KL1Ø Student Guide

TABLE OF CONTENTS

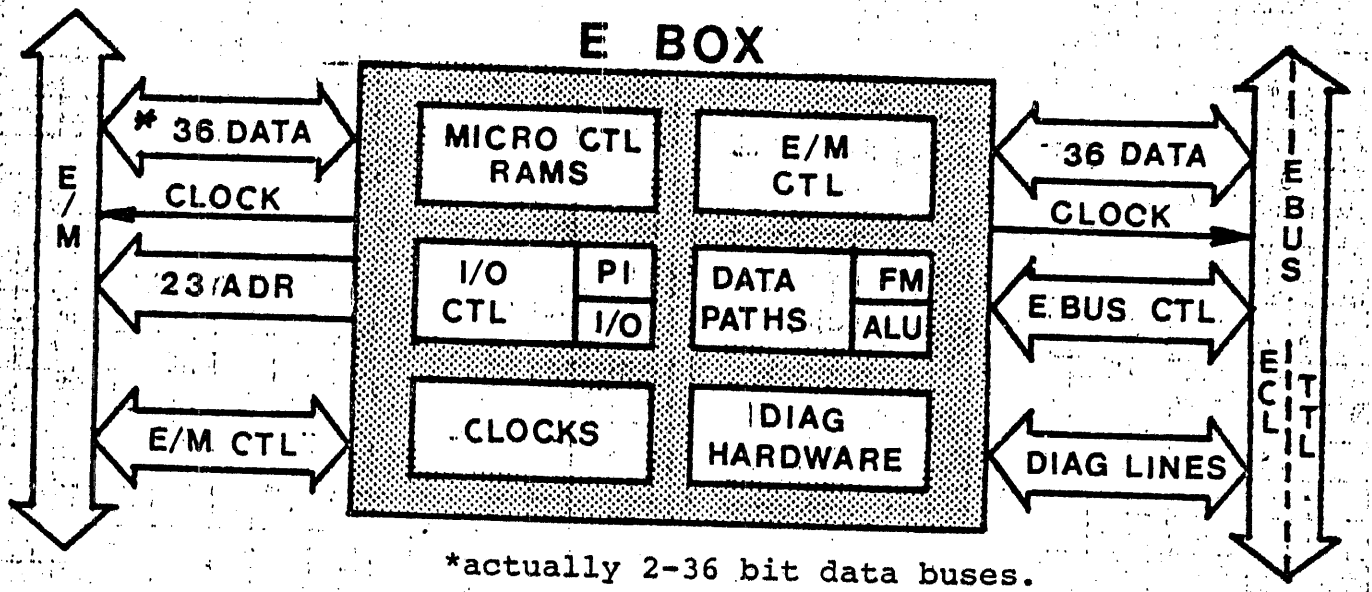
	Page
E BOX	
Block Diagrams	E 1-1
Basic Machine Cycle	E 1-2
Data Path	E 1-3
D-RAM	E 1-4
C-RAM	E 1-7
Micro Program Flow Chart	E 1-16
Instruction Cycle Overview	E 1-18
Page Fail Handling	E 1-27
Simplified Clock Control	E 1-29
Simplified PI System Block	E 1-3Ø
PI Timing Chart	E 1-34
Meter Board Overview	E 2-1
M BOX	
General Overview	1-1
Cache Memory	1-1
Memory Buffers	1-1
Address Modification and Verification Logic	1-1
Channel Logic	1-4
Cycle Control Logic	1-4
General Interface	2-1
E/MBox Interface Lines	2-1
SBus Interface Lines	2-5
Channel Control to Cache Interface	2-8
FUNCTIONAL DESCRIPTION	
Data Flow	3-1
Memory Buffer Input Mixer	3-1
Memory Buffers	3-1
Memory Buffer Output Mixer	3-1
Cache Data Input Mixer	3-3
Cache Memory	3-3
Page Table Input Mixer	3-3
Channel Data Input Mixer	3-3
Channel Data Buffer	3-3
CBus Output Buffer	3-3
MB Channel Buffer	3-3
Channel Data Reverse Mixer	3-3
CCW Buffer	3-4
CCW Input Mixer	3-4
Cache Memory	3-4
Functional Description	3-4
Cache Read Operation	3-6
Cache Write Operation	3-9
Cache Use Logic	3-9

Pager 3-13
General 3-13
Modes of Operation 3-13
Pager Functional Description 3-14
Operation 3-19
PMA Selection 3-20
General 3-20
Cache Addressing 3-23
Address Selection 3-24
Core Control 3-29
Acknowledge Pulse COntrol Logic 3-31
Data Valid Pulse Control Logic 3-33
Error Checking and Reporting 3-34
Address Parity Checking 3-34
Data Parity Checking 3-35
Timeout Errors 3-37
Error Flags 3-37
Diagnostic Logic 3-40
MBox Functional Operation 3-41

General Control Logic 4-1
Detailed Description 4-4
Cache Control 4-4
MB Control 4-9
Core Control 4-12
EBox Requests 4-17
CSH EBox Cycle 4-18
MB REQ Generation 4-42
MB Cycle 4-42
Writeback Cycle 4-44
Page Refill Cycle 4-46
Cache Clearer Control 4-46
Cache Clearer Request 4-48
CCA Cycle 4-48
Channel Cycle 4-51

DTE-2Ø

General Introduction D 1-1
Basic Programming Overview D 1-4
Door bell function D 1-6
Byte Xfer function D 1-9
Error Overview D 1-9
Diagnostic Overview D 1-1Ø
Interface Communication D 1-12
Data Xfer Signals D 1-13
DTE RAM D 1-14
RAM Load Sequence D 2-1
RAM READ Sequence D 2-3
DEX - Deposit D 2-5
DEX - Examine D 2-11
TO 11 Transfer D 2-13
TO 1Ø Transfer D 2-18
BR D 2-23
NPR D 2-24
KL1Ø PI D 2-26
Diagnostic Bus Operation D 2-29
KLIØ CONO, CONI Formats D 2-37
EBus Parity D 2-38



*actually 2-36 bit data buses.

E BOX FUNCTIONS

A. EXECUTES INSTRUCTIONS.

B. PROVIDES ALL PROCESSOR CLOCKS.

C. HANDLES INTERRUPTS.

D. GENERATES A VIRTUAL MEMORY ADDRESS TO THE M BOX.

E. GENERATES I/O COMMANDS TO DEVICES.

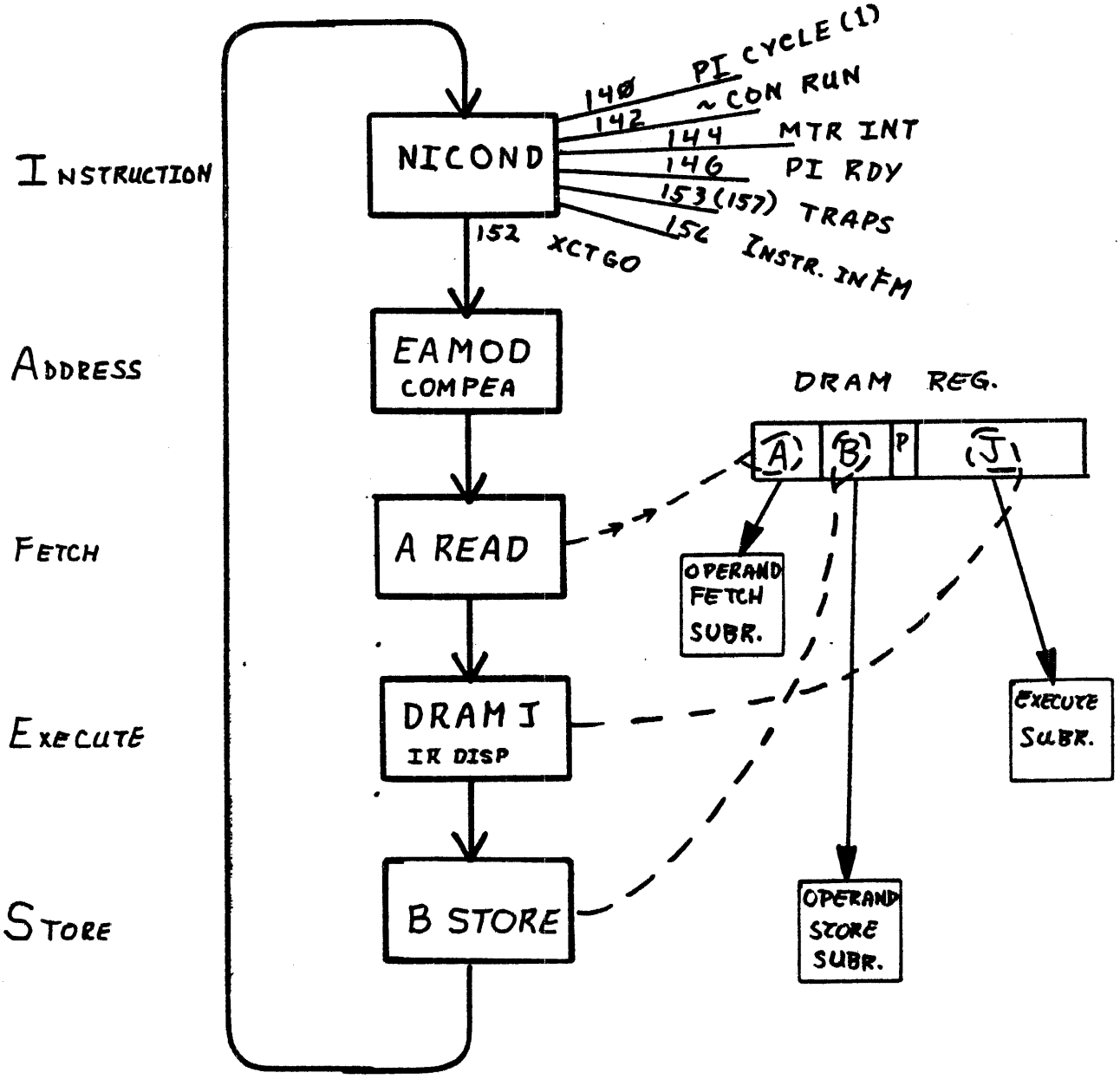
F. PROVIDES A METER WHICH KEEPS TRACK OF SYSTEM PERFORMANCE.

G. SEQUENCES INSTRUCTIONS.

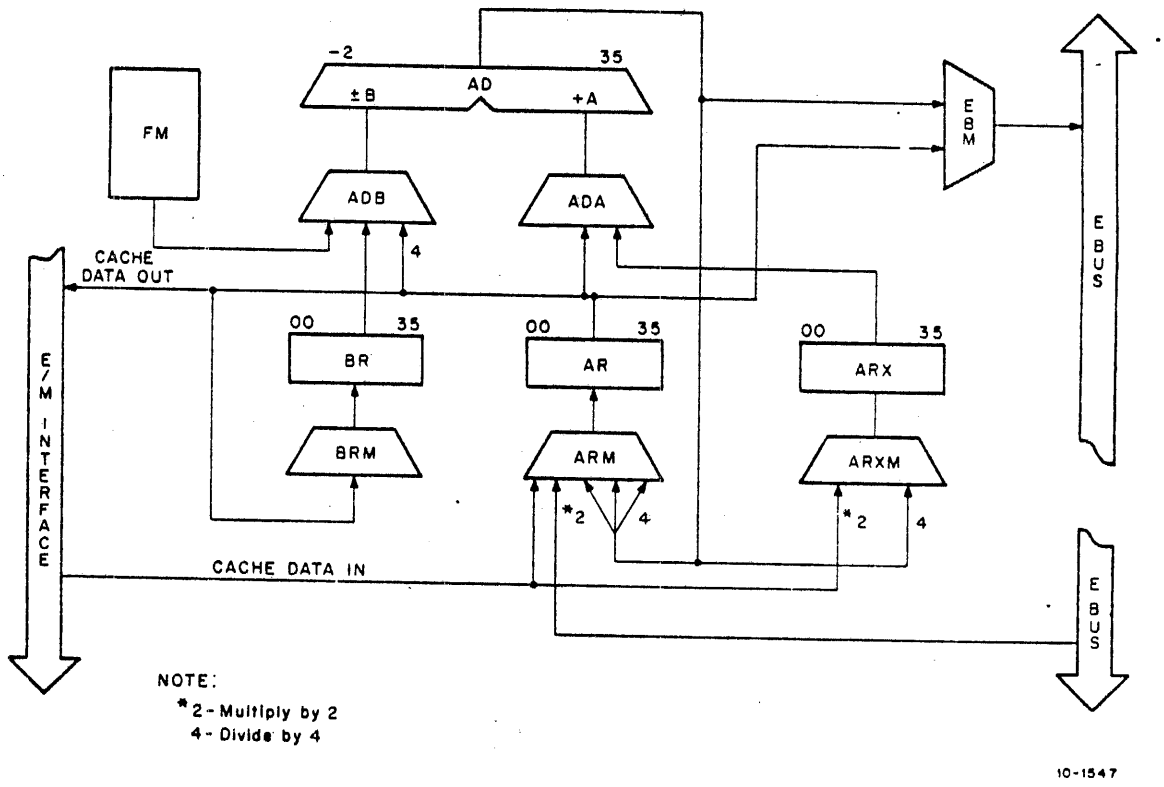
H. PROVIDES A DATA PATH BETWEEN THE DTE-20 AND KL MEMORY.

I. CONTAINS DIAGNOSTIC HARDWARE TO HELP THE 11/40 ISOLATE MALFUNCTIONS IN THE E BOX.

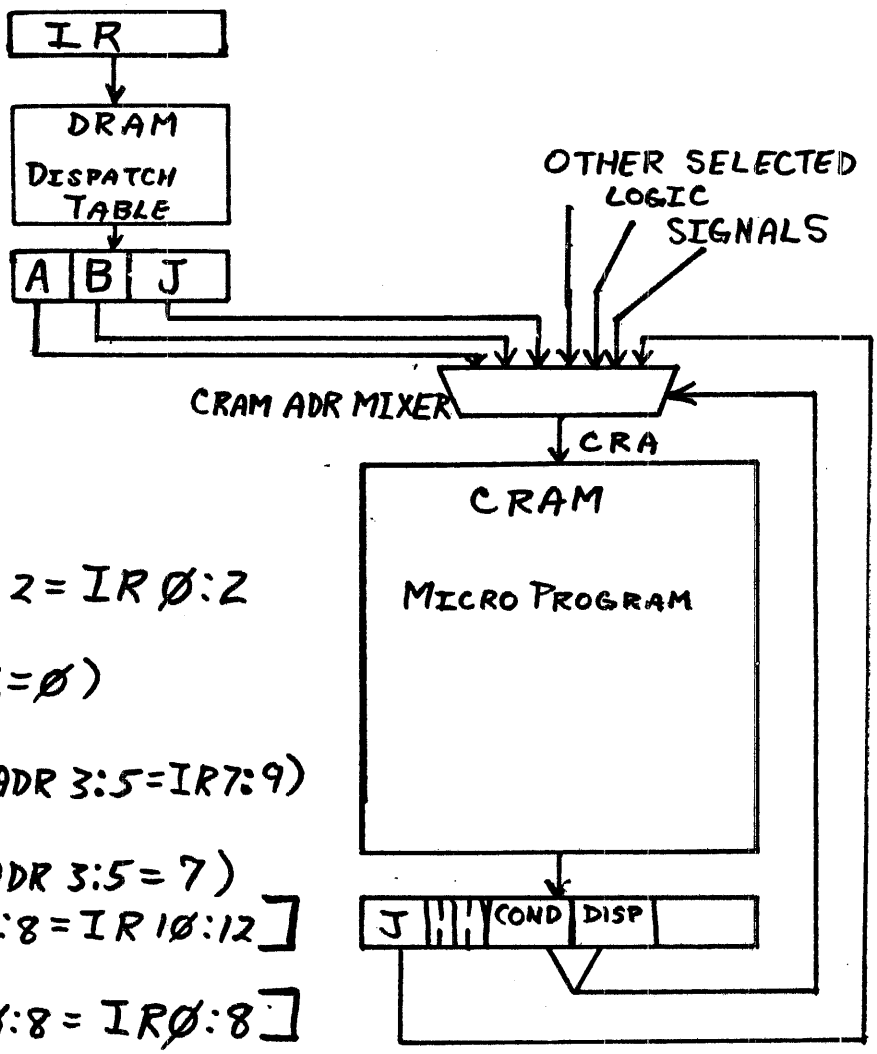
BASIC MACHINE CYCLE



SEE.
EBOX 2-95



EBox Data Path Simplified



DRAM
ADDRESS

IF [IR Ø:2 = 7]

THEN [DRADR Ø:2 = IR Ø:2

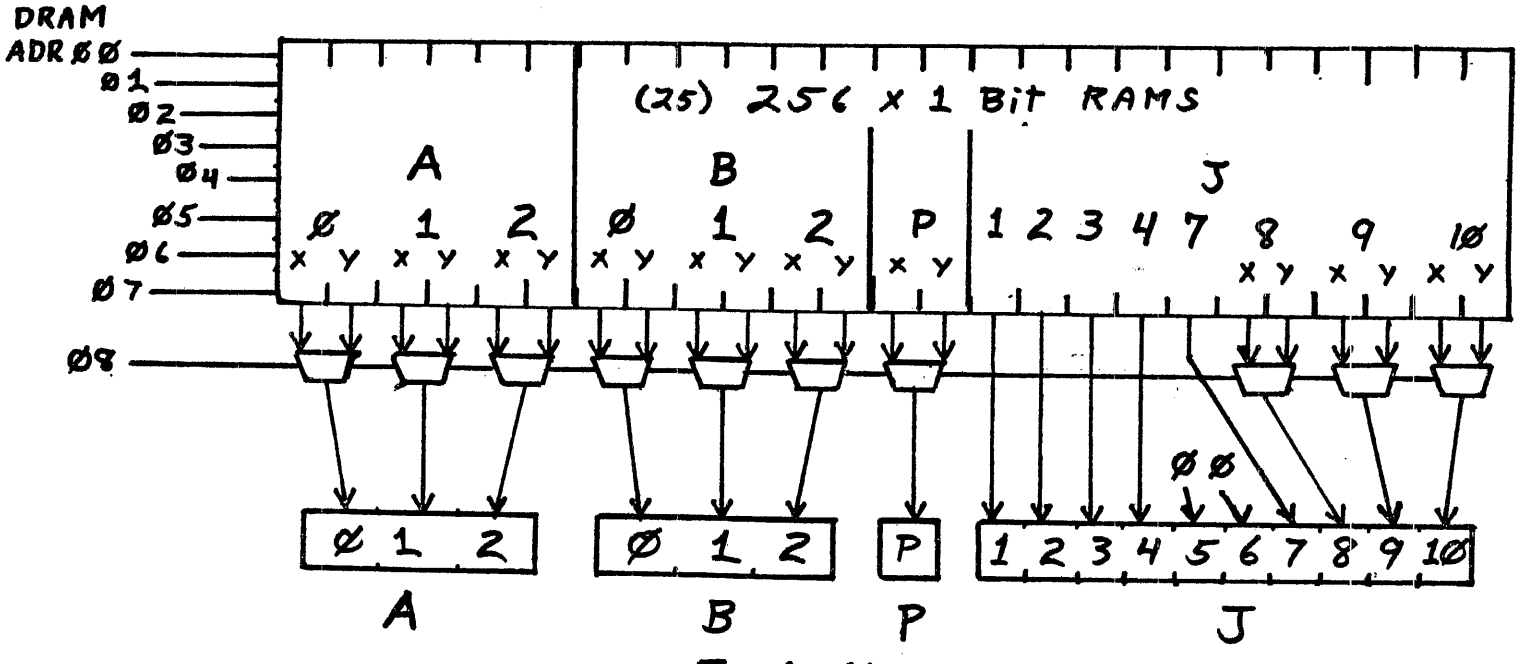
AND IF(IR 3:6=Ø)

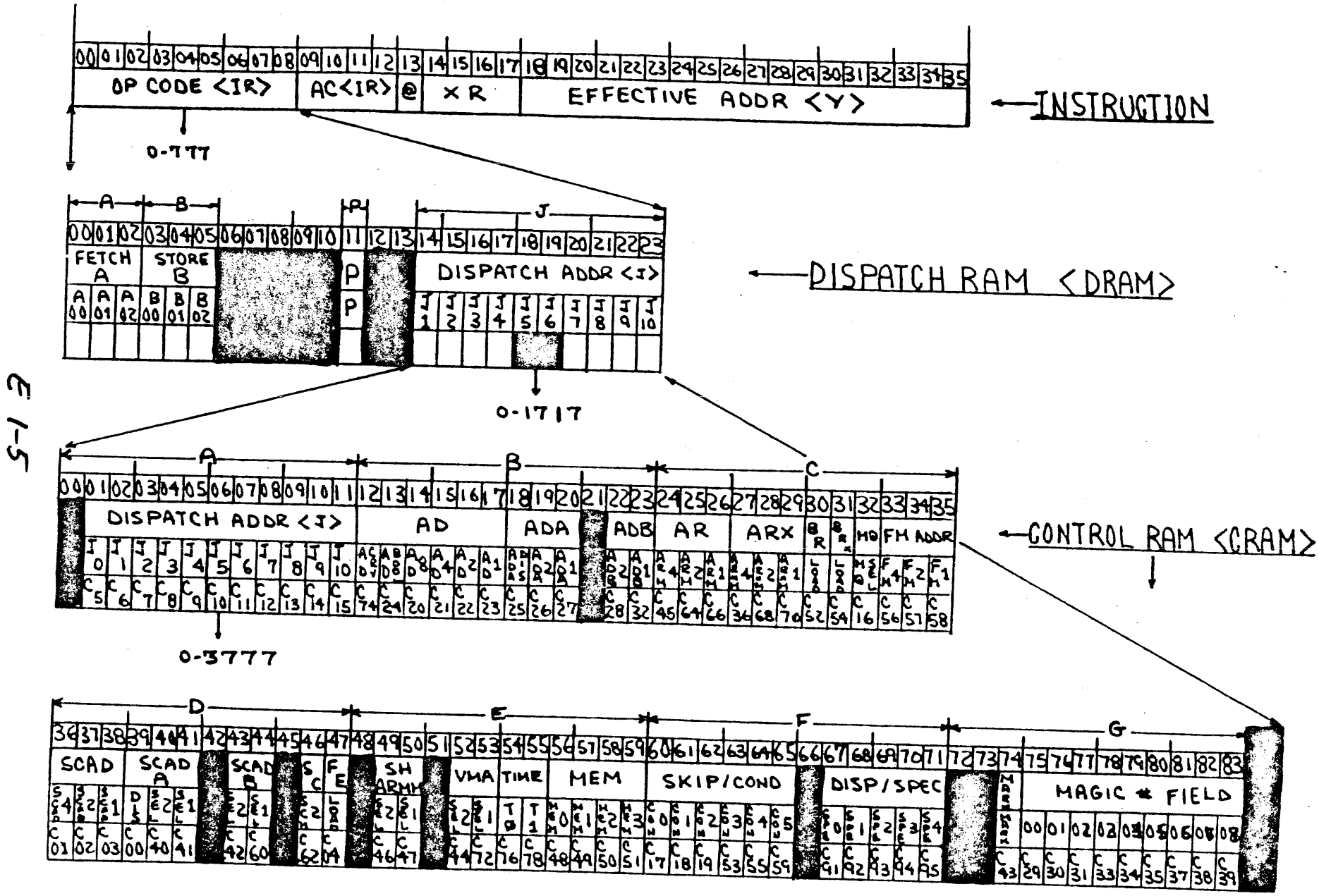
THEN(DR ADR 3:5=IR7:9)

ELSE(DRADR 3:5 = 7)
AND DRADR 6:8 = IR 1Ø:12]

ELSE [DR ADR Ø:8 = IRØ:8]

DRAM

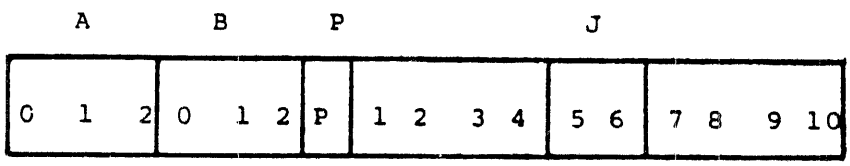



OP CODE <IR>
AC<IR>
@
XR
EFFECTIVE ADDR <Y>
INSTRUCTION
0-777
FETCH A
STORE B
P
P
DISPATCH ADDR <J>
DISPATCH RAM <DRAM>
0-1717
DISPATCH ADDR <J>
AD
ADA
ADB
AR
ARX
FM ADDR
CONTROL RAM <CRAM>
0-3777
SCAD
SCAD A
SCAD B
SH
VMA
TIME
MEM
SKIP/COND
DISP/SPEC
MAGIC # FIELD

DISPATCH RAM (DRAM) WORD FORMAT

E Box D-16

DRAM word found on M8522 module in Slot 45.



Bits 5 & 6 always zero

DRAM A
Bit Functions

0	Immediate
1	Immediate - Prefetch
2	Not Used
3	Write Test
4	Read
5	Read - Prefetch
6	Read - Write (separate cycles)
7	Read - Pause - Write

DRAM B
Bit Functions

B Store 0, 1, 2

1	Double AC
2	Double both
3	Self
5	AC
6	Memory
7	Both

B - 1 - 2 Flt Store

1	AC
2	Memory
3	Both

B0 Inverts SJC Tests

0 IF Cry 0 = 0, PC SKIP
1 IF Cry 0 = 1, PC SKIP

B/Skip/Jump/Comp

0	SJC L, E
1	SJC E
2	SJC L
3	SJC Never
4	SJC G
5	SJC NOT EQUAL
6	SJC G, E
7	SJC Always

Cannot use B & B-1-2 simultaneously

Can use B & B0 together

A FIELD

MICROCODE BASE ADDRESS (CRAM) 47

MICRO WORD POSITION	0	1	2	3	4	5	6	7	8	9	10	11
SIGNAL NAME	N.U.	J00	J01	J02	J03	J04	J05	J06	J07	J08	J09	J10
MODULE SLOT NUMBER	[redacted]	50	50	50	44	44	44	44	42	42	42	42
MODULE PIN NUMBER	[redacted]	DV2	CK2	BP2	ED2	DV2	CK2	BP2	ED2	DV2	CK2	BP2
CRAM PHYSICAL BIT		05	06	07	08	09	10	11	12	13	14	15

MODEL B SAME

EBOX/A-16

B FIELD MAJOR DATA PATH

	AD CONTROLS ALU FUNCTIONS						ADA/ADXA SELECTS AD A			ADB/ADXB SELECTS AD B		
MICROWORD POSITION	12	13	14	15	16	17	18	19	20	21	22	23
SIGNAL NAME	CARRY IN CRY	BOOL	SEL8	SEL4	SEL2	SEL1	D15	SEL2	SEL1	N.U.	SEL2	SEL1
MODULE SLOT NUMBER	42	50	52	52	52	52	50	50	50		44	42
MODULE PIN NUMBER	CF2	ES2	ES2	EP2	CJ2	BF1	EP2	CJ2	BF1		ES2	ES2
CRAM PHYSICAL BIT	74	24	20	21	22	23	25	26	27		28	32

ARITHMETIC FUNCTIONS

00	A + X CRY	01	A + AND CB
03	A*2	02	A + AND
06	A + B	44	OR + 1
11	A - B - 1		
17	A - 1	05	OR + AND CB
40	A + 1	07	A + OR
43	A*2 + 1	52	AND + OR CB
46	A + B + 1	53	A + OR CB
50	OR CB + 1		
51	A - B	15	AND CB - 1
54	X CRY - 1	16	AND - 1
		17	A - 1

BOOL BIT 01

BOOLEAN FUNCTIONS

20	SET CA	31	XOR
21	OR C (NAND)	32	B
22	OR CA	33	OR
23	1's	34	0's
24	AND C (NOR)	35	AND CB
25	SET CB	36	AND
26	EQV	37	A
27	OR CB		
30	AND CA		

CARRY FUNCTIONS

36	CRY 0 IF A & B ≠ 0
37	GEN CRY 0 IF A ≠ 0
60	GEN CRY 0 IF A = 1's
71	CRY 0 IF A GE B UNSIGNED

SEE KL MAINT GUIDE P. 2-?

0	AR
1	ARX
2	MQ
3	PC

BIT 18 is ADA/ADXA ENABLE

0	EN
1	0's

0	FM
1	BR*2
2	BR
3	AR*4

SEL 1 ON PREVENTS DEFAULT SELECTION OF F.M.

FBOX/A-17

C FIELD

Fm BLOCK

	AR/ARM			ARX/ARXM			BR	BRX	MQ	FM ADR		
MICROWORD POSITION	24	25	26	27	28	29	30	31	32	33	34	35
SIGNAL NAME	SEL4	SEL2	SEL1	SEL4	SEL2	SEL1	LOAD	LOAD	SEL	4	2	1
MODULE SLOT NUMBER	50	50	50	40	44	44	42	42	40	40	40	40
MODULE PIN NUMBER	FK2	DR2	CF2	ES2	DR2	CF2	FP2	AR2	ED2	FP2	FK2	AR2
CRAM PHYSICAL BIT	45	64	66	36	68	70	52	54	16	56	57	58

AR/ARM (24–26)

0 AR
0 ARMM IF SPEC 22
1 CACHE
2 AD
3 E BUS
4 SH
5 AD*2
6 ADX
7 AD*.25

ARX/ARXM (27–29)

0 ARX
1 CACHE
2 AD
3 MQ
4 SH
5 ADX*2
6 ADX
7 ADX*.25

BR (30): 0 BR; 1 AR

BRX (31): 0 BRX; 1 ARX

MQ (32): 0 MQ; 1 SH

IF SPEC/MQ SHIFT

0 MQ*2
1 MQ*.25

IF COND/REG CTL

0 MQ SEL
1 MQM SEL

FM ADR (33–35)

0 AC0
1 AC1
2 XR
3 VMA
4 AC2
5 AC3
6 AC4 *CURRENT BLOCK AC+4*
7 #B#AC5 *CURRENT BLOCK +5*

MODEL B
6 = AC##
7 = #B#

6 CURRENT BLOCK AC+
MODEL B
6 CURRENT BLOCK AC+ MAGIC #
7 BLOCK AND AC SELECTED BY # FIELD

EBOX/A-17

E 1-9

D FIELD MINOR DATA PATH

	SCAD			SCADA				SCADB			SC	FE
MICROWORD POSITION	36	37	38	39	40	41	42	43	44	45	46	47
SIGNAL NAME	4	2	1	DIS1	SEL2	SEL1	N.U.	SEL2	SEL1	N.U.	SEL2	LOAD
MODULE SLOT NUMBER	52	52	52	52	52	52		52	52		52	50
MODULE PIN NUMBER	DV2	CK2	BP2	ED2	FP2	FK2		AR2	DR2		CF2	ED2
CRAM PHYSICAL BIT	01	02	03	00	40	41		42	60		62	04

0 A
1 A - B -1
2 A + B
3 A - 1
4 A + 1
5 A - B
6 OR
7 AND

0 FE
1 AR 0 - 5
2 AR EXP
3 #

BIT 39 = 1
ENABLES 0's

1 BYTE POINTER P FIELD

2 <AR 01-08> XOR <AR 00>

3 SIGN EXTENDED WITH #00

0 SC
1 AR6-11
2 AR0-8
3 #

1 BYTE POINTER S FIELD

0 = SC

1 = SCAD

WITH SPEC 13,
0 = FE

1 = AR SHIFT

0 = FE

1 = SCAD

MODEL B SAME

E BOX/A-18

E FIELD

	SH/ARMM				VMA		TIME		MEMORY			
MICROWORD POSITION	48	49	50	51	52	53	54	55	56	57	58	59
SIGNAL NAME	N.U.	SEL2	SEL1	N.U.	SEL2	SEL1	T00	T01	00	01	02	03
MODULE SLOT NUMBER		50	50		50	42	40	40	44	44	44	44
MODULE PIN NUMBER		AR2	AV2		FP2	DR2	DR2	CF2	FP2	FK2	AR2	AV2
CRAM PHYSICAL BIT		46	47		44	72	76	78	48	49	50	51

SH FIELD

0 SHIFT AR,ARX
1 AR
2 ARX
3 AR SWAP

ARMM FIELD

0 AR0-8←#
1 AR1-8←AR0
EXP←SCAD
2 AR0-8←SCAD
EXP←SCAD
3 AR0-5←SCAD
POS←SCAD

MODEL B
VMAX FIELD
0 - VMAX
1 PC SEC
2 AREV SEC
3 AD12-17

0 VMA
1 PC
2 PC + 1
3 AD
0 - NOOP
1 - PC → VMA
2 - VMA ADR
3 - AD → VMA

0 2T
1 3T
2 4T
3 5T

0 NO OP
1 ARL IND
2 MB WAIT
3 SEC 0
4 A READ
5 B WRITE
6 FETCH
7 REG FUNCTION

IF X ADDR

10 AD FUNCTION
11 EA CALCULATION
12 LOAD AR
13 LOAD ARX
14 READ-WRITE
15 READ-PAUSE-WRITE
16 WRITE
17 IFET

OLD STYLE MCL BOX

IF NOT X ADDR

10 A IND
11 BYTE IND
12 LOAD AR
13 LOAD ARX
14 AD FUNCTION
15 BYTE READ
16 WRITE
17 READ-PAUSE-WRITE

MODEL B
3 - RESTORE VMA
10 - AD FUNCTION
11 - EA CALC
14 - RW
15 - READ-PAUSE-WAIT
16 - WRITE
17 - UNCOND. FETCH

14 - READ TEST WRITABILITY

EBOX/A-18

F FIELD

COND MODEL B SPEC

MICROWORD POSTIION	60	61	62	63	64	65	66	67	68	69	70	71
SIGNAL NAME	00	01	02	03	04	05	N.U.	00	01	02	03	04
MODULE SLOT NUMBER	40	40	40	42	42	40		45	45	45	45	45
MODULE PIN NUMBER	DV2	CK2	BP2	FK2	AV2	AV2		DM2	DT2	EF1	EM2	ES2
CRAM PHYSICAL BIT	17	18	19	53	55	59		91	92	93	94	95

66 = CALL FUNCTION

NON-SKIP FUNCT.

0	NO OP
1	LD AR 0-8
2	LD AR 9-17
3	LD AR 18-35
4	AR CLR
5	ARX CLR
6	ARL IND SEE G FIELD
7	REG CTL
10	FM WRITE
11	PCF ← #
12	FE SHRT
13	AD FLAGS
14	LOAD IR
15	SPEC INSTR
16	SR ← #
17	SEL VMA
20	DIAG FUNC
21	EBOX STATE
22	EBUS CTL
23	MBOX CTL
24	SPARE
25	SPARE
26	SPARE
27	SPARE
30	VMA ← #
31	VMA ← #+TRAP
32	VMA ← #+MODE
33	VMA ← #+AR32-35
34	VMA ← #+PI*2
35	VMA DEC
36	VMA INC
37	LD VMA HELD

MODEL B
57 - LOCAL AC ADDR
50 PC SEL 0
75 VMA SEL 0
25 LONG EN

SKIP FUNCT.

40	SPARE
41	EVEN PAR
42	BR 0
43	ARX 0
44	AR 18
45	AR 0
46	AC # 0
47	SC 0
50	SC LT 36
51	SCAD 0
52	SCAD ≠ 0
53	ADX 0
54	AD CRY 0
55	AD 0
56	AD ≠ 0
57	SPARE
60	FETCH
61	KERNEL
62	USER
63	PUBLIC
64	RPW REF
65	PI CYCLE
66	-EBUS GRANT
67	-EBUS XFER
70	INTRPT
71	-START
72	RUN
73	I/O LEGAL
74	P!S XCT
75	EBOX PF
76	AC REF
77	-MTR REQ

DISPATCHES
MODIFY CRAM BASE ADDRESS (A-FIELD)

MODEL B
20 STACK UPDATE

0	DIAG
1	DRAM J
2	DRAM A READ
3	RETURN
4	PG FAIL
5	SR
6	NICOND
7	SH 0-3
30	MUL
31	DIV
32	SIGNS
33	DRAM B
34	BYTE
35	NORM
36	EA MOD
37	EA TYPE

NON-DISPATCHES

10	NO OP
11	INH CRY 18
12	MQ SHIFT
13	SCM ALT
14	CLR FPD
15	LD PC
16	XCRY AR 0
17	GEN CRY 18
20	SEC HOLD
21	CALL
22	ARL IND SEE G FIELD
23	MTR CTL
24	FLAG CTL
25	SAVE FLAGS
26	SP MEM CYCLE
27	AD LONG

MODEL B UPDATE
IF XADR: 20 = STACK POINTER
21 = SPARE

MAGIC NUMBER FIELD

MACROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CJ2	BF1	EP2	CJ2	BF1	EP2	CJ2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

MODEL B

ENABLES REQUIRED

KL PAGE: 1=CALL

LONG PC: ARØ-8, 1-LD

USED TO ADDRESS FAST MEMORY

LOAD

CLR

1=ARR 11=ARR+MQ
2=ARL 13=AR+MQ
3=AR 14=ARX+MQ
4=ARX 16=ARL+ARX+MQ
6=ARL+ARX
7=AR+ARX
10=MQ 17=AR+ARX+MQ

ARL

0=ARL
0=ARMM IF BIT 76=
1=CACHE
2=AD
3=EBUS
4=SH
5=AD*2
6=ADX
7=AD*.25

F-FIELD
IF COND DECODE = 6
THEN ARL IND SEL
F FIELD
COND 6
SPEC 22
E FIELD
MEMORY 1

IF COND DECODE = 7
THEN COND/REG CTL

AR CTL

1=ARR LOAD
2=AR9-17 LOAD
4=ARØ-8 LOAD
6=ARL LOAD

EXPTST

1=
AR -EXP

IF COND DECODE = 7
THEN COND/REG CTL AND MQ/MQ SEL
IF CFIELD BIT 32 = 0

THEN

MQ CTL

0=MQ
1=MQ*2
2=MQ*.5
3=0'S

IF CFIELD BIT 32 = 1
COND/REG CTL AND MQ/MQM SEL

THEN

0=SH
1=MQ*.25
2=IS
3=AD

G FIELD #2

MAGIC NUMBER FIELD

MICROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CJ2	BF1	EP2	CJ2	BF1	EP2	CJ2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

F-FIELD - IF COND/PCF-#
DECODE 11

PC FLAGS	
20=TRAP 1	424 DIV CHK
40=TRAP 2	620 FLOV
100=FPD	624 FDV CHK
420=AROV	630 FXU

F-FIELD - IF SPEC/FLG CTL
FOR JRST
DECODE 24

FLAG CTL	
20=SET FLAGS	502 PI DISMISS
412=PORTAL	602 JFCL
420=RSTR FLAGS	622 JFCL+LD FLAGS
442=HALT	

F-FIELD - IF COND/SPEC INSTR
DECODE 15

SPEC INSTR	
4=INSTR ABORT	100=INH PC+1
10=INTRPT INH	200=KERNEL CYCLE
10=CONT	302=HALT RUN FLOP
20=PXCT	310=CONS XCT
40=SXCT	704=SET PI CYCLE

E-FIELD - IF MEM/FETCH
DECODE 6

FETCH	
201=COMP	400=UNCONO
202=SKIP	502=JUMP
203=TEST	503=JFCL

F-FIELD - IF SPEC/SP MEM CYCLE
DECODE 26

SP MEM	
1=PAGING INH	
2=CACHE INH	111=EPT
10=EPT EN	200=USER
20=UPT EN	221=UPT
31=PT	400=FETCH
40=SECØ	431=PT FETCH
100=EXEC	511=EPT FETCH
101=UNPAGED EXEC	621=UPT FETCH

E-FIELD - IF MEM/REG FUNC
DECODE 7

MREG FUNC	
407=SBUS DIAG	505=WR REFILL FAM
140=MAP	601=LOAD CCA
502=READ UBR	602=LOAD UBR
503=READ EBR	603=LOAD EBR
504=READ ERA	MODEL B
	606=LOAD CCA

F Box/1-21 E 1-14

MAGIC NUMBER FIELD

MICROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CJ2	BF1	EP2	CJ2	BF1	EP2	CJ2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

F-FIELD - IF COND/MBOX CTL
DECODE 23

MBOX CTL

00 NORMAL	21 CLR PT LINE
01 PT DIR CLR	100 SET I/O PF ERR
10 PT WR	200 SET PAGE FAIL
20 PT DIR WR	

F-FIELD - IF COND/EBUS CTL
DECODE 22

EBUS CTL

0=REL EEBUS	26 DATA O
1=INPUT	27 DATA I
2=DATA I/O	30 I/O INIT
4=DISABLE CS	60 EBUS DEMAND
10=CTL ← IR	100 REL EBUS
20=EBUS NO DEMAND	400 GRAB EBUS
	200 REQ EBUS

F-FIELD - IF COND/DIAG FUNC
DECODE 20

DIAG FUNC

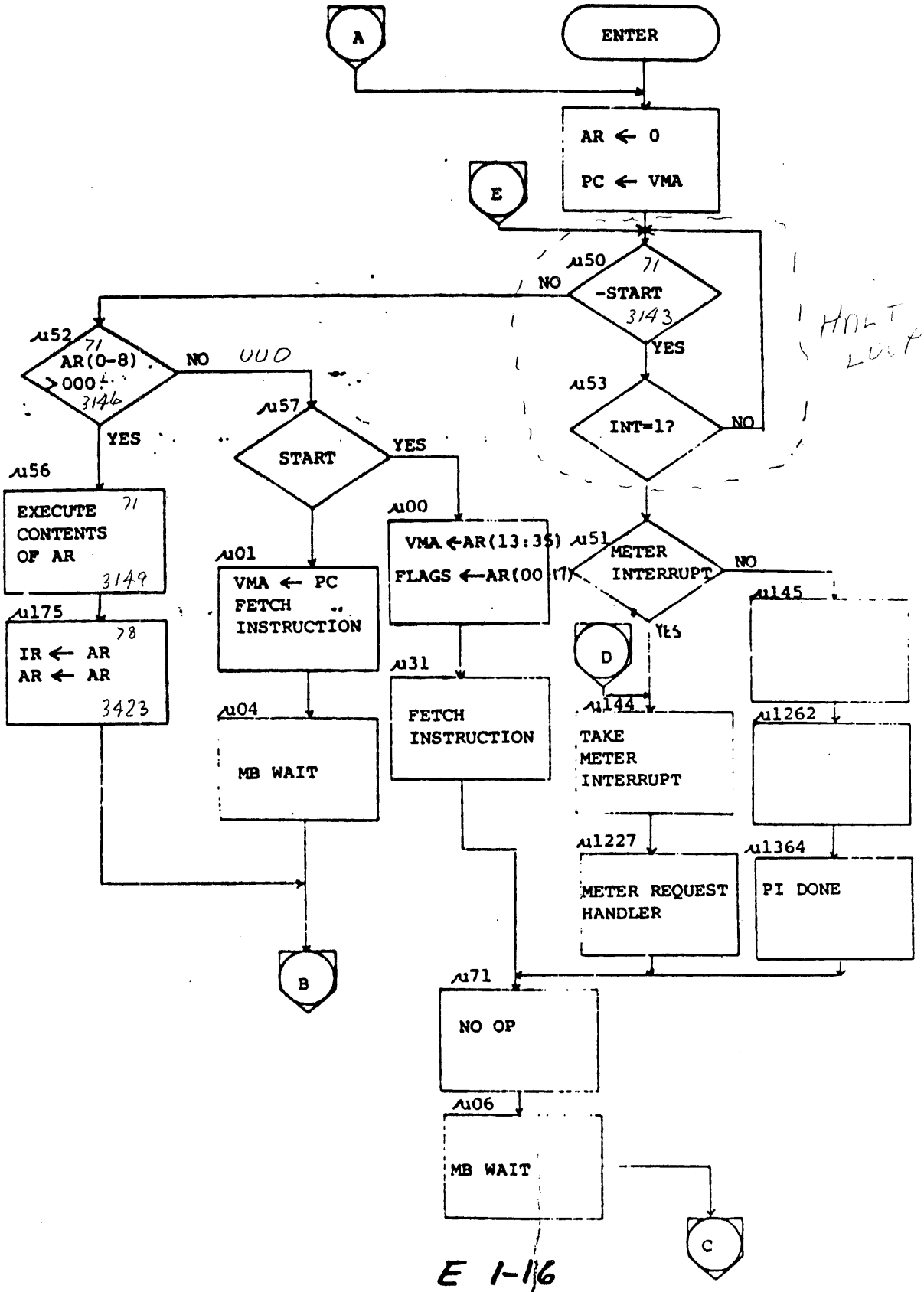
400 .5 uSEC	511 DATAI PAG (L)
404 LD PA LEFT	511 RD PERF CNT
405 LD PA RIGHT	512 CONI APR (L)
406 CONO MTR	512 RD EBOX CNT
407 CONO TIM	513 DATAI APR
414 CONO APR	513 RD CACHE CNT
415 CONO PI	514 RD INTRVL
416 CONO PAG	515 RD PERIOD
417 DATAO APR	516 CONI MTR
425 LD AC BLKS	517 RD MTR REQ
426 LD PCS+CWSX	530 CONI PI (PAR)
500 CONI PI (R)	531 CONI PAG
501 CONI PI (L)	567 RD EBUS REG
510 CONI APR (R)	620 DATAO PAG
510 RD TIME	

F-FIELD - IF SPEC/MTR CTL
DECODE 23

METER FUNCTIONS

0 CLR TIM	4 LD PA LH
1 CLR PERF	5 LD PA RH
2 CLR E CNT	6 CONO MTR
3 CLR M CNT	7 CONO TIM

MICROPROGRAM FLOWCHART REVISION 126
(GENERALIZED)



E 1-16

G FIELD #4

MAGIC NUMBER FIELD

MICROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CU2	BF1	EP2	CU2	BF1	EP2	CU2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

MODEL B ONLY

6 = AC#
3 2 = JUST AC#
3 3 = AC <OR> AC#

MEM/EA CALC

EA CALC

400 = LOAD AR
200 = LOAD ARX
100 = PAUSE
040 = WRITE
620 = BYTE RD
610 = BYTE IND
041 = PUSH
421 = POP AR
440 = LD AR+WR
240 = LD ARX+WR
20 = PREV EA
10 = INDIRECT
2 = EA
1 = STACK
221 = POP ARX
621 = POP AR-ARX
42 = WRITE (E)
402 = LD AR (EA)

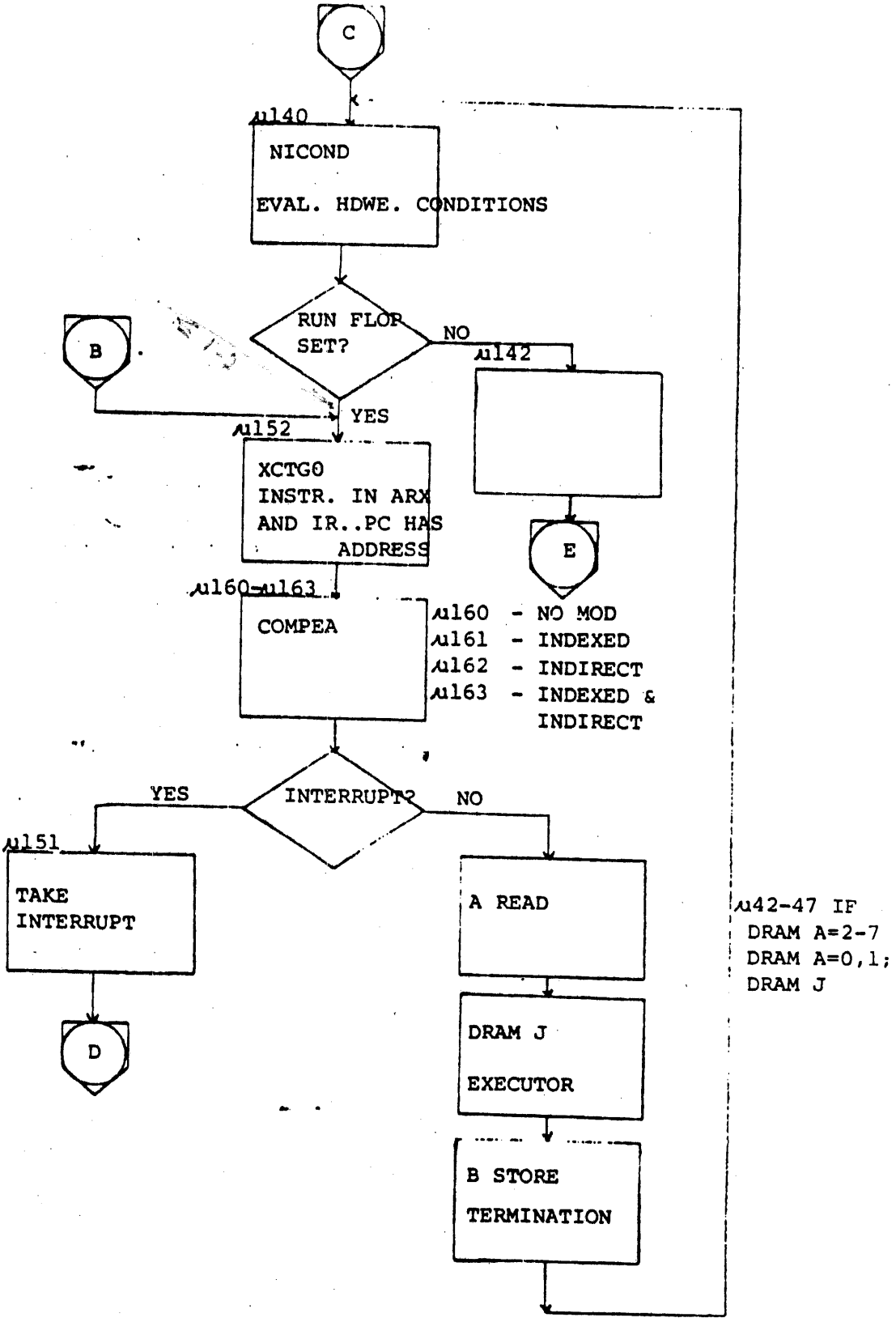
IF XADDR (ARX)

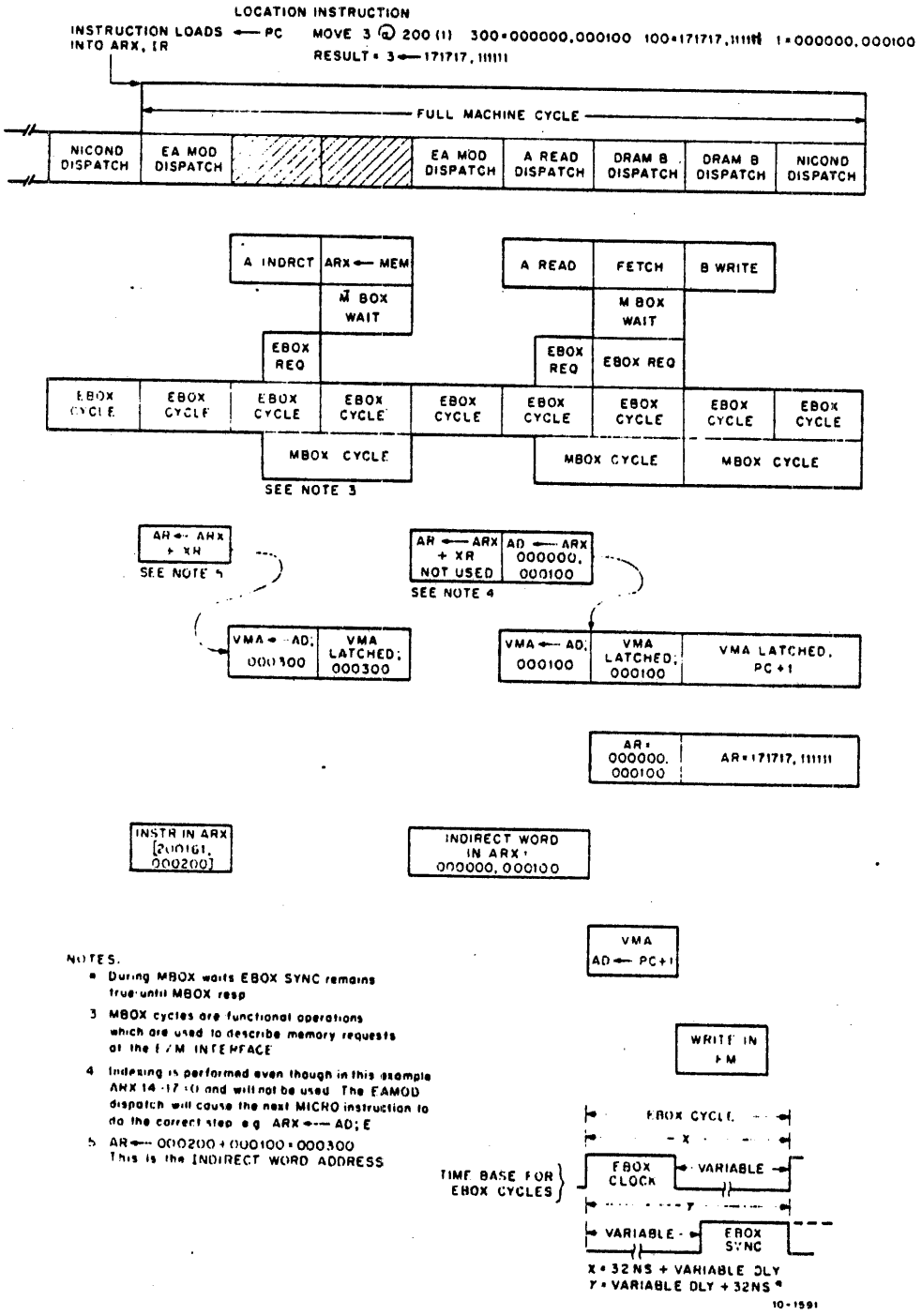
230 = A IND

IF NOT XADDR

630 = A IND

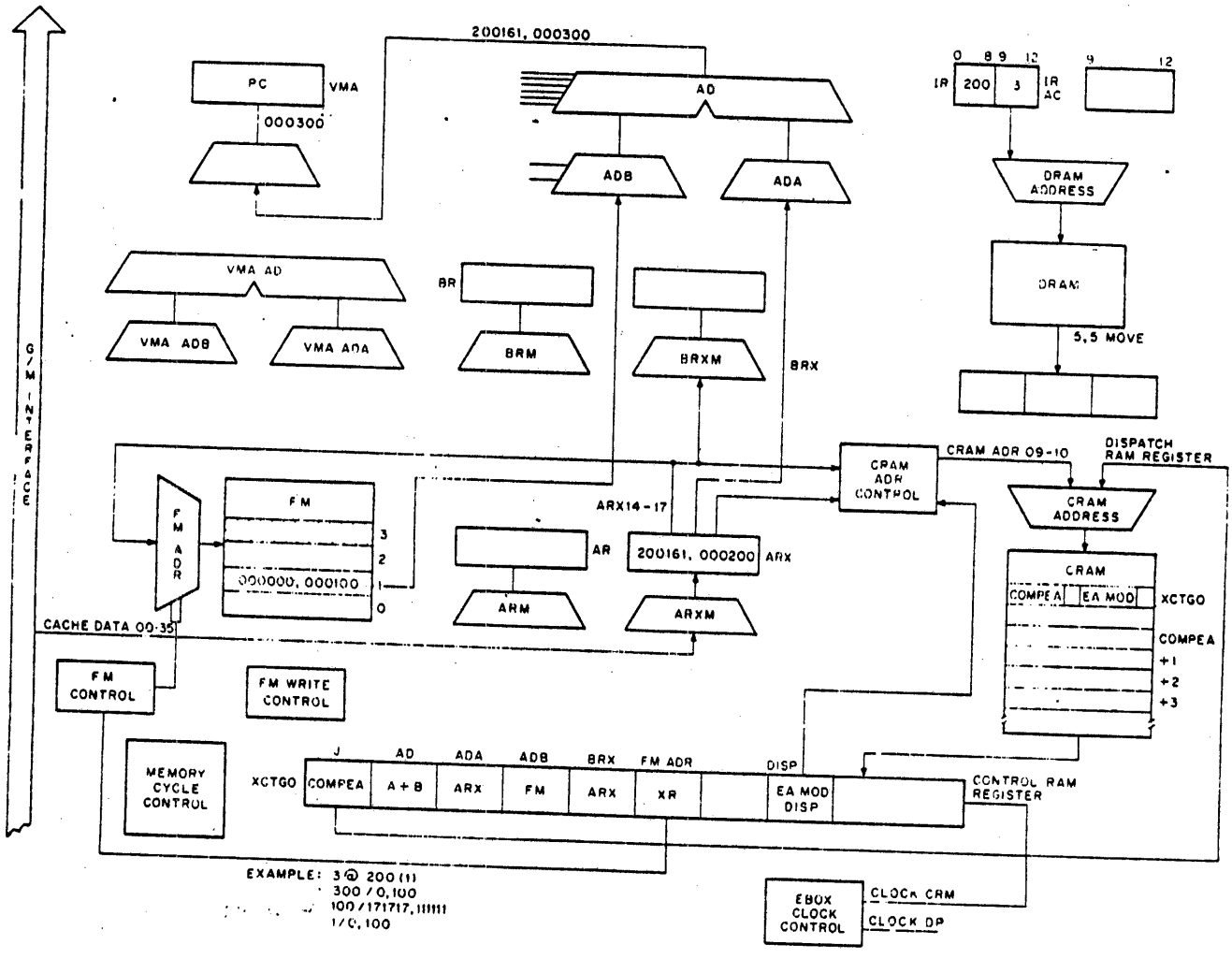
INDIRECT AT 1ST. EA CALC TIME TO BOTH AR AND ARX AS IN MODEL A


C
μ140
NICOND
EVAL. HDWE. CONDITIONS
RUN FLOP SET?
NO
YES
B
μ142
E
μ152
XCTG0
INSTR. IN ARX AND IR..PC HAS ADDRESS
μ160-μ163
COMPEA
μ160 - NO MOD
μ161 - INDEXED
μ162 - INDIRECT
μ163 - INDEXED & INDIRECT
INTERRUPT?
YES
NO
μ151
TAKE INTERRUPT
D
A READ
DRAM J
EXECUTOR
B STORE
TERMINATION
μ42-47 IF
DRAM A=2-7
DRAM A=0,1;
DRAM J



Basic Machine Cycle Overview (Sheet 1)

EBOX

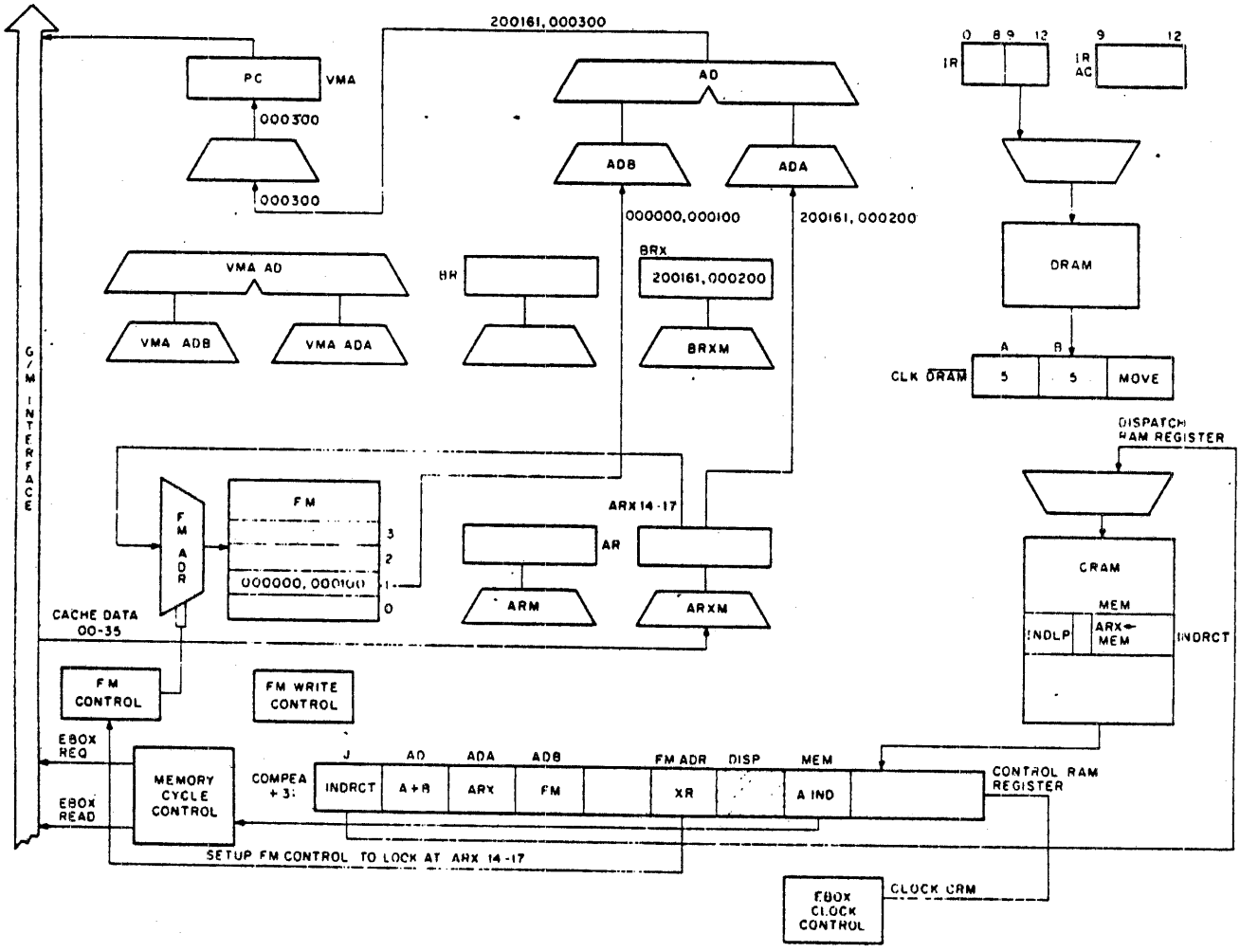


U152 = 0160, 0001, 0020, 0000, 0000, 0036, 0000

con

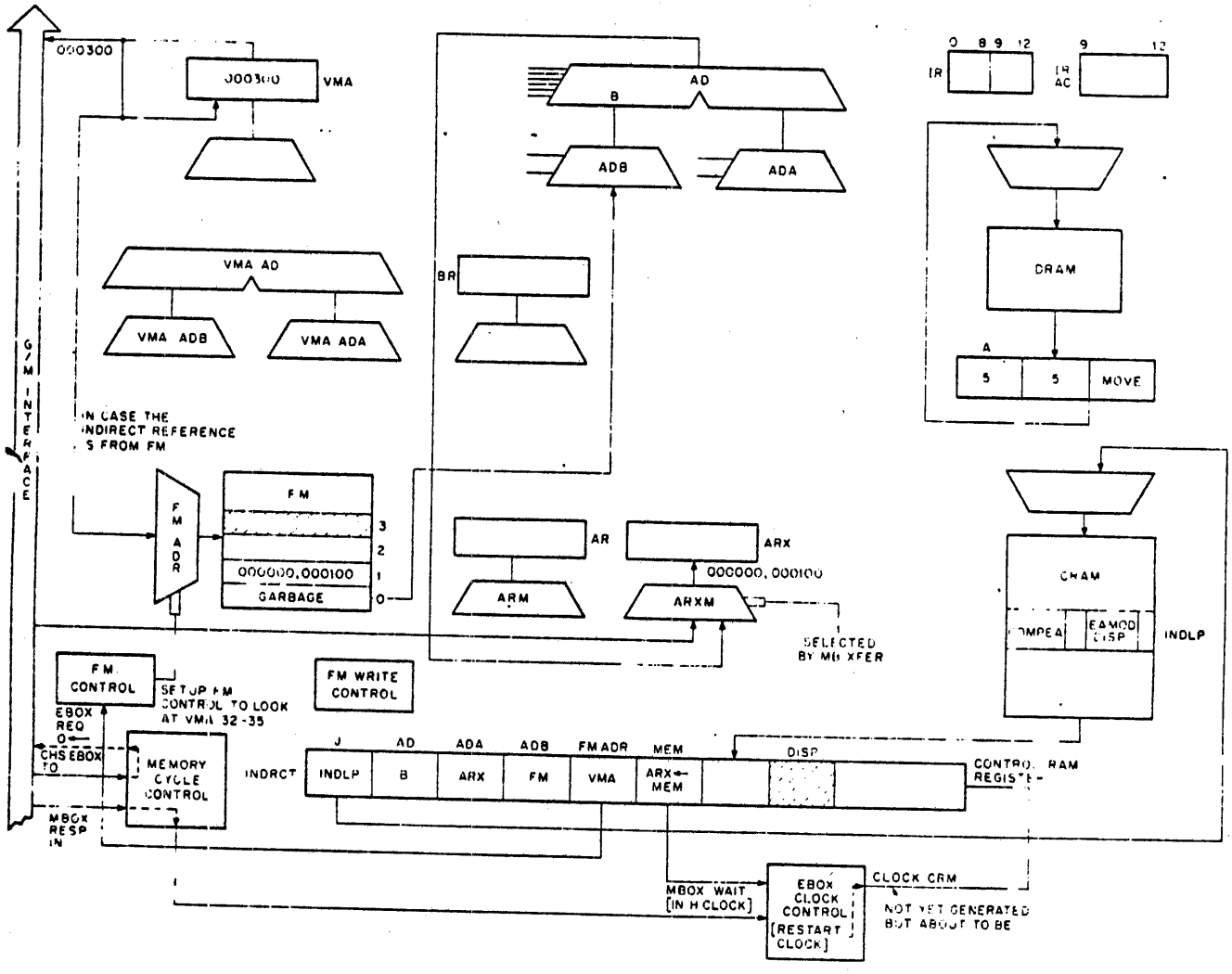
Instruction Cycle: NICOND DISPATCH → XCTGO

E BOX 2-27



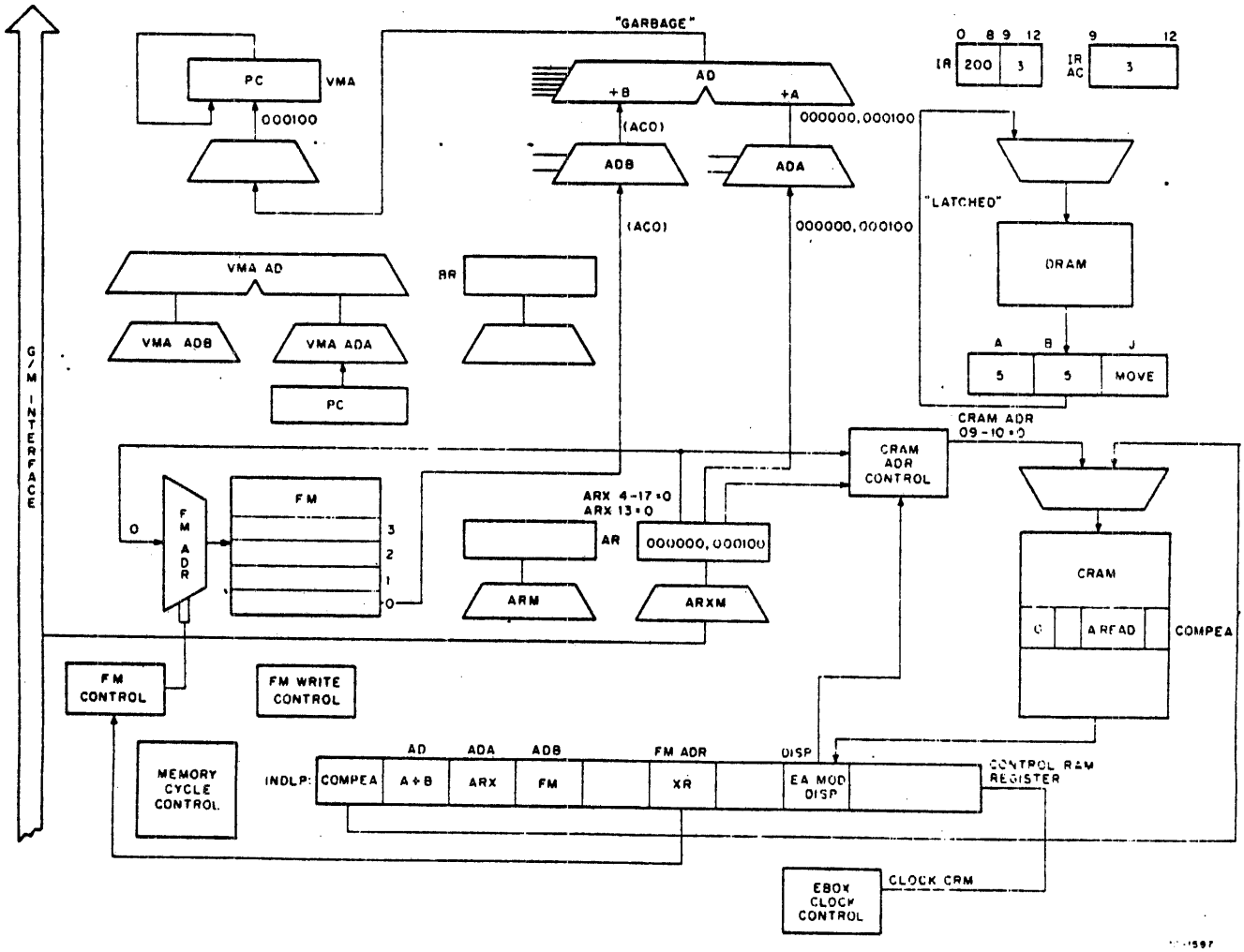
Set Up and Make Indirect Word Request

E 1-20



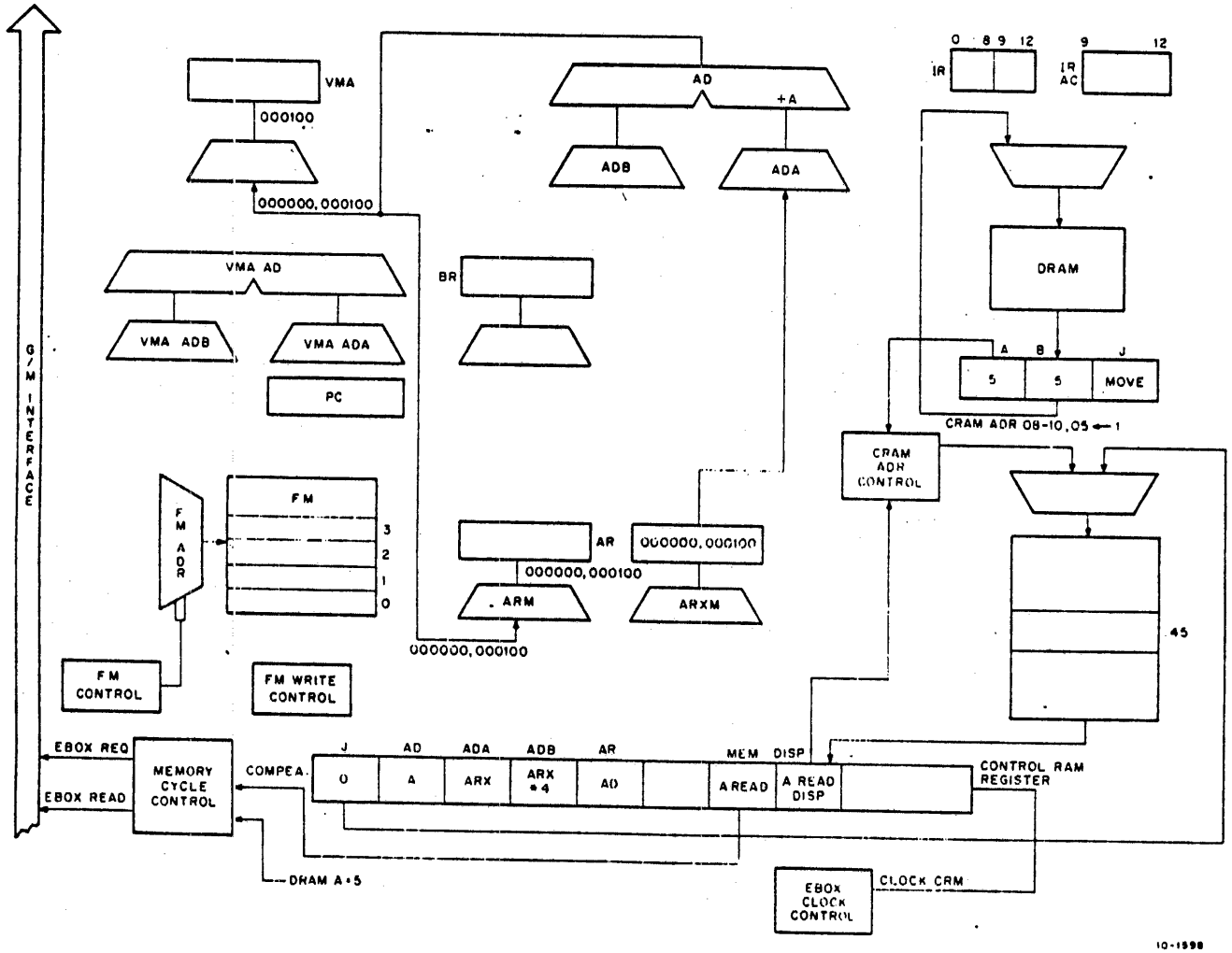
MBox Responds To Indirect Request

E-Box 2.39



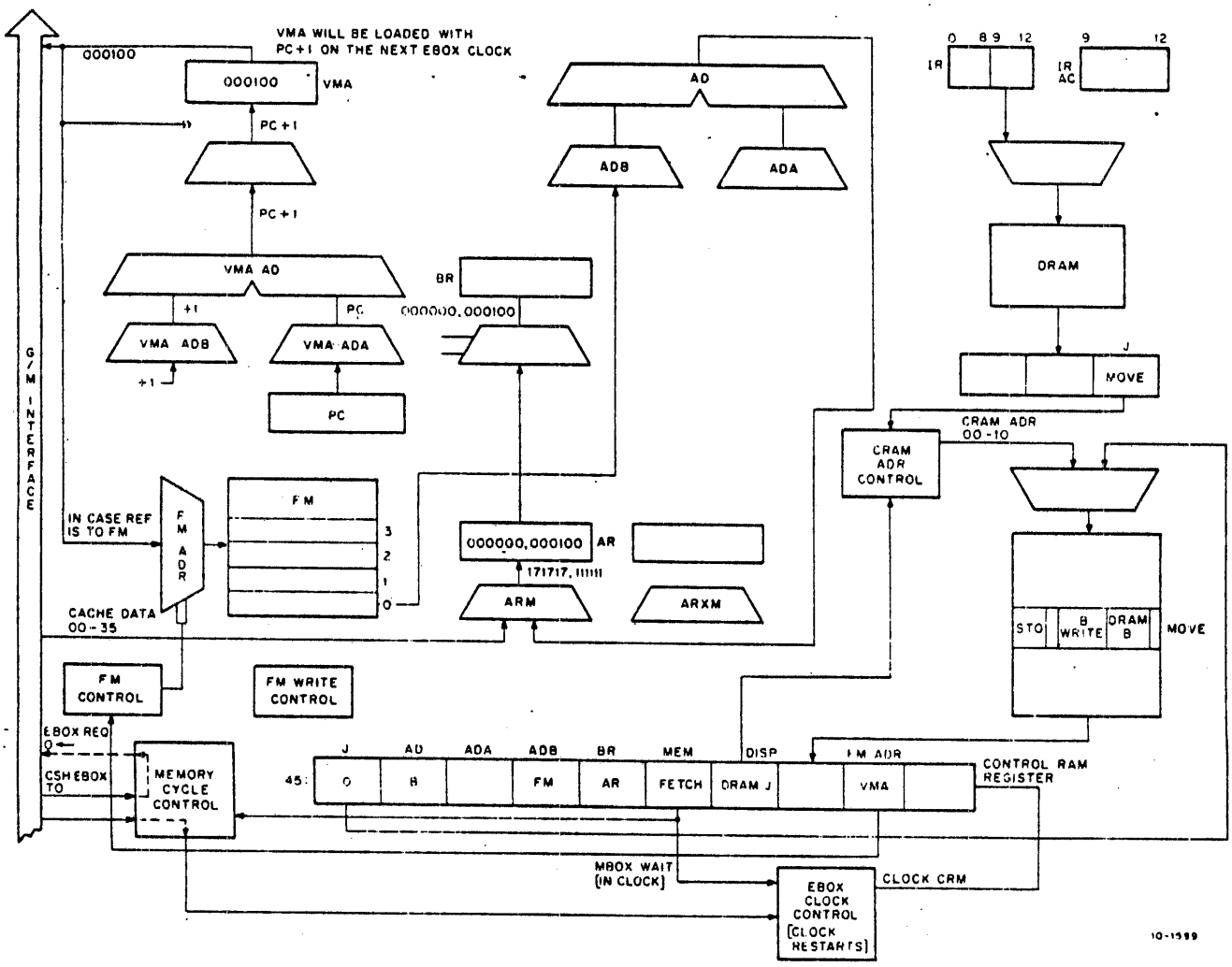
Address Calculation Continues

Rev 2.411



A READ Dispatch Set Up Data Fetch

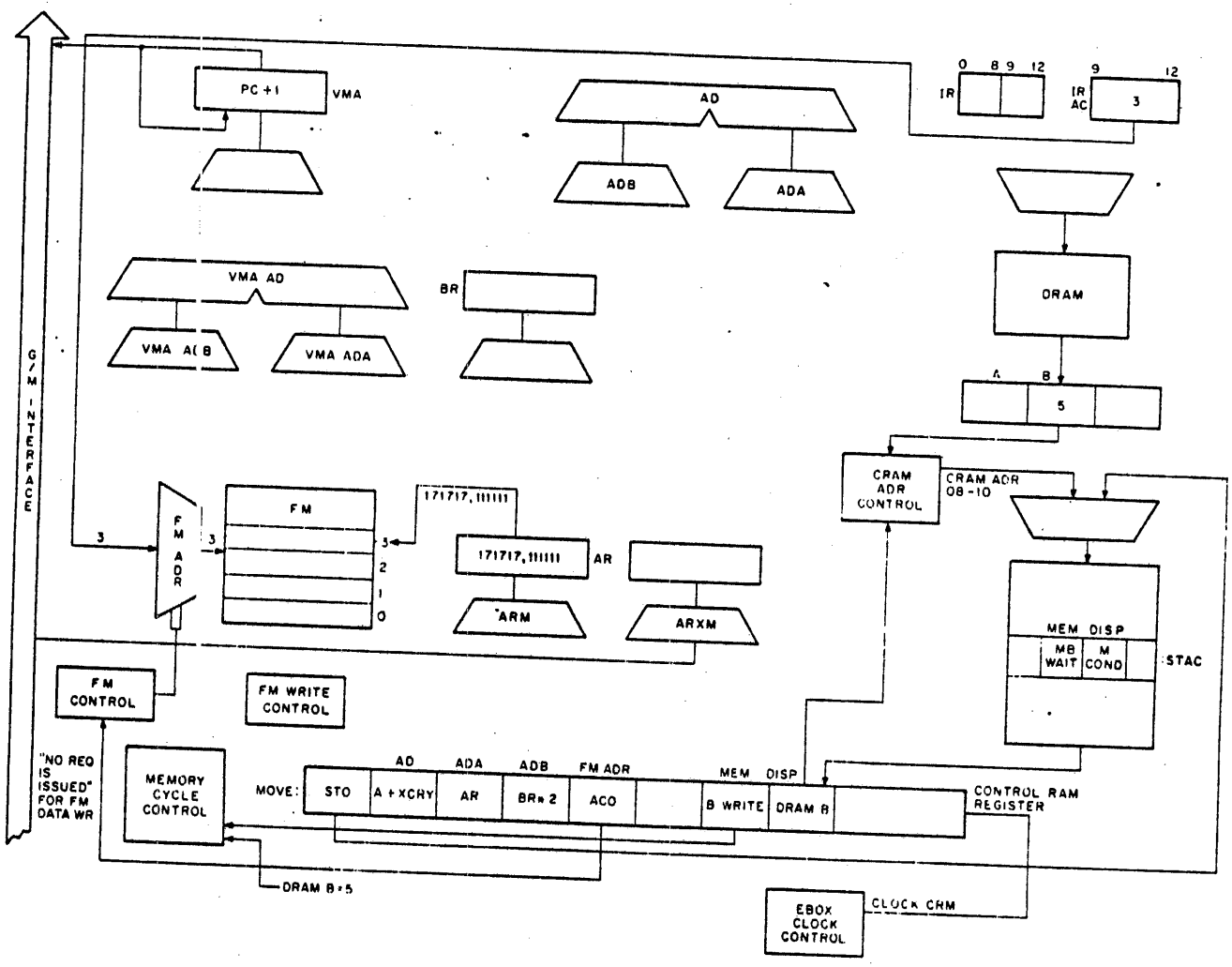
6 Aug 2-13



MBox Responds With Data Word Requests

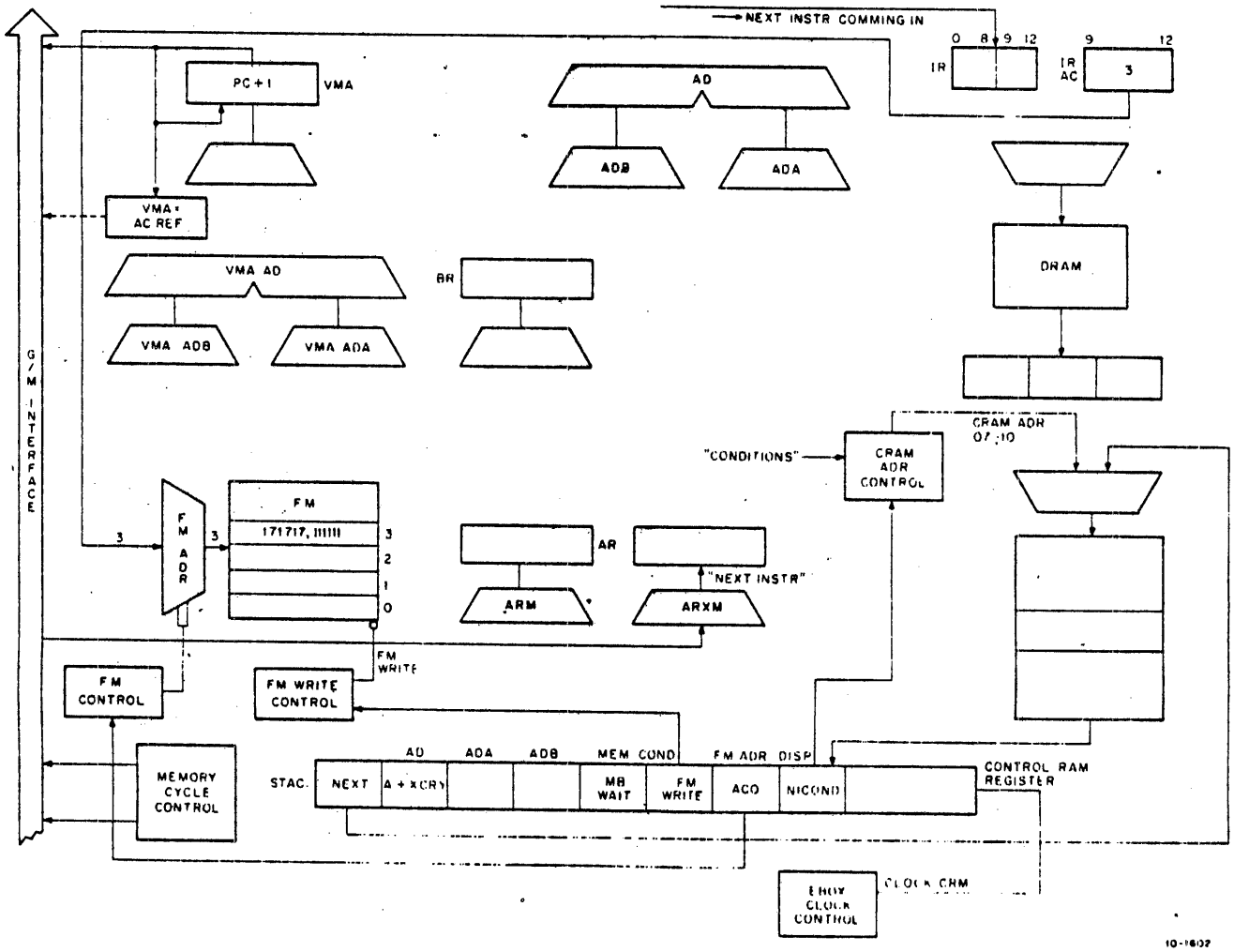
EBOX-2.47

E 1-24



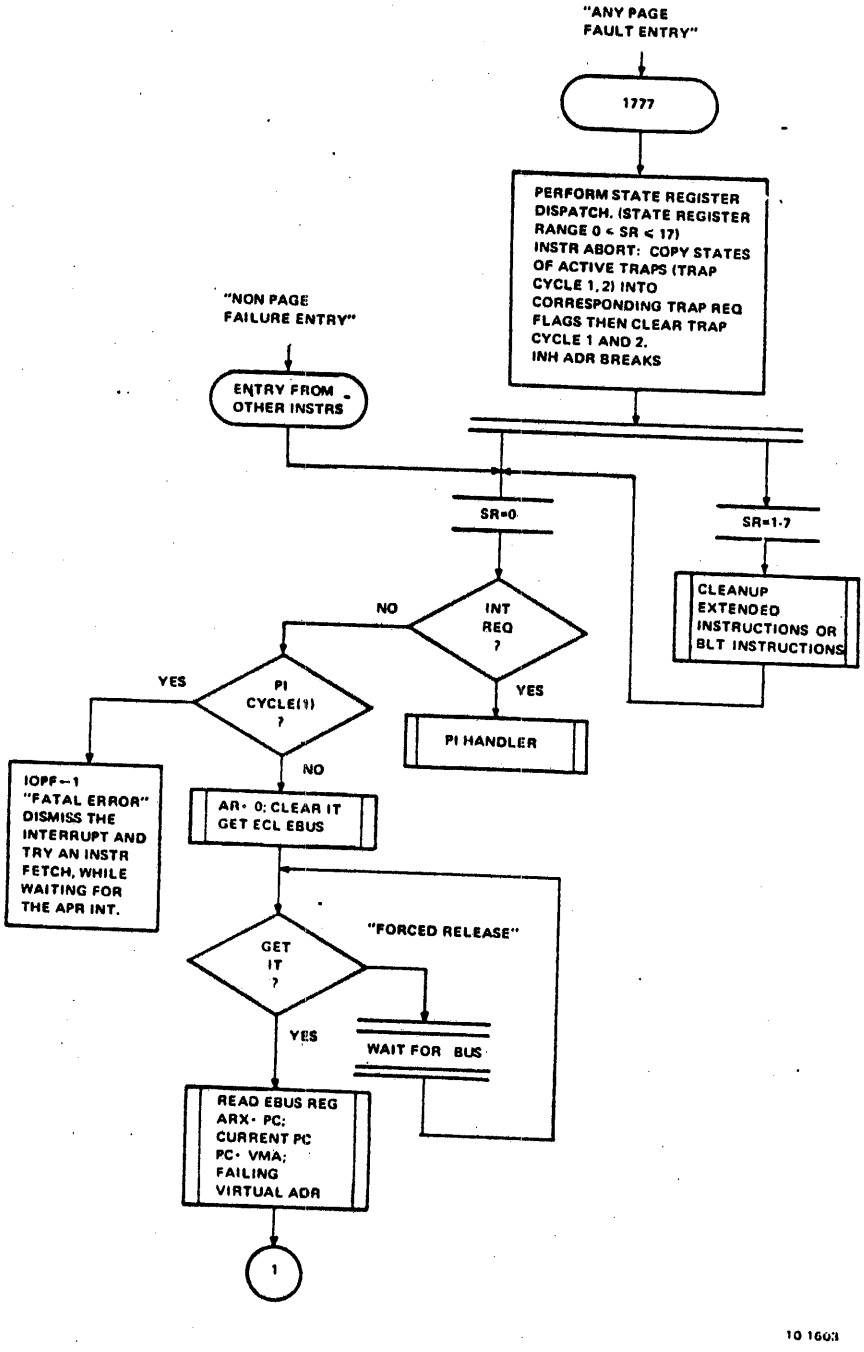
Executor Set Up For Store Cycle

5 Nov 2-51

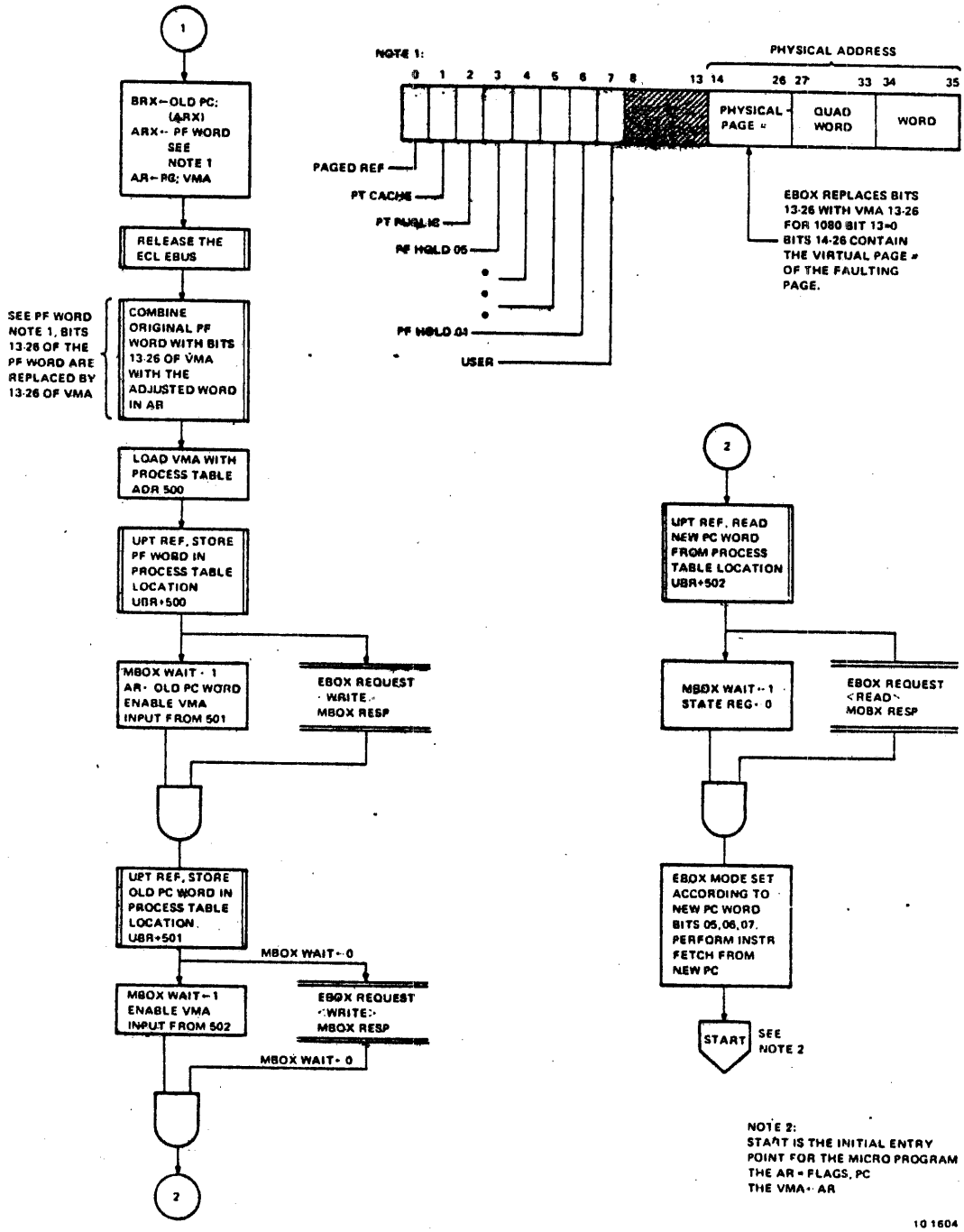


Finish Store Cycle, Perform NICOND Dispatch

E Box 2-53



Page Fail Handling (Sheet 1 of 2)

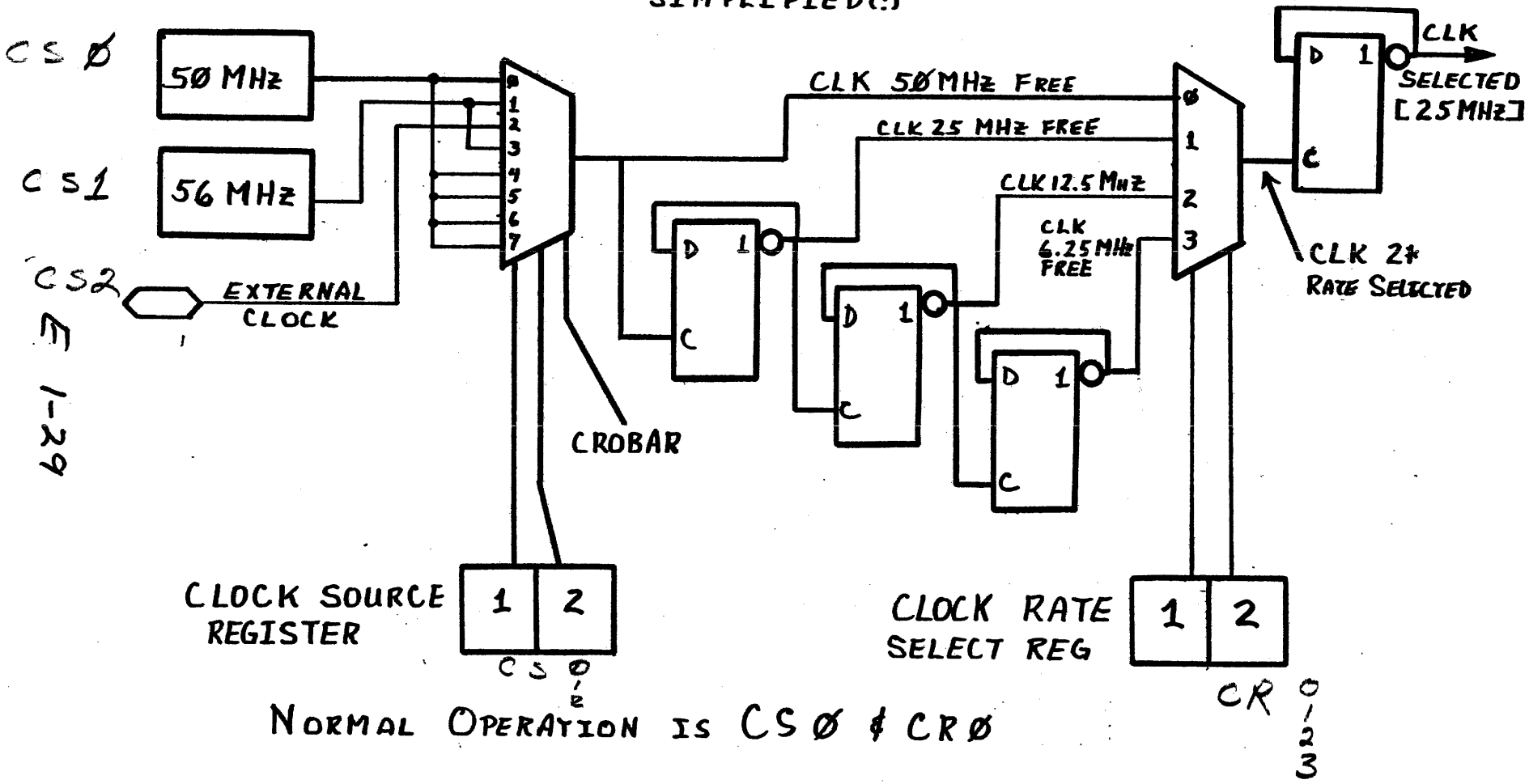


Page Fail Handling (Sheet 2 of 2)

KLDCP -3

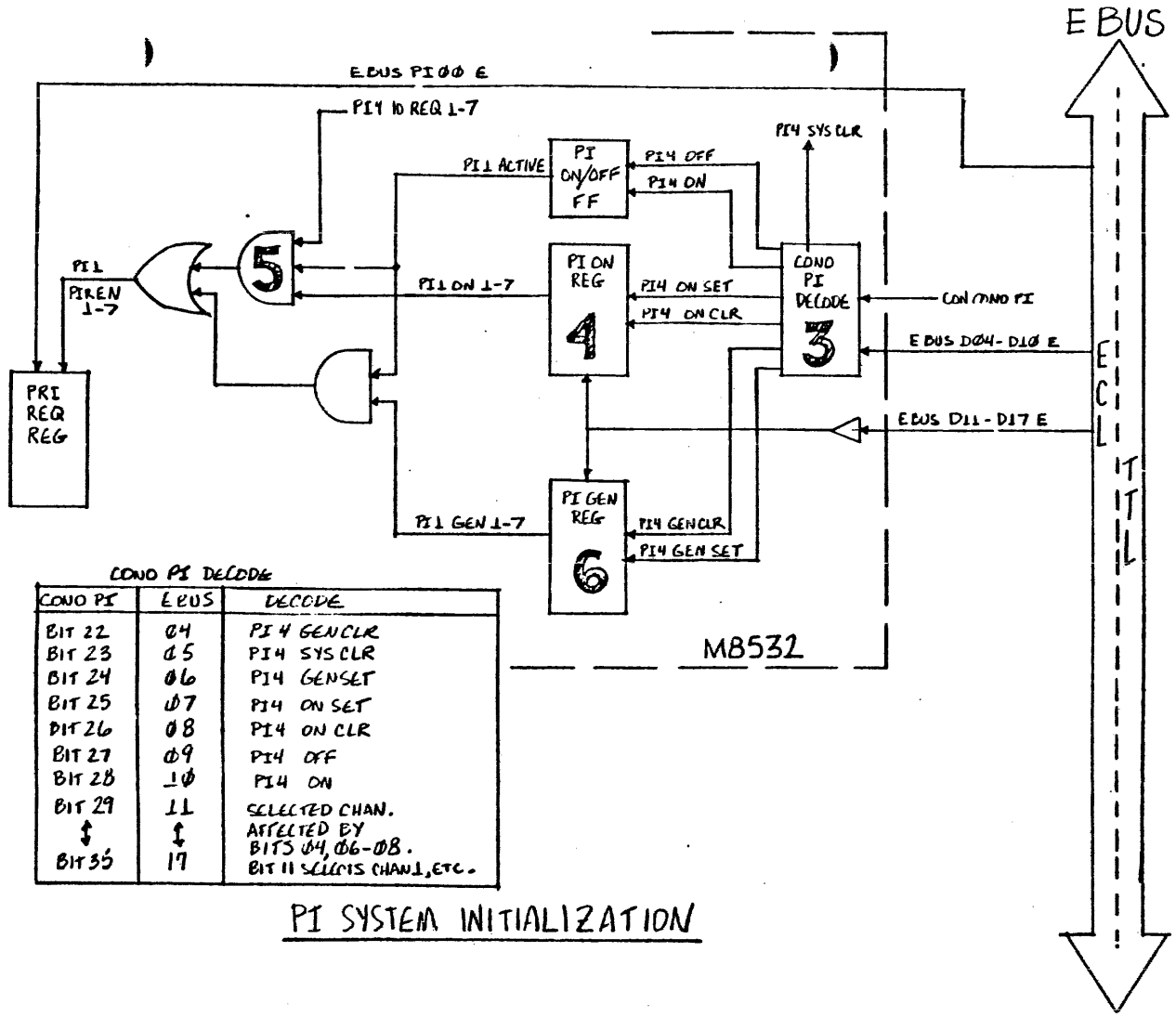
CLOCK CONTROL

SIMPLIFIED(?)



NORMAL OPERATION IS CS Ø & CR Ø

1-29



COND PI DECODE

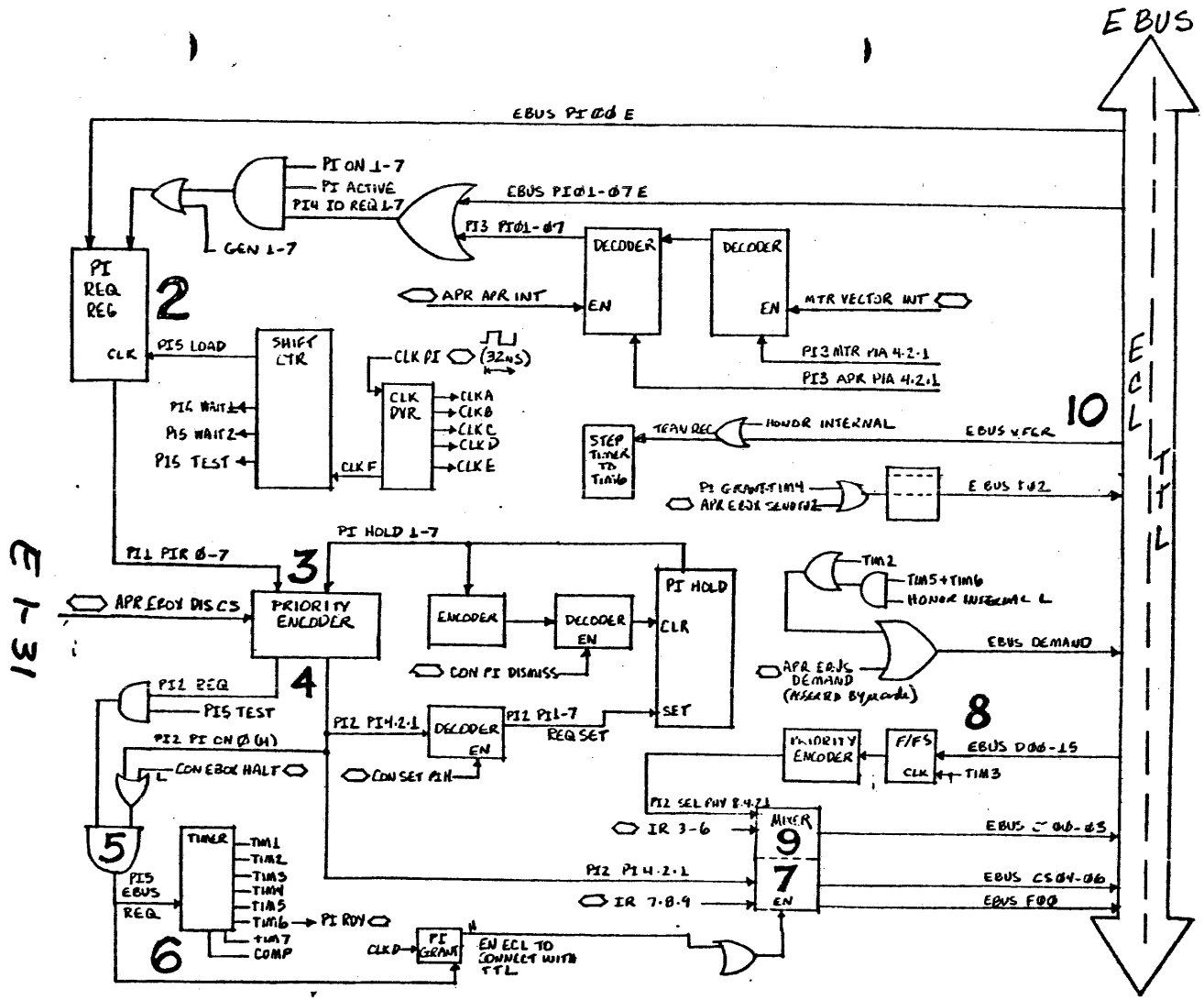
COND PI	EBUS	DECODE
BIT 22	Ø4	PI 4 GEN CLR
BIT 23	Ø5	PI4 SYS CLR
BIT 24	Ø6	PI4 GEN SET
BIT 25	Ø7	PI4 ON SET
BIT 26	Ø8	PI4 ON CLR
BIT 27	Ø9	PI4 OFF
BIT 28	1Ø	PI4 ON
BIT 29	11	SELECTED CHAN. AFFECTED BY BITS Ø4, Ø6-Ø8. BIT 11 SELECTS CHAN 1, ETC.
↕	↕	
BIT 35	17	

PI SYSTEM INITIALIZATION

PI SYSTEM INITIALIZATION

1. CPU CONTROLS THE PI SYSTEM THROUGH THE [redacted]
2. THE IMMEDIATE PORTION OF THE COND PI (18-35) IS PLACED ON THE EBUS (ØØ-17).
3. WHEN THE PI MODULE RECEIVES CON COND PI FROM THE EBOX IT WILL DECODE EBUS Ø4-1Ø AS SHOWN IN THE COND PI DECODE TABLE.
4. THE PI ON REG WILL SPECIFY WHICH CHANNELS (1-7) WILL BE ALLOWED TO INTERRUPT THE SYSTEM. PI ON SET WILL LOAD THE SELECTED CHANNELS FROM BITS 11-17 AT COND PI TIME.
5. WHEN A PI4 10 REQ 1-7 SIGNAL (INT REQ) IS ACTIVE, IF ITS CORRESPONDING CHAN # IS ON IN THE PI ON REG AND PI 1 ACTIVE IS PRESENT, THE INT CHAN # WILL BE LOADED INTO THE PRI REQ REG SO THE INT CAN BE PROCESSED.
   NOTICE THAT CHAN ØØ IS NOT AFFECTED BY THIS.
6. SOFTWARE CAN GENERATE INTERRUPTS VIA THE PI GEN REG. AS LONG AS PI 1 ACTIVE IS PRESENT.

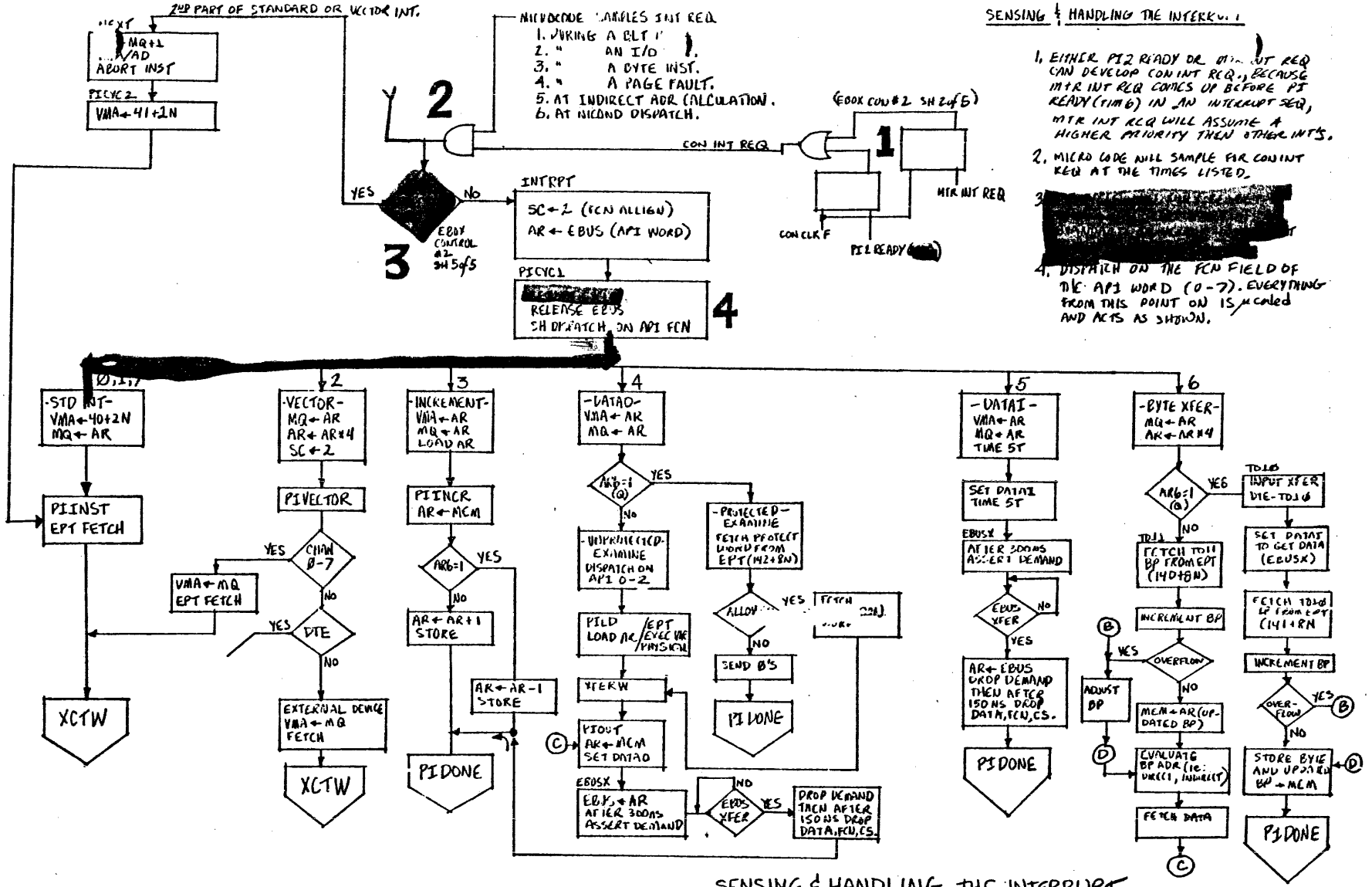
Ø1-3



E131

HONORING THE INTERRUPT

1. [redacted] THE PI SYSTEM HAS BEEN INITIALIZED AND ALL CHANNELS CAN INTERRUPT.
2. THE PI REQUEST REG WILL LOAD ALL PI REQUESTING CHANNEL #'S WHEN CLK'D WITH PIS LOAD, WHICH IS ACTIVE WHEN THE PI SYSTEM IS NOT PROCESSING AN INTERRUPT.
3. THE PRIORITY ENCODER SELECTS THE HIGHEST PI REQ. (APR EBOX DIS CS IS HI, THEN PIRØ, PIH1, PIR1, PIR2, PIR2, ETC)
4. THE PRI ENCODER GENERATES PI2 REQ (ANY REQ) AND THE OCTAL REPRESENTATION OF THE HI PRI INT ON PI 4-2-1.
5. PIS TEST ALLOWS PI2 REQ TO BE "FELT" AT THIS GATE. IF THE EBOX IS NOT HALTED, EBUS REQ WILL BE GENERATED. IF THE EBOX IS HALTED, ONLY PIØ WILL CAUSE E BUS REQ AT THIS TIME.
6. E BUS REQ ENABLES THE PI TIMER AND SETS PI GRANT. FOR EXT DEVICES THIS ALLOWS THE TTL AND ECL TO "LINK-UP" AND ALLOW COMMUNICATION BETWEEN THE DEVICES AND THE PI.
7. PI GRANT ALSO CAUSES THE SELECTED CHANNELS CHAN# TO BE PLACED ON CS Ø4-Ø6 AND ENABLES FØØ (PI SERVED). WHEN DEMAND IS GENERATED AT TIM2 THIS PRESET INFO WILL BE CLOCKED INTO THE DEVICES RECOGNIZING THE CHAN#.
8. THE DEVICES ON THE SEL CHAN WILL RESPOND WITH THEIR PHYS CONT # ON EBUS DØØ:15 (ONE BIT/DEVICE) SEE TABLE 1. AT TIM3 THESE #'S ARE LOADED INTO A REG AND PRESENTED TO ANOTHER PRIORITY ENCODER.
9. THE HI PRI PHYS CONT # (Ø=HI) IS PRESENTED TO A MIXER AND AT TIM4 IS GATED TO EBUS CS ØØ:Ø3 (ALSO THE CHAN # IS PLACED ON EBUS CSØ4:Ø6) TIM4·PI GRANT CAUSES FØ4·FØ2 (PI ADR IN). DEMAND IS ASSERTED AT TIM5.
10. THE FULLY SELECTED DEVICE RESPONDS WITH E BUS XFER, AND PLACES AN API WORD ON BITS ØØ:35 OF THE E BUS. EBUS XFER CAUSES THE TIM6 SIGNAL WHICH IN TURN GEN'S PI RDY TO THE E BOX. (ALERTS THE E BOX)



SENSING & HANDLING THE INTERRUPT

1. EITHER PI2 READY OR MTR INT REQ CAN DEVELOP CON INT REQ., BECAUSE MTR INT REQ COMES UP BEFORE PI READY (TIM 6) IN AN INTERRUPT SEQ, MTR INT REQ WILL ASSUME A HIGHER PRIORITY THEN OTHER INTS.
2. MICRO CODE WILL SAMPLE FOR CON INT REQ AT THE TIMES LISTED.
3. [redacted]
4. DISPATCH ON THE FCN FIELD OF THE API WORD (0-7). EVERYTHING FROM THIS POINT ON IS uCoded AND ACTS AS SHOWN.

SENSING & HANDLING THE INTERRUPT

PI SYS SH 3 OF 4

E1-32

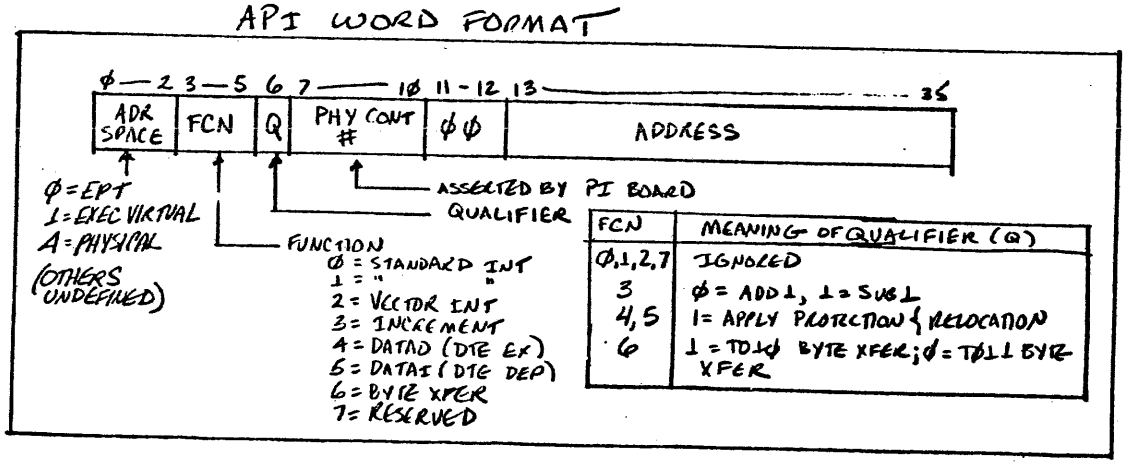
– DIAGNOS & FUNCTIONS –

DIAG FUNCTION			EBUS LINE USE**						
	DS Ø5	DS Ø6	D11	D12	D13	D14	D15	D16	D17
READ PI ON REG	Ø	Ø	PI ON 1	PI ON 2	PI ON 3	PI ON 4	PI ON 5	PI ON 6	PI ON 7
READ PI GEN REG	Ø	1	PI GEN 1	PI GEN 2	PI GEN 3	PI GEN 4	PI GEN 5	PI GEN 6	PI GEN 7
READ CS Ø1-Ø3, Ø5, Ø6	1	Ø	EBUS CS Ø5	EBUS CS Ø6	EBUS COMMAND	EBUS CS ØØ	EBUS CS Ø1	EBUS CS Ø2	EBUS CS Ø3
READ PI STATE	1	1	PI2 TIMER DONE	PI5 GRANT	PI2 STATE HOLD	EBUS CS Ø4	PI2 HONOR INT'NAL	PI2 READY	PI5 EBUS REQ

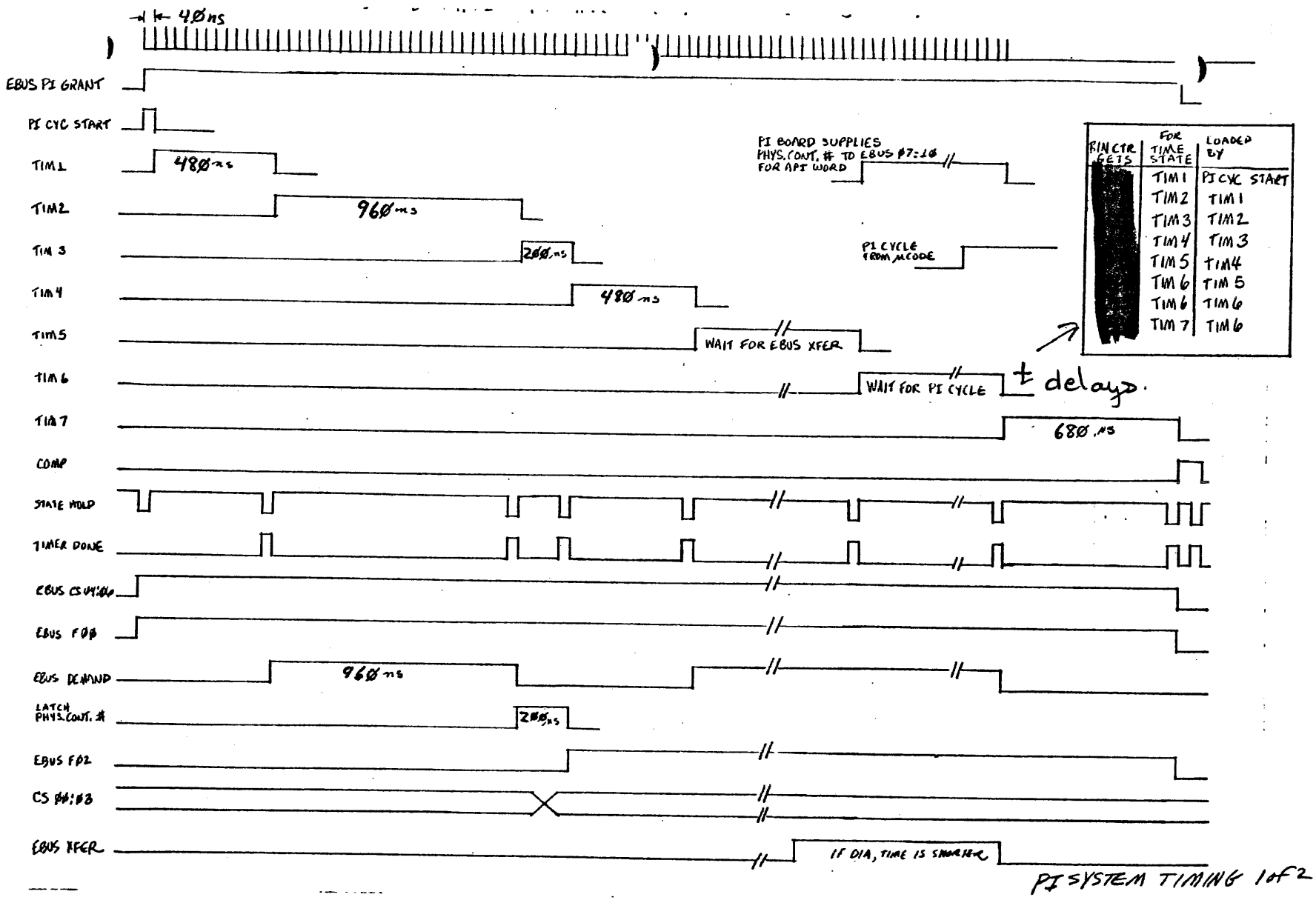
** EBUS D03-D09 ALWAYS GETS PIH1-7 AND EBUS D10 ALWAYS GETS PI ACTIVE..

TO ALLOW THE INDICATED "BITS" TO BE CATED TO THE EBUS LINES, THE PI SYSTEM MUST RECEIVE "DIAG READ FUNC 10x" AND NOT (-) DIAG 04 ...

API WORD FORMAT



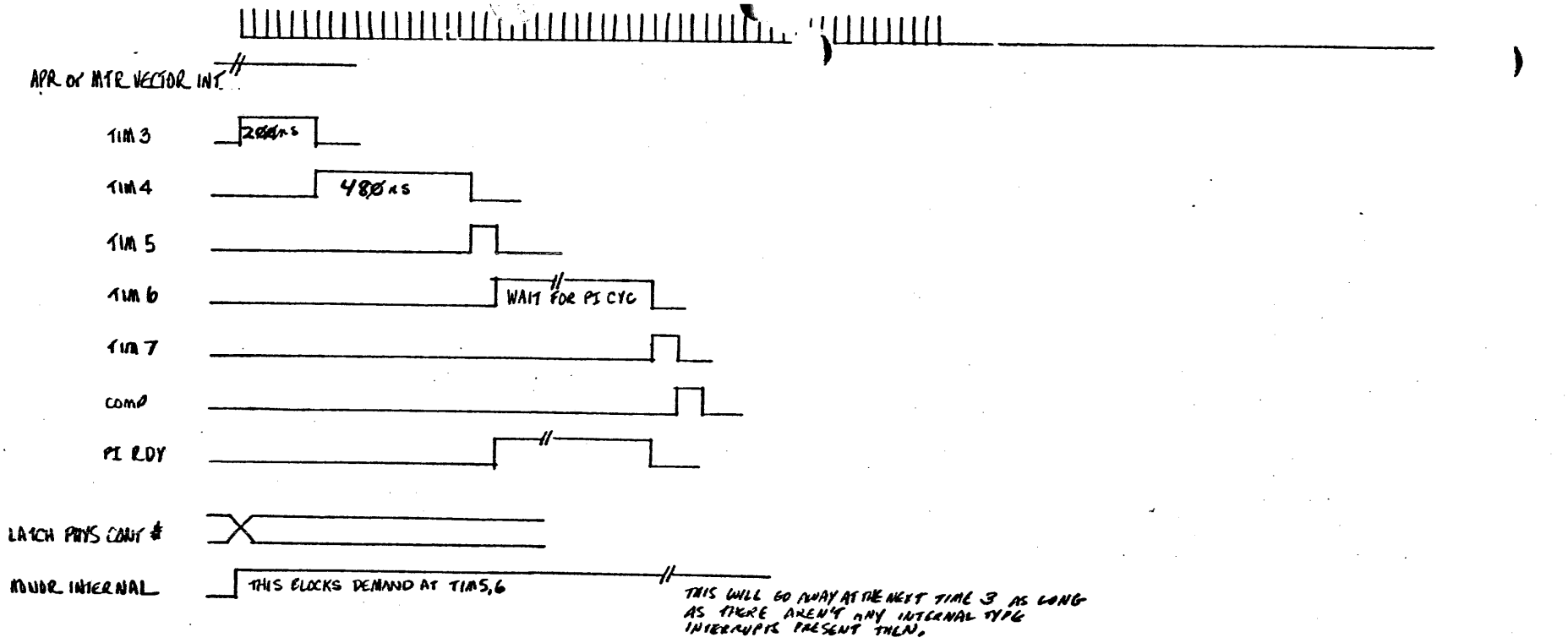
FCN	MEANING OF QUALIFIER (Q)
Ø,1,2,7	IGNORED
3	Ø = ADD 1, 1 = SUB 1
4,5	1 = APPLY PROTECTION { RELOCATION
6	1 = TO Ø BYTE XFER; Ø = TO 1 BYTE XFER



PI SYSTEM TIMING 1 of 2

E1-34

261-6 2-82



E1-35

PI SYSTEM TIMING 2 of 2

CHAPTER 1
OVERVIEW

1.1 GENERAL INFORMATION

The KL10 CPU has the following built-in program clocks, each providing a different timing or counting function:

Interval Timer
Time Base
Performance Analysis Counter
Accounting Meters

The *Interval Timer* is similar to the DK10 Real-Time Clock in KA10/KI10 systems. It provides a programmable source of $2^{12} - 1$ interrupts with 10 μs resolution and a choice of two possible timing intervals ranging from 10 μs to 40.95 ms.

The readable *Time Base* is a long-term clock for measuring elapsed time with 1 μs resolution. (Long-term power line frequency time base is provided by the front end processor.) It uses a 1.0 MHz (0.005%) frequency source, derived and down-counted from the basic machine clock, that gives less than 5 seconds of drift (gain or loss) over a 24-hour period.

The *Performance Analysis Counter* is a tool for testing and evaluating the KL10 system. It monitors either the duration or the rate of occurrence of several hardware signals from various parts of the CPU. The signals, chosen for their usefulness in evaluating system performance, define machine states and conditions not easily measured by software techniques. Any number of the available signals can be monitored and they are selected by means of a Boolean expression loaded by the program.

The *Accounting Meters* include an EBox Busy Meter that counts when the EBox is executing microcode and a Memory Cycle Meter that counts the number of EBox memory references. The two meters provide a reproducible measure of the processor resources used by a program. They are used for billing users and for benchmark or comparison purposes.

The hardware associated with the program clocks is contained on the M8538 Meter Board, a hex-height module mounted in the CPU cabinet.

1.2 INSTRUCTION SET SUMMARY

Operations on the program clocks are performed by I/O instructions directed to internal devices TIM, MTR, and PAG. Device mnemonics TIM and MTR (device codes 020 and 024) are used exclusively for the clocks. TIM is used for operations on the Interval Timer, Time Base, and the Performance Analysis Counter. MTR is used for operations on the Accounting Meters and for miscellaneous clock control. In addition, a DATAO to the KL10 paging system, PAG (device code 010), optionally stores the current value of the Accounting Meters during a context switch. The I/O instructions are summarized in Table 1-1.

Table 1-1
Instruction Set for Program Clocks

Instruction	Function
CONO TIM	Set Interval Timer On/Off, Period (12 bits); Clear Interval Timer Counter, Done, Overflow.
CONI TIM	Read Interval Timer On, Period (12 bits), Counter (12 bits), Done, Overflow.
DATAO TIM	Not Used
DATAI TIM* (RDTIME)	Read Time Base (60 bits)
BLKO TIM (WRPAE)	Set Performance Analysis Counter Enables, Clocking Mode; Clear Performance Analysis Hardware Counter.
BLKI TIM* (RDPERF)	Read Performance Analysis Counter (60 bits).
CONO MTR	Set Accounting Meter On, Enables; Set Time Base On/Off; Clear Time Base Hardware Counter; Set Interval Timer PIA.
CONI MTR	Read Accounting Meter On, Enables; Read Time Base On, Read Interval Timer PIA.
DATAO MTR	Not used
*DATAI MTR (RDEACT)	Read EBox Busy Meter (60 bits).
BLKO MTR	Not used
*BLKI MTR (RDMACT)	Read Memory Cycle Meter (60 bits).
DATAO PAG	Store Accounting Meters if bit 02 = 1 and bit 18 = 0.

*Double-Precision Instructions

1.3 BASIC OPERATION

The Interval Timer consists of a 12-bit counter, a Period register, and circuitry to compare the contents of the two. A vector interrupt (to EPT location 514) is generated when the incrementing counter reaches a value equal to the Period register contents previously loaded by the program. When the match occurs, a DONE flag sets to generate the interrupt and the counter is automatically reset to 0 in preparation for timing the next interval.

If the program loads a value into the Period register that is less than the current value of the Interval Timer counter (a programming error), an interrupt will occur. The interrupt request occurs when the counter, with no match to reset it to 0, increments to its maximum count and sets an OVERFLOW flag. The OVERFLOW and DONE flags, which indicate the nature of the PI requests generated, can be read by the CONI TIM instruction. The instruction also reads the up-counter contents. This allows the programmer to check the current value before loading the Period register, thereby preventing counter overflows from occurring.

The CONO TIM instruction furnishes the main control for the Interval Timer. Besides loading the Period register, it turns the up-counter on or off and allows the counter to be cleared at the same time. It also allows the DONE and OVERFLOW flags to be cleared so that PI requests can be dismissed. The PI channel number assignment (PIA) is loaded by the CONO MTR instruction.

While the counter associated with the Interval Timer is 12 bits and fully contained on the Meter Board, the counters associated with the other program clocks are 60 bits and (to save hardware) partially implemented in system memory. Memory is used in that each 60-bit clock has a corresponding double word in a process table (EPT/UPT). Only a clock's low order 16 bits are contained on the Meter Board. When a 16-bit hardware counter goes half-full (the high order bit = 1), the corresponding double word in memory is updated with that value; that is, the hardware counter is added to the contents of the EPT/UPT location associated with the clock. Thus, the double word in memory can be regarded as the value of the full clock at the most recent time that the hardware counter was half-full. Process table locations are listed in Table 1-2.

Table 1-2
Process Table Locations for Program Clocks

Clock	Process Table	Locations
Time Base	EPT	510, 511
EBox Busy Meter	UPT	504, 505
Memory Cycle Meter	UPT	506, 507
Performance Analysis Counter	EPT	512, 513

The 60-bit clocks are updated by the microprogram, not by the Meter Board hardware. When a hardware counter goes half-full, the Meter Board posts a request for service. At its next opportunity and using CPU hardware, the microprogram reads and clears the hardware counter and adds its value to the appropriate double word in the EPT/UPT. No PI level is assigned for the update; the addition (double-precision binary add) takes place regardless of the state or level of the PI system.

The DATAO PAG instruction can also cause a 60-bit clock update by specifying that the current value of the Accounting Meters be stored (bit 18 = 0) when a new UBR address is loaded (bit 02 = 1). Operation is the same except that a hardware counter going half-full does not initiate the update (counters can be any value) and two hardware counters are read, cleared, and their values added to UPT locations. (There are two Accounting Meters.) Because the update operation occurs before the new UBR address is loaded, the UPT locations that are updated to the current value of the Accounting Meters are for the old user. Thus, during a context switch, the instruction saves the Accounting Meters for the old user and clears the hardware counters for the new user.

The 60-bit clocks are read by double-precision DATAI and BLKI instructions to internal devices TIM and MTR. Although they have different operation codes, DATAI and BLKI are executed by the microprogram in the same fashion. In a process similar to the clock update, the appropriate hardware counter is read, but not cleared, and added to the corresponding double word in memory. The sum is then written in E and E+1, where E is the effective memory address specified by the DATAI or BLKI.

The Time Base and Accounting Meters are controlled by the CONO MTR instruction. Provision is made to turn the clocks on or off, clear the Time Base's hardware counter, and set the Accounting Meter enables. When turned on, the Accounting Meters always count in user mode. Through the use of the enables, it can also be made to count in executive mode when the machine is at some PI level, at no PI level, or independent of PI level. Control information set by the CONO can be read by the CONI MTR instruction.

The Performance Analysis Counter is controlled by the BLKO TIM instruction. It sets the clock's enable bits and clocking mode and it provides for clearing the 16-bit hardware counter. The enable bits are grouped to form the terms of a Boolean expression in which each term corresponds to a hardware signal or to a number of related signals. A single term will be true when the signals are in the state defined by the enables. Also, all of the terms must be true to satisfy the input gating to the clock. Thus each term must have a "don't care" bit or some other means to enable it when the machine state it represents is not to be monitored. The clocking mode set by the BLKO is either duration mode or event mode. In duration mode, the clock counts at one-half the basic machine clock rate as long as the Boolean expression is true. In event mode, it counts whenever the Boolean expression goes from false to true.

SECTION 1
OVERVIEW

1.1 GENERAL

The M Box is the memory interface in the KL-10 system. The M Box contains a cache memory, four memory buffers (MBs), address modification and verification logic, channel logic, and cycle control logic (see Figure 1-1). The functional elements provide the E Box and channels with access to core memory.

1.2 CACHE MEMORY

The Cache is a 2048 word data buffer where instructions and data are stored and maintained as the E Box issues memory reference requests. The purpose of Cache is to increase the efficiency of the machine by reducing memory access time and decreasing the number of main memory cycles. This is accomplished by recognizing that in the course of running a program, an individual instruction may be executed many times, i.e., a program loop. Therefore, once data has been moved from core to Cache, it can be stored or fetched very quickly (typically 125 ns) thus speeding up the program execution. Initially, as the E Box makes requests for words, memory cycles are granted by the M Box, core is accessed, and four words are received (the one requested plus the associated three) and stored in Cache. Considering that it is likely the E Box will request the next consecutive word, this word will already be in cache and will be readily available since a core cycle won't be required. When the E Box makes a request for a word that is not already in Cache, the M Box grants another core cycle that brings in four more words. Therefore, a program residing in less than 2048 locations would make references to the slower core memory only upon the first access of every fourth location, and thereafter all memory references would be made to the faster Cache memory.

1.3 MEMORY BUFFERS (MBs)

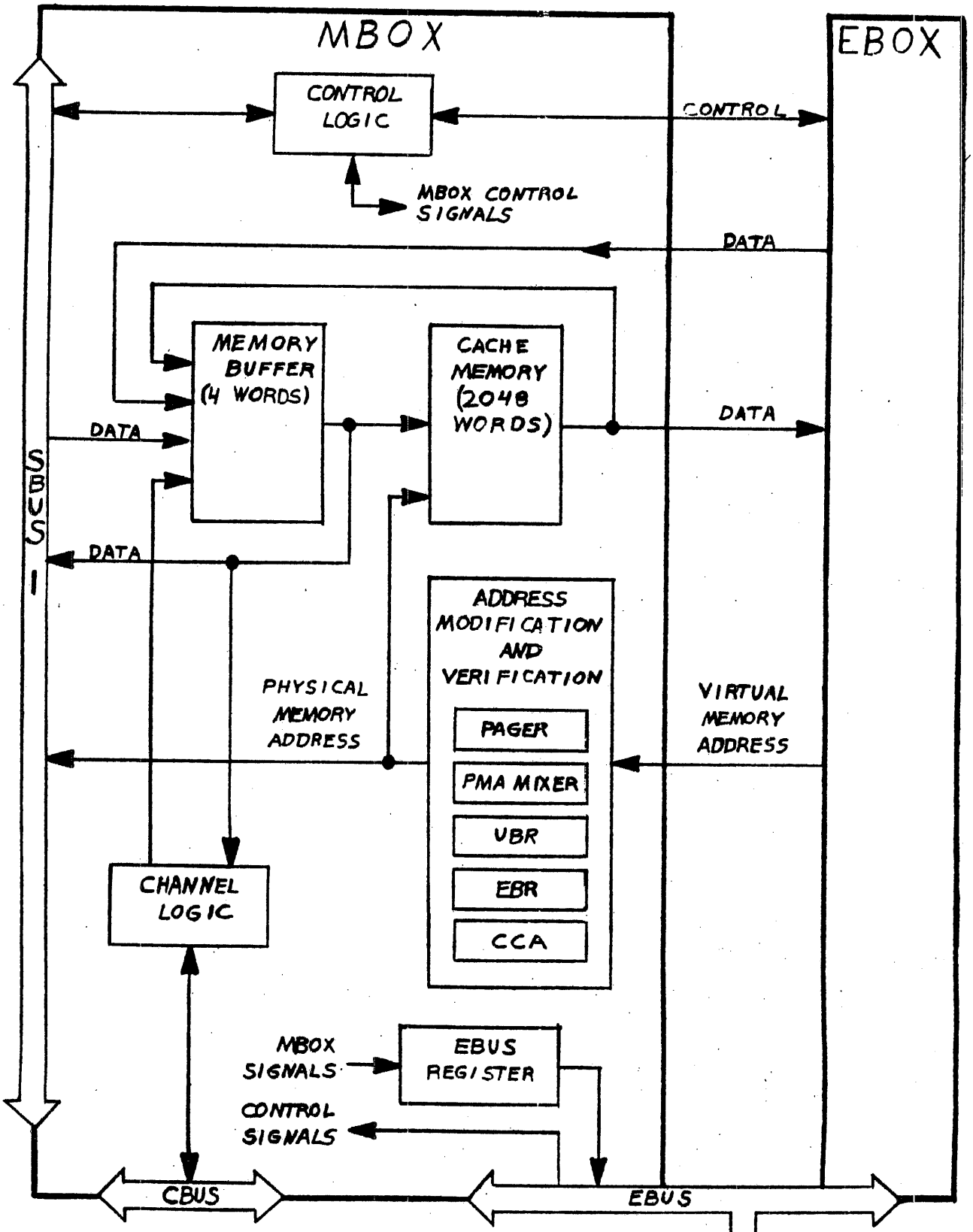
The MBs provide four words of intermediate storage between core and cache or core and the channel or vice versa.

1.4 ADDRESS MODIFICATION AND VERIFICATION LOGIC

This logic consists of a pager, physical memory address mixer (PMA), user and executive base registers, CCA register, and the necessary control logic. Its main purpose is to convert virtual memory addresses (VMA) received from the E Box into physical addresses for use by cache and core. A simplified form of this conversion is shown in Figure 1-2.

The Pager contains a page table storage area that is filled as the E Box makes paged requests for words for which the page table has no valid physical page address. When in the KI paging mode a page refill mechanism is employed to automatically fetch page table entries from either the user on executive process tables located in core. When in the KL paging mode page refills are handled by the E Box.

The PMA is assembled by the M Box to accommodate the type of request. All addresses that may be needed are made available to the PMA at all times. Depending on the type of request, the PMA is controlled to enable the



M BOX 1-2

Figure 1-1 M Box Simplified Block Diagram

SEE PRINTSET 23

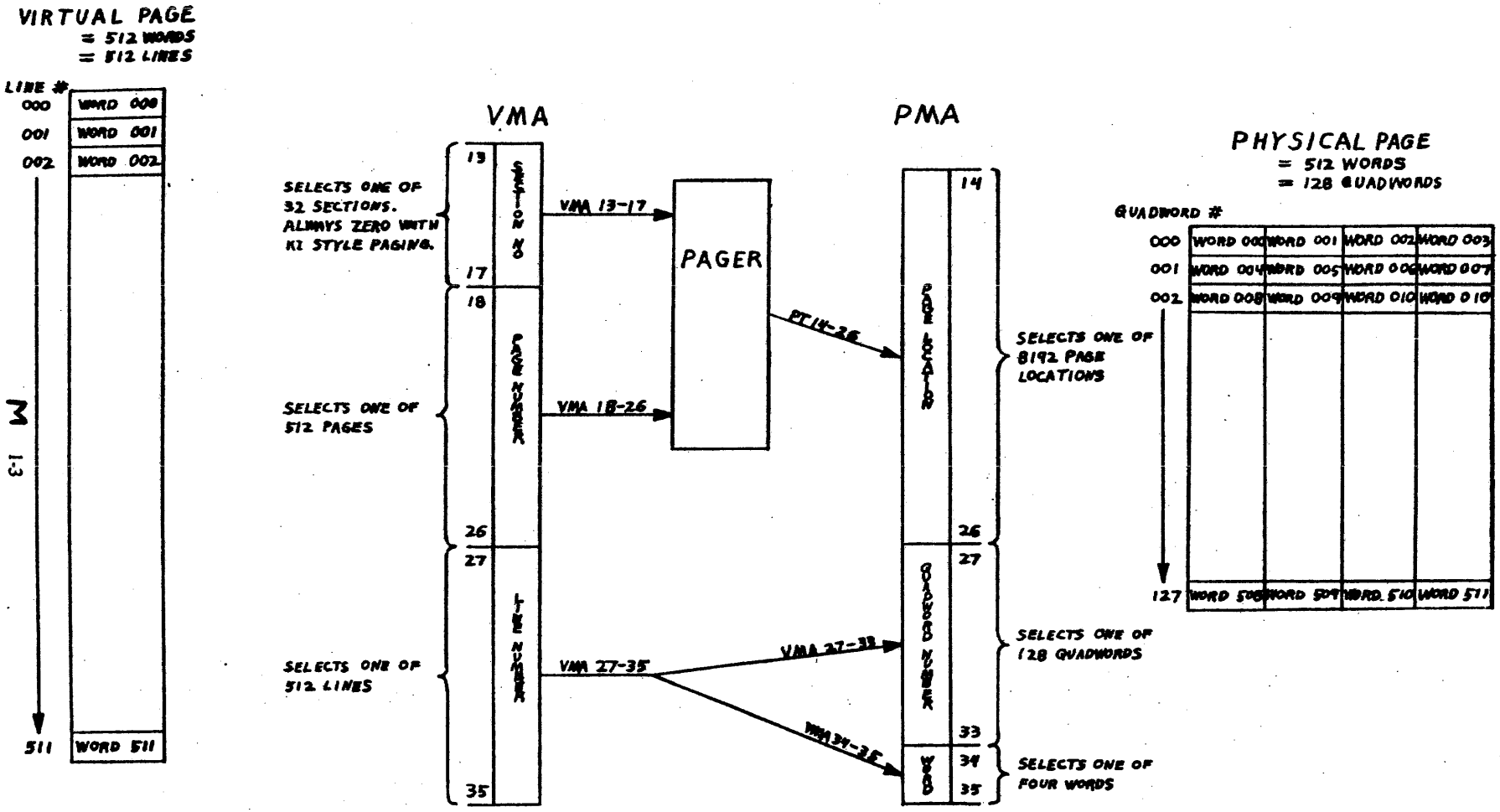


Figure 1-2 Simplified VMA to PMA Conversion

desired address mixture. The PMA gets the virtual section, page, and line address from the E Box VMA. The physical page address is selected from the page table, or from the CCW BUS if performing a channel operation. In addition, the PMA has access to the User Base Register (UBR), Executive Base Register (EBR) and the Cache Clearer Address register (CCA) which are loaded at some point with an appropriate address from the VMA.

1.5 CHANNEL LOGIC

The M Box may also be equipped with eight integral data channels. These channels interface with the Cache and the MBs to form the data path between core memory and the Channel Bus. The channels interact with the Cache to maintain the integrity of the flow of data between core memory and mass storage.

1.6 CYCLE CONTROL LOGIC

Besides the functional elements introduced above, the M Box contains control logic to execute operations and maintain order. The controls are:

a. Cache Control
b. Channel Control
c. MB Control
d. Core Control
e. Cache Clearer Control

These controls operate independently of each other until a requested operation is completed. Requests can be issued by the E Box, the C Bus, or by the controls themselves. This control structure increases the efficiency of the machine in that several operations can be going on at the same time.

On a priority basis, the M Box grants and executes all memory requests made by the E Box and up to eight high-speed data channels. Once a request is granted the M Box will remain busy for a number of clock ticks. To assure the channels adequate service, they are provided with a higher priority than an E Box request.

SECTION 2
INTERFACE

2.1 GENERAL

To interface the MBox with the ⌒ system three sets of controls are employed. These controls interface the MBox to the EBox, SBus & Channel Control.

2.2 E/MBOX INTERFACE LINES (See Figure 2-1)

The interface lines that carry data, timing and control information between the EBox and MBox are listed, together with their function, below.

CLK	A 31.25 nanosecond timing signal that originates on the CLK module in the E Box and is distributed to all M Box boards.
CLK EBOX REQ	Issued by the E Box to request service (E-Box Cycle).
CSH EBOX TO IN	Asserted when the cache cycle control honors an E-Box request. This signal clears the CLK EBOX REQ line.
VMA AC REF A	Asserted by the E Box to abort the requested cycle should the request turn out to be an accumulator (fast memory) reference.
MCL VMA READ	Asserted by the E-Box to request the word specified by address lines VMA 13 - 35.
MCL VMA WRITE	Asserted by the E-Box to request that the word contained on data lines AR 00 - 35 be written into the location specified by address lines VMA 13 - 35.
MCL VMA PAUSE	When asserted with MCL VMA READ, the Read-Pause portion of a Read-Pause-Write cycle is requested. When asserted with MCL VMA WRITE, a Write-Pause cycle is requested.
APR EBOX LOAD REG	Asserted by the E-Box when loading the user base register (UBR), executive base register (EBR), or the cache clearer address register (CCA).
APR EBOX READ REG	Asserted by the E-Box when reading the user base register (UBR), executive base register (EBR), cache clearer address register (CCA) or the error address register (ERA). The contents of the specified register is transferred to the E-Bus register where it can be read by executing DIAG READ FUNC 16X. This line is also asserted when loading the refill table and during an E-Box MAP request.

M Box 2-10

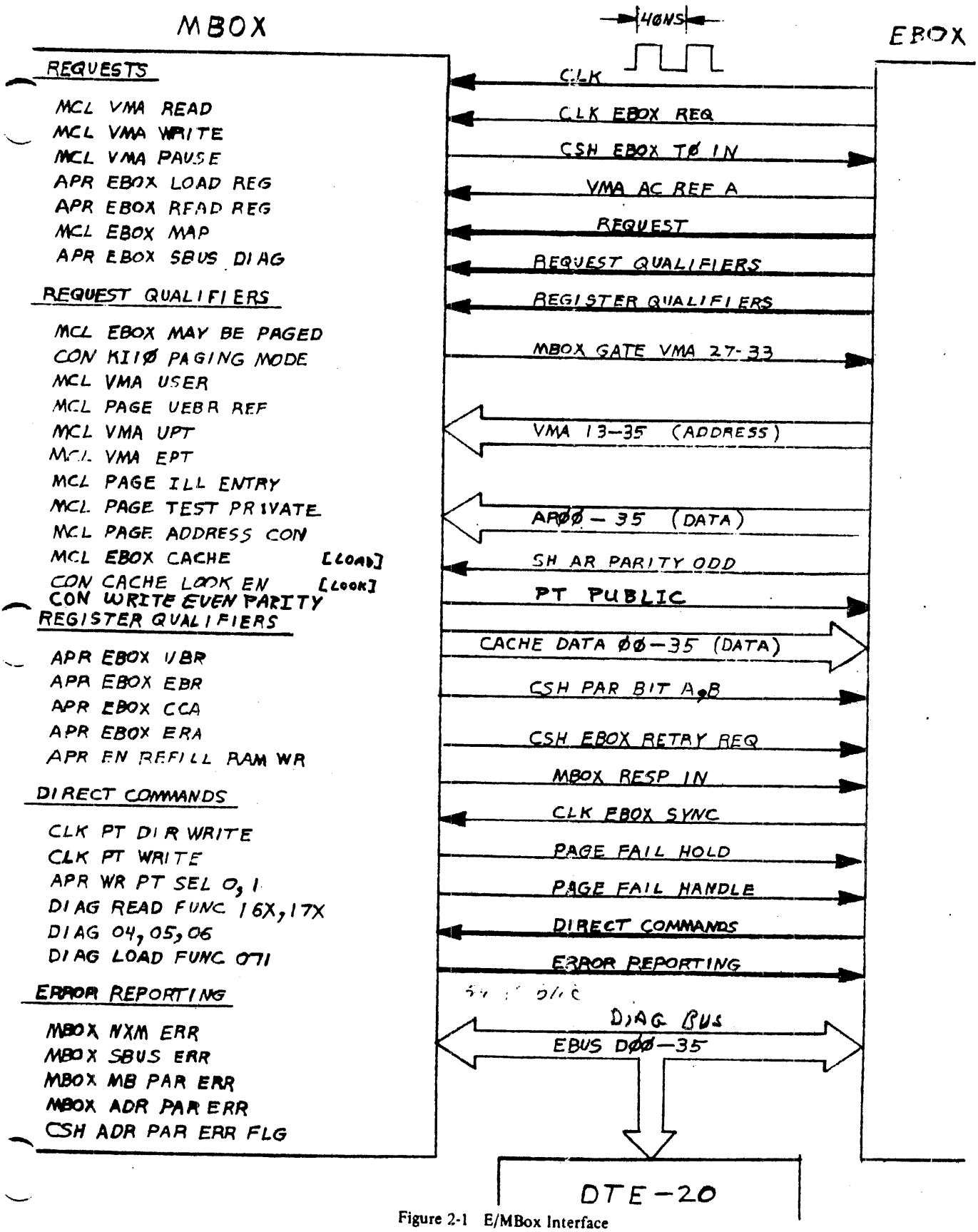


Figure 2-1 E/MBox Interface

MCL EBOX MAP	This line requests a cycle that transforms the virtual page number contained on address lines VMA13–35 into a physical page location. The physical page location and its assigned descriptor bits are transferred from the page table to the E-Bus register where they can be read by executing DIAG READ FUNC 16X.
APR EBOX SBUS DIAG	This function allows the E-Box to send a control word to the memory controllers (MA-20, DMA-20) and to receive a word of diagnostic information back.
MCL EBOX MAY BE PAGED	Asserted by the E-Box to indicate that the reference is paged.
CON KI10 PAGING MODE	Indicates KI paging mode when asserted and KL paging made when negated.
MCL VMA USER	Asserted by the E-Box when the reference is to the user address space.
MCL PAGE UEBR REF	Asserted by the E-Box when a reference is made to the USER or EXECUTIVE process table.
MCL VMA UPT	Asserted by the E-Box when a reference is made to the user process table.
MCL VMA EPT	Asserted by the E-Box when a reference is made to the executive process table.
MCL PAGE ILL ENTRY	Asserted by the E-Box to force a page fail condition in the M-Box to abort the current request. This is necessary if the previous instruction was fetched from a concealed page and it was not a portal instruction.
MCL PAGE TEST PRIVATE	Asserted by the E-Box for a non-instruction reference in the public mode to check whether the referenced page is concealed. If the page is not concealed PAGED FAIL HOLD is asserted.
MCL PAGE ADDRESS COND	Asserted along with MCL PAGE ILL ENTRY when the E-Box detects an address break condition. This forces a page fail in the E Box.
MCL EBOX CACHE	Asserted by the E-Box to allow references to cache. When negated all references are to core.
CON CACHE LOOK EN	Is normally asserted by the E-Box to allow references to cache. Is negated during a one word read from core.
APR EBOX UBR	Asserted by the E-Box when the user base register (UBR) is being loaded or read.
APR EBOX EBR	Asserted by the E-Box when the executive base register (EBR) is being loaded or read.
APR EBOX CCA	Asserted by the E-Box when the cache clearer address register (CCA) is being loaded or read. Accumulator bits AC10 through AC12 and virtual memory address bits VMA 14 through VMA 26 contain the following control information when loading the CCA register.

APR EBOX CCA (cont)	• AC10 is set when clearing only one page from cache and reset when the entire cache is to be cleared. • AC11 is set when the written words in cache are to be written back to core. • AC12 is set when the cache entries are to be invalidated. • VMA 14–26 contain the page number to be cleared when clearing only one page.
APR EBOX ERA	Asserted by the E-Box when the error address register (ERA) is being read. This register is loaded when a parity or non-existent memory error is detected.
APR EN REFILL RAM WR	This line is asserted to load the use bit refill table. When this signal is received by the M-Box, VMA lines 27–33 must contain the address into which the data contained on VMA lines 18–20 is to be loaded.
MBOX GATE VMA 27–33	Asserted by the M-Box when an E-Box cycle is granted to route VMA lines 27–33 directly to cache.
VMA 13–35	These lines may contain a virtual memory address or data to be loaded into a register.
VMA 27G–33G	These address lines are gated to the cache whenever an E-Box cycle is started.
AR00–35 and SH AR PARITY ODD	These lines contain data and parity from the E-Box.
CACHE DATA 00–35 and CSH PAR BIT A,B	These lines carry data and parity from the M-Box to the IR, AR, and ARX registers in the E-Box.
CSH EBOX RETRY REQ	This line is asserted by the M-Box when a read or write to core is required and core is busy, or when a refill or writeback cycle is required. This signal is also generated if the E-Box attempts to write into a cache location that a core cycle is in the process of filling. It forces the E-Box to set up the interface lines to retry the current request.
MBOX RESP IN	Asserted by the M-Box after the request has been processed to completion.
CLK EBOX SYNC	Asserted by the E-Box to reset MBOX RESP IN and to complete the handshaking procedure.
PAGE FAIL HOLD	This signal is generated by the M-Box whenever a page table parity error is detected or when a page test fails.
PAGE FAIL HANDLE	Asserted by the M-Box if the directory or page table does not contain a valid entry during a KL paging operation.
CLK PT DIR WRITE	Asserted by the E-Box during KL paging to write or clear an entry in the page table directory.

C' K PT WRITE	Asserted by the E-Box during KL paging to write or clear an entry in the page table.
APR WR PT SEL 0,1	These two lines are used to select the page table directory or a portion of the page table when clearing a specified location.
DIAG READ FUNC 16X, 17X	Asserted in conjunction with the DIAG04, 05, 06 lines to select the M-Box control signals to be placed on the E-Bus for diagnostic purposes.
DIAG 04, 05, 06	These lines select a mixer input when performing a diagnostic read function.
DIAG LOAD FUNC 071	Asserted by the E-Box to set up the cache input mixer. The code presented on E-Bus 30–35 determines which input will be selected.
MBOX NXM ERR	The non-existent memory flag is set if all requested words have not been received from core 64 microseconds after SBUS START was asserted.
MBOX SBUS ERR	This flag is generated if the DMA-20 detects a data parity error during a core read or write or if one of the memories fails to send an acknowledge within 32 microseconds of the request.
MBOX MB PAR ERR	This flag is set when a memory buffer parity error is detected. Parity is checked whenever data is moved out of a MB.
MBOX ADR PAR ERR	This flag is set when the DMA-20 detects a parity error on ADR 14–35, RD RQ, WR RQ, and RQ 0–3 lines.
CSH ADR PAR ERR FLG	This flag is set when a cache directory parity error is detected.
EBUS D00–D35	The E-Bus data lines allow the E-Box and DTE-20 to read diagnostic information from and send control information to the M-Box.

2.3 SBUS INTERFACE LINES (See Figure 2-2) INTERNAC

The interface between the DMA-20 and the M-Box provides the means to transfer data, commands, and status information between the two devices. The function of these lines is described below.

SBUS 1 CLK EXT	The SBus clock is a 125 ns square wave that drives the internal DMA-20 clock generator, which in turn provides phased (deskewed) 125 and 62 ns clock signals. The Phase A clocks are generated on the fall time of the external clock. Equivalent Phase B clocks are generated on the subsequent rise of the external clock.
SBUS 1 D00–35	These 36 bidirectional data lines transfer data to the M-Box during a read cycle and the read portion of a read/modify/write cycle. Likewise, the lines transfer data to the DMA-20 during a write cycle and the write portion of a read/modify/write cycle.
SBUS 1 DATA PAR	This bidirectional line transfers odd parity for the associated data words during read/write cycles. The write data parity is checked in the DMA-20 and transferred to the storage module. The read parity is read out of memory, checked in the DMA-20, and transferred to the M-Box. Should a read parity error be detected the DMA-20 will set appropriate error flags and transfer the bad parity to the M-Box.

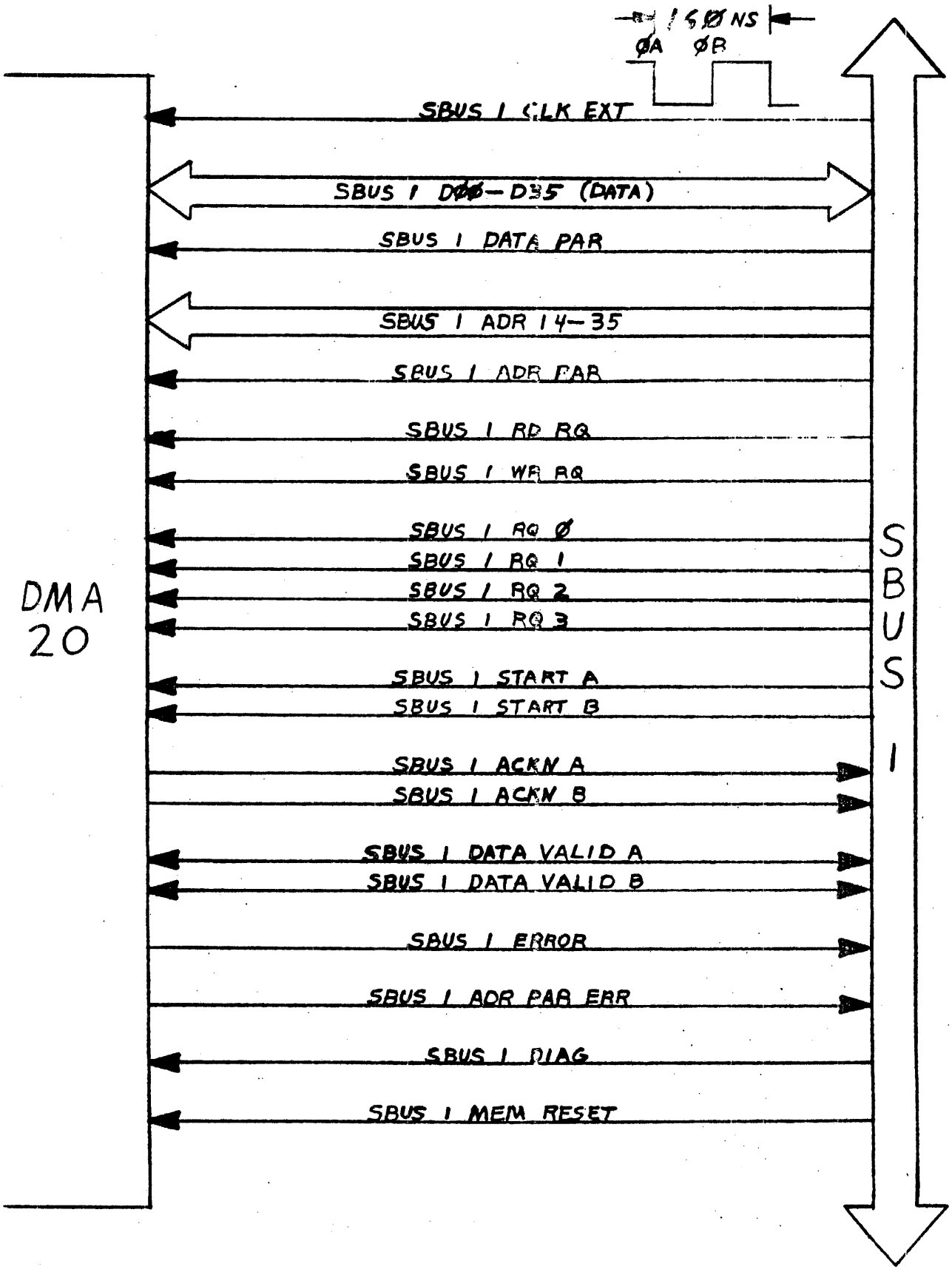


Figure 2-2 SBUS Interface

SBUS 1 ADR 14–35	These address lines designate the specific address (physical location) to read/write in the selected storage module. Bits 14–35 address a quadword; bits 34–35 addresses the first word to be accessed within the quadword.
SBUS 1 ADR PAR	The address parity line transfers odd parity for the address (ADR 14–35), read request (RD RQ), write request (WR RQ), and the request 0 through 3 (RQ 0–3) lines. Should an address parity error be detected the cycle request is inhibited and an error (SBUS 1 ADR PAR ERR) is generated.
SBUS 1 RD RQ	The M-Box asserts this line when it is requesting a memory read cycle. With both SBUS RD RQ and SBUS WR RQ asserted the M-Box is requesting a read/modify/write cycle from the addressed storage module.
SBUS 1 WR RQ	The M-Box asserts this line when it is requesting a memory write cycle. With both SBUS WR RQ and SBUS RD RQ asserted the M-Box is requesting a read/modify/write cycle from the addressed storage module.
SBUS 1 RQ 0–3	These four request lines specify which word (s) in a quadword are to be accessed or written.
SBUS 1 START A	The M-Box asserts this line on Phase A of the clock to initiate a memory cycle.
SBUS 1 START B	This signal is identical to START A except it is received on Phase B of the clock.
SBUS 1 ACKN A	This signal is generated on Phase A of the clock and indicates the storage module has acknowledged receiving an address and request. An acknowledge pulse is returned to the M-Box for each word requested.
SUBS 1 ACKN B	This line is identical to SBUS ACKN A except it is returned to the M-Box coincident with the Phase B clock.
SBUS 1 DATA VALID A	This bidirectional line transfers a signal from the DMA-20 in sync with the Phase A clock that indicates data and parity are available to the M-Box during a read cycle or read portion of a read/modify/write cycle. A DATA VALID pulse is generated when each requested word becomes available. Only during the write portion of a read/modify/write cycle is DATA VALID sent to the DMA-20. In this case it represents an equivalent write restart signal which initiates the write portion of the cycle in the DMA-20.
SBUS 1 DATA VALID B	This line is identical to DATA VALID A except it is generated on Phase B of the clock.
SBUS 1 ERROR	When this line is true detection of a read or write parity error from a storage module or a 32 microsecond storage module time out is indicated. The E Box may retrieve the particular error flag by executing the appropriate diagnostic cycle.
SBUS 1 ADR PAR ERR	This line indicates a parity error on the ADR 14–35, RD RQ, WR RQ, and RQ 0–3 lines.

SBUS 1 DIAG	The diagnostic line requests the diagnostic function specified by the word on the data line
SBUS 1 MEM RESET	Generates a power clear condition to initialize the logic in the DMA-20.

2.4 CHANNEL CONTROL TO CACHE INTERFACE

A summary of the channel control to cache interface is presented below. The interface signals the grouped according to their function.

a. Control Commands

CCL3 CHAN REQ	Issued by the channels to request service.
CCL2 HOLD MEM	Asserted by the channels if the channels have requests backed up. By asserting this signal, the channel is assured of the next core cycle by inhibiting an EBox Request from initiating a core cycle.
CSH CHAN CYC	Asserted when the cache cycle control starts processing the CHAN request. This signal informs the channel that it can start writing the MBs in the case of channel write operation or start looking for words ready to be taken from the MBs in the case of channel read operations.
CCL START MEM	Asserted by the channel during channel write operations after the first word is loaded into the MBs. Subsequent words are moved into the MBs at 125 ns intervals assuring the core control has a word to move to core when it is ready. The core control moves words to core at 187 ns intervals. During channel read operations, the cache cycle control starts the core cycle when it is ready.
CCL CH MB SEL 1--2	The channel places a two bit code on these lines to select the correct MB to be loaded during channel write operations or read during channel read operations.
CCL CH LOAD MB	Asserted by the channel to load the selected MB during channel write operations.
MB 0--3 HOLD IN	Asserted by the cache cycle control and/or the core cycle control during a channel read operation to load the MBs and to inform the channel that the corresponding word is ready to be taken.
CCL CH MB TEST PAR	Asserted by the channel to check the parity of the selected word before it is taken from the MB during channel read operations.

b. Request Qualifiers

CCL CHAN TO MEM	Asserted by the channel to specify a channel write operation is to be executed. When negated, a channel read operation is executed.
CCW CHAN WD0--3 RQ	These four signals are asserted by the channel to specify the words to be read or written.

b. (cont)

CCL CHAN EPT	Asserted by the channel to read or write the executive process table. The EPT is read to fetch the initial CCW and is written to store the channel status at the end of a transfer. The cache cycle control will automatically select the correct address for referencing the EPT.

c. Error Reporting Commands

CHAN PAR ERR	Asserted for one clock period when the MB parity check fails during a channel read or channel write operation. During channel write operations, parity is checked when the channel asserts CCL CH MB TEST PAR and during channel read operations, parity is checked when the cache cycle control or the core cycle control loads the word into the MB. This signal informs the channel that a data parity error occurred during the transfer.
CHAN ADR PAR ERR	Asserted for one clock period when the SBus address parity check fails during channel read or channel write operations
CHAN NXM ERR	Asserted for one clock period when the NXM counter in the MBox times out (64 μs). This counter times out if one of the ACKN pulses for the requested words is not received from the memory.

d. Address Lines

CCN CHA 14–35	Physical memory address from channel.

e. Data Lines

Data Buffer and path is an integral part of the MB modules.

f. Clocks

Clocks are distributed to the channels from the EBox.

SECTION 3
FUNCTIONAL DESCRIPTION

3.1 DATA FLOW

The data paths that are implemented to route data between the EBox, the CBus, and the SBus are shown in Figure 3-1. The desired path is selected by either the cache, core, or channel control logic, depending on the request that is being executed. The components that make up the data path are listed and described below.

3.1.1 Memory Buffer Input Mixer

The MB IN mixer is a 36-bit, five-input mixer that provides a means for selecting the input to the memory buffers. The MB IN Mixer is controlled by the MB IN SEL 1-2-4 code. By adjusting the select code, the MBs can be loaded with data from the sources listed in Table 3-1.

Table 3-1 MB IN Select Codes

Source	MB IN SEL 4	2	1
CACHE	0	0	X
AR	0	1	0
CH BUFFER	0	1	1
SBUS	1	0	X
CCW BUFFER	1	1	X

X = Don't Care

3.1.2 Memory Buffers (MBs)

The four MBs are 36-bit memory buffer registers for temporarily holding the data as it is moved from the source to the destination registers or RAMs. In effect the MBs serve as a buffer to normalize (compensate for the differences in speed) the transfer of data between the source and destination. The MBs are set up so that MB0 stores word 0 of a quad word group, MB1 stores word 1, and etc.

3.1.3 Memory Buffer Output Mixer

The MB SEL mixer is a 36-bit four-input mixer that selects the contents of one of the four MBs when transferring the data to the one of the following destinations.

a. Cache Data Input Mixer
b. Page Table Input Mixer
c. CCW Input Mixer
d. Channel Data Input Mixer
e. SBUS

The output of the mixer is also checked for odd parity whenever the data is being transferred.

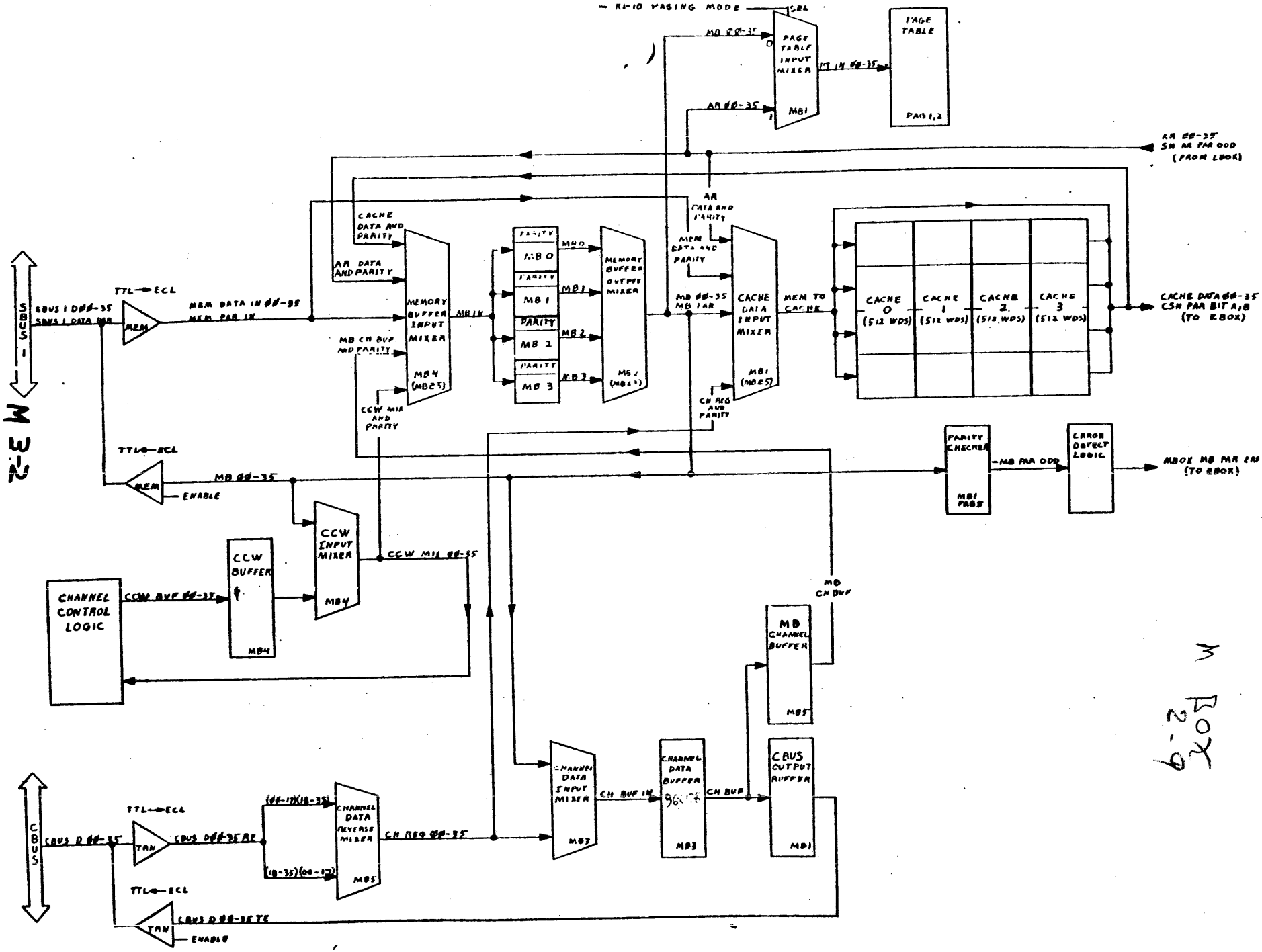


Figure 3-1. MBOX DATA PATHS

3.1.4 Cache Data Input Mixer

The MEM TO C mixer is a 36-bit, four-input mixer, that provides a means for adjusting the data source for the cache memory. The MEM TO C mixer is controlled by the MEM TO C SEL 1-2 code produced by the cache cycle control when a Cache cycle is started. Table 3-2 lists the paths that may be established by the mixer.

Table 3-2 MEM TO C Select Codes

Data Path	MEM TO C SEL 1-2 CODE	Function
AR → CSH	0	EBOX WRITE
MB → CSH MB → AR	1	EBOX READ
SBUS → CSH SBUS → AR	2	EBOX READ OR EBOX SBUS DIAG
0 → CSH	3	DIAG Function

3.1.5 Cache Memory

The Cache memory contains storage for 2048 words (512 quad words) in four identical quarters (cache 0, 1, 2, and 3). There is also a bypass route around cache so that data can be sent from the MBs or SBUS directly to the AR.

3.1.6 Page Table Input Mixer

The PT IN mixer is a 36-bit, two input mixer that is controlled by the -CON KI10 PAGING MODE line. If KI10 style paging is being performed, the input to the page table is from the MBs. If KL10 style paging is selected the input is from the AR.

3.1.7 Channel Data Input Mixer

The CH BUF IN mixer is a 36-bit, two-input mixer that manipulated by the channel control to move data into the CHAN BUF from the selected MB during a channel read operation or from the CBus data lines during a channel write operation.

3.1.8 Channel Data Buffer

The Channel Data Buffer (CH BUF) contains 16 locations of buffer storage for each channel; consequently, there are 128 locations in the CH BUF to accommodate all eight channels. The CH BUF is addressed by CH BUF ADR 00–06 which is a function of the selected channel and the buffer location to be read or written. This address is formed by the channel control.

3.1.9 CBUS output Buffer

The CBUS OUT BUF is a 36-bit register that holds the word to be moved from the CH BUF to the CBUS.

3.1.10 MB Channel Buffer

The MB CH BUF is a 36-bit register that holds the word to be moved from the CH BUF to the MB via the MB IN mixer during a channel write operation.

3.1.11 Channel Data Reverse Mixer

The CH REG mixer-latch is a 36-bit two-input mixer combined with a 36-bit register (Latch). This mixer-latch is controlled by the channel control to adjust the two half words coming in from the CBus (each half word is one word from the drive) in the correct order to accommodate both forward and reverse read operation of the drive before moving the word into the CH BUF.

3.1.12 CCW Buffer

The channel CCW Buffer (CCW BUF) contains two locations of storage for each channel; consequently, there are 16 locations in the CCW BUF to accommodate all eight channels. This buffer contains the Word Count (WC), the Address, the Command List Pointer and status information for each channel. The CCW BUF is addressed by CCW BUF ADR 00–03 which is a function of the selected channel and the buffer location to be read or written. This address is formed by the channel control.

3.1.13 CCW Input Mixer

The CCW mixer is a 36-bit, two-input mixer that is controlled by the channel control in executing the following operations:

a. Transfer a newly fetched CCW that was placed into a MB by the core cycle control from the MB to the CCW BUF.

b. Transfer the Address (ADR) or the Command List Pointer (CLP) from the CCW BUF to the CCW register when the channel issues a request to read or write memory.

c. Transfer the status from the CCW BUF to the MBs when the channel issues a request to store the status words.

3.2 CACHE MEMORY

3.2.1 Functional Description

The cache memory consists of solid-state, random access memory chips and control logic, that provides 2048 words of high-speed buffering between main memory (core) and the processor (E-Box). The Cache storage elements are logically divided into four identical quarters and are referred to as Cache 0, 1, 2, and 3 (see Figure 3-2). Each quarter is capable of storing 128 quadwords (512 words) thus providing a total storage capacity of 2048 words in a structure similar to core memory. At any given time, Cache may contain data whose distribution can range from up to four complete pages from anywhere in core, to four words from each of 512 different pages.

To identify each quadword group the Cache contains a directory that stores the physical page number of the quadword. The directory also contains the following bits:

- A valid bit for each of the four words within the quadword. A words valid bit is set when the word is fetched from core and stored in cache, or whenever the E-Box writes the word in cache.

- A written bit for each of the four words in the quadword. A words written bit is set whenever the word is written into by the E-Box thus superseding the data contained in core.

As the Cache is filled with data and instructions the associated locations of the directory are updated to specify the physical page address of the quadword and specify which words were fetched from core (valid bits). This continues until the Cache is filled. Thereafter, the least recently used quadwords are supplanted by new instructions and data as they are needed. Words that have been written into the Cache by the EBox are identified by updating the directory written bits accordingly so that they can be moved back to core before they are supplanted.

The cache also contains a use table whose output is used to determine which of the cache quarters (cache 0, 1, 2, or 3) a quadword will be written into. This determination is based on the recency of the data contained in each of the quarters. A new quadword is always brought into the location that contains the least recently used data. This convention ensures that a given quadword location in each of the four cache quarters will never contain more than one quadword from a given page. Therefore, a conflict will never occur when comparing the address with the contents of the directory to determine if the desired word is in the cache. This feature of refilling the Cache also tends to keep data and instructions that are used more frequently in the Cache longer.

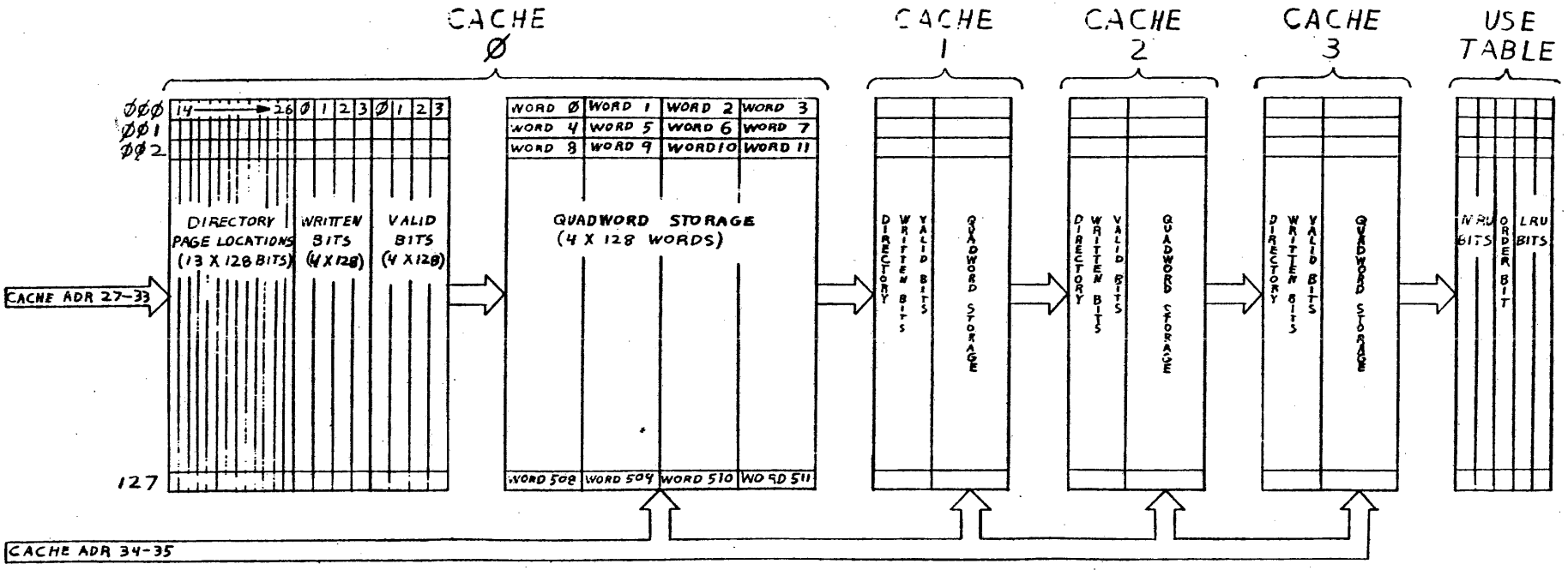
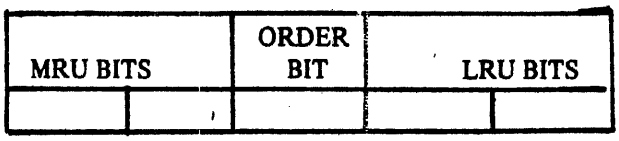


Figure 3-2 Cache Functional Organization

The Use table contains five bits of coded information for each of the 128 quadword locations. The five bit code is broken down as follows:



MRU Bits – These two bits contain the binary number of the cache quarter that was Most Recently Used.

LRU Bits – These two bits contain the binary number of the cache quarter that was Least Recently Used.

Order Bit – The order bit indicates the sequence in which the two remaining quarters were used. A 0 indicates they are in ascending order while a 1 indicates descending order. For example:

Cache Quarter				Use Table Code		
MRU			LRU	MRU	ORDER	LRU
0	1	2	3	00	0	11
0	2	1	3	00	1	11
1	0	3	2	01	0	10
1	3	0	2	01	1	10

Whenever data is fetched from or stored in cache, the use table is updated to reflect any changed status. When the system is initialized (powered on) a cache sweep operation is performed that resets all valid and written bits and places the code 00011 throughout the use table thus purging any illegal patterns. An illegal pattern is defined as one that has identical bits in the MRU and LRU fields.

3.2.2 Cache Read Operation (See Figures 3-3 and 3-4)
When a cache read operation is attempted, the cache is accessed and one of three conditions will exist. These conditions and the action taken by cache control is as follows:

a. The requested word is in cache. This is determined by comparing the physical memory address with the output of the addressed (quadword) location within each of the four cache directories. If a match is found a check is made to see if any of the four words, within the addressed quadword is valid. If so, the signal CSH (X) VALID MATCH is generated enabling that cache quarter's data storage area and updating the Use bits. A further check is then made to see if the requested word (specified by PMA 34-35) is valid, and if it is, RD FOUND is generated. RD FOUND causes MBOX RESP to be generated to inform the EBox that the requested word (selected from the quadword by PMA 34-35) is available on the CACHE DATA 00–35 lines.

b. The requested word is not in cache but at least one of the associated words of the quadword is valid. In this case the signal RD FOUND is not generated as it was in example a. [redacted]. The condition ANY VALID MATCH and -RD FOUND generates a core read request to bring in all the non-valid words of the quadword, and to update the use bits. The requested word is the first one sent by core and it is routed directly to the EBox. It is then written into cache and its valid bit is set. Any other requested words will be received from core in ascending modulo four order. As they become available, they are written into cache (via the MBs) and their valid bits are set.

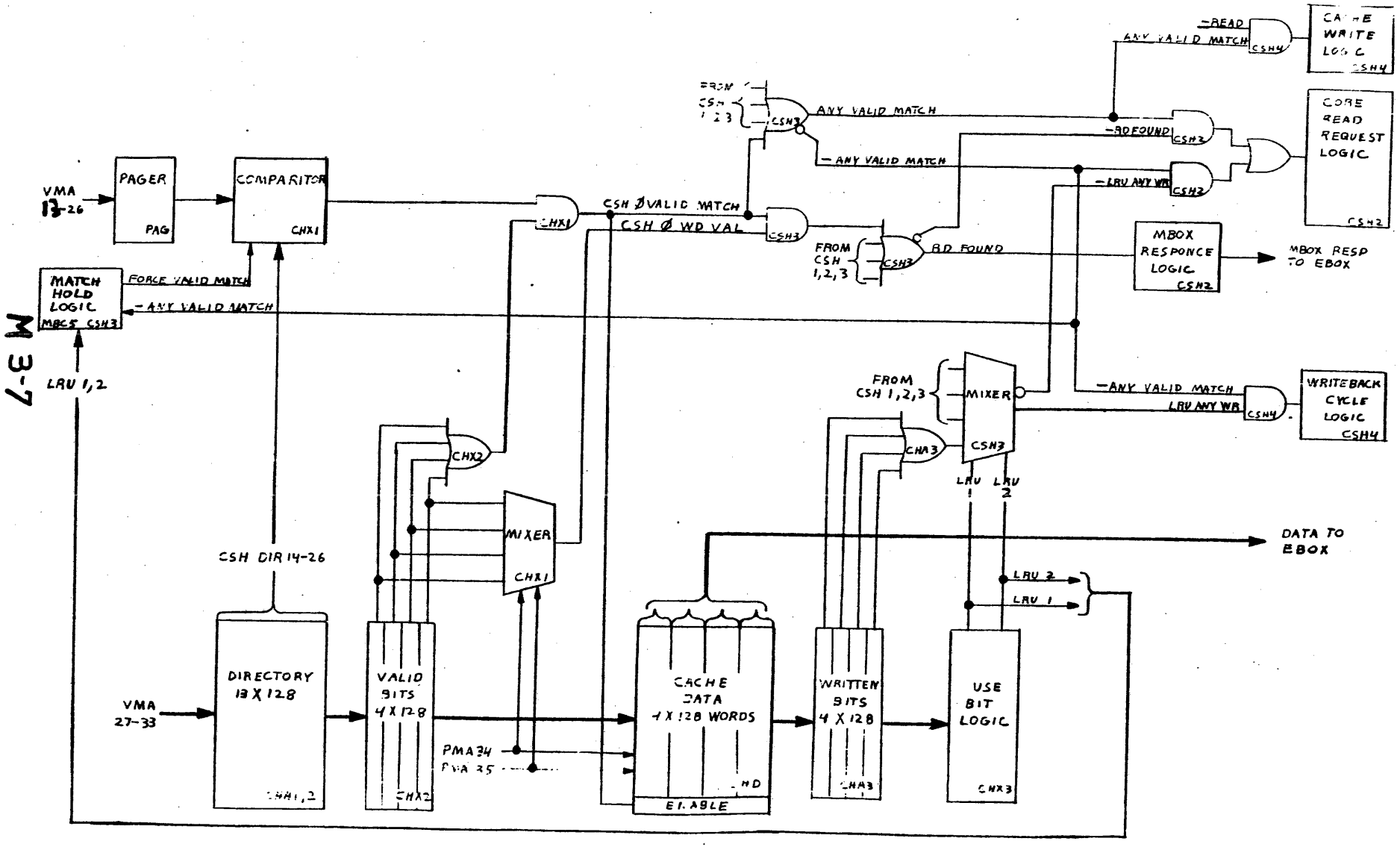


Figure 3-3 Memory Reference Decision Paths

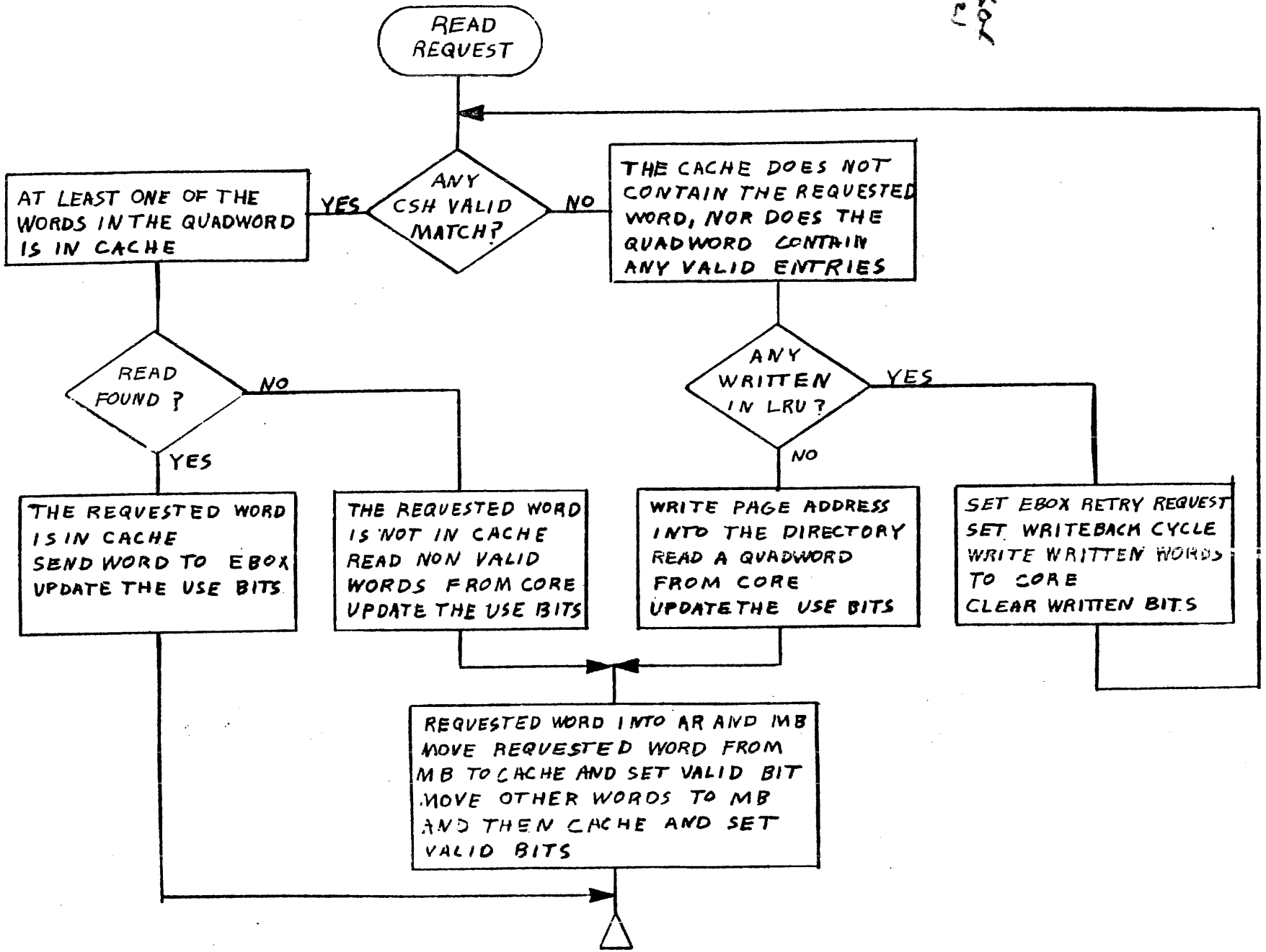


Figure 3-4 Read Operation Simplified Flowchart

c. The requested word and none of the associated words of the quadword are in cache. In this case a check is made to see if the addressed quadword in the least recently used (LRU) cache quarter has any of its four written bits set. If any written bits are set (LRU ANY WR) the EBox is told to retry the request (EBox RETRY REQ) and a cache writeback cycle is initiated. The writeback cycle writes the written words back into core and clears the written bits.

If no written bits are set (-LRU ANY WR) a core read request for the entire quadword is issued. The physical page address of the quadword is written into the directory of the LRU cache quarter and the use bits are updated. The requested word is the first one sent by core and is routed directly to the EBox. It is then written into the LRU cache quarter and its valid bit set. The other three words of the quadword are received from core in ascending modulo four order. As they become available, they are written into cache (via the MBs) and their valid bits are set.

3.2.3 Cache Write Operation (See Figures 3-3 and 3-5)
When a cache write operation is attempted the cache is accessed and one of two conditions will exist. These conditions and the action taken by cache control is as follows:

a. One or more words belonging to the quadword group that the EBox is attempting to write into is in cache and is valid (ANY VALID MATCH is generated). In this case the word is written into the addressed location, the valid and written bits are set and the use table is updated.

b. If the Cache does not have a record of the quadword (address does not match and/or no valid bits are set) the least recently used (LRU) Cache Block is checked to see if any written bits are set. If none of the written bits are set, then the group is available for use. In this case the addressed word in the selected group is simply written, and the corresponding directory address and use bits are updated. The valid and written bit locations are also set. If one or more written bits are set, core must be updated before the LRU Cache Block can be used. Core is updated by initiating a writeback cycle. This cycle causes all written words in the LRU Block to be moved to the MBs and then to core. As each word is moved to the respective MB, the written bit for the word is cleared. After all words are on their way to core, the EBox request is retried. This time, no written bits will be set permitting this block to be used for the current request as described before.

3.2.4 Cache Use Logic (See Figure 3-6)
The Cache Use Logic consists of two RAMs plus control logic. One RAM contains the use information while the other RAM contains update information for the use table and is named the refill table. The refill table contains entries for all possible history combinations as a function of the four Cache quarters which turns out to be 128 entries. After the Cache is initialized for full Cache service, only 96 out of the 128 locations are required to provide the use history update information because 32 combinations are illegal. The 32 illegal combinations will only be encountered during initialization.

The refill Table is three bits wide and is structured into the following two fields:

a. ORDER: Bit 0

b. LRU: Bits 1 and 2

Collectively, the contents of the refill Table represents the refill algorithm of the Cache. The refill algorithm can be adjusted by changing the sequence of the bit patterns to bypass one, two, or three Cache quarters in any combination. Normally, the algorithm is set to use all four Cache quarters equally. Table 3-3 specifies the bit patterns and the sequence of these patterns for using all Cache quarters equally.

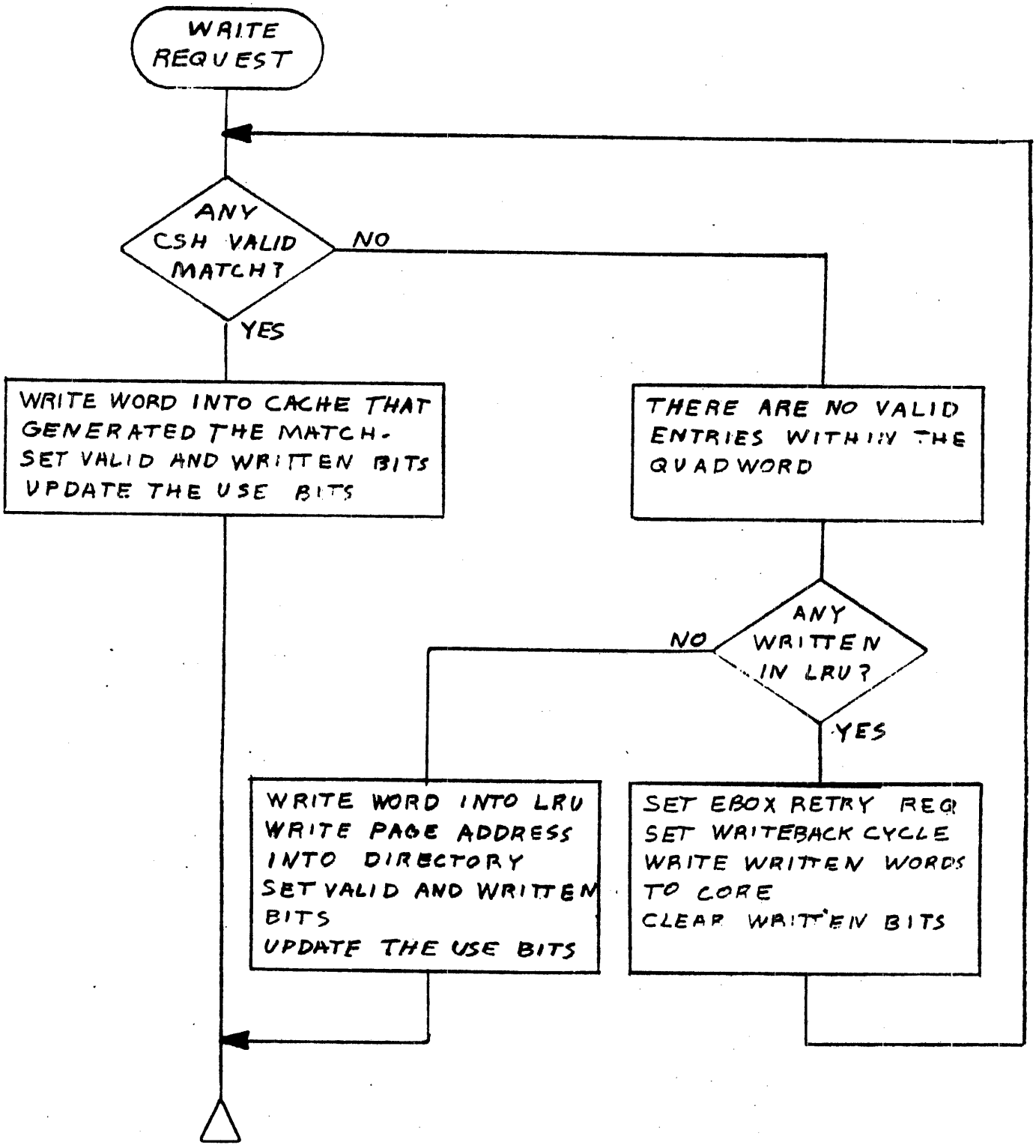


Figure 3-5 Write Operation Simplified Flowchart

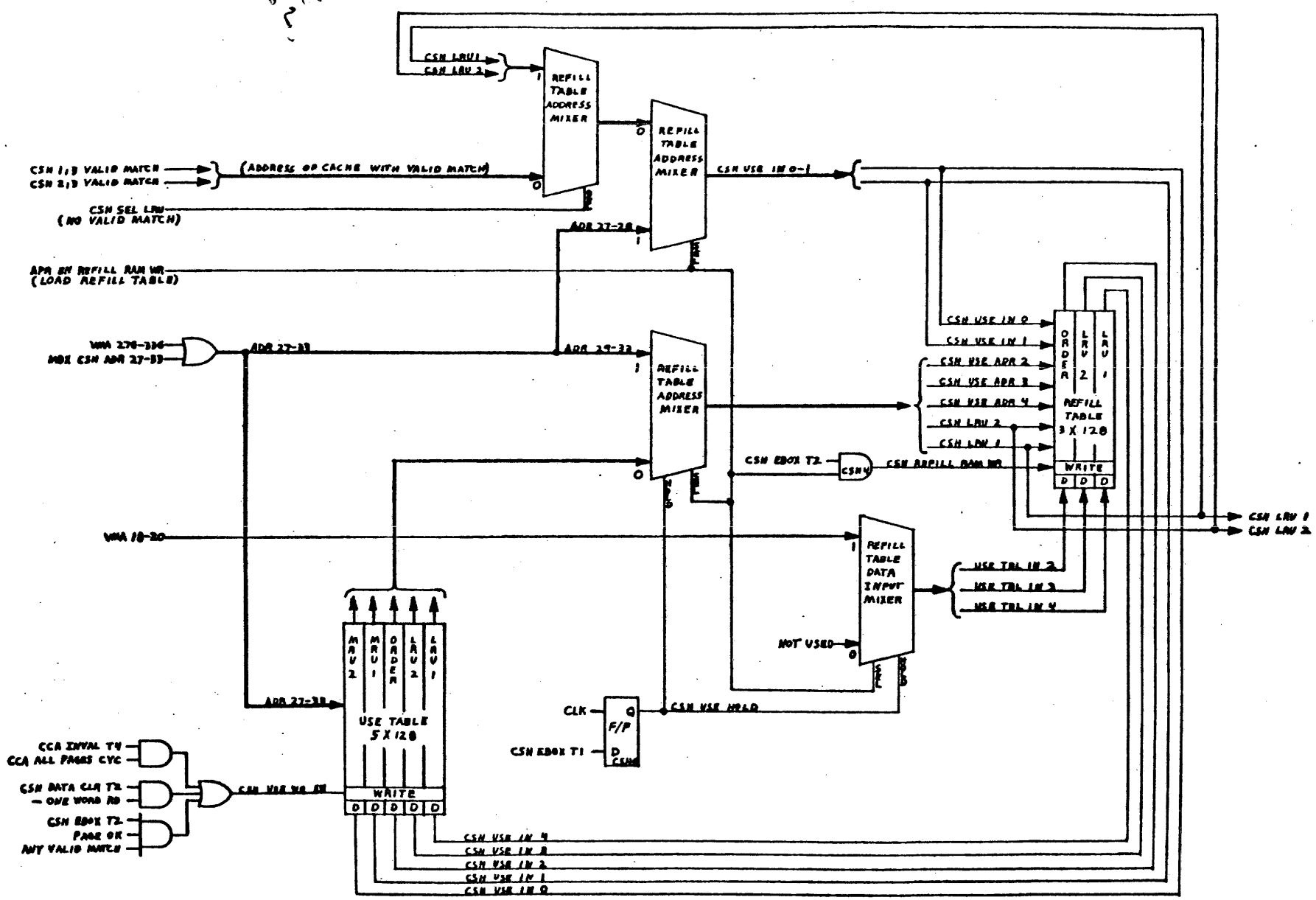


Figure 3-6 Cache Use Table Logic

Table 3-3 Cache Refill Algorithm (All Ca he Quarters)

OLD CONTENTS OF USE TABLE	ADDRESS OF CACHE QUARTER BEING ACCESSED			
	00	01	10	11
0 0 0 0 0	0 0 0	0 0 0	0 1 1	0 0 0
0 0 0 0 1	0 0 1	0 1 1	0 0 1	0 0 1
0 0 0 1 0	0 1 0	0 1 0	0 1 1	0 1 0
0 0 0 1 1	0 1 1	0 1 1	0 1 1	0 1 0
0 0 1 0 0	1 0 0	0 0 0	0 0 1	0 0 0
0 0 1 0 1	1 0 1	0 1 0	0 0 1	0 0 1
0 0 1 1 0	1 1 0	0 1 0	0 0 1	0 1 0
0 0 1 1 1	1 1 1	0 1 1	0 1 1	0 0 1
0 1 0 0 0	0 1 1	0 0 0	0 0 0	0 0 0
0 1 0 0 1	0 0 1	0 0 1	1 1 1	1 0 1
0 1 0 1 0	0 1 0	0 1 0	1 1 1	1 1 0
0 1 0 1 1	0 1 1	0 1 1	1 1 1	1 1 0
0 1 1 0 0	0 1 0	1 0 0	0 0 0	0 0 0
0 1 1 0 1	0 0 1	1 0 1	0 0 0	1 0 1
0 1 1 1 0	0 1 0	1 1 0	0 0 0	1 1 0
0 1 1 1 1	0 1 1	1 1 1	1 1 1	0 0 0
1 0 0 0 0	1 1 1	0 0 0	0 0 0	1 0 0
1 0 0 0 1	0 0 1	1 1 1	0 0 1	1 0 1
1 0 0 1 0	0 1 0	1 1 1	0 1 0	1 1 0
1 0 0 1 1	1 1 1	1 1 1	0 1 1	1 0 1
1 0 1 0 0	0 0 1	0 0 0	1 0 0	1 0 0
1 0 1 0 1	0 0 1	0 0 0	1 0 1	1 0 1
1 0 1 1 0	0 1 0	0 0 0	1 1 0	1 1 0
1 0 1 1 1	1 1 1	1 1 1	1 1 1	1 0 0
1 1 0 0 0	1 1 0	1 0 0	1 0 0	0 0 0
1 1 0 0 1	1 0 1	1 1 0	1 0 1	0 0 1
1 1 0 1 0	1 1 0	1 1 0	1 0 1	0 1 0
1 1 0 1 1	1 1 1	1 1 0	1 1 1	0 1 1
1 1 1 0 0	1 0 1	1 0 0	1 0 0	1 0 0
1 1 1 0 1	1 0 1	1 0 0	1 0 1	1 0 1
1 1 1 1 0	1 1 0	1 1 0	1 0 0	1 1 0
1 1 1 1 1	1 1 1	1 1 0	1 1 1	1 1 1

M Box 3-63

During normal operation the refill table is addressed by the contents of the use table in conjunction with a two bit code that specifies the cache directory that yielded a match or by a two bit code that specifies the LRU cache quarter if no match occurred. When the refill table is loaded (DIA EN REFILL WR RAM) the table is addressed by address bits 27-33 which are the same as those used to address the cache directory. After the refill table is loaded by the EBox at power up, a Cache SWEEP instruction to invalidate the entire cache must be executed by the EBox to initialize the directories and the use table. In the Cache Directory, all valid and written bits are cleared; in the Use Table all entries are initialized to reflect the refill algorithm for full Cache service. This purges all illegal bit patterns from the table. The illegal patterns for full cache service are those where the contents of the MRU field is the same as the contents of the LRU field of the use table. After the use table is initialized it will contain "00 0 11" in every location indicating that the order of use of each cache quarter is 3, 2, 1, and 0.

3.2.4.1 Use Table Operation (See Figure 3-6) – The use table is updated during a CSH EBOX CYC that is executing an EBOX READ or WRITE request for which the cache is to be used. If the cache contains a valid entry (ANY VALID MATCH), even though the desired word may not be in the cache (-RD FOUND) the Use Table is updated by asserting WR USE BITS. If the cache does not contain a valid entry (-ANY VALID MATCH), the use table is updated by asserting WR USE BITS one clock tack after CSH DATA CLR T2 is asserted. The major difference between these two cases, besides the timing, is the way the refill table is addressed and the data for the MRU field of the addressed use table location is derived. For the case where a valid entry is found, ANY VALID MATCH is asserted which causes ANY VAL HOLD latch to be set. This inhibits the CSH SEL LRU gate to make sure that the two high order bits of the refill table address and the data for the MRU field of the use table is a two bit code that identifies the cache that yielded the valid entry. For the case where a valid entry is not found in the cache, ANY VALID MATCH is not asserted which causes ANY VAL HOLD to remain cleared. This enables the CSH SEL LRU gate and selects the two high-order bits of the refill table address for the data for the MRU field of the Use Table. These two bits are a two-bit code that identifies the LRU Cache.

3.3 PAGER

3.3.1 General

User programs reference instructions and data via virtual addresses. These addresses are not absolute (physical core addresses) since any given page can reside anywhere in core when the program is running. The monitor determines where the entire program will reside and will also, if a contiguous segment is not available, assign core on a page by page basis. Therefore, since user programs are allocated core dynamically, the transformation from virtual address to physical address must be performed dynamically. As the monitor assigns core to a user program, the user process table and associated page tables are created to specify where in physical core the user program resides. This information is specified on a page-by-page basis. Then, when a given user program is given time to run by the scheduler, paging data is transferred from the user process table to the hardware tables in the Pager. The hardware tables include a page table and a directory.

The page table storage area contains 512 locations that store a 13 bit address plus five access bits. These storage locations are divided into four sets. Each set is identified as to section number, user or executive status, and validity by a 128 location directory.

As the running programs reference memory the pager is filled. Eventually the pager will have enough entries to eliminate the need for further references to memory for paging information. At this point, all virtual addresses can be transformed by the entries in the hardware tables providing the running program confines itself to one section.

NOTE

Conflicts between User and Exec addresses or section addresses can cause thrashing when switching address space.

When a given user program runs out of time, all entries in the hardware tables are invalidated by setting the NOT VALID bits in the directory table and the procedure is repeated for the next scheduled program.

The Pager transforms the virtual page address into a physical page address and checks the page access bits every time the EBox makes a paged read or write request.

The transformation, essentially, is the replacement of the virtual section and page address with the physical page address. Both pager tables are automatically filled as virtual addresses are referenced by the user program. These entries are then used to determine if the entries are valid and if so to use the desired entry (addressed entry) as a replacement for the virtual section and page number.

If the Pager does not contain a valid entry, one of two courses of action can take place. With KI-10 Style Paging the MBox starts a page refill cycle to fetch four words (8 entries) from the process table and then retries the request. If after refilling the Page Table, the request cannot be honored because of the state of the access bits, the EBox is informed that a Page Fail condition occurred. The EBox must then take an alternate course of action and retry the request. With KL-10 Style Paging the MBox clears the addressed Page Table location and informs the EBox that a page fail condition occurred. The EBox must then calculate the physical page address, write the address into the Page Table and retry the request.

3.3.2 Modes of Operation

The pager is designed to operate in either the KI or KL paging mode. The paging mode is selected by the EBox which asserts or negates the 'CON KI10 PAGING MODE' E/M Box interface line. When operating in the KI paging mode, the page refills are executed by the MBox. This is accomplished by using the page pointers that are contained in the user and executive base registers. Extended addressing (sectioning) is not implemented thus the users program is limited to 256K of core storage. In the KL paging mode, refills are handled by the EBox when requested by the MBox. The EBox calculates the physical page address, writes it into the page table, and then retrys the request.

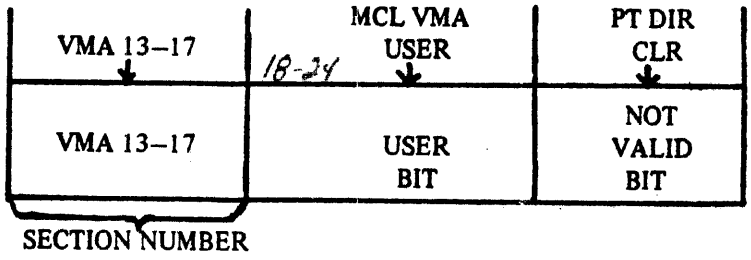
3.3.3 Pager Functional Description

The pager consists of two hardware tables, associated address, enable, and write drivers, and logic for detecting illegal page references (See Figure 3-7). One table serves as a directory and the other as the page table entry storage area. The directory contains 128 locations for storing virtual section numbers and the page table contains 512 locations for storing physical page numbers. Each directory entry implicitly identifies four page table entries. These tables also contain status bits to identify valid entries and access privileges. If the virtual section address matches the contents in the directory and the NOT VALID bit is cleared, then the corresponding four entries in the page table are current for the running process and may be accessible and legal for transforming the virtual address into the physical address.

3.3.3.1 Addressing – When the EBox makes a paged memory reference the page table and its directory are addressed by a function of the virtual User/Executive section and page address. EBOX USER, and bit 17 of the virtual section address are Exclusive-ORed with bit 19 and 20 of the virtual page address. This modifies bits 19 and 20 of the PT address as a function of the section address and the User/Executive mode. This modified page address is used to distribute the entries in the table as is shown in figure 3-8 and prevents identical page entries from different sections from occupying the same table locations.

In the KI Mode references outside of section 0 will not occur. The directory table is addressed by the seven high order bits of PT address (PT ADR 18–24) and the Page Table is addressed by all nine PT address bits (PT ADR 18–26). Therefore, for a given virtual address, one directory entry and one Page Table entry are selected. When the pager is addressed by the EBox a comparison is simultaneously made to determine if the directory entry (virtual section address) is valid, and the same as the virtual section address presented by the EBox. The User bit is also checked to see if it matches the current status of the processor.

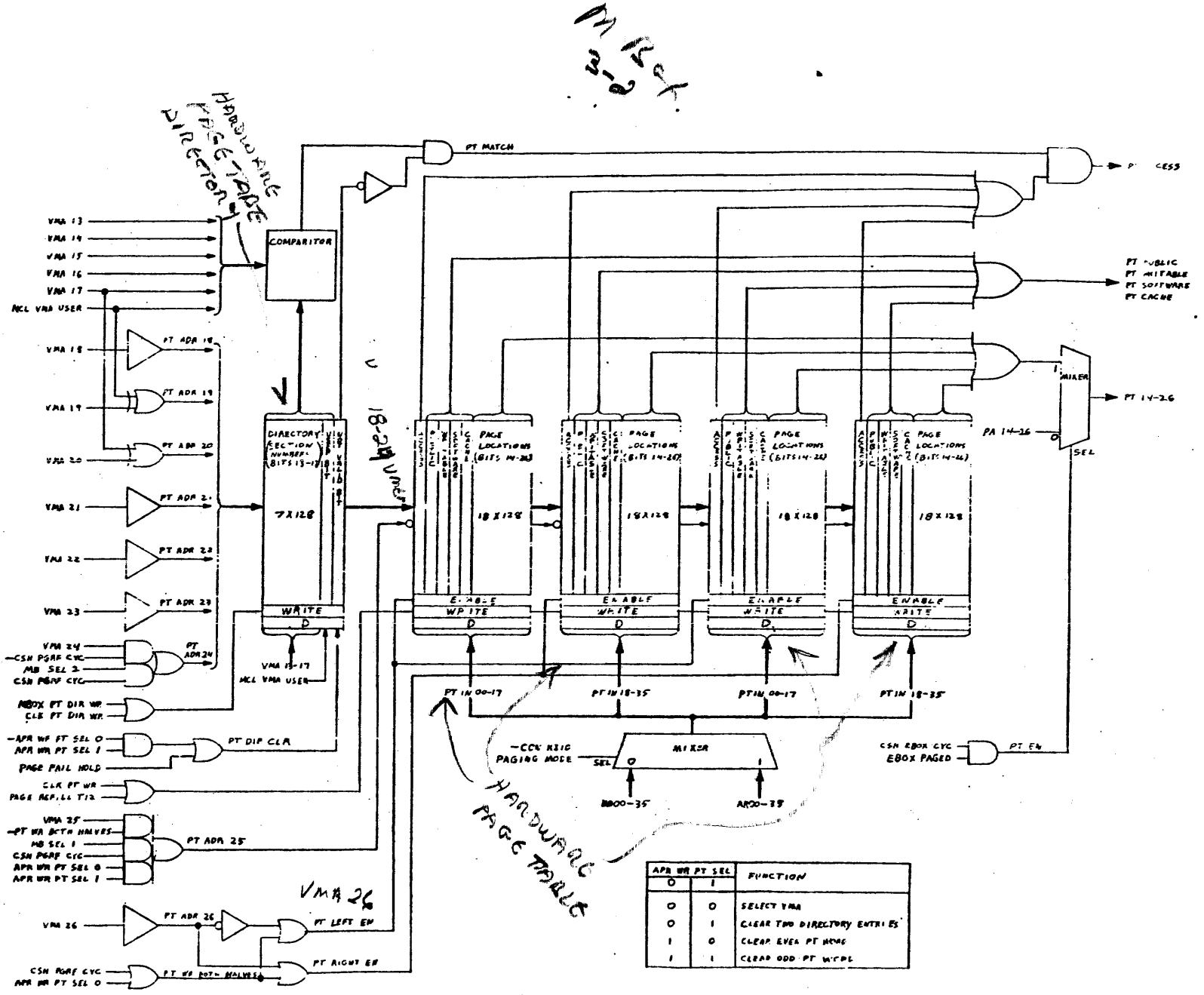
3.3.3.2 Page Table Directory Entries – Each entry in the page table directory consists of 7 bits in the following format:



- SECTION NUMBER – Each of the 128 directory entries identifies the section to which the four corresponding entries in the page table belongs. When operating in the KI paging mode, the section number is always zero.

- USER BIT – When the user bit is set, the corresponding entries in the page table are identified as belonging to the user program. When the user bit is cleared, the entries pertain to the executive program.

- NOT VALID BIT – The 128 not valid bits are set (two at a time) by the monitor whenever a new user is granted running time in the processor. As valid entries are brought in from the process tables in core and stored in the hardware page table, the NOT VALID BIT for those entries is cleared.



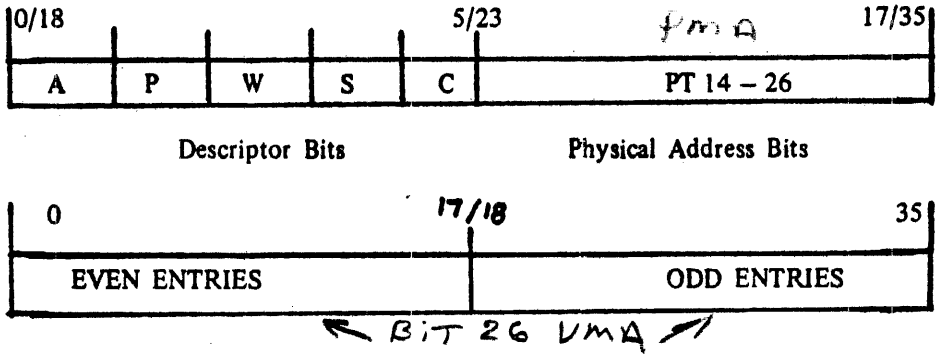
APR WR PT SEL 0	APR WR PT SEL 1	FUNCTION
0	0	SELECT VMA
0	1	CLEAR TWO DIRECTORY ENTRIES
1	0	CLEAR EVEN PT WORD
1	1	CLEAR ODD PT WORD

Figure 3-7 Pager Simplified Logic

PAGE TABLE ADDRESS (OCTAL)	ACTUAL ADDRESSED LOCATION (OCTAL)			
	EVEN SECTIONS 0, 2, 4, 6, 8, etc.		ODD SECTIONS 1, 3, 7, 9, 11, etc.	
	EXECUTIVE	USER	EXECUTIVE	USER
000–077	000–077	200–277	100–177	300–377
100–177	100–177	300–377	000–077	200–277
200–277	200–277	000–077	300–377	100–177
300–377	300–377	100–177	200–277	000–077
400–477	400–477	600–677	500–577	700–777
500–577	500–577	700–777	400–477	600–677
600–677	600–677	400–477	700–777	500–577
700–777	700–777	500–577	600–677	400–477

Figure 3-8 Page Table Address Hash Functions

3.3.3.3 Page Table Entries – The first five bits (ACCESS, PUBLIC, WRITEABLE, SOFTWARE and CACHE) of each Page Table entry are page descriptor bits that specify what type of entry (what kind of page) it is. These bits along with the physical address are transferred from the core table to the hardware table when the user program references a page that does not have a valid entry in the directory. Each entry in the page table consists of 18 bits that are layed out in the following format:



PHYSICAL ADDRESS BITS (PT 14–26)
These bits reference the actual page location that the virtual page number received from the E-Box has specified.

DESCRIPTOR BITS

These bits define the type of page, set up control functions and together with the request qualifiers from the E-Box determine whether a reference by the user or executive program is legal. The function of these bits is as follows:

- ACCESS BIT (PT ACCESS) This bit is set by the Monitor when the user program is allocated running time and brought into core storage. If the bit is not set, the corresponding page is still in mass storage or not accessible at all.

- PUBLIC BIT (PT PUBLIC) When this bit is set, the page is identified as being accessible from the public mode. If the bit is reset, the page is accessible from the concealed mode and access from the public mode will cause a page failure unless it is made via a portal instruction. Referencing a portal instruction causes the processor to switch from public to concealed mode. The concealed mode remains in effect until a reference is made to a public area.

- WRITABLE BIT (PT WRITABLE) This bit must be set in order to write into the page. If a write operation is attempted while the bit is reset, a page fail will occur. The writable bit is typically used to protect programs shared between users.

- SOFTWARE BIT (PT SOFTWARE) This bit is set when a page that is writable is written into. Therefore, absence of a software bit identifies pages that have not been written into and consequently do not have to be transferred to mass storage when the user is swapped out.

- CACHE BIT (PT CACHE) When this bit is set, the page will be maintained in cache memory. When cleared all words from the page will be read from and stored in core memory. This bit is reset by the Monitor to permit two processors to share the data contained in a page.

These bits along with the request qualifiers presented by the EBox are used to determine whether a given reference by the user program or executive program is legal.

3.3.3.4 Paging Checks – Each time the E-Box makes a paged reference to memory, the pager tests the request to see if it is legal and checks to see if there are any hardware failures. The signal PAGE OK is generated and sent to the cache control logic if any one of the following types of references are made under the given conditions.

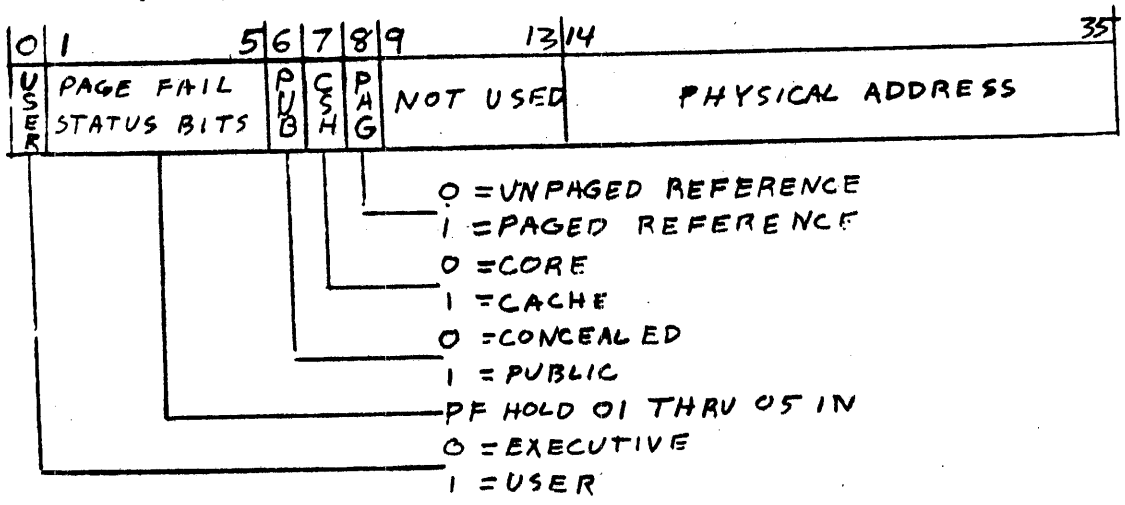
- A reference using either the user or executive base register is made and the previous instruction was not fetched from a proprietary area (not an illegal entry).

- An executive or user paged reference is made and the page entry is located in the table. It is accessible, public, and is writable or not being written into. The previous instruction was not fetched from a proprietary area and the page table parity is correct (odd).

- An executive paged reference is made and the page entry is located in the table. It is accessible, is not being written into, the previous instruction was not fetched from a proprietary area, and the page table parity is correct. This case applies to references to concealed (not public) pages. Remember the supervisor is allowed to read from concealed pages but not write into them.

- An executive unpaged reference is made. Since unpaged executive references are no longer permitted, this gate is functionally inactive.

Whenever a page test fails, the PAGE FAIL line is raised and five page fault bits (PF HOLD 01–05 IN) are assembled into a status word that describes the nature of the problem. If the pager is operating in the KL paging mode, the PF E-BOX HANDLE line may also be raised at this time. The PAGE FAIL signal is sent to the cache control logic where it enables the page fail time states. These time states transfer page fail status words, four control bits, plus the physical address to the EBus register. The EBox is then notified of the problem via the PAGE FAIL HOLD line. The EBox can examine the word in the EBus register, evaluate the failure, and take the appropriate remedial action. The page fail status word has the following format:



The conditions that cause PAGE FAIL to be generated and the associated Page Fail status bits are listed in Table 3-4.

Table 3-4 Page Fail Conditions

PAGE FAIL CONDITION	PF EBOX HANDLE	PF HOLD XIN 01 THRU 05
A paged reference was made in the KL paging mode and the directory did not contain a valid entry.	1	10001
A paged reference was made and the access bit was cleared. This indicates that the page is not in core and a reference to mass storage is required.	X	00WS1
The previous instruction was fetched from a proprietary area and was not a portal instruction. The E-Box is forcing a page fail by asserting MCL PAGE ILL ENTRY.	0	10001
The E-Box has detected an address break condition and is forcing a page fail by asserting MCL PAGE ADDRESS COND and MCL PAGE ILL ENTRY.	0	10011
The E-Box made a non-instruction paged write reference (PAGE TEST PRIVATE) to an executive page. If the page was concealed (-PT PUBLIC) page fail was generated.	X	10001
The E-Box made a non-instruction paged reference to a user page (PAGE TEST PRIVATE). If the page was concealed (-PT PUBLIC) page fail was generated.		
A refill error occurred during a paged reference. This condition is caused by a hardware failure. The paging hardware could not find the virtual page listed in the page table or the page map in core.	0	10010
The E-Box attempted a write operation and the page was not writable.	X	01001
During a paged reference a page table parity error was detected.	0	10101

X = 0 – KI Paging Mode
1 – KL Paging Mode

W = 0 – Not Writable
1 – Writable

S = 0 – Has not been written into
1 – Has been written into

T = 0 – Read
1 – Write

3.3.4 Operation

3.3.4.1 KI Style Paging – When the EBox issues a request to read or write paged memory it also asserts EBOX KI PAGING MODE. This allows the MBox to automatically refill the page table when required.

The page table must be refilled when a valid entry is not found in the directory. When the page table needs to be refilled the Pager asserts PAGE REFILL and the cache control will then execute a Refill cycle. This cycle fetches eight page table entries (4 words) from the process table (executive or user depending on the EBox request qualifiers). If one or more of the needed words are in cache, these words will be taken from cache instead of core. In either case, the words are moved into the MBs. From the MBs the words are moved into the page table one at a time. During the Refill cycle the PT ADR bits 24–26 are modified to move the words from the MBs into the correct page table locations. VMA 26 is blocked to select both halves of the page table (PT RIGHT and LEFT EN are asserted). VMA 24 and 25 are blocked and are replaced with two bit codes that corresponds to which MB is selected (MB SEL 1-2). PT ADR 18–23 remains unchanged. The resulting PT address then changes only when another MB is selected to move another word into the page table. At the same time the page table is written, the directory is also updated. The directory is updated by storing the virtual section address and the state of EBOX USER and validating the entry. One entry is placed in the directory for every four words that are written into the page table. When all the words have been written into the page table the cache control retries the request.

If the directory contains a valid entry and the EBox requested a legal operation, the Pager asserts PAGE OK. This signal informs the cache control to simply transform the virtual section and page address into the physical address by selecting the address from the page table. The directory contains a valid entry if the NOT VALID bit is cleared, and the USER bit and the virtual address in the table matches the address and EBOX USER signal presented with the request. The reference is legal if page descriptor bits allow the request. Refer to paragraph 3.3.3.3.

If the directory contains a valid entry and the EBox requested an illegal operation, the Pager asserts PAGE FAIL. The page test logic of the Pager senses that the EBox requested an illegal operation by checking the page descriptor bits of the referenced page. When the Pager asserts PAGE FAIL, the cache control asserts PAGE FAIL HOLD to inform the EBox that the page test failed. The page fail status word is transferred into the EBus register (LOAD EBUS REG) to allow the EBox to read the word and evaluate the failure and take remedial action. The format of the page fail status word was discussed in paragraph 3.3.3.4.

3.3.4.2 KL Style Paging – When the EBox issues a request to read or write paged memory it issues the request with KI PAGING MODE negated. This prevents the MBox from exeucting the refill operation and forces the MBox to assert PF EBOX HANDLE and PAGE FAIL HOLD in the event a valid entry is not found in the page table. PF EBOX HANDLE is asserted by the MBox only when the EBox specifies the KL PAGING MODE. The MBox asserts PAGE FAIL HOLD but not PF EBOX HANDLE, only when an illegal reference in accordance with the page descriptor bits of the page table entry was made.

NOTE
The pager will never assert PAGE REFILL when the EBox specifies that the KL Paging Mode is to be used.

The page table must be refilled when a valid entry is not found in the directory or when an entry in the page table is not accessible (ACCESS bit is cleared). To refill the page table the pager asserts PAGE FAIL and PF EBOX HANDLE. The cache control then asserts PAGE FAIL HOLD and transfers the Page fail status word into the EBus register. The EBox recognizes that the page test failed because a valid entry was not found in the page table because of the fact that both PF EBOX HANDLE and PAGE FAIL HOLD was asserted. The EBox then issues a request to read the EBus register and the page fail status word is evaluated to determine what kind of refill operation is required.

If a valid entry is not found in the directory, the EBox will clear four entry locations in the page table. Bits 25 and 26 of the PT address are set up and EBOX PT WR is asserted by the EBox to select and clear the selected words. The EBox sets up the correct address by presenting a two bit code to the MBox via the EBOX PT WR SEL0 and 1 control lines. The codes and their functions are defined in Table 3-5.

Table 3-5 Page Table Write Select Codes

EBOX PT WR SEL		
0	1	Functions
0	0	Select VMA address
1	0	Clear even PT word
0	1	Clear two directory entries
1	1	Clear odd PT word

MBox 2-18

After the EBox has cleared the even and odd words in the page table, it issues process table read requests to fetch a valid page table entry. When a valid page table entry is found it is written (EBOX PT WR) into the page table. At the same time the page table entry is written, the virtual section address is written into the directory (EBOX PT DIR WRITE). During this operation the tables are addressed by virtual address bits 18–26 since the EBox presents a code of "00" on the EBOX PT WR SEL control lines. At this point, one of the four locations in the page table that corresponds to the validated entry in the directory will have an accessible entry so that the original request can be retried by the EBox.

If a valid entry is found in the directory, but an accessible entry is not found in the page table, the EBox will fetch the entry and write it into the page table as described above.

Before the EBox writes a page table entry it checks the W bit of the entry. If the EBox intends to read from this page and the W bit is set, it clears the W and sets the S bit before writing the entry into the page table. Consequently, a page fail condition will be sensed by the Pager if the EBox issues a write request for that page. When this occurs the EBox checks the PF code to see if the S bit is set. If the S bit is set the EBox clears the S bit, sets the W bit and writes the entry back into the page table. To indicate the page will be written the EBox also updates the core status table (CST). After these operations are done, the EBox retries the original request. This scheme speeds up swapping the program out to mass storage since only those pages that were written will be identified and swapped out.

If the MBox presents a page fail code other than those indicating a refill operation is required or the write test failed, the EBox will evaluate the failure and take appropriate remedial action.

The EBox can also clear the entire directory. This is done whenever another user program is started. To clear an entry in the directory the EBox sets up VMA address bits 18-23, places code "01" on the EBOX PT WR SEL 0 and 1 control lines and asserted EBOX PT DIR WRITE. The EBox must execute this operation 64 times to clear the entire directory.

3.4 PMA SELECTION

3.4.1 General

The address paths implemented in the MBox are shown in Figure 3-9. These paths are implemented to facilitate the formation of the appropriate cache and core addresses and to address the various RAMS in the MBox. The addressable RAMs include the page table, cache, use table, CCW buffer and CH buffer.

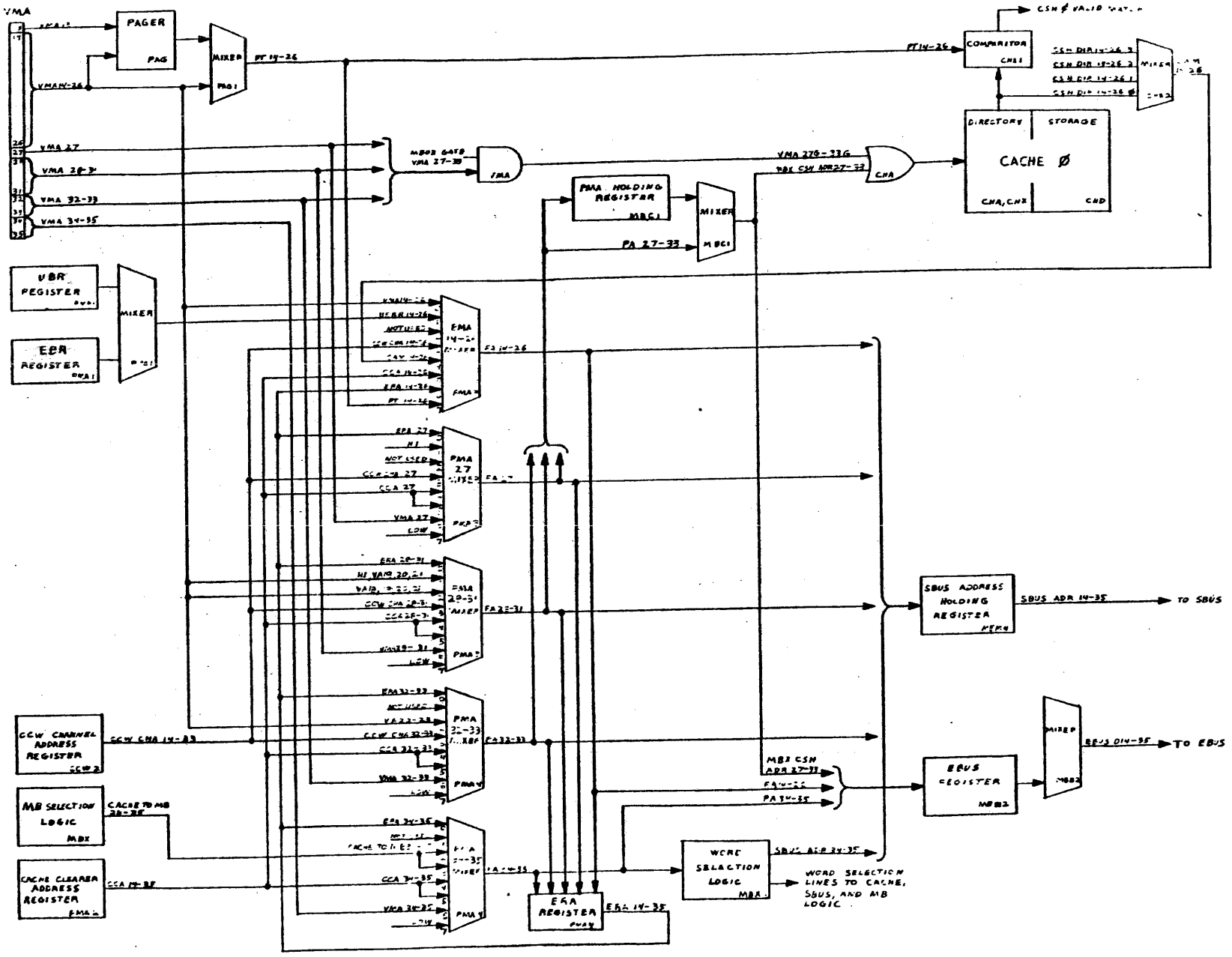


Figure 3-9 PMA Selection Simplified Logic

Any memory request, whether from the channel or from the EBox must be accompanied with an address. The address accompanying EBox Requests is supplied by the VMA. The CCW BUF provides the address when the channel makes a request. For EBox requests other than references to memory, the VMA serves as an address and/or data source. For example, the VMA serves as a data source when loading the UBR, EBR, or CCA and as an address and data source when loading the Cache refill table. The function of the elements involved in forming the physical memory address is as follows:

a. Physical Memory Address Mixer (PMA)

The PMA is a 22-bit eight-input mixer that receives various types of addresses for forming the desired physical memory address for a given cache cycle. The correct physical memory address is formed by the PMA under explicit control of the cache cycle control. The desired address mixture is selected and held when a particular cache cycle is started. This address is then used to address the cache, and core memory if a core cycle is started.

b. Page Table (PT) and Page Table Directory (PT DIR)

The page table contains 512 entries which are associated with (indexed by) entries in the page table directory. Each page table directory entry identifies four adjacent entries in the page table and therefore contains 128 entries. Both the page table and the directory are addressed by the virtual section and page address every time a cache EBox cycle is started.

c. User and Executive Base Registers (UBR and EBR) and Cache Clearer Address Register (CCA)

The UBR, EBR and CCA registers are loaded from the VMA. The contents of these registers are made available to the PMA so that the correct physical memory address can be formed by the PMA.

d. PMA Hold Register

The PMA hold register is loaded when a core read cycle is started. This address is then used to move the words coming in from the MBs into the cache. This address is held since the EBox can issue another request for a cache cycle after the first word comes in from core.

e. Cache Directory.

The cache directory contains one physical memory page address location for each corresponding quadword location in the Cache data buffer. This address is made available to the PMA so that the correct physical memory address can be formed by the PMA for a write back operation.

The cache directory is addressed by the VMA, PMA or the refill address from the PMA HOLD register depending on the particular cache cycle being executed as outlined in Table 3-6.

Table 3-6 Cache Directory Address Sources

Cache Cycle	Address Source
CSH MB CYC	PMA HOLD 27-33
CSH CHAN CYC	PMA 27-33
CSH EBOX CYC	VMA 27-33
CSH CCA CYC	PMA 27-33
CSH PAGE REFILL CYC	PMA 27-33
CSH WRITEBACK CYC	PMA 27-33

M-Box
2-61

f. Cache Address Mixer (CAM)

The cache address mixer is a 13-bit four-input mixer that provides the means for distributing the address from the appropriate cache directory quarter to the PMA during a writeback operation. The mixer is controlled by the CAM SEL 1-2 code which is a function of the cache quarter in which the written words are located.

g. Channel Command Word (CCW) Register and CCW Buffer

The CCW buffer contains the word count, address command list pointer (CLP) and status bits. The CLP or the address is transferred to the CCW Register and held when the channel issues a request so that the address can be selected by the PMA for distribution to the SBus.

3.4.2 Cache Addressing

The cache cycles that check the cache directory and/or the page table must allow for logic and RAM transit time. Approximately 90 nanoseconds (3 MBox clock ticks) are required from the time the address is presented to the page table and cache before their contents can be checked to decide the next step in the cache cycle. The page table is addressed by the VMA when the EBox issues the request. However, the cache address varies with the cycle to be executed and is presented to the cache when the cycle is started. A summary of the sources that contribute to forming the Cache address presented in Table 3-7.

Table 3-7 Cache Address Combinations

Cycle	Address Source	
	Cache Directory (27–33)	Cache Data (27–35)
MB	PMA HOLD	PMA HOLD + MB SEL 1-2
CHAN	PMA CHA	PMA CHA + CTOMB
EBOX	VMA	PMA VMA
CCA	PMA CCA	PMA CCA
REFILL	PMA QUADWORD POINTER	PMA QUADWORD WD POINTER + CTOMB
WRITEBACK		
CCA REQ	PMA CCA	PMA CCA
EBOX REQ	PMA VMA	PMA VMA + CTOMB

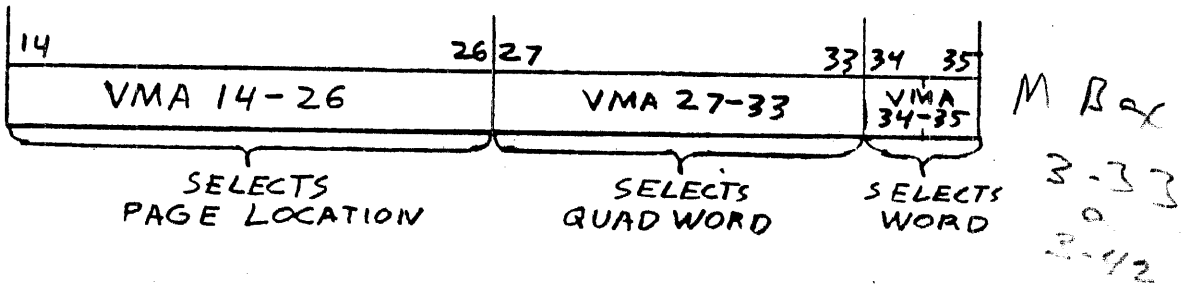
The Cache is addressed by the 9 least significant bits of the 22-bit physical address or the 23-bit virtual address. These bits point to a word within a page and are not subject to modification by the paging mechanism in the system since only entire pages can be relocated through the paging mechanism. All nine address bits are used to address the data portion of the cache while only the seven high order bits address the directory portion of the cache. This has the effect of addressing one data word in each cache and a directory entry for each cache. Consequently, if one of the directory entries matches the page address that was presented with the request and the valid bit for the word in the associated cache is set, then the desired word is in the cache. Note that only one word is addressed in each cache while the cache directory in conjunction with the valid bit of the word specifies which cache has the requested word.

3.4.3 Address Selection

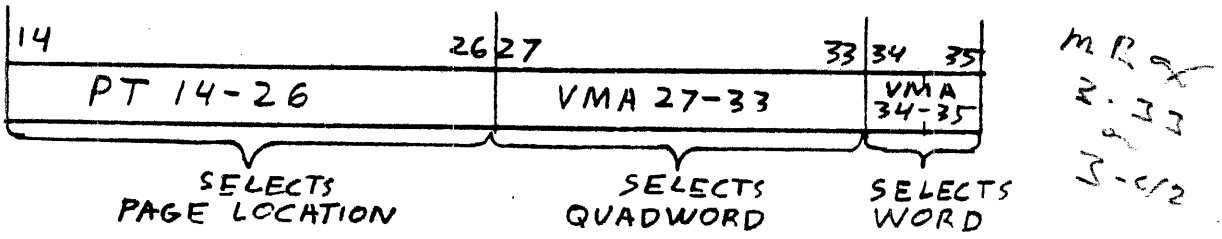
3.4.3.1 MB Cycle – During the CSH MB cycle the cache address is provided by the PMA HOLD register and the MB control. The seven high order bits of the nine bit cache address are supplied by the PMA HOLD register and the two low order bits are a function of which MB is selected (MB SEL 1-2) at the time. The PMA HOLD register is loaded when a core read cycle is started and is held for the duration of the cycle (CORE RD IN PROGRESS). The MB control provides the two low order bits of the cache address (MB SEL 1-2) to move the word into the correct location of the data portion of the cache. The contents of the PMA HOLD register and the MB SEL 1-2 control lines are selected (REFILL ADR EN) to address the cache every time a cache MB cycle is executed.

3.4.3.2 EBox Cycle – During the CSH EBOX cycle the cache address is provided by the VMA. The seven high order bits of the cache address are not passed through the PMA to minimize the transit time, thereby, permitting the cache control to check the contents of the cache directory somewhat earlier than would otherwise be possible. This consequently speeds up the CSH EBOX cycle.

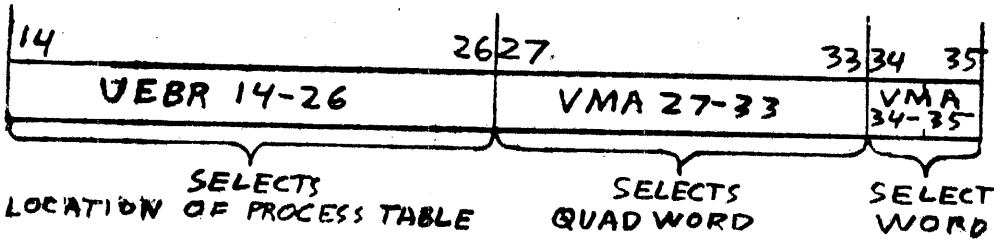
For an unpaged memory reference, the PMA simply supplies the VMA address unchanged as shown below.



In the case of a paged reference, the valid contents of the page table (the physical page address) is combined with (linked or concatinated) the virtual word address of the page as shown below.



For references to the process tables the contents of the User or Executive Base Register (UBR or EBR) depending on whether EBox UPT or EPT is asserted by the EBox is linked with the virtual word address of the referenced page as shown below.

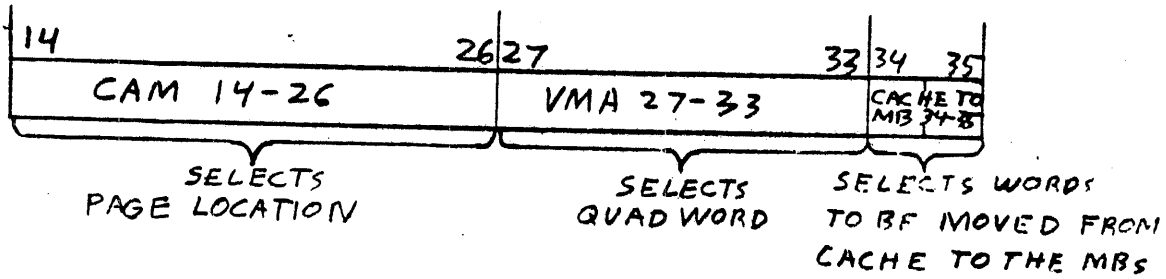


3.4.3.3 Writeback Cycle – During a cache writeback cycle the cache address is provided by either the CCA or the VMA depending on which request was granted. If an EBox Request was granted the cache writeback cycle is entered from a cache EBox cycle and the PMA control selects the VMA to address the cache. If a CCA Request was granted the cache writeback cycle is entered from a cache clearer cycle and the PMA control selects the CCA register to address the Cache.

Words written into cache by the EBox are written back to core before the contents of the LRU cache quarter is supplanted with a word(s) from another page. Written words in the cache are also written back to core when the EBox issues a command to clear the cache (SWEEP instruction to validate core). All words within the QUADWORD with their written bit set are written back to core in numerical order (0 → 1 → 2 and then 3). This is because the new word to be stored there is from a different page.

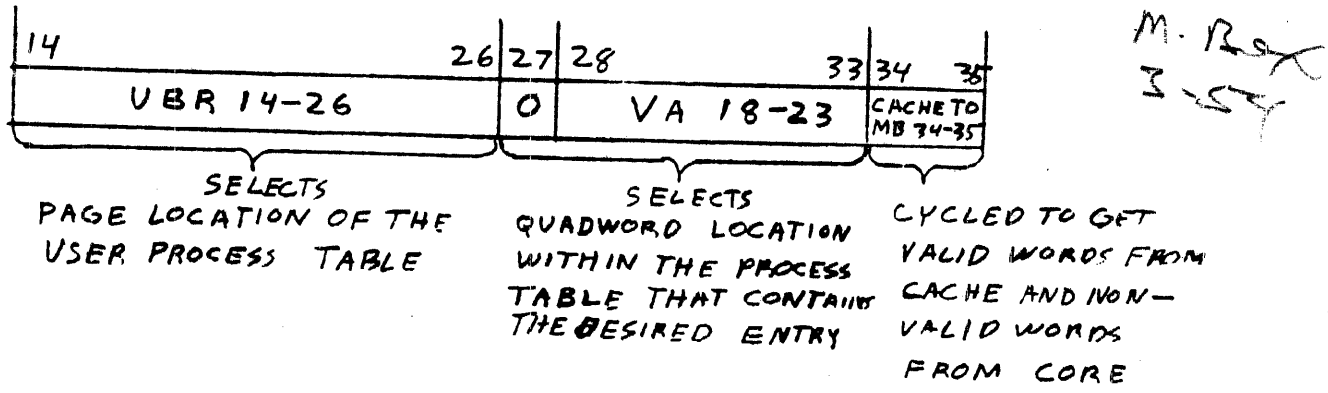
that is caused by an EBox Request

The address for a writeback cycle is assembled by the PMA as follows:



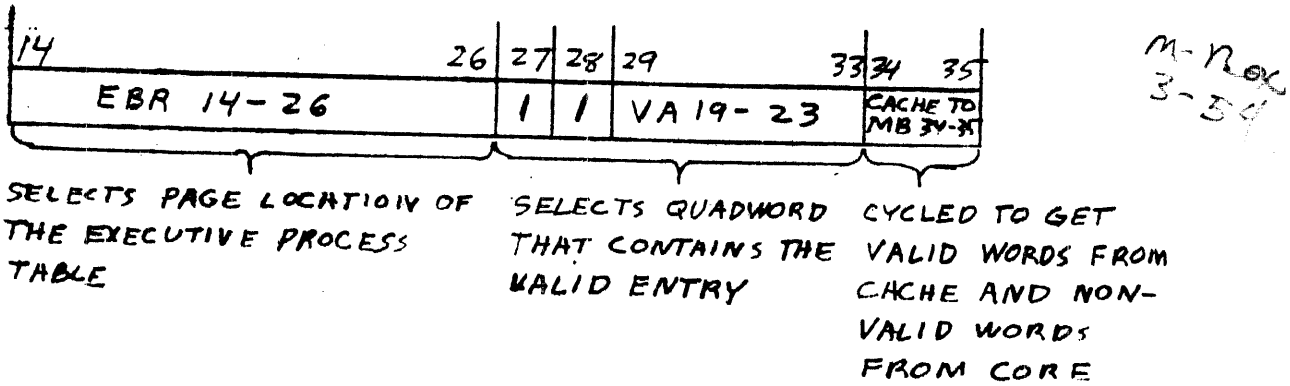
3.4.3.4 Page Refill Cycle (KI Paging Mode) – During a page refill cycle the cache address is supplied by the PMA. The seven high order bits constitute a quadword pointer into the process table (EPT or UPT) and the two low order bits are a function of which words in the cache are not valid. If some of the words in the cache are valid, the low-order two address bits are used to move the valid words into the MBs; (refer to MB control description) otherwise, these address bits are not needed. The address mix depends on whether the memory reference is to the user or the executive address space. If the memory reference is to the executive address space, the specific address mix also depends on whether the reference is to the "per process area" to the upper executive area or to the lower executive area. Consequently, depending on the state of EBOX USER (1 = User space; 0 = Executive space), and virtual page address (VMA 18–26), one of four possible addresses will be configured. Figure 3-10 shows the layout of the user and executive address space and their relationship to the process tables.

For the case where the EBox makes a memory reference to the user address space, all of which is paged, the SBUS address for the page refill cycle is configured as shown below.



UBR 14–26 points to the physical page in core that contains the user process table; VMA 18–23 points to the quadword in the process table that contains the page table entry of the referenced virtual page and cache to MB 34-35 and cycled to get any valid words from cache. Bit 27 of the SBus address is jammed to "zero" to select the lower half (locations 0–377_8) of the user process table which contain the 512 Page Table entries (two entries per location) for the user address space.

For the case where the EBox makes a memory reference to the lower executive address space (pages 000–337_8) the SBus address for the page refill cycle is configured as shown below.



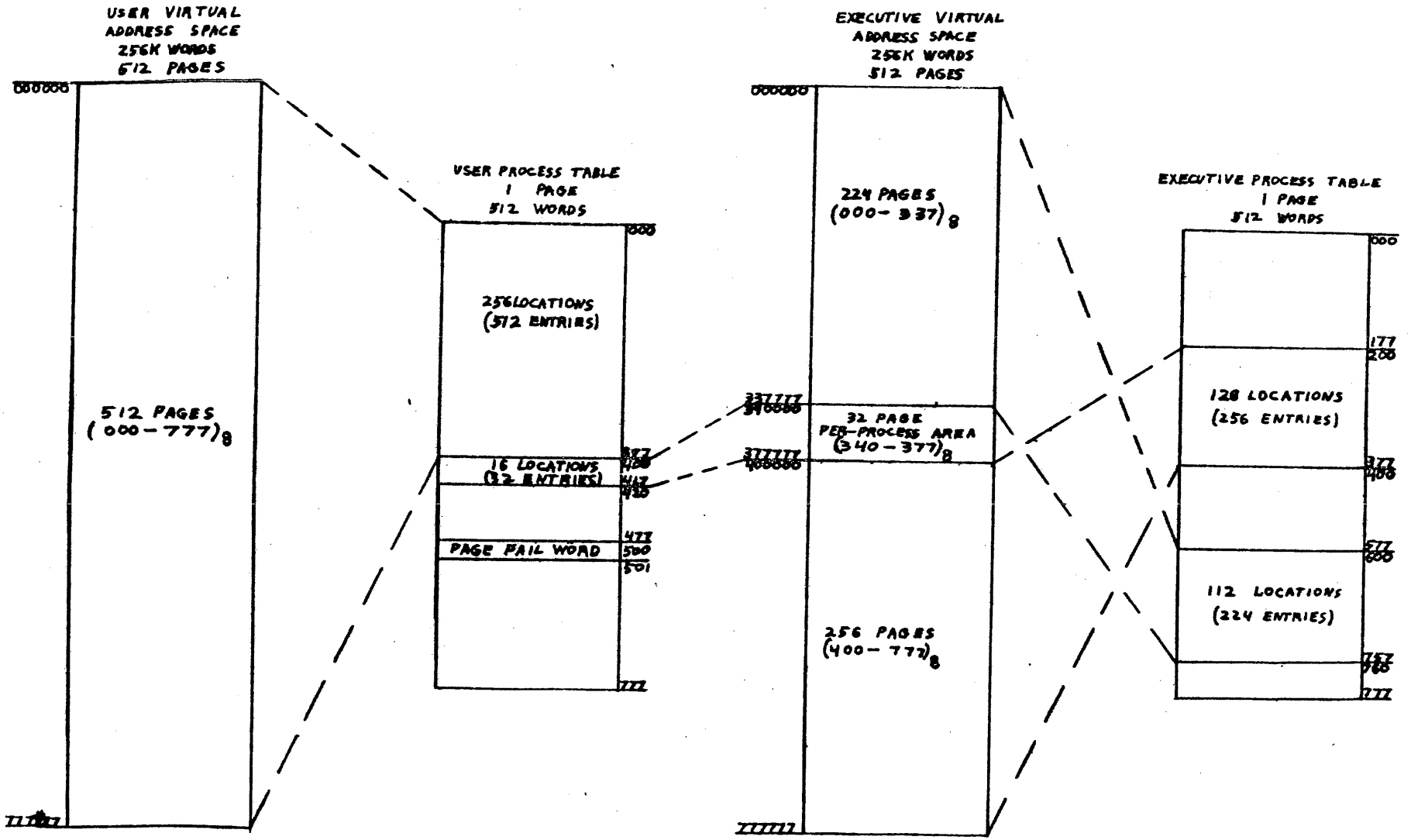
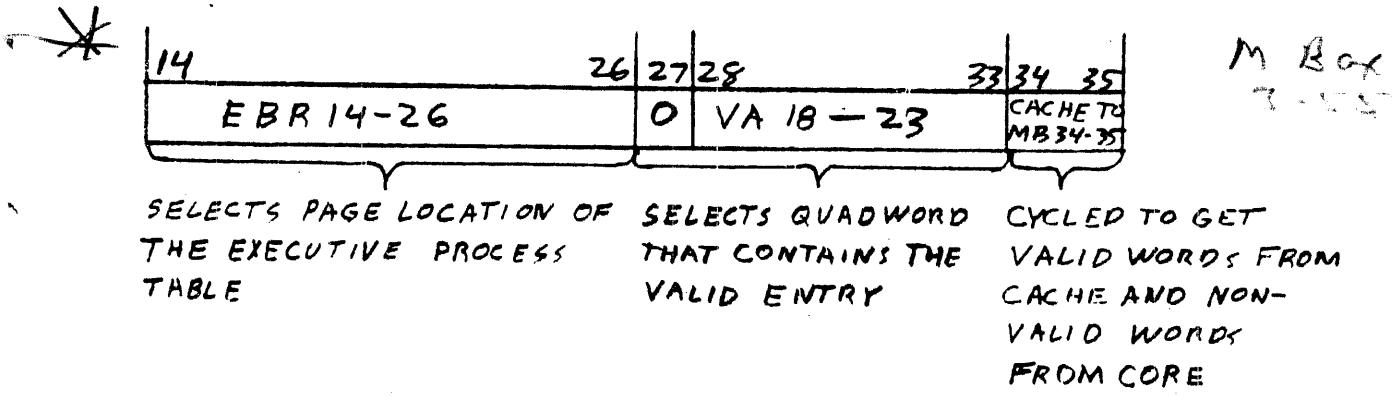


Figure 3-10 User and Executive Address Space Allotments

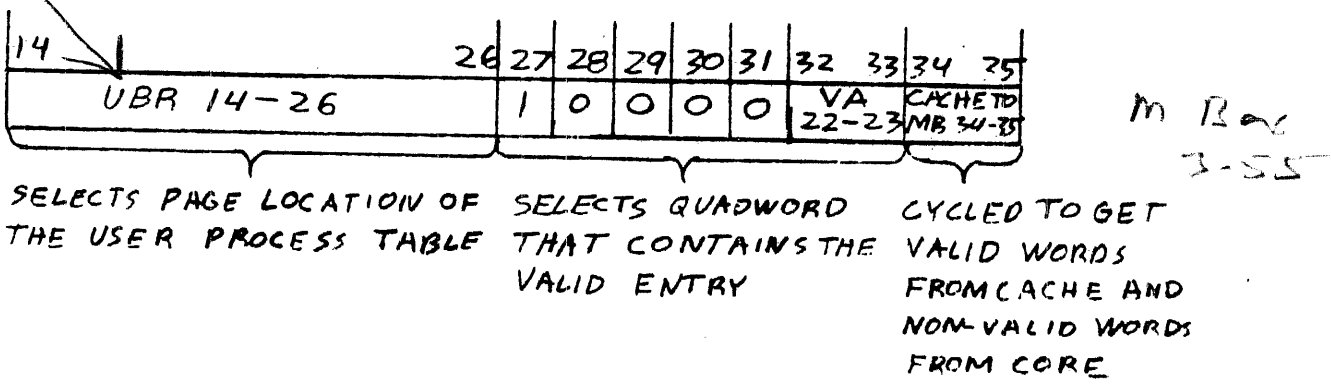
EBR 14–26 points to the physical page in core that contains the executive process table; VMA 19–23 points to the quadword location in the process table that contains the page table entry of the referenced virtual page; and cache to MB 34-35 are cycled to get any valid words from cache. Bits 27 and 28 are jammed to "one" to select the upper quarter (locations 600–777) of the executive process table of which locations 600–757 contain the 224 page table entries (two entries per location) for the lower executive address space.

For the case where the EBox makes a memory reference to the upper Executive address space (pages 400–777$_8$) the SBus address for the page refill cycle is configured as shown below.



EBR 14–26 points to the physical page in core that contains the executive process table; VMA 18–23 points to the quadword location in the process table that contains the page table entry of the referenced virtual page; and cache to MB 34-35 are cycled to get any valid words from cache. Bit 27 of the SBus address is jammed to "zero" to select the lower half (locations 000–377$_8$) of the executive process table of which location 200–377$_8$ contain the 256 page table entries (two entries per location) for the upper executive address space.

For the case where the EBox makes a memory reference to the paged executive address space defined to be the "per process area" (pages 340–377$_8$) the SBus address for the page refill cycle is configured as shown below.



DTE-20

1.1 INTRODUCTION

Each central processor in a KL10 system may have from one to four PDP-11 processors attached, each serving as a "front end" processor. Each PDP-11 is connected to the KL10 by a separate interface called the DTE20 Console Processor Interface, or simply the 10-11 Interface. The following are some of the possible front end functions:

1. Handling unit record equipment
2. Handling asynchronous communications equipment
3. Handling synchronous communications equipment
4. Providing a long term power line frequency clock
5. Diagnosing the KL10 Central Processor and other functional components in the system
6. Running a dedicated real-time data acquisition system
7. Bootstrapping the KL10 system.

In terms of basic features, the DTE20 generates parity for Deposit data and detects parity errors for both Examine data and byte transfers over the EBus. The DTE20 connects to the PDP-11 as a standard Unibus peripheral and communicates via interrupt or device address. Up to four DTE20s may be connected to a PDP-11. In a system consisting of four KL10 Central Processors, there may be four PDP-11/40 processors, where each processor can communicate with all KL10s in the system. It is possible to have up to four DTE20s on each PDP-11 in the KL10 system, and each KL10 processor may have 1, 2, 3, or 4 DTE20s connected to it via the EBus.

The DTE20 uses the NPR (Direct Memory Access) and BR (Vector Interrupt) features of the PDP-11. In addition, the DTE20 contains logic to detect PDP-11 core memory parity errors during NPR transfers, provided that the memory being accessed contains the parity option (MFU11 UP).

The DTE20 provides the following capabilities:

1. Console functions at Examine and Deposit, restricted or unrestricted.

2. Doorbell function, where the PDP-11 can interrupt the KL10 Central Processor and vice versa.

3. High speed simultaneous two-way transfer of variable byte data between the PDP-11 and KL10 memory.

4. Diagnostic bus for the PDP-11 to diagnose the KL10.

5. KL10-initiated bootstrap startup of the PDP-11 mechanism (diagnostic bus) to load th microcode into the CRAM, execute PDP-10 instructions, and start or stop the KL10 Cr tral Processor.

The following terminology will give some perspective on the front end and its relationship to the DTE20.

PDP-11 Communication Region – This region consists of an area of KL10 core memory defined by the deposit relocation and protection word in the Executive Process Table (EPT). This area is written by the PDP-11 using protected deposits, and read by the KL10. It is used for coordination of status, preparing for byte transfer operations, and passing limited amounts of data. Each PDP-11 in the system has a separate communication region in the KL10 memory, which it alone can modify.

KL10 Communication Region – This region is defined solely by the KL10 software and is separate from the PDP-11 communication region. It can be written by the KL10, but may be read by the PDP-11 using protected Examines. This area is used to coordinate status, prepare byte transfer operations, and pass limited amounts of data (Figure 1-1).

DTE 1-2

TABLES/MAPS
PAGE 20
28

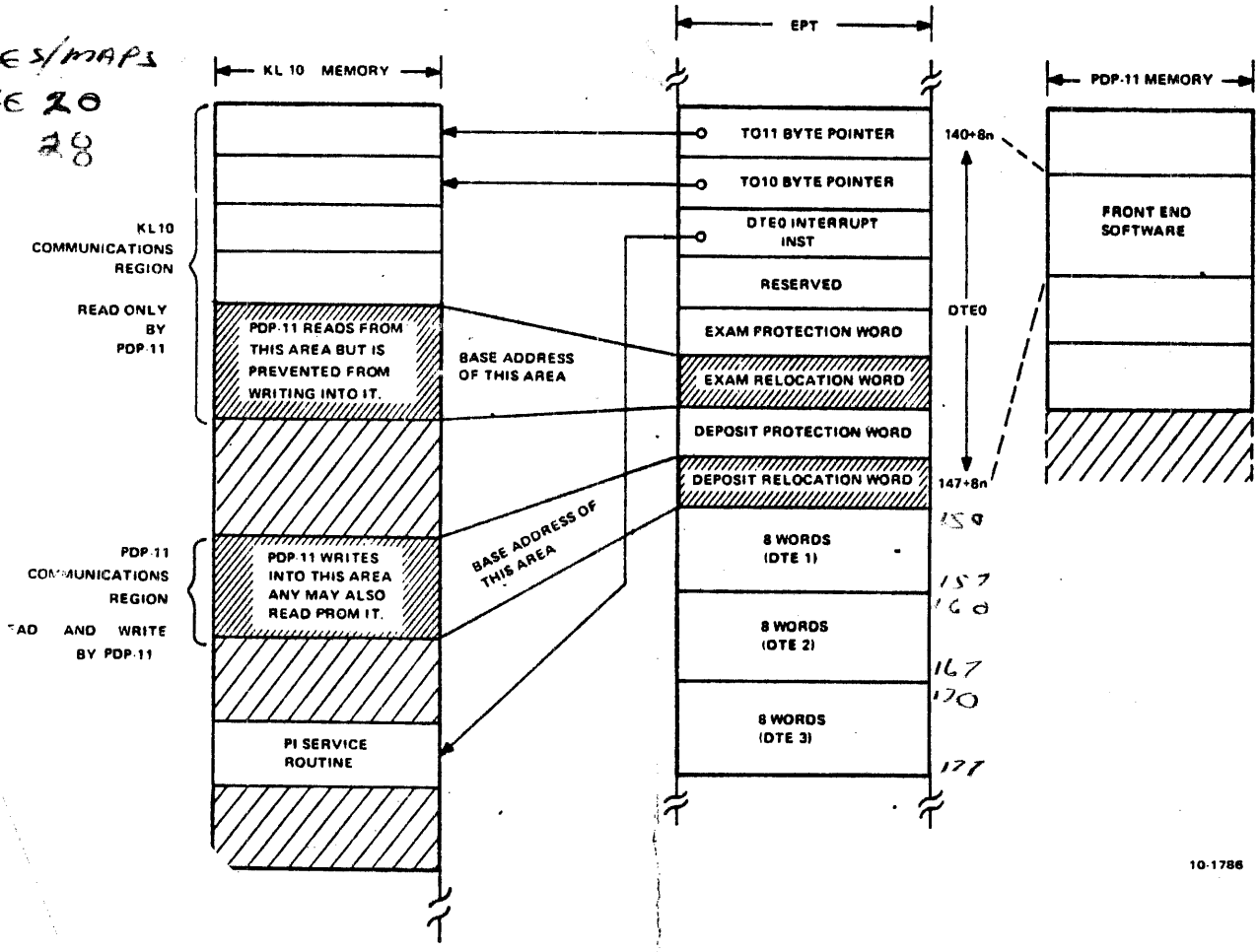


Figure 1-1 Overview Communications Region

D 1-2

Restricted Front End – A restricted front end is a PDP-11 system with a DTE20 that does not have diagnostic privileges. A restricted front end is prevented from using the diagnostic bus. A restricted front end can only access KL10 memory after the KL10 has performed a CONO (Conditions Out) to allow use of DTE PI0. After this has been done, the restricted front end can only examine or transfer bytes from the KL10 communication region and only deposit or transfer bytes to its own PDP-11 communication region.

Privileged Front End – A privileged front end is a PDP-11 attached to a KL10 via a DTE20 that can use the diagnostic bus and perform unrestricted Deposits.

Protected Examine or Deposit – A protected Examine or Deposit is an Examine or Deposit that is relocated and range checked by the KL10. The relocation and protection for Examine is separate from that of Deposit. A privileged front end can override the Examine and Deposit protection checks. A restricted front end cannot override these checks.

For addressing purposes, each controller is permanently assigned a unique device, or Controller Select (CS) code. A total of four Controller Select codes has been assigned because up to four controllers (interfaces) can be implemented in a KL10 system. Each interface is also assigned a physical number according to the physical slots in which the interface module will reside. These are indicated below. Both of these are hard-wired on the KL10 backplane. The specific Controller Select (CS) codes and physical number (n) assignments are as follows:

Interface	Controller Select (CS) Codes	Physical Number n_{10}
0	200	8
1	204	9
2	210	10
3	214	11

The device code is used to address the interface and the physical number is used to identify the interrupting interface.

Eight locations are assigned to each DTE20 in the KL10 Executive Process Table as follows:

Location	Name
140 + 8*n	To 11 Byte Pointer
141 + 8*n	To 10 Byte Pointer
142 + 8*n	DTE20 Interrupt Instruction
143 + 8*n	Reserved for DEC Hardware
144 + 8*n	Examine Protect Word
145 + 8*n	Examine Relocation Word
146 + 8*n	Deposit Protect Word
147 + 8*n	Deposit Relocation Word

NOTE: n = 0, 1, 2 or 3

TABLE PARS ← 20 +28

Figure 1-2, API Word Format, illustrates the basic format of this word. The DTE20 allows the software to set the following fields of this 36-bit word:

Address Space Field 0-2
Unused bits 11-12
Address Field 13-35.

DTE 1-4

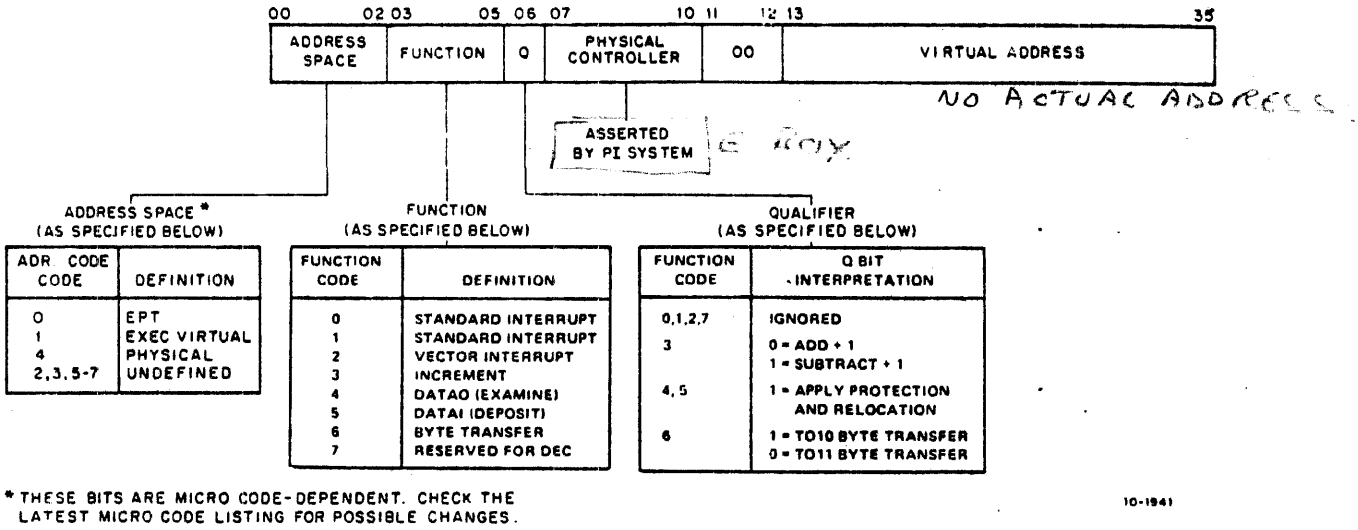


Figure 1-2 API Word Format

DTE 1-4

The Priority Interrupt (PI) board in the EBox supplies the physical controller number field [7-10]. The DTE20 asserts Qualifier (Q) bit 6, for all Examines and Deposits by a restricted front end, whether protected or not.

The DTE20 asserts Qualifier for all protected Examines and Deposits by a privileged front end and does not assert it if the privileged front end makes an unprotected Examine or Deposit.

1.2 BASIC PROGRAMMING OVERVIEW

To specify a 36-bit PDP-10 data word, three PDP-11 words are used. They are Deposit/Examine Data Word 1, Deposit/Examine Data Word 2, and Deposit/Examine Data Word 3.

To specify a 23-bit PDP-10 address, two PDP-11 words are used. They are Ten Address Word 1 and Ten Address Word 2. The high order part of Ten Address Word 1 is used for control. Ten Address Word 1 specifies whether an Examine or Deposit is to be done. For a privileged front end, the protect off bit in the Ten Address Word 1 can be set by the software to allow an unprotected Examine or Deposit. On unprotected operations, the space field specifies the type of address: Executive Process Table (EPT), Exec Virtual, or Physical Address, which may refer to core memory or ACs.

The Examine or Deposit function is started when the PDP-11 program writes the Ten Address Word. No program interrupts are generated on the KL10 or the PDP-11 side to signal completion of the Examine or Deposit. Therefore, the PDP-11 program must check for completion by looking at the status DEXDONE bit. The DTE20 clears DEXDONE when the PDP-11 writes Ten Address Word 2, so the software never needs to. Data in TENAD1, TENAD2, DEXWD1, DEXWD2, DEXWD3 remain intact after an operation. Therefore, the PDP-11 may perform repeated protected Examines or Deposits merely by writing the TENAD2 word each time. An Examine followed by a Deposit (changing only TENAD1 and TENAD2) will result in moving data from one KL10 core location to another. For unprotected operations, the PDP-11 must reload the protect off bit (PRTOFF) between each operation (Figures 1-3 and 1-4).

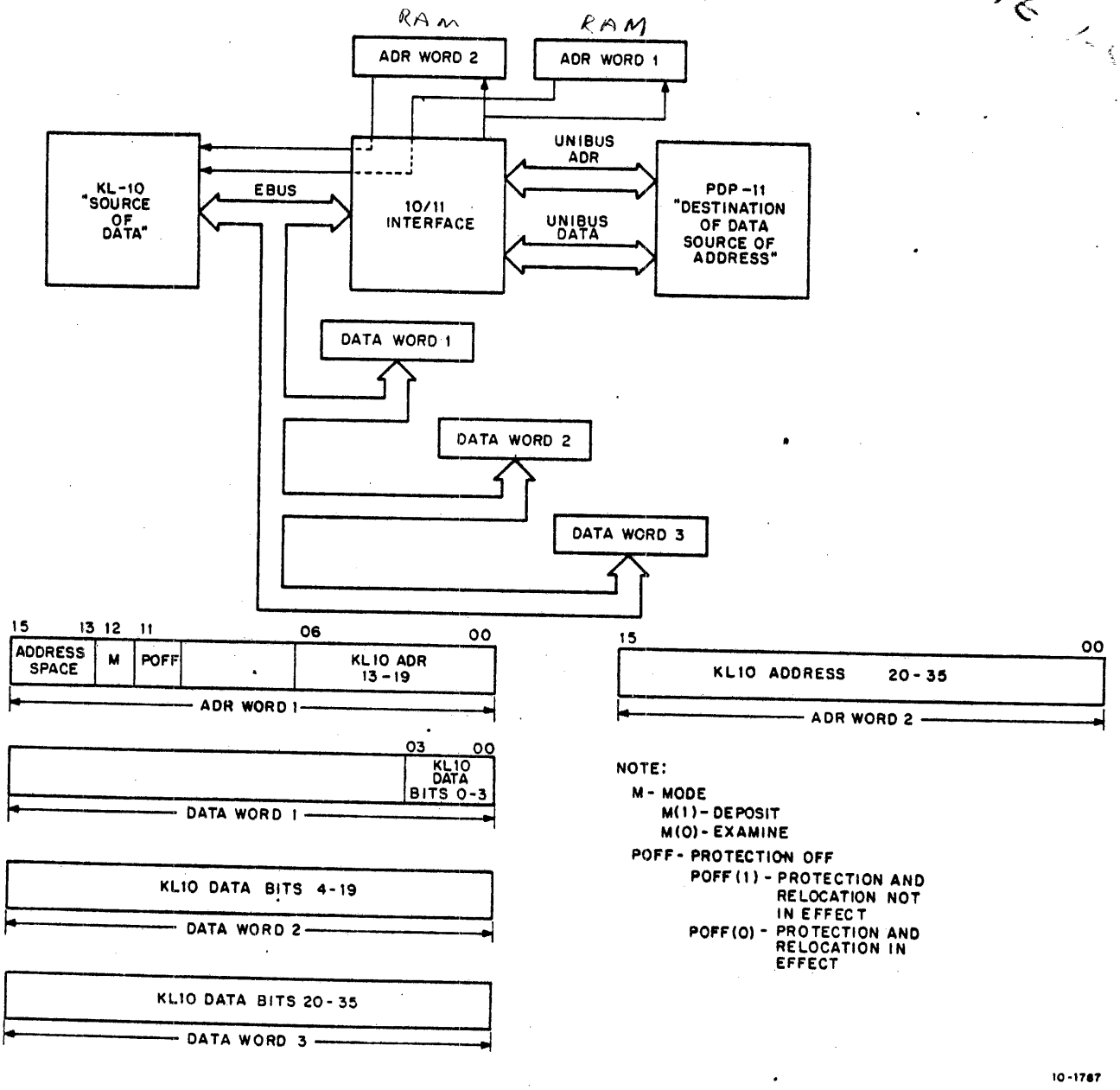


Figure 1-3 Examine Overview

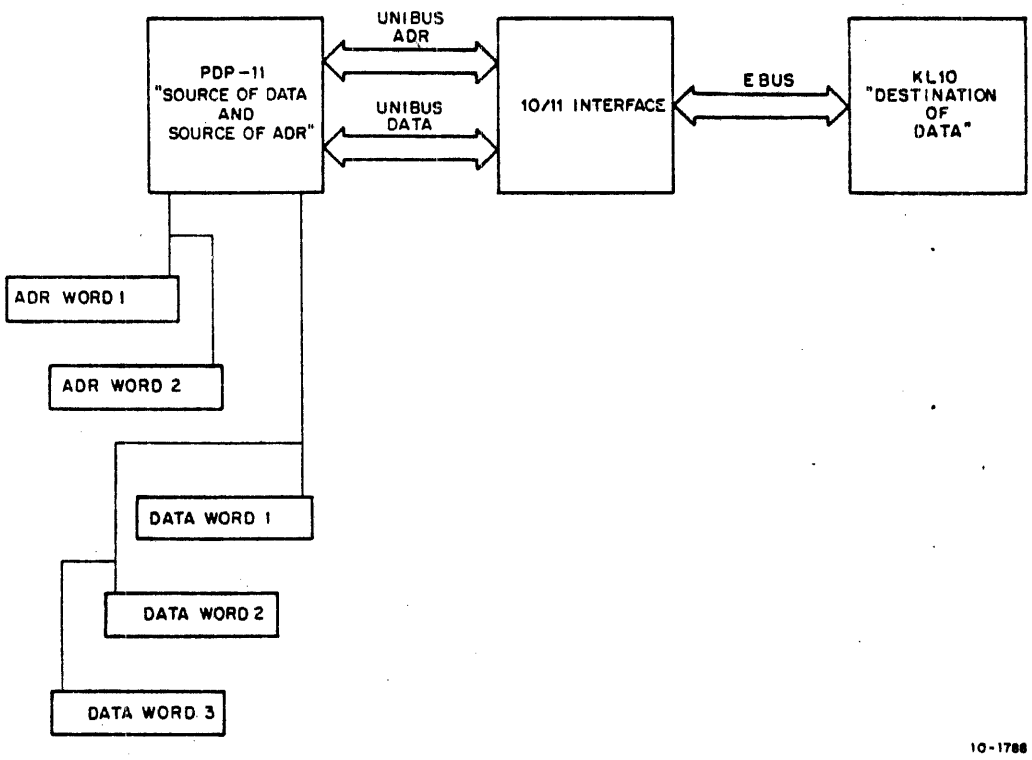


Figure 1-4 Deposit Overview
DTE 1-4

1.3 DOORBELL FUNCTION

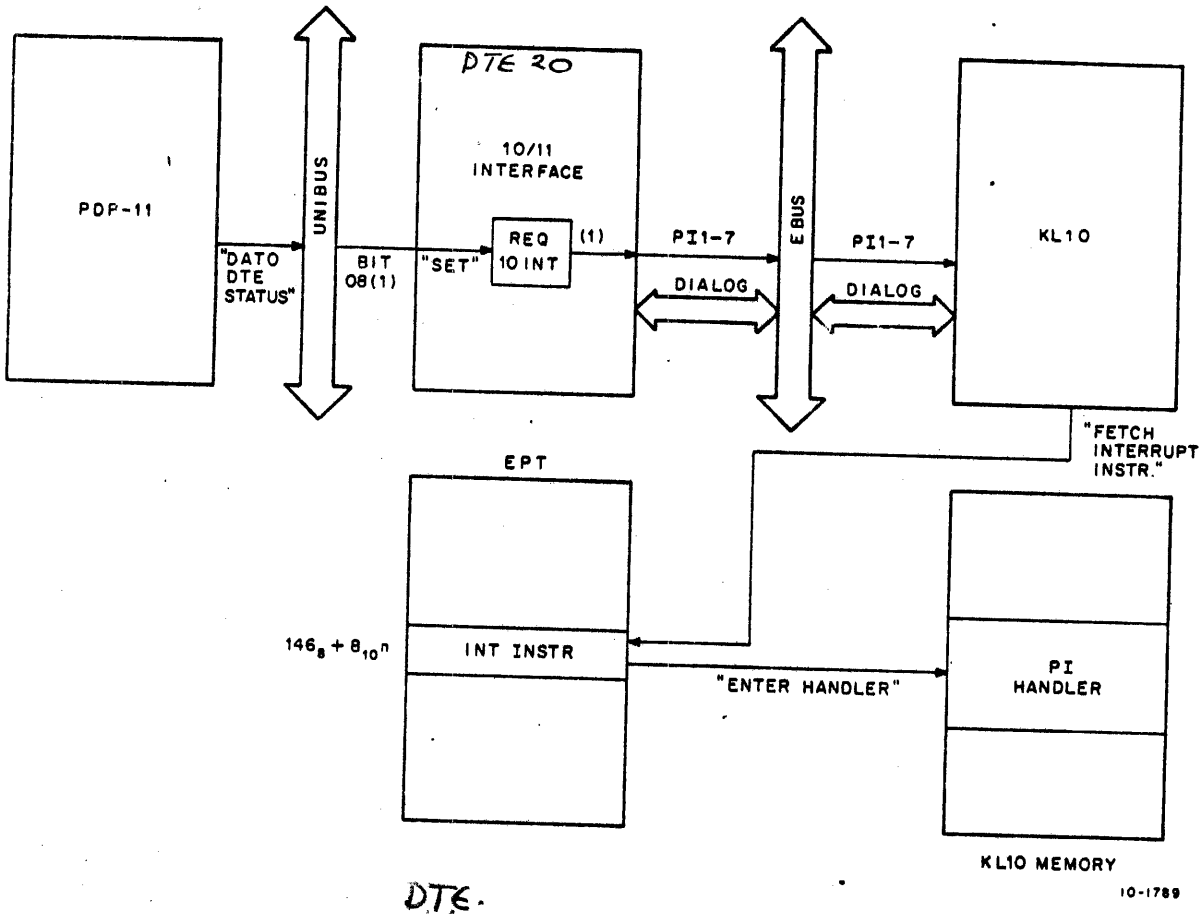
TABLE/MAPS 8

The doorbell function allows each KL10 to interrupt each PDP-11 connected by a DTE20 and vice versa.

The doorbell consists of a programmable interrupt and a status bit. In order for the PDP-11 to interrupt the KL10, the PDP-11 sets the request 10 interrupt flip-flop (bit 08) in the PDP-11 status word. When this bit is set, the DTE20 generates an interrupt in the KL10 with a status bit set in the CONI word (bit 26) indicating that the PDP-11 CPU has programmed an interrupt of the KL10 (Figure 1-5).

This procedure works in a reversed but identical manner for the KL10 interrupting the PDP-11. The KL10 sets the 10 requesting 11 interrupt by doing a CONO to the DTE20. The PDP-11 discovers the cause for the interrupt by looking at bit TO10DB (bit 11) in status. Communication is done via a word (or words) in the communication region in KL10 memory. A word (or words) is chosen and Deposit and Examine features are used by the PDP-11 to gain access to these words (Figure 1-6).

This mechanism is used by either processor to indicate to the other processor that it is powering down. For example, if the KL10 determines that its power is disappearing, it will set a bit in a word that is assigned for power failure notification. The KL10 then interrupts the PDP-11. The PDP-11, as part of its standard routine, will always check for the KL10 power fail bit in the communication region. In this way, the PDP-11 is notified that the KL10 power is disappearing. In a similar way the PDP-11 could interrupt the KL10 on every tick of the power line clock (50 or 60 Hz).



DTE.

Figure 1-5 PDP-11 Rings KL10 Doorbell

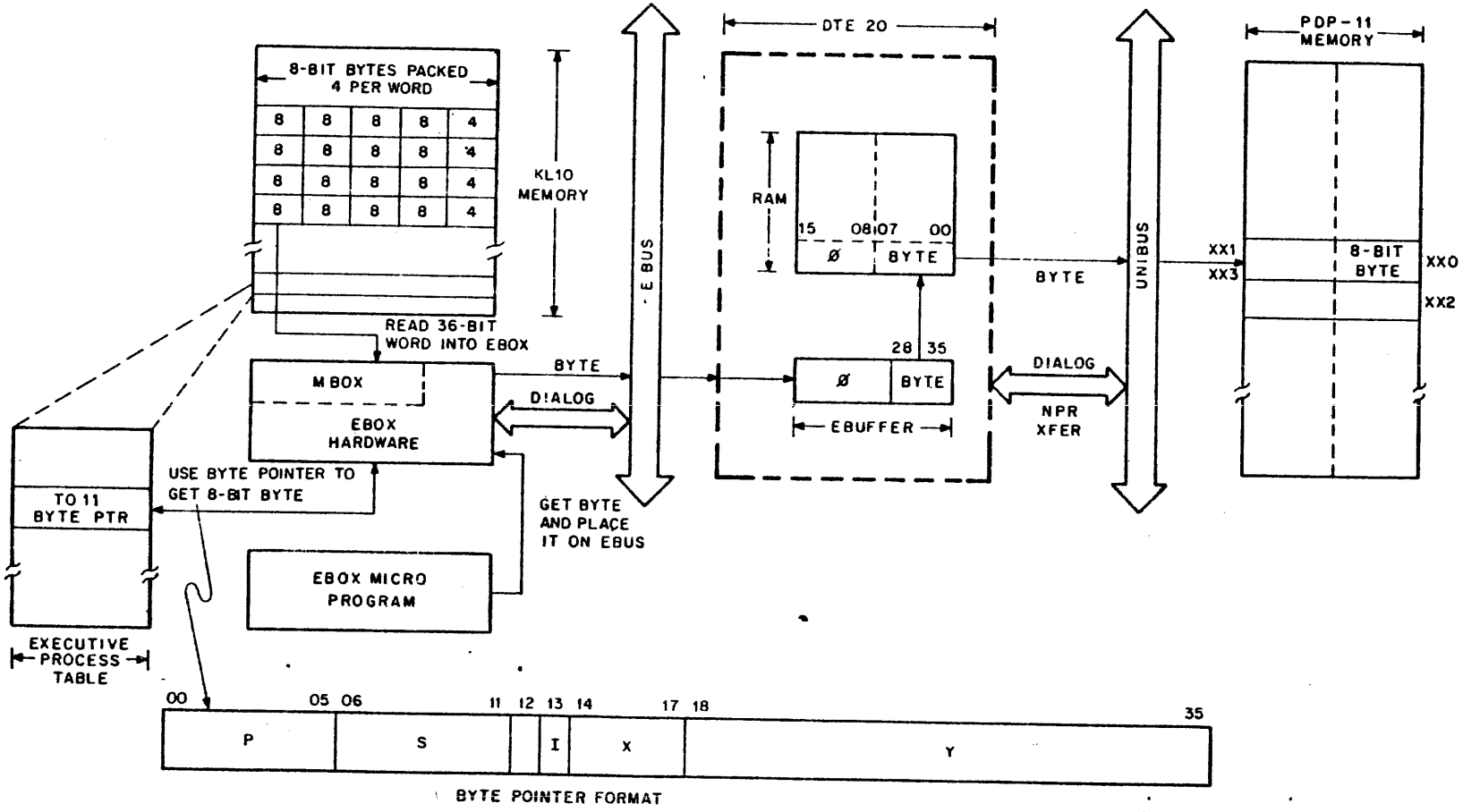


Figure 1-6 TO11 Byte Transfer Overview

DTE 1-8

8-1 D

1.4 BYTE TRANSFER FUNCTION

During the byte transfer function, the DTE20 transfers fields of information between the PDP-11 and the EBox. On the KL10 side, the fields are of variable length and are accessed through a PDP-10 byte pointer. On the PDP-11 side, the fields are either 8 bits wide and are stored in consecutive bytes or are 16 bits wide and are stored in consecutive words. If the field into which the information is being stored is narrower than the field from which it was read, as many of the rightmost bits as will fit are stored. If the field into which the information is being stored is wider than the field from which it was read, the information is right-aligned and padded with zeroes on the left.

To perform a transfer, the following actions must be done:

- The PDP-11 should specify the transfer rate (delay between transfers) and address bits 17-16 (this can be done once at system startup). If it is not specified, an undetermined transfer rate will occur to one of the four 32K memory regions.
- The PDP-11 must specify whether byte or word mode is to be used in the PDP-11.
- The sender must specify the address of the source string. The KL10 controls the address of the data either to or from the KL10 via byte pointers in the EPT. The PDP-11 controls the address on its side via two locations in the DTE (one word for each direction of transfer).
- The receiver must specify whether it alone (scatter write) or both CPUs are to receive normal termination interrupts (I bit = 1).

Information in the form of bytes may be stored in the PDP-11 as either one variable sized byte per PDP-11 16-bit word (1 to 16 bits of data) or one variable sized byte per 8-bit PDP-11 byte (1 to 8 bits of data). Byte addresses are specified in the KL10 using regular KL10 byte pointers in the EPT. Byte pointers are interpreted in Exec Virtual Address space.

CAUTION

The index field of the byte pointers should be zero. Otherwise, the EBox will index using the current contents of the Executive or User Index register at the time of the transfer. Indirection should not be used because the indirect word will not be incremented as with all byte pointer operations.

1.5 ERROR OVERVIEW

The DTE20 will generate/check parity on Deposit/Examine data (36 bits). It will not check or generate parity for CONI, CONO, DATAO, or API words. The software will check for errors by examining the termination words. The parity scheme also imposes one restriction on the byte pointer used for TO11 transfers. A byte size larger than 16 bits cannot be used unless the bits to the left of the rightmost 16 bits contain even parity. If a parity error occurs, the error termination bit status and the EBus parity error flag status will be set. If an Examine operation was in progress when a TO11 transfer operation has an error termination due to an EBus parity error, it is not possible for the software to determine if the Examine operation has a parity error. The EBus parity error is fatal, and is treated so by the Monitor. When a parity error occurs, the bad data is stored in the RAM and can be retrieved for error reporting. The DTE20 sometimes swaps the left and right bytes for byte mode prior to writing the bytes into the RAM. Therefore, the termination TO11 address word should be examined to determine if the left and right halves were swapped. If the termination address is even, the bytes were swapped. (This applies only to transfers in byte mode.)

1.6 DIAGNOSTIC OVERVIEW

The interface contains many features that enable diagnosing of the interface. It is designed to be diagnosed using three basic methods:

1. Without using or disturbing the EBus
2. With loopback on the EBus but without the KL10 or without the KL10 running
3. With the KL10 running.

The interface is primarily checked out in a single-step manner. Full speed operation may only be checked with a running KL10; DIAG1 contains the Diagnose 10/11 Interface bit. When DIAG1 is set, the following occurs. The interface clock is disabled and single step operation commences. Interrupts are inhibited from being sent to the KL10. The interface operates in the normal manner except that EBus operations never complete because no interrupts are issued to the KL10. Therefore, a bit is provided that enables setting EBus Complete, allowing the operation to continue.

The interface control is run by an up-counter and three decoders. The decoders are selected by the major state flip-flops. The up-counter is loadable by the rightmost four bits of DIAG Word 2. This enables any minor logic state to be executed. The major states are not loadable; however, they naturally cycle until a condition occurs that indicates the operation is ready to take place. These major state bits are readable.

1.6.1 Diagnosing the KL10

All KL10 diagnostic functions and console functions (except Deposit and Examine) are performed over the diagnostic portion of the EBus. This specification explains the operation of the diagnostic bus.

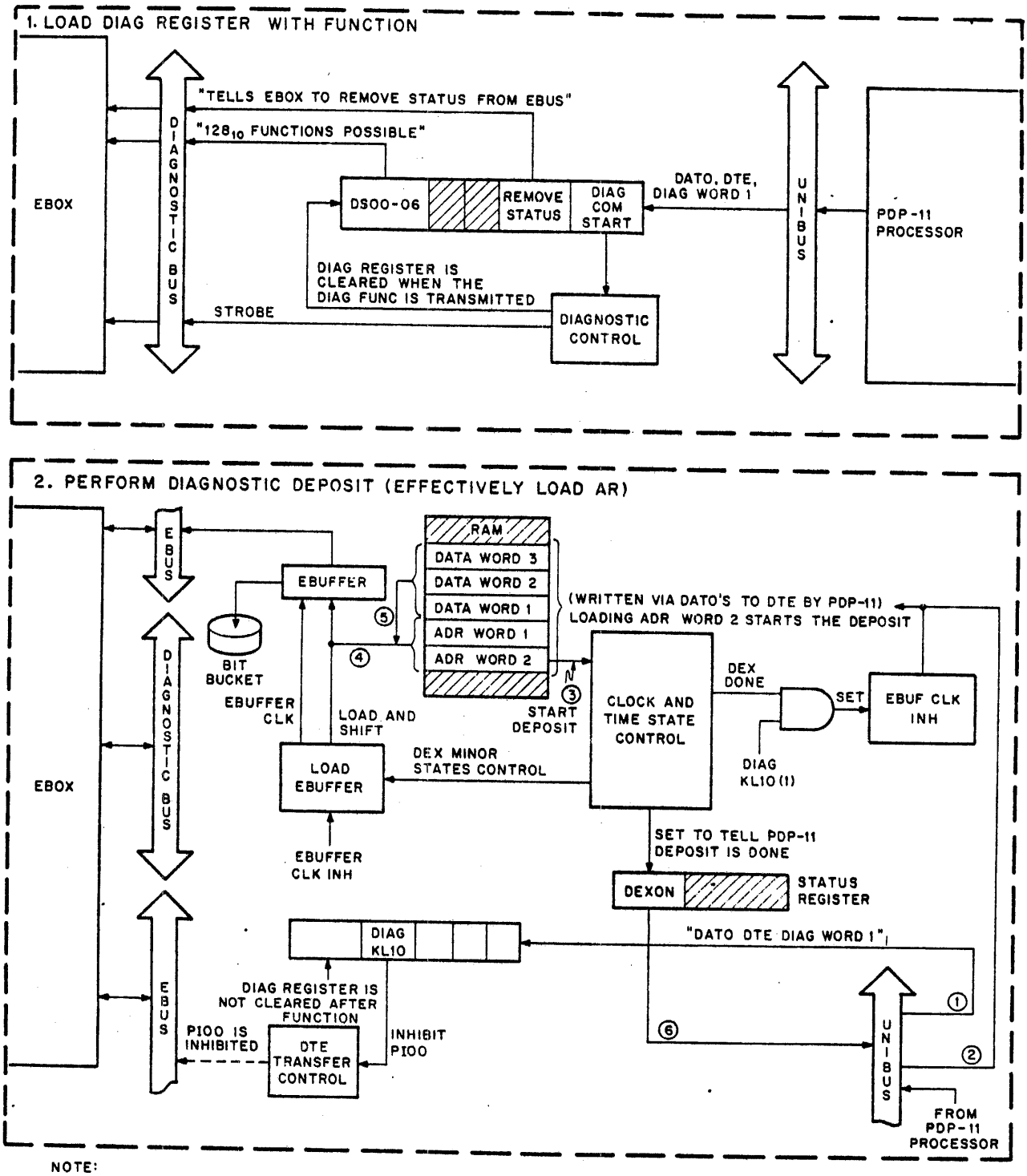
The diagnostic bus contains the following ten signal lines:

DS00-06	Diagnostic Select (DS) Lines – The PDP-11 sends encoded diagnostic functions to the KL10 on these lines. These lines can be read by the PDP-11 at any time, even while the rest of the EBus is active for other devices.
DIAG STROBE	Diagnostic Strobe – This line is asserted to indicate that the Diagnostic Select lines are stable, and that the indicated function should be performed.
DFUNC (Actual Mnemonic is Remove Status)	Diagnostic Function – When true, this causes the KL10 to disable the basic CPU status from the DS lines, switch the translator (only for the DS lines) to convert TTL to ECL, and put the EBus translator under control of DS bits 00 and 01.

1.6.1.1 Diagnostic Bus Control

Diagnostic CPU Status Read – All bits in DIAG Word 1 must be loaded with zeros. The CPU status may then be read from the DS lines after 1 μs has been allowed for the lines to settle (Figure 1-7).

Diagnostic Functions Only (i.e., no 36-bit transfers) – The desired function code bits should be set along with DIAG Command Start (DIAG1 PDCOMST) and Remove Status (DIAG1 [DFUNC]). This will result in the function being sent to the KL10. When DIAG Command Start is a zero, the function has been sent. All function bits must be loaded with the desired value each time a new command is sent. The DIAG Send bit has no effect upon this operation. DIAG KL10 must not be set or a 36-bit data transfer will take place. This operation should not take more than 2.0 μs.



NOTE:

4 Loading ADR1 is not necessary but loading ADR2 is required to begin the deposit operation

5 The clock and time state control loads and shifts DATAWORD 1, 2 and 3 into the EBUFFER which effectively discards ADR words 2 and 1

6 PDP-11 samples state, to determine when deposit is done; then may issue diagnostic function "LOADAR" causing EBOX to read the DATA from the EBUFFER into the AR. See 1

10-1792

DTE UNIT DESCRIPTION

Figure 1-7 Diagnostic Overview

1.6.1.2 Diagnostic Functions with 36-Bit Data Transfer

Sending Data to the KL10 – No other operations (i.e., byte string transfers) may be in progress while doing 36-bit diagnostic data transfers. The data should be loaded into DEX WD1-3 (same bit assignments as with a Deposit or Examine). DIAG KL10 should then be set and a Deposit operation should be started. When the transfer is complete (DEXDON SET), the diagnostic function should be loaded as described above, the DIAG KL10, DIAG Send, DIAG Command Start, and DIAG Function set. The operation is complete when DIAG Command Start is on a zero.

Receiving Data From the KL10 – The diagnostic function should be loaded with DIAG KL10 set, DFUNC set (Remove Status), DIAG Send clear, and DIAG Command Start set. When DIAG Command Start is clear, the function is complete and the data is in DEX WD1-3. No other operations (i.e., byte string transfers) may be in progress during this operation.

All the KL10 diagnostic functions are disabled when the privileged restricted mode switch is set to restricted mode. This bit can be tested by Reading Status (RM). When the switch is set to restricted mode, status (RM) is set (i.e., the device is restricted and cannot send diagnostic functions).

1.7 INTERFACE COMMUNICATION

The DTE20 can communicate directly with three devices in the system:

1. EBox via EBus
2. PDP-11 processor via Unibus
3. PDP-11 memory via Unibus.
4. DIAG BUSS - MBox - METER - some REG

This communication, when over the Unibus, is in a master-slave relationship (Figure 1-8). During any bus transfer, either the DTE20, the PDP-11 processor, or the PDP-11 memory has control of the bus. The controlling device is considered the bus master, and the device being controlled is considered the slave. Also, communication on the Unibus is interlocked between the DTE20 and either the PDP-11 processor or the memory. Each control signal issued by the master device must be acknowledged by a similar response from the slave device. Thus, communication is independent of bus length and of the response time between the master and the slave. When the DTE20 requests the bus, the handling of the request depends on the location of the interface in a priority structure. The following factors must be considered to determine the priority of the request:

1. The processor's priority is set under program control to one of eight levels using bits 7, 6, and 5 in the processor's Status register. These three bits set a priority level that inhibits granting bus requests (BR) on the same or lower levels.

2. Bus requests from external devices, i.e., DTE20 can be made on any one of five request lines. A non-processor request (NPR) has the highest priority and its request is granted by the processor between bus cycles of an instruction execution.

3. When more than one device is connected to the same bus request line, the one that is electrically closest to the PDP-11 processor has the higher priority.

Data Transfer Signals

Name	Mnemonic	No. of Lines
Data	D (00:35)	36
Controller Sel	CS (00:06)	7
Function	F (00:02)	3
Demand	DEM	1
Acknowledge	ACK	1
Transfer	XFER	1

DTE 1/17

Priority Transfer Signals DTE 1/18

Name	Mnemonic	No. of Signals
Controller Sel	CS (04:06)	3
Controller Sel	CS (00:03)	4
Function	F (00:02)	3
Demand	DEM	1
Acknowledge	ACK	1
Transfer	XFER	1

Data Transfer Signals DTE 1/14

Name	Mnemonic	No. of Lines
Data	D (15:00)	16
Address	A (17:00)	18
Control	C0, C1	2
Master Sync	MSYN	1
Slave Sync	SSYN	1
Parity	PA, PB	2
Interrupt	INTR	1
		Total: 41

Data Transfer Commands DTE 1/18,19 TABLE 1-3

F00	F01	F02	Operation
0	0	0	CONO
0	0	1	CONI
0	1	0	DATAO
0	1	1	DATAI
1	0	0	PI SERVED
1	0	1	PI ADDRESS IN

PDP-11 Device Registers DTE 1/20

Interface	Assignment
0	774400
1	774440
2	774500
3	774540

Data Transfer Operations DTE 1/1C

C1	C0	Operation
0	0	DATI -- data in
0	1	DATIP -- data in pause
1	0	DATO -- data out
1	1	DATOB -- data out byte

MOST IMPORTANT

1 OF 4?

DTE RAM

Fig 3/12 DTE/3-18

DTE 20 OPERATION 8-10

16X16

11/40 ADDRESS	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
174 400	ADDRESS 17 16 — DELAY COUNT - BYTE TRANSFERS
174 402	20 — DEX WORD 3 — 35
174 404	04 — DEX WORD 2 — 19
174 406	DEX WORD 1 — 00 — 03
174 410	ADDRESS SPACE — DEP EX — Q — 0 0 0 0 — 13 DEX ADDRESS 1 — 19
174 412	20 — DEX ADDRESS 2 — 35
174 414	I — 0 0 0 — TO10 BYTE COUNT
174 416	I — Z — BM — 0 — TO11 BYTE COUNT
174 420	TO10 TRANSFER ADDRESS (-11)
174 422	TO11 TRANSFER ADDRESS (-11)
174 424	TO10 DATA
174 426	TO11 DATA
174 430	DIAGNOSTIC WORD 1
174 432	DIAGNOSTIC WORD 2
174 434	STATUS REGISTER
174 436	DIAGNOSTIC WORD 3

EPT, DTE REGION IN KL10

N = 0, 1, 2, 3	
140 + 8N	TO11 BYTE POINTER
141	"	TO10 BYTE POINTER
142	"	DTE INTERRUPT INSTRUCTION
143	"	RESERVED
144	"	EXAMINE PROTECTION
145	"	EXAMINE RELOCATION
146	"	DEPOSIT PROTECTION
147	"	DEPOSIT RELOCATION

PDP-11 DTE#	Vector Addr or BR	DTE RAM S.A.	KL10 DTE#	Duc Code	Phy#	EPT Loc.
0	774	174400-36	0	200	10 (8)	140-7
1	770	174440-76	1	204	11 (9)	150-7
2	764	174500-36	2	210	12 (10)	160-7
3	760	174540-76	3	214	13 (11)	170-7

DTE /1-21

Addressable Register Summary

Register Name	Accessible By:	PDP-11 INSTR	DTE20 ADR	KL10 INSTR	FCN CODE
DIAG3	PDP-11	DATO, DATI	XXX36	-	-
STATUS	BOTH PDP-11 and KL10	DATO, DATI	XXX34	CONO, CONI	0,1
DIAG2	PDP-11	DATO, DATI	XXX32	-	-
DIAG1	PDP-11	DATO, DATI	XXX30	-	-
TO11 DATA	PDP-11	DATO, DATI	XXX26		
TO10 DATA	PDP-11	DATO, DATI	XXX24	-	-
TO11 ADR	PDP-11	DATO, DATI	XXX22	-	-
TO10 ADR	PDP-11	DATO, DATI	XXX20	-	-
TO11 BYTE CNT	PDP-11	DATO, DATI	XXX16	-	-
TO10 BYTE CNT	BOTH PDP-11 and KL10	DATO, DATI	XXX14	DATAO	2
ADDRESS WORD 2	PDP-11	DATO, DATI	XXX12		-
ADDRESS WORD 1	PDP-11	DATO, DATI	XXX10		-
DATA WORD 1	PDP-11	DATO, DATI	XXX06		-
DATA WORD 2	PDP-11	DATO, DATI	XXX04	-	-
DATA WORD 3	PDP-11	DATO, DATI	XXX02		-
DELAY COUNT	PDP-11	DATO, DATI	XXX00	-	-

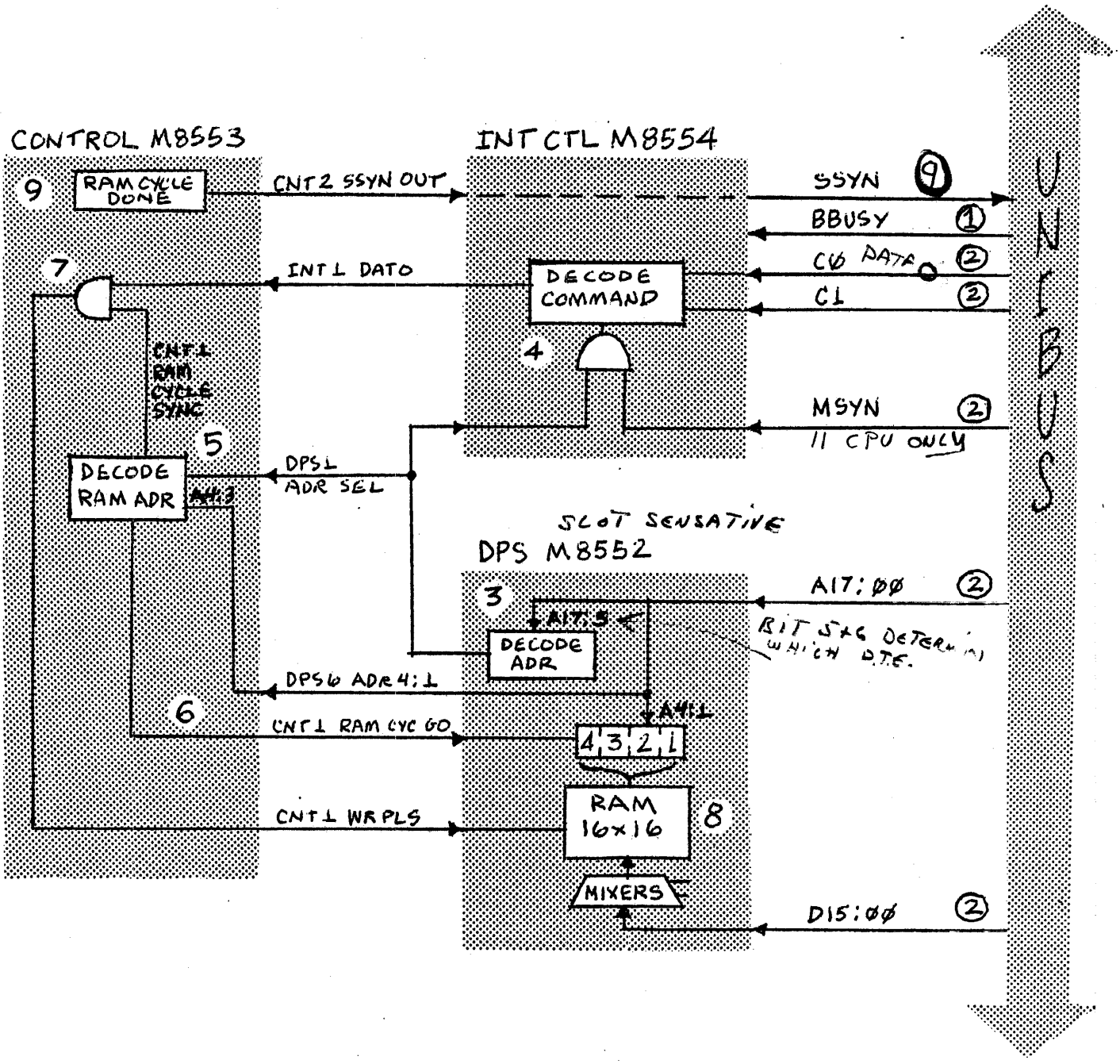
Each DTE has 2 PI Assignments on the E Bus.

PI Ø - DEX, TO1Ø, TO11 Data Xfers.
PI A 1-7 - Doorbell & Status.

Each DTE Can Interrupt the PDP-11.

BR (4) for DEX DONE, STATUS, DOORBELL
NPR for TO1Ø, TO11 Data Xfers.

D 1-15



RAM LOAD SEQUENCE

Description:

1. PDP-11 becomes master of Unibus and asserts BBUSY.

2. PDP-11 places DATO command on CØ and C1; address on A15:ØØ, ADR17 and ADR16; data to be loaded to RAM on D15:ØØ and activates MSYN to begin operation.

3. DPS M8552 decodes address bits 17:5 as the DTE select, and generates ADR SEL(H) to Control and INT Control if the address matches the wired option.

4. INT Control samples MSYN and ADR SEL and allows the DTE to decode CØ and C1. In this case, the decoded command must equal DATO.

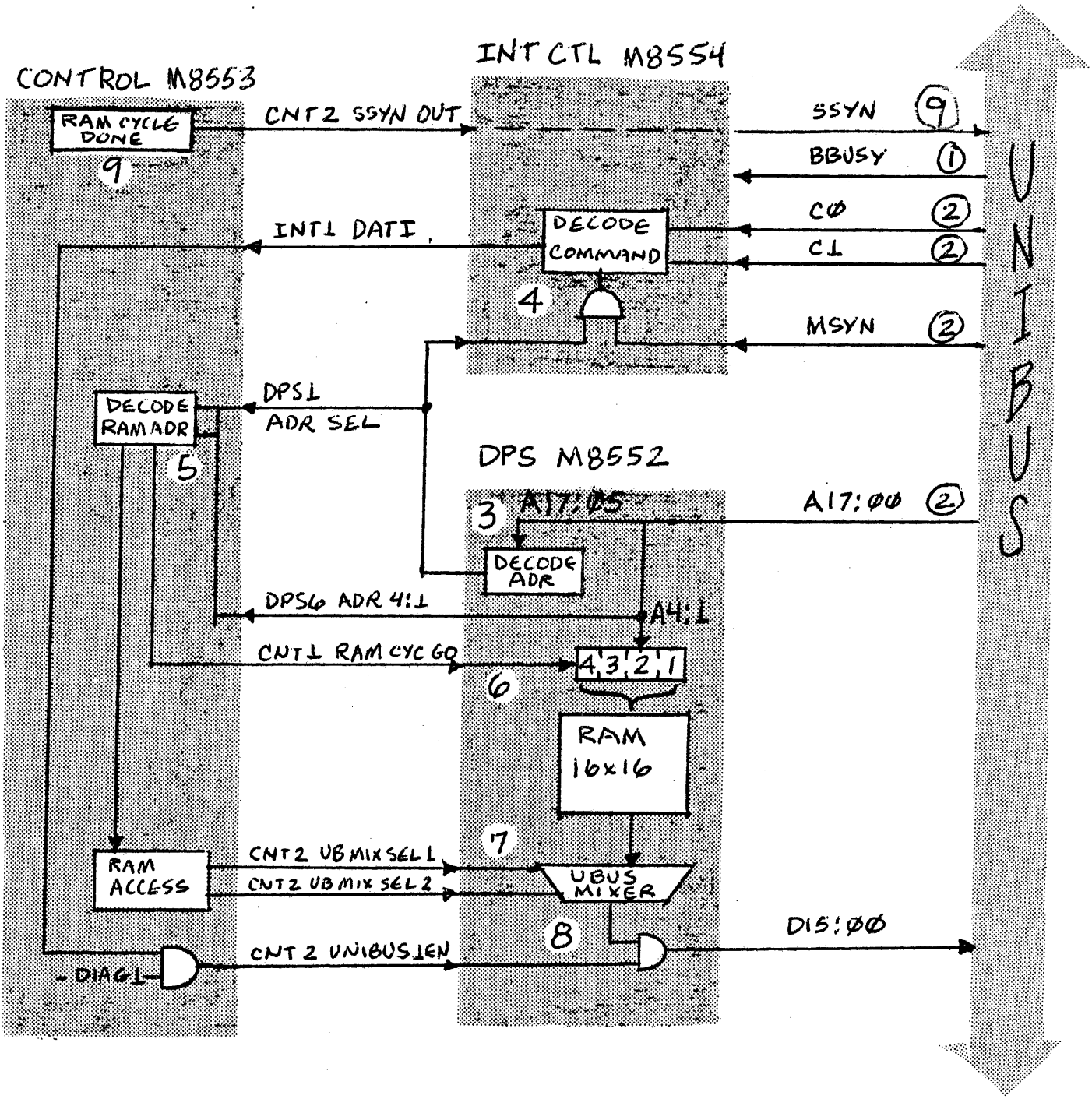
5. Control samples ADR SEL and ADR 4:3 to generate a RAM cycle. If ADR 4:3 are both present, this would not be a RAM cycle, but a register cycle.

6. Control generates "RAM CYCLE GO" to DPS which allows the selected RAM address (4:1) to be "gated" to the RAM address lines.

7. Control determines that data is to be written to the RAM by "ANDING" DATO and RAM CYCLE SYNC and generates "WR PLS" (write pulse) to DPS M8552.

8. DPS M8552 seeing the "WR PLS" will now write the data from D15:ØØ into the RAM address specified by A4:1.

9. Control recognizing that the RAM cycle is complete will generate "SSYN OUT" which is passed to the Unibus via interrupt control. This will cause the PDP-11/4Ø to drop MSYN and ultimately terminate this operation. SSYN will drop after MSYN drops.

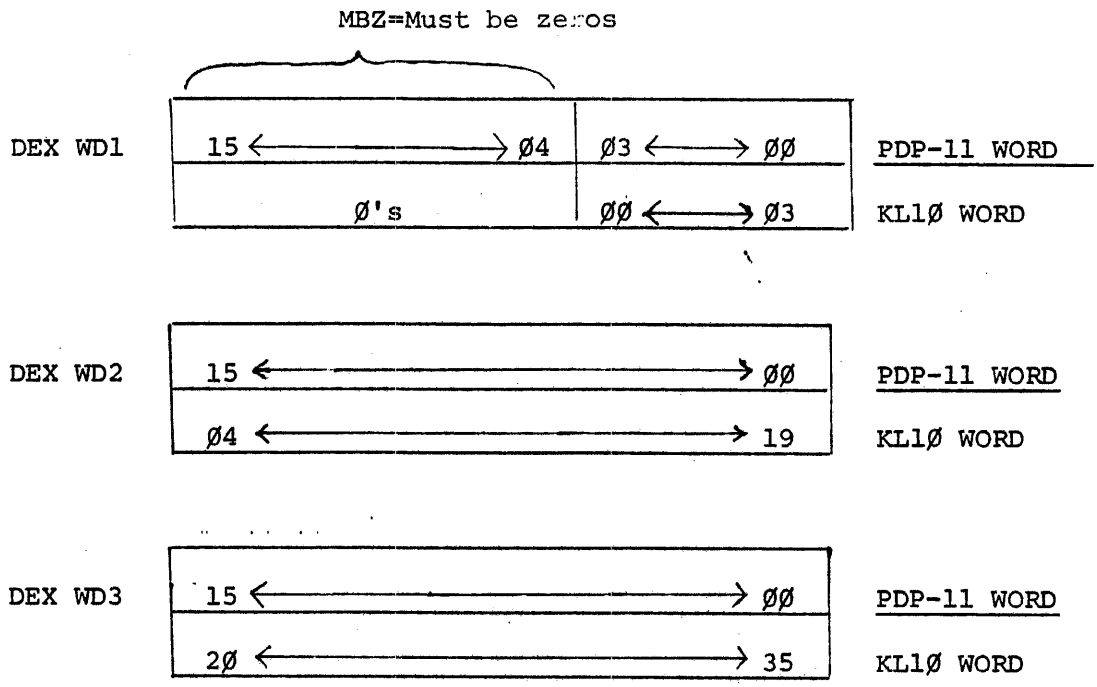


RAM READ SEQUENCE

SEE D2-3

Description:

1. PDP-11 becomes master of Unibus and asserts BBUSY.
2. PDP-11 places DATI command on CØ and C1; address on A15:ØØ, ADR 17 and ADR 16 and activates MSYN to begin operation.
3. DPS M8552 decodes address bits 17:5 as the DTE select and generates ADR SEL(H) to control and interrupt control if the address matches the wired option.
4. INT control sensing both ADR SEL and MSYN will allow the DTE to decode CØ and C1. In this case, the decoded command must equal DATI.
5. Control samples ADR SEL and ADR 4:3 to generate a RAM cycle.
6. Control then generates "RAM CYCLE GO" which allows bits 4:1 to be "gated" to the RAM address circuitry. This will be the address of the desired word to be read from the RAM. "READ" is enabled due to the absence of write pulse which was generated in Figure 3 by DATO.
7. The RAM data 15:ØØ is presented to a mixer which allows various sources to provide information to the Unibus. In this case, due to the RAM access, UB MIX SEL (2) and (1) will be asserted low, allowing the mixer to pass RAM data.
8. The data is enabled to the Unibus because of DATI and this not being a diagnostic register 1 sequence.
9. Control recognizing that the RAM cycle is complete will generate SSYN OUT which is passed to the Unibus via interrupt control. This will ultimately terminate the READ RAM sequence.



DEPOSIT/EXAMINE WORDS 1, 2, & 3

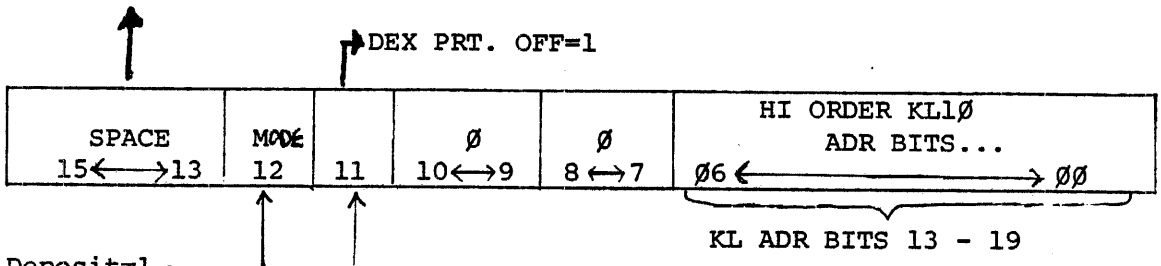
SEE DTE 2-9

SEE D1-14

See DTE 2-9

Ø=EPT
1=EXEC VIRTUAL

4=PHYSICAL
5-7=UNDEFINED

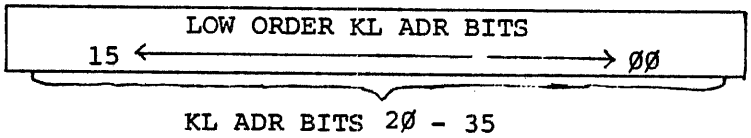


TEN ADR WD 1

Deposit=1
Examine=Ø

DEX protect off.
If Bit=1, DEX's are not relocated and no protection check is made. The ADR space is designated by Bits 15 - 13 of this word.

The F/F which this bit sets, is cleared after each DEX operation. Therefore, to do repetetive unprotected operations, ADR WD 1 must be loaded each time with bit 11=1.



Ten ADR WD 2

DEX ADDRESS WD 1 & 2

DTE 2-9

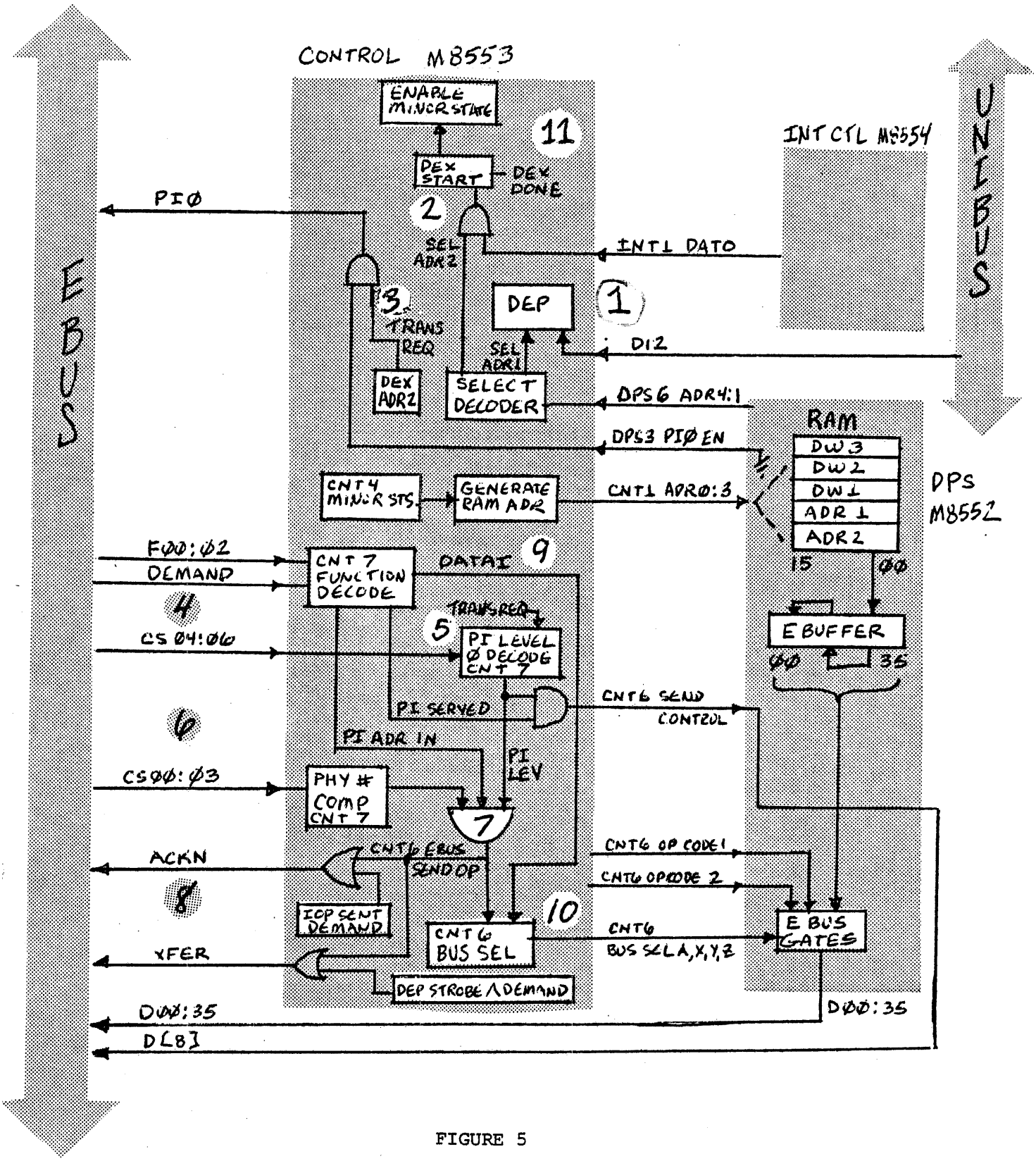


FIGURE 5

DEPOSIT OPERATION

TO KL10

SEE DTE/2-9

SEE D2-7

DESCRIPTION:

1. When address word 1 is loaded to the RAM from the PDP-11/4Ø, the select decoders on M8553 will generate SEL ADR1 which clocks Ubus Bit 12 into the deposit latch. (Bit 12=1 for deposit and therefore sets the latch.) This will set up the control circuitry to do a deposit.

2. The loading of address word 2, will allow the select decodes to generate SEL ADR2, which sets the DEX start flop. DEX Start set locks the DTE into the DEX major state thereby enabling the DEX minor state decoder. Minor states ADR1 and ADR2 are generated and are used to pack ADR1 and 2→E Buffer.

3. Minor state DEX ADR2 generates TRANS REQ which will force an interrupt to the E-Box on PI level Ø. This also halts the stepping of the minor states until the E-Bus interrupt sequence is complete.

4. When the E-Box PI system establishes this interrupt as highest priority, it will assert:

   a. FØØ:Ø2 = PI servied = 4
   b. CSØ4:Ø6 = Interrupting Chan# = Ø
   c. Demand

5. Control decodes CHAN# Ø with CNT4 TRANS REQ active. If the decode is satisfied, PI served and PI level generate "send control" to DPS M8552. "Send Control" is merely "Gated" through the DPS and onto the E-Bus as "E-Bus'D[8]" (This will represent the physical controller #.) Bit 8 will represent the PHYS CONT # for the DTE.

6. The E-Box will examine this physical controller number and select the DTE as highest priority by asserting:

   a. FØØ=Ø2=PI ADR IN = 5
   b. CSØ4=Ø6=INTERRUPTING CHAN#
   c. CSØØ=Ø3=PHYSICAL CONTROLLER SELECTED
   d. DEMAND

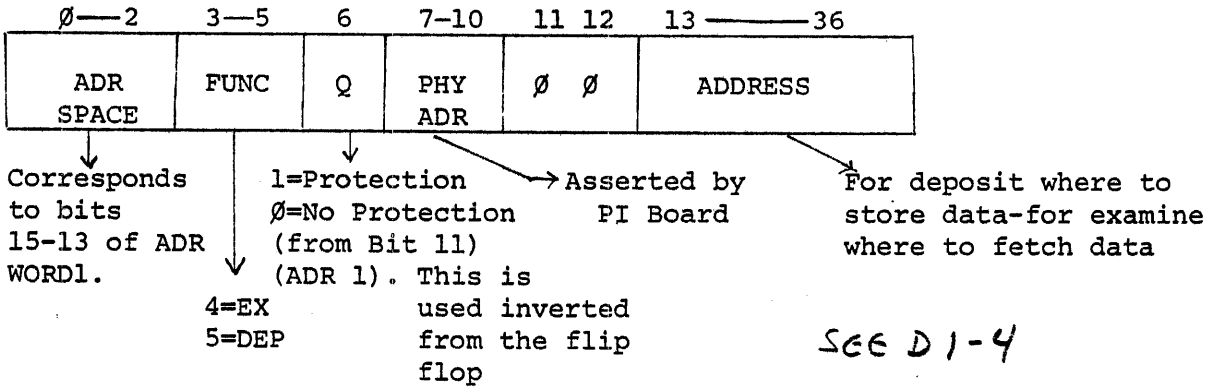
7. If the PHY#, PI LEV, and PI ADR IN are all present in the DTE, E-Bus SEND OP is generated on Control Print 6 of 8.

8. E-Bus send op will allow E-Bus ACKN, E-Bus Transfer, and E-Bus Enables to be generated which ultimately allow the API word to be passed to the E-Bus on the DØØ-35 lines. E-Bus ACKN de-selects the DIA for this transfer. E-Bus transfer "TELLS" the E-Box PI system to generate "PI READY".

The API word is formed on DPS M8552 by packing ADR1 and ADR2 and other controlling signals into the E-Buffer.

The first two minor states of the DEX major state will accomplish this (DEX ADR1, DEX ADR2). These signals will allow the proper RAM addresses to be generated to select the previously loaded ADR words.

The API word formed in the E-Buffer is as shown below:



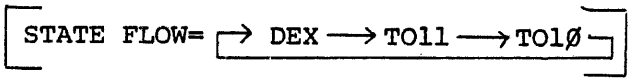
9. E-Box will drop demand, causing acceptance of the API word and sets IOP SENT.* The E-Box will then generate a DATAI command because the API FUNC code = deposit (5_8). To send the DATAI Command, the E-Box generates:

A. FØ:2 = 3 ($Ø11_2$) DATAI
B DEMAND

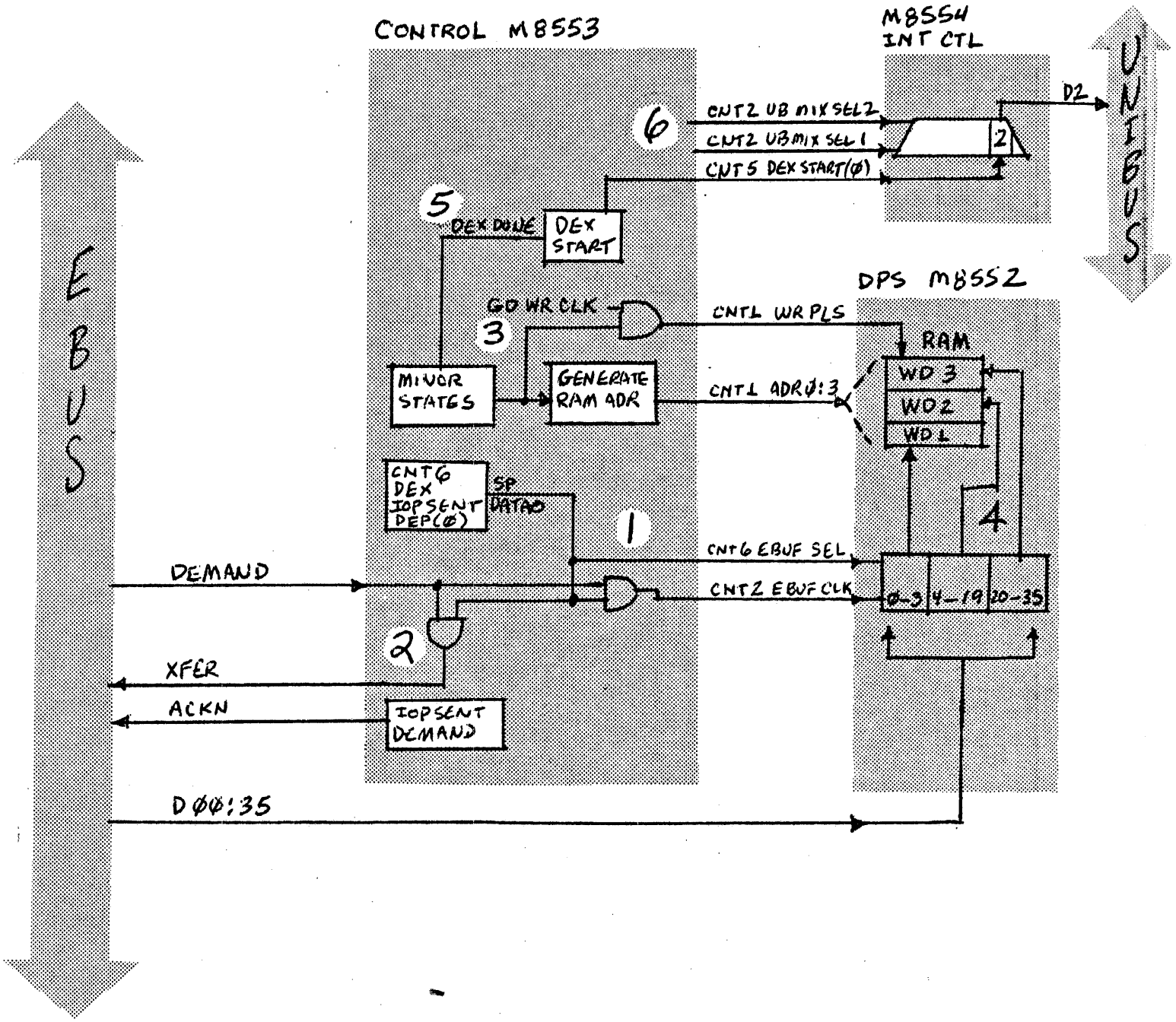
*The DATAI will only be accepted by the DTE that has "IOP SENT" active. "IOP SENT" indicates that the DTE is fully locked into the E-Bus. The CS lines aren't used in this case to select the DTE for the acceptance of the DATAI. "IOP SENT" releases the gated clocks via bus complete so that the minor states can be restarted from ADR2 and also resets trans req.

10. The DATAI will generate the E-Bus SEL enables A, X, Y, and Z, to allow the 36 bit word packed from RAM WD1, WD2, WD3, to be gated to the E-Bus. The E-Buffer packing is accomplished during minor states DEX WD1, DEX WD2, and DEX WD3. E-Bus ACKN and transfer are generated for the word transfer at this time also.

11. Demand dropping will ultimately terminate this operation. The minor state decoder will generate DEX Done which clears DEX Start. This will "unlock" the DTE from DEX major state. Control then begins toggling from state to state sampling conditions to lock into the next state desired.



The RAM locations are not changed, therefore to do a repetitive operation, the PDP-11/4Ø would simply have to reload DEX ADR2 into the RAM. For unprotected operations, the -11 must reload the protect off bit (ADR1 BIT11) each time.



DTE 2-13

FIGURE 6
EXAMINE OPERATION

DESCRIPTION:

The first nine steps of the description of Figure 5, the deposit operation, are essentially identical with the first nine steps required here. Therefore, they are eliminated from this figure. The differences are described below.

A. Figure 5, Step 1 - for the examine operation, Bit 12=Ø
B. Figure 5, Step 9 - E-Box will generate a DATAO command because of the IOP FCN (4_8).

1. DEX, IOP SENT, and the deposit latch not set generate CNT6 SP DATAO. SP DATAO and DEMAND generate E-Buf CLK which gates the 36 bit word from E-Bus DØØ:35 into the E-Buffer.

2. SP DATAO and DEMAND also generate CNT6 TRANS which in turn generates E-Bus XFER. E-Bus ACKN is generated due to IOP SENT and DEMAND.

3. The minor state signal DEX WD1 generates an address of 3_8 which is gated to the RAM. E-Buffer bits Ø-3 are also presented to the RAM data lines at this time. When WR PLS is activated, the data is written to location 3 of the RAM.

4. DEX WD2 and DEX WD3 will then follow in order to generate the appropriate RAM addresses and select the proper portions of the E-Buffer to be written to the RAM. WR PLS must therefore be activated two more times to accomplish this.

5. The minor state decoder generates DEX DONE which clears the DEX Start Flip Latch.

6. The clearing of DEX START and UB Mixer SEL 2 and 1 being low will allow bit 2 to be asserted on the Unibus.

7. The looping program in the 11 has been sampling the DTE status and when it sees bit 2 finally active, the 11 program will proceed to an area that allows the 3 RAM locations just filled to be read into the 11 memory.

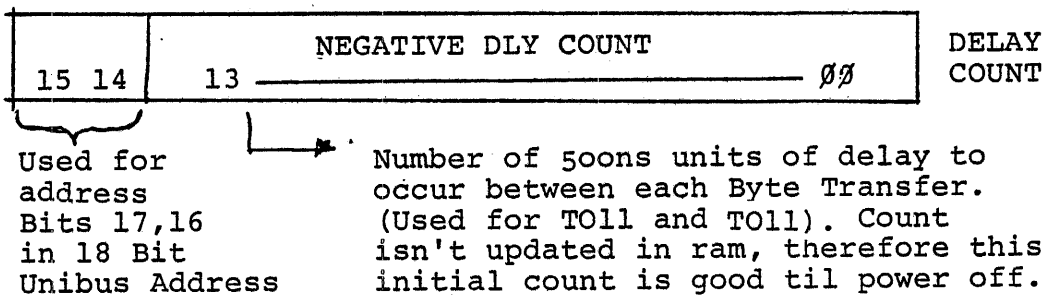
8. Demand in the meantime from the E-Bus, has been dropped and the examine operation will terminate.

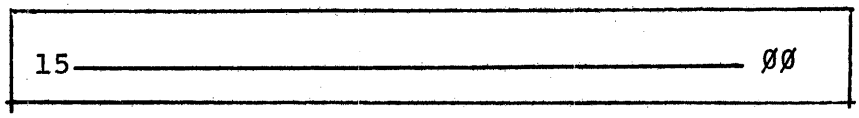
9. The DTE will begin toggling from major state to major state.

TO11 TRANSFER RAM REQUIREMENTS:

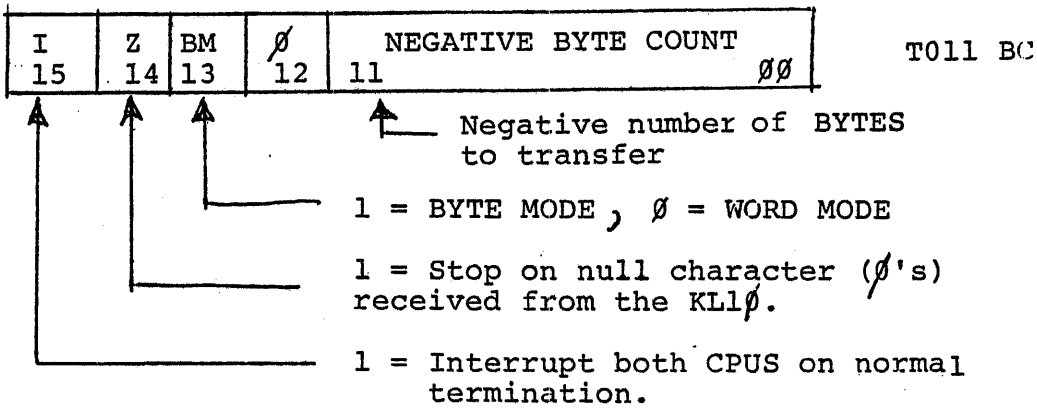
RAM ADR	WORD	USAGE	PDP-11 LOC.
ØØ	TO11 Delay Count	Negative number used to create delay between byte transfer.	164ØØØ+40*N
Ø7	TO11 Byte Count (BC)	Negative count of the number of bytes to be transferred.	164Ø16+4Ø*N
11	TO11 (PDP-11) ADR	Initial PDP-11 ADR to store data.	164Ø22+40*N

WORD FORMATS





Byte ADR in PDP-11 where to store the Byte received from the KL1Ø. Other 2 bits for full 18 bit address come from 15, 14 of DLY count word.



D 2-13
DTE 2-21

TOII TRANSFER

TOII-XFER

DTE DETECTS TOII ADR AND BC LOAD. THIS GENERATES TOII RDY ON CNT 5.

LOCK DTE INTO TOII TRANS MAJOR STATE ON CNT 4 - INITIATE MINOR STATES

TOII DLY RD — MINOR STATE

TRANSFER TOII DLY CNT → ABC REG

CNT⊥ ADR Ø:3 → DPS4 (RAM ADR = ØØ)
CNT⊥ DLY CNT RD ∧ CNT4 GD SHF CLK ; CNT⊥ ABC LOAD
STEP TO NEXT MINOR STATE

TOII DLY INC

THIS MINOR STATE MAINTAINED UNTIL ABC REG IS INCREMENTED TO ZERO OR INTERRUPTED FOR LEX OPERATION

DPS1 ABC REG 13 ∧ CNT1 DLY INC ; INHIBIT ST CLK → CNT4 BIN CTR.

ABC INCREMENTED TO Ø

Y → A (SH 1 OF 4)

N

DPS1 ABC 13 = Ø

DEX START ENABLED

Y → G (SH 4 OF 4)

N

INCREMENT ABC REGISTER +1

CNT1 DLY INC ∧ CNT4 GD WR CLK ; CNT⊥ ABC INC

A (SH 1 OF 4)

INCREMENT TO NEXT MINOR STATE

CNT4 GD ST CLK ; CNT4 TOII BC RD

TOII BC RD

TRANSFER TOII BC → ABC REG

CNT⊥ ADR Ø:3 → DPS4 (RAM ADR = Ø7)
CNT⊥ BC RD ∧ CNT4 GD SHF CLK ; CNT⊥ ABC LOAD

BC = Ø DPS4

Y → B (SH 4 OF 4)

N

INCREMENT TO NEXT MINOR STATE

CNT4 GD ST CLK ; CNT4 TOII BC INC

TOII BC INC

INCREMENT BC IN ABC REG BY +1

CNT1 BC INC ∧ CNT4 GD WR CLK ; CNT⊥ ABC INC

INCREMENT TO NEXT MINOR STATE

CNT4 GD ST CLK ; CNT4 TOII I/O FUNC

TOII I/O FUNC

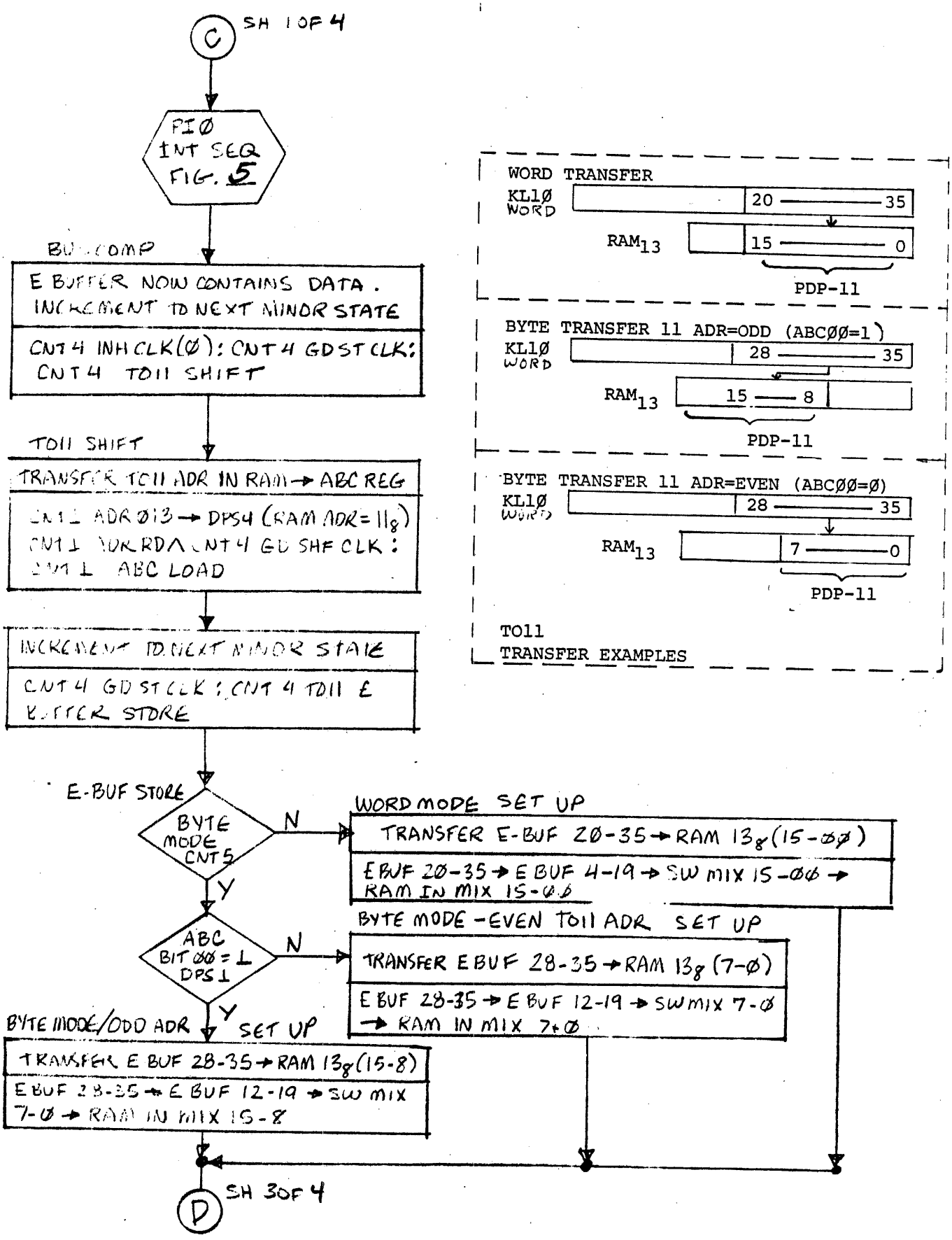
TRANSFER UPDATED BC → RAM $Ø7_8$
INTERRUPT KL1Ø ON PIØ
HOLD THIS MINOR STATE UNTIL EBUS DIALOGUE COMPLETE (API FCN = 6, Q = Ø)

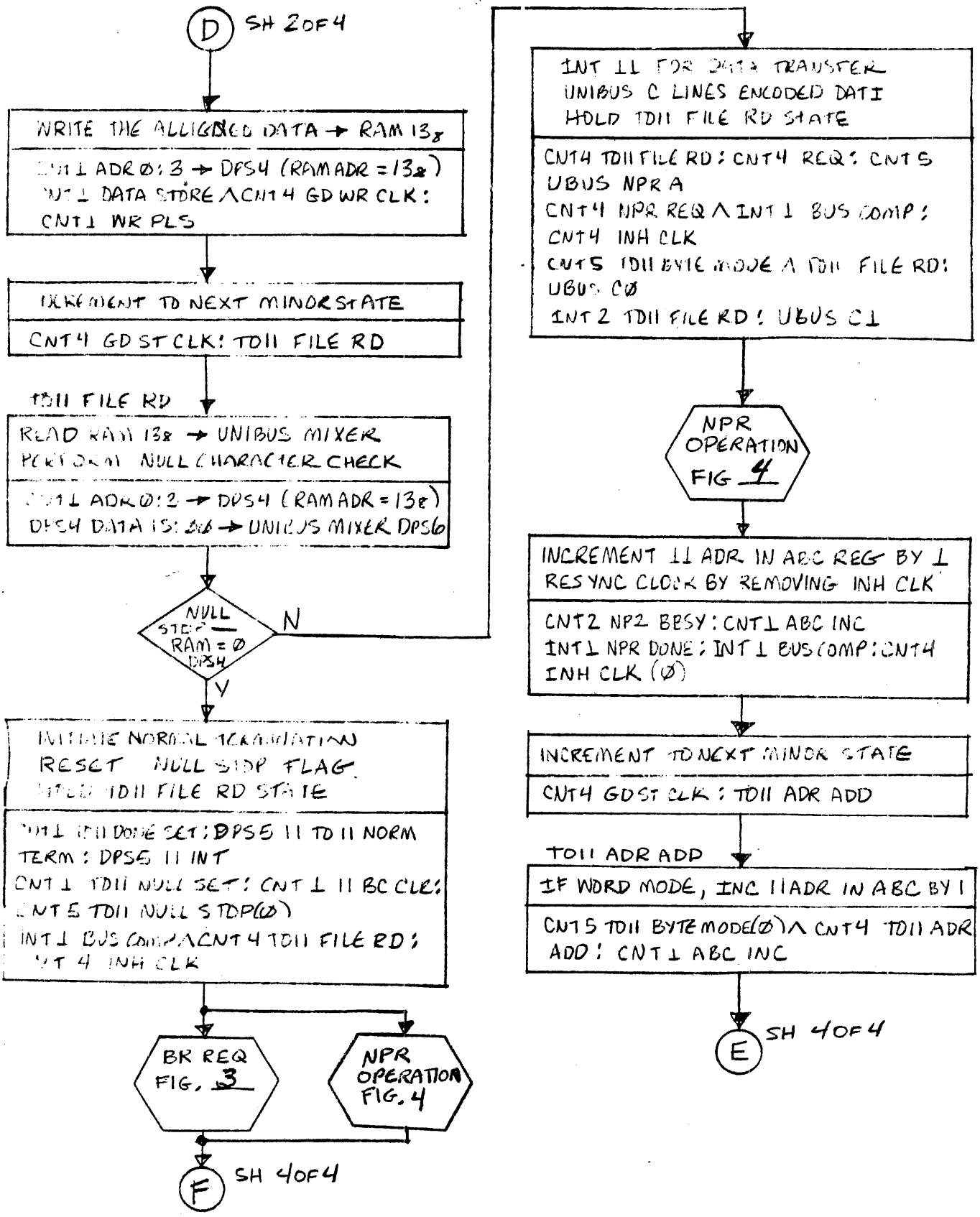
INT1 BUSCOMP ∧ CNT4 TOII I/O FUNC ; CNT4 INH CLK

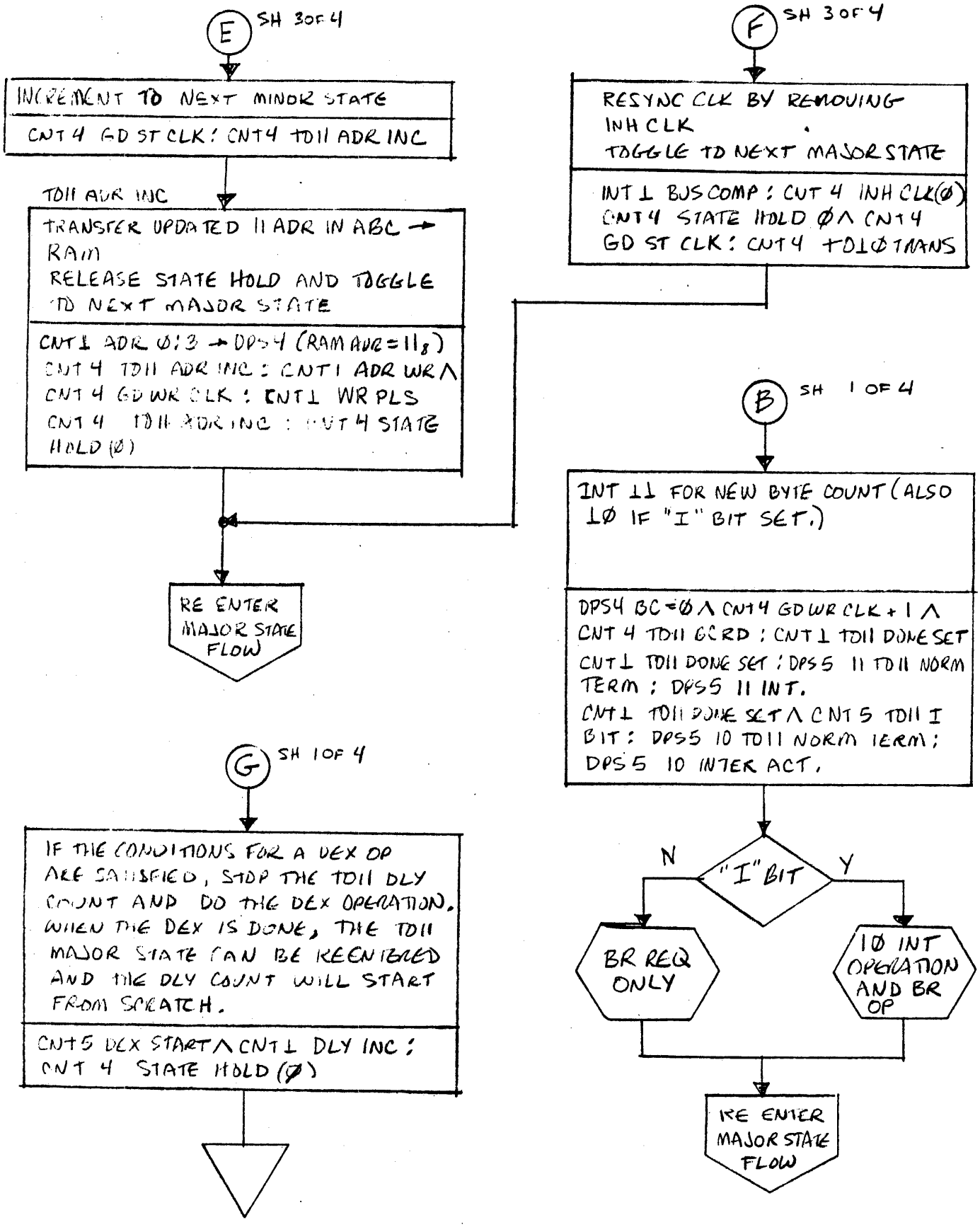
SH 2 OF 4

C

FIGURE 1, TOII TRANS SHEET 1 OF 4


C
SH 1 OF 4
FIØ
INT SEQ
FIG. 5
BU...COMP
E BUFFER NOW CONTAINS DATA.
INCREMENT TO NEXT MINOR STATE
CNT 4 INH CLK(Ø); CNT 4 GD ST CLK;
CNT 4 TO11 SHIFT
TO11 SHIFT
TRANSFER TO11 ADR IN RAM → ABC REG
CNT ADR Ø13 → DPS4 (RAM ADR = 11₈)
CNT ADR RD ∧ CNT 4 GD SHF CLK;
CNT ABC LOAD
INCREMENT TO NEXT MINOR STATE
CNT 4 GD ST CLK; CNT 4 TO11 E
BUFFER STORE
E-BUF STORE
BYTE MODE CNT 5
N
Y
WORD MODE SET UP
TRANSFER E-BUF 20-35 → RAM 13₈ (15-ØØ)
EBUF 20-35 → E BUF 4-19 → SW MIX 15-ØØ →
RAM IN MIX 15-ØØ
ABC BIT ØØ = ⊥ DPS ⊥
N
Y
BYTE MODE - EVEN TO11 ADR SET UP
TRANSFER EBUF 28-35 → RAM 13₈ (7-Ø)
E BUF 28-35 → E BUF 12-19 → SW MIX 7-Ø
→ RAM IN MIX 7-Ø
BYTE MODE/ODD ADR SET UP
TRANSFER E BUF 28-35 → RAM 13₈ (15-8)
E BUF 28-35 → E BUF 12-19 → SW MIX
7-Ø → RAM IN MIX 15-8
D
SH 3 OF 4
WORD TRANSFER
KL1Ø WORD
20
35
RAM13
15
0
PDP-11
BYTE TRANSFER 11 ADR=ODD (ABCØØ=1)
KL1Ø WORD
28
35
RAM13
15
8
PDP-11
BYTE TRANSFER 11 ADR=EVEN (ABCØØ=Ø)
KL1Ø WORD
28
35
RAM13
7
0
PDP-11
TO11
TRANSFER EXAMPLES


D
SH 2 OF 4
WRITE THE ALLIGNED DATA → RAM 13₈
CNT 1 ADR Ø:3 → DPS4 (RAM ADR = 13₈)
INT 1 DATA STORE ∧ CNT 4 GD WR CLK:
CNT 1 WR PLS
INCREMENT TO NEXT MINOR STATE
CNT 4 GD ST CLK: TDII FILE RD
TDII FILE RD
READ RAM 13₈ → UNIBUS MIXER
PERFORM NULL CHARACTER CHECK
CNT 1 ADR Ø:2 → DPS4 (RAM ADR = 13₈)
DPS4 DATA IS: Ø → UNIBUS MIXER DPS6
NULL STOP — RAM = Ø DPS4
N
Y
INITIATE NORMAL TERMINATION
RESET NULL STOP FLAG
HOLD TDII FILE RD STATE
CNT 1 TDII DONE SET; DPS5 11 TO 11 NORM TERM: DPS5 11 INT
CNT 1 TDII NULL SET: CNT 1 11 BC CLR;
CNT 5 TDII NULL STOP(Ø)
INT 1 BUS COMP ∧ CNT 4 TDII FILE RD;
CNT 4 INH CLK
BR REQ FIG. 3
NPR OPERATION FIG. 4
F
SH 4 OF 4
INT 11 FOR DATA TRANSFER
UNIBUS C LINES ENCODED DATI
HOLD TDII FILE RD STATE
CNT 4 TDII FILE RD: CNT 4 REQ: CNT 5 UBUS NPR A
CNT 4 NPR REQ ∧ INT 1 BUS COMP; CNT 4 INH CLK
CNT 5 TDII BYTE MODE ∧ TDII FILE RD: UBUS CØ
INT 2 TDII FILE RD: UBUS C1
NPR OPERATION FIG 4
INCREMENT 11 ADR IN ABC REG BY 1
RESYNC CLOCK BY REMOVING INH CLK
CNT 2 NP2 BBSY: CNT 1 ABC INC
INT 1 NPR DONE; INT 1 BUS COMP: CNT 4 INH CLK (Ø)
INCREMENT TO NEXT MINOR STATE
CNT 4 GD ST CLK: TDII ADR ADD
TDII ADR ADD
IF WORD MODE, INC 11 ADR IN ABC BY 1
CNT 5 TDII BYTE MODE(Ø) ∧ CNT 4 TDII ADR ADD: CNT 1 ABC INC
E
SH 4 OF 4



TDII TRANS SHEET 4 OF 4

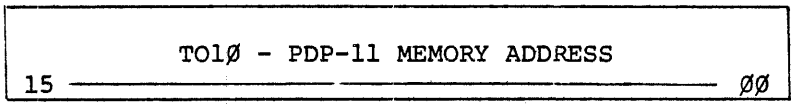
D 2-17

TO1Ø TRANSFER REQUIREMENTS

RAM ADR	WORD	USAGE	PDP-11 LOC
ØØ	DLY COUNT	(See TO11 Xfer)	164ØØØ+4Ø*N
1Ø	TO1Ø ADR	PDP-11 Address of Source Data	164Ø2Ø+40*N

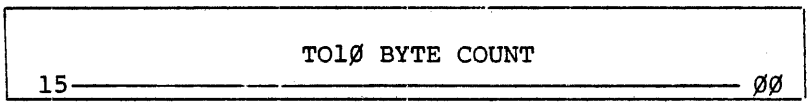
DIAG Register 3 Bit ØØ specified byte or word mode and must be loaded accordingly via a register load sequence from the PDP-11.

WORD FORMATS:



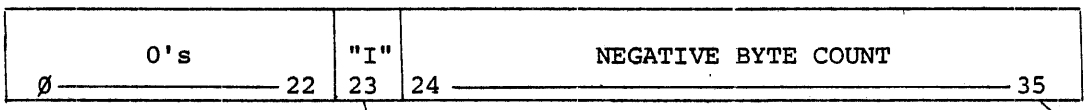
TO1Ø AD

Byte address of source string. Updated as each byte/word is transferred. At end of transfer points to byte which would have been transferred next...



TO1Ø BC

Negative byte count - readable and writable by the PDP-11. DTE-2Ø will not start transfer until 10 sets this register with a DATAO.



DATAO WORD FORMAT

TO1Ø "I" bit -
1=set I bit for TO1Ø transfer. [INT both 10 & 11 on normal term]
Ø=don't set "I" bit for TO1Ø transfer. [INT 1Ø only on normal term]

Twc's compliment of number of characters. Left to be transferred.
EX: BC=$ØØØØ_8$=Ø bytes transferred

BC=7777_8=1 byte transfer

BC=7773_8=5 byte transfer

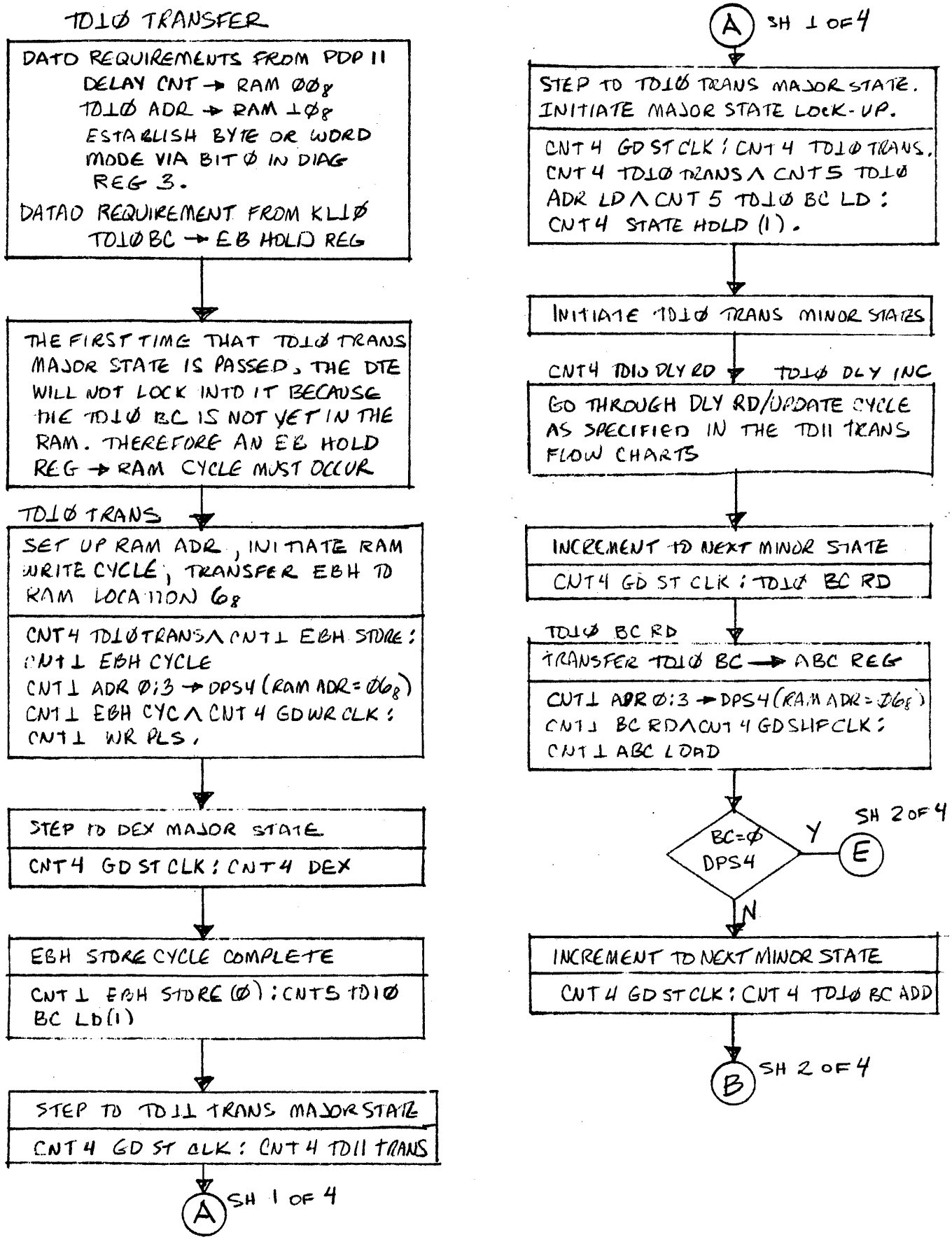
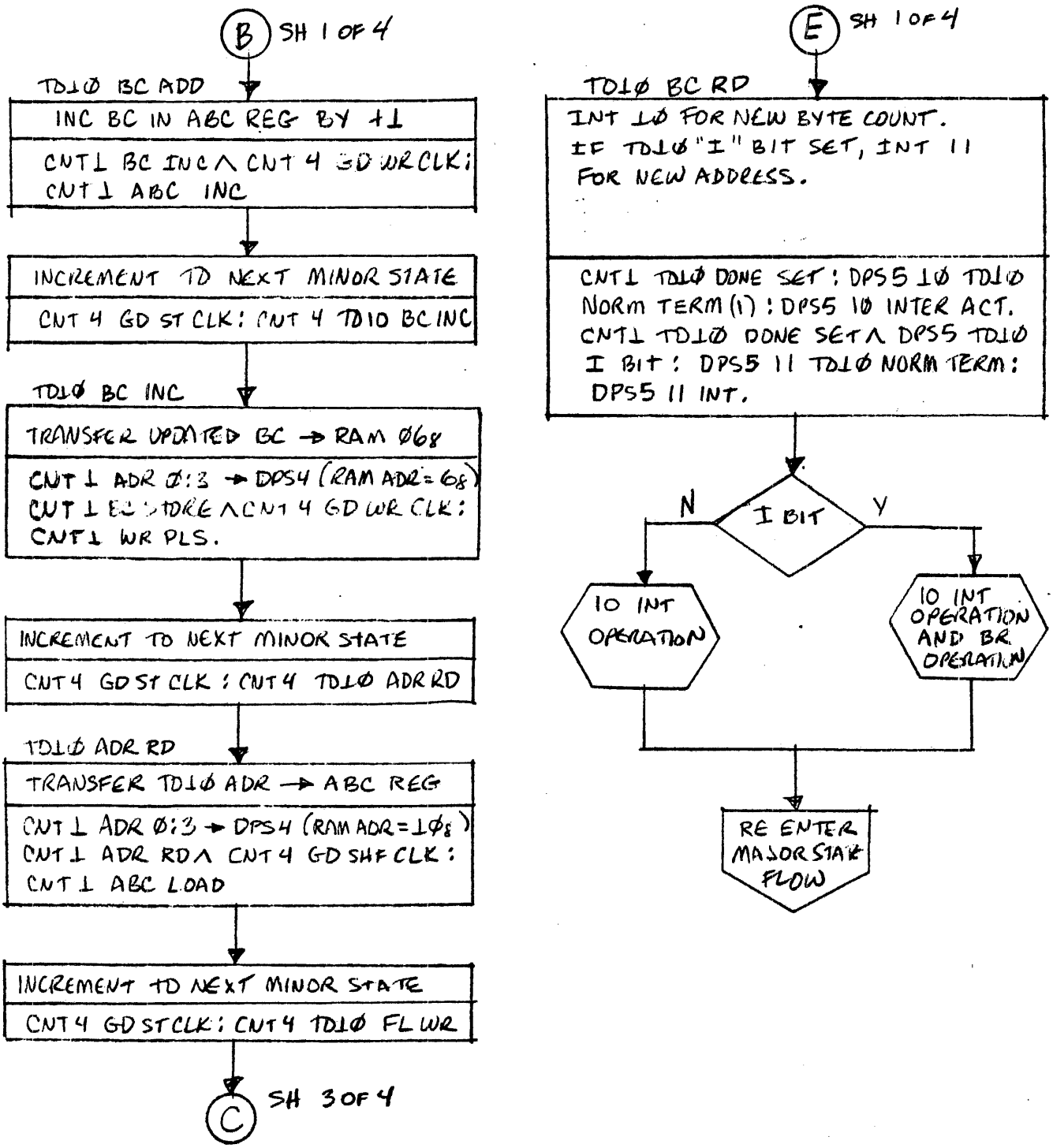
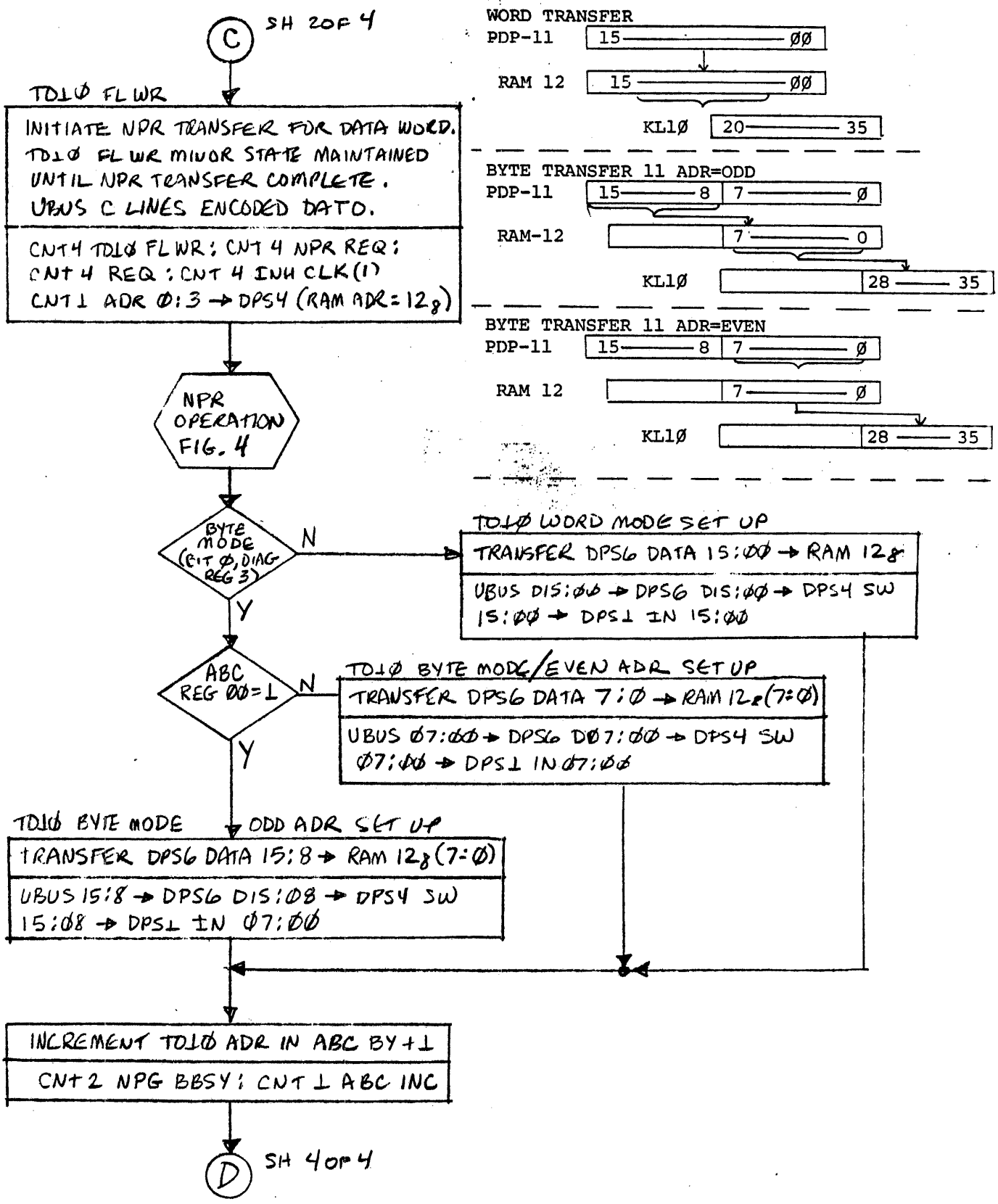


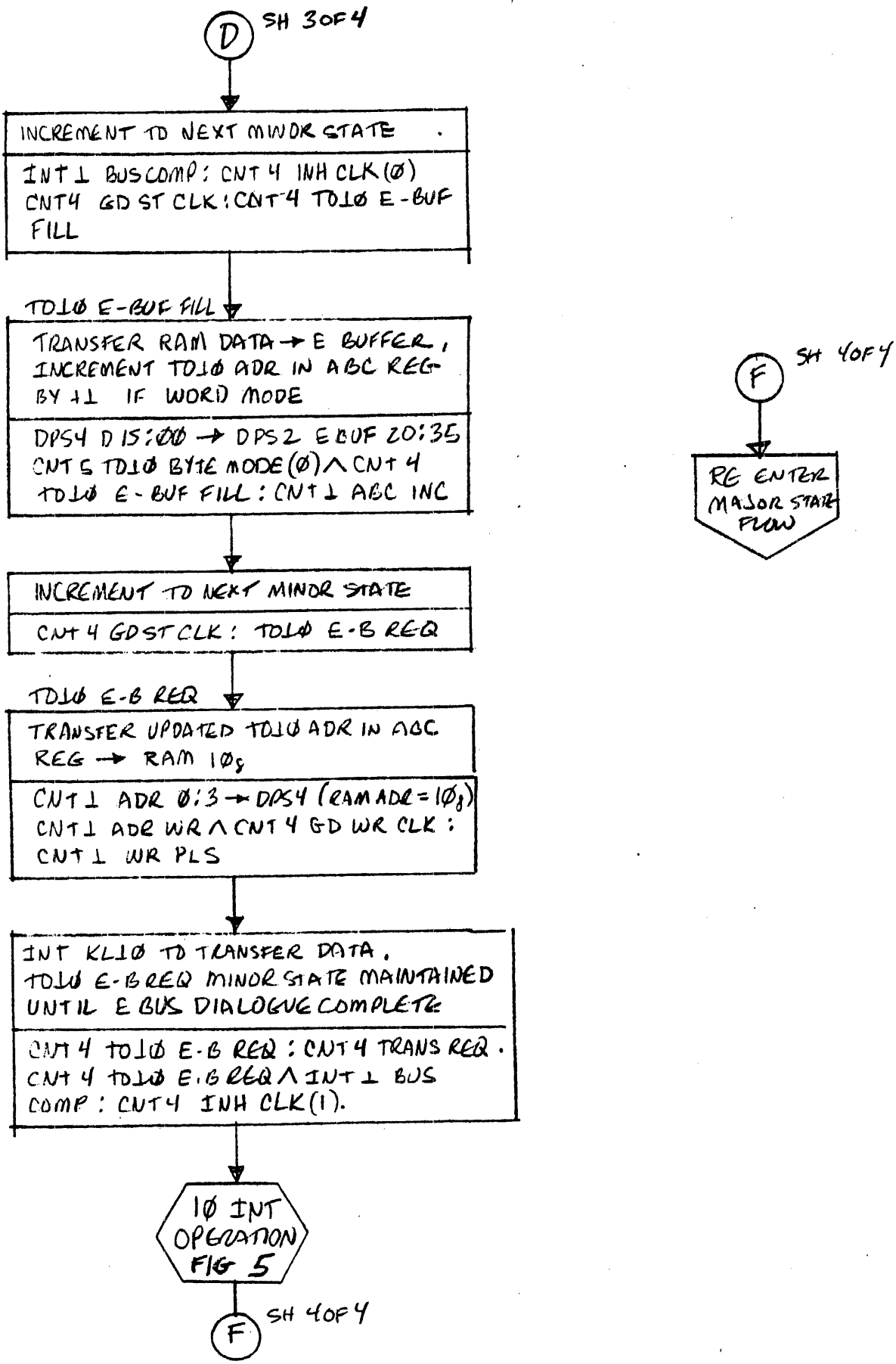
FIGURE 2. TD1Ø TRANS SHEET 1 OF 4


B
SH 1 OF 4
TD1Ø BC ADD
INC BC IN ABC REG BY +1
CNT1 BC INC ∧ CNT 4 GD WR CLK;
CNT1 ABC INC
INCREMENT TO NEXT MINOR STATE
CNT 4 GD ST CLK: CNT 4 TD10 BC INC
TD1Ø BC INC
TRANSFER UPDATED BC → RAM Ø68
CNT1 ADR Ø:3 → DPS4 (RAM ADR=68)
CNT1 EC STORE ∧ CNT 4 GD WR CLK:
CNT1 WR PLS.
INCREMENT TO NEXT MINOR STATE
CNT4 GD ST CLK : CNT4 TD1Ø ADR RD
TD1Ø ADR RD
TRANSFER TD1Ø ADR → ABC REG
CNT1 ADR Ø:3 → DPS4 (RAM ADR=1Ø8)
CNT1 ADR RD ∧ CNT 4 GD SHF CLK :
CNT1 ABC LOAD
INCREMENT TO NEXT MINOR STATE
CNT4 GD ST CLK: CNT4 TD1Ø FL WR
C
SH 3 OF 4
E
SH 1 OF 4
TD1Ø BC RD
INT 1Ø FOR NEW BYTE COUNT.
IF TD1Ø "I" BIT SET, INT 11
FOR NEW ADDRESS.
CNT1 TD1Ø DONE SET : DPS5 1Ø TD1Ø
NORM TERM (1) : DPS5 10 INTER ACT.
CNT1 TD1Ø DONE SET ∧ DPS5 TD1Ø
I BIT : DPS5 11 TD1Ø NORM TERM :
DPS5 11 INT.
I BIT
N
Y
IO INT OPERATION
IO INT OPERATION AND BR OPERATION
RE ENTER MAJOR STATE FLOW



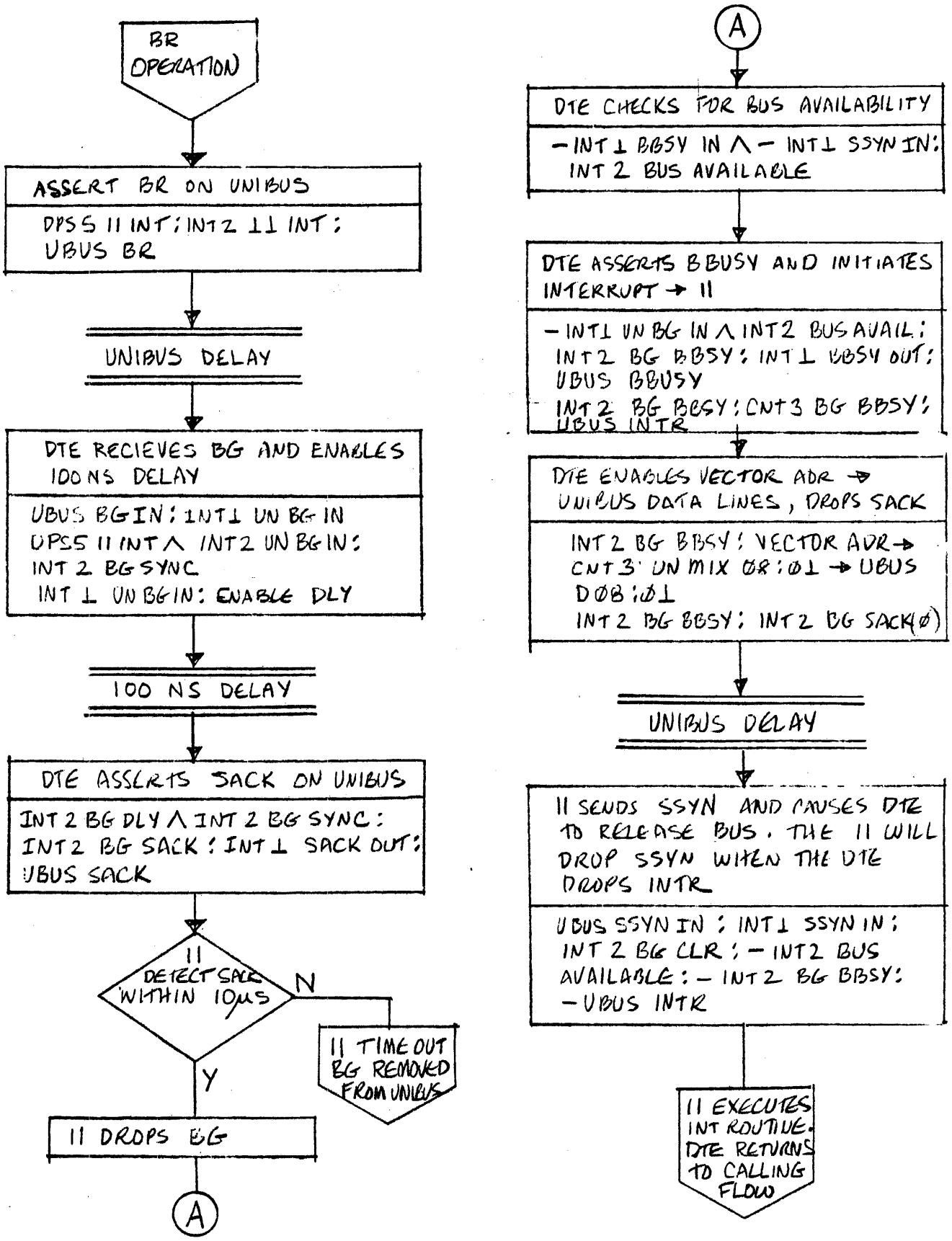
TD1Ø TRANS SHEET 3 OF 4

D2-21



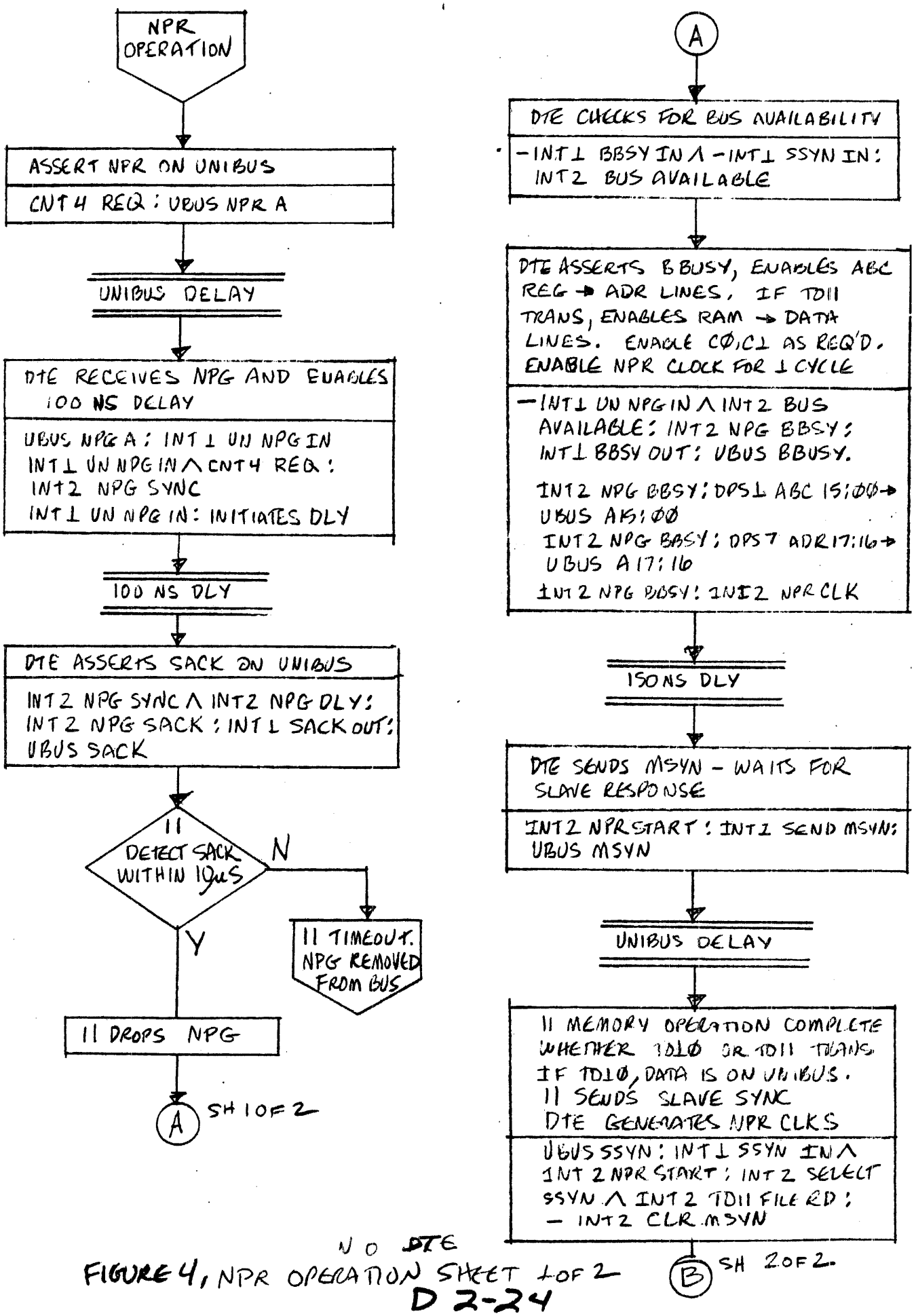
TO1Ø TRANS SHEET 4OF 4

D 2-22



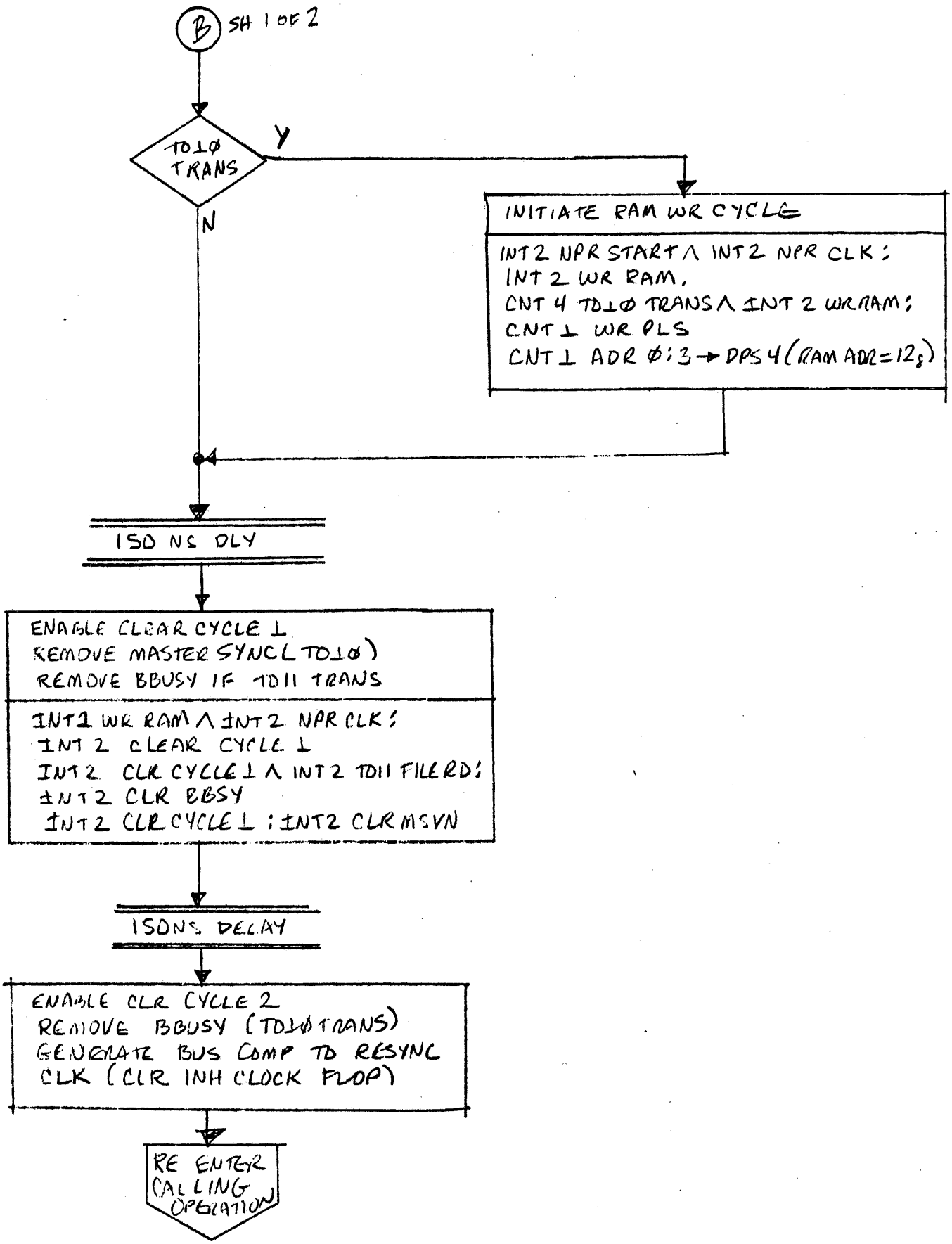
NO DTE-

FIGURE 3, BR REQUEST OPERATION SH ⊥ OF ⊥



NO DTE

FIGURE 4, NPR OPERATION SHEET 1 OF 2



NPR OPERATION SHEET 2 OF 2

D 2-25

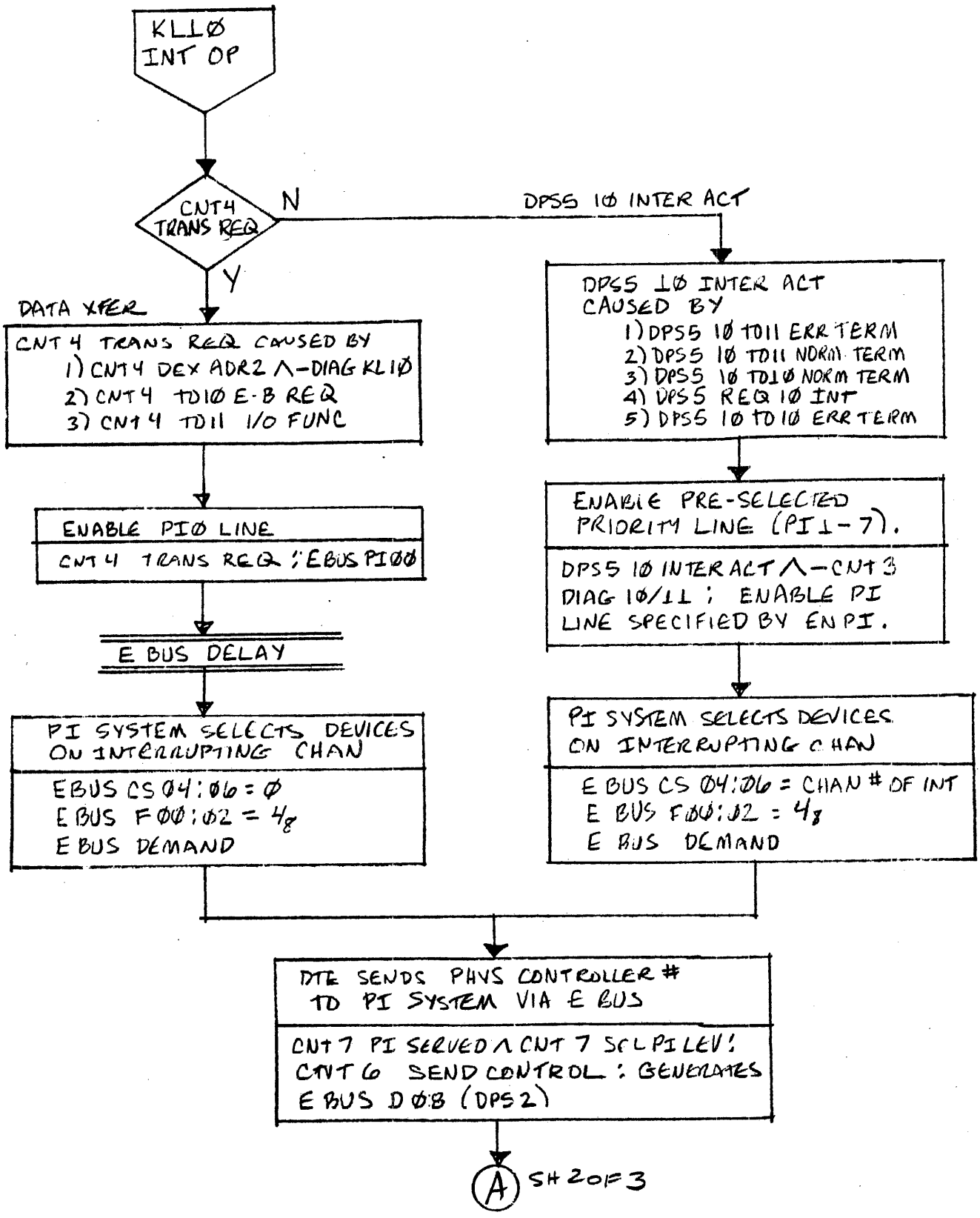
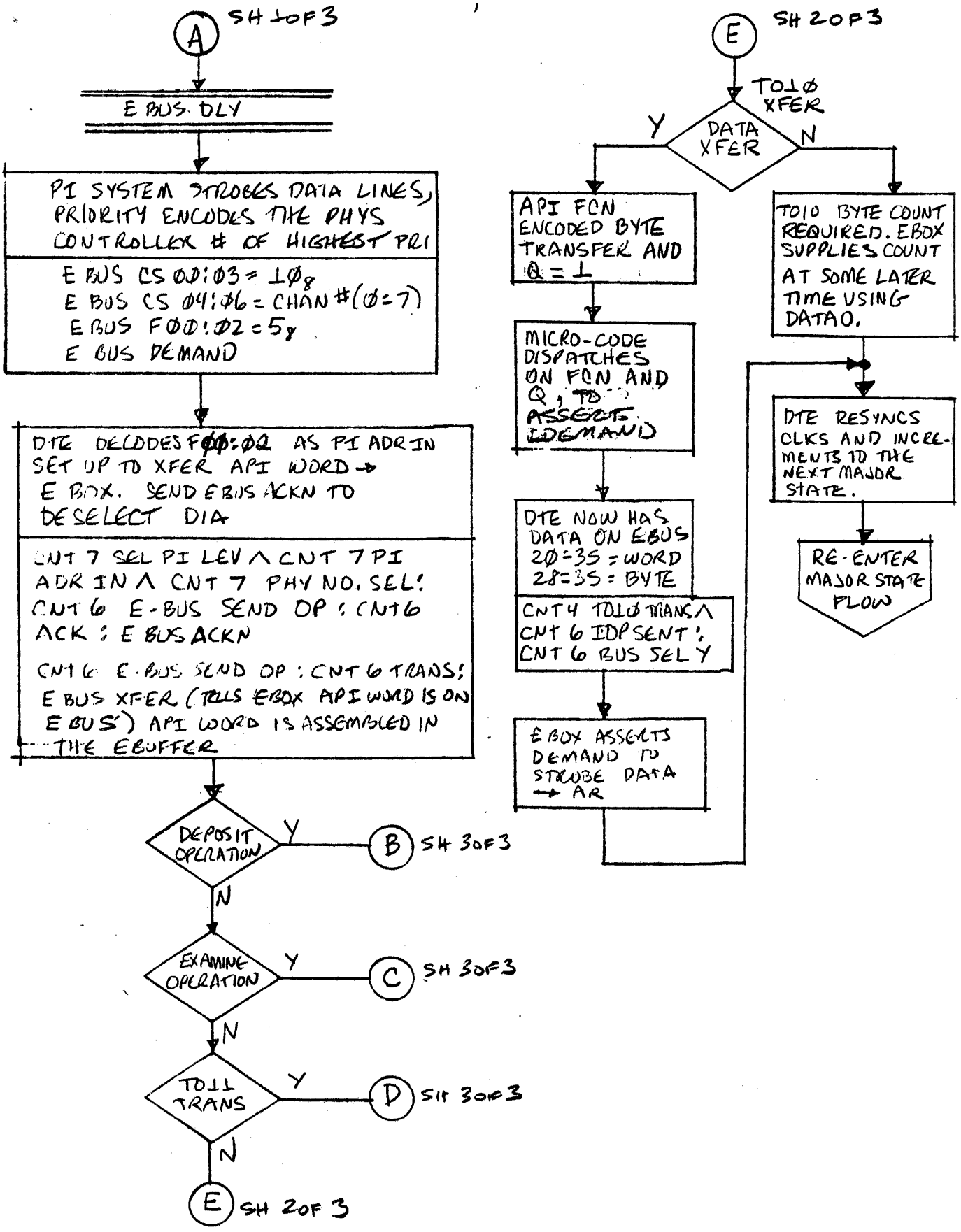
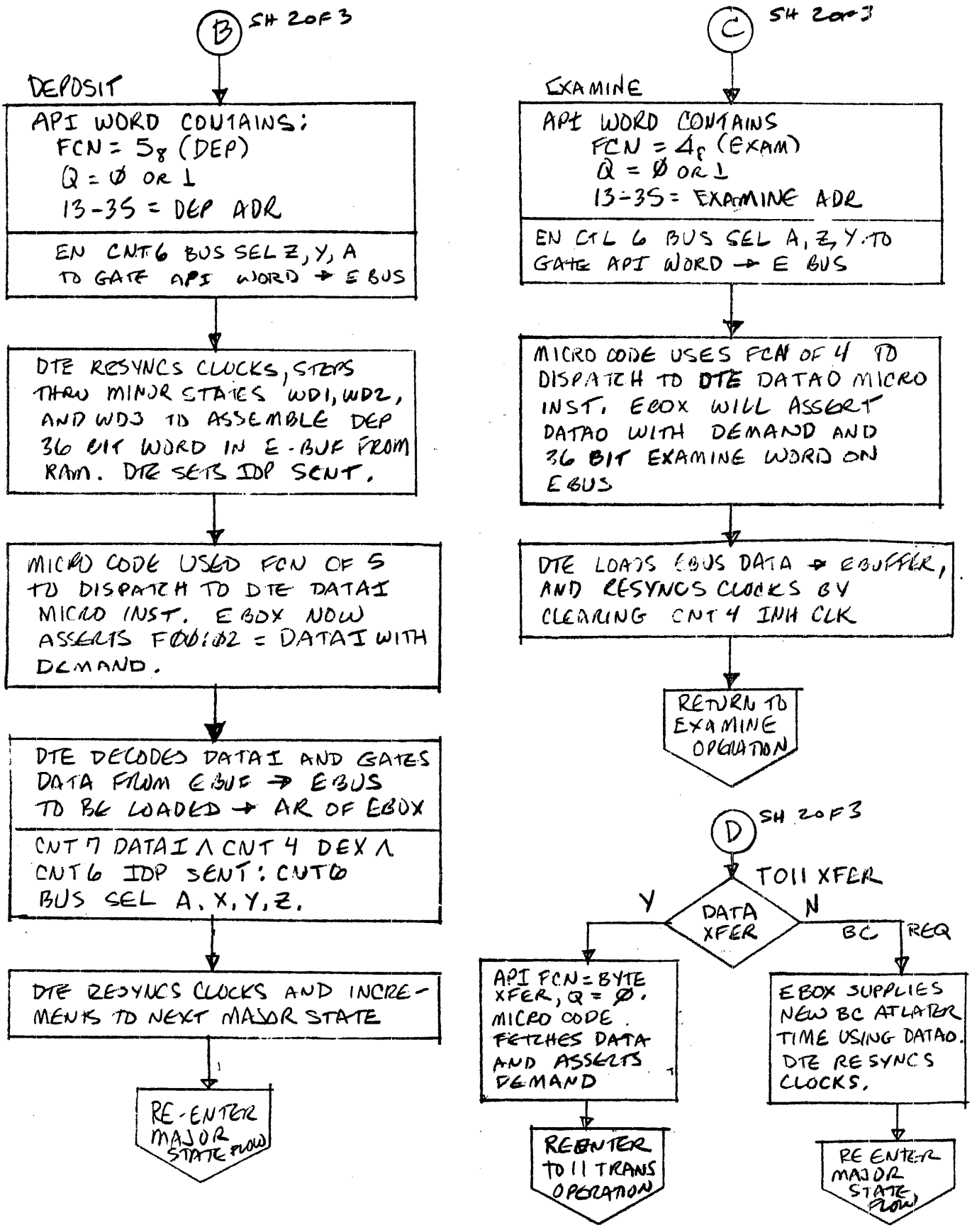


FIGURE 5. KL10 INTERRUPT SHEET 1 OF 3

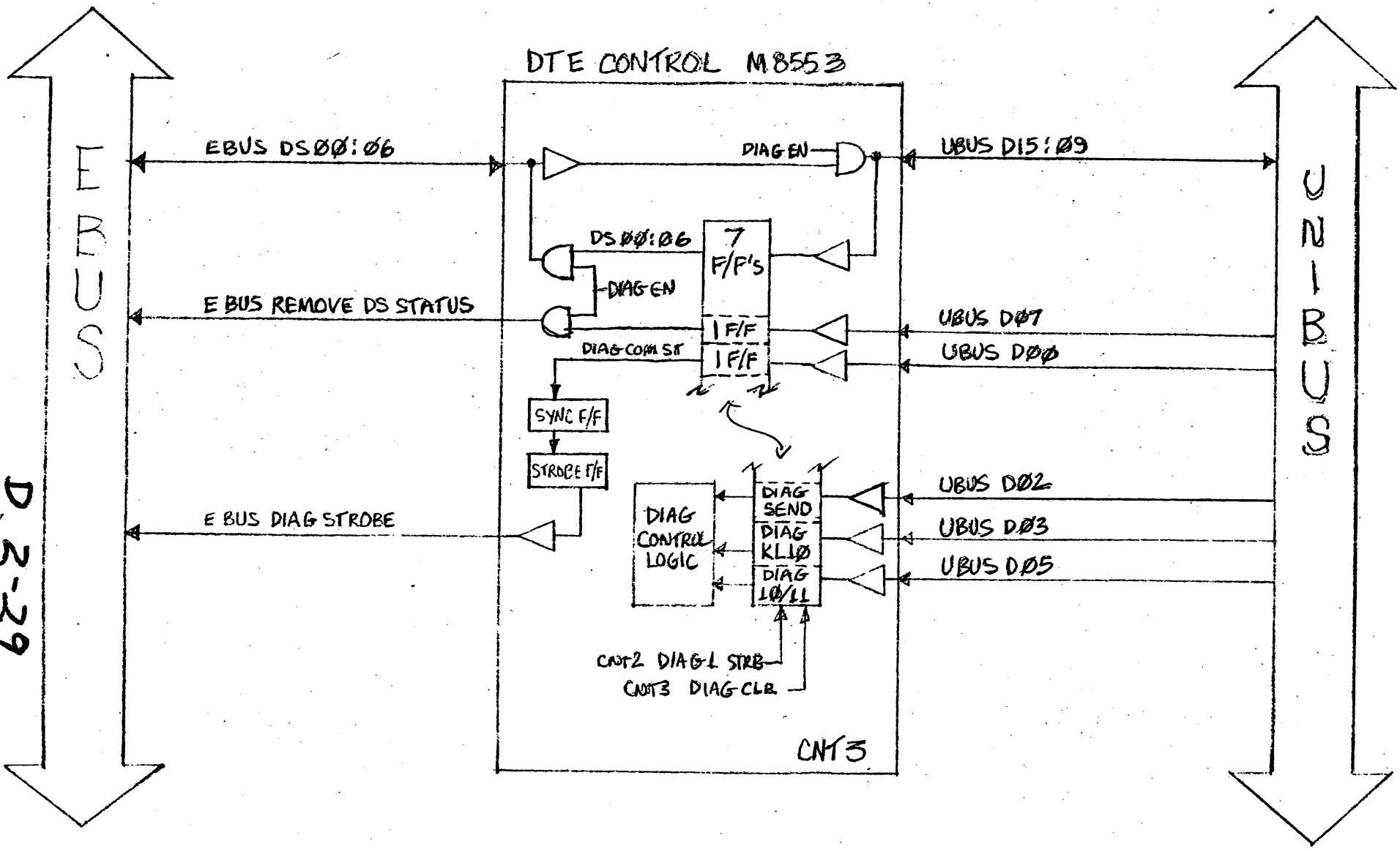


KL10 INTERRUPT SHEET 2 OF 3

D 2-27



KL10 INTERRUPT SHEET 3 OF 3



DTE DIAGNOSTIC BUS

D 2-29
Fig 8 & 9 Page 50 Print supp.

THERE ARE 3 WAYS TO USE THE DIAGNOSTIC BUS. THEY ARE;

1. DIAG CPU STATUS READ
2. DIAG FUNCTION ONLY
3. DIAG FUNCTION WITH 36 BIT DATA XFER
   a) → KL10
   b) → PDP-11/40

(1) DIAG CPU STATUS READ

PDP-11/40 DOES A LOAD REGISTER OPERATION TO DIAG 1, WITH:

UBUS D07 = 0 = - EBUS REMOTE STATUS, ALL OTHER UBUS D BITS = 0

DPS6 ADR 4:1 : CNT 2 SEL DIAG 1; CNT 2 DIAG 1 STRB; ENABLE UBUS DATA → DIAG 1 REG (CNT 3)

AFTER 1 μSEC, PDP-11 CAN READ LINES WITH A DIAG REG 1 READ OP (DATI → DIAG 1) TO ENABLE BASIC CPU STATUS → UNIBUS D15:9. STATUS BITS ARE:

DS	UBUS D	USAGE (1)	
00	15	0	NOT USED
01	14	0	
02	13	0	
03	12	0	
04	11	KL CLK ERROR STOP	
05	10	KL RUN F/F	
06	09	HALT – μCODE IS IN HALT LOOP	

11 PROGRAM CAN EVALUATE DATA

(2) DIAG FUNC ONLY

THE PDP-11/40 LOADS DIAG 1 WITH THE DESIRED FUNCTION, (UBUS IS:09 → DS 00:06) AND UBUS D00 = 1 (DIAG COMMAND START). D07 = 1 (REMOTE STATUS)

(3) → (A) SH 2 OF 3

THE ENCODED FUNCTION IS PLACED ON THE E BUS DS LINES. E BUS DIAG STROBE IS THEN ASSERTED TO INDICATE THAT THE DS LINES ARE STABLE AND THE INDICATED FUNCTION SHOULD BE PERFORMED.

CNT 3 DIAG DS 00:06 → E BUS DS 00:06 (BIT 07 = 1)
CNT 3 DIAG COM START (1) ∧ CNT 7 WR CLK (1) : CNT 3 SYNC (1)
CNT 3 SYNC (1) ∧ CNT 7 WR CLK (1); CNT 3 STROBE (1); E BUS DIAG STROBE

CLEAR DIAG REG 1 AND DROP STROBE

CNT 3 STROBE (1) ∧ CNT 7 WR CLK (1); CNT 3 DIAG CLR (1); CLRS DIAG REG 1
CNT 3 DIAG COM START (0); CNT 3 STROBE (0)

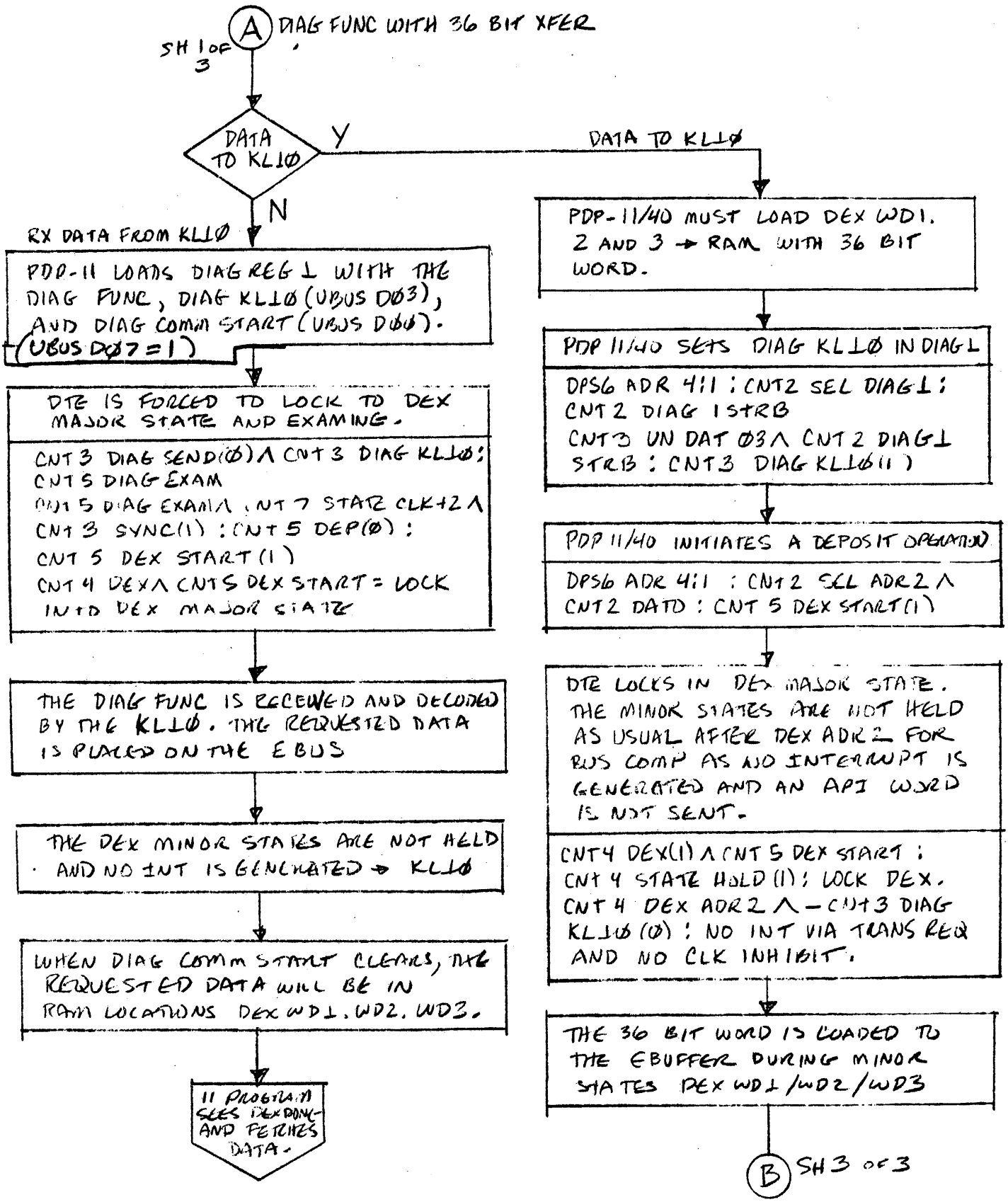
DIAG CLR IS DROPPED AT WR CLK + 1

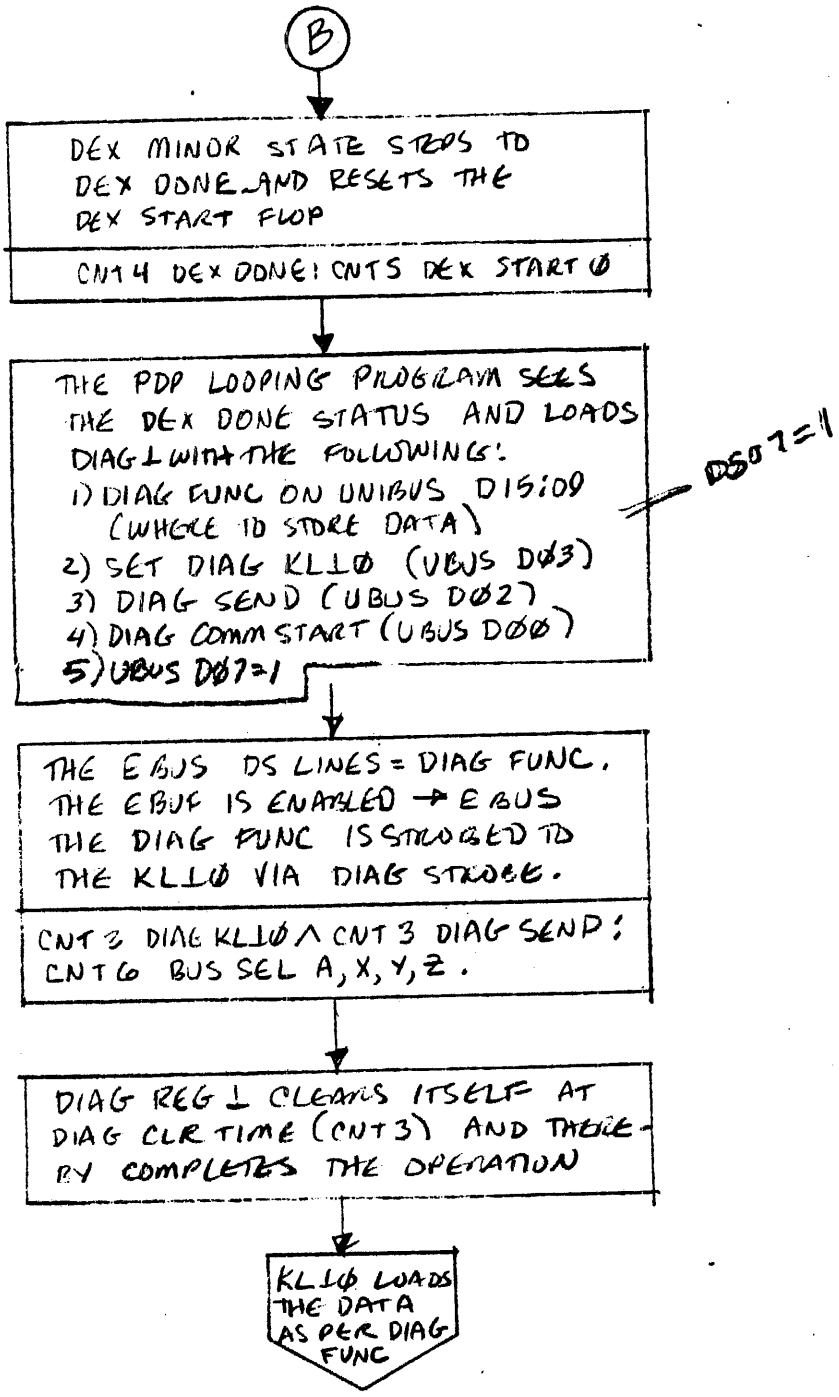
CNT 7 WR CLK + 1 : CNT 3 DIAG CLR (0)

DIAG FUNCTION IS EXECUTED BY KL

DTE/DIAGNOSTIC BUS OPERATION SHEET 1 OF 3

D 2-30


SH 1 OF 3
A
DIAG FUNC WITH 36 BIT XFER
DATA TO KL10
Y
N
DATA TO KL10
RX DATA FROM KL10
PDP-11 LOADS DIAG REG 1 WITH THE DIAG FUNC, DIAG KL10 (UBUS D03), AND DIAG COMM START (UBUS D00). (UBUS D07 = 1)
PDP-11/40 MUST LOAD DEX WD1, 2 AND 3 → RAM WITH 36 BIT WORD.
DTE IS FORCED TO LOCK TO DEX MAJOR STATE AND EXAMING.
CNT 3 DIAG SEND(0) ∧ CNT 3 DIAG KL10; CNT 5 DIAG EXAM
CNT 5 DIAG EXAM ∧ CNT 7 STATE CLK+2 ∧ CNT 3 SYNC(1) : CNT 5 DEP(0) : CNT 5 DEX START (1)
CNT 4 DEX ∧ CNT 5 DEX START = LOCK INTO DEX MAJOR STATE
PDP 11/40 SETS DIAG KL10 IN DIAG 1
DPS6 ADR 4:1 : CNT 2 SEL DIAG 1; CNT 2 DIAG 1 STRB
CNT 3 UN DAT 03 ∧ CNT 2 DIAG 1 STRB : CNT 3 DIAG KL10 (1)
PDP 11/40 INITIATES A DEPOSIT OPERATION
DPS6 ADR 4:1 : CNT 2 SEL ADR 2 ∧ CNT 2 DATO : CNT 5 DEX START (1)
THE DIAG FUNC IS RECEIVED AND DECODED BY THE KL10. THE REQUESTED DATA IS PLACED ON THE EBUS
DTE LOCKS IN DEX MAJOR STATE. THE MINOR STATES ARE NOT HELD AS USUAL AFTER DEX ADR 2 FOR BUS COMP AS NO INTERRUPT IS GENERATED AND AN API WORD IS NOT SENT.
CNT 4 DEX (1) ∧ CNT 5 DEX START : CNT 4 STATE HOLD (1); LOCK DEX. CNT 4 DEX ADR 2 ∧ — CNT 3 DIAG KL10 (0) : NO INT VIA TRANS REQ AND NO CLK INHIBIT.
THE DEX MINOR STATES ARE NOT HELD AND NO INT IS GENERATED → KL10
WHEN DIAG COMM START CLEARS, THE REQUESTED DATA WILL BE IN RAM LOCATIONS DEX WD1, WD2, WD3.
THE 36 BIT WORD IS LOADED TO THE EBUFFER DURING MINOR STATES DEX WD1/WD2/WD3
11 PROGRAM SEES DEX DONE AND FETCHES DATA.
B
SH 3 OF 3


B
DEX MINOR STATE STEPS TO DEX DONE AND RESETS THE DEX START FLOP
CNT 4 DEX DONE: CNTS DEX START Ø
THE PDP LOOPING PROGRAM SEES THE DEX DONE STATUS AND LOADS DIAG⊥ WITH THE FOLLOWING:
1) DIAG FUNC ON UNIBUS D15:09 (WHERE ID STORE DATA)
2) SET DIAG KL⊥Ø (UBUS DØ3)
3) DIAG SEND (UBUS DØ2)
4) DIAG COMM START (UBUS DØØ)
5) UBUS DØ7=1
DØ7=1
THE EBUS DS LINES = DIAG FUNC. THE EBUF IS ENABLED → EBUS THE DIAG FUNC IS STROBED TO THE KL⊥Ø VIA DIAG STROBE.
CNT 2 DIAG KL⊥Ø ∧ CNT 3 DIAG SEND: CNT 6 BUS SEL A, X, Y, Z.
DIAG REG⊥ CLEARS ITSELF AT DIAG CLR TIME (CNT 3) AND THEREBY COMPLETES THE OPERATION
KL⊥Ø LOADS THE DATA AS PER DIAG FUNC

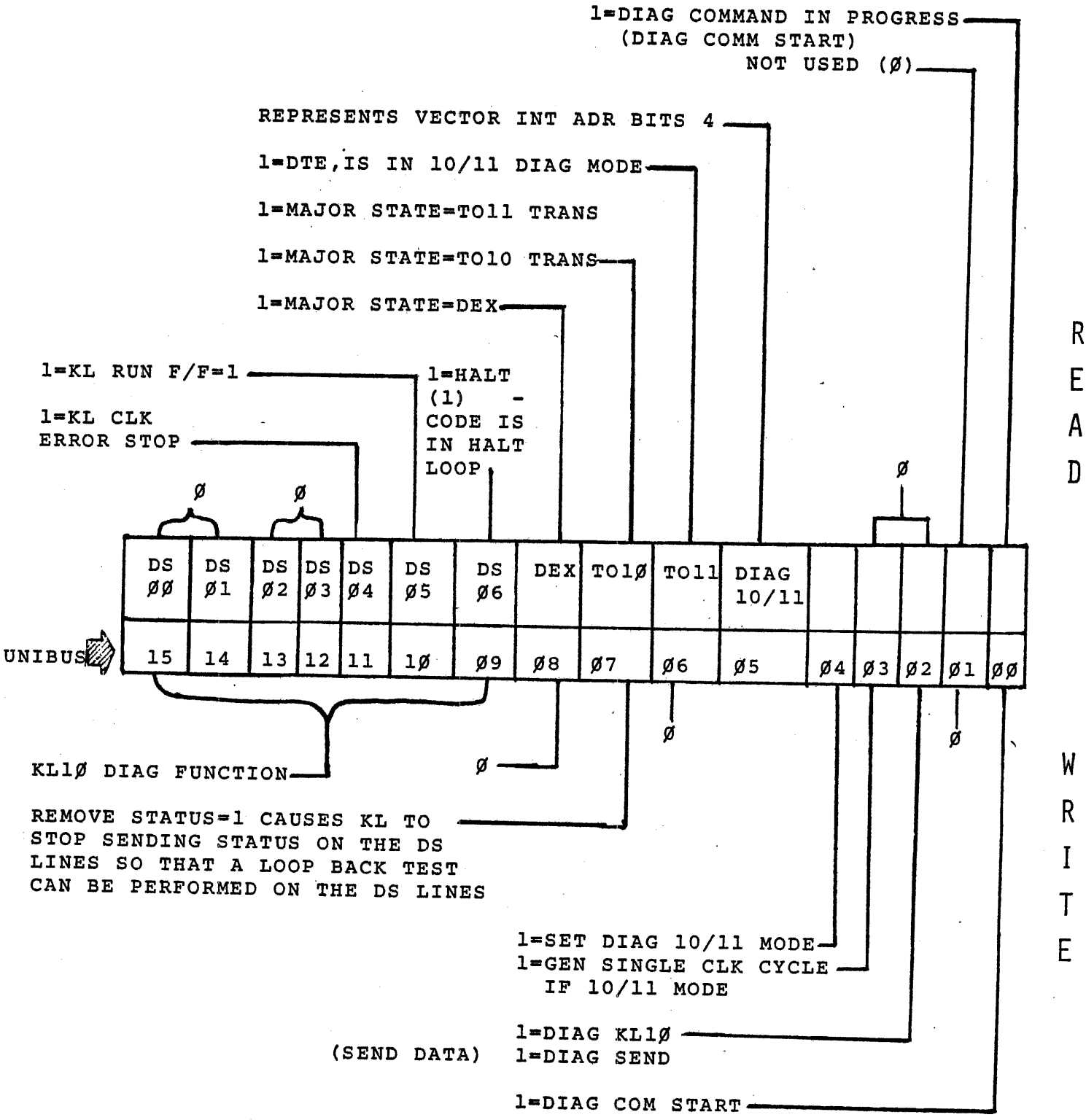


FIGURE 1 - DIAGNOSTIC WORD 1

 DTE 1-30,31

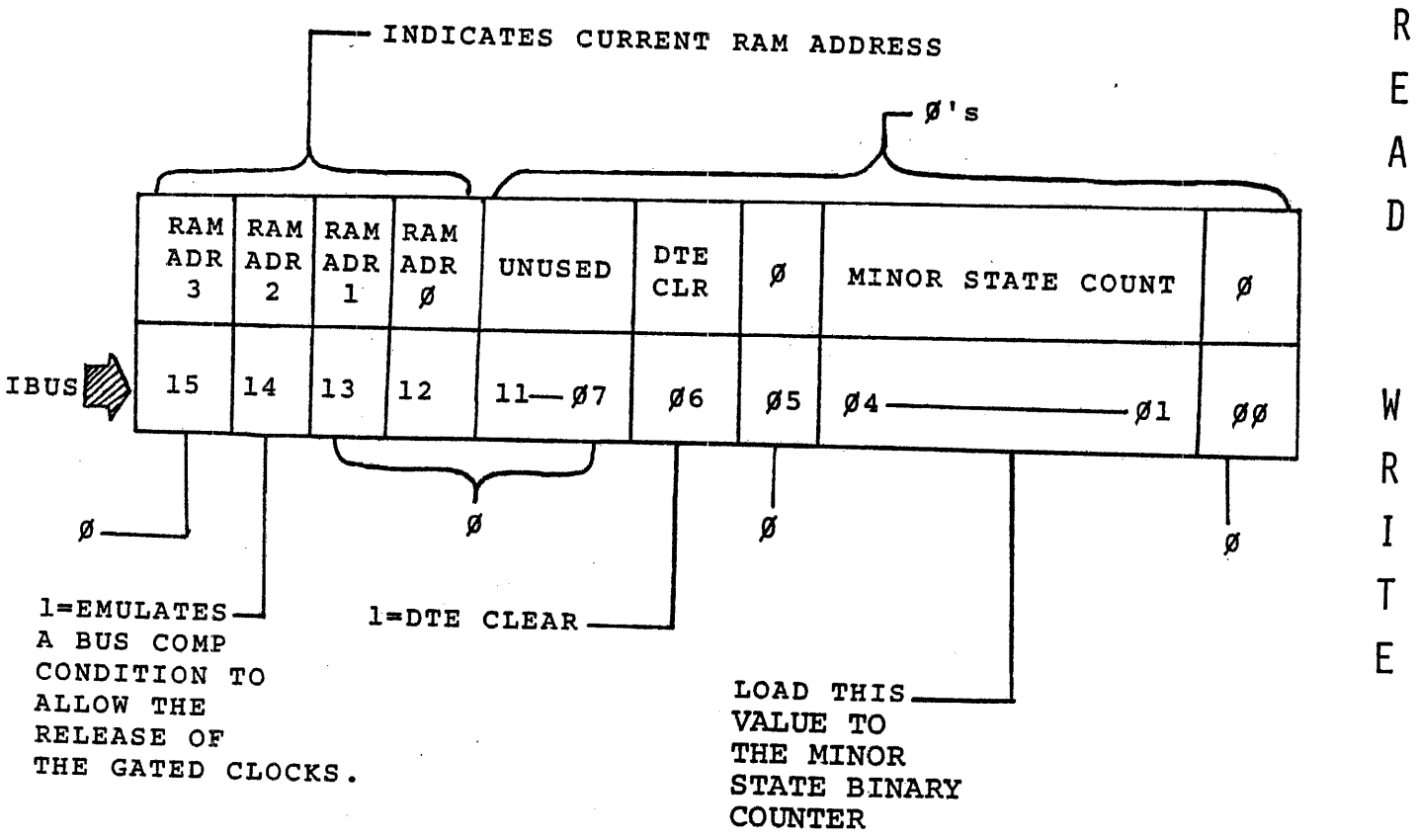


FIGURE 2 DIAG WORD 2

DTE 1-32

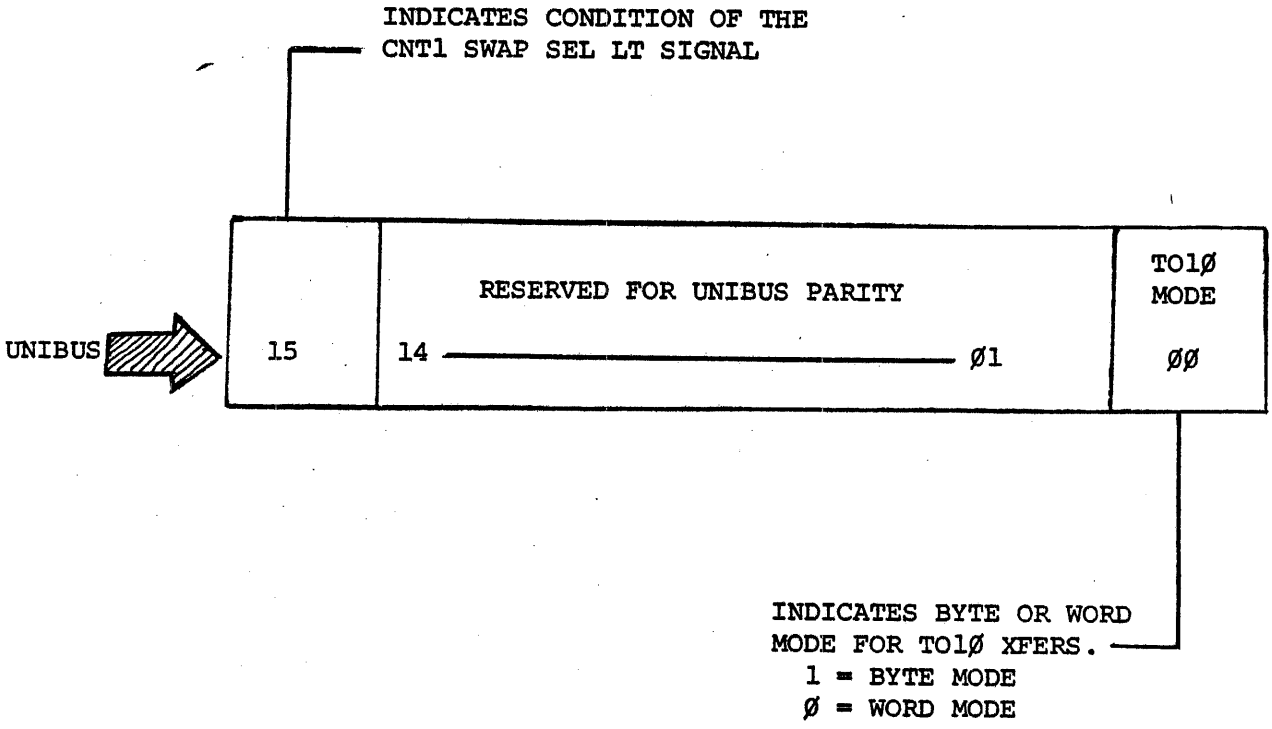


FIGURE 3 DIAG WORD 3

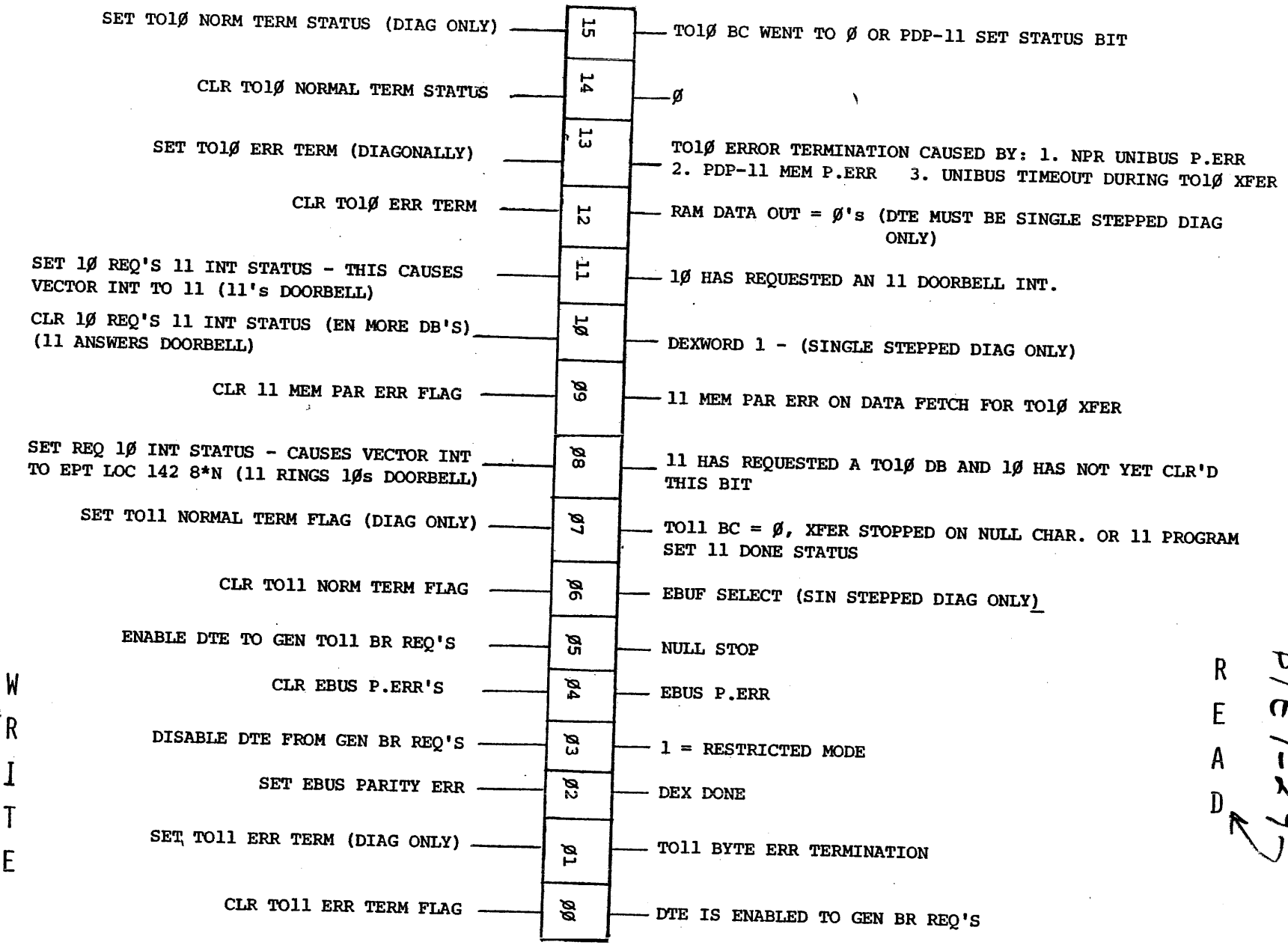


FIGURE 4 STATUS WORD

D 2-36 DTE 1-28

DTE 1-29

KL-10 CONO, CONI FORMATS

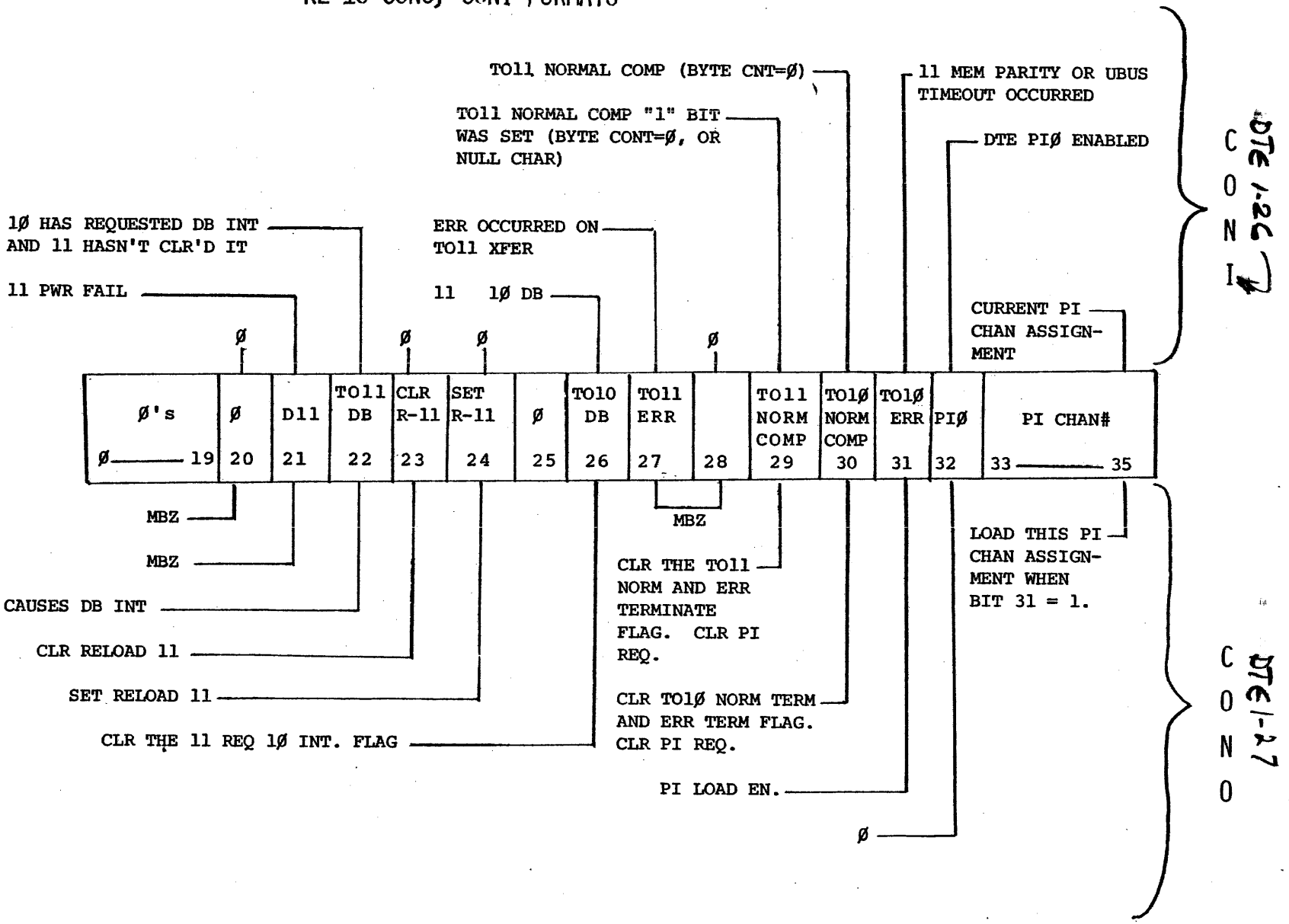
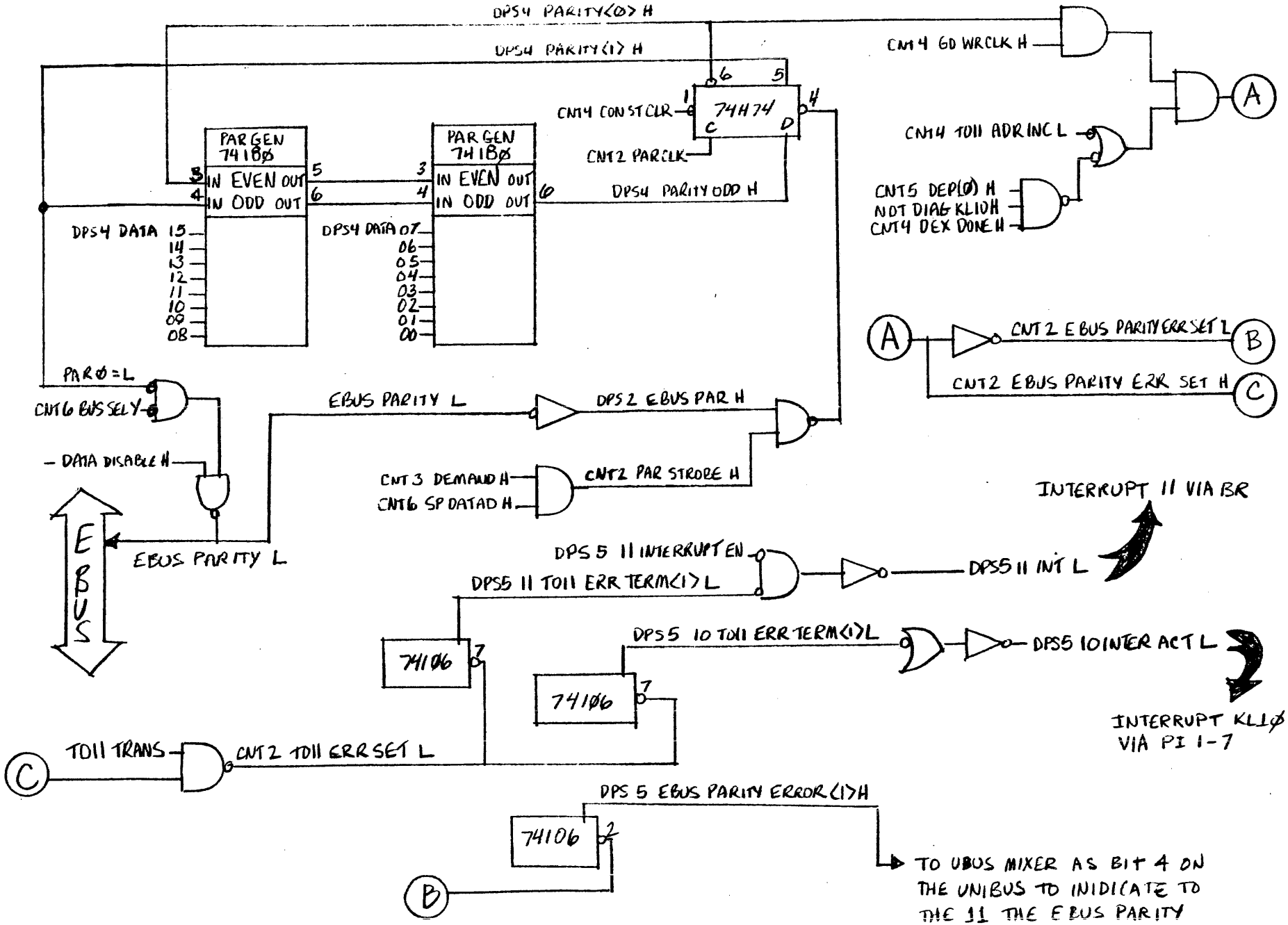


FIGURE 5, KL10 CONO, CONI FORMATS



D 2-38

EBUS PARITY LOGIC

The DTE will generate and check parity on deposit examine data and byte string data. It will not generate parity for CONI's to the DTE and will not detect parity for CONO's or DATAO's for the DTE. The parity will be odd parity.

When a parity error occurs, the bad data is in the RAM and can be retrieved by the 11/4Ø for error reporting.

Turn to Figure 1, E-Bus parity logic. This figure will be used to help describe the generation and detection of E-Bus parity.

First, the parity generation. The parity will be generated for 36 bits in a deposit operation and for 16 bits in a TO1Ø transfer. If the TO1Ø transfer specifies byte mode, the parity generation is on 16 bits, therefore to insure correct parity on the transmitted byte, the other 8 bits in the TO1Ø data RAM location must equal Ø's.

On Figure 1, locate the two 7418Ø parity

generation chips.

The data input to the chips is the 16 bit RAM output. Therefore, to generate parity for a 36 bit deposit operation, the parity is accumulated each time a DEX word is read from the RAM and placed into the E-Buffer. It is important to note that the unused bits in DEX Word 1 must be zeroed or bad parity could occur. For TO1Ø byte transfers, the parity is generated at the time the byte or word is loaded to the E-Buffer to be transmitted.

Assume a 16 bit transfer is going to take place and the word to be transferred is being read from the RAM to the E-Buffer. The RAM data is also felt by those two parity chips. Also, assume for simplicity, the 16 bits are all zeroes or even parity. Therefore all inputs, 15 through ØØ will be low to the chips. Pins 3 and 4 on the chips will also help determine the chip output. These Signals come from the parity flip flop to the right of the parity chips. This parity flip flop is forced to a Ø condition at each control 4 con state clear

time. This is generated each time a major state toggles. Therefore, the parity flip flop will always start in the Ø condition. That makes output 5 low and output 6 high.

Following the signals back again to the left-most parity chip, this allows the in Even signal at pin 3 to be high. With even data on 15 through Ø8 and the in even signal high, the chip will output a high on pin 5 and a low on pin 6. With even data on 7 through Ø of the next parity chip and its in EVEN high, pin 6 of this chip will be asserted low. This low is presented to the D input of the parity flip flop. When it's clocked with control 2 parity clock, the flip flop will remain at a Ø condition. This flip flop will now be used to generate the odd parity bit to the E-Bus. Follow pin 5 of the flip flop to the left and down to an "and" gate. This is at a low level. When control 6 bus select Y is generated to gate the data to the E-Bus it will also be asserted low to this "and" gate. The high from this "and" gate is passed inverted to the

E-Bus as E-Bus parity L and thereby asserts the odd parity for the even data just sent out.

In the case of a 36 bit deposit operation, control 6 bus Sel Y is not generated until the entire 36 bit word has been packed to the E-Buffer from the RAM. Therefore, the parity bit generation is an accumulative process between the parity flip flop and the parity chips. The parity flip flop will be clocked with the parity chips output each time a DEX word is read from the RAM to the E-Buffer. After the 3 DEX words are read from the RAM, the bus select Y signal will allow the accumulated parity bit to be transmitted along with the data.

The E-Bus parity line will be asserted low to represent a 1 condition for the parity bit.

At this time, you should stop the tape and review the parity generation sequence. It may be helpful to you to associate Figure 1 with the parity logic in the prints. So stop the tape and do this.

Let's take a look at the logic used in parity detection. Again, we'll assume a condition. Assume a TO11 transfer is taking place, word mode is present, and the 16 bits contain an even number of 1's - thereby making the E-Bus parity bit active to force odd parity.

On Figure 1, the E-Bus parity line will be asserted low, representing the active parity bit. This is inverted high and gated to the parity flip flop at demand and special DATAO time. Pin 4 of the parity flip flop, the preset pin, is asserted low. This forces the flip flop to the ones condition and therefore "captures" the parity bit from the E-Bus. When the 16 bits of data is written to the RAM from the E-Buffer, the RAM outputs will reflect the RAM inputs, therefore the data being written can be felt at the parity chips. Assume data lines 15 through 8 contain an even number of ones. Pin 3 on this chip is low and pin 4 is high, reflecting the parity flip flop outputs. Therefore, with the in odd line active and even data on the data lines, the

leftmost parity chip with assert pin 6 at a high level. Assuming bits 7 through zero are even, the high level on this chip at pin 4 will force the chip to assert its pin 6 at a high level. The DPS4 parity odd signal being high when clocked into the parity flip flop will force the flip flop to a 1's condition, making pin 6 low and 5 high. Follow pin 6 to the right from the flip flop. If a parity error occured pin 6 would be high. In our case, the odd parity was computed correctly making pin 6 low. Let's assume that the parity was incorrect and pin 6 was asserted high at gated write clock time, making the "and" gate. The "and" gate just before connection A will propagate the parity error signal for two different operations as can be seen by the inputs on the "or" gate here. One of the times is at TO11 minor state ADR increment. This is the last thing to occur for a TO11 transfer. The other condition for propagating the parity error is at DEX done time for an examine not occurring in KL1Ø diagnostic mode. Therefore, a parity error is generating at the

completion of a TO11 trans or examine operation if the DPS 4 parity flip flop is in a Ø condition making pin 6 high. You should be able to follow the rest of the sequence for generating the appropriate interrupts on your own.

DTE-20

1.1 INTRODUCTION

Each central processor in a KL10 system may have from one to four PDP-11 processors attached, each serving as a "front end" processor. Each PDP-11 is connected to the KL10 by a separate interface called the DTE20 Console Processor Interface, or simply the 10-11 Interface. The following are some of the possible front end functions:

1. Handling unit record equipment
2. Handling asynchronous communications equipment
3. Handling synchronous communications equipment
4. Providing a long term power line frequency clock
5. Diagnosing the KL10 Central Processor and other functional components in the system
6. Running a dedicated real-time data acquisition system
7. Bootstrapping the KL10 system.

In terms of basic features, the DTE20 generates parity for Deposit data and detects parity errors for both Examine data and byte transfers over the EBus. The DTE20 connects to the PDP-11 as a standard Unibus peripheral and communicates via interrupt or device address. Up to four DTE20s may be connected to a PDP-11. In a system consisting of four KL10 Central Processors, there may be four PDP-11/40 processors, where each processor can communicate with all KL10s in the system. It is possible to have up to four DTE20s on each PDP-11 in the KL10 system, and each KL10 processor may have 1, 2, 3, or 4 DTE20s connected to it via the EBus.

The DTE20 uses the NPR (Direct Memory Access) and BR (Vector Interrupt) features of the PDP-11. In addition, the DTE20 contains logic to detect PDP-11 core memory parity errors during NPR transfers, provided that the memory being accessed contains the parity option (MFU11 UP).

The DTE20 provides the following capabilities:

1. Console functions at Examine and Deposit, restricted or unrestricted.

2. Doorbell function, where the PDP-11 can interrupt the KL10 Central Processor and vice versa.

3. High speed simultaneous two-way transfer of variable byte data between the PDP-11 and KL10 memory.

4. Diagnostic bus for the PDP-11 to diagnose the KL10.

5. KL10-initiated bootstrap startup of the PDP-11 mechanism (diagnostic bus) to load the microcode into the CRAM, execute PDP-10 instructions, and start or stop the KL10 Central Processor.

The following terminology will give some perspective on the front end and its relationship to the DTE20.

PDP-11 Communication Region – This region consists of an area of KL10 core memory defined by the deposit relocation and protection word in the Executive Process Table (EPT). This area is written by the PDP-11 using protected deposits, and read by the KL10. It is used for coordination of status, preparing for byte transfer operations, and passing limited amounts of data. Each PDP-11 in the system has a separate communication region in the KL10 memory, which it alone can modify.

KL10 Communication Region – This region is defined solely by the KL10 software and is separate from the PDP-11 communication region. It can be written by the KL10, but may be read by the PDP-11 using protected Examines. This area is used to coordinate status, prepare byte transfer operations, and pass limited amounts of data (Figure 1-1).

DTE 1-2

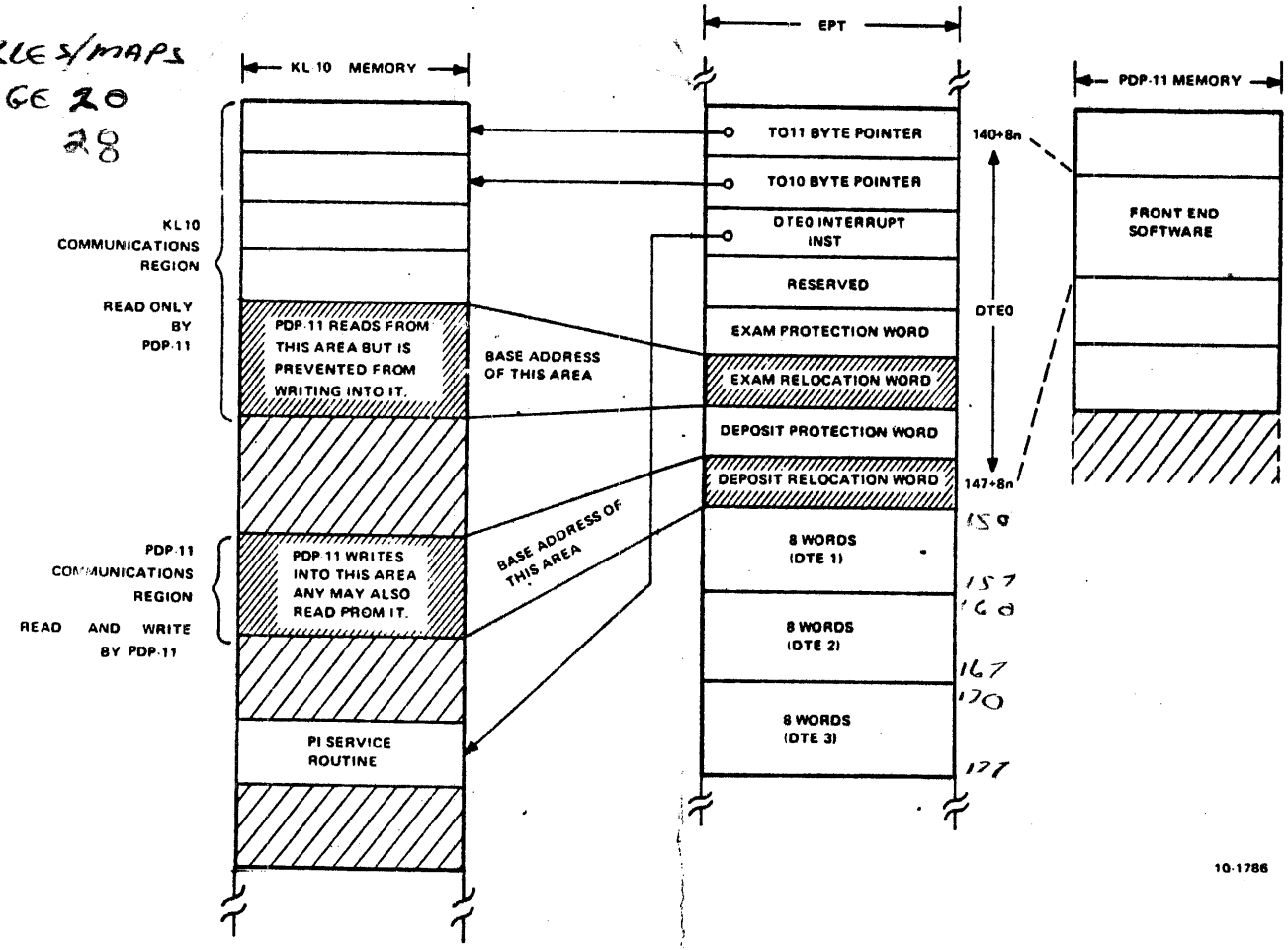
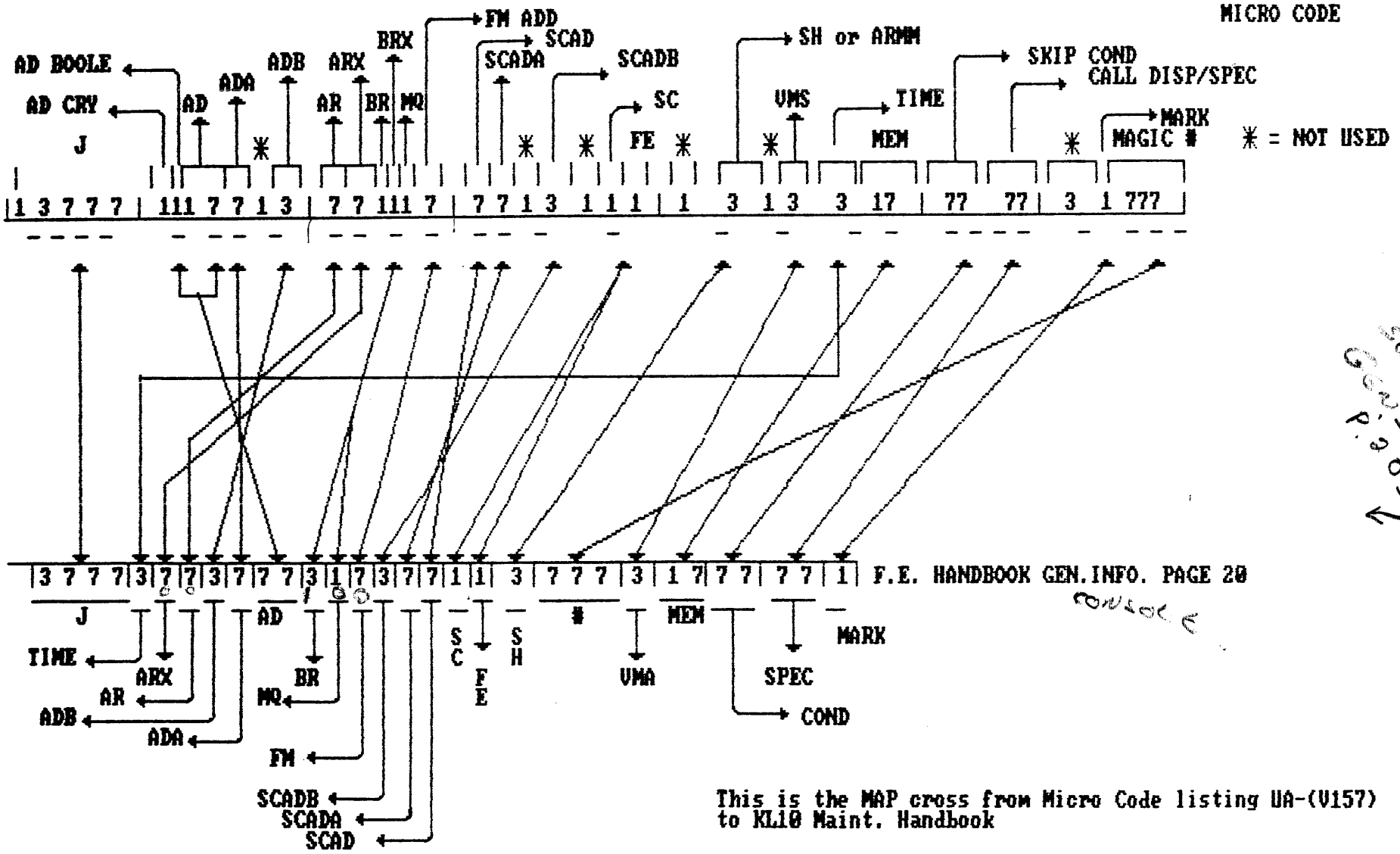


Figure 1-1 Overview Communications Region



This is the MAP cross from Micro Code listing UA-(V157)
to KL10 Maint. Handbook

by WILLIAMSON

1080,2040,2060 ENGINEERING FUNCTIONAL SPEC - CHAP 2.8

TO: KL10 LIST, J. PARSLOW (200 FILE)

TITLE: KL10 PAGING - REV 2

STATUS: FINISHED

FILE: [EFS]CH2S08.SPC

PDM #: 200-200-037-02

DATE: 5 DEC 75

SUPERSEDED MEMOS: KL10 PAGING PROPOSAL, D. MURPHY, 5 JUNE 73
NEW PAGING DESIGN, D. MURPHY, 31 MAY 74

SUPERSEDED SPECS: NONE

ENGINEER: D. MURPHY, J. LEONARD

APPROVED:

EDITOR: D. MURPHY

TYPIST: J. MCCARTHY

REVIEWED: 6 AUG 75

DISTRIBUTED:

ABSTRACT

THIS CHAPTER DESCRIBES THE KL10 PAGING FACILITIES. THESE FACILITIES ARE A CONSIDERABLE ADVANCE OVER KI10 PAGING, ALTHOUGH THE KL10 ALSO PROVIDES A KI10 PAGING MODE. KL10 PAGING PROVIDES FOR EFFICIENT PROGRAM WORKING SET MANAGEMENT AND DEMAND PAGING. IT ALSO PROVIDES FOR EXTENSIVE SHARING OF DATA AND PROGRAMS ON A PAGE-BY-PAGE BASIS.

REVISION HISTORY

REV	DESCRIPTION	CHG NO	ORIG	DATE	APPD BY	DATE
1	REVIEW CHANGES			6 AUG 75		
2	CLEANUP			5 DEC 75		

THE KL10 PAGING FACILITIES ARE INTENDED TO SUPPORT SOPHISTICATED OPERATING SYSTEM FEATURES, INCLUDING IN PARTICULAR:

1. EFFICIENT PROGRAM WORKING SET MANAGEMENT AND DEMAND PAGING.

2. EXTENSIVE SHARING OF DATA AND PROGRAMS ON A PAGE-BY-PAGE BASIS.

MUCH OF THE MECHANISM DESCRIBED HEREIN IS IMPLEMENTED BY KL10 MICROCODE RATHER THAN SPECIFIC HARDWARE. THE COMBINATION OF HARDWARE AND MICROCODE WHICH IMPLEMENTS THIS SPECIFICATION WILL BE REFERRED TO AS THE KL10 PAGER. THE KL10 ALSO SUPPORTS KI10 PAGING AS A SPECIAL MODE. KI10 PAGING IS NOT DESCRIBED IN THIS CHAPTER.

THIS PAGING DESIGN ALSO SUPPORTS THE EXTENDED ADDRESSING FACILITIES OF THE KL10 PROCESSOR, AND IT IS EXTENDABLE BEYOND THE 32-SECTION IMPLEMENTATION OF THE KL10. THE PHYSICAL CORE ADDRESS FIELDS SUPPORT UP TO 27 BITS (134 MILLION WORDS) OF PHYSICAL CORE MEMORY; THE KL10 IMPLEMENTS 22 BITS (4 MILLION WORDS) OF. THIS. THERE IS ALSO AN IMPLICIT LIMIT OF 2**23 (8 MILLION) PAGES PER DISK STRUCTURE (4 BILLION WORDS).

N.B. "CORE" IN THIS DISCUSSION IS NOT MEANT TO SPECIFY A TECHNOLOGY, BUT A MEMORY SYSTEM WITH ACCESS TIME AND CAPACITY SIMILAR TO CORE.

ADDRESS SPACES

THE USER ADDRESS SPACE IS HOMOGENEOUS AND CONSISTS OF 32 EQUAL SECTIONS. SECTION 0 IS NOT TREATED IN ANY SPECIAL MANNER BY THE PAGING FACILITIES. THE SECTION TABLE FOR THE USER ADDRESS SPACE RESIDES IN THE UPT AND CONSISTS OF 32 SECTION POINTERS. THE EXEC ADDRESS SPACE ALSO CONSISTS OF 32 EQUAL SECTIONS. ITS SECTION TABLE RESIDES IN THE EPT AND CONSISTS OF 32 SECTION POINTERS. THE MONITOR SOFTWARE CAN EFFECTIVELY DIVIDE THE EXEC ADDRESS SPACE INTO PER-PROCESS AND PER-JOB AREAS THROUGH THE USE OF INDIRECT POINTERS (SEE APPENDIX), HENCE NO SUCH DIVISION NEED BE BUILT INTO THE PAGER. IN A MULTI-PROCESSING SYSTEM, EACH CPU HAS ITS OWN EPT AND HENCE ITS OWN EXEC SECTION TABLE.

SECTION POINTERS

A SECTION POINTER REPRESENTS AN ENTIRE SECTION, I.E., A 256K-WORD ADDRESS SPACE. IT POINTS TO A PAGE TABLE WHICH IN TURN CONTAINS POINTERS REPRESENTING EACH PAGE OF THAT SECTION.

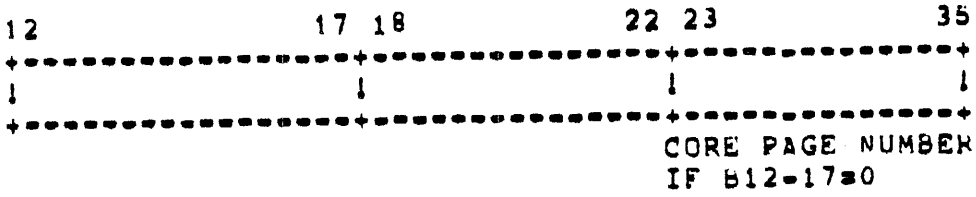
THE SECTION POINTER MAY BE IMMEDIATE, SHARED OR INDIRECT FORMAT. IN ANY CASE, IT MUST YIELD THE PHYSICAL CORE ADDRESS OF A PAGE CONTAINING A PAGE TABLE.

PAGE TABLES

A PAGE TABLE IS A PAGE CONTAINING PAGE POINTERS FOR ONE SECTION. THERE ARE 512 PAGES PER SECTION AND EACH POINTER IS ONE 36-BIT WORD. HENCE A PAGE TABLE IS A FULL PAGE.

STORAGE ADDRESSES (PAGE ADDRESSES)

A STORAGE ADDRESS IDENTIFIES A PAGE OF PHYSICAL STORAGE IN CORE OR ON SOME OTHER MEDIUM. THE FORMAT OF THE STORAGE ADDRESS DETERMINES BOTH THE MEDIUM AND THE ADDRESS WITHIN THAT MEDIUM. STORAGE ADDRESSES ARE FOUND IN PAGE POINTERS AND IN SPT ENTRIES (BELOW), AND ARE 24-BIT QUANTITIES.



IF BITS 12-17 ARE 0, THE ADDRESS REFERS TO CORE MEMORY. BITS 23-35 ARE THE PHYSICAL CORE PAGE NUMBER OF THE PAGE (B18-22 MBZ). ONLY VIRTUAL ADDRESS REFERENCES WHICH TRANSLATE TO PHYSICAL CORE ADDRESSES CAN BE COMPLETED BY THE PROCESSOR.

IF BITS 12-17 ARE NOT 0, THEN THE ADDRESS IS SOMETHING OTHER THAN PHYSICAL CORE (E.G., DISK OR DRUM). A VIRTUAL ADDRESS REFERENCE WHICH TRANSLATES TO A NON-CORE ADDRESS CANNOT BE COMPLETED BY THE PROCESSOR, AND A TRAP TO THE MONITOR MUST BE INITIATED. THE FORMAT OF NON-CORE ADDRESSES IS IRRELEVANT TO THE PAGER BEYOND THE CONVENTION THAT BITS 12-17 ARE NOT 0.

PAGE POINTERS

THERE ARE THREE TYPES OF PAGE POINTERS, IMMEDIATE, SHARED, AND INDIRECT. THE POINTER TYPE IS ENCODED IN BITS 0-2 OF THE POINTER AS FOLLOWS:

0	NO ACCESS
1	IMMEDIATE
2	SHARED
3	INDIRECT

n/u
0 1 2 3 4 5 6 7 — 11
P W C

4-7 Unused

4-7 NOT USED, RESERVED.

EACH PAGE POINTER CONTAINS ACCESS BITS WHICH DETERMINE WHAT TYPES OF REFERENCES MAY BE MADE TO THE PAGE. THE ACCESS BITS ARE HANDLED IN THE SAME WAY REGARDLESS OF WHICH OF THE THREE POINTER TYPES IS BEING INTERPRETED.

P (PUBLIC,B3) IF THIS BIT IS OFF, THE PAGE MAY ONLY BE REFERENCED BY PROGRAMS RUNNING IN CONCEALED OR KERNEL MODE. SEE KI10 REFERENCE MANUAL FOR ADDITIONAL DETAILS.

W (WRITE,B4) IF THIS BIT IS OFF, WRITE REFERENCES MAY NOT BE DONE TO THE PAGE.

C (CACHE,B6) IF THIS BIT IS ON, DATA IN THIS PAGE MAY BE PLACED IN THE CACHE.

B5,7-11 THESE BITS ARE NOT USED BY THE PAGER AND ARE RESRVED FOR FUTURE SPECIFICATION BY DEC.

IMMEDIATE POINTER

AN IMMEDIATE POINTER CONTAINS THE PHYSICAL ADDRESS OF THE ASSOCIATED PAGE IN BITS 12-35. THIS POINTER TYPE IS ALSO KNOWN AS PRIVATE SINCE THE PAGE IS PRIVATE TO THE PAGE TABLE WHICH CONTAINS THE POINTER.

SHARED POINTER

A SHARED POINTER CONTAINS AN INDEX WHICH POINTS INTO THE SPT (SPECIAL/SHARED PAGES TABLE). THE ASSOCIATED SPT ENTRY THEN CONTAINS THE PHYSICAL ADDRESS FOR THE PAGE. THE SPT ENTRY IS FOUND AT THE PHYSICAL CORE ADDRESS GIVEN BY THE SUM OF THE SPT BASE REGISTER AND THE SPT INDEX FROM THE SHARED POINTER. THIS IS KNOWN AS A SHARED POINTER BECAUSE MANY PAGE TABLES MAY CONTAIN SHARED POINTERS TO THE SAME PHYSICAL PAGE. REGARDLESS OF THE NUMBER OF PAGE TABLES HOLDING A PARTICULAR SHARED POINTER, THE PHYSICAL ADDRESS IS RECORDED ONLY ONCE IN THE SPT. HENCE THE MONITOR MAY MOVE THE PAGE WITH ONLY ONE ADDRESS TO UPDATE.

INDIRECT POINTER

THE INDIRECT POINTER IDENTIFIES ANOTHER PAGE TABLE AND A POINTER WITHIN THAT PAGE TABLE. IT CAUSES THE NEW POINTER TO BE FETCHED AND INTERPRETED. HENCE, THE INDIRECT POINTER IS USED TO MAKE A PAGE OF ONE ADDRESS SPACE EXACTLY EQUIVALENT TO A PAGE OF ANOTHER ADDRESS SPACE.

THE INDIRECT POINTER IDENTIFIES THE OBJECT PAGE TABLE USING AN

SPT INDEX. THE PHYSICAL ADDRESS OF THE PAGE TABLE IS FOUND IN THE ASSOCIATED SPT ENTRY JUST AS FOR A SHARED POINTER. THIS IS DONE SO THAT THE PHYSICAL ADDRESS OF A PAGE TABLE MAY BE KEPT IN ONE PLACE AND NEED NOT BE DUPLICATED IN ANY PAGE POINTERS. ONCE THE OBJECT PAGE TABLE HAS BEEN FOUND (AND DETERMINED TO BE IN CORE), THE PAGE NUMBER FIELD OF THE INDIRECT POINTER IS USED AS AN INDEX TO SELECT A NEW POINTER WORD FROM THE PAGE TABLE. THIS NEW POINTER MAY BE OF ANY OF THE THREE POINTER TYPES OR IT MAY BE NO-ACCESS. THE ACCESS BITS OF THE NEW POINTER ARE "AND"ED WITH THOSE OF THE INDIRECT POINTER. THUS ACCESS TO A PAGE IS PROHIBITED IF EITHER POINTER WOULD PROHIBIT IT. THE PAGER WILL INTERPRET INDIRECT POINTERS TO AN ARBITRARY DEPTH BUT MUST BE ABLE TO TERMINATE INDIRECT POINTER INTERPRETATION TO SERVICE A PRIORITY INTERRUPT IN THE CASE OF LONG INDIRECT CHAINS OR INDIRECT LOOPS.

SPT

THE SPT (SPECIAL/SHARED PAGES TABLE) HOLDS PHYSICAL ADDRESSES FOR PAGES WHICH ARE SHARED AMONG MANY PAGE TABLES (PROCESSES) OR WHICH ARE USED IN SOME SPECIAL WAY, E.G., AS PAGE TABLES. A PAGER REGISTER (AC BLOCK 6, WORD 3) HOLDS THE BASE ADDRESS OF THE SPT. THE SPT INDEX FOUND IN POINTERS IS ADDED TO THE SPT BASE ADDRESS TO FORM THE PHYSICAL CORE ADDRESS OF THE ASSOCIATED ENTRY. THE SPT ENTRY CONTAINS A PHYSICAL ADDRESS IN BITS 12-35. BITS 0-11 ARE IGNORED BY THE PAGER AND ARE USED AS A SHARE COUNT BY THE MONITOR.

CORE STATUS TABLE

IN ORDER TO DYNAMICALLY MANAGE CORE IN A VIRTUAL MEMORY ENVIRONMENT, IT IS EXTREMELY IMPORTANT FOR THE MONITOR TO BE ABLE TO OBTAIN INFORMATION ABOUT THE MEMORY REFERENCES GENERATED BY USER JOBS. THE CORE STATUS TABLE (CST) IS USED TO RECORD AND HOLD SUCH INFORMATION.

THE BASE ADDRESS OF THE CST IS HELD IN A PAGER REGISTER (AC BLOCK 6, WORD 2). THIS IS ADDED TO THE PHYSICAL CORE PAGE NUMBER FROM A STORAGE ADDRESS TO FORM THE ADDRESS OF THE ASSOCIATED CST ENTRY. A CST ENTRY IS USED AND UPDATED IN THE FOLLOWING MANNER:

1. FETCH THE CST ENTRY.
2. IF BITS 0-5 ARE 0, THE PAGE IS INACCESSIBLE AND A TRAP TO THE MONITOR IS INITIATED.
3. THE CST ENTRY IS "AND"ED WITH A MASK BEING HELD IN A PAGER REGISTER (CSTMSK, AC BLOCK 6, WORD 0).
4. THE RESULT IS IORED WITH THE QUANTITY IN A SECOND PAGER REGISTER (CSTDATA, AC BLOCK 6, WORD 1).

5. IF THE CURRENT VIRTUAL ADDRESS REFERENCE IS A WRITE REFERENCE (AND THE ACCESS BITS PERMIT A WRITE), A 1 IS IORED INTO BIT 35.

6. THE RESULT IS STORED BACK INTO THE CST.

THIS PROCEDURE ALLOWS THE MONITOR TO SET CERTAIN FIELDS AND MERGE OTHERS. TYPICALLY, ONE FIELD IS SELECTED AS AN "AGE", AND A VALUE REPRESENTING TIME IS SET INTO THIS FIELD. THUS PHYSICAL PAGES MAY BE ORDERED BY TIME OF LAST REFERENCE. ANOTHER FIELD IS TYPICALLY USED TO RECORD WHAT PROCESSES REFERENCE A PAGE. THIS IS DONE BY ASSIGNING A BIT POSITION TO EACH ACTIVE PROCESS AND PLACING A 1 IN THAT BIT POSITION IN THE PAGER DATA WORD. THUS THE CST WORD FOR A PAGE WILL HAVE ONES IN THE BIT POSITIONS OF EACH OF THE PROCESSES WHICH REFERENCED IT. ADDITIONALLY, THE MODIFIED BIT (B35) WILL BE SET IF THE PAGE HAS BEEN MODIFIED. THE MONITOR NEED NOT SWAP OUT PAGES TO WHICH ONLY READ REFERENCES HAVE BEEN DONE.

POINTER INTERPRETATION FLOW

THE FOLLOWING IS A DESCRIPTION OF THE POINTER INTERPRETATION ALGORITHM. IN THE FOLLOWING FLOW, THE TERM "USER" REFERS TO WHETHER AN EXEC OR USER SECTION IS BEING REFERENCED. THIS IS THE SAME AS THE PROCESS STATE BIT, EXCEPT UNDER PXCT (WHERE THE PROCESSOR CAN BE IN EXEC MODE WHILE REFERENCING A USER SECTION). THIS FLOW IS ALSO REPRESENTED BY A FLOW CHART IN THE FIGURES SECTION OF THIS DOCUMENT

1. INITIALIZE LOCAL PAGER VARIABLES P, W, AND C TO 1.

2. FETCH SECTION POINTER. THE SECTION POINTER IS FOUND IN THE USER SECTION TABLE IN THE UPT (LOCATION USECT THROUGH USECT+37) IF THE REFERENCE IS TO A USER SECTION, AND THE EXEC SECTION TABLE IN THE EPT (LOCATION ESECT THROUGH ESECT+37) IF THE REFERENCE IS TO AN EXEC SECTION. (USECT=ESECT=440)

3. TRAP IF A NO-ACCESS SECTION POINTER. A SECTION MAY BE NON-EXISTENT FOR THE RUNNING PROCESS, IN WHICH CASE THE SECTION POINTER WILL BE 0. BITS 0-2 OF THE SECTION POINTER ARE USED AS A CODE FIELD JUST AS WITH PAGE POINTERS. CODE 0 IS NO-ACCESS, CODE 1 IS IMMEDIATE, CODE 2 (SHARE) IS NURMAL SECTION POINTER, CODE 3 (INDIRECT) MAY ALSO BE USED, OTHER CODES ARE NOT DEFINED.

4. UPDATE ACCESS BITS, AND P, W, C (LOCAL PAGER VARIABLES) WITH SECTION POINTER BITS 3, 4, 6 RESPECTIVELY.

5. IF THE CODE IN BITS 0-2 IS 1, BITS 23-35 CONTAIN THE PAGE TABLE ADDRESS. OTHERWISE, FETCH THE PAGE TABLE

PHYSICAL ADDRESS. FETCH FROM THE PHYSICAL CORE ADDRESS GIVEN BY THE SUM OF THE SPT BASE ADDRESS AND THE SPT INDEX. THE SPT INDEX IS BITS 18-35 OF THE LAST POINTER FETCHED (A SECTION POINTER OR AN INDIRECT POINTER).

6. TRAP IF THE PAGE TABLE IS NOT IN CORE. IF THE PHYSICAL ADDRESS OBTAINED IN STEP 5 DOES NOT CONTAIN 0 IN BITS 12-17, THEN THE PHYSICAL ADDRESS IS NOT A CORE ADDRESS AND A TRAP IS INITIATED. IF THE SECTION POINTER WAS INDIRECT, FETCH A NEW SECTION POINTER FROM THIS PAGE TABLE, THE ENTRY GIVEN BY BITS 9-17 OF THE INDIRECT POINTER, AND GO TO 3.

7. CHECK AND UPDATE THE CST FOR THE PAGE TABLE. FETCH FROM THE PHYSICAL CORE ADDRESS GIVEN BY THE SUM OF THE CST BASE ADDRESS AND THE PHYSICAL CORE PAGE NUMBER. TRAP IF BITS 0-5 OF THE CST ENTRY ARE 0. OTHERWISE. AND CSTMSK, IOR CSTDATA, AND STORE THE RESULT BACK INTO CORE. THE MODIFIED BIT SHOULD NOT BE CHANGED HERE.

8. FETCH THE PAGE POINTER. THE PAGE POINTER IS FETCHED FROM THE PHYSICAL CORE ADDRESS CONSISTING OF THE CORE PAGE NUMBER OF THE PAGE TABLE IN BITS 14-26, AND THE CURRENT VIRTUAL PAGE NUMBER IN BITS 27-35. THE CURRENT PAGE NUMBER CAME FROM VMA BITS 18-26 OR FROM BITS 9-17 OF THE LAST INDIRECT POINTER.

9. UPDATE ACCESS BITS. THE P, W, AND C BITS ARE ANDED WITH THE RESPECTIVE PAGER VARIABLES.

10. DISPATCH ON POINTER TYPE.

IF A NO-ACCESS POINTER (CODE 0), INITIATE A TRAP.

IF AN INDIRECT POINTER, (CODE 3) TAKE BITS 9-17 AS THE NEW CURRENT PAGE NUMBER, AND TAKE BITS 18-35 AS AN SPT INDEX IDENTIFYING A NEW PAGE TABLE. LOOP BACK TO STEP 5.

IF A SHARE POINTER, FETCH THE SPT ENTRY GIVEN BY THE SUM OF THE SPT BASE REGISTER AND BITS 18-35 OF THE SHARE POINTER.

IF AN IMMEDIATE POINTER, CONTINUE TO STEP 11.

11. TRAP IF PAGE NOT IN CORE. IF BITS 12-17 OF THE PHYSICAL ADDRESS ARE NOT 0, THE ASSOCIATED PAGE IS NOT IN CORE AND A TRAP TO THE MONITOR IS INITIATED.

12. CHECK AND UPDATE THE CST FOR THE PAGE. FETCH FROM THE PHYSICAL CORE ADDRESS GIVEN BY THE SUM OF THE CST BASE ADDRESS AND THE PHYSICAL CORE PAGE NUMBER. TRAP IF BITS 0-5 OF THE CST ENTRY ARE 0. OTHERWISE, AND CSTMSK, IOR CSTDATA, AND IOR B35 IF A WRITE REFERENCE

AND W=1. STORE THE RESULT BACK INTO CORE.

13. COMPLETE THE VIRTUAL ADDRESS REFERENCE AND/OR LOAD PAGING MEMORY.

INITIATE ILLEGAL WRITE TRAP IF WRITE AND NOT-W.

THE PHYSICAL ADDRESS FOR THE REFERENCE CONSISTS OF THE PHYSICAL CORE PAGE NUMBER IN BITS 14-26, AND VMA BITS 27-35 IN BITS 27-35.

INSTRUCTIONS RELEVANT TO PAGING.

SEE CHAP 2.6 OF KL10 FUNCTIONAL SPECS, KL10 INTERNAL IO INSTRUCTIONS, FOR ADDITIONAL INFORMATION.

1. LOAD UBR (DATAO PAG,) LOAD EBR (CONO PAG,).

2. CLEAR PAGING MEMORY (DATAO PAG, CONO PAG,).

3. CLEAR PAGING MEMORY ENTRY FOR MONITOR VIRTUAL ADDRESS E (CLRPT E).

4. PAGING ON/OFF (CONO PAG,).

5. DECLARE SPT BASE ADDRESS; CST BASE ADDRESS; CSTMSK; CSTDATA. (DONE AS DEPOSITS INTO RESERVED AC BLOCK.)

PAGE FAIL DATA

THE FOLLOWING DATA IS STORED WHEN A PAGE FAIL TRAP IS INITIATED:

1. VIRTUAL ADDRESS OF REFERENCE

2. USER BIT, PUBLIC BIT, WRITE REFERENCE BIT

3. PAGE FAIL CODE

SPECIFIC PAGE FAIL CODES ARE GENERATED AND STORED IN THE PAGE FAIL WORD FOR THE FOLLOWING CONDITIONS:

1. PROPRIETARY VIOLATION

2. REFILL ERROR

3. ADDRESS COMPARE

4. PAGE TABLE PARITY ERROR

5. MEMORY DATA PARITY ERROR

ALL OTHER PAGE FAIL CONDITIONS STORE ONLY THE DATA DESCRIBING THE REFERENCE (ADDRESS, USER, PUBLIC, WRITE) AND THE SOFTWARE WILL DETERMINING THE CAUSE OF THE PAGE FAIL AND THE PROPER ACTION.

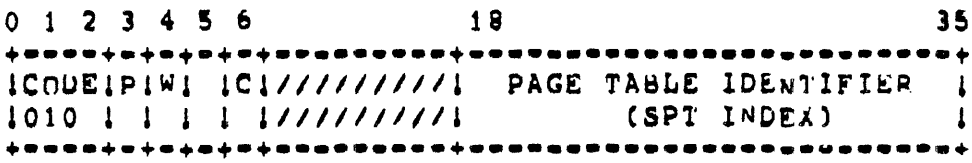
KL10 PAGING - WORD FORMATS

SECTION POINTER

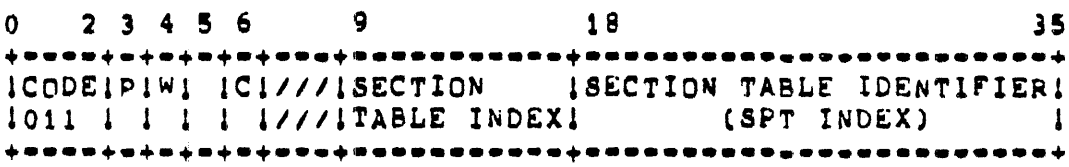
THE SECTION POINTER IS FOUND IN THE USER OR EXEC SECTION TABLE. (PART OF UPT OR EPT.)

SECTION POINTER PROVIDES (VIA THE SPT) THE PHYSICAL ADDRESS OF THE PAGE TABLE FOR THE GIVEN SECTION.

CODE:		
	0	NO-ACCESS (TRAP)
	1	IMMEDIATE
	2	SHARE
	3	INDIRECT
	4-7	UNUSED, RESERVED



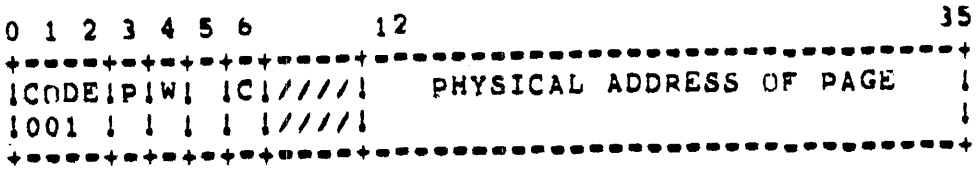
NORMAL SECTION POINTER (CODE = 2)



INDIRECT SECTION POINTER (CODE = 3)

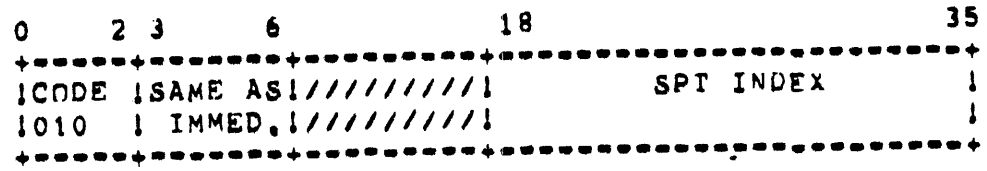
PAGE POINTERS

FOUND IN PAGE TABLES



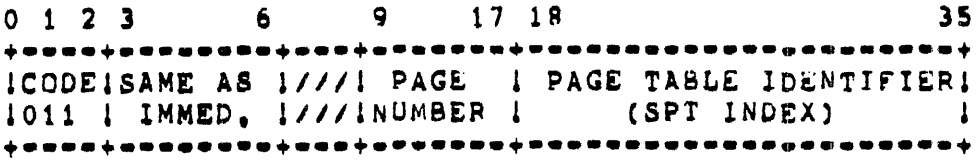
IMMEDIATE POINTER (CODE FIELD = 1)

B12-35 GIVE PHYSICAL ADDRESS OF PAGE
IF B12-17 >< 0, PAGE NOT IN CORE-TRAP
IF B12-17 = 0, B23-35 GIVE CORE PAGE NUMBER OF PAGE, B18-22 MBZ



SHARED POINTER (CODE FIELD = 2)

B18-35 GIVE SPT INDEX (SPTX). SPTX + SPT BASE ADDRESS = PHYSICAL CORE ADDRESS OF WORD HOLDING PHYSICAL ADDRESS OF PAGE.

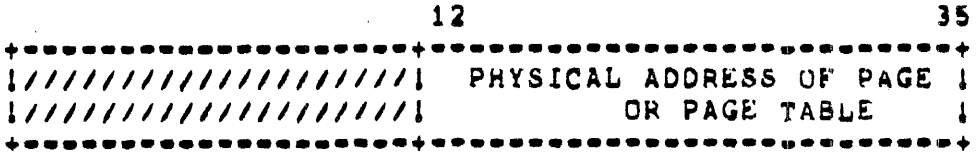


INDIRECT POINTER (CODE FIELD = 3)

THIS POINTER TYPE CAUSES ANOTHER POINTER TO BE FETCHED AND INTERPRETED. THE NEW POINTER IS FOUND IN WORD N (B9-17) OF THE PAGE ADDRESSED BY C(SPT + SPTX).

SPT ENTRY

FOUND IN THE SPT, I.E., WHEN FETCHING C(SPT +SPTX)

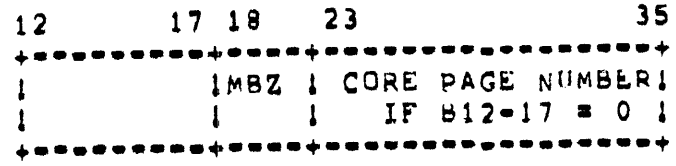


B12-35 GIVE PHYSICAL ADDRESS OF PAGE.

THE BASE ADDRESS (PHYSICAL CORE ADDRESS) OF THE SPT RESIDES IN ONE AC OF THE RESERVED AC BLOCK.

PHYSICAL STORAGE ADDRESS

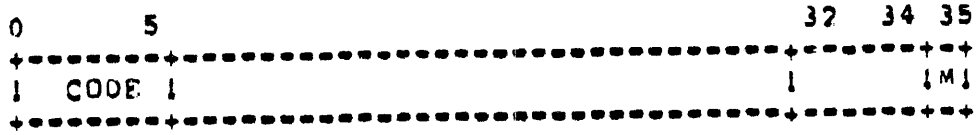
FOUND IN B12-35 OF IMMEDIATE POINTERS AND SPT ENTRIES.



IF B12-17 = 0, THEN B23-35 ARE CORE PAGE NUMBER (I.E., B14-26 OF PHYSICAL CORE ADDRESS) OF PAGE AND B18-22 MBZ. IF B12-17 >< 0, THEN ADDRESS IS NOT CORE AND PAGER TRAPs.

CORE STATUS TABLE ENTRY

FOUND WHEN FETCHING C(CBR + CORE PAGENO)



B0-5 ARE CODE FIELD:

0 = UNAVAILABLE, TRAP

1-77 = AVAILABLE

B32-34 RESERVED FOR FUTURE HARDWARE SPECIFICATION.

B35 IS "MODIFIED" BIT, SET ON ANY WRITE REF TO PAGE.

WHENEVER A CORE PAGE NUMBER (PHYS. ADR WITH B12-17=0) IS FOUND DURING POINTER INTERPRETATION, THE CST WORD AT CBR + CORE PAGENO IS UPDATED AS FOLLOWS:

1. FETCH C(CBR+CORE PAGENO)
2. TRAP IF B0-5 =0
3. AND WITH MASK FROM RESERVED AC BLOCK
4. IOR WITH DATA FROM RESERVED AC BLOCK
5. IOR B35 IF WRITE REFERENCE

QUANTITIES IN HARDWARE REGISTERS (RESERVED AC BLOCK)

SPT SPT BASE REGISTER

CBR CST BASE REGISTER

CSTMSK CST UPDATE MASK

ANDED WITH CST WORD DURING UPDATE

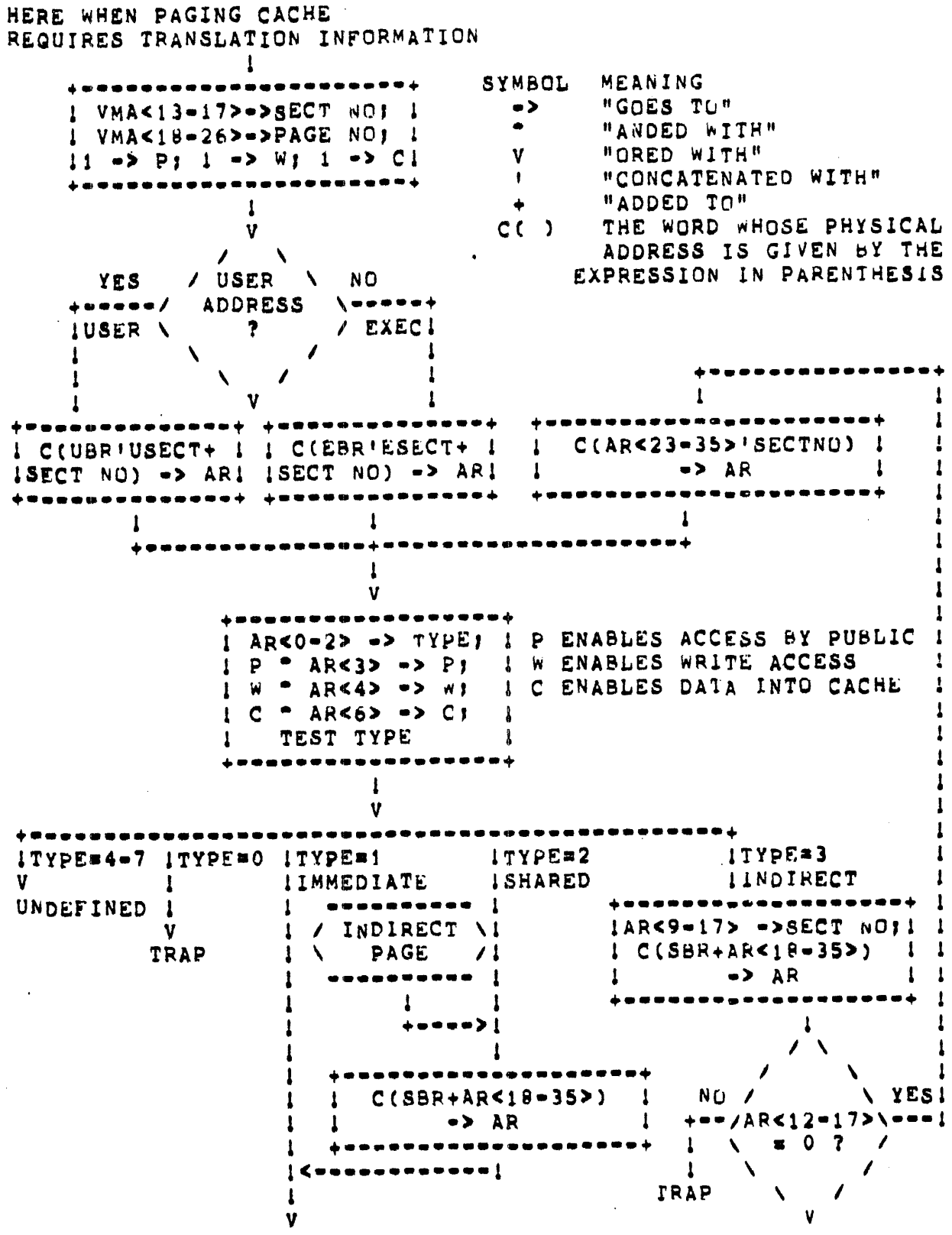
(B32-35 MUST BE ALL 1'S TO PRESERVE EXISTING CST INFORMATION)

CSTDATA CST UPDATE DATA

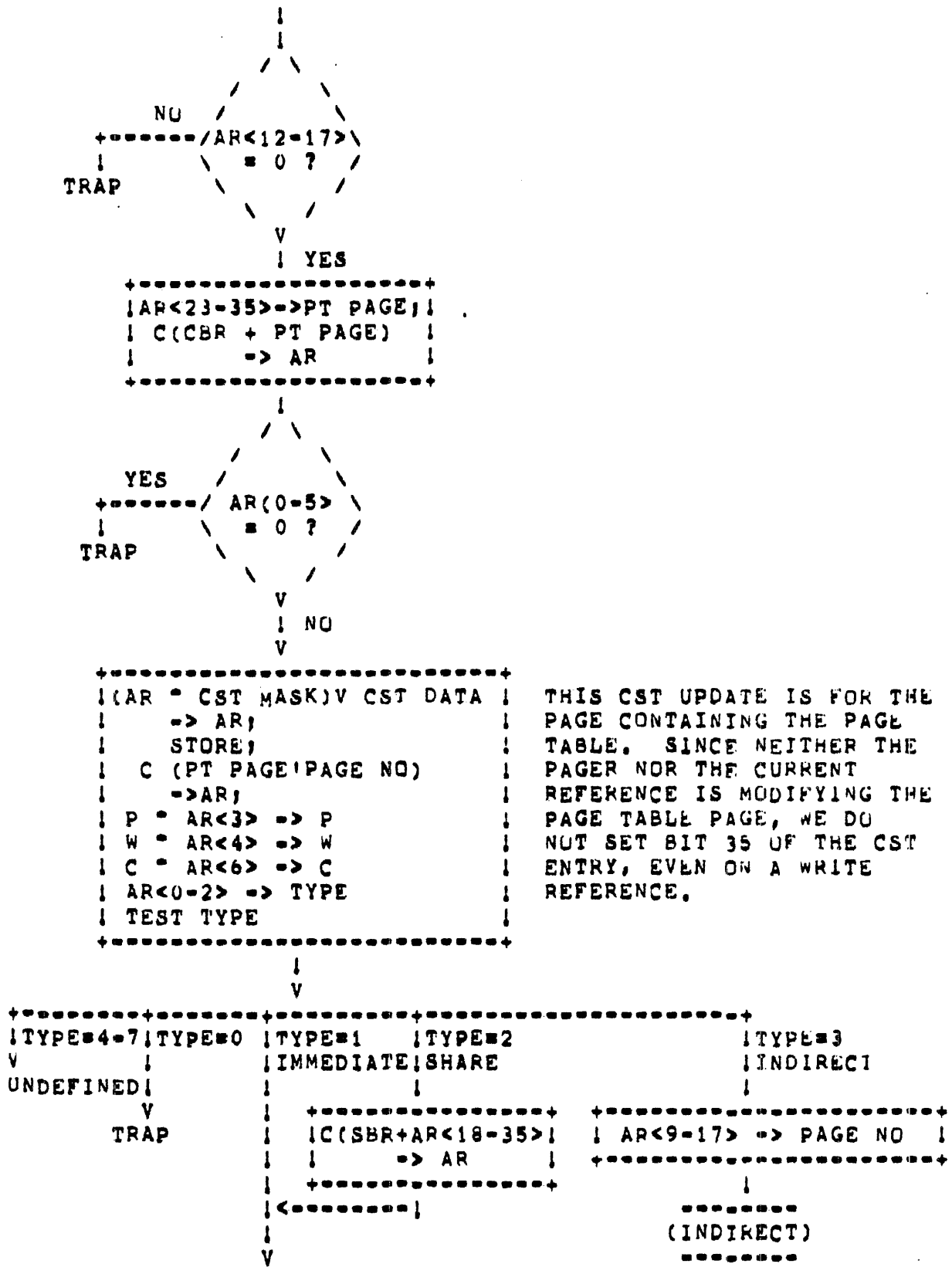
IORED WITH CST WORD DURING UPDATE

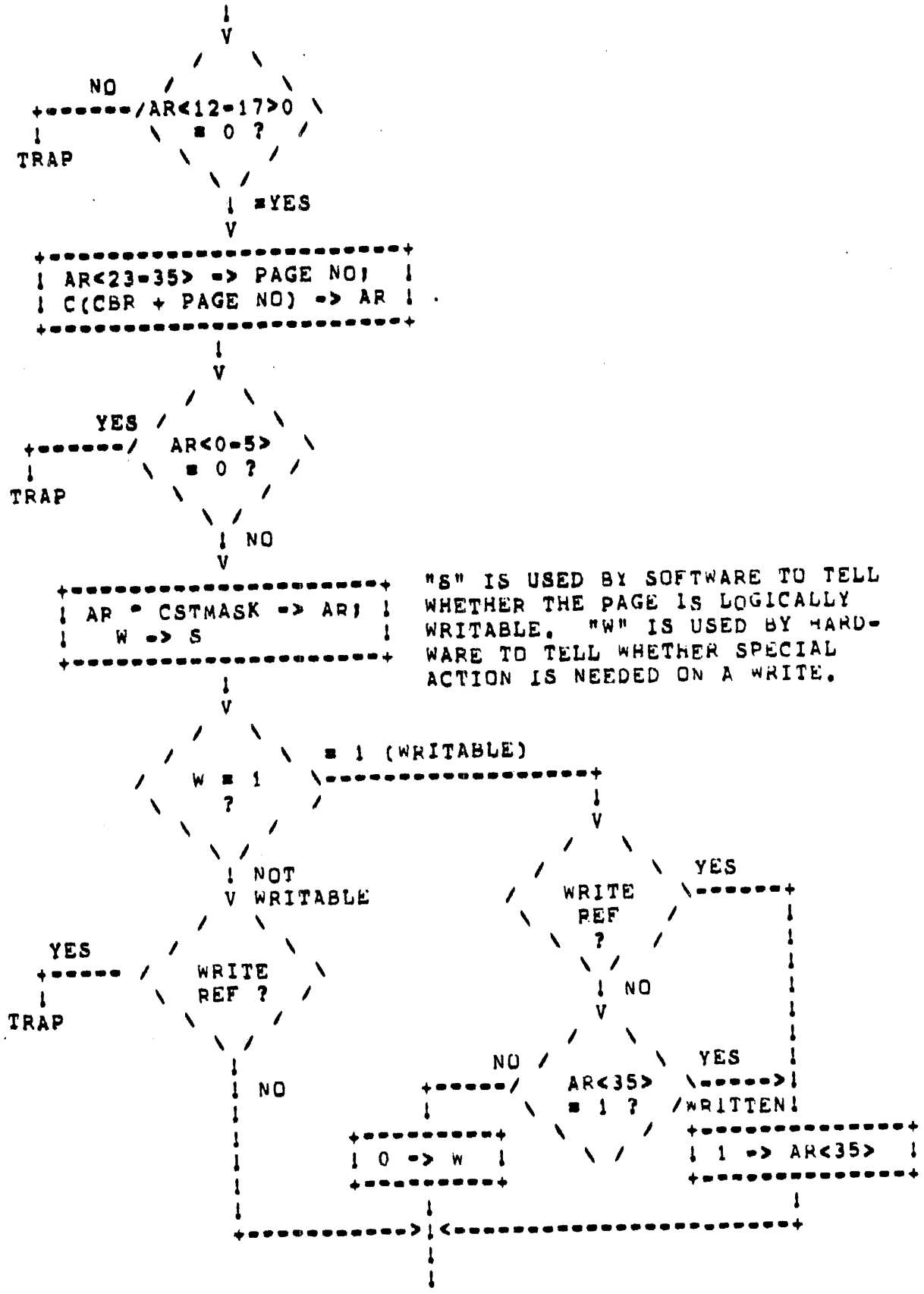
(B32-35 MUST BE ALL 0'S TO PRESERVE EXISTING CST INFORMATION)

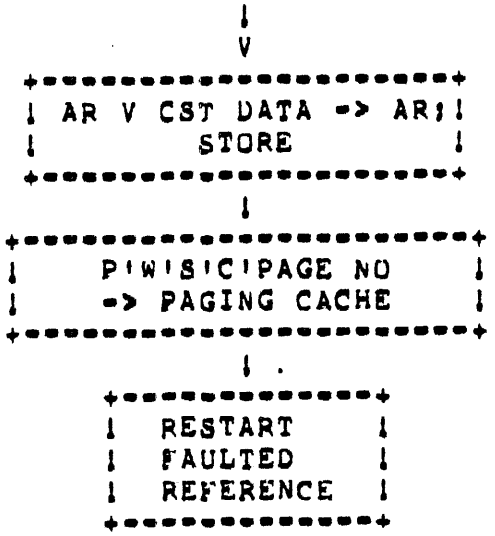
ALL UNSPECIFIED BITS AND FIELDS ARE RESERVED FOR FUTURE SPECIFICATION BY DEC.



SYMBOL	MEANING
->	"GOES TO"
*	"ANDED WITH"
V	"ORED WITH"
!	"CONCATENATED WITH"
+	"ADDED TO"
C( )	THE WORD WHOSE PHYSICAL ADDRESS IS GIVEN BY THE EXPRESSION IN PARENTHESIS


NO
AR<12-17> = 0 ?
TRAP
YES
AR<23-35>=>PT PAGE
C(CBR + PT PAGE) => AR
YES
AR(0-5> = 0 ?
TRAP
NO
(AR * CST MASK)V CST DATA => AR
STORE
C (PT PAGE!PAGE NO) =>AR
P * AR<3> => P
W * AR<4> => W
C * AR<6> => C
AR<0-2> => TYPE
TEST TYPE
THIS CST UPDATE IS FOR THE PAGE CONTAINING THE PAGE TABLE. SINCE NEITHER THE PAGER NOR THE CURRENT REFERENCE IS MODIFYING THE PAGE TABLE PAGE, WE DO NOT SET BIT 35 OF THE CST ENTRY, EVEN ON A WRITE REFERENCE.
TYPE=4-7
TYPE=0
UNDEFINED
TRAP
TYPE=1
IMMEDIATE
TYPE=2
SHARE
C(SBR+AR<18-35> => AR
TYPE=3
INDIRECT
AR<9-17> => PAGE NO
(INDIRECT)


NO
AR<12-17>0 = 0 ?
TRAP
=YES
AR<23-35> => PAGE NO
C(CBR + PAGE NO) => AR
YES
AR<0-5> = 0 ?
TRAP
NO
AR * CSTMASK => AR
W => S
"S" IS USED BY SOFTWARE TO TELL WHETHER THE PAGE IS LOGICALLY WRITABLE. "W" IS USED BY HARDWARE TO TELL WHETHER SPECIAL ACTION IS NEEDED ON A WRITE.
W = 1 ?
= 1 (WRITABLE)
NOT WRITABLE
YES
WRITE REF ?
TRAP
NO
WRITE REF ?
YES
NO
AR<35> = 1 ?
YES
WRITTEN
0 => W
1 => AR<35>


AR V CST DATA -> AR
STORE
P
W
S
C
PAGE NO
-> PAGING CACHE
RESTART
FAULTED
REFERENCE

APPENDIX I - EPT/UPT SYMBOLS USED

ESECT = 440 SECTION TABLE FOR EXEC SECTIONS 0-37

USECT = 440 SECTION TABLE FOR USER SECTIONS 0-37

APPENDIX II

THE FOLLOWING DISCUSSES SOME OF THE DESIGN AND IMPLEMENTATION ISSUES OF THE PAGER.

CACHED PAGING DATA

THE KL10 HAS A "PAGING MEMORY" WHICH HOLDS VIRTUAL-TO-PHYSICAL MAPPING INFORMATION. THIS IS EFFECTIVELY A CACHE OF PAGING DATA WHICH HAS BEEN FETCHED FROM MEMORY AND/OR DETERMINED BY POINTER INTERPRETATION. NOTE, HOWEVER, THAT IT IS TOTALLY DISTINCT FROM THE MEMORY CACHE, USUALLY REFERRED TO AS SIMPLY "THE CACHE". WHEN A VIRTUAL ADDRESS REFERENCE IS REQUESTED, THE PAGING MEMORY IS FIRST CHECKED TO SEE IF THE CORRESPONDING PHYSICAL ADDRESS IS PRESENT. IF IT IS, THE REFERENCE CAN PROCEED WITH NO DELAY. IF IT IS NOT, THE DATA MUST BE OBTAINED FROM CORE.

THE KL10 PAGING MEMORY IS IMPLEMENTED AS A TABLE WITH ONE ENTRY FOR EACH OF THE 512 PAGES OF THE VIRTUAL ADDRESS SPACE. (THE KL10 IMPLEMENTED THE EQUIVALENT FUNCTION WITH ASSOCIATIVE MEMORY.) THE USER AND EXEC ADDRESS SPACES USE THE SAME 512 ENTRIES, BUT THE INDEX IS OFFSET DIFFERENTLY SO AS TO REDUCE CONFLICTS. AT ANY TIME THEN, THE PAGING MEMORY WILL BE HOLDING MAPPING INFORMATION FOR MOST OF THE PAGES ACTIVELY BEING USED BY THE RUNNING PROGRAM. WHEN THE MONITOR TAKES ANY ACTION WHICH WOULD INVALIDATE SOME EXISTING VIRTUAL-TO-PHYSICAL ADDRESS ASSOCIATIONS, THE PAGING MEMORY MUST BE PARTIALLY OR COMPLETELY CLEARED. SUCH CASES INCLUDE:

1. CHANGE OF USER PROCESS - THE ENTIRE USER ADDRESS SPACE CHANGES, SO THE ENTIRE PAGING MEMORY MUST BE CLEARED.

2. REMOVAL OF ONE PAGE FROM CORE OR REMOVAL OF A POINTER FROM THE USER PROCESS PAGE TABLE - THE ENTIRE PAGING MEMORY MUST BE CLEARED SINCE SHARED AND INDIRECT POINTERS MAY HAVE CAUSED THE ONE PHYSICAL PAGE TO APPEAR IN SEVERAL VIRTUAL PAGES.

3. IN SOME CASES, THE MONITOR WILL MAP A PAGE INTO THE EXEC MAP FOR LOCAL USE. WHEN THIS PAGE IS UNMAPPED, ONLY THAT ONE ASSOCIATION NEED BE CLEARED FROM THE PAGING MEMORY. THE PAGER PROVIDES A FUNCTION TO CLEAR THE ASSOCIATION FOR A PARTICULAR VIRTUAL ADDRESS. THIS MAY BE USED IN THIS CASE TO REDUCE SUBSEQUENT RELOAD

OVERHEAD.

IN KL10 PAGING MODE, WHEN THE PAGING MEMORY FAILS TO CONTAIN THE DATA FOR THE REQUESTED VIRTUAL ADDRESS, A SPECIAL TRAP IN THE MICROCODE OCCURS. AFTER SAVING VULNERABLE ACTIVE EBOX DATA, THE MICROCODE INVOKES THE POINTER TRACING AND INTERPRETATION ALGORITHM DESCRIBED ABOVE. IF THE POINTER INTERPRETATION YIELDS A VALID CORE ADDRESS, IT AND THE ACCESS INFORMATION ARE LOADED INTO THE PAGING MEMORY, THEN THE EBOX ACTIVE REGISTERS ARE RESTORED AND THE ORIGINAL MEMORY REFERENCE IS REQUESTED AGAIN.

THE PAGER IS REQUIRED TO MAINTAIN THE MODIFIED BIT IN THE CORE STATUS TABLE. THIS MEANS THAT THE FIRST WRITE REFERENCE TO A PAGE MUST BE DETECTED EVEN IF PREVIOUS READ REFERENCES HAVE BEEN MADE. THE MICROCODE IMPLEMENTS THIS AS FOLLOWS:

> ON A PAGING MEMORY RELOAD, THE WRITE ACCESS (W) BIT IS SET IN THE PAGING MEMORY ONLY IF THE CURRENT MEMORY REFERENCE IS A WRITE (AND WRITE IS LEGAL FOR THE PAGE). THUS IF THE FIRST REFERENCE TO A PAGE IS READ, THE W BIT IN THE CORRESPONDING PAGING MEMORY ENTRY WILL BE SET TO 0, AND A SBSEQUENT WRITE REFERENCE WILL CAUSE ANOTHER TRAP TO THE MICROCODE. ON THIS SECOND TRAP, THE POINTER INTERPRETATION WILL BE REPEATED AND THE PAGING MEMORY RELOADED, THIS TIME WITH THE W BIT SET.

IT HAS BEEN SUGGESTED THAT THE MICROCODE COULD USE THE UNUSED BIT (FORMERLY THE S BIT) IN THE PAGING MEMORY TO RECORD WHETHER OR NOT THE PAGE IS WRITABLE AND THEREBY AVOID THE SECOND POINTER TRACE IN THE WRITE-AFTER-READ CASE. THE PERFORMANCE IMPROVEMENT APPEARS TO BE VERY SMALL HOWEVER AND AT PRESENT DOES NOT SEEM WORTH THE ADDITIONAL MICROCODE WHICH WOULD BE REQUIRED.

CACHE CONSIDERATIONS

IN A ONE-CPU CONFIGURATION, IT DOES NOT MATTER LOGICALLY WHETHER MICROCODE PAGING REFERENCES ARE CACHED. SINCE THE PAGING MEMORY IS AVAILABLE TO HOLD PAGING DATA, IT WOULD SEEM REDUNDANT TO ALSO HOLD PAGE POINTERS IN THE CACHE. ALSO, IT APPEARS UNLIKELY THAT REFERENCES ARE MADE TO PAGE POINTERS VERY OFTEN. ON THE OTHER HAND, THE SECTION 0 SECTION POINTERS (EXEC AND USER) AND THE ASSOCIATED SPT WORDS ARE REFERENCED ON EVERY PAGE MEMORY RELOAD. HENCE, IT IS UNCLEAR WHETHER A PERFORMANCE ADVANTAGE IS OBTAINED BY CACHING OR NOT CACHING PAGE REFILL REFERENCES.

MULTI-CPU CONSIDERATIONS

IN A MULTI-CPU CONFIGURATION USING KL10S, EACH PROCESSOR WOULD HAVE ITS OWN PAGING MEMORY, AND IN GENERAL, ALL OF THEM MUST BE

CLEARED ON ANY EVENT WHICH REQUIRES CLEARING ONE. THEREFORE THE MONITOR MUST HAVE A SIGNAL MECHANISM TO QUICKLY REQUEST THE OTHER PROCESSORS TO EXECUTE THE APPROPRIATE PAGER INSTRUCTIONS.

THE CODE FIELD OF THE CORE STATUS TABLE MAY BE USED TO PREVENT ACCESS TO CERTAIN PAGES BY OTHER PROCESSORS WHEN ONE PROCESSOR BEGINS SOME HOUSEKEEPING FUNCTION ON THOSE PAGES (E.G., SWAPPING OUT TO DISK). IT DOES NOT, HOWEVER, PROVIDE ANY MECHANISM TO LIMIT ACCESS TO EXACTLY ONE PROCESSOR.

IN A MULTI-CPU CONFIGURATION, THE SEVERAL PROCESSORS WILL BE UPDATING THE CORE STATUS TABLE SIMULTANEOUSLY, AND SOME MECHANISM (E.G., READ-PAUSE-WRITE) MUST BE USED TO ENSURE THAT DATA IS NOT LOST BECAUSE OF TWO PROCESSORS SIMULTANEOUSLY UPDATING THE SAME ENTRY. OBVIOUSLY, MICROCODE CST REFERENCES CANNOT BE CACHED, AND THE MONITOR MUST ENSURE THAT PROGRAM REFERENCES TO THE CST ARE ALSO NOT CACHED.

IT IS DESIRABLE THAT ALL MICROCODE PAGING REFERENCES BE UNCACHED IN A MULTI-CPU CONFIGURATION, OTHERWISE THE SOFTWARE MUST DO A CACHE CLEAR ALSO EVERY TIME IT DOES A PAGING MEMORY CLEAR OF ANOTHER PROCESSOR.

DIVISION OF EXEC ADDRESS SPACE

TYPICALLY THE MONITOR USES AREAS OF THE EXEC ADDRESS SPACE IN DISTINCT WAYS:

1. SYSTEM-WIDE CODE AND DATA. E.G., THE MONITOR CODE, SCHEDULER AND SWAPPER DATA BASES, ETC.

2. PER-JOB DATA. E.G., OPEN FILE DATA, FORK STRUCTURE DATA, ETC.

3. PER-PROCESS DATA. E.G., LOCAL STACK, LOCAL VARIABLES, ETC.

IN ADDITION TO PERMANENT CONTENTS, EACH OF THESE AREAS WILL ALSO HAVE FILE OR PROCESS PAGES DYNAMICALLY MAPPED IN AS NEEDED.

THE INDIRECT POINTER MECHANISM MAY BE USED TO CONFIGURE THE EXEC ADDRESS SPACE IN THIS MANNER, AND IT DOES NOT REQUIRE THAT ANY BOUNDARIES BE WIRED INTO HARDWARE (AS ON THE KI10, EXEC PAGES 340-377). TO IMPLEMENT THE DIVISION ABOVE, THE MONITOR WOULD HAVE THREE PARTIAL PAGE TABLES:

1. THE SYSTEM MONITOR MAP RESIDING IN RESIDENT STORAGE;

2. THE JOB MAP RESIDING IN A PAGE SWAPPED WITH THE JOB;

3. THE PROCESS MAP RESIDING IN A PAGE SWAPPED WITH THE PROCESS.

THE THREE MAPS WOULD COVER MUTUALLY EXCLUSIVE AREAS OF THE MONITOR ADDRESS SPACE. EACH WOULD HAVE AN SPT SLOT ASSIGNED TO HOLD ITS PHYSICAL CORE ADDRESS.

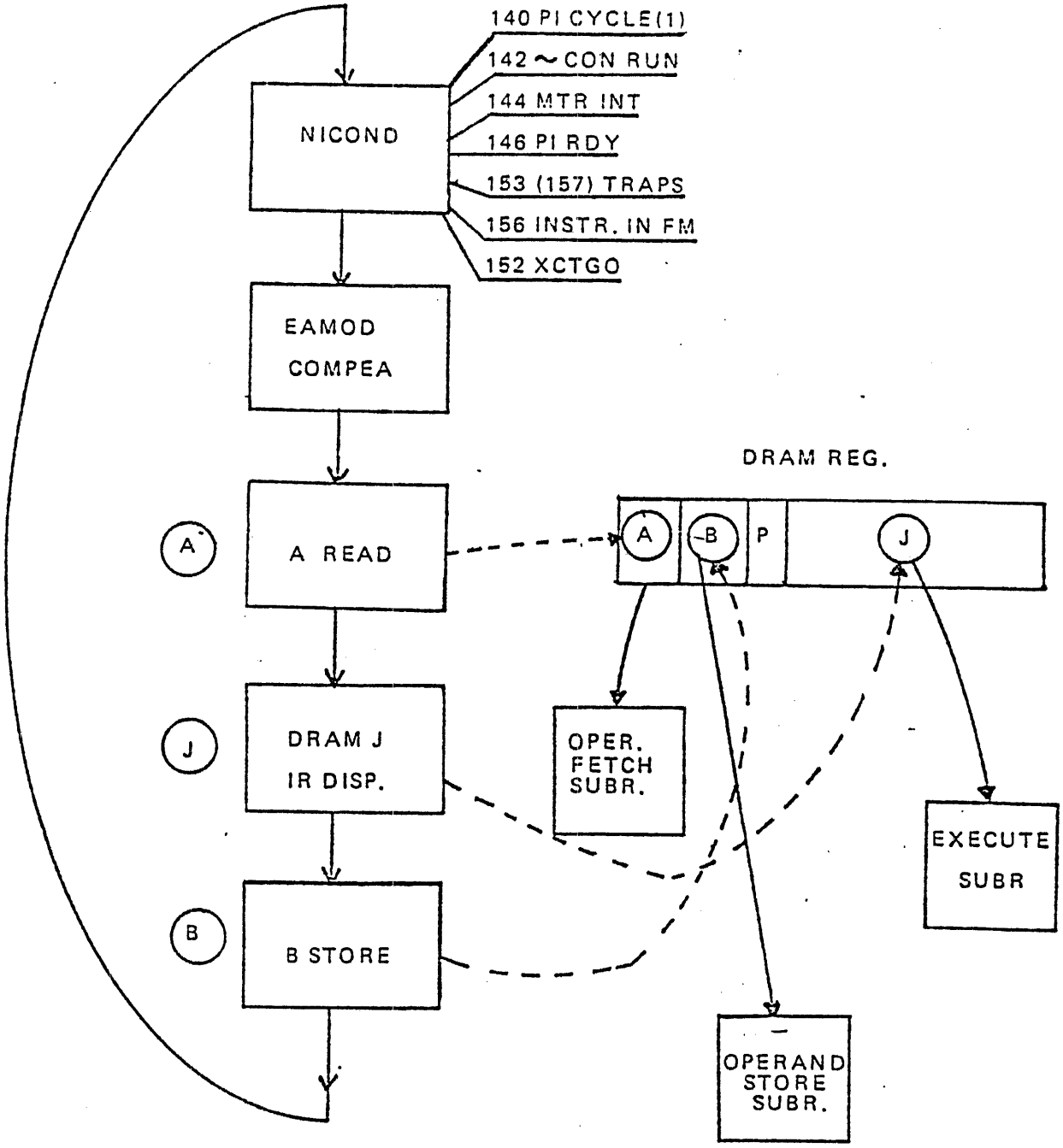
ALL POINTER TRACES WOULD FIRST REFERENCE THE SYSTEM MONITOR MAP AFTER HAVING INTERPRETED THE EXEC SECTION 0 POINTER. FOR SYSTEM-WIDE PAGES, THE MAP WOULD CONTAIN A NORMAL PAGE POINTER TO THE APROPRIATE PAGE. FOR PER-JOB PAGES, THE MAP WOULD CONTAIN AN INDIRECT POINTER TO THE JOB MAP, AND FOR PER-PROCESS PAGES, THE MAP WOULD CONTAIN AN INDIRECT POINTER TO THE PROCESS MAP. IN ORDER TO AVOID CHANGING ALL OF THE PER-JOB AND PER-PROCESS POINTERS ON A CONTEXT SWITCH, TWO SPT ENTRIES WOULD BE RESERVED FOR THE "CURRENT" JOB MAP AND PROCESS MAP ADDRESSES, AND THE JOB AND PROCESS POINTERS WOULD ALWAYS USE THESE RESERVED ENTRIES. THAT IS, WHEN ANY PROCESS IS STARTED, ITS JOB MAP CORE ADDRESS IS COPIED INTO THE RESERVED JOB SPT ENTRY, AND ITS PROCESS MAP CORE ADDRESS IS COPIED INTO THE RESERVED PROCESS SPT ENTRY. THESE SPT ENTRIES BECOME IN EFFECT TWO BASE REGISTERS FOR THE TWO MAPS, AND THE MONITOR MUST SET THEM JUST AS THOUGH THEY WERE IMPLEMENTED IN HARDWARE.

[END OF KLPAG.SPC]

THE GUIDE TO UNDERSTANDING MICRODE FLOWS IN KL BASED SYSTEMS

Compiled by George Allen

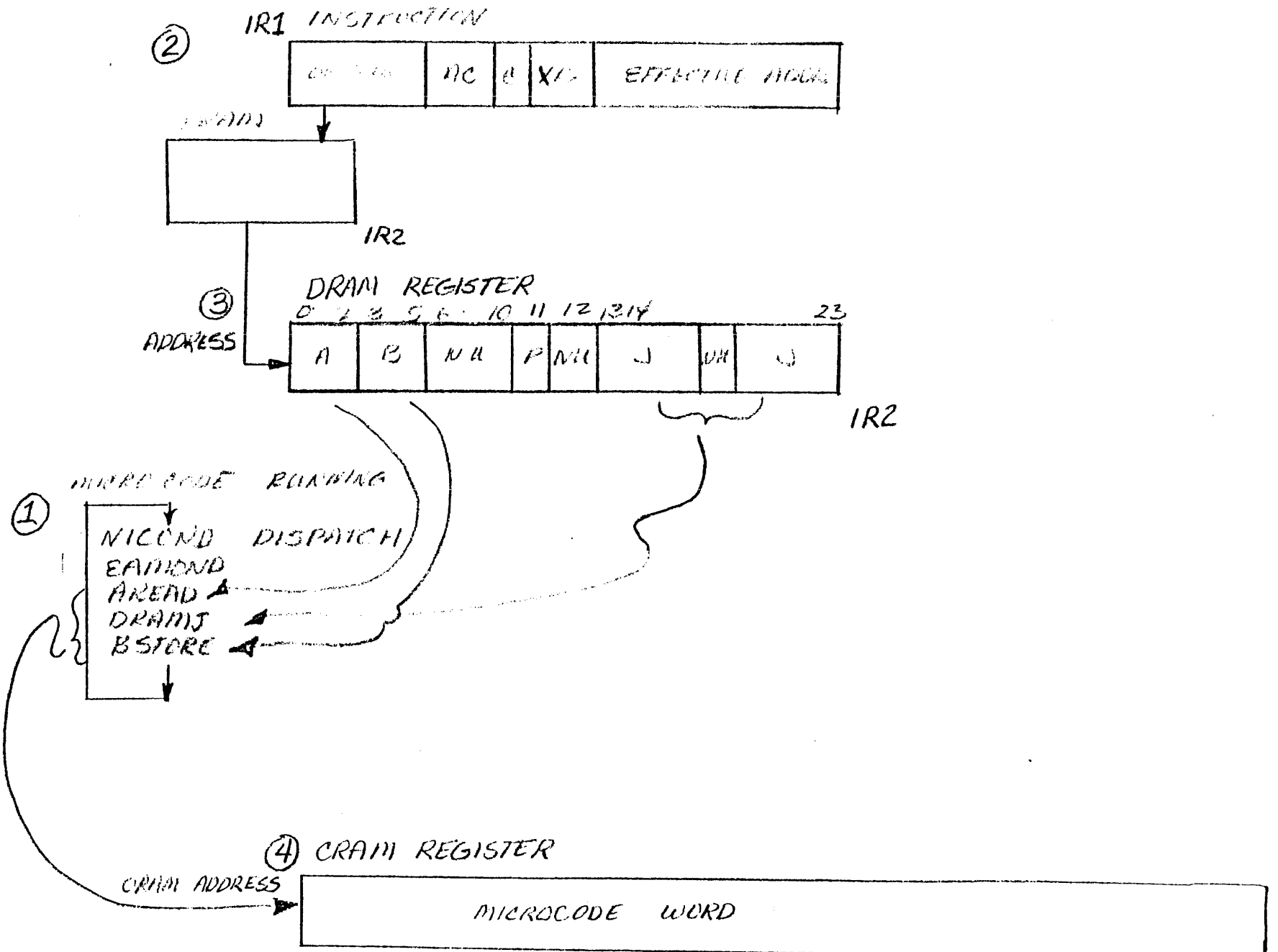
Tymshare Educational Servic

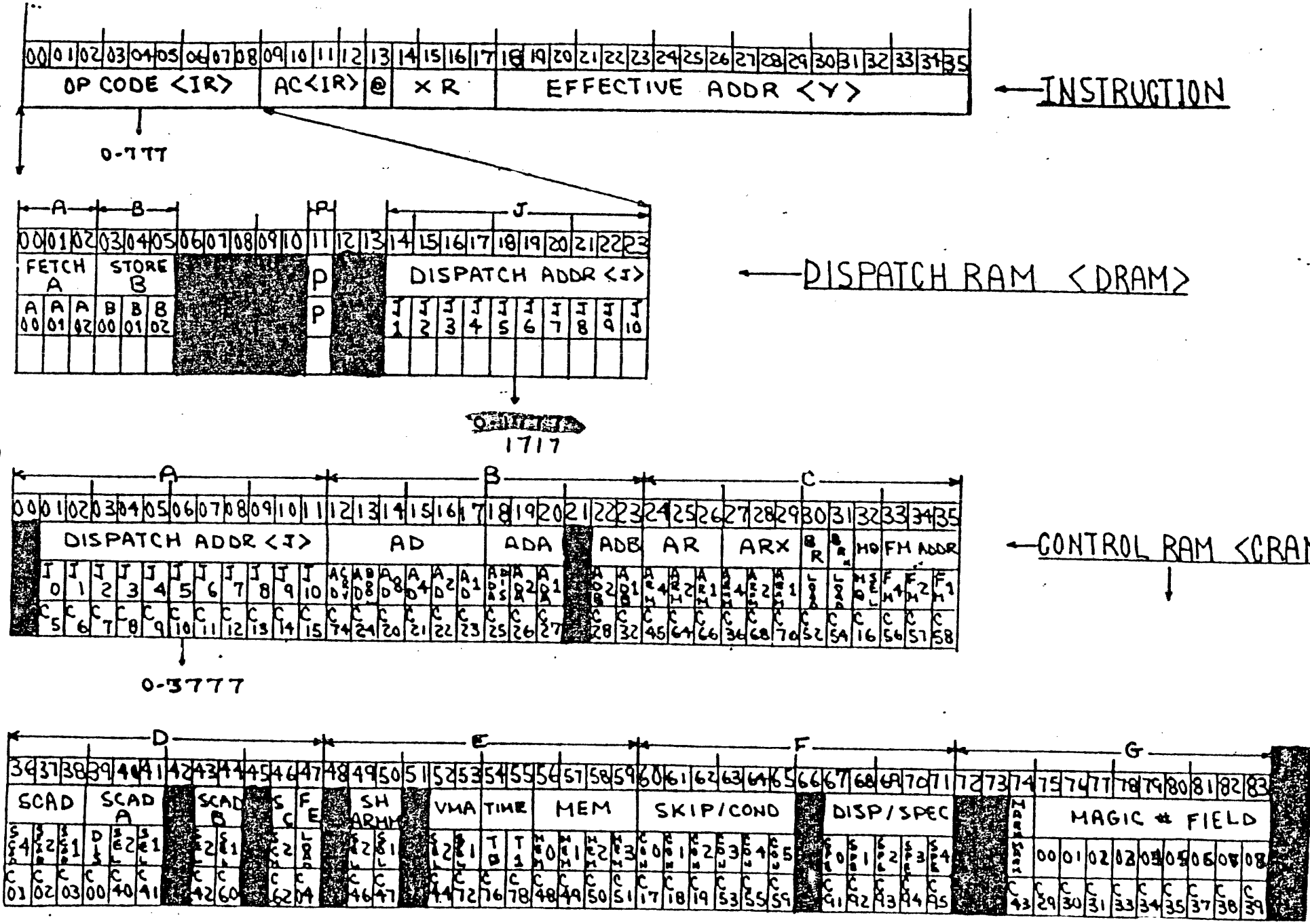


BASIC MACHINE CYCLE

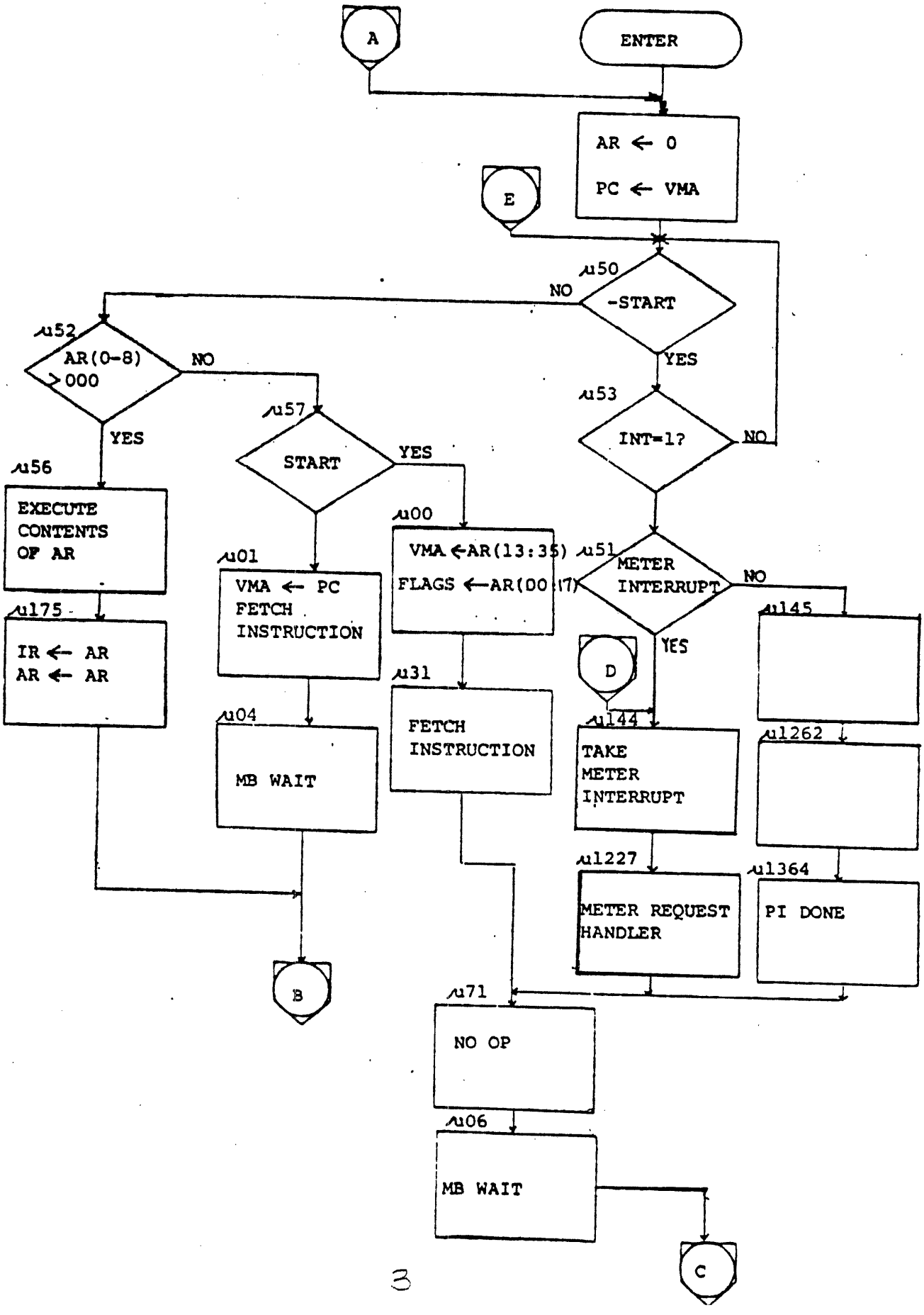
FIGURE 2-3A

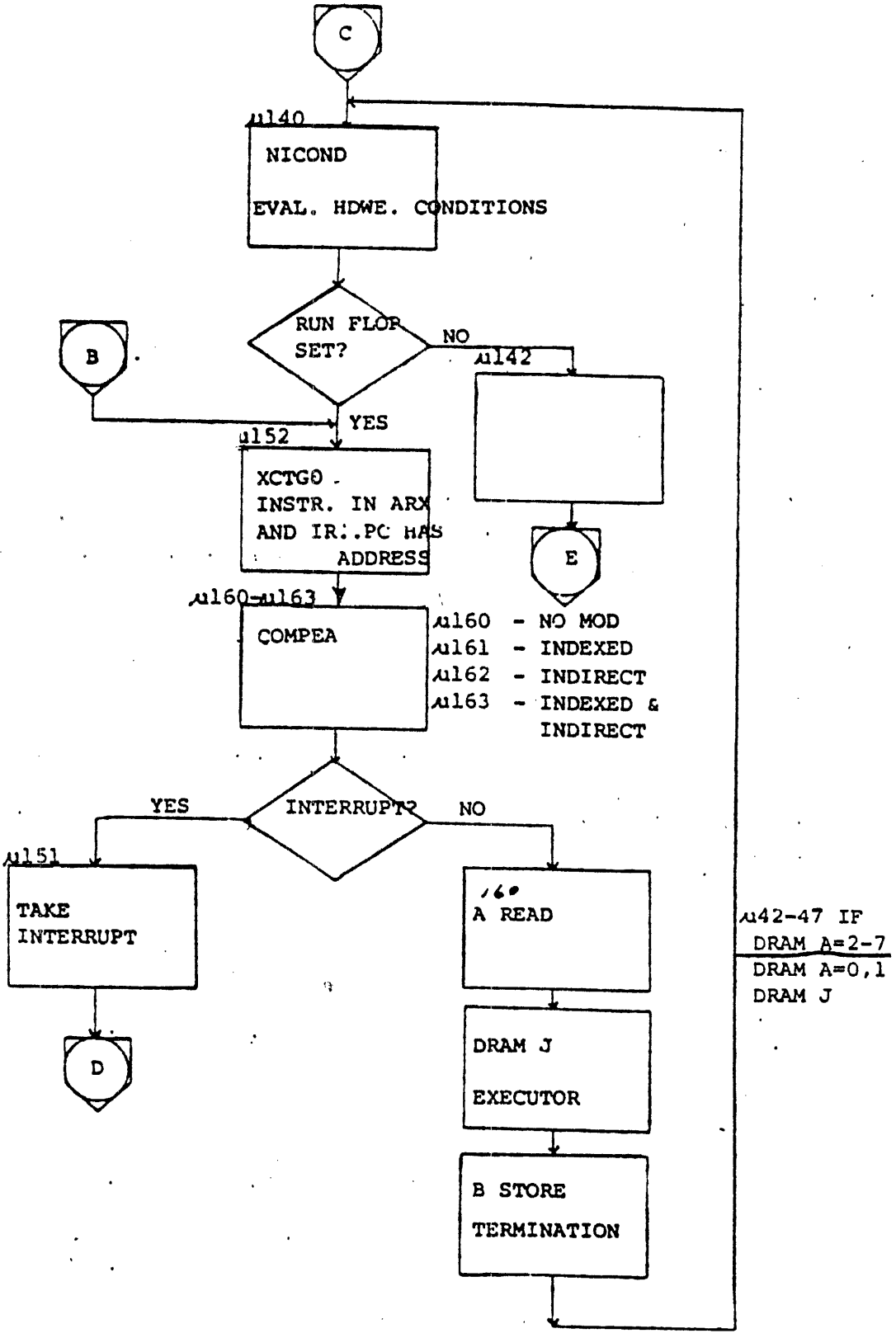
INSTRUCTION EXCUTION




00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
OP CODE <IR>
AC <IR>
@
X R
EFFECTIVE ADDR <Y>
INSTRUCTION
0-777
A
B
P
J
FETCH A
STORE B
DISPATCH ADDR <J>
DISPATCH RAM <DRAM>
1717
DISPATCH ADDR <J>
AD
ADA
ADB
AR
ARX
FM ADDR
CONTROL RAM <CRAM>
0-3777
D
E
F
G
SCAD
SCAD A
SCAD B
SC
FE
SH
VMA
TIME
MEM
SKIP/COND
DISP/SPEC
MAGIC # FIELD

MICROPROGRAM FLOWCHART REVISION 126
(GENERALIZED)

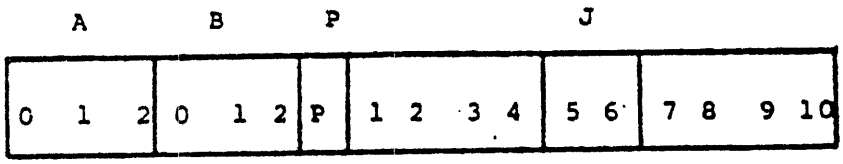



C
μ140
NICOND
EVAL. HDWE. CONDITIONS
RUN FLOP SET?
NO
B
μ142
YES
μ152
XCTGO - INSTR. IN ARX AND IR:.PC HAS ADDRESS
E
μ160-μ163
COMPEA
μ160 - NO MOD
μ161 - INDEXED
μ162 - INDIRECT
μ163 - INDEXED & INDIRECT
YES
INTERRUPT?
NO
μ151
TAKE INTERRUPT
D
A READ
DRAM J
EXECUTOR
B STORE
TERMINATION
μ42-47 IF
DRAM A=2-7
DRAM A=0,1
DRAM J

DISPATCH RAM (DRAM) WORD FORMAT

DRAM word found on M8522 module in Slot 45.

48



Bits 5 & 6 always zero

DRAM A Bit Functions

0	Immediate
1	Immediate - PF
2	Not Used
3	Write Test
4	Read
5	Read - PF
6	Read - Write
7	Read - Pause - Write

DRAM B Bit Functions

B Store 0, 1, 2

1	Double AC
2	Double both
3	Self
5	AC
6	Memory
7	Both

B - 1 - 2 Flt Store

1	AC
2	Memory
3	Both

B0 Inverts Tests

0	Cry 0 = 0
1	Cry 0 = 1

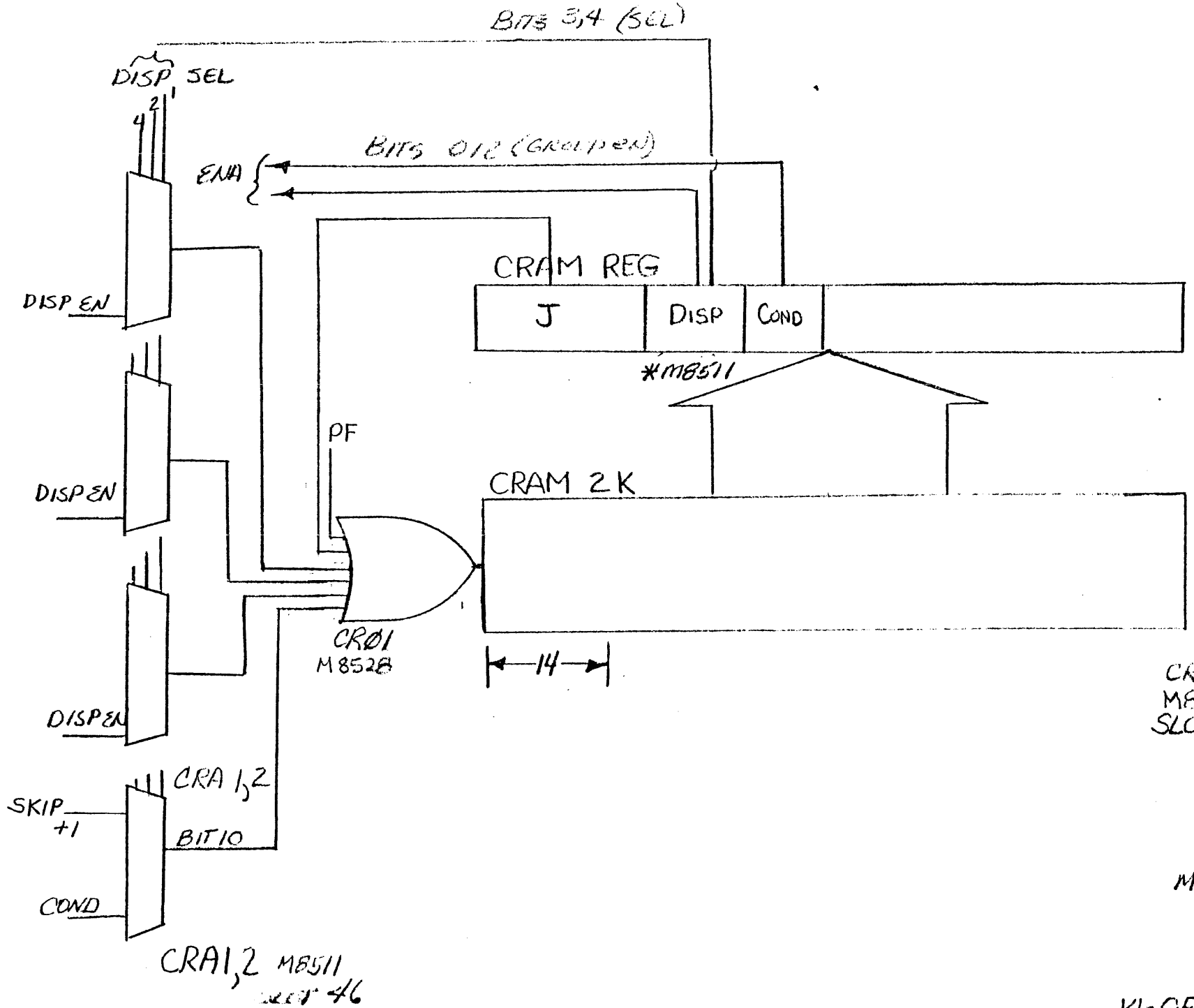
B/Skip/Jump/Comp

0	SJC L, E
1	SJC E
2	SJC L
3	SJC Never
4	SJC G
5	SJC Never
6	SJC G, E
7	SJC Always

Cannot use B & B-1-2 simultaneously

Can use B & B0 together

MICROCORE ADDRESSING



CR M8528
SLOT 40 x 14
42
44
50
52
M8528 46 x 5

A FIELD

MICROCODE [redacted]

MICRO WORD POSITION	0	1	2	3	4	5	6	7	8	9	10	11
SIGNAL NAME	N.U.	J00	J01	J02	J03	J04	J05	J06	J07	J08	J09	J10
MODULE SLOT NUMBER		50	50	50	44	44	44	44	42	42	42	42
MODULE PIN NUMBER		DV2	CK2	BP2	ED2	DV2	CK2	BP2	ED2	DV2	CK2	BP2
CRAM PHYSICAL BIT		05	06	07	08	09	10	11	12	13	14	15

C FIELD

	AR/ARM			ARX/ARXM			BR	BRX	MQ	FM ADR		
MICROWORD POSITION	24	25	26	27	28	29	30	31	32	33	34	35
SIGNAL NAME	SEL4	SEL2	SEL1	SEL4	SEL2	SEL1	LOAD	LOAD	SEL	4	2	1
MODULE SLOT NUMBER	50	50	50	40	44	44	42	42	40	40	40	40
MODULE PIN NUMBER	FK2	DR2	CF2	FF2	DR2	CF2	FP2	AR2	ED2	FP2	FK2	AR2
CRAM PHYSICAL BIT	45	64	66	36	68	70	52	54	16	56	57	58

0 AR
0 ARMM IF SPEC 22
1 CACHE
2 AD
3 E BUS
4 SH
5 AD*2
6 ADX
7 AD*.25

0 ARX
1 CACHE
2 AD
3 MQ
4 SH
5 ADX*2
6 ADX
7 ADX*.25

0 BR / 1 AR

0 BRX / 1 ARX

0 MQ / 1 SH

IF SPEC/MQ SHIFT

0 MQ*2
1 MQ*.25

IF COND/REG CTL

0 MQ SEL
1 MQM SEL

0 AC0
1 AC1
2 XR
3 VMA
4 AC2
5 AC3
6 AC4
7 #B#

E FIELD

	SH/ARMM				VMA		TIME		MEMORY			
MICROWORD POSITION	48	49	50	51	52	53	54	55	56	57	58	59
SIGNAL NAME	N.U.	SEL2	SEL1	N.U.	SEL2	SEL1	T00	T01	00	01	02	03
MODULE SLOT NUMBER		50	50		50	42	40	40	44	44	44	44
MODULE PIN NUMBER		AR2	AV2		FP2	DR2	DR2	CF2	FP2	FK2	AR2	AV2
CRAM PHYSICAL BIT		46	47		44	72	76	78	48	49	50	51

SH FIELD

0 SHIFT AR,ARX
1 AR
2 ARX
3 AR SWAP

ARMM FIELD

0 #
1 EXP SIGN
2 SCAD EXP
3 SCAD POS WITH ARL IND

0 VMA
1 PC
2 PC + 1
3 AD

0 2T
1 3T
2 4T
3 5T

0 NO OP
1 ARL IND
2 MB WAIT
3 SEC 0
4 A READ
5 B WRITE
6 FETCH
7 REG FUNCTION

IF X ADDR

10 AD FUNCTION
11 EA CALCULATION
12 LOAD AR
13 LOAD ARX
14 READ-WRITE
15 READ-PAUSE-WRITE
16 WRITE
17 IFET

IF NOT X ADDR

10 A IND
11 BYTE IND
12 LOAD AR
13 LOAD ARX
14 AD FUNCTION
15 BYTE READ
16 WRITE
17 READ-PAUSE-WRITE

G FIELD #1

MAGIC NUMBER FIELD

MACROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CU2	BF1	EP2	CU2	BF1	EP2	CU2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

ENABLES REQUIRED

1=CALL

ARØ-8
1-LD

USED TO ADDRESS FAST MEMORY

ARL IND

CLR

1=ARR
2=ARL
3=AR
4=ARX
6=ARL+ARX
7=AR+ARX
10=MQ
11=ARR+MQ
13=AR+MQ
14=ARX+MQ
16=ARL+ARX+MQ
17=AR+ARX+MQ

LOAD ARL

0=ARL
0=ARMM IF BIT 76
1=CACHE
2=AD
3=EBUS
4=SH
5=AD*2
6=ADX
7=AD*.25

CONO/REG CTL

AR CTL

1=ARR LOAD
2=AR9-17 LOAD
4=ARØ-8 LOAD
6=ARL LOAD

EXPTST
1=
AR -EXP

CONO/REG CTL AND MQ/MQ SEL

CONO/REG CTL AND MQ/MQM SEL

MQ CTL

0=MQ
1=MQ*2
2=MQ*.5
3=0'S

0=SH
1=MQ*.25
2=IS
3=AD

G FIELD #3

MAGIC NUMBER FIELD

MICROWORD POSITION	72	73	74	75	76	77	78	79	80	81	82	83
SIGNAL NAME	NOT USED	NOT USED	CRAM MARK	#00	#01	#02	#03	#04	#05	#06	#07	#08
MODULE SLOT #			52	44	44	44	42	42	42	40	40	40
MODULE PIN #			AV2	EP2	CU2	BF1	EP2	CU2	BF1	EP2	CU2	BF1
CRAM PHYS BIT #			43	29	30	31	33	34	35	37	38	39

CONO/MBOX CTL

MBOX CTL

00 NORMAL	21 CLR PT DIR LINE
01 PT DIR CLR	100 SET I/O PF ERR
10 PT WR	200 SET PAGE FAIL
20 PT DIR WR	

CONO/EBUS CTL

EBUS CTL

0=REL EEBUS	26 DATA O
1=INPUT	27 DATA I
2=DATA I/O	30 I/O INIT
4=DISABLE CS	60 EBUS DEMAND
10=CTL -IR	100 REL EBUS
20=EBUS NO DEMAND	400 GRAB EBUS

CONO/DIAG FUNC

DIAG FUNC

400 .5 uSEC	511 DATAI PAG (L)
404 LD PA LEFT	511 RD PERF CNT
405 LD PA RIGHT	512 CONI APR (L)
406 CONO MTR	512 RD EBOX CNT
407 CONO TIM	513 DATAI APR
414 CONO APR	513 RD CACHE CNT
415 CONO PI	514 RD INTRVL
416 CONO PAG	515 RD PERIOD
417 DATAO APR	516 CONI MTR
425 LD AC BLKS	517 RD MTR REQ
426 LD PCS+CWSX	530 CONI PI (PAR)
500 CONI PI (R)	531 CONI PAG
501 CONI PI (L)	567 RD EBUS REG
510 CONI APR (R)	620 DATAO PAG
510 RD TIME	