CUSTOMER SERVICE

TEK TIP NO. 65-75-S11

Date January 8, 1968 Ref. E.O. No.

Subject: SIGMA 5 CPU ENGINEERING NOTES AND PHASE SEQUENCE CHARTS

Related Model Numbers: 8201 Sigma 5 CPU
8218 Sigma 5 Floating Point Option

Distribution: Customer Service

Void After:

Technical Discussion: Attached are Sigma 5 CPU Engineering Notes and Phase Sequence Charts. This material, which reflects the general operation of the Sigma 5 CPU, is for reference only. The simplified equations shown in the phase sequence charts are functional and representative in nature. These equations are not necessarily complete nor shown as implemented. The specific logic equations must be consulted to determine how a given function is implemented.

The drawing numbers of logic equations to be used in conjunction with the attached material are:

1. Sigma 5 CPU Equations 133263
2. Sigma 5 Floating Point Equations 134106

Attachments: Sigma 5 CPU Engineering Notes and Phase Sequence Charts:

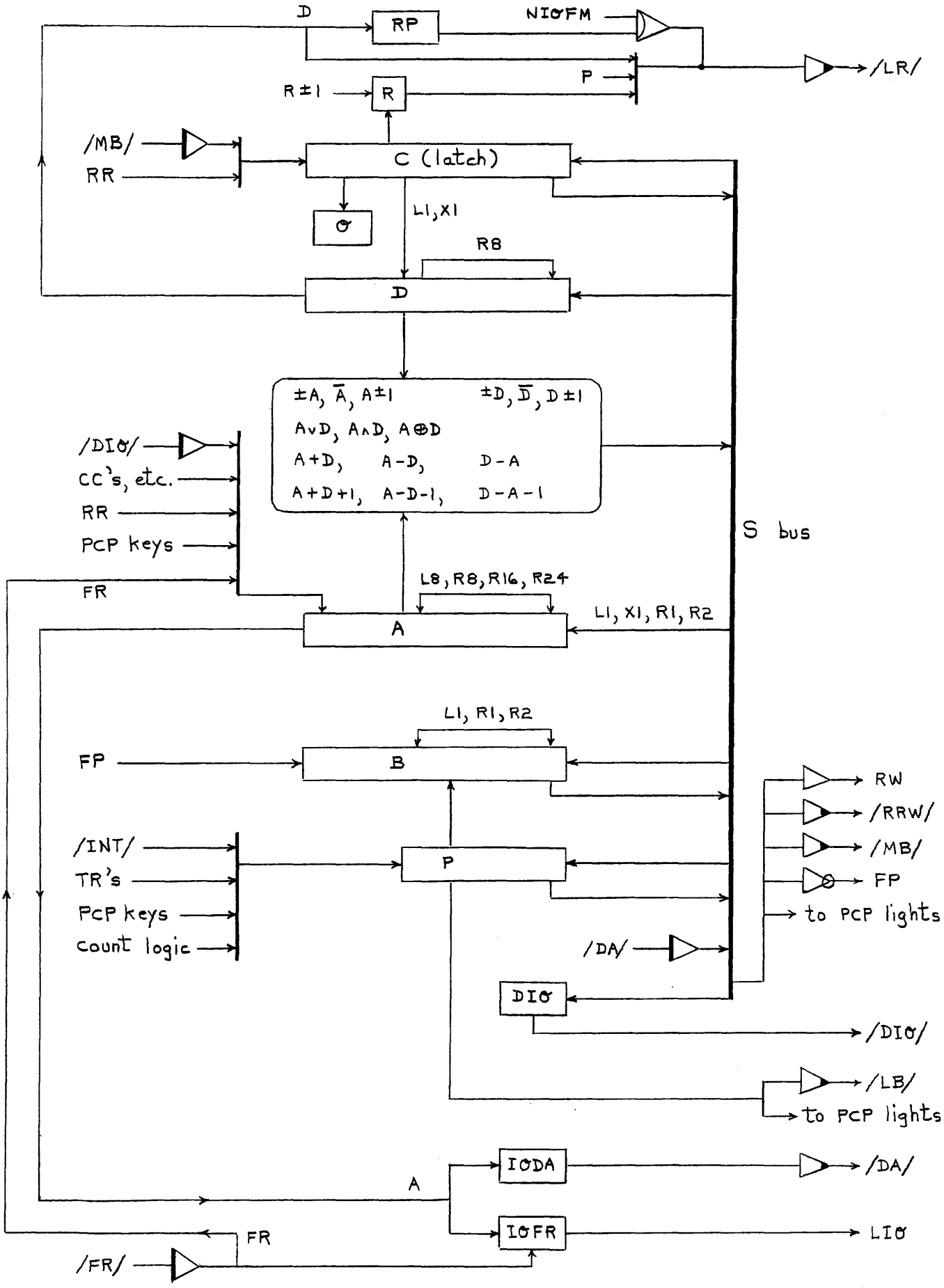
Prepared By: Evan Bell Date: 1/8/68

Approved: Evan T. Bell Date: 1/8/68
Product Support Engineer

Approved: Date: 1-8-68
Manager, Product Support Engineering

INDEX

SIGMA 5 INSTRUCTION PHASE SEQUENCE CHARTS


D
RP
NIOFM
/LR/
P
R±1
R
/MB/
RR
C (latch)
O
LI, XI
R8
D
±A, Ā, A±1
±D, D̄, D±1
AvD, A∧D, A⊕D
A+D, A-D, D-A
A+D+1, A-D-1, D-A-1
/DIO/
CC's, etc.
RR
PCP keys
FR
S bus
L8, R8, R16, R24
LI, XI, RI, R2
A
LI, RI, R2
FP
B
RW
/RRW/
/MB/
FP
to PCP lights
/INT/
TR's
PCP keys
count logic
P
/DA/
DIO
/DIO/
/LB/
to PCP lights
IODA
/DA/
A
FR
IOFR
LIO
/FR/

NOTES ON PCP PHASES

1. INTERRUPTS ARE INHIBITED DURING PCP PHASES BY DISABLING (S/INTRAP)

   (S/INTRAP) = N(PCP2.NKRUN).NPCPACT , WHERE PCPACT = PCP1+PCP3+PCP4+PCP5+PCP6

2. I/O IS NOT DISABLED DURING PCP2 BUT IS INHIBITED DURING PCPACT

   (S/IOEN) = IOFS.PCP2/1 + ---
   SCINH = PCPACT + ---

3. THE SIGNALS LISTED BELOW ARE TRUE WHEN THE SWITCHS INDICATED ARE ENERGIZED

   SWK1 — "CLEAR" (CPU RESET AND SYSTEM RESET) OR "LOAD"
   SWK2 — "INSERT PSW1" OR "INSERT PSW2"
   SWK3 — "STORE INSTR ADDR" OR "STORE SELECT ADDR"
   SWK4 — "INSTR ADDR INCREMENT" OR "DISPLAY INSTR ADDR" OR "DISPLAY SELECT ADDR"
   SWK5 — "DISPLAY SELECT ADDR" OR "STORE SELECT ADDR"
   SWK6 — "COMPUTE RUN" OR "COMPUTE STEP"
   SWK12 = SWK1 + SWK2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 10 OR PCP 6		BRPCP1 = N(INT.IEN).NFUEXU.NIOSC .HALT/1.PH10 S/PCP1 = BRPCP1 + PCP6	ENTERING PCP1 AT PH10 IS IHIBITED IF AN INTERRUPT REQ. IS PRESENT AND IEN IS TRUE
PCP 1	PRESET D→S FOR PCP2	S/SXD = PCP1 R/PCP1 = . S/PCP2 = PCP1 + RESET/KS	

PCP PHASES 1 of 6

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	NO CONTROL SW. ACTIVE ⇒ R/HALT FF	R/HALT = PCP2.NKAS/B	
	INHIBIT INTERRUPTS DURING IDLE	(S/INTRAP) = I. PCP2 .NKRUN	
	D → S → DISPLAY LIGHTS	S/SXD = PCP2/1.NRESET/C.NKCLRPSW/B .NIOCON	SXD - ALSO PRESET DURING PCP1
		PCP2/1 = PCP2 .NPCP3	
	COMPUTE STEP, RUN ⇒ PRESET CXS	(S/CXS) = PCP2/1.NIOCON. (KSTEP/B+KRUN/B)	
PCP 2	CLEAR PSW1 ⇒ 0 → PSW1	PSW1XS = KCLRPSW1.NIOCON.NKAS/B (R/CCn) = (R/CC) (R/CC) = CCXS/0 CCXS/0 = PSW1XS PXS = PSW1XS	S = 0's because S/SXD is inhibited. h = 1→4
IDLE PHASE	CLEAR PSW2 ⇒ 0 → PSW2	PSW2XS = KCLRPSW2.NIOCON.NKAS/B	
	CPU RESET ⇒ 020000000 → D	R/RPn = RPXS = PSW2XS DX = RESET S/D6 = RESET/C	n = 24 → 27
	0 → PSW1 0 → PSW2 25 → P SETBRP	PSW1XS = RESET PSW2XS = RESET S/Pn = RESET/C , PX = PSW1XS S/BRP = RESET/C	S = 0's , S/SXD INHIBITED n = 26, 29 ↔ 31
		R/PCP2 = PCP3 S/PCP3 = (PCP2/1.NIOCON.NDCSTOP) .(CLEARMEM + INT.KRUN + NHALT.KAS/1.KAS/2)	
	ANY CONTROL SWITCH (OTHER THAN RESET) ACTIVATED — GO TO PCP3		

PCP 2 of 6

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP 2 & 3	D → S		PRESET DURING PCP2
	0 → A	AX = PCP3	
	PRESET A → S FOR PCP 4	S/SXA = PCP3	
	COMPUTE STEP, RUN ⇒ S → C	CXS	PRESET DURING PCP2
	⇒ BRPH10	BRPH10 = PCP3. SWK6	
	DISPLAY SEL. ADDR. / STORE SEL. ADDR } P → B	BXP = PCP3. SWK5	
	INSERT PSW1 ⇒ PSW1 (except P) → A	AXPSW1 = KPSW1/B. PCP3	
	INSERT PSW2 ⇒ PSW2 → A	AXPSW2 = KPSW2/B. PCP3	
	CLEARMEM / INSERT PSW1,2 / LOAD } ⇒ S → B	BXS = PCP3. SWK12	
	NIOFS ⟹ RESET IOSC	R/IOSC = PCP3. NIOFS	
		R/PCP2 = . PCP3	
		R/PCP3 = .	
		S/PCP4 = PCP3. NBRPH10	

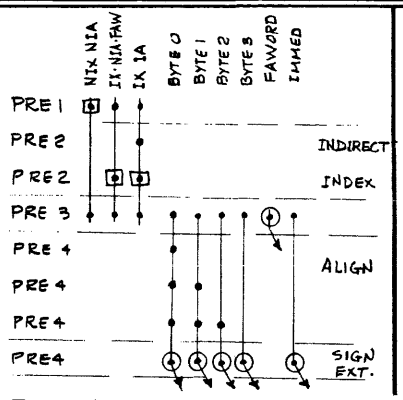
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP 4	PRESET FOR PCP5 – AVD → S	(S/SXAORD) = PCP4	
	DISPLAY SEL. ADDR. / STORE SEL. ADDR. } ADDR SW → P	PXK = PCP4. SWK5	
	STORE INST. ADDR / CLEARMEM / LOAD } PRESET S → MB	S/MBXS = PCP4. SWK1 + PCP4. SWK3	PRESET FOR PCP5
	DISPLAY INSTADDR / INST. ADDR INCR } ⇒ S/DRQ	S/MRQ/2 = PCP4. SWK4	
		S/DRQ = S/MBXS + S/MRQ/2 + ---	
	INSERT PSW1 / INSERT PSW2 / CLEAR DATA / CLEARMEM / LOAD } ⇒ S → D	DXS = PCP4. KCLEAR/B + PCP4. SWK12	
	INSERT PSW1 ⇒ P → S	SXP = PCP4. KPSW1/B. NDIS	
	INSERT PSW1 / INSERT PSW2 / ENTER DATA } DATA SW → A	AXK = PCP4. SWK2 + PCP4. KENTER/B	
	ENTER DATA	S/CXS = PCP4. KENTER/B	PRESET FOR PCP5 – TO SAVE NEW DATA IN C IF I/O GETS IN DURING IDLE.
	LOAD ⇒ 20 → P	S/P26 = PCP4. KFILL/B	
		PX = PCP4. KFILL/B	
	INST. ADDR INCR ⇒ P+1 → P	PUC31 = PCP4. KINCRE/B	
		R/PCP4 = .	
		S/PCP5 = PCP4 + BRPCP5	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP 5	AVD → S PRESET B → S FOR PCP6	PRESET IN PCP4 S/SXB = PCP5. NBRPCP5	
	INSERT PSW1 ⇒ S ↛ PSW1	PSW1XS = PCP5. KPSW1/B	
	INSERT PSW2 ⇒ S ↛ PSW2	PSW2XS = PCP5. KPSW2/B RPXS = PSW2XS	
	ENTER DATA ⇒ S ↛ D S ↛ C	DXS = PCP5. KENTER/B CXS	PRESET IN PCP4
	DISPLAY INST. ADDR DISPLAY SEL. ADDR INST ADDR INCRE } ⇒ C ↛ D	DXC = PCP5. SWK4	
	STORE INST. ADDR STORE SEL ADDR CLEAR MEM LOAD } ⇒ WRITE S → MB	MBXS = PRESET IN PCP4	
	CLEARMEM LOAD } ⇒ P+1 ↛ P S/DRQ BRPCP5	PUC31 = PCP5. SWK1 S/MBXS = PCP5. SWK1 BRPCP5 = PCP5. CLEARMEM + PCP5. N(P28. P31). KFILL/B S/SXA = BRPCP5	
	LOAD ⇒ LOAD ↛ A	AXLOAD = PCP5. KFILL/B S/MRQ/2 = BRPCP5	
		R/PCP5 = . S/PCP6 = PCP5. NBRPCP5	

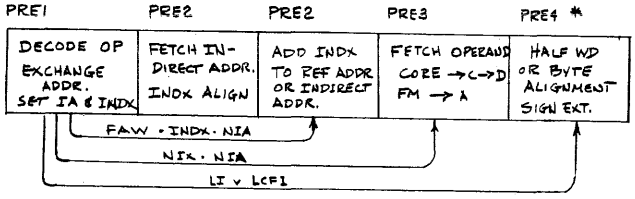
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PCP 6	B → S PRESET IN PCP5		
	S/HALT	S/HALT = PCP6	
	STORE SEL ADDR DISPLAY SEL ADDR } ⇒ S ↛ P	PXS = PCP6. SWK5	
	INSERT PSW1 INSERT PSW2 CLEARMEM LOAD } ⇒ S ↛ D	DXS = PCP6. SWK12	
	LOAD ⇒ 25 ↛ P	PX = PCP6. KFILL/B S/P26,29,31 = RESET/C	
	02000000 ↛ D	RESET/C = PCP6. KFILL/B S/D6 = RESET/C	
		R/PCP6 = . S/PCP1 = PCP6	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS

PRE



PREPARATION :
PERFORMS THOSE FUNCTIONS WHICH ARE GENERALLY COMMON TO ALL INSTRUCTIONS



▣ PRE/12 ⇒ END OF EFFECTIVE ADDR. COMPUTATION, PRE1 OR PRE2
⊙ PRE/34 ⇒ LAST PHASE OF PREPARATION

* PRE4 MAY LAST FOR 4 (T5L) CLOCKS

PREP. TIMING

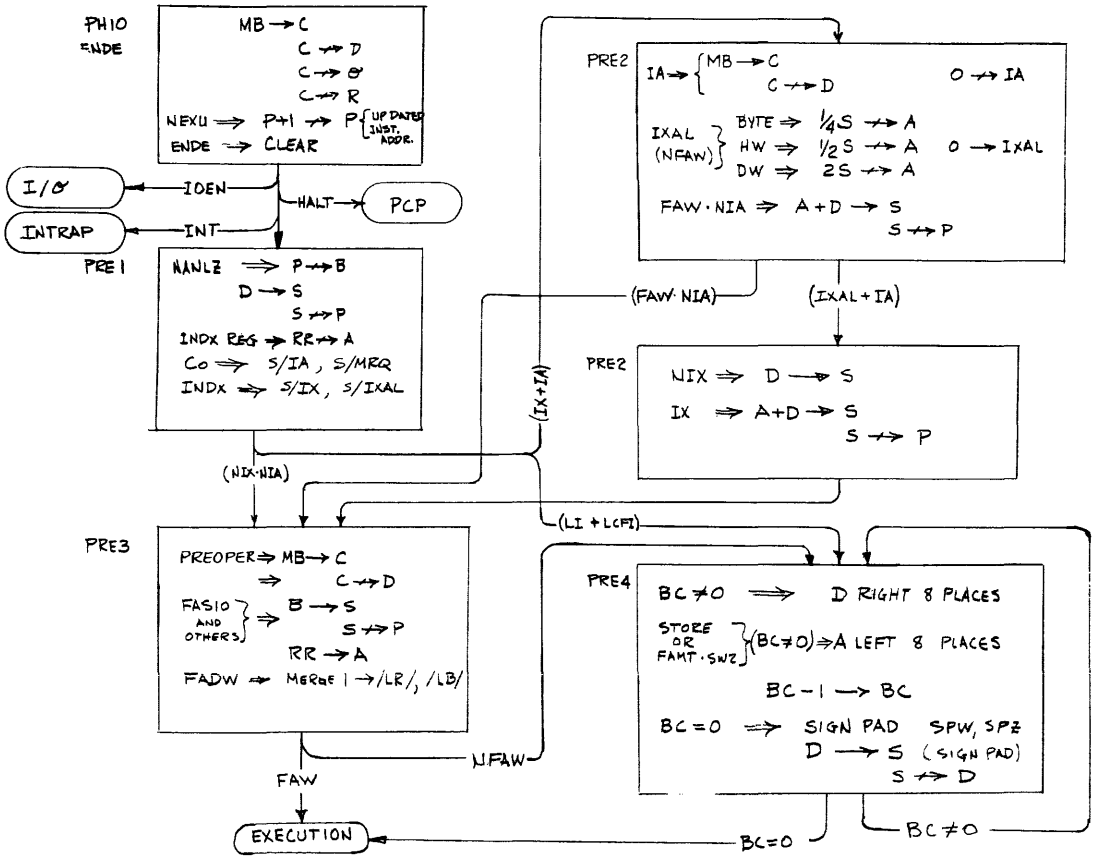
	PREP TIME +INCR
WORD · NIA · NIX CORE	1.12 μs
WORD · NIA · NIX · NO CORE	0.66 μs
LI OR LCFI	0.56 μs
IMM WITH FM OPER	0.94 μs
IA	+ 1.12
IY	+ 0.38
IA · IX	+ 1.22

FAMILY ADDR. SIGNALS

FABYTE ⇒ BYTE ADDR INST.
FAHW ⇒ HALF WORD INST.
FAIM ⇒ IMMEDIATE ADDR. INST.
FAW ⇒ WORD ADDR. INST
FADW ⇒ DOUBLE WORD ADDR. INST.
FAWORD ⇒ WORD OR DOUBLEWORD INST.

PREPARATION

1 of 13



PREP.

2 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 1 T5L	NANLZ ⇒ P → B; D → S	BxP/1 = BRP · PRE1	MOVE PROG. ADDR TO B. IF NOT IMMED INST. MOVE OPERAND REF ADDR TO P
		Sn = PRn	
	NFAIM ⇒ S → P	PXS = NFAIM · PRE 1	
	D12|4 → /LR/; RR → A	/LR/ = D12-14 · LRXD	X FIELD TO LR
		S/An = RRn · AXRR	INDX. INCR → A
	INDX ⇒ SET INDEX FLIP FLOP	S/IX = INDX · PRE1	
	INDX · NFAW ⇒ SET INDEX ALIGN	S/IXAL = INDX · PRE1 · NFAW	
	CO ⇒ {SET IA; CORE MEM REQ}	S/IA = CO · PRE1	REQUEST INDIRECT ADDR.
		S/MRQ/2 = CO · PRE1 · NFAIM	
	FAW · N(S/IA) · INDX ⇒ PRESET A+D → S	S/SXAPD = FAW · PRE1 · NCO · INDX	
	NFAW · INDX ⇒ PRESET A → S	S/SXA = (S/IXAL)	
	LI, LCFI ⇒ PRESET D → S	S/SXD = (LI + LCFI) · PRE1	
	→ SIGN EXTEND	SPIM = (LI + LCFI) · PRE1	
		S/SPW = SPIM · D12	
		S/SPZ = SPIM · ND12	
	LI, LCFI ⇒ BRANCH TO PRE4	BRPRE4 = (LI + LCFI) · PRE1 · NANLZ · NCO	
	INDX INDIRECT ⇒ BRANCH TO PRE2	BRPRE2 = INDX · PRE1 + CO · PRE1	
	ENABLE TRAP TO '40' ⇒ PRETR	S/PRETR = PRE1 · NANLZ	
	IDLE MODE ⇒ SET HALT	S/HALT = NKRUN · PRE1 · NFUEXU	

PREP. 3 of 13

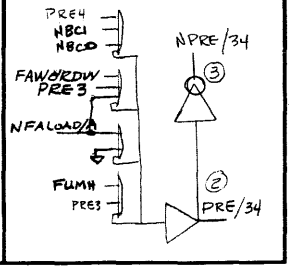
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 2	INDEX REG. ALIGN		PRE2 STATES
	IxAL ⇒ {0 → IXAL; A → S}	R/IXAL = 0	WORD INDEX ONLY
	BYTE ALIGN ⇒ ¼S → A	Sn = PRn	
	HALF WD ALIGN ⇒ ½S → A	AXSR2 = IXAL · PRE2 · FABYTE	
	DOUBLE WD. ALIGN ⇒ 2S → A	AXSR1 = IXAL · PRE2 · FAHW	
	S → P	AXSL1 = IXAL · PRE2 · FADW	
		PXS = PRE2	
	INDIRECT ADDR.		
	Clear IA ⇒ 0 → IA	R/IA = 0	
	IA ⇒ {MB → C; C → I}	Cn = MBn · CXMB	
		DXC = IA · PRE2	
	SUM BUS PRESETS		
	IA to BUS ⇒ D → S	S/SXD = IA · NIX · PRE2	
	IA PLUS INDX ⇒ A+D → S	S/SXAPD = IA · IX · PRE2 + IXAL · PRE2	
	RA PLUS INDX ⇒ A+D → S		
	SUSTAIN PRE 2	BRPRE2 = IXAL · PRE2 + IA · PRE2	
	NBRPRE2 ⇒ PRE/12	PRE/12 = NIA · NIXAL · PRE2	
	PRE/12 = (PRE1 + PRE2) · NBPPRE2		

PREP. 4 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE /12	PRE/12 ⇒ END OF ADDR VARIENT	PRE/12 = PRE1 · NINDX · NCO + PRE2 · NIA · NIXAL	
	PREOPER ⇒ SET MEM REQUEST	S/MRQ/2 = PREOPER · PRE/12	FETCH OPERAND PREOPER IS ONLY THOSE INST WHICH REQ. READING THE EFF. ADDR FROM MEMORY. PREOPER IS QUALIFIED WITH NANLZ
	FABRANCH ⇒ SET MEM REQUEST EXECUTE SET DATA RELEASE ENABLE FM READ ⇒ PRESET RR→A	S/MRQ/1 = FABRANCH · PRE/12 · NANLZ S/DRQ/2 = ØU6 · ØL7 · PRE/12 S/AXRR = PRE/12	
	AD, CD, STD, SD, CLR, S, FAMULNH, FASEL ⇒ PRESET MERGE 1 → /LR/	(S/LR31/1) = ØU1 · ØL0 + ØU1 · ØL1 + ØU1 · ØL5 + ØU1 · ØL8 + ØU3 · ØL9 + ØU2 · ØL5 (S/LR31/12) = FAMULNH + FASEL + (S/LR31/1) (S/LR31) = (S/LR31/12) · PRE/12	
	FADW EXCEPT FLOAT PT AND LAD ⇒ PRESET MERGE 1 → /LB/	(S/P31/1) = FADW · PRE/12 · N(Ø4 · Ø5) · N(ØU1 · ØLB)	
	NINDEX ⇒ D → S	Sn = PRn	
	INDEX ⇒ A+D → S FAIØ ⇒ S → D NFAIM ⇒ S → P	Sn = (PRn ⊕ Kn) SXADD DXS = FAIØ. PRE/12 PXS = NFAIM · PRE1 + PRE2	
	FADW ⇒ CRUSH S31	S31INH = FADW · PRE/12	
	SET T8L PRESET B → S	S/T8L = PRE/12 S/SXB = PRE/12. NFAIM	

PREF.

5 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 3 T8L OR DR	PROGRAM ADDR. BACK TO P B → S S ↛ P	Sn = Bn · SXB PXS = FAS10. NFAIM. PRE3. NANLZ + FUWAIT · PRE3 · NANLZ + FASEL · PRE3 · NØL7 · NANLZ + FUEXU · PRE3 · NANLZ + FARWD · PRE3 · NANLZ + FAMDS · NFAIM. PRE3. NANLZ	TRANSFER PROGRAM ADDR. BACK TO (P)
	NFAWORD ⇒ LOAD BYTE COUNTER	S/BC0 = NP32. PRE3. NPRE/34. ØU1 S/BC1 = NP33. PRE3. NPRE/34. ØU7	BYTE COUNT DECODE BC0 BC1 (BC=1) → 0 1
	NFAWORD ⇒ SET SIGN EXTENSION	S/SPW = SPIM · D12 + FAHW · C16 · P32 · PRE3 SPIM = FAIM · PRE3 S/SPZ = SPIM · ND12 · + FAHW · NC16 · P32 · PRE3 + FABYTE · P32 · P33 · PRE3	PRE/34
	NFAWORD · NFUMH ⇒ PRESET D → S BRANCH TO PRE4	S/SXD = PRE3 · BRPRE4 BRPRE4 = PRE3 · NPRE/34	
	FAST, FUMI, FASEL ⇒ PRESET S → C	S/CXS = FUMI · PRE3	

PREP.

6 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 3 CONT	PREOPER ⇒ { MB → C ; C → D }	Cn = MBn · CXMB DXC = PREOPER · PRE3	NFADW (C) : EL (D) : EL FADW/1 (C) : ELv1 (D) : ELv1
	UNCONDITIONAL EXCEPT: FAST/M, FAPSD, STFC } { RR → A ; READ INHIBIT }	S/An = RRn · AXRE · NAXRRINH AXRRINH = FASTORE · PRE3 · ØL4 + FAPSD · PRE3 + FAST/M · PRE3 + FARWD · ØLC · PRE3	
	STCF ⇒ CF → A2431	AXFC = FASTORE · PRE3 · ØL4	
	XPSD ⇒ PSW1 → A	AXPSW1 = FAPSD · PRE3	
	FAST/M · Ø6 ⇒ CC → A2831 ; CC → MC0407	AXS = ; AXCC = FAST/M · PRE3 · Ø6	← STACK MULTIPLE AND LM, STM
	FAST/M · NØ6 ⇒ 1 → MC	S/MC7 = FAST/M · PRE3 · NØ6	← STACK WORD
	FAIØ ⇒ 0 → /LR/	LRXZ = FUSIØ · PRE3	
	FADW/1, FAST, CLM ⇒ P−1 → P	PDC31 = FADW/1 · PRE3 + FAST · PRE3 + FACOMP/L · PRE3 · Ø41	FADW/1, FAST, CLM ⇒ (P) : EW.ADDR.
	FULAD ⇒ { SET MEM. REQ. ; P+1 → P }	S/MRQ/2 = FALOAD/A · ØU1 · PRE3 PUC31 = FULAD · PRE3	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 4 T5L	SIGN PAD ⇒ { D → S ; S → D ; S → C }	Sn = PRn * ; DXS = BCZ · PRE4. NFULAD ; S/CXS = FADIVH. PRE4	* SEE SIGN PAD LOGIC. PROPOGATES ARE CRUSHED (SPZ) OR ENABLED (SPW) UPWARD FROM SIGN POSITION
	FULAD { MB → C ; C → D ; (P-1) → P }	Cn = MBn . CXMB ; DXC = FULAD. PRE4 ; PDC31 = FULAD. PRE4	
	RIGHT ALIGN D ⇒ LOAD LEFT ALIGN A ⇒ STORE	DXDR8 = NBCZ. PRE4 AXAL8 = FASTORE. NBCZ. PRE4 + FAMT. SW2. NBCZ. PRE4	
	PRESET SIGN PAD PRESET D → S PRESET NEGITIVE SIGN PRESET POSITIVE SIGN	S/SXD = PRE4. (BC=1) S/SPW = FAHW. D08. (BC=1). PRE4 S/SPZ = FAHW. ND08. (BC=1). PRE4 + FABYTE. (BC=1). PRE4	
	DECREMENT BYTE COUNTER	BCDC1 = NBCZ. PRE4	
	COMPARE BYTE ⇒ 0 → A0023	AXZ/012 = FACOMP/1. ØU7. PRE4	
	BC ≠ 0 ⇒ SUSTAIN PRE4	BRPRE4 = PRE4 · NBCZ	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP END PHASE	ADDER PRESETS		PRE/34 IS THE LAST CLOCK DURING PREP. WHEN REG ALIGNMENT TAKES PLACE, PRE/34 IS COINCIDENT WITH PRE4 · (BC=0). ELSE PRE/34 IS COINCIDENT WITH PRE3.
	A + D → S	S/SXAPD = FAADD · (PRE/34 + PH2) + FAST/A · (PRE3 + PH1/F) + FAST/A · PH8	
PRE /34	A − D → S	S/SXAMD = FASUB · (PRE/34 + PH2)	
	D − A → S	S/SXDMA = FUPLM · (PRE3 + PH1/F) + FUPLM · PH8	
	A → S	S/SXA = FASTORE · PRE/34 + FARWD · (PRE/34 + PH2) · NRZ + FAMT · PRE/34 + FAPSD · PRE3 + FUMMC · PRE3	
	D → S	S/SXD = FUINT · PRE3 + FALCEP · PRE/34 + FALOAD · (PRE/34 + PH2) + FUXW · PRE3	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP END PHASE	ADDER PRESET CONT.		
	D − 1 → S	S/SXDM1 = FUPLW · (PRE3 + PH1/F + PH8)	
PRE /34	D + 1 → S	S/SXDP1 = FUPSW · (PRE3 + PH1/F + PH8)	
	A + 1 → S	S/SXAP1 = FUBIR · PRE3	
	A − 1 → S	S/SXAM1 = FUBDR · PRE3	
	− D → S	S/SXMD = FALOAD/C · (PRE/34 + PH2)	
	A ∧ D → S	S/PRXAD = FASEL · PRE3 · NBL7 + BU4 · BLB · PRE3	
	NA ∧ D → S	S/PRXNAD = FASEL · PRE3 · BL7	
	A ⊕ D → S	S/SXAEORD = BU... BL8 · PRE3	
	A ∨ D → S	S/SXAORD = BU4 · BL9 · PRE3	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP END PHASE	CORE READ/WRITE	(S MRQ) = [(S/MRQ/1) + (S/MRQ/2) + (S/MRQ/3) + (S/MBXS)] • N(ANLZ · PRE3)	
	SET MEMORY REQUEST	(S/MRQ/1) = FUINT · PRE3 + FUWAIT · PRE3 + FAS10 · PRE3	
PRE /34	SET MEM. REQ AND DATA REL.	(S/MRQ/2) = FAPSD · (PRE/34 + PH2) + (FAST/M · PRE3 · N0U0) · 0LA + (S/MBXS)	
	SET MEMORY REQUEST DATA REL. DELAYED	(S/MRQ/3) = FADW/1 · (PRE/34 + PH2) + FACOMP/L · 0U1 · PRE3	ALL MEMORY REQUESTS ARE INHIBITED IF (ANLZ · PRE3)
	SET WRITE MEM. REQ.	(S/MBXS) = FASTORE · PRE/34 + FAMT · SW2 · PRE/34 + FAPSD · PRE3 · 07	
	PRESET B → S	S/SXB = FUBAL. PRE3 + FALOAD/A · (PRE/34 + PH2) + FAPSD · PRE3	

PREP.

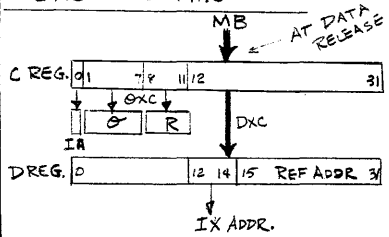
11 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP END PHASE	FAST MEMORY READ	S/AXRR = FUSF.D23 · PRE3 + FUDW. NR31 · PRE3 + FASTORE · PRE/34 · N02 + (FAST/M · PRE3 · N0U0) · N0LA + FUMMC · PRE3 + FUS · PRE3	
	PRESET RR → A		
PRE /34	MERGE 1 → LR31	S/LR31 = FADW/1 · PRE3 + FUMMC · PRE3	
	FAST MEMORY WRITE		
	PRESET S → RW	S/RW = FUXW · PRE3 · NANLZ + FAS11 · N0L1 · (PRE/34 + PH2) + FUBAL · PRE3 · NANLZ + FUBDR · PRE3 · NANLZ + FUBIR · PRE3 · NANLZ	
	FAST → SET SW8	S/SW8 = FAST PRE3	SW8 IS PHI SEQUENCE FLIP FLOPS
	FUMSP → SET SW7	S/SW7 = FAST · PRE3 · N04 + FULAWORDW · ND0. PRE/34 + FALOAD/A · 0U5 · ND16 · PRE/34	SW 7 INDICATES SECOND PASS TO FAST SEQUENCE AND SIGN OF LOAD ABSOLUTE.

PREP.

12 of 13

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP END PHASE	(STM) ⇒ P−1 → P	PDC31 = (FAST/M · PRE3 · NØUØ) · NOLA	
	(LM) ⇒ R−1 → R	RDC31 = (FAST/M · PRE3 · NØUØ) · ØLA	
PRE /34	TIMING SELECT FLIP FLOPS T/L	S/T1L = FACOMP/1 · (PRE/34 + PH2) + FAST · PRE3 + (ANLZ · PRE3) + FACOMP/L · (PRE/34 + PH2)	
	BRANCH LOGIC: (FADIV) ⇒ SET PH3	BRPH3 = FUSF · PRE3 · NANLZ · D23	
	(FAMUL) ⇒ SET PH3	+ FAMDS · PRE/34 · NBRPH5 · NANLZ	
	(LPSD) ⇒ SET PH3	+ FAPSD · PRE3 · NANLZ · N87	
	(ANLZ) ⇒ SET PH5	BRPH5 = (ANLZ · PRE3)	
	(FUS) ⇒ SET PH5	+ FUS · PRE3	
	(FUSF · ND23) ⇒ SET PH5	+ FUSF · PRE3 · ND23 ·	
	(LM V STM) ⇒ SET PH6	BRPH6 = (FAST/M · PRE3 · NØUØ), NANLZ	
	FUSF · D23 ⇒ SET PH3		
	(FAMT · SW0) ⇒ SET PH8	BRPH8 = FAMT · SW2 · PRE/34	
	(EXU) ⇒ SET PH10	BRPH10 = FUEXU · PRE3 · NANLZ	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 10	END OF EXECUTION ⇒ ENDE	ENDE = EXC · PH10	ENDE IS PHASE 10 FOR ALL INST.
	MB → C	Cn = MBn · CxMB	EXC IS SET WITH PRE1 AND INDICATES THAT INST. EXECUTION PRECEDED THIS PH10.
	C → D	DXC = PH10	
	$C_{1-7} \rightarrow \sigma$, $C_{8-11} \rightarrow R$ →	σXC = PH10	
T5L OR DR			① FUNCTIONS WHICH PROG ADDR IS NOT UPDATE ARE (A.) EXECUTE INST (B.) IOSERVICE CALL (C.) INTERRUPT (D.) HALT (E) P HOLD
	UPDATE PROGRAM ADDR.		
	INCR. PROG. ADDR IF NOT AS NOTED ① → P + 1 → P	PUC31 = N(FUEXU.END). NIOSC . PH10. NHALT. NINT. NKAHOLD	
	DECR PROG. ADDR IF INST. IS TO BE REPEATED ② → P - 1 → P	PDC31 = FUEXU · (INT + IOSC) · ENDE + FUMMC · NMCZ · ENDE · (INT + IOSC)	② INST. IS REPEATED IF MMC OF EXU CHAIN WAS TERMINATED BY INTERRUPT OR IO SERVICE CALL
	REFERANCE ADDR. ⇒ PRESET D→S	S/SXD = PH10	

ENDE

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 10	IOENABLE ⇒ IOEN	S/IOEN = IOSC · PH10 · NIOINH	
	INTERRUPT ENABLE ⇒ IEN	IEN = KRUN · PH10 · NIOSC	
	SET ARITH OVERFLOW TRAP	S/TRAP = AM. CC2 · PH10 · OVERIND	
	INDEX ⇒ PRESET $D_{08-11} \rightarrow$ /LR/; PRESET RR → A	S/LRXD = σXC	
		S/AXRR = PH10	
	ENABLE PRE1 ⇒ PRE1EN	PRE1EN = N(S/TRAP) · N(S/INTRAP) · NIOSC · NHALT	
		S/PRE1 = PRE1EN · PH10	
	CLEAR & RESET/A	CLEAR = PH10 + RESET/B	
		RESET/A = CLEAR + ...	
	CLEAR ⇒ ZERO → PHASE F/F, NBR, SWn, MCn, ANLZ, BCn, DIOTn, EXC, IX, OVERIND, DIOWD, DIOIND, INTRAP		

ENDE

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT END OF PREPARATION NFADW/1 ⇒ (D) : EW " ⇒ (P) : PROG. ADDR. FADW/1 ⇒ (D) : EW v1 " ⇒ (P) : Eff. ADDR. " ⇒ (B) : PROG ADDR. LOAD RELATED EVENTS DURING PREP. LI, LCFI ⇒ { SIGN PAD IMMED.; PRESET D→S; BRANCH TO PRE4 FALOAD, FALCFP ⇒ PRESET D → S FALOAD/C ⇒ PRESET −D→S FAS10, FADW/1 ⇒ SET MEMORY REQUEST FAS11 ⇒ SET FAST MEM WRITE FADW/1 ⇒ MERGE1 → LR	INSTRUCTION FAMILIES FALOAD: LI, LB, LH, LW, LD FALOAD/C: LCH, LCW, LCD FALCFP : LCFI, LCF, LRP FALCF : LCFI, LCF FADW/1 : LD, LCD; ALSO: AD, SD, CD FAS10 : LI, LB, LH, LW, LCF LCFI, LRP LCH, LCW, plus others FAS11 : LI, LB, LH, LW, LD LCH, LCW, LCD SPIM = (FULI + FULCFI) · PRE1 S/SXD = (FULI + FULCFI) · PRE1 BRPRE4 = (FULI + FULCFI) · PRE1 · NANL2 S/SXD = FALOAD · (PRE/34 + PH2) + FALCFP · PRE/34 S/SXMD = FALOAD/C · (PRE/34 + PH2) S/MRQ/1 = FAS10 · PRE/34 S/MRQ/3 = FADW/1 · (PRE/34 + PH2) S/RW = FAS11 · (PRE/34 + PH2) · NOL1 S/LR31 = FADW/1 · (NANL2 · PRE3)	INST DESCRIBED BY THIS SEQUENCE THESE FAMILIES INCLUDE SEVERAL TYPES OF INSTRUCTIONS PRE/34 IS LAST PHASE OF PREPARATION FAS10: FETCH NEXT INST FADW/1: FETCH EW

"FALOAD": LI (22), LB (72), LH (52), LW (32), LD (12), LCH (5A), LCW (3A), LCD (1A), LCFI (02), LCF (70), LRP (2F)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1	FALOAD, FALCFP ⇒ D → S; S → RR FALOAD/C ⇒ −D → S FALCF ⇒ { S2427 → CC; S2931 → FC FULRP ⇒ S2327 → RP FAS11 ⇒ SET CC3 & CC4 LOAD POSITIVE ⇒ SET CC3 LOAD NEGITIVE ⇒ SET CC4 FADW/1 { PRESET B→S; PRESET RR→A FULCD ⇒ HOLD END CARRY FAS10 BRANCH TO END	Sn = PRn S/RRn = Sn · RW CCxS/3 = FALCF. PH1 . R30 FCXS = FALCF. PH1. R31 RPXS = FULRP. PH1 TESTS = FAS11 . (PH1 + PH3) S/CC3 = SGTE . TESTS S/CC4 = So . TESTS S/SXB = FADW/1. PH1 S/AXRR = FADW/1 · PH1 KOOHOLD = FADW/1 · PH1 S/SWO/NZ = KOOHOLD BRPH10 = FAS10 . PH1	SGTE: SUM GREATER THAN ZERO
PH 2	RR → A MB → C C → D B → S S → P	S/An = RRn . AXRR Cn = MBn · CXMB DXC = FADW/1. PH2 Sn = Bn . SXB PXS = FADW/1. PH2	
DR	PRESETS FULD ⇒ PRESET D→S FULCD ⇒ PRESET −D→S FADW/1 ⇒ {PRESET FM WRITE; MEM REQ	S/SXD = FALOAD · (PRE/34 + PH2) S/SXMD = FALOAD/C · (PRE/34 + PH2) S/RW = FAS11 · (PRE/34 + PH2) S/MRQ/3 = FADW/1 · PH2	

FALOAD

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH3	±D ⟶ S S ⟶ RR	$S_n = PR_n$ $S/RR_n = S_n \cdot RW$	
T8L	LD, LCD ⇒ SET CC3, CC4 VIA TESTS AND S3263Z LOAD POSITIVE ⇒ SET CC3 LOAD NEGITIVE ⇒ SET CC4 BRANCH TO ENDE	TESTS = FAS11 · (PH1 + PH3) S3263Z = NSW0 · NTESTS S/CC3 = SGTZ · TESTS S/CC4 = SO . TESTS BRPH10 = FADW/1. PH3	
PH10 DR	NORMAL ENDE		

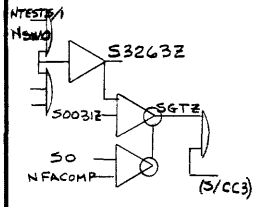
FALOAD 3 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREPARATION LAD ⇒ (D) : EW1 FALOAD/A · NOU1 ⇒ (D) : EW FALOAD/A ⇒ (A): RR; (P): EW ADDR.; (B): PROG. ADDR. FALOAD/A ⇒ PRESET B ⟶ S POS SIGN ⇒ SET SW7	INSTRUCTION FAMILY FALOAD/A: LAH, LAW, LAD FULAWORDW: LAW OR LAD S/SXB = FALOAD/A · (PRE/34 + PH2) S/SW7 = FULAWORDW · ND0 · PRE/34 + FALOAD/A. OU5. ND16. PRE/34	FALOAD/A · NOU1 ⇒ LAH, LAW RR⟶A IS AUTO FUNCTION AND CONTENTS OF RR ARE NOT USED BY THIS SEQUENCE.
PH1 T5L	B ⟶ S FALOAD/A · NOU1 ⇒ S ⟶ P FALOAD/A ⇒ SET CORE MEM REQ. ⇒ SET FM WRITE POS. SIGN ⇒ SET D ⟶ S NEG. SIGN ⇒ SET −D ⟶ S MERGE 1 ⟶ /LR31/	$S_n = B_n \cdot SXB$ PXS = FALOAD/A · NOU1 · PH1 S/MRQ/3 = FALOAD/A · (PH1 + PH3) S/RW = FALOAD/A · (PH1 + PH3) S/SXD = FALOAD/A · (PH1 + PH3) · SW7 S/SXMD = FALOAD/A · (PH1 + PH3) · NSW7 S/LR31 = FULAD · PH1	S/MRQ/3 ⇒ CORE REQ WITH DATA RELEASE AUTO. SET ON FOLLOWING CLOCK. LAD: FETCH EW0 NLAD: FETCH NEXT INST

"FALOAD/A" (Load Absolute): LAH (5B), LAW (3B), LAD (1B) 1 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2	±D → S S →→ RR	$S_n = PR_n$ S/RRn = $S_n \cdot Rw$	LOAD EWO
T8L	FALOAD/A ⇒ PRESET B → S SET CC3 AND CC4 VIA TESTS SUM ≠ 0 ⇒ SET SWO END CARRY HOLD OVERFLOW ⇒ SET CC2 N(LAD) ⇒ BRANCH TO PH10	S/SxB = FALOAD/A · (PRE/34 + PH2) TESTS = FALOAD/A · PH2 S/SWO = K00HOLD · NS0031Z K00HOLD = FALOAD/A · PH2 S/FL3 = K00HOLD · K00 PROBOVER = FALOAD/A · PH2 · N01 BRPH10 = FALOAD/A · PH2 · N0U1	NO OVFL INDICATION FOR LAH INST
PH3	B → S S →→ P LAD ⇒ {SET CORE MEM. REQ SET FM WRITE	$S_n = B_n \cdot SxB$ PxS = FALOAD/A · PH3 S/MRQ/3 = FALOAD/A · (PH1 + PH3) S/RW = FALOAD/A · (PH1 + PH3)	LAD – ONLY FETCH NEXT INST
DR	MB → C C →→ D POS. SIGN ⇒ PRESET D → S NEG. SIGN ⇒ PRESET –D → S FL3 ⇒ PRESET K31	$C_n = MB_n \cdot MBxC$ DxC = FALOAD/A · PH3 S/SxD = FALOAD/A · (PH1 + PH3) · NDO S/SxMD = FALOAD/A · (PH1 + PH3) · DO S/K31 = FALOAD/A · PH3 · FL3	

FALOAD/A 2 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4	±D → S S →→ RR	$S_n = PR_n \cdot$ S/RRn = $S_n \cdot Rw$	
T8L	SET CC3 AND CC4 VIA TESTS LOAD ≠ 0 ⇒ set CC3 SUM GREATER THAN ZERO OVERFLOW ⇒ SET CC2 BRANCH TO ENDE	TESTS = FALOAD/A · PH4 S/CC3 = SGTZ · TESTS S3263Z = NTESTS/1 · NSWO SGTZ = NS0063 · NS0 PROBOVER = FALOAD/A · PH4 S/OVERIND = PROBOVER S/CC2 = PROBOVER · (S00 ⊕ S0) BRPH10 = FALOAD/A · PH4	S0063 = 0 CC3 = 0 CC4 = 0 S0063 > 0 CC3 = 1 CC4 = 0 
PH10 (ENDE) DR	NORMAL ENDE	S/TRAP = ENDE · CC2 · AM · OVERIND	

FALOAD/A 3 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREPARATION (D) : EW (A) : RR (P) : PROG. ADDRESS AND ⇒ PRESET A∧D → S OR ⇒ PRESET A∨D → S EOR ⇒ PRESET A⊕D → S FAS10 ⇒ SET CORE MEM. REQ. FAS11 ⇒ SET FM WRITE	INSTRUCTION FAMILIES FAS10 : AND, OR, EOR, PLUS OTHERS FAS11 : AND, OR, EOR, PLUS OTHER ØU4·ØL8 : EOR ØU4·ØL9 : OR } FALOGIC ØU4·ØLB : AND S/PRXAD = ØU4·ØLB·PRE3 S/SXAORD = ØU4·ØL9·PRE3 S/SXAEORD = ØU4·ØL8·PRE3 S/MRQ/1 = FAS10·PRE/34 S/RW = FAS11·(PRE/34 + PH2)·NØL1	} THESE FAMILIES INCLUDE SEVERAL TYPES OF INSTRUCTIONS FETCH NEXT INST.

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

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1 T8L	AND ⇒ A∧D → S OR ⇒ A∨D → S EOR ⇒ A⊕D → S S → RR FAS11 ⇒ SET CC1 & CC2 VIA TESTS FAS10 ⇒ BRANCH TO ENDE	} Sn = PRn S/RRn = Sn · RW TESTS = FAS11·(PH1+PH3) S/CC3 = SGTZ. TESTS S/CC4 = SO · TESTS BRPH10 = FAS10·PH1	WHERE n IS DEFINED AS EACH OF 32 BITS SEE FAARITH, PH3 SEQUENCE FOR TESTS DETAILS
PH 10 ENDE DR	NORMAL ENDE		

OR, EOR, AND

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREVIOUS INSTRUCTION NFADW/1 → (D) : EW (A) : RR003! (P) : PROG ADDR FADW/1 → (D) : EWv1 (A) : RRv! (P) : EW ADDR. (B) : PROG ADDR. EVENTS DURING PREP. FAADD ⇒ PRESET A + D → S FASUB ⇒ PRESET A - D → S FAS10 / FADW/1 ⇒ SET MEMORY REQUEST FAS11 ⇒ SET FAST MEMORY WRITE FADW/1 ⇒ MERGE 1 → LR	INSTRUCTION FAMILIES FAARITH : AD, AI, AW, AH, SD, SW, SH FAADD : AD, AI, AW, AH, AWM FASUB : SD, SW, SH, CD, CI, CW, CH, CB CLM, CLR FAS10 : AW, AI, AH, SW, SH, PLUS OTHERS FAS11 : AD, AW, AI, AH, SD SW, SH PLUS OTHERS FADW/1 : AD, SD ALSO LD, LCD, CD S/SXAPD = FAADD · (PRE/34 + PH2) S/SXAMD = FASUB · (PRE/34 + PH2) S/MRQ/1 = FAS10 · PRE/34 S/MRQ/3 = FADW/1 · (PRE/34 + PH2) S/RW = FAS11 · (PRE/34 + PH2) · NOL1 S/LR31 = FADW/1 · (NANLZ · PRE3)	← INST(S) COVERED BY THIS SEQUENCE THESE FAMILIES INCLUDE SEVERAL TYPES OF INSTRUCTIONS, SEE COMBINED LISTS P. PRE/34: LAST PHASE OF PREP PH2: REPEAT FOR FADW/1 FAS10: FETCH NEXT INST. FADW/1: FETCH EW

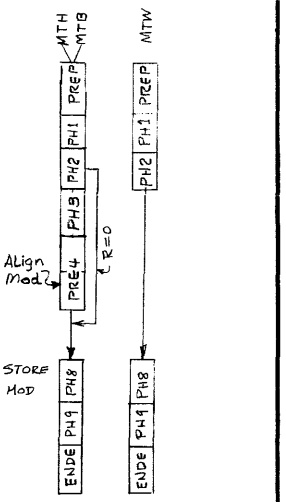
"FAARITH": AI(20), AH(50), AW(30), AD(10), SH(58), SW(38), SD(18)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1	FAADD ⇒ A + D → S FASUB ⇒ A - D → S S → RR FAS11 ⇒ PROBE CC1 & CC2 VIA TESTS FAARITH ⇒ PROBEOVERFLOW SET CC1 IF END CARRY	Sn = (PRn ⊕ Kn) · SXADD SXADD = GXAD + GXNAD + GXAND + L31 RRn = Sn · Rw TESTS = FAS11 · (PH1 + PH3) PROBEOVER = FAARITH · (PH1 + PH3) CC1XK00 = FAARITH · (PH1 + PH3)	WHERE n is defined as each of 32 bits FADW/1 ⇒ S → RRv1 SEE PH3 OF THIS SEQUENCE FOR TESTS, OVERFLOW, AND END CARRY DETAILS FOR DOUBLEWORD ADD MOST SIGNIFICANT HALF.
TIL	FADW/1 ⇒ PRESET B → S PRESET RR → A SAVE S ≠ 0 HOLD END CARRY FAS10 ⇒ BRANCH TO ENDE END CARRY ⇒ SET CC1 OVERFLOW ⇒ SET CC2 POSITIVE RESULT ⇒ SET CC3 NEGITIVE RESULT ⇒ SET CC4	S/SXB = (FADW/1 · PH1) S/AXRR = (FADW/1 · PH1) S/SWO/NZ = K00HOLD · NS00317 S/FL3 = K00HOLD · K00 K00HOLD = FADW/1 · PH1 + FALOAD/A · PH2 BRPH10 = FAS10 · PH1 S/CC1 = CC1XK00 S/CC2 = PROBOVER. (S00 ⊕ S0) S/OVERIND = PROBOVER S/CC3 = SGTZ. TESTS S/CC4 = S0. TESTS	

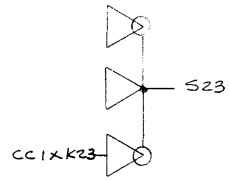
FAARITH

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2	FADW/1 ⇒ B → S	$S_n = B_n \cdot SXB$	FADW/1 ONLY
	S → P	$PXS = FADW/1 \cdot PH2$	XFER PROG ADDR TO P REG.
	MB → C	$C_n = MB_n \cdot CXMB$	
	C → D	$DXC = FADW/1 \cdot PH2$	
	RR → A	$S/An = RR_n \cdot AXRR$	
DR	FAADD ⇒ PRESET A+D → S	$S/SXAPD = FAADD \cdot (PRE/34 + PH2)$	
	FASUB ⇒ PRESET A−D → S	$S/SXAMD = FASUB \cdot (PRE/34 + PH2)$	
	FADW/1 ⇒ SET CORE MEM. REQ.	$S/MRQ/3 = FADW/1 \cdot (PRE/34 + PH2)$	FETCH NEXT INST
	DOUBLE PRECISION ADD OR SUBTRACT END CARRY GOES TO LSB CARRY K00 [PH2] → K31 [PH3]	$S/K31 = FADW/1 \cdot PH2 \cdot FL3$	K31 IS BUILT UPSIDE DOWN
	FAS11 ⇒ PRESET WRITE TO FM	$S/RW = FAS11 \cdot (PRE/34 + PH2) \cdot NOLI$	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3	FAADD ⇒ A+D → S	$S_n = (PR_n \oplus K_n) \cdot SXADD$	MOST SIGNIFICANT HALF OF DOUBLEWORD ADD.
	FASUB ⇒ A−D → S	$SXADD = GXAD + GXNAD + GXAND + K31$	
	S → RR	$RR_n = S_n \cdot Rw$	
	PROBE CC3 & CC4 WITH TESTS	TESTS = FAS11 · (PH1 + PH3)	
TIL	NSWO ⇒ S3263 = 0	S3263Z = NTESTS/1 · NSWO	SWO (S3263 ≠ 0) INTO SGTZ (SUM GREATER THAN ZERO)
		SGTZ = (S0063 ≠ 0) · NS0 · NFACOMP	
	POSITIVE RESULT ⇒ SET CC3	S/CC3 = SGTZ · TESTS	CONDITION CODES SET IN PHASE 1 ARE OVERRIDEN BY R/CC DURING PH3
	NEGITIVE RESULT ⇒ SET CC4	S/CC4 = S0 · TESTS	
	PROBE OVER FLOW	PROBEOVER = FAARITH · (PH1 + PH3)	
		S/CC2 = PROBEOVER · (S00 ⊕ S0)	
		S/OVERIND = PROBEOVER	
	SET CC1 IF END CARRY	CC1XK00 = FAARITH · (PH1 + PH3)	
		S/CC1 = CC1XK00 · K00	
	BRANCH TO ENDE	BRPH10 = FADW/1 · PH3	
ENDE	AM (ARITH. MASK) ⇒ TRAP TO '43' ON OVERFLOW	S/TRAP = ENDE · CC2 · AM · OVERIND	
		S/TR30 = ENDE. CC2. AM. OVERIND	
		S/TR31 = ENDE CC2. AM. OVERIND	
DR	NORMAL ENDE		

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE	CONTENTS OF REGISTERS (A) : RR (NOT USED) (D) : EFF. LOCATION (B) : PROG. ADDR. (P) EFF. ADDR. PRESET A → S	S/SXA = FAMT · PRE/34	FAMT MTW (33) MTH (53) MTB (73)
PH 1 T5L	(R≠0) ⇒ SET SW2 R POSITIVE (NR28) ⇒ R ↛ A R NEGITIVE (R28) ⇒ NR ↛ A R > 0 ⇒ PRESET A+D → S R < 0 ⇒ PRESET D-A-1 → S R = 0 ⇒ PRESET D → S INTRAP ⇒ SET INTERRUPT CLOCK ENABLE NINTRAP ⇒ SET T1IL TIMING	S/SW2 = FAMT·PH1·NRZ AXR = FAMT·PH1·NR28 AXNR = FAMT·PH1·R28 S/SXAPD = FAMT·PH1·NR28·NRZ S/SXDMAM1 = FAMT·PH1·R28 S/SXD = FAMT·PH1·RZ S/CEINT = FAMT·PH1·INTRAP S/T1IL = FAMT·PH1·NINTRAP	

"FAMT": MTB (73), MTH (53), MTW (33)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2 TIME SEE *	MTH, MTB · (R≠0) ⇒ LOAD BYTE COUNTER R>0 ⇒ A+D → S R<0 ⇒ D-A-1 → S R=0 ⇒ D → S S → A NINTRAP: SET CC3 & CC4 VIA TESTS RESULT POS ⇒ SET CC3 NEG ⇒ SET CC4 END CARRY ⇒ SET CC1 BYTE END CARRY FUMTB ⇒ SET FLAG3 OVERFLOW ⇒ SET CC2 HALF WD OVFL MTB ⇒ CRUSH S23 PHASE 2 CONTINUED	S/BC0 = FAMT. PH2. Ø1. NP32. NRZ S/BC1 = FAMT. PH2. Ø4. NP33. NRZ BCX = FAMT. PH2. Ø1. NRZ Sn = (Kn ⊕ PRn). SXADD + PRn AXS = FAMT. PH2 TESTS = FAMT. PH2. NINTRAP SGTZ = (S0063≠0). NSO. NFACOMP S/CC3 = TESTS. SGTZ S/CC4 = TESTS. S0 CC1XK00 = FAMT. PH2. NINTRAP CC1XK23 = FAMT. PH2. NINTRAP. Ø4 S/FL3 = CC1XK23 PROBOVER = FAMT·PH2. NINTRAP PROBOVER/H = FAMT. PH2. NINTRAP. Ø5 OVERIND = PROBOVER S/CC2 = PROBOVER. (S00 ⊕ S0) + PROBOVER/H. (S15 ⊕ S16) S23 = I. CC1XK23	* TIMING NINTRAP, R≠0 = T8L R=0 = T5L INTRAP = T5L 

FAMT

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2	INTRAP ⇒ { EXIT HIGHEST PRIORITY; ARM HIGHEST PRIORITY; TRIGGER COUNT ZERO }	LEVACT = FAMT. PH2. INTRAP LEVARM = FAMT. PH2. INTRAP CNTZERO = FAMT. PH2. INTRAP. S0031Z	
	PRESET A → S (R≠0) . MTW ⇒ SET CORE WRITE { R=0 OR MTW } ⇒ BRANCH TO PH8	S/SXA = FAMT · (PRE/34 + PH2) S/MBXS = FAMT. PH2 N01. NRZ BRPH8 = FAMT · PH2 · (RZ + N01)	
PH 3	{ MTH. OVFL. NINTRAP OR MTB . NINTRAP } ADJUST SIGN	FUMTSIGN = FAMT. PH3. NINTRAP · (CC2 + N0U5)	
	MTH. OVFL ⇒ EXCHANGE CC3 ↔ CC4	FUMTOVER = FUMTSIGN · NFL3 S/CC3 = FUMTOVER . CC4 S/CC4 = FUMTOVER . CC3	
T5L	MTB ⇒ TEST BYTE = 0	S/CC3 = FUMTSIGN . FL3. NS1631Z	S1623 WILL BE ZERO DUE TO PH2 (CRUSH S23)
	FUMTSIGN ⇒ CC3 & CC4 RESET	R/CC3 = FUMTSIGN R/CC4 = FUMTSIGN	
	BRANCH TO PRE4 FOR PREWRITE ALIGN	BRPRE4 = FAMT . PH3	

FAMT 3 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 4	LEFT ALIGN A	AXAL8 = FAMT · SW2 · NBCZ · PRE4	
T5L	BRANCH TO PH8 SET A → S SET CORE MEM. WRITE BC ≠ 0 SUSTAIN PH4	BRPH8 = FAMT · SW2 · PRE/34 S/SXA = FAMT · PRE/34 S/MBXS = FAMT · SW2 · PRE/34 BPPRE4 = PRE4 . NBCZ	
PH 8 DR	A → S S → MB	Sn = PRn MBn = Sn · MBXS	
PH 9 T5L	B → S → P SET CORE MEM REQ RESET INTRAP	PXSXB = NFAFL · NFAMDS · PH9 S/MREQ/2 = PXSXB R/INTRAP = FAMT · PH9	
PH 10 ENDE	NORMAL ENDE		

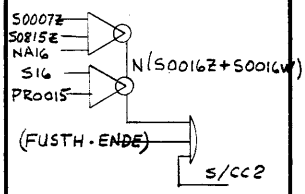
FAMT 4 of 4

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREPARATION STCF ⇒ (A) : BYTE ALIGNED CF STB ⇒ (A) : BYTE ALIGNED $R_{24:31}$ STH ⇒ (A) : HW ALIGNED $R_{16:31}$ STW ⇒ (A) : $R_{00:31}$ STD ⇒ (A) : R v1 XW, AWM ⇒ (A) : R00:31; (D) : EW (B) : PROG. ADDR. (P) : EFF. ADDR STD ⇒ (P) : EFF. ADDR v1 EVENTS DURING PREP STCF ⇒ INHIBIT R → A; ENABLE CF → A STCF, STB, STH ⇒ LEFT ALIGN A AWM ⇒ PRESET A+D → S XW ⇒ PRESET D → S; SET FM WRITE FASTORE ⇒ SET CORE MEM REQ; PRESET A → S STH, STD ⇒ PRESET RR → A	INSTRUCTION FAMILIES FASTORE: STB, STH, STW, STD, STCF, (XW·NPRE), (AWM·NPRE) FASTORE/1 : STD, (XW·NPRE), (AWM·NPRE) AXRRINH = FASTORE · PRE3 · ŌL4 AXFC = FASTORE · PRE3 · ŌL4 AXALB = FASTORE · PRE4 · NBCE S/SXAPD = FAADD · (PRE/34 + PH2) S/SXD = FUXW · PRE3 S/RW = FUXW · PRE3 S/MBXS = FASTORE · PRE/34 S/SXA = FASTORE · PRE/34 S/AXRR = FASTORE · PRE/34 · NŌ2	THE TRANSFER R → A IS AN AUTO FUNCTION IN PRE3. FAADD IS QUALIFIED WITH (FUAWM · PRE3) STD ⇒ STORE Rv1 NSTD ⇒ STORE R

"FASTORE": STB (75), STH (55), STW (35), STD (15), STCF (74), XW (44), AWM (66) 1 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH1	AWM ⇒ A + D → S; S → A	$S_n = (PR_n + K_n) \cdot N(PR_n \cdot K_n) \cdot SXADD$ AXS = FUAWM · PH1	
	XW ⇒ D → S; S → RR	$S_n = PR_n$ $R_n = S_n \cdot R_W$	
XW OR AWM T8L	FASTORE ⇒ A → S; S → MB	$S_n = PR_n$ $MB_n = S_n \cdot MBXS$	
	STH, STD ⇒ RR → A	$S/A_n = RR_n \cdot AXRR$	
	FASTORE/1 ⇒ PRESET A → S; SET CORE WRITE REQ.	S/SXA = FASTORE/1 · PH1 S/MBXS = FASTORE/1 · PH1	
STORE DR	STD ⇒ RESET P31	PDC31 = FASTORE · PH1 · ŌU1	
	AWM, XW ⇒ SET CC3 AND CC4 VIA TESTS	TESTS = FUXW · PH1 + FUAWM · PH1	
	AWM, OVERFLOW ⇒ SET CC2	PROBOVER = FUAWM · (PH1 + PH3)	
	AWM, END CARRY ⇒ SET CC1	CC1XL00 = FUAWM · (PH1 + PH3)	
	NFASTORE/1 ⇒ BRANCH TO PH7	BRPH7 = FASTORE · NFASTORE/1 · PH1	

FASTORE 2 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 DR	FASTORE/1 ⇒ { A → S ; S ↛ MB } BRANCH TO PH9	$S_n = PR_n$ $MB_n = S_n \cdot MBXS$ $BRPH9 = FASTORE \cdot PH2$	
PH9 T5L	B → S S ↛ P PRESET A → S	$SXB = PXSXB$ $PXS = PXSXB$ $S/SXA = FASTORE \cdot PH9$	
PH10 DR	A → S STH ⇒ {SET CC2 IF S0016 ≠ 0 OR ALL ONES AWM ⇒ SET TRAP IF OVERFLOW. AM NORMAL ENDE	$S_n = PR_n$ $S/CC2 = N(S0016Z + S0016W) \cdot (FUSTH \cdot ENDE)$ $S/TRAP = ENDE \cdot CC2 \cdot AM \cdot OVERIND$ $S/TR30 = ENDE \cdot CC2 \cdot AM \cdot OVERIND$ $S/TR31 = ENDE \cdot CC2 \cdot AM \cdot OVERIND$	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENT OF REGISTERS AT THE END OF PREPARATION (C) : EW (D) : EW (A) : RRv1 LS, CS ⇒ (P) : PROG. ADDR. STS ⇒ (P) : EW ADDR. STS ⇒ (B) : PROG. ADDR. END OF PREP. LS, CS ⇒ PRESET A∧D → S STS ⇒ PRESET NA∧D → S FASEL ⇒ PRESET S → C	FAMILY SIGNALS FASEL = LS, STS, CS FACOMP = CS., OTHER COMP INST(S). $S/PRXAD = FASEL \cdot PRE3 \cdot N0L7$ $S/PRXNAD = FASEL \cdot PRE3 \cdot 0L7$ $S/CXS = FASEL \cdot PRE3$	SELECTIVE INST(S). CS - COMPARE SEL. 45 LS - LOAD SEL 4A STS - STORE SEL 47
PH 1 T5L	LS, CS ⇒ A∧D → S STS ⇒ NA∧D → S S ↛ C FASEL ⇒ {PRESET A → S ; PRESET RR → A}	$S_n = PR_n$ $C_n = S_n \cdot CXS$ $S/SXA = FASEL \cdot PH1$ $S/AXRR = FASEL \cdot PH1$	LS, CS ⇒ (C) : [EW ∧ (RRv1)] STS ⇒ (C) : [EW ∧ ($\overline{RRv1}$)]

"FASEL" : LS (4A), STS (47), CS (45)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2 T5L	A → S Ś ↛ D RR ↛ A	S_n = PR n DXS = FASEL · PH2	(D): RRV1
	CS, STS ⇒ PRESET A∧D → S LS ⇒ PRESET A∧ND → S	S/A$_n$ = RR$_n$ · AXRR S/PRXAD = FASEL · PH2 · NØLA S/PRXAND = FASEL · PH2 · ØLA	(A): RR
PH 3 T5L	CS, STS ⇒ A∧D → S LS ⇒ A∧ND → S S ↛ A	S_n = PR n AXS = FASEL · PH3	CS, STS ⇒ (A): [RR∧(RRV1)] LS ⇒ (A): [RR∧(R̄R̄V1̄)]
	LS, STS ⇒ PRESET AVD → S CS ⇒ PRESET A−D → S	S/SXAORD = FASEL · PH3 · NØL5 S/SXAMD = FASEL · PH3 · ØL5	
	LS, CS ⇒ SET CORE MEM REQUEST STS ⇒ SET CORE MEM. WRITE	S/MRQ/1 = FASEL · PH3 · NØL7 S/MBXS = FASEL · PH3 · ØL7	FETCH NEXT INST. STORE RESULT IN EW
	LS ⇒ SET FM WRITE CS ⇒ SET T1IL C ↛ D	S/RW = FASEL · PH3 · ØLA S/T1IL = FASEL · PH3 · ØL5 DXC = FASEL. PH3	

FASEL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 4 STS DR CS T1IL LS T8L	LS, STS ⇒ AVD → S CS ⇒ A−D → S	S_n = PR$_n$ S_n = (PR$_n$ ⊕ K$_n$) · SXADD	LS ⇒ [R∧(R̄V1̄)] V[EW∧(RV1)] CS ⇒ [(R)∧(RV1)]:[EW∧(RV1)]
	STS ⇒ S ↛ MB LS ⇒ S ↛ RR	MB$_n$ = S_n · MBXS RR$_n$ = S_n · RW	STS ⇒ [(R)∧(RV1)] V[EW∧(R̄V1̄)]
	CS, LS ⇒ SET CC3 AND CC4 ACCORDING TO TESTS	SGTZ = (S0031 ≠ 0) · NS00 · FACOMP SGTZ = (S0031 ≠ 0) · NS0 · NFACOMP TESTS = FASEL · PH4 · NØL7 S/CC3 = SGTZ TESTS S/CC4 = S0 TESTS	SIGN EXTEND FOR CS FOR DETAIL ON TESTS SEE FAARITH PH3
	CS, LS ⇒ BRANCH TO ENDE STS ⇒ BRANCH TO PH9	BRPH10 = FASEL · PH4 · NØL7 BRPH9 = FASEL · PH4 · ØL7	
PH 9 T5L	STS ⇒ B → S ↛ P; SET CORE MEM REQ.	PXSXB = NFAFL · NFAMDS · PH9 MRQ/2 = PXSXB	
PH10 ENDE DR	NORMAL ENDE	SEE ENDE SEQU[illegible]E	

FASEL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END OF PREP	CONTENTS OF REGISTERS (D) : SW (P) : PROG. ADDR. SET MEM REQUEST (NEXT INST.) PRESET D → S	S/MRQ/1 = FUINT · PRE3 S/SXD = FUINT · PRE3	
PH 1 T5L	D → S S → CC CLEAR D0003 SET MERGE 1 → /LR/ PRESET D → S PRESET FM WRITE	Sn = PRn CCXS/0 = FUINT · PH1 DX/0A = FUINT · PH1 S/LR31 = FUINT · PH1 S/SXD = FUINT · (PH1 + PH3) S/RW = FUINT · (PH1 + PH3)	
PH 2 T8L	D → S S1631 → RR DOWN ALIGN D IF R FIELD IS ODD R31 = 1 ⇒ BRANCH TO ENDE	Sn = PRn S/RRn = Sn · RW RWXZ/01 = FUINT · PH2 DXDR8 = FUINT · PH2 BRPH10 = FUINT · PH2 · R31	RWXZ/01 TRANSFERS ZEROS TO RR0015

INT (6B) 1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3 T5L	PRESET D → S PRESET FM WRITE DOWN ALIGN D	S/SXD = FUINT · (PH1 + PH3) S/RW = FUINT · (PH1 + PH3) DXDR8 = FUINT · PH3	
PH 4 T8L	D → S S1631 → RR BRANCH TO ENDE	Sn = PRn S/RRn = Sn · RW RWXZ/01 = FUINT · PH4 BRPH10 = FUINT · PH4	RWXZ/01 INHIBITS BYTES 0 AND 1 TO RR
PH 10 ENDE T5L	NORMAL ENDE		

INT 2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP.	CONTENTS OF REGISTERS AT THE END OF PREP.	INSTRUCTION FAMILIES	
	NFADW/1 ⇒ (D) : EL (ALIGNED); (A) : RR; (P) : PROG. ADDR.	FACOMP/1 : CD, CI, CW, CH, CB	
		FASUB : CD, CI, CW, CH, CB, PLUS OTHERS	
		FAS10 : CI, CW, CH, CB, PLUS OTHERS	
		FADW/1 : CD, PLUS OTHERS	
	FADW/1 ⇒ (D) : EWv1; (A) : RRv1; (P) : EW ADDR.; (B) : PROG. ADDR.		
	END OF PREP.		
	FASUB ⇒ PRESET A-D → S	S/SXAMD = FASUB · (PRE/34 + PH2)	
	FAS10, FADW/1 ⇒ SET CORE MEM REQ.	S/MRQ/1 = FAS10 · (PRE/34 + PH2)	FETCH NEXT INST.
		S/MRQ/3 = FADW/1 · (PRE/34 + PH2)	FETCH EW
	FACOMP/1 ⇒ SET TIIL	S/TIIL = FACOMP/1 · (PRE/34 + PH2)	
	FADW/1 ⇒ MERGE 1 → LR	S/LR31 = FADW/1 · NANLZ · PRE3	
	CB ⇒ CLEAR A0023	AXZ/012 = FACOMP/1 · OUT · PRE4	

"FACOMP/1" : CI(21), CB(71), CH(51), CW(31), CD(11)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1	FASUB ⇒ A-D → S	$S_n = (K_n \oplus PR_n) \cdot SXADD$	
	FAS11 ⇒ SET CC3 & CC4 VIA TESTS	TESTS = FAS11 · (PH1 + PH3)	
		SGTZ = (S0031 ≠ 0) · NS00 · FACOMP	
		S/CC3 = SGTZ · TESTS	
TIIL		S/CC4 = S0 · TESTS	
	NFADW/1 ⇒ PRESET A ∧ D → S	S/PRXAD = FACOMP/1 · PH1 · NOUT	
	FACOMP/1 ⇒ SET T8L	S/T8L = FACOMP/1 · PH1 ·	
	FADW/1 ⇒ PRESET B → S	S/SXB = (FADW/1 · PH1)	
	PRESET RR → A	S/AXRR = (FADW/1 · PH1)	
	SAVE S ≠ 0	S/SW0/NZ = K00HOLD · NS0031Z	
	HOLD END CARRY	S/FL3 = K00HOLD · K00	
	FAS10 ⇒ BRANCH TO ENDE	BRPH10 = FAS10 · PH1	

FACOMP/1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2 DR	CD ⇒ { B → S; S → D; MB → C; C → D; RR → A }	$S_n = B_n \cdot SXB$ PXS = FADW/1 · PH2 $C_n = MB_n \cdot CXMB$ DXC = FADW/1 · PH2 $S/A_n = RR_n \cdot AXRR$	XFER. PROG ADDR TO P REQ.
	FASUB ⇒ PRESET A-D → S	S/SXAMD = FASUB · (PRE/34 + PH2)	
	FADW/1 ⇒ SET CORE MEM. REQUEST	S/MRQ/3 = FADW/1 · (PRE/34 + PH2)	
	DOUBLE PRECISION SUBTRACT END CARRY GOES TO LSB CARRY K00[PH1] → FL3 → K31[PH3]	S/K31/3 = FADW/1 · PH2 · FL3	
	FACOMP/1 ⇒ SET TIIL	S/TIIL = FACOMP/1 · (PRE/34 + PH2)	

FACOMP/1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3 TIIL	FASUB ⇒ A-D → S	$S_n = (Kn \oplus PRn) \cdot SXADD$	
	SET CC3 & CC4 WITH TESTS	TESTS = FAS11 · (PH1 + PH3)	
		S3263Z = NTESTS/1 · NSW0	
		SGTZ = (S0063 ≠ 0) · NS00 · FACOMP	
	COMPARE RESULT POS ⇒	S/CC3 = SGTZ · TESTS	
	COMPARE RESULT NEG ⇒	S/CC4 = S0 · TESTS	
	CD ⇒ BRANCH TO ENDE	BRPH10 = FADW/1 · PH3	
PH 10 ENDE	NFADW/1 ⇒ { A∧D → S; IF (S≠0) THEN 1 → CC2 }	$S_n = PRn$ S/CC2/NZ = FACOMP/1 · ENDE · N0U1	
DR	NORMAL ENDE		

FACOMP/1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREPARATION CLR ⇒ (D) : EW (A) : RRv1 (P) : EW ADDR (B) : PROG. ADDR. CLM ⇒ (D) : EW v1 (A) : RR (P) : EW ADDR (B) : PROG ADDR. END OF PREPARATION CLM ⇒ SET CORE MEMORY REQ. FASUB ⇒ PRESET A-D → S FACOMP/L ⇒ SET TIIL	FAMILY SIGNALS COMP/L : CLR, CLM FASUB : CLR, CLM PLUS OTHERS FACOMP : CLR, CLM, CD, CI, CW, CH, CB CS S/MRQ/3 = FACOMP/L · PRE3 · Ø4 7 S/SXAMD = FASUB · (PRE/34 + PH2) S/TIIL = FACOMP/L (PRE/34 + PH2)	COMPARE LIMITS INSTRUCTIONS CLR - COMPARE WITH LIMITS IN REGISTER CLM - COMPARE WITH LIMITS IN MEMORY MRQ/3 AUTO SETS DATA RELEASE ONE CLOCK LATER THAN MEMORY REQUEST. TIIL IS REQUIRED TO ALLOW TESTS WHEN ADDER → FM IS NOT PREFORMED

"FACOMP/L" : CLR (39), CLM (19)

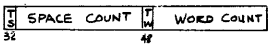
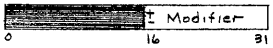
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1 TIIL	A-D → S FACOMP/L ⇒ TESTS SET CC3 IF S>0 SET CC4 IF S<0 FACOMP/L ⇒ PRESET B → S CLR ⇒ PRESET RR → A	$S_n = (PR_n \oplus K_n) \cdot SXADD$ TESTS = FACOMP/L · (PH1 + PH3) S/CC3 = SGTZ · TESTS S/CC4 = SO · TESTS SGTZ = (S0031 ≠ 0) · N500 · FACOMP S/SXB = FACOMP/L · PH1 S/AXRR = FACOMP/L · PH1 · Ø4 3	CLM ⇒ [-EW v1 + RR] CLR ⇒ [-EW + RRv1]
PH 2 (CLM) DR (CLR) T5L	CLM ⇒ MB → C C → D CLR ⇒ RR → A FACOMP/L ⇒ B → S S → P FACOMP/L ⇒ XFER CC3 → CC1 XFER CC4 → CC2 PRESET A-D → S SET CORE MEMORY REQ. SET TIIL	$C_n = MB_n \cdot CXMB$ DXC = FACOMP/L · PH2 · Ø4 1 $S/A_n = RR_n \cdot AXRR$ $S_n = B_n \cdot SXB$ PXS = FACOMP/L · PH2 S/CC1 = CC3 · FACOMP/L · PH2 S/CC2 = CC4 · FACOMP/L · PH2 S/SXAMD = FASUB · (PRE/34 + PH2) S/MRQ/1 = FACOMP/L · PH2 S/TIIL = FACOMP/L · (PRE/34 + PH2)	CLM ⇒ (D) : EW CLR ⇒ (A) : RR

FACOMP/L

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3 TIL	FACOMP/L ⇒ A − D ⟶ S FACOMP/L ⇒ { TESTS SET CC3 IF S>0 SET CC4 IF S<0 BRANCH TO ENDE	$S_n = (K_n \oplus PR_n) \cdot SXADD$ TESTS = FACOMP/L · (PH1 + PH3) S/CC3 = SGTZ · TESTS S/CC4 = S0 · TESTS SGTZ = (S0031 ≠ 0) · NS00 · FACOMP BRPH10 = FACOMP/L · PH3	CLM, CLR ⇒ [EW + RR] BRPH10 SETS DATA RELEASE
PH 10 ENDE DR	NORMAL ENDE		

FACOMP/L

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END OF PREP	CONTENTS OF REGISTERS (P) : PROG. ADDR. SET CORE MEM. REQUEST	S/MRQ/1 = FUWAIT · PRE3	
PH 1 T5L	SET HALT BRANCH TO ENDE	S/HALT/1 = FUWAIT · PH1 BRPH10 = FUWAIT PH1	
PH 10 ENDE	INHIBIT ADVANCE OF PROG ADDR BRANCH TO PCP1 INHIBIT PRE1 SEE PCP SEQUENCE	PUC31 = NFUEXU · PH10 $\underline{\text{NHALT}}$ · ... BRPCP = NFUEXU · HALT/1 · ENDE · NIOSC PRE1EN = N(S/TRAP) · N(S/INTRAP) · NIOSC · $\underline{\text{NHALT}}$	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	**CONTENTS OF REGISTERS** FAST ⇒ (C): SPW1 (D): SPW1 FAST/M ⇒ (B): PROG. ADDR FAST ⇒ (P): SPW0 ADDR LM, STM ⇒ (P): 1ST EFF. LOCATION MSP (A): RR (MODIFER) (FAST/M·06) (A): CC, (MC): CC } NO. OF WORDS PSW, PLW ⇒ (MC): 1 **PRESET CONDITIONS WITH PRE3** FAST/C ⇒ PRESET A+D → S PLM ⇒ PRESET D-A → S PSW ⇒ PRESET D+1 → S PLW ⇒ PRESET D-1 → S FAST ⇒ SET SW8 MSP ⇒ SET SW7 LM ⇒ R-1 → R; SET CORE MEM REQ. STM ⇒ PRESET RR → A; FAST → P-1 → P LM, STM ⇒ BRANCH TO PHASE 6 FAST ⇒ PRESET S → C; SET TIIL	**INSTRUCTION FAMILIES** FAST : PSM, PSW, PLM, PLW, MSP FAST/A : PSM, PSW, PLM, PLW FAST/M : PSM, PLM, PSW, PLW, LM, STM FAST/L : PLM, PLW, LM FAST/S : PSM, PSW, STS FAST/C : PSM, MSP FAST/A ⇒ STACK POINTER DOUBLEWORD SPW0·(R)  SPW1 - (RV1)  MSP ⇒ STACK POINTER MODIFIER (R)  S/SXAPD = FAST/C·(PRE3+PH1/F+PH8) S/SXDMA = FUPLM·(PRE3+PH1/F+PH8) S/SXDPI = FUPSW·(PRE3+PH1/F+PH8) S/SXDM1 = FUPLW·(PRE3+PH1/F+PH8) BRSW8 = FAST·PRE3 S/SW7 = FAST·PRE3·NØ4 RDC31 = FAST/M·PRE3·NØU0·ØLA S/MRQ/2 = FAST/M·PRE3·NØU0·ØLA S/AXRR = FAST/M·PRE3·NØU0·NØLA PDC31 = FAST/M·PRE3·NØU0·NØLA + FAST·PRE3 BRPH6 = FAST/M·PRE3·NØU0·NANL2 S/CXS = FAST·PRE3 S/TIIL = FAST·PRE3	

"FAST": PSW (09), PLW (08), PSM (0B), PLM (0A), MSP (13), LM (2A), STM (2B)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1/A	FAST/C ⇒ D+A → S PLM ⇒ D-A → S PSW ⇒ D+1 → S PLW ⇒ D-1 → S } CRUSH S16 S1631 → C1631 ZERO → C0015	PH1/A = PH1·SW8·FAST Sn = (Kn ⊕ PRn)·SXADD S16INH = FAST·PH1/A CXS/0 = CXS·N(FAST·PH1/A) CXS/1 = CXS·N(FAST·PH1/A) Cn(BYTE3) = Sn·CXS Cn(BYTE4) = Sn·CXS	UP DATE WORD COUNT TO C REG NØ6 ⇒ (C): WC±1 Ø6 ⇒ (C): WC±A
TIIL	A16 ⊕ K16 (WORD COUNT OVFL OR UNDFL) ⇒ SET SW3 TS (D0) ⇒ SET SW5 TW (D16) ⇒ SET SW6 WORD COUNT = 0 ⇒ SET SW4 FAST ⇒ DOWN ALIGN D SUSTAIN PH1 HOLD PH1 AND STEP SW8 - SW14 STEP TO SW9	S/SW3 = (A16-K16)·FAST·PH1/A S/SW5 = D0·FAST·PH1/A S/SW6 = D16·FAST·PH1/A S/SW4 = [N(A16-K16)·S1631Z]·FAST·PH1/A DXDR8 = FAST·PH1/A BRPH1/1 = FAST·PH1·N[(NSW7·PH1/C) + PH1/G + (SW3·PH1/C) + (S1631Z·N(A16-K16))·PH1/C] STEP815 = NBRSW8·NBRSW10·NBRPH11·NBRSW13·NBRSW15·NRESET S/SW9 = SW8·STEP815	

FAST

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1/B T8L	PLM ⟹ PRESET A+D → S PLW ⟹ PRESET D+1 → S PSW ⟹ PRESET D-1 → S FAST/C ⟹ PRESET D-A → S FAST ⟹ {DOWN ALIGN D; SET T11L} STEP TO SW10	S/SXAPD = FUPLM · PH1/B S/SXDP1 = FUPLW · PH1/B S/SXDM1 = FUPSW · PH1/B S/SXDMA = FAST/C · PH1/B DXDR8 = FAST · PH1/B S/T11L = FAST · PH1/B S/SW10 = SW9 · STEP 815	PH1/B = PH1 · SW9
PH 1/C T11L	PLM ⟹ A+D → S PLW ⟹ D+1 → S PSW ⟹ D-1 → S FAST/C ⟹ D-A → S CRUSH SIG S → A C → D A16 ⊕ K16 (SPACE COUNT OVFL. OR UNDFL.) ⟹ SET SW1 (SPACE COUNT = 0) ⟹ SET SW2 FAST/A·NSW7 (FIRST PASS) → (SECOND PASS) → FAST ⟹ SET CORE MEM REQ. FIRST PASS - PLM ⟹ PRESET A-1 → S (SECOND PASS) ⟹ SUSTAIN PH1	PH1/C = PH1 · SW10 · FAST Sn = (Kn ⊕ PRn) · SXADD SIGINH = FAST · PH1/C AXS = FAST · PH1/C · SW7 DXC = FAST · PH1/C S/SW1 = (A16 - K16) · FAST · PH1/C S/SW2 = [N(A16-K16) · S1631Z] · FAST · PH1/C SW7 ⟹ SET INDICATES STORE NEW WORD, SPACE AND NEW TSA R/SW7 = FAST · SW7 · PH1/C S/MRQ/3 = FAST · PH1/C S/SXAM1 = FUPLM · PH1/C · NSW7 BRPH1/1 = FAST · PH1 · N[(NSW7 · PH1/C) + (PH1/C · SW3) + (A16 - K16) · PH1/C]	(A): NEW SC (D): NEW WC (SW1): OVFL: SC UNDFL (SW2): SC = 0 GO TO PH2 IF ABORT OR TRAP OR FIRST PASS.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1/D T8L	FAST ⟹ ALIGN A STEP TO SW12	PH1/D = PH1 · SW11 · FAST AXAL8 = FAST · PH1/D BRSW12 = SW11 · STEP 815	
PH 1/E DR	MB → C FAST ⟹ UP ALIGN A PRESET A∨D → S TS (SW5) ⟹ SET A0 TW (SW6) ⟹ SET A16 SET CORE MEM WRITE SPW1 ADDR ⟹ P+1 → P STEP TO SW13	PH1/E = PH1 · SW12 · FAST Cn = MBn · CXMB AXAL8 = FAST · PH1/E S/SXADRD = FAST · PH1/E S/A0 = FAST · PH1/E · SW5 S/A16 = FAST · PH1/E · SW6 S/MBXS = FAST · PH1/E PUC31 = FAST · PH1/E BRSW13 = SW12 · STEP 815	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1/F	MERGE SPW1 ⇒ A v D → S	PH1/F = PH1 · SW13 · FAST	
	S → MB	Sn = PRn	STORE NEW SPW1
	SET CORE MEM WRITE	MBn = Sn · MBXS	
DR	C → D	S/MBXS = FAST · PH1/F	(D) : TSA
	MSP ⇒ RR1631 → A1631; RR16 → A15	DXC = FAST · PH1/F	
		AXRR/2 = FUMSP · PH1/F	AXRR/3 SAME AS... WHERE /3 GOES TO BYTE 3 AND /2 GOES TO BYTE 2. DIRECT NOT PRESET.
		S/A15 = FUMSP · PH1/F	
	FAST/C ⇒ PRESET A+D → S	S/SXAPD = FAST/C · (PRE3 + PH1/F + PH8)	
	PLM ⇒ PRESET D-A → S	S/SXDMA = FUPLM · (PRE3 + PH1/F + PH8)	
	PSW ⇒ PRESET D+1 → S	S/SXDP1 = FUPSW · (PRE3 + PH1/F + PH8)	
	PLW ⇒ PRESET D-1 → S	S/SXDM1 = FUPLW · (PRE3 + PH1/F + PH8)	MSP ⇒ (A) : MODIFIER
	FAST/M ⇒ CC → A	AXCC = FAST/M · (PRE3 + PH1/F + PH8) · %6	FAST/A ⇒ (A) : NO. OF WORDS
	SPW0 ADDR. TO P ⇒ P-1 → P	PDC31 = FAST · PH1/F	
	SUSTAIN PHASE	BRPH1/1 = FAST · PH1 · N[PH1/C ...	
	STEP TO SW	BRSW14 = SW13 · STEP815	

FAST 5 of 11

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1/G	FAST/C A+D → S; PLM D-A → S; PLW D-1 → S; PSW D+1 → S	PH1/G = PH1 · SW14 · FAST	
		Sn = (Kn ⊕ PRn) · SXADD	STORE UPDATED TSA
DR	S → MB	MBn = Sn · MBXS	MULTIPLE ⇒ TSA ± CC WORD ⇒ TSA ± 1 MSP ⇒ TSA + ER
	DROP PHASE 1	BRPH1/1 = FAST · PH1 · NPH1/G	
	BRANCH TO PH9	BRPH9 = FAST · PH1/G	
PH 2	PLM ⇒ A-1 → S; S → D	Sn = (Kn ⊕ PRn) · SXADD	FROM PH1/C
		DXS = FUPLM · PH2	PLM ⇒ (D): CC-1
	R FIELD TO A ⇒ R → A	AXR = FUPLM · PH2	DETERMINE STARTING REG. IF PULL MULTIPLE
PLM T8L	PRESET A+D → S	S/SXAPD = FUPLM · PH2	
ELSE T5L	TRAP CONDITIONS		
	SW3 · NSW6 OR SW1 · NSW5 ⇒ SET TRAP	S/TRAP = FAST · PH2 · SW3 · NSW6 + FAST · PH2 · SW1 · NSW5	
		S/TR30 = "	
	SW1 + SW3 ⇒ SET ABORT	S/FASTABORT = FAST · PH2 · SW1 + FAST · PH2 · SW3	NOTE FASTABORT IS BUILT WITH TWO FLIP FLOPS FASTF1 & FASTF2

FAST 6 of 11

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3	SPW0 ⇒ MB → C (TOP OF STACK ADDRESS)	Cn = MBn · CxMB	(C): TSA
	C ⇢ D	DxC = FAST/A · PH3	(D): TSA
DR	PLM ⇒ A+D → S	Sn = (Kn ⊕ PRn) · SXADD	
	S ⇢ R	RxS = FUPLM · PH3	(R): 1ST REG
	NABORT / FAST/A ⇒ PRESET S → C	S/CxS = FAST/A · PH3 · NFASTF1	
	SPW1 ADDR ⇒ P+1 ⇢ P	PUC31 = FAST/A · PH3	
	ABORT ⇒ SET MEM. REQ	S/MRQ/2 = FASTABORT · PH3	
PH 4	P → S	SxP = FAST . PH4	
	S ⇢ C	Cn = Sn · CxS	FOR PUSH⇒(C): SPW1 ADDR.
	S ⇢ A	AxS = FAST. PH4	FOR PULL ⇒ (A): SPW1 ADDR.
T5L	ABORT CONDITIONS		
	SW3 · SW6 OR SW1 · SW5 ⇒ BRANCH TO PHASE 9	BRPH9 = FAST/A · PH4 · SW3 · SW6 + FAST/A · PH4 · SW1 · SW5	
	NABORT ⇒ PRESET D → S	S/SxD = FAST/A · PH4 . NBRPH9	
	ABORT ⇒ MB → C	Cn = MBn . CxMB	
	C ⇢ D	DxC = FASTABORT . PH4	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5	D → S	Sn = PRn	
	S ⇢ P	PxS = FAST/A · PH5	
	FAST/L ⇒ SET MEMORY REQ. FETCH 1ST WORD	S/MRQ/2 = FAST/L · PH5	
	R+1 ⇢ R	RUC31 = FAST/L · PH5	
	FAST/S ⇒ PRESET FM READ	S/AxRR = FAST/S · PH5	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 6 DR	FAST/L ⇒ MB → C; C → D; D → S; S → RR	Cn = MBn · CXMB DXC = FAST/L · PH6 S/SXD = FAST/L · PH6 · NMCZ S/RW = FAST/L · PH6 · NMCZ	
	FAST/S ⇒ RR → A; A → S; S → MB	S/AXRR = FAST/S · PH6 S/SXA = FAST/S · PH6 · NMCZ S/MBXS = FAST/S · PH6 · NMCZ	
	(P) AND (R) COUNT CONTROL		
	FAST/S LM ⇒ P+1 → P	PUC31 = FAST/S · PH6 + FAST/L · PH6 · NØUO	
	PLM ⇒ P-1 → P	PDC31 = FAST/L · PH6 · ØUO	
	FAST/S LM ⇒ R+1 → R	RUC31 = FAST/S · PH6 + FAST/L · PH6 · NØUO	
	PLM ⇒ R-1 → R	RDC31 = FAST/L · PH6 · ØUO	
	DECREMENT MACRO COUNTER	MCDC7 = FAST/M · PH6 · NIØEN	
	(PLM, PLW) MC=0 ⇒ PRESET A→S	S/SXA = FAST/L · PH6 · ØUO · MCZ	
	(PSM, PSW) MC=0 ⇒ PRESET C→S	S/SXC = FAST/S · PH6 · ØUO · MCZ	
	IØ SERVICE CALL ENABLE (MC>4)	IØENG = FAST/M · NMC0005Z · PH6	
	MC ≠ 0 ⇒ SUSTAIN PHASE 6	BRPH6 = FAST/M · PH6 · NMCZ	
	(LM, STH) · (MC=0) ⇒ BRANCH TO 9	BRPH9 = FAST/M · PH6 · MCZ · NØUO	

FAST 9 of 11

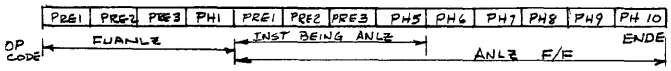
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 7 T5L	PULL ⇒ A → S PUSH ⇒ C → S SPW1 ADDR ⇒ S → P SET CORE MEM REQ.	Sn = PR + Cn · SXC PXS = FAST/A · PH7 S/MRQ/2 = FAST/A · PH7	(P): SPW1 ADDR.
PH 8 TIIL	MB → C; C → D; CC → A	Cn = MBn · CXMB DXC = FAST/A · PH8 AXCC = FAST/M · (PRE3 + PH1/F + PH8) · ØG AXZ = FAST/M · (PRE3 + PH1/F + PH8)	
	FAST/C ⇒ PRESET A+D → S PLM ⇒ PRESET D-A → S PSW ⇒ PRESET D+1 → S PLW ⇒ PRESET D-1 → S	S/SXAPD = FAST/C · (PRE3 + PH1/F + PH8) S/SXDMA = FUPLM · (PRE3 + PH1/F + PH8) S/SXDP1 = FUPSW · (PRE3 + PH1/F + PH8) S/SXDM1 = FUPLW · (PRE3 + PH1/F + PH8)	
	FAST/A ⇒ SET SW8; PRESET C → S	BRSW8 = FAST/A · PH8 S/SXC = FAST/A · PH8	
	FAST ⇒ SET TIIL TIMING	S/TIIL = FAST · PH8	
	SPWO ADDR ⇒ P-1 → P	PDC31 = FAST/A · PH8	

FAST 10 of 11

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 9 T5L	B ⟶ S S ⟶/⟶ P SET CONDITION CODE SW3 · SW6 ⇒ ABORT SET CC3 SW1 · SW5 ⇒ ABORT SET CC1 SW2 (SPACE COUNT = 0) ⇒ SET CC2 SW4 (WORD COUNT = 0) ⇒ SET CC4 ABORT ⇒ PRESET D→S	Sn = Bn . SxB PXSxB = NFAFL · NFAMDS · PH9 S/MRQ/2 = PXSxB FASTNABORT = FAST. PH9. NFASTABORT S/CC3 = FAST · PH9 · SW3 S/CC1 = FAST · PH9 · SW1 S/CC2 = FASTNABORT. PH9. SW2 S/CC4 = FASTNABORT. PH9 . SW4 R/CC = FAST · PH9 S/SxD = FASTABORT. PH9	
PH 10	NORMAL ENDE ABORT ⇒ { CORRECT CC2 & CC4 CRUSH S > 0 CRUSH S16 CRUSH S00	TESTS = FASTABORT. ENDE S/CC4 = FASTABORT. ENDE. S16 3Z S/CC2 = FASTABORT ENDE. S01 15Z SGTZ = I. FASTABORT. ENDE S16 = I. FASTABORT. ENDE S00 = I. FASTABORT. ENDE	⇐ TESTS. S17 3Z ⇐ TESTS. S0007Z. S08 15Z

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END PREP	CONTENT OF REGISTERS AT THE END OF PREP (A) : CONTROL IMAGE ADDRESS (B) : PROG. ADDRESS PRE3 PRESETS SET 1 → LR31 FAST MEMORY READ PRESET A → S	S/LR31 = FUMMC.PRE3 S/AXRR = FUMMC.PRE3 S/SXA = FUMMC.PRE3	Note A MAPPING OR READ PROTECT MMC INSTRUCTION IS DECODED AS AN ILLEGAL INSTRUCTION AND WILL TRAP TO X'40'
PH 1	A → S S → P S → D RRu1 → A SET CORE MEM REQUEST AND DATA RELEASE PRESET A → S	Sn = PRn PXS = FUMMC.PH1 DXS = FUMMC.PH1 S/An = RRn.AXRR S/MRQ/2 = FUMMC.PH1 S/SXA = FUMMC.(PH1+PH3)	TRANSFER COUNT AND CONTROL START TO A
PH 2 DR	A → S S → MC0007 S → P MB → C PRESET C → S	Sn = PRn MCXS = FUMMC.PH2 PXS = FUMMC.PH2 Cn = MBn.CXMB S/SXC = FUMMC.PH2	(A): RRu1 (P): CONTROL IMAGE ADD (D): " " " (C): MEMORY LOCK CONTROL IMAGE
PH 3	C → S S → A PRESET A → S PRESET WRITE TO LOCK BRANCH TO PH6	Sn = PRn AXS = FUMMC.PH3 S/SXA = FUMMC.(PH1+PH3) S/LOCKW = FUMMC.PH3 BRPH6 = FUMMC.PH3	(P): CONTROL START
PH 6 T8L	A → S S0007 → LOCK P1520+1 → P1520 BC - 1 → BC MC - 1 → MC ALIGN A LEFT 1 BYTE PRESET A → S SUSTAIN PH6 LAST PH6 ⇒ (BC=1) (BC=1) { UP DATE WORD CT. IN A MERGE 1 → /LR/ PRESET FM WRITE	Sn = PRn S/LOCKW = FUMMC.PH6.N(BC=1) PUC20 = FUMMC.PH6 BCDC1 = FUMMC.PH6 MCDC1 = FUMMC.PH6.BCZ AXAL8 = FUMMC.PH6 N(BC=1) S/SXA = FUMMC.PH6 BRPH6 = FUMMC.PH6.N(BC=1) AXMC = FUMMC.PH6.(BC=1) S/LR31 = FUMMC.PH6.(BC=1) S/RWXS = FUMMC.PH6.(BC=1)	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 7	P1520 → S A → S S → A S → RRu1 PRESET WRITE TO FM PRESET D+1 → S	SXP = FUMMC PH7 Sn = PRn + Pn · SXP AXS = FUMMC. PH7 S/RRn = Sn . RWXS S/RW = FUMMC. PH7 S/SXDP1 = FUMMC. PH7	LOAD A AND RRu1 WITH NEW START, COUNT
PH 8	D+1 → S S → D S → P S → RR PRESET A → S PRESET CORE MEMORY REQUEST AND DATA RELEASE NOT(INTERRUPT OR IOSC) AND NOT MC=0 ⇒ BRANCH TO PH 2	Sn = (Kn ⊕ PRn).SXADD DXS = FUMMC. PH8 PXS = FUMMC. PH8 S/RRn = Sn . RWXS S/SXA = FUMMC. BRPH2 S/MRQ = FUMMC. BRPH2 BRPH2 = FUMMC. PH8, NMCZ . N(INT + IOSC)	BRPH2 ⇒ INST CONTINUED, ELSE INST. EXIT
PH 9	B → S S → P PRESET CORE MEMORY REQUEST AND DATA REL.	PXSXB = NFAFL. NFAMDS. PH9 S/MRQ/2 = PXSXB	
PH 10	NORMAL ENDE EXCEPT IF FUMMC DID NOT FINISH DECREMENT P. THIS WILL RETURN TO FUMMC INSTRUCTION AFTER INTERRUPT OR IOSC	PDC31 = FUMMC. PH10 NMCZ (INT+IOSC)	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREP. (C): } INSTRUCTION TO BE ANALIZED (D): } (P): EFF. ADDR OF INST. (B): PROG. ADDR.	 FAMILY SIGNAL (ADDRESSING) FAIM : ALL IMMEDIATE INSTS. FABYTE: ALL BYTE INSTS. FAHW : ALL HALF WORD INSTS. FAW : ALL WORD ADDRESS INSTS. FADW : ALL DOUBLE WORD INSTS.	
PH 1 T5L	C0107 → 01-7 PRESET D → S PRESET D1214 → /LR/ PRESET RR → A SET ANLZ FLIP-FLOP BRANCH TO PRE1	0XC = FUANLZ · PH1 S/SXD = FUANLZ · PH1 S/LRXD = 0XC S/AXRR = FUANLZ · PH1 S/ANLZ = FUANLZ · PH1 S/PRE1 = NCLEAR · FUANLZ · PH1 NBR = N(FUANLZ · PH1)	INST. TO BE ANALIZED IS LOADED INTO INSTRUCTION REGISTER THE ANLZ FLIP FLOP IS REQUIRED TO MAINTAIN ANALIZE SEQUENCE.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE 1 ANLZ T5L	D → S S → P INDEX ⇒ { SET INDEX FLIP FLOP; SET INDEX ALIGN; D1214 → LR; RR → A } WORD INDX NOT INDIRECT ⇒ PRESET A+D → S INDIRECT ADDR. ⇒ { SET IA; SET CORE MEM. REQ. }	Sn = PRn PXS = NFAIM · PRE/12 S/IX = INDX · PRE1 S/IXAL = INDX · PRE1 · NFAW /LR283/ = D1214 · LRXD S/An = RRn · AXRR S/SXAPD = FAW · PRE1 · INDX · NCO S/IA = Co · PRE1 S/MRQ/2 = CO · PRE1 · NFAIM	(P) : INST ADDR. (A) : INDX NFAIM INHIBITS CORE FETCH DURING ANLZ INST. THIS IS AN ILLEGAL INST. HOW EVER DURING ANLZ, THE ILLEGAL INST. TRAP IS DISABLED. THEREFORE THIS REQUEST CAN NOT BE ALLOWED.
PRE 3 ANLZ T8L	SET CONDITION CODE TO ANALIZED INST ADDR. CLASS BRANCH TO PH5 SET T11L CLEAR 0 REGISTER	S/CC1 = N01 · (ANLZ · PRE3) + N03 · NFAIM · (ANLZ · PRE3) S/CC2 = FADW · (ANLZ · PRE3) + FAHW · (ANLZ · PRE3) S/CC3 = ANLZ · IA S/CC4 = FAIM · (ANLZ · PRE3) R/CC = (ANLZ · PRE3) BRPH5 = ANLZ · PRE3 S/T11L = ANLZ · PRE3 0XZ = " "	(LIVLCFI) BRANCH TO PRE 4 IS DISABLED.
PRE 2 ANLZ	INDEXING AND INDIRECT ADDRESS OPERATIONS : SEE PREPARATION NFAW · NIA – T5L + T8L NFAW · IA – MEM. CYCLE + T8L NFAW · NIX · IA – MEM.CYC.+T5L	FAW · NIA – T8L FAW · IA – MEM.CYCLE + T8L	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 T1L	DEAD PHASE TO ALLOW FAMILY SIGNALS TO DIE OUT.	ADDER TO BUS FUNCTIONS OCCUR DEPENDENT ON ANALIZED INST. HOWEVER WRITE TO CORE OR FM IS INHIBITED.	
PH 6 T5L	P → S CC1 ⇒ S → A NCC1 ⇒ S → 2A NCC1 ⇒ P32 → A31 PRESET A → S	SxP = ANLZ · PH6 AxS = CC1 · ANLZ · PH6 AxSL1 = NCC1 · ANLZ · PH6 A31xP32 = NCC1 · ANLZ · PH6 S/SXA = ANLZ · PH6	} DISPLACEMENT ALIGNMENT
PH 7 T5L	A → S FABYTE* ⇒ S → 2A FADW* ⇒ S → ½A PRESET A → S NFAIM* ⇒ PRESET FM WRITE * FAMILY BYTE ADDRESS OF ANALYSED INST. FABYTE, ETC. HAS BEEN CLEARED BY 0X2 IN PRE3. ADDR. GROUP HELD IN COND. CODE SETTING	Sn = PRn AXSL1 = NCC1 · NCC2 · ANLZ, PH7 A31xP33 = NCC1 · NCC2 · ANLZ · PH7 AXSR1 = CC1 · CC2 ANLZ · PH7 S/SXA = ANLZ · PH7 S/RW = ANLZ · PH7 · NCC4, NOL1	} DISPLACEMENT ALIGNMENT SET ONLY IF TYPE OF INST WAS NOT IMMEDIATE ADDRESSING TYPE

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 8 T8L	A → S NFAIM* ⇒ S → RW	Sn = PRn S/RRn = Sn · RW	
PH 9 T5L	B → S → P SET CORE MEM. REQUEST	PXSxB = NFAFL · NFAMDS · PH9 S/MRQ/2 = PXSxB	FETCH NEXT INST
PH 10 ENDE DR	NORMAL ENDE ZERO → ANLZ	SEE ENDE SEQUENCE R/ANLZ = CLEAR	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE	CONTENTS OF REGISTERS AT THE END OF PREPARATION	FAMILY SIGNAL	
		FABRANCH : BAL, BCS, BCR BDR, BIR, EXU	
	NEXU ⇒ (B): UPDATED PROG. ADDR. (P): EFFECTIVE ADDR.		
	FABRANCH ⇒ (A) : RR		
	EXU ⟹ (P): UPDATED PROG. ADDR.	PXS = FUEXU · PRE3 · NANLZ	MOVE PROG. ADDR BACK TO (P).
	FABRANCH ⟹ SET MEM REQUEST	S/MRQ/1 = FABRANCH · PRE/12.NANLZ	
	BIR ⟹ PRESET A+1 → S	S/SXAP1 = FUBIR · PRE3	
	BDR ⟹ PRESET A-1 → S	S/SXAM1 = FUBDR · PRE3	
	BAL, BIR, BDR ⟹ PRESET FM WRITE	S/RW = FUBAL · PRE3 · NANLZ + FUBIR · PRE3 · NANLZ + FU BDR · PRE3 · NANLZ	
	EXU ⟹ BRANCH TO ENDE	BRPH10 = FUEXU · PRE3 · NANLZ	
	BAL ⟹ PRESET B → S	S/SXB = FUBAL · PRE3	

"FABRANCH" : EXU (67), BCS (69), BCR (68), BIR (65), BDR (64), BAL (6A)

1 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 1	BAL ⟹ B → S BIR ⟹ A+1 → S BDR ⟹ A-1 → S S ↛ RR	$S_n = B_n \cdot SXB$ $S_n = (Kn \oplus PRn) \cdot SXADD$ $RR_n = S_n \cdot RW$	BIR : BRANCH UNTIL NEG RR = ZERO OR POS. BDR : BRANCH UNTIL POS RR = ZERO OR NEG.
BCS OR BCR T5L	FABRANCH ⇒ SET DATA RELEASE BCS · (CC∧R ≠ 0), BCR · (CC∧R = 0), BAL ⟹ BRANCH TO ENDE	S/DRQ/2 = FABRANCH · PH1 BRPH10 = FUBCS · PH1 · (R · CC) + FUBCR · PH1 · N(R · CC) + FUBAL · PH1	GO TO ENDE FOR PROGRAM LOOP (BRANCH). GO TO PH9 FOR PROGRAM END LOOP (NO BRANCH)
BIR BDR T11L	BIR · (S < 0), BDR · (S > 0) ⟹ SET PH10 * (ENDE)	S/PH10 = FUBIR · PH1 · NBRPH9 + FUBDR PH1 · SGTZ	* SPECIAL MECH. TO AVOID LEVEL PILE UP. KILL NBR BY (FUBDR · PH1)
BAL T8L	BCS · (CC∧R = 0), BCR · (CC∧R ≠ 0), BIR · (S ≥ 0) ⟹ BRANCH TO PH 9	NBR = I. FABRANCH. PH1 BRPH9 = FUBCS · PH1 · N(R · CC) + FUBCR · PH1 · (R · CC) + FUBIR · PH1 · NS0	
	BDR · (S ≤ 0) ⟹ SET PH9 *	S/PH9 = FUBDR · PH1 · NSGTZ	
PH 9 DR	B → S ↛ P SET MEMORY REQUEST	PXSXB = NFAFL · NFAMDS · PH9 S/MRQ/2 = PXSXB	NO BRANCH PLACE PROG. ADDR INTO (P) AND REQUEST NEXT INST IN SEQUENCE.
PH 10 ENDE DR	NORMAL ENDE WITH ADDITIONAL REQUIREMENT REQUIRE FOR EXU LISTED		
	EXU ⟹ INHIBIT P+1 → P; IF (INT + IOSC) − P-1 → P	PUC31 = NFUEXU · PH10 · NHALT · NINT · NIOSC PDC31 = FU EXU · (INT + IOSC) · ENDE	
	IOENABLE ⟹	IOEN = IOSC · PH10 · NIOINH	

FABRANCH

2 of 2

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
Prelim	SET TRAP	S/TRAP = (S/TRAP)·NRESET	
	SET INTRAP	S/INTRAP = (S/INTRAP)·NRESET	
	SET INTRAP1	S/INTRAP1 = (S/INTRAP)·NRESET	
	SET INTRAP2	(S/INTRAP2) = (S/INTRAP)	
	INHIBIT SET PREI	PREIEN = N(S/TRAP)·N(S/INTRAP)	
	CLEAR	CLEAR = (S/INTRAP)	
INTRAP1 · INTRAP2	SET CEINT	(S/CEINT) = INTRAP1·INTRAP2·NTRAP	
	SET DRQ	N(S/DRQ/2) = I·INTRAP1·TRAP	
	BRP ⇒ P ↔ B	BXP = BRP·INTRAP1	
	RESET BRP	(R/BRP) = INTRAP1	
	C → D	NDXC = I·INTRAP2	REDUNDANT TERM
	RESET INTRAP2	R/INTRAP2 = •	
	CLEAR P	PX = INTRAP1	REDUNDANT TERM
INTRAP1	INT ⇒ INT_{0-8} → P_{28-31}	PXINT = NTRAP·(INTRAP1·NINTRAP2)	PXINT ⇒ ENTER ACTIVE TO INTERRUPT LOGIC
	CLEAR P	PX = INTRAP1	
	TRAP ⇒ TR_{28-31} → P_{28-31}	PXTR = TRAP·(INTRAP1·NINTRAP2)	
	SET DRQ	N(S/DRQ/2) = I·INTRAP1·TRAP	REDUNDANT TERM
	SET MRQ	N(S/MRQ/2) = I*+·INTRAP1·NINTRAP2	
	SET DRQ	(S/DRQ) = I·N(S/MRQ/2)	
	RESET CEINT	R/CEINT = •	
	RESET INTRAP1	(R/INTRAP1) = I·INTRAP2	
	SET INTRAP2	(S/INTRAP2) = (INTRAP1·NINTRAP2)	
ARE + DR			
INTRAP2	MB → C → D	NDXC = I·INTRAP2	
	→ Ø	NØXC = B·N(NINTRAP1·INTRAP2)	
	→ R	RXC = I·N(NINTRAP1·INTRAP2)	
		RX = I·N(NINTRAP1·INTRAP2)	
	SET SXD	N(S/SXD) = B·N(NINTRAP1·INTRAP2)	
	RESET INT	(R/INT) = I·N(NINTRAP1·INTRAP2)	
	RESET INTRAP2	R/INTRAP2 = •	
	SET PREI	N(S/PREI) = I*+·INTRAP2·NINTRAP1	
DR			

INTRAP SEQUENCE

1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END OF PREP	CONTENTS OF REGISTERS (A): PSW1 (B): PROG ADDR (P): CURRENT PSW1 ADDR XPSD { NTRAP ⇒ PRESET A → S TRAP ⇒ PRESET A-1 → S ⇒ SET CORE MEM. WRITE LPSD ⇒ { BRANCH TO PH3 SET CORE MEM. REQ.	AXPSW1 = FAPSD · PRE3 AXZ = FAPSD · PRE3 S/SXA = FAPSD · PRE3 · NTRAP S/SXAM1 = FAPSD · PRE3 · TRAP S/MBXS = FAPSD · PRE3 · 07 BRPH3 = FAPSD · PRE3 · NOT. NANLZ S/MRQ/2 = FAPSD · (PRE/34 + PH2)	
PH 1 DR	NTRAP ⇒ A → S TRAP ⇒ A-1 → S S → MB XPSD ⇒ { CURRENT PSW2 ADDR. → P CURRENT PSW2 → A SET CORE MEM WRITE PRESET A → S TRACCs → TRs	Sn = PRn + (PRn ⊕ Kn) · SXADD MB n = Sn · MBXS PUC31 = FAPSD (PH1 + PH3) AXPSW2 = FAPSD · PH1 S/MBXS = FAPSD · PH1 S/SXA = FAPSD · PH1 S/TR28 = FAPSD · PH1 · TRACC1	XPSD: STORE CURRENT PSW1 NOTE ⇒ { TRACC1 → TR28 TRACC2 → TR29 TRACC3 → TR30 TRACC4 → TR31
PH 2 DR	A → S S → MB SET CORE MEMROY REQ NEXT PSW1 ADDR → P	Sn = PRn MBn = Sn · MBXS S/MRQ/2 = FAPSD · (PRE/34 + PH2) PUC31 = FAPSD · PH2	XPSD: STORE CURRENT PSW2

"FAPSD": LPSD (0E), XPSD (0F)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3 DR	FAPSD ⇒ { MB → C C → D SET CORE MEM. REQUEST NEXT PSW2 → P PRESET A + D → S XPSD (IF TRAP · R29) ⇒ TR's → A	Cn = MBn · CXMB DXC = FAPSD · PH3 S/MRQ/2 = FAPSD · PH3 PUC31 = FAPSD · PH3 S/SXAPD = FAPSD · PH3 AXTR = FAPSD · PH3 · 07 · R29 · TRAP AXZ = FAPSD · PH3	FETCH NEXT PSW1 LOAD A WITH TRs OR ZERO
PH 4 DR	MB → C C → D A+D → S S → PSW1 S → P XPSD ⇒ IF TRAP, TRACC → CC FAPSD ⇒ { SET MEM REQUEST PRESET D → S LPSD · R30 ⇒ INTERLOCK CLOCK WITH ARE	Cn = MBn · CXMB DXC = FAPSD · PH4 Sn = (Kn ⊕ PRn) · SXADD PSW1XS = FAPSD · PH4 PXS = FAPSD · PH4 CCXTRACC = FAPSD · PH4 · 07 · TRAP S/MRQ/1 = FAPSD · PH4 S/SXD = FAPSD · PH4 S/CEINT = FAPSD · PH4 · NOT · R30	FETCH NEXT PSW2 LOAD NEXT PSW1 IF XPSD · TRAP MERGE CC WITH TRACCS

FAPSD

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 T5L	LOAD NEXT PSW2 LPSD ⇒ {R30, LEVACT; R31, LEVARM} FAPSD {D → S; S → PSW2} LPSD ⇒ ZERO → INT INHIBITS FAPSD ⇒ RESET INTRAP = TRAP = TRACC BRANCH TO ENDE	PSW2xS = FAPSD · PH5 LEVACT = FAPSD · PH5 · N07 · R30 LEVARM = FAPSD · PH5 · N07 · R30 · R31 Sn = PRn PSW2 = FAPSD · PH5 PSW2xS/1 = PSW2xS · N07 (R/INTRAP) = (R/TRAP) (R/TRAP) = FAPSD · PH5 (R/TRACC) = FAPSD · PH5 BRPH10 = FAPSD.PH5	INTERRUPT INHIBITS XPSD ⇒ MERGE CURRENT SETTING WITH NEW LPSD ⇒ LOAD NEW VALUES
PH 10 DR	NORMAL ENDE		

FAPSD 3 of 3

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE	CONTENTS OF REGISTERS (B) : PROGRAM ADDR.		
PH 1 T5L	R ↔ TRACC 1, 2, 3 & 4	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	SET CONDITION CODES DURING XPSD
	SET TR28 CAL3 OR CAL4 ⇒ SET TR30 CAL2 OR CAL4 ⇒ SET TR31	S/TR28 = (FACAL·PH1)·NTRAP·NSTRAP S/TR30 = (FACAL·PH1)·06 S/TR31 = (FACAL·PH1)·07·NSTRAP·NTRAP	MODIFIES ADDRESS OF INST POINTED TO BY XPSD
	SET TRAP	(S/TRAP) = (FACAL·PH1)	PROCEED TO INTRAP SEQ.
INTRAP	SEE INTRAP SEQUENCE		

"FACAL": CAL1 (04), CAL2 (05), CAL3 (06), CAL4 (07) 1 of 1

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END OF PREP	CONTENTS OF REGISTERS (P) : PROG. ADDR. (B) : EFFECTIVE ADDR. WD ⟹ (A) : RR NRZ ⟹ PRESET A ⟶ S RD ⟹ CLEAR A	B1619Z ⟹ INTERNAL MODE S/SXA = FARWD · (PRE/34 + PH2) · NRZ AXRRINH = FARWD · 0LC · PRE3	
PH 1 T5L	WD · N(INTERNAL MODE) ⟹ A ⟶ S ⟹ S ⟶ DI0'0031 B ⟶ DI0'3247 NRZ ⟹ PRESET A ⟶ S INTERNAL MODE ⟹ KSS ⟶ CC; INTERRUPT INH; WRITE DIRECT WD SET WRITE LINE	Sn = PRn DI0XS = FARWD · PH1 · 0LD · NB1619Z DI0XB = FARWD · PH1 S/SXA = FARWD · (PH1 + PH3) · NRZ CCXRWD = FARWD · PH1 · B1619Z INHXWD = CCXRWD · 0LD · B26 WDINT = CCXRWD · 0LD · B25 S/DI0WD = FARWD · PH1 · 0LD	
CONT NEXT PAGE	WD · (INTENAL MODE) · B27 ⟹ SET INTERRUPT INHIBITS	S/CIF = INHXWD · B29 · B27 R/CIF = INHXWD · B29 S/II = INHXWD · B30 · B27 R/II = INHXWD · B30 S/EI = INHXWD · B31 · B27 R/EI = INHXWD · B31	

"FARWD": RD(6C), WD(6D)

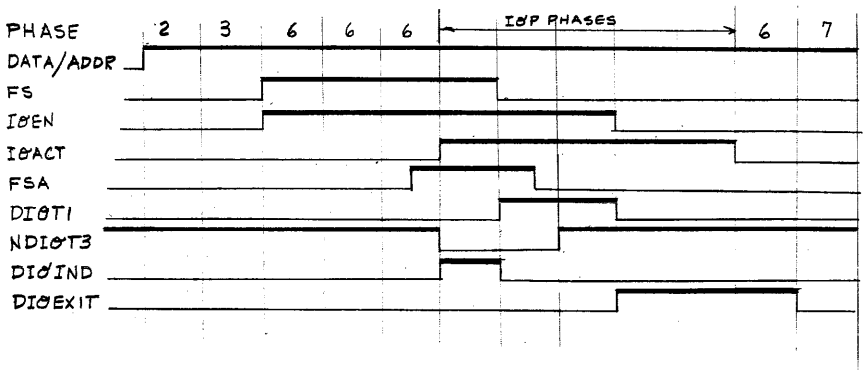
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	READ SENSE SWITCHES INTERNAL MODE	S/CC1 = CCXRWD.KSS1 S/CC2 = CCXRWD.KSS2 S/CC3 = CCXRWD.KSS3 S/CC4 = CCXRWD.KSS4 R/CC = CCXRWD	
	ALARM CONTROL	S/ALARM = WDINT. B31 R/ALARM = WDINT. + RESET	
	AUDIO TOGGLE	S/MUSIC = WDINT. B30. NMUSIC R/MUSIC = WDINT. B30 AUDIO = MUSIC. NALARM + ALARM. KRUN. 1KC	
	READ AND RESET MEMORY FAULT INDICATORS MFI 0-7 ⟶ A2331 ZERO ⟶ A0024 ZERO ⟶ MRI	RDXMFI = CCXRWD . 0LC . B27 AXPARITY = RDXMFI MFR = RDXMFI · NRZ + RESET	
	INTERNAL MODE ⟹ BRPH8	BRPH8 = FARWD. PH1 · B1619Z	

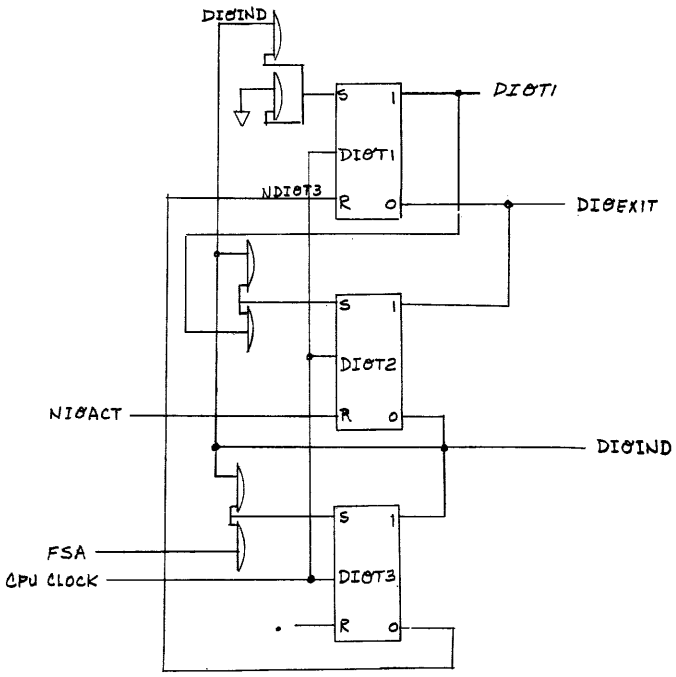
FARWD

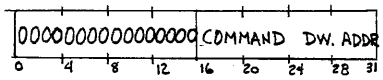
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2 T5L	A → S EWDM ⇒ S_{1631} → DAT1631 WD · INTERRUT CONTROL ⇒ EWDM PRESET A → S	S_n = PRn DATn = Sn · EWDM EWDM = NB16 · NB17 · NB18 · B19 · DIOWD S/SXA = FARWD · (PRE/34 + PH2) · NRZ	
PH 3 T5L	A → S EWDM ⇒ S_{1631} → DAT1631 PRESET A → S DECREMENT MC BRANCH TO PH6 SET FUNCTION STROBE	Sn = PRn DATn = Sn · EWDM S/SXA = FARWD · (PH1 + PH3) · NRZ MCDC7 = FARWD · PH3 BRPH6 = FARWD · PH3 S/DIOFS = FARWD · PH3	
PH 6 T5L	A → S S_{1631} → DAT1631 PRESET A → S IO ENABLE DIOEXIT ⇒ CLEAR DIOWD NDIOEXIT ⇒ SUSTAIN PH6 FSA ⇒ ENABLE DIOIND DIOIND ⇒ DIO → DIO(REG) DIO51 → CC3 DIO52 → CC4	Sn = PRn DATn = Sn · EWDM S/SXA = FARWD · PH6 · NDIOEXIT · NRZ IOEN6 = FARWD · PH6 · NEWDM · NDIOEXIT · NMC0005Z R/DIOWD = DIOEXIT BRPH6 = FARWD · PH6 · NDIOEXIT DIOIND = FSA · DIOT3 · NDIOT2 = DIOxDIO S/CC3 = DIOIND · DIO51 S/CC4 = DIOIND · DIO52	NOTE SEE SHEET 5 FOR PHASE 6 CONTROL.

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 7 T8L	DIO → A PRESET A → S RD ⇒ SET FM WRITE	AxDIO = FARWD · PH7 S/SXA = FARWD · PH7 S/RW = FARWD · PH7 · OLC · NRZ	
PH 8 T8L	A → S RD · NRZ ⇒ S → RR BRANCH TO PH10 NEXT INST ⇒ SET MEM REQ	Sn = PRn S/RRn = Sn · RW BRPH10 = FARWD · PH8 S/MRQ/1 = FARWD · PH8	
PH 10 T8L	NORMAL ENDE		

DIRECT I/O CONTROL LOGIC





PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PREP	CONTENTS OF REGISTERS AT THE END OF PREPARATION (B): PROG. ADDR (D) & (P): IOPADDR SIO ⇒ (A): R[0]	INSTRUCTION FAMILY SIGNALS FAIO: SIO, TIO, TDV, HIO, AIO FAIO/1: SIO, TIO, TDV, HIO	1ST CMD ADDR
	EVENTS DURING PREP P → B D → S NINDX ⇒ S → P IA ⇒ {MB → C; C → D} INDX A+D → S S → D S → P SIO ⇒ {0 → /LR/; R[0] → A}	BXP/1 = BRP · PRE1 Sn = PRn PXS = NFAIM · PRE1 DXC = IA · PRE2 S/SXAPD = FAW · PRE1 · INDX · NCO + IA · IX · PRE2 DXS = FAIO · PRE/12 PXS = PRE2 LRXE = FUSIO · PRE3 S/AXRR = PRE/12  WORD IN R[0]	

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

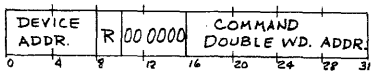
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH. 1	FUNCTION LINES TO MIOP AIO SIO TIO TDV HIO 02 → /FNC0/ 1 0 0 0 0 06 → /FNC1/ 1 0 0 1 1 07 → /FNC2/ 0 0 1 0 1	/FNC0C/ = BCD * B.02 /FNC1C/ = BCD * B.06 /FNC2C/ = BCD * B.07	
	FUNCTION LINES TO IIOP FUSIO → /SIO/ N06.07 → /TIO/ N02.06.N07 → /TDV/ 06.07 → /HIO/ FUAIO → /AIO/	/SIO/ = BCD * B.FUSIO. (NIOCON.NPH10) /TIO/ = BCD * B.N06.07. (NIOCON.NPH10) /TDV/ = BCD * B.N02.06.N07. (NIOCON.NPH10) /HIO/ = BCD * B.06.07. (NIOCON.NPH10) /AIO/ = BCD * B.FUAIO. (NIOCON.NPH10)	
	SET IOP ADDR LINES P21 → SW5 → /IOPA0/ P22 → SW6 → /IOPA1/ P23 → SW3 → /IOPA2/ NSIO ⇒ CLEAR A PRESET ⇒ D → S DEVICE ADDR D2431 → D0007 X'20' → P	S/SW5 = (FAIO. PH1). P21 /IOPA0C/ = BCD * B. SW5 S/SW6 = (FAIO. PH1). P22 /IOPA1C/ = BCD * B. SW6 S/SW3 = (FAIO. PH1). P23 /IOPA2C/ = BCD * B. SW3 AXE = FA10. PH1. NFUSIO S/SXD = (FAIO · PH1) DXDR8 = (FAIO. PH1) PX = (FAIO. PH1) S/P26 = (FAIO. PH1)	SW FLIP/FLOPS HOLD IOP ADDR. LINES UNTIL INST. COMPLETED. ALIGN DEVICE ADDR. CORE ADDR. FOR MIOP TRANSFER DOUBLEWORD

FAIO

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 2	$D \longrightarrow S$ $S_{0007} \rightarrow A_{0007}$	$S_n = PR_n$ $AxS/0 = AxS/2$ $AxS/2 = (FAIO.\ PH2)$	DEVICE ADDR. TO A_{0007}
	$R \neq 0 \Rightarrow$ SET A_{09} R IS EVEN $\Rightarrow$ SET A_{08} CLEAR A_{0815}	$(S/A9) = (FAIO.\ PH2).\ NRZ$ $(S/A8) = (FAIO.\ PH2).\ NRZ.\ NR31$ $AZ/1 = (FAIO.\ PH2)$	GENERATE R FOR TRANSFER TO X'20'
	MIOP, NOT AIO $\Rightarrow$ PRESET CORE WRITE	$S/MBXS = FAIO/1.PH2.N(IOPOP.IOADD)$ $S/MRQ = S/MBXS$ $S/DRQ = S/MBXS$	
	PRESET $A \longrightarrow S$	$S/SXA = (FAIO \cdot PH2)$	
	CLEAR CC1 AND CC2	$R/CC1 = (FAIO \cdot PH2)$ $R/CC2 = (FAIO.\ PH2)$	
	PROCEED LOW $\Rightarrow$ SET SWO	$S/SWO = (FAIO.PH2).\ NPR$	
	INTEGRAL IOP ADDR	$IOPADD = NSW3.NSW5.NSW6$	
		$IOAXST = FAIO.\ PH2$	

FAIO

3 of 15

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 3	$A \longrightarrow S$ MIOP, NAIO $\Rightarrow$ $S \longrightarrow MB$ $S \rightarrow D$	$S_n = PR_n$ $MB_n = S_n \cdot MBXS$ $DXS = (FAIO.\ PH3)$	WRITE COMM TO CORE X'20' FOR MIOP ONLY
			
	DEVICE ADDR. $\Rightarrow$ $A \rightarrow IOFR$ $A \rightarrow IODA$	$IOFRXA = FAIO.\ PH3-B$ $IODAXA = (FAIO.\ PH3)$	DEVICE ADDR. TO DEVICE SUBCONTROLLER
	NAIO. R31 $\Rightarrow$ $P+1 \rightarrow P$	$PUC31 = (FAIO/1.PH3).\ R31$	CORE ADDR. = X'21'
	PROCEED LOW $\Rightarrow$ SET SWO	$S/SWO = (FAIO.\ PH3).\ NPR$	
	NSWO $\Rightarrow$ BRPH3	$BRPH3 = FAIO.\ PH3.\ NSWO$	WAIT FOR PROCEED TO GO LOW
	SWO $\Rightarrow$ SET CONTROL STROBE	$S/IOCONST = (FAIO.\ PH3).\ SWO$	

FAIO

4 of 15

INTEGRAL IOP FAST MEMORY (D.C. CHANNEL)

CONTENT	READ PRESET	WRITE PRESET	WORD ADDR IOFM	IOFR8	IOFR9
STATUS, BITS 0 TO 15 — BYTE ADDRESS BITS 15 TO 31	S/AXRR/2	S/RW/2	1	0	0
FLAGS, BITS 0 TO 7 — BYTE ADDR, BITS 8 & 9 — BYTE COUNT BITS 16 TO 31	S/AXRR/3	S/RW/3	1	0	1
BYTE COUNT (ONLY) BITS 16 TO 31	S/AXRR/6	NOT DONE	1	0	1
MOST SIGNIFICANT BYTE ⇒ COMMAND DBW ADDR. BITS 24 TO 31	S/AXRR/4	S/RW/4	1	1	0
LEAST SIGNIFICANT BYTE ⇒ COMMAND DBW ADDR BITS 24 TO 31	S/AXRR/4 AND S/AXRR/3	S/RW/4 AND S/AXRR/3	1	1	1

STATUS BITS

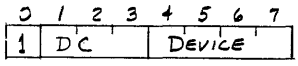
BIT 0	READ BACKWORD	BIT 9	TRANM. DATA ERROR
BIT 1	CHAINING MODIFIER	BIT 10	TRANSM. MEM. ERROR
BIT 2	ZERO BYTE COUNT INT.	BIT 11	MEMORY ADDR ERROR
BIT 3	CHANNEL END INT	BIT 12	IOP MEMORY ERROR
BIT 4	UNUSUAL END INT	BIT 13	IOP CONTROL ERROR
BIT 8	INCORRECT LENGTH	BIT 14	IOP HALT

FLAGS

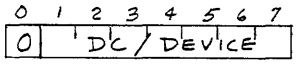
BIT 0	DATA CHAIN
BIT 1	INT. AT 0 BYTE CT.
BIT 2	COMMAND CHAIN
BIT 3	INT AT CHAN. END
BIT 4	HALT ON TRANS. ERR.
BIT 5	INT UNUSUAL END
BIT 6	SUPPRESS INCR. LENGTH
BIT 7	SKIP

DEVICE CONTROLLER ADDRESS DECODE

MULTIUNIT DEVICE CONTROLLER
IOFR0 = 1



SINGLE DEVICE CONTROLLER
IOFR0 = 0



LIOn = IOFRp WHERE n = 3-7, P = 1-7 AS PER TABLE BELOW

	CHANNEL SELECT	IOFR0 = 1	IOFR0 = 0
SELECT ONE OF 4 GROUPS	LIO3 =	0	IOFR3
	LIO4 =	0	IOFR4
SELECT ONE OF EIGHT CHANNELS	LIO5 =	IOFR1	IOFR5
	LIO6 =	IOFR2	IOFR6
	LIO7 =	IOFR3	IOFR7

STANDARD EIGHT CHS. * L/IO1Bi = NLIO3 . NLIO4

OPTIONAL CHANNELS (EIGHT EACH)
L/IO2Bi = NLIO3 . LIO4
L/IO3Bi = LIO3 . NLIO4
L/IO4Bi = LIO3 LIO4

* EQUATIONS NOT COMPLETE

PRESET FLIP FLOPS

IOFM
S/IOFM = S/AXRR/2 + S/AXRR/3 + S/AXRR/4 + S/AXRR/6 + S/RW/2 + S/RW/3 + S/RW/4

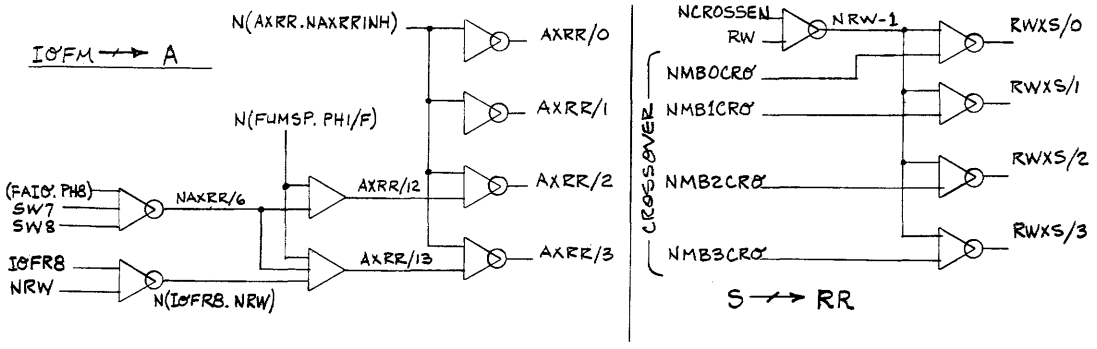
IOFM8
S/IOFM8 = S/AXRR/4 + S/RW/4

IOFM9
S/IOFM9 = S/AXRR/3 + S/AXRR/6 + S/RW/3

AXRR
S/AXRR = S/AXRR/2 + S/AXRR/3

RW
S/RW = S/RW/2 + S/RW/3 + S/RW/4

BYTE DRIVER LOGIC



PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 4	IOCONST ⟶ /CNSTC/	/CNSTC/ = BCD * B. IOCONST	CONTROL STROBE TO IOP(S) (IOPEX = MIOP CABLE)
	CNST ⟶ INTEGRAL IOP	CNST = IOCONST. IOPOP. NIOPEX + /CNST/. IOPOP. IOPEX	
	NIOPADD, AIO. NIOIR ⟹ IIOP PASS CNST ON	CNSTI = FAIO/1. NIOPADD. CNST + FUAIO. NIOIR. CNST	NOT INTEGRAL IOP ADDR. AND NO IIOP INT. PENDING
	INTEGRAL IOP PROCEED	PRI = CNSTI. [LAST IOP]	
	CNST {AIO. IOIR, IOPADD} ⟹ IIOP PROCEED	S/SW2 = FAIO/1. PH4. IOPADD. CNST + FUAIO. PH4. IOIR. CNST	INTEGRAL IOP ADDR. OR INT. PENDING
	WAIT FOR PROCEED ⟹ BRPH4	BRPH4 = FAIO. PH4. SW0. NSW2	
PH 4 (MIOP)	MIOP ADDRESSED PROCEED ⟶ RESET SW0	R/SW0 = FAIO. PH4. PR	NSW0 = MIOP PROCEED
	CLEAR ⟶ IOCONST	R/IOCONST = FAIO. PH4. NSW0	
	COND1, 2 ⟶ CC1, CC2	S/CC1 = FAIO. PH4. NSW0. COND1 S/CC2 = FAIO. PH4. NSW0. COND2	
	NRZ ⟹ READ CORE SET MRQ & DRQ	S/MRQ/2 = FAIO. PH4. NSW0. NRZ	
	RZ ⟹ BRPH9	BRPH9 = FAIO. PH4. NSW0. RZ	NO REG. STATUS REQ'D.

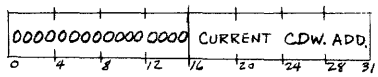
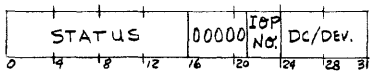
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 4	INTEGRAL IOP ADDRESSED (SECOND CLOCK) SET FUNCTION STROBE	IOPADD = NSW3. NSW5. NSW6	(SW0 = SW2 = 1) ⟹ IIOP DURING SECOND PH4 SW2 IS RESET
		S/IOFS = FAIO. PH4. SW2	
	PRESET [IOFM(00)] ⟶ A	S/AXRR/2 = FAIO. PH4. SW2 S/IOFM = S/AXRR/2	READ PREVIOUS STATUS AND BYTE ADDRESS
	START SEQUENCE SW8-15	BRSW8 = FAIO. PH4. SW2	
	RESET SW2	R/SW2 = FAIO. PH4. SW2	
PH 5	MIOP ADDRESSED		NSW0 ⟹ MIOP
	MB ⟶ C C ⟶ D [IOFM(00)] ⟶ A	Cn = MBn · CXMB DXC = FAIO. PH5. NSW0	R EVEN ⟹ X'20' ⟶ D R ODD ⟹ X'21' ⟶ D
	NAIO. NR31 ⟹ P+1 ⟶ P	S/An = RRn · AXRR PUC31 = FAIO/1. PH5. NSW0. NR31	INCREMENT P IF R EVEN
	SET CORE READ, DRQ DELAYED	S/MRQ/3 = FAIO/1. PH5. NSW0. NR31	READ X'21' IF R EVEN
	R31, AIO ⟹ PRESET D⟶S; PRESET S ⟶ FM	S/SXD = FAIO. PH5. NSW0. R31 + FUAIO. PH5. NSW0	
		S/RW = FAIO/1. PH5. NSW0. R31 + FUAIO. PH5. NSW0	AIO ⟹ X'20' CONTAINS [STATUS, IO ADDR]
	MIOP ADDRESSED GO TO PH6	S/PH6 = PH5. NBR. NIOEN	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 SW8 NSW7	INTEGRAL IOP ADDRESSED		
	NSW7 ⇒ SUSTAIN SW8	BRSW8 = FAIO. PH5. SW8. NSW7	WAIT FOR DEVICE CONTROLLER RESPONSE
	/AVO/+/FSL/ ⇒ SET SW7	S/SW7 = FAIO. PH5. SW8. NSW7. FSL + FAIO. PH5. SW8. NSW7. AVO	D.C. RESPONSE RECEIVED FSL ⇒ FUNCTION STROBE ACKNOWLEDGE AVO ⇒ FUNCTION NOT ACCEPTED
	NAIO ⇒ PRESET [FM(01)] → A	S/AXRR/6 = FAIO/1. PH5. SW8. NSW7	
	AIO ⇒ {CLEAR IODA; PRESET [FM(00)] → A}	IODAX = FUAIO. PH5. SW8. NSW7 S/AXRR/2 = FUAIO. PH5. SW8	
PH 5 SW8 SW7	NAIO ⇒ [FM(01)] → A1631 AIO ⇒ [FM(00)] → A	S/An = RRn. AXRR/6 S/An = RRn. AXRR	NAIO ⇒ BYTE CT → A1631 AIO ⇒ STATUS → A (FROM INT. ACKN'D DEVICE)
	SW7 ⇒ SET CONDITION CODE	AXRR/6 = FAIO/1. PH5. SW8. SW7 S/CC1 = FAIO. PH5. SW8. SW7. NDOR S/CC2 = FAIO. PH5. SW8. SW7. NIOR	
	RESET SW7 SIO ⇒ CLEAR OLD STATUS	R/SW7 = FAIO. PH5. SW8. SW7 SIOSP/1 = FUSIO. PH5. SW8. SW7	
		S/RW/2 = SIOSP/1. DOR. IOR	
	AIO ⇒ {/FR/ → IOFR; PRESET A → S; PRESET [FM(00)] → A}	IOFRXFR = (FUAIO. P5. 8. 7) IOFRX = (FUAIO. P5. 8. 7) S/SXA = (FUAIO. P5. 8. 7) S/AXRR/2 = FUAIO. PH5. SW8	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	VALID START ⇒ VALST	VALST = FUSIO. NCC1. NCC2	
PH 5 SW9	VALST ⇒ ZERO → S → IOFM(00)	RRn = Sn. RW	CLEAR OLD STATUS
	NAIO ⇒ FR → A0007	S/An = FRn. AXFR	
	AIO ⇒ A → S	AXFR = FAIO/1. PH5. SW9	
	PRESET A → S	Sn = PRn	
	AIO ⇒ [IOFM(00)] → A	S/SXA = FAIO. PH5. SW9	
	NAIO ⇒ {PRESET Ru1 → /LR/; PRESET WRITE TO Ru1}	S/An = RRn. AXRR	AIO ⇒ STATUS → A STATUS TO Ru1
		S/LR31 = FAIO/1. PH5. SW9	
		S/RW = FAIO/1. PH5. SW9. NRZ	
	AIO ⇒ PRESET WRITE TO [IOFM(00)]	S/RW/2 = FUAIO. PH5. SW9	
	RZ, AVO ⇒ BRANCH TO SW13	BRSW13 = FAIO/1. PH5. SW9. RZ + FAIO. PH5. SW9. AVO	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 SW 10	A → S NAIO ⇒ { S → Ru1 ZERO → RW15	Sn = PRn Rwn = Sn . RWxS Rw15 = S15 . RWxS/1 . RW15xZ NRWXZ = FAIO/1 . PH5 . SW10	LOAD IO STATUS BUT INHIBIT S15 WHICH INDICATES A BUSY SELECTOR IOP
	AIO ⇒ { S → [IOFM(00)] ZERO → { IO STATUS INTERRUPT BITS	RWn = Sn . RWxS RW3 = S3 . RWXS . N(FUAIO . PH5 . SW10) Rw4 = S4 . " . " RW5 = S5 . " . "	AIO STATUS RETURNED WITH INTERRUPT STATUS BITS CLEARED
	AIO ⇒ { IOFR → A0007 IOP STATUS → A0815 CLEAR TO BYTE 1	AxFR = FUAIO . PH5 . SW10 IOAxST = IOINST IOINST = FUAIO . PH5 . SW10 . (A_2 + A_3 + A_4 + NIOFRO) AZ/1 = FUAIO . PH5 . SW10	DEVICE ADDR → A
	NAIO, NR31 ⇒ PRESET [IOFM(10)] → A	S/AxRR/4 = FAIO/1 . PH5 . SW10 . NR31	
	NAIO, R31 ⇒ BRANCH TO SW13	BRSW13 = FAIO/1 . PH5 . SW10 . R31	ODD R FIELD ONE STATUS WORD REQUIRED

FAIO 10 of 15

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 SW 11	NAIO ⇒ { [IOFM(10)] → A PRESET [IOFM(11)] → A	S/An = RRn . AxRR/3 S/AXRR/4 = FAIO/1 . PH5 . SW11 S/AXRR/6 = FAIO/1 . PH5 . SW11	READ MSB OF CDW ADDRESS TO A2431
	AIO ⇒ A0008 → A2431	AxAR24 = FUAIO . PH5 . SW11	TRANSFER DEVICE ADDR. TO BYTE 3 OF A
PH 5 SW 12	NAIO ⇒ { A2431 → A1623 IOFM(11) → A2431	AxAL8 = FAIO/1 . PH5 . SW12 S/An = RRn . AxRR/3	LOAD MSB OF CDW ADDRESS TO A1623 AND READ LSB OF CDW TO BYTE 3 OF A
	PRESET A → S PRESET WRITE TO R	S/SXA = FAIO . PH5 . SW12 S/RW = FAIO . PH5 . SW12 . NRZ	
	AIO ⇒ { DA → S S → A0007 A12 ⇒ SET CC2	SxDA = FUAIO . PH5 . SW12 AxS/2 = FUAIO . PH5 . SW12 S/CC2 = FUAIO . PH5 . SW12 . A12	READ DEVICE STATUS TO BYTE 0 OF A CC2 INDICATES UNUSUAL INTERRUPT
	STATUS → A SIO ⇒		
	AIO ⇒		

FAIO 11 of 15

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 5 SW 13	A → S S → RR	Sn = PRn RWn = Sn . RWxS	STATUS → (R)
	DROP FUNCTION STROBE DROP CONNECT STROBE	R/IOFS = FAIO. PH5. SW13 R/IOCNST = FAIO. PH5. SW13	DISCONNECT DEVICE CONTROLLER
	VALST {PRESET D-1 → S PRESET S → [IOFM (10)]	S/SxDM1 = FAIO. PH5. SW13. VALST S/RW/4 = FAIO. PH5. SW13. VALST	IF SIO VALID START STORE NEW CDW. ADDR
	NVALST ⇒ BRANCH TO PH9	BRPH9 = FAIO. PH5. SW0. SW13. NVALST	NVALST ALSO INCLUDE ALL FAIO EXCEPT SIO
PH 5 SW 14	SIO. VALST ONLY D-1 → S S1623 → [IOFM(10)] SEE EXPLANATION OF IOFM ADDRESSING ON P.	Sn = (Kn ⊕ PRn) . SXADD RWn = Sn . RWXS/2	MSB NEW CDW ADDR TO IOFM 10
	S → A	AxS = FAIO. PH5. SW14	
	IIOP ⇒ BRANCH TO PH9	BRPH9 = FAIO. PH5. SW0. SW14	

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 6	MIOP ADDRESSED R31 + AIO ⇒ D → S	Sn = PRn	MIOP STATUS → R
	S → RR S → D	RWn = Sn . RWxS DxS = FAIO/1 . PH6. NR31	
	SET DRQ DELAYED	S/DRQ = MRQ P1	
	DOWN ALIGN D	DxDR8 = FAIO. PH6	R EVEN ALIGN CDW ADDRESS
DRQ	R31 + AIO ⇒ BRANCH TO PH9	BRPH9 = FAIO. PH6. NSW0. R31 + FUAIO. FAIO. PH6	R ODD OR AIO BRANCH TO PH9
PH 7	MB → C	Cn = MBn . CxMB	
	DOWN ALIGN D CLEAR D0015	DxDR8 = FAIO. PH7 NDxDR8/0, /1 = FAIO. PH7	
	PRESET D → S	C/SxD = FAIO . PH7	
	PRESET WRITE TO FM	S/RW = FAIO. PH7	
PH 8	D → S	Sn = PRn	
	S → RR	RWn = Sn. RWXS	[X'20'] → R
	C → D	DxC = FAIO. PH8	[X'21] → D

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 9	MIOP ADDRESSED		
	NAIO, R EVEN ⇒ IOBR9	IOBR9 = FAIO/1. PH9. NR31. NRZ	R EVEN STATUS → Ru1
	IOBR9 ⇒ {PRESET D → S; PRESET S → Ru1}	S/SXD = IOBR9. NSW0 S/RW = IOBR9. NSW0 S/LR31 = FAIO. PH9. NSW0	
	INTEGRAL IOP ADDRESSED		
	UP ALIGN LSB CDW ADDR.	AXAL8 = FAIO. PH9. SW0	
	VALST {PRESET A → S; PRESET S → [IOFM(11)]}	S/SXA = FAIO. PH9. SW0. VALST S/RW4 = FAIO. PH9. SW0. VALST S/RW3 = FAIO. PH9. SW0. VALST	PRESETS TO STORE LSB CDW ADDR.
	EITHER IOP		
	B → S S → P	SXB = PXSXB PXS = PXSXB MRQ/2 = PXSXB	FETCH NEXT INSTRUCTION
DRQ			

FAIO

14 of 15

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH 10 END	MIOP ADDRESSED		
	REVEN ⇒ {D → S; S → Ru1}	Sn = PRn RWn = Sn • RWXS	
	INTEGRAL IOP ADDRESS		
	VALST A → S; S → [IOFM(11)]	Sn = PRn RWn = Sn • RWXS/2	
	SEE NORMAL END		

FAIO

15 of 15

MULTIPLY: general sequence of events

MW, MI

1) put multiplicand in C (and D insignificantly). (extend sign if MI)
2) put multiplier in B (via A)
3) iterate, clearing A and B0001 on the first clock of PH6
   (double length product is formed in AB)
4) set CC2 if all bits in A* are ≠ B0
5) store A* in R (replaced by B later if R is odd)
6) store B in Rv1

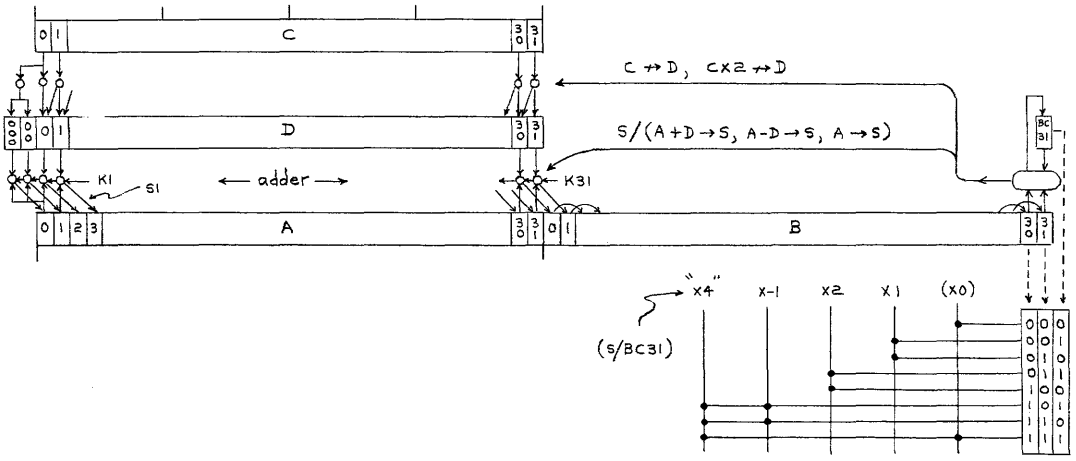
MH

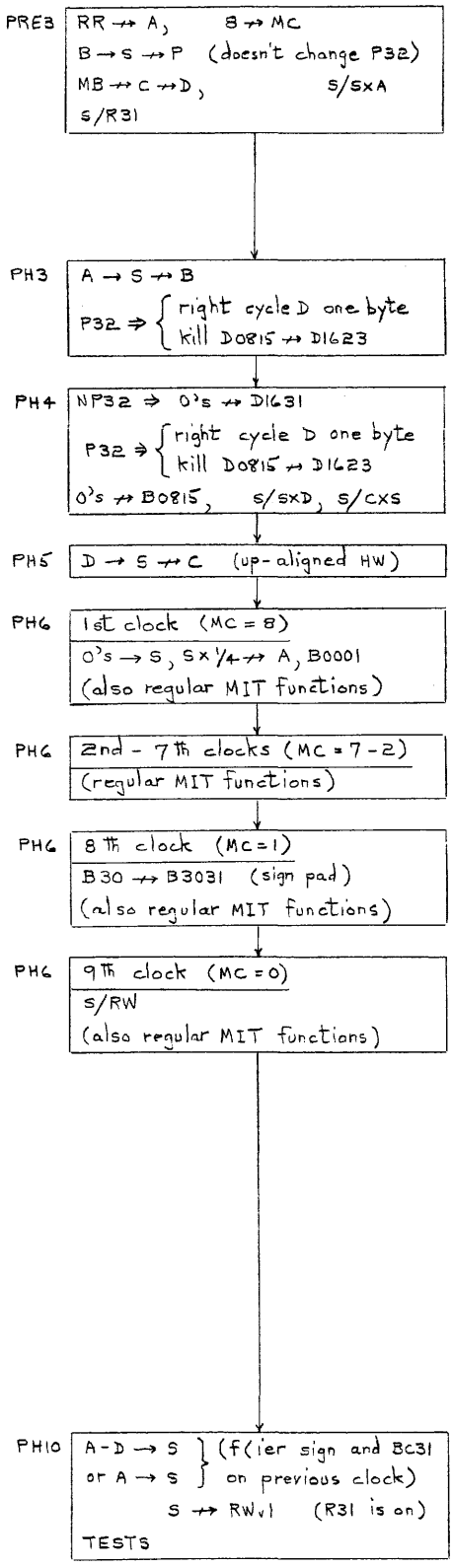
1) put multiplicand in C (and D insignificantly), upward-aligned, hence C1631 = 0.
2) put multiplier in B (via A) with 0's in B0815 (B1631 contains the 16 bit multiplier)
3) iterate, clearing A (and B0001) on the first clock of PH6
   (single length product is formed in A)
4) store A* in Rv1

(note: The reference manual considers (R) to be the multiplicand, though the hardware uses (R) as the multiplier. Obviously the product is unaffected by this contradiction in nomenclature.)

*: will be A-D in certain cases where multiplier is negative. .

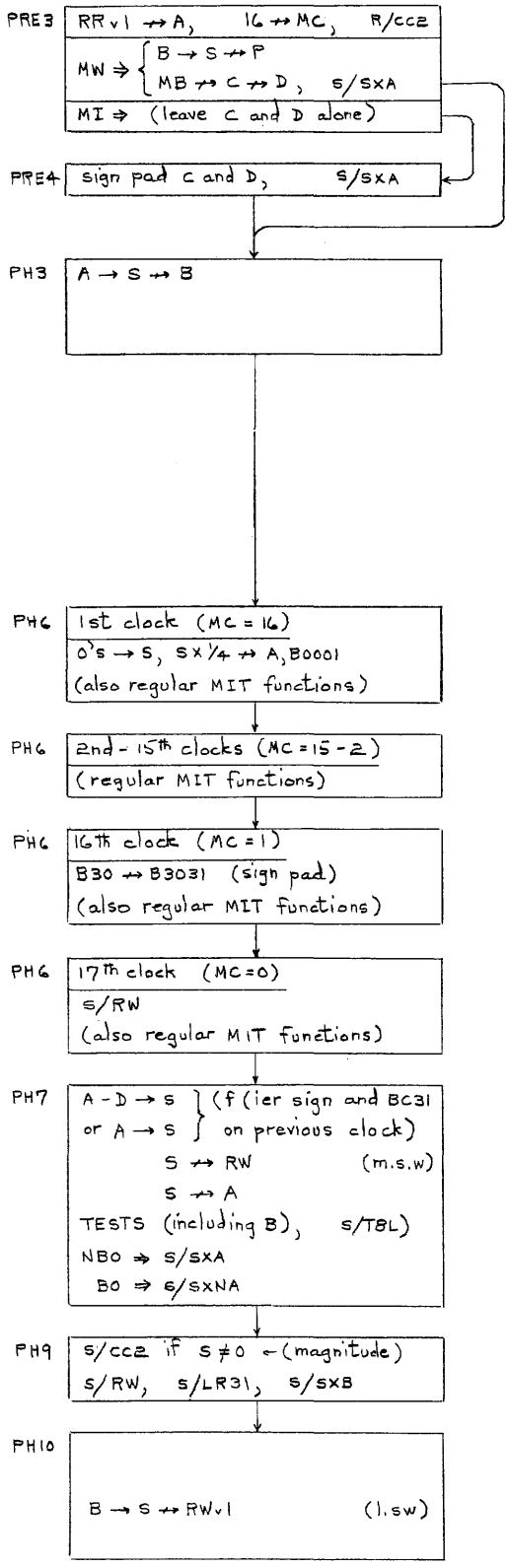
Register organization during MULTIPLY iterations:





MH : EHW × R1631 → R∨1

"FUMH"



MW : EW × R∨1 → R, R∨1
MI : I_{se} × R∨1 → R, R∨1

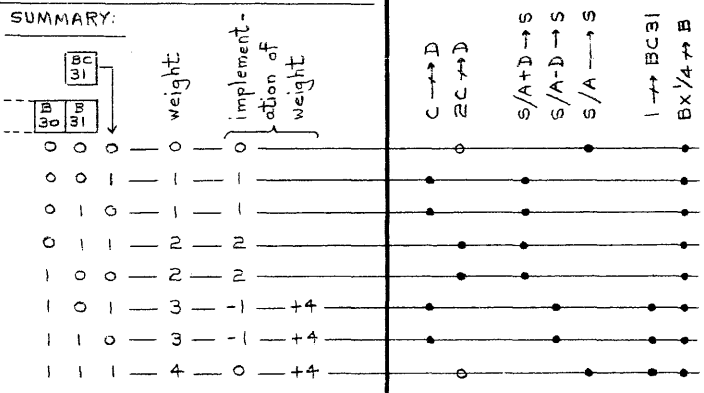
"FAMULNH"

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Family signals : FUMH (MH) FAMULNH (MW, MI) FAMUL (MH, MW, MI) FAMDS (MH, MW, MI, DW, DH, S, SF)	 = 0U5.0L7 = 0U3.0L7 + 0U2.0L3 = FAMULNH + FUMH = FAMULNH + FUMH + ---	 (EHW x R1631 → Rv1) (EW x (Rv1) → R, Rv1)
PRE3	B → S S → P if MW or MH	SXB : S/SXB = PRE/12 PXS = FAMDS.PRE3.NANL2.NFAIM	
	RRv1 → A if NMH RR → A if MH } multiplier	AXRR : S/AXRR = PRE/12 LR31 : S/LR31 = PRE/12. (S/LR31/12) ← FAMULNH	
	MB → C → D if NMI (C, D unchanged if MI) } multiplicand	DXC = PREOPER.PRE3	
	16 → MC if NMH 8 → MC if MH } set iter. ctr.	S/MC3 = PRE3.FAMULNH S/MC4 = PRE3.FUMH	
	set R reg. to Rv1 if MH S/A → S if NMI (for A → S → B in PH3) R/CC2 if NMH (for magnitude test)	S/R31 = FUMH.PRE3 S/SXA = FAMUL.PRE/34 ← low if MI R/CC2/1 = FAMDS.NFUMH.PRE3	(for product → Rv1 in PH10, and to render R (if even) unchanged if I/O)
	S/PH3 if NMI S/PRE4 if MI	BRPH3 = FAMDS.NBRPH5.NANL2.PRE/34 BRPRE4 = PRE3.NANL2.NPRE/34	
PRE4	(entered only if MI) The IMMEDIATE OPERAND in C and D is sign padded in both C and D as per chart on "PREPARATION"		
	S/A → S S/PH3	S/SXA = FAMUL.PRE/34 BRPH3 = FAMDS.NBRPH5.NANL2.PRE/34	
PH3	A → S S → B	(Preset in PRE3 or PRE4) BXS = FAMUL PH3	
	NMW ⇒ S/PH6	BRPH6 = FAMULNH.PH3	
	MW ⇒ { Right-cycle D 1 byte, 0's → D1623, S/PH4 } if P32	DXDR8 = FUMH.PH3.P32 DXDR8/2 = DXDR8.NFUMH (NBR is high)	(partial up-align HW if initially in l.s. HW)
PH4	(entered only if MH)		
	P32 ⇒ { Right-cycle D 1 byte, 0's → D1623 }	DXDR8 = FUMH.PH4.P32 DXDR8/2 = DXDR8.NFUMH	(complete up-align HW if initially in l.s. HW)
	NP32 ⇒ 0's → D1631	DX/2 = DX/3 = FUMH.PH4 ← redun if P32	(HW initially in m.s. HW)
	0's → B0815 (clear B1415 for sign logic)	BX/1 = FUMH.PH4	
	S/D → S S/CXS } (for D → S → C in PH5) (advance to PH5)	S/SXD = FUMH.PH4 S/CXS = FUMH.PH4	

Sheet 1 of 3

MNEMONIC: MW (37), MI (23), MH (57)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5	(entered only if MH) D → S ↛ C	(preset in PH4)	(up-aligned HW ↛ C)
PH6	(ITERATION phase) (17 clocks if MW or MI, 9 clocks if MH)		
	CONTROL SIGNALS:		
	MIT (handles direct logic)	MIT = FAMUL.PH6	
	MIT/1 (handles preset logic)	MIT/1 = FAMUL.NIOEN.(PH6 + S/PH6/I0)	(similar to "BCON" in Σ7)
	REGISTER CONTROL - PRESET LOGIC:		
	SUMMARY:		



FROM ABOVE SUMMARY:		
enable D	DX = MIT/1 + DXC + ···	(for benefit of 2C ↛ D)
C ↛ D	DXC = MIT/1.(B31 ⊕ BC31)	
2C ↛ D	DXCLI = MIT/1.NDXC	(insig when S/A → S)
C0 ↛ D00 and D000	S/D00 = S/D000 = C0.(DXC + DXCLI)	(sign extension)
S/A+D → S	S/SXAPD/1 = MIT/1.N(S/SXAMD/1).N(S/SXA/1)	
S/A-D → S	S/SXAMD/1 = MIT/1.B30.(B31 ⊕ BC31)	
S/A → S	S/SXA/1 = MIT/1.(NB30.NB31.NBC31 + B30.B31.BC31)	
1 ↛ BC31 (holds until reset)	S/BC31 = (S/SXAMD/1)	(set-reset logic convenient for I/O interruptability)
0 ↛ BC31 (" " set)	R/BC31 = MIT/1.NB30 + CLEAR	
S30/1 ↛ B0, S31/1 ↛ B1 (product bits)	S/B0 = BXBR2.S30/1 ← S30.B0001E.N/1 + FL1.(S/PH6/I0)	
	S/B1 = BXBR2.S31/1 ← S31.B0001E.N/1 + FL2.(S/PH6/I0)	
B0029 ↛ B0231	BXBR2 = MIT/1 (= MIT)	
REGISTER CONTROL - DIRECT LOGIC:		
S000 ↛ A0, S00 ↛ A1, S0029 ↛ A0231	AXSR2 = MIT	(adder extension; (A00 red. to A0))
S30 ↛ FL1, S31 ↛ FL2 (2-bit extension of A (for I/O))	S/FL1 = S30.MIT; R/FL1 = (MIT + CLEAR) S/FL2 = S31.MIT; R/FL2 = (" + ")	
CONTROL FUNCTIONS - GENERAL:		
MC-1 ↛ MC	MCDC7 = MIT/1	
sustain PH6 until MC = 0	BRPH6 = FAMDS.PH6.NMC2.NBRPH10.NFSHEX	

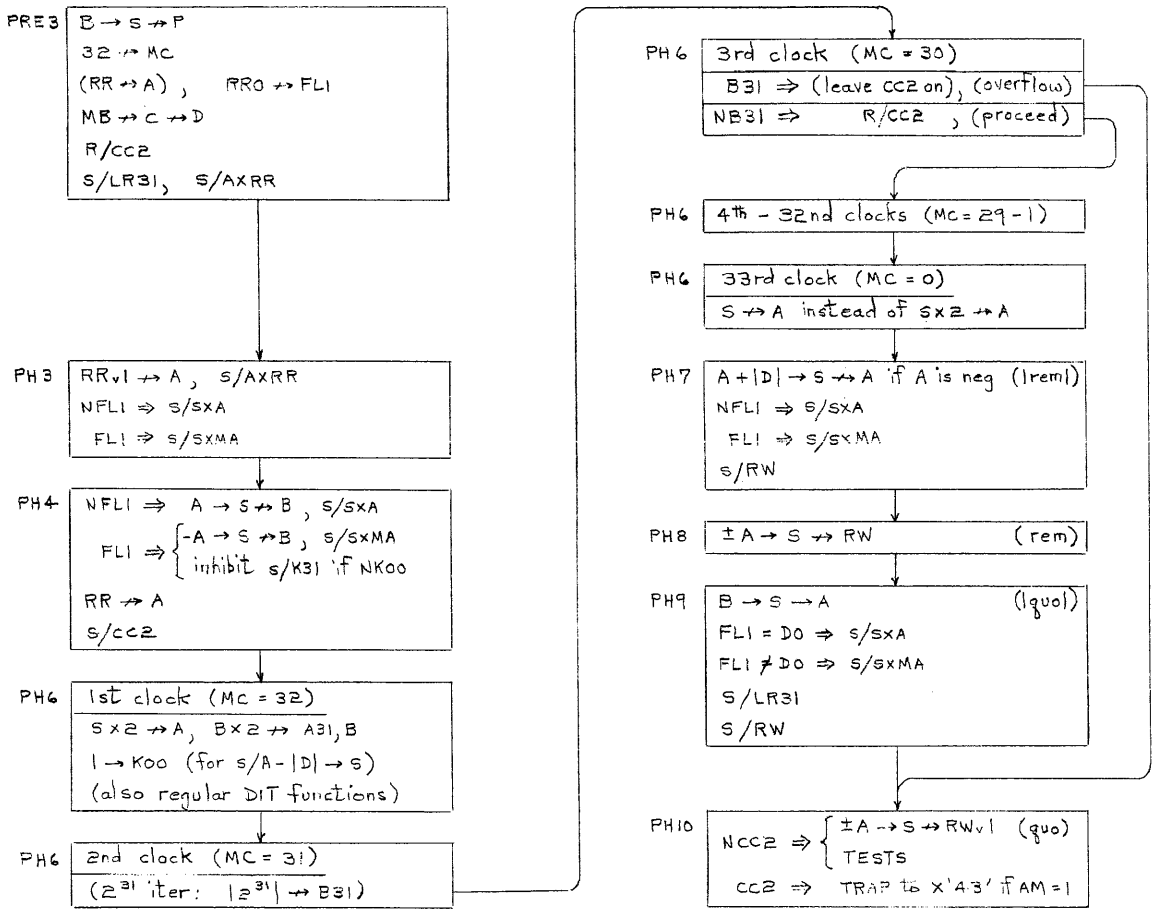
Sheet 2 of 3

MNEMONIC: MW, MI, MH

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(Cont'd)		
	ON THE FIRST CLOCK: 0's → A, B0001	(MC=16 if MW, MI; MC=8 if MH) (nothing has been preset → S)	 (clear accumulator)
	ON THE NEXT TO FINAL CLOCK: B30 → B3031 (B2829=0 at this time, hence cannot interfere with the above	(MC=1) S/B3031 = B30.MIT.(MC=1)	 (extend sign 2 bit posit.) due to BXBR2
	ON THE FINAL CLOCK: MIT/1 sets up the sign iteration logic, the result of which will be on S during the next clock. (will be S/A-D→S if (negative multiplier).(BC31=0); otherwise will be S/A→S) S/MRQ (for next instruction) S/RW (insig if NMH.Rodd) MH ⇒ S/PH10, S/DRQ NMH ⇒ advance to PH7	(MC=0) S/MRQ/1 = FAMDS.PH6.NBRPH6.NIOEN S/RW = MIT.MCZ BRPH10 = FUMH.PH6.MCZ, S/DRQ = BRPH10 (NBR is high)	 (TESTS next clock)
	I/O SERVICE CALL: IOEN6 (enable entry) inhibit S/PH6 if IOEN	 IOEN6 = FAMDS.PH6.NFPRR.NFSHEX.IOEN6/1 ← MC0005Z S/PH6 = BRPH6.NCLEAR.NIOEN	(excludes final 4 iter.)
	when returning from I/O (at (S/PH6/IO) time), the C→D function - normally performed by I/O logic - is inhibited to enable C→D or 2C→D to take place (MIT/1 is raised by (S/PH6/IO)); also, 2 product bits (stored in FL1,2) are put in B0,1 as B shifts right. inhibit C→D FL1 → B0 FL2 → B1	 DXC = IOPH1.SW13.NBXBR2 ← low S/B0 = BXBR2.S30/1 ← FL1.(S/PH6/IO) S/B1 = BXBR2.S31/1 ← FL2.(S/PH6/IO)	
PH7	(bypassed if MH) A → S or A-D → S S → RW (store 2^{63}-2^{32} of prod.) S → A S/T8L NB0 ⇒ S/A→S B0 ⇒ S/NA→S (for magnitude test) TESTS (including B register) S/PH9	 (preset in PH6) AXS = (FAMULNH.PH7) S/T8L = (") S/SXA = (").NB0 S/SXNA = (").B0 TESTS = (") TESTS/1 = (") BRPH9 = (")	 m.s.w. of product (NS0031Z is slow (PH9)) (cc3, cc4 control)
PH9	(bypassed if MH) S/CC2 if S≠0 (i.e. if the top 33 product bits are not all the same) S/RW S/LR31 S/B→S (to store l.s.w. of prod.) S/DRQ	S/CC2 = NS0031Z.(S/CC2/NZ) ← FAMULNH.PH9 S/RW = FAMDS.PH9 S/LR31 = FAMULNH.PH9 S/SXB = FAMULNH.PH9 S/DRQ/1 = PH9	(T8 timing)
PH10	NMH ⇒ B → S S → RW∨1 MH ⇒ A → S or A-D → S S → RW∨1 TESTS ENDE	(preset in PH9) (preset in PH6) (RW preset in PH6, R31 set in PRE3) TESTS = FUMH.ENDE ENDE = PH10.EXC	2^{31}-2^{0} of product (replaces 2^{63}-2^{32} if Rodd) entire product 2^{31}-2^{0} (cc3, cc4 control)

Sheet 3 of 3

MNEMONIC: MW, MI, MH



"NFADIVH" (DW with even R) : R, Rv1 ÷ EW ↛ Rv1 , remainder ↛ R (sign matched to numerator)

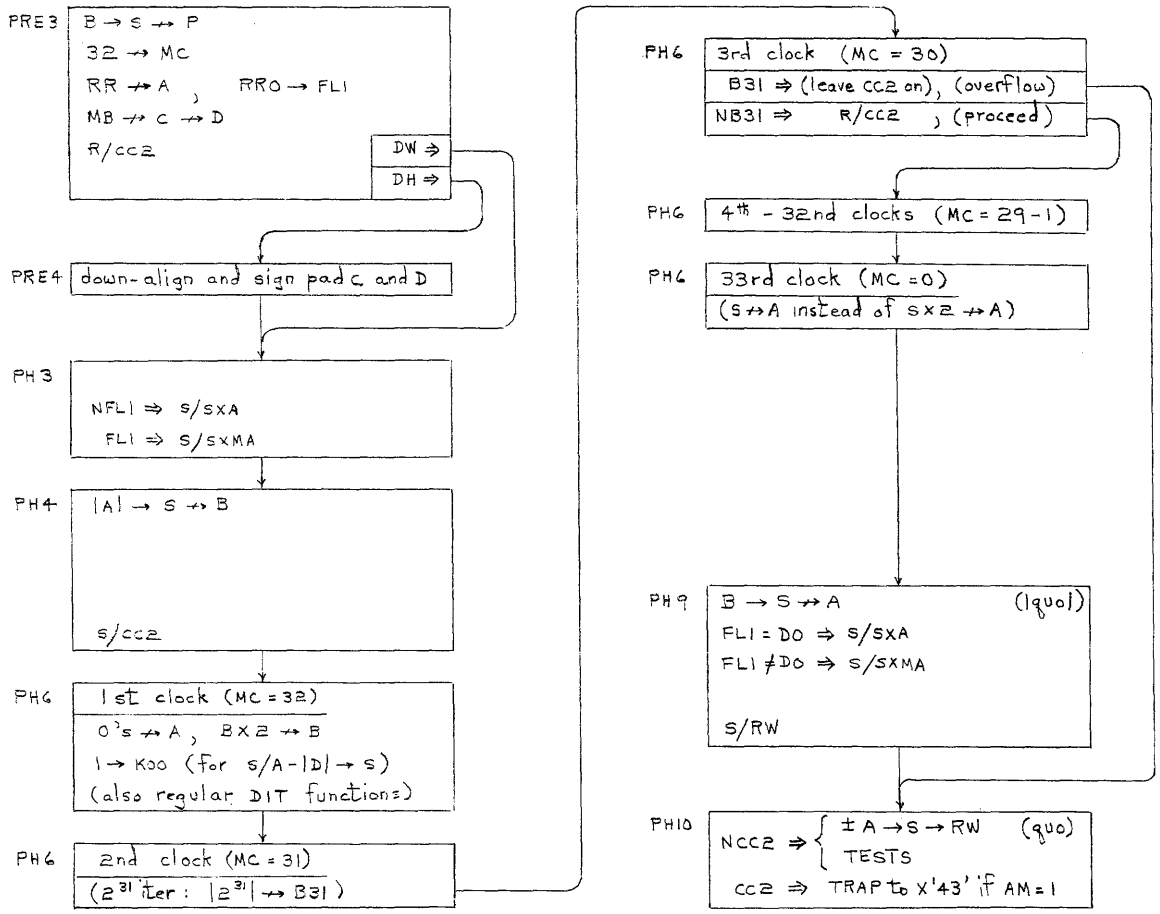
DIVIDE : general sequence of events

1) Put 2 x |numerator| in AB (2^0 bit being in B30 in all cases)
2) Put denominator in D (down-aligned if DH) (i.e. 2^0 bit is in D31)
3) iterate, left-shifting the |quotient| into B. (Indicate overflow if |quotient| ≥ 2^{31})
4) store remainder in R if DW with even R (otherwise discard it) } don't store if overflow (CC2=1)
5) store quotient in Rv1 if DW or R if DH }

Differences in nomenclature :

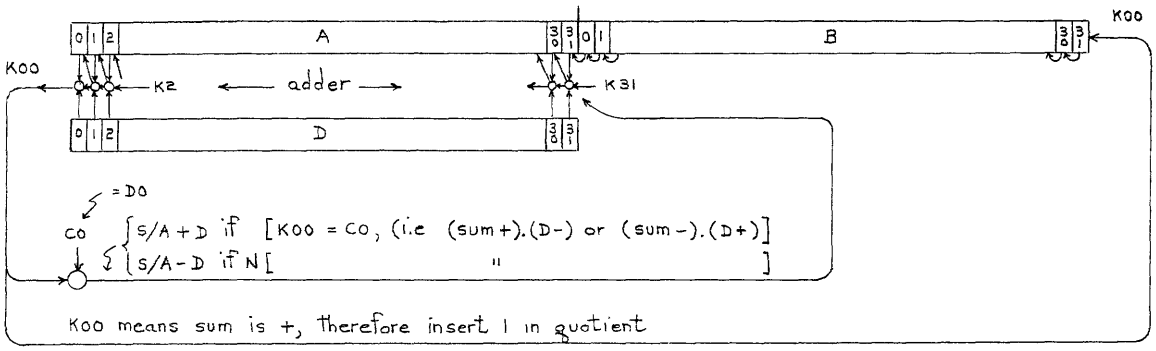
Reference Manual : DIVIDEND ÷ DIVISOR

Hardware : NUMERATOR ÷ DENOMINATOR



"FADIVH" (DW with odd R) : R ÷ EW ↛ R, (discard remainder)
or DH : R ÷ EHW → R, (" ")

Register organization during DIVIDE iterations:



PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
	Family signals (FUDW.NR31): [DW with even R]	= 0U3.0L6.NR31 operation: R,Rv1 ÷ EW → Rv1, remainder → R	
	FADIVH: [DW with odd R or DH]	= 0U3.0L6.R31 + 0U5.0L6 operation: R ÷ EW or EHW → R, (discard remainder)	
	FADIV [DW or DH] FAMDS [DW or DH or MI, MW, MH, S, SF]		
PRE3	B → S S → P	Sn = Bn.SXB, (S/SXB = PRE/12) PXS = FAMDS.PRE3.NANLZ.NFAIM	
	RR → A RR0 → FLI (store numerator sign) MB → C → D (denominator)	An = RRn.AXRR, (S/AXRR = PRE/12) S/FLI = PRE3.RR0 DXC = PREOPER.PRE3	
	32 → MC R/CC2 (for overflow test and control)	S/MC2 = FADIV.PRE3 R/CC2/1 = FAMDS.NFUMH.PRE3	
	S/LR31, S/AXRR } if [DW with even R]	S/LR31 = (FUDW.NR31.PRE3) S/AXRR = (")	} for numer. l.s.w → A
	S/PH3 if DW S/PRE4 if DH	BRPH3 = FAMDS.NBRPHS.NANLZ.PRE/34 BRPRE4 = PRE3.NANLZ.NPRE/34	} PRE/34 high if DW
PRE4	(entered only if DH) The HALF WORD in C and D is down-	aligned and sign-padded as per chart	on "PREPARATION"
	S/PH3 when finished	BRPH3 = FAMDS.NBRPHS.NANLZ.PRE/34	
PH3	RRv1 → A if [DW with even R] (A retains R if N["])	(preset in PRE3)	(numer. l.s.w.)
	NFLI ⇒ S/A → S FLI ⇒ S/-A → S } S/|A| → S	S/SXA = NFLI.(S/SX/FLI) ← FADIV.PH3 S/SXMA = FLI.(") ←	+ FUDW.NR31.PH3 (redun. - mech. conv.)
	S/AXRR if [DW with even R]	S/AXRR = FUDW.NR31.PH3	
PH4	|A| → S S → B (|numer.|)	(preset in PH3) BXS = FADIV.PH4	(l.s.w. if DW with even R)
	NFLI ⇒ S/A → S FLI ⇒ S/NA+Carry → S don't set K31 if K00 = 0 } if [DW with even R]	S/SXA = NFLI.(S/SX/FLI) S/SXMA = FLI.(") ← FUDW.NR31.PH4 N(S/K31/1) will be high if NK00, holding N(") = N(K00.(S/K31/3) + (S/K31/2).(S/K31/3)), where (S/K31/3) is high and (S/K31/2) = N(FAMDS.PH4), hence is low, Therefore N(S/K31/1) reduces to NK00	NK31 on (i.e. K31 off)
	RR → A if [DW with even R] S/CC2 S/PH6	An = RRn.AXRR S/CC2/1 = (FADIV.PH4) BRPH6 = (")	(m.s.w. of numer.)

Sheet 1 of 3

MNEMONIC: DW (36), DH (56)

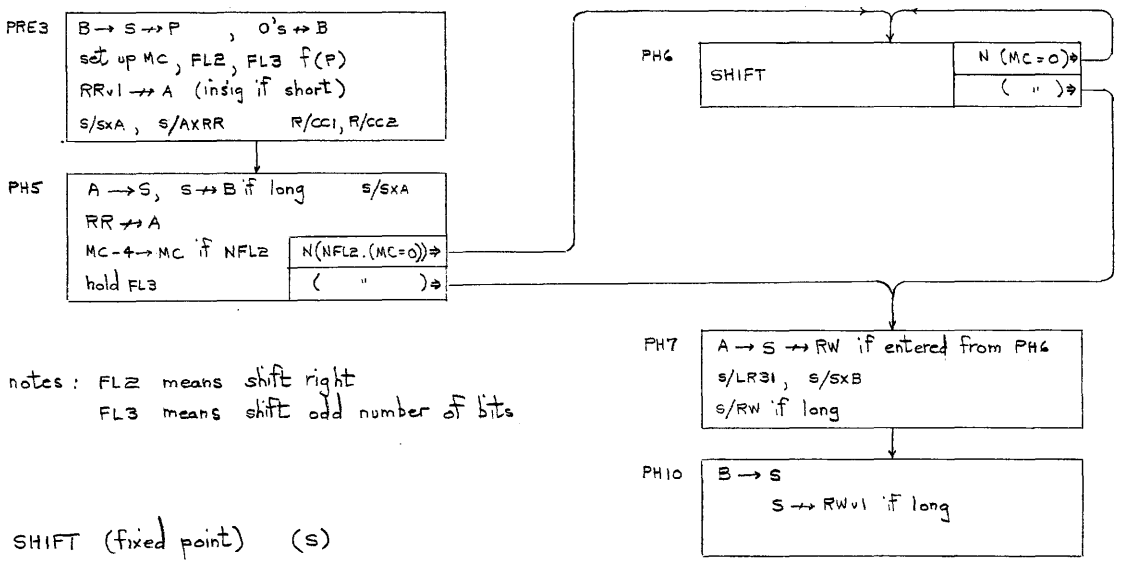
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(ITERATIONS PHASE) - 33 clocks		
	CONTROL SIGNALS:		
	DIT/1 (handles preset logic)	DIT/1 = FADIV.NIOEN.(PH6 + (S/PH6/IO)).N(FADIVH.MC≠)	
	CONTROL FUNCTIONS:		
	MC - 1 → MC	MCDC7 = DIT/1	(initially set to 32)
	sustain PH6 until MC=0 or until overflow detected	BRPH6 = FAMDS.PH6.NFSHEX.NMC≠.NBRPH10	
	REGISTER CONTROL, etc.:		
	S/A+D → S if CO = K00	S/SXAPD/1 = DIT/1.N(CO⊕K00)	K00 means positive residue
	S/A-D → S if CO ≠ K00	S/SXAMD/1 = DIT/1.(CO⊕K00)	
	S0131 → A0030, except on final clock	AXSL1 = FADIV.PH6.NMC≠	residue x 2 → AB
	B0 → A31, except on final clock	S/A31 = AXSL1.A31EN/1 ← B0.FAMDS.PH6	
	B0131 → B0030	BXBL1 = FADIV.PH6	
	K00 → B31	S/B31 = BXBL1.B31EN/1 ← K00.FADIV	|quotient| bit = 1
	ON THE 1st CLOCK:	(MC = 32, CC2 = 1)	
	1 → K00	K00 = G0003 = FADIV.K00/1 ← CC2.MC2	(to cause S/A-|D| → S)
	NA + Carry → S if [DW with even R] FL1		
	A → S if ["] NFL1	preset in PH4	2^{63}-2^{32} of |numer.|
	0's → S if N["]		
	0's if N[DW with even R]		
	S0131 → A0030, B0 → A31	(see REGISTER CONTROL)	(2x|numer.| → AB)
	B0131 → B0030		
	1 → B31 (insig)		(K00 is forced high)
	ON THE 2nd CLOCK:	(first iteration - for |quotient| 2^{31} bit)	(MC = 31)
	A-|D| → S, SX2 → A0030	(see REGISTER CONTROL)	
	BX2 → A31, B0030		
	K00 → B31 (2^{31}|quotient| bit)		(K00 means + residue)
	ON THE 3rd CLOCK:	(OVERFLOW TEST)	(MC = 30)
	B31 = 1 means OVERFLOW since it indicates |quotient| ≥ 2^{31}		
	B31 ⇒ raise DIVOVER	DIVOVER = B31.(DIT.(MC=30)) ← FADIV.CC2.MC6.NMC7	
	B31 ⇒ leave CC2 on	(no reset active)	(was set in PH4)
	B31 ⇒ S/PH10, S/MRQ, S/DRQ	BRPH10 = DIVOVER, S/MRQ/1 = FAMDS.PH6.NBRPH6.NIOEN, S/DRQ = BRPH10	
	B31 ⇒ stop sustaining PH6	BRPH6 = NBRPH10.FAMDS.PH6.NFSHEX.NMC≠	
	NB31 ⇒ R/cc2	R/cc2/2 = NB31.(DIT.(MC=30)) ← FADIV.CC2.MC6.NMC7	
	NB31 ⇒ (continue iterations)		(quo. 2^{30} → B31 this clock)
	4th - 32nd CLOCKS:		
	(iterations for quo. 2^{29} -- 2^{1} → B)		(MC = 29 thru 1)

MNEMONIC: DW, DH

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(Cont'd)		
	ON THE 33rd (final) CLOCK:		(MC=0)
	inhibit SX2 ↛ A	AXSLI = (FADIV.PH6).NMCZ	
	enable S ——↛ A	AXS = (").MCZ	(residue X 1 ↛ A)
	(B still shifts left with 2° gue	bit ↛ B31)	
	s/MRQ (for next instruction)	s/MRQ/1 = FAMDS.PH6.NBRPH6.NIOEN	
	stop sustaining PH6	BRPH6 = NMCZ.FAMDS.PH6.NFSHEX.N	BRPH10
	N[DW with even R] ⇒	(discard remainder)	
	inhibit S/A±D → S	DIT/1 is qualified by N(FADIVH.MCZ)	(see REGISTER CONTROL)
	s/PH9	BRPH9 = FADIVH.MCZ	
	[DW with even R] ⇒	(save remainder)	
	s/A±D → S	DIT/1 is still high	
	advance to PH7	NBR is high	
			(exclude 1st 3 and final 4 iterations)
	I/O SERVICE CALL:		N(FADIV.CC2+MC0005Z)
	IOEN6 (enable entry)	IOEN6 = FAMDS.PH6.NFPRR.NFSHEX.NIO	EN6/1
	inhibit s/PH6 if IOEN	s/PH6 = BRPH6.NCLEAR.NIOEN	
	DIT/1, used for preset logic, is inhibited by IOEN on exit and enabled by		(s/PH6/IO) on return,
	hence is one clock ahead	of PH6 when PH6 is interrupted	
	B31 → K00 when (s/PH6/IO)	K00 = G0003 = FADIV.K00/1 ← B31.(s/PH6/IO)	(for S/A±D → S)
PH7	(entered if [DW with even R]. no overflow)		
	A ±D → S	(preset on last clock of PH6)	
	S ↛ A if A is neg.	AXS = FUDW.NR31.PH7.NB31	
	NFLI ⇒ S/A → S	s/SXA = NFLI.(S/SX/FLI) ┐ FADIV.PH7	
	FLI ⇒ S/-A → S	s/SXMA = FLI.(") ┘	
	s/RW	s/RW = FUDW.NR31.PH7	(restored remainder)
PH8	(entered if [DW with even R]. no overflow)		
	S ↛ RW	(PRESET IN PH7)	remainder ↛ R with polarity matched to original numerator.
PH9	B → S (Phase by-passed if overflow)	SXB = (FADIV.PH9).NDIS	
	S ↛ A	AXS = (")	
	s/A → S if +/+ or -/-	s/SXA = (").N(FLI ⊕ DO)	(quotient)
	s/-A → S if +/- or -/+	s/SXMA = (").N(s/SXA)	
	s/RW	s/RW = FAMDS.PH9	
	s/LR31 if [DW with even R]	s/LR31 = FUDW.NR31.PH9	
	s/DRQ	s/DRQ/1 = PH9	
PH10	NO OVERFLOW (CC2=0) ⇒	(entry from PH9)	
	±A → S		
	S ↛ RWv1 if [DW with even R]	(preset in PH9)	quotient ↛ Rv1
	S ↛ RW if N["]		" ↛ R
	TESTS (CC3, CC4 control)	TESTS = FADIV.ENDE.NCC2	
	OVERFLOW (CC2=1) ⇒	(entry from PH6 in which RW is not set)	
	(don't change SCRATCHPAD, CC	3 or CC4)	
	TRAP to x'43' if AM=1	s/TRAP = ENDE.CC2.AM.0	VERIND ← = FADIV + ···
		s/TR30 = s/TR31 = PH10.CC2.AM.0	VERIND ←
	ENDE	ENDE = PH10.EXC	

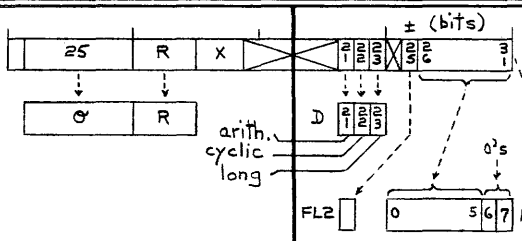
Sheet 3 of 3

MNEMONIC: DW, DH



notes : FL2 means shift right
FL3 means shift odd number of bits

SHIFT (fixed point) (S)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END of PREP (PRE3)		Family signals : FUS : S FASH : S, SF FAMDS : S, SF,	MI, MW, MH, DW, DH
	P2S ⇒ { 1 → FL2 (store direction) NP2630 → MC0105 P31 → FL3 (odd no. bits)	S/FL2 = PRE3.P2S, (R/FL2 = CLEAR) MCXNPL1 = FASH.PRE3.P2S S/FL3 = PRE3.P31 MCXPL2 = FASH.PRE3.NP2S	P2S = 1 means right { even ⇒ (2|P|-4) → MC odd ⇒ (2|P|-2) → MC 4|P| → MC
	NP2S ⇒ P2631 → MC0005		
	B → S S → P	Sn = Bn.SXB, (S/SXB = PRE/12) PXS = FAMDS.PRE3.NANLZ.NFAIM	= (S/LR31/1) = OU2.OL5)
	RRv1 → A (insig if short) 0's → B (for short left shift) S/A → S (insig if short) S/AXRR (for RR → A)	An = RRn.AXRR, (S/AXRR = PRE/12 BXZ = FAMDS.PRE3 S/SXA = FUS.PRE3 S/AXRR = FUS.PRE3	, S/LR31 = PRE/12.(S/LR31/12)
	R/CC1 (for parity check) R/CC2 (for overflow test)	R/CC1/1 = FASH.PRE3 R/CC2/1 = FAMDS.NFUMH.PRE3	
	S/PH5	BRPH5 = FUS.PRE3	

Sheet 1 of 3

MNEMONIC: S (25)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5	A → S	Sn = An . AXRR	
	S ↛ B if long	BXS = FUS.PH5.D23	(B remains = 0 if short)
	RR ↛ A	An = RRn . AXRR	
	sustain FL3 (on if bit count is odd)	R/FL3 = N(FUS.PH5)	(for first clock of PH6)
	S/A → S	S/SXA = FASH.PH5	
	LEFT SHIFT CASE:		
	MC - 4 ↛ MC (4|P|-4 ↛ MC)	MCDC7 = (FASH.PH5.NFL2)	(FUS holds MC6 and MC7 = 0, hence down-counting starts at MC5)
	shift count = 0 ⇒ { S/PH7 / S/MRQ	BRPH7 = (").MCZ S/MRQ/1 = (").MCZ	(for next instruction)
PH6	(SHIFTING PHASE)		
	CONTROL SIGNALS:		
	SFT	SFT = FASH.NIOEN.PH6	
	SUM BUS:	{ S/SXA = FASH.PH5	(preset)
	A → S	+ SFT	(sustain)
		+ FASH.(S/PH6/IO)	(return from I/O service)
	INITIAL REGISTER CONTENTS:		

short	R	(0's)
	A	B
long	R	R∨1

LEFT SHIFT CASE: Initially MC contains 4|P|-4, where P is the number of bit positions to be shifted. AB shifts left by 1's until MC has down-counted (by 4's) to zero.

REGISTER CONTROL:

S0131 ↛ A0030
B0 ↛ A31 (insig if short)
A0 ↛ A31 if short cyclic
B0131 ↛ B0030
A0 ↛ B31 if long cyclic

AXSL1 = SFT.NFL2.NFSHEX
S/A31 = AXSL1.A31EN/1 ← { = B0.FAMDS.PH6 + A0.(FUS.D22.ND23).PH6
BXBL1 = SFT.NFL2.NFSHEX
S/B31 = BXBL1.B31EN/1 ← A0.(FUS.D22.D23)

CONDITION CODES:

Toggle CC1 as 1's shift through A0 (parity check)
1 ↛ CC2 if overflow

FUS/1 = SFT.NFL2.FUS (i.e. shift left)
S/CC1 = FUS/1.A0.NCC1
R/CC1/2 = FUS/1.A0
S/CC2 = FUS/1.(A0 ⊕ A1)

RIGHT SHIFT CASES:

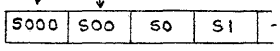
1) Odd no. of bits: (FL3 is on during first clock.) Initially MC contains 2|P|-2. AB shifts right one position and MC counts down by 4, yielding 2|P|-6. When the shift is finished, shifting takes place by 2's until MC has down-counted (by 4's) to zero.

REGISTER CONTROL:

S00 ↛ A0
S0030 ↛ A0131
S31 ↛ B0 if long
B0030 ↛ B0131

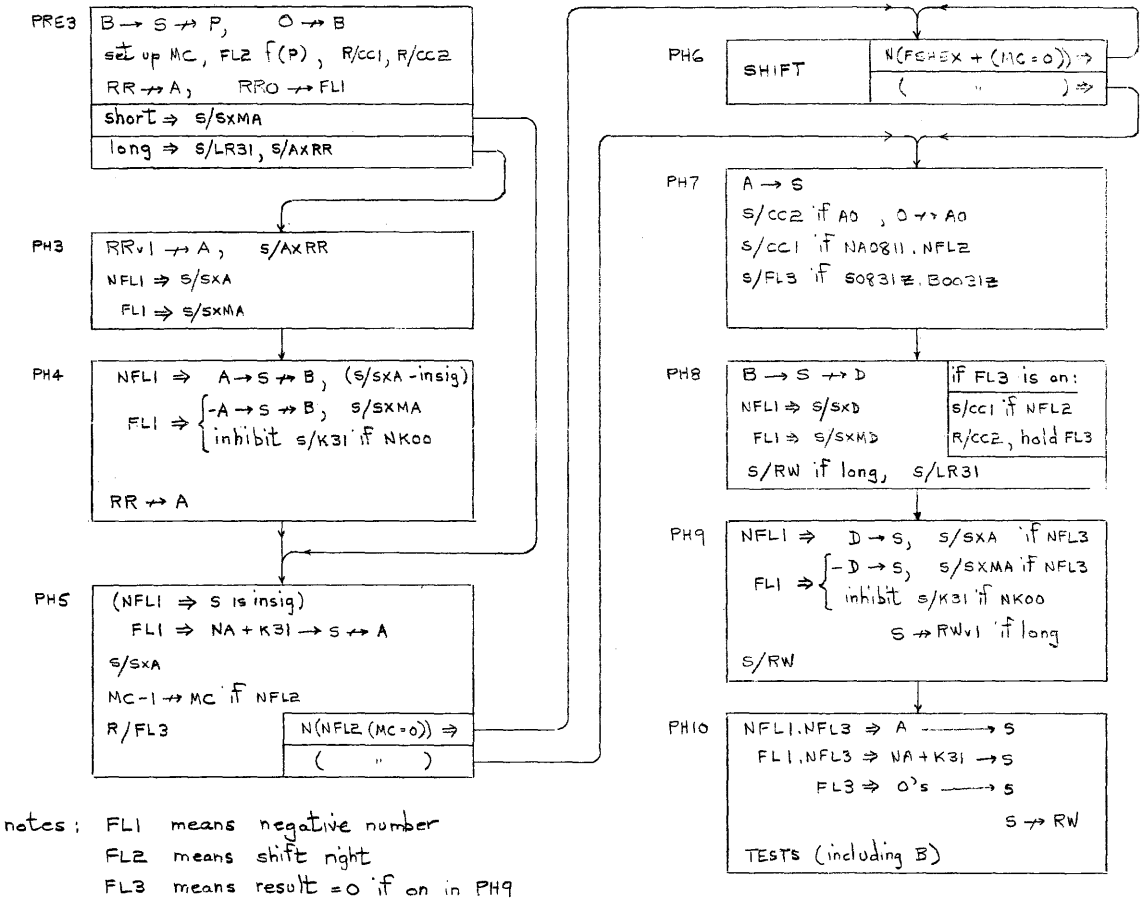
AXSR1 = SFT.FL2.FL3 (S00 described next page)
S/B0 = BXBR1.S31/1 ← S31.B0001EN/1 ← FASH.D23
BXBR1 = SFT.FL2.FL3

MNEMONIC: S

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(Cont'd)		
	2) Even no. of bits: (FL3 is off during first clock) Initially MC contains right by 2's until MC has down-counted (by 4's) to zero)		2|P|-4. AB shifts
	REGISTER CONTROL:		
	S000 → A0	AXSR2 = SFT.FL2.NFL3.NFSHEX	(S000 described below)
	S00 → A1		(S00 " ")
	S0029 → A0231		
	S30 → B0 (if long)	S/B0 = BXBR2.S30/1 ← S30.B0001EN/	1 ← FASH.D23
	S31 → B1 (if long)	S/B1 = BXBR2.S31/1 ← S31.B0001EN/	1 ← FASH.D23
	B0029 → B0231	BXBR2 = SFT.FL2.NFL3.NFSHEX	
	PHASE CONTROL, etc:		
	MC-4 → MC	MCDC7 = SFT (FUS holds MC6 and MC7	=0, hence down-count by 4's)
	(MC ≠ 0) ⇒ sustain PH6	BRPH6 = FAMDS.PH6.NMCZ.NFSHEX.NBRPH10	(keep shifting)
	(MC = 0) ⇒ advance to PH7	(NBR is high)	
	s/MRQ (for next instr.)	S/MRQ/1 = FAMDS.PH6.NBRPH6.NIOEN	
	s/RW (to store A)	S/RW = SFT.MCZ.FUS	
	MISC:		
	sum bus extension:	short cyclic / long cyclic	arithmetic
	S000 = A30.	(FUS.D22.ND23) + B30.(FUS.D22.D23) + A00 ←	(A0.N(FUS.ND21))
	S00 = A31.	(") + B31.(") + A00 ←	(")
			
			(MC<4) ↓
	I/O SERVICE CALL:		
	IOEN6 (enable entry)	IOEN6 = FAMDS.PH6.NFPRR.NFSHEX.IOE	N6/1 ← NMC0005Z
	inhibit s/PH6 if IOEN	S/PH6 = BRPH6.NCLEAR.NIOEN	
	" SFT " "	SFT = FASH.PH6.NIOEN	
PH7	A → S	Sn = An.AXRR	
	S → RW	RWn = Sn.RW	
	(note: S→RW does not take place if PH6 was skipped, but that is the case of shift zero positions, hence there would be no change in the contents of R.)		
	s/LR31 (for B→S→RW|1 if long)	S/LR31 = (FUS.PH7)	
	s/RW if long (for B→S→RW|1 if long)	S/RW = (").D23.NR31 ← inhib	its store when R is odd, hence R will contain the M.S.W. in this case.
	s/B→S (for B→S→RW|1 if long)	S/SXB = FASH.PH7	
	S/DRQ	S/DRQ = BRPH10	
	S/PH10	BRPH10 = FUS.PH7	
PH10	B→S, S→RW|1 if long	(preset in PH7)	
	ENDE	ENDE = PH10.EXC	

Sheet 3 of 3

MNEMONIC: S



notes: FL1 means negative number
FL2 means shift right
FL3 means result = 0 if on in PH9

SHIFT FLOATING (SF)

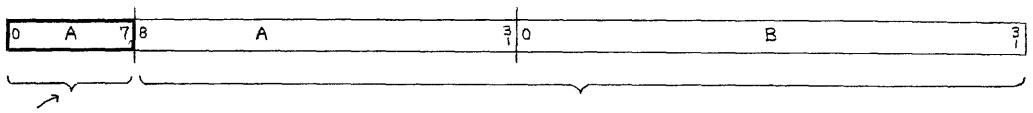
SHIFT FLOATING: general sequence of events

1) Put the absolute value of the number in AB
2) Shift the mantissa by the number of hexes specified unless early stop:
   LEFT: decrement exponent, stop if exponent underflow (s/cc2), or if mantissa = 0 or is ≥ 16 (s/cc1).
   RIGHT: increment " , " " " overflow (s/cc2), " " " = 0.
3) Store the result (negated if originally negative). (Store true zero if mantissa = 0.)

note: nomenclature differences:

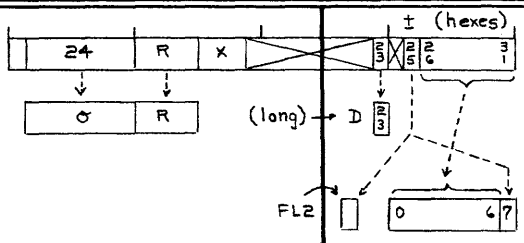
Reference manual		Hardware
CHARACTERISTIC	-	EXPONENT
FRACTION	-	MANTISSA

Register organization:



(exponent: counts up or down but does not shift)

(|mantissa|: shifts left by 1's or right by 2's. B will = 0 throughout a short shift)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
END of PREP PRE3		Family signals: FUSF : SF FASH : SF, S FAMDS : SF, S,	MI, MW, MH, DW, DH
	P25 ⇒ { 1 ↛ FL2 (store direction) NP2631 ↛ MC0106, 1 ↛ MC7 NP25 ⇒ P2631 ↛ MC0005	S/FL2 = PRE3.P25; (R/FL2 = CLEAR) MCXNPL1 = FASH.PRE3.P25 MCXPL2 = FASH.PRE3.NP25	P25 = 1 means RIGHT (2|P|-1) ↛ MC 4|P| ↛ MC
	B → S S ↛ P	Sn = Bn.SXB, (S/SXB = PRE/12) PXS = FAMDS.PRE3.NANLZ.NFAIM	
	RR ↛ A RR0 ↛ FL1 (store sign) 0's ↛ B (for short cases)	An = RRn.AXRR, (S/AXRR = PRE/12) S/FL1 = PRE3.RR0, (R/FL1 = CLEAR) BXZ = FAMDS.PRE3	(replaced in PH3 if long)
	short ⇒ { S/-A → S S/PH5	S/SXMA = (FUSF.PRE3.ND23) BRPH5 = (")	(for |A| → S ↛ A in PH5 in case A is neg.)
	long ⇒ { S/AXRR S/LR31 S/PH3 } for RRv1 ↛ A	S/AXRR = (FUSF.PRE3.D23) S/LR31 = (") BRPH3 = FAMDS.PRE/34.NBRPH5.NANLZ	
	R/CC1 (for normalization test) R/CC2 (for exp. over/underflow test)	R/CC1 = FASH.PRE3 R/CC2 = FAMDS.NFUMH.PRE3	
PH3	(entered if long only) RRv1 ↛ A (l.s.w.) NFL1 ⇒ S/A → S FL1 ⇒ S/-A → S S/AXRR (for m.s.w)	An = RRn.AXRR S/SXA = NFL1.(S/SX/FL1) S/SXMA = FL1.(") ← FUSF.D23.PH3 S/AXRR = FUSF.D23.PH3	(LR31 is high) (S/|A| → S, (l.s.w.))
PH4	(entered if long only) |A| → S S ↛ B (l.s.w.) NFL1 ⇒ S/A → S (insig.) { FL1 ⇒ S/NA + Carry → S { don't set K31 if K00 = 0 RR ↛ A (m.s.w)	(see PH3) BXS = FUSF.D23.PH4 S/SXA = NFL1.(S/SX/FL1) S/SXMA = FL1.(") ← FUSF.D23.PH4 { N(S/K31/1) will be high if NK00, holding N(") = N(K00.(S/K31/3) + (S/K31/2) where (S/K31/3) is high, and (S/K31/2) = N(FAMDS.PH4), hence is low, Therefore N(S/K31/1) reduces An = RRn.AXRR	(S/|A| → S, (m.s.w)) NK31 on (i.e. K31 off) .(S/K31/3)) to NK00
PH5	-A ——→ S if short NA + Carry → S if long and no. is neg. S ↛ A if no. is neg. S/A → S MC-1 ↛ MC if left shift shift count = 0 ⇒ { S/PH7 S/MRQ (for next instr.)	(see PRE3) (see PH4) AXS = FUSF.PH5.FL1 S/SXA = (FASH.PH5) MCDC7 = (").NFL2 BRPH7 = (").NFL2.MCZ S/MRQ/1 = (").NFL2.MCZ	(A → S if pos. no. - insig) (don't change A if pos. no.) (4|P|-1 ↛ MC) (NFL2 term needed for fixed point shift)

Sheet 1 of 3

MNEMONIC: SF (24)

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(SHIFTING PHASE)		
	CONTROL SIGNALS		
	FUSF/1	FUSF/1 = FUSF.NIOEN.(PH6 + PH7)	(floating shift only)
	SFT	SFT = FASH.NIOEN.PH6	(any shift)
	FNORM (mantissa is normalized)	FNORM = (A8+A9+A10+A11).NFL2.MC6.MC7	(left only, on 1 mod.4 clocks)
	FL3 (mantissa = zero indicator)	S/FL3 = FUSF/1.S0815Z.S1631Z.B0031Z	
	FSHEX (stop shifting)	FSHEX = FUSF.FNORM	(normalized on left shift)
		+ FUSF.FL3	(mantissa = zero)
		+ FUSF.A0	(exponent has under/overflowed)
	SUM BUS		
	A → S0831	S/SXA = FASH.PH5	(preset)
		+ SFT	(sustain)
		+ FASH.(S/PH6/IO)	(return from I/O service)
	(0's → S0007 - exponent field)	PRXAD/0 and PRXAND/0 are each inhibited by FUSF/1	
	A REGISTER EXPONENT FIELD: A0007 are isolated from the rest of A by inhibiting the reset enable (AX/0) by FUSF/1, also since S0007 are held = 0, no 1's can set A0007 via the shift paths. A0007 are under the control of AUC7 and ADC7 which count the exponent once for each hex shifted.		
	A0: if A0 becomes a 1 it signifies exponent overflow. It will be reset on the following clock by R/A0 = FUSF/1, after having performed it's functions (i.e. set cc2 and raise FSHEX). It must be cleared before preparing the result for scratch-pad for compatibility with E7.		
	INITIAL REGISTER CONTENTS: The AB registers contain the absolute value of the floating point number (with B containing zeros in the short case). The illegal number "minus 1" will yield a result of true zero.		
	LEFT SHIFT CASE: Initially MC contains 4|P|-1, where P is the number of hexes to be shifted. The mantissa in AB shifts left by 1's until the MC has down-counted to zero or until FSHEX stops the shift due to normalization, exponent underflow, or mantissa originally being zero.		
	REGISTER control:		
	S0831 → A0731 (S0007 = 0)	AXSL1 = SFT.NFL2.NFSHEX	(FNORM prevents S8 → A7 by raising FSHEX)
	B0 → A31	S/A31 = AXSL1.A31EN/1 ← B0.FAMDS.PH6	
	B0131 → B0030, (0's → B31)	BXBL1 = SFT.NFL2.NFSHEX	
	A0007 - 1 → A0007	ADC7 = FUSF/1.NFL2.NMC6.NMC7	(on 0 mod.4 clocks)
	S/CC1 if mantissa is normalized	S/CC1 = FUSF/1.FNORM	(on 1 mod.4 clocks)
	RIGHT SHIFT CASE: Initially MC contains 2|P|-1, where P is the number of hexes to be shifted. The mantissa in AB shifts right by 2's until the MC has down-counted to zero or until FSHEX stops the shift due to exponent overflow or mantissa originally being zero or being shifted off the end of AB if long or A if short.		
	REGISTER control:		
	S0829 → A1031 (0's → A0809)	AXSR2 = SFT.FL2.NFL3.NFSHEX	(NFL3 is for fixed point)
	S30 → B0, S31 → B1 } if long	S/B0 = BXBR2.S30/1 ← S30.B0001EN/1 ← FASH.D23	
		S/B1 = BXBR2.S31/1 ← S31.B0001EN/1 ← FASH.D23	
	B0029 → B0231	BXBR2 = SFT.FL2.NFL3.NFSHEX	(NFL3 is for fixed point)
	A0007 + 1 → A0007	AUC7 = FUSF/1.FL2.NMC7.NCC2	(on 0 mod.2 clocks)
		↑ (prevents surplus count in PH7 in case where exp. overflow causes FSHEX)	

Sheet 2 of 3

MNEMONIC: SF

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6	(cont'd) PHASE CONTROL: MC - 1 → MC (MC ≠ 0).NFSHEX ⇒ sustain PH6 (MC = 0) + FSHEX ⇒ advance to PH7 s/MRQ MISC: s/cc2 if exponent under/overflow I/O SERVICE CALL: IOEN6 (enable entry) inhibit s/PH6 if IOEN (SFT and FUSF/1 are each inhibited by IOEN)	MCDC7 = SFT BRPH6 = FAMDS.PH6.NMCZ.NFSHEX.NBRPH10 (NBR is high) s/MRQ/1 = FAMDS.PH6.NBRPH6.NIOEN s/CC2 = FUSF/1.A0 IOEN6 = FAMDS.PH6.NFPRR.NFSHEX.IOEN6/1 ← NMC0005Z s/PH6 = BRPH6.NIOEN.NCLEAR	(for next instruction) (MC < 4) ↓
PH7	(PROBE RESULT) (FUSF/1 is still high) s/cc1 if left and mantissa normalized s/cc2 if exponent over/underflow R/A0 (render sign bit +) s/FL3 if mantissa = 0 s/B → s	FUSF/1 = FUSF.NIOEN.(PH6 + PH7) s/CC1 = FUSF/1.FNORM ← (A8+A9+A10+A11).NFL2.MC6.MC7 s/CC2 = FUSF/1.A0 R/A0 = FUSF/1 s/FL3 = FUSF/1.S0815Z.S1631Z.B0031Z s/SXB = FASH.PH7	(will be low (in left case) when FSHEX caused by exp. underflow) (A is on S0831) ↑ always high if short
PH8	B → S S → D NFL1 ⇒ s/D → s FL1 ⇒ s/-D → s s/LR31 s/RW if long IF MANTISSA = 0: s/cc1 if left shift R/cc2 hold FL3 on (for S contr. in PH9)	Sn = Bn.SXB DXS = (FUSF.PH8) s/SXD = (").NFL1 s/SXMD = (").FL1 s/LR31 = (") s/RW = (").D23 s/cc1/1 = (").FL3.NFL2 R/cc2/1 = (").FL3 R/FL3 = N(")	(B = 0 if short) { negate l.s.w if original sign was negative. (0 is considered normalized) (cancel over/underflow ind.)
PH9	± D → S, S → RW/1 if long IF MANTISSA ≠ 0: NFL1 ⇒ s/A → s {FL1 ⇒ s/NA + Carry → s {don't set K31 if K00 = 0 s/RW s/DRQ	(preset in PH8) s/SXA = NFL1.(S/SX/FL1) s/SXMA = FL1.(") ← FUSF.PH9.NFL3 (s/K31/2) = N(FUSF.PH9) --- (see discussion in PH4) s/RW = FAMDS.PH9 s/DRQ/1 = PH9	(l.s.w.)
PH10	A or NA + Carry or 0's → S → RW s/cc3 if result > 0 s/cc4 " " < 0 ENDE	(preset in PH9) TESTS = (FUSF.ENDE) TESTs/1 = (") ENDE = PH10.Exc	(true zero if mantissa = 0) (includes B reg in test)

Sheet 3 of 3

MNEMONIC: SF

FLOATING POINT BOX OPTION

Table of contents:

σ decoding:

	$\overline{\sigma 2}$		$\sigma 2$	
$\overline{\sigma 6}.\overline{\sigma 7}$	FSL	1C	FSS	3C
$\overline{\sigma 6}.\sigma 7$	FAL	1D	FAS	3D
$\sigma 6.\overline{\sigma 7}$	FDL	1E	FDS	3E
$\sigma 6.\sigma 7$	FML	1F	FMS	3F

FAFL = $\overline{\sigma 1}.\sigma 3.\sigma 4.\sigma 5$

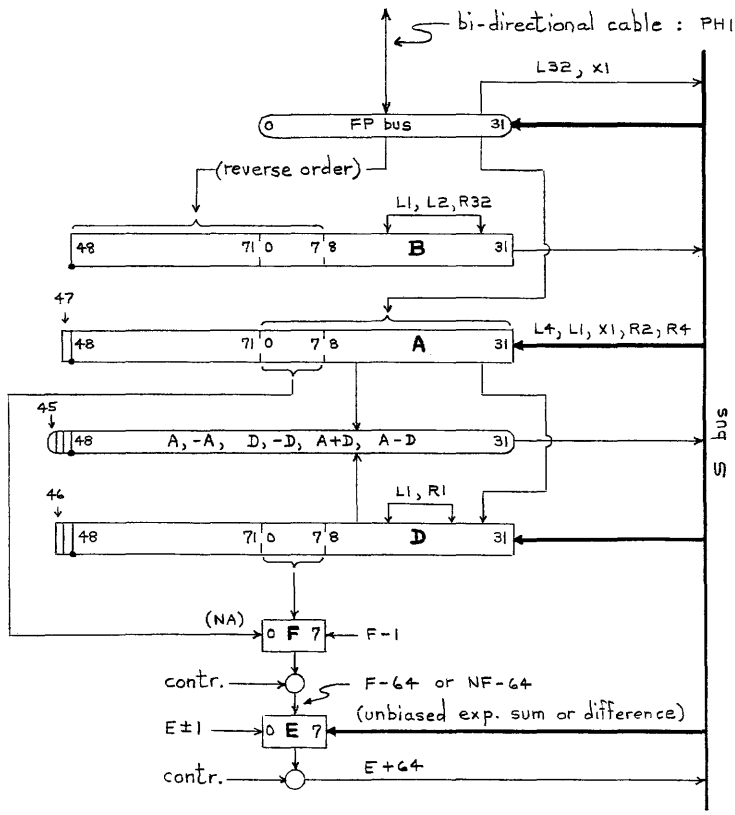
Notes on documentation:

1) CPU and BOX activity are combined on the sequence charts. *CPU functions are all printed on a slant.*
2) The "General sequence of events" notes on pages 302, 310, and 318 are intended only to give an impression of what is to take place. They cannot be either complete or accurate in every detail without becoming too complicated to serve their intended purpose.

Misc.:

1) To simulate absence of box: Ground 12F01 (kills FPCON), ground 23P07 (raises NFOPTION). (pts. in CPU)

OVERALL BLOCK OF BOX:



Principal use of registers:

	ADD/SUB	MUL	DIV
← B :	postnorm ctr.	multiplier (backwards)	quotient
← A :	augend, scaler	product, scaler	numerator, scaler
← D :	addend, sum	multiplicand	denominator
← F :	exponent buffer	iterations ctr.	iterations ctr.
← E :	align. ctr., sum exp.	prod. exp.	quo. exp.

Condition Code Settings for Floating-point Instructions

Condition Code 1	2	3	4	Meaning if no trap to location X'44' occurs	Meaning if trap to location X'44' occurs
0	0	0	0	A x 0, 0/A, or -A + A① with FN=1 (normal results)	*②
0	0	0	1	N < 0	*
0	0	1	0	N > 0	*
0	1	0	0	*②	divide by zero
0	1	0	1	*	overflow, N < 0 (always trapped)
0	1	1	0	*	overflow, N > 0 (always trapped)
③ 1	0	0	0	-A + A①	-A + A
1	0	0	1	N < 0 (> 2 postnormalizing shifts)	N < 0 (> 2 postnormalizing shifts)
1	0	1	0	N > 0 (> 2 postnormalizing shifts) FS=0, FN=0, and no underflow	N > 0 (> 2 postnormalizing shifts) FS=1, FN=0, and no underflow with FZ=1
1	1	0	0	underflow with FZ=0 and no trap by FS=1①	*
1	1	0	1	*	underflow, N < 0 (FZ=1)
1	1	1	0	*	underflow, N > 0 (FZ=1)

① result set to true zero

② "*" indicates impossible configurations

③ applies to add and subtract only where FN=0

CONTROL FLAGS (in PSW):

FZ ("floating zero"):
when = 1 causes trap to X'44' if underflow
when = 0 causes ZEROS to be stored (except where significance trap occurs)

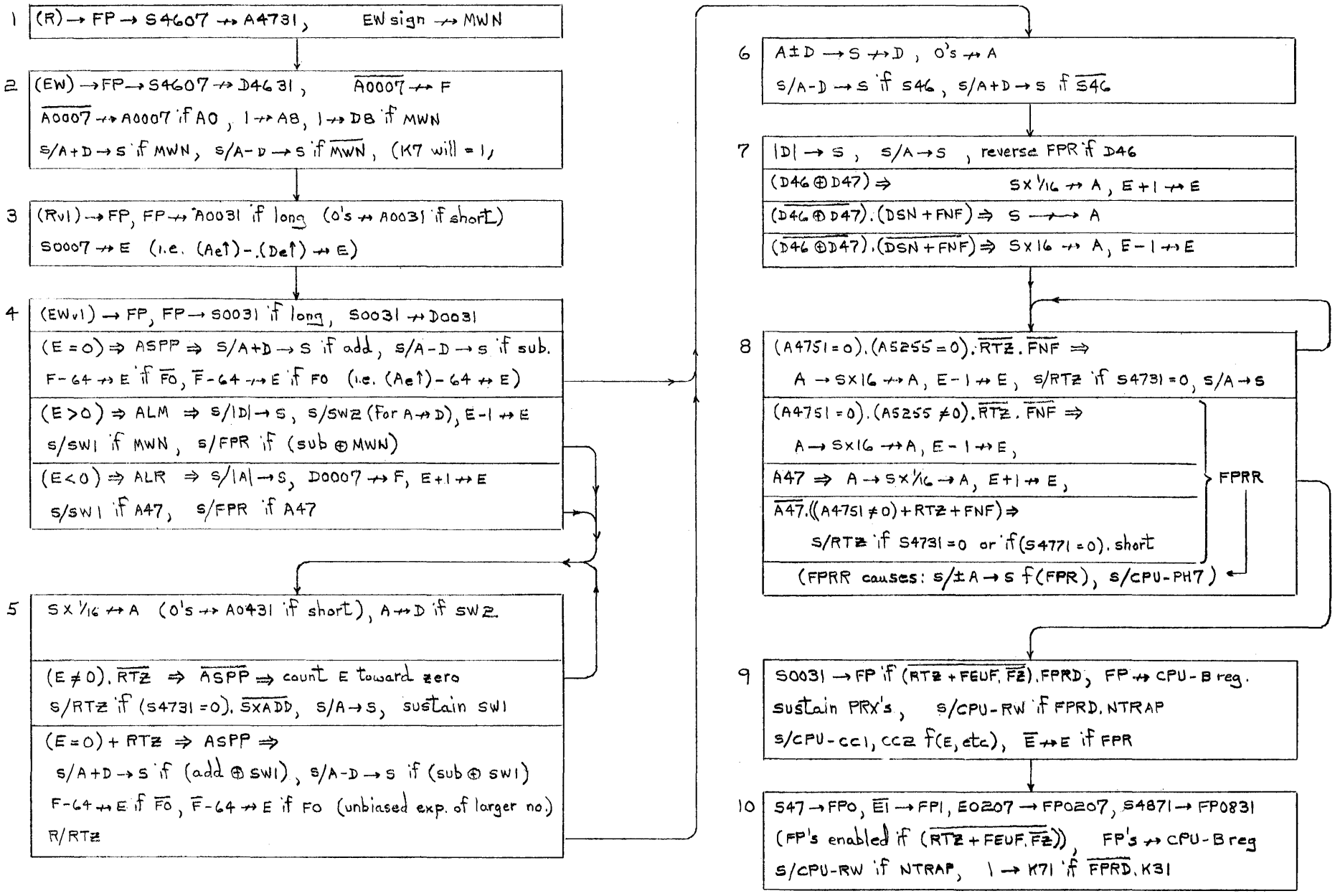
ADD/SUB : General sequence of events:

1) Put augend (R) in A
2) Put addend (EW) in D
3) Put (augend exponent) − (addend exponent) in E
4) Hexadecimally right shift the mantissa with the smaller exponent by the difference in E
5) Put the larger of the two exponents in E
6) Add/Subtract, result goes to D
7) Scale |result| until normalized (unless = 0, or left shifting inhibited by FN = 1). $0 \leq |result| < 1$ is put in A
8) Assemble adjusted exponent (E) and mantissa (A) on S in proper polarity
9) Store S if not TRAP

ADD/SUB control flags:

FN (called FNF in hardware):
when = 1 inhibits normalization of results
when = 0 allows normalization of results and setting of cc1 if top 3 hexes of |unnormalized result| = 0

FS (significance trap):
when = 1 causes trap to X'44' if FN = 0 and top 3 hexes of |unnormalized result| = 0


1
(R) → FP → S4607 → A4731, EW sign → MWN
2
(EW) → FP → S4607 → D4631, A0007 → F
A0007 → A0007 if A0, 1 → A8, 1 → D8 if MWN
S/A+D → S if MWN, S/A-D → S if MWN, (K7 will = 1)
3
(R v1) → FP, FP → A0031 if long (0's → A0031 if short)
S0007 → E (i.e. (Ae↑)-(De↑) → E)
4
(EW v1) → FP, FP → S0031 if long, S0031 → D0031
(E=0) ⇒ ASPP ⇒ S/A+D → S if add, S/A-D → S if sub.
F-64 → E if F0, F-64 → E if F0 (i.e. (Ae↑)-64 → E)
(E>0) ⇒ ALM ⇒ S/|D| → S, S/SW2 (For A → D), E-1 → E
S/SW1 if MWN, S/FPR if (sub ⊕ MWN)
(E<0) ⇒ ALR ⇒ S/|A| → S, D0007 → F, E+1 → E
S/SW1 if A47, S/FPR if A47
5
S x 1/16 → A (0's → A0431 if short), A → D if SW2
(E≠0).RTZ ⇒ ASPP ⇒ count E toward zero
S/RTZ if (S4731=0).SXADD, S/A → S, sustain SW1
(E=0)+RTZ ⇒ ASPP ⇒
S/A+D → S if (add ⊕ SW1), S/A-D → S if (sub ⊕ SW1)
F-64 → E if F0, F-64 → E if F0 (unbiased exp. of larger no.)
R/RTZ
6
A±D → S → D, 0's → A
S/A-D → S if S46, S/A+D → S if S46
7
|D| → S, S/A → S, reverse FPR if D46
(D46 ⊕ D47) ⇒ Sx1/16 → A, E+1 → E
(D46 ⊕ D47).(DSN+FNF) ⇒ S → A
(D46 ⊕ D47).(DSN+FNF) ⇒ Sx16 → A, E-1 → E
8
(A4751=0).(A5255=0).RTZ.FNF ⇒
A → Sx16 → A, E-1 → E, S/RTZ if S4731=0, S/A → S
(A4751=0).(A5255≠0).RTZ.FNF ⇒
A → Sx16 → A, E-1 → E,
A47 ⇒ A → Sx1/16 → A, E+1 → E,
A47.((A4751≠0)+RTZ+FNF) ⇒
S/RTZ if S4731=0 or if (S4771=0).short
FPRR
(FPRR causes: S/±A → S f(FPR), S/CPU-PH7)
9
S0031 → FP if (RTZ+FEUF.FZ).FPRD, FP → CPU-B reg.
sustain PRX's, S/CPU-RW if FPRD.NTRAP
S/CPU-cc1, cc2 f(E, etc), E → E if FPR
10
S47 → FP0, E1 → FP1, E0207 → FP0207, S4871 → FP0831
(FP's enabled if (RTZ+FEUF.FZ)), FP's → CPU-B reg
S/CPU-RW if NTRAP, 1 → K71 if FPRD.K31

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE3	(will not be reached if NFOPTION and	PRE2 was entered)	
	CLOCK TIMING: (T8L unless I/O service call, in which case FPCLEN/2 rejects		T5L clocks to box)
	S/T8L	S/T8L = PRE/12	(initial turn-on (redun.))
		+ FAFL.NIOACT.NPH10	(hold until IOACT or PH10)
		+ FAFL.(S/PH6/IO)	(return from I/O)
	RR → A	(S/AXRR = PRE/12)	(augend - m.s.w.)
	MB → C → D (from even location)	DXC = PREOPER.PRE3	(addend - m.s.w.)
	(address forced even	by S31INH in earlier phase (see "PREPARATION"))	
	1 → P31 } if long (request	PUC31 = (FAFL.N02.PRE3.NANL2)	(P31 = 0)
	S/MRQ } l.s.w. of addend)	S/MRQ/1 = (")	(inhibited if NFOPTION)
	S/NA → S	S/SXNA = FAFL.PRE/34	
	S/PH1 if FOPTION	S/PH1 = PRE/34.NBR	
	(TRAP if NFOPTION)		
	FPCON → box	FPCON = FAFL.PRE3	} start the box
	S/PH1 (in box)	S/PH1 = FPCON.NPH1	on next clock)
		↑ (prevent S/PH1	
PH1	NA → S	(preset in PRE3)	
PH1	NS → FP	FP_n = I.S_n.FPXS ← = NPH8.NDIS	(R) → FP
	FP0 → S47 (→ S46)	} SXFP/U = S4607XFP = PH1.NFPDIS	(augend mantissa - m.s.w.)
	FP0831 → S4871		
	FP0007 → S0007	SXFP/4 = S4607XFP = PH1.NFPDIS	(augend exponent)
	0's → S0831	(no enable hi)	
	S → A	AXS = PH1	
	DO → FPCON	FPCON = FAFL.PH1.DO	
	FPCON → MWN	S/MWN = FPCON.PH1; R/MWN = PH1	(store addend sign)
	0's → B (for PH7, 8)	BX = PH1	(post-normalization ctr.)
	0's → E (for PH3)	EX = PH1	(exponent reg.)
	0's → F (for PH2)	FX = PH1	(augend exponent reg.)
	S/LR31 } if long	S/LR31 = (FAFL.N02.PH1)	} (for augend-l.s.w.)
	S/AXRR }	S/AXRR = (")	
	S/ND → S	S/SXND = FAFL.PH1	
	S/PH2	S/PH2 = PH1.NBR	
	S/PH2	S/PH2 = PH1	
PH2	ND → S	(preset in PH1)	
PH2	NS → FP	FP_n = I.S_n.FPXS ← = NPH8.NDIS	(EW) → FP
	FP0 → S47 → S46	} SXFP/U = S4607XFP = PH2.NFPDIS	(addend mantissa - m.s.w.)
	FP0831 → S4871		
	FP0007 → S0007	SXFP/4 = S4607XFP = PH2.NFPDIS	(addend exponent)
	0's → S0831	(no enable hi)	
	S → D	DXS = PH2	
	NA0007 → F	FXNA = PH2.N06	(store augend exponent)

FAS (3D), FAL (1D), FSS (3C), FSL (1C)

MNEMONIC: FP: ADD/SUB

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 PH2 cont'd.	PRESET FOR EXPONENT ARITH:		
	NA0007 → A0007 if A0 = 1	S/A1 = (PH2.A0).NA1, (similarly A2 - A8)	
	1 → A8	AX/L = AXL' = (PH2.A0)	(un-invert augend exponent)
	1 → D8 } if addend is neg.	S/A8 = PH2.NMUL	(K7 control)
	S/A + D → S } if addend is pos.	S/D8 = PH2.MWN	(for (Ae↑) + (De↓)+1 → S)
	S/A - D → S	S/SXAPD = S/SXAPD/1 = PH2.MWN.NMUL	(for (Ae↑) + N(De↑)+1 → S)
		S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = PH2	
	RRv1 → A	(preset in PH1)	K7 = G8 = 1
	S/DRQ (for EWv1) } if long	S/DRQ/1 = FAFL.N02.PH2	(augend - l.s.w.)
	S/NA → S	S/SXNA = FAFL.PH2	
	S/PH3	S/PH3 = PH2.NBR	
	S/PH3	S/PH3 = PH2	
PH3 PH3	(Ae↑) - (De↑) → S	(preset in PH2)	
	S ↔ E	S/En = Sn.PH3	(unbiased exponenent difference ↔ E)
	NA → S	(preset in PH2)	
	NS → FP	FPn = I.Sn.FPXS ← = NPH8.NDIS	((Rv1) → FP if long, (R) → FP if short-insig.)
	FP ↔ A0031 if long	AXFP = PH3.N02	
	0's ↔ A0031 if short	AX/L = AXL = PH3	(augend - l.s.w.)
	MBv1 ↔ C ↔ D if long	DXC = FAFL.N02.PH3	(addend - l.s.w.)
	S/ND → S	S/SXND = FAFL.PH3	
	S/PH4	S/PH4 = PH3.NBR	
	S/PH4	S/PH4 = PH3	
PH4 PH4	ND → S	(preset in PH3)	
	NS → FP	FPn = I.Sn.FPXS ← = NPH8.NDIS	((EWv1) → FP if long, (EW) → FP if short-insig.)
	FP → S0031 if long	SXFP/4 = SXFP/A = S0031XFP = PH4.N02.NFPDIS	
	0's → S0031 if short	(no enable hi)	
	S ↔ D0031	DX/L = DXS/L = PH4	(addend l.s.w)
	Clear Condition Codes	R/CC = (FAFL.PH4)	
	S/B → S	S/SXB = (")	
	S/PH5	S/PH5 = PH4.NBR	
	→ see separate chart: "FAFL PH5-6"		
	(PH4, PH4 continued on next page)		

Sheet 2 of 6

MNEMONIC: FP: ADD/SUB

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 PH4 cont'd.	EXISTING CONDITIONS: A contains (R), (Rv1)/0's B " 0's D " (EW), (De) in D0007 E " (Ae↑) − (De↑) F " N(Ae)	(mantissa of augend) ((EWv1)/0's is being clocked into D0031 this phase)	
	IF E = 0 : ASPP ⇒ S/A+D → S if add S/A−D → S if sub F−64 ↛ E if NF0 NF−64 ↛ E if F0 S/PH6 (add/sub phase)	ASPP = PH4.N06.E0003Ƶ.E0407Ƶ S/SXAPD = S/SXAPD/1 = ASPP.07.NSW1 ← (R/SW1 = NPH9) S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = ASPP EXFM64 = ASPP.NF0 EXNFM64 = ASPP.F0 S/PH6 = ASPP	} (uninverted, unbiased A exponent ↛ E)
	IF E>0 : ALM ⇒ S/|D| → S S/SW1 if MWN S/FPR if (sub⊕MWN) E−1 ↛ E S/A ↛ D S/PH5 (alignment phase)	ALM = PH4.N06.NE0.NASPP S/SXD = ND46.S/SXAVD ← PH4.ALM S/SXMD = D46.S/SXAVD ← S/SW1 = (S/SW1/1) = ALM.MWN S/FPR = ALM.N(07⊕MWN) EDC7 = ALM S/SW2 = ALM.PH4.NPH7 ← (mech. conv.) S/PH5 = PH4.N06.NASPP	(R) > (EW) (reversing sign of D oper.) (result of PH6 will be in reverse polarity) (down-count toward zero)
	IF E<0 : ALR ⇒ S/|A| → S S/SW1 if A47 S/FPR if A47 E+1 ↛ E D0007 ↛ F S/PH5 (alignment phase)	ALR = PH4.N06.E0.NRTƵ ← (R/RTƵ = PH1) S/SXA = NA47.S/SXAVA ← S/SXMA = A47.S/SXAVA ← = PH4.ALR S/SW1 = ALR.A47 S/FPR = ALR.A47 EUC7 = ALR FXD = ALR.PH4 S/PH5 = PH4.N06.NASPP	(EW) > (R) (reversing sign of A oper.) (result of PH6 will be in reverse polarity) (up-count E toward zero) (larger exponent ↛ F)
PH5 or PH6 PH5	(ALIGNMENT PHASE − entered only if E ≠ 0 in PH4) FIRST CLOCK ONLY: A ↛ D } if E>0 in PH4 |D| → S } |A| → S " E<0 " " (inhibit S/RTƵ if number → S	DXA = PH5.SW2 ← (repeater) } (preset in PH4) requires negation hence involves carry	(larger operand ↛ D) propagation)
	ALL CLOCKS: SX 1/16 ↛ A 0's ↛ A0407 if short ↩ "GUARD DIGIT"	AXSR4 = AXSR4/1 = PH5.N06 S/A4 = S0.AXSR4-A.N02 ← (long only)	- (similary A5, A6, A7) logic permits right alignment into A0003

MNEMONIC: FP: ADD/SUB

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 or PH6 PH5 cont'd.	ALL BUT FINAL CLOCK ((E≠0).NRTZ): E+1 → E if E<0 E−1 → E if E>0 sustain SW1 S/A → S (for Sx1/16 → A) S/RTZ if 4731 = 0 sustain PH5 ON FINAL CLOCK ((E=0)+RTZ): S/A+D → S if (add ⊕ SW1) S/A−D → S if (sub ⊕ SW1) F−64 → E if NF0 NF−64 → E if F0 R/RTZ S/PH6	ALM = PH5.N06.NE0.NASPP ALR = PH5.N06.E0.NRTZ NASPP = N(PH5.N06.(E0003Z.E0407Z +RTZ)) EUC7 = ALR EDC7 = ALM S/SW1/1 = ALM.MWN + ALR.SW1 S/SxA = PH5.N06.NASPP S/RTZ = PH5.SZU.SZL.NASPP.NSXADD ← NPRXNAND.NGXADD S/PH5 = PH5.N06.NASPP ASPP = PH5.N06.(E0003Z.E0407Z +RTZ) S/SXAPD = S/SXAPD/1 = ASPP.(N07.SW1 + 07.NSW1) S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = ASPP EXFM64 = ASPP.NF0 EXNFM64 = ASPP.F0 R/RTZ = ASPP S/PH6 = ASPP	((E>0).NRTZ) ((E<0).NRTZ) N((E=0)+RTZ) (R/SW1 = NPH9) (uninverted, unbiased exponent of larger number.)
PH5 or PH6 PH6	(ADD/SUB PHASE) A ± D → S S → D S/A+D → S if NS46 S/A−D → S if S46 } S/|D| → S 0's → A (for 0±D → S) S/PH7	(preset in PH4 or PH5 by ASPP) DXS = PH6.N06 S/SXAPD reduces to PH6.N06.N(PR46 ⊕ K46) S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = PH6.N06 AX = PH6.N06 S/PH7 = PH6.N06	} (1-bit extension on left handles overflows) (special mech.)
PH6 PH7	(FIRST ADJUST |SUM| PHASE) |D| → S reverse FPR if D46 (D46 ≠ D47) ⇒ Sx1/16 → A E+1 → E (D46 = D47).N(DSN+FNF) ⇒ Sx16 → A E−1 → E Bx2 → B, 1 → B67 (D46 = D47).(DSN + FNF) ⇒ S → A S/A → S S/PH8	(preset in PH6) S/FPR = (PH7.N06.D46).NFPR R/FPR = (") AXSR4/1 = PH7.N06.(D46 ⊕ D47) AXSR4 = AXSR4/1 EUC7 = AXSR4/1.NPH5 AXSL4/1 = PH7.N06.NAXSR4/1.NDSN.N(FNF.N06) AXSL4 = AXSL4/1 EDC7 = AXSL4/1.N(PH5.DIV) BxBL1 = AXSL4/1.N06, S/B67 = BxBL1.N06 AXS = PH7.N06.NAXSR4/1.NAXSL4/1 S/SXA = PH7.N(S/PH7) + AXSL4/1.NFPRR S/PH8 = PH7.N(S/PH7)	} (FPR could have been set in PH4) (postnorm ctr.)
PH6 PH8	(FINAL ADJUST |SUM| PHASE) (A contains the |SUM| in the range CONTROL SIGNALS: ASN (A "Simple-Normalized") FPRR (Floating Point Result Ready)	0 ≤ S ≤ 1) (inhibit postnorm.) → high if A4751 ≠ all zeros (i.e. 1/16 ≤ S ≤ 1) or if FNF = 1 ASN = N(A47.A48.A49.A50.A51).N(NA47.A48S1Z.N(FNF.N06)) FPRR = PH8.NDIV.ASN (1/16 ≤ S ≤ 1 or FNF = 1) + PH8.NDIV.NA5255Z (ASN assured next clock) + PH8.RTZ.NPH5 (S = 0 (2nd PH8 clock))	

Sheet 4 of 6

MNEMONIC: FP: ADD/SUB

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH6 PH8 cont'd.	RTZ (Result is zero)	S/RTZ = PH8.SZU.SZL.NASPP.NSXADD → + PH8.SZU.02.N06 (needed for FNF=1 case where significance in guard digit only)	(S4731 = 0) (S4771 = 0) if short
	NASN ⇒ ((0≤A<1/16).(FNF=0)) A → S Sx16 → A E-1 → E Bx2 → B, 1 → B67 S/RTZ if S=0 S/A → S sustain PH8 } if (A5255=0).NRTZ ← raise FPRR if (A5255≠0)+RTZ	AXSL4/1 = PH8.NDIV.NASN (preset in PH7 or PH8) AXSL4 = AXSL4/1 EDC7 = AXSL4/1.N(PH5.DIV) BXBL1 = AXSL4/1.N06, S/B67 = BX (see above) S/SXA = AXSL4/1.NFPRR S/PH8 = PH8.NFPRR FPRR = PH8.NA5255Z.NDIV + PH8	BL1.N06 (postnorm. ctr.) .RTZ.NPH5
	ASN.A47 ⇒ (A=1) A → S S x 1/16 → A E+1 → E ← raise FPRR	(preset in PH7) (NOTE: S46 forced =0, otherwise = A47) AXSR4 = AXSR4/1 ← = PH8.NDIV.A47 EUC7 = NPH5.AXSR4/1 ← FPRR = PH8.NDIV.ASN	
	ASN.NA47 ⇒ ((1/16≤A<1)+(FNF=1) A → S S/RTZ if S=0 (FNF=1 case) ← raise FPRR	(preset in PH7) (see above) FPRR = PH8.NDIV.ASN	(don't shift registers)
	→ FPRR ⇒ (Floating Point Result Ready) stop CLOCK → box if I/O service call has CPU : FPCLEN/1 = NFPRR + NIOEN.NICIN S/PH7 S/MRQ (for next instruction) S/A → S if NFPR S/-A → S if FPR S/PH9	S/PH7 = PH6.NBR ← = NBRPH6 = N(FAFL.PH6.NFPRR) S/MRQ/1 = FAFL.PH6.NBRPH6.NIOEN S/SXA = FPRR.NFPR S/SXMA = FPRR.FPR S/PH9 = FPRR	} (put result in proper polarity)
PH7 PH9	CONTROL SIGNALS USED IN PH9-10: FPRD TRAP FEOF FEUF	FPRD = N02 TRAP = FEOF + FEUF.N(FEUF.NFZ) = FEUF.FZ + B65.N06.FS FEOF = NE0.E1.NRTZ FEUF = E0.NE1.NRTZ.N(B65.N06.FS.NFZ)	(result double length) (exp. overflow) ((exp. underflow).FZ) (significance trap) (exp. overflow) (exp. underflow)
	A or -A → S S0031 → FP if not Z 0's → FP if (short FP → B S/B → S S/LR31 S/RW if long and not trap case	(preset in PH8 by FPRR) FPXSL = PH9.NFPDIS.FPRD.NRTZ.N(FEUF.NFZ) or result zero or underflow and FZ=0) BXFP = FAFL.PH7 S/BXS = FAFL.PH7 S/LR31 = FAFL.PH7 S/RW = S/RW/FP = PH9.FPRD.NTRAP	} (result - l.s.w.)
	S/CC1 if exponent underflow or S/CC2 if exponent underflow or	if FN=0 and >2 post-normalizing shifts required (includes RTZ). S/CC1 = S/CC1/FP = PH9.FEUF + PH9.N06.B65 overflow. S/CC2 = S/CC2/FP = PH9.FEUF + PH9.FEOF	

Sheet 5 of 6

MNEMONIC: FP: ADD/SUB

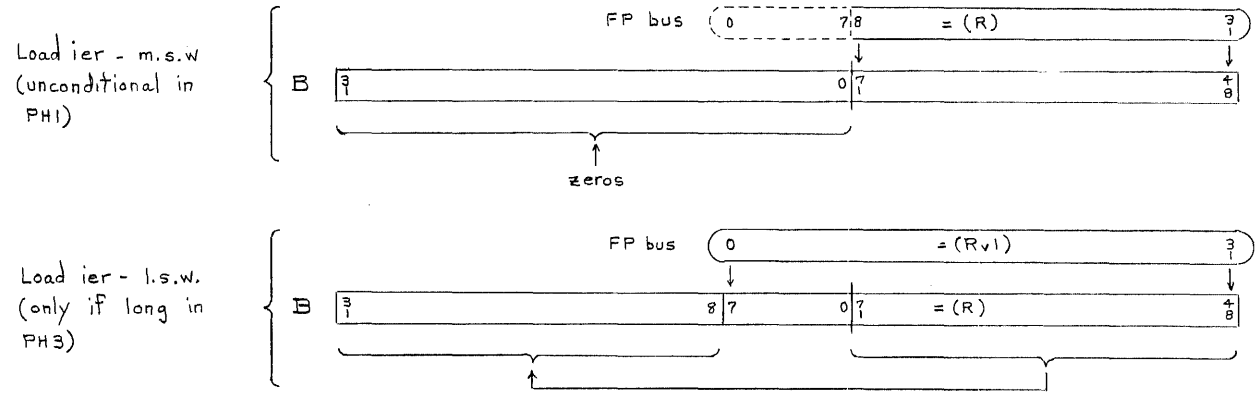
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7 PH9 cont'd.	S/A → S if NFPR S/-A → S if FPR	S/SXA = PH9.NFPR S/SXMA = PH9.FPR	} (sustaining from PH8)
	NE ↛ E if FPR. (E0 = E1) ↳ qualifier	EXNE = PH9. FPR.NTRAP.N(FEUF, NFZ) prevents interchanging FEUF and FEOF	(invert exp. when result neg.)
	S/PH8 S/PH10	S/PH8 = PH7.NBR S/PH10 = PH9	
PH8 PH10	B → S S ↛ RW, 1 if long and not trap S ≠ 0 ⇒ 1 ↛ SW0 (for TESTS)	} (preset in PH7) S/SW0 = N50031Z. (S/SW0/NZ) ← = FAFL.N02.PH8	(result - l.s.w.)
	A or -A → S S47 → FP0 (exp.) { NE1 → FP1 (+64 bias) E0207 → FP0207 S4871 → FP0831 FP ↛ B	(preset in PH9) ← { note: K71 is forced high in short "-A" case to assure proper truncation: K71 = G0003 = PH10.NFPRD.K31 FPSXU = PH10.NFPDIS.NRTZ.N(FEUF, NFZ) (FPXS = NPH8.NDIS -- blocks S (=B) → FP) BXFP = FAFL.PH8	} (result m.s.w)
	S/B → S S/RW if not trap case	S/SXB = FAFL.PH8 S/RW = S/RW/FP = PH10.NTRAP	
	S/DRQ S/PH10 R/PH10 (no more box action)	S/DRQ = BRPH10 = FAFL.PH8 S/PH10 = BRPH10 = FAFL.PH8 (no set term hi)	
PH10	B → S S ↛ RW	} (preset in PH8)	(result - m.s.w.)
	TESTS (cc3, cc4 control) (note: SW0 = 1 causes S > 0 for (ADD/SUB).(FN=1) case only, where result is +, exp. = -64, and significance in l.s.w. only.)	TESTS = FAFL.ENDE indication for double length zero test	; This is needed for
	Stop sustaining TBL	S/TBL = FAFL.NIOACT.NPH10	
	TRAP to X'44' if RW is off	{ S/TRAP = (FAFL.ENDE.NRW) S/TR29 = (")	} (NRW means TRAP in the box was hi last cl.)
	ENDE	ENDE = PH10.EXC	

Sheet 6 of 6

MNEMONIC: FP: ADD/SUB

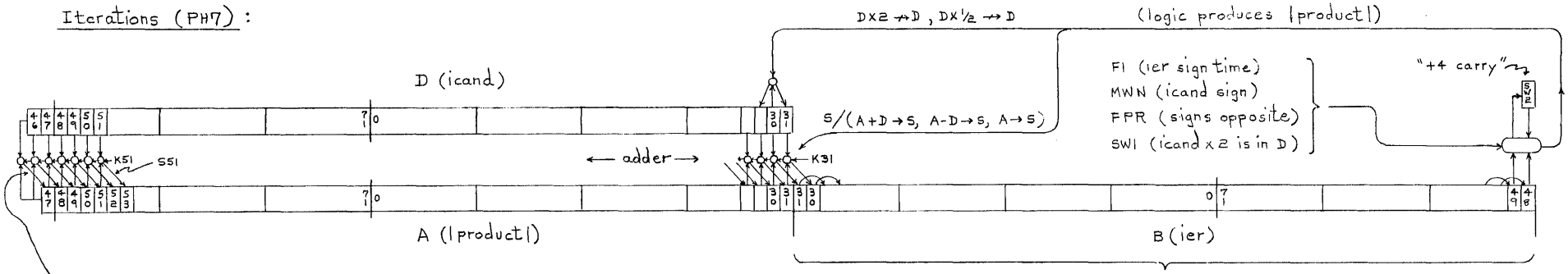
General sequence of events:

1) Put multiplier (R) in A, and backwards in B
2) Put multiplicand (EW) in D
3) Put exponent sum in E
4) Normalize (D) if necessary; early exit if = 0
5) " (A) " " ; " " " = 0
   (B doesn't change but 2 2-bit iterations are deleted for each hex of A shift required)
6) Clear A
7) Iterate, producing $1/256 \leq |prod.| < 1$ in AB
8) Normalize AB
9) Assemble adjusted exponent (E) and mantissa (A) on S in proper polarity
10) Store S if not TRAP



Loading the multiplier into B - backwards

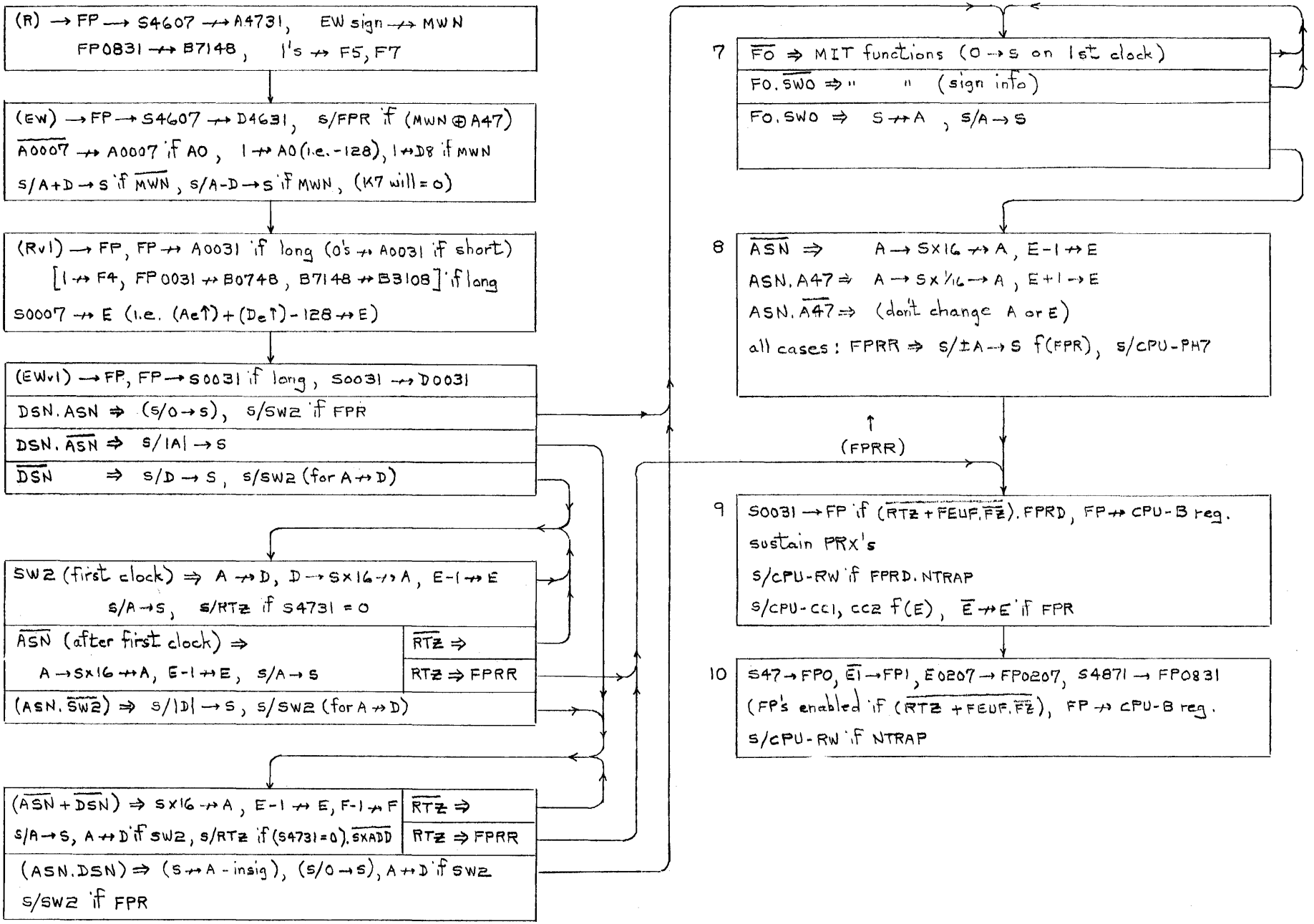
Iterations (PH7):



S45 = PR46.K45, which red. to PR46.$\overline{K46}$ + G46

Note that the B register is "played backwards" (This enabled efficient use of the FT41 flip-flop module)

MUL


1. (R) → FP → S4607 ↛ A4731, EW sign ↛ MWN
FP0831 ↛ B7148, 1's ↛ F5, F7
2 (EW) → FP → S4607 ↛ D4631, S/FPR if (MWN ⊕ A47)
A0007 ↛ A0007 if A0, 1 ↛ A0 (i.e. -128), 1 ↛ D8 if MWN
S/A+D → S if MWN, S/A-D → S if MWN, (K7 will = 0)
3 (R v 1) → FP, FP ↛ A0031 if long (0's ↛ A0031 if short)
[1 ↛ F4, FP 0031 ↛ B0748, B7148 ↛ B3108] if long
S0007 ↛ E (i.e. (Ae↑) + (De↑) - 128 ↛ E)
4 (EW v 1) → FP, FP → S0031 if long, S0031 ↛ D0031
DSN. ASN ⇒ (S/0 → S), S/SW2 if FPR
DSN. ASN ⇒ S/|A| → S
DSN ⇒ S/D → S, S/SW2 (for A ↛ D)
5 SW2 (first clock) ⇒ A ↛ D, D → Sx16 ↛ A, E-1 ↛ E
S/A → S, S/RTZ if S4731 = 0
ASN (after first clock) ⇒
A → Sx16 ↛ A, E-1 ↛ E, S/A → S
RTZ ⇒
RTZ ⇒ FPRR
(ASN. SW2) ⇒ S/|D| → S, S/SW2 (for A ↛ D)
6 (ASN + DSN) ⇒ Sx16 ↛ A, E-1 ↛ E, F-1 ↛ F
S/A → S, A ↛ D if SW2, S/RTZ if (S4731 = 0). SXADD
RTZ ⇒
RTZ ⇒ FPRR
(ASN. DSN) ⇒ (S ↛ A - insig), (S/0 → S), A ↛ D if SW2
S/SW2 if FPR
7 FO ⇒ MIT functions (0 → S on 1st clock)
FO. SWO ⇒ " " (sign info)
FO. SWO ⇒ S ↛ A, S/A → S
8 ASN ⇒ A → Sx16 ↛ A, E-1 ↛ E
ASN. A47 ⇒ A → Sx 1/16 → A, E+1 → E
ASN. A47 ⇒ (don't change A or E)
all cases: FPRR ⇒ S/±A → S f(FPR), S/CPU-PH7
(FPRR)
9 S0031 → FP if (RTZ + FEUF. FZ). FPRD, FP ↛ CPU-B reg.
sustain PRX's
S/CPU-RW if FPRD. NTRAP
S/CPU-cc1, cc2 f(E), E ↛ E if FPR
10 S47 → FP0, E1 → FP1, E0207 → FP0207, S4871 → FP0831
(FP's enabled if (RTZ + FEUF. FZ), FP ↛ CPU-B reg.
S/CPU-RW if NTRAP
311
MUL
MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE3	(will not be reached if NFOPTION and PRE2 was entered)		
	CLOCK TIMING: (T8L unless I/O service call, in which case FPCLEN/2 rejects TSL clocks to box)		
	S/T8L	S/T8L = PRE/12 + FAFL.NIOACT.NPH10 + FAFL.(S/PH6/IO)	(initial turn-on (redun.)) (hold until IOACT or PH10) (return from I/O)
	RR → A	(S/AXRR = PRE/12)	(ier - m.s.w.)
	MB → C → D (from even location) (address forced even)	DXC = PREOPER.PRE3	(icand - m.s.w.) (by S31INH in earlier phase (see "PREPARATION"))
	1 → P31 S/MRQ } if long (request l.s.w. of icand)	PUC31 = (FAFL.N02.PRE3.NANL2) S/MRQ/1 = (")	(P31 = 0) (inhibited if NFOPTION)
	S/NA → S	S/SXNA = FAFL.PRE/34	
	S/PH1 if FOPTION (TRAP if NFOPTION)	S/PH1 = PRE/34.NBR	
	FPCON → box S/PH1 (in box)	FPCON = FAFL.PRE3 S/PH1 = FPCON.NPH1 ↳ (prevents S/PH1 on next clock)	} start the box
PH1 PH1	NA → S NS → FP	(preset in PRE3) $FP_n = I.S_n.FPXS \leftarrow = NPH8.NDIS$	(R) → FP
	FP0 → S47 (→S46) FP0831 → S4871	SXFP/U = S4607xFP = PH1.NFPDIS	(ier mantissa - m.s.w.)
	FP0007 → S0007 0's → S0831	SXFP/4 = S4607xFP = PH1.NFPDIS (no enable hi)	(ier exponent)
	S → A	AXS = PH1	
	FP3108 → B4871 0's → B0031	BXFP/U = BXFPU = PH1.MUL Bx = PH1	(ier mantissa m.s.w. → B, backwards)
	0's → E (for PH3)	Ex = PH1	(exponent reg.)
	5 → F (enable F) 0 → SW0	S/F5 = S/F7 = BXFP/U = BXFPU = PH1.MUL Fx = PH1 R/SW0 = Bx = PH1	} (iterations ctr.)
	D0 → FPCON FPCON → MWN	FPCON = FAFL.PH1.D0 S/MWN = FPCON.PH1; R/MWN = PH1	(store icand sign)
	S/LR31 S/AXRR } if long S/ND → S	S/LR31/1 = (FAFL.N02.PH1) S/AXRR = (") S/SXND = FAFL.PH1	} (for ier l.s.w.)
	S/PH2 S/PH2	S/PH2 = PH1.NBR S/PH2 = PH1	
PH2 PH2	1 → FPR if operand signs ≠	S/FPR = PH2.06.(MWN ⊕ A47); R/FPR = PH1	

MNEMONIC: FP: MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 PH2 cont'd	ND → S NS → FP FP0 ⟶ S47 → S46 FP0831 ⟶ S4871 FP0007 ⟶ S0007 0's ⟶ S0831 S ⟶↠ D	(preset in PH1) FP_n = I. S_n.FPXS ← = NPH8.NDIS SXFP/U = S4607XFP = PH2.NFPDIS SXFP/4 = S4607XFP = PH2.NFPDIS (no enable hi) DXS = PH2	(EW) → FP (icand mantissa - m.s.w.) (icand exponent)
	PRESET FOR EXPONENT ARITH: NA0007 ↠ A0007 if A0=1 1 ↠ A0 S/A+D → S if icand is pos. S/A-D → S } if icand is neg. 1 ↠ D8	S/A1 = (PH2.A0).NA1, (similarly A2-A8) AX/L = AXL = (PH2.A0) S/A0 = PH2.MUL S/SXAPD = S/SXAPD/1 = PH2.NMWN.MUL S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = PH2 S/D8 = PH2.MWN	(un-invert ier exponent) (minus 128) ⟶ (for (Ae↑)+(De↑)+0-128→S) (for (Ae↑)+N(De↓)+0-128→S) K7=0 since PR8=G8=0 (ier - l.s.w.)
	RRv1 ↠ A S/DRQ (for EWv1) } if long S/NA → S	(preset in PH1) S/DRQ/1 = FAFL.NO2.PH2 S/SXNA = FAFL.PH2	
	S/PH3 S/PH3	S/PH3 = PH2.NBR S/PH3 = PH2	
PH3 PH3	(Ae↑)+(De↑)-128 → S S ↠ E	(preset in PH2) S/E_n = S_n.PH3	(unbiased exponent sum ↠ E)
	NA → S NS → FP FP ↠ A0031 if long 0' ↠ A0031 if short	(preset in PH2) FP_n = I. S_n.FPXS = NPH8.NDIS AXFP = PH3.NO2 AX/L = AXL = PH3	((Rv1) → FP if long, (R) → FP if short-insig.) (ier - l.s.w.)
	B4871 ↠ B0831 FP0700 ↠ B0007 FP3108 ↠ B4871 1 ↠ F4 } if long	BXFP/L BXFP/L BXFP/U S/F4 = BXFP/L } = BXFP = PH3.NO2.MUL	(double-length ier ↠ B, backwards) (change F from 5 to 13)
	MBv1 ↠ C ↠ D if long S/ND → S	DXC = FAFL.NO2.PH3 S/SXND = FAFL.PH3	(icand - l.s.w.)
	S/PH4 S/PH4	S/PH4 = PH3.NBR S/PH4 = PH3	
PH4 PH4	ND → S NS → FP FP → S0031 if long 0's → S0031 if short S ↠ D0031	(preset in PH3) FP_n = I. S_n.FPXS = NPH8.NDIS SXFP/4 = SXFP/A = S0031XFP = PH4.NO2.NFPDIS (no enable hi) DX/L = DXS/L = PH4	((EWv1) → FP if long, (EW) → FP if short-insig.) (icand - l.s.w.)
	Clear Condition Codes S/B → S	R/CC = FAFL.PH4 S/SXB = FAFL.PH4	
	S/PH5 ↳ see separate chart: "FAFL PH5-6"	S/PH5 = PH4.NBR	

Sheet 2 of 6

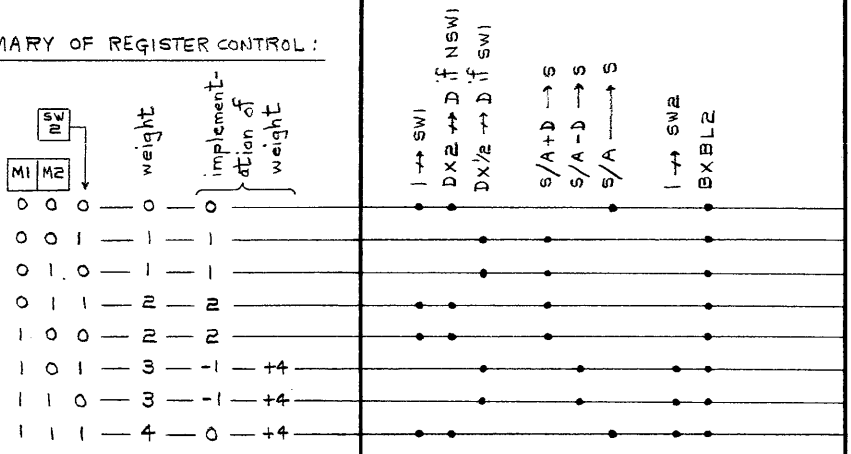
MNEMONIC: FP : MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 PH4	EXISTING CONDITIONS: A contains (R), (Rv1)/0's B7148 " (R), (Rv1) if long B3148 " (R) if short D " (EW), (De) in D0007 E " (Ae1) + (De1) - 128 F " 5 if short, 13 if long DSN.ASN ⇒ S/PH7 S/SW2 if +x- or -x+ DSN.NASN ⇒ S/|A| → S S/PH6 NDSN ⇒ S/D → S S/A ↛ D S/PH5	(mantissa of multiplier) (mantissa of multiplier (without sign bit), with least significant bit in B48. Reverse-loading of B is for mech. conv.) ((EWv1)/0's is being clocked into D0031 this phase.) (tentative product exponent (unbiased)) (number of hexes - minus one - in multiplier) S/PH7 = PH4.MUL.06.DSN.ASN S/SW2 = (S/PH7).NPH7.MUL.FPR S/SXA = NA47.SXAVA ← S/SXMA = A47.SXAVA ← = PH4.06.DSN.N(S/PH7) S/PH6 = PH4.06.DSN.N(S/PH7) S/SXD = PH4.06.NDSN S/SW2 = S/SW2/1 = PH4.06.NDSN S/PH5 = PH4.06.NDSN	(multiplier carry ff.)
PH5 or PH6 PH5	(entered only if multiplicand requires prenormalization) (SW2 + NASN) ⇒ N(S/PH6) ⇒ A ↛ D (first clock only) D → S (") A → S (after first clock) SX16 ↛ A E - 1 ↛ E S/RTZ if S=0 (on first clock) RTZ ⇒ raise FPRR s/PH9, etc. - (see FPRR functions at end of PH8) NRTZ ⇒ sustain PH5 (NSW2.ASN) ⇒ (multiplicand is "simple-normalized" - 2nd clock or later) S/|D| → S S/A ↛ D S/PH6	N(S/PH6) = N(PH5.06.ASN.NSW2) DXA = PH5.SW2 ← (repeater, set in PH4) (save multiplier) (preset in PH4) S/SXA = NFPRR.AXSL4/1 ← AXSL4 = AXSL4/1 ← = PH5.06.N(S/PH6) EDC7 = N(PH5.DIV).AXSL4/1 ← = PH5.06.N(S/PH6) S/RTZ = PH5.SZU.SZL.NASPP.NSXADD FPRR = PH5.06.RTZ S/PH5 = PH5.06.N(S/PH6).NRTZ S/SXD = ND46.SXAVD ← S/SXMD = D46.SXAVD ← S/SW2 = S/SW2/1 ← S/PH6 = ← = PH5.06.ASN.NSW2	(multiplicand = 0 ∴ product = 0; store zero result)
PH5 or PH6 PH6	(entered from PH5 or conditionally from PH4) ON THE FIRST CLOCK: |A| → S if entry from PH4 |D| → S " " " PH5 A ↛ D " " " PH5 (NASN + NDSN) ⇒ N(S/PH7) ⇒ S/A → S (for clocks after first) SX16 ↛ A E - 1 ↛ E F - 1 ↛ F (to delete 2 2-bit iterations per hex of normalization required)	(preset logic) DXA = PH6.SW2 (return "simple-normalized" multiplicand ↛ D) N(S/PH7) = N(PH6.06.ASN.DSN) (multiplier not "simple-norm'd.") S/SXA = NFPRR.AXSL4/1 ← AXSL4 = AXSL4/1 ← EDC7 = N(PH5.DIV).AXSL4/1 ← = PH6.06.N(S/PH7) FDC7 = PH6.MUL.N(S/PH7)	|multiplier| → S

Sheet 3 of 6

MNEMONIC: FP: MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 or PH6 PH6 cont'd	S/RTZ if S=0 (on first clock) RTZ ⇒ raise FPRR, s/PH9, etc. - (see FPRR functions at end of PH8) NRTZ ⇒ sustain PH6	S/RTZ = PH6.SZU.SZL.NASPP.NSXADD FPRR = PH6.RTZ.NPH5 S/PH6 = PH6.06.N(S/PH7).NRTZ	(multiplier = 0 ∴ product = 0; store zero result)
	(ASN.DSN) ⇒ (multiplicand is "simple normalized" and iterations count has been adjusted for un-normalized multiplier) S ↦ A (insig, needed for DIV.) S/PH7 1 ↦ SW2 if +x− or −x+	AXS = PH6.06.ASN.DSN S/PH7 = PH6.06.ASN.DSN S/SW2 = (S/PH7).NPH7.MUL.FPR	(multiplier carry f.f.)
PH5 or PH6 PH7	(ITERATIONS PHASE) (entered from PH4 or PH6) SUMMARY OF REGISTER CONTROL:		PH7 duration: short: 14 clocks * long: 30 clocks * *: subtract 2 clocks for each hex of ier prenormalization



CONTROL SIGNALS:

MIT	(high except final clock)	MIT = PH7.MUL.N(F0.SW0)	(high on final 2 clocks)
M1	(multiplier:	M1 = B49.(NF1.NFPR) + NB49.(NF1.FPR)	+ (MWN.F1)
M2	B if +x+ or −x−	M2 = B48.(") + NB48.(")	) + (")
SW2	NB+1 if +x− or −x+)	S/SW2 = MIT.MUL.M1.N(S/SXAPD/1)	(hi on first clock if FPR)
SW1	(on when D contains multiplicand x 2)	S/SW1 = MIT.N(MIT.(M2⊕SW2))	
SW0	(high on even numbered clocks)	S/SW0 = NSW0.BXBL2 ← = MIT	R/SW0 = BX

REGISTER CONTROL:

DX2 ↦ D (multiplicand x 2 ↦ D)	DXDL1 = MIT.N(MIT.(M2⊕SW2)).NSW1	
DX ½ ↦ D (" X1 ↦ D)	DXDR1 = (MIT.(M2⊕SW2)).SW1	
S/A+D ⟶ S (S=0 on first clock, hence A clears)	S/SXAPD = S/SXAPD/1 = MIT.N(S/SXAMD/1).N(S/SXA)	
S/A−D ⟶ S	S/SXAMD = S/SXAMD/1 = (MIT(M2⊕SW2)).M1	
S/A ⟶ S	S/SXA = MIT.(NM1.NM2.NSW2 + M1.M2.SW2)	
"S45" ↦ A47	S/A47 = S/A47/2 = (G46 + PR46.NK46).BXBL2 ← = MIT	
S4629 ↦ A4831	AXSR2 = MIT	
S3031 ↦ B3130	S/B31 = S30.BXBL2, S/B30 = S31.BXBL2	(B register is played backwards)
B3146 ⟶ B2948	BXBL2 = MIT (supplies multiplier bits)	

CONTROL FUNCTIONS - GENERAL:

F−1 ↦ F on even numbered clocks	FDC7 = MIT.SW0
sustain PH7 until final clock	S/PH7 = MIT

Sheet 4 of 6

MNEMONIC: FP: MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 or PH6 PH7 cont'd	ON THE FINAL CLOCK : S ↠ A (sign iteration) S/A → S S/PH8	((F = -1).(SW0 = 1)) AXS = PH7.N(S/PH7).MUL S/SXA = PH7.N(S/PH7) S/PH8 = PH7.N(S/PH7)	
PH6 PH8	(ADJUST |PRODUCT| PHASE) NASN ⇒ (1/256 ≤ A < 1/16) A → S SX16 ↠ A B3128 ↠ A2831 E - 1 ↠ E ASN.A47 ⇒ (A = 1) A → S S x 1/16 ↠ A E + 1 ↠ E ASN.NA47 ⇒ (1/16 ≤ A < 1) (nothing is changed)	(1/256 ≤ |product| ≤ 1) AXSL4/1 = PH8.NDIV.NASN (preset in PH7) AXSL4 = AXSL4/1 S/A28 = B31, A2831XB ← PH8.MUL.NASN. (similarly, A29, A30, A31) EDC7 = AXSL4/1.N(PH5.DIV) AXSR4/1 = PH8.NDIV.A47 (preset in PH7) AXSR4 = AXSR4/1 (note: S46 is inhibited by PH8 to prevent setting A50) EUC7 = AXSR4/1.NPH5	
	FPRR ⇒ (Floating Point Result Ready) raise FPRR { (Prod. ≠ 0) (ier = 0) (icand = 0) }	FPRR = PH8.NDIV.(ASN + NAS255Z) + PH5.06.RTZ + PH6.RTZ.NPH5	(all cases in PH8) (2nd clock of PH5) (2nd clock of PH6)
	stop CLOCK → box if I/O service S/PH7 S/MRQ (for next instruction)	call has CPU: FPCLEN/1 = NFPRR + S/PH7 = PH6.NBR ← = NBRPH6 = S/MRQ/1 = FAFL.PH6.NBRPH6.N	NIOEN.NIOIN N(FAFL.PH6.NFPRR) IOEN
	S/A → S if +x+ or -x- S/-A → S if +x- or -x+ S/PH9	S/SXA = FPRR.NFPR S/SXMA = FPRR.FPR S/PH9 = FPRR	} (put result in proper polarity)
PH7 PH9	CONTROL SIGNALS USED IN PH9-10: FPRD (result double length) TRAP FEOF FEUF A or -A → S S0031 → FP if not 0's → FP if (short and R is odd or result = 0 or underflow FP ↠ B S/B → S S/LR31 S/RW if (long or R is even). not trap s/cc1 if exponent underflow s/cc2 if exponent underflow or overflow	FPRD = N02 + MUL.NR31 TRAP = FEOF + FEUF.N(FEUF.NFZ) = FEUF.FZ FEOF = NEO.E1.NRTZ FEUF = E0.NE1.NRTZ.N(B65.N06.FS.NFZ); (preset in PH5, 6, or 8 by FPRR) FPXSL = PH9.NFPDIS.FPRD.NRTZ.N (FEUF.NFZ and FZ = 0) BXFP = FAFL.PH7 S/BXS = FAFL.PH7 S/LR31 = FAFL.PH7 S/RW = S/RW/FP = PH9.FPRD.NTRAP S/CC1 = S/CC1/FP = PH9.FEUF S/CC2 = S/CC2/FP = PH9.FEUF + PH9.FEOF	(long) (even R address) (exp. overflow) (exp. underflow).(FZ = 1) (exp. overflow) (exp. underflow) } (result-1, s, w)

Sheet 5 of 6

MNEMONIC: FP : MUL

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7 PH9 cont'd.	S/A → S if NFPR S/-A → S if FPR	S/SXA = PH9.NFPR S/SXMA = PH9.FPR	} (sustaining from PH8)
	NE → E if FPR. (E0 = E1) ↳ qualifier	EXNE = PH9.FPR.NTRAP.N(FEUF,NFZ) prevents interchanging FEUF and FEOF	(invert exp. when result neg.)
	S/PH8 S/PH10	S/PH8 = PH7.NBR S/PH10 = PH9	
PH8 PH10	B → S S → RW, 1 if long and not trap S ≠ 0 ⇒ 1 → SW0 (for TESTS)	} (preset in PH7) S/SW0 = NS0031Z.(S/SW0/NZ) ← = FAFL.N02.PH8	(result - l.s.w.)
	A or -A → S S47 → FP0 (exp.) { NE1 → FP1 (+64 bias) E0207 → FP0207 S4871 → FP0831 FP → B	(preset in PH9) ← { note: K71 is forced high in short "-A" case to assure proper truncation; K71 = G0003 = PH10.NFPRD.K31 FPSXU = PH10.NFPDIS.NRTZ.N(FEUF,NFZ) (FPXS = NPH8.NDIS -- blocks S(=B) → FP) BXFP = FAFL.PH8	} (result m.s.w)
	S/B → S S/RW if not trap case	S/SXB = FAFL.PH8 S/RW = S/RW/FP = PH10.NTRAP	
	S/DRQ S/PH10 R/PH10 (no more box action)	S/DRQ = BRPH10 = FAFL.PH8 S/PH10 = BRPH10 = FAFL.PH8 (no set term hi)	
PH10	B → S S → RW	} (preset in PH8)	(result - m.s.w.)
	TESTS (cc3, cc4 control) (note: SW0 = 1 causes S > 0 (ADD/SUB).(FN=1) case only, where result is +, exp. = -64, and significance in l.s.w. only.)	TESTS = FAFL.ENDE indication for double length zero test; This is needed for	
	Stop sustaining TBL	S/TBL = FAFL.NIOACT.NPH10	
	TRAP to X'44' if RW is off	{ S/TRAP = (FAFL.ENDE.NRW) S/TR29 = (")	} (NRW means TRAP in the box was hi last cl.)
	ENDE	ENDE = PH10.EXC	

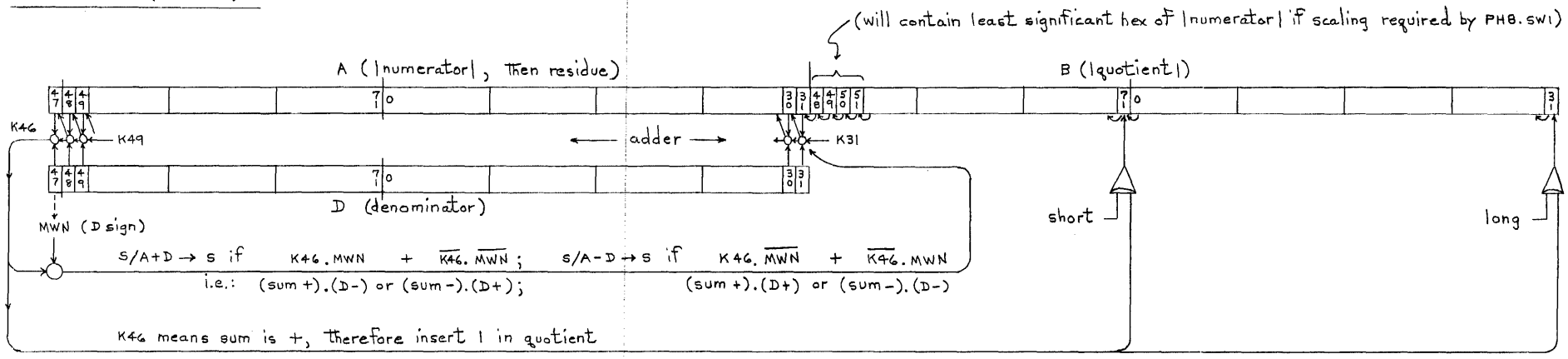
Sheet 6 of 6

MNEMONIC: FP: MUL

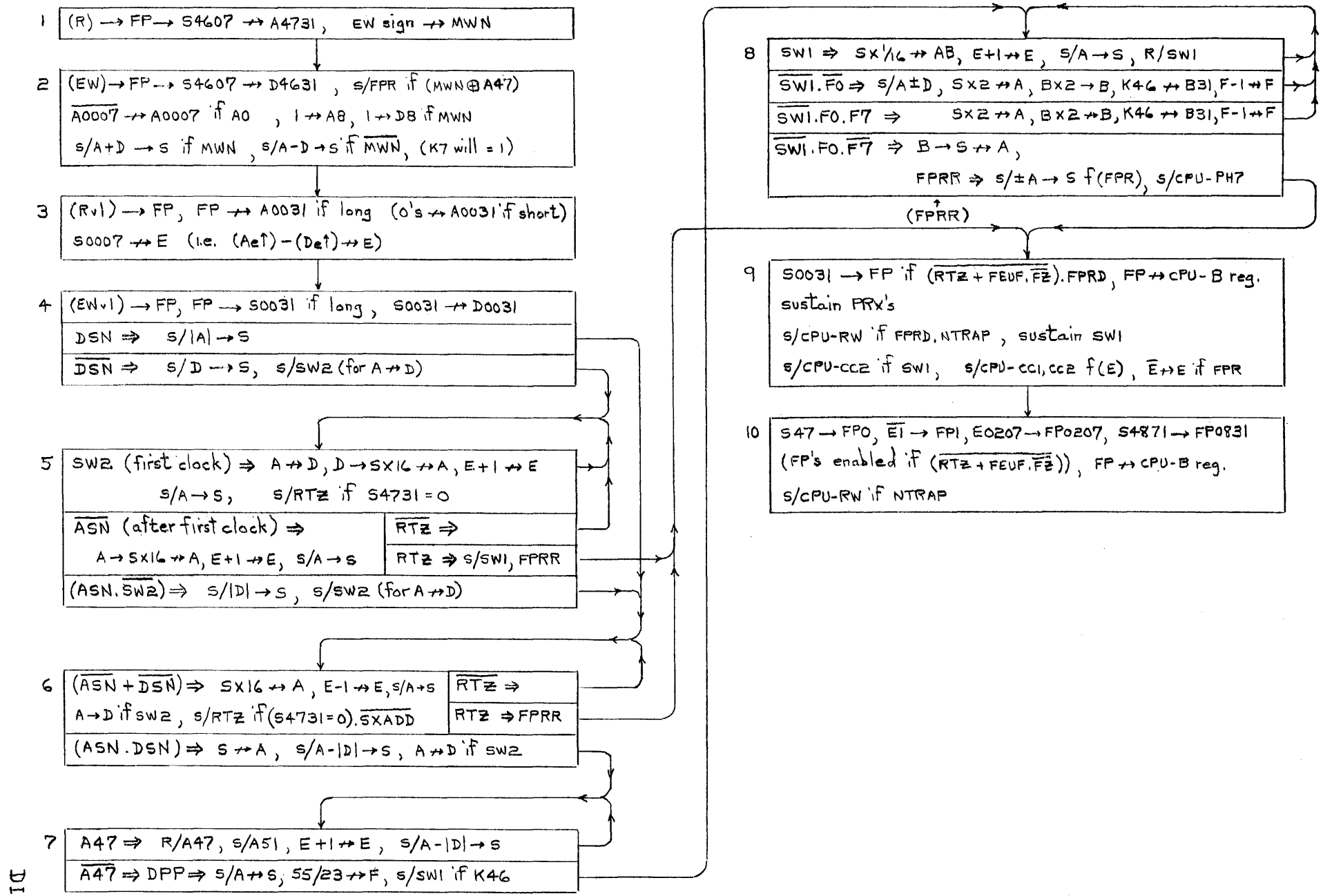
General sequence of events:

1) Put numerator (R) in A
2) Put denominator (EW) in D
3) Put (numerator exponent) - (denominator exponent) in E
4) Normalize (D) if necessary ; TRAP and set CC2 if = 0
5) " (A) " " ; early exit if = 0
6) Put |normalized numerator| in A
7) Hexadecimally right shift A until |A| < |D| (0-2 clocks)
8) Iterate, producing 1/16 ≤ |quotient| < 1 in B
9) Put |quotient| in A
10) Assemble adjusted exponent (E) and mantissa (A) on S in proper polarity
11) Store S if not TRAP

ITERATIONS (PH8. $\overline{SW1}$):



DIV


1
(R) → FP → S4607 ↦ A4731, EW sign ↦ MWN
2
(EW) → FP → S4607 ↦ D4631, S/FPR if (MWN⊕A47)
A0007 ↦ A0007 if A0, 1 ↦ A8, 1 ↦ D8 if MWN
S/A+D → S if MWN, S/A-D → S if MWN, (K7 will = 1)
3
(R v 1) → FP, FP ↦ A0031 if long (0's ↦ A0031 if short)
S0007 ↦ E (i.e. (Ae↑) - (De↑) ↦ E)
4
(EW v 1) → FP, FP → S0031 if long, S0031 ↦ D0031
DSN ⇒ S/|A| → S
DSN ⇒ S/D → S, S/SW2 (for A ↦ D)
5
SW2 (first clock) ⇒ A ↦ D, D → SX16 ↦ A, E+1 ↦ E
S/A → S, S/RTZ if S4731 = 0
ASN (after first clock) ⇒
A → SX16 ↦ A, E+1 ↦ E, S/A → S
RTZ ⇒
RTZ ⇒ S/SW1, FPRR
(ASN.SW2) ⇒ S/|D| → S, S/SW2 (for A ↦ D)
6
(ASN + DSN) ⇒ SX16 ↦ A, E-1 ↦ E, S/A → S
A → D if SW2, S/RTZ if (S4731 = 0).SXADD
RTZ ⇒
RTZ ⇒ FPRR
(ASN.DSN) ⇒ S ↦ A, S/A-|D| → S, A ↦ D if SW2
7
A47 ⇒ R/A47, S/A51, E+1 ↦ E, S/A-|D| → S
A47 ⇒ DPP ⇒ S/A ↦ S, 55/23 ↦ F, S/SW1 if K46
8
SW1 ⇒ SX1/16 ↦ AB, E+1 ↦ E, S/A → S, R/SW1
SW1.F0 ⇒ S/A±D, SX2 ↦ A, BX2 → B, K46 ↦ B31, F-1 ↦ F
SW1.F0.F7 ⇒ SX2 ↦ A, BX2 ↦ B, K46 ↦ B31, F-1 ↦ F
SW1.F0.F7 ⇒ B → S ↦ A,
FPRR ⇒ S/±A → S f(FPR), S/CPU-PH7
(FPRR)
9
S0031 → FP if (RTZ + FEUF.FZ).FPRD, FP ↦ CPU-B reg.
sustain PRx's
S/CPU-RW if FPRD.NTRAP, sustain SW1
S/CPU-CC2 if SW1, S/CPU-CC1, CC2 f(E), E ↦ E if FPR
10
S47 → FP0, E1 → FP1, E0207 → FP0207, S4871 → FP0831
(FP's enabled if (RTZ + FEUF.FZ)), FP ↦ CPU-B reg.
S/CPU-RW if NTRAP

DIV

DIV

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PRE3	(will not be reached if NFOPTION	and PRE2 was entered)	
	CLOCK TIMING: (T&L unless I/O service call, in which case FPCLEN/2 rejects T&L clocks to box)		
	s/T&L	s/T&L = PRE/12	(initial turn-on (redun.))
		+ FAFL. NIOACT. NPHIO	(hold until IOACT or PHIO)
		+ FAFL. (s/PH6/IO)	(return from I/O)
	RR → A	(s/AxRR = PRE/12)	(numer. - m.s.w)
	MB → C → D (from even location)	DxC = PREOPER. PRE3	(denom. - m.s.w)
	↳ (address forced even by S31INH in earlier phase (see "PREPARATION"))		
	1 → P31 } if long (request	PUC31 = (FAFL. NO2. PRE3. NANLZ)	(P31 = 0)
	s/MRQ } l.s.w. of denom)	s/MRQ/1 = (")	(inhibited if NFOPTION)
	s/NA → s	s/SXNA = FAFL. PRE/34	
	s/PH1 if FOPTION	s/PH1 = PRE/34. NBR	
	(TRAP if NFOPTION)		
	FPCON → box	FPCON = FAFL. PRE3	} start the box
	s/PH1 (in box)	s/PH1 = FPCON. NPH1	on next clock)
		↳ (prevents s/PH1	
PH1	NA → S	(preset in PRE3)	
PH1	NS → FP	$FP_n = I.S_n \cdot FPXS \leftarrow = NPH8.NDIS$	(R) → FP
	FP0 → S47 (→ S46)	SXFP/U = SX4607XFP = PH1.NFPDIS	(numer. mantissa - m.s.w.)
	FP0831 → S4871		
	FP0007 → S0007	SXFP/4 = SX4607XFP = PH1.NFPDIS	(numer. exponent)
	0's → S0831	(no enable hi)	
	S → A	AXS = PH1	
	0's → B (for PH8)	BX = PH1	(|quotient| reg.)
	0's → E (for PH3)	EX = PH1	(exponent reg.)
	0's → F (for PH7)	FX = PH1	(iterations ctr.)
	DO → FPCON	FPCON = FAFL. PH1. DO	
	FPCON → MWN	s/MWN = FPCON. PH1; R/MWN = PH1	(store denom. sign)
	s/LR31 } if long	s/LR31/1 = (FAFL. NO2. PH1)	} (for numer. l.s.w.)
	s/AXRR }	s/AXRR = (")	
	s/ND → s	s/SXND = FAFL. PH1	
	s/PH2	s/PH2 = PH1. NBR	
	s/PH2	s/PH2 = PH1	
PH2	ND → S	(preset in PH1)	
PH2	NS → FP	$FP_n = I.Sn.FPXS \leftarrow = NPH8.NDIS$	(EW) → FP
	FP0 → S47 → S46	SXFP/U = (SX4607XFP = PH1.NFPDIS)	(denom. mantissa - m.s.w.)
	FP0831 → S4871		
	FP0007 → S0007	SXFP/4 = (")	(denom. exponent)
	0's → S0831	(no enable hi)	
	S → D	DxS = PH2	
	1 → FPR if operand signs ≠	s/FPR = PH2. 06. (MWN ⊕ A47); R/FPR = PH1	

 FDS (3E), FDL (1E) MNEMONIC: FP: DIV

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH2 PH2 cont'd	PRESET FOR EXPONENT ARITH: NA0007 → A0007 if A0 = 1 1 → A8 1 → D8 } if denom. is neg. S/A+D → S S/A-D → S if denom. is pos. RRv1 → A S/DRQ (for EWv1) } if long S/NA → S S/PH3 s/PH3	{ S/A1 = (PH2.A0).NA1, (similarly A2-A8) AX/L = AxL = (PH2.A0) S/A8 = PH2.NMUL S/D8 = PH2.MWN S/SXAPD = S/SXAPD/1 = PH2.MWN.NMUL S/SXAMD = N(S/SXAPD).S/SXAMD/2 ← = PH2 (preset in PH1) S/DRQ/1 = FAFL.N02.PH2 S/SXNA = FAFL.PH2 S/PH3 = PH2.NBR S/PH3 = PH2	(un-invert numer. exponent) (K7 control) (for (Ae↑) + (De↓) + 1 → S) (for (Ae↑) + N(De↑) + 1 → S) K7 = G8 = 1 (numer. - l.s.w.)
PH3 PH3	(Ae↑) - (De↑) → S numer. denom S → E NA → S NS → FP FP → A0031 if long 0's → A0031 if short MBv1 → C → D if long S/ND → S S/PH4 S/PH4	(preset in PH2) S/En = Sn.PH3 (preset in PH2) FPn = I.Sn.FPXS ← = NPH8.NDIS AXFP = PH3.N02 AX/L = AXL = PH3 DXC = FAFL.N02.PH3 S/SXND = FAFL.PH3 S/PH4 = PH3.NBR S/PH4 = PH3	} (unbiased exponent difference → E) } ((Rv1) → FP if long, (R) → FP if short-insig.) } (numer. - l.s.w.) (denom - l.s.w.)
PH4 PH4	ND → S NS → FP FP → S0031 if long 0's → S0031 if short S → D0031 Clear Condition Codes S/B → S S/PH5 → see separate chart: "FAFL PH5-6" (PH4, PH4 continued on next page)	(preset in PH3) FPn = I.Sn.FPXS ← = NPH8.NDIS SXFP/4 = SXFP/A = S0031XFP = PH4.N02.NFPDIS (no enable hi) DX/L = DXS/L = PH4 R/CC = (FAFL.PH4) S/SXB = (FAFL.PH4) S/PH5 = PH4.NBR	} ((EWv1) → FP if long, (EW) → FP if short-insig.) (denom. - l.s.w.)

Sheet 2 of 6

MNEMONIC: FP: DIV

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH4 PH4	EXISTING CONDITIONS: A contains (R), (Rv1)/0's B " 0's D " (EW), (De)in D0007 E " (Ae↑) − (De↑) F " 0's	(mantissa of numerator) ((EWv1)/0's is being clocked in D0031 This phase) (tentative quotient exponent)	
	DSN ⇒ S/|A| → S S/PH6	(denominator is "simple normalized") S/SXA = NA47. SXAVA ← S/SXMA = A47. SXAVA ← = PH4.06.DSN. N(S/PH7) S/PH6 = PH4.06.DSN.N(S/PH7)	
	NDSN ⇒ S/D → S S/SW2 (for A ↛ D in PH5) S/PH5	S/SXD = PH4.06.NDSN S/SW2 = S/SW2/1 = PH4.06.NDSN S/PH5 = PH4.06.NDSN	
PH5 or PH6 PH5	(entered only if denominator requires prenormalization) (SW2 + NASN) ⇒ N(S/PH6) ⇒ A ↛ D (first clock only) {D → S (") {A → S (after first clock) S x 16 ↛ A E+1 ↛ E	N(S/PH6) = N(PH5.06.ASN.NSW2) DXA = PH5.SW2 ← (repeater, set in PH4) (save numerator) (preset in PH4) S/SXA = AXSL4/1 ← AXSL4 = AXSL4/1 ← = PH5.06.N(S/PH6) EUC7 = PH5.DIV.N(S/PH6)	
	{S/RTZ if S = 0 (on first clock) RTZ ⇒ {raise FPRR S/PH9, etc. − (see S/SW1 NRTZ ⇒ sustain PH5	S/RTZ = PH5.SZU.SZL.NASPP.NSXADD FPRR = PH5.06.RTZ FPRR functions at end of PH8) S/SW1 = PH5.DIV.RTZ S/PH5 = PH5.06.N(S/PH6).NRTZ	} (denom. = 0 means overflow -- will cause trap)
	(NSW2.ASN) ⇒ (denominator is "simple normalized" − 2nd clock or later) S/|D| → S S/A ↛ D S/PH6	{S/SXD = ND46.S/SXAVD ← {S/SXMD = D46.S/SXAVD ← S/SW2 = S/SW2/1 ← S/PH6 = ← = (PH5.06.ASN.NSW2)	
PH5 or PH6 PH6	(entered from PH4 or PH5) ON THE FIRST CLOCK: |A| → S if entry from PH4 |D| → S } " " " PH5 A ↛ D }	} (preset logic) DXA = PH6.SW2	} |numerator| → S "simple normalized" denom. ↛ D
	(NASN + NDSN) ⇒ N(S/PH7) ⇒ S/A → S (for clocks after first) S x 16 ↛ A E − 1 ↛ E	N(S/PH7) = N(PH6.06.ASN.DSN) (numerator requires normalization) S/SXA = NFPRR.AXSL4/1 ← = PH6.06.N(S/PH7) AXSL4 = AXSL4/1 ← EDC7 = N(PH5.DIV).AXSL4/1	

Sheet 3 of 6

MNEMONIC: FP : DIV

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5 or PH6 PH6 cont'd	S/RTZ if S=0 (on first clock) RTZ ⇒ raise FPRR; s/PH9, etc. - (see FPRR functions at end of PH8) NRTZ ⇒ sustain PH6	S/RTZ = PH6.SZU.SZL.NASPP.NSXADD FPRR = PH6.RTZ.NPH5 S/PH6 = PH6.06.N(s/PH7).NRTZ	(numerator = 0 with denominator ≠ 0 ∴ quo = 0)
	(ASN.DSN) ⇒ (both numerator and denominator are simple normalized) S ↠ A S/A-|D| → s s/PH7	AXS = PH6.06.ASN.DSN {s/SXAPD/1 = MWN.DIV.(s/PH7) s/SXAMD/1 = NMWN.DIV.(s/PH7)} = PH6.06.ASN.DSN S/PH7 =	|numerator| ↠ A
PH6 PH7	(scaling phase) (A47 = 1) ⇒ (can occur only if original numerator was = -1 or -1/16) (numer. = +1) 0 ↠ A 1 ↠ AS1 } A x 1/16 ↠ A E+1 ↠ E sustain PH7 S/A-|D| → s	AX = (PH7.DIV.A47) S/AS1 = (") EUC7 = (") S/PH7 = ← {s/SXAPD/1 = MWN.DIV.(s/PH7) s/SXAMD/1 = NMWN.DIV.(s/PH7)} = PH7.DIV.A47	
	(A47 = 0) ⇒ DPP ⇒ 23 ↠ F if short (00010111) 55 ↠ F if long (00110111)	DPP = PH7.DIV.NA47 (S/F3 = S/F5 = S/F6 = S/F7 = DPP) ("), S/F2 = DPP.N02	(numer. < +1)
	S/A → S s/SW1 if |numer| ≥ |denom| s/PH8	S/SXA = PH7.N(s/PH7) s/SW1 = DPP.K46.DIV S/PH8 = PH7.N(s/PH7)	(K46 means ⊤ sum)
(PH8.SW1): (entered only if |numer.| ≥ |denom.|) (lasts one clock.) PH6			
	A → S S x 1/16 ↠ A S2831 ↠ B48S1 E+1 ↠ E R/SW1 S/A → S sustain PH8 (with SW1 = 0)	(preset in PH7) AXSR4 = AXSR4/1 = PH8.DIV.SW1 S/B48 = S/B48/1 = S28.(PH8.DIV.SW1), similarly B49, B50, B51 EUC7 = AXSR4/1.NPH5 R/SW1 = NPH9 S/SXA = PH8.DIV.SW1 S/PH8 = PH8.NFPRR	
(PH8.NSW1): (Iterations phase.) (26 clocks if short; 58 clocks if long) PH6			
	CONTROL SIGNALS: DIT (all but final 2 clocks) DIT/1 (all but final clock) FPRR (final clock)	DIT = (PH8.DIV.NSW1).NF0 DIT/1 = (").NFPRR FPRR = PH8.DIV.F0.NF7	(F > -1) (F > -2) (F = -2)
	CONTROL FUNCTIONS: F-1 ↠ F (count iterations) sustain PH8	FDC7 = DIT/1 S/PH8 = PH8.NFPRR	

MNEMONIC: FP: DIV

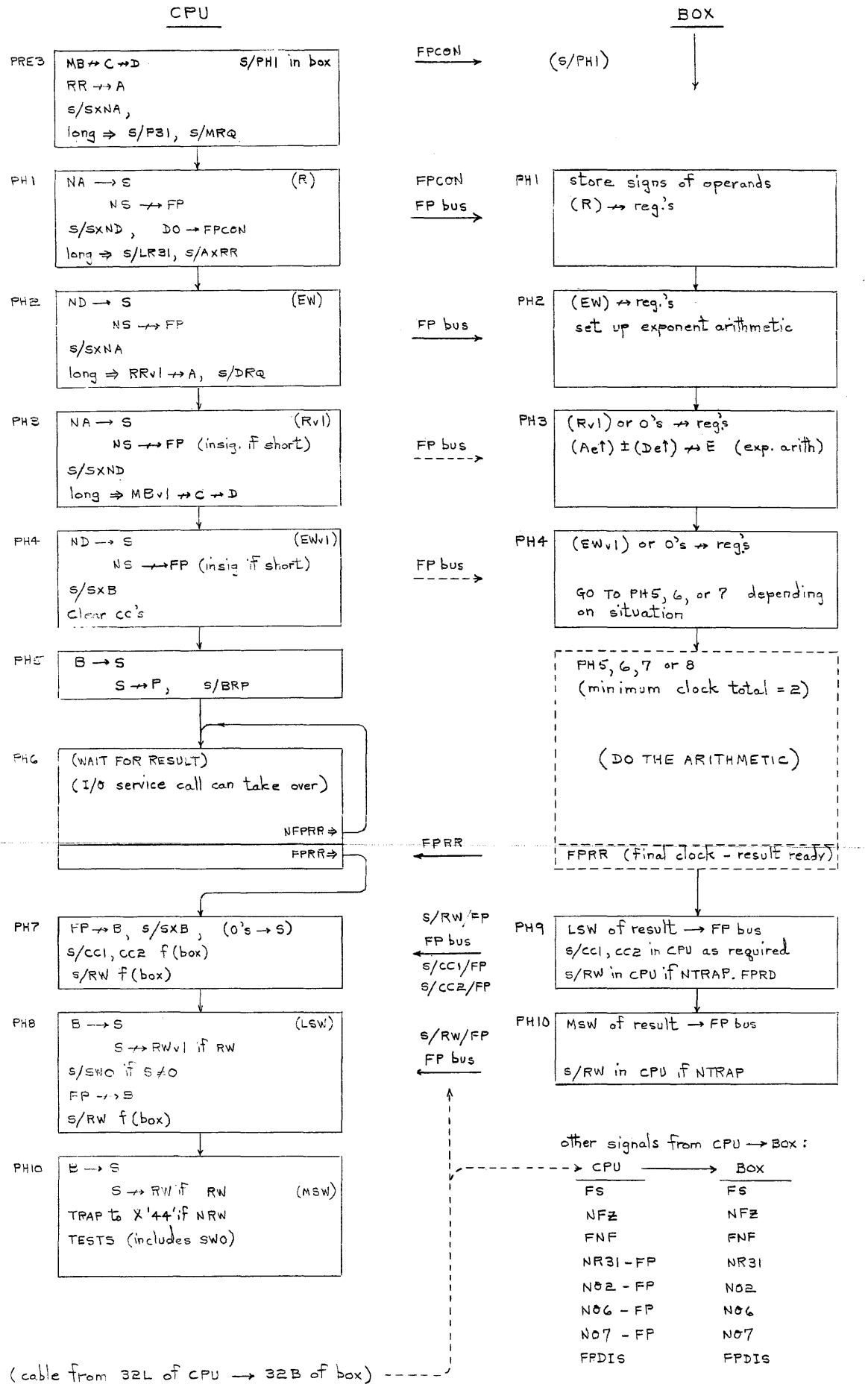
PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
(PH8 .NSW1)			
PH6 cont'd	REGISTER CONTROL, etc.:		
	S/A+D → S if (K46⊕SXADD⊕MWN)	S/SXAPD reduces to DIT. (K46⊕SXADD⊕MWN)	(spec. mech.)
	S/A-D → S if N(")	S/SXAMD = N(S/SXAPD). S/SXAMD/2← = DIT	
	S4831 ↛ A4730	AXSLI = DIT/1	
	B48 ↛ A31	S/A31 = AXSLI-7. B48	(for PH8.SW1 case)
	B4931 ↛ B4830	BXBLI = DIT/1	
	K46 ↛ B31 if long	S/B31 = BXBLI-L. K46. SW0	} |quotient| bit = 1
	K46 ↛ B71 if short	S/B71 = BXBLI-U. K46. NSW0	}
	ON THE 1st CLOCK: (DIT. NSXADD)	(F = 55 if long, 23 if short --- i.e. initial setting)	
	A → S	(preset by previous clock)	
	K46 = 0, SXADD = 0	(arithmetic unit not adding - i.e: A → S was preset)	
	hence the operation amounts to: [AB x 2 ↛ AB, and S/A - |D| → S]	- i.e. "go for quo. 2^{-1}"	
	S/SW0 if long	S/SW0 = BXBLI-L. N02	(for K46 ↛ B71/B31)
	ON THE NEXT TO FINAL CLOCK:	(F = -1)	
	(don't S/A±D → S)	(see REGISTER CONTROL - DIT is low	because F0 = 1)
	FPRR ⇒ (Floating Point Result Ready)	← (F=-2)	
	raise FPRR { (1/16 ≤ |quo| < 1); (denom = 0); (denom ≠ 0). (numer = 0)	FPRR = PH8. DIV. F0. NF7	(FINAL CLOCK PH8.NSW1)
		+ PH5. 06. RTZ	(2nd clock of PH5)
		+ PH6. RTZ. NPH5	(2nd clock of PH6)
	stop CLOCK → box if I/O service	call has CPU: FPCLEN/1 = NFPRR +	NIOEN. NIOIN
	S/PH7	S/PH7 = PH6. NBR← = NBRPH6 = N(	FAFL. PH6. NFPRR)
	S/MRQ (for next instruction)	S/MRQ/1 = FAFL. PH6. NBRPH6. NIOEN	
	B → S (0 → S47)	SXB = FPRR. DIV. NSDIS	(B = 0 if NPH8)
	S ↛ A	AXS = FPRR. DIV	(|quo| ↛ A)
	S/A → S if +/+ or -/-	S/SXA = FPRR. NFPR	} (put result in
	S/-A → S if +/- or -/+	S/SXMA = FPRR. FPR	} proper polarity)
	S/PH9	S/PH9 = FPRR	
PH7 PH9	CONTROL SIGNALS USED IN PH9-10:		
	FPRD	FPRD = N02	(result double length)
	TRAP	TRAP = SW1	(divide by zero)
		+ FEOF	(exp. overflow)
		+ FEUF. N(FEUF. NFZ) = FEUF. FZ	(exp. underflow). (FZ = 1)
	FEOF	FEOF = NE0. E1. NRTZ	(exp. overflow)
	FEUF	FEUF = E0. NE1. NRTZ. N(B65. N06. FS. NFZ);	(exp underflow)
	A or -A → S	(preset in PH5, 6, or 8 by FPRR)	
	S0031 → FP if not, 0's → FP if (short or result = 0 or underflow and FZ = 0)	FPXSL = PH9. NFPDIS. FPRD. NRTZ. N(FEUF. NFZ)	} (result - l.s.w.)
	FP ↛ B	BXFP = (FAFL. PH7)	}
	S/B → S	S/BXS = (")	}
	S/LR31	S/LR31 = (")	}
	S/RW if long and not trap	S/RW = S/RW/FP = PH9. FPRD. NTRAP	}
	S/CC1 if exponent underflow	S/CC1 = S/CC1/FP = PH9. FEUF	
	S/CC2 if exponent underflow or overflow or if divide by zero	S/CC2 = S/CC2/FP = PH9. FEUF + PH9. FEUF + PH9. SW1	

MNEMONIC: FP: DIV

PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH7 PH9 cont'd.	S/A → S if NFPR S/-A → S if FPR	S/SXA = PH9.NFPR S/SXMA = PH9.FPR	(sustaining from PH8)
	NE ↛ E if FPR. (E0 = E1) ↳ qualifier	EXNE = PH9.FPR.NTRAP.N(FEUF, NFZ) prevents interchanging FEUF and FEOF	(invert exp. when result neg.)
	S/PH8 S/PH10	S/PH8 = PH7.NBR S/PH10 = PH9	
PH8 PH10	B → S S ↛ RW, 1 if long and not trap S ≠ 0 ⇒ 1 ↛ SW0 (for TESTS)	(preset in PH7) S/SW0 = N50031Z.(S/SW0/NZ) ← = FAFL.N02.PH8	(result - l.s.w.)
	A or -A → S S47 → FP0 (exp.) NE1 → FP1 (+64 bias) E0207 → FP0207 S4871 → FP0831 FP ↛ B	(preset in PH9) ← note: K71 is forced high in short "-A" case to assure proper truncation: K71 = G0003 = PH10.NFPRD.K31 FPXSU = PH10.NFPDIS.NRTZ.N(FEUF.NFZ) (FPXS = NPH8.NDIS -- blocks S (=B) → FP) BXFP = FAFL.PH8	
	S/B → S S/RW if not trap case	S/SXB = FAFL.PH8 S/RW = S/RW/FP = PH10.NTRAP	(result m.s.w)
	S/DRQ S/PH10 R/PH10 (no more box action)	S/DRQ = BRPH10 = FAFL.PH8 S/PH10 = BRPH10 = FAFL.PH8 (no set term hi)	
PH10	B → S S ↛ RW	(preset in PH8)	(result - m.s.w.)
	TESTS (CC3, CC4 control) (note: SW0 = 1 causes S > 0 (ADD/SUB).(FN=1) case only, where result is +, exp. = -64, and significance in l.s.w. only.)	TESTS = FAFL.ENDE indication for double length zero test; this is needed for	
	Stop sustaining TBL	S/TBL = FAFL.NIOACT.NPH10	
	TRAP to X'44' if RW is off	S/TRAP = (FAFL.ENDE.NRW) S/TR29 = (")	(NRW means TRAP in the box was hi last cl.)
	ENDE	ENDE = PH10.EXC	

Sheet 6 of 6

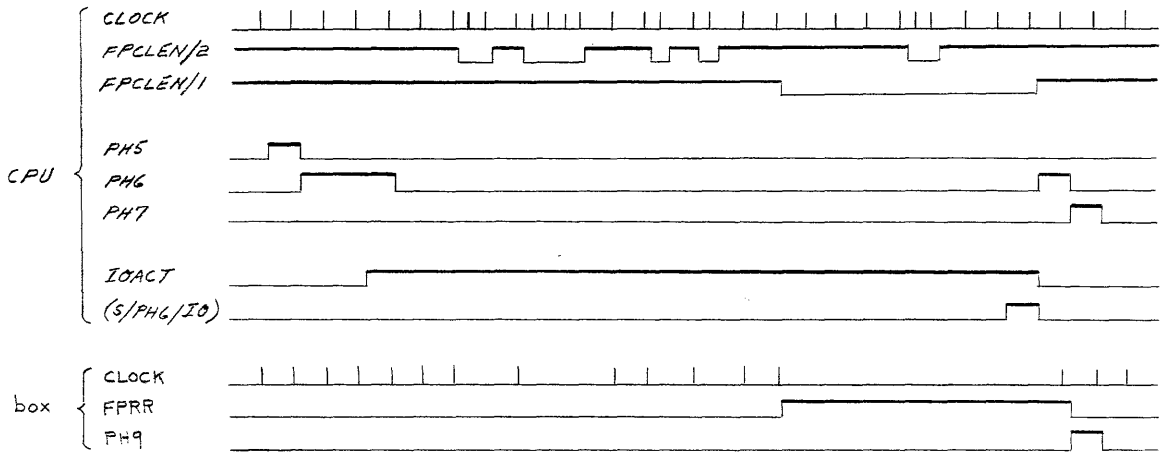
MNEMONIC: FP: DIV



CPU ↔ BOX "BIG PICTURE"

NOTE: Since PH5 and PH6 in the CPU can occur during a variety of box phases, the activity is shown on this separate chart. The last phase where the CPU feeds the box is PH4; the box starts feeding the CPU in PH7 (box PH9).

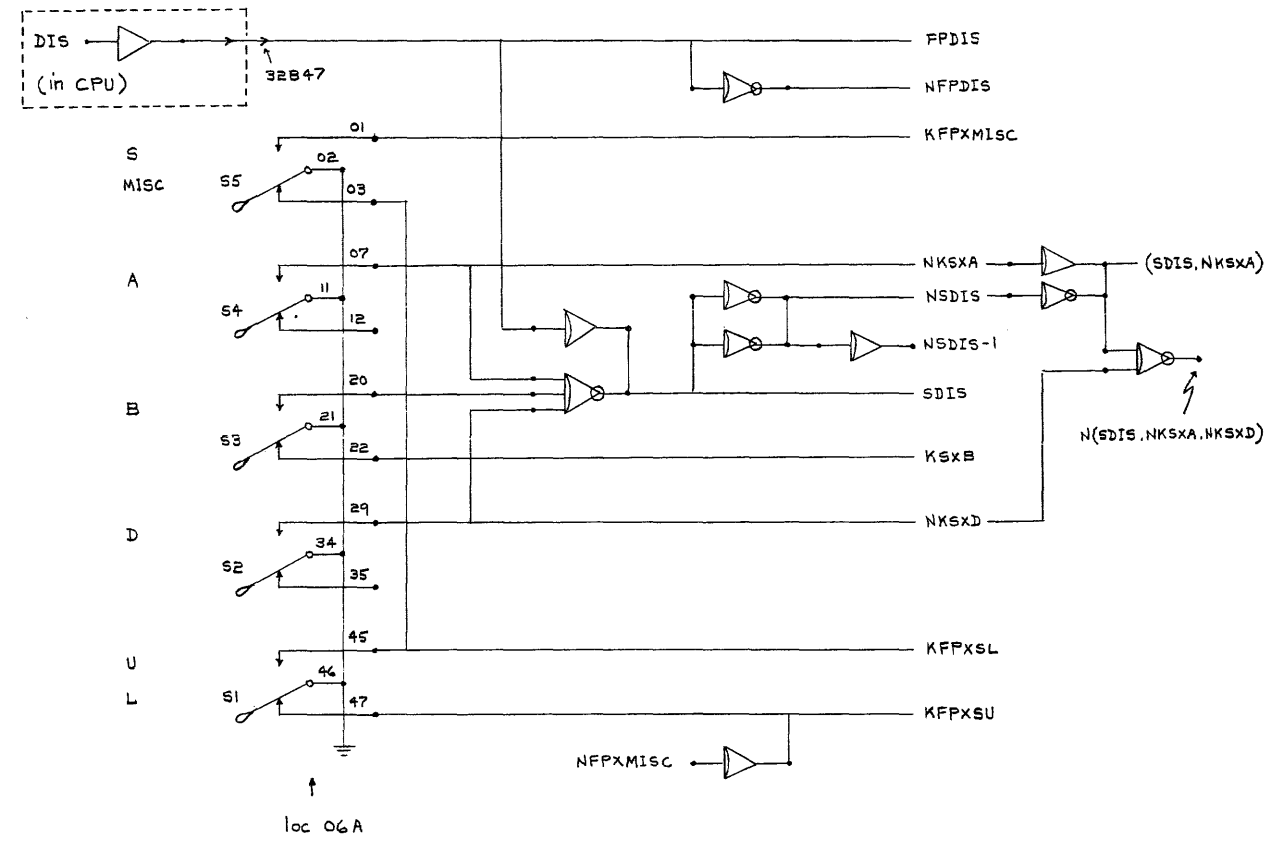
During PH6 an I/O service call can grab the CPU, but the box will continue to compute its results, subject to CPU control of CLOCK to the box. Example:



PHASE	FUNCTION PERFORMED	SIGNALS INVOLVED	COMMENTS
PH5	(can concur with box PH5, 6, or 7,	but never with FPRR)	(lasts one clock)
	B → S S → P S/BRP	(preset in PH4 by FAFL.PH4) PXS = FAFL.PH5 S/BRP = FAFL.PH5	} (program addr. → P) (set prog. addr. pointer)
	(The above releases the B register for use later as receiver of results from the box)		
	S/PH6	S/PH6 = PH5.NBR	
PH6	(Can concur with box PH5, 6, 7, or	8)	
	"WAIT PHASE": terminated by or interrupted by	FPRR (from box) IOACT = IOEN + IOIN	(FP Result Ready) (I/O service call)
	NIOACT.NFPRR ⇒ sustain PH6 enable I/O entry	S/PH6 = NIOEN.NCLEAR.BRPH6 ← = IOEN6 = FAFL.PH6.NFPRR.NDIOEXIT.	FAFL.PH6.NFPRR NFSHEX.IOEN6/1
	IOACT ⇒ S/TBL control	S/TBL = FAFL.NIOACT.NPH10 + FAFL.(S/PH6/IO) + terms not associated with	(low during IOACT) (final clock of IOACT) FAFL
	block CLOCK → box if TSL or FPRR	FPCLEN/2 = NTSEN	(low if TSEN)
	S/PH6 on final clock	FPCLEN/1 = NFPRR + NIOEN.NIOIN BRPH6 = IOPH1.SW13.(S/PH6/IO)	(low if FPRR.IOACT)
	NIOACT.FPRR ⇒ S/MRQ (for next instruction) S/PH7	S/MRQ/1 = FAFL.PH6.NBRPH6.NIO S/PH7 = PH6.NBR ← = NBRPH6	EN

Sheet 1 of 1

MNEMONIC: FAFL PH5-6



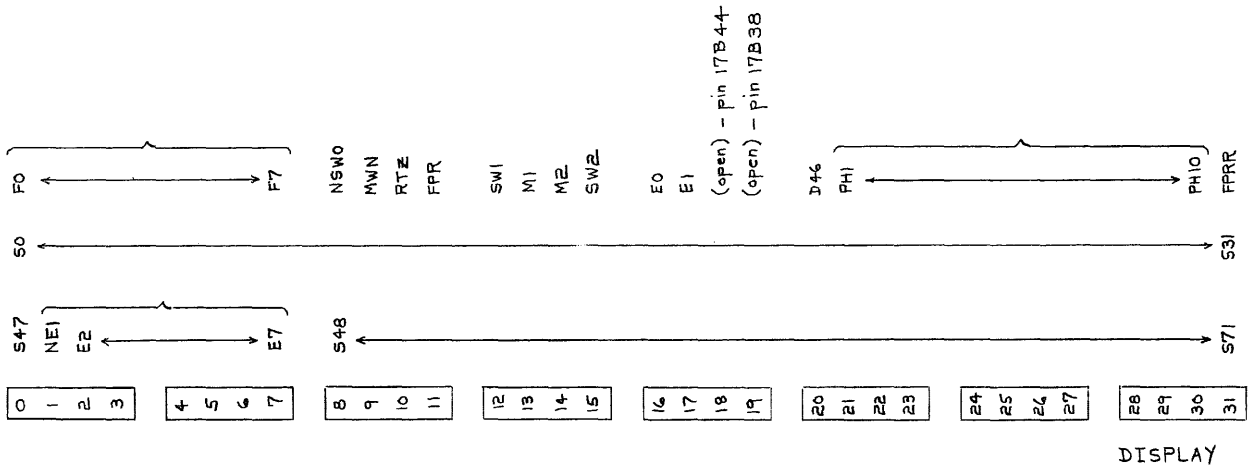
down/up		MISC	SL (norm)	SU (norm)	AL	AU	BL	BU	DL	DU
MISC/S	S5	0	1	1	1	1	1	1	1	1
— /A	S4	x			1	1				
— /B	S3	x					1	1		
— /D	S2	x							1	1
SL/SU	S1	x		1		1		1		1

↑ 0's will be displayed when input to s is FP

CPU :

REGISTER SELECT rotary switch must be on EXT to display the box

REGISTER DISPLAY Toggle switch must be ON to obtain displays selected by the switches in module 06A in the box. (OFF position will display normal FP bus info.)



FPDIS = DIS from CPU. High if DISPLAY SWITCH is on when SINGLE CLOCKING

SW5 (top) → down ⇒ MISC → FP

SW5 (top) → up → SW1 (bottom) → down ⇒ S0031 → FP0031

SW1 (bottom) → up ⇒ S47 → FP0, NE1 → FP1, E0207 → FP0207, S4871 → FP0831

(note: FP → S qualified by NFPDIS to prevent latch resulting from FP → S)

SDIS = FPDIS.(KSXA + KSXB + KSXB) (S4 or S3 or S2 up)

Causes removal of normal logic from S and substitutes:

A → S if S4 up, B → S if S3 up, D → S if S2 up

(a merge takes place if more than one switch is up)

defines BOX → CPU cases

NFPX = N(FPDIS + PH9 + PH10) (raises "Box" end of FP bus to enable CPU to control it in early phases)

FPXMISC = FPDIS.KFPXMISC

FPXSL = FPDIS.KFPXSL + NFPDIS.PH9.NRTZ.N(FEUF,NFZ), FPRD

FPXSU = FPDIS.KFPXSU + NFPDIS.PH10.NRTZ,N(FEUF,NFZ)

SXFP
- S47 → S46, FP0 → S47, FP0831 → S4871: SXFP/U = NFPDIS.(PH1 + PH2)
- FP0007 → S0007: SXFP/4 = NFPDIS.(PH1 + PH2 + PH4.N02)
- FP0831 → S0831: SXFP/A = NFPDIS.PH4.N02

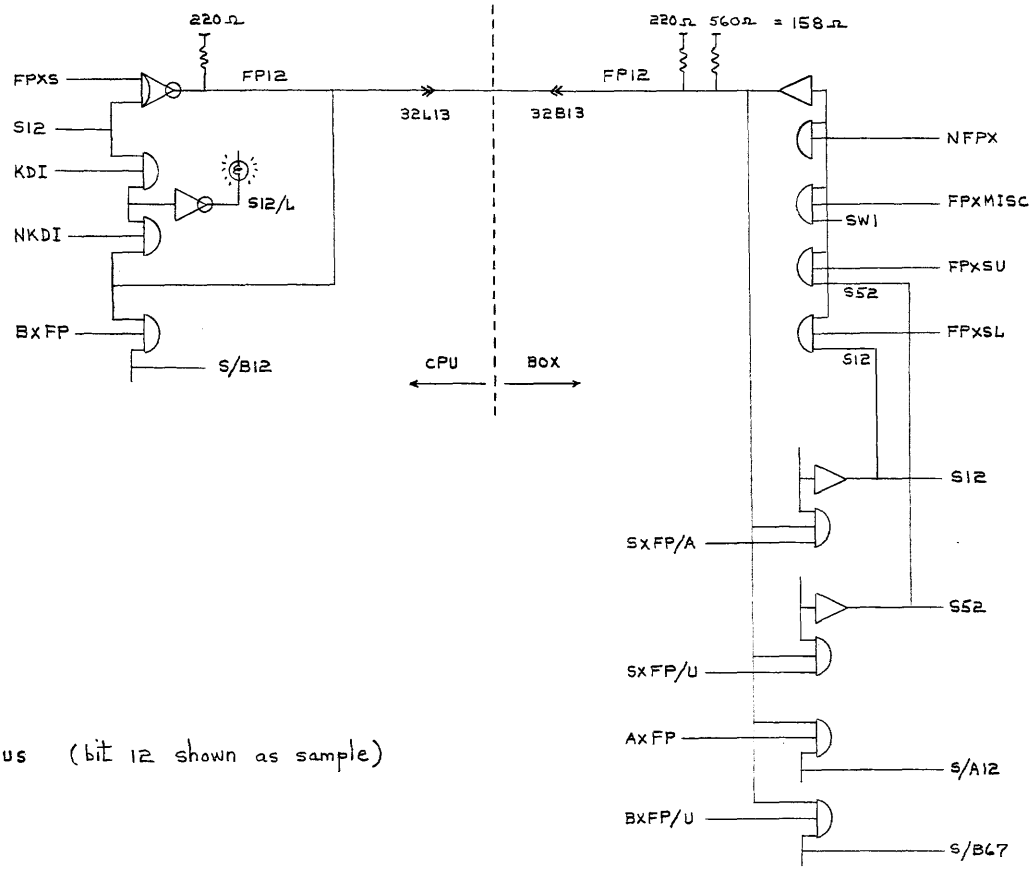
SXB = SDIS.KSXB + NSDIS.FPRR.DIV

PRXAD/ = SDIS.(KSXA + KSXD) + NSDIS.PRXAD

PRXAND/ = SDIS.KSXA + NSDIS.PRXAND

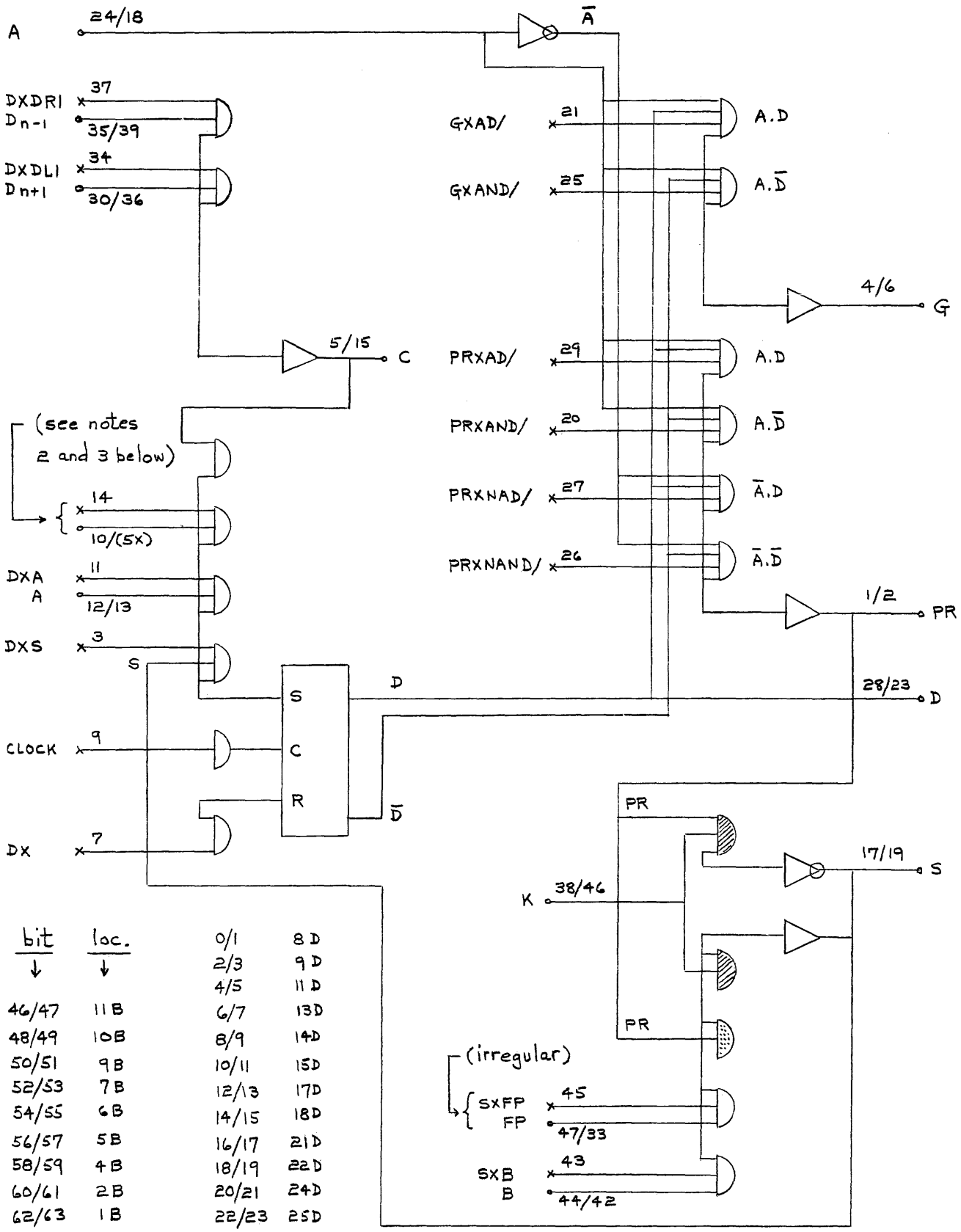
PRXNAD/ = SDIS.KSXD + NSDIS.PRXNAD

PRXNAND/, GX/'s, and K31 are all qualified by NSDIS



FP bus (bit 12 shown as sample)

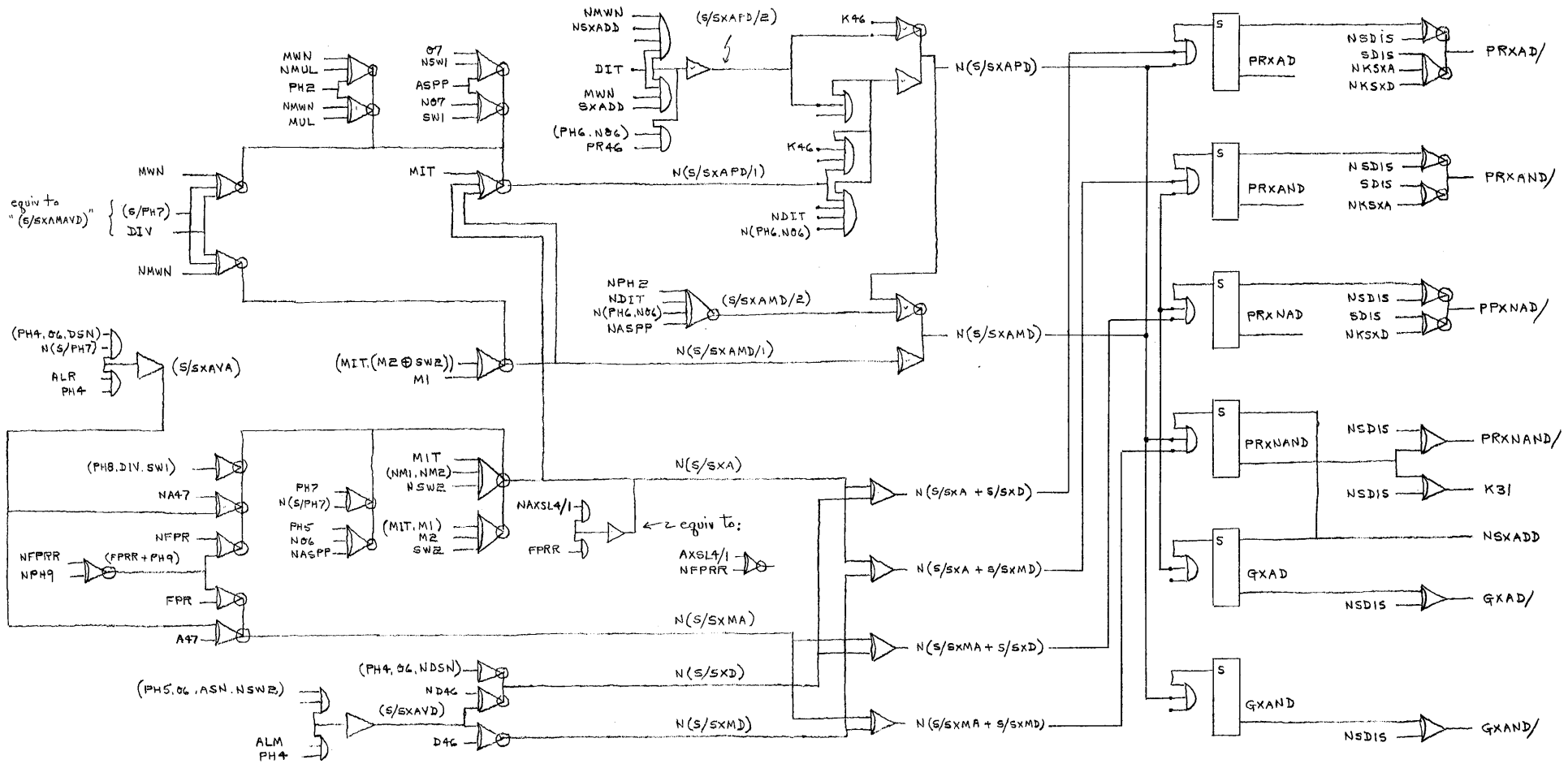
BUS SYSTEM



bit ↓	loc. ↓		
		0/1	8 D
		2/3	9 D
		4/5	11 D
46/47	11 B	6/7	13 D
48/49	10 B	8/9	14 D
50/51	9 B	10/11	15 D
52/53	7 B	12/13	17 D
54/55	6 B	14/15	18 D
56/57	5 B	16/17	21 D
58/59	4 B	18/19	22 D
60/61	2 B	20/21	24 D
62/63	1 B	22/23	25 D
64/65	1 D	24/25	26 D
66/67	2 D	26/27	27 D
68/69	6 D	28/29	30 D
70/71	7 D	30/31	31 D

1) grounded gates and unused pins not shown.
2) pin 14: PH2 on bit 8 (and 9- insig.), gnd. on others.
3) pin 10: MWN on bit 8 (open or C_{n-1} on others - insig.)

LT42 ADDER MODULE (½)



(see equations for mechanization details)

e.g. might be

(repeaters, played upside-down)