CUSTOMER SERVICE

TEK TIP NO. 65-76-10

Date 1-5-67

Ref. E.O. No. N.A.

Subject:

SIGMA 7 CPU ENGINEERING NOTES
AND PHASE SEQUENCE CHARTS

Related Model Numbers:

8401 Sigma 7 Central Processor
Change Level: E

Distribution:

Customer Service

Void After:

Technical Discussion:

Attached are Sigma 7 CPU Engineering Notes and Phase Sequence Charts. These charts are for reference only and reflect the general operation of the Sigma 7 CPU. Simplified equations shown in the phase sequence charts are functional and representative in nature. These equations are not necessarily complete nor are shown as implemented. The specific logic equations must be consulted to determine how a given function is implemented.

The central processor logic equation drawing numbers are:

"FRAME 1"	131971
"FRAME 2"	131972
"FRAME 3"	131973

Attachments: Sigma 7 CPU Engineering Notes and Phase Sequence Charts, pages i, ii, and 1 through 84.

Prepared By: Alan Mitchell Date: 1-5-67

Approved: Alan Mitchell Date: 1-5-67
Product Support Engineer

Approved: Date: 1-5-67
Manager, Product Support Engineering

INDEX

SIGMA 7 INSTRUCTION PHASE SEQUENCE CHARTS

Mnemonic	Code	Instruction-Name	Page
LOAD/STORE			
LI	22	Load Immediate	9
LB	72	Load Byte	9
LH	52	Load Halfword	9
LW	32	Load Word	9
LD	12	Load Doubleword	10
LCH	5A	Load Complement Halfword	9
LAH	5B	Load Absolute Halfword	9
LCW	3A	Load Complement Word	9
LAW	3B	Load Absolute Word	9
LCD	1A	Load Complement Doubleword	17
LAD	1B	Load Absolute Doubleword	11
LS	4A	Load Selective	12
LM	2A	Load Multiple	64
LCFI	02	Load Conditions & Floating Control Immediate	10
LCF	70	Load Conditions & Floating Control	10
XW	46	Exchange Word	13
STB	75	Store Byte	14
STH	55	Store Halfword	14
STW	35	Store Word	14
STD	15	Store Doubleword	13
STS	47	Store Selective	15
STM	2B	Store Multiple	64
STCF	74	Store Conditions & Floating Control	14
ANALYZE/INTERPRET			
ANLZ	44	Analyze	70
INT	6B	Interpret	70
FIXED-POINT ARITHMETIC			
AI	20	Add Immediate	16
AH	50	Add Halfword	16
AW	30	Add Word	16
AD	10	Add Doubleword	17
SH	58	Subtract Halfword	16
SW	38	Subtract Word	16
SD	18	Subtract Doubleword	17
MI	23	Multiply Immediate	26
MH	57	Multiply Halfword	28
MW	37	Multiply Word	26
DH	56	Divide Halfword	32
DW	36	Divide Word	30
AWM	66	Add Word to Memory	21
MTB	73	Modify & Test Byte	21
MTH	53	Modify & Test Halfword	21
MTW	33	Modify & Test Word	21
COMPARISON			
CI	21	Compare Immediate	18
CB	71	Compare Byte	18
CH	51	Compare Halfword	18
CW	31	Compare Word	18
CD	11	Compare Doubleword	17
CS	45	Compare Selective	19
CLR	39	Compare with Limits in Register	20
CLM	19	Compare with Limits in Memory	20
LOGICAL			
OR	49	OR Word	21
EOR	48	Exclusive OR Word	21
AND	4B	AND Word	21
SHIFT			
S	25	Shift	34
SF	24	Shift Floating	37
CONVERSION			
CVA	29	Convert by Addition	63
CVS	28	Convert by Subtraction	63

Mnemonic	Code	Instruction-Name	Page
FLOATING-POINT ARITHMETIC			
FAS	3D	Floating Add Short	40
FAL	1D	Floating Add Long	40
FSS	3C	Floating Subtract Short	40
FSL	1C	Floating Subtract Long	40
FMS	3F	Floating Multiply Short	45
FML	1F	Floating Multiply Long	45
FDS	3E	Floating Divide Short	50
FDL	1E	Floating Divide Long	50
DECIMAL			
DL	7E	Decimal Load	80
DST	7F	Decimal Store	80
DA	79	Decimal Add	80
DS	78	Decimal Subtract	80
DM	7B	Decimal Multiply	80
DD	7A	Decimal Divide	80
DC	7D	Decimal Compare	80
DSA	7C	Decimal Shift Arithmetic	80
PACK	76	Pack Decimal Digits	80
UNPK	77	Unpack Decimal Digits	80
BYTE STRING			
MBS	61	Move Byte String	55
CBS	60	Compare Byte String	55
TBS	41	Translate Byte String	57
TTBS	40	Translate & Test Byte String	57
EBS	63	Edit Byte String	60
PUSH DOWN			
PSW	09	Push Word	65
PLW	08	Pull Word	65
PSM	0B	Push Multiple	65
PLM	0A	Pull Multiple	65
MSP	13	Modify Stack Pointer	65
EXECUTE/BRANCH			
EXU	67	Execute	82
BCS	69	Branch on Conditions Set	83
BCR	68	Branch on Conditions Reset	83
BIR	65	Branch on Incrementing Register	83
BDR	64	Branch on Decrementing Register	83
BAL	6A	Branch and Link	82
CALL			
CAL1	04	Call 1	84
CAL2	05	Call 2	84
CAL3	06	Call 3	84
CAL4	07	Call 4	84
CONTROL			
LPSD	0E	Load Program Status Doubleword	72
XPSD	0F	Exchange Program Status Doubleword	72
LRP	2F	Load Register Pointer	74
MMC	6F	Move to Memory Control	68
WAIT	2E	Wait	84
RD	6C	Read Direct	74
WD	6D	Write Direct	74
INPUT/OUTPUT			
SIO	4C	Start Input/Output	76
HIO	4F	Halt Input/Output	76
TIO	4D	Test Input/Output	76
TDV	4E	Test Device	76
AIO	6E	Acknowledge Input/Output Interrupt	76

TABLE OF CONTENTS (by page no.)

1 C.P.U. Registers
2 OPCODES (OU & OL decoding), Register end-bits
3 Adder & carry system
4 PREPARATION PHASES
6 P.C.P. PHASES

9 LI, LW, LH, LB, LCW, LCH, LAW, LAH
10 LD, LCF, LCFT
11 LAD
12 LS
13 XW, STD
14 STW, STH, STB, STCF
15 STS

16 AI, AW, AH, SW, SH
17 AD, SD, LCD, CD

18 CI, CW, CH, CB
19 CS
20 CLM, CLR

21 EOR, OR, AND, AWM, MTW, MTH, MTB

23 MULTIPLY (general info.)
26 MW, MI
28 MH

29 DIVIDE (general info.)
30 DW
32 DH

33 SHIFTS (general info)
34 S
36 SF

39 FLOATING POINT (general info.)
40 FAL, FAS, FSL, FSS
45 FML, FMS
50 FDL, FDS

55 MBS, CBS
57 TBS, TTBS
60 EBS

63 CVA, CVS
64 LM, STM
65 PLW, PSW, PLM, PSM, MSP
68 MMC
70 INT, ANLZ
72 XPSD, LPSD
74 LRP, RD, WD
76 SIO, TIO, TDV, HIO, AIO

78 INTRAP SEQUENCE
80 DA, DS, DL, DST, DC, DM, DSA, DD, PACK, UNPK
82 EXU, BAL
83 BCS, BCR, BDR, BIR
84 WAIT, CAL1, CAL2, CAL3, CAL4

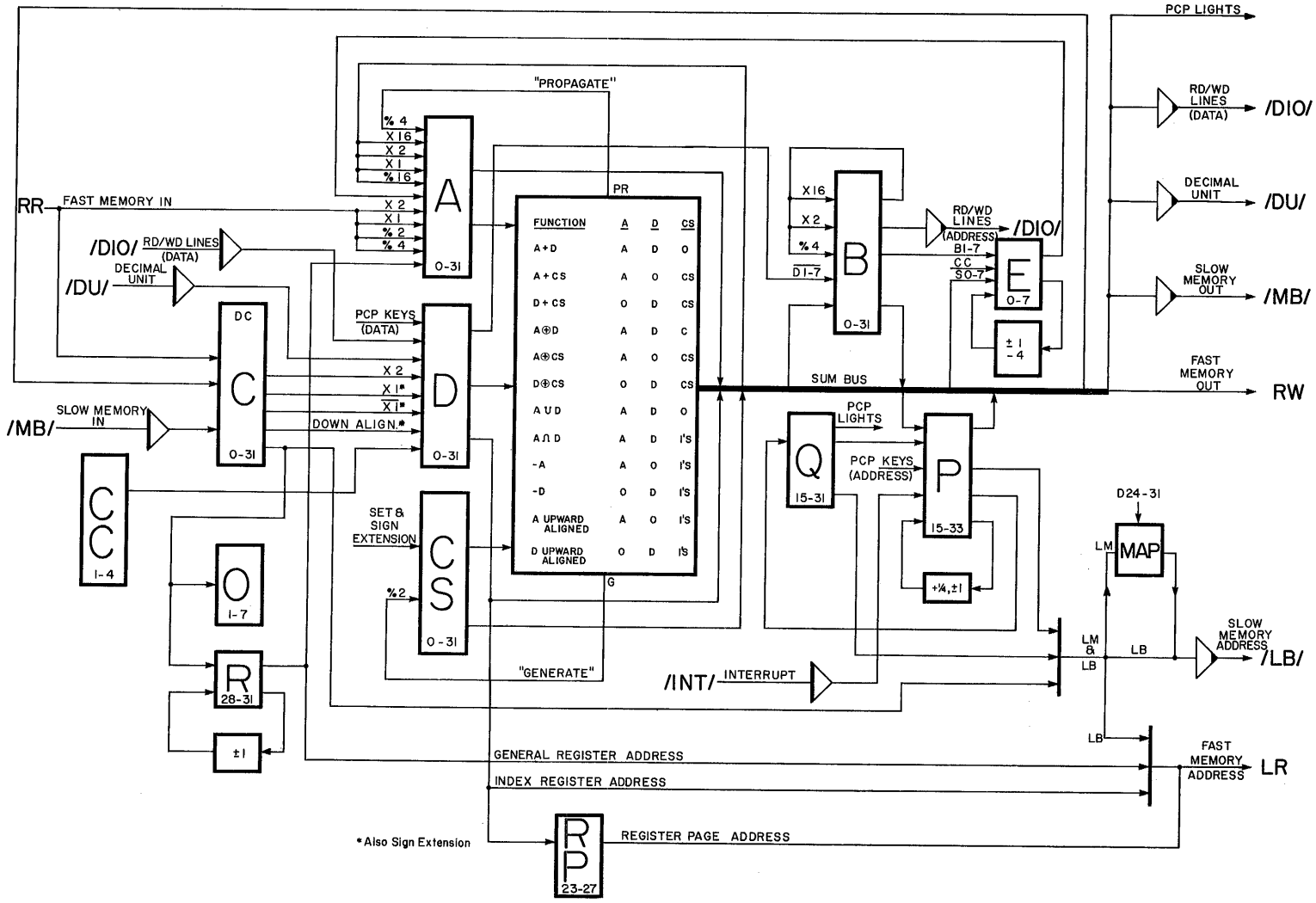
MISC:

Down-align ½WD : see p16, PH1

Down-align BYTE : see p18, PH1

TESTA : see p16, following PH5
TESTA/1 : see p18, following PH4

SIGMA 7 CPU REGISTERS



SIGMA 7 OPCODE CHART

	AI	TTBS	CBS	AD	AW	AH	LCF	0L0
	CI	TBS	MBS	CD	CW	CH	CB	0L1
LCFI	LI			LD	LW	LH	LB	0L2
	MI		EBS	MSP	MTW	MTH	MTB	0L3
CAL1	SF	ANLZ	BDR				STCF	0L4
CAL2	S	CS	BIR	STD	STW	STH	STB	0L5
CAL3		XW	AWM		DW	DH	PACK	0L6
CAL4		STS	EXU		MW	MH	UNPK	0L7
PLW	CVS	EOR	BCR	SD	SW	SH	DS	0L8
PSW	CVA	OR	BCS	CLM	CLR		DA	0L9
PLM	LM	LS	BAL	LCD	LCW	LCH	DD	0LA
PSM	STM	AND	INT	LAD	LAW	LAH	DM	0LB
		SIO	RD	FSL	FSS		DSA	0LC
		TIO	WD	FAL	FAS		DC	0LD
LPSD	WAIT	TDV	AIO	FDL	FDS		DL	0LE
XPSD	LRP	HIO	MMC	FML	FMS		DST	0LF
0U0	0U2	0U4	0U6	0U1	0U3	0U5	0U7	

REGISTER END-BITS, ETC

SHIFT LEFT 1 LOGIC :

A31 : AxSL1 x A31EN/2, [where A31EN/2 = B0.FAMDSF/1 + S0.ALCYC + K46.FAFLD.PH12 ← FAMDSF.N(FAFLD.PH11.P26 + FAFLD.PH12)]

B31 : BxBL1 x B31EN/1, [where B31EN/1 = BC31 + S0.FASHFx.C22.C23 + B0.NFAMDSF]
+ DxNC.FADIV.PH11 + K46.FAFLD.PH11

SHIFT LEFT 4 LOGIC :

A28-31 : AxSL4 x A28-31EN/1, [where A28-31EN = B0-3.(FASH.C23 + FAFLM.PH13) + A0-3.ALCYC]

B28-31 : BxBL4/1 x S0-3, [where BxBL4/1 = BxBL4.FASHFx.C22.C23]

SHIFT RIGHT 2 LOGIC :

A47 : AxPRR2F (A47 + D46)

A48 : " (A47 ⊕ D46)

A0 : " . PR70 + AxPRR2.A0EN/1,
[where A0EN/1 = (A0+D71)(FASHFx.C21 + FAMUL) + B30(FASHFx.C22.C23) + A30.ARCYC

A1 : " . PR71 + AxPRR2.A1EN/1
[where A1EN/1 = (A0 ⊕ D71)(") + B31.(") + A31. "

B47 : cleared by MIT FAFLM

B48,49 : RN.MIT FAFLM ← (sign pad)

B0 : BxBR2.FAMDSF.PR30.N(FAMDSF/M.NMIT) + BxBR2.NFAMDSF.B30

B1 : " . " .PR31.N(") + " . " .B31

B2 : " . " .(PR32 ⊕ G33) + " . .B0

B3 : " . " .(B1 ⊕ BC1 ⊕ CS33) + " . " .B1

B4 : " .(MIT FAFLM).B70 + " .N(MIT.FAFLM).B2

B5 : " .(").B71 + " .N(").B3

B8,9 : " .(shift FAFLM).RN ← (sign pad) + " .N(shift FAFLM).B6,7

BC31 : SFTR2.B30, etc. (for long, non-cyclic, odd-numbered, fixed point shifts)

SHIFT RIGHT 4 LOGIC (floating-point only) :

A47, 48, 49 are cleared by AxSR4

A50 : AxSR4.(FMOF.A47≠)

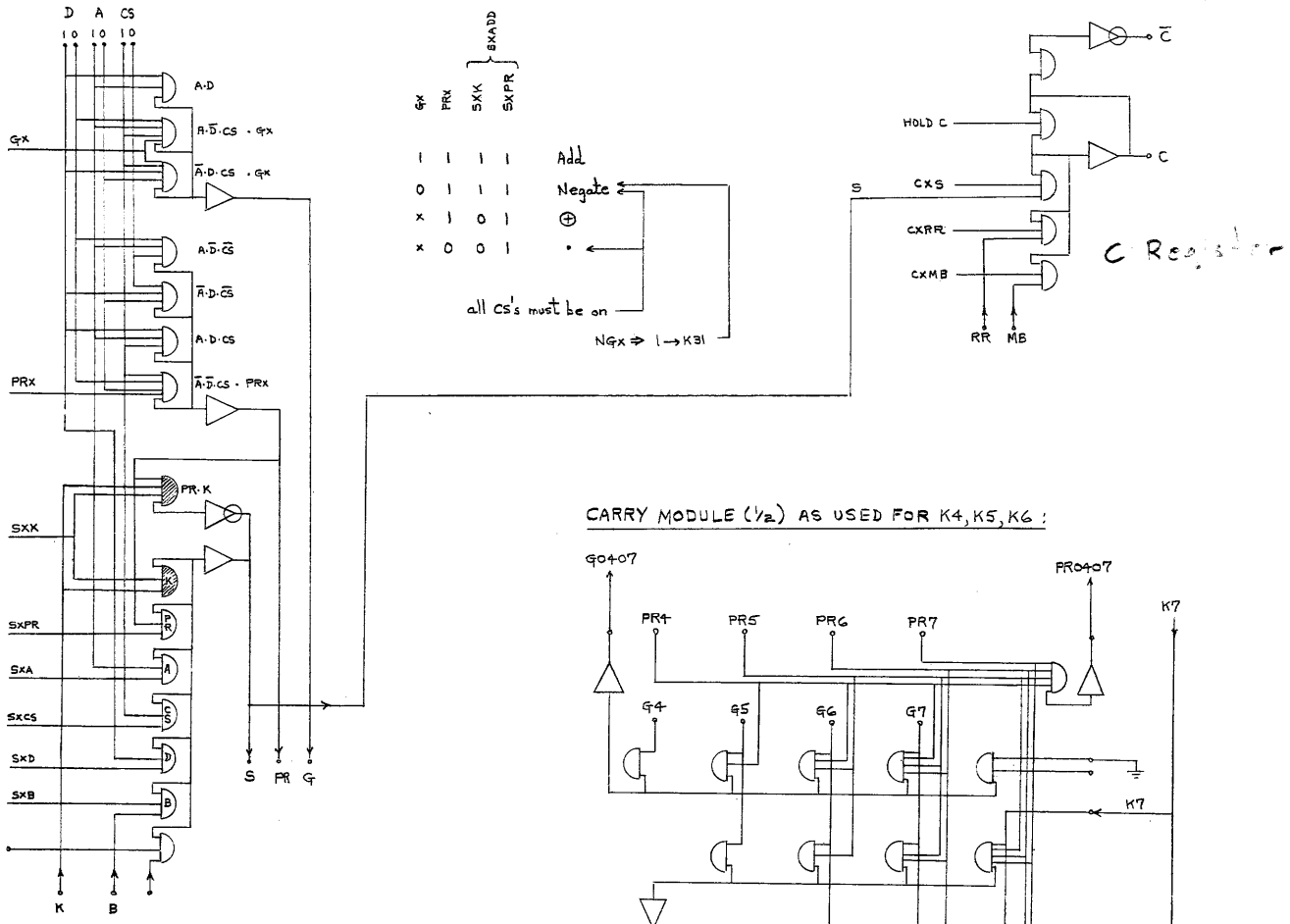
B0, 1, 2 : (FAFLD.PH10).S28, 29, 30

SHIFT INSTRUCTION CONTROL BITS : C21 ⇒ ARITHMETIC, C22 ⇒ CYCLIC, C23 ⇒ LONG

ALCYC = FASHFx.C22.NC23 + NFAMDSF

ARCYC = ALCYC.NANLZ

ADDER MODULE (½):

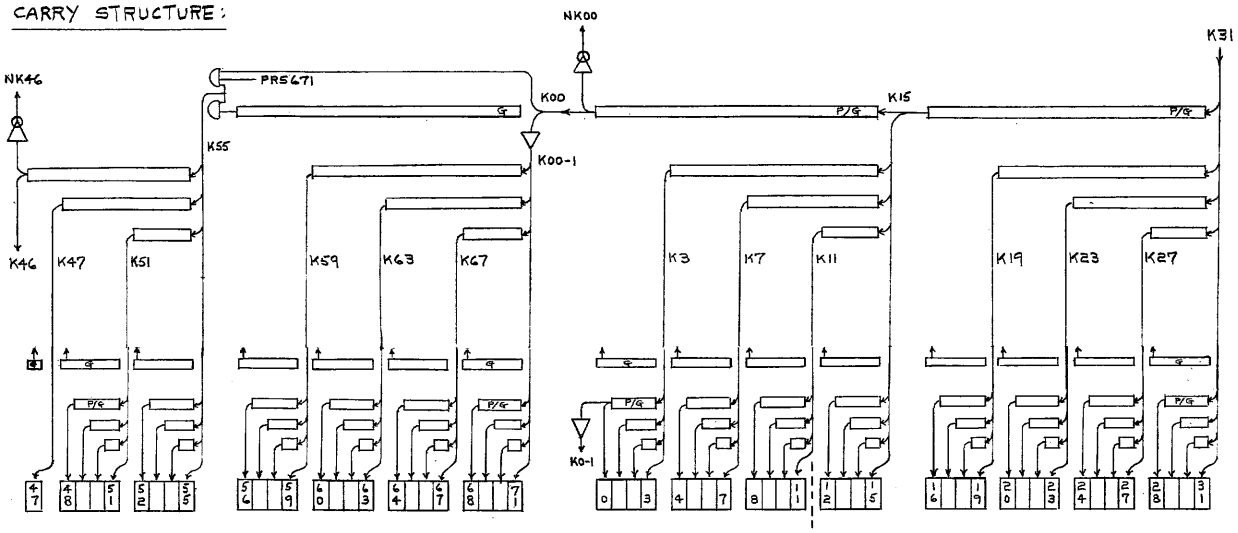


NOTE: when SXK and SXPR are high (SXADD case)

$S = N(PR.K).(PR+K)$, which reduces to $PR \oplus K$

CARRY MODULE (½) AS USED FOR K4, K5, K6:

CARRY STRUCTURE:



PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE1	P → Q	QXP = PRE1 . NANLZ	
	Reset A and E registers	AX/1 = PRE1 , EX/1 = PRE1	
T6RL	NINTRAPF ⇒ Reset B	BX/1 = PRE1. NINTRAPF	
	D12-14 → LR	(S/LRXD) = GXC	GXC = ENDE (Previous Phase)
	Set Request from C ⇒ Set ARQ	S/ARQ = RQC	
	Indexed ⇒ Set IX	S/IX = INDX. PRE1	NINDX = .NC12. NC13. NC14 +.(NC3. NC4. NC5)
	Indexed. Byte Add. ⇒ (RR) → A right 2	AXRRR2 = GU7 . INDX . PRE1	
	Indexed . HW Add ⇒ (RR) → A right 1	AXRRR1 = GU5 . INDX . PRE1	
	Indexed . Word Add ⇒ (RR) → A	AXRR = FAW. INDX . PRE1	
	Indexed. DW Add ⇒ (RR) → A left 1	AXRRL1 = FADW. INDX. PRE1	
	Not Indexed . PRERQ ⇒ Set RQ	S/RQ = PRERQ . NIA. PRE1. NINDX	
	Indirect. Not immediate ⇒ RQC	RQC = CO. PRE1 .(C3 + C4 + C5)	RQC = Use Address from C
	Not Indexed. Operand required ⇒ RQC	RQC = PRE1. NINDX . PREGPRQ/1 . NANLZ + PRE1. NINDX . PREGPRQ/2 . NANLZ	
	Provide timing differential between Address to Memory and Memory Request.	ATE = RQC . MAP . TP140 . NKSC + RQC . NMAP. TP100. NKSC + MAP. NCRG. (TR240. NTR270) + NMAP. NCRG. (TR180. NTR210) + CRG. (TR100. NTR140)	
	RQC. Single Clock Mode ⇒ MRQ	MRQ = RQC . KSC	
	Indirect. C31 ⇒ 1 → LB31	LB31/2 = LMXC . C31 . NAG	
	Not Indirect. Two Operands ⇒ 1 → LB31	LB31/2 = PREFADG. LMXC . NCO	
	Some Immediate Instr ⇒ Set PH1	(S/PH1/1) = PREIM . PRE1	for LCFI, AI, LI, CBS, MBS, EBS
	All other Instructions ⇒ Set PRE2	S/PRE2 = NPREIM. PRE1 . N(S/INTRAPF)	
	Disable ENDE	ENDE = I. NENDE I . PRE1	Fast cutoff of ENDE signal
	Select Timing Signal to Rate Meter	T4RL = PREP	
		RATE/L = PRE1	
		RWDIS = PRE1	
	Trapping Logic: Illegal Opcode Slave . Priv. Inst. Non Implemented	(S/TRAP/1) = FAILL . (PRE1 . NANLZ) + FAPRIV. (PRE1. NANLZ) . NMASTER + FANIMP. (PRE1. NANLZ)	
		S/TRACC1 = FAILL. PRE1, NANLZ . NTRAP	
		S/TRACC3 = FAPRIV. PRE1. NMASTER . NANLZ . NTRAP	
		S/TR31 = FANIMP . (PRE1 . NANLZ)	

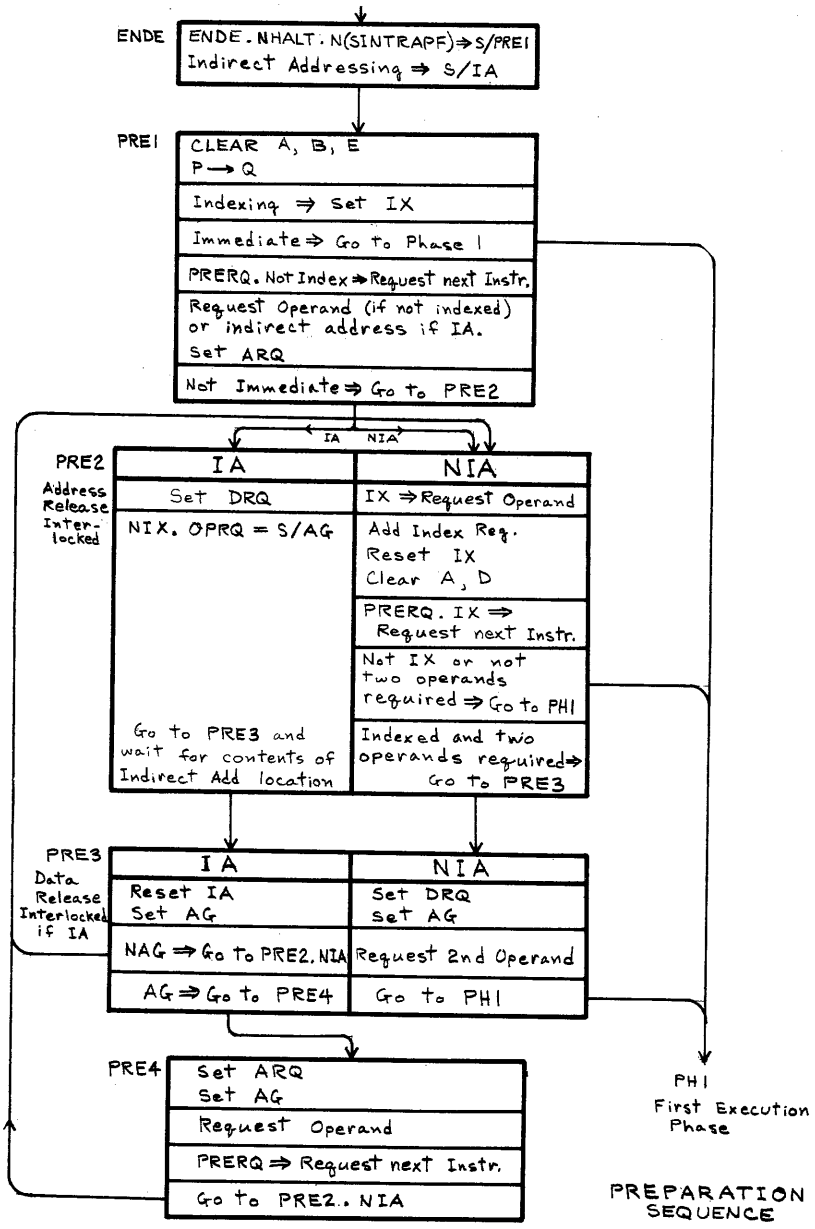
PREPARATION

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2	Not Indirect ⇒ D+A → S	SXADD = (PRE2 . NIA) . NSDIS	
	⇒ S → P	PXS = (PRE2 . NIA)	
T4RL	⇒ Reset IX	R/IX = (PRE2 . NIA)	
	⇒ Reset A	AX/1 = (PRE2. NIA)	
	⇒ Reset D	DX/1 = (PRE2. NIA)	
	Indexed . Not Indirect ⇒ Set ARQ	S/ARQ = (PRE2. NIA) . IX	
	Indexed. PRERQ ⇒ Set RQ	S/RQ = (PRE2.NIA). IX. PRERQ	
	Indirect ⇒ Set DRQ	S/DRQ = PRE2 . IA + PRE2.NIA.GPRQ . N(PREDG.IX).NFABRANCH	
	⇒ Go to PRE3	S/PRE3 = PRE2. IA. N(S/INTRAPF)	
	Two Operands . Indexed ⇒ Go to PRE3	S/PRE3 = PRE2. PREDG. IX. N(S/INTRAPF	
	Indirect. Not Indexed . Opr Req ⇒ Set AG	S/AG = PRE2. IA. NIX. GPRQ . NANLZ	via (S/AG/1)
	Not Indirect . Opr Req ⇒ MRQ	MRQ = (PRE2, NIA). PREDG. NANLZ	+ PRE2.NIA.GPRQ.NANLZ
	Not Indirect. Not Two Opr ⇒ To PH1	(S/PH1/1) = NPREDG. (PRE2 . NIA)	
Add Rel	Not Indirect. Not Indexed . Two Opr ⇒ To PH1	(S/PH1/1) = PREDG. NIX. (PRE2. NIA)	
	(S/PH1) ⇒ set EXU	S/EXU = (S/PH1/1) . NCLEAR	
	AG. PREDG ⇒ 1 → LB31	LB31/2 = LMXC . AG . PREDG	
	enable T4RL clock	T4RL = PREP	
	Two Operands. Indexed ⇒ S/LB31/1	S/LB31/1 = PRE2.NIA. PREDG. IX	
PRE3	Indirect ⇒ C → D	DXC/6 = IA. PRE3	
	Reset IA	R/IA = (PRE3 + CLEAR)	
T4RL	Indirect. NAG ⇒ To PRE2	S/PRE2 = PRE3. NAG. IA. N(S/INTRAPF)	
	AG ⇒ To PRE4	S/PRE4 = PRE3. AG . N(S/INTRAPF)	
	Set AG	S/AG = PRE3	
	Not Indirect ⇒ MRQ	MRQ = PRE3 . NIA . NANLZ	
Data Rel	⇒ Set DRQ	S/DRQ = PRE3. NIA	
	⇒ Go to PH1	(S/PH1/1) = PRE3. NIA	
	Enable T4RL Clock	T4RL = PREP	

PREP.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE4	Set AG Memory Request Go to PRE2 (with IA reset) PRERQ ⇒ Set RQ	S/AG = PRE4 RQC = PRE4 S/PRE2 = PRE4. N(S/INTRAPF) S/RQ = PRE4. PRERQ	
T4RL	Set ARQ Select T4RL Clock Two operands. LMXC. not long floating ⇒ 1 → LB31	S/ARQ = RQC T4RL = PREP LB31/2 = LMXC . AG . PREDO.N (FAFL.N02)	

PREP.



PREPARATION SEQUENCE

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP2·PCP3	NO CONTROL SWITCH ACTIVATED ⇒ Reset HALT FF.	R/HALT = .PCP2.NKAS/B	
	D → DISPLAY LIGHTS	NSXD = .PCP2.NRESET/B.NSDIS	
T&L	$\overline{\text{HALT}}$ · ANY CONTROL SWITCH, OTHER THAN CPU RESET, ACTIVATED ⇒ GO TO PCP2·PCP3	S/PCP3 = .PCP2.NHALT.NCLEAR.KAS/1.KAS/2.NPCP3 + .PCP2.NHALT.NCLEAR.CLEARMEM.NPCP3	
	INTERRUPT REQUEST · RUN · NOT PARITY ERROR HALT · NOT ADDRESS STOP HALT ⇒ Reset HALT FF	R/HALT = (INT.NDCSTOP).PCP2.KRUN/B	
	CPU RESET ⇒ X'02000000' → D	NDX/1 = .RESET	No Op Instruction → D
	⇒ X'25' → Q	S/D6 = .RESET/B S/Q26 = .RESET/B (also Q29 & Q31 QX12 = .RESET QX13 = .RESET	typical term
	⇒ 0 → PSW1 (excluding Q) ⇒ 0 → PSW2 ⇒ 0 → P ⇒ set HALT	PSW1XS = .RESET PSW2XD = .RESET PX/1 = .CLEAR S/HALT = RESET	

PCP PHASES

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP2·PCP3	Q → P	NPXQ = .PCP3	
	INSERT PSW1 or PSW2 ⇒ D → S → A	NSXD = .PCP3.KPSW/B NAXS = .PCP3.KPSW/B	
T&L	GO TO PCP4	S/PCP4 = .PCP3.(NPCP7.NENDE) R/PCP2 = .PCP3 R/PCP3 = . S/T&L = PCP3.(KPSW/B + NKIDLE/B)	

PCP

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP4	CLEAR DATA ⇒ 0 ↦ D	NDX/1 = .PCP4.KCLEARD/B	
	ENTER DATA ⇒ DATA SWITCHES ↦ D	NDXK = .PCP4.KENTERD/B	
TGL/TBL	DISPLAY SELECT ADDRESS + STORE SELECT ADDRESS ⇒ SELECT ADDRESS ↦ P	NPXK = +.PCP4.KDISPLAK/B +.PCP4.KSTORK/B	
	LOAD ⇒ 20_{16} ⇒ P	NPX20 = .PCP4.KFILL/B	
	RUN + STEP ⇒ D → S ↦ C	NSXD = .PCP4.NKIDLE/B N(S/CXS) = .PCP3.NKIDLE/B	
	INSERT PSW1 ⇒ P → S → C ↦ D	NSXP = .PCP4.KPSW1/B N(S/CXS) = .PCP3.KPSW/B NDXC/6 = .PCP4.KPSW1/B	
	⇒ PSW1 ↦ D	NDXPSW1 = .PCP4.KPSW1/B	
	INSERT PSW2 ⇒ PSW2 ↦ D	NDXPSW2 = .PCP4.KPSW2/B	
	INCREMENT INST. ADDR ⇒ P+1 ↦ P ⇒ Set DRQ ⇒ Set Request	NPCTP1 = .PCP4.KINCRE/B N(S/DRQ) = .PCP4.KINCRE/B NMRQ = .PCP.(S/DRQ)	
	DISPLAY OR STORE ⇒ Set DRQ ⇒ Set Request	N(S/DRQ) = +.PCP4.KSTOR/B +.PCP4.KDISPLA/B	
	GO TO PCP5	R/PCP4 = . S/PCP5 = .PCP4.(NPCP7.NENDE)	

PCP

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP5	RUN + STEP ⇒ ENDE ⇒ P+1 ↦ P ⇒ C ↦ D ⇒ C ↦ R TGL ⇒ C ↦ Ø ⇒ GO TO PRE1	NENDE = .PCP5.NKIDLE/B PCTP1 = .ENDE NDXC/6 = .ENDE NØRXC = .ENDE NØXC = .ENDE S/PRE1 = .ENDE.NHALT.N(S/INTRAPF)	Implemented as: PCTP1=I.NPCTP1.NENDE I.PCTP1DIS
	STORE ⇒ D → S → MB ⇒ Write Word ⇒ Inhibit Protect Fail	NSXD = .PCP5.KSTOR/B NMW = .PCP5.KSTOR/B PRØTD = .FAILD.NPCP5	
	DISPLAY ⇒ C ↦ D	NDXC/6 = .PCP5.KDISPLA/B	
	INCREMENT INST ADDR ⇒ C ↦ D ⇒ P ↦ Q	NDXC/6 = .PCP5.KINCRE/B NQXP = .PCP5.KINCRE/B	
	INSERT PSW1 OR PSW2 ⇒ DATA SWITCHES ↦ D	NDXK = .PCP5.KPSW/B	
	LOAD ⇒ Set DRQ ⇒ Set Request	N(S/DRQ) = .PCP5.KFILL/B NMRQ = .PCP.(S/DRQ)	
	$\overline{\text{RUN OR STEP}}$ ⇒ GO TO PCP6·$\overline{\text{PCP7}}$	S/PCP6 = .PCP5.(NPCP7.NENDE) R/PCP5 = .	
Data Rel.			

PCP

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP6·PCP7	INSERT PSW1 ⇒ D→S↠P	NSXD = .PCP6.NPCP7.KPSW1/B.NDIS	
	⇒ S↠PSW1	NPXS = .PCP6.NPCP7.KPSW1/B	
		PSW1XS = .PCP6.NPCP7.KPSW1/B	
T6L	INSERT PSW2 ⇒ D↠PSW2	PSW2XD = .PCP6.NPCP7.KPSW2/B	
	CLEAR MEMORY ⇒ Set DRQ	N(S/DRQ) = .PCP6.CLEARMEM	
	⇒ Set Request	NMRQ = .PCP.(S/DRQ)	
	⇒ Write Word	NMW = .PCP6.CLEARMEM	
	⇒ P+1↠P	NPCTP1 = .PCP6.CLEARMEM	
	⇒ Inhibit Crossover	CRO = .NCLEARMEM.(Address 0 thru 15)	
	⇒ Inhibit Protect Fail	PROTD = .FAILD.NPCP6	
	⇒ Stay in PCP6	S/PCP7 = (S/PCP7).(NPCP7.NENDE)	
		(S/PCP7) = .(PCP6.NCLEARMEM).NKFILL/B	
	⇒ P→Q	(R/PCP6) = .PCP7	
		QXP = CLEARMEM	
	LOAD ⇒ Set DRQ	N(S/DRQ) = .PCP6.NPCP7.KFILL/B	
	⇒ Set Request	NMRQ = .PCP.(S/DRQ)	
	⇒ Write Word	NMW = .PCP6.KFILL/B	
	⇒ P+1↠P	NPCTP1 = .PCP6.KFILL/B	
	⇒ Force Load Bootstrap on S	S7I = .KFILL/B.P28	typical term
	⇒ Inhibit Protect Fail	PROTD = .FAILD.NPCP6	
	⇒ GO TO PCP6·PCP7 for write in 29_{16}	(S/PCP7) = .P28.PCP6.NCLEARMEM	
	$\overline{\text{CLEAR MEMORY+FILL}}$ ⇒ GO TO PCP6·PCP7	(S/PCP7) = .(PCP6.NCLEARMEM).NKFILL/B	
	Reset AHCL	AHCL = N(NAHCL)	
Data Rel.		NAHCL = N(AHCL.NPCP6)	

PCP

5 of 6

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP6·PCP7	CLEAR MEMORY ⇒ SAME AS PCP6·NPCP7		
	LOAD ⇒ Write Word	NMW = .PCP6.KFILL/B	
	⇒ P+1↠P	NPCTP1 = .PCP6.KFILL/B	
T6L	⇒ Force Fill Bootstrap on S	S7I = .KFILL/B.P28	typical term
	⇒ Inhibit Protect Fail	PROTD = .FAILD.NPCP6	
	GO TO PCP1	(S/PCP1) = .PCP7.NPCP3	
		(R/PCP6) = .PCP7 + RESET	
		(R/PCP7) = .	
	INSERT PSW1 or PSW2 ⇒	DXC/6 = .PCP7.KPSW/B	
	A→S→C↠D	(S/CXS) = .PCP6.KPSW/B	
		SXA = .PCP7.KPSW/B	
	P→Q	QXP = .PCP7.KPSW/B	
Data Rel.			
PCP1	$\overline{\text{Set INTRAPF FF}}$ ⇒ Go to PCP2.$\overline{\text{PCP3}}$	(S/PCP2) = .PCP1.NCLEAR.NPCP3	
		R/PCP1 = .	
T6L	set HALT F.F.	S/HALT = .PCP1	

PCP

6 of 6

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	LI (22) LW (32) LH (52) LB (72) LCW (3A) LCH (5A) ● FAS26 (also covers LAW) ● ● ● ● ● ● FAS23 ● ● ● ● ● ● FAS21 (" " LAW, LAH) ● ● ● ● ● PRERQ (" " " " , LCF, AW, AH, SW, SH, CW, CH, CB, CLR, EOR, OR, AND)		
ENDP	LI ⇒ { C12 ↛ D0011 , C1231 ↛ D1231 MRQ (access next intr. f(P)) NLI ⇒ S/RQ (" " " f(Q))	DXC/4 = OU2.(N04.N05.N07).PRE1 MRQ = (OU2.(N04.N05.N07)+(N01.N03).0L2).NANL2.PRE1 S/RQ : f(PRERQ)	(also covers AI) (also covers AI, LCFI)
PHI T4RL	NLI ⇒ Q ↛ P LI ⇒ (D already contains info) LW ⇒ C ↛ D LH ⇒ C_H ↛ D (down-aligned) LB ⇒ C_B ↛ D (" - ") LCW ⇒ NC ↛ D, 1 ↛ CS31 LCH ⇒ { NC_H ↛ D (down-aligned) 1 ↛ CS31 S/DRQ S/PHS S/T10L	PXQ = PRERQ.PHI DXC/6 = OU3.(N04.N05).PHI DXC/5 = OU5.(").PHI DXCBP = OU7.(").PHI DXNC/1 = (N01.03).(04.N05.N07).PHI DXNC/3 = (OU5.(").PHI) CSX1/8 = (") S/DRQ = (FAS23.PHI) BRPHS = (") S/T10L = (")	(also covers AW, CW, MTW) (" " AH, CH, MTH) (" " LCF, CB, MTB) (" " SD, LCD, SW) (" " SH) (" " ")
PHS T10L	D+CS31 → S ENDE S ↛ RW S ↛ A } CC3,4 contr. S/TESTA LCW ⇒ { 1 ↛ CC2 if overflow TRAP to 67 if overflow.AM	SXADD = (FAS21.PHS); ENDE = FAS21.PHS RW = (") AXS = (") S/TESTA = (") S/CC2 = PROBEOVER.OVER; RCC2 = PROBEOVER S/TRAP = TROVER; S/TR30 = S/TR31 = TROVER.N(S/TRACC4/1)	{ OVER = f(ND0.K0) PROBEOVER = FAS26.PHS TROVER = PROBEOVER.OVER.AM

LI (22), LW (32), LH (52), LB (72), LCW (3A), LCH (5A) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	LAW LAH (Load Absolute Values) (3B) (5B) ● ● FAS10 ● ● FAS21 (also covers LI, LW, LH, LB, LCW, LCH) ● FAS26 (" " LCW) ● ● PRERQ (" " LW, LH, LB, LCW, LCH, LCF, AW, AH, SW, SH, CW, CH, CB, CLR, EOR, OR, AND)		
ENDP	S/RQ (access next instr. f(Q))	S/RQ : f(PRERQ)	
PHI T4RL	MB ↛ C C ↛ D if LAW C_H ↛ D " LAH (down-aligned) Q ↛ P S/T4L	(preparation) DXC/6 = (N01.03).0LB.PHI DXC/5 = OU5.(N05.06.07).PHI PXQ = PRERQ.PHI S/T4L = FAS10.PHI	(also covers LAD) (" " MTH)
PH2 T4L	S/NGX if D is negative S/DRQ S/PHS S/T10L	S/NGX = (FAS10.PH2).D0 S/DRQ = (") S/PHS = (") S/T10L = (")	
PHS T10L	ENDE D → S if D is pos. -D → S " " " neg. S ↛ RW S ↛ A } CC3,4 contr. S/TESTA LAW ⇒ { 0 ↛ CC2 1 ↛ CC2 if overflow TRAP to 67 if overflow.AM } EW = -2^{31}	ENDE = (FAS21.PHS) SXADD = (") + NFAMDSF.NGX ← via SXADD/1 (negate case) RW = (") AXS = (") S/TESTA = (") R/CC2 = PROBEOVER = FAS26.PHS S/CC2 = FAS10.PHS.K0 ← (impossible if LAH) S/TRAP = TROVER; S/TR30 = S/TR31 = TROVER.N(S/TRACC4/1)	{ (note: OVER is inhibited by FAS10 since it is unsuitable for negation using NGX) (TROVER = FAS10.PHS.AM.K0)

LAW (3B), LAH (5B) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	LD (12) Load Doubleword	PRED0, FAS16, FULD	
PH1 T4RL	MBv1 → C C → D s/LR31/2 s/RQ (access next instr. f(Q)) s/DRQ (for access EW) s/PH3 s/T8L	(prep.) DXC/6 = (N01.03).(N04.N05.N07).PH1 s/LR31/2 = (0U1.(N05.N07).PH1) s/RQ = (") s/DRQ = PRED0.PH1 BRPH3 = (FULD.PH1) s/T8L = (")	(also covers AD, AW, LW) (" " AD, SD, LCD) (" " " " ") (request set by prep.)
PH3 T8L	D → S (EWv1) S → RWv1 S → A (for zero test) MB → C C → D Q → P s/DRQ s/PH5 s/T8L	SXD = (FULD.PH3) RW = (") AXS = (") (see PH1) DXC/6 = (") PXQ = PRED0.PH1 s/DRQ = (FAS16.PH3) BRPH5 = (") s/T8L = FULD.PH3	
PH5 T8L	ENDE D → S (EW) S → RW S → A 1 → A31 if A ≠ 0 } CC3, 4 contr. s/TESTA (note: A contains EWv1)	ENDE = (FAS16.PH5) SXD = FULD.PH5 RW = (FAS16.PH5) AXS = (") A31X1 = FULD.PH5.NA0031Z s/TESTA = FAS16.PH5	

LD (12) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	Load Conditions and Floating Control: LCF (70) LCFI (02)	FAS17 = 0U7.0L0 + 0U0.0L2 PRERQ = (01.03).(N04.N05).N0L3.NANL2	
ENDP	s/RQ if LCF (access next instr f(Q)) MRQ if LCFI (" " f(P))	s/RQ = f(PRERQ) MRQ = (N01.N03).0L2.NANL2.PRE1	(also covers LI)
PH1 T4RL	MB → C ; CB → D (down-aligned) if LCF (D contains instr. if LCFI) s/NPRX (for D2431 → S0007 in PH2) s/DRQ Q → P if LCF s/T8L (for buffered s's → cc's)	DXCBP = 0U7.(N04.N05).PH1 (Dxc/6 raised by previous ENDE) s/NPRX = FAS17.PH1 s/DRQ = FAS17.PH1 PXQ = PH1.PRERQ s/T8L = FAS17.PH1	(align. f(P32,P33)) (already down aligned)
PH2 T8L	ENDE D2431 → K2330 → S0007 S0 → CC1 S1 → CC2 S2 → CC3 S3 → CC4 } if BIT 10 = 1 (stored in R30) S5 → FS S6 → FZ S7 → FNF } if BIT 11 = 1 (stored in R31)	ENDE = (FAS17.PH2) SXUAB = (") s/CC1 = S0.CCXS s/CC2 = S1.CCXS s/CC3 = S2.CCXS s/CC4 = S3.CCXS } CCXS = FAS17.PH2.R30 s/FS = S5.FSZNXS s/FZ = S6.FSZNXS s/FNF = S7.FSZNXS } FSZNXS = FAS17.PH2.R31	(up-align byte) resets = CCXS resets = FSZNXS

LCF(70), LCFI(02) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	LAD (1B) Load Absolute Double word	FAS16, FAS19, PREDO, FULAD	
PH1 T&RL	MB∨1 → C C → D S/NGX S/DRQ (for access EW)	(prep.) DXC/6 = (NO1.03).0LB S/NGX = FULAD.PH1 S/DRQ = (PREDO.PH1)	(request set by prep.)
PH2 T6L	-D → S S → A NK00 → S00 → K00H MB → C S/LR31/2 S/T8L	SXADD = FAS16.PH2 + NGX.NFAMDSF AXS = FAS16.PH2 (normal action) (see PH1) S/LR31/2 = (FULAD.PH2) S/T8L = (")	
PH3 T8L	NC0 ⇒ D → S C0 ⇒ A → S S → RW∨1 S/BWZ if A≠0 (for 1 → A31 in PH5) 0 → A NC0 ⇒ C → D C0 ⇒ NC → D NK00H → CS31 (stored end carry)	SXD = (FULAD.PH3).NC0 SXA = (").C0 RW = (") S/BWZ = NA0031Z.(S/BWZ/1) ← FAS16.NOL9.PH3; R/BWZ = CLEAR AX/1 = FAS19.PH3 DXC/6 = (FULAD.PH3).NC0 DXNC = (").C0 CSX1/8 = (").C0.NK00H	+EW ⇒ +(EW∨1) → R∨1 - " ⇒ -(") → R∨1

LAD (1B) 1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3	(Continued) Q → P MRQ/1 S/DRQ } access next instr. S/PH5 S/T10L	PXQ = MRQ/1 + PREDO.PH3 MRQ/1 = FULAD.PH3 S/DRQ = (FAS16.PH3) BRPH5 = (") S/T10L = FAS19.PH3	} redundant Q → P is for mechanization convenience
PH5 T10L	ENDE D+CS31 → S S → RW CC3,4 contr. { S → A; 1 → A31 if A≠0 in PH3; S/TESTA } 0 → CC2 1 → CC2 if overflow TRAP to 67 if overflow.AM	ENDE = (FAS16.PH5) SXADD = FAS19.PH5 RW = (FAS16.PH5) AXS = (") A31X1 = FAS16.BWZ S/TESTA = FAS16.PH5 R/CC2 = (PROBEOVER) = FAS19.PH5 S/CC2 = (").OVER S/TRAP = (TROVER) S/TR30 = (").N(S/TRACC4/1) S/TR31 = (").N(") where (") = PROBEOVER.AM.OVER	+EW ⇒ EW → R -EW ⇒ NEW+e.c. → R (render A≠0 if l.s.w. ≠0) OVER = f(NA0.ND0.K0) assured by: 0 → A C → D if NC0 NC → D if C0 } in PH3

LAD 2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	LS (4A): EW∧(Rv1) ∪ R∧N(Rv1) → R (reduces to EW∧R → R if R is odd), >0 → CC3, <0 → CC4		FAS6, FAS15
ENDP	S/LR31/2	S/LR31/2 = PRE2, NIA, FULS	
PH1 T4RL	MB → C C → D RRv1 → A S/NPRX (for A∧D → S) S/T4L	(prep.) DXC/6 = FAS6, PH1 AXRR = FAS15, PH1 S/NPRX = FAS6, PH1 S/T4L = FULS, PH1	
PH2 T4L	A∧D → S S → B } EW∧(Rv1) → B 1's → CS 0 → D } for N(Rv1) → C → D S/CXS S/PH4	SXPR = NPRX BXS = FAS6, PH2 CSX1 = (FULS, PH2) DX/1 = (") S/CXS = (") BRPH4 = (")	
PH4 T6L	A ⊕ CS → S S → C C → D } N(Rv1) → D RR → A S/NPRX S/DRQ MRQ/1 } for next instr. S/T8L	SXPR = (FULS, PH4) (CXS is on) DXC/6 = (") AXRR = (") S/NPRX = (") S/DRQ = (FAS6, PH4) MRQ/1 = (") S/T8L = FULS, PH4	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T8L	ENDE A∧D → S (R∧N(Rv1)) B → S (EW∧(Rv1)) S → RW S → A S/TESTA	ENDE = FAS15, PH5 SXPR = NPRX SXB = FULS, PH5 RW = (01, N02), (04, N05, 06), PH5 AXS = (FAS15, PH5) S/TESTA = (")	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	Exchange Word : (R ↔ EWL) XW (46)	FAS9 = 004.0L6	(also covers EOR, OR, AND)
PH1 T4RL	MB →→ C C →→ D RR →→ A MRQ, S/DRQ } for write S/T8L (insig considering DRQ; mech. conv.)	(preparation) DXC/6 = (FAS9.PH1) AXRR = (") MRQ = FUXW.PH1 S/DRQ = (FAS9.PH1) S/T8L = (")	(also covers AWM) (needed for EOR, OR, AND)
PH2 T8L	A → S S →→ MB (R →→ EWL) MRQ/1, S/DRQ } for next instr. S/T8L	SXA = (FAS9.PH2) MW = (") MRQ/1 = (") S/DRQ = (") S/T8L = (")	(FAS9.PH2 implies XW only since EOR, OR, and AND skip PH2)
PH3 T8L	ENDE D → S S →→ RW (EW →→ R) S →→ A, S/TESTA } CC3,4 contr.	ENDE = (FAS9.PH3) SXD = FUXW.PH3 RW = (FAS9.PH3) AXS = (") S/TESTA = (")	

XW (46)

1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
ENDP minus 1	S/LR31/2	S/LR31/2 = (PRE1.NIA + PRE3.IA).FAFRR ← 001.0L5	
ENDP	RRv1 →→ A MRQ, S/DRQ } for write S/LB31/1	AXRR = FAS14.(PRE2.NIA) MRQ = (FUSTD.(").NANL2) S/DRQ = (") S/LB31/1 = (")	
PH1 T4RL	A → S, S →→ MBv1 } Rv1 →→ MWv1 RR →→ A MRQ, S/DRQ } for write S/PH4 S/T4L (insig)	SXA = (N01.03).0L5.PH1 MW = (N01.03).0L5.PH1 AXRR = (FUSTD.PH1) MRQ = (") S/DRQ = 03.0L5.PH1 BRPH4 = (FUSTD.PH1) S/T4L = FAS14.PH1	(for STS)
PH4 T4L	A → S, S →→ MB } R → MW MRQ/1, S/DRQ } for next instr	SXA = FUSTD.PH4 MW = FAS14.PH4 MRQ/1 = (FUSTD.PH4) S/DRQ = (")	
PH5 T6L	ENDE	ENDE = FAS14.PH5	
NOTES	STD (15) covered by FAS14 and FUSTD ↑ also covers STS		

STD (15)

1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	STW (35), STH (55), STB (75), STCF (74)	FAS18 = 0U7 (N04.05.N06) + (02.03).0L5 +	(01.03).0L5
PRE2.NIA:			
	NSTCF ⇒ RR → A	AXRR = PRE2.NIA.FAS18.0L5	(info is down-aligned if STH,STB)
	STCF ⇒ CC1 → D24	S/D24 = CC1 (FUSTCF.PRE2.NIA.NANL2)	( )=0L4.0U7.PRE2.NIA.NANL2
	CC2 → D25	S/D25 = CC2 (")	
	CC3 → D26	S/D26 = CC3 (")	0 → D caused by DX/1 = PRE2.NIA
	CC4 → D27	S/D27 = CC4 (")	
	FS → D29	S/D29 = FS (")	
	FZ → D30	S/D30 = FZ (")	
	FNF → D31	S/D31 = FNF (")	
	S/NPRX (for upward aligning)	S/NPRX = (PRE2.NIA.FAS18.NANL2)	(insig if STW)
	MRQ, S/DRQ (for WRITE)	MRQ = ("); S/DRQ = (")	
	will be > T6L because of		
PH1 T4RL	STW ⇒ A → S; S → MW	SXA = ((N01.03).0L5.PH1)	(SXPR is high because of NPRX
		MW = (")	but D=0; also A∨(A∧D)=A)
	note: in the STH, STB, and STCF cases the K (carries) signals = A/D x 2 (i.e. $(A/D)_n \rightarrow G_n \rightarrow K_{n-1}$), caused by NPRX		
	STH ⇒ A1631 → K1530 → S0015; " → S1631; S → MH	SXUAH = (FUSTH.PH1)	
		MWH = (")	
	STB, STCF ⇒ A/D2431 → K2330 → S0007; " → " → S0815; " → " → S1623; " → S2431; S → MB	SXUAB = (0U7.(N04.05.N06).PH1)	
		MWB = (")	

STW (35), STH (55), STB (75), STCF (74) 1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1	(Continued)		
	1 → CS15 (for ½WD "significance test")	S/CS15 = FUSTH.PH1	
	MRQ/1, S/DRQ for next instr.	MRQ/1 = (FAS18.PH1) + 03.0L5.PH1; S/DRQ = (")	
	S/T8L if STH (for "significance" test)	S/T8L = FUSTH.PH1	
PH2 T8L	ENDE	ENDE = FAS18.PH2	
	STH ⇒ 0 → CC2; 1 → CC2 if signifance in A0015	R/CC2 = (FUSTH.PH2)	
		S/CC2 = (")(A16.NK00 + NA16.NA0015≠)	(A16 = 1).(A0015 ≠ all 1's); (A16 = 0).(A0015 ≠ all 0's)

STW, etc. 2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	STS(47): R ∧ (RvI) v EW ∧ N(RvI) → EWL	(reduces to R v EW → EWL if R is odd) FAS	6 and FAS14 are high
ENDP minus 1	s/LR31/2	s/LR31/2 = (PRE1.NIA + PRE3.IX).FAFRR/1 ← 0U4.0L7	
ENDP	RRvI → A 1's → CS	AXRR = FAS14.(PRE2.NIA) CSXI = FUSTS.(")	
PH1 T4RL	A ⊕ CS → S S → A } N(RvI) → A MB → C C → D } EW → D s/NPRX s/CXRR s/TIOL	SXPR = (FUSTS.PH1) AXS = (") (prep.) DXC/6 = (FAS6.PH1) s/NPRX = (") s/CXRR = 0U4(N04.05.07) s/TIOL = (s/CXRR)	
PH2 TIOL	A ∧ D → S S → B } EW ∧ N(RvI) → B RR → C s/LR31/2 enable T4RL	SXPR = NPRX BXS = FAS6.PH2 (CXRR is on) s/LR31/2 = (FUSTS.PH2) T4RL = (")	
PH3 T4RL	RRvI → A (RvI → A) C → D (R → D) s/NPRX MRQ, s/DRQ } for write	AXRR = (FUSTS.PH3) DXC/6 = (") s/NPRX = (") MRQ = (") s/DRQ = (")	

STS (47)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T6L	A ∧ D → S (R ∧ (RvI)) B → S (EW ∧ N(RvI)) S → MB MRQ/1, s/DRQ } access next instr.	SXPR = NPRX SXB = 0U4.(N04.05.07).PH4 MW = FAS14.PH4 MRQ/1 = (FAS6.PH4) s/DRQ = (")	
PH5 T6L	ENDE	ENDE = FAS14.PH5	

STS

2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	AI (20) AW (30) AH (50) SW (38) SH (58)	FAS12 = (N01.02).0L0 + (0U3 + 0U5).(N05.N06.N07)	
		PRERQ = (0U3.N05 + (01.03).(N04.N05) + (01.N02)).0LB).N0L3.NANL≠	
ENDP	s/RQ if NAI (access next instr. f (Q)) MRQ if AI (" " " f (P)) [C12 → D0011, C1231 → D1231] if AI	s/RQ : f (PRERQ) MRQ = 0U2.(N04.N05.N07).NANL≠.PRE1 DXC/4 = 0U2.(").PRE1	(also covers LI) (")
PH1 T4RL	AW ⇒ C → D SW ⇒ NC → D, 1 → CS31 AH ⇒ {C0C16 → D0015; C0015 → D1631 if NP32; C1631 → D1631 if P32} SH ⇒ {NC0C16 → D0015; NC0015 → D1631 if NP32; NC1631 → D1631 if P32; 1 → CS31} RR → A Q → P if NIA s/PHS s/DRQ (for next instr) s/T10L	DXC/4 = 0U3.(N04.N05).PH1 DXNC/1 = (N01.03).(04.N05.N07) (down-align ½ WD): DXC/5 = 0U5.(N04.N05).PH1 (causes DXC/2 if P32 or DXCR16 if NP32) (down-align inverted ½ WD): DXNC/3 = 0U5.(04.N05.N07).PH1 (causes DXNC/2 if P32 or DXNCR16 if NP32) CSX1/8 = 0U5.(04.N05.N07).PH1 AXRR = (FAS12.PH1) PXQ = PRERQ.PH1 BRPHS = (FAS12.PH1) s/DRQ = (") s/T10L = (")	(also covers CW, LW, MTW) (" " SD, LCD, LCW) (also covers CH, LH, MTH) (also covers LCH) (also covers LCH)

AI (20), AW (30), AH (50), SW (38), SH (58)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PHS T10L	ENDE A + D + CS31 → S S → RW S → A s/TESTA } CC3,4 control 0 → CC1 1 → CC1 if end carry 0 → CC2 1 → CC2 if overflow s/TRAP s/TR30 s/TR31 } trap to 67 if overflow.AM	ENDE = (FAS12.PHS) SXADD = (") RW = (") AXS = (") s/TESTA = (") R/CC1 = (") S/CC1 = (").K00 R/CC2 = (PROBEOVER) = FAS12.PHS S/CC2 = (").OVER S/TRAP = (TROVER) S/TR30 = (").(N(s/TRACC4/1)) s/TR31 = (").(")	OVER = f(A0.D0.NK0 + NA0.ND0.K0) TROVER = OVER.AM.PROBEOVER
Clock following ENDE TESTA functions: 0 → CC3 1 → CC3 if A > 0 0 → CC4 1 → CC4 if A < 0	(TESTA was set last clock) R/CC3 = (TESTA) S/CC3 = (").NTESTA/1.NA0.NA0031≠ R/CC4 = (") S/CC4 = (").NTESTA/1.A0	

AI, etc.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	AD (10) SD (18) LCD (1A) CD (11) (add, subtract, load complement, compare) - doubleword. FAS2 3 16 22 PRED0	(also covers LD, LAD) (" " CLM, CLR) (" " CLM, LD, LAD)	
ENDP	s/LR31/2 if NLCD	s/LR31/2 = (S/PH1/1).(FAS3 + OU1.(N04.N05.N06))	
PH1 T4RL	MBv1 → C C → D if AD NC → D, 1 → CS31 if NAD RRv1 → A if NLCD (A contains 0 if LCD) s/LR31/2 (for write) if NCD S/DRQ (for access EW) s/RQ (initiate next instr. access f(Q)) s/T10L if NCD	(prep.) DXC/6 = (N01.03).(N04.N05.N07).PH1 DXNC/1 = ((").(04.N05.N07)+FAS2).PH1 AXRR = FAS22.PH1 s/LR31/2 = OU1.(N05.N07).PH1 S/DRQ = (PRED0.PH1) s/RQ = OU1((N05.N07)+(N05.N06)).PH1 S/T10L = (FAS3 + FULCD).PH1	+ EWv1 → D, CS31 - EWv1 → D + Rv1 → A (request set by prep.)
PH2 T10L / T6L	Timing: T10L if NCD; T6L if CD A + D + CS31 → S S → RWv1 if NCD S → A (for zero test) NK00 → S00 → K00H (store end carry info.)	SXADD = (FAS16 + FAS22).PH2 RW = (FAS3 + OU1.(04.N05.N07)).PH2 AXS = (FAS16 + FAS22).PH2 (normal action)	Rv1 + EWv1 if AD Rv1 - EWv1 if SD, CD - EWv1 if LCD

AD (10), SD (18), LCD (1A), CD (11)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2	Continued MB → C C → D if AD NC → D if NAD ENABLE T6RL if AD, SD S/T4L if LCD	(see PH1) DXC/6 = FUAD.PH2 DXNC = (FAS2 + FULCD).PH2 T6RL = FAS3.PH2 S/T4L = FULCD.PH2	
PH3 T4L / T6L / T6RL	Timing: T4L if LCD, T6L if CD, T6RL S/BWZ if A ≠ 0 (for 1 → A31 in PH5) 0 → A if LCD RR → A if NLCD NK00H → CS31 (stored end carry) Q → P S/DRQ S/PH5 S/T10L if NCD	if AD, SD. S/BWZ = NA0031Z.(S/BWZ/1) ← (FAS16 + FAS22).N01.9.PH3; R/BWZ = CLEAR AX/1 = FAS19.PH3 AXRR = FAS22.PH3 CSX1/8 = NK00H.(S/BWZ/1) ← PXQ = (PRED0.PH3) S/DRQ = (") BRPH5 = (FAS16 + FAS22).PH3 S/T10L = (FAS19 + FAS3).PH3	
PH5 T10L / T6L	Timing: T10L if NCD, T6L if CD ENDE A + D + CS31 → S S → RW if NCD CC3,4 contr { S → A; 1 → A31 if A ≠ 0 in PH3; S/TESTA; (A0⊕D0⊕K00) → S00 → K00H; S/TESTA/1 if CD } 1 → CC1 if end-carry if AD, SD 1 → CC2 if overflow if NCD TRAP to 67 if overflow. AM if NCD	ENDE = (FAS16 + FAS22).PH5 SXADD = (FAS19 + FAS22).PH5 RW = (FAS3 + FAS16).PH5 AXS = ((FAS16 + FAS22).PH5) A31X1 = (") S/TESTA = ( ) S00XN = FAS22.PH5 S/TESTA/1 = (S/TESTA).(N04.N06.07) S/CC1 = FAS3.PH5.K00; R/CC1 = FAS3.PH5 S/CC2 = PROBEOVER.OVER; R/CC2 = PROBEOVER ← S/TRAP, S/TR30, S/TR31 : f(TROVER) ← PROBE	(render A ≠ 0 if l.s.w ≠ 0) (signif. if CD; means R < EW) causes NK00H.(NA0031) → CC3 " K00H.(") → CC4 (FAS3 + FAS19).PH5 OVER.AM.OVER

AD, etc.

2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	CI (21), CW (31), CH (51), CB (71)	FAS11 PRERQ	
PRE2.NIA	RR → A if NCB RR2431 → A2431, 0's → A0023 1's → CS (for NA → B) S/RQ if NCI (access next instr. f(Q)) 0 → D (for A⊕CS → PR → S → B in PH1)	AXRR = (PRE2.NIA).FAS11.NFUCB AXRR/3 = (").FUCB CSX1 = (").FAS11 S/RQ : f(PRERQ) DX/1 = PRE2.NIA	(needed for CI)
PH1 T4RL	MB → C if NCI CW ⇒ C → D CH ⇒ C0C16 → D0015 (sign pad) C0015 → D1631 if NP32 C1631 → D1631 if P32 CB ⇒ (0 → D0023) if P32 P33 = 0 0 C0007 → D2431 " = 0 1 C0815 → " " = 1 0 C1623 → " " = 1 1 CI ⇒ C12 → D0011, C1231 → D1231 A⊕CS → S, S → B } (NR → B) S/NPRX (for A∧D → S → A) S/T4L MRQ/1 if CI Q → P	(prep.) DXC/6 = OU3.(N04.N05).PH1 down-align ½WD: DXC/5 = OU5.(N04.N05).PH1 DXCR16 = NP32.(DXC/5) DXC/2 = P32.(") down-align byte: DXCBP = OU7.(N04.N05).PH1 DXCR24 = NP32.NP33.(DXCBP) DXCR16/1 = NP32.P33.(") DXCR8 = P32.NP33.(") DXC/1 = P32.P33.(") DXC/4 = FUCI.PH1 SXPR = (FAS11.PH1) BXS = (") S/NPRX = (") S/T4L = (") MRQ/1 = FUCI.PH1 PXQ = MRQ/1 + PRERQ.PH1	(also covers AW, LW, MTW) (also covers AH, LH, MTH) (also covers LCF, LB, MTB) (CS contains all 1's)

CI (21), CW (31), CH (51), CB (71)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 T4L	A∧D → S, S → A } (R∧EW → A) S/T4L	SXPR = NPRX AXS = (FAS11.PH2) S/T4L = (")	
PH3 T4L	0 → CC2 1 → CC2 if A≠0 (i.e. if R∧EW ≠ 0) B → S, S → A, 1 → CS31 } (-R → A, CS31) S/DRQ	R/CC2 = (FAS11.PH3) S/CC2 = (").NA0031Z SXB = (") AXS = (") CSX1/8 = (") S/DRQ = (")	
PH4 T6L	ENDE A+D+CS31 → S, S → A } (EW-R → A) S/K00H if N(A0⊕D0⊕K00) (i.e. EW ≥ R) S/TESTA S/TESTA/1	ENDE = (FAS11.PH4) SXADD = (") AXS = (") S/K00H = S00 = NK00⊕S00X, where S00X = (A0⊕D0).FAS11.NS00XN S/TESTA = FAS11.PH4 S/TESTA/1 = (S/TESTA).(N04.N06.07)	
Clock following ENDE: 0 → CC3 1 → CC3 if R > EW 0 → CC4 1 → CC4 if EW > R		R/CC3 = TESTA S/CC3 = TESTA/1.NK00H.NA0031Z R/CC4 = TESTA S/CC4 = TESTA/1.K00H.NA0031Z	

CI, etc.

2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
X	CS(45): R∧(Rv1) : EW∧(Rv1) (red. to R:E∧R if R is odd) ; R<EW → CC4 , R>EW → CC3; FAS6,15 are high		
ENDP	S/LR31/2	S/LR31 = PRE2.NIA.FUCS	
PH1 T4RL	MB → C C → D RRv1 → A S/NPRX S/CXRR S/T10L (for RR → C)	(prep.) DXC/6 = FAS6.PH1 AXRR = FAS15.PH1 S/NPRX = FAS6.PH1 S/CXRR = 004.(N04.05.07).PH1 S/T10L = (S/CXRR)	
PH2 T10L	A∧D → S S → B } EW∧(Rv1) → B RR → C C → D } R → D S/NPRX S/T4L	SXPR = NPRX BXS = FAS6.PH2 (CXRR is on) DXC/6 = (FUCS.PH2) S/NPRX = (") S/T4L = (")	
PH3 T4L	A∧D → S S → A } R∧(Rv1) → A S/CXS	SXPR = NPRX AXS = (FUCS.PH3) S/CXS = (")	
PH4 T6L	B → S S → C NC → D 1 → CS31 } -[EW∧(Rv1)] → D, CS31 MRQ/1 S/DRQ } for next instr.	SXB = 004.(N04,05,07).PH4 (CXS is on) DXNC/1 = FUCS.PH4 MRQ/1 = (FAS6.PH4) S/DRQ = (")	

CS (4S)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T6L	ENDE A+D+CS31 → S S → A } R-EW → A (masked) NK00 → S00 → K00H (result negative) S/TESTA (for CC3,4 contr.) S/TESTA/1	ENDE = FAS15.PH5 SXADD = FUCS.PH5 AXS = FAS15.PH5 (normal action) S/TESTA = FAS15.PH5 S/TESTA/1 = (S/TESTA).(N04.N06.07)	(note: selected contents of R and EW are treated as positive integer magnitudes)
Clock following ENDE : S/CC3 if K00H.NA0031≠ (R > EW) S/CC4 if K00H. " (R < EW)		S/CC3/1 = (TESTA/1.NA0031≠).NK00H S/CC4/1 = (").K00H	R/CC3 = TESTA R/CC4 = "

CS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	CLM (19) Compare with limits in MEMORY CLR(39) " " " " REGISTER	FAS1 , FAS22, PREDO FAS1 , FAS22, PRERQ	R>EWv1 Rv1>EW s/ CC1 ↓ ; R<EWv1 Rv1<EW CC2 ↓ ; R>EW " CC3 ↓ ; R<EW " CC4 ↓
ENDP	⇒ S/LR31/2 ⇒ S/RQ (init next instr access f(Q))	S/LR31/2 = FUCLR.PRE2.NIA S/RQ : f(PRERQ)	(for R v 1)
PH1 T4RL	⇒ MBv1 → C ⇒ MB → C NC → D, 1 → CS31 ⇒ RR → A ⇒ RRv1 → A ⇒ S/RQ (init next instr access f(Q)) ⇒ S/DRQ (for access EW) ⇒ Q → P	(prep.) DXNC/1 = FAS1.PH1 AXRR = FAS22.PH1 S/RQ = OU1.(NO5.N06).PH1 S/DRQ = (PREDO.PH1) PXQ = PRERQ.PH1	(request set by prep.)
PH2 T4L	A+D+CS31 → S S → A (A0⊕D0⊕K00) → K00H S/TESTA S/TESTA/1 (CC3,4 contr.) ⇒ MB → C ⇒ NC → D, (1 → CS31 - insig) ⇒ (NEW remains in D S/T4L CLM CLR	SXADD = (FAS22.PH2) AXS = (") SOOXN = (FAS1.PH2) S/TESTA = ( ) S/TESTA/1 = (S/TESTA).FAS1 (see PH1) DXNC/1 = OU1.OL9.PH2 S/T4L = FAS1.PH2	CLM ⇒ R - (EWv1) → A CLR ⇒ (Rv1) - EW → A (R< EWv1 or Rv1 < EW)

CLM (19), CLR (39)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3 T4L	0 → CC3 1 → CC3 if NK00H.(A≠0) 0 → CC4 1 → CC4 if K00H.(A≠0) RR → A 1 → CS31 Q → P if CLM S/DRQ S/PH5	R/CC3 = TESTA S/CC3 = TESTA/1.NA00312.NK00H R/CC4 = TESTA S/CC4 = TESTA/1.NA00312.K00H AXRR = FAS22.PH3 CSX1/8 = FAS1.PH3 PXQ = PREDO.PH3 S/DRQ = (FAS22.PH3) BRPH5 = (")	CLM CLR R > EWv1 Rv1 > EW R < EWv1 Rv1 < EW
PH5 T4L	A+D+CS31 → S S → A (A0⊕D0⊕K00) → K00H S/TESTA S/TESTA/1 0 → CC1 CC3 → CC1 0 → CC2 CC4 → CC2 ENDE	SXADD = (FAS22.PH5) AXS = (") SOOXN = (") S/TESTA = (") S/TESTA/1 = (S/TESTA).FAS1 R/CC1 = (FAS1.PH5) S/CC1 = (").CC3 R/CC2 = (") S/CC2 = (") CC4 ENDE = FAS22.PH5	R > EWv1 Rv1 > EW R < EWv1 Rv1 < EW
Clock following ENDE	1 → CC3 if NK00H.(A≠0) 1 → CC4 if K00H.(")	(same as PH3)	R > EW R < EW

CLM, CLR

2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	EOR (48) EW ⊕ R → R OR (49) EW ∨ R → R AND (4B) EW ∧ R → R	FAS9 = 0U4((04.N05.N06)+(04.N05.07)) PRERQ = ((01.N02)(0L8+0LB)+0U4.0L9).N9L3.NANLZ	
ENDP	s/RQ (access next inst f(Q))	s/RQ : f(PRERQ)	
PH1 T4RL	MB → C C → D RR → A s/NPRx if AND s/PH3 s/T8L s/DRQ Q → P	(preparation) DXC/6 = (FAS9.PH1) AXRR = (") s/NPRx = 0U4.(04.N05.06) BRPH3 = FAS9.04.PH1 s/T8L = (FAS9.PH1) s/DRQ = (") PXQ = PRERQ.PH1	
PH3 T8L	ENDE EOR ⇒ PR → S (A ⊕ D) OR ⇒ A → S, D → S (A ∨ D) AND ⇒ PR → S (A ∧ D) S → RW S → A, s/TESTA } CC3, 4 contr.	ENDE = FAS9.PH3 SXPR = FUEOR.PH3 SXA = (FUOR.PH3) SXD = (") SXPR = NPRX RW = (FAS9.PH3) AXS = (") s/TESTA = (")	

EOR (48) OR (49) AND (4B)

1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	AWM (66) MTW (33) MTH (53) MTB (73) FAS8 FAS13 FAS7 FAS24	= 0U3.0L3 + 0U6.0L6 = (01.03).0L3 = (01.03).0L3 + (02.03).0L3 = (01.03).0L3 + (02.03).0L3 + 0U6.0L6	
PH1 T4RL	⇒ C → D ⇒ C_H → D ⇒ C_B → D ⇒ RR → A ⇒ R2831 → A2831, R28 → CS0027 ⇒ MRQ ⇒ MRQ if R≠0 ⇒ s/DRQ s/T10L } for write	DXC/6 = (0U3.(N04.N05)+FAS8).PH1 DXC/5 = 0U5.(").PH1 DXCBP = 0U7.(").PH1 AXRR = FUAWM.PH1 AXR = (FAS7.PH1) CSX1/2 = (").R28 MRQ = FUAWM.PH1 MRQ = (FAS8.PH1).NRZ s/DRQ = (") s/T10L = FAS24.PH1	(down-aligned (Sign → D0015)) (" - " ($0^{'}s$ → D0023)) R (down-aligned and sign-extended) → A, CS
PH2 T10L	A+D+CS → S S → A ⇒ S → MW ⇒ R/MAPDIS if INTRAPF	SXADD = (FAS24.PH2) AXS = (") MW = FAS8.PH2 R/MAPDIS = FUMTW.PH2.INTRAPF	

AWM (66), MTW (33), MTH (53), MTB (73)

1 of 3

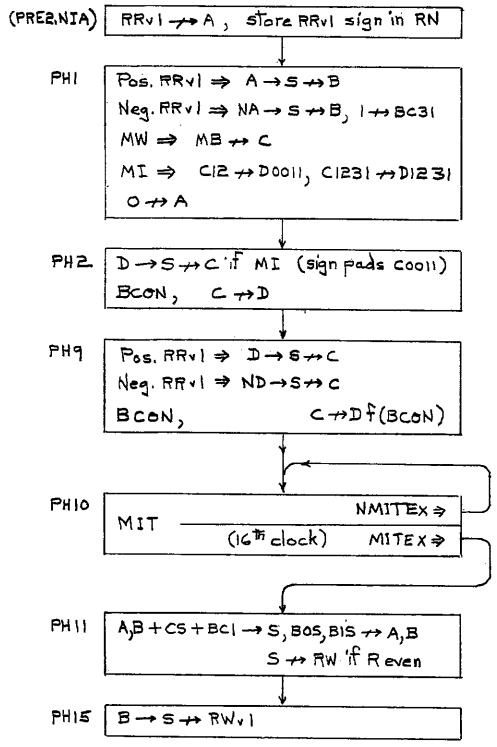
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2	(PH2 cont'd) AWM, MTW, MTH, MTB, NINTRAPF 0 → CC1,2,3,4 ⇒ 1 → CC1 if K00 ⇒ 1 → CC2 if OVER ⇒ 1 → CC2 if S15 ≠ S16 ⇒ K23 → MWN ⇒ MRQ/1 (for next instr.); S/PH4 ⇒ 0 → D; S/NPRX (for up-align A); MRQ if R≠0 (for write) ⇒ S/DRQ for read next instr. ⇒ S/DRQ for WRITE in PH3	R/CC = (PH2.NINTRAPF).FAS24 S/CC1 = (").FAS24.K00 S/CC2 = (").FAS8.OVER S/CC2 = (").FUMTH.(S15 ⊕ S16) S/MWN = FUMTB.PH2.K23; R/MWN = CLEAR MRQ/1 = (PH2.NINTRAPF).FAS8 BRPH4 = FAS8.PH2 Dx/1 = (FAS13.PH2) S/NPRX = (FAS13.PH2) MRQ = (").NRE S/DRQ = FAS24.PH2	(does not cover +R in MTB) ++: AWM,MTW = CS0 A0 D0 K0 = 0 0 0 1 --: AWM = 0 1 1 0 MTW = 1 0 1 0 (store end-carry from BYTE) (write is inhibited when modifier = 0 to permit test of data in protected memory)
PH3 T&L	(PH3 SKIPPED) = ( ) ⇒ A16:31 → S00:15, S16:31; S → A; S → MH ⇒ A24:31 → S (all bytes); 0 → A00:07, S → A08:31; S → MB; 1 → CC1 if K23 in PH2 ⇒ MRQ/1; S/DRQ access next inst ⇒ R/MAPDIS if INTRAPF	SXUAH = (FUMTH.PH3) AXS = (") MWH = (") SXUAB = (FUMTB.PH3) AXS/B = (") MWB = (") S/CC1 = (PH3.NINTRAPF).FUMTB.MWN MRQ/1 = ((").FAS24) S/DRQ = (") R/MAPDIS = FAS13.PH3.INTRAPF	(up-aligned sum → A for proper CC3,4 setting in overflow cases) (clear A0 for TESTA, which, in MTB sets CC3 if result≠0)

AWM, MTW, MTH, MTB

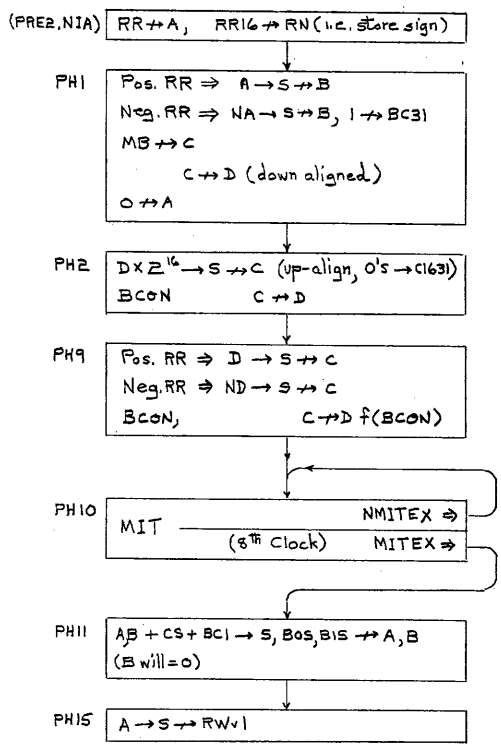
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T&L	CASE 1 : NINTRAPF ⇒ S/TESTA (cc3,4 contr) ENDE TRAP to 67 if overflow.AM	S/TESTA = (FAS24.PH4.NINTRAPF) ENDE = (") TROVER = (").AM.CC2 S/TRAP = (TROVER) S/TR30 = (").N(S/TRACC4/1) S/TR31 = (").N(")	TESTA causes: 1 → CC3 if (Result > 0) 1 → CC3 if (" < 0).MTB 1 → CC4 if (" < 0).NMTB AWM, MTW, MTH only
	CASE 2 : INTRAPF ⇒ INT16 INT17 INT23 if A=0 CEINT B → S; S → P (access instr.f(B)) MRQ S/DRQ R/INTRAPF (advance to PH5) } if NAWM	/INT16/ = (FAS7.PH4.INTRAPF) /INT17/ = (") /INT23/ = (").A0031E CEINT = (") SxB = FAS7.PH4 PxS = (FAS7.PH4.INTRAPF) MRQ = (") S/DRQ = (") R/INTRAPF = (")	(exit highest priority interrupt) (arm " " ") (trigger count zero ") (sync. with interrupt clock)
PH5 T&L	(Entered only if INTRAPF was on) ENDE	ENDE = FAS24.PH5	

AWM, MTW, MTH, MTB

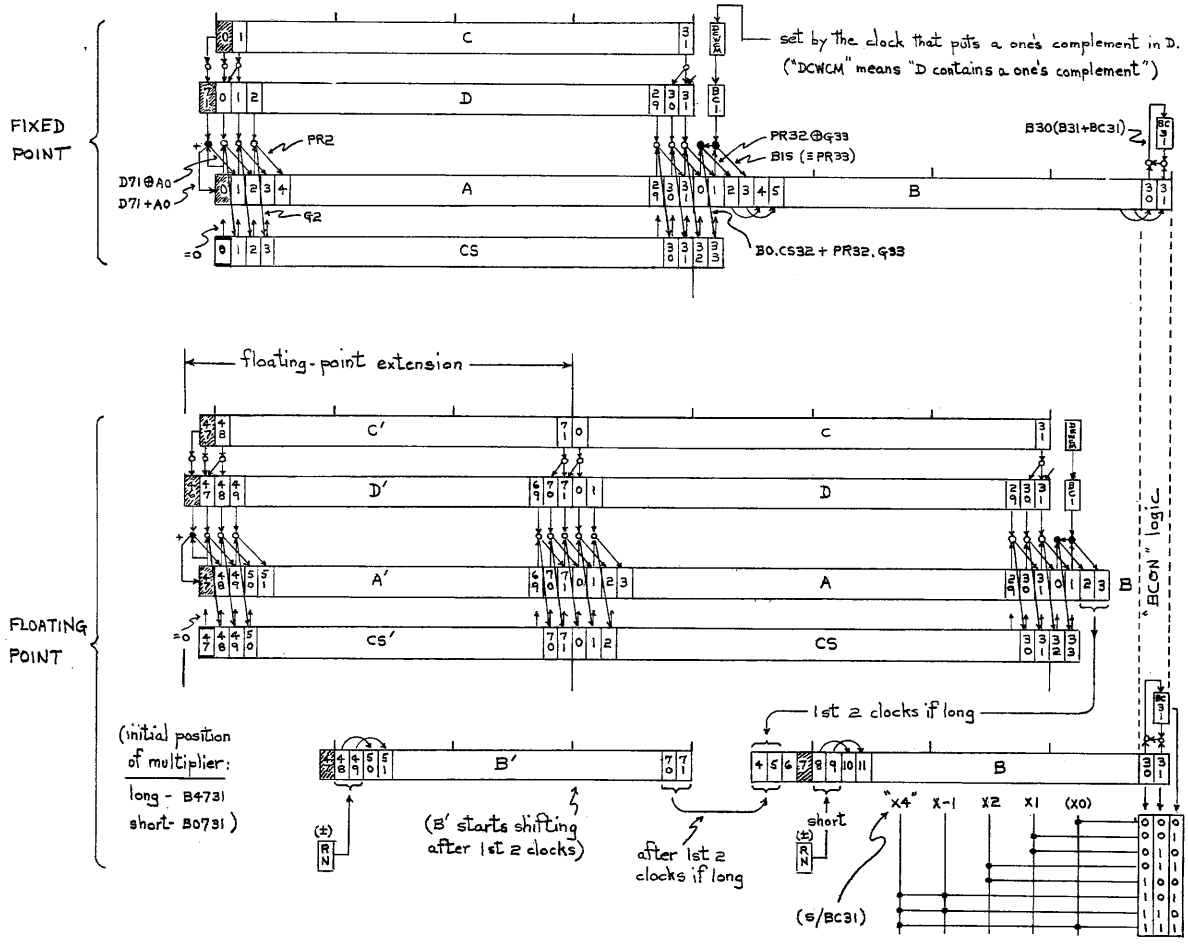
"FAMULNH": MW, MI



"FAMULH": MH



MULTIPLY - REGISTER ORGANIZATION:



MULTIPLY NOTES (fixed and floating):

MPP (multiply preparation) ⇒

- 0 → A"
- C" → D"
- 1 → CS" if (MWN⊕RN) } for generation of |product|
- BCON
- sPH9
- s/CXS , reset 'interruptable ff

} Floating point only (clock preceding PH9)

BCON (multiplier control) ⇒

- 1 → DXCM if $\overline{B30}\cdot B31\cdot\overline{BC31} + \overline{B30}\cdot\overline{B31}\cdot BC31$ (for C → D)
- 1 → DXCLIM if $B30\cdot\overline{B31}\cdot\overline{BC31} + \overline{B30}\cdot B31\cdot BC31$ (for Cx2 → D)
- 1 → DXNCM if $B30\cdot B31\cdot\overline{BC31} + B30\cdot\overline{B31}\cdot BC31$ (for $\overline{C}$ → D)
- 1 → BC31 if $B30\cdot B31\cdot(\overline{BC31}) + B30\cdot BC31$ + FLMC·BC31 (not qualified by BCON (directly))
- 1 → DCWCM if CCWCM (DXCM + DXCLIM) + $\overline{CCWCM}$ DXNCM

B x $\frac{1}{4}$ → B (does not apply to B', which shifts during MIT only)

- PR30 → B0 } during MIT only; 0 → B0,1 during 2 BCON's preceding MIT.
- PR31 → B1 }
- 1 → B2 if (Q33⊕PR32)
- BIS → B3 (BIS = B1 ⊕ BC1 ⊕ CS33)
- 1 → B4 if B2 ($\overline{\text{FLM MIT}}$) + B70 (FLM MIT)
- 1 → B5 " B3 (") + B71 (")
- 1 → B8 if B6 ($\overline{\text{short FLM}}$) + RN (short FLM)
- 1 → B9 " B7 (") + RN (")
- RN → B48,49
- (0 → B47)

C → D f(BCON) ⇒

- $C_{0-31} \rightarrow D_{0-31}$
- $C_0 \rightarrow D_{71}$ if fixed
- $C_{47-71} \rightarrow D_{47-71}$ if floating
- $C_{47} \rightarrow D46$

} if DXCM (Cx1 → D)

- 1 → D31 if CCWCM
- $C_{0-31} \rightarrow D_{71-30}$
- $C_{47-71} \rightarrow D_{46-70}$ if floating

} if DXCLIM (Cx2 → D)

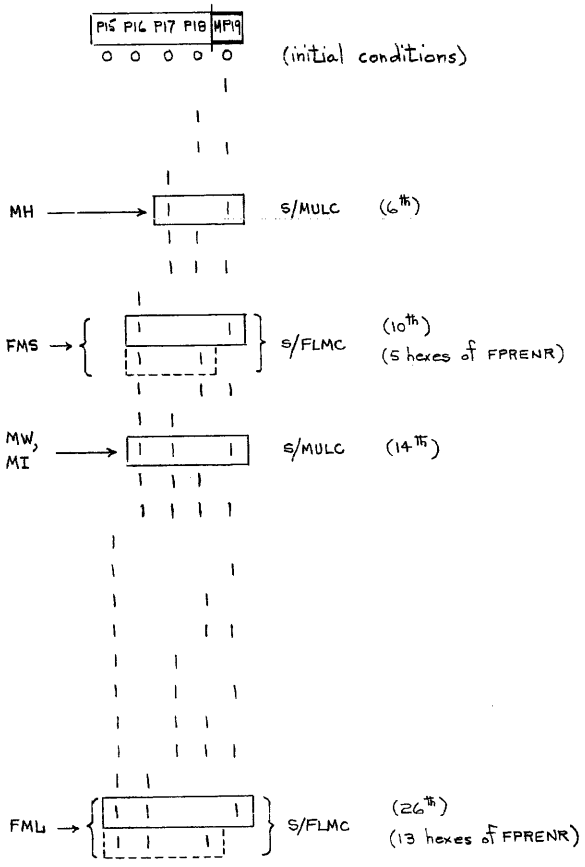
- $\overline{C}_{0-31} \rightarrow D_{0-31}$
- $\overline{C}_0 \rightarrow D71$ if fixed
- $\overline{C}_{47-71} \rightarrow D_{47-71}$ if floating
- $\overline{C}_{47} \rightarrow D46$

} if DXNCM ($\overline{C}$x1 → D)

CCWCM = MUL.RN + FLM (MWN⊕RN)
("C CONTAINS ONES' COMPLEMENT")

(note: when a 1's complement is put in D, the "+1" owed is "paid" to BC1 one clock later)

ITERATIONS COUNTING (detect MITEX minus 2):



MIT (MULTIPLY ITERATIONS) ⇒ (1st clock = 2 x TIL, others = TIL)

$PR \times \frac{1}{4} \longrightarrow AB$

B_{0-3} (see BCON)

$PR_{0-29} \rightarrowtail A_{2-31}$

$1 \rightarrowtail A1$ if $(D71 \oplus A0)$ MUL + PR71 FLM

$1 \rightarrowtail A0$ if $(D71 + A0)$ MUL + PR70 FLM

$PR_{47-69} \rightarrowtail A_{49-71}$

$1 \rightarrowtail A48$ if $(D46 \oplus A47)$ } if FLM

$1 \rightarrowtail A47$ if $(D46 + A47)$ }

$B' \times \frac{1}{4} \longrightarrow B'$

(A → S; insig - mech. conv. - will be in display lights)

FIXED: $B2,3 \rightarrowtail E2,3$ (merge) (for zero check l.s.w. of product)

$G \times \frac{1}{2} \longrightarrow CS$

$1 \rightarrowtail CS33$ if $B0.CS32 + G33.PR32$

$G_{0-31} \rightarrowtail CS_{1-32}$

$G_{47-71} \rightarrowtail CS_{48-0}$ if FLM (keep CS0 off if $\overline{FLM}$), ($0 \rightarrowtail CS47$)

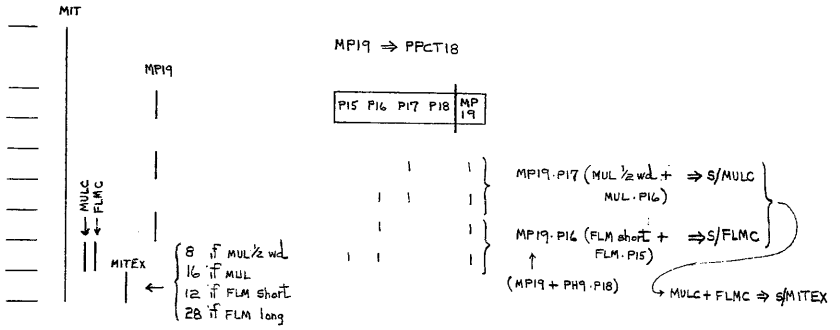
$C \rightarrowtail D$ f(BCON) (see separate sheet)

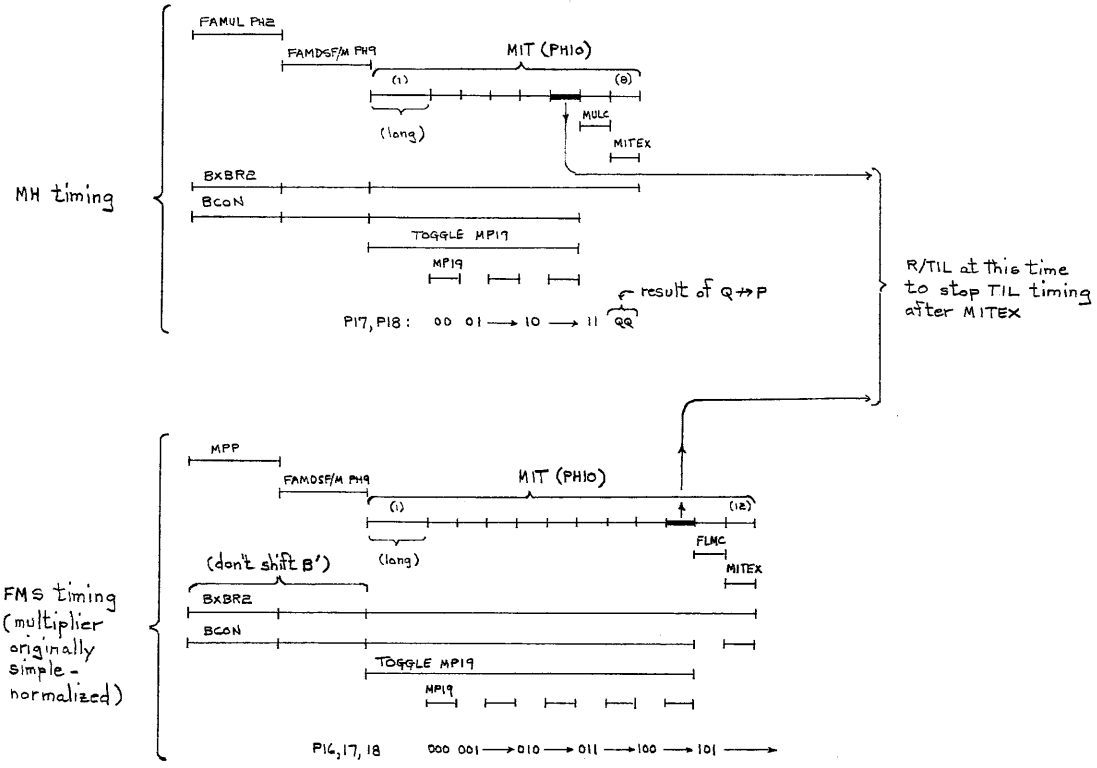
$1 \rightarrowtail BC1$ if MIT. DCWCM (using CSX&R1)

sustain PH10 if MIT $\overline{MITEX}$

BCON: high if $MIT.[\overline{MULC + FLMC + MITEX \cdot MUL}]$ + other slow terms (2 clocks preceding MIT)

Toggle MP19 if $MIT.[\overline{\quad\quad\quad\quad}]$





PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PHASE PRECEDING PRE2.NIA	S/LR31/2 (for RRv1 →→ A in PRE2.NIA)	= (PRE1.NIA + PRE3.Ix + PRE4). FAFRR/1 ← = (OU3, NO4, 05, 06) ← = FAMULW + (PRE1.NIA + PRE3.IA). FAFRR ← = (OU2, OL3) ← = FAMULI	
PRE2.NIA: T4RL	RRv1 →→ A RR0 →→ RN (store sign for PH1,2) 1 →→ CS (for NA →→ S in PH1 if RN)	AXRR = FAMDSF. PRE2.NIA S/RN = RR0. RNxRR0 ← = PRE2.NIA. RNxRR0/1 CSX1 = PRE2.NIA. FAMUL	multiplier →→ A. ← = FAMULNH; R/RN = CLEAR. (negative multiplier case)
PH1 T4RL	A ⟶ S if A is pos. A⊕CS ⟶ S, 1 →→ BC31 if " " neg. S →→ B	SXA = (FAMUL. PH1). NRN SXPR = ("). RN S/BC31 = ("). RN BXS = FAMDSF. PH1	|multiplier| →→ B, BC31. (avoids need for sign iteration)
	MB →→ C if FAMULW C12 →→ D0011, C1231 →→ D1231 if FAMULI (imm.) C0C16 →→ MWN (insig)	(preparation control) DXC/4 = FAMULI. PH1 S/MWN = C0C16. FAMDSF. PH1	multiplicand →→ C. multiplicand →→ D (with sign pad) (immediate) (MB0 ⟶ C0C16)
	S/CXS if FAMULI (imm.) 0 →→ A (clear A for iterations) 0 →→ P (clear iterations ctr.) R/CC2 (for magnitude test in PH15) S/T4L if FAMULW	S/CXS = FAMULI. PH1 AX/1 = FAMDSF. PH1 PX = FAMUL. PH1 R/CC2 = FAMDSF. NFAMULH. PH1 S/T4L = FAMUL. OU3. PH1	(for D ⟶ S →→ C →→ D in PH2)

MI (23), MW (37) "FAMULNH" 1 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 T4L or T6L	(TIMING: T4L if FAMULW; T6L for FAMULI) D ⟶ S, S →→ C if FAMULI (imm.) C →→ D (all cases) 1 →→ CS, S/CXS (for inversion of C. if multiplier neg.) B x 1/4 →→ B (0 →→ B0001) BCON (described separately) general: set-up DXCM, DXNCM, DXCLIM, S/PH9	SXPR = FAMULI. PH2 (PR = D since A = CS = 0) (CXS was set by FAMULI. PH1) DXC/6 = (FAMUL. PH2) CSX1 = ("). RN S/CXS = ("). RN BXBR2 = (") BCON = BXBR2. FAMDSF/M BC31 ff's as function of B30, B31, BC31) BRPH9 = (FAMUL. PH2)	32 bit multiplicand (assembled) →→ C →→ D (already in C if FAMULW) (RN is surplus (mech. conv.)) (for D⊕CS ⟶ S →→ C in PH9) interrogate multiplier $2^{1,0}$
PH9 T6L	D⊕CS ⟶ S (i.e. ND→S), S →→ C if neg. multiplier C →→ D f(BCON) general: C →→ D if DXCM, 1 →→ DCWCM if (CCWCM. (DXCM + DXCLIM) + NCCWCM. DXNCM), etc. complement" (= FAMUL. RN)) B x 1/4 →→ B (0 →→ B0001) BCON (see PH2) 0 →→ D (related to BCON logic) S/MIT (MIT is a fast FAMDSF/M. PH10) S/TIL if not single clocking	SXPR = FAMDSF/M. PH9 (CXS was set in PH2 if RN) (described separately) NC →→ D if DXNCM, 2xC →→ D if DXCLIM, 0 →→ D if not the foregoing, BXBR2 = (FAMDSF/M). PH9 BCON = ("). BXBR2 DX/1 = ("). PH9 S/MIT = ("). PH9 S/TIL = ("). PH9. NKSC. N((S/FLMC/1). P18).	inverted multiplicand →→ C if multiplicand was negated in PH1 (otherwise C remains unchanged) (f(multiplier $2^{1,0}$)) (CCWCM means "C contains one's interrogate multiplier $2^{3,2}$ (is held on until MITEX)

MI, MW 2 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH10 TIL	= MIT (multiply iterations) (16 clocks, the first of which is 2xTIL long)		
	NOTE: Register organization, end bit control, and other such details are described separately (under "MIT FUNCTIONS"). Only general control functions are mentioned below.		
	(PR = A⊕D⊕CS) x 1/4 → A	AXPRR2 = (MIT)	product generation
	(G = A.D + A.CS + D.CS) x 1/2 → CS	CSXGR1 = (")	
	Bx 1/4 → B (PR3031 → B0001)	BXBR2 = (")	
	BCON (see PH2)	BCON reduces to BXBR2.N (MULC + MITEX)	interrogate multiplier 2^{31-4}
	B0203 → E0203 (merge)	(path enabled by FAMUL MIT)	product $2^{27-0} \neq 0$
	(A → S (insig.))	SXA = FAMDSF.PH10	
	MP19: high on all even clocks but the last	S/MP19 red. to MIT. N (MULC + MITEX).NMP19	5-bit iter. ctr. (doesn't count during last two clocks)
	P1518 + MP19 → P1518	PCTP5 = MP19	
	ON THE 14th clock:	(next to last even clock)	(ctr = 13)
	S/MULC	S/MULC = (MP19.(S/MULC/1)), where (S/MULC/1) = FAMULNH.P16.P17	
	R/TIL initiate end of TIL	R/TIL red. to (") + NMIT	(TIL timing lasts thru. PH10)
	S/MRQ/M	S/MRQ/M = (")	
	ON THE 15th clock: (MULC)	(next to last clock - last C→D f(BCON))	f (multiplier $2^{31,30}$)
	S/MITEX	S/MITEX = MULC	(ctr still = 13)
	MRQ, Q → P } go for next instruction	MRQ = MRQ/M PX = PXQ = MULC	1 level MRQ/1
	ON THE 16th clock: (MITEX)	(last clock)	
	(0 → D)	(BCON logic)	
	R/MIT (repeater)	S/MIT = (MIT.NMITEX.NCLEAR)	
	stop sustaining PH10	BRPH10 = (")	(store 2^{63-32} of prod. next phase)
	S/T10L if R is even	S/T10L = FAMULNH.MITEX.NR31	

MI, MW

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH11 T6L or T10L	(TIMING: T6L if R is odd, T10L if R is even)		
	A + CS + K31 → S	SXADD = (FAMDSF/M.PH11)	(Assimilated prod. $2^{63-32} \rightarrow S$
	S → A	AXS = (")	(K31 is end-carry from B assimilation))
	S → RW if R is even	RW = FAMULNH.PH11.NR31	
	(B + CS32, CS33, BC1) → B	BXB = FAMDSF/M.PH11	(assimilate prod. 2^{31-0})
	(") → E0003 (merge)	(path provided by FAMUL.PH11)	(prod. $2^{31-28} \neq 0$)
	1 → CS31 for magnitude test	CSX1/8 = (FAMUL.PH11)	
	S/LR31/2 (for S → RWv1)	S/LR31/2 = (")	
	S/DRQ	S/DRQ = (")	
	S/PH15	BRPH15 = (")	
	S/T8L	S/T8L = (")	
PH15 T8L	B → S	SXB = FAMULNH.PH15	Assimilated prod. $2^{31-0} \rightarrow S$
	S → RWv1	RW = FAMDSF.PH15	(LR31/2 is on)
	for CC3,4 contr. { (set-up to test 64 bit product): 1 → A31 if E ≠ 0	A31X1 = FAMULNH.PH15.NEZ	E≠0 means prod. $2^{31-0} \neq 0$
	S/TESTA	S/TESTA = FAMDSF.NFASHFX.PH15	
	1 → CC2 if $2^{63-32} \neq 2^{31}$	S/CC2 = FAMULNH.PH15.(NB0.NA0031Z + B0.NK00)	(CS31 = 1, D = 0)
	ENDE	ENDE = FAMDSF.PH15	

MI, MW

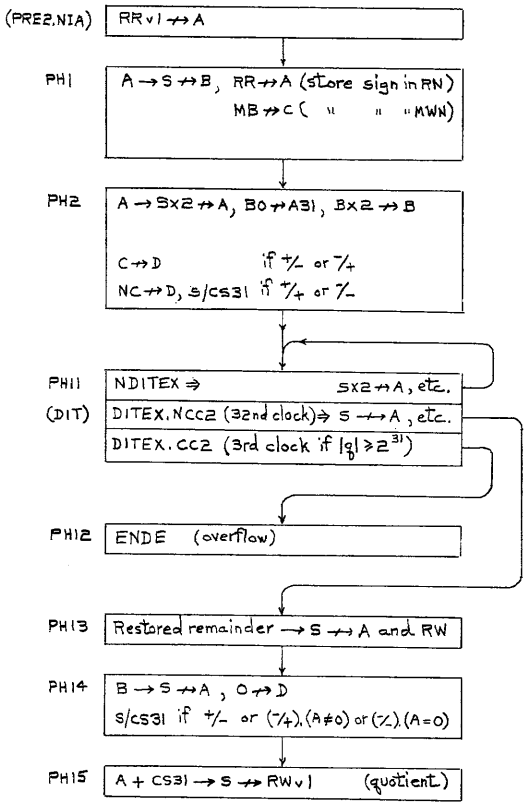
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2.NIA: T4RL	RR → A (multiplier → A) RR16 → RN (store multiplier sign) 1 → CS (for NA → S in PH1 if RN)	AXRR = FAMDSF.PRE2.NIA S/RN = RR16.FAMULH.PRE2.NIA; R/RN = CLEAR CSX1 = PRE2.NIA.FAMUL	multiplier consists of RR1631. (0015 are ignored) (neg. multiplier case)
PH1 T4RL	A → S if pos. multiplier A⊕CS → S, 1 → BC31 " neg. " S → B	SXA = (FAMUL PH1).NRN SXPR = S/BC31 = (").RN BXS = FAMDSF.PH1	(multiplier) → B1631, BC31. (B0015 are insig)
	MB → C C0C16 → D0015 (sign pad) C0015 → D1631 if NP32 C1631 → D1631 if P32 (C0C16 → MWN (insig.))	(preparation contr.) (C0C16 = RR0 if NP32, RR16 if P32) DXC/5 = 005.(N04.05.06).PH1 S/MWN = FAMDSF.PH1	downward aligned half-word multiplicand → D (will be up-aligned in PH2)
	0 → A (clear for up-align D and for iterations) S/CXS (for D x 2^{16} → C in PH2) 1 → NPRX S/T8L 0 → P (clear iterations ctr.)	AX/1 = FAMDSF.PH1 S/CXS = (FAMULH.PH1) S/NPRX = (") S/T8L = (") PX = FAMUL.PH1	set-up for upward alignment of multiplicand
PH2 T8L	$D_i . CS_i \rightarrow K_{i+1}$ K x 2^{15} → S0015 0 → S1631 S → C C → D	(CS = all 1's, A = 0, D is down aligned) SXUAH/1 = FAMULH.PH2	upward aligned half-word multiplicand → C → D (0 → D1631)
	1 → CS S/CXS } if multiplier was negative	(CXS was set in PH1) DXC/6 = (FAMUL.PH2) CSX1 = (").RN S/CXS = (").RN	set up for ND → S → C in PH9

MH (57) "FAMULH" 1 of 2

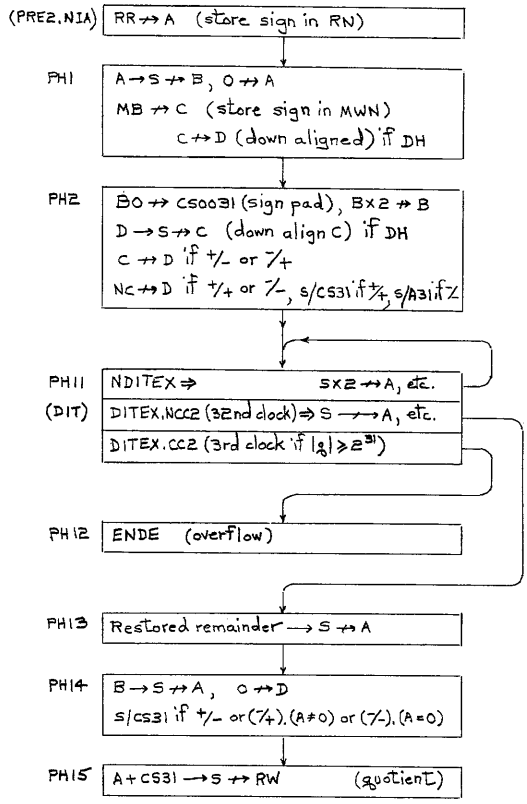
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2	(continued) B x 1/4 → B (0 → B0001) BCON S/PH9	same as FAMULNH	interrogate multiplier $2^{1,0}$
PH9 T6L	All operations same as PH9 of FAMULNH. Summary: D⊕CS → S → C if RN, C → D f(BCON) ← f($2^{1,0}$ of multiplier), Bx1/4 → B, BCON (for $2^{3,2}$ of multiplier), 0 → D (related to BCON logic), S/MIT, S/T1L.		(C unchanged if NRN),
PH10 T1L	= MIT (multiply iterations) (8 clocks, the first of which is 2x T1L long) Operations same as PH10 of FAMULNH except that: 1) There are 8 clocks instead of 16 2) The last iteration is for multiplier $2^{15,16}$ instead of $2^{31,30}$ 3) Terminal iterations are initiated on the 6th clock instead of the 14th (i.e. (S/MULC/1) = FAMULH.P17) 4) The product (2^{31-0}) will have no significance reaching B 5) T10L will not be set		
PH11 T6L	A+CS+K31 → S → A } carry assimilation B+CS32, CS33, BC1 → B } also: 1 → CS31 (insig.), S/LR31/2, S/DRQ, S/PH15, S/T8L, (as in FAMULNH)	AXS = SXADD = (FAMDSF/M.PH11) BXB = (")	product 2^{31-0} → A 0 → B0019; (garbage in B2031)
PH15 T8L	A → S S → RW.1 S/TESTA (CC3,4 contr.) ENDE	SXA = FAMULH.PH15 RW = (FAMDSF.PH15) S/TESTA = ( ).NFASHFx ENDE = ( )	

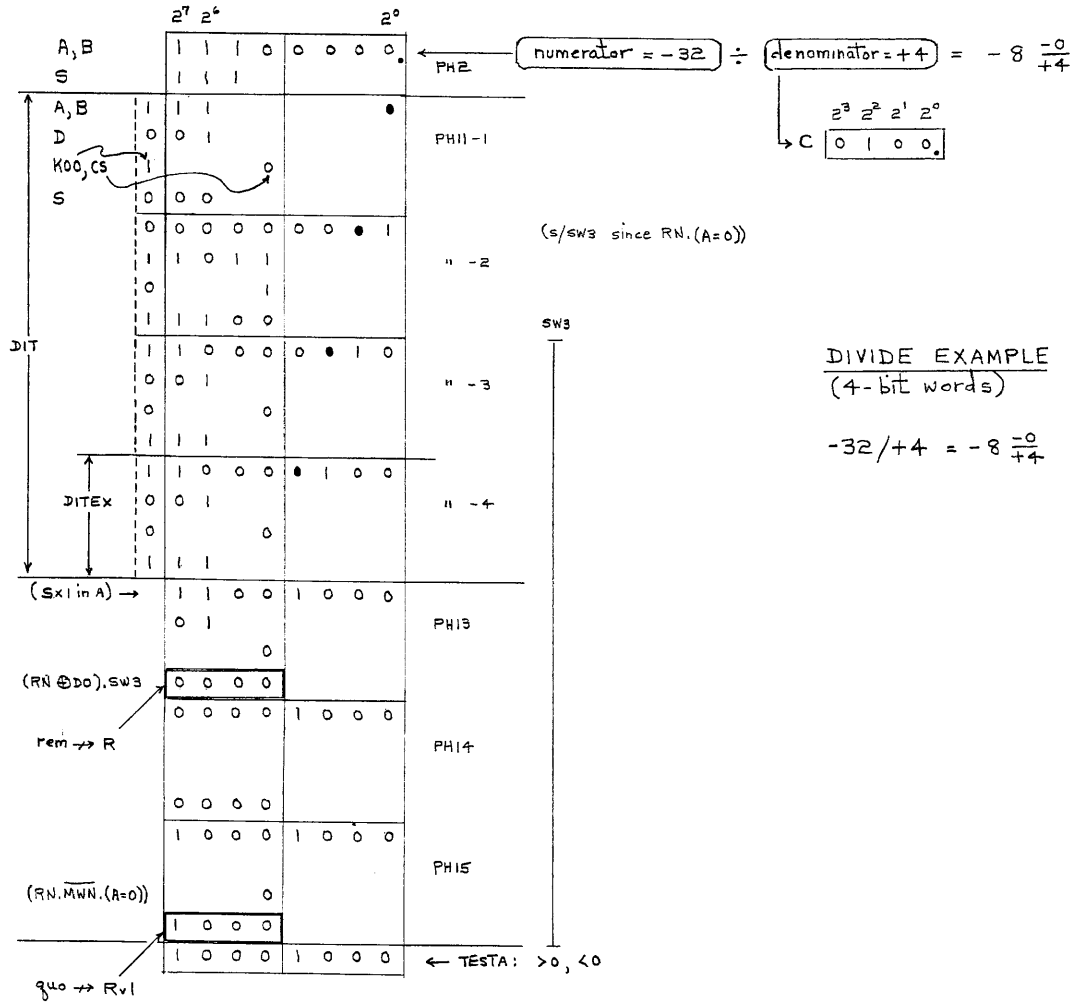
MH 2 of 2

"FADIVW": DW. Reven



"FADIVH": DH or DW.Rodd





PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PHASE PRECEDING (PRE2.NIA): S/LR31/2		S/LR31/2 = (PRE1.NIA + PRE3.IX + PRE4).FAFRR/1, where FAFRR/1 = OU3.N04.05.06	for RRv1 → A in PRE2.NIA
PRE2.NIA: RRv1 → A		AXRR = FAMDSF.PRE2.NIA	l.s.w. of numer, $(2^{31}-2^{0})$
PH1 T4RL	A → S → B 1 → BWZ if A0131 = 0 (for O.F. test) R → A RR0 → RN (reset by CLEAR) MB → C MB0 → COC16 → MWN (") R/CC2 (for O.F. test) 0 → P (clear for use as iterations ctr.) S/IEN (start interruptability)	BXS = FAMDSF.PH1, SXA = FADIV.PH1 S/BWZ = FADIV.PH1.A0131Z AXRR = PH1.RNXRR0/2 ← = FADIVW RNXRR0 = RR0.RNXRR0 ← PH1.RNXRR0/2 (by preparation control) S/MWN = FAMDSF.PH1.COC16 R/CC2 = FAMDSF.NFAMULH.PH1 PX = FADIV.PH1 S/IEN = FAMDSF.NFAMUL.PH1	l.s.w. of numer → B, "B WAS ZERO" (almost). m.s.w. of numer → A $(2^{63}-2^{32})$, store sign of numer. denom. $(2^{31}-2^{0})$, store sign of denom.
PH2 T6L	A → SX2 → A B0 → A31 BX2 → B 0 → B31 (insig) NC → D, 1 → CS31 if N(MWN ⊕ RN) C → D if (") P+1 → P (pre-count iterations ctr.) S/PH11 (start iterations)	AXSLI = SXPR = FADIVW.PH2 (D = CS = zero) S/A31 = AXSLI.A31EN/2 ← B0.FAMDSF BXBLI = FADIV.PH2 (no set term active) DXNC/1 = (FADIV.PH2).N(MWN ⊕ RN) DXC/6 = (").(") PCTP1 = (") BRPH11 = (")	scale numerator to align numer. 2^{31} over denom. 2^{0} so first iteration yields quo. 2^{31}. denom → D, with polarity opposite to numer. (mech. convenience in PH11)
	NOTES: FADIV = OU3.0L6+OU5.0L6 FADIVW = FADIV.N01.NR31 FADIVH =	(DW or DH) (DW if R is even) : perform DW (DW if R is odd or DH) : " DH	

DW (36) (R must be even; if R is odd, see DH) "FADIVW" 1 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH11 T6L	"DIT" Divide iterations (32 clocks if no overflow, 3 clocks if overflow)		
	CONTROL SIGNALS:		
	DIT (divide iterations)	DIT = FAMDSF/D.PH11	
	DITEX (last clock of DIT)	DITEX = FADIV.P26 + CC2	32nd clock or "O.F. detected"
	CONTROL FUNCTIONS:		
	P+1 → P (count iter.) } if NDITEX	PCTP1 = (DIT.NDITEX)	(P = 1 initially)
	sustain PH11 } if NDITEX	BRPH11 = (")	
	MRQ/1 (Q → P, etc.) } if DITEX	MRQ/1 = (DIT.DITEX)	includes O.F.
	R/IEN (stop interruptability) } if DITEX	R/IEN = (")	
	S/PH13 } when P = 32 (i.e. no O.F.) } if DITEX	BRPH13 = (FADIV.PH11.P26)	excludes O.F.
	S/T10L } when P = 32 (i.e. no O.F.) } if DITEX	S/T10L = (").FADIVW	
	(advance to PH12 if O.F.) } if DITEX		
	REGISTER CONTROL:		
	A+D+CS31 → S	SXADD = (DIT)	
	SX2 → A } if P ≠ 32	AXSLI = (").N(FADIV.P26)	residue x 2 → A
	B0 → A31 } if P ≠ 32	S/A31 = AXSLI.A31EN/2 ← = FAMDSF/1.B0	
	S → A if P = 32 } residue	AXS = FADIV.PH11.P26	residue to A for remainder rest.
	BX2 → B } quotient	BXBLI = DIT	quotient is the 1's complement of |quo| when ± or ∓.
	1 → B31 } quotient	S/B31 = DXNC.FADIV.PH11	
	NC → D } if (MWN ⊕ K00)	DXNC/D reduces to (FADIV.PH11).(MWN ⊕ K00)	polarity of denom clocked into D, CS is opposite to that of residue clocked into A.
	1 → CS31 } if (MWN ⊕ K00)		
	C → D if N(MWN ⊕ K00) } ± denom.	DXC/D " " (").N(")	
	NMWN.NK00 + MWN.K00 (C+).(S−) + (C−).(S+)		
	SW3: if on after final iteration, means zero	residue was hit (for neg. numer. case): S/SW3 = (FADIV.PH11).DO.A0031Z; R/SW3 = (").A31 + CLEAR	

DW 2 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH11 T&L	(Continued) OVERFLOW LOGIC: $1 \rightarrow$ CC2 if $\lvert n\rvert \div \lvert d\rvert \geq 2^{31}$ (sign bit posit. of quo) $\lvert q\rvert > 2^{31} \rightarrow$ $\lvert q\rvert \geqslant 2^{31} \rightarrow$ $\lvert q\rvert = 2^{31} \rightarrow$ CC2 ⇒ DITEX (perform DITEX functions and inhibit NDITEX functions) (3rd clock) (DITEX = CC2) also: s/DRQ s/TRAP s/TR30 s/TR31 } trap to 67_{10} if AM=1 (advance to PH12)	s/CC2 = FADIV.PH11.P2&29Z.P30.NP31.DIVOVER (i.e. probe DIVOVER after the 1st iteration (2nd clock, P=2) DIVOVER = RN.NDO (- numer).(- residue) + NRN.DO (+ numer).(+ residue) + DO.A0031Z.BWZ (residue = 0) (note: BWZ means B0029 = 0 at this time; DO means + residue) s/DRQ = (FADIV.PH11.CC2) (ENDE follows) s/TRAP = ( ).AM s/TR30 = ( ).AM s/TR31 = ( ).AM	
PH12 T&L	(entered only if overflow detected in PH11) ENDE	ENDE = FADIV.PH12	
PH13 T10L	(Remainder restoration phase) N(RN⊕DO).NA0031Z ⇒ A+D+CS31 → S N(").A0031Z ⇒ (0 → S) (").NSW3 ⇒ A → S (").SW3 ⇒ (0 → S) S ↛ A (For quotient adjustment in PH14) S ↛ RW	NOTE: (RN⊕DO) means numerator and residue have like signs SXADD = FADIV.PH13.N(RN⊕DO).NA0031Z (unlike signs).(residue ≠ 0) (+ or − numerator cases) (residue = 0) SXA = FADIV.PH13.(RN⊕DO).NSW3 (like signs).(|residue| ≠ |denom|) (negative numerator case where residue hit 0) (").(" |=| ") AXS = FADIV.PH13 (zero remainder requires increasing |quo| in neg. numer. case) RW = FADIVW.PH13 remainder ↛ R	

DW 3 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH14 T&L	(adjust quotient set-up phase) (note: if the quotient is negative, it is a 1's complement) B → S ↛ A 0 ↛ D 1 ↛ CS31 if: (+/−) or (−/+).(rem ≠ 0) (in (−/+) case, if rem = 0, a −1 is owed to the quotient; this is "paid by withholding or (−/−).(rem = 0) s/DRQ s/LR31/2 s/T10L	AXS = SXB = (FADIV.PH14) DX/1 = (") s/CS31 = (CSX1/8) = (FADIV.PH14).NRN.MWN s/CS31 = (") = (").RN.NMWN.NA0031Z (") s/CS31 = CSX1/8 = FADIV.PH14.RN.MWN.A0031Z (") s/DRQ = FAMDSF.PH14 s/LR31/2 = FADIVW.PH14 s/T10L = FAMDSF.PH14	bring un-adjusted quo ↛ A. (clear for A+CS31 → S) (+1 for 2's complement) (+1 owed to + quo. because rem=0) (ENDE FOLLOWS) for quotient ↛ RW.1 (S ↛ RW.1 follows)
PH15 T10L	ENDE A+CS31 → S (adjusted quotient) S ↛ RW.1 (") for CC3,4 control { S ↛ A (") s/TESTA	ENDE = FAMDSF.PH15 SXADD = FADIV.PH15 RW = FAMDSF.PH15 AXS = FADIV.PH15 s/TESTA = FAMDSF.NFASHFX.PH15	(LR31/2 was set in PH14)

DW 4 of 4

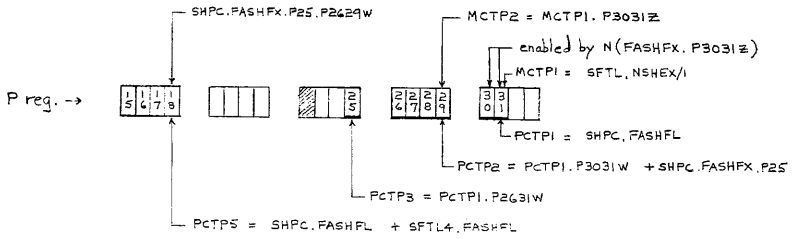
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2.NIA	R → A ((2^{31-0}) → A (numer.) RRO → RN (store numer. sign)	AXRR = FAMDSF.PRE2.NIA S/RN = RRO.RNXRRO ← PRE2.NIA.RNXRRO	1 ← FADIVH; (R/RN = CLEAR)
PH1 T4RL	A → S S → B 1 → BW2 if A0131 = 0 } numer. MB → C C → D (down aligned) MB0 → COC16 → MWN if upper ½WD MB16 → COC16 → MWN " lower " " } denom. 0 → P (clear iterations ctr) 0 → CC2 (for o.f. test) S/IEN (start interruptability) S/CXS if DH (but not in DW.R31 case) 0 → A (for sign pad, etc.)	SXA = FADIV.PH1 BXS = FAMDSF.PH1 S/BW2 = FADIV.PH1.A0131Z; R/BW2 = CLEAR (prep.) DXC/5 = OUS.(NO4.05.06).PH1 S/MWN = FAMDSF.PH1.COC16; R/MWN = CLEAR PX = FADIV.PH1 R/CC2 = (FAMDSF.PH1).NFAMULH S/IEN = (").NFAMUL S/CXS = FADIVH.PH1.OUS AX/1 = FAMDSF.PH1	(2^{31-0}) → B (numer.) (for overflow test) (down-aligned denom. to D; (sign stored in MWN)) (D remains = 0 if DW.R31) (for D → S → C in PH2)
PH2 T6L	D → S } if DH (but not in S → C } DW.R31 case) CASE 1 +/+ : NC → D 1 → CS31 } (- denom → D,CS) CASE 2 +/- : C → D (+ denom → D) CASE 3 -/+ : C → D (+ denom → D) 1's → CS0031 (sign pad numer.) CASE 4 -/- : NC → D 1 → A31 } (- denom → D,A) 1's → CS0031 (sign pad numer.)	SXD = FADIVH.PH2 (CXS is on - see PH1) DXNC/1 = (FADIV.PH2).N(MWN ⊕ RN) DXC/6 = (").(") DXC/6 = (").(") CSX1 = FADIVH.PH2.RN DXNC/1 = FADIV.PH2.N(MWN ⊕ RN) A31X1 = A31X1/1 = FADIVH.PH2.MWN.RN CSX1 = FADIVH.PH2.RN (+ S/CS31 = DXNC/1 - redun.)	(down-aligned, sign-padded denom. → C if DH, (C already contains 32 bit denom. in DW.R31 case)). (put denominator into D, CS31/A31 in polarity opposite to numer; also, put sign pad of neg. numer. in CS)

DH (56) (includes DW (36), where R is odd) "FADIVH". 1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2	(Continued) B x 2 → B 0 → B31 (insig) (sign pad is handled by CS (see above)) P+1 → P (pre-count iterations ctr.) S/PH11	BXBL2 = FADIV.PH2 (no set term active) PCTP1 = (FADIV.PH2) BRPH11 = (")	(scale numerator so 2^{31} is "over" denom. 2^{0} for first iteration; hence first iteration yields quotient 2^{31})
PH11 T6L	"DIT" (32 clocks) (same as FADIVW except that T10L	is not set by DITEX)	(divide iterations)
(PH12) T6L	(entered only if overflow detected in ENDE	PH11 ($^{n}/_{0}$ or $-2^{31}/-2^{0}$)) ENDE = FADIV.PH12	(overflow case only)
PH13 T6L	(Same as FADIVW except that: 1) Timing is T6L instead of T10L 2) S → RW is inhibited (i.e. the	remainder is discarded)	(remainder restoration)
PH14 T6L	(same as FADIVW except that S/LR31	/2 does not take place)	(bring un-adjusted quo. to A and adjust CS31)
PH15 T10L	(same as FADIVW except that: quotient → RW (instead of RW.1) (ENDE)	ENDE = FAMDSF.PH15	(store quotient in R)

DH 2 of 2

SHIFT INSTRUCTIONS (notes on control signals):



SHPC contr. { FASHFL: [nu] toggle by SFTR2 ; FASHFX: [30] set by PH1.P30, toggle by SFTR2

sBWZ = PH1.P25 (direction of shift: 0 ⇒ left, 1 ⇒ right)

SHEX = PH9.NNSHEX/1 (exit from shifting phase)

NSHEX/1 = N (P2531Z (left shift cases)
+ P1517W (right shift cases, floating point limit - 14 hexes)
+ NP25. NA0811Z. FASHFL (normalized number in floating left shift)
+ E0) (exponent overflow/underflow)

SFTL = PH9. NBWZ

SFTR = PH9. BWZ

SFTL1 = FASHFX [SFTL. (P30 + P31) + (hi at SHEX time) SFTR. P18. P31]

SFTL4 = SFTL. NSFTL1. NSHEX/1

SFTR2 = SFTR. NSHEX/1

PCTE1 = SHPC. FASHFL ; MCTE1 = SFTL4. FASHFL

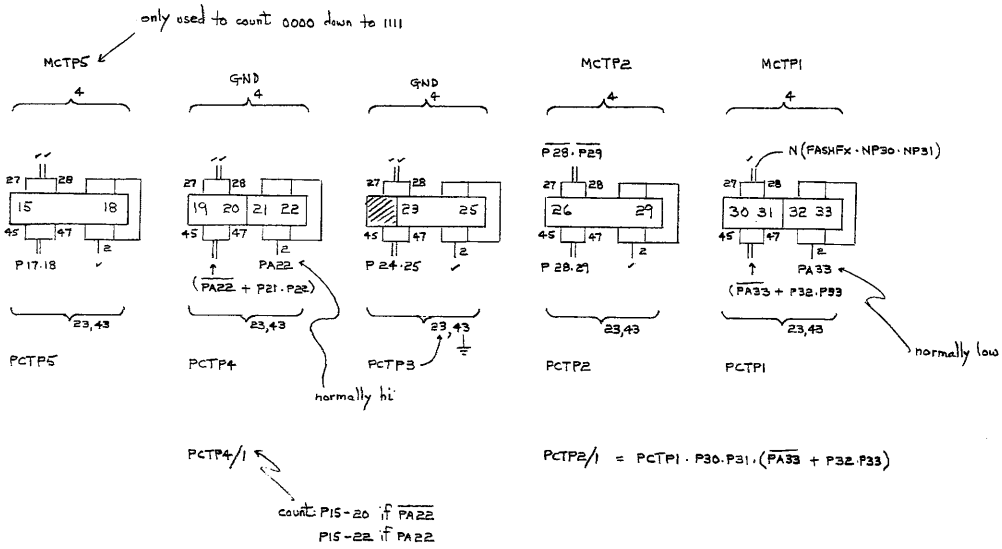
PARITY: toggle CC1 by SFTPAR, which = FASHFX. SFTL4. (A0⊕A1⊕A2⊕A3) + SFTL1. NBWZ. A0 (implies FASHFX; exclude right shift)

FIXED POINT OVERFLOW: set CC2 by FASHFX. SFTL4. N(A0004Z + A0004W) + SFTL1 · NBWZ. (A0⊕A1)

FLOATING POINT "NORMALIZED": set CC1 by FASHFL. SHEX. NA0811Z + FASHFL. PH15. NBWZ. RTZ

" " EXPONENT OVERFLOW/UNDERFLOW: set CC2 by FASHFL. PH15. E0. NRTZ

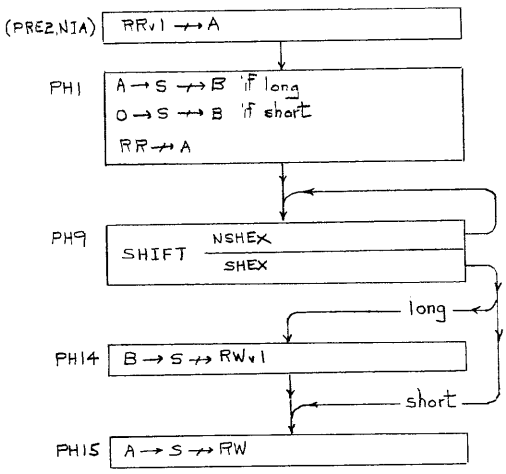
P REGISTER CONTROL

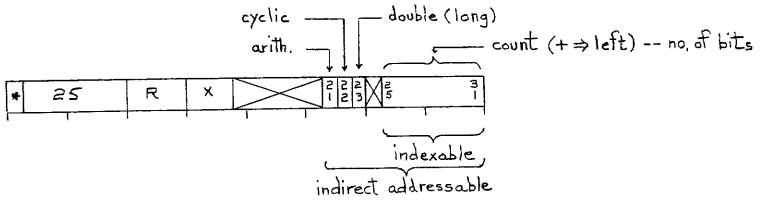


PCTP4/1 ↔ count: P15-20 if PA22 ; P15-22 if PA22

PCTP2/1 = PCTP1 · P30. P31. (PA33 + P32. P33)

"FASHFX" : S





FASH . PH9 (fixed)

FIXED POINT - LEFT

P25 = BWZ = 0 (+)

2^2 2^0 (binary positions)

15 18 25 26 29 30,31

1 { 0 0 0 0 | 00 00 | AB C 0 | X X X X | Y Z — $\overline{P3031Z}$ (0-3 clocks)

down-count P3031, shift AB left by 1's, toggle cc1 if A0, set cc2 if (A0⊕A1)

2 { 00 00 00 00 AB C0 XX XX 00 — P3031Z . $\overline{P2629Z}$ (0-15 clocks)

down-count P2629, shift AB left by 4's, toggle cc1 if (A0⊕A1⊕A2⊕A3), set cc2 if N(A0004Z + A0004W)

3 { 00 00 00 00 AB C0 0000 00 — P3031Z . P2629Z (1 clock)

SHEX (raised by P2531Z)

FIXED POINT - RIGHT

P25 = BWZ = 1 (-)

SHPC (P30 ↛ SHPC during PH1, represents 2^1)

15 18 25 26 29 30,31

1 { 0 0 0 0 | 00 00 | AB C 1 | X X X X Y | Y Z — P1517W (1-32 clocks)

[Toggle SHPC --- SHPC up-counts P2629 ---- carry from P26 down-counts P1518]
shift AB right by 2's (B30 ↛ BC31 to extend B in case of over-shift (i.e. P31=1))

2 { 11 11 00 00 AB C1 0000 0 YZ — P1517W

SHEX (raised by P1517W) shift AB left 1 if P18 . P31 (1 clock)

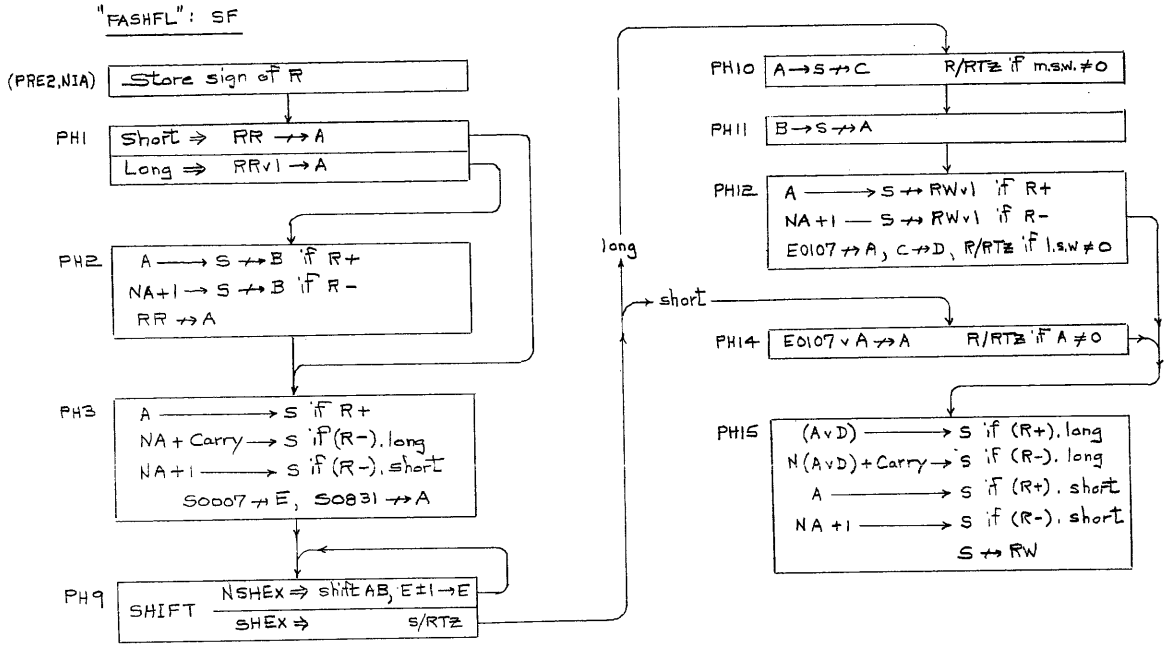
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
(Phase preceding PRE2.NIA):	S/LR31/2 =	(PRE1.NIA + PRE3.IA). GU2. GLS	for RRv1 during PRE2.NIA
PRE2.NIA T4 RL	RRv1 ↛ A	AXRR = FAMDSF.PRE2.NIA	
PH1 T4 RL	A → S ↛ B if long, 0 → S ↛ B if short RR↛A RR0 ↛ RN (store sign) (insig) P25 ↛ BWz (store direction) R/cc1 (for parity check in PH9) R/cc2 (for overflow check in PH9) S/IEN (render instruction interruptable) 0 ↛ P15-22 (for right shift case) P30 ↛ SHPC (") S/PH9	SXA = FASHFx.C23.PH1, BXS = FAMDSF.PH1 AXRR = RNXRR0/2.PH1 S/RN = RR0.RNXRR0, where RNXRR0 includes RNXRR0/2.PH1 (R/RN = CLEAR) S/BWz = FASH.PH1.P25, R/BWz = CLEAR R/cc1 = FASH.PH1. R/cc2 = FAMDSF.NFAMULH.PH1 S/IEN = FAMDSF.NFAMUL.PH1 Px/2 = FASH.PH1 S/SHPC = FASHFx.PH1.P30 BRPH9 = FASHFX.PH1	} RNXRR0/2 includes FASHFX SHEx ⇒ Q25 ↛ P25 } can be set only in left shift case will be reset by SHEx
PH9 T6L	SHIFTING PHASE. (Details are listed on separate sheets; only general control is mentioned here) NP25 ⇒ shift left by 1's and by 4's until SHEX (= P2531z) causes exit P25 ⇒ shift right by 2's (and left by 1 if add) until SHEx (= P1517W) causes exit A → S A → PR (continued)	 SXA = FASH.PH9 (D and CS are cleared, hence PR = A)	 (1 - 19 clocks) (2 - 33 clocks) for AXSL4, AXSL1 for AXPRR2

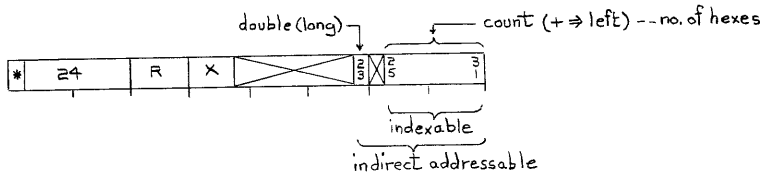
S (25) "FASHFX" 1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9 T6L	(Cont'd) Sustain PH9	BRPH9 = FASH.PH9.NSHEX/1, where, NSHEX/1 = N(NP25.P2629z.NP30.NP31	(shifting in process) + P15.P16.P17)
	Functions performed on last clock (i.e.	SHEX, which = FASH.PH9.NNSHEX/1)	(" shift exit")
	R/IEN (terminate interruptability) MRQ/1 (Q ↛ P, etc) S/T10L (store in next phase) short only { S/PH15, S/DRQ } long only { S/PH14, S/LR31/2 }	R/IEN = SHEX MRQ/1 = SHEX.N(FASHFL.C23) S/T10L = SHEX.FASHFx BRPH15 = SHEX.FASHFx.NC23 S/DRQ = SHEX.FASHFx.NC23 BRPH14 = SHEX.FASHFx.C23 S/LR31/2 = SHEX.FASHFx.C23	(P and C are free) short: (A → S ↛ RW next phase) long: (B → S ↛ RWv1 next phase)
PH14 T10L	(entered if long only) B → S ↛ RWv1 S/T10L S/DRQ	SXB = FASHFx.PH14; RW = FASHFX.PH14 S/T10L = FAMDSF.PH14 S/DRQ = FAMDSF.PH14	
PH15 T10L	A → S ↛ RW ENDE	{ SXADD = FASH.PH15.NRTz (note: ADD reduces to A; NRTz is high) RW = FAMDSF.PH15 ENDE = FAMDSF.PH15	

S 2 of 2

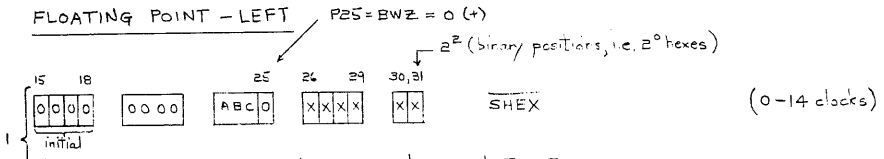
"FASHFL": SF





FASH. PH9 (floating)

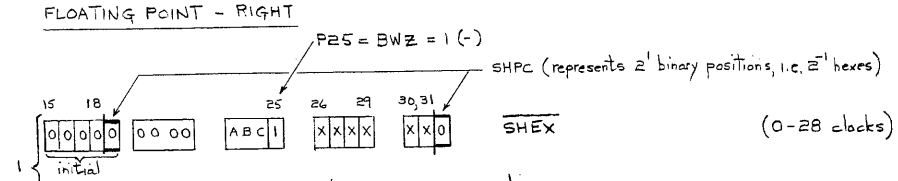
FLOATING POINT – LEFT



1 { down-count P2631, up-count P1518, down-count E0007

2 { SHEX can be raised by: (1 clock)
1) P2531Z, i.e. shift count has been complied with, or
2) P1517W, i.e. 14 hexes of shift have taken place (hence result must = 0), or
3) AC911Z, i.e. the number is normalized, or
4) E0, i.e. the exponent has underflowed. (E will = 11 11 11 11)

FLOATING POINT – RIGHT



1 { toggle SHPC --- SHPC up-counts P1531, P1518, and E0007
AB shifts right by 2's (i.e. one hex for each cycle of SHPC)

2 { SHEX can be raised by: (1 clock)
1) P2531Z, i.e. shift count has been complied with, or
2) P1517Z, i.e. 14 hexes of shift have taken place (hence result must = 0), or
3) E0, i.e. the exponent has overflowed. (E will = 10 00 00 00)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2.NIA T4 RL	RRO → RN (store sign)	S/RN = RRO·RNXRRO, where RNXRRO = RNXRRO/1 (R/RN = CLEAR)	1, PRE2.NIA, and where RNXRRO/1 = FASHFL
	RR → A (insig) S/LR31/2 if long 0's → D	AXRR = FAMDSF.PRE2.NIA S/LR31/2 = FASHFL.PRE2.NIA.C23 DX/1 = PRE2.NIA	(for RRV1 → A in PH1 if long) (for −A→S)
PH1 T4 RL	RRV1 → A if long RR → A if short	AXRR = FASHFL.PH1	(LR31/2 is on) (" " off)
	P25 → BWZ (store direction)	S/BWZ = FASH.PH1.P25, R/BWZ = CLEAR	
	R/CC1 (for normalize check in PH9 and PH15)	R/CC1 = FASH.PH1	
	R/CC2 (for exponent overflow/underflow test in PH15)	R/CC2 = FAMDSF.NFAMULH.PH1	
	S/IEN (render instruction interruptable)	S/IEN = FAMDSF.NFAMUL.PH1	
	0 → P15-22 (clear "limit counter")	PX/2 = FASH.PH1	
	S/NGX if negative operand	S/NGX = FASHFL.PH1.RN	
	R/KOOH (to cause $\overline{A}+1 \rightarrow S$ next phase in FASHFL when negation takes place (NGX), $\overline{A} \rightarrow S$ if KOOH is on, and $\overline{A}+1 \rightarrow S$ if KOOH is off.)	in negative operand case; [K31 (on the adder) = NGX.N(KOOH.FASHFL), hence S/KOOH = SOO = NKOO.N(FASHFL.PH1) ; (R/KOOH is always high)	
	S/PH3 if short enable T6RL if long	BRPH3 = FASHFL.PH1.NC23; T6RL = FASHFL.PH1.C23	
PH2 T6 RL	RR → A (entered if long only) S → B	AXRR = FASHFL.PH2 BXS = FASHFL.PH2	
	+ operand : A → S − operand : $\overline{A}+1 \rightarrow S$	SXADD = FASHFL.PH2 ; ADD = A since NGX is off. " = " ; ADD = $\overline{A}+1$ since NGX and NKOOH are on.	
	NKOO → KOOH (store inverted end-carry for double precision negation.)	S/KOOH = SOO = NKOO	
	S/NGX (to continue negation)	S/NGX = FASHFL.PH2.RN	

SF (24) "FASHFL" 1 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3 T6L	SO831 → A0831, (0 → A0007), (mantissa) SO007 → E (exponent)	AXS/3 = FASHFL.PH3 EXS = FASHFL.PH3	("−1" is converted to zero) (SO will = 0 except in above)
	+ operand : A → S − operand : $\overline{A} \rightarrow S$ if input carry = 0 $\overline{A}+1 \rightarrow S$ if input carry = 1	SXADD = FASHFL.PH3; ADD = A since NGX is off " = " ; ADD = $\overline{A}$ " " is on " = " ; ADD = $\overline{A}+1$ " " " on	and KOOH is on " " " off ← (KOO in PH2)
	S PH9	BRPH9 = FASHFL.PH3	
PH9 T6L	SHIFTING PHASE (Details are listed on separate sheets; only general control is mentioned here) NOTE: The number in AB is the absolute value of the of the mantissa		
	NP25 ⇒ shift left by 4's until SHEX (= P2531Z + P1517W + E0 + NA0811Z) causes exit P25 ⇒ " right " 2's " " (= P2531Z + P1517W + E0) causes exit		(1-15 clocks) (1-29 clocks)
	S/RTZ (set up "RESULT ZERO" test)	S/RTZ = (S/RTZ/3) = FASHFL.PH9.N06 (mech. conv.)	
	A → S	SXA = FASH.PH9	for AXSL4
	A → PR	(D and CS are cleared, hence PR = A)	for AXPRR2
	sustain PH9	BRPH9 = FASH.PH9, NSHEX/1, where NSHEX/1 = N(NP25.P2629Z.NP30.NP31 + P15.P16.P17 + E0 + NA0811Z.FASHFL.NP25)	(shifting in process)
	Functions performed on last clock (i.e.	SHEX, which = FASH.PH9.NNSHEX/1)	("shift exit)
	R/IEN (terminate interruptability)	R/IEN = SHEX	
	short only : MRQ/1 (Q → P, etc) S/PH14	MRQ/1 = SHEX.N(FASHFL.C23) BRPH14 = SHEX.FASHFL.NC23	(P and C are free)
	long only : S/CXS	S/CXS = SHEX.FASHFL.C23	(for A → S → C in PH10)

SF 2 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH10 T6L	(entered if long only) A → S ↛ C R/RTZ if NA0031Z (zero test m.s.w.)	SXA = FAMDSF.PH10 ; (CXS was set in PH9) { S/RTZ = (S/RTZ/3) = FASHFL.A0031Z.RTZ.N06 (mech. conv.) { R/RTZ = (always hi.)	(save m.s.w.) (RTZ was set in PH9; drop if A ≠ 0)
PH11 T6L	(entered if long only) B → S ↛ A (R/RTZ if NA0031Z - redundant) S/NGX if original operand negative R/K00H S/LR31/2 S/T10L	SXB = FASHFL.PH11 ; AXS = FASHFL.PH11 hold RTZ if A0031Z - (see PH10) S/NGX = FASHFL.PH11.RN S/K00H is inhibited by (FASHFL.PH11) (see PH1) S/LR31/2 = FASHFL.PH11 S/T10L = FASH.PH11	(l.s.w. ↛ A) (contents of A same as PH10) } set up for Ā+1 → S ↛ RWv1 in PH12 (for RWv1)
PH12 T10L	+ operand: A → S (entered if long only) − operand: Ā+1 → S NK00 ↛ K00H S/NGX S ↛ RWv1 R/RTZ if NA0031Z (zero test l.s.w.) E0107 ↛ A0107 (0 ↛ rest of A) C ↛ D (c0007 will contain zeros) MRQ/1 (Q ↛ P, etc) S/DRQ S/PH15 S/T10L	SXADD = FASHFL.PH12 " = " S/K00H = S00 = NK00 S/NGX = FASHFL.PH12.RN RW = FASHFL.PH12 hold RTZ if A0031Z - (see PH10) AXE = FASHFL.PH12 (AXE raises AX) DXC/6 = FASHFL.PH12 MRQ/1 = FASHFL.PH12 S/DRQ = FASH.PH12 BPH15 = FASHFL.PH12 S/T10L = FASH.PH12	} logic same as PH2 store shifted l.s.w. (E0 ↛ A0 blocked by FAMDSF) (m.s.w. ↛ D0831)

SF 3 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH14 T6L	(entered if short only) AvE0107 ↛ A (merge) R/RTZ if NA0031Z S/NGX if original operand negative R/K00H S/DRQ S/T10L	AXE/1 = FASHFL.PH14 (does not raise AX) hold RTZ if A0031Z - (see PH10) S/NGX = FASHFL.PH14.RN S/K00H is inhibited by (FASHFL.PH14) (see PH1) S/DRQ = FAMDSF.PH14 S/T10L = FAMDSF.PH14	(A0007 contains zeros) } for Ā+1 → S ↛ RW in PH15
PH15 T10L	(final phase of both short and long) + operand (non-zero result): short: A → S long: AvD → S − operand (non-zero result) short: Ā+1 → S long: (ĀvD)+1 → S if end carry in PH12 = 1 : (ĀvD) → S " " " " " =0 (0 → S if zero result in mantissa) S ↛ RW S ↛ A 1 ↛ A3; if result not zero (merge) } For CC3, CC4 contr. S/TESTA s/cc1 if result = 0 in left shift case s/cc2 if result ≠ 0 and exp. over/underflow	} SXADD = FASH.PH15.NRTZ } controlled by NGX and K00H RW = FAMDSF.PH15 AXS = FASHFL.PH15 A31X1 = FASHFL.PH15.NRTZ (for long case where S/TESTA = FAMDSF.NFASHFX.PH15 S/CC1 = FASHFL.PH15.NBWZ.RTZ S/CC2 = FASHFL.PH15.E0.NRTZ	A0107 has exp; D0831 has mantissa (other bits = 0) (exponent made = 0) exp = 0, m.s.w = 0, l.s.w ≠ 0) (see also PH9)

SF 4 of 4

FLOATING POINT CONTROL FLAGS (FZ, FN, FS):

FZ. Floating zero. (Applies to all floating operations)

FZ = 0: Underflow causes the result to be set equal to true zero, the UCC set to 11 and the LCC set to 00.
(Exception: if a trap results from significance checking the result of an addition or subtraction, an underlow generated in the process of postnormalizing is ignored when FZ = 0)

FZ = 1 Underflow causes a trap to 68. Result storage does not occur, UCC is set to 11, and the LCC will be set to reflect the polarity of the result (01 for <0, 10 for >0)

for add and subtract only:

FN Floating normalize. (Applies to add and subtract only)

FN = 0: Results of additions or subtractions are postnormalized. CC2 is set if more than two postnormalizing shifts are required or if the result is zero.

FN = 1: Inhibit postnormalization of results of additions or subtractions. UCC will inevitably remain = 00

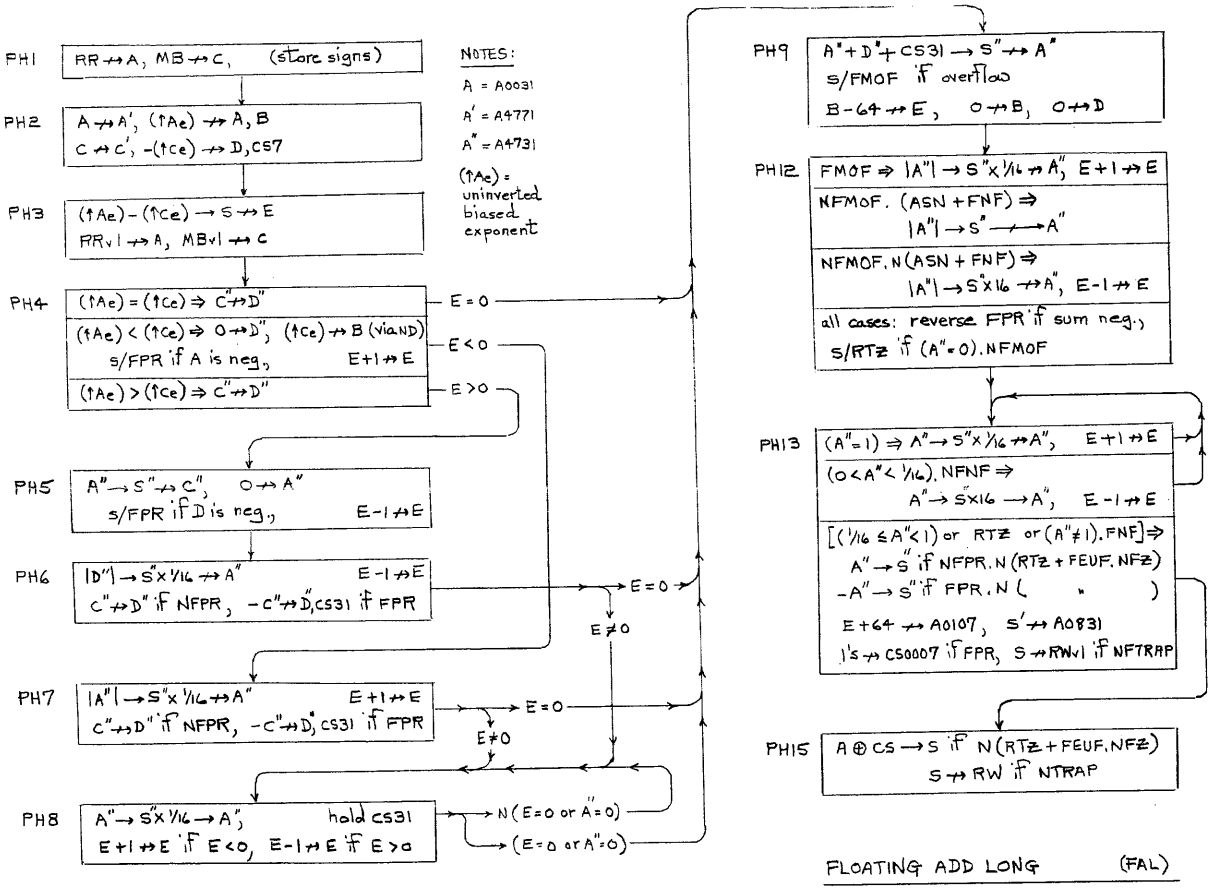
FS Floating significance (Applies to add and subtract only)

FS = 1 with FN = 0 will cause a trap to 68 if more than two postnormalizing shifts are required or if the result is zero. Result storage does not occur, but LCC will be set to indicate the result (00 for zero, 01 for <0, 10 for >0). CC1 is set, and CC2 will be set if an underflow resulted from postnormalizing and FZ = 1.

C.C. SUMMARY:

UCC	LCC	NO TRAP	TRAP to 68
00	00	* N×0, 0/N, or [A-A or A+(-A) with FN=1]	—
	01	N<0	—
	10	N>0	—
	11	—	—
01	00	—	divide by zero } always trapped
	01	—	overflow, n<0 } always trapped
	10	—	overflow, n>0 } always trapped
	11	—	—
† 10	00	* A-A or A+(-A) } FS=0 and FN=0 and no underflow	A-A or A+(-A) } FS=1 and FN=0 and not (underflow with FZ=1)
†	01	N<0 } >2 postnormalizing shifts	N<0 } >2 postnormalizing shifts
†	10	N>0 } >2 postnormalizing shifts	N>0 } >2 postnormalizing shifts
†	11	—	—
11	00	* Underflow with FZ=0 and no trap by FS=1	—
	01	—	underflow, N<0 } FZ=1
	10	—	underflow, N>0 } FZ=1
	11	—	—

* result set to true zero
† applies to add and subtract only where FN=0
— indicates impossible configurations



FLOATING ADD LONG (FAL)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
T4RL PRE2.NIA	RR → A (insig.)	AxRR = FAMDSF.PRE2.NIA	(repeated significanly next phase)
PH1 T4RL	A → S → B (insig) RR → A (m.s.w. of augend) RR0 → RN (store sign of augend) MB → C (m.s.w of addend) MB0 → COC16 → MWN (store addend sign) s/CXCL32 (for C0 → C47, C0831 → C4871 in PH2) R/CC1 (for exp. underflow test in PH15) R/CC2 (" " under/overflow " " ") 1's → CS0007 (to invert A0007 if A is neg.) s/IEN (start interruptability)	BxS = FAMDSF.PH1, SxA = FAFL.PH1 AxRR = RNxRR0/2.PH1 s/RN = RR0.RNxRR0; RNxRR0 = RNxRR0/2, PH1 (R/RN = CLEAR) s/MWN = FAMDSF.PH1.COC16, (R/MWN = CLEAR) s/CXCL32 = FAFL.PH1 R/CC1 = FAFL.PH1 R/CC2 = FAMDSF.NFAMULH.PH1 CSx1/5 = FAFL.PH1 s/IEN = FAMDSF.NFAMUL.PH1	(for long FAFLM) RNxRR0/2 = FAFL (also used for significance test in PH15) exp. bits are inverted when neg, no
PH2 T6L	A0 → A47, A0831 → A4871 (mantissa) C0 → C47, C0831 → C4871 (") Exponent differencing set-up: ($RR_e - MB_e$) - augend : A ⊕ CS0007 → S + " : A → S S → A0007 - addend : C → D + " : $\overline{C}$ → D 1 → CS7 (for 2's complement) S → B (save uninverted augend exp.)	AxAL32 = FAFL.PH2 CxCL32 (F.F. set in PH1) SxPR = FAFL.PH2.RN SxA = FAFL.PH2.NRN AXS/1 = FAFL.PH2 DxC/6 = FAFL.PH2.(MWN ⊕ FAFLM) ← MWN DxNC = FAFL.PH2.N(MWN ⊕ FAFLM) ← NMWN CS7x1 = FAFL.PH2.NFAFLM BxS = FAFL.PH2.N(FAFLM, N02)	m.s.w. of augend → A extension " " addend → C " uninverted augend exp → A0107 (0 → A0, A0831) inverted addend exp → D (only D0007 are signif.) "+1" → CS0007 (only B0007 are signif.)

FAL (1D), FAS (3D), FSL (1C), FSS (3C) "FAFLAS" 1 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 T&L	(continued) Access of l.s.w. of operands: (bits 0031) S/LB31/1 MRQ, S/DRQ } if long S/CX/1 if short } (addend) S/LR31/2 (augend) enable T&RL if long	S/LB31/1 = (FAFL.PH2) MRQ = (").N02 S/DRQ = (").N02 S/CX/1 = (").02 S/LR31/2=(") T&RL = (").N02	causes MB∨1 → C in PH3 " 0 → C " " (for RR∨1 → A in PH3 if long) (0 → A " " " short)
PH3 T&L if short, T&RL if long	A4771 → S4771 → B4771 (insig) A+D+CS7 → S0007 → E0007 (RR_e, $-MB_e$) 0 → P (insig) 0 → A4731 S4771 → A4771 (regenerate A4771) RR∨1 → A0031 if long (zeros if short) 1 → NGX if RN (insig) 0 → C0031 if short MB∨1 → C0031 if long	SXA/3 = BXS/1 = FAFL.PH3 SXK/1 = SXPR/1 = EXS = FAFL.PH3 PX =(FAFL.PH3) AX/1=(") AXS/4 = (FAFL.PH3.N(FAFLM.ASN)) AXRR = (").N02 S/NGX = (").RN see PH2 "	(for long FAFLM) unbiased exponent difference (for FAFLMD) S4771=A4771; see SXA/3 above l.s.w. of augend (for FAFLMD) l.s.w. of addend
PH4 T&L	NOTE: B0107 holds the augend exp., D0107 holds the inverted addend exponent. CASE 1: E=0 (exponents equal - set up for add/subtract) C → D if add $\overline{C}$ → D } if subtract 1 → CS31 S/PH9, S/T8L	DXC/6 = (FAFLAS.PH4.EZ).07 DXNC/1 = (").N07 " = ("). " BRPH9=S/T8L = (")	

FAL, etc.

2 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T&L	(continued) CASE 2: E>0 (augend (A) > addend (C); right shift |C|) C → D (addend) S/CXS (advance to PH5)	DXC/6 = FAFLAS.PH4.NEO.NEZ S/CXS = "	(for A → S → C in PH5)
	CASE 3: E<0 (addend (C) > augend (A); right shift |A|) 0 → D 1 → NGX, 1 → FPR } if A negative $\overline{D0107}$ → B0107 (larger exp. → B) E+1 → E (precount E toward zero) S/PH7 S/T8L	DX/1 = (FAFLAS.PH4) S/NGX = (FAFLAS.PH4.EO).RN S/FPR = (").RN BXND = (") PCTE1 = (") BRPH7 = (") S/T8L = (")	set up for |A| → S in PH7 remembers polarity reversal D0107 contains inverted exp.
PH5 T&L	(entered from PH4 if E>0) A → S → C 0 → A 1 → NGX if addend (D) is negative E-1 → E (pre-count E toward zero) 1 → FPR if the |addend| is opposite to that desired for the add or subtract (i.e. S/T8L (advance to PH6)	SXA = (FAFLAS.PH5); CXS was set in PH4. AX/1 = (") S/NGX = (").MWN MCTE1 = (") S/FPR = (FAFLAS.PH5)(07.MWN + N07.NMWN) or S/T8L = (")	larger no → C set up for |D| → S in PH6 ADD with NEG. addend SUB " POS. ")

FAL, etc.

3 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6 T8L	(entered only from PH5 (E is now ≥0) |D| → SX 1/16 → A { C → D if $\overline{FPR}$ { $\overline{C}$ → D, 1 → CS31 if FPR { E-1 → E (count toward zero) { if E ≠ 0: s/PH8 { if E = 0: s/PH9, s T8L	- (shift |original addend|) SXADD = AXSR4 = FAFLAS.PH6 DXC/6 = (FAFLAS.PH6).NFPR DXNC/1 = (").FPR MCTE1 = (") BRPH8 = ( ).NE≠ BRPH9 = s/T8L = ( ).E≠	(NGX was set in PH5 if MWN) { original augend → D, polarity as per FPR (see PH5) (go to "keep shifting" phase) (go to ADD phase), (E counts to -1)
PH7 T8L	(entered only from PH4 if E was <0 (E now is ≤0)) - (shift |original augend|) |A| → SX 1/16 → A { C → D if (ADD.$\overline{FPR}$ + SUB.FPR) { $\overline{C}$ → D, 1 → CS31 if (") { E+1 → E (count toward zero) { if E ≠ 0: (advance to PH8) { if E = 0: s/PH9, s/T8L	SXADD = AXSR4 = FAFLAS.PH7 DXC/6 = (FAFLAS.PH7).(07 ⊕ FPR) DXNC/1 = (").N(") PCTE1 = (") BRPH9 = s/T8L = (").E≠	(NGX was set in PH4 if RN) { original addend → D as per ADD/SUB, but reversed if FPR was set in PH4 (go to "keep shifting" phase) (go to ADD phase), (E counts to +1)
PH8 T6L	(entered either from PH6 or PH7, but only if the original exponent difference was >1) { A4771 → S4771 } if short { 0's → S0031 } { A4731 → S4731 if long { SX 1/16 → A { E+1 → E if E<0 } (count twd. zero) { E-1 → E if E ≥0 } hold CS31 (for PH9) sustain PH8 if (E≠0).(A4731≠0) advance to PH9 if (E=0)+(A4731=0) s/T8L if (E=0) by (A4731=0) since A, D, CS will have two clocks to allow for carry propagation.	SXA/3 = (FAFLAS.PH8).02 SXA = (").NO2 AXSR4 = (") PCTE1 = (").EO MCTE1 = (").NEO R/CS31 = N(") BRPH8 = (").NE≠.NA4731≠ s/T8L = (").E≠; (note: T8L not necessary if PH9 reached	(0-8 clocks (short); 0-15 (long)) { in effect, in short operations significance is lost beyond A0003. This 4 bit group is the "guard digit" (terminal state will = -1 unless exit caused by A4731≠) (repeater FF)

4 of 9

FAL, FAS, FSL, FSS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9 T8L or T6L ↑ (see PH8)	"ADD PHASE" (can be entered from: PH4 if exponents were equal; PH6 or PH7 if differed by more than 1; PH8 if exponents differed by 1) (NOTE: If FPR is on, it signifies the polarity of the sum obtained in this phase is reversed) A + D + CS31 → S → A s/FMOF if overflow (mantissa) B0107 - 64 → E0007 0 → B (clear for postnorm. ctr. use) 0 → D 1 → NGX } set up for |A| → S in PH12 MRQ/1 R/IEN (stop interruptability) s/PH12 s/T8L	SXADD = AXS = FAFLAS.PH9 s/FMOF = (FAFLAS.PH9).(A47.D47.NK47 + NA47.ND47.K47); (does not include -1) { NB1 → E0001, B0207 → E0207, gated by { EXB = (FAFLAS.PH9) BX/1 = (") DX/1 = (") s/NGX = (") MRQ/1 = (") R/IEN = (") BRPH12 = (") s/T8L = (")	{ unbiased exponent of sum → E
PH12 T8L	|A| → S: $\overline{A}$+1 → S if negative sum A → S " positive " CASE 1: FMOF - shift |A| right SX 1/16 → A 1 → A50 if A4731≠ E+1 → E (adjust exp.)	SXADD = (FAFLAS.PH12).(A47 ⊕ FMOF) SXA = (").N(") (1≤A<2; -2≤A<-1) AXSR4 = FMOF s/A50 = FMOF.A4731≠ PCTE1 = FMOF { s/FMOF - see PH9; R/FMOF is always high	(NGX is on) (S47 represents +2°) (A = -2 case, yields + 2/16)

5 of 9

FAL, etc.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH12 T8L	(continued)		
	CASE 2: $\overline{FMOF}$. (ASN + FNF) – S → A (|A| → A)	(1/16 ≤ A < 1 or -1 ≤ A < -1/16 or (inhibit post-normalization). $\overline{FMOF}$) AXS = FAFLAS.PH9.NFMOF.NFPPN/2.	(NFPPN/2 reduces to ASN + FNF)
	CASE 3: $\overline{FMOF}$. ($\overline{ASN}$. $\overline{FNF}$) – S×16 → A; E-1 → E (adjust exp); 1 → BC31 (for post-norm. ctr.)	((-1/16 ≤ A < 1/16) and post-normalize → shift |A| left. AXSL4 = FPPN, MCTE1 = ", S/BC31 = " } FPPN = FPPN/2 = FAFLAS.PH12.NFMOF.NASN.FPPN/1, where FPPN/1 reduces to NFNF	
	S/RTZ if $\overline{FMOF}$. A4731Z	S/RTZ = (S/RTZ/1), which reduces to FAFLAS.PH12.NFMOF.A4731Z	result = 0
	0 → D (redun.); 1 → NGX; S/G0003/1 if short (causes "K71"); S/LR31/2; S/T10L if long	DX/1 = (FAFL.PH12); S/NGX = ("); (S/G0003/1) = (").NFPRD; reset by CLEAR; S/LR31/2 = ("); S/T10L = FPRD.PH12	} set up for negation and short truncation (for S → RWV1 if long)
	FPR logic: reverse FPR if the sum generated in PH9 was negative.	S/FPR = (S/FPR) = (FAFLAS.PH12).(A47 ⊕ FMOF).NFPR; R/FPR = (R/FPR) = (").(")	
PH13 TGL or T10L	(1-6 clocks if short (all TGL); 1-13 clocks if long (first and last T10L, all others TGL)) NOTES: A contains the absolute value of the sum; if FPR is on (from control in PH4, 5, 12) it signifies the result is negative and suitable adjustments will be made when storing the result.		

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH13	(continued)		
	CASE 1: (FAFL.PH13.NFPRR) "RESULT NOT READY" FOR STORAGE; requires shift left or right. (T10L for first clock if long)		
	A → S; sustain PH13; " T10L if long and < 2 shifts; " LR31/2; " NGX; BX2 → B (signif. if FPPN-below)	SXA = (FAFL.PH13.NFPRR); BRPH13 = ( ); S/T10L = ( ).N(FPPN.A52552).FPRD; S/LR31/2 = ( ); S/NGX = ( ); BXBL1 = ( )	(for post-norm. ctr.)
	A47 ⇒ (FAFL.PH13.A47) (i.e. |sum| = +1; can happen if PH9 produced -1/16 and FNF is off, or -1) Sx1/16 → A (logical); E+1 → E (adjust exp.)	AXSR4 = (FAFL.PH13.A47); PCTE1 = (") } (0-1 clock only; advance to CASE 2)	
	$\overline{A4751Z}$. RTZ . $\overline{FNF}$ ⇒ FPPN (i.e. 0<|sum|<1/16 and "FN" is not inhibiting post-normalization) (NOTE: 0-5 clocks if short (all TGL); 0-12 clocks if long (first clock T10L, all others TGL))		
	SX16 → A (0's → A2831); E-1 → E (adjust exp.); 1 → BC31 (for post-norm. ctr.)	AXSL4 = FPPN, MCTE1 = ", S/BC31 = " } FPPN = FPPN/3 = FAFL.PH13.A4751Z., FPPN/1, where FPPN/1 reduces to NRTZ.NFNF	
	Post-normalization counting logic: B was cleared in PH9 and shifts left by one's in PH13 until the result is ready for storage. BC31 is set each time the result is post-normalized (in PH12 or PH13). As B shifts left, B30 ← B31 ← BC31, hence if B31 contains a 1 and post-normalizing is still in process, it signifies > 2 shifts are required; also if B30 contains a 1 when the result is ready for storage (or in PH15), it means > 2 shifts took place.		
	Conditionally "kill" underflow indication: 1 → E1 (merge) exponent if > 2 shifts required and flags require trap on significance	E1X1 = FAFLAS.FPPN.FS.NFZ.E0.B31	(Merge +64 with negative but not on underflow)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH13	(Continued)		
	CASE 2: $\overline{A47}.(\overline{A4751Z} + RTZ + FNF) \Rightarrow$ (i.e. $1/16 \leq \lvert SUM \rvert < 1$,	FPRR: "RESULT READY" FOR STORAGE (1 clock: T10L if long, T6L if short) or SUM = 0 or ($\lvert SUM \rvert < 1$ and "FN" is inhibiting post-normalization))	
	$\overline{A}+K \rightarrow S$ if sum neg. $A \rightarrow S$ " " pos. $0 \rightarrow S$ if result = 0 or if exponent underflow with the trap flag (FZ) off (note that the underflow indication is killed if >2 postnormalizing shifts were required and the FS, $\overline{FZ}$ configuration existed in the flags; hence when significance trap occurs, CC3 and CC4 will be set as a function of the sum rather than by zero, forced by FEUF. $\overline{FZ}$)	SXADD = FPRR.N(RTZ + FEUF.NFZ).FPR SXA = " .N(").NFPR	K = K31 due to NGX; also "K71" is high (for proper truncation) if G0003/1 was set in PH12.
	$S \rightarrow\rightarrow RW \vee 1$ if long	RW = (FPRR).FPRD.NFTRAP	inhibit if trap.
	S4871 $\rightarrow\rightarrow$ A0831 E+64 $\rightarrow\rightarrow$ A0107 (0 $\rightarrow\rightarrow$ A0)	AXSR32 = (").N(FAFLD.02) AXE = (") (i.e. $\overline{E1} \rightarrow$ A1, E0207 $\rightarrow$ A0207)	m.s. mantissa bits. uninverted exp. plus bias.
	1's $\rightarrow\rightarrow$ CS0007 if result neg.	CSX1/5 = (").FPR (for sign insertion	and exponent inversion)
	S/DRQ	S/DRQ = (")	
	S/T8L	S/T8L = (")	
	S/PH15	BRPH15 = (")	
	S/RTZ if short and A4771 = 0	S/RTZ = (S/RTZ/3) = FPRR.NFPRD.N06.A4771Z	needed for short case where FN=1 and significance is confined to A0003 (guard digit)
	S/TRAP, S/TR29 } trap to 68_{10}	S/TRAP = (FPRR).FTRAP S/TR29 = (").FTRAP	
	Trap conditions are: overflow (unconditional) underflow (if FZ on) insignificant result (if $\overline{FN}$.FS)	FTRAP = + FEOF (which = NEO.E1.NRTZ.FAFL) + FEUF.FZ (where FEUF = EO.NE1.NRTZ.FAFL) + FRINSIG.FS, (where FRINSIG = FAFLAS.B30 + FAFLAS.NFNF.RTZ)	>2 post-norm shifts (implies $\overline{FN}$) (result = 0). $\overline{FN}$

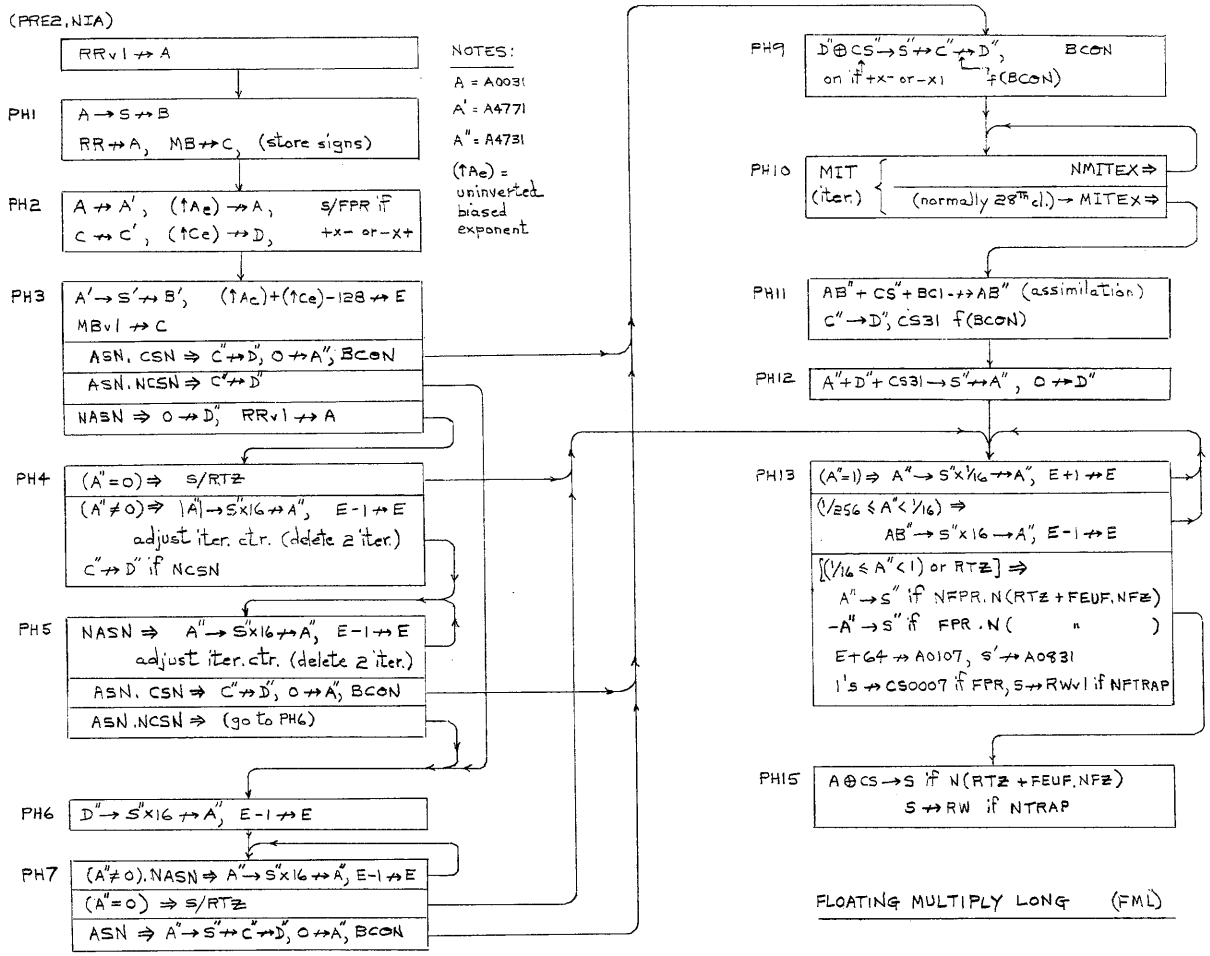
FAL, FAS, FSL, FSS

8 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH15 T8L	ENDE	ENDE = FAMDSF.PH15	
	$A \oplus CS \rightarrow S$ if result $\neq 0$ (see PH13) $S \rightarrow\rightarrow RW$ if NTRAP	SXPR = FAFL.PH15.N(RTZ + FEUF.NFZ) RW = FAMDSF.PH15; RWDIS = TRAP.NINTRAPF	CS0007 are on if result neg. (inhibited by trap)
	for CC3,4 contr.: $S \rightarrow\rightarrow A$	AXS = FAFL.PH15	(needed for sign bit)
	for CC3,4 contr.: 1 $\rightarrow\rightarrow$ A31 (merge) if result $\neq 0$ indication (CC3 = 1) instead of = 0 in a case where the result is positive, the biased exponent = 0, and significance is confined to the l.s. 32 bits of a long operation)	A31X1 = FAFL.PH15.N(RTZ + FEUF.NFZ)	(needed to assure >0) (needed for unnormalized (FN=1) ADD/SUB
	for CC3,4 contr.: S/TESTA	S/TESTA = FAMDSF.NFASHFX.PH15	
	1 $\rightarrow\rightarrow$ CC1	S/CC1 = FAFL.PH15.FEUF + FAFL.PH15.FRINSIG ← see PH13	(exp. underflow) (insignificant result). $\overline{FN}$
	1 $\rightarrow\rightarrow$ CC2	S/CC2 = FAFL.PH15.FEUF (exp. underflow) + FAFL.PH15.FEOF	(inhibited if FS Trap an FZ = 0) (exp. overflow)

FAL, etc.

9 of 9



PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
(Phase preceding PRE2.NIA):	S/LR31/2 if long=	(PRE1.NIA + PRE3.IX + PRE4).OV1.OLF	for RRv+1 during PRE2.NIA
PRE2.NIA: T4 RL	long: RRv1 → A short: RR → A	AxRR = FAMDSF.PRE2.NIA " = "	l.s.w. of multiplier multiplier
PH1 T4 RL	A → S → B	BxS = FAMDSF.PH1, SxA = FAFL.PH1	(0 → B4771 since A4771 = 0)
	RR → A (redundant if short)	AxRR = RNxRR0/2.PH1	m.s.w of multiplier
	RR0 → RN (store multiplier sign)	S/RN = RR0.RNxRR0; RNxRR0 = RNxRR0/2.PH1	RNxRR0/2 = FAFL
		(R/RN = CLEAR)	
	MB → C		m.s.w of multiplicand
	MB0 → COC16 → MWN (store multiplicand sign)	S/MWN = FAMDSF.PH1.COC16, (R/MWN = CLEAR)	
	S/CXCL32 (for C0 → C47, C0831 → C4871 in PH2)	S/CXCL32 = FAFL.PH1	
	R/cc1 (for exp. underflow test in PH15)	R/cc1 = FAFL.PH1	
	R/cc2 (" " under/overflow " " ")	R/cc2 = FAMDSF.NFAMULH.PH1	
	1's → CS0007 (to invert A0007 if A is neg.)	CSx1/5 = FAFL.PH1	exp. bits are inverted when neg.no.
	S/IEN (start interruptability)	S/IEN = FAMDSF.NFAMUL.PH1	
PH2 T6L	A0 → A47, A0831 → A4871 (mantissa)	AXAL32 = FAFL.PH2	m.s.w. of multiplier → A extension
	C0 → C47, C0831 → C4871 (mantissa)	CXCL32 (F.F. set in PH1)	" " multiplicand → C "
	Exponent summing setup:		
	− multiplier: A⊕CS0007 → S	SXPR = FAFL.PH2.RN	uninverted multiplier
	+ " : A → S	SxA = FAFL.PH2.NRN	exp. → A0107
	S → A0007	AxS/1 = FAFL.PH2	(0 → A0, A0831)
	− multiplicand: C̄ → D	DxNC = FAFL.PH2.N(MWN⊕FAFLM) ← = MWN	uninverted multiplicand
	+ " : C → D	DxC/6 = FAFL.PH2.(MWN⊕FAFLM) ← = NMWN	exp. → D (only D0007 are signif.)
	1 → CS0 (to remove 2x bias from sum)	S/CS0 = FAFLM.PH2	"−128" → CS0007

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 T6L	(continued) S ↛ B if short : (replace contents of B with short multiplier; only bits 8-31 are significant and S0831 reduce to A0831 regardless of whether S is controlled by SxA or SXPR (as per above)) 1 ↛ FPR if final product is to be negative Access of l.s.w. of operands : (bits 0031) s/LB31/1 MRQ } if long s/DRQ } s/cx/1 if short } (multiplicand) s/LR31/2 (multiplier) enable T6RL if long	BxS = FAFL.PH2.N(FAFLM.N02) S/FPR = (S/FPR) = FAFLMD.PH2.(MWN⊕RN) S/LB31/1 = (FAFL.PH2) MRQ = (").N02 s/DRQ = (").N02 s/cx/1 = (").02 s/LR31/2 = (") T6RL = (").N02	(0 ↛ B4771 since A4771 = 0) } causes MBv1 ↛ C in PH3 " 0 ↛ C " " (for RRv1 ↛ A in PH3 if long) (0 ↛ A " " " short)
PH3 T6L if short, T6RL if long	A4771 → S4771 ↛ B4771 (insig. if short) A + D + CS0 → S0007 ↛ E0007 0 ↛ P 0 ↛ C0031 if short MBv1 ↛ C0031 if long 0 ↛ A4731 ASN case: (A not "simple-normalized") S4771 ↛ A4771 (regenerate A4771) RRv1 ↛ A0031 if long (zeros if short) S/NGX if multiplier negative s/T8L 0 ↛ D (advance to PH4 to normalize |A|)	SXA/3 = FAFL.PH3, BXS/1 = FAFL.PH3 SXK/1 = SXPR/1 = EXS = FAFL.PH3 PX = FAFL.PH3 see PH2 " AX/1 = FAFL.PH3 (all of A will clear unless AXS/4 = (FAFL.PH3.N(FAFLM.ASN)) AxRR = (").N02 S/NGX = (").RN S/T8L = (").FAFLMD DX/1 = FAFLMD.PH3	(does not affect B0031 or S0031) (unbiased exponent sum) (P1518 used as iterations ctr.) set terms below are active) (ASN case) S4771 = A4771; see SXA/3 above. for |A| → S in PH4

FML, FMS

2 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3	(continued) ASN case: (A is "simple-normalized") C ↛ D (multiplicand) 0 ↛ A (multiplier is in B) ASN.CSN : s/PH6 ASN.CSN ⇒ MPP, which causes: MPP ⇒ { 0 ↛ A; C ↛ D; 1 ↛ CS if signs opposite; s/CXS; BCON; s/PH9; R/IEN; s/T8L }	DXC/6 = FAFLM.PH3.ASN AX/1 is high and no set terms are active BRPH6 = FAFLM.PH3.ASN.NCSN MPP = FAFLM.PH3.ASN.CSN AX/1 = MPP DXC/6 = MPP CSX1 = MPP.(MWN⊕RN) S/CXS = (S/CXS/1) = MPP BCON = BxBR2.FAMDSF/M.etc, which red.to MPP BRPH9 = MPP R/IEN = MPP S/T8L = (S/CXS/1) = MPP	to simple-normalize multiplicand prepare for iterations } redundant in PH3 but needed if MPP rises in PH5 or PH7 (positive product scheme) for D⊕CS ↛ C ↛ D f(BCON) functions described separately pre-iterations Phase stop interruptability
PH4 T8L	(entered only if the multiplier is not normalized) if A=0: s/PH13, s/RTZ, MRQ/1, R/IEN, s/LR31/2, sT10L if not (short.Rodd) |A| → S FPRENR causes: } insig if A=0 SX16 ↛ A E - 1 ↛ E up-count P1518 if A≠0 C ↛ D if NCSN (advance to PH5 if A≠0) NOTE: normalization of the |A|, instead of A, avoids the case of the smallest possible negative multiplier (all 1's) being normalized to become "-1". This would require special control logic to circumvent PH10.	{ S/RTZ = BRPH13 = R/IEN = S/LR31/2 = (S/RTZ/1), which MRQ/1 = (S/RTZ/1).FAFLMD, S/T10L = (S/RTZ/1).FPRD SXADD = FAFLMD.PH4 FPRENR = FAFLMD.PH4.NASN AXSL4 = FPRENR MCTE1 = FPRENR PCTP5 = PCTP5/1 = FPRENR, NA4731Z DXC/6 = FAFLMD.PH4.NCSN	} reduces to FAFLM.PH4.A4731Z = (result = 0) Ā+1 if NGX, A if GX. "prenormalize R operand." |A| shifts left one hex. down-count exponent. delete 2 multiply iterations. (for D → S in PH6)

FML, FMS

3 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T6L	(entered only if multiplier was an unnormalized non-zero number) (1-5 clocks if short, 1-13 clocks if long) $\overline{ASN} \Rightarrow$ FPRENR : A → SX16 ↛ A E-1 ↛ E } (0-4 clocks if short, 0-12 " " long) up-count P15I8 sustain PH5 ASN . $\overline{CSN}$: (advance to PH6) ASN . CSN ⇒ MPP S/PH9, etc - see descr. PH3	FPRENR = FAFLMD.PH5.NASN SXA = FAFL.PH5, AXSL4 = FPRENR MCTE1 = FPRENR PCTP5 = PCTP5/3 = FPRENR.NA4731Z BRPH5 = FPRENR.PH5 MPP = FAFLM.PH5.ASN.CSN	} normalize, etc (same as PH4) to normalize multiplicand prepare for iterations
PH6 T6L	(entered only if multiplicand is unnormalized; can be entered from PH3 or PH5) D → S FPRENM causes SX16 ↛ A E-1 ↛ E } insig. if D = 0 S/CXS S/T8L (advance to PH7)	SXD = FAFLMD.PH6 FPRENM = FAFLMD.PH6.N(PH7.ASN) = FAFLMD.PH6 AXSL4 = FPRENM MCTE1 = FPRENM.FAFLM S/CXS = FPRENM.NA4731Z + (S/CXS/1) S/T8L = (S/CXS/1) ← = FAFLMD.PH6	C is in D (+ or - no.) "prenormalize M operand" multiplicand X 16 ↛ A down-count exponent (for A→S↛C↛D in PH7)
PH7 T6L T8L	(entered from PH6 only) (1-6 clocks if short, 1-14 clocks if long; maximum applies only to "all 1's" multiplicand) if A = 0 : S/PH13, etc. same as in PH4 (first clock only)	(S/RTZ/1) reduces to FAFLMD.PH7.A4731Z	result = 0

FML, FMS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7	(continued) if A ≠ 0 : $\overline{ASN} \Rightarrow$ FPRENM : A → SX16 ↛ A } insig if A = 0 E-1 ↛ E S/CXS } only if A ≠ 0 sustain PH7 ASN ⇒ MPP S/PH9, etc. - see descr. PH3 S ↛ C (all cases)	FPRENM = FAFLMD.PH7.N(PH7.ASN) = FAFLMD.PH7.NASN SXA = FAFLMD.PH7, AXSL4 = FPRENM MCTE1 = FPRENM.FAFLM S/CXS = FPRENM.NA4731Z BRPH7 = FPRENM.NA4731Z MPP = FAFLM.FAFLMD.PH7.ASN (surplus, mech. convenience) CXS is on throughout PH7	} continue prenormalization (1-5 clocks if short, 1-13 " " long) prepare for iterations (For A→S↛C↛D when MPP)
PH9 T8L	(entered from PH3, PH5, or PH7 as function of MPP) D⊕CS → S ↛ C C ↛ D f(BCON) BCON (described separately) 0 ↛ D (related to BCON logic) S/MIT (MIT is a fast PH10.FAMDSF/M) S/TIL (for single level timing) S/FLMC if significance in the original multiplier existed only in the least significant hex. (i.e. only two clocks of PH10 to be used)	SXPR = FAMDSF/M.PH9; CXS was set by MPP, CS's are on if (MWN⊕RN) (described separately) BCON = BXBR2.FAMDSF/M, etc, which reduces to DX/1 = (FAMDSF/M.PH9) S/MIT = (") (repeater) S/TIL = (").NKSC.N((S/FLMC/1).P18) S/FLMC = PH9.P18.(S/FLMC/1), where (S/FLMC/1) = FAFLM.P16(P15+02)	(match multiplicand sign to multiplier for |product|) FAMDSF/M.PH9 (sustained in PH10 until MITEX) inhibit when single clocking or when only two PH10 clocks

FML, FMS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH10 TIL →	= "MIT" (multiply iterations) (12 clocks if short, 28 clocks if long; subtract 2 x number of PH5 clocks) (TIL except when only 2 clocks are required; timing will be T6L in that case) NOTE: Register organization, bit control, and other such details are described separately (under "MIT functions"). Only general control functions are noted below.		
	FLMC is high during the next to last clock MITEX " " " " last clock MP19 " " " { even numbered clocks, excluding last.	s/FLMC = MP19. (s/FLMC/1) ← see phase 9 s/MITEX = FLMC s/MP19 = BCON. (s/MP19/1), which reduces to MIT. NFLMC. NMITEX. NMP19	} resets always high } 5-bit iterations ctr
	up-count PIS18 when MP19 = 1	PCTP5 = MP19	
	BCON	BCON = BxBR2. FAMDSF/M. NFLMC. etc, which reduces to MIT. NFLMC	
	sustain PH10 until last clock	BRPH10 = MIT. NMITEX. NCLEAR	
	R/TIL (initiate end of TIL (2 clocks in adv.))	R/TIL = MP19. (S/FLMC/1) + NMIT	(NMIT for initial reset)
	R/MIT (repeater) s/T8L (advance to PH11) } on last clock (MITEX)	S/MIT = MIT. NMITEX. NCLEAR s/T8L = MITEX. FAFLM	
PH11 T8L	(Assimilation phase) A + CS + K31 → S ↛ A B + CS32, CS33, BC1 ↛ B	SXADD = FAMDSF/M. PH11, AXS = FAMDSF/M. PH11 BxB = FAMDSF/M. PH11	K31 is end carry from B (affects B0007 only)
	(D contains zero) C ↛ D f(BCON) 1 ↛ CS31 if 1's complement ↛ D MRQ/1 s/T8L (advance to PH12)	described separately CSX1/8 = PH11 (S/DCWCM) ← (BCON LOGIC) MRQ/1 = FAFLM. PH11 s/T8L = FAFLMD. PH11	} sign iteration set-up

FML, FMS

6 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH12 T8L	(sign iteration) A + D + CS31 → S ↛ A 0 ↛ D s/NGX s/G0003/1 if short. Rodd (causes "K71") s/LR31/2 s/T10L if not (short. Rodd)	SXADD = FAFLM. PH12, AXS = FAFLM. PH12 DX/1 = FAFL. PH12 s/NGX = FAFL. PH12 (s/G0003/1) = FAFL. PH12. NFPRD, reset by CLEAR s/LR31/2 = FAFL. PH12 s/T10L = FPRD. PH12	completes |product| } set up for negation and short truncation for S ↛ RWv1 if long or Reven
PH13 T6L or T10L	(1-2 clocks, both T6L if (short. Rodd), otherwise both T10L). (NOTE: product is an absolute value: $\frac{1}{256} \leq n \leq 1$, or = 0)		
	CASE 1: (FAFL. PH13. NFPRR): "RESULT NOT READY" FOR STORAGE; requires shift left or right (1 clock if req'd) A → S sustain PH13 " T10L if not (short. Rodd) " LR31/2 " NGX Bx2 ↛ B (insig)	SXA = (FAFL. PH13. NFPRR) BPH13 = (") s/T10L = ("). N(FPPN. A5255Z). FPRD s/LR31/2 = (") ← high except in certain FAFLAS cases s/NGX = (") BxBL1 = (")	(for FAFLAS)
	A47 ⇒ (FAFL. PH13. A47), (i.e. Sx 1/16 ↛ A (logical) E+1 ↛ E	|product| = 1, e.g. -1/16 x -1/16 results in -1 x -1, hence prod. = +1) AXSR4 = (FAFL. PH13. A47) PCTE1 = (")	(reduces to 1 ↛ AS1, i.e. 1/16) up-count exponent
	A4751Z . $\overline{RT2Z}$ ⇒ FPPN (i.e. Sx16 ↛ A B0003 ↛ A2831 E-1 → E 1 ↛ BC31 (insig)	|product| = $\frac{1}{256} \leq n < \frac{1}{16}$) AXSL4 = FPPN bit path provided by AXSL4. FAFLM. PH13 MCTE1 = FPPN s/BC31 = FPPN	down-count exponent (for FAFLAS)

FML, FMS

7 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH13	(Continued)		
	CASE 2: $\overline{A47}.(\overline{A4751Z}+RTZ) \Rightarrow$ FPRR: "RESULT READY" FOR STORAGE (i.e. |product| = $1/16 \leq$ n < 1, or = 0)		
	$\overline{A}+K \longrightarrow S$ if product negative $A \longrightarrow S$ " " positive $0 \longrightarrow S$ " result zero or exp. underflow	SXADD = FPRR.N(RTZ+FEUF.NFZ).FPR SXA = " .N(").NFPR with the trap flag (FZ) off (i.e., neither of above)	K = K31 due to NGX; also, if (short.Rodd), "K7i" is on for proper truncation; inhibit if trap (ref. PH12)
	S0031 → RW.i if not (short.Rodd)	RW = FPRR.FPRD.NFTRAP	
	S4871 → A0831 E+64 → A0107 (0 → A0)	AXAR32 = FPRR.N(FAFLD.02) AXE = FPRR ($\overline{E1}$ → A1, E0207 → A0207)	m.s. mantissa bits uninverted exponent plus bias
	1's → CS0007 if result negative	CSX1/S = FPRR.FPR (for sign insertion	(sign inserted next phase)
	S/DRQ	S/DRQ = FPRR	and exponent inversion)
	S/T8L	S/T8L = "	
	S/PH15	BRPH15 = "	
	Trap if result not zero and (exp. overflow + exp. underflow with trap flag (FZ) on):		
		FTRAP = FEOF + FEUF.FZ	
		FEOF = FAFL.NRTZ.NE0.E1	overflow: unbiased exp. > 63
		FEUF = " . " .E0.NE1	underflow: " " < -64
	S/TRAP	S/TRAP = (FPRR.FTRAP)	
	S/TR29 (for trap address 68_{10})	S/TR29 = (")	

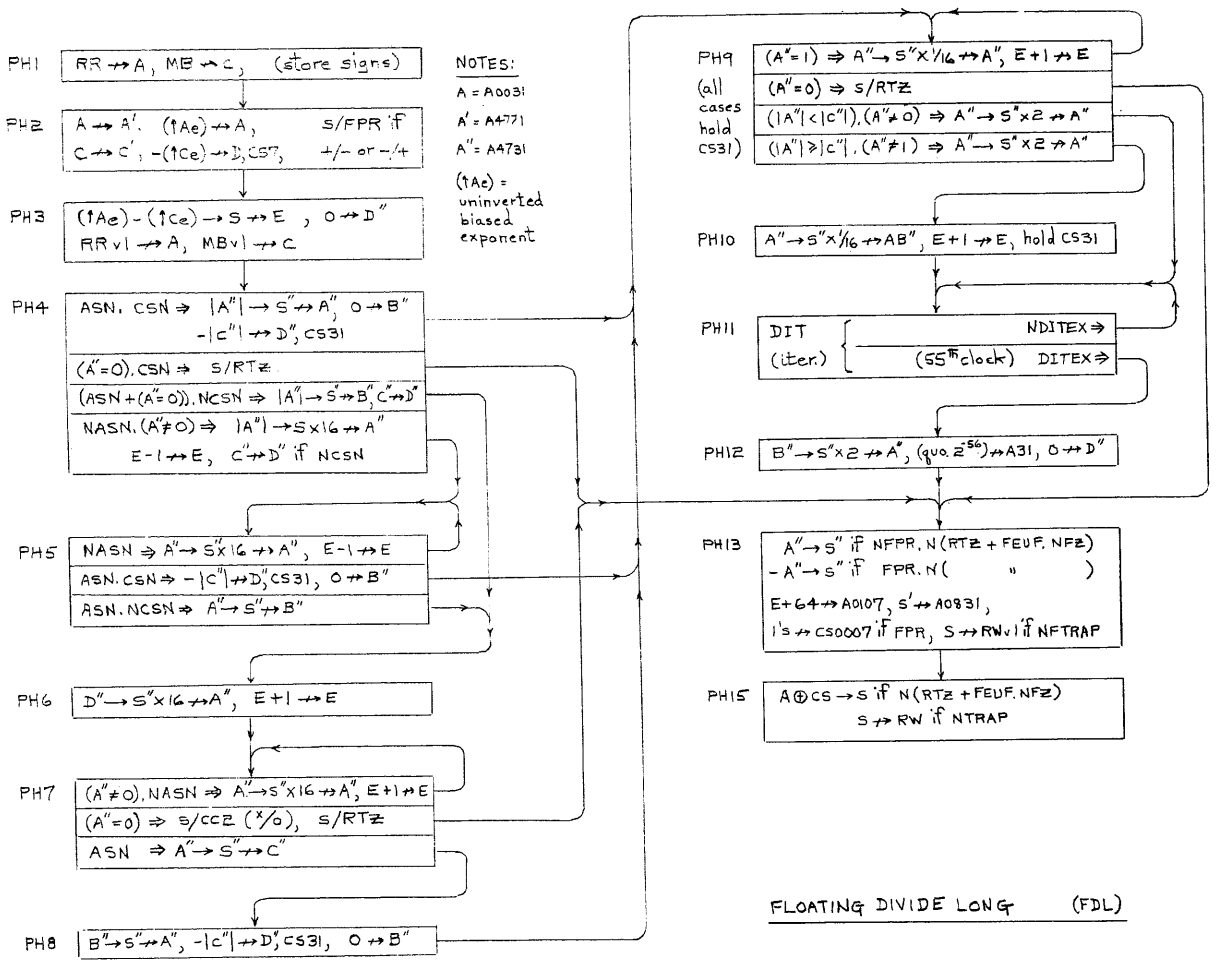
FML, FMS

8 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH15 T8L	ENDE	ENDE = FAMDSF.PH15	
	A ⊕ CS → S if result not zero S → RW if NTRAP	SXPR = FAFL.PH15.N(RTZ + FEUF.NFZ)	CS0007 are on if result neg. (inhibited by TRAP)
	for CC3, CC4 contr.: S → A; 1 → A31 if result not zero (merge); S/TESTA	RW = FAMDSF.PH15; RWDIS = TRAP.NINTRAPF AXS = FAFL.PH15 A31X1 = FAFL.PH15.N(RTZ + FEUF.NFZ) S/TESTA = FAMDSF.NFASHFX.PH15	(needed for sign bit) (needed for certain FAFLAS cases)
	1 → CC1 if underflow 1 → CC2 " " or overflow } and result ≠ 0	S/CC1 = FAFL.PH15.FEUF S/CC2 = FAFL.PH15.(FEUF + FEOF)	

FML, FMS

9 of 9



FLOATING DIVIDE LONG (FDL)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
T4RL PRE2. NIA:	RR → A (insig)	AXRR = FAMDSF. PRE2. NIA	(repeated significantly next phase)
PH1 T4RL	A → S → B (insig) RR → A (m.s.w. of numerator) RRO → RN (store sign of numer.) MB → C (m.s.w. of denominator) MBO → COC16 → MWN (store denom. sign) S/CXCL32 (for C0 → C47, C0831 → C4871 in PH2) R/CC1 (for exp. underflow test in PH15) R/CC2 (for " under/overflow " " " , or "Divide by zero test in PH7) 1's → CS0007 (to invert A0007 if A is neg.) S/IEN (start interruptability)	BXS = FAMDSF. PH1, SXA = FAFL. PH1 AXRR = RNXRRO/2. PH1 S/RN = RNXRRO. RRO; RNXRRO = RNXRRO/2. PH1 (R/RN = CLEAR) (by preparation control) S/MWN = FAMDSF. PH1. COC16, (R/MWN = CLEAR) S/CXCL32 = FAFL. PH1 R/CC1 = FAFL. PH1 R/CC2 = FAMDSF. NFAMULH. PH1 CSx1/S = FAFL. PH1 S/IEN = FAMDSF. NFAMUL. PH1	(for long FAFLM) RNXRRO/2 = FAFL exp. bits are inverted when neg. no.
PH2 T6L	A0 → A47, A0831 → A4871 (mantissa) C0 → C47, C0831 → C4871 (") Exponent differencing set-up (numer. exp - numer: A ⊕ CS0007 → S + " : A → S S → A0007 - denom: C → D + " : C̄ → D 1 → CS7 (for 2's complement) S → B (insig)	AXAL32 = FAFL. PH2 CXCL32 (F.F. set in PH2) minus denom. exp.): SXFR = (FAFL. PH2). RN SXA = ("). NRN AXS/1 = (") DXC/6 = ("), (MWN ⊕ FAFLM) ← MWN DXNC = ("). N(") ← NMWN CS7X1 = ("). NFAFLM BXS = ("). N(FAFLM. N02)	m.s.w. of numer. → A extension " " denom. → C " uninverted numerator exp. → A0107 (0 → A0, A0831) inverted denominator exp → D, (only D0007 are signif.) "1" → CS0007 (For FAFLAS and FAFLM)

FDL (1E), FDS (3E) "FAFLD"

1 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 T&L	(continued)		
	Access of l.s.w. of operands (bits 0031)		
	s/LB31/1	s/LB31/1 = (FAFL.PH2)	causes MBv1 → C in PH3
	MRQ } if long } (denominator)	MRQ = ("), N02	" 0 → C " "
	s/DRQ } if long } (denominator)	s/DRQ = ("), N02	(for RRv1 → A in PH3 if long)
	s/CX/1 if short } (denominator)	s/CX/1 = ("), 02	(0 → A " " " short)
	s/LR31/2 (numerator)	s/LR31/2 = (")	algorithm generates |quotient|
	enable T&RL if long	T&RL = ("), N02	
	s/FPR if final quotient is to be negative	s/FPR = (s/FPR) = FAFLMD.PH2.(MWN ⊕ RN)	
PH3 T&L if short	A4771 → S4771 → B4771 (insig)	SXA/3 = BXS/1 = FAFL.PH3	(for long FAFLM)
	A + D + CS7 → S0007 → E0007 (numer. exp. − denom. exp)	SXK/1 = SXPR/1 = EXS = FAFL.PH3	unbiased exp. difference
	0 → P (clear iterations ctr.)	PX = (FAFL.PH3)	
T&RL if long	0 → A4771	AX/1 = (")	
	A4771 → S4771 → 4771 (regenerate)	AXS/4 = (FAFL.PH3.N(FAFLM.ASN)) ; SXA/3 =	FAFL.PH3
	RRv1 → A0031 if long (clears if short)	AXRR = ("), N02	l.s.w. of numerator
	1 → NGX if RN (for |A| → S in PH4)	S/NGX = ("), RN	
	0 → C0031 if short	see PH2	l.s.w. of denominator
	MBv1 → C0031 if long		
	0 → D	DX/1 = FAFLMD.PH3	(for |A| → S in PH4)
	S/T8L	S/T8L = (FAFL.PH3.N(FAFLM.ASN)).FAFLMD	
PH4 T8L	(purpose: process numerator − convert to absolute value, zero check, prenormalize if necessary, examine denom.)		
	CASE 1: (ASN+(A4731=0)).$\overline{CSN}$: (denom. of unknown magnitude)		
	s/PH6	BRPH6 = (FAFLD.PH4).NCSN.(ASN + A4731Z)	set-up for zero checking and prenormalizing denom.
	C → D	DXC/6 = (FAFLMD.PH4).NCSN	
	|A| → S	SXADD = (")	
	S → A (insig - replaced in PH6)	AXS = (FAFLD.PH4).ASN	
	S → B (save for PH8)	BXS = (").NCSN	

FDL, FDS

2 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T8L	(continued)		
	CASE 2: (A4731=0).CSN ⇒ (S/RTZ/1), which reduces to FAFLD.PH4.A4731Z.CSN. (0 ÷ non-zero denom.)		
	s/PH13 (next to end phase)	BRPH13 = (S/RTZ/1)	
	s/RTZ (result = 0 indicator)	s/RTZ = (")	(R/RTZ = CLEAR)
	MRQ/1 (go for next instruction)	MRQ/1 = (").FAFLMD	(Q → P, etc)
	R/IEN (terminate interruptability)	R/IEN = (")	(was set in PH1)
	s/LR31/2 (for S → RWv1 in long case)	s/LR31/2 = (")	(not needed if short)
	s/T10L if long	s/T10L = (").FPRD	(needed because S → RWv1)
	CASE 3: (A4731 ≠ 0)		
	$\overline{ASN}$: ⇒ FPRENR	FPRENR = FAFLMD.PH4.NASN	prenormalize "R" operand
	(numerator not simple-normalized (includes =0)) (i.e. $0 \le n < 1/16$, $-1/16 \le n < 0$)		
	|A| → SX16 → A	AXSL4 = FPRENR	|numer.| x 16 → A
	E−1 → E (adj. exp)	MCTE1 = "	(insignificant if A=0)
	up-count P1518 if A≠0 (insig.)	PCTP5 = PCTP5/3 = FPRENR.NA4731Z	(for FAFLM)
	(advance to PH5)		
	C → D if $\overline{CSN}$	DXC/6 = FAFLMD.PH4.NCSN	(for PH6)
	ASN: (numerator is simple-normalized) (i.e. $1/16 \le n < 1$, $-1 \le n < -1/16$)		
	|A| → S → A	AXS = FAFLD.PH4.ASN	|numer.| → A
	$\overline{CSN}$:		
	(see CASE 1)		
	CSN: ⇒ DPP	DPP = FAFLD.PH4.ASN.CSN	set-up for divide iterations
	DPP: C → D if MWN	DXC/6 = DPP.MWN	−|Denom| → D,CS
	DPP: $\overline{C}$ → D, 1 → CS31 if $\overline{MWN}$	DXNC/1 = DPP.NMWN	−|Denom| → D,CS
	DPP: 0 → B	BX/1 = DPP	Clear B for PH9, 10, 11
	DPP: S/PH9	BRPH9 = DPP	
	DPP: S/T8L	S/T8L = DPP	
	ALL CASES: |A| → S	SXADD = FAFLMD.PH4	(NGX was set in PH3 if RN)

FDL, FDS

3 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T6L	(entered from PH4 if $(\overline{ASN}.)(A\neq 0)$ (1-5 clocks if short, 1-13 clocks if long) CASE 1: $\overline{ASN} \Rightarrow$ FPRENR $A \rightarrow S \times 16 \twoheadrightarrow A$ $E-1 \twoheadrightarrow E$ (adj. exp.) sustain PH5 up-count PIS18 (insig.) CASE 2: ASN $(1/16 \leq A < 1)$ $\overline{CSN}$ (advance to PH6) $CSN \Rightarrow DPP$ s PH9, etc. (descr. in PH4 - CASE 3) BOTH CASES: $A \rightarrow S$ $S \twoheadrightarrow B$ if $\overline{CSN}$	FPRENR = FAFLMD.PH5'.NASN AXSL4 = (FPRENR) MCTE1 = (") BRPH5 = (").PH5' PCTP5 = PCTP5/3 = FPRENR.NA4731Z DPP = FAFLD.PH5'.ASN.CSN SXA = FAFL.PH5' BXS = FAFLD.PH5'.NCSN	continue prenormalizing "R" operand (A contains a non-zero pos. no.) for FAFLM (to simple normalize denom.) (save normalized |numer.| for return to A in PH8)
PH6 T6L	(entered only if $\overline{CSN}$ - from PH4 or PH5. $D \rightarrow S$ FPRENM causes: $S \times 16 \twoheadrightarrow A$ $E+1 \twoheadrightarrow E$ S/CXS } insig if D = 0 S/T8L (advance to PH7)	D contains the denominator where $0 \leq d < 1/16$ SXD = (FAFLMD.PH6) FPRENM = (").N(PH7.ASN) = FAFLMD.PH6 AXSL4 = (FPRENM) PCTE1 = (").FAFLD S/CXS = (").NA4731Z + (S/CXS/1) S/T8L = (S/CXS/1) = FAFLMD.PH6	or $-1/16 \leq d < 0$) (C ↠ D in PH4) prenormalize "M" operand denom. x 16 ↠ A adjust exp. (for A → S ↠ C ↠ D in PH7)

FDL, FDS

4 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7 T6L / T8L ↯ T8L on 1st clock only)	(entered from PH6 only) (1-6 clocks if short, 1-14 clocks if long; maximum applies only to "all 1's" denom., which becomes "-1") CASE 1: (A = 0) (DIVIDE BY ZERO) $1 \twoheadrightarrow CC2$ (n ÷ 0 overflow) s/PH13, RTZ, etc. (descr. in PH4 - CASE 2) CASE 2: (A ≠ 0) $\overline{ASN} \Rightarrow$ FPRENM: $A \rightarrow S \times 16 \twoheadrightarrow A$ $E+1 \twoheadrightarrow E$ } insig if A = 0 S/CXS sustain PH7 } only if A ≠ 0 ASN: (advance to PH8) BOTH CASES: $A \rightarrow S \twoheadrightarrow C$	S/CC2 = FAFLD.PH7.A4731Z (S/RTZ/1) reduces to FAFLMD.PH7.A4731Z FPRENM = FAFLMD.PH7.N(PH7.ASN) = FAFLMD.PH7.NASN AXSL4 = (FPRENM), SXA = FAFLMD.PH7 PCTE1 = (").FAFLD S/CXS = (").NA4731Z BRPH7 = (").NA4731Z CXS is on throughout PH7	(CC's will = 0100) (set next to last Phase) continue prenorm. denom. (1-5 clocks if short 1-13 clocks if long) last clock puts simple-normalized denom. in C
PH8 T6L	(entered from PH7 only) $B \rightarrow S \twoheadrightarrow A$ $DPP \Rightarrow$ s/PH9, etc. (desr. in PH4, CASE 3)	SXB = AXS = FAFLD.PH8 DPP reduces to FAFLD.PH8	retrieve normalized |numer.| (ASN is high)
PH9 T8L or T6L	(entered by DPP from PH4, 5, or 8) (1-2 clocks; 2 clocks only if |numer| = +1, in which case 2nd CLOCK is T6L) NOTE: A contains |numer.|, which is $1/16 \leq n \leq 1$ or $n = 0$ (initial condition in this phase) D contains -|denom.|, where |denom| is $1/16 \leq d \leq 1$ hence, $\|A\| \div \|D\|$ could produce a |quotient| in the range $1/16 \leq q \leq 16$ if $n \neq 0$, PH9 and 10 scale the numerator to assure a quotient in the range $1/16 \leq \|q\| < 1$ (i.e. normalized)		

FDL, FDS

5 of 9

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9	(continued)		
	CASE 1: A47 (means A = +1, hence	$1 \le \|q\| \le 16$) (possible only if orig. numer = -1 or -1/16	T&L timing
	A → Sx 1/16 → A (produces + 1/16)	AXSR4 = (FAFLD.PH9.A47)	assures $1/16 \le q \le 1$
	E+1 → E (adj. exp.)	PCTE1 = (")	
	sustain PH9	BRPH9 = (")	(T&L not sustained)
	(advance to CASE 2)		
	CASE 2: $\overline{A47}$ (means $1/16 \le A < 1$ or A = 0, hence $1/16 \le \|q\| < 16$)		
	(TIMING: T8L if not entered from CASE 2. T&L when entered from CASE 2; T&L is suffient since K46 is fast enough to gate functions, as the only bits to change on the adder are A4751.)		
	A → Sx2 → A (scale for 2^{-1} iteration)	AXSL4 = FAFLD.PH9.NA47	A31 ← B0, where B0 = 0
	if (n ≥ d).(n ≠ 1): ($1 \le \|q\| < 16$)		
	(advance to PH10)		to shift numerator right 1 hex
	if (n < d).(n ≠ 0): ($1/16 \le \|q\| < 1$)		
	s/PH11, s/T8L } start iterations	BRPH11 = (FAFLD.PH9.NK46.NA4731≠)	$\overline{A47}$ implied by $\overline{K46}$ since D47=1
		s/T8L = (")	(NK46 is 4 levels old in T&L case; 6 levels if T8L)
	if (n = 0): (q = 0)		
	s/PH13, s/RT≠, etc. (described in PH4, CASE 2.)	(s/RT≠/1) reduces to FAFLD.PH9.A4731≠	
	ALL CASES:		
	A → S	SXA = (FAFLD.PH9)	
	hold CS31 (for 2^{-1} iteration)	R/CS31 = N(")	
PH10 T&L	(entered only if n ≥ d in PH9, CASE 2.)	($1 \le q < 16$)	
	A → Sx 1/16 → A	AXSR4 = (FAFLD.PH10), SXA = FAMDSF, PH10 gated by (")	assures $1/16 \le q < 1$
	S2830 → B0002	PCTE1 = (")	B = 0, S31 = 0
	E+1 → E (adj. exp.)		

FDL, FDS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH10 T&L	(continued)		
	hold CS31 (for 2^{-1} iteration)	R/CS31 = N(FAFLD.PH10)	
PH11 T8L	"DIT" Divide iterations (23 clocks if short, 55 if long) (2^{-1} iteration through next to last iteration; (2^0=0))		
	(initial conditions: A contains 2x|adjusted numer.|, (D,CS) contain -|denom.|; also $1/8 \le \|c\| \le \|A\| < 2\|c\|$)		
	CONTROL SIGNALS:		
	DIT (divide iterations)	FAMDSF/D.PH11	
	DITEX (last clock of DIT)	DITEX = (FAFLD).02.(P27.P29.P30) + (").P26.(")	short: P=22 (23rd clock) long: P=54 (55th ")
	CONTROL FUNCTIONS:		
	P+1 → P iterations ctr. } if $\overline{DITEX}$	PCTP1 = (DIT.NDITEX)	P was cleared in PH3
	sustain PH9	BRPH9 = (")	
	" T8L	S/T8L = FAFLD.PH11	
	(advance to PH12) } if DITEX		
	MRQ/1 (Q → P, etc.	MRQ/1 = (DIT.DITEX)	
	R/IEN (stop interruptability)	RIEN = (")	
	REGISTER CONTROL:		
	A+D+CS31 → Sx2 → A; B0 → A31 until P26=1 } residue	SXADD = DIT; AXSL1 = DIT.N(FADIV.P26) = DIT	residue x 2 → A
		S/A31 = AXSL1.A31EN/2 ← B0, FAMDSF/1 ← FAMDSF.N(FAFLD.(PH11.P26+PH12))	
	Bx2 → B; K46 → B31 } quotient	BXBL1 = DIT; S/B31 = K46.FAFLD.PH11	|quotient| K46 means pos. residue
	C → D if (MWN⊕K46); $\overline{C}$ → D if (MWN⊕K46); 1 → CS31 } ± denom.	DXC/D reduces to (FAFLD.PH11).N(MWN⊕K46); DXNC/D " " (").("); S/CS31 = DXNC/D	polarity of denom. clocked into D,CS is opposite to residue clocked into A.
	MWN.$\overline{K46}$ + $\overline{MWN}$.K46 = (C-).(S-) + (C+).(S+)		

FDL, FDS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH12 T8L	(final iteration) {B → SX2 → A K46 → A31} completes |quotient| {0 → D s/NGX s/G0003/1 if short (insig)} set up for negation s/LR31/2 (for S → RWv1 if long) s/T10L if long	AXSL1 = SXA = FAFLD.PH12 S/A31 = AXSL1.A31EN/2 ← K46.FAFLD.PH12 DX/1 = (FAFL.PH12) s/NGX = (") (s/G0003/1) = (").NFPRD s/LR31/2 = (") s/T10L = FPRD.PH12	2^{-24} if short, 2^{-56} if long (quotient is "right aligned") "FLOATING POINT RESULT DOUBLE"
PH13 T10L if long, T6L if short	(next to last phase. 1 clock.) NOTE: the |quotient| is in A0831 if short or in A4831 if long. The most significant (RTZ is on if either operand = 0) FPRR is high: "FL. PT. RESULT READY" {$\bar{A}$ + K → S if quotient neg. A → S " " pos. 0 → S if underflow with FZ=0, or if RTZ S → RWv1 if long (and not trap) {S4871 → A0831 if long S0871 → A0831 if short E+64 → A0107 (0 → A0) 1's → CS0007 if result neg. s/DRQ s/T8L s/PH15	FPRR = NA47.(NA4751Z + FAFLD.02 + RTZ) SXADD = (FPRR).N(RTZ + FEUF.NFZ).FPR SXA = (").N(").NFPR RW = (").FPRD.NFTRAP AXSR32 = ( ).N(FAFLD.02) AXS/3 = FAFLD.PH13.02 AXE = (FPRR), (i.e. $\overline{ET}$ → A1, E0207 → A0207) csx1/5 = (").FPR (for sign insertion and exponent inversion) s/DRQ = (") s/T8L = (") BRPH15 = (")	hex. is non-zero (unless RTZ). (reduces to FAFLD.PH13). {K = K31 due to NGX, ("K71" will be high insignificantly in the short case since S4707 are ignored). m.s. bits of quotient, entire quotient, uninverted exp. plus bias,

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH13	(continued) TRAP LOGIC: {overflow (unconditional) underflow (if FZ=1) divide by zero (unconditional) (inhibit S → RWv1) {s/TRAP s/TR29} trap to 68_{10}	FTRAP = + FEOF + FEUF.FZ + CC2 RW = FPRR.FPRD.NFTRAP s/TRAP = (FPRR.FTRAP) s/TR29 = (")	FEOF = (FAFL.NRTZ).NE0.E1 FEUF = (").E0.NE1 see s/CC2 in PH7
PH15 T8L	ENDE A ⊕ CS → S if not (quo = 0 or div by 0) S → RW if NTRAP for CC3,4 contr. {S → A 1 → A31 (merge) if result ≠ 0 s/TESTA 1 → CC1 if exp. underflow 1 → CC2 " " under/overflow	ENDE = FAMDSF.PH15 SXPR = FAFL.PH15.N(RTZ + FEUF.NFZ) RW = FAMDSF.PH15 ; RWDIS = TRAP.NINTRAPF AXS = (FAFL.PH15) A31X1 = (").N(RTZ + FEUF.NFZ) s/TESTA = FAMDSF.NFASHFX.PH15 s/CC1 = (FAFL.PH15.FEUF) s/CC2 = (") + (FAFL.PH15.FEOF)	CS0007 are on if neg. result, (inhibited by TRAP.) (insig. - needed for long FAFLAS) (inhibited if div. by zero) CC2 will be on if div. by zero

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Set DRQ	S/DRQ = FABØ/1 . EXU	Sets DRQ on every clock from PH1 thru PH13.
	Generate PROTECTDIS	PROTECTDIS = FABØ	PROTECTDIS while PHA is off
PH1	C12-31 → D12-31 , C12 → D0-11 R → LR (R≠0) ⇒ RR → A	DXC/4 = FABS . PH1 R→LR occurs unless preset otherwise AXRR = FABS. PH1. NRZ	(extend sign of displacement)
T4RL	0 → A	AX/1 = FABS. PH1 S/LR31/2 = FABS. PH1 S/TGRL = FABØ/1. PH1	for Rul → LR in PH2
PH2	A+D → S → B	SXADD = FABØ/2 . PH2 BXS = FABØ . PH2	
	Rul → LR	Preset in PH1	
T6RL	RR → A	AXRR = FABØ/2 . PH2 S/LR31/2 = FABØ . PH2 S/T10L/1 = FABØ. PH2	for Rul → LR in PH3
PH3	Rul → LR D → S → RW	Preset in PH2 SXD = FABØ/2 . PH3 RW = FABØ . PH3	(save displacement)
T10L	0 → D CBS ⇒ 0 → CC3 0 → CC4 Branch to Phase 5	DX/1 = FABS . PH3 R/CC3 = FABS. PH3. NØ7 R/CC4 = FABØ/1 . PH3. NØ7 BRPH5 = FABØ/1 . PH3 S/T8L = FABØ/1 . PH3	

MBS (61), CBS (60)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5	Shift B right 2 (B30 → Bo , B31 → B1) Shift A right 2 via PR (PR30 → Ao, PR31 → A1) A → S → E	BXBR2 = FABS/PH5 AXPRR2 = FABS/PH5 SXA = FABØ/2 . PH5 EXS = FABØ/2 . PH5	These circular shifts save byte addressing bits in positions 0 & 1.
T8L	(Ao-7 ≠ 0) ∩ NINT ⇒ Set Word Mode	S/SW1 = FABS . PH5. NA0007Z . NINT	SW1 used for Word Mode flip-flop
	(Ao-7 = 0) ∪ INT ⇒ Go to PH6A	S/PHA = FABS. PH5. A0007Z + FABS. PH5. INT S/T8L = FABØ/1. PH5	
PH6	B → S → P, (So → P32 S1 → P33)	SXB = FABS . PH6 PXS/1 = FABØ/2 . PH6	
T8L	MBS. R≠0. NBO. NBI. NAO. NAI. E>3 ⇒ S/WM	S/SW1 = (FABS. Ø7). PH6. (NBO. NBI. NAO. NAI) . NRZ. NEØ5Z.	Set Word Mode indicator.
	WM ∪ R≠0 ∪ CBS ⇒ Set Request	MRQ = FABS. PH6. SW1 + FABS, PH6. NRZ + FABS. PH6. NØ7	
	Reset Word Mode indicator (WM)	R/SW1 = FABS . PH6	
PH7	P → LB , MB → C WM ⇒ C → D	Request Set in PH6 DXC/6 = FABS. PH7. SW1	Word → D (MBS only)
	NWM ⇒ C → D24-31 (Down. Align.)	DXCBP = FABS. PH7. NSW1	Byte → D (Byte selected by P32,33).
	WM ⇒ P+1 → P R≠0 . NWM ⇒ P+¼ → P	PCTP1 = FABS. PH7. SW1 PCTP1 = FABS. PH7. NRZ PA33 = FABS. PH7. NRZ	Count P for next word. Count P for next byte.
	Protect Fail ⇒ Set PF (flip-flop SW2)	S/SW2 = FABØ/2. PH7. PROTECTD	Remember Memory Protect failure (will be used in PH13 to cause branch to exit sequence).
Data Rel.		S/T4L = FABØ/1 . PH7	

MBS, CBS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH8 T4L	P → S → B (P32 → S0, P33 → S1)	SXP = FABS. PH8 BXS = FABS. PH8 S/T4L = FABO/1. PH8	
PH9 T4L	A → S → P S0 → P32, S1 → P33 0 → A CBS ⇒ S/NGX for −D → S NPF ⇒ Set Request MBS · NWM ⇒ Prepare for Up. Align.	SXA = FABO/1 . PH9 PXS/1 = FABO/1. PH9 AX/1 = FABS . PH9 (S/NGX) = FABS. PH9. NO7 MRQ = FABS . PH9 . NSW2 S/NPRX = FABS. 07. NSW1. PH9	Preset for PH10 Preset for PH10
PH10 T6L Data Rel	P → LB MBS. WM ⇒ D → S → MB CBS ⇒ MB → C, C → D24-31 WM ∩ NPF ∩ NPROTECT FAIL ⇒ E−4 → E, P+1 → P MBS. NWM. NPF. N(PROTECT FAIL) ⇒ E−1 → E, ⇒ P+¼ → P CBS ⇒ −D → S → A Protect Fail ⇒ Set PF MBS ⇒ Branch To Phase 13	SXD = FABS. 07. PH10 . SW1 MW = FABS, 07. PH10 . SW1 SXUAB = FABS. 07 . PH10. NSW1 MWB = FABS. 07. PH10. NSW1 DXCBP = 06. 0L0. PH10. NPHA ES4 = FABS/10 . SW1 PA33 = FABS/10 . NSW1 MCTE1 = FABS/10 PCTP1 = FABS/10 FABS/10 = FABS. PH10, NSW2. NPROTECTD. 07 AXS = FABS. PH10 . NO7 S/SW2 = FABO . PH10. PROTECTD BRPH13 = FABS, 07. PH10	Write Word Byte selected by P32,33 Write Byte Down Align., byte selected by P32,33 Count E and P for Word Mode or Byte Mode. (MBS only) AX/1 and S/NGX are Preset.
PH11 T6L	A+D → S → A	SXADD = FABS . PH11 AXS = FABS . PH11 S/T8L = FABS . PH11	

MBS, CBS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH12 T8L	NPF. A0 ⇒ Set CC3 NPF. NA0 . (A≠0) ⇒ Set CC4 NPF. (A=0) ⇒ E−1 → E ⇒ P+¼ → P	(S/CC3/1) = FABS. PH12 . NSW2 . A0 S/CC4 = FABS. PH12. NSW2. NA0. NA00312 MCTE1 = FABS. PH12. NSW2 . A00312 PCTP1 = FABS . PH12. NSW2 . A00312 PA33 = FABS . PH12. NSW2. A00312	Count for each byte (CBS only)
PH13 T6L	P → S → A, (P32 → S0, P33 → S1) PF ∪ INT ∪ A≠0 ∪ E=0 ⇒ Go to PH6A N(PF ∪ INT ∪ A≠0 ∪ E=0) ⇒ Go to PH6	SXP = FABS . PH13 AXS = FABS . PH13 BRPH6 = FABS . PH13 S/PHA = FABS. PH13. (SW2+INT+NA00132+E2) S/T8L = FABS . PH13	
PH6A T8L	A → S, S X2 → A0-30, S0 → A31 B X2 → B0-30, B0 → B31	SXA = FABSA . PH6 AXSL1 = FABSA . PH6 BXBL1 = FABSA . PH6 S/LR31/2 = FABSA . PH6 S/CXRR = FABSA . PH6 S/T10L = (S/CXRR)	Preset for PH7A
PH7A T10L	A → S S X2 → A0-30, S0 → A31 E → A0-7 R01 → LR RR → C	SXA = FABSA . PH7 AXSL1 = FABSA . PH7 AXE = FABSA . PH7 LR31/2, CXRR } Preset in PH6A S/LR31/2 = FABSA . PH7 S/T8L = FABSA . PH7	Preset for PH8A
PH8A T8L	A → S → RW R01 → LR	SXA = FABOA/2 . PH8 RW = FABOA/2 . PH8 LR31/2 set in phase 7A S/T8L = FABOA/2 . PH8	

MBS, CBS

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9A T8L	B → S SX2 → A0-30 S0 → A31 NC → D WM ∪ NPF.EZ ⇒ 1 → CS31 Set Request (for next instruction)	SXB = FABOA/2 . PH9 AXSL1 = FABOA . PH9 DXNC = FABSA . PH9 CSX1/8 = FABSA.PH9. NSW2. N(N07.CC3 + N07.CC4) MRQ/1 = FABOA/1 . PH9	also causes Q → P
PH10A T6L	A + D + CS → S S → A 0 → D WM ∩ PF ⇒ 1's → CS0-29 Set DRQ Set Interruptible	SXADD = FABOA/2 . PH10 AXS = FABSA . PH10 DX/1 = FABSA . PH10 CSX1/1 = FABSA . PH10 . SW1 . SW2 S/DRQ = FABOA . PH10 S/IEN = FABOA . PH10 S/T10L = FABOA/2 . PH2	
PH11A T10L Data Rel	R → LR NR31 . (R≠0) ⇒ A+CS → S → RW CBS . (CC3 ∪ CC4) ∪ (E=0) ⇒ ENDE PF ⇒ Set trap PF ⇒ Set TRACC4	always done unless set otherwise SXADD = FABSA . PH11 . NR31 . NRZ RW = FABSA . PH11 . NR31 . NRZ ENDE = FABSA . PH11 . N07 . CC3 + FABOA/1 . PH11 . N07 . CC4 + FABOA . PH11 . EZ S/TRAP = (S/TRACC4/1) (S/TRACC4/1) = FABOA . PH11 . SW2 . NTRAP	

MBS, CBS

5 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Set DRQ for all NPHA Phases Generate PROTECTDIS for all NPHA Phases Preset LR31/2 for PH1	S/DRQ = FABO/1 . EXU PROTECTDIS = FABO S/LR31/2 = FATR . (PRE2 . NIA)	
PH1 T4RL	C12-31 → D12-31, 0's → D0-11 R ∪ 1 → LR RR → A	DXC/3 = FATR . PH1 LR31/2 set in PRE2 AXRR = FATR . PH1 T6RL = FABO/1 . PH1	
PH2 T6RL	A → S S → E S → B (R≠0) ⇒ RR → A R → LR 0 → A (R=0) ⇒ 1's → CS0-7 Set LR31/2 for PH3	SXA = FATR . PH2 EXS = FATR . PH2 BXS = FABO . PH2 AXRR = FATR . PH2 . NRZ normally occurs unless preset otherwise AX/1 = FATR . PH2 CSX1/5 = FATR . PH2 . RZ S/LR31/2 = FABO . PH2 S/T10L = FABO . PH2	
PH3 T10L	R ∪ 1 → LR A + D + CS → S S → RW B + 4 → B (B30 → B0, B31 → B1) Set FP (First Pass Flip-flop) Reset PF (Protect Fail Flip-flop) TTBS ⇒ 0 → CC4 Branch to PH5	LR31/2 set in PH2 SXADD = FATR . PH3 RW = FATR . PH3 BXBR2 = FATR . PH3 S/SW1 = FATR . PH3 R/SW2 occured at ENDE (via CLEAR) R/CC4 = FABO/1 . PH3 . N07 BRPH5 = FABO/1 . PH3 S/T8L = FABO/1 . PH3	Using SW1 for FP Using SW2 for PF

TBS(41), TTBS(40)

1 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T8L	FP ⇒ B → S S ↠ P, (S0 → P32, S1 → P33) Reset FP PF ∪ Protect Fail ∪ (E=0) ∪ INT ∪ CC4 ⇒ Branch to PH9A N[PF ∪ Protect Fail ∪ (E=0) ∪ INT ∪ CC4] ⇒ Set Request P → LB Not FP ⇒ D24-31 → K → S0-7, etc ⇒ S → MB Protect Fail ⇒ Set PF NFP ∩ N(PF ∪ PROTECT Fail ∪ CC4) ⇒ P+¼ ↠ P Set LR31/2 for PH6	SXB = FATR5.SW1 PXS/1 = FATR5.SW1 R/SW1 = FATR.PH5 S/PHA = FATR5.(SW2+PROTECTD+CC4+EZ+INT) BRPH9 = FATR5.(SW2+PROTECTD+CC4+EZ+INT) MRQ = FATR5.N(SW2+PROTECTD+CC4+EZ+INT) Normally occurs unless set otherwise SXUAB = FATR.NSW1.PH5 MWB = FATR.NSW1.PH5 S/SW2 = FATR.PH5.PROTECTD PCTP1 = FATR5.NSW1.N(SW2+PROTECTD+CC4) PA33 = FATR5.NSW1.N(SW2+PROTECTD+CC4) S/LR31/2 = FABO.NFABS.PH5 S/T8L = FABO/1.PH5	Upward Alignment. Byte selected by P32-33
PH6 T8L Data Rel	P → LB MB → C C ↠ D24-31 (Down Align) P → S S ↠ B, (P32 → S0, P33 → S1) R∪1 → LR RR ↠ A Protect Fail ⇒ Set PF	Normally occurs unless set otherwise Request made in PH5 DXCBP = FATR.PH6 SXP = FABO/1.PH6 BXS = FATR.PH6 Normally occurs unless set otherwise AXRR = FATR.PH6 S/SW2 = FATR.PH6.PROTECTD	Byte selected by P32-33
PH7 T6L	A+D → S S ↠ A 0's → D	SXADD = FATR.PH7 AXS = FATR.PH7 DX/1 = FATR.PH7 T4L = FABO/1.PH7	

TBS, TTBS

2 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH8 T4L	A → PR, PR÷4 ↠ A, (PR30 → A0, PR31 → A1)	AXPRR2 = FATR.PH8 S/T4L = FABO/1.PH8	
PH9 T4L	A → S S ↠ P, (S0 → P32, S1 → P33) Set Request E-1 ↠ E	SXA = FABO/1.PH9 PXS/1 = FABO/1.PH9 MRQ = FATR.PH9 MCTE1 = FATR.PH9	
PH10 T6L Data Rel	P → LB MB → C C → D24-31 (Down Align) B → S S ↠ P, (S0 → P32, S1 → P33) 0 ↠ A PROTECT FAIL ⇒ Set PF TBS ∩ N(PF ∪ Protect Fail) ⇒ Request 1's ↠ CS, set NPRX TBS ⇒ Branch to Phase 5	Request made in PH9 DXCBP = FATR.PH10 SXB = FATR.PH10 PXS/1 = FATR.PH10 AX/1 = FATR.PH10 S/SW2 = FABO.PH10.PROTECTD MRQ = FATR.07.PH10.NPROTECTD.NSW2 S/NPRX = FATR.PH10 BRPH5 = FATR.07.PH10 S/T8L = FATR.07.PH10 S/LR31/2 = (0U4.0L0.NPHA).PH10 S/CXS = (0U4.0L0.NPHA).PH10 S/T10L = (0U4.0L0.NPHA).PH10	Byte selected by P32-33 for up align in PH5 or 11 } For TTBS only. Preset for PH11
PH11 T10L	D24-31 → K → S0-7 S → C ↠ D RR ↠ A R∪1 → LR 1's ↠ CS, set NPRX	SXUAB = FATR.PH11 DXC/6 = FATR.PH11 AXRR = FATR.PH11 LR31/2 set in PH10 S/NPRX = FATR.PH11 T4RL = FATR.PH11	CXS set in PH10 {Set for "AND" operation in Phase 12.

TBS, TTBS

3 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH12 T4RL	A∩D → S S → A	NPRX set in PH11 AXS = FATR . PH12 S/T8L = FATR . PH12	
PH13 T8L	NPF ∩ (A0-7 ≠0) ⇒ Set CC4 R → LR NPF ∩ NR31 ∩(R≠0) ∩ (A0-7≠0) ⇒ A → S ⇒ S → RW Branch to Phase 5	S/CC4 = (FATR . NSW2 . NA0007Z) . PH13 normally occurs unless set otherwise SXA = (FATR . NSW2 . NA0007Z) . NRZ . NR31 . PH13 RWB0 = (FATR . NSW2 . NA0007Z) . NRZ . NR31 . PH13 BRPH5 = FATR . PH13 (S/T8L/1) = FAB0/1 . PH13	Write byte 0 only
PH9A T8L	P → S Sx2 → A0-30, S0 → A31 Set Request, Q → P PF ∪ CC4 ⇒ E+1 → E	SXP = FATRA . PH9 AXSL1 = FAB0A . PH9 MRQ/1 = FAB0A . PH9 PCTE1 = FATRA . PH9 . SW2 + FATRA . PH9 . CC4	
PH10A T6L	A → S Sx2 → A0-30, S0 → A31 E → A0-7 Set Interruptible Set DRQ Set LR31/2 for PH11A	SXA = FATRA . PH10 AXSL1 = FATRA . PH10 AXE = FATRA . PH10 S/IEN = FATRA . PH10 S/DRQ = FATRA . PH10 S/LR31/2 = FATRA . PH10 S/T8L = FATRA . PH10	

TBS, TTBS

4 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH11A T8L Data Rel	Ru1 → LR A → S S → RW TTBS ∩ E=0 ⇒ 0 → CC4 PF ⇒ Set TRAP ⇒ Set TRACC4 TTBS ∩ CC4 ∪ E=0 ⇒ ENDE	LR31/2 set in PH10A SXA = FATRA . PH11 RW = FATRA . PH11 R/CC4 = FATRA . NO7 . PH11 . EZ (S/TRAP/1) = (S/TRACC4/1) (S/TRACC4/1) = FAB0A . PH11 . SW2 . NTRAP ENDE = FAB0A . PH11 . EZ = FAB0A/1 . PH11 . CC4 . NO7	

TBS, TTBS

5 of 5

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 T4RL	C12-31 → D12-31 0's → D0-11 R → LR (RR) → A Disable Memory protection until PHA is set.	} DXC/3 = FUEBS. PH1 AXRR = FUEBS. PH1 S/LR31/2 = FABØ/2 S/CXS = FABØ/2 S/T10L = FABØ/2 PROTECTDIS = FABØ	Displacement → D } Preset for PH2
PH2 T10L	CC1-4 → DU0-3 Ø4-7 → DU4-7 1 → DU8 1 → DU9 A+D → S S → B S → C (RR) → A Ru1 → LR	DUXCC = FUEBS. PH2 DUXØ = FUEBS. PH2 DUCLOCK = FUEBS. PH2 DUSTART = FUEBS. PH2 SXADD = FABØ/2. PH2 BXS = FABØ. PH2 CX3 preset in PH1 AXRR = FABØ/2. PH2 LR31/2 preset in PH1 S/LR31/2 = FABØ. PH2 S/T10L = FABØ. PH2	 } Preset for PH3
PH3 T10L	D → S S → RW Ru1 → LR Co-7 → D24-31 0 → D (insig)	SXD = FABØ/2. PH3 RW = FABØ. PH3 LR31/2 preset in PH2 DXCR24 = FUEBS. PH3 S/T10L = FUEBS. PH3 DX/1 = FABS. PH3	save displacement

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T10L	D → S S24-31 → DU0-7 1 → DU8 Reset PF (Protect Fail' ff) Shift B right 2 (circular) S/CXRR S/T10L 0 → D (so PR = A next phase)	SXD = FUEBS. PH4 DUXS = FUEBS. PH4 DUCLOCK = FUEBS. PH4 Done by CLEAR at previous ENDE BXBR2 = FUEBS. PH4 (S/CXRR) = FUEBS. PH4 S/T10L = (S/CXRR) S/LR31/2 = FUEBS. PH4 DX/1 = FUEBS. PH4	 Using SW2 for PF } Preset for PH5
PH5 T10L	(entered from PH4, PH14, or PH15) (A0-7 =0) ∪ PF ∪ INT ∪ DUT ⇒ To PH8A (A0-7 =0) ∪ PF ∪ INT ∪ DUT ⇒ Shift A right 2 (circular) A → S S → E (RR) → C Ru1 → LR	S/PHA = FUEBS.PH5.(SW2+SW3+A00072+INT) BRPH8 = " " " AXPRR2 = FUEBS. PH5. N(") SXA = FABØ/2. PH5 EXS = FABØ/2. PH5 CXRR } preset in PH4, 14, or 15 LR31/2 } S/LR31/2 = FABØ. NFABS. PH5 S/T10L = FUEBS. PH5	 Shift via PR Displacement → C For Ru1 → LR in PH8A
PH6 T10L	A → S S → P, S0 → P32, S1 → P33 Set Request Set DRQ	SXA = FUEBS. PH6 PXS/1 = FUEBS. PH6 MRQ = FUEBS. PH6 S/DRQ = FUEBS. PH6	
PH7 T6L Data Release	MB → C → D24-31 (Down Align) B → S S → P, S0 → P32, S1 → P33 PROTECTD → Set PF Set Request	DXCPB = FUEBS. PH7 SXB = FUEBS. PH7 PXS/1 = FUEBS. PH7 S/SW2 = FABØ/2. PH7. PROTECTD MRQ = FUEBS. PH7 S/T10L = FUEBS. PH7	Byte selected by P32,33 Using SW2 for PF ff

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH8 T10L Data Release	NDRQ ⇒ D → S S → DU0-7 1 → DUB NDRQ ⇒ Repeat Phase 8 1's → CS, Set NPRX DU10 ⇒ Set DRQ DRQ ⇒ A → S ⇒ S → P, S0 → P32, S1 → P33 ⇒ 0's → A ⇒ MB → C → D24-31 DRQ . DU12 ⇒ Set DBR DRQ . DU12 . PROTECTD ⇒ Set PF	SXD = FUEBS8.NDRQ DUXS = FUEBS8. NDRQ DUCLOCK = FUEBS . PH8 BRPH8 = FUEBS8. NDRQ S/NPRX = FUEBS . PH8 S/DRQ = FUEBS. PH8. DUEND SXA = FUEBS8. DRQ PXS/1 = FUEBS8. DRQ AX/1 = FUEBS8. DRQ DXCBP = FUEBS8. DRQ S/SW4 = FUEBS8. DRQ . DU12 S/SW2 = FUEBS8. DRQ. DU12. PROTECTD S/T10L = FUEBS. PH8 R/SW4 = FUEBS8	Byte of source pattern → DU DU10 is End Field Down Align (byte selected by P32, P33) SW4 used for Decimal Byte Request (DBR) ff.
PH9 T10L	Upward Align D24-31 → S0-7, 8-15, etc. NDRQ ⇒ S → DU0-7 1 → DUB NDRQ ⇒ Repeat PH9 1's → CS, Set NPRX DU10 ⇒ Set DRQ DRQ ⇒ DU0-7 → D24-31 DRQ.DBR. DU12 ⇒ Set DUT DRQ . NDU12 ⇒ Set Request ⇒ set DRQ	SXUAB = FUEBS . PH9 DUXS = FUEBS9. NDRQ DUCLOCK = FUEBS . PH9 BRPH9 = FUEBS9. NDRQ S/NPRX = FUEBS . PH9 S/DRQ = FUEBS9 . DUEND DXDU = FUEBS9 . DRQ S/SW3 = FUEBS9.DRQ. DU12. SW4 MRQ = FUEBS9. DRQ. NDU12 S/DRQ = FUEBS9 . DRQ . NDU12	Byte of Decimal No. → DU Edited byte → D SW3 used for Decimal unit trap (DUT) flipflop
PH10 T6L Data Release	Upward Align D24-31 → S0-7, 8-15, etc S → MB (byte) N(DUT ∪ PF ∪ PROTECTD) ⇒ P + 1/4 → P ⇒ E - 1 → E PROTECTD ⇒ Set PF	SXUAB = FUEBS . PH10 MWB = FUEBS. PH10 PCTP1 = FUEBS. PH10. N(SW2 + SW3 + PROTECTD) PA33 = " " " MCTE1 = " " " Set SW2 = FABO. PH10, PROTECTD	Write Byte selected by P32,33

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH11 T6L	P → S, P32 → S0, P33 → S1 S×2 → A0-30, S0 → A31	SXP = FUEBS . PH11 AXSL1 = FUEBS . PH11	
PH12 T6L	B → S S → P, S0 → P32, S1 → P33	SXB = FUEBS. PH12 PXS/1 = FUEBS. PH12	
PH13 T6L	P + 1/4 → P A → S S×2 → A8-30, S0 → A31 E → A0-7	PCTP1 = FUEBS. PH13 PA33 = " " SXA = " " AXSL1/1 = " " AXE = " " S/T10L = " "	
PH14 T10L	1 → DUB 0 → D N(PF ∪ DUT). DU0 → Set CC1 N(PF ∪ DUT). DU1 → Set CC2 N(PF ∪ DUT). DU2 ⇒ Set CC3 N(PF ∪ DUT). DU3 → Set CC4 P → S, P32 → S0, P33 → S1 N(PF ∪ DUT). NDU1 . (CC2 ∪ DBR) ⇒ S → B N(PF ∪ DUT). DU4 ⇒ 1's → CS (PF ∪ NDU5) ⇒ { Go PH5, S/CXRR } N(PF ∪ NDU5) ⇒ 0 → R addr. lines merge 1 with R addr. S/T10L	DUCLOCK = FUEBS. PH14 . NPH3 DX/1 = FUEBS. PH14 S/CC1 = (FUEBS14. N(SW2 + SW3)).DU0 S/CC2 = (").DU1 S/CC3 = (").DU2 S/CC4 = (").DU3 (R/CC) = FUEBS14. N(SW2 + SW3) SXP = FUEBS . PH14 BXS = FUEBS14. N(SW2 + SW3). NDU1. CC2 + FUEBS14. N(SW2+SW3). NDU1 . SW4 CSK1 = FUEBS14. N(SW2+SW3) . DU4 BRPH5 = FUEBS14. (NDU5 + SW2) S/CXRR = FUEBS. BRPH5 R/LRXR = FUEBS14. NBR S/LR31/2 = FUEBS14 S/T10L = (S/CXRR) + FUEBS14 . NBR	Sets register address to 1 for use in PH15 (red. to FUEBS14)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH15 T10L	1 → LR A + CS → S S → RW (bytes 1-3) Go to Phase 5	Preset in PH14 SXADD = FUEBS . PH15 RWB1 = " " RWB2 = " " RWB3 = " " BRPH5 = " " S/LR31/2 = " " (S/CXRR) = FUEBS . BRPH5 S/T10L = (S/CXRR)	} Preset for Phase 5
PH8A T10L	(entered from Phase 5) R ∪ 1 → LR A → S S → RW NC → D Shift B left 1 (circular)	LR31/2 set in Phase 5 SXA = FAB0A/2 . PH8 RW = FAB0A/2 . PH8 DXNC = FUEBSA . PH8 BXBL1 = FUEBSA . PH8 S/TBL = FAB0A/2 . PH8	(prepare to subtract displacement in PH10A)
PH9A TBL	B → S S x 2 → A0-30, S0 → A31 1 → CS31	SXB = FAB0A/2 . PH9 AXSL1 = FAB0A . PH9 CSX1/8 = FUEBSA . PH9 S/CXS = " " S/T10L = " "	for S → C in PH10A

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH10A T10L	A+D+CS → S S → C C → D 0 → A Set A9 E=0 ⇒ Set Memory Request Q → P Set Interruptible Set DRQ	SXADD = FUEBSA . PH10 CXS preset in PH9A DXC/6 = FUEBSA . PH10 AX/1 = " " A9X1 = " " MRQ = FUEBSA . PH10 . EZ PXQ = FUEBSA . PH10 . PXQ S/IEN = FAB0A . PH10 S/DRQ = FAB0A . PH10 S/T10L = FAB0A/2 . PH10	
PH11A T10L Data Release	A+D → S S → RW (bytes 1, 2 & 3) R → LR E=0 ⇒ ENDE PF ∪ DUT ⇒ Set TRAP DUT ⇒ Set TR29 ⇒ Set TR31 PF ⇒ Set TRACC4	SXADD = FUEBSA . PH11 RWB1 = FUEBSA . PH11 . NPREI . N(CROF . ATE) RWB2 = " " " " RWB3 = " " " " Normally occurs unless set otherwise ENDE = FAB0A . PH11 . EZ (S/TRAP/1) = (S/TR29/1) . NPH8 + (S/TRACC4/1) (S/TR29/1) = FUEBSA . PH11 . SW3 S/TR31 = (S/TR29/1) . N(S/TRACC4/1) (S/TRACC4/1) = FAB0A . PH11 . SW2 . NTRAP	SW3 used for Decimal Unit Trap ff and SW2 used for Protect Fail ff.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Set DRQ each execution phase CVA ⇒ Set LR31/2 in preparation	FACV = 0U2. (04, N05. 06) (S/DRQ) = FACV. EXU (S/LR31/2) = FACV. 02. (PRE2. NIA)	Opcode decoding signals
PH1 T4RL	RR → A CVS ⇒ 1 → E7 Set Request Go to Phase 3 CVA ⇒ Ru1 → LR (note: B reg. was cleared in PRE1)	NAXRR = FACV. PH1 E7X1 = FUCVS. PH1 MRQ = FUCVS. PH1 BRPH3 = FUCVS. PH1 Preset in (PRE2. NIA) phase	Number to be converted → A E counts repititions of PH3. R → LR if CVS
PH2 T6L	A → S → B (Only CVA goes thru this phase) 0's → A Reset CC1 Set CX/1	BXS = FACV. PH2 SXA = FACV. PH2 AX/1 = FACV. PH2 (R/CC1) = FACV. PH2 (S/CX/1) = FACV. PH2	
PH3 T10L	(CVS. E<32) ∪ (CVA · B0 · E<32) ⇒ Set Request CVA ⇒ C → D CVS ⇒ NC → D 1 → CS31 A + D + CS → S (CVA . B30) ∪ (CVS . K00) ⇒ S → A CVS . K00 ⇒ 1 → B31 CVA . B30. K00 ⇒ Set CC1 E + 1 → E Shift B left one (cyclic) (CVS + SW1) · (E ≠ 33) ⇒ P+1 → P PH3 continued on next page	MRQ = FUCVS. PH3. NE2 + FACV. PH3. NE2. B0 DXC/6 = FACV. PH3. 07 DXNC = FUCVS. PH3 CX1/8 = FUCVS. PH3 SXADD = FACV. PH3 AXS = FACV. PH3. 07. B30 + FUCVS. PH3. K00 B31X1 = FUCVS. PH3. K00 S/CC1 = FACV. PH3. 07. B30 . K00 PCTE1 = FACV. PH3 BXBL1 = FACV. PH3 PCTP1 = FACV. PH3. SW1. N(E2. E7) + FUCVS. PH3. N(E2. E7)	MB → C, P → LB, 0s → C for 1st Phase 3 of CVA. Add or Subtract. result to A. CC1 was cleared in PH2. Count iterations.

CVA (29), CVS (28)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3 (cont.) Data Release	set SW1 E ≠ 33 ⇒ Repeat PH3 E = 33 ⇒ Go to PH4 Set Request, Q → P	S/SW1 = FACV. PH3 BRPH3 = FACV. PH3. N(E2. E7) MRQ/1 = FACV. PH3. (E2. E7) S/T10L = FACV. PH3	
PH4 T10L	A → S → RW CVS ⇒ Set LR31/2 for Ru1 → LR in PH5	SXA = FACV. PH4 RW = FACV. PH4 (S/T8L) = FACV. PH4 (S/LR31/2 = FUCVS. PH4	R → LR. If CVS: Remainder → R If CVA: Converted Number → R
PH5 T8L Data Release	CVS ⇒ B → S S → RW S → A Set TESTA ENDE	SXB = FUCVS . PH5 RW = FUCVS . PH5 AXS = FUCVS . PH5 S/TESTA = FACV . PH5 ENDE = FACV . PH5	

CVA, CVS

2 of 2

FAST (FAST includes Stack Instructions)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS	
PH1 T4RL	CC → E LM ⇒ Go to PH3A Set PHA Set Request STM ⇒ Go to PH1A Set PHA Set Interruptible Set DRQ	EXCC = FAST. PH1. (04. N05. 06) BRPH3 = FAST. PH1. 0LA . 02 S/PHA = FAST. PH1. 02 MRQ = FAST. PH1. 02 . N07 BRPH1 = FAST. PH1. 0LB. 02 S/PHA = FAST. PH1. 02 S/IEN = FAST. PH1. (04. N05. 06) S/DRQ = FAST. EXU + FASTA. EXU	Word Count → E4 thru E7 Access 1st Word DRQ is set for all phases of execution	LM STM
PH1A T6L	C → D Reset Interruptible Set Request RR → A R+1 → R SXADD	DXC/6 = FASTA . PH1 R/IEN = FASTA . PH1 MRQ = FASTA . PH1 AXRR = FASTA . PH1. (04. N05. 07) PCTR = FASTA . PH1. 07 SXADD = FASTA. PH1 S/T8L = FASTA . PH1	not significant not significant	STM
PH2A T8L Data Rel	A → S → MB R+1 → R E-1 → E E ≠ 1 ⇒ Set Request ⇒ P+1 → P ⇒ RR → A ⇒ Repeat PH2A E = 1 ⇒ Go Phase 13A ⇒ Set Request	MW = FASTA . PH2 SXA = FASTA . PH2 PCTR = FASTA . PH2 MCTE1 = FASTA. PH2 MRQ = FASTA. PH2 . N(E=1) PCTP1 = FASTA. PH2 . N(E=1) AXRR = FASTA. PH2 . N(E=1). N0L9 BRPH2 = FASTA. PH2. N(E=1). N0L9 BRPH13 = FASTA. PH2 . (E=1). 02 MRQ/1 = FASTA. PH2. (E=1). 02 S/T8L = FASTA . PH2	P → LB, Write Word (E=1) = NE4. NE5. NE6. E7 E0 thru E3 are ignored. Q → P,	STM

LM (2A), STM (2B)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS	
PH3A T6L Data Rel	MB → C → D D → S → A (E ≠ 1) ⇒ P+1 → P ⇒ Set Request Reset Interruptible Go to Phase 5A	DXC/6 = FASTA . PH3 SXD = FASTA . PH3 AXS = FASTA . PH3 PCTP1 = FASTA . PH3 . N(E=1). 0LA MRQ = FASTA . PH3 . N(E=1). 0LA R/IEN = FASTA . PH3 BRPH5 = FASTA . PH3. 0LA	1st Word → D } not significant (E=1) = NE4. NE5. NE6. E7 E0 thru E3 are ignored.	LM
PH5A T6L Data Rel.	C → D D → S → RW R+1 → R E-1 → E E ≠ 1 ⇒ Repeat PH5A E ≠ 1 or 2 ⇒ Set Request ⇒ P+1 → P E = 1 ⇒ Set Request Go to PH13A	DXC/6 = FASTA . PH5 SXD = FASTA . PH5 RW = FASTA . PH5 PCTR = FASTA . PH5 MCTE1 = FASTA . PH5 BRPH5 = FASTA, PH5, 0LA . N(E=1) MRQ = FASTA . PH5. N(E=1 OR 2) . 0LA PCTP1 = FASTA . PH5. N(E=1 OR 2). 0LA MRQ/1 = FASTA . PH5. (E=1). N07. 02 BRPH13 = FASTA . PH5. 0LA. 02. (E=1)	Word → D Word → Register Q → P	LM
PH13A T6L Data Rel	END	ENDE = FASTA . PH13	P+1 → P etc.	LM STM

LM, STM

2 of 2

PHASE	PLW PSW PLM PSM MSP	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
			S/DRQ = FAST. EXU + FASTA. EXU	Set DRQ for all phases
PH1 T4RL		CC → E Set Interruptible CC → D28-D31 CC=0 ⇒ 1 → D27 RR16-31 → A16-31 1 → A31 S/NGX	EXCC= FAST. PH1. (04. N05. 06) S/IEN= FAST. PH1. (04. N05. 06) DXCC = FAST. PH1. (04. N05. 06) D27X1 = FAST. PH1. CC2. N02. 04. N05. 06 AXRR/2 = FAST. PH1. 0L3 A31X1/1 = FAST. PH1. N06 (S/NGX) = FAST. PH1. N02	Set number of words into least sig end of A or D Prepare for negation
PH2 T6L		-A or -D → B16-31 1's → CS and prepare for upward alignment	G1619/1 = FAST. PH2 BXS = FAST. PH2 S/NPRX = FAST. PH2	Hold -word count in B (G1619/1 causes 0's → B0-15 while presetting of NGX in PH1 causes the count to be negated
PH3 T6L		Upward Align Halfword S → A0-A15 Clear D	SXUAH = FAST. PH3 AXS/2 = FAST. PH3 DX/1 = FAST. PH3	This results in A16-31 or D16-31 → A0-15 0's → A16-31
PH4 T6L		B ∪ A → S → A Set NGX Set Request set LB31/1	SXB = FAST. PH4 SXA = FAST. PH4 AXS = FAST. PH4 (S/NGX) = FAST. PH4 MRQ = FAST. PH4 (S/LB31/1) = FAST. PH4	Combine Positive & Neg. halves of Modifier → A (Pos. in A0-15, Neg. in A16-31) Prepare to negate modifier for "pushes". Access the space/Word Count from memory

PLW (08), PSW (09), PLM (0A), PSM (0B), MSP (13) (STACK INSTRUCTIONS) 1 of 6

PHASE	PLW PSW PLM PSM MSP	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 T6L Data Rel		MB → C → D PU1 → LB -A → S → A for Pushes	DXC/6 = FAST. PH5 AXS = FAST. PH5. 07 G1619 = FAST. PH5 (S/T10L/1) = FAST. PH5	(EAU) → D Negate Hold K15=1
PH6 T10L		A+D+CS → S → B Prevent carry from Words to space field D16·$\overline{S16}$ ⇒ Set SW2 D0·S0 ⇒ Set SW1 $\overline{D16}$·S16 ∪ $\overline{D0}$·S0 ⇒ Set TRAP, TR30 D16·$\overline{S16}$ ∪ D0·$\overline{S0}$ ⇒ Set PHA Go to PH14A Set CXS for PH7 S/T8L	SXADD = FAST. PH6 K15 = 1. FAST. PH6 BXS = FAST. PH6 S/SW2 = FAST. PH6. D16. NS16 S/SW1 = FAST. PH6. D0. NS0 (S/TR30/1) = FAST. PH6 S/PHA = FAST. PH6. D16. NS16 BRPH14 = FAST. PH6. D16. NS16 S/PHA = FAST. PH6. D0. NS0 BRPH14 = FAST. PH6. D0. NS0 S/CXS = (S/CXS/1) = FAST. PH6 S/T8L = (S/CXS/1) = FAST. PH6	Modify Space/Word Cnt. and store in B. Word Over/Underflow Space Over/Underflow Push/Pull Overflow & Trap To Phase 14A to abort instruction if trap mask bit is on.
PH7 T8L		A → S → C → D16-D31 0's → A Set Request	SXA = FAST. PH7 DXC/2 = FAST. PH7 AX/1 = FAST. PH7 MRQ = FAST. PH7	Increment (or Decrement) into D16-31; sign in D16 Request for TSA

PLW, etc. 2 of 6

PHASE	PLW PSW PLM PSM MSP	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH8		MB → C → D	DXC/6 = FAST . PH8	TSA → D
		D → S → A	SXD = FAST . PH8	
T6L			AXS = FAST. PH8. (ØLA + ØL3)	Increment/Decrement → A
		Set CXS	S/CXS = FAST . PH8 . NØL3	Preset for PH9
Data Rel				
PH9		P → S → C	SXP = FAST . PH9	Hold EA in C
		Set PHA	S/PHA = FAST . PH9	
T6L		Go To PH1A	BRPH1 = FAST. PH9. NØL3	(MSP goes to PH10A)
		1 → CS1	CSX1/8 = FAST. PH9 . NØL8. NØL3	
		A16 → CS0-15	CSX1/3 = FAST. PH9. A16	for sign extension (Note A is clear except for PLM
Data Rel		set Request	MRQ = FAST . PH9. ØL3	
PH1A		C → D	DXC/6 = FASTA . PH1	(EA) → D
		Reset Interruptible	R/IEN = FASTA . PH1	
		A+D+CS → S → P	PXS = FASTA. PH1. NØ2	Modified TSA → P
T6L		Set Request	MRQ = FASTA . PH1.	
		RR → A	AXRR = FASTA . PH1. (Ø4. NØ5. Ø7)	
		R+1 → R	PCTR = FASTA. PH1. Ø7	
		Go to Phase 3A	BRPH3 = FASTA. PH1. (Ø4. NØ5. NØ7)	
			SXADD = FASTA . PH1	
			S/T8L = FASTA . PH1	

PLW, PSW, PLM, PSM, MSP

3 of 6

PHASE	PLW PSW PLM PSM MSP	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2A		A → S → MB	MW = FASTA . PH2	P → LB Write Word
			SXA = FASTA . PH2	
		R+1 → R	PCTR = FASTA . PH2	
		E-1 → E	MCTE = FASTA. PH2	
T8L		E≠1 ⇒ Set Request	MRQ = FASTA . PH2. N(E=1). ØLB	
		P+1 → P	PCTP1 = FASTA. PH2. N(E=1). ØLB	
		RR → A	AXRR = FASTA. PH2 . N(E=1) . NØL9	
		Repeat Phase 2A	BRPH2 = FASTA. PH2. N(E=1). NØL9	
		PSW ∪ [PSM. E=1] ⇒ Go to PH8A	BRPH8 = FASTA. PH2. (E=1). NØ2	
			+ FASTA . PH2. ØL9 . NØ2	
		⇒ 0 S → A	AX/1 = FASTA. PH2. (E=1) . NØ2	
			+ FASTA . PH2. ØL9	
		⇒ S/CXS	S/CXS = FASTA . PH2. (E=1). NØ2	
			+ FASTA . PH2. ØL9	
			S/T8L = FASTA . PH2	
Data Rel				
PH3A		MB → C → D	DXC/6 = FASTA . PH3	P → LB Read 1st Word
		D → S → A	SXD = FASTA . PH3	
			AXS = FASTA. PH3	EA → A
		Go to PH5A if PLM	BRPH5 = FASTA. PH3 . ØLA	
T8L		P+1 → P	PCTP1 = FASTA . PH3 . N(E=1). ØLA	Prepare to Read 2nd Word
		set Request	MRQ = FASTA . PH3 . N(E=1). ØLA	
		Reset Interruptible	R/IEN = FASTA . PH3	
		set CXS	S/CXS = FASTA . PH3 . ØL8	
Data Rel		set T8L	S/T8L = FASTA . PH3.	
PH4A		P → S → C	SXP = FASTA . PH4	TSA → C
T8L			S/T8L = FASTA . PH4	

PLW, etc.

4 of 6

PHASE	PLW PSW PLM PSM MS	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5A T8L Data Rel	• • •	C → D D → S → RW R+1 → R E−1 → E E≠1 ⇒ Repeat PH5A E≠1 or 2 ⇒ Set Request ⇒ P+1 → P	DXC/6 = FASTA . PH5 SXD = FASTA . PH5 RW = FASTA . PH5 PCTR = FASTA. PH5 MCTE1 = FASTA. PH5 BRPH5 = FASTA. PH5. GLA. N(E=1) MRQ = FASTA. PH5. N(E=1 OR 2). GLA PCTP1 = FASTA. PH5. N(E=1 OR 2). GLA	Word → D (PLM) TSA → D (PLW) Write Word in Register
PH6A T6L	• • •	A → S → P 1's → CS 0's → A CC → D28−31 Set Request CC=0 ⇒ 1 → D27 Set NGX Set CXS for S→C in PH7A set T8L	PXS = FASTA . PH6 SXA = FASTA . PH6 CSX1 = FASTA . PH6 AX/1 = FASTA . PH6 DXCC = FASTA . PH6 . GLA . NOZ MRQ = FASTA . PH6 . GLA D27X1 = FASTA . PH6. GLA . CCZ S/NGX = FASTA. PH6. GLA. NOZ S/CXS = FASTA. PH6 . GLB S/T8L = FASTA.PH6	EA → P
PH7A T8L Data Rel	• •	D−1 → S → C → D −D → S → A MB → C → D Set Request Go to Phase 10A	SXADD = FASTA . PH7 AXS = FASTA . PH7 . GLA DXC/6 = FASTA. PH7 MRQ = FASTA . PH7 BRPH10 = FASTA. PH7	New TSA → D P → LB
PH8A T8L	• • •	P → S → C	SXP = FASTA . PH8 CXS was preset in phase 2A	Final TSA → C

PLW, etc.

PHASE	PLW PSW PLM PSM MS	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9A T6L	• • •	D → S → P C → D Set Request	PXS = FASTA . PH9 SXD = FASTA . PH9 DXC/6 = FASTA . PH9 MRQ = FASTA. PH9	EA → P TSA → D
PH10A T6L	• • • • •	D+A → S → MB Set Request Reset Interruptible (for Pul → LB in PH11A)	MW = FASTA . PH10 SXADD = FASTA. PH10 MRQ = FASTA. PH10 R/IEN = FASTA. PH10 (S/LB31/1) = FASTA. PH10	P → LB, Write New TSA
PH11A T6L data Rel	• • • • •	B → S → MB Pul → LB S → A Set Request for next Instr.	MW = FASTA. PH11 SXB = FASTA. PH11 AXS = FASTA. PH11 MRQ/1 = FASTA. PH11	Count → EAul Pul → LB, Write Count → A for testing Request and Q → P
PH12A T6L Data Rel	• • • • •	SW1 → CC1 (A1−15 = 0) ⇒ Set CC2 SW2 → CC3 (A17−31 = 0) ⇒ Set CC4 END	S/CC1 = FASTA. PH12. SW1 S/CC2 = FASTA. PH12 . A0115Z S/CC3 = FASTA. PH12 . SW2 S/CC4 = FASTA. PH12. A1731Z (R/CC) = FASTA. PH12 ENDE = FASTA. PH12	
PH14A T8L	• • • • •	D → S → A Set Request Go to Phase 12A	SXD = FASTA . PH14 AXS = FASTA . PH14 MRQ/1 = FASTA . PH14 BRPH12 = FASTA . PH14	Request and Q → P

PLW, etc.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2	Set SW4 D12 → SW1 D13 → SW2 D14 → SW3	S/SW4 = FUMMC. PRE2 S/SW1 = FUMMC. NSW4. PRE2. D12 S/SW2 = FUMMC. NSW4. PRE2. D13 S/SW3 = FUMMC. NSW4. PRE2. D14	1st pass indicator Function code bits into SW1, 2, 3 on first PRE2. (2nd PRE2 occurs if indirect address)
PH1 T4RL	RR → A (R → LR) PRESET 1 → LR31 for PH2 0 → P32, P33	AXRR = FUMMC. PH1 S/LR31/2 = FUMMC. PH1 PX/1 = FUMMC. PH1 S/T4RL = FUMMC. PH1	
PH2 T4RL	A → S → P RR → A (RU1 → LR) Set Request Set Address Release	SXA = FUMMC. PH2 PXS = FUMMC. PH2 AXRR = FUMMC. PH2 MRQ = FUMMC. PH2 S/ARQ = FUMMC. PH2 S/T4RL = FUMMC. PH2	Image address → P
PH3 T4RL Addr. Rel.	A → S → E0-E7, P15-P31 RR → A (R → LR) Disable mapping Set Data Release	SXA = FUMMC. PH3 EXS = FUMMC. PH3 PXS = FUMMC. PH3 AXRR = FUMMC. PH3 S/MAPDIS = FUMMC. PH3 S/DRQ = FUMMC. PH3	

MMC (6F)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 T6L Data Rel. on 1st Clock	P byte ≠ 3 ⇒ +1 → P33 (SW1 = 1) ⇒ +1 → P22 (SW1 ≠ 1) ⇒ +1 → P20 C → D24-D31 P byte ≠ 0 ⇒ D24-D31 → Byte of Map D24-D31 → Byte of PCB D24-D31 → Byte of Locks N(P32, P33) ⇒ Repeat 4 (P32, P33) ⇒ 0 → P32, P33 Reset SW4	PA33 = FUMMC. PH4. N(P32. P33) PCTP1 = FUMMC. PH4. N(P32. P33) PA22 = N(FUMMC. PH4. NSW1) PCTP4/1 = FUMMC. PH4. NSW4 DXCBP = FUMMC. PH4 ← if SW1 ← if SW2 (see special equations at end.) ← if SW3 BRPH4 = FUMMC. PH4. N(P32. P33) PX/1 = FUMMC. P33 R/SW4 = FUMMC. PH4 S/T8L = FUMMC. PH4	Increment P byte addr. Increment Control Start Address at P20 or P22 depending on function. Downward Align Load 3 bytes here and 4th byte in phase 5. Nothing loaded on 1st clock of Phase 4 because data is read from memory then.
PH5 T6L	P → S → B D24-D31 → Byte of Map D24-D31 → Byte of PCB D24-D31 → Byte of Locks 0 → D 1 → CS31 E-1 → E	SXP = FUMMC. PH5 BXS = FUMMC. PH5 if SW1 if SW2 (see special equations at end.) if SW3 DX/1 = FUMMC. PH5 CSX1/8 = FUMMC. PH5 MCTE1 = FUMMC. PH5	Preserve Byte Address Load 4th byte

MMC

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6 T6L	A+D → S ↛ P S ↛ A Set Request if Count≠0 set Address Release Interlock	PXS = FUMMC . PH6 SXADD = FUMMC . PH6 AXS = FUMMC . PH6 MRQ = FUMMC . PH6 . NEZ S/ARQ = FUMMC . PH6 S/SW4 = FUMMC . PH6 . INT . NEZ	Increment address Access next word SW4 used here to Remember interrupt is pending.
PH7 T6L Addr Rel	B → S ↛ P Count≠0 and no Interrupt ⇒ Return to PH4 Set Data Release Interlock	SXB = FUMMC . PH7 PXS = FUMMC . PH7 BRPH4 = FUMMC . PH7 . NSW4 . NEZ S/DRQ = FUMMC . PH7 S/T8L = FUMMC . PH7	
PH8 T6L	A → S → RW (R on LR) Enable mapping	PA22 = FUMMC (PH4+PH8) . NSW1 PCTP4/1 = FUMMC . PH8 SXA = FUMMC . PH8 RW = FUMMC . PH8 R/MAPDIS = FUMMC . PH8	} Increment P at P20 or P22 depending on function Store new R

MMC

3 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9 T8L	Set Interruptible Preset 1 → LR31 E ↛ A P → S ↛ B Request next instruction Set Data Release Interlock	S/IEN = FUMMC . PH9 S/LR31/2 = FUMMC . PH9 AXE = FUMMC . PH9 SXP = FUMMC . PH9 BXS = FUMMC . PH9 MRQ/1 = FUMMC . PH9 . NSW4 S/DRQ = FUMMC . PH9	{ Not set if exiting due to interrupt.
PH10 T8L	B ∪ A → S ↛ RW (RU on LR) Generate ENDE if no interrupt.	SXB = FUMMC . PH10 SXA = FUMMC . PH10 RW = FUMMC . PH10 ENDE = (FUMMC . NSW4) . PH10	
	Special Equations MAPW = MMCW . SW1 PCBW = MMCW . SW2 LOCKW = MMCW . SW3 MMCW = FUMMC . PH5 +FUMMC . PH4 . (NP32 . NP33) MAPWXD = FUMMC . EXU		

MMC

4 of 4

FUINT = ØU6.ØLB

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 T4RL Data Rel	$C_{4-15} \rightarrow D_{20-31}$, 0's $\rightarrow D_{0-19}$	DXC/9 = FUINT . PH1 S/T8L = FUINT . EXU . NENDE	T8L for remaining phases. Operand → C → D
PH2 T8L	D → S → RW (R → LR) C → D Set Request for next Instruction	{ SXD = FUINT . PH2 { RW = FUINT . PH2 DXC/6 = FUINT . PH2 (MRQ/1) = FUINT . PH2 S/LR31/2 = FUINT . PH2	$Operand_{4-15} \rightarrow R_{20-31}$ causes Q → P also Rul → LR in next phase
PH3 T8L	0's → RW (Rul → LR) Request Data Release on next clock	RW = FUINT . PH3 preset above S/LR31/2 = FUINT . PH3 S/DRQ = FUINT . PH3	Clear Rul so that Rul_{0-15} will be clear in final result Rul → LR in next phase
PH4 T8L Data Rel.	D → S → CC S → RW (Bytes 2 & 3 only) (Rul → LR) ENDE	SXD = FUINT . PH4 S/CC1 = FUINT . PH4 . S0 S/CC2 = FUINT . PH4 . S1 S/CC3 = FUINT . PH4 . S2 S/CC4 = FUINT . PH4 . S3 { (R/CC) = FUINT . PH4 { RWB2 = FUINT . PH4 { RWB3 = FUINT . PH4 ENDE = FUINT . PH4	$Operand_{0-4}$ → CC's $Operand_{16-31} \rightarrow Rul_{16-31}$

INT (6B) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 NANLZ T4RL	C → D $C_{0-7} \rightarrow O_{0-7}$ Set ANLZ flip-flop set PRE1 set T8L Inhibit clearing of LMXC first time thru PH1. (SW2 is set in PRE1) Inhibit S/PH2 (via NBR)	DXC/6 = FUANLZ . NANLZ . PH1 ØXC = FUANLZ . NANLZ . PH1 + ENDE S/ANLZ = FUANLZ . PH1 S/PRE1 = FUANLZ . NANLZ . PH1 . N(S/INTRAPF) S/T8L = FUANLZ . NANLZ . PH1 LMXC = B . NAR . I . (PH1 . FUANLZ . NSW2) . I - - - NBR = I . FUANLZ . NANLZ	ØXC sets LMXC ANLZ is used to distinguish between the two PH1's of this instruction and to inhibit functions normally performed by the instruction being analyzed.
PRE1 ANLZ T8L Preparation phases	Normal preparation sequence takes place to compute effective address, Disable P → Q Disable S/RQ (via PRERQ) Disable memory requests except for indirect addresses Set SW2 for control of LMXC set DRQ if immediate ANLZ PREPARATION CONT ON NEXT PAGE	QXP = PRE1 . NANLZ PRERQ = I . NPRERQ/1 B . NØL3 . NANLZ { MRQ = PRE1 . ØU2 . (NØ4 . NØ5 . NØ7) . NANLZ + PRE1 . (NØ1 . NØ3) . ØL2 . NANLZ + (PRE2 . NIA) . IX . ØPRQ . NANLZ + (PRE2 . NIA) . PREDØ . NANLZ + PRE3 . NIA . NANLZ + PRE2 . NIA . FUSTD . NANLZ + PRE2 . NIA . FAS18 . NANLZ RQC = PRE1 . NINDX . PREØPRQ/1 . NANLZ + PRE1 . NINDX . PREØPRQ/2 . NANLZ + PRE1 . CO S/SW2 = ANLZ . PRE1 S/DRQ = PREIM . ANLZ . PRE1	← Request allowed for indirect address. (in case indirect address was made)

ANLZ (44) 1 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
ANLZ Prep.	(cont.) Set Condition code to indicate opcode class. Force Analyze Opcode into O register upon completing preparation phases. suppress advancement to PH1 set SW1	(S/CC1/1) = (ANLZ.PRE1).FAIM.(NO1.NO3).(NO4.NO5) S/CC1 = FADW.(ANLZ.PRE1) (S/CC1/1) = (ANLZ.PRE1).FAW (R/CC1) = (ANLZ.PRE1) S/CC2 = FADW.(ANLZ.PRE1) (S/CC2/1) = OU5.(ANLZ.PRE1) R/CC2 = (ANLZ.PRE1) R/CC3 = (ANLZ.PRE1) (S/CC4/1) = (ANLZ.PRE1).FAIM R/CC4 = (ANLZ.PRE1) S/CC3 = (ANLZ.PRE1).DO OX/1 = (S/PH1/1) . ANLZ NBR = ... I.ANLZ.(S/PH1/1) S/SW1 = ANLZ.(S/PH1/1)	for Immediate word type Double word type Word type Double word type Halfword type Immediate type Indirect addressed
ANLZ.SW1 : NO PHASE IS SET. (This dummy T4RL	reset SW1 set PH1	pulse time allows FA's and FU's to settle) R/SW1 = ANLZ S/PH1 = ANLZ.SW1	
PH1 ANLZ T6L	NCC4 ⇒ P → S, P32 → S0, P33 → S1 ⇒ S → A 0's → D Request next instruction	SXP = ANLZ/1 . PH1 AXS = ANLZ/1 . PH1 DX/1 = ANLZ . PH1 MRQ/1 = ANLZ . PH1 ANLZ/1 = ANLZ . NCC4	also causes PXQ
PH2 ANLZ T6L	NCC4 . N(CC1 . NCC2) ⇒ A → S ⇒ Shift left one (cyclic)	SXA = PH2 . N(CC1 . NCC2) . ANLZ/1 AXSL1 = ANLZ/1 . N(CC1 . NCC2) . PH2	

ANLZ

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3 ANLZ T6L	NCC1 . NCC2 . NCC4 ⇒ A → S ⇒ Shift left one (cyclic) CC1 . CC2 . NCC4 ⇒ Shift right two Set DRQ Set T10L	SXA = NCC1 . NCC2 . PH3 . ANLZ/1 AXSL1 = ANLZ/1 . NCC1 . NCC2 . PH3 AXPRR2 = ANLZ/1 . CC1 . CC2 . PH3 (S/DRQ) = ANLZ . PH3 (S/T10L/1) = ANLZ . PH3	
PH4 T10L data release	Reset ANLZ flip-flop NCC4 ⇒ S → RW NCC4 ⇒ A → S ENDE	R/ANLZ = PH4 + CLEAR RW = ANLZ/1 . PH4 SXA = ANLZ . NCC4 . PH4 ENDE = ANLZ . PH4	

ANLZ

PHASE	FAPSD 0V0·04·05·06 FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1	LPSD ⇒ Go To PH5 SET REQUEST	NBRPH5 = FAPSD · PH1 · N07 MRQ = FAPSD · PH1 · N07	Suppress Mapping If---
T4RL	XPSD ⇒ RESET MAPDIS XPSD · NR30 ⇒ SET MAPDIS XPSD · INTRAPF ⇒ GO To PH3 SET REQUEST XPSD · NINTRAPF ⇒ P → Q Q → P PSW1 → D	R/MAPDIS = FAPSD · PH1 · 07 S/MAPDIS = FAPSD · PH1 · 07 · NR30 NBRPH3 = FAPSD · PH1 · 07 · INTRAPF MRQ = FAPSD · PH1 · 07 · INTRAPF NQXP = FAPSD · PH1 · 07 · N(INTRAPF) NPXQ = FAPSD · PH1 · 07 · N(INTRAPF) NDX PSW1 = FAPSD · PH1	
PH2	P → S → B	NSXP = FAPSD · PH2 NBXS = FAPSD · PH2 NPXQ = FAPSD · PH2 MRQ = FAPSD · PH2	
T6L	Q → P SET REQUEST		
PH3	P + 1 → P SET REQUEST	PCTP1 = FAPSD · PH3 MRQ = FAPSD · PH3 NSXD = FAPSD · PH3 NSXB = FAPSD · PH3 NMW = FAPSD · PH3 NDXPSW2 = FAPSD · PH3	[P → LB] WRITEWORD
T6L	D + B → S → MB PSW2 → D data release		

XPSD (0F), LPSD (0E)

1 of 4

PHASE	FAPSD FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4	P + 1 → P SET REQUEST	PCTP1 = FAPSD · PH4 MRQ = FAPSD · PH4 NSXD = FAPSD · PH4 NMW = FAPSD · PH4	[P → LB] WRITE WORD
T6L	D → S → MB data release		
PH5	P + 1 → P SET REQUEST	PCTP1 = FAPSD · PH5 MRQ = FAPSD · PH5 NDXC/6 = FAPSD · PH5	[P → LB]
T6L	MB → C → D data release		
PH6	MB → C → D D → S → PSW1	NDXC/6 = FAPSD · PH6 NSXD = FAPSD · PH6 NPSW1XS = FAPSD · PH6 NAXS = FAPSD · PH6	[P → LB]
T6L	→ A RESET MAPDIS LPSD ⇒ 0 → CI 0 → II 0 → EI TRAP.XPSD ⇒ TRACC → CC S → CC data release	R/MAPDIS = FAPSD · PH6 R/CI = FAPSD · PH6 · N07 R/II = FAPSD · PH6 · N07 R/EI = FAPSD · PH6 · N07 NCCXTRACC = FAPSD · PH6 · TRAP · 07 NCCXS = FAPSD · PH6	

XPSD, LPSD

2 of 4

PHASE	FAPSD FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7 T6L	D2, D3 → WRITE KEY D5 ⇒ 1 → CI D6 ⇒ 1 → II D7 ⇒ 1 → EI R28 ⇒ D23-D27 → Rp23-Rp27 Lpsd · R30 ⇒ 1 → INT 16 Lpsd . R30.R31 ⇒ 1 → INT 17 A→S → P TRAP · R29 ⇒ TRACC → D28-D31 RESET TRAP RESET INTRAPF BRING UP R/D0-31 Xpsd · NR29 ⇒ GO TO PH9 SET REQUEST Lpsd ⇒ ALLOW CLOCK INTERLOCK WITH INT10	S/WK0 = FAPSD · PH7 · D2 S/WK1 = FAPSD · PH7 · D3 S/CI = FAPSD · PH7 · D5 S/II = FAPSD · PH7 · D6 S/EI = FAPSD · PH7 · D7 NRPXD = FAPSD · PH7 · R28 INT16 = FAPSD · PH7 · N07 · R30 INT17 = FAPSD · PH7 · N07 · R30.R31 NSXA = FAPSD · PH7 NPXS = FAPSD · PH7 NDXTRACC = FAPSD · PH7 · TRAP · R29 R/TRAP = FAPSD · PH7 R/INTRAPF = FAPSD · PH7 NDX/1 = FAPSD · PH7 NBRPH9 = FAPSD · PH7 · 07 · NR29 MRQ = FAPSD · PH7 · 07 · NR29 NCEINT = FAPSD . PH7 . N07 . R30	UNCONDITIONALLY
PH8 T6L	D+A → S → P SET REQUEST	NPXS = FAPSD · PH8 SXADD = FAPSD · PH8 MRQ = FAPSD · PH8	

XPSD, LPSD

PHASE	FAPSD FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH9 T6L	Reset TRACC	R/TRACC = FAPSD . PH9	
PH10 T6L	ENDE data release	ENDE = FAPSD · PH10	
	SPECIAL S/DRQ = FAPSD · NPH7 · NPH8 · EXU		

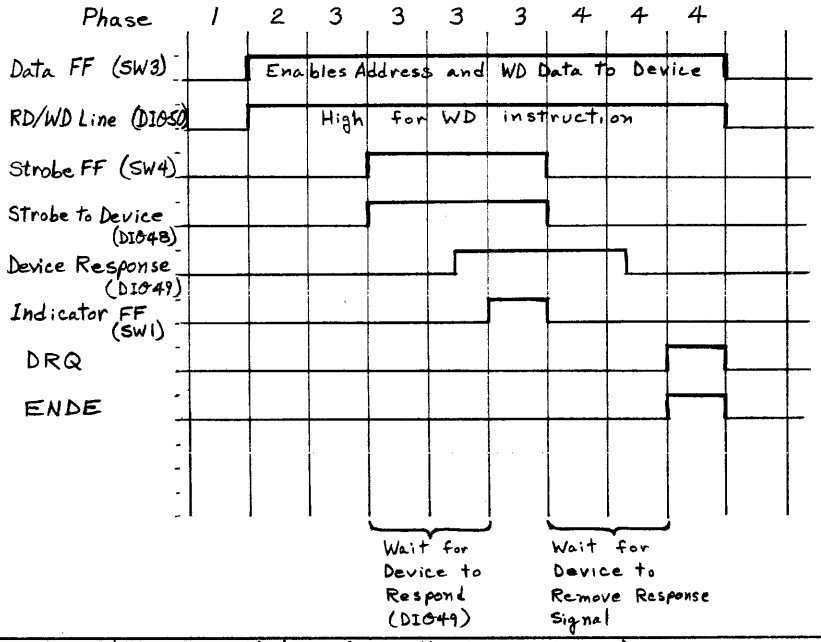
XPSD, LPSD

PHASE	FULRP = (QU2 · QLF) FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 T4RL	MB → C → D	NDxC/6 = FULRP · PH1	
PH2 T6L	D23 – D27 → RP23 – RP27 SET REQUEST	NRPxD = FULRP · PH2 MRQ/I = FULRP · PH2 S/DRQ = FULRP · PH2	
PH3 T6L	ENDE DATA RELEASE	ENDE = FULRP · PH3	

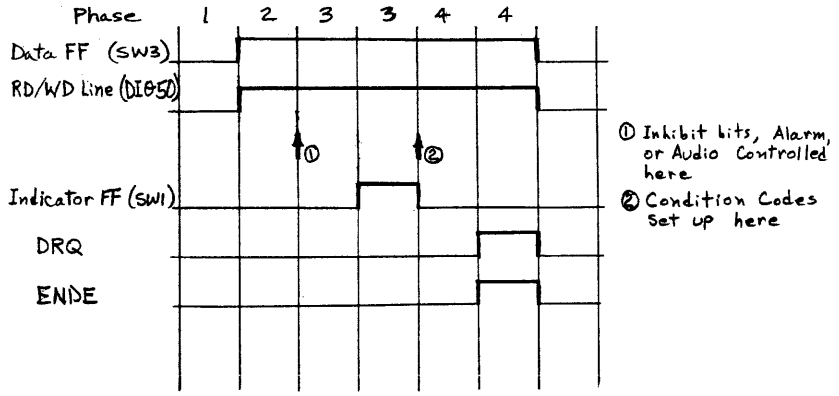
LRP(2F)

WD, RD Instructions
Simplified Timing Diagram
(Ignoring Circuit and line delays)

WD, RD



WD Internal Control Mode (Bits 16-19 equal zero)



RD (6C), WD (6D)

Family: FARWD = 0U6. (04.05.N06)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Some important signals used in WD and RD instructions.	INHXWD = 0U6·0LD·PH2·B16RZ·B26 WDINTL = 0U6·0LD·PH2·B1619Z B1619Z = NB16·NB17·NB18·NB19 FARWD = 0U6·04·05·N06	} For internal Control Mode of WD instruction Decode of B Reg. Bits Family (WD, RD) Decode
PH1	Set Memory Request and Q→P P → S → B 0 → A R → LR RR → A	MRQ/1 = FARWD. PH1 SXP, BXS = FADI0. PH1 AX/1 = FARWD. PH1 AXRR = FARWD. PH1	Request for next instr. Eff. Address → B No special control for R → LR
T4RL	Set DATA Flip-flop Reset Indicator Flip-flop Reset Strobe Flip-Flop	S/SW3 = FADI0. PH1 } These, as well as SW3 above, are already reset by CLEAR during previous ENDE	} Multi-purpose Flip-flops, SW1, SW3, and SW4 are used here, "Data FF" is SW3 "Indicator" is SW1 "Strobe" is SW4
PH2	DATA FF ⇒ B16-31 → DI032-47 DATA FF. WD ⇒ A → S ⇒ 1 → DI050 WD. DATA FF. R≠0 ⇒ S → DI00-31	DI0XB = FARWD. SW3 SXA = 0LD. 0U6. SW3 DI050X1 = 0LD. 0U6. SW3 DI0XS = 0LD. 0U6. SW3. NRZ	Address Lines Data → S for WD RD/WD Line Data Out for WD if R≠0
T6L	WD·(B16-19=0)·B26·B27·B29 ⇒ Set CI WD·(B16-19=0)·B26·B29 ⇒ Reset CI WD·(B16-19=0)·B26·B27·B30 ⇒ Set II WD·(B16-19=0)·B26·B30 ⇒ Reset II WD·(B16-19=0)·B26·B27·B31 ⇒ Set EI WD·(B16-19=0)·B26·B31 ⇒ Reset EI	S/CIF = INHXWD. B27. B29 R/CIF = INHXWD. B29 S/II = INHXWD. B27. B30 R/II = INHXWD. B30 S/EI = INHXWD. B27. B31 R/EI = INHXWD. B31	Control of Inhibit Bits contained in PSW2
	WD·(B16-19=0)·B25·B31 ⇒ Set Alarm WD·(B16-19=0)·B25 ⇒ Reset Alarm WD·(B16-19=0)·B25·B30·$\overline{\text{MUSIC}}$ ⇒ Set MUSIC WD·(B16-19=0)·B25·B30 ⇒ Reset MUSIC ALARM FF ⇒ ALARM LIGHT ON MUSIC FF. $\overline{\text{ALARM FF}}$ +ALARM FF. RUN. 1KC ⇒ AUDIO	S/ALARM = WDINTL. B25. B31 R/ALARM = WDINTL. B25 + RESET S/MUSIC = WDINTL. B25. B30. NMUSIC R/MUSIC = WDINTL. B25. B30 ALARM/L = ALARM AUDIO = MUSIC·NALARM +(ALARM. KRUN/B). 1KC	Alarm Control Audio Control (Toggles) Signal to Alarm Light Signal to Speaker

RD, WD

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3	DI049 + (B16-19=0) ⇒ Set Indicator $\overline{\text{Indicator}}$ ⇒ Stay in PH3 ⇒ Set Strobe FF Indicator ⇒ Reset Strobe FF	S/SW1 = (FARWD. PH3). DI049. NSW1 +(FARWD. PH3). B1619Z. NSW1 BRPH3 = FARWD. PH3. NSW1 S/SW4 = (FARWD. PH3). NSW1 R/SW4 = FADI0. SW1	Using SW1 for "Indicator" Using SW4 for "Strobe"
T6L	Strobe ⇒ 1 → DI048 Indicator·DI051·(B16-19≠0) + (B16-19=0)·KSS3 ⇒ Set CC3 Indicator·DI052·(B16-19≠0) + (B16-19=0)·KSS3 ⇒ Set CC4 (B16-19=0)·KSS1 ⇒ Set CC1 (B16-19=0)·KSS2 ⇒ Set CC2 Indicator ⇒ Reset CC1,2,3,4 Indicator ⇒ DI00-31 → D Indicator ⇒ Reset Indicator R → LR	DI048X1 = FARWD. SW4 (S/CC3/1) = (FARWD. SW1). NB1619Z. DI051 S/CC3 = (FARWD. SW1. B1619Z). KSS3 (S/CC4/1) = (FARWD. SW1). NB1619Z. DI052 S/CC4 = (FARWD. SW1. B1619Z). KSS4 S/CC1 = (FARWD. SW1. B1619Z). KSS1 S/CC2 = (FARWD. SW1. B1619Z). KSS2 (R/CC) = (FARWD. SW1) DXDI0 = (FARWD. SW1). NB1619Z R/SW1 = FADI0	Set Condition Code from Sense Switches or from External Device R → LR occurs normally without special control.
	RD·(B16-19=0)·Indicator·R≠0·B27 ⇒ PFSR RD·(B16-19=0)·$\overline{\text{Indicator}}$·B27 ⇒ Parity Error1-8 → D24-31	PFSR = (FARWD. SW1). (B1619Z. N07. NRZ. B27) DXPARITY = FARWD. (B1619Z. N07. NRZ. B27) PH3	Reset Parity Indicators with short pulse on /PFSR Puts MFL0-7 into D24-31
PH4	$\overline{\text{DI049}}$ + (B16-19=0) ⇒ Set DRQ $\overline{\text{DRQ}}$ ⇒ Stay in Phase 4 DRQ ⇒ Reset Data FF	S/DRQ = FARWD. PH4. NDI049 +FARWD. PH4. B1619Z BRPH4 = FARWD. PH4. NDRQ R/SW3 = CLEAR	Wait in Phase 4. CLEAR = ENDE + ----
T6L / T8L	DRQ ⇒ ENDE R → LR RD. R≠0 ⇒ D → S → RW set T8L if not ENDE	ENDE = FARWD. PH4. DRQ RW = 0LC. 0U6. PH4. NRZ SXD = 0LC. 0U6. PH4. NRZ S/T8L = FARWD. PH4. NENDE	Load Input Data

RD, WD

Family: FAIO, FAIO/1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1	P → S → B S → A 1's → CS	BXS, SXP = FAIO. PH1 AXS = FAIO. PH1 S/NPRX = FAIO. PH1	P contains I/O ADDRESS 14 → CS for up. align. in PH2.
T4RL	Set Data FF Reset Indicator FF Set MAPDIS	S/SW3 = FADIO. PH1 (R/SW1) = FADIO + CLEAR S/MAPDIS = FAIO. PH1 (S/NLRXR) = FAIO. PH1 T6RL = FAIO. PH1	Using SW3 for Data FF Using SW1 for Indicator Prevent Mapping 0$ → LR for PH2 select T6RL clock for PH2
PH2	$A_{24-31} \rightarrow K \rightarrow S_{0-7, 8-15, 16-23, 24-31}$ $S_{0-7} \rightarrow A_{0-7}$ 0's → A 0's → LR SIO ⇒ $RR_{16-31} \rightarrow A_{16-31}$ R≠0 · R31 ⇒ 1 → A8	SXUAB = FAIO. PH2 AXS/1 = FAIO. PH2 AX/1 = FAIO. PH2 (Preset during PH1) AXRR/2=OLC. OU4. PH2 A8X1 = FAIO. PH2. NRZ. NR31	Upward Align 'A' Reg Clear all 'A' Reg not being set. Address Register 0 (Reg. 0 contains first command address)
T6RL	R≠0 ⇒ 1 → A9 AIO ⇒ Set Memory Request Set ARQ Data FF · 02 ⇒ FNC0 Data FF · 06 ⇒ FNC1 Data FF · 07 ⇒ FNC2 Data FF · B21 ⇒ IOPA0 Data FF · B22 ⇒ IOPA1 Data FF · B23 ⇒ IOPA2	A9X1 = FAIO. PH2. NRZ MRQ = FAIO/1. PH2 S/ARQ = FAIO. PH2 IOPXFCAD = FAIO. SW3 (Note this occurs for all phases that SW3 is set)	} Function Code to IOP's } IOP Address to IOP's
	$20_{16} \rightarrow P$ ($20_{16} = 32_{10}$)	NPX20 = FAIO. PH2	Load Memory Address 20_{16} into P Register.

SIO (4C), TIO (4D), TDV (4E), HIO (4F), AIO (6E)

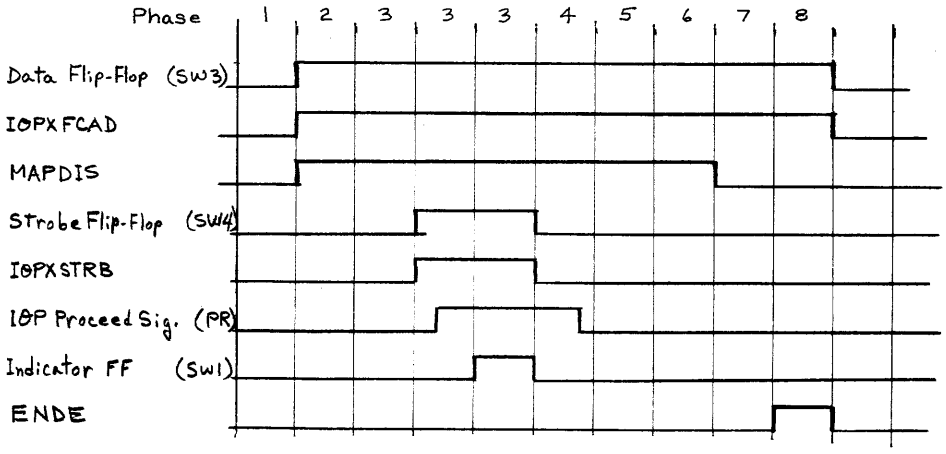
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	P is normally on LB		
PH3	$20_{16} \rightarrow LB$ A → S → MB IOP PROCEED · STROBE ⇒ Set Indicator IOP PROCEED · Indicator ⇒ Set STROBE Indicator ⇒ Reset Strobe Indicator ⇒ Stay in Phase 3 ⇒ Set DRQ	SXA, MW = FAIO. PH3 S/SW1 = FAIO3. PR. NSW1. SW4 S/SW4 = FAIO3. NPR. NSW1 R/SW4 = FADIO. SW1 BRPH3 = FAIO3. NSW1 S/DRQ = FAIO3. NSW1	20_{16} loaded into P in PH2 Using SW1 for Indicator Using SW4 for Strobe Wait in PH3 until SW1 is set.
T6L	Indicator · IOP CC1 Signal ⇒ S/CC1 Indicator · IOP CC2 Signal ⇒ S/CC2 Indicator ⇒ Set Memory Request AIO · Indicator · (R≠0) · NCOND1 ⇒ Go to PH6 Indicator · [(R=0) + COND1] ⇒ Go to PH8 ⇒ Q → P ⇒ R/MAPDIS ⇒ Set DRQ	(S/CC1/1) = FAIO3. COND1. SW1 (S/CC2/1) = FAIO3. COND2. SW1 MRQ = FAIO3. SW1 BRPH6 = (OU6. OLE. NRZ. NCOND1). SW1 BRPH8 = (FAIO. RZ + FAIO. COND1). SW1 PXQ = (FAIO. RZ + FAIO. COND1). PH3. SW1 R/MAPDIS = (FAIO. RZ + FAIO. COND1). SW1 S/DRQ = (FAIO. RZ + FAIO. COND1). SW1	Note: FAIO3 = FAIO. PH3 (next instruction address)
	Set ARQ STROBE ⇒ 1 → IOP Strobe line No Branch ⇒ 1 → LB31 (This is being set up for use in PH4)	S/ARQ = FAIO. PH3 IOPXSTRB = FAIO3. SW4 (S/LB31/1) = FAIO. PH3. NBR	Strobe → IOP via CNST line Note: No Data Release on 1st Clock in PH3.
Data Rel Add Rel	Indicator Set ⇒ Reset Indicator	(R/SW1) = FADIO S/SW1 = NSW1 ← see set terms for SW1 at top of sheet	
PH4	$21_{16} \rightarrow LB$ Set Memory Request Set DRQ R31 ⇒ Q → P ⇒ Reset MAPDIS	S/MRQ = FAIO. PH4 S/DRQ = FAIO. PH4 PXQ = FAIO. PH4. R31 (R/MAPDIS) = FAIO. PH4. R31	This is done by loading 20_{16} into P in PH2 and setting LB31/1 in PH3. Only one response word stored if R31=1. Enable Mapping Again.
T6L Add. Rel			
PH5	MB → C → D $20_{16} \rightarrow LB$ if $\overline{R31}$ $\overline{R31}$ ⇒ Set ARQ	DXC/6 = FAIO. PH5 S/ARQ = FAIO. PH5. NR31	MB represents cell 21 20_{16} preset into P in PH2 but if R31=1 then contents of Q is in P.
T6L Data Rel			

SIO, etc.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6 TGL Add Rel	$\overline{R31}$ + AIO ⇒ MRQ/1 ⇒ S/DRQ Reset MAPDIS Set Ru1 → LR for use in PH7	MRQ/1 = FAIO . PH6 . (NR31+02) S/DRQ = FAIO . PH6 . (NR31+02) (R/MAPDIS) = FAIO . PH6 S/LR31/2 = FAIO . PH6 (S/T8L/1) = FAIO . PH6	(for next instruction) Enable Mapping
PH7 T8L Data Rel	MB → C Ru1 → LR $\overline{AIO}$ ⇒ C_{0-15} → D_{16-31}, C_0 → D_{0-15} ⇒ D → S → RW AIO ⇒ C → D ⇒ 20_{16} → LB Set DRQ Except for $\overline{AIO}$. R31	Automatic when Request is made — Preset in Phase 6 DXCR16 = FAIO/1 . PH7 SXD, RW = FAIO/1 . PH7 DXC/6 = 0U6 . GLE . PH7 — Preset into P in Phase 2 (S/T8L/1) = FAIO . PH7 (S/DRQ) = FAIO . PH7	(MB represents cell 20 or the next instruction) (D represents cell 21)
PH8 T8L Data Rel	P → LB R → LR R≠0 . $\overline{CC1}$. ($\overline{R31}$ + AIO) → RW $\overline{AIO}$ ⇒ D_{16-31} → S_{16-31} Reset Data FF ENDE AIO ⇒ D → S	RW = FAIO . PH8 . NRZ . NCC1 . (NR31+02) SXD/2 = FAIO/1 . PH8 R/SW3 = CLEAR ENDE = FAIO . PH8 SXD = FAIO . PH8 . 02	} Normal Operation unless otherwise preset. Write into Register R. Halfword on S from cell 20 Word on S.

Simplified Timing Diagram
(Ignoring Circuit and Line Delays)

SIO, TIO, TDV, HIO with Even, Nonzero R Field



Repeat this phase until IOP responds with PR signal then set Indicator.

Repeat this phase until IOP signal PR is low then set Strobe

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	S/INTRAPF		
	NENDE.NPRE1+ ENDE.(S/TRAP).N(S/BRQ/1)⇒S/BRQ	S/BRQ = (S/INTRAPF).NENDE.NPRE1 + ENDE.(S/TRAP).I.N(S/BRQ/1)	set BRQ if trap or interrupt is to point at current instruction rather than next instruction
		(S/BRQ/1) = FABØ.PH3 + FABR.PH3 + FUBAL	Branch instruction that trap because of memory protect or non-existent address point at the branch instruction itself.
	S/DRQ	S/DRQ = (NENDE + AHCL/1)(S/INTRAPF)	If trap is for non-existent address, set DRQ even if ENDE.
	CLEAR	CLEAR = (S/INTRAPF)	
	S/INTRAPF	S/INTRAPF = (S/INTRAPF)	
	S/INTRAP1	S/INTRAP1 = (S/INTRAPF).NRESET	

INTRAP SEQUENCE - (SET INTRAPF)

1 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
INTRAP1 INTRAP2	P ⇒ S ⇔ A	AXS = INTRAP1	
		SXP = INTRAP1.NSDIS	
	0 ⇒ D	DX/1 = INTRAP1	
	1's ⇒ CS	CSX1 = INTRAP1.NSDIS	
	S/DRQ	S/DRQ = INTRAP1	
	CLEAR	CLEAR = INTRAP1	
	R/HALT	R/HALT = INTRAP1	
	S/MAPDIS	S/MAPDIS = INTRAP1	disable mapping
	NTRAP ⇒ prevent stopping clocks after next clock if single clocking.	SCEN = N(.INTRAP1.NINTRAP2.NTRAP)	associated with CEINT in INTRAP1.INTRAP2
	S/TIOL	S/TIOL = INTRAP1.NINTRAP2	
	S/INTRAP2	S/INTRAP2 = INTRAP1.NRESET	
Timing is function of previous clock			
Data Rel.			

INTRAP SEQUENCE - (INTRAP1.NINTRAP2)

2 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
INTRAP1 INTRAP2 T10L	P → S ⇒ A	AXS = INTRAP1 SXP = INTRAP1. NSDIS	
	0 ⇒ D	DX/1 = INTRAP1	
	1's ⇒ CS	CSX1 = INTRAP1. NSDIS	
	S/DRQ	S/DRQ = INTRAP1	
	CLEAR	CLEAR = INTRAP1	
	R/HALT	R/HALT = INTRAP1	
	S/MAPDIS	S/MAPDIS = INTRAP1	
	NTRAP ⇒ Synchronize next CPU clock with interrupt clock	CEINT = INTRAP1.INTRAP2.NTRAP	
	TRAP ⇒ TR28-31 ⇒ P28-31 S/P25	PXTR = INTRAP1.INTRAP2. TRAP S/P25 = PXTR S/P28 = TR28. PXTR	typical term
	INT. NTRAP ⇒ INTO -8 → P23-31	PXINT = INTRAP1.INTRAP2.NTRAP.INT S/P23 = INTO. PXINT	typical term
	NTRAP. NINT ⇒ S/P25	PXTR = INTRAP1.INTRAP2.(TRAP+NINT)	causes interrupt system fault trap to 40_{16}
	MRQ	MRQ = INTRAP1. INTRAP2	
	R/INTRAP1	R/INTRAP1 = INTRAP2 + RESET	
	R/LMXC	LMXC = LMXC. N(INTRAP1. INTRAP2)	
Data Rel.	R/AHCL	AHCL = AHCL. N(INTRAP1. INTRAP2)	

INTRAP SEQUENCE - (INTRAP1. INTRAP2)

3 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
INTRAP1 INTRAP2 T6L	A-1 → S ⇒ B	SXADD = (NINTRAP1.INTRAP2). NSDIS BXS = (")	
	C ⇒ D	DXC/6 = (")	
	C_0 ⇒ IA C_{1-7} ⇒ O_{1-7}	ORXC = (")	
	C_{8-11} ⇒ R_{28-31}		
	S/PRE1	S/PRE1 = (")	
	P ⇒ Q	QXP = INTRAP2	
Data Rel.			

INTRAP SEQUENCE - (NINTRAP1. INTRAP2)

4 of 4

FADE 0U7[04+N04·05·06]

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 TARL		S/DRQ = FADE.EXU S/TIOL = FADE.EXU DUCLOCK = FADE.NPH3.NPH9.NPREP	} for all phases
PH2 TIOL DUCLOCK	15 → R PRESET UPWARD ALIGN DSA ⇒ P→S ↔ A (DST+UNPK) ⇒ BR TO PH5 PRESET RR→C SET INTERRUPTIBLE DM → DU12 04-07 → DU4-DU7 R28-R31 → DU0-DU3 START → DU9 DSA ⇒ PRESET S↔C	RX1 = FADE·PH2 (S/NPRX) = FADE·PH2 AXS = FADE·PH2·0LC SXP = FADE·PH2·0LC BRPH5 = FADE·PH2·(05·06·07) (S/CXRR) = FADE·PH2·(05·06·07) (S/IEN) = FADE·PH2 DUMDM = FADE·PH2.DM DUX0 = FADE·PH2 DUXR = FADE·PH2 DUSTART = FADE·PH2 (S/CXS) = FADE·PH2·0LC DUCLOCK = FADE.NPH3.NPH9.NPREP	15 → R SEND START AND DETAILS OF INSTRUCTION TO DU
PH3 TIOL	DSA ⇒ SET REQUEST UPWARD ALIGN 1/2 WD DSA ⇒ 0 ↔ P32·P33	MRQ = FADE·PH3·N0LC SXUAH = FADE·PH3 PX/1 = FADE·PH3·0LC (R/SW1) = FADE.EXU	PREPARE TO LOAD CORE OPERAND OR TO LOAD SHIFT NUMBER.

DA(79), DS(78), DL(7E), DST(7F), DC(7D), DM(7B), DSA(7C), DD(7A), PACK(76), UNPK(77)

FADE

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 TIOL DUCLOCK	DOWNWARD ALIGN D24-D31 → S ↔ DU (Pbyte=3)·N(ENDFIELD) ⇒ SET REQUEST "ENDFIELD" ⇒ 1 ↔ SW1 ⇒ SET RR→C SW1=0 ⇒ REPEAT PH4 P+1/4 → P (DL+PACK)(SW1) ⇒ BR TO PH6 [READ ⇒ P→LB, MB→C] data Release	DXCBP = FADE·PH4 DUXS = FADE·PH4 SXD/1 = FADE·PH4 MRQ = FADE·PH4·P32·P33·NDUEND·NSW1 S/SW1 = (FADE·EXU)·DUEND·NENDE (S/CXRR) = FADE·PH4·DUEND BRPH4 = FADE·PH4·NSW1 PCTP1 = FADE·PH4 PA33 = FADE·PH4 (R/SW1) = FADE.EXU DUCLOCK = FADE.NPH3.NPH9.NPREP BRPH6 = FADE·PH4·(05·06·N07)·SW1	C ↔ D USING P byte 1ST BYTE TO DU IS JUNK SPECIAL DU MUST BRING UP END FIELD DURING NEXT TO LAST BYTE TRANSFER LOOP UNTIL DU SAYS IT HAS ALL THE SPECIFIED BYTES INCREMENT P BY 1 byte WHOLE MEMORY WORD GOES TO C REGISTER
PH5 TIOL DUCLOCK	RR → C C DOWNWARD ALIGN ↔ D D24-D31 → S ↔ DU E6 E7 = 01 ⇒ R-1 ↔ R N(ENDFIELD) ⇒ REPEAT PH5 ENDFIELD·(DM+DD) ⇒ RESET INTERRUPTIBLE 0 → A, E-1 → E	(S/CXRR) = FADE·PH5 DXCR24 = FADE·PH5·NE6·E7 DXCR16/1 = FADE·PH5·E6·NE7 DXCR8 = FADE·PH5·E6·E7 DXC/1 = FADE·PH5·NE6·NE7 DUXS, NSXD/1 = FADE·PH5 MCTR = FADE·PH5·NE6·E7 BRPH5 = FADE·PH5·NDUEND R/IEN = FADE·PH5·DUEND·(N05·06) AX/1 = FADE·PH5, MCTE1 = FADE.PH5 DUCLOCK = FADE.NPH3.NPH9.NPREP	BYTE TO DU LOOP DU MUST BRING UP ENDFIELD WHILE LAST BYTE IS IN D REGISTER

DA, etc.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6 T10L DUCLOCK	N ENDFIELD ⇒ REPEAT 6 ENDFIELD ⇒ 1's → R 0 → E DUCABLE → D24-31 CLOCK → DU ENDFIELD ⇒ SET CONDITION CODES FROM DU CABLE INFO ENDFIELD.(DC + DU5) ⇒ GO TO PH9 SET REQUEST N(PACK + UNPK) ⇒ RESET INTERRUPTABLE (DM + DD).INT ⇒ DUMDM DU4 ⇒ SET TRAP (ADDR X'45')	BRPH6 = FADE.PH6.NDUEND RX1 = FADE.PH6.DUEND EX/1 = FADE.PH6.DUEND DXDU = FADE.PH6 DUCLOCK = FADE.NPH3.NPH9.NPREP S/CC1 = CCXDU.DU0 R/CC1 = CCXDU.DU6 S/CC2 = CCXDU.DU1 R/CC2 = CCXDU.DU7 S/CC3/1 = CCXDU/1.DU2 R/CC3 = CCXDU/1 S/CC4/1 = CCXDU/1.DU3 R/CC4 = CCXDU/1 CCXDU = FADE.PH6 DUEND CCXDU/1 = CCXDU.N(05.06.07).DUEND.NDU5 BRPH9 = FADE.PH6.DUEND.(DU7.OLD+DU5) MRQ/1 = FADE.PH6.DUEND.(DU7.OLD+DU5) R/IEN = FADE.PH6.04 DUMDM = FADE.PH6.INT.(N05.06) (S/TRAP/1) = (S/TR29/1).NPH8 S/TR31 = (S/TR29/1)+--- (S/TR29/1) = CCXDU.DU4	WAIT UNTIL DU READY 15 → R Not DST OR UNPK DU5 ⇒ ABORT INSTR. INHIBIT INTERRUPTS
PH7 T10L DUCLOCK	DU CABLE → D24-31 (DST + UNPK) ⇒ SET WRITE BYTE PRESET UPWARD ALIGN CLOCK → DU ENDFIELD ⇒ S/SW1 OTHERWISE RESET SW1	DXDU = FADE.PH7 MRQ = FADE.PH7.(05.06.07) S/NPRX = FADE.PH7 DUCLOCK = FADE.NPH3.NPH9.NPREP S/SW1 = FADE.EXU.DUEND.NEND R/SW1 = FADE.EXU	

DA, etc.

3 of 4

PHASE — FADE

FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH8 T10L DUCLOCK	DU → D24-D31 1's → CS NSW1 ⇒ REPEAT PH8 NSW1(DST + UNPK) ⇒ SET REQUEST SW1 ⇒ S/INTERRUPTIBLE SW1·N(CC1·CC2) ⇒ SET REQUEST (END) (PST+UNPK) ⇒ D24-31 → S → MB (DST+UNPK).NSW1 ⇒ P+1/4 → P ⇒ SET WRITE BYTE N(DST+UNPK) ⇒ D24-D31 → S → RW N(DST+UNPK) ⇒ E-1 → E N(DST+UNPK).(E6,E7=0) ⇒ R-1 → R ENDFIELD ⇒ S/SW1	DXDU = FADE·PH8 CSX1 = FADE·PH8 BRPH8 = FADE·PH8·NSW1 MRQ = FADE·PH8·NSW1·(05·06·07) (S/IEN) = FADE·PH8·SW1 MRQ/1 = FADE·PH8·SW1·N(N05.06.CC1.CC2) SXUAB = FADE·PH8· PCTP1 = FADE·PH8·(05·06·07)·NSW1 PA33 = FADE·PH8·(05·06·07)·NSW1 MWB = FADE·PH8·(05·06·07)·NSW1 SXAUB = FADE·PH8 RWB0 = (FADE·PH8)N(05·06·07)·NE6·E7 RWB1 = (FADE·PH8)·N(05·06·07)·E6·NE7 RWB2 = (FADE·PH8)·N(05·06·07)·E6·E7 RWB3 = (FADE·PH8)·N(05·06·07)·NE6·NE7 MCTE1 = FADE·PH8·N(05·06·07) MCTR = FADE·PH8·N(05·06·07)·NE6·E7 S/SW1 = (FADE·EXU)·DUEND·NENDE	LOOP WITHIN PH8 result back to DEC. ACC UPWARD ALIGNED P byte UPWARD ALIGNED SPECIAL
PH9 T10L	N(CC1·CC2) ⇒ ENDE	ENDE = FADE·PH9·N(N05.06.CC1.CC2)	

DA, etc.

4 of 4

PHASE / FUNCTION PERFORMED		SIGNALS INVOLVED	COMMENTS
		FUEXU = 0U6. 0L7	Opcode decoding Signal
PRE2.NIA:	set DRQ	S/DRQ = PRE2.NIA.0PRQ.FUEXU.NFABRANCH	
PH1 T4RL	MB → C →→ D Q →→ P Set DRQ	DXC/6 = FUEXU. PH1 PXQ = FUEXU. PH1 S/DRQ = FUEXU. PH1	
PH2 T6L Data Release	ENDE Suppress P+1 →→ P Suppress Instruction Protect	ENDE = FUEXU. PH2 PCTPIDIS = I. NENDE I. NTRAP. NHALT. NFUEXU. NKAHOLD (S/TRACC4/1) = PROTECTI. ENDE. NFUEXU. NTRAP	Causes C → 0, R, D

EXU (67)

1 of 1

PHASE / FUNCTION PERFORMED		SIGNALS INVOLVED	COMMENTS
		FUBAL = 0U6. 0LA	Opcode decoding signal
PH1 T4RL	P →→ Q Q → P	QXP = FUBAL. PH1 PXQ = FUBAL. PH1 S/TBL = FUBAL. PH1	
PH2 T8L	P → S → RW P →→ Q Q →→ P Set Request	RW = FUBAL. PH2 SXP = FUBAL. PH2 QXP = FUBAL. PH2 PXQ = FUBAL. PH2 MRQ/1 = FUBAL. PH2 S/DRQ = FUBAL. PH2	PXQ in case of Trap
PH3 T6L data Release	ENDE PROTECT FAIL ⇒ Set BRQ	ENDE = FUBAL. PH3 S/BRQ = ENDE. (S/TRAP). I. N(S/BRQ/1) where N(S/BRQ/1) = I. FUBAL	BRQ indicates that return addres is in Q register

BAL (6A)

1 of 1

FABC = (⊖U6 . ⊖4 . N⊖5 . N⊖6)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2.NIA go to PH3 T4RL do not set T4RL		BRPH3 = FABC . PRE2 . NIA T4RL = I . FABC . PRE2 . NIA	if branch
PH3 T6L	BCS . (R.CC) } ⇒ ENDE +BCR . N(R.CC) }	ENDE = FABC . PH3 . ⊖7 . (R.CC) +FABC . PH3 . N⊖7 . N(R.CC)	(R.CC) = R28 . CC1 +R29 . CC2 +R30 . CC3 +R31 . CC4
	BCS . N(R.CC) } ⇒ Q ↔ P +BCR . (R.CC) } set Request suppress Protect Fail	MRQ/1 = FABC . PH3 . ⊖7 . N(R.CC) +FABC . PH3 . N⊖7 . (R.CC) (S/TRACC4/1) = PROTECTD . NPROTECTDIS where . NPROTECTDIS = I . (FABR+FABCNBRANCH) and (FABR+FABCNBRANCH) = FABC . PH3 . ⊖7 . N(R.CC) +FABC . PH3 . N⊖7 . (R.CC)	if don't branch (S/TRACC4/1) sets trap and TRACC4
data release	ENDE . PROTECTFAIL ⇒ set BRQ	S/BRQ = FABC . PH3 . ENDE . (S/TRAP) via (S/BRQ/1)	BRQ indicates return address from trap is in Q
PH4 T6L	set DRQ	S/DRQ = FABC . PH4	
PH5 T6L data release	ENDE	ENDE = FABC . PH5	No branch

BCS (69), BCR (68)

1 of 1

FABR = (⊖U6 . N⊖4 . ⊖5 . N⊖6)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE2 . NIA T4RL	RR ↔ A BIR ⇒ 1 ↔ CS31 BDR ⇒ 1's ↔ CS go to PH2 do not set DRQ do not set T4RL	AXRR = FABR . PRE2 . NIA CSX1/8 = FABR . PRE2 . NIA . ⊖7 CSX1 = FABR . PRE2 . NIA . N⊖7 BRPH2 = FABR . PRE2 . NIA S/DRQ = PRE2 . NIA . NFABRANCH . ... NFABRANCH = I ⁂ + . FABR T4RL = I . FABR . PRE2 . NIA	R ↔ A +1 ↔ CS -1 ↔ CS data release for the operand (the branched to instruction) is in PH3.
PH2 T6L	A + CS → S ↔ A	AXS = (FABR . PH2) SXADD = " S/TIOL = " S/DRQ = "	R±1 ↔ A
PH3 T10L	A → S ↔ RW	SXA = FABR . PH3 RW = FABR . PH3 RWDIS = FABR . ENDE . (PROTECTD + PROTECTI)	R±1 ↔ R suppress changing R if branching, and protect fail on branched to instruction.
	BDR . (A>0) } ⇒ ENDE +BIR . (A<0) }	ENDE = FABR . PH3 . N⊖7 . NAO . NA0031Z +FABR . PH3 . ⊖7 . AO	if branch
	BDR . N(A>0) } ⇒ Q ↔ P +BIR . N(A<0) } set Request suppress Protect Fail	MRQ/1 = FABR . PH3 . N⊖7 . N(NAO . NA0031Z) +FABR . PH3 . ⊖7 . NAO (S/TRACC4/1) = PROTECTD . NPROTECTDIS and NPROTECTDIS = I . (FABR+FABCNBRANCH) and (FABR+FABCNBRANCH) = FABR . ⊖7 . NAO (via (BRQ/1))? +FABR . N⊖7 . N(NAO . NA0031Z)	if don't branch (S/TRACC4/1) sets TRAP and TRACC4
data release	ENDE . PROTECTFAIL ⇒ set BRQ	S/BRQ = FABR . PH3 . ENDE . (S/TRAP)	BRQ indicates return address from trap is in Q.
PH4 T6L	set DRQ	S/DRQ = FABR . PH4	
PH5 T6L data release	ENDE	ENDE = FABR . PH5	No branch

BDR (64), BIR (65)

1 of 1

FUWAIT = (GU2 · GLE)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1 T4RL	SET REQUEST SET HALT	MRQ/1 = FUWAIT · PH1 S/HALT = FUWAIT PH1 S/DRQ = FUWAIT PH1	Q → P
PH2 T6L	ENDE DATA RELEASE	ENDE = FUWAIT · PH2	

WAIT (2E)

1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
		FACAL = GU0. (NG4. G5)	Opcode decoding signal
PH1 T4RL	R → TRACC1, 2, 3, & 4 Set TR28 CAL3 or CAL4 ⇒ Set TR30 CAL2 or CAL4 ⇒ Set TR31 Set TRAP	S/TRACC1 = (FACAL. PH1. NTRAP. NSTRAP). R28 S/TRACC2 = (FACAL. PH1. NTRAP. NSTRAP). R29 S/TRACC3 = (FACAL. PH1. NTRAP. NSTRAP). R30 S/TRACC4 = (FACAL. PH1. NTRAP. NSTRAP). R31 S/TR28 = (FACAL. PH1. NTRAP. NSTRAP) S/TR30 = FACAL. PH1. G6 (S/TR31/1) = FACAL. PH1. G7. NSTRAP (S/TRAP/1) = FACAL . PH1	Set Up Address of Trap Location
PH2 T6L	Proceed to INTRAP Sequence	S/INTRAPF = TRAP . NINTRAPF	

CAL1 (04), CAL2 (05), CAL3 (06), CAL4 (07)

1 of 1