94	a	m	45	0	a	m	JNZR a,m Jump Not Zero	If (R_a)≠0, (R_m)→P	–	–	–
95		d	45	1		d	LJNE d Local Jump Not Equal	If (CC) indicates ≠ or not 0, (P)+d→P	–	–	–
96	a	m	45	2	a	m	JNZ a,y,m Jump Not Zero	If (R_a) ≠ 0, Y→P	–	–	–
97	a	m	45	3	a	m	JNZ a,*y,m Jump Not Zero	If (R_a) ≠ 0, (Y)→P	–	–	–
98	a	m	46	0	a	m	JPR a,m Jump Positive	If (R_a)≥0, (R_m)→P	–	–	–
99		d	46	1		d	LJGE d Local Jump Greater or Equal	If (CC) indicates ≥ or +, (P)+d→P	–	–	–
9A	a	m	46	2	a	m	JP a,y,m Jump Positive	If (R_a) ≥ 0, Y→P	–	–	–
9B	a	m	46	3	a	m	JP a,*y,m Jump Positive	If (R_a)≥0, (Y)→P	–	–	–
9C	a	m	47	0	a	m	JNR a,m Jump Negative	If (R_a) < 0, (R_m) → P	–	–	–
9D		d	47	1		d	LJLS d Local Jump Less	If (CC) indicates < or –, (P)+d→P	–	–	–
9E	a	m	47	2	a	m	JN a,y,m Jump Negative	If (R_a) < 0, Y→P	–	–	–
9F	a	m	47	3	a	m	JN a,*y,m Jump Negative	If (R_a)<0, (Y)→P	–	–	–
A0	a	m	50	0	a	m	FSUR a,m # Floating Point Subtract (Register)	(R_a, $R_{a\oplus1}$) – (R_m, $R_{m\oplus1}$) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A1	a	m	50	1	a	m	FSUI a,m # Floating Point Subtract (Indirect)	(R_a, $R_{a\oplus1}$) – (Y*, Y*⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A3	a	m	50	3	a	m	FSU a,y,m # Floating Point Subtract (Index)	(R_a, $R_{a\oplus1}$) – (Y, Y⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A4	a	m	51	0	a	m	FAR a,m # Floating Point Add (Register)	(R_a, $R_{a\oplus1}$) + (R_m, $R_{m\oplus1}$) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A5	a	m	51	1	a	m	FAI a,m # Floating Point Add (Indirect)	(R_a, $R_{a\oplus1}$) + (Y*, Y*⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A7	a	m	51	3	a	m	FA a,y,m # Floating Point Add (Index)	(R_a, $R_{a\oplus1}$) + (Y, Y⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A8	a	m	52	0	a	m	FMR a,m # Floating Point Multiply (Register)	(R_a, $R_{a\oplus1}$) * (R_m, $R_{m\oplus1}$)→ R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
A9	a	m	52	1	a	m	FMI a,m # Floating Point Multiply (Indirect)	(R_a, $R_{a\oplus1}$) * (Y*, Y*⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
AB	a	m	52	3	a	m	FM a,y,m # Floating Point Multiply (Index)	(R_a, $R_{a\oplus1}$)*(Y, Y⊕1) → R_a, R_{a+1}; Res.→R_{a+2}, R_{a+3}	0	X	X
AC	a	m	53	0	a	m	FDR a,m # Floating Point Divide (Register)	(R_a, $R_{a\oplus1}$)/(R_m, $R_{m\oplus1}$) → R_a, R_{a+1}; Rem.→ R_{a+2}, R_{a+3}	0	X	X
AD	a	m	53	1	a	m	FDI a,m # Floating Point Divide (Indirect)	(R_a, $R_{a\oplus1}$)/(Y*,Y*⊕1) → R_a, R_{a+1}; Rem.→R_{a+2}, R_{a+3}	0	X	X
AF	a	m	53	3	a	m	FD a,y,m # Floating Point Divide (Index)	(R_a, $R_{a\oplus1}$)/(Y,Y⊕1) → R_a, R_{a+1}; Rem.→R_{a+2}, R_{a+3}	0	X	X
B0	a	m	54	0	a	m	LARR a,m] Load Address Register (Register)	• (R_m) → AR_r	–	–	–
B1	a	m	54	1	a	m	LARI a,m] Load Address Register (Indirect)	• (Y*)→AR_r	–	–	–
B3	a	m	54	3	a	m	LARM a,y,m] Load Address Register Multiple (Index)	• (Y, ..., Y+u)→AR_r,...,AR_{r+u}	–	–	–
B4	a	m	55	0	a	m	SARR a,m Store Address Register (Register)	• (AR_r)→R_m	–	–	–
B5	a	m	55	1	a	m	SARI a,m Store Address Register (Indirect)	• (AR_r)→Y*	–	–	–
B7	a	m	55	3	a	m	SARM a,y,m Store Address Register Multiple (Index)	• (AR_r,...,AR_{r+u}) → Y,..., Y+u	–	–	–
B8	a	m	56	0	a	m	MDR a,m # Multiply Double (Register)	(R_a, $R_{a\oplus1}$)*(R_m, $R_{m\oplus1}$)→R_a, R_{a+1}, R_{a+2}, R_{a+3}	0	0	X
B9	a	m	56	1	a	m	MDI a,m # Multiply Double (Indirect)	(R_a, $R_{a\oplus1}$)*(Y*,Y*⊕1) → R_a, R_{a+1}, R_{a+2}, R_{a+3}	0	0	X
BB	a	m	56	3	a	m	MD a,y,m # Multiply Double (Index)	(R_a, $R_{a\oplus1}$)*(Y, Y⊕1) → R_a, R_{a+1}, R_{a+2}, R_{a+3}	0	0	X
BC	a	m	57	0	a	m	DDR a,m # Divide Double (Register)	(R_a, R_{a+1}, R_{a+2}, R_{a+3})/ (R_m, $R_{m\oplus1}$) → R_{a+2}, R_{a+3}; Rem.→R_a, R_{a+1}	0	X	X
BD	a	m	57	1	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
BF	a	m	57	3	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
C0	a	m	60	0	a	m	LLRS a,m Logical Right Single Shift (Literal)	Shift (R_a) right m places, zero fill	0	0	X
C1	a	m	60	1	a	m	LARS a,m Algebraic Right Single Shift (Literal)	Shift (R_a) right m places, sign fill	0	0	X
C2	a	m	60	2	a	m	LLRD a,m Logical Right Double Shift (Literal)	Shift (R_a, $R_{a\oplus1}$) right m places, zero fill	0	0	X
C3	a	m	60	3	a	m	LARD a,m Algebraic Right Double Shift (Literal)	Shift (R_a, $R_{a\oplus1}$) right m places, sign fill	0	0	X
C4	a	m	61	0	a	m	LALS a,m Algebraic Left Single Shift (Literal)	Shift (R_a) left m places, zero fill	0	X	X
C5	a	m	61	1	a	m	LCLS a,m Circular Left Single Shift (Literal)	Shift (R_a) left circularly m places	0	0	X
C6	a	m	61	2	a	m	LALD a,m Algebraic Left Double Shift (Literal)	Shift (R_a, $R_{a\oplus1}$) left m places, zero fill	0	X	X
C7	a	m	61	3	a	m	LCLD a,m Circular Left Double Shift (Literal)	Shift (R_a, $R_{a\oplus1}$) left circularly m places	0	0	X
C8	a	m	62	0	a	m	LSU a,m Subtract (Literal)	(R_a) – m → R_a	X	X	X
C9	a	m	62	1	a	m	LSUD a,m Subtract Double (Literal)	(R_a, $R_{a\oplus1}$) – m → R_a, $R_{a\oplus1}$	X	X	X
CA	a	m	62	2	a	m	LA a,m Add (Literal)	(R_a)+m→R_a	X	X	X
CB	a	m	62	3	a	m	LAD a,m Add Double (Literal)	(R_a, $R_{a\oplus1}$) + m → R_a, $R_{a\oplus1}$	X	X	X
CC	a	m	63	0	a	m	LL a,m Load (Literal)	m → R_a	0	0	X
CD	a	m	63	1	a	m	LC a,m Compare (Literal)	(R_a) : m	X	X	X
CE	a	m	63	2	a	m	LMUL a,m Multiply (Literal)	($R_{a\oplus1}$) * m → R_a, $R_{a\oplus1}$	0	0	X
CF	a	m	63	3	a	m	LDIV a,m Divide (Literal)	(R_a, $R_{a\oplus1}$)/m → $R_{a\oplus1}$; Rem. → R_a	0	X	X

CPU REPERTOIRE (CONT.)

HEXADECIMAL FORMAT OP	a	m	OCTAL FORMAT o	f	a	m	CODING FORMAT INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9–8 CC
D0	a	m	64	0	a	m	LIR a,m] Load Inter-Register	(R_m*) → R_a[3]	0	0	X
D2	a	m	64	2	a	m	LMR a,y,m Load Multiple-Register	(Y...Y+15-a) → R_a...R_{15}	–	–	–
D3	a	m	64	3	a	m	BSU a,y,m Byte Subtract	(R_a) – (Y)$_{byte}$ → R_a	X	X	X
D4	a	m	65	0	a	m	SIR a,m] Store Inter-Register	(R_a) → R_m*[3]	0	0	X
D6	a	m	65	2	a	m	SMR a,y,m Store Multiple-Register	(R_a...R_{15}) → (Y...Y+15-a)	–	–	–
D7	a	m	65	3	a	m	BA a,y,m Byte Add (Index)	(R_a)+(Y)$_{byte}$ → R_a	X	X	X
DB	a	m	66	3	a	m	BC a,y,m Byte Compare (Index)	(R_a) : (Y)$_{byte}$	X	X	X
DC	a	m	67	0	a	m	UM1 a,m User Macro (Software)	Reserved for user macro instructions	–	–	–
DC	a	m	67	0	a	m	UM2 a,m User Macro (Software)	Reserved for user macro instructions	–	–	–
DD	a	m	67	1	a	m	UMI a,m User Macro (Software)	Reserved for user macro instructions	–	–	–
DE	a	m	67	2	a	m	UMK a,y,m User Macro (Software)	Reserved for user macro instructions	–	–	–
DF	a	m	67	3	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
E0	a	m	70	0	a	m	LPAR a,m] Load Physical Address (Register)	MAP (Y*, (R_{m+1})) → Y**, Y** → R_a, R_{a+1}	0	0	X
E1	a	m	70	1	a	m	LPAI a,m] Load Physical Address (Indirect)	MAP ((Y*), (R_{m+1})) → Y**, Y** → R_a, R_{a+1}	0	0	X
E2	a	m	70	2	a	m	LPAK a,y,m] Load Physical Address (Constant)	MAP (Y, (R_{m+1})) → Y**, Y** → R_a, R_{a+1}	0	0	X
E3	a	m	70	3	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
E7	a	m	71	3	a	m	LMAP a,y,m] Load Mapped	MAP (Y, (R_{m+1})) → Y**, (Y**) → R_a	0	0	X
EB	a	m	72	3	a	m	SMAP a,y,m] Store Mapped	MAP (Y, (R_{m+1})) → Y**, (R_a) → Y**	–	–	–
ED	a	m	73	1	a	m	LPLI a,m] Load Physical Location (Indirect)	(R_m, R_{m+1}) → Y**, (Y**) → R_a	0	0	X
EF	a	m	73	3	a	m	LPL a,y,m] *Load Physical Location (Index)	Y, R_m, R_{m+1} → Y**, (Y**) → R_a	0	0	X
F1	a	m	74	1	a	m	SPLI a,m] Store Physical Location (Indirect)	Y_{Rm}, R_{m+1} → Y**, (R_a) → Y**	–	–	–
F3	a	m	74	3	a	m	SPL a,y,m] *Store Physical Location (Index)	Y_{Rm}, R_{m+1} → Y**, (R_a) → Y**	–	–	–

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

INDIRECT WORD FORMAT

IW 1
IW 2

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

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

3	2	1	0
m			

4-bit m-field of the RL format instructions

Byte Format – 8-bit unsigned integer

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

Single-Length Format

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIGN BIT															

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIGN BIT	CHARACTERISTIC							FRACTION							

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

↑ RADIX POINT

STATUS REGISTER 2 FORMAT

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

Bit 10: Illegal instruction; memory resume error, parity error, protect fault; or floating point (FP) interrupt data*

Bits 9–8: Indirect control bits for R_B
Bits 7–6: Indirect control bits for R_A
Bits 5–4: Indirect control bits for R_C
Bits 3–2: Indirect control bits for R_E

00 Normal Addressing
01 Normal Addressing
10 Indirect Addressing w/o Indexing
11 Indirect Addressing w Indexing

* Bits 7–0 are interpreted as follows:

7	6	5	4	3	2	1	0	
I	I	C	C	C	C	X	X	IOC memory resume error, parity error, instruction fault, or protection fault
0	0	0	0	0	0	1	0	CPU memory resume error, parity error, instruction fault or protection fault
Instruction Register bits 11 – 4								Floating point underflow or overflow interrupt

where: II – The IOC number

CCCC – The channel number, where appropriate; otherwise the a-field of the command

XX – The IOC instruction type, as follows:
00 – Input chain
01 – Output chain
11 – I/O command

INSTRUCTION FORMATS

INSTRUCTION TYPE

RL

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

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

RR
RI, TYPE 2

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

RI, TYPE 1

OP	d

RK, RX

OP	a	m
y		

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

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

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)

CORDIC FUNCTIONS (OPTIONAL MATHPAC INSTRUCTIONS)

HEXADECIMAL FORMAT			OCTAL FORMAT	CODING	FUNCTION	INPUT PARAMETERS			OUTPUT PARAMETERS		
OP	a	m	o f a m	FORMAT		R_s	R_{s+1}	R_{s+2}	Y→R_s	X→R_{s+1}	W→R_{s+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	$2\sqrt{x}$	$W=1/2\ \text{log}_e x$ $=\tanh^{-1}\frac{x-1}{x+1}$
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
I 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_s and R_{s+1} must be the same
The radix point for W = Constant in hyperbolic 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 207C 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. See page 20.

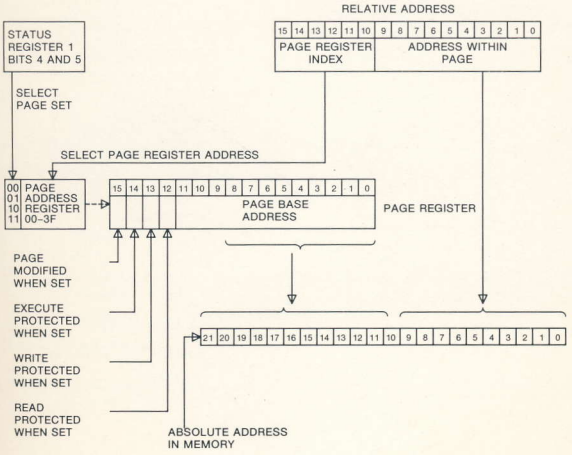
INTERRUPT PRIORITY

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 (35 RR)	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

MEMORY ADDRESS GENERATION



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)
80–6F	EXTERNAL INTERRUPT WORD STORAGE (IOC)

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

MRC DISPLAY

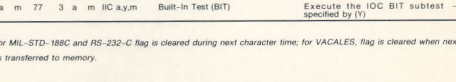
DISPLAY CODE	INFORMATION DISPLAYED	
000	MRC State	AXX– RUN (Program Run) –= blank XFXX PWR (Power Fault) XXFX PROG (Instruction Fault) –XXS STOP – = blank s = STOP condition s = 0 Power up or Master Clear 1 Jump-stop 1 2 Jump-stop 2 3 Unconditional jump-stop 4 Stop key depression 5 Breakpoint stop 6 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	$3\text{–}0_2$ = Register
110–11F	General Register Set 1	$3\text{–}0_2$ = Register
200–2FF	Page Registers, 00–3F	$7\text{–}6_2$ = Page register set $5\text{–}0_2$ = 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	$4\text{–}7_2$ = Channel $3\text{–}0_2$ = Channel memory location
B00–BFF	IOC Channel Status	$4\text{–}7_2$ = Channel $3\text{–}0_2$ = Channel status location
C00–C0F	IOC Output Data	$3\text{–}0_2$ = 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	

INPUT OUTPUT INSTRUCTIONS

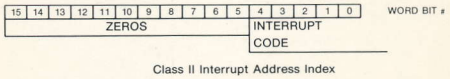
HEXADECIMAL FORMAT OP	a	m	OCTAL FORMAT o	f	a	m	CODING FORMAT	INSTRUCTION	OPERATION	SR1 BITS 11 C	10 OV	9-8 CC
E0	0	0	70	0	00	00	ACR 0	Channel Control	Master clear all channels	–	–	–
E0	0	4	70	0	00	04	ACR 4 CCR 0,4	Channel Control	Enable external interrupts, all channels; Set External Interrupt Enable (EIE) line	–	–	–
E0	0	5	70	0	00	05	ACR 5 CCR 0,5	Channel Control	Disable external interrupts channel a; all channels; Clear External Interrupt Enable (EIE) line	–	–	–
E0	a	6	70	0	a	06	CCR a,6	Enable Selected Interrupts	Enable Class III, Priority 2,3,4 interrupts, channels 0 to a–1	–	–	–
E0	a	7	70	0	a	07	CCR a,7	Disable Selected Interrupts	Disable Class III, Priority 2,3,4 interrupts, channels 0 to a–1	–	–	–
E0	a	8	70	0	a	10	CCR a,8	Channel Control	Master clear, channel a	–	–	–
E0	a	9	70	0	a	11	CCR a,9	Clear Input on Channel a				
E0	a	A	70	0	a	12	CCR a,A	Clear Output on Channel a				
E0	a	C	70	0	a	14	CCR a,C	Channel Control	Enable external interrupts, channel a; Set External Interrupt Enable (EIE) line	–	–	–
E0	a	D	70	0	a	15	CCR a,D	Channel Control	Disable external interrupts,channel a, Clear External Interrupt Enable (EIE) line	–	–	–
E0	a	E	70	0	a	16	CCR a,E	Channel Control	Enable Class III, Priority 2,3,4 interrupts, channel a	–	–	–
E0	a	F	70	0	a	17	CCR a,F	Channel Control	Disable Class III, Priority 2,3,4 interrupts, channel a	–	–	–
								INPUT/OUTPUT INSTRUCTIONS – COMMAND INSTRUCTIONS				
E6	a	2	71	2	a	02	ICK a,y	Initiate Input Chain	Y→IOCM$_D$, initiate input chain	–	–	–
E6	a	6	71	2	a	06	OCK a,y	Initiate Output Chain	Y→IOCM$_C$, initiate output chain	–	–	–
E6	a	m	71	2	a	m	WIMK a,y,m	Write Control Memory	Y→IOCM$_m$, channel a	–	–	–
E6	a	m	71	2	a	m	WCMK am,y	Write Control Memory				
E7	a	m	71	3	a	m	WIM a,y,m	Write Control Memory	(Y)→IOCM$_m$, channel a	–	–	–
EB	a	m	72	3	a	m	RIM a,y,m RCM am,y	Read Control Memory	Channel a, (IOCM$_m$) → Y	–	–	–
F8	a	m	76	0	a	m	SICR a,m	Serial Interface Control	Set or clear serial channel a discretes	–	–	–
FB	a	m	76	3	a	m	SST a,y,m	Store Serial Status	Channel a status bits per m → Y	–	–	–
								INPUT/OUTPUT INSTRUCTIONS – CHAIN INSTRUCTIONS				
E2	a	m	70	2	a	m	LCMI a,y,m	Load Control Memory				
E3	0	0	70	3	00	00	IO 0,y	Input Data	(Y,Y⊕1) → BCW, BAP; initiate transfer	–	–	–
E3	1	0	70	3	01	00	IO 1,y	Output Data	(Y,Y⊕1) → BCW, BAP; initiate transfer	–	–	–
E3	2	0	70	3	02	00	IO 2,y	External Function	(Y,Y⊕1) → BCW, BAP; initiate transfer	–	–	–
E3	3	0	70	3	03	00	IO 3,y	Force External Function	(Y,Y⊕1) → BCW, BAP; initiate transfer	–	–	–
E6	0	m	71	2	00	m	LCMK m,y	Load Control Memory	Y→IOCM$_m$	–	–	–
E7	0	m	71	3	00	m	LCM m,y	Load Control Memory	(Y)→IOCM$_m$	–	–	–
EB	0	m	72	3	00	m	SCM m,y	Store Control Memory	(IOCM$_m$)→Y	–	–	–
EC	0	0	73	0	00	00	HCR	Halt Chain	Halt chaining (chaining)	–	–	–
EC	1	0	73	0	01	00	IPR	Interrupt Processor	Generate chain interrupt (chaining)	–	–	–
EF	0	0	73	3	00	00	ZF y	Zero Flag	0→Y$_{15,14}$	–	–	–
EF	1	0	73	3	01	00	SF y	Set Flag	1→Y$_{15,14}$	–	–	–
EF	2	0	73	3	02	00	TF y	Test and Set Y	0 → Y$_{15,14}$ set condition			
EF	4	m	73	3	04	m	ZB y,m	Clear Bit	0 → Ym			
EF	5	m	73	3	05	m	SB y,m	Set Bit	1 → Ym			
EF	7	m	73	3	07	m	CB y,m	Compare Bit to Zero	Ym:0 set condition			
F2	0	0	74	2	00	00	SJC 0,y	Serial Jump (Unconditional)	Unconditional Y → CAP; clear flag	–	–	–
F2	1	0	74	2	01	00	SJC 1,y	Serial Jump (Conditional)	Serial Jump if suppress flag not set. No jump for MIL-STD-1397 or NAT-STD-4153. (1)			
F2	2	0	74	2	02	00	SJC 2,y	Serial Jump (Conditional)	Serial Jump if monitor flag set. No jump for MIL-STD-1397 or NAT-STD-4153. (1)			
F2	4	0	74	2	04	00	SJMC 4,y	Serial Jump (Conditional)	Jump if condition bit (bit 15) in I/O status word is set.			
F2	8	0	74	2	10	00	SJMC 8,y	Serial Jump (Conditional)	Y → CAP if Input Buffer is active			
F2	9	0	74	2	11	00	SJMC 9,y	Serial Jump (Conditional)	Y → CAP if Output Buffer is active			
F2	A	0	74	2	12	00	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			
F4	0	m	75	0	00	m	SFSC m	Search for sync	Perform functions per m-designator	–	–	–
F8	0	m	76	0	00	m	CSIR m	Serial Interface Control	Set or clear serial channel discrete function	–	–	–
FB	0	m	76	3	00	m	CSST y,m	Store Serial Status	Serial status bit per m → Y	–	–	–
FF	a	m	77	3	a	m	IIC a,y,m	Built-In Test (BIT)	Execute the IOC BIT subtest specified by (Y)	–	–	–

(1) 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.

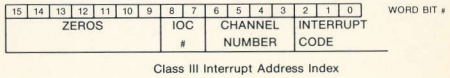
INTERRUPT ENTRANCE ADDRESS INDEX



Class I Interrupt Address Index



Class II Interrupt Address Index



Class III Interrupt Address Index

MIL-STD-1397 PARALLEL OPERATING MODES

MODE REGISTER 15 – 5	4	3	2	1	0	MODE OF OPERATION
	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	0	X	X	X	PERIPHERAL INPUT CHANNEL (PIC)
	1	1	X	X	X	PERIPHERAL INPUT CHANNEL (PIC)
NOT USED						

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

MODE REGISTER 15–4	3	2	1	0	MODE OF OPERATION
	0	0	0	0	OFF
	0	0	0	1	
	0	0	1	0	
	0	0	1	1	
	0	1	0	0	
	0	1	0	1	
	0	1	1	0	
	0	1	1	1	
	1	0	0	0	OFF
	1	0	0	1	COMPUTER TO COMPUTER, LOOP TEST, 16-BIT
	1	0	1	0	COMPUTER TO PERIPHERAL, 16-BIT
	1	0	1	1	COMPUTER TO COMPUTER, 16-BIT
	1	1	0	0	COMPUTER TO PERIPHERAL, 32-BIT
	1	1	0	1	COMPUTER TO COMPUTER, 32-BIT
	1	1	1	0	COMPUTER TO PERIPHERAL, DUAL CHANNEL, 32-BIT
	1	1	1	1	COMPUTER TO COMPUTER, DUAL CHANNEL, 32-BIT
NOT USED					

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
NOT USED																

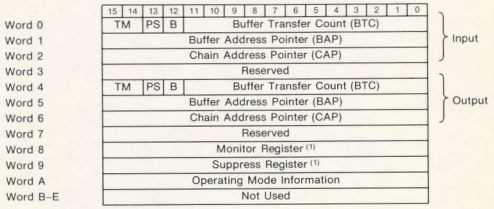
(1) Set by hardware

I/O CONTROL MEMORY ADDRESS SELECTION

a-VALUE	WORD	m-VALUE	MIL-STD-1397 A, B, C	MIL-STD-1397 TYPE D SERIAL NAT-STD-4153	RS-232-C MIL-STD-188C, VACALES	NAT-STD-4156
Channel	0	0	Input BWC	Input BWC	Input BWC	Input BWC
designator	1	1	Input BAP	Input BAP	Input BAP	Input BAP
for	2	2	Input CAP	Input CAP	Input CAP	Input CAP
command	3	3	Reserved*	Reserved*	Reserved*	Reserved*
instructions,	4	4	Output BWC	Output BWC	Output BWC	Output BWC
not used	5	5	Output BAP	Output BAP	Output BAP	Output BAP
for	6	6	Output CAP	Output CAP	Output CAP	Output CAP
chaining	7	7	Reserved*	Reserved*	Reserved*	Reserved*
instructions	8	8	Reserved*	Reserved*	Monitor Word	Reserved*
	9	9	Reserved*	Reserved*	Suppress Word	--
	A	A	Operating Mode	Operating Mode	Serial Mode	
						Reserved*
						Reserved*
	B	B	--	--	--	Reserved*
	C-F		Not Used	--	--	--

*Reserved addresses may be used for future enhancements.

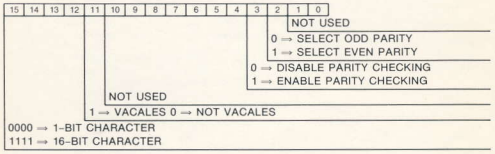
I/O CONTROL MEMORY



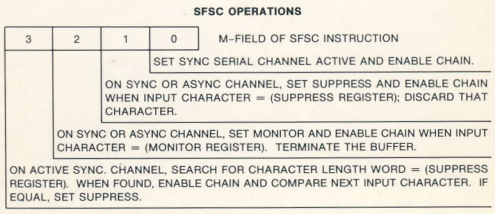
TM = 00 – Abort the transfer. For input, continue accepting the input data, but do not write it into memory.
TM = 01 – Transfer 8-bit bytes.
TM = 10 – Transfer 16-bit words.
TM = 11 – Transfer 32-bit double words.
PS = 0 – Use page register set 0.
PS = 1 – Use page register set 2 if the channel number 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

VACALES OPERATING MODES



SFSC OPERATIONS



OPERATION CODE 3D, SET SUPPRESS, SET MONITOR AND SEARCH FOR SYNC m-DESIGNATOR FUNCTIONS

m-VALUE	INTERFACE	FUNCTION
0000	Sync	Disable sync, disable monitor, disable set suppress, and enable next chain instruction.
0000	Async	Disable set monitor, disable set suppress, and enable next chain instruction.
0001	Sync	Hold sync active, disable set monitor, disable set suppress, and enable next chain instruction.
0010	Async	Enable set suppress, disable set monitor, and enable next chain instruction.
0011	Sync	Hold sync active, enable set suppress, disable set monitor, and enable next chain instruction.
0100	Async	Enable set monitor, disable set suppress, and enable next chain instruction.
0101	Sync	Hold sync active, enable set monitor, disable set suppress, and enable next chain instruction.
0110	Async	Enable set monitor, enable set suppress, enable next chain instruction.
0111	Sync	Hold sync active, enable set monitor, enable set suppress, enable next chain instruction.
1000	–	No operation and enable next chain instruction.
1001	Sync, MIL-STD-188C, RS-232-C	Enable search for sync, disable set suppress after sync is established, and disable set monitor, disable chaining until function is complete.
1001	Sync, VACALES	Enable search for sync, disable set suppress after sync is established, and disable set monitor, disable chaining until function is complete.
1010	–	No operation and enable next chain instruction.
1011	Sync, MIL-STD-188C, RS-232-C	Enable search for sync, enable set suppress, and disable set monitor; disable chaining until function is complete.
1011	Sync, VACALES	Enable search for sync bit-by-bit, enable set suppress, and disable set monitor; disable chaining until function is complete.
1100	–	No operation and enable next chain instruction.
1101	Sync, MIL-STD-188C, RS-232-C	Enable search for sync, disable set suppress after sync is established, and enable set monitor; disable chaining until function is complete.
1101	Sync, VACALES	Enable search for sync character compare, set suppress flag if the next character compares, disable set monitor and disable set suppress on subsequent characters; enable next chain instruction after the character compare.
1110	–	No operation and enable next chain instruction.
1111	Sync, MIL-STD-188C, RS-232-C	Enable search for sync, enable set monitor and enable set suppress; disable chaining until function is complete.
1111	Sync, VACALES	Enable search for sync character compare, set suppress flag if the next character compares, enable set monitor and enable set suppress on subsequent characters; enable next chain instruction after the character compare.

STORE STATUS BIT INTERPRETATION FOR VACALES

BIT	FUNCTION	DESCRIPTION
2^1	OVERRUN	THE SERIAL I/O DID NOT TRANSFER TO MEMORY BEFORE ANOTHER I/O WORD WAS RECEIVED.
2^2	PARITY ERROR	THE SERIAL I/O DETECTED A PARITY ERROR ON AN INPUT DATA WORD.
2^3	SYNC ERROR	THE INBOUND DISCRETE CONTROL LINE, SYNC ERROR, WAS SET BY AN EXTERNAL DEVICE.

OPERATION CODE 38, CHANNEL CONTROL INSTRUCTION m-DESIGNATOR AND a-DESIGNATOR

ULTRA SYMBOL	CODING FORMAT m-VALUE	CODING FORMAT a-VALUE	OPERATION
	0	0	Master Clear – All channels – Deactivate all chains – Terminate all I/O transfers – Disable all external interrupt data storage and clear all EIE lines – Clear all pending Class III interrupts, disable further generation of Class III Priority 2, 3, and 4 interrupts – Clear intercomputer time-out function – Clear serial monitor and suppress flags.
	1	–	Illegal
ACR	2	–	Illegal
(1)	3	–	Illegal
	4	–	Enable all channels' external interrupt data storage; set all EIE lines.
	5	–	Disable all channels' external interrupt data storage; clear all EIE lines.
	6	0	Enable all channels' external interrupt monitors to allow Class III, priority 2, 3, and 4 interrupt generation. If external interrupt data were stored while monitors where disabled, generate the Class III, priority 2 interrupt.
	6	1–F	Enable external interrupt monitors for all channels with priority lower than the channel defined by a to allow Class III, priority 2, 3, and 4 interrupt generation. If external interrupt data was stored while monitors were disabled, generate the Class III, priority 2 interrupt.
	7	0	Disable priority 2, 3, and 4 interrupt generation for all channels.
	7	1–F	Disable priority 2, 3, and 4 interrupt generation by channels with priority lower than the channel defined by a.
	8	0–F	Master Clear the channel defined by a (see m = 0 above).
	9	0–F	Master Clear input on channel defined by a.
	A	0–F	Master Clear output on channel defined by a.
CCR	B	–	Illegal
(2)	C	0–F	Enable the channel external interrupt data storage; set EIE line or send command.
	D	0–F	Disable the channel external interrupt data storage; clear EIE line.
	E	0–F	Enable the channel Class III priority 2, 3, and 4 interrupt generation (see m = 6 above).
	F	0–F	Disable the channel Class III priority 2, 3, and 4 interrupt generation.

NOTES: (1) Operations affecting all channels collectively (command or chaining)
(2) Operations affecting only the channel specified by the a-designator (command) or the associated channel (chaining)
For all I/O instructions, RK and RX formats, Y = y (indexing and indirect addressing cannot be used).

JUMP (CHAINING) a-DESIGNATOR JUMP CONDITIONS

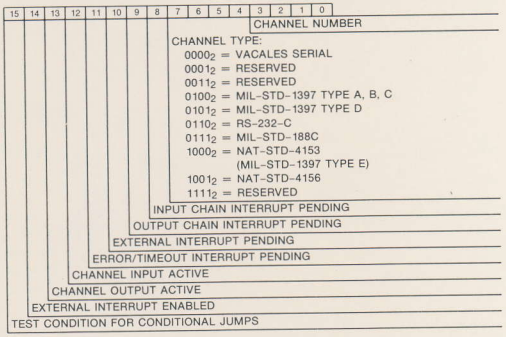
ULTRA SYMBOL	a-VALUE	JUMP CONDITION
SJC	0	Unconditional jump
	1	Serial jump if suppress flag not set (for MIL-STD-188C and RS-232-C, flag is cleared during next character time; for VACALES, flag is cleared when next character is transferred to memory). (No jump for MIL-STD-1397 or NAT-STD-4153.)
	2	Serial jump if monitor flag set (for MIL-STD-188C and RS-232-C, flag is cleared when next character is transferred to memory). (No jump for MIL-STD-1397 or NAT-STD-4153.)
SJMC	4	Jump if condition bit (bit 15) in the I/O status word is set.
	A	Jump if input buffer is active.
	B	Jump if output buffer is active.
	C	Jump if external function buffer is active. (No jump for MIL-STD-188C, RS-232-C, or VACALES.)

OPERATION CODE 3E, DISCRETE SET/CLEAR FUNCTIONS

m-VALUE	FUNCTION	MIL-STD-188C/VACALES LINE DESIGNATOR MIL-STD-188C	MIL-STD-188C/VACALES LINE DESIGNATOR VACALES	RS-232-C DISCRETE	RS-232-C LINE DESIGNATOR
0	Enable Loop Test	(1)	(1)	(1)	–
1	(internal)	(1)	(1)	(1)	–
2	Disable Loop Test	Not Used	Not Used	Not Used	–
3	(internal)	Not Used	Not Used	Not Used	–
4	Noop	J	J	J (non-Std.)	J
5	Noop	J	J	J (non-Std.)	J
6	On	H	Transmitter Prep.	Dsbl. Ring Indicator	–
	Off				
7	On	H	Transmitter Prep.	Intrpt. (1) Enbl. Ring Indicator	–
8	Off	G	G	Intrpt.(1)	CA
9		G	G	Request to Send	CA
A	Off	F	F	Request to Send	CH
B	On	F	F	New Sync	CH
C	Off	D	D	New Sync	CD
D	On	D	D	Data Terminal Ready	CD
E	Off	A	Loop Back (external)	Data Terminal Ready	–
	On			Modem Loop Test (external)	
F	Off	A	Loop Back (external)	Modem Loop Test (external)	–
	On				

(1) Internal function – no interface line affected.

I/O STATUS WORD



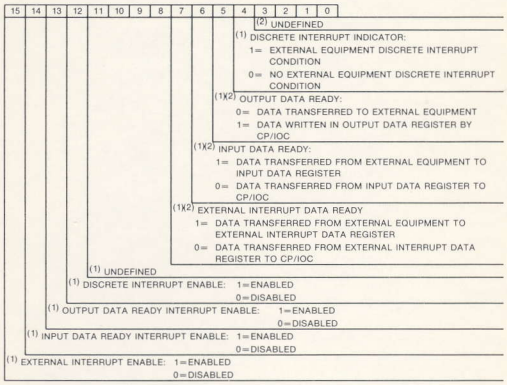
STATUS WORD INTERPRETATION

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.

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

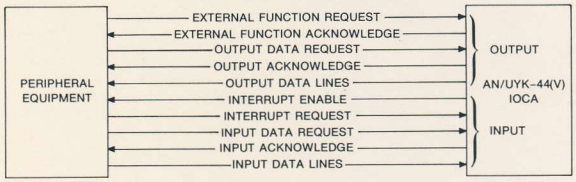
ADDRESS X – EXTERNAL INTERRUPT WORD
x+1 – INPUT DATA WORD
x+2 – OUTPUT DATA WORD
x+3 – MMIO CONTROL/STATUS WORD

I/O CONNECTOR AND CABLE TYPES

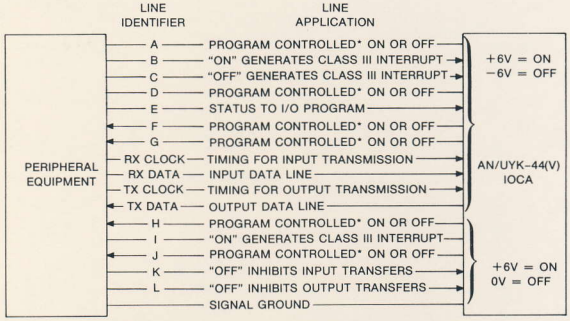
CHANNEL TYPE	CABLE TYPE	CONN. PER CHAN.	MOUNTED I/O CONNECTOR	MATING I/O CONNECTOR	CONT.
MIL-STD-1397 Type A, B, C	2U45 2AU40	2	INPUT D38999/20WG35AN OUTPUT D38999/20WG35AA	INPUT D38999/20WG35SN OUTPUT D38999/20WG35SA	79
MIL-STD-1397 Type D	RG-11 RG-12	2	COAXIAL AMPHENOL PN34475-1050	COAXIAL AMPHENOL PN53250-1000	2
NAT-STD-4153 (MIL-STD-1397 Type E)	TRF-58 TRF-8	2	TRIAXIAL TROMPETER PN BJ80	TRIAXIAL TROMPETER PN80-14A	2
MIL-STD-188C, VACALES	2U19	1	D38999/20WE26AN	D38999/26WE26SN	26
RS-232-C		1	D38999/20WE26AN	D38999/26WE26SN	26
NAT-STD-4156	2U-10 (10 twisted pair with overall braid shield)	1	D38999/20WE26AA	D38999/26WE26SA	26

MIL-STD-1397 PARALLEL INTERFACE

COMPUTER TO PERIPHERAL EQUIPMENT INTERFACE
(8, 16, or 32 BIT PARALLEL TRANSFERS)

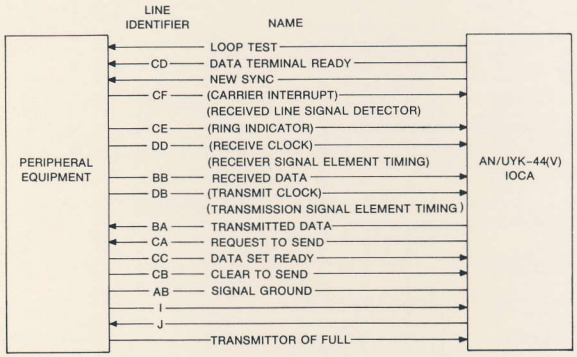


MIL-STD-188C AND VACALES INTERFACE

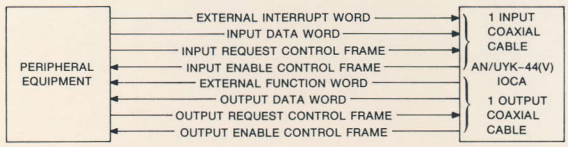


* PROGRAM CONTROLLED LINES ARE ASSIGNED FUNCTIONS ACCORDING TO THE NEED OF THE PARTICULAR EQUIPMENT CONNECTED TO THE CHANNEL.

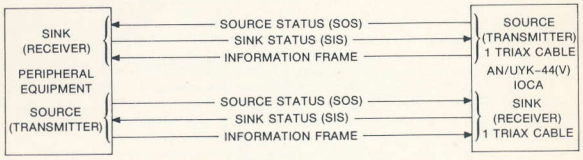
RS-232-C INTERFACE



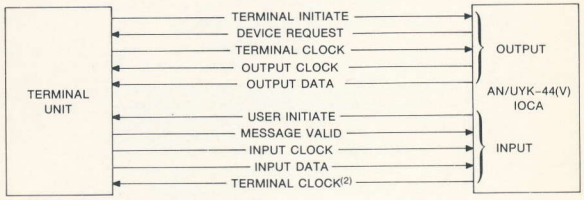
MIL-STD-1397 TYPE D INTERFACE



NAT-STD-4156 INTERFACE



NAT-STD-4153 (MIL-STD-1397 TYPE E) INTERFACE



(2) USED ONLY FOR AN/UYK-44(V) TO AN/UYK-44(V) COMMUNICATION

ASCII CHARACTER SET

ASCII	OCT	HEX	DEC
NUL	000	00	000
SOH	001	01	001
STX	002	02	002
ETX	003	03	003
EOT	004	04	004
ENQ	005	05	005
ACK	006	06	006
BEL	007	07	007
BS	010	08	008
HT	011	09	009
LF	012	0A	010
VT	013	0B	011
FF	014	0C	012
CR	015	0D	013
SO	016	0E	014
SI	017	0F	015
DLE	020	10	016
DC1	021	11	017
DC2	022	12	018
DC3	023	13	019
DC4	024	14	020
NAK	025	15	021
SYN	026	16	022
ETB	027	17	023
CAN	030	18	024
EM	031	19	025
SUB	032	1A	026
ESC	033	1B	027
FS	034	1C	028
GS	035	1D	029
RS	036	1E	030
US	037	1F	031
SP (Space)	040	20	032
! (Exclamation)	041	21	033
" (Quotes)	042	22	034
# (Number)	043	23	035
$ (Dollar Sign)	044	24	036
% (Percent)	045	25	037
& (Ampersand)	046	26	038
' (Apostrophe)	047	27	039
((Left Parenthesis)	050	28	040
) (Right Parenthesis)	051	29	041
* (Asterisk)	052	2A	042
+ (Plus)	053	2B	043
, (Comma)	054	2C	044
- (Minus)	055	2D	045
. (Period-Dec. Pt.)	056	2E	046
/ (Slash)	057	2F	047
0	060	30	048
1	061	31	049
2	062	32	050
3	063	33	051
4	064	34	052
5	065	35	053
6	066	36	054
7	067	37	055
8	070	38	056
9	071	39	057
: (Colon)	072	3A	058
; (Semicolon)	073	3B	059
< (Less than)	074	3C	060
= (Equal)	075	3D	061
> (Greater than)	076	3E	062
? (Question Mark)	077	3F	063

ASCII	OCT	HEX	DEC
@ (At)	100	40	064
A	101	41	065
B	102	42	066
C	103	43	067
D	104	44	068
E	105	45	069
F	106	46	070
G	107	47	071
H	110	48	072
I	111	49	073
J	112	4A	074
K	113	4B	075
L	114	4C	076
M	115	4D	077
N	116	4E	078
O	117	4F	079
P	120	50	080
Q	121	51	081
R	122	52	082
S	123	53	083
T	124	54	084
U	125	55	085
V	126	56	086
W	127	57	087
X	130	58	088
Y	131	59	089
Z	132	5A	090
[(Left Bracket)	133	5B	091
\ (Backslash)	134	5C	092
] (Right Bracket)	135	5D	093
^ (Circumflex)	136	5E	094
_ (Underline)	137	5F	095
` (Grave Accent)	140	60	096
a	141	61	097
b	142	62	098
c	143	63	099
d	144	64	100
e	145	65	101
f	146	66	102
g	147	67	103
h	150	68	104
i	151	69	105
j	152	6A	106
k	153	6B	107
l	154	6C	108
m	155	6D	109
n	156	6E	110
o	157	6F	111
p	160	70	112
q	161	71	113
r	162	72	114
s	163	73	115
t	164	74	116
u	165	75	117
v	166	76	118
w	167	77	119
x	170	78	120
y	171	79	121
z	172	7A	122
{ (Left Brace)	173	7B	123
` (Acute Accent)	174	7C	124
} (Right Brace)	175	7D	125
~ (Tilde)	176	7E	126
DEL	177	7F	127

Note: The undefined ASCII characters are not used.

MDS COMMAND SUMMARY

Representations of command use modified Backus–Naur notation.

[ ] means optional

{ } means choose one

MDS CONTROL COMMANDS (paragraph 2.4)

RESET	Reset the MDS
FORM [{H,HEX,O,OCT}]	Select data display and entry format
GRS [{0,1,C}]	Select general-register set for reference by MDS
PRS [{0,1,2,3,C}]	Select page-register set for reference by MDS
IOC [{0,1,2,3}]	Select IOC for reference by MDS

MAINTENANCE PANEL SWITCH EMULATION COMMANDS (paragraph 3.2)

MC	Master Clear (paragraph 3.2.1)
JS1 [{ON,OFF}]	Enable/Disable Jump Stop 1 (paragraph 3.2.2)
JS2 [{ON,OFF}]	Enable/Disable Jump Stop 2 (paragraph 3.2.2)
BOOT [{1,2}]	Select bootstrap option (paragraph 3.2.6)
LOAD	Start bootstrap load (paragraph 3.2.3)
RTCS [{OFF,EXT,1K,32K}]	Set RTC Switch (paragraph 3.2.4)
ICT [{ENA,DIS}]	Enable/Disable Intercomputer Timeouts (paragraph 3.2.5)
PWFI	Initiate simulated power fault (paragraph 3.2.7)
PWFC	Clear simulated power fault (paragraph 3.2.7)
CFI	Clear status display PWFLT and PGFLT fault indicators (paragraph 3.2.8)

REGISTER REFERENCE COMMANDS (paragraph 3.3)

Control Registers (paragraph 3.3.1)

reg [{val}]	Change value in specified register
P[val]	Change the contents of the P register
IR	Inspect the contents of the Instruction Register
SR1[val]	Change the contents of Status Register 1
SR2[val]	Change the contents of Status Register 2
RTCL[val]	Change the contents of RTC register lower half
RTCU[val]	Change the contents of RTC register upper half
MON[val]	Change the contents of the monitor clock register

General Registers (paragraph 3.3.2)

GR	Display the contents of all 16 registers in the selected general register set.
GRrg[val]	Change the contents of general register (rg) in the selected register set.

Page Register (paragraph 3.3.3)

PR	Display the contents of all 64 registers in the selected combined-page register set.
PRrg[val]	Change the contents of page register rg in the selected page register set.
CR	Display the contents of all 64 registers in the selected CPU page-register set.
CRrg[val]	Change the contents of register (rg) in the selected page-register set.
MR	Display the contents of all 64 registers in the selected MAE page-register set.
MRrg[val]	Change the contents of register (rg) in the selected page-register set.
N[,value]	Change the contents of the next higher general register page register, control memory location, main memory location, bit signature word, or diagnostic signature word; depending on which type of these items was last accessed by MDS software (paragraph 5.3).

NOTE: For all above commands, if the value or options were omitted from the command entry, the current setting or value of the switch or register will be displayed.

PROGRAM EXECUTION COMMANDS (paragraph 3.4.1)

RUN [reladr] R [reladr]	Start program execution and set P register to relative starting address (reladr) specified (paragraph 34.1).
RUNJ [reladr] RJ [reladr]	Start program execution, set P register to relative starting address (reladr) specified, and continue until a jump instruction has been executed (paragraph 34.2).
STOP	Stop an executing program (paragraph 3.4.3)
STEP [steps][reladr] S [steps][reladr]	Op step the MRP for the specified number of steps and set P register to the relative starting address (reladr) address specified (paragraph 3.4.4)
RUNI RI	Start execution for all IOC(s) (paragraph 3.4.5)
STOPI	Stop execution for all IOC(s) (paragraph 3.4.6)
STEPI [num,ioc] SI [num,ioc]	Op-step ioc for num steps (paragraph 3.4.7)
BP [[0,1,2,3,C,D,E] [C,D,E,R,W,X,S,F]address]	Select, clear, enable, disable, or change breakpoints (paragraph 3.4.8)
HIST	Display P-jump history (paragraph 3.4.9)

IOC COMMANDS (section 4)

CM ch	Display the contents of all 16 control memory locations for the selected channel on the selected IOC (paragraph 4.2).
CM ch [wrnm][val]	Change the contents of control memory location (wrnm) for the selected channel on the selected IOC (paragraph 4.2).
CHST ch	Inspect channels ch (paragraph 4.3)
IOST [ioc]	Inspect selected ioc status (paragraph 4.4)

MAIN MEMORY COMMANDS (section 5)

M adr[value]	Inspect and change adr referenced and place designated value (1: numeric; 2: ASCII) in specified main memory location (paragraph 5.2)
D [adr][len] DUMP [adr][len]	Display (dump) block of memory in constant (single numeric) format (paragraph 5.4)
DA [adr][len] DUMPA [adr] [len]	Display block of memory from memory block starting address (adr), len long, in ASCII format (paragraph 5.4)
CONS adr,len,value	Store a constant at memory block starting address (adr), len long, of specified numeric value in several locations (paragraph 5.5)
COPY from, len,dest	Duplicate the contents of designated (from) consecutive memory locations of len length to a different group of consecutive memory locations (dest) (paragraph 5.6)
COMP blkaddr1, len, blkadr2 [d]	Compare two blocks of memory, blkadr1 and blkadr2, of len length, and display the differences (d) (paragraph 5.7)

DIAGNOSTICS (section 6)

TEST	Initiate a BIT (paragraph 6.2)
TPxx [,value]	Inspect [or change] a value in test parameter word xx (paragraph 6.3)
FS	Display contents of FS words (paragraph 6.4)

ANGULAR EQUIVALENTS

BIT	ANGLE (DEGREES)	ANGLE (RADIANS)
15	180	3.14159
14	90	1.57079
13	45	0.78540
12	22.5	0.39270
11	11.25	0.19635
10	5.625	0.09818
9	2.8125	0.04909
8	1.4063	0.02454
7	0.7031	0.01227
6	0.3516	0.006136
5	0.1758	0.003068
4	0.08789	0.001534
3	0.04395	0.000767
2	0.02197	0.000383
1	0.01099	0.000192
0	0.005493	0.000096