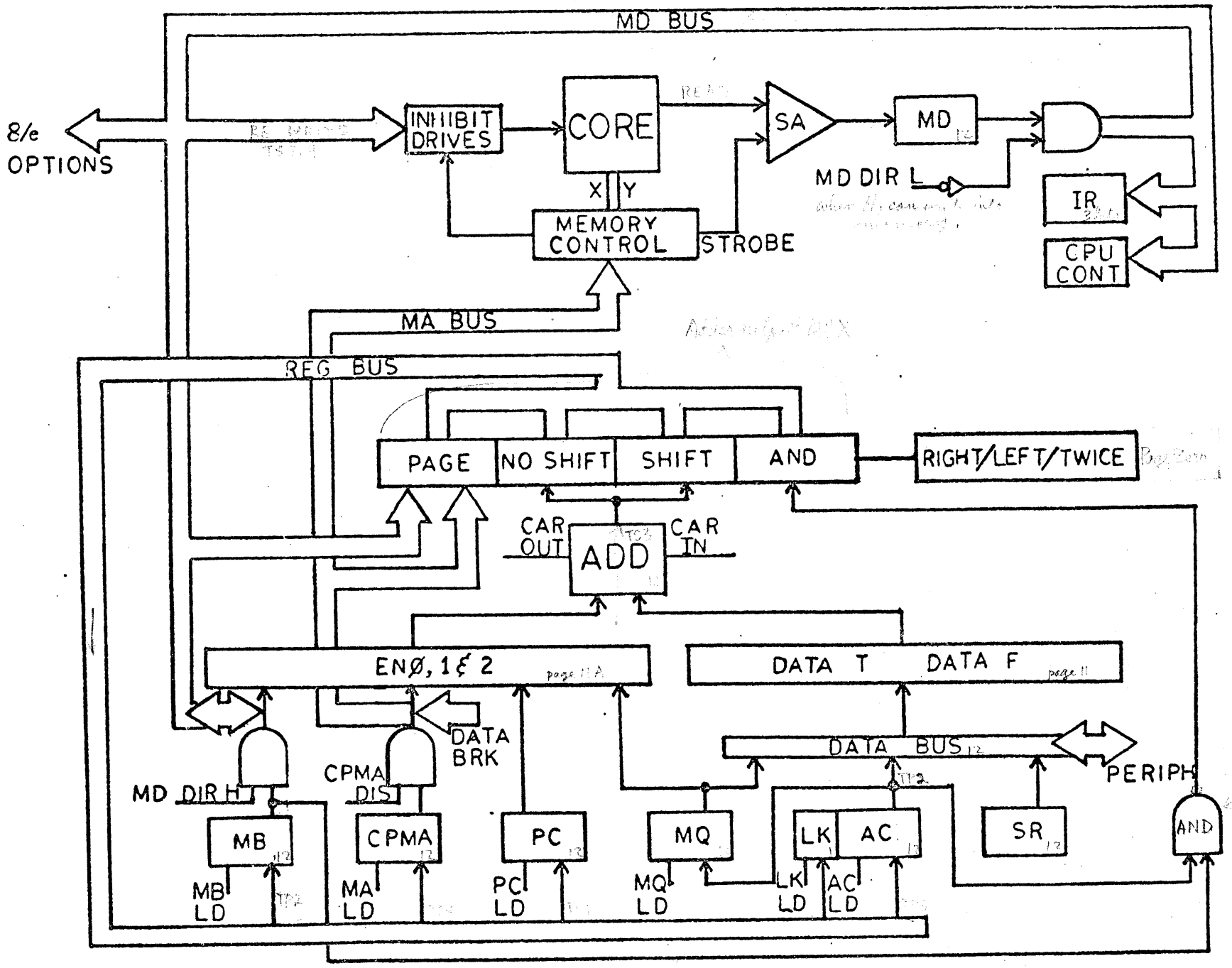
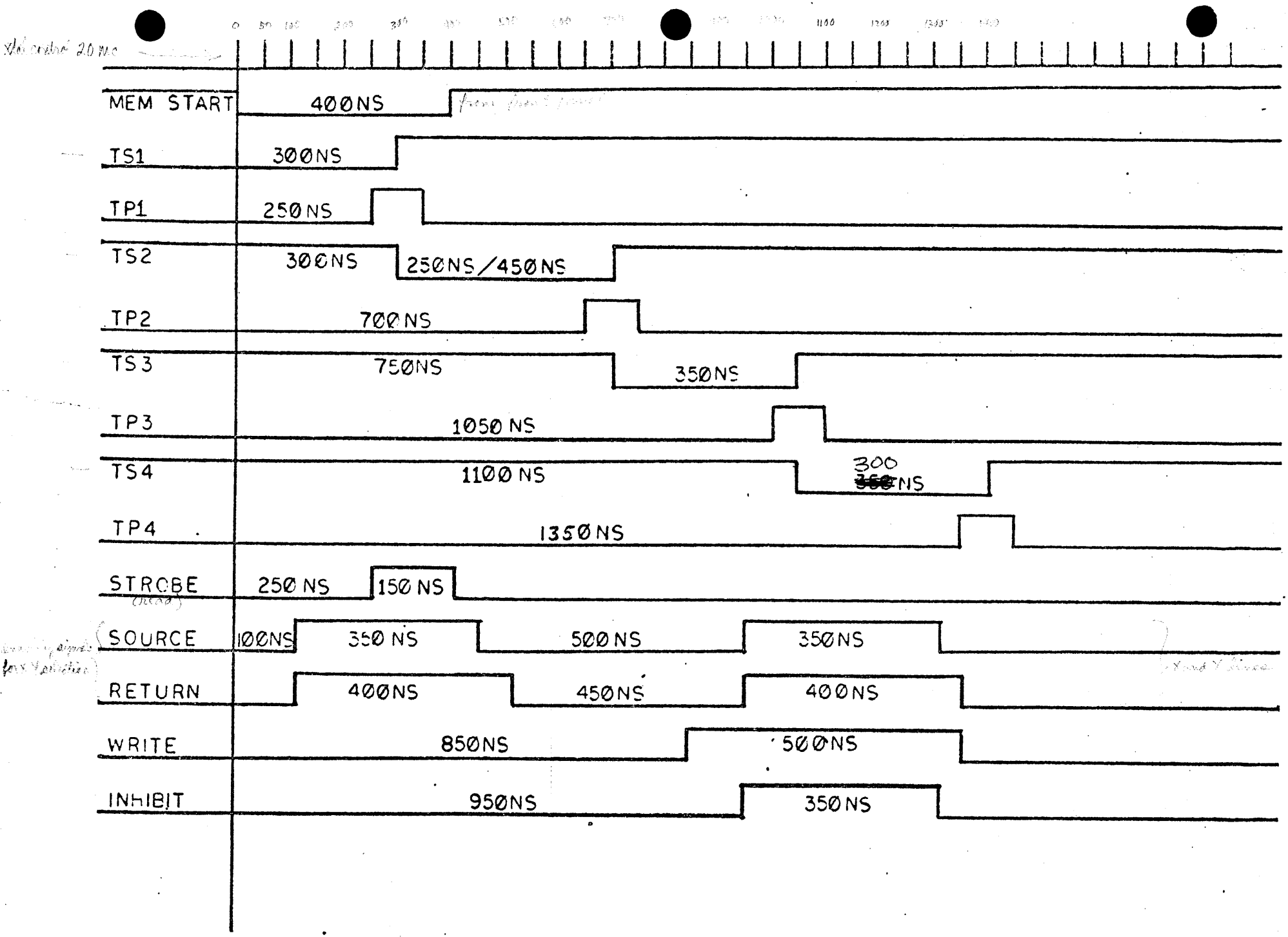
INSTRUCTOR: ARLEN SIMMS

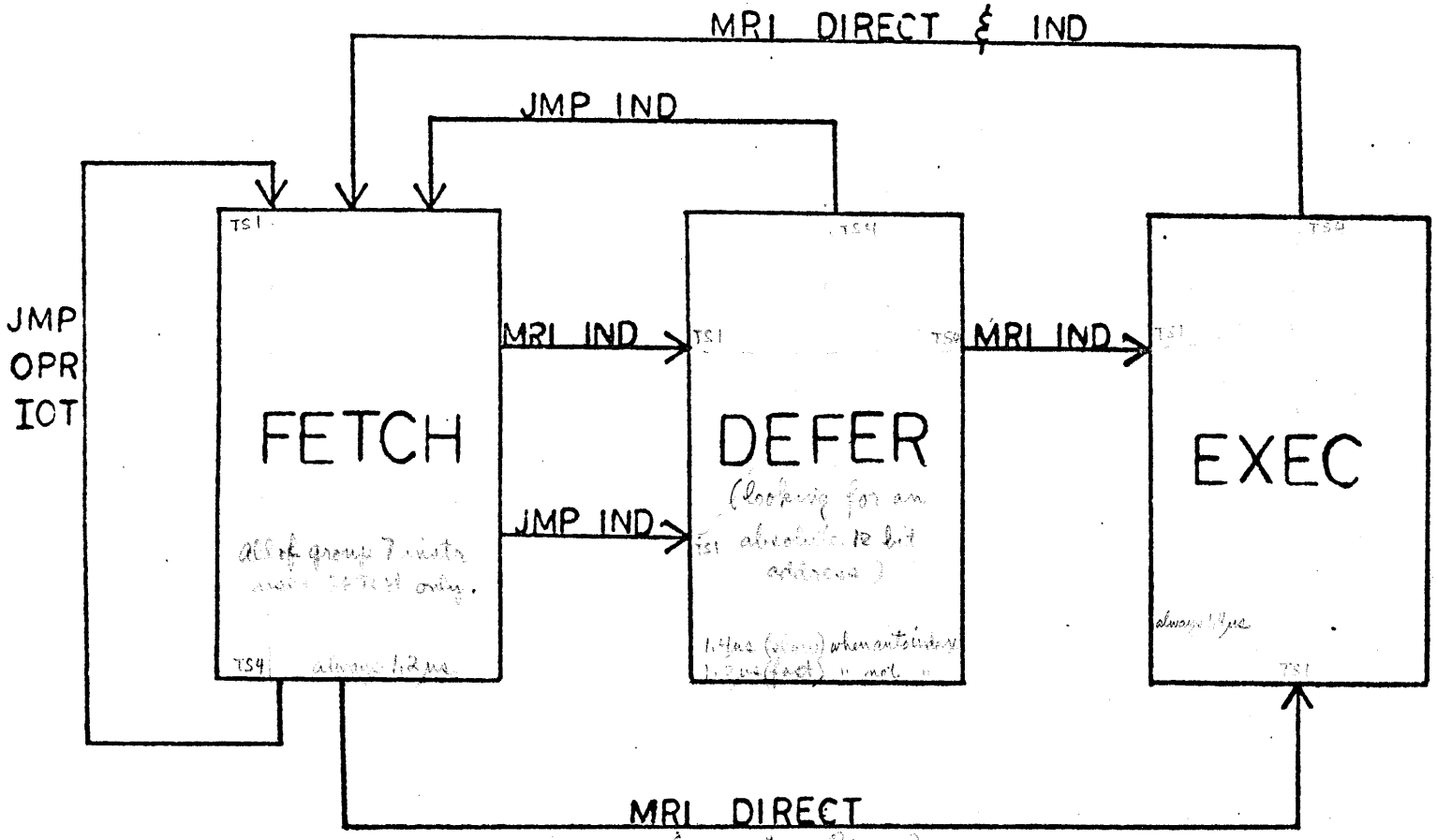
PDP-8e
HARDWARE FAMILIARIZATION
AND
INTERFACING

TABLE OF CONTENTS


MD BUS
&/e
OPTIONS
INHIBIT
DRIVES
CORE
SA
MD
MD DIR L
IR
CPU
CONT
X
Y
MEMORY
CONTROL
STROBE
MA BUS
REG BUS
PAGE
NO SHIFT
SHIFT
AND
RIGHT/LEFT/TWICE
CAR
OUT
ADD
CAR
IN
ENØ, 1 & 2
DATA T
DATA F
DATA
BRK
CPMA
DIS
MD DIR H
DATA BUS
PERIPH
MB
CPMA
PC
MQ
LK
AC
SR
AND
MB
LD
MA
LD
PC
LD
MQ
LD
LK
LD
AC
LD


MEM START
400NS
TS1
300NS
TP1
250NS
TS2
300NS
250NS/450NS
TP2
700NS
TS3
750NS
350NS
TP3
1050 NS
TS4
1100 NS
300
NS
TP4
1350NS
STROBE
250 NS
150 NS
SOURCE
100NS
350 NS
500 NS
350NS
RETURN
400NS
450NS
400NS
WRITE
850NS
500NS
INHIBIT
950NS
350 NS

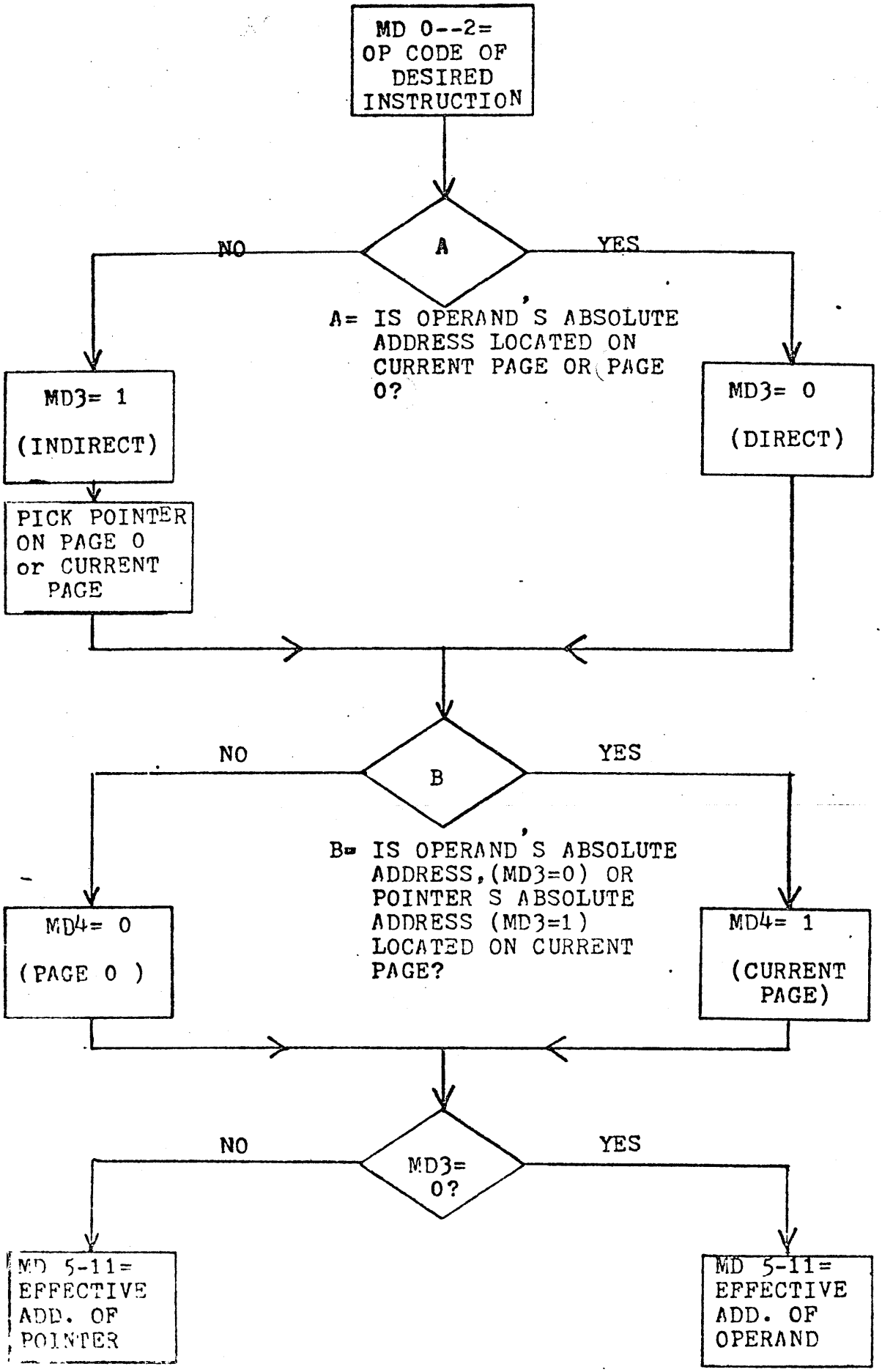
TIMING GEN.



MAJOR STATES GEN.

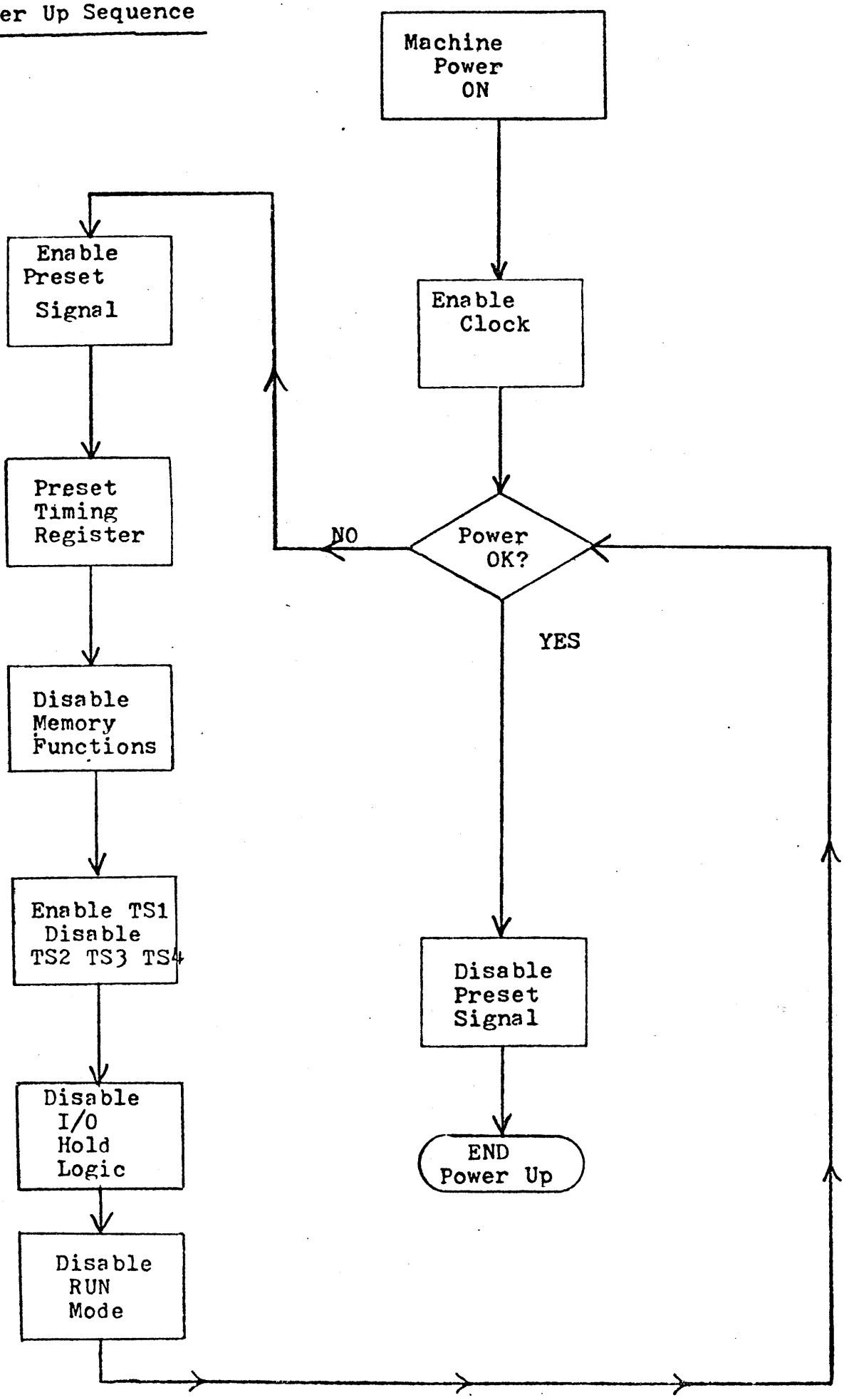
3 PROCESSOR STATES

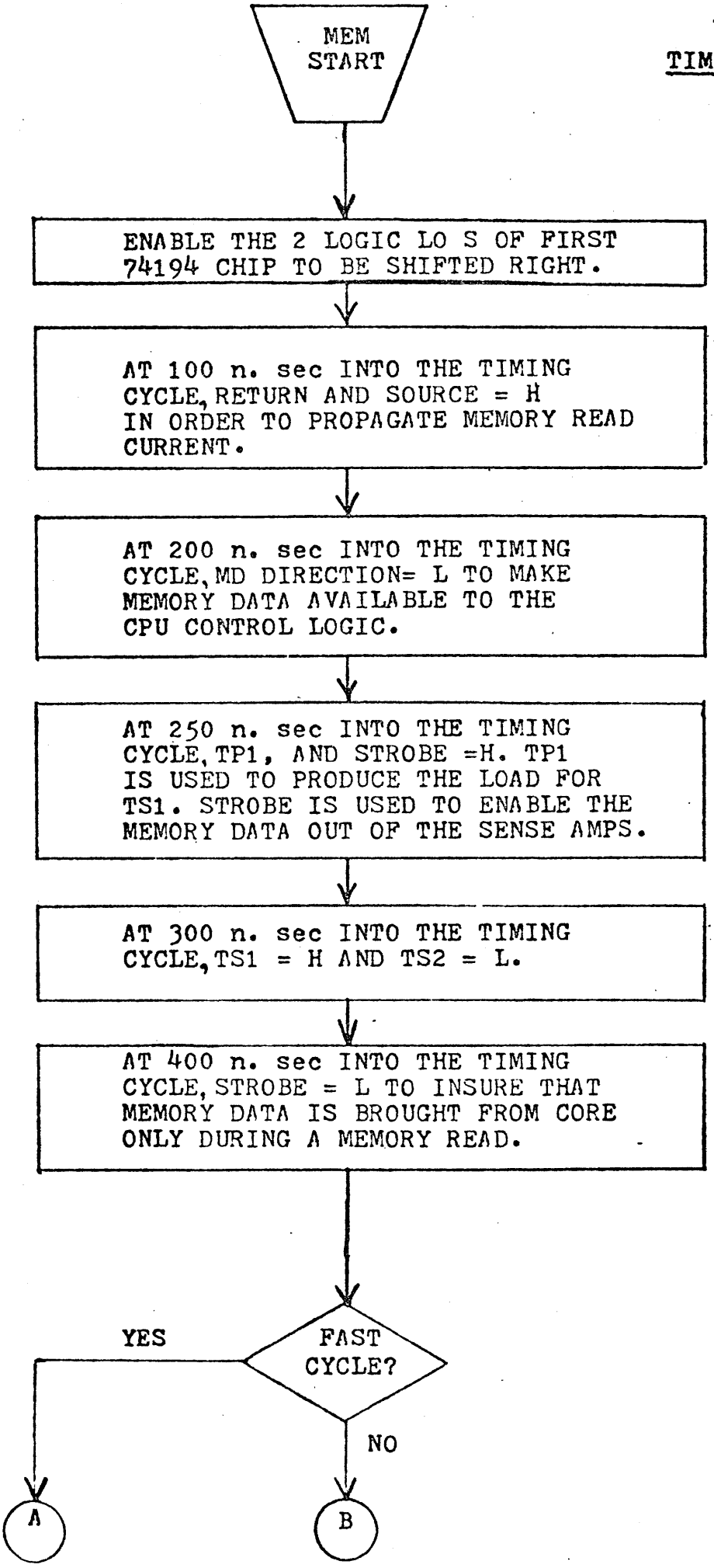
PDP 8/e
MRI INSTUCTION CODING
FLOW DIAGRAM

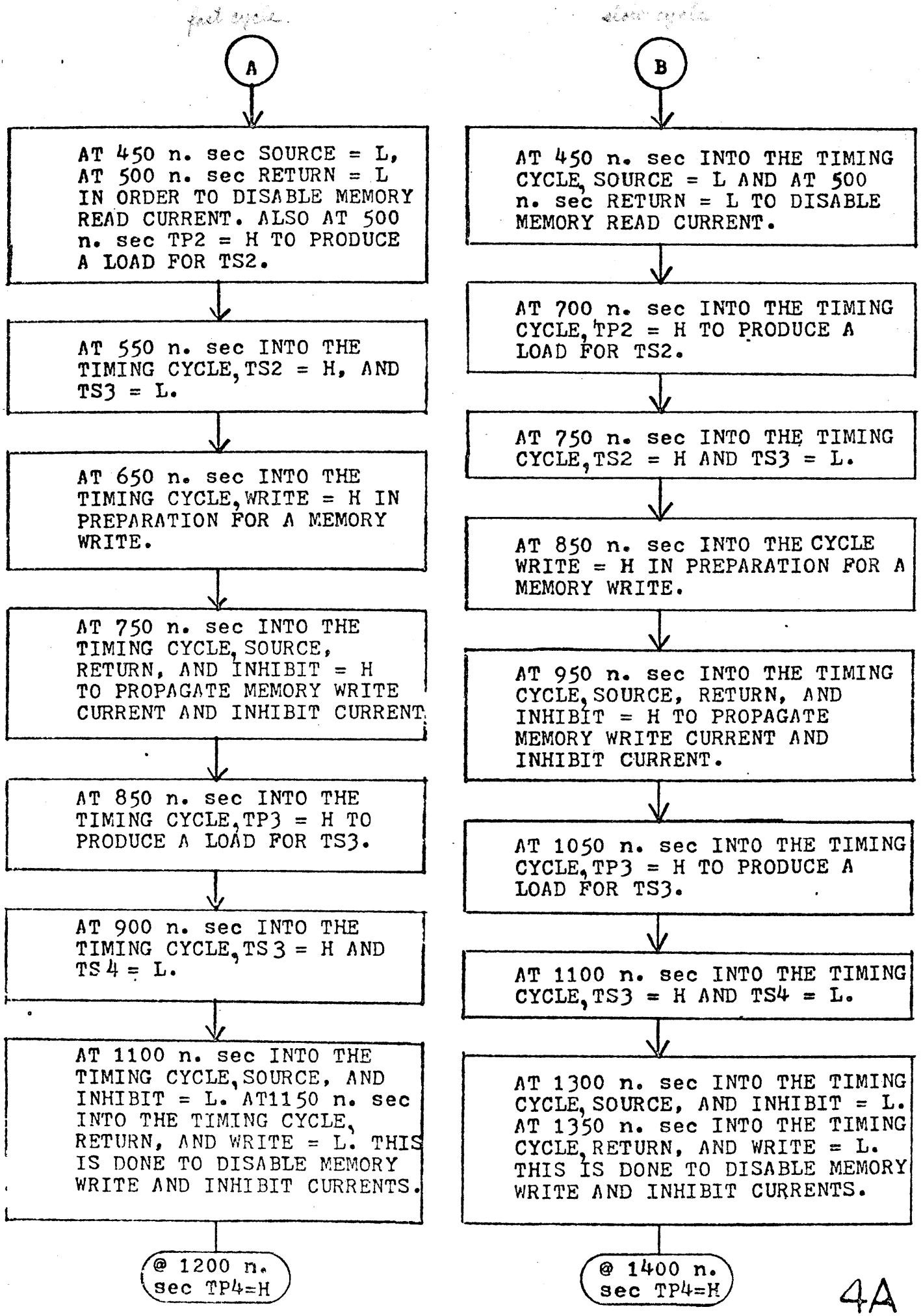


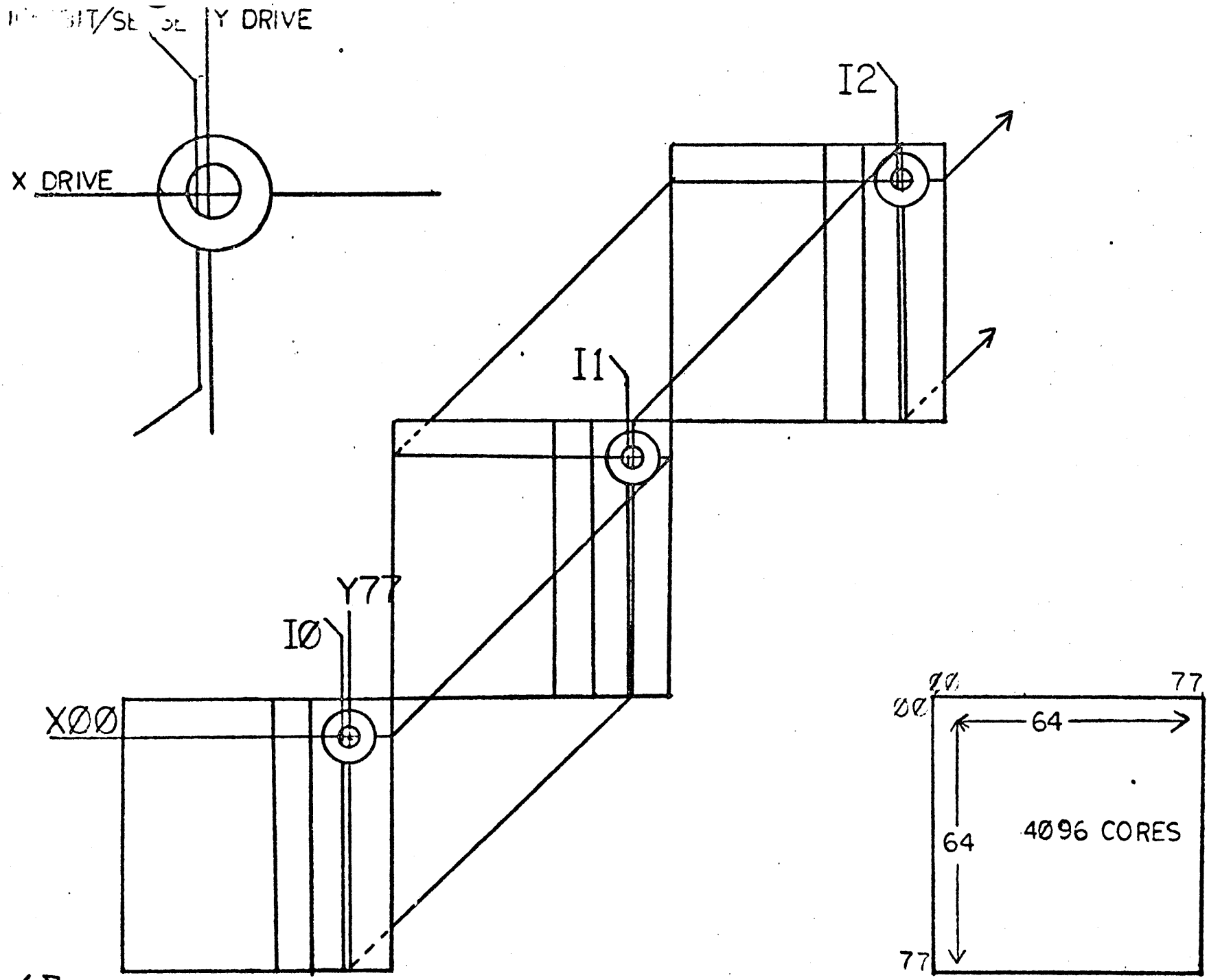
PDP 8/e

Power Up Sequence

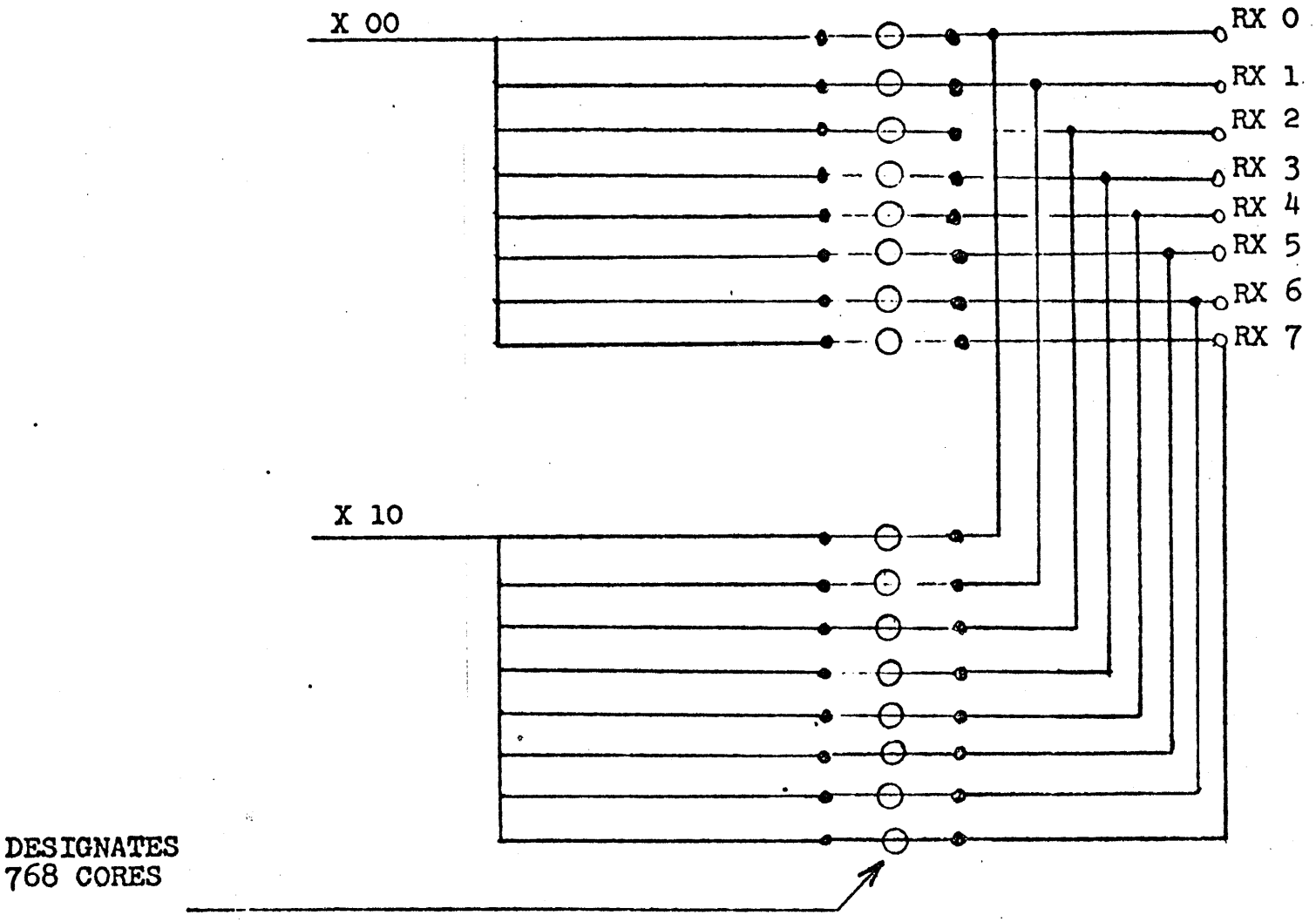



MEM START
ENABLE THE 2 LOGIC LO S OF FIRST 74194 CHIP TO BE SHIFTED RIGHT.
AT 100 n. sec INTO THE TIMING CYCLE, RETURN AND SOURCE = H IN ORDER TO PROPAGATE MEMORY READ CURRENT.
AT 200 n. sec INTO THE TIMING CYCLE, MD DIRECTION= L TO MAKE MEMORY DATA AVAILABLE TO THE CPU CONTROL LOGIC.
AT 250 n. sec INTO THE TIMING CYCLE, TP1, AND STROBE =H. TP1 IS USED TO PRODUCE THE LOAD FOR TS1. STROBE IS USED TO ENABLE THE MEMORY DATA OUT OF THE SENSE AMPS.
AT 300 n. sec INTO THE TIMING CYCLE, TS1 = H AND TS2 = L.
AT 400 n. sec INTO THE TIMING CYCLE, STROBE = L TO INSURE THAT MEMORY DATA IS BROUGHT FROM CORE ONLY DURING A MEMORY READ.
FAST CYCLE?
YES
NO
A
B


A
AT 450 n. sec SOURCE = L, AT 500 n. sec RETURN = L IN ORDER TO DISABLE MEMORY READ CURRENT. ALSO AT 500 n. sec TP2 = H TO PRODUCE A LOAD FOR TS2.
AT 550 n. sec INTO THE TIMING CYCLE, TS2 = H, AND TS3 = L.
AT 650 n. sec INTO THE TIMING CYCLE, WRITE = H IN PREPARATION FOR A MEMORY WRITE.
AT 750 n. sec INTO THE TIMING CYCLE, SOURCE, RETURN, AND INHIBIT = H TO PROPAGATE MEMORY WRITE CURRENT AND INHIBIT CURRENT
AT 850 n. sec INTO THE TIMING CYCLE, TP3 = H TO PRODUCE A LOAD FOR TS3.
AT 900 n. sec INTO THE TIMING CYCLE, TS3 = H AND TS4 = L.
AT 1100 n. sec INTO THE TIMING CYCLE, SOURCE, AND INHIBIT = L. AT1150 n. sec INTO THE TIMING CYCLE, RETURN, AND WRITE = L. THIS IS DONE TO DISABLE MEMORY WRITE AND INHIBIT CURRENTS.
@ 1200 n. sec TP4=H
B
AT 450 n. sec INTO THE TIMING CYCLE, SOURCE = L AND AT 500 n. sec RETURN = L TO DISABLE MEMORY READ CURRENT.
AT 700 n. sec INTO THE TIMING CYCLE, TP2 = H TO PRODUCE A LOAD FOR TS2.
AT 750 n. sec INTO THE TIMING CYCLE, TS2 = H AND TS3 = L.
AT 850 n. sec INTO THE CYCLE WRITE = H IN PREPARATION FOR A MEMORY WRITE.
AT 950 n. sec INTO THE TIMING CYCLE, SOURCE, RETURN, AND INHIBIT = H TO PROPAGATE MEMORY WRITE CURRENT AND INHIBIT CURRENT.
AT 1050 n. sec INTO THE TIMING CYCLE, TP3 = H TO PRODUCE A LOAD FOR TS3.
AT 1100 n. sec INTO THE TIMING CYCLE, TS3 = H AND TS4 = L.
AT 1300 n. sec INTO THE TIMING CYCLE, SOURCE, AND INHIBIT = L. AT 1350 n. sec INTO THE TIMING CYCLE, RETURN, AND WRITE = L. THIS IS DONE TO DISABLE MEMORY WRITE AND INHIBIT CURRENTS.
@ 1400 n. sec TP4=H


SIT/SE
Y DRIVE
X DRIVE
I2
I1
Y77
IØ
XØØ
ØØ
77
64
64
4096 CORES
77

EXAMPLE OF X READ LINE LAYOUT



MEMORY ADDRESS SELECTION

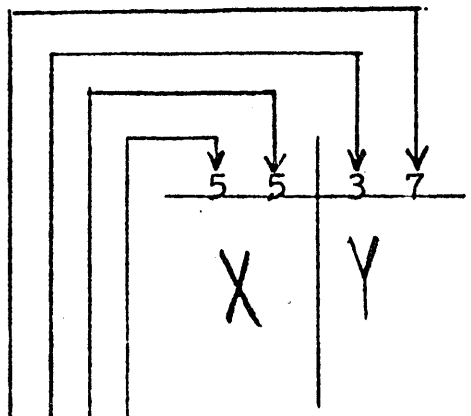


FIG. 1

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 0---2, WILL SELECT A READ OR WRITE LINE IN THE UPPER LEFT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 3---5, WILL SELECT A READ OR WRITE LINE IN THE UPPER RIGHT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 6---8, WILL SELECT A READ OR WRITE LINE IN THE LOWER LEFT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 9---11, WILL SELECT A READ OR WRITE LINE IN THE LOWER RIGHT HAND QUADRANT.

FIG. 2

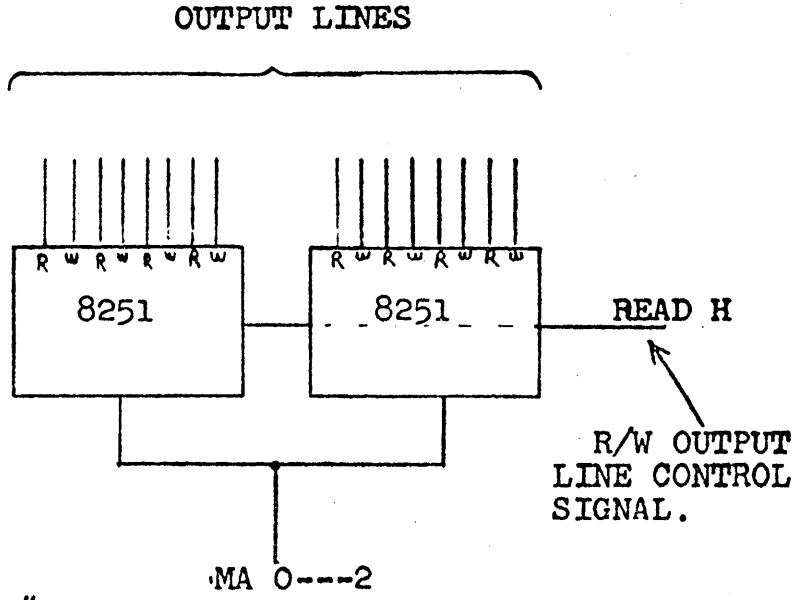
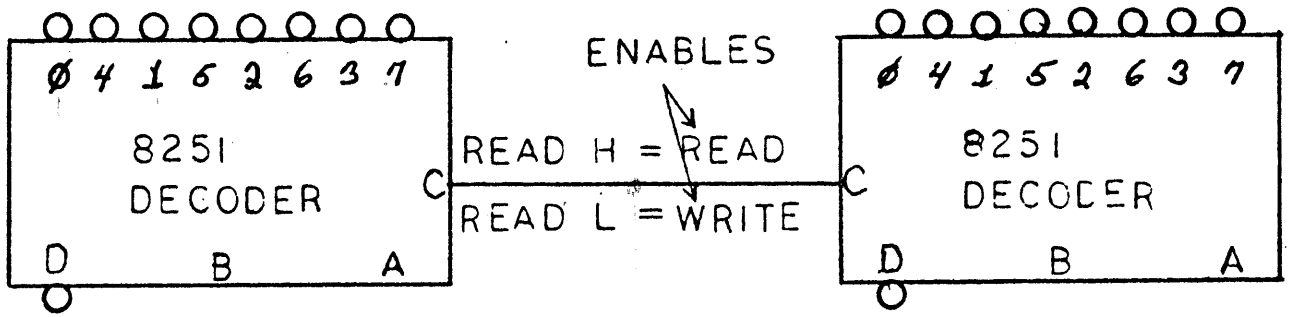


FIG. 2. ILLUSTRATES THE UPPER LEFT HAND QUADRANT OF PRINT 3. EACH 8251 HAS 8 OUTPUT LINES; 4 FOR READ AND 4 FOR WRITE FUNCTIONS. ONE READ OR WRITE LINE(DEPENDING ON THE FUNCTION) MUST BE SELECTED IN ORDER TO SELECT THE DESIRED ADDRESS. THIS IS DONE BY THE COMBINATION OF THE MA BITS(THE OUTPUT LINES WIRED AS SHOWN IN HANDOUT). THE ABOVE INFORMATION IS APPLICABLE TO THE OTHER 3 QUADRANTS OF PRINT 3.

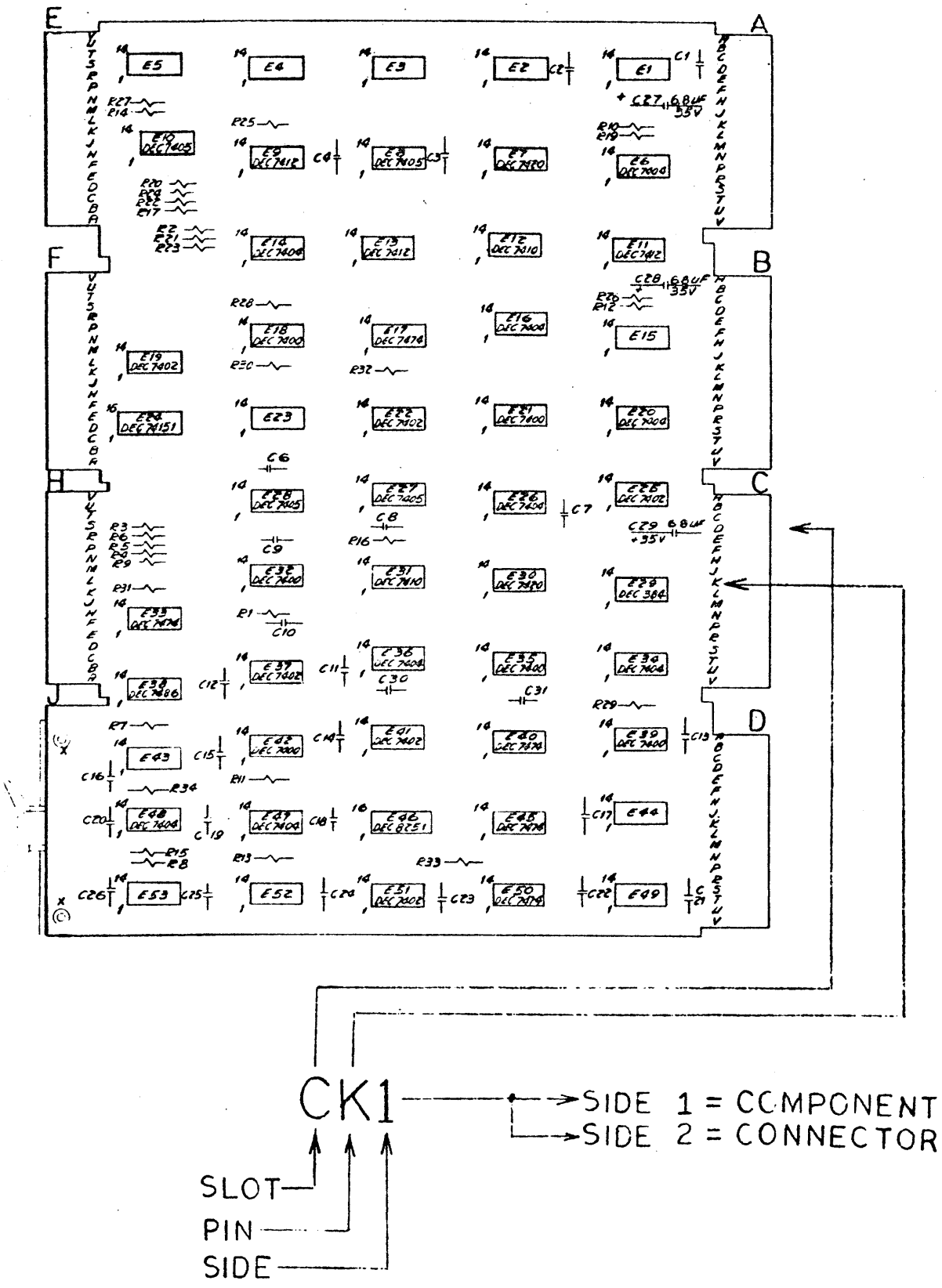
MEMORY ADDRESS DECODER

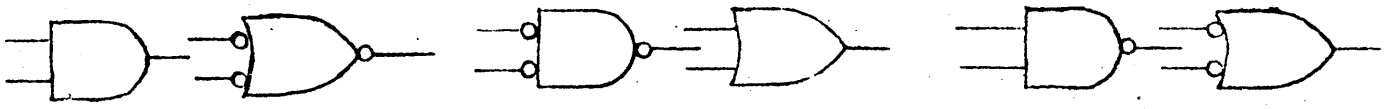
WHEN READING, USE OUT ACTIVE PINS 4,5,6,7
WHEN WRITING, USE OUT ACTIVE PINS 0,1,2,3



MA 0 1 2 3 4 5 6 7 8 9 10 11	MA INPUT D B A	CUT ACTIVE PINS READ	CUT ACTIVE PINS WRITE
0 0 0	L L L	4	0
0 0 1	L L H	5	1
0 1 0	L H L	6	2
0 1 1	L H H	7	3

MA 0 1 2 3 4 5 6 7 8 9 10 11	MA INPUT D B A	OUT ACTIVE PINS READ	OUT ACTIVE PINS WRITE
1 0 0	L L L	4	0
1 0 1	L L H	5	1
1 1 0	L H L	6	2
1 1 1	L H H	7	3

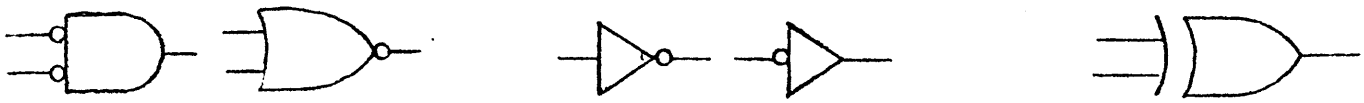

CK1
SIDE 1 = COMPONENT
SIDE 2 = CONNECTOR
SLOT
PIN
SIDE



A	B	C
H	H	H
H	L	L
L	H	L
L	L	L

A	B	C
H	H	H
H	L	H
L	H	H
L	L	L

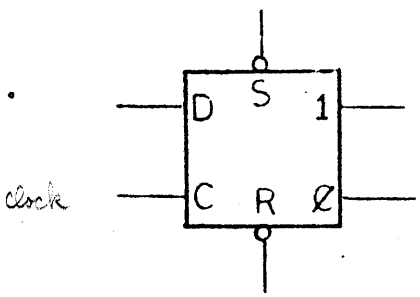
A	B	C
H	H	L
H	L	H
L	H	H
L	L	H



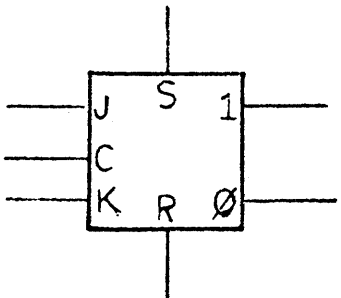
A	B	C
H	H	I
H	L	L
L	H	L
L	L	H

H	L	H
L	H	L

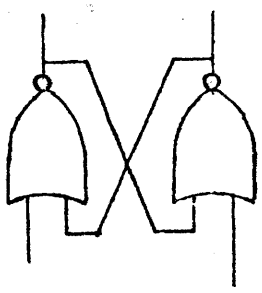
A	B	C
H	H	L
H	L	H
L	H	H
L	L	L



D



JK



RS

o ≡ low

— ≡ Hi

ADDER LOGIC TRUTH TABLES

EN0	EN1	EN2	RESULT
L	L	L	PC
L	L	H	MD
L	H	L	MQ
L	H	H	MA
H	X	X	ZERO

DATA T	DATA F	RESULT
L	L	COMPLEMENT
L	H	TRUE
H	L	ZERO

PAGE	RIGHT	LEFT	TWICE	RESULT
L	L	L	L	CURRENT PAGE
X	L	L	H	AND
X	L	H	L	RIGHT 2
X	L	H	H	RIGHT 1
X	H	L	L	LEFT 2
X	H	L	H	LEFT 1
X	H	H	L	SWAP
X	H	H	H	NO SHIFT
H	L	L	L	PAGE ZERO

I/O SIGNALS

C0	C1	C2	RESULT
L	L	L	DATA → PC
L	L	H	DATA → AC
L	H	L	PC + DATA → PC
L	H	H	AC → DATA, CLR AC
H	L	L	DATA → PC
H	L	H	AC/DATA ored → AC
H	H	L	PC + DATA → PC
H	H	H	AC/DATA ored → AC

ADDER CONDITIONS

LEFT	RIGHT	CAR IN	SUM	CAR OUT
H	H	H	H	H
H	L	H	L	H
L	L	H	H	L
H	H	L	L	H
H	L	L	L	H
L	L	L	L	L

Adder Output Mux

RIGHT	LEFT	TWICE	PAGE Z	USE
L	L	L	L	CURRENT PAGE
L	L	H	X	AND
L	H	L	X	RTR
L	H	H	X	RAR
H	L	L	X	RTL
H	L	H	X	RAL
H	H	L	X	BYTE SWAP
H	H	H	X	NO SHIFT
L	L	L	H	PAGE ZERO

LEFT	RIGHT	CI	SUM	CO
H	H	H	H	H
H	L	H	L	H
L	L	H	H	L
H	H	L	L	H
H	L	L	H	L
L	L	L	H	L

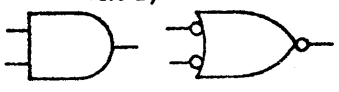
ENO	EN1	EN2	ADDER IN
L	L	L	PC
L	L	H	MD
L	H	L	MQ
L	H	H	CPMA
H	X	X	ZERO

DATA T	DATA F	ADDER IN
		DATA BUS
L	L	DATA BUS
L	H	DATA BUS
H	L	ZERO

IND1	IND2	RESULT
L	L	AC → BUS
L	H	BUS
H	L	MQ → BUS
H	H	STATUS

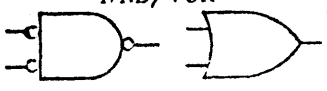
C0	C1	C2	RESULT
L	L	L	DATA → PC
L	L	H	DATA → AC
L	H	L	PC + DATA → PC
L	H	H	AC → DATA, 0 → AC
H	L	L	DATA → PC
H	L	H	AC ored DATA → AC
H	H	L	PC + DATA → PC
H	H	H	AC ored DATA → AC

+AND/-OR



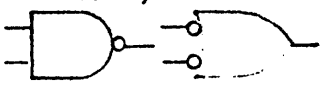
L	L	L
L	H	L
H	L	L
H	H	H

-AND/+OR



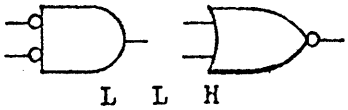
L	L	L
L	H	H
H	L	H
H	H	H

+NAND/-NOR



L	L	H
L	H	H
H	L	H
H	H	L

-NAND/+NOR

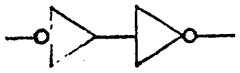


L	L	H
L	H	L
H	L	L
H	H	L

XOR

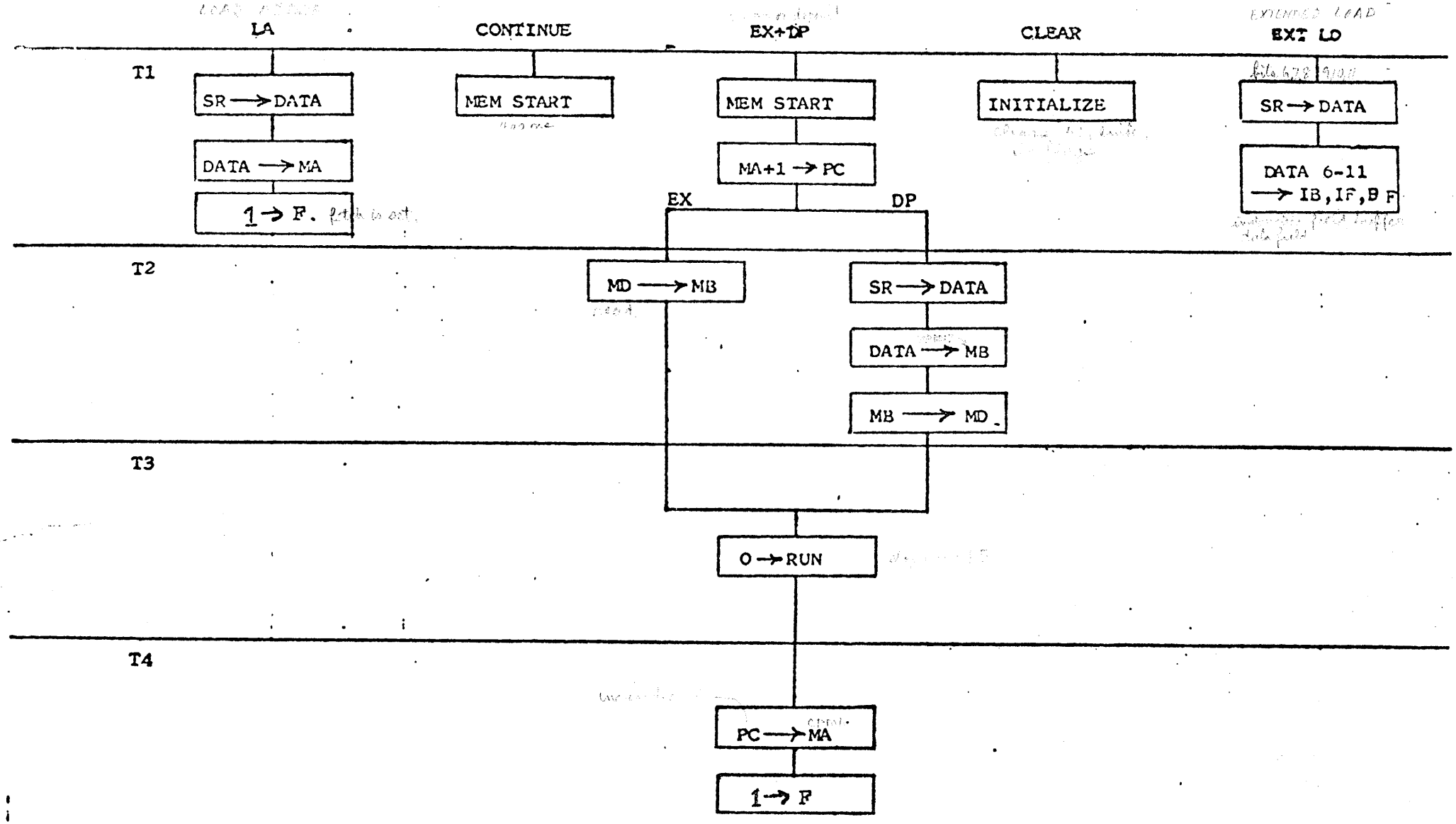
L	L	L
L	H	H
H	L	H
H	H	L

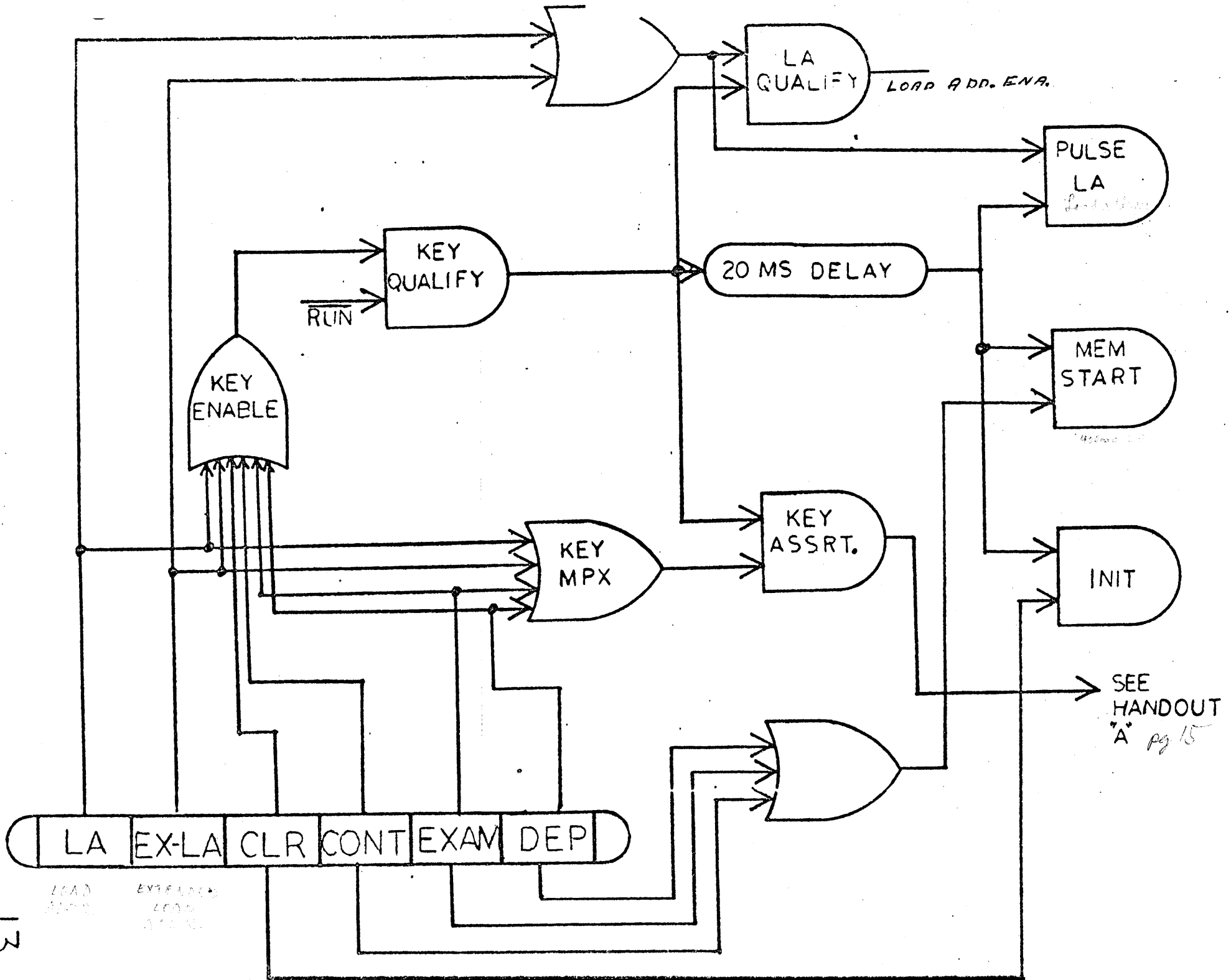
INVERTER

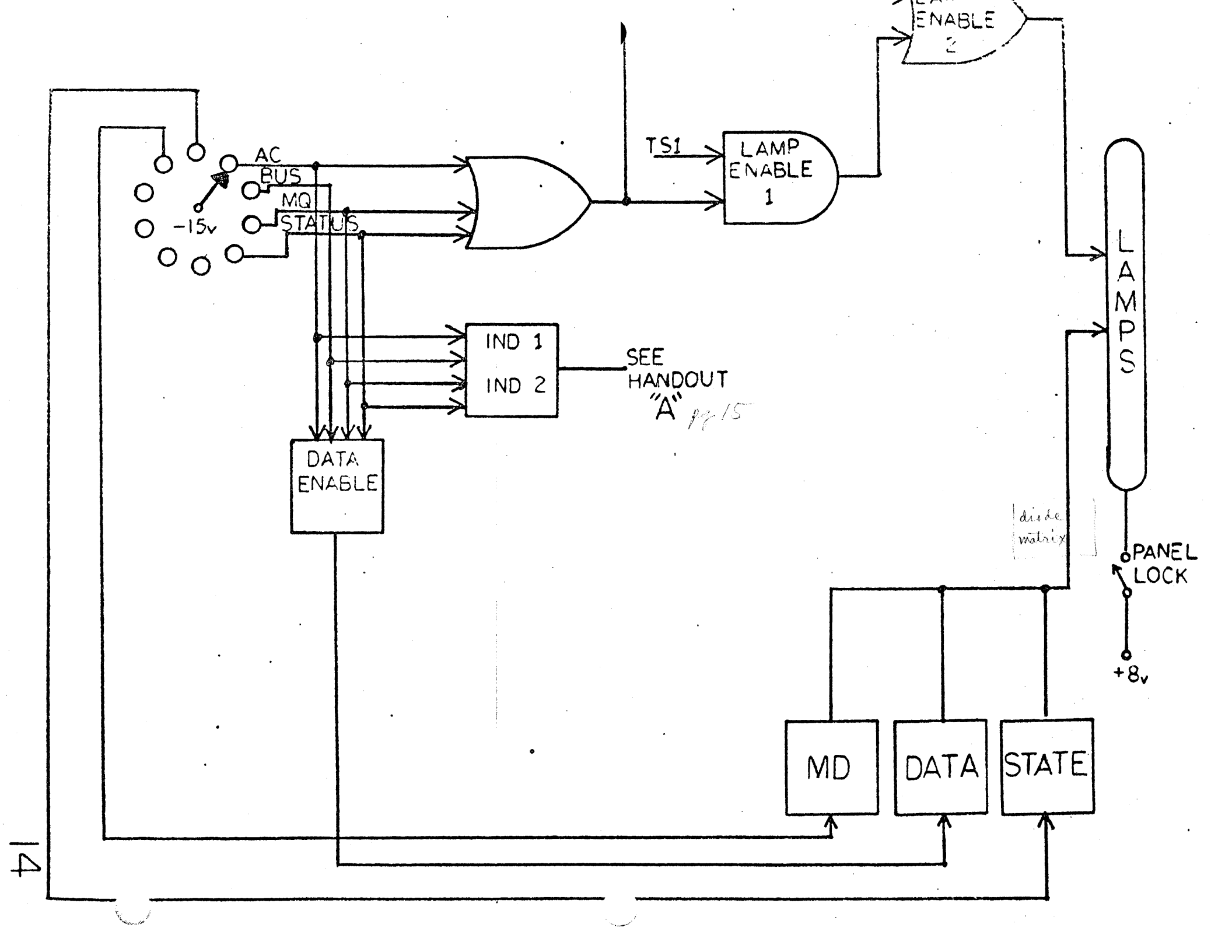


L	H
H	L

PDP - 8E MANUAL KF - FLOW DIAGRAM FRONT PANEL




LA QUALIFY
LOAD ADD. ENA.
PULSE LA
KEY QUALIFY
RUN
20 MS DELAY
MEM START
KEY ENABLE
KEY MPX
KEY ASSRT.
INIT
SEE HANDOUT "A" pg 15
LA
EX-LA
CLR
CONT
EXAM
DEP


ENABLE
2
AC
BUS
MQ
STATUS
-15v
TS1
LAMP
ENABLE
1
LAMPS
IND 1
IND 2
SEE
HANDOUT
"A"
pg 15
DATA
ENABLE
divide
matrix
PANEL
LOCK
+8v
MD
DATA
STATE
14

PDP 8/e

OSR is hard wired in front panel off

KEY FUNCTION SIGNALS

HANDOUT A

IND 1	IND 2.	Result
L	L	AC ⟶ BUS
L	H	BUS ⟶ BUS
H	L	MQ ⟶ BUS
H	H	STATUS ⟶ BUS

KEY LA

(A)	KEY CONTROL	H	Qualification for MA LOAD
(B)	MS DISABLE	L	Disables MAJOR STATES and IR
(C)	MD DIR. IN.	L	NO use for KEY LA
(D)	BK DATA CONTROL	L	NO use for KEY LA
(E)	SR----BUS	H	Enables SR to the DATA BUS

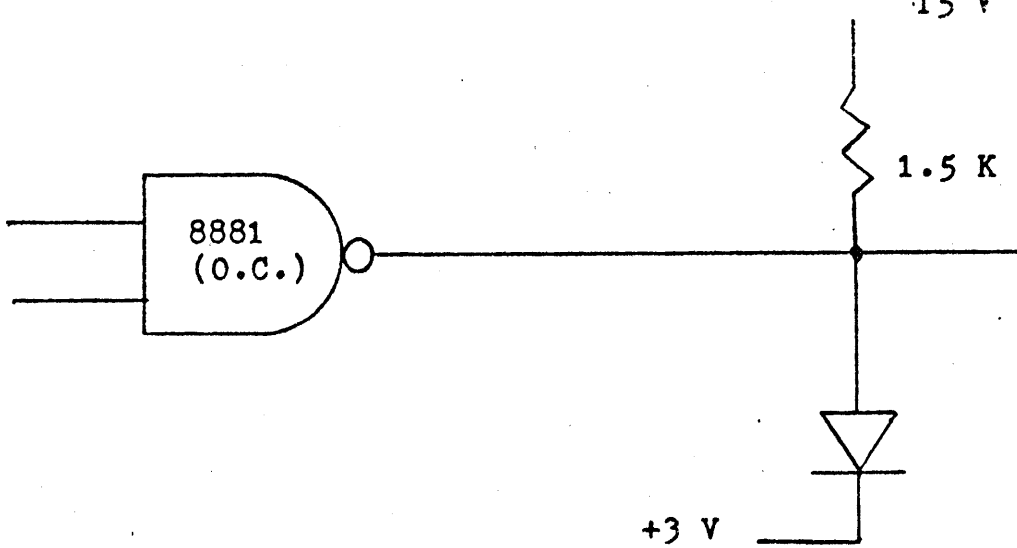
KEY EXAM

(A)	KEY CONTROL	L	Allows one timing cycle only
(B)	MS DISABLE	L	Disables MAJOR STATES and IR
(C)	MD DIR, IN.	L	Enables MEM ⟶ MD LINES
(D)	BK DATA CONTROL	L	Enables MD ⟶ MB
(E)	SR----BUS	L	Disables SR ⟶ BUS

KEY DEP

(A)	KEY CONTROL	L	Allows one timing cycle only
(B)	MS DISABLE	L	Disables MAJOR STATES and IR
(C)	MD DIR. IN.	H	Disables MEM ⟶ MD LINES
(D)	BK DATA CONTROL	H	Disables MD ⟶ MB
(E)	SR----BUS	H	Enables SR ⟶ BUS

BUS LOGIC CONCEPT AND 724 POWER SUPPLY REQUIREME



THE 1.5K RESISTOR AND THE DIODE ARE LOCATED ON THE M 8320 (or the M 832) BUS LOADS CARD.

H724 POWER SUPPLY

OUTPUT VOLTAGE	+15 V	+5 V	-15 V	+8 V
CURRENT OUTPUT	1 A	20 A	8 A	2 A
CURRENT USED	.8 A	6 A	4.5 A	1.25 A
UNUSED CURRENT	.2 A	14 A	3.5 A	

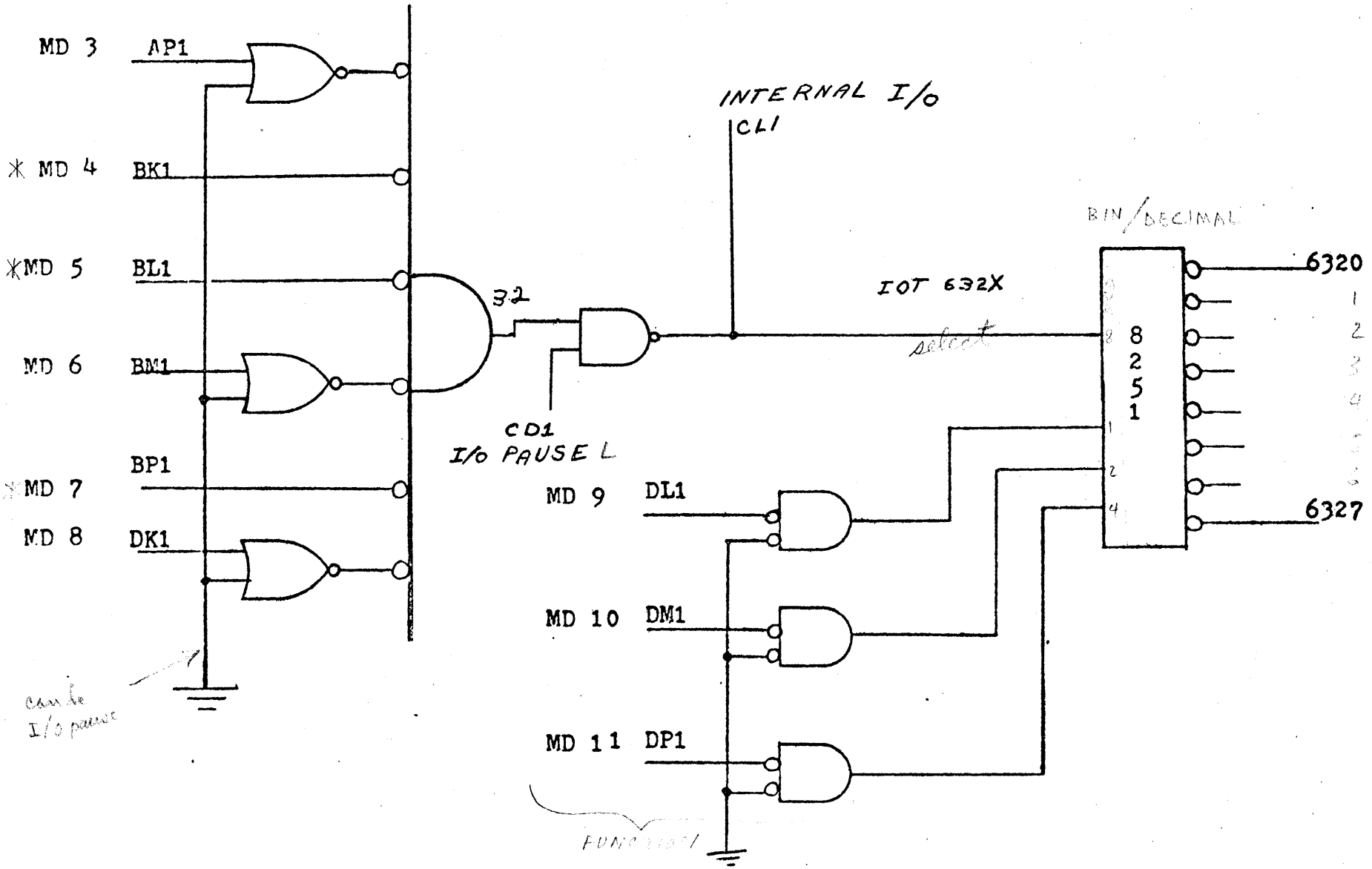
INTERN.

DEVICE CODE SELECTIO!

(632X)

011 010

Device code (code 32) Remember "1" is LO



INTERNAL I/O

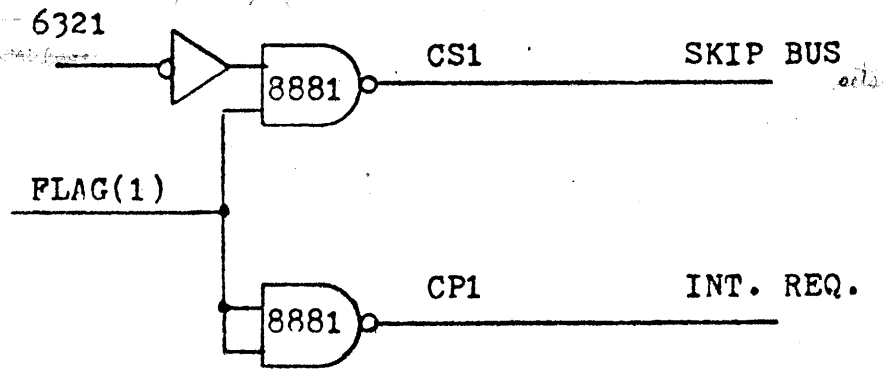
EXAMPLE INSTRUCTION SET

INSTRUCTION	FUNCTION
6321----------	CHECK FLAG AND SKIP IF =(1)
6322----------	SKIP ONCE IF FLAG "A" = 1, SKIP ONCE IF FLAG "B" = 1, SKIP TWICE IF FLAG "A" AND FLAG "B" ARE = 1.
6324----------	TRANSFER AC TO DEVICE
6325----------	TRANSFER AC TO DEVICE, 0---AC
6326----------	TRANSFER DATA FROM DEVICE TO AC.
6327----------	TRANSFER DATA TO THE PC.

device may have multiple flags

GATING FOR SKIP AND INT. REQ.

FIG. 1

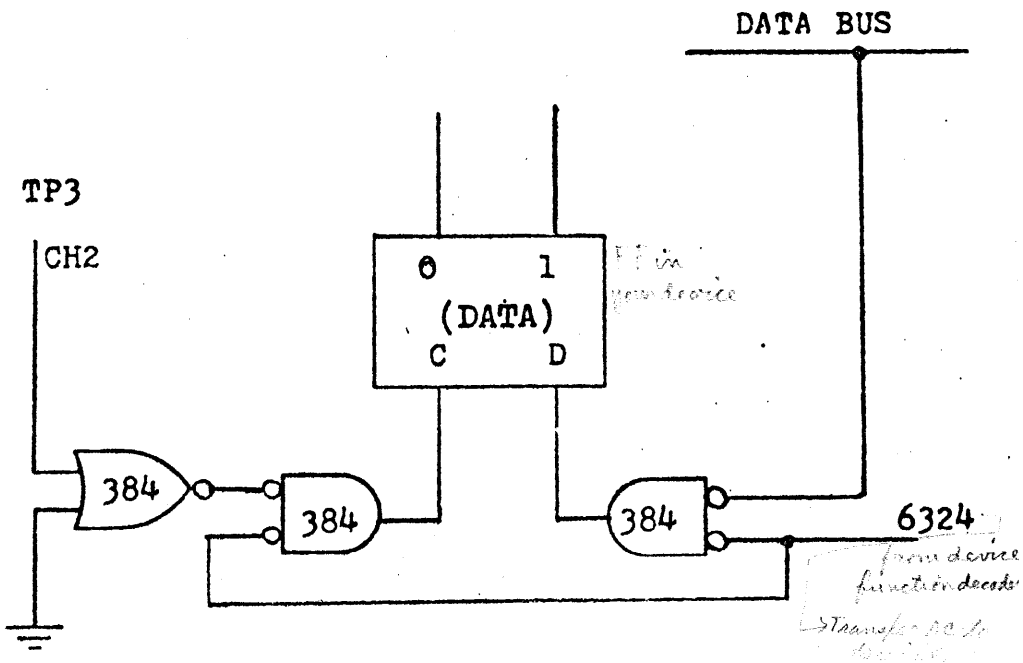


THE PURPOSE OF THIS GATING IS TO ALLOW SENSING OF SKIP CONDITIONS AND ACKNOWLEDGING INTERRUPT REQUESTS FROM THE DEVICE.

I/O OUTPUT TRANSFER GATING

C0	C1	C2	
H	H	H	= AC∨DATA----AC

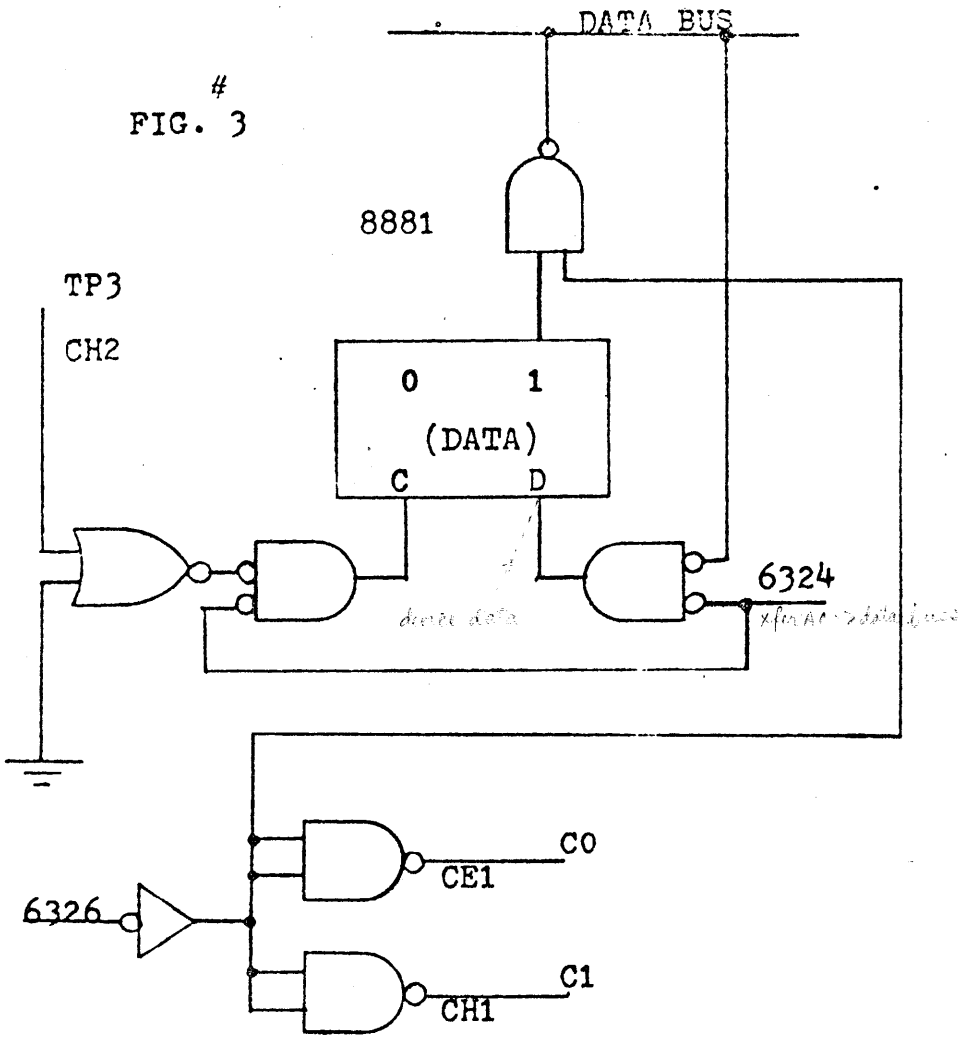
FIG. 2



THE PURPOSE OF THIS GATING IS TO ALLOW THE TRANSFER OF THE AC TO A REGISTER IN THE DEVICE.

DATA----AC GATING

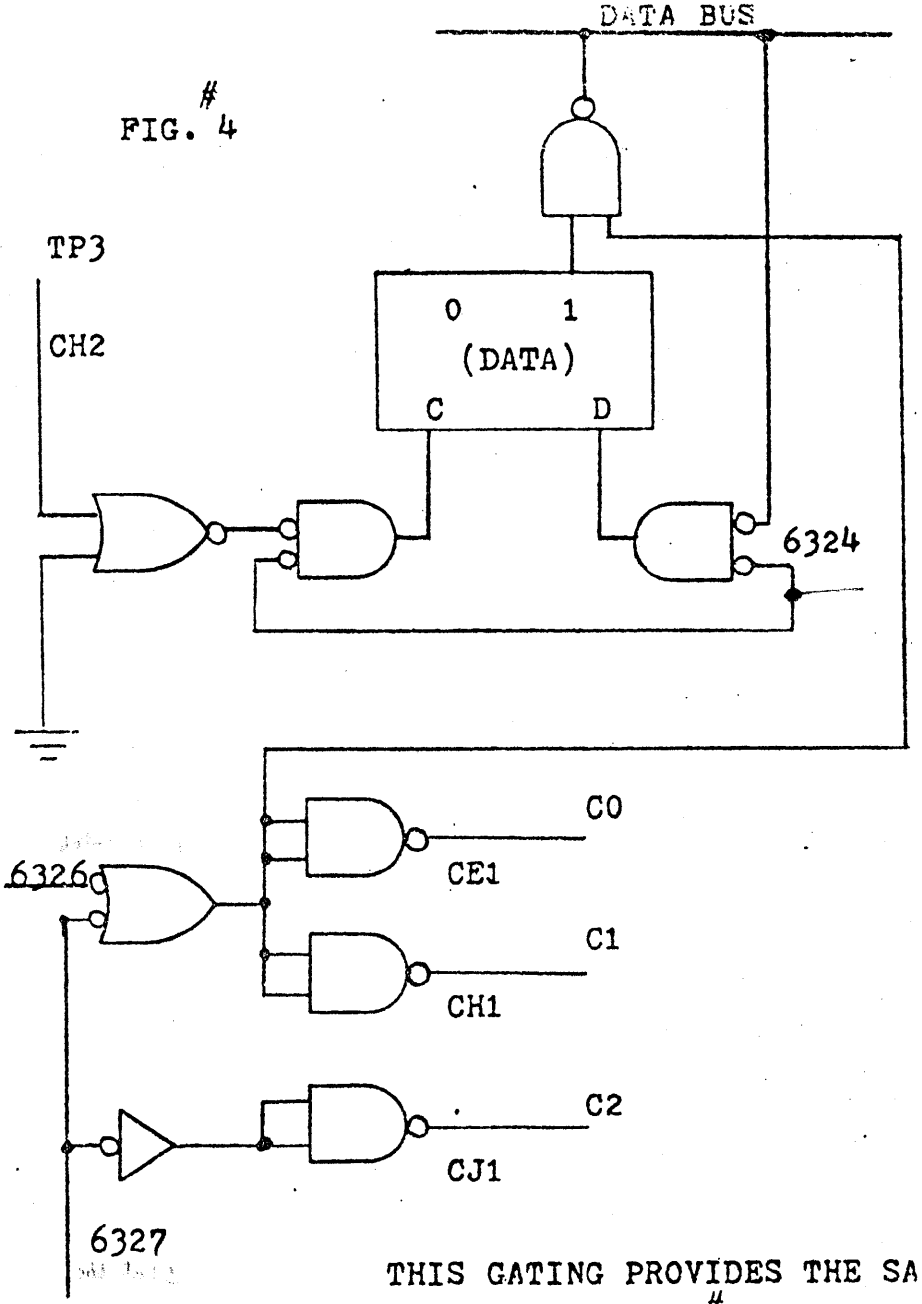
FIG. #3



THIS GATING HAS THE SAME FUNCTION AS THAT OF FIG.#2 PLUS THE DATA FROM THE DEVICE REGISTER CAN BE TANSFERED TO THE AC.

DATA ⟶ AC = Co=L, C1=L, C2=H

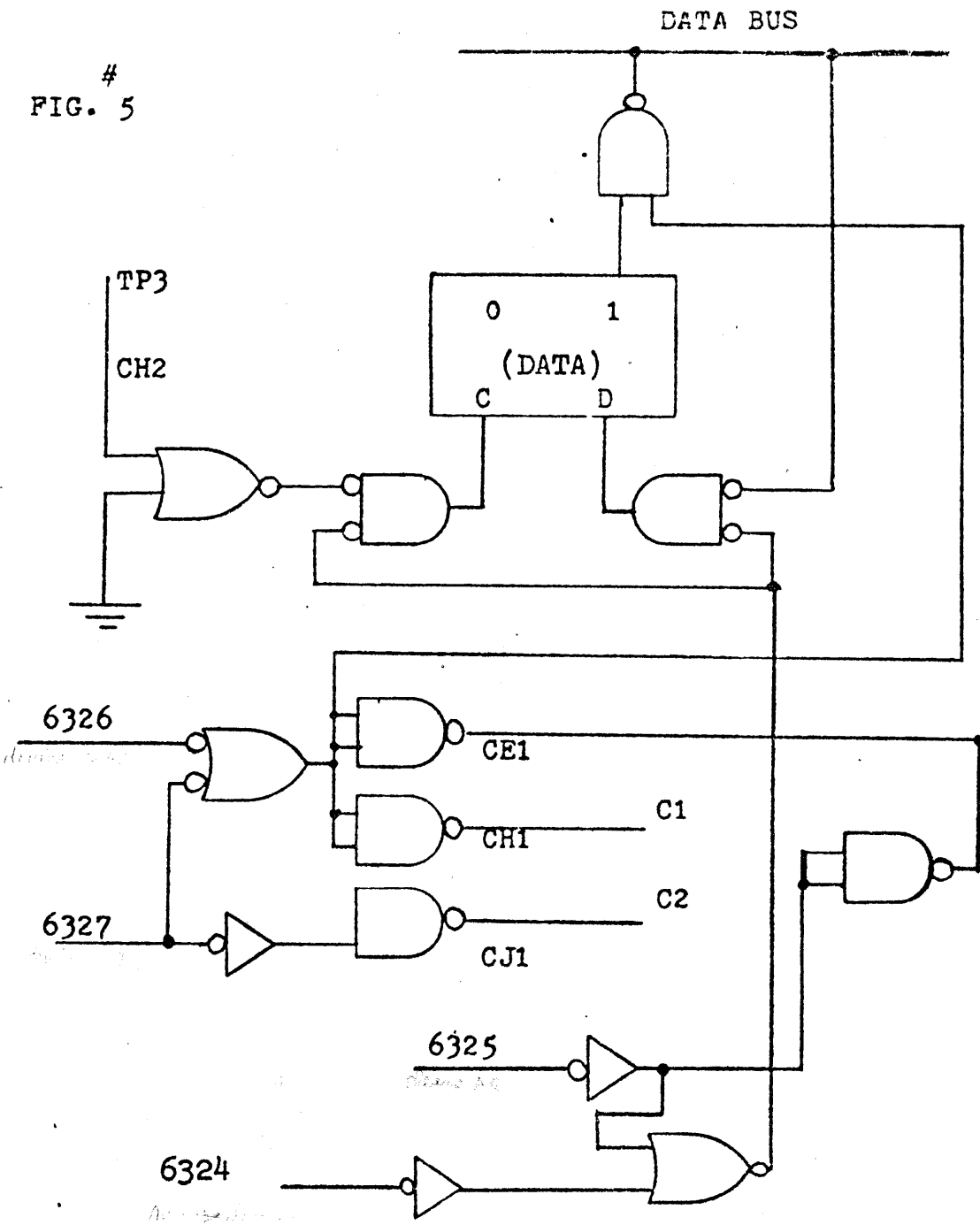
FIG. #4



THIS GATING PROVIDES THE SAME FUNCTIONS AS THOSE OF FIG. # 2 and 3 PLUS THE TRANSFERING OF DATA ⟶ PC.

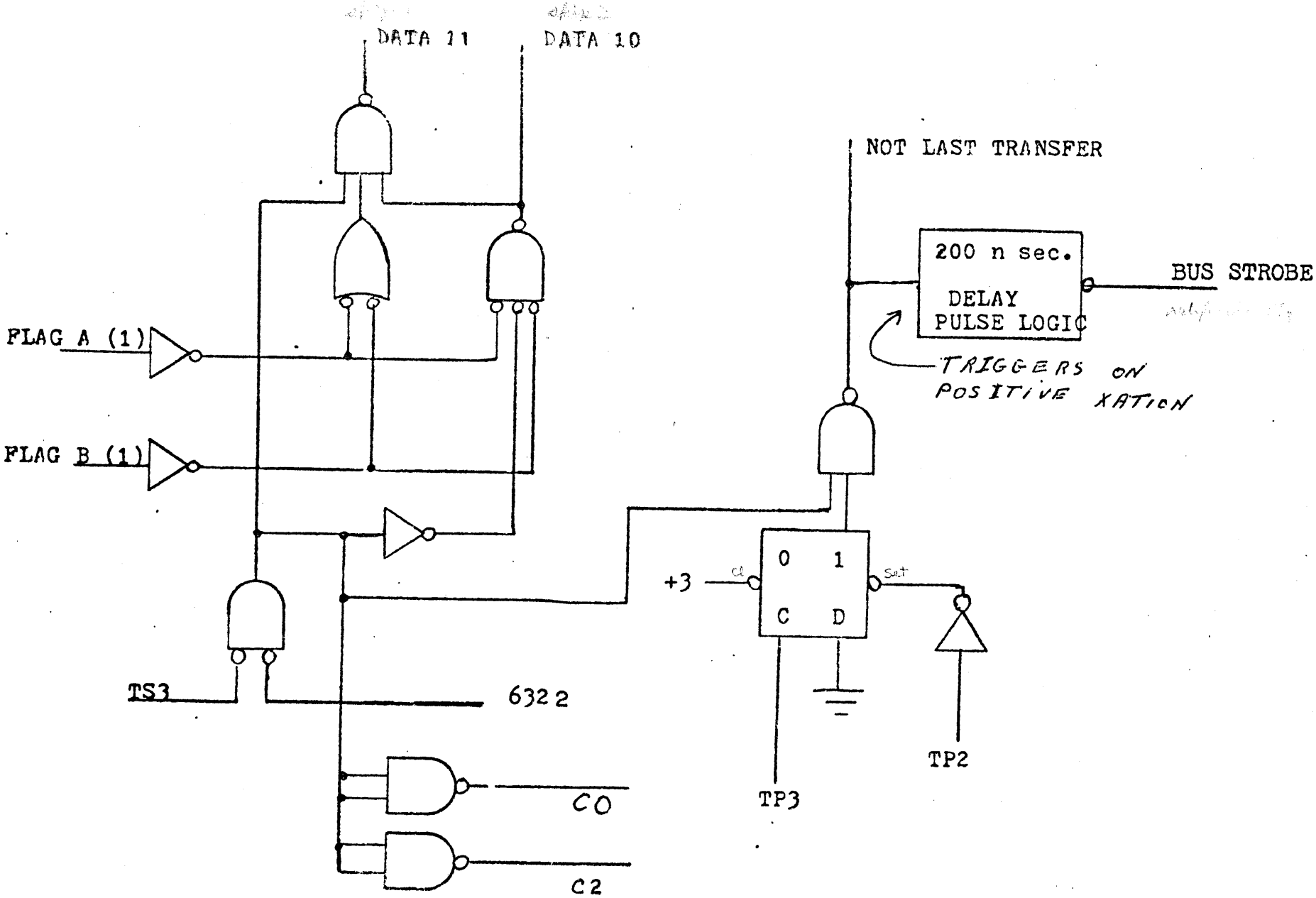
(DATA ⟶ PC= CO=L, C1=L, C2=H)

FIG. # 5



THIS GATING PROVIDES THE SAME FUNCTIONS AS THOSE OF FIG. # 2,3, and 4 PLUS THE 0 ⟶ AC AFTER THE AC HAS BEEN TRANSFERED TO THE DATA BUS.(0 ⟶ AC @ BUS STROBE

AC ⟶ DATA, 0 ⟶ AC
C0=L, C1=H, C2=H


DATA 11
DATA 10
NOT LAST TRANSFER
200 n sec.
DELAY
PULSE LOGIC
BUS STROBE
TRIGGERS ON
POSITIVE XATION
FLAG A (1)
FLAG B (1)
+3
0
1
C
D
TS3
6322
CO
C2
TP3
TP2

1. BEFORE THE CPU IS ABLE TO ACKNOWLEDGE ANY INTERRUPT REQUESTS FROM DEVICES,THE INTERRUPT SYSTEM MUST BE ENABLED. THE FOLLOWING IS A PROCEDURE IN ORDER TO ENABLE THE INTERRUPT SYSTEM.
   a) AN ION INSTRUCTION,6001,MUST BE PERFORMED. THIS INSTRUCTION WILL "TURN ON" THE INTERRUPT SYSTEM ONLY HALF WAY.
   b) ANY INSTRUCTION THAT FOLLOWS THE 6001 WILL FULLY ENABLE THE INTERRUPT SYSTEM.

2. WHEN THE INTERRUPT SYSTEM HAS BEEN ENABLED, THE CPU IS CAPABLE OF ACKNOWLEDGING INTERRUPT REQUESTS. THE FOLLOWING IS A GENERAL DESCRIPTION OF ACKNOWLEDGING AN INTERRUPT REQUEST:
   a) THE CPU WILL ACKNOWLEDGE AN INTERRUPT REQUEST AT TP3 TIME OF ANY MAJOR STATE PROVIDING THAT FETCH IS THE NEXT MAJOR STATE TO OCCUR. BY ACKNOWLEDGING THE INTERRUPT REQUEST THE NORMAL OCCURENCES OF TS4 ARE DISABLED. INSTEAD OF PERFORMING PC---MA (OR PC + 1----MA FOR A SKIP CONDITION) 0's ARE FORCED TO THE MA. THEREFORE; THE NEXT ABSOLUTE ADDRESS TO BE REFERENCED IS 0000. ALONG WITH FORCING THE CPU TO GO TO ADDRESS 0000, THE IR IS FORCED TO DECODE A JMS INSTUCTION.

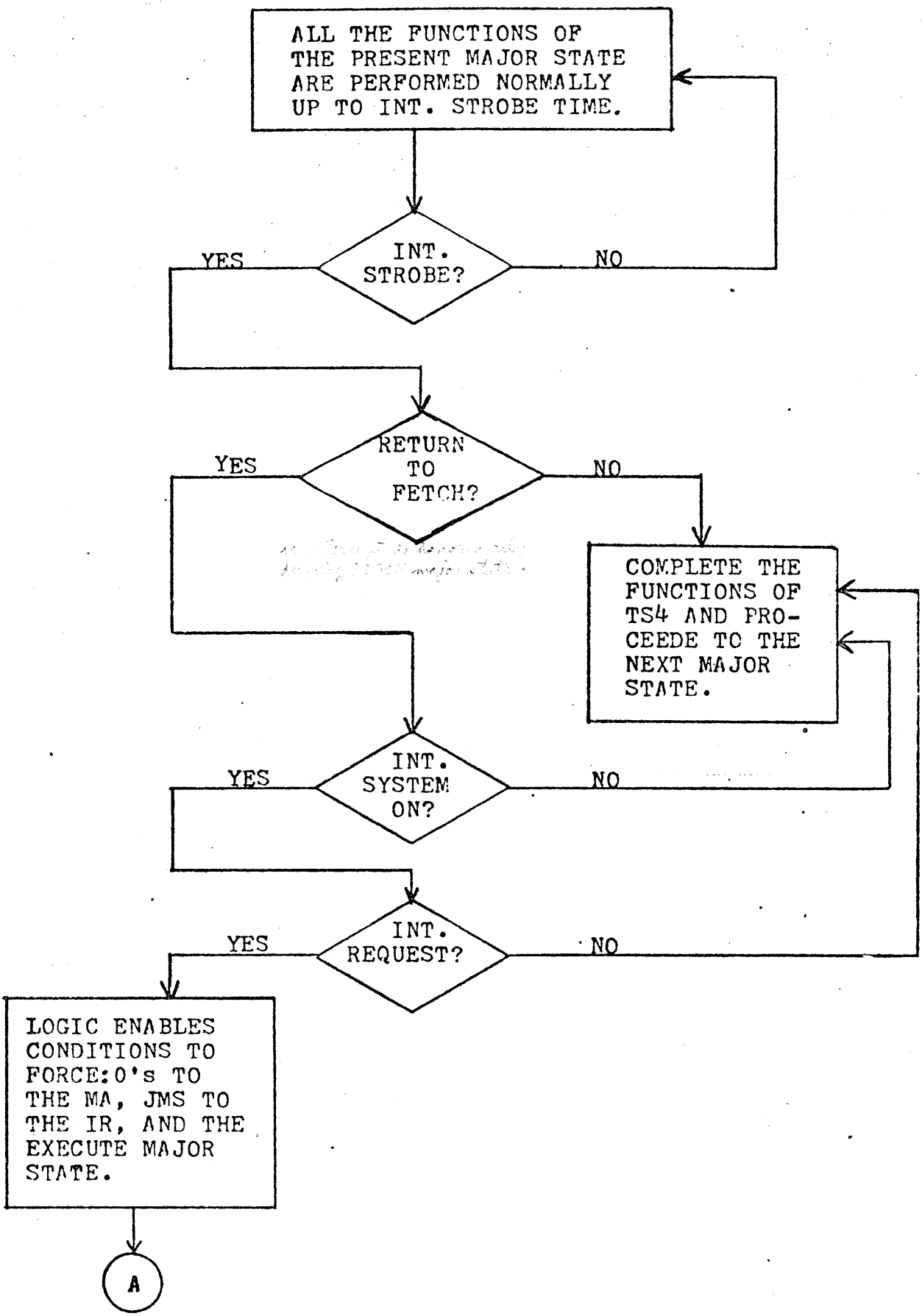
NOW THE CPU IS FORCED TO PERFORM A JMS INSTRUCTION AT ADDRESS 0000. SINCE THE CPU's IR IS FORCED TO DECODE A JMS AND ABSOLUTE ADDRESS 0000 WAS FORCED TO THE MA, THE CPU CAN ELIMINATE THE NEED FOR A FETCH CYCLE. THE CONTROLING LOGIC FOR CHANGING MAJOR STATES IS FORCED TO PRODUCE THE CONDITIONS THAT WILL ALLOW THE MAJOR STATE OF EXECUTE TO OCCUR NEXT. THE VALUE OF THE PC (WHICH IS THE NEXT INSTRUCTION'S ADDRESS FOR THE MAIN PROGRAM) WILL BE STORED IN ADDRESS 0000. ADDRESS 0000 IS NOW THE ENTRANCE POINT FOR RETURNING TO THE MAIN PROGRAM. THE FIRST INSTRUCTION OF THE SUBROUTINE WILL BE PERFORMED AT ADDRESS 0001. AT THIS POINT AN INTERROGATION ROUTINE IS EXECUTED (SEE HANDOUT #2s).

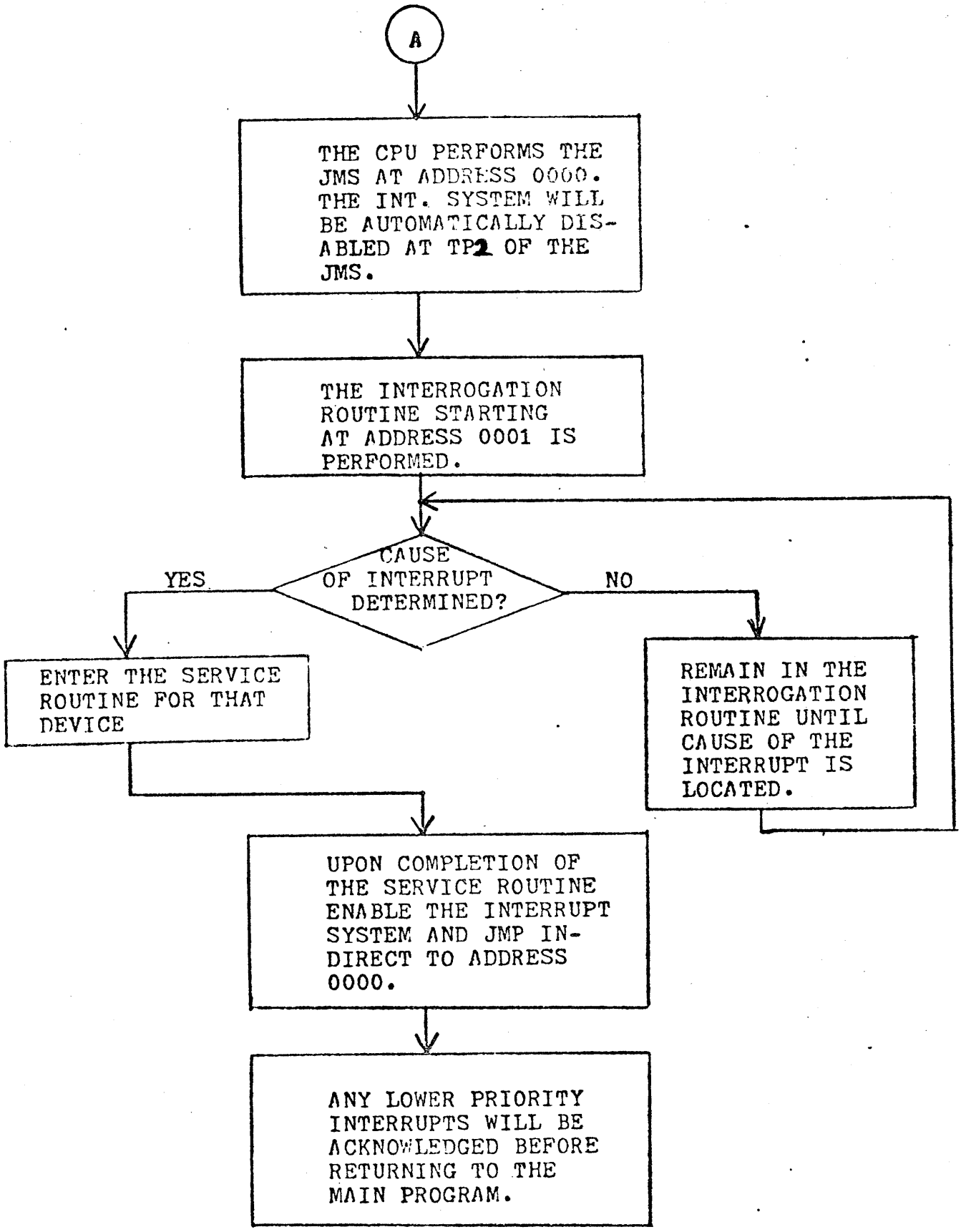
AS THE JMS IS EXECUTED,THE INT. SYS. IS AUTOMATICALY SHUT OFF.

3. UPON THE COMPLETION OF THE INTERROGATION AND THE SERVICE ROUTINES, PROCEDURES FOR RETURNIG TO THE MAIN PROGRAM ARE AS FOLLOWS:

   a) BECAUSE THE INTERRUPT SYSTEM WAS DISABLED DURING THE JMS, WHICH ALLOWED THE INTERRUPT BEING PROCESSED NOT TO BE INTERRUPTED BY ANOTHER INTERRUPT, THE INTERRUPT SYSTEM SHOULD BE ENABLED TO ACKNOWLEDGE ANY LOWER PRIORITY INTERRUPTS BEFORE RETURNING TO THE MAIN PROGRAM. THIS IS ACCOMPLISHED BY PERFORMING AN ION

INSTRUCTION AT THE NEXT TO THE LAST ADDRESS OF THE SERVICE ROUTINE PROGRAM. THE ION INSTRUCTION IS THEN FOLLOWED BY THE JMP INDIRECT TO ADDRESS 0000. IF ANY LOWER PRIORITY INTERRUPTS ARE PRESENT AT THIS TIME, THEY WILL BE ACKNOWLEDGED (ACCORDING TO PRIORITY) AND THE ORIGINAL CONTENTS OF ADDRESS 0000, WHICH HOLDS THE PC OF THE FIRST INTERRUPT WILL NOT BE CHANGED.


ALL THE FUNCTIONS OF THE PRESENT MAJOR STATE ARE PERFORMED NORMALLY UP TO INT. STROBE TIME.
INT. STROBE?
YES
NO
RETURN TO FETCH?
YES
NO
COMPLETE THE FUNCTIONS OF TS4 AND PROCEEDE TO THE NEXT MAJOR STATE.
INT. SYSTEM ON?
YES
NO
INT. REQUEST?
YES
NO
LOGIC ENABLES CONDITIONS TO FORCE:0's TO THE MA, JMS TO THE IR, AND THE EXECUTE MAJOR STATE.
A


A
THE CPU PERFORMS THE JMS AT ADDRESS 0000. THE INT. SYSTEM WILL BE AUTOMATICALLY DISABLED AT TP2 OF THE JMS.
THE INTERROGATION ROUTINE STARTING AT ADDRESS 0001 IS PERFORMED.
CAUSE OF INTERRUPT DETERMINED?
YES
NO
ENTER THE SERVICE ROUTINE FOR THAT DEVICE
REMAIN IN THE INTERROGATION ROUTINE UNTIL CAUSE OF THE INTERRUPT IS LOCATED.
UPON COMPLETION OF THE SERVICE ROUTINE ENABLE THE INTERRUPT SYSTEM AND JMP INDIRECT TO ADDRESS 0000.
ANY LOWER PRIORITY INTERRUPTS WILL BE ACKNOWLEDGED BEFORE RETURNING TO THE MAIN PROGRAM.

INTERROGATION ROUTINE

EXAMPLE PROGRAM

```
0000/ PC (XXXX)
0001/ DCA 3050
0002/ GTF 6004
0003/ DCA 3051
0004/ JMP I 5405
0005/ 7000

7000/ 6XXX
7001/ SKP 7410
7002/ JMP 5250
7003/ 6XXX
7004/ SKP 7410
7005/ JMP 5300
7006/ 6XXX
7007/ SKP 7410
7010/ JMP 5350
```

DESCRIPTION OF BASIC INTERROGATION ROUTINE

THE ABOVE PROGRAM IS AN EXAMPLE OF A BASIC INTERROGATION ROUTINE. IN ADRESS 0000 THE PC (WHICH IS THE ADDRESS OF THE NEXT INSTRUCTION FOR THE MAIN PROGRAM) IS STORED FROM THE FORCED JMS OF THE INTERRUPT SYSTEM. THE DCA 3050 LOCATED IN ADDRESS 0001 IS USED TO STORE THE AC VALUE IN CASE THE DEVICE CAUSING THE INTERRUPT TRANSFERS INFORMATION TO THE AC. THE CONTENTS OF ADDRESS 0002, A GTF, IS USED TO BRING THE VALUE OF THE LINK INTO THE AC. THE GTF IS FOLLOWED BY A DCA 3051 , IN LOCATION 0003, SO THAT THE VALUE OF THE LINK MAY BE STORED IN MEMORY. THE JMP I LOCATED IN ADDRESS 0004 IS USED TO EXIT PAGE "0". THE REASON FOR EXITING PAGE"0" IS THAT IT IS USED FOR AUTO INDEX PURPOSES AS WELL AS DIRECT ACCESS FROM ALL PAGES.

THE JMP I TAKES THE PROGRAM TO ADDRESS 7000 WHERE THE INTERROGATION ROUTINE BEGINS. THE FIRST PART OF THE PROGRAM WAS WRITTEN FOR "HOUSE CLEANING" PURPOSES. THE INSTRUCTION LOCATED IN ADDRESS 7000, LISTED AS 6XXX, WILL CHECK INTERRUPT FLAG OF THE FIRST DEVICE (PRIORITY 0) FOR A SKIP CONDITION. IF THAT DEVICE'S INTERRUPT FLAG WAS SET, A SKIP CONDITION WOULD RESULT CAUSING THE PROGRAM TO REFERENCE ADDRESS 7002 INSTEAD OF 7001. THE INSTRUCTION AT ADDRESS 7002, AJMP 5250, WILL ENTER THE SERVICE ROUTINE FOR DEVICE PRIORITY 0.

EXAMPLE PROGRAM

IF DEVICE 0'S INTERRUPT FLAG WAS NOT SET, THE PROGRAM WOULD REFERENCE ADDRESS 7001. AN UNCONDITIONAL SKIP, 7410, LOCATED AT ADDRESS 7001 WILL CAUSE THE PROGRAM TO SKIP OVER THE JMP 5250. THUS THE ENTRANCE POINT FOR THE SERVICE ROUTINE OF DEVICE 0 IS SKIPPED OVER.

THE SAME SEQUENCE OF CHECKING THE INTERRUPT FLAGS APPLIES TO THE OTHER TWO DEVICES.

THE NEXT PORTION OF THE EXAMPLE PROGRAM DEMONSTRATES HOW TO ENTER THE MAIN PROGRAM AND ALSO ENABLE THE INTERRUPT SYSTEM. THIS PORTION OF THE PROGRAM WILL BE EXECUTED AFTER THE MAIN PORTION OF THE SERVICE ROUTINE HAS BEEN COMPLETED.

(AS AN EXAMPLE, DEVICE PRIORITY 1 WILL BE USED)

```
7124/ CLA  7200
7125/ TAD  1051
7126/ RTF  6005
7127/ CLA  7200
7130/ TAD  1050
7131/ ION  6001
7132/ JMP  5400
```

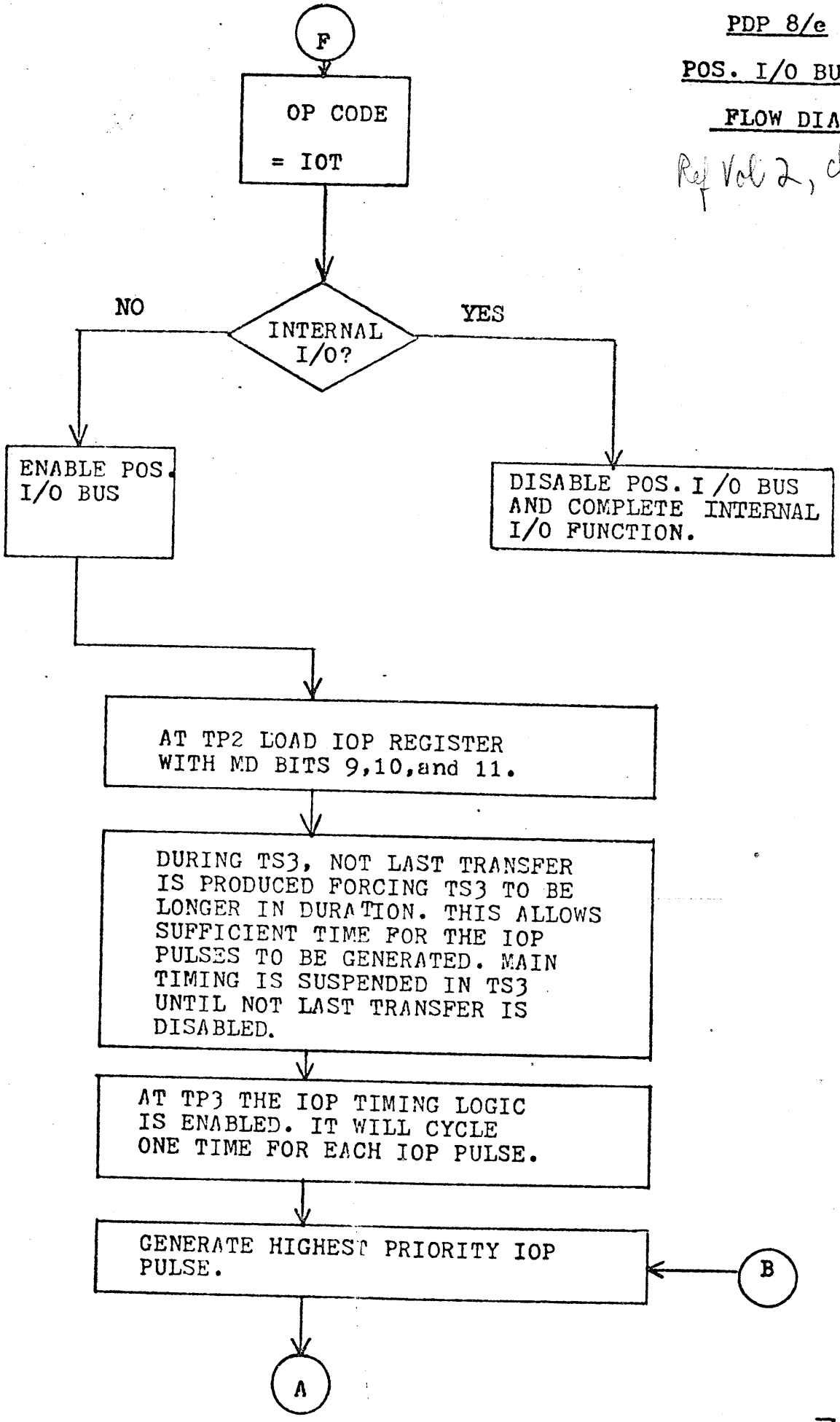
AT LOCATION 7124, A CLA CLEARS THE AC IN PREPARATION FOR RESTORING THE VALUES THAT WERE STORED IN MEMORY BY THE "HOUSE CLEANING" PORTION OF THE PROGRAM. THE TAD INSTRUCTION, 1051, LOCATED AT ADDRESS 7125 WILL BRING THE ORIGINAL CONDITION OF THE LINK, PRIOR TO THE INTERRUPT, INTO THE AC. FOLLOWING THE TAD INSTRUCTION IS A RTF INSTRUCTION. THIS INSTRUCTION WILL RETURN THE ORIGINAL CONTENTS OF THE LINK, WHICH IS NOW IN THE AC, BACK INTO THE LINK. THE NEXT LOCATION, 7127, HOLDS A CLA INSRUCTION. THIS WILL BE USED TO CLEAR THE AC IN PREPARATION FOR RESTORING THE ORIGINAL AC, PRIOR TO THE INTERRUPT. AFTER THE AC HAS BEEN CLEARED BY THE CLA, THE TAD INSTRUCTION, 1050, AT ADDRESS 7130 WILL BRING THE ORIGINAL CONTENTS OF THE AC, PRIOR TO THE INTERRUPT, BACK INTO THE AC. THE 6001 INSTRUCTION, ION, WILL PARTIALLY ENABLE THE INTERRUPT SYSTEM. THE LAST INSTRUCTION, 5400, WILL FULLY ENABLE THE INTERRUPT SYSTEM AND ALLOW THE PROGRAM TO RETURN TO ADDRESS 0000. FROM THIS POINT THE PROGRAM CAN ENTER THE MAIN PROGRAM OR ACKNOWLEDGE ANY LOWER PRIORITY INTERRUPTS THAT MAY BE PRESENT.

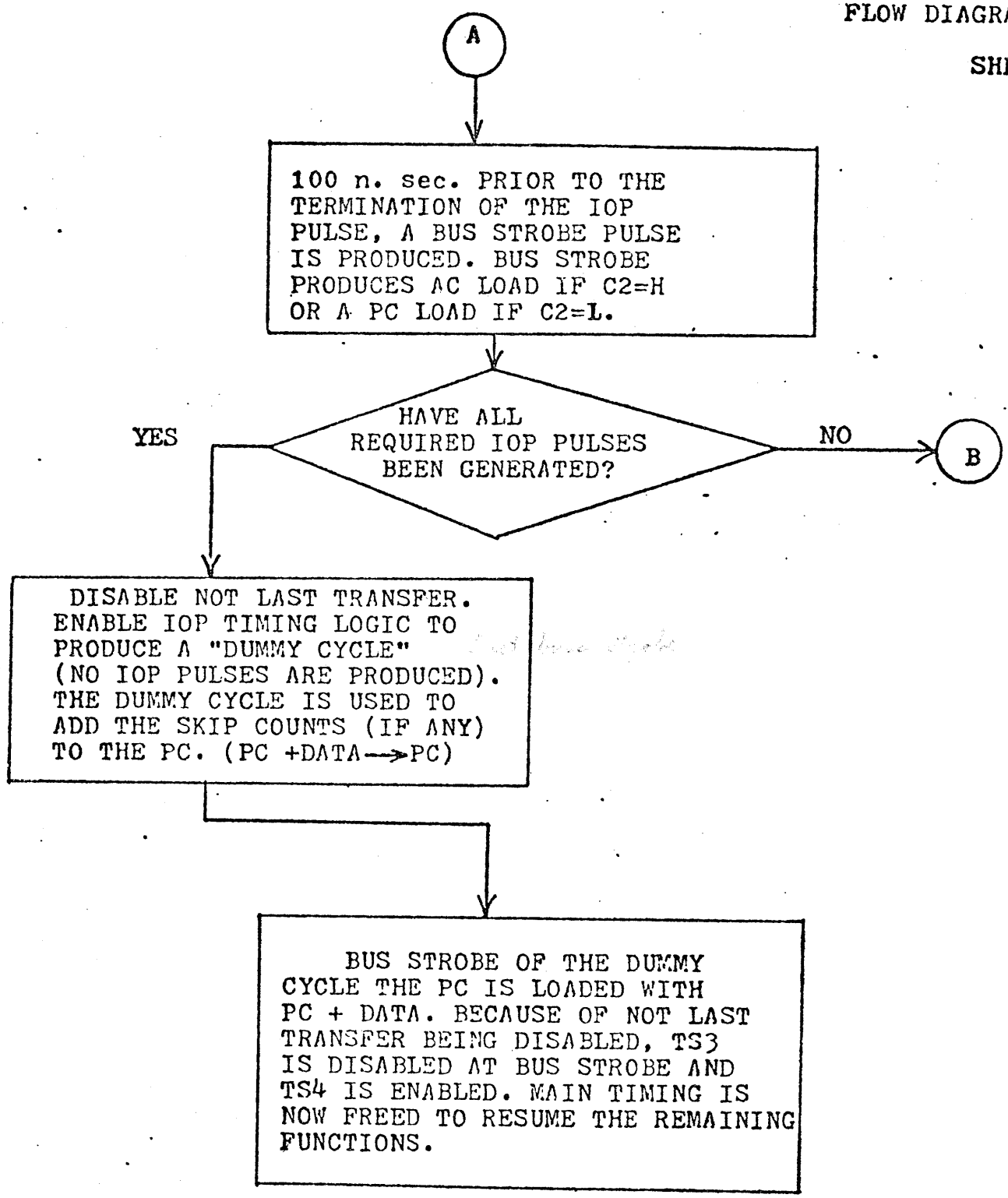
PDP 8/e

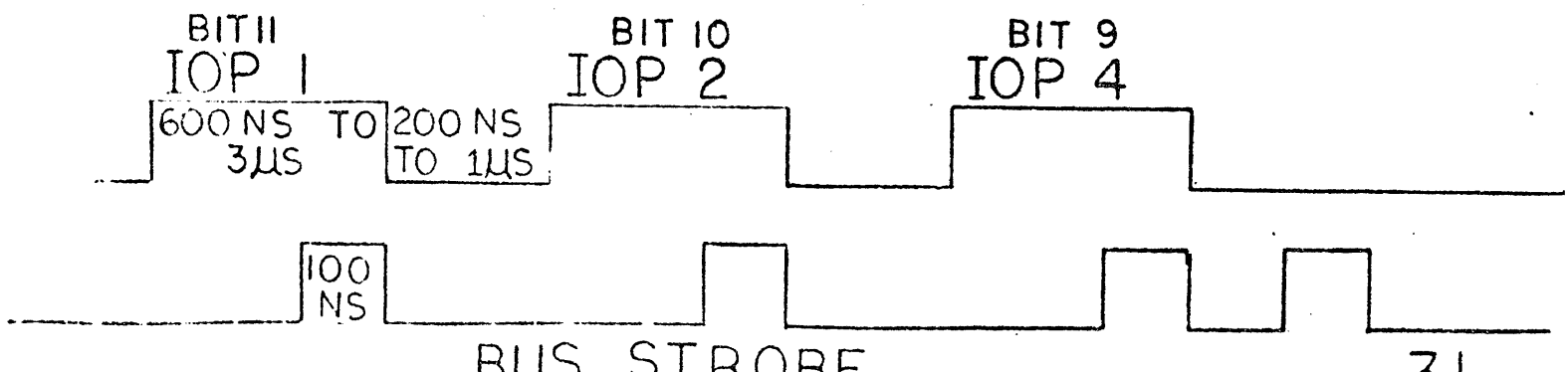
POS. I/O BUS

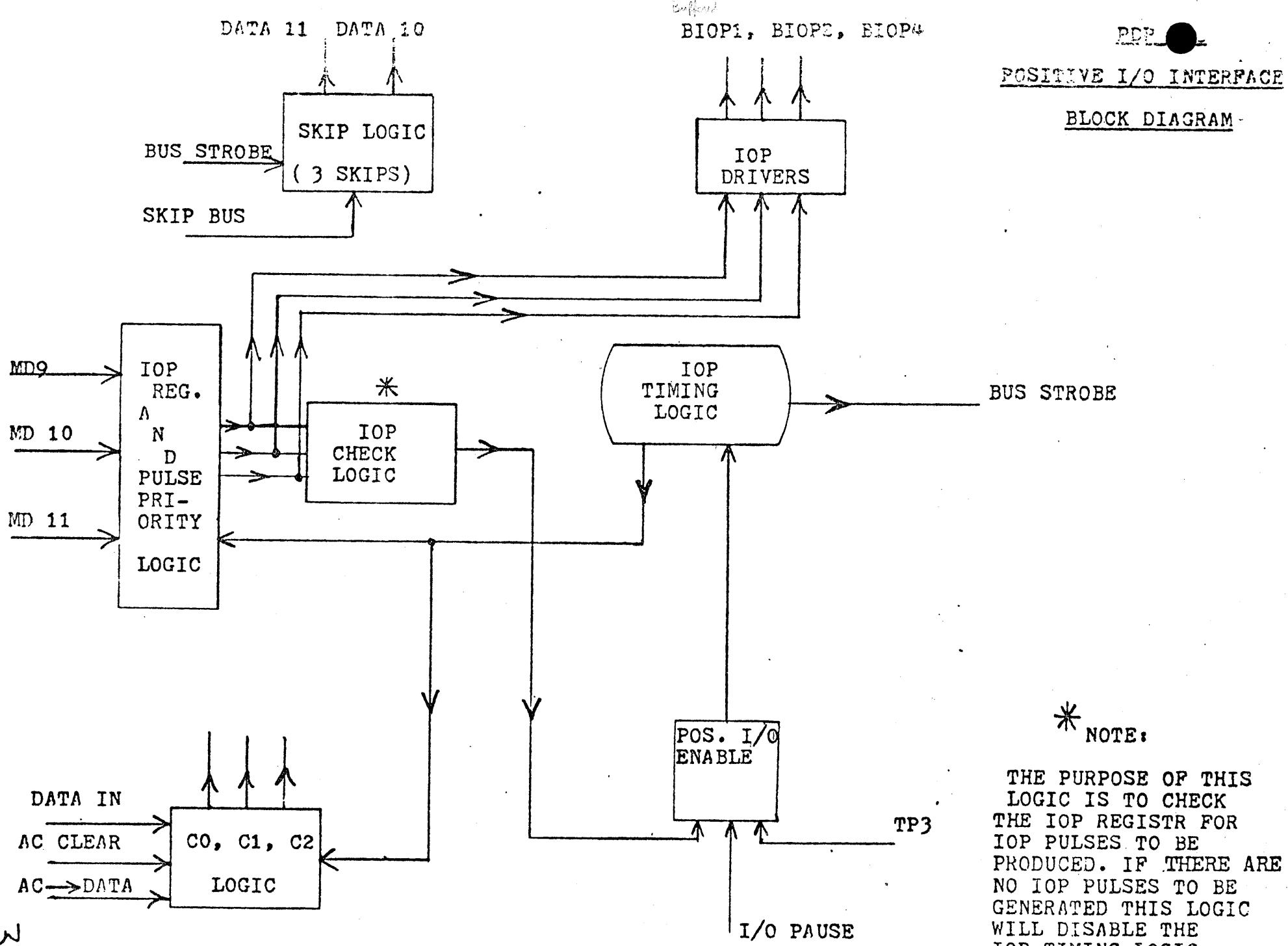
FLOW DIAGRAM

Ref Vol 2, ch 9

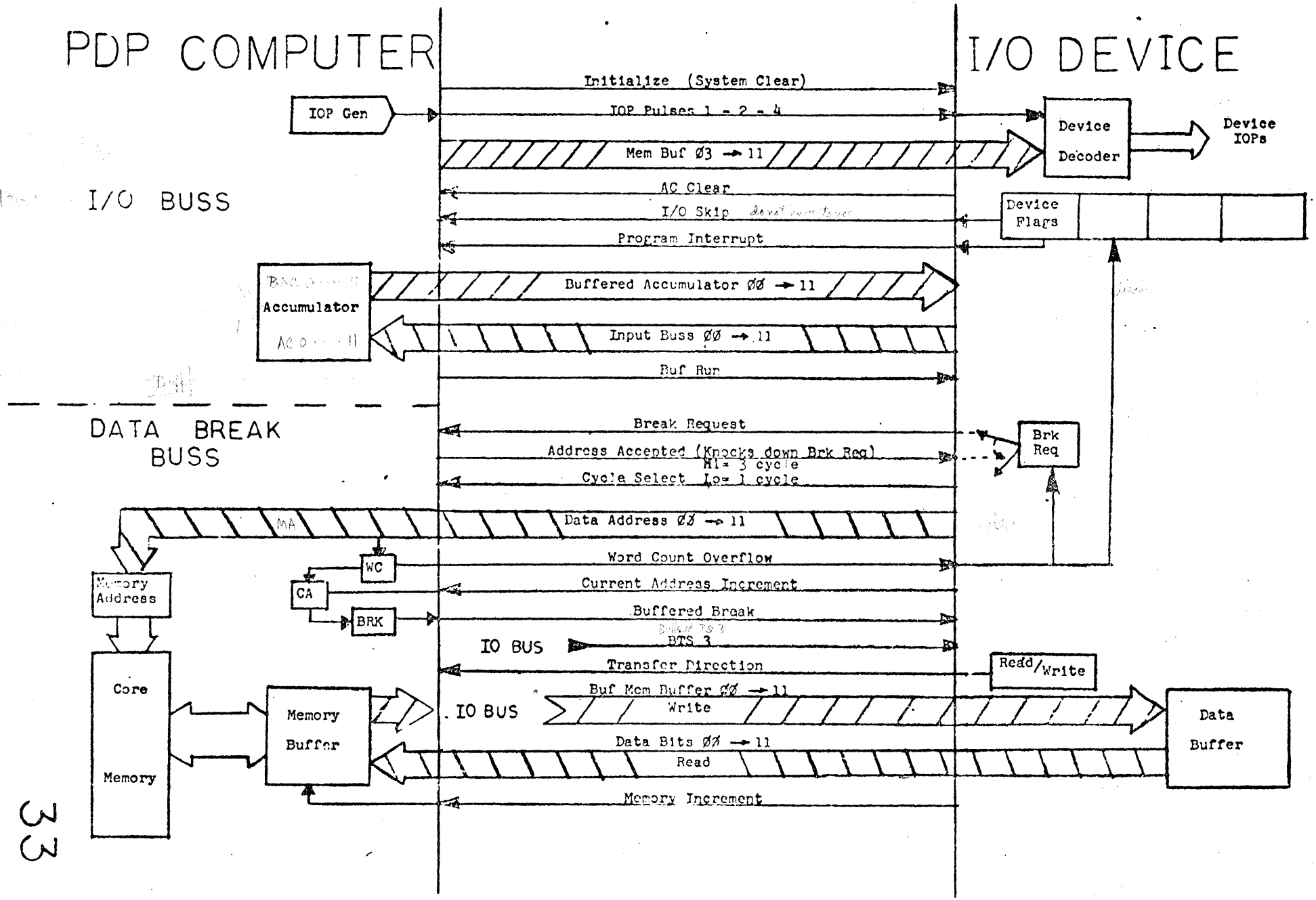



A
100 n. sec. PRIOR TO THE TERMINATION OF THE IOP PULSE, A BUS STROBE PULSE IS PRODUCED. BUS STROBE PRODUCES AC LOAD IF C2=H OR A PC LOAD IF C2=L.
HAVE ALL REQUIRED IOP PULSES BEEN GENERATED?
YES
NO
B
DISABLE NOT LAST TRANSFER. ENABLE IOP TIMING LOGIC TO PRODUCE A "DUMMY CYCLE" (NO IOP PULSES ARE PRODUCED). THE DUMMY CYCLE IS USED TO ADD THE SKIP COUNTS (IF ANY) TO THE PC. (PC +DATA→PC)
BUS STROBE OF THE DUMMY CYCLE THE PC IS LOADED WITH PC + DATA. BECAUSE OF NOT LAST TRANSFER BEING DISABLED, TS3 IS DISABLED AT BUS STROBE AND TS4 IS ENABLED. MAIN TIMING IS NOW FREED TO RESUME THE REMAINING FUNCTIONS.


BIT 11
IOP 1
600 NS TO 3μS
200 NS TO 1μS
BIT 10
IOP 2
BIT 9
IOP 4
100 NS
BUS STROBE



IOP pulse generating


PDP COMPUTER
I/O DEVICE
Initialize (System Clear)
IOP Gen
IOP Pulses 1 - 2 - 4
Device Decoder
Device IOPs
Mem Buf Ø3 → 11
I/O BUSS
AC Clear
I/O Skip
Program Interrupt
Device Flags
Accumulator
Buffered Accumulator ØØ → 11
Input Buss ØØ → 11
Buf Run
DATA BREAK BUSS
Break Request
Brk Req
Address Accepted (Knocks down Brk Req)
Hi = 3 cycle
Cycle Select Lo = 1 cycle
MA
Data Address ØØ → 11
WC
Word Count Overflow
Memory Address
CA
Current Address Increment
BRK
Buffered Break
IO BUS
BTS 3
Transfer Direction
Read/Write
Core Memory
Memory Buffer
Buf Mem Buffer ØØ → 11
Write
Data Buffer
Data Bits ØØ → 11
Read
Memory Increment

STATE FUNCTIONS
OF
3 CYCLE BREAK

(logic number only)

WORD COUNT

THE PURPOSE OF THIS STATE IS TO INCREMENT A MEMORY WORD LOCATED AT A KNOWN MEMORY ADDRESS. THIS KNOWN MEMORY ADDRESS IS SPECIFIED BY THE 3 CYCLE DEVICE. THE MEMORY WORD LOCATED AT THIS ADDRESS IS THE 2'S COMPLIMENTED VALUE OF THE NUMBER TRANSFERS TO BE PERFORMED.

IF THE MEMORY WORD IS INCREMENTED TO 0000, WORD COUNT OVERFLOW IS SENT TO THE DEVICE TO INHIBIT ANY MORE BREAK REQ.

CURRENT ADDRESS

THE PURPOSE OF THIS STATE IS TO SPECIFY AN ADDRESS AT WHICH THE TRANSFER TAKES PLACE.

THIS FUNCTION CAN BE DONE 2 WAYS. 1) A KNOWN VALUE, LOCATED AT THE INCREMENTED VALUE OF THE MEMORY ADDRESS OF THE WORD COUNT STATE WILL BE INCREMENTED EACH TIME THIS STATE IS REFERENCED. THIS INCREMENTED VALUE IS USED FOR THE MA OF THE BREAK STATE, THUS ALLOWING SEQUENTIAL DATA TRANSFERS FROM MEMORY. 2) THE KNOWN VALUE NEED NOT BE INCREMENTED ALLOWING DATA TRANSFERS AT A CONSTANT ADDRESS. HOWEVER; THIS METHOD REQUIRES A BACKGROUND PROGRAM.

BREAK

THE PURPOSE OF THIS STATE IS TO PERFORM THE TRANSFERS. THERE ARE 4 FUNCTIONS WHICH MAY BE ACCOMPLISHED 1) DATA MAY BE TRANSFERED FROM THE DEVICE TO THE CPU. 2) DATA FROM THE CPU MAY BE TRANSFERED TO THE DEVICE. 3) DATA FROM THE DEVICE MAY BE ADDED WITH A MEMORY WORD FROM THE CPU. 4) A MEMORY WORD OF THE CPU MAY BE INCREMENTED.

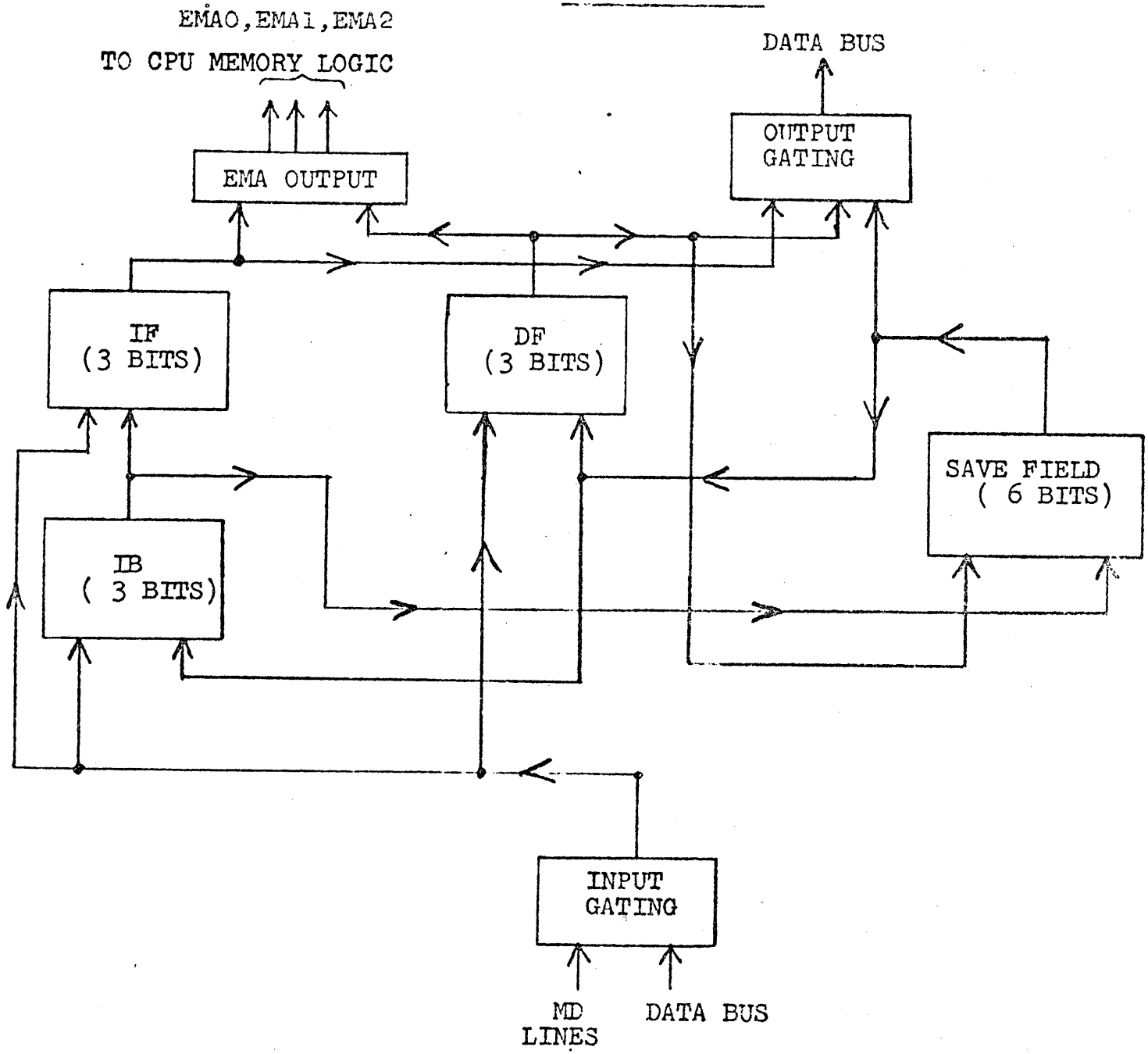
ALL 4 FUNCTIONS ARE PERFORMED AT A MEMORY ADDRESS SPECIFIED BY THE CURRENT ADDRESS STATE.

OPTIONS FOR BREAK MAJOR STATE

WRITE (DATA OUT) DI (0) INC (0)	(MEM) ⟶ MDL ⟶ PERIPH.	CONTENTS OF MEMORY SENT TO PERIPH AND WRITTEN BACK INTO MEMORY
READ (DATA IN) DI (1) INC (0)	DATA LINES ⟶ DATA BUS ⟶ MB MDL ⟶ (MEM)	DATA FROM PERIPH WRITTEN INTO MEMORY
INC (MEM) DI (0) INC (1)	(MEM) ⟶ MDL DATA BUS = 1 DATA BUS + MDL ⟶ MB ⟶ (MEM)	CONTENTS OF MEMORY +1 WRITTEN BACK INTO MEMORY (USED AS A COUNTER)
DATA ADDED TO DI (1) INC (1)	(MEM) ⟶ MDL DATA LINES ⟶ DB DATA BUS + MDL ⟶ MB ⟶ (MEM)	BREAK DATA IS ADDED TO THE CONTENTS OF MEMORY AND WRITTEN BACK INTO MEMORY

EXTENDED [redacted] CONTROL

BLOCK DIAGRAM



FUNCTIONS OF INSTUCTION FIELD AND DATA FIELD REGISTERS

THE FOLLOWING INFORMATION DESCRIBES THE RELATIONSHIP BETWEEN THE CPU MAJOR STATES AND EXTENDED MEMORY FUNCTIONS.

KEY FUNCTIONS: ALTHOUGH THE CPU IS IN NO MAJOR STATE, EXTENDED MEMORY MUST STILL CONTROL WHAT FIELD WILL BE REFERENCED DURING KEY FUNCTIONS. BECAUSE OF THIS, THE PROGRAMMER HAS A CHOICE OF ANY FIELD (LIMITED, OF COURSE, TO THE AMOUNT OF CORE ON THE SYSTEM) TO MANUALLY DEPOSIT OR EXAMINE DATA. THE CONTENTS OF THE IF REGISTER DETERMINES WHICH FIELD WILL BE REFERENCED DURING KEY FUNCTIONS..

FETCH: THE CONTENTS OF THE IF REGISTER WILL DETERMINE WHAT FIELD WILL BE REFERENCED TO OBTAIN INSTRUCTIONS. THIS CONDITION IS TRUE FOR ANY FETCH MAJOR STATE.

DEFER: THE CONTENTS OF THE IF REGISTER WILL DETERMINE FROM WHAT FIELD THE POINTER ADDRESS WILL BE OBTAINED.* HOWEVER; AT TP4 TIME OF DEFER EITHER THE IF OR THE DF REGISTER CAN BE THE REGISTER THAT WILL SPECIFY THE FIELD

AT TP4 TIME OF EACH MAJOR STATE EXT. MEM. CONT. LOGIC DECIDES IF THE IF OR DF WILL SPECIFY THE NEXT FIELD TO REFERENCE.

INSTRUCTION FIELD AND DATA FIELD

REGISTERS

DEFER---------- (cont.) TO BE REFERENCED FOR THE EXECUTE MAJOR STATE. THIS IS DEPENDENT UPON WHAT TYPE OF INSTRUCTION IS BEING PERFORMED.

IF AN AND, TAD, ISZ, OR DCA IS BEING PERFORMED, THE CONTENTS OF THE DF REGISTER WILL DETERMINE WHICH FIELD WILL BE REFERENCED TO OBTAIN DATA FOR THE EXECUTE MAJOR STATE. IF THE CONTENTS OF THE DF ≠ IF THE PROGRAMMER NOT ONLY HAS THE CAPABILITY OF OBTAINING DATA FROM ANY MEMORY PAGE; HE ALSO HAS THE CAPABILITY OF REFERENCING A DIFFERENT MEMORY FIELD IN ORDER TO OBTAIN DATA. ON THE OTHER HAND; IF THE DF = IF, A NORMAL INDIRECT IS PERFORMED IN THE SAME FIELD THE INSTRUCTION AND THE POINTER WERE OBTAINED.

IF A JMS OR JMP IS BEING PERFORMED, THE DF REGISTER WILL NOT SPECIFY THE FIELD TO BE REFERENCED IN EXECUTE; THE IF REGISTER WILL.THE REASON FOR THIS IS THAT A JMP NEVER ENTERS THE EXECUTE STATE. IF THE DF REGISTER COULD SPECIFY WHICH FIELD TO REFERENCE DURING THE NEXT MAJOR

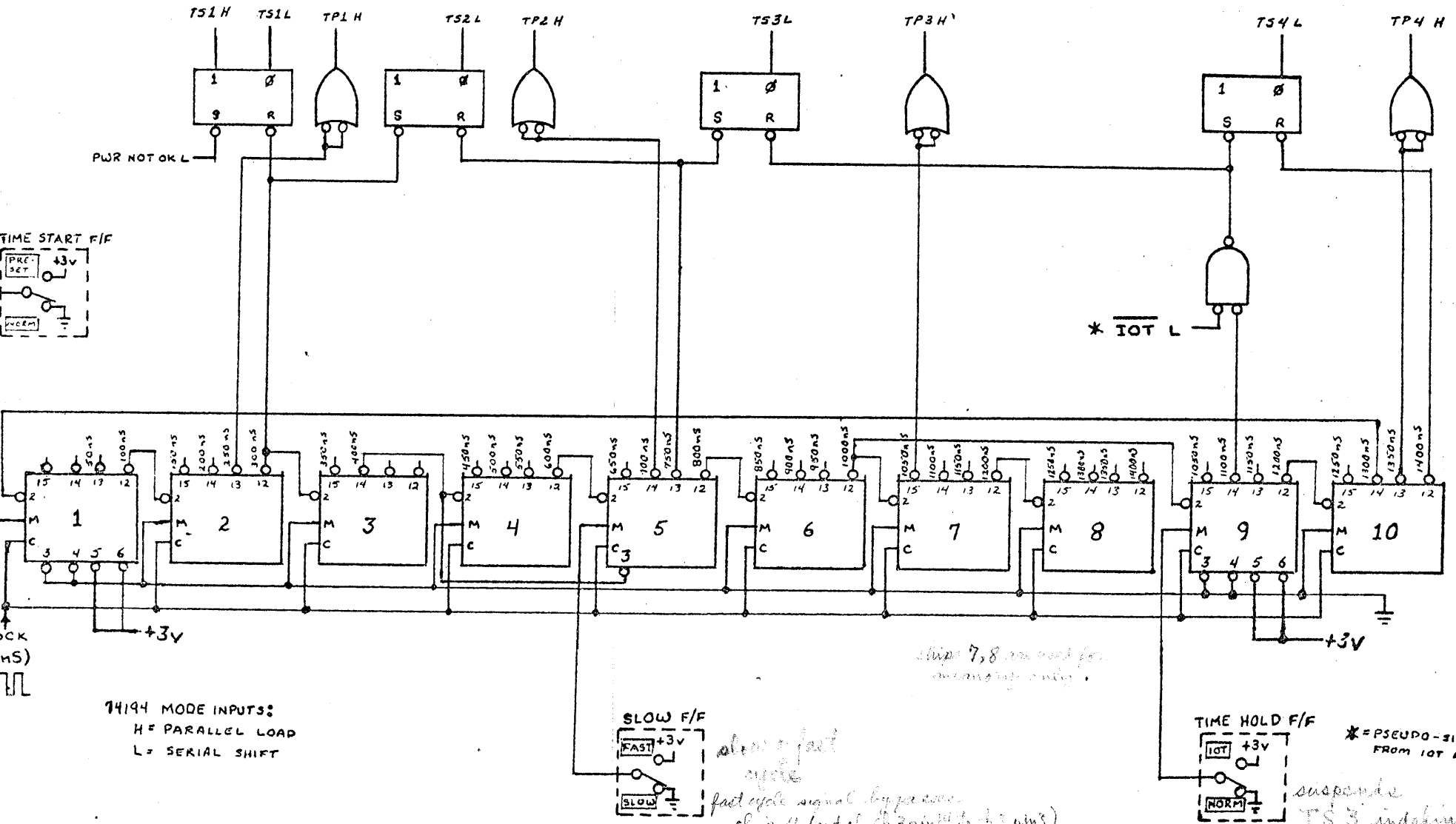
INSTRUCTION FIELD AND DATA FIELD

REGISTERS

DEFER (cont.) ---------- STATE, WHICH WOULD BE FETCH, THE NEXT INSTRUCTION WOULD BE OBTAINED FROM THE FIELD SPECIFIED BY THE DF REGISTER. THIS WOULD DEFEAT THE PURPOSE OF THE DF REGISTER AS IT IS USED ONLY TO OBTAIN DATA; NOT INSTRUCTIONS.

BECAUSE THE JMP INDIRECT CANNOT USE THE DF REGISTER TO SPECIFY A FIELD TO REFERENCE FOR THE EXECUTE MAJOR STATE, THE JMS INSTRUCTION IS FORCED TO DO THE SAME. IF THE JMS INSTRUCTION COULD USE THE DF REGISTER TO SPECIFY WHICH FIELD TO REFERENCE FOR THE EXECUTE MAJOR STATE, THE PC WOULD BE STORED IN THAT PARTICULAR FIELD. HOWEVER; UPON RETURNING FROM THE SUBROUTINE, WHICH REQUIRES THE USE OF A JMP INDIRECT TO THE ADDRESS WHERE THE PC IS LOCATED, THE FIELD WHERE THE PC IS STORED COULD NOT BE REFERENCED. WHY? A JMP INDIRECT CANNOT REFERENCE A FIELD SPECIFIED BY THE DF REGISTER.

EXECUTE ---------- THE EXECUTE MAJOR STATE CAN REFERENCE A FIELD SPECIFIED BY THE IF OR DF. (SEE ABOVE INFORMATION OF DEFER FOR EXPLANATION)



M8330 TIMING GENERATOR- SIMPLIFIED SCHEMATIC DIAGRAM

M830 (Alt) has different chip, changes pins.

not very good.

PDP-8/E INTERFACING COURSE

TRAINING NOTES

PRELIMINARY (day one only)

MARCH 27, 1973

ABOUT THIS MANUAL

This manual has been written to be used in conjunction with the PDP 8/e Family Interfacing Seminar, and as such should not be taken as the only reference for such information.

INDEX

DRAWINGS

PDP-8/e INTERFACING SEMINAR
DAILY BREAKDOWN

DAY 1

A. Block Diagram
B. Timing
C. IOT Instructions
D. Omnibus Signals

DAY 2

A. Internal Interfacing
B. External Interfacing

DAY 3

A. Data Break
B. Program Interrupt

VII. Data Break
- A. Purpose
- B. Theory of operation
  1. Block diagram
  2. Timing diagrams
  3. Interrelation to C.P.
  4. Sample usage

VIII. Program Interrupt
- A. Purpose
- B. Theory of operation
- C. Hardware implementation

PROCESSOR TIMING

Processor operations are broken down into cycles which are in turn broken down into time states and time pulses. Each cycle represents a reference to memory in which a word is removed from or put into memory, thus for each data word that an instruction requires from memory a new cycle is initiated. Consequently, there may be one, two, or three cycles for each instruction. Our main concern with timing is not the individual cycles, but rather the various subdivisions of the cycle. Each cycle is divided into four parts referred to as time periods. A time period consists of a rather long time state (approx. = 300ns.) and a short time pulse at its end (100 ns.). The four time periods are referred to as T1, T2, T3, and T4. (See timing diagram No. INT-8).

The time states in each cycle are used to enable data, that is they are used to determine what data is to be transferred. The time pulses are used to load registers, which is to say that they determine where the selected data is to go. Since the sequence for timing is a time state followed by a time pulse, this means that the processor first figures out what data is to be transferred and then were to put it. You will note that there are four time periods just as there are four major registers. Generally (though not exclusively) each time period is dedicated to a particular register, as follows:

1. T1 is used for modification of the PC.
2. T2 is used for modification of the MB.
3. T3 is used for modification of the AC.

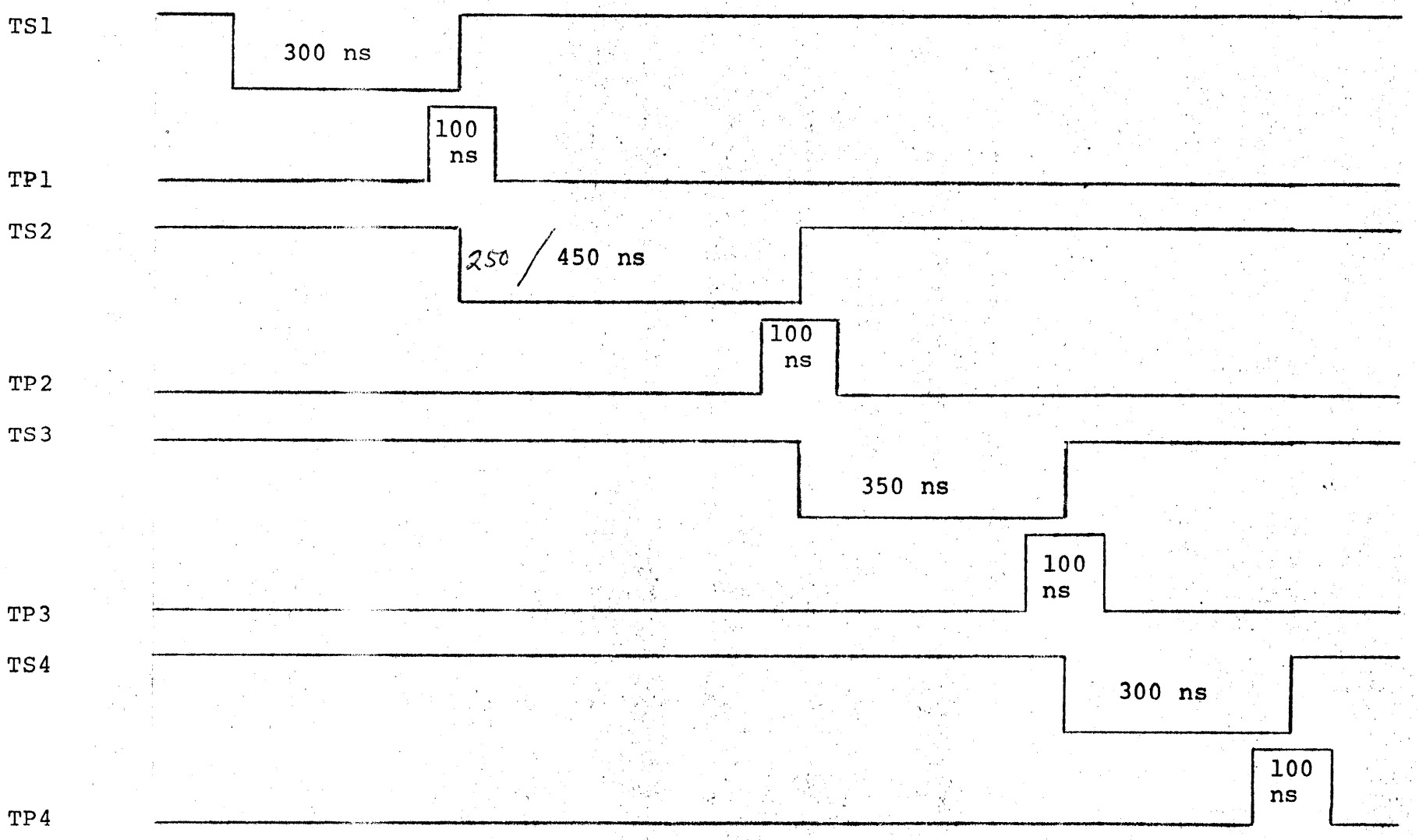
(note that since it is the AC to and from

which all data transfers take place for I/O operations, ~~that~~ these operations will have to take place during time 3, with time state 3 being used to enable data onto and off of the bus, and time pulse 3 being used to load the data into the AC or some peripheral buffer.)

4. T4 is used to modify the MA. (CPMA).

When the type of interface being used is of the internal variety it is up to the user to see to it that all operations with the bus are concluded within the confines of T3. While if the interface is external, the positive I/O bus option will stall the processor timing in order to allow sufficient time to communicate with a peripheral that might be located at the end of a considerable length of cable (up to 50ft. in the case of the positive I/O. For a diagram of timing as applicable to the Positive I/O see drawing No. INT-7. (missing)

CENTRAL PROCESSOR TIMING Int-8



The instructions used by the computer to communicate with the peripherial devices are called IOT instructions. These instructions take the form of 12 bit numbers and carry three pieces of information.

1. The first three bits from the OP CODE, which is a 6 to specify that the instruction is an IOT.
2. The next six bits carry the device code, that is, a number which specifies the device to be dealt with.
3. The last three bits of the instruction contain the function bits. These three bits specify to the device the function it is to perform. These last three bits can be interrpreted in one of two ways.
    1. If the device is an internal type interface the last three bits are converted from binary to octal to produce one of eight different commands.
    2. If the device uses an external type interface. The last three bits are picked up by the positive I/O interface which will in turn generate one, two, or three IOP pulses. The peripheral will then use these pulses to determine what function to perform.

It should be kept in mind that when the C.P. executes these instructions that all 12 bits of the instruction code are available to the user on the MD bus lines. It is from here that they are taken

to be used by the device to select a device code as well as to determine the function. It is not necessary for the device to consider the first three bits of the instruction, since the C.P. will generate the signal I/O PAUSE whenever it detects an IOT instruction. It will be shown later that the signal I/O PAUSE must be used by the device interface to gate all its data to and from the C.P.

OMNIBUS SIGNALS

A complete list of all signals on the omnibus is presented in drawing INT-9. The signals shown with an * are the primary signals used in interfacing and are discussed in the section entitled "Signal Description". All signals on the Bus are low true (that is, they are grounded for assertion) except for those marked with a #, which are asserted by allowing them to float high (approx. = 3 volts). Note that a bus loads module in the computer pulls all low true signals to a high when they are not asserted.

BRK CYCLE	BL2	*
BRK DATA CONT	BK2	*
BRK IN PROG	BE2	*
BUS STROBE	CK1	*
CPMA DISABLE	CU1	*
C0	CE1	*
C1	CH1	*
C2	CJ1	*
DEFER	DK2	
DATA BUS 0	AR1	*
DATA BUS 1	AS1	*
DATA BUS 2	AU1	*
DATA BUS 3	AV1	*
DATA BUS 4	BR1	*
DATA BUS 5	BS1	*
DATA BUS 6	BU1	*
DATA BUS 7	BV1	*
DATA BUS 8	DR1	*
DATA BUS 9	DS1	*
DATA BUS 10	DU1	*
DATA BUS 11	DV1	*
EMA 0	AD2	
EMA 1	AE2	
EMA 2	AH2	
EXECUTE	DL2	
FETCH	DJ2	
F SET	DP2	
GROUND	AC1	
	AF1	
	AN1	
	AT1	
	AC2	
	AF2	
	AN2	
	AT2	
	BC1	
	BF1	
	BN1	
	BT1	
	BC2	
	BF2	
	BN2	
	BT2	
	CC1	
	CF1	
	CN1	
	CT1	
	CC2	
	CF2	
	CN2	
	CT2	
	DC1	

GROUND (cont.)	DF1	
	DN1	
	DT1	
	DC2	
	DF2	
	DN2	
	DT2	
IR 0	DD2	
IR 1	DE2	
IR 2	DH2	
IND 1	CU2	
IND 2	CV2	
INITIALIZE	CR1	*#
INTERNAL I/O	CL1	*
INTERRUPT IN PROG	BP2	
INTERRUPT RQST	CP1	*
INTERRUPT STROBE	BD2	
INHIBIT	AP2	
I/O PAUSE	CD1	*
KEY CONTROL	DU2	
LINK	AV2	
LINK DATA	CR2	
LINK LOAD	CS2	
LOAD ADDR ENABLE	BM2	
MD 00	AK1	*
MD 01	AL1	*
MD 02	AM1	*
MD 03	AP1	*
MD 04	BK1	*
MD 05	BL1	*
MD 06	BM1	*
MD 07	BP1	*
MD 08	DK1	*
MD 09	DL1	*
MD 10	DM1	*
MD 11	DP1	*
MA 00	AD1	*
MA 01	AE1	*
MA 02	AH1	*
MA 03	AJ1	*
MA 04	BD1	*
MA 05	BE1	*
MA 06	BH1	*
MA 07	BJ1	*
MA 08	DD1	*
MA 09	DE1	*
MA 10	DH1	*
MA 11	DJ1	*
MA,MS LOAD CONTR	BH2	*
MD DIR	AK2	
MEM START	AJ2	
MS DISABLE	CV1	*

* = primary signals used in interfacing.

NOT LAST XFER	CM1	*
OVERFLOW	BJ2	
POWER OK	BV2	
PULSE LOAD ADDR	DR2	
RETURN	AR2	
ROM ADDR	AU2	
RUN	BU2	*
SKIP	CS1	*
SOURCE	AL2	
STOP	DS2	*
STROBE	AM2	
SWITCH	DV2	
TP 1	CD2	
TP 2	CE2	
TP 3	CH2	*
TP 4	CJ2	
TS 1	CK2	
TS 2	CL2	
TS 3	CM2	*
TS 4	CP2	
USER MODE	DM2	
WRITE	AS2	
MISC.		
TEST POINTS	AA1	
	AB1	
	BA1	
	BB1	
	CA1	
	CB1	
	DA1	
	DB1	
RESERVED	BR2	
	BS2	
VOLTAGES		
+5	AA2	
	BA2	
	CA2	
	DA2	
-15	AB2	
	BB2	
	CB2	
	DB2	

DATA BUS BITS These are the 12 lines that carry data to and from the AC during internal I/O operations.

NOTES:

MD BUS BITS These are the 12 lines that carry data to and from memory. In interfacing they are primarily used to carry the device codes to all peripherals during I/O operations, and they also carry the function bits to the Internal interface ~~deives~~ devices.

NOTES:

MA BUS BITS These 12 lines carry the address information to the memory. It is here that the new data break address is applied if the data break interface is being used.

NOTES:

CPMA DISABLE This signal is used to remove the CPMA from the MA bus. This feature is only used by the DATA BREAK interface which supplies its own MA information.

NOTES:

MA, MS LOAD CONTROL This signal when asserted prohibits the MA and the major states from being loaded and thus changed. This is asserted during a data break transfer where the data break interface supplies its ~~won~~ own major states. It also removes the CPMA from the MA bus. This is done because the DATA BREAK interface also supplies its own MA.

NOTES:

MS DISABLE This signal is used to disable the processors major states during break transfers.

NOTES:

INTERNAL I/O This signal is controlled by the various peripherals and is used to distinguish between the internal and the external type of peripheral. When asserted it means internal, and when not asserted it indicates that the device is External, at which time it activates the positive I/O interface.

NOTES:

NOT LAST XFER When asserted this signal will cause the C.P. timing to stall. That is, it will cause the timing to remain in TS3 once it is entered, until a BUS STROBE occurrs after this signal is no longer asserted.

NOTES:

BUS STROBE This is used as a substitute load pulse in the absence of TP3. It is used by EXTERNAL interfaces only, since this type of interface requires the stalling of C.P. timing. Keep in mind that no data transfers can be completed without loading the data into some register, and that a load pulse is required to do the loading.

NOTES:

C0, C1, C2 These are the three control signals used by an INTERNAL interface to control transfers to and from the AC.

NOTES:

TP3 TP3 is the pulse that the C.P. normally uses to load the AC and to clock the skip flip-flop. It is used for this by the INTERNAL interfaces, but since it is absent when using the EXTERNAL interfaces (due to the Positive I/O interface) it must be compensated for by the generation of BUS STROBES.

TS3 It is during TS3 that all peripheral transfers to and from the AC must take place. It is also the time state in which the C.P. stalls when the Positive I/O is used. For INTERNAL type interfaces the user must gate his data on and off the bus with TS3. (Note that the signal I/O PAUSE occurrs during TS3 so that it is the best signal to use for this purpose).

NOTES:

SKIP This is the skip bus used by INTERNAL INTERFACES only. If it it asserted during I/O PAUSE time a skip flip-flop in the C.P. will set causing the next sequential instruction in the program to be skipped.

NOTES:

I/O PAUSE This signal, when asserted, states that the processor is executing an IOT instruction. Because of this, I/O PAUSE is used as the enabling level to gate data onto and off of the data bus. It can also be used to gate the skip conditions onto the skip bus.

NOTES:

INTERRUPT REQUEST This signal is asserted by a peripheral that wishes to trigger an interrupt sequence. If the interrupts are active when this signal is asserted, the C.P. will unconditionally execute a JMS 0000.

NOTES:

BRK CYCLE This signal indicates that the data break interface is in the break major state and that data is being transferred into or out of memory.

NOTES:

INITIALIZE This signal is generated in one of two ways, it is generated from the front panel by pressing the clear key, and it also can be generated by the CAF (clear ALL FLAG) instruction. In either case the signal is used to clear the AC and the LINK as well as the flags in all the peripherals.

NOTES:

RUN Whenever RUN is asserted it simply indicates that the C.P. is in the process of executing instructions. (C.P. not halted)

NOTES:

STOP If STOP is asserted it will cause the processor to halt at the completion of the next cycle. (That is, the next reference of memory)

NOTES:

DEVICE SELECTION

When an IOT instruction is executed by the C.P. it is up to the individual devices to determine which one is being referenced by the instruction. This is accomplished by device codes. The device codes, as discussed in the section on IOT instructions, are the two digit numbers located in the center six bits of the instruction. These bits are available on the MD bus (bits 3 through 8). The purpose of the device selector logic in the interface is to compare the code coming from the instruction with the code selected for the device. This is done as indicated in drawings INT - 3 and INT- 5 . It is a further function of the device selector to generate a signal that can be used to enable all logic in the interface once it is determined that this is the device being referenced.

INTERNAL vs. EXTERNAL I/O

For all input-output operations, the processor is controlled by the following signals:

C0, C1, C2

BUS STROBE

SKIP

An internal interface is one in which the device interface itself directly controls these signals on the OMNIBUS to govern the I/O operation.

An external interface utilizes the Positive I/O BUS option to control these signals and in effect puts an interface between the C.P. and the actual device interface.

INTERNAL INTERFACES

ADVANTAGE: The primary advantage to utilizing the internal type of interface is speed. Since the IOP generator is not required a great savings in time can be realized by employing this type of interface. Where a typical external I/O instruction may require 4.25 micro seconds to execute, a comparable instruction for an internal device would only take 1.2 us.

DISADVANTAGE: The main disadvantage to an internal interface is the complexity of the design. While an internal interface is not basically a difficult design problem, it is none-the-less more difficult than employing the external type design. The reason for this is that when using an external interface, all of the C.P. control is accomplished by the Positive I/O Bus option.

CONSIDERATIONS: As with any type interface, we must consider the problem of device selection. Since this is the same for all interfaces you will find it covered under a separate section entitled: "DEVICE SELECTION".

For the internal interface it is important that the unit assert the signal INTERNAL I/O L whenever it detects it's device code. This is necessary because it is the only way the Positive I/O option has of knowing if it should generate IOP pulses. If INTERNAL I/O is asserted it in effect turns off the Positive I/O. Even if no external interfaces currently exist in the system it is still good policy to assert the signal since an external device may be added in the future requiring the use of the Positive I/O.

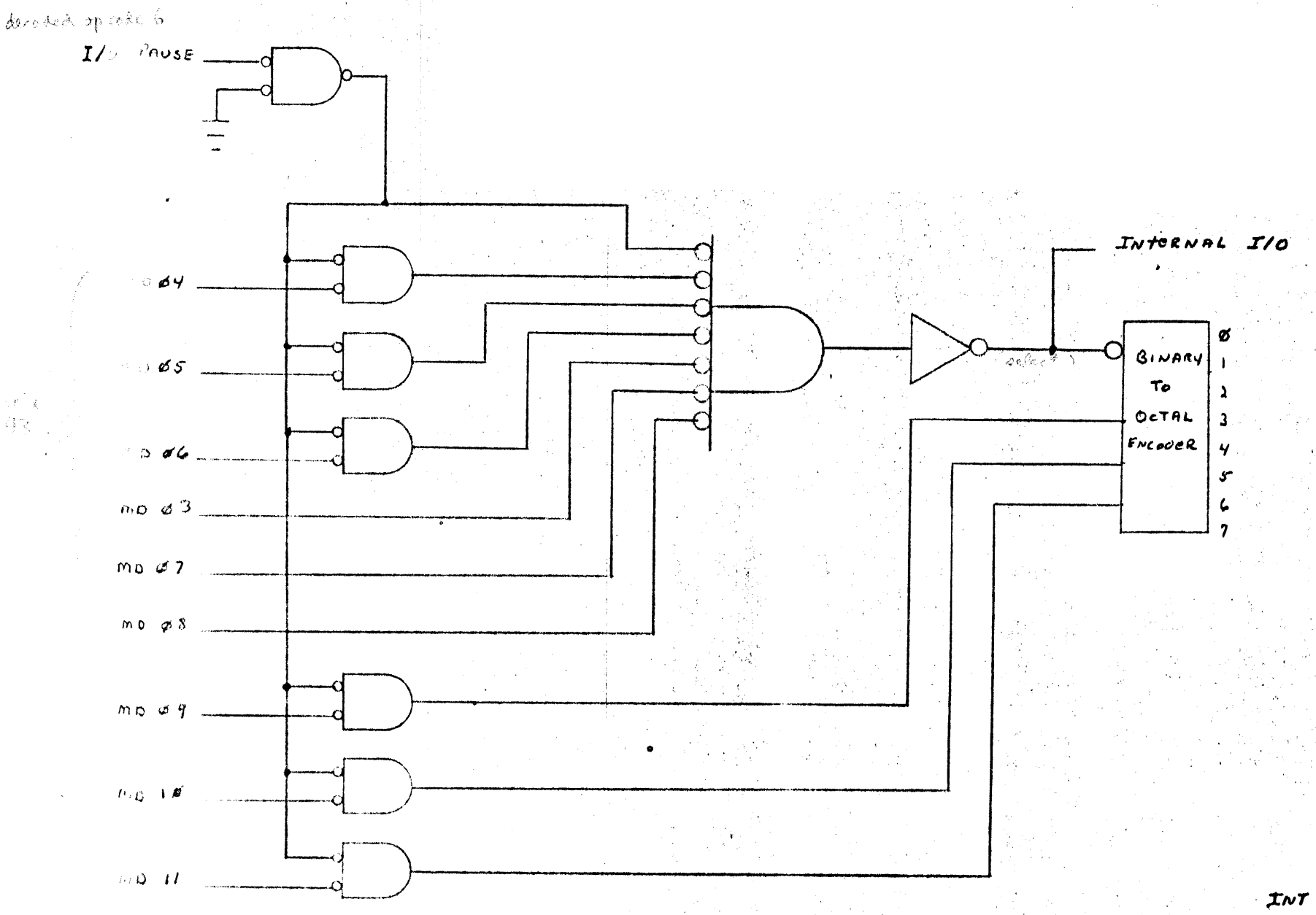
Once device selection has been accomplished it is next necessary to decode the individual IOT instructions. Since an internal interface uses all possible combinations of IOT instruction bits 9, 10, and 11 it is now necessary to do a binary to octal conversion on these three bits. There are two easy ways that this can be accomplished.

1. If you are assembling the interface by using standard modules, a DEC type M161 Binary to decimal converter would be best.
2. If you are designing your own modules from an IC level, a Signetics 8251 Binary to Decimal converter will do the job.

In either case use the output of your ~~device selector~~ function decoder to activate the binary to octal converter. (See drawing #INT-3)

After the three function bits from the IOT instruction have been converted, the resultant octal output will be used to control the C0, C1, and C2 signals which control processor operation. (See chart #INT-2) Keep in mind that all transfers are completed with the occurance of a Bus Strobe signal, which is generated by the C.P. at TP_3 time.

Internal Device Select and I/O Decode
(Device code 43)



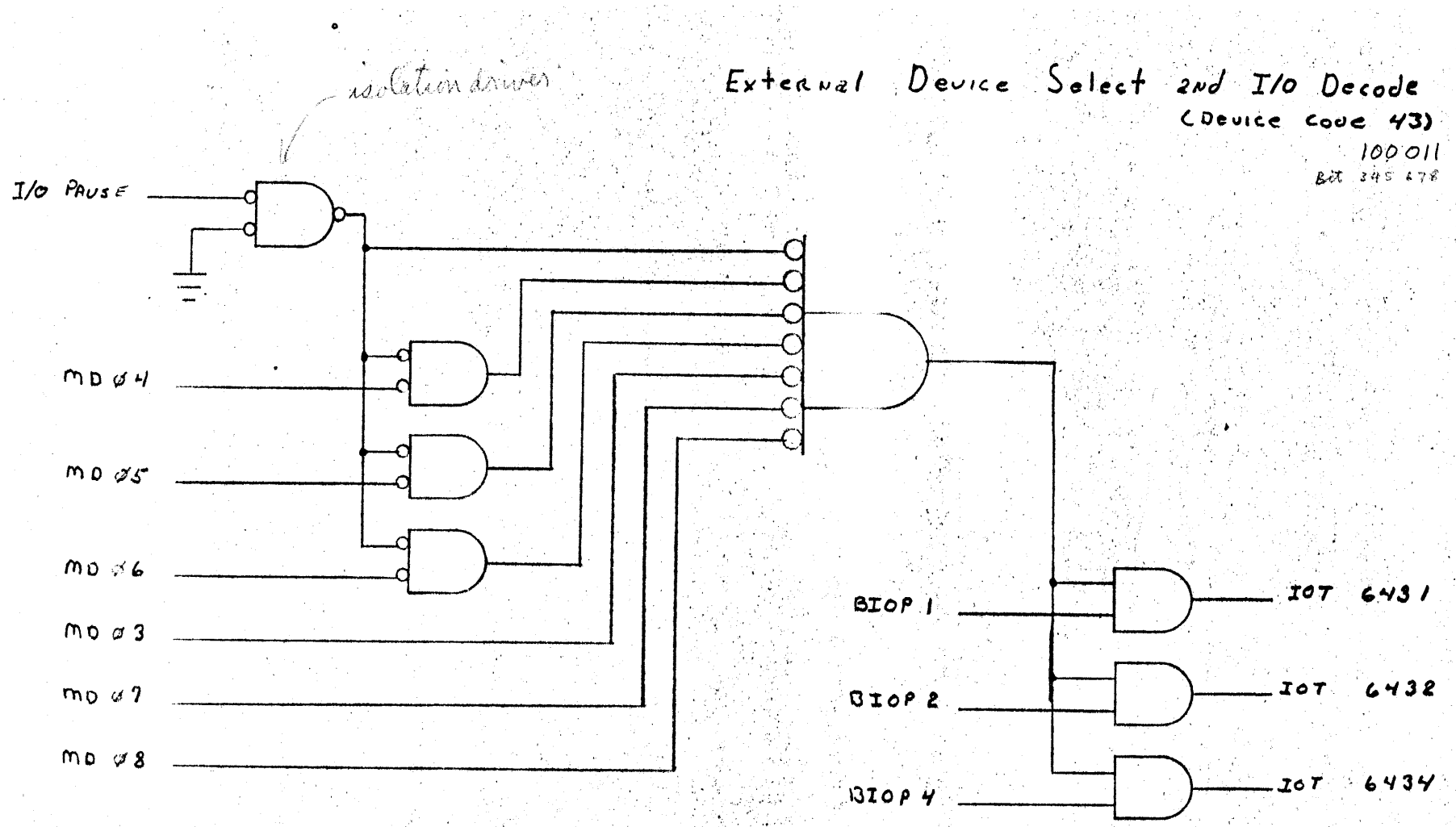
CONTROL LINE TRUTH TABLE

YPE OF XFER	C0,C1,C2	AC⟶BUS	BUS⟶ADDER	AC LOAD	PC LOAD
C OUTPUT C UNCHANGED	H H H	YES	YES	YES	NO
C OUTPUT C CLEARED	L H H	YES	NO	YES	NO
C INPUT C OR INPUT	H L H	YES	YES	YES	NO
C INPUT ATA TO AC	L L H	NO	YES	YES	NO
C INPUT C PLUS DATA	L H L	NO	YES	NO	YES
C INPUT ATA TO PC	L L L	NO	YES	NO	YES

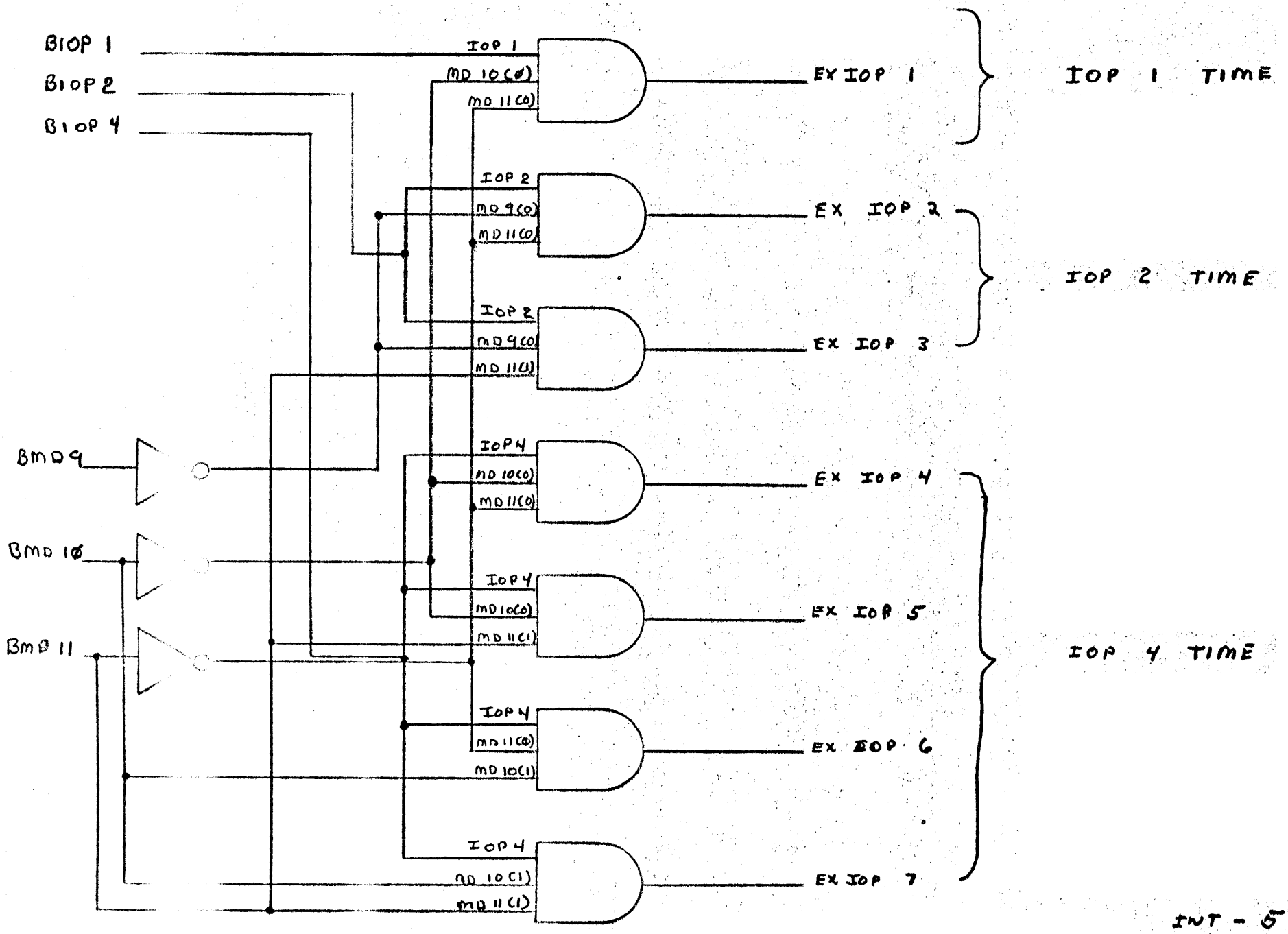
ADVANTAGE: The advantage to external rather than internal interfaces is simplicity. Since you are using the Positive I/O option to control all input-output functions it is not necessary to design any logic to control the C lines, or any other control signals in the processor.

DISADVANTAGE: The problem with external interfaces is the simple fact that they are comparitively slow, requiring nearly three times the instruction execution time of commands for internal interfaces. This is brought about by the need to generate up to three very long (approx. 750 ns) I/O pulses.

CONSIDERATIONS: After the interface has decoded the device code, the output of the decoder should be used to gate with the incomming IO pulses to decode the appropriate instructions. A typical logic circuit is shown in figure INT-4. This particular diagram illustrates how only three IOT's can be decoded. It is possible to decode up to 7 different functions even though only three pulses are generated. This can be accomplished by gating the three IOP pulses with the various combinations of bits 9, 10, and 11 as shown in figure INT-5. It should be noted that even though seven combinations are decoded there are still only three discrete time periods for these functions. The three time periods being derrived from the three IOP pulses.


isolation drivers
External Device Select 2nd I/O Decode
(Device code 43)
100011
Bit 345 678
I/O PAUSE
MD Ø4
MD Ø5
MD Ø6
MD Ø3
MD Ø7
MD Ø8
BIOP 1
BIOP 2
BIOP 4
IOT 6431
IOT 6432
IOT 6434

Extended IOP Generation



Since the positive I/O interface option will handle the bulk of the input-output operations, there are only two signals which the designer must concern himself with. These are:

1. $\overline{\text{SKIP}}$
2. $\overline{\text{AC CLEAR BUS}}$

$\overline{\text{SKIP}}$: This signal is used to allow a peripheral to control program sequence. When the positive I/O generates any of its three IOP's, it monitors this signal. If the signal is asserted (grounded) during the IOP, an internal counter will be incremented in the positive I/O. At the end of the IOT instruction the counter will be automatically added to the Program Counter in the processor, thus giving the interface the ability to skip from Ø to 3 instructions in the processor's program.

$\overline{\text{AC CLEAR BUS}}$: The purpose of this signal when asserted is to prevent the processor from enabling the AC on to the Data Bus. Since your interface will transmit data to the processor via the bus, the use of this signal becomes quite important. To see its application in I/O Data Transfers check the chart on INT-6.

TYPE OF XFER	AC CLEAR BUS	PERIPH DATA → BUS	BUS DATA → PERIPH
PERIPH → AC	L	YES	NO
PERIPH ∧ AC → AC	H	YES	NO
0 → AC	L	NO	NO
AC → PERIPH AC UNCHANGED	H	NO	YES
AC → PERIPH AC CLEARED	L	NO	YES

DATA TRANSFER TRUTH TABLE FOR EXTERNAL INTERFACES Int-6

POSITIVE I/O INTERFACE

The positive I/O interface option serves as a link between the users peripheral and the PDP-8/e. As seen in the discussion on external interfaces the positive I/O does not take care of all interfacing problems, some signals still must be controlled by the users interface. The following is a discussion of the functions that are controlled by the interface, and a description of how it operates

Since not all peripherals on a system will be of the external type, the positive I/O will not always be active. The positive I/O is activated by not grounding the signal INTERNAL I/O. This indicates that the assumed type of interface is to be external.

Once the interface is activated, it will pick up bits 9, 10, and 11 of the IOT instruction. These bits will be used to determine which IOP pulses will be generated. If bit 11 is set, IOP 1 will be generated; if bit 10 is set IOP 2, and if bit 09 is set IOP 4 will be gen. The duration of these pulses and the separation between them is adjustable, with the normal settings being 750 ns for duration and 300 ns for separation. The function of each pulse is not determined by the positive I/O, it is left to the users interface to decide what each pulse is to be used for. There is however, a standard usage for these pulses which is:

IOP 1 Normally used for testing flags and generating skip conditions.

IOP 2 Normally used for clearing buffers and the AC

IOP 4 Normally use for the actual data transfers.

It should be carefully noted that these are only the normal usages and that they are not manditory. Each pulse can be used in any manner desired. It should also be noted that only the desired pulses are generated. For example, if an IOT instruction of the form 6xx5 were executed, IOP 1 would be generated immediately followed by IOP 4 (following the normal separation delay).

Since the use of each I/O pulse is left to the user, the Positive I/O must consider each possible application at the time the pulse is generated. The possible applications being:

1. Skip the next instruction on a given condition.
2. Transfer data to the AC.
3. Transfer data to the peripheral (AC cleared).
4. Transfer data to the peripheral (AC unchanged).

Because of the varied possibilities, the positive I/O will cause the following common events to occur each time that an IOP is generated:

1. If the device grounds the signal SKIP BUS the positive I/O will increment a skip counter at Bus Strobe time.
2. Unless the peripheral grounds the signal AC CLEAR BUS the AC will be placed onto the data bus.
3. At bus strobe time the positive I/O will unconditionally load the AC from the data bus.

(For an explanation of how to implement data transfers using these common events see the section on External Interfaces.) After all IOP pulses have been generated, the positive I/O interface will then take what ever value is in the skip counter and add it to the Program

Counter thus causing the computer to skip either 0, 1, 2, or three times depending on how many skip conditions were encountered during the IOP generation time.

BUS STROBE TIME Exactly 100 ns before the end of each IOP pulse the positive I/O interface will generate a BUS STROBE signal which can be used as a load pulse in the C.P., hence, BUS STROBE TIME occurs 100 ns before the end of each IO pulse.

INTERVENTION WITH PROCESSOR TIMING Since a great deal of time is required to generate the I/O pulses, it is necessary for the Positive I/O generator to stall the processor timing. This is accomplished as follows:

When the positive I/O determined that it is to be used (the presence of an IOT instruction with the absence of the signal INTERNAL I/O) it generates a signal to the C.P. called NOT LAST XFER L, this causes the processor to remain in time state three once it is entered and to remain there until such time as the interface lifts this signal and generates one additional BUS STROBE signal. The occurrance of a BUS STROBE without the signal NOT LAST XFER L, will restart C.P. timing which will pick up at time state 4, and run normally from there. The following should be noted:

1. Even though the C.P. is forced to wait in time state 3, time pulse 3 will still be generated at the usual time. (It is the absence of TP3 later on in the I/O timing that makes the presence of BUS STROBES necessary, since there is no other pulse present that can be used for loading puposes).

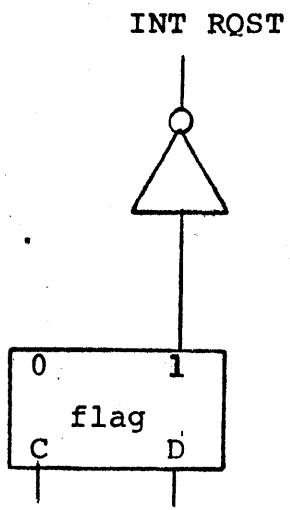
2. Even though the C.P. is stalled, the memory timing will run to its normal completion and simply wait for a new cycle to begin (this will take place when the processor is restarted).

For a graphic illustration of timing see drawing number INT-7.

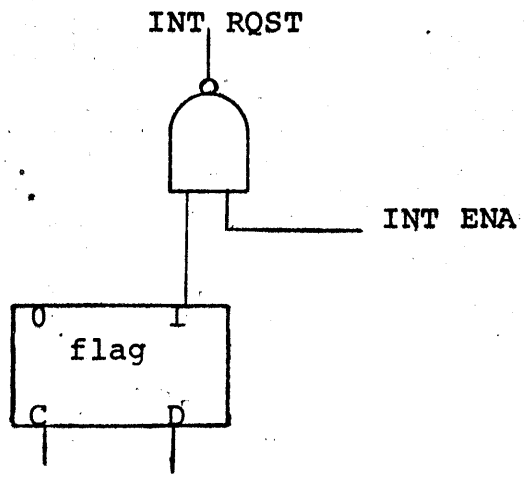
PROGRAM INTERRUPT

Because of the great difference in timing between the C.P. and most peripherals, it becomes necessary from a programming standpoint to determine if a device is ready to transfer data to or from the C.P. before the transfer actually takes place. This is accomplished in one of two ways; first, the programmer could issue a skip instruction which would cause the program to skip the next sequential instruction if the device ready flag is set, indicating that the device is ready to transfer. The second technique is to have the device flag automatically and notify the C.P. when it is ready. This is a far better method since it does not require the C.P. to dedicate all of its time to flag checking. This second method can be accomplished by using the program interrupt feature of the computer. This works by having the hardware monitor all device flags (to accomplish this all the programmer need do is issue an ION instruction). If the hardware detects a flag it will automatically execute a JMS 0000.

From an interface standpoint all that is necessary to use the interrupt facility is to design the device flag so that when it sets it will assert the signal INT REQST on the OMNIBUS. (See drawing no. INT.- 10 .) Since the hardware in the C.P. is designed with only one interrupt request line it must monitor all flags tied to it, consequently if the programmer wishes to ignore a given device he has no easy way of doing it. For this reason it is desirable to have the device flag designed so that the programmer can issue a command that would eliminate thatparticular flag from the interrupt request line. This can be accomplished by using the technique shown in drawing no. INT.- 11 .



INTERRUPT REQUEST Int-10



INTERRUPT ENABLE Int-11

DATA BREAK

When interfacing mass storage devices to the computer such as disks and magnetic tape units, a new interfacing problem is encountered that did not appear in the simpler devices. With a non-mass storage device such as the teletype, we were dealing with speeds that were so slow that we were able to transfer data to the computer via the program interrupt system. Under that system whenever the device had a word ready to transfer to the C.P. it set a flag which generated an interrupt request. When the C.P. saw the request it did a JMS 0000 and a program was then run using individual IOT instructions to obtain the data from the peripheral, or transfer to the peripheral. The problem is that this takes time. In fact it takes an absolute minimum of 16 us. before the program could reach the word that the device had waiting. If the device were fast enough that it could have a new word ready in less than 16 us. the program could not possibly access the data fast enough. Because of this speed problem, and also due to the fact that in a mass storage device we are normally going to be transferring large quantities of data, the DATA BREAK system was designed.

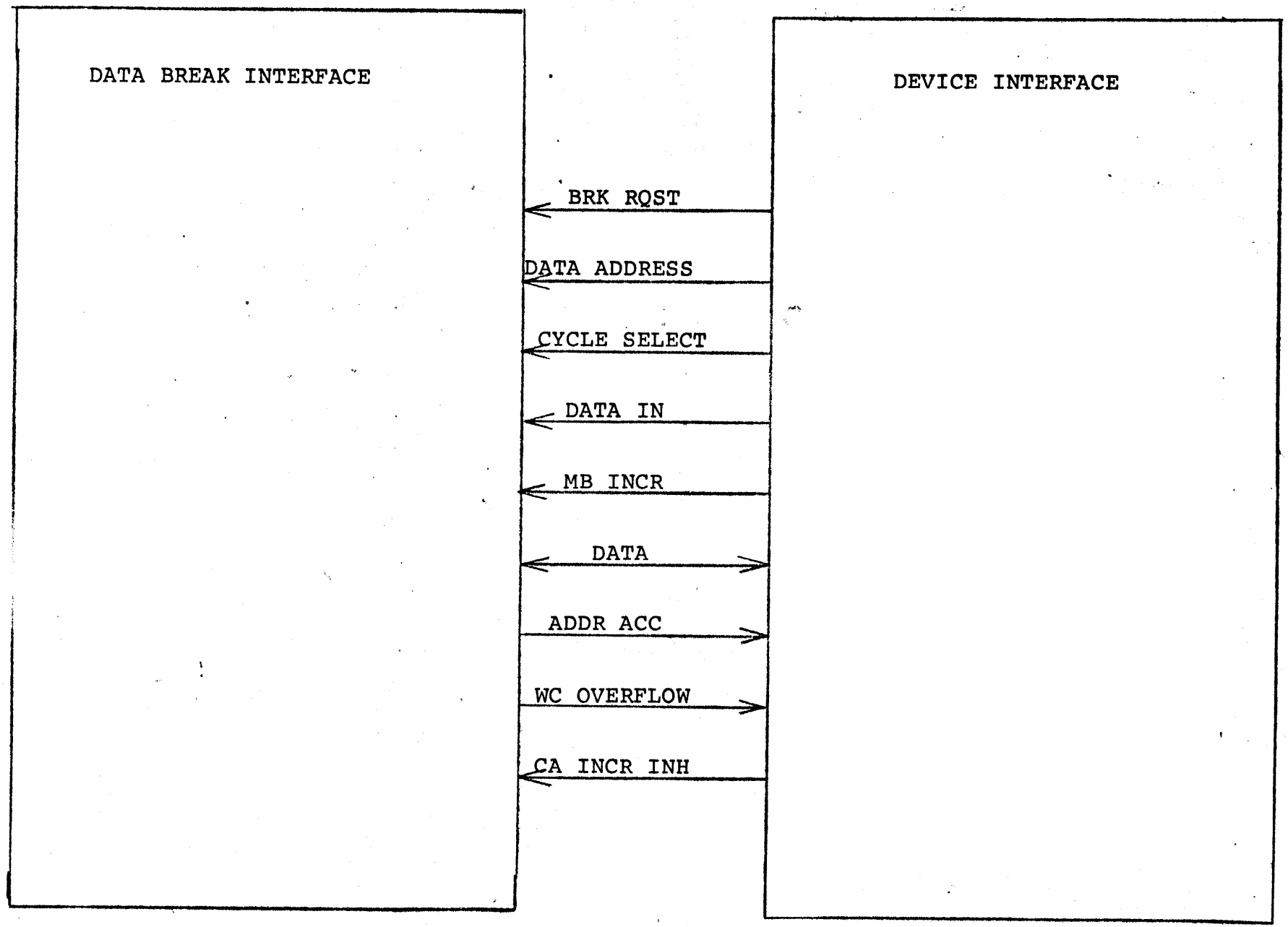
When using data break the theory is that the peripheral will set a flag as each data word is ready for transfer. Once the C.P. acknowledges the flag it will allow the device to transfer the data directly into or out of memory. This direct memory access will take place completely without program intervention. That is to say that the transfer will be handled completely by the hardware. Since

the DATA BREAK INTERFACE module does the bulk of the work in a direct memory transfer, we will concentrate on the control information which must pass from the device itself to the DATA BREAK interface.

DATA BREAK OPERATION There are two forms of data break transfers to be considered, first ~~there~~ single cycle data break. This is the fastest of the two, and also the most complex, involving a rather fancy interface. The second called THREE CYCLE DATA BREAK is three times slower, but requires a far less complex interface. The first to be discussed will be single cycle.

SINGLE CYCLE DATA BREAK The basic theory of single cycle break is that when the device is ready to transfer a word it sets a BREAK REQUEST flag. The device then sends an address to the data break interface. This is the address at which the transfer will take place. The interface will then accept a word from the peripheral and send it to memory at that address, or else send a word to the device from that memory address. This entire operation takes place between normal procssor cycles. In effect the interface steals a cycle from processor time in order to use memory for the peripheral. Since the hardware does all the work, and since this happens between cycles of standard instructions, no program intervention is necessary. The exact sequence, and the signals necessary to make it work is as follows: (See Drawing No. INT. 12).

1. The device generates a BRK RQST signal to the DB (data break) interface.
2. In addition to the request the device must specify the address at which the transfer is to take place.
3. As soon as the DB interface detects that the C.P.



DATA BREAK SIGNAL CHART Int-12

has completed the current cycle, it will enter its own major state called BREAK. At this time it will generate MA DIS to remove the CPMA from the MA bus. In its place it will put the address specified by the peripheral. To notify the device that this has taken place the DB Interface will generate a signal to the device called ADDRESS ACCEPTED.

4. Since it is possible for data to be going to or from memory the device must also specify the direction of data transfer. This is done at the same time that the address is sent by use of the signal DATA IN. If DATA IN is at ground, the data will go from the device to the memory, otherwise the data will go from memory to the device. It is also possible at this point that rather than transfer data to or from memory, the device merely wants to increment a number in memory. This can be accomplished by use of the MB INCREMENT signal. If MB INCREMENT is grounded and the DATA IN signal is high, the DB interface will cause the content of the specified memory location to be incremented. If MB INCREMENT is low and so is DATA IN, the DB interface will take a data word from the peripheral and add it to the data in the memory location specified. When using the MB INCREMENT signal if the increment or the addition causes the CP to produce a 13 bit result a signal will be sent to the peripheral called W.C. OVERFLOW.

5. Once the transfer is completed, the DB interface will lift the signal MA DISABLE, thus allowing the C.P. to continue on with its program as if nothing had happened.

LEST WE FORGET Since the transfer mentioned above occurs without any program intervention, it is necessary to arrive at some method of determining when all the necessary data has been transferred. This can only be done by the device itself. It will be necessary for the device to keep track of the number of words transferred, and when all the words are transferred, notify the C.P. that this has taken place. This can be done via the interrupt facility. The actual word counting is best handled by loading into the device (under program control at the beginning of the program) a negative number representing the number of words to be transferred. As each word is sent the device must increment this value and check to see if the result is zero. If it is then all words have been sent, and an interrupt can be generated, if not then the device will simply get its next word and continue breaking until the transfer is completed.

One other consideration is that the device will have to update the address it sends to the DB interface as each word is sent so that all words will go into or come from sequential memory locations.

You can see from the above considerations that the interface in the device can become rather complex, since it necessatates the use of two 12 bit, self incrementing, programmable registers. And in addition the register used to count words must be able to detect a result of zero so that it can generate an interrupt. The advantage to the three cycle break approach is that these two functions are

performed automatically in the C.P.

THREE CYCLE DATA BREAK The actual transfer of data under three cycle break is the same as for single cycle. The only difference is that in three cycle the counting of words and the updating of the address is taken care of in this processor. This makes it necessary for the break to take three cycles rather than the single cycle used earlier, hence the name three cycle break. The three cycle break is initiated in the same manner as the single cycle, that is, it is still necessary to generate a break request, send the address, specify the direction of transfer, and whether or not to use the MB increment feature. The difference is as follows:

1. The address specified must be an even address. this address is taken as the location in memory of the word count. When the break is started the computer will go to the address specified where it will remove the contents of that memory location and increment. If this number ever goes to zero the DB INTERFACE will generate the signal WC OVERFLOW. Even if the word count should go to zero at this point the DB INTERFACE will still complete the current data transfer. This cycle is referred to as the WORD COUNT major state.
2. After the computer counts off the current word, the DB interface will then take the original address and add 1 to it.

This new address is the pointer to the memory location which contains the address at which the data transfer is to take place. Once the new address is generated, the DB interface enters the CURRENT ADDRESS major state, where it takes the incremented address and obtains the content of that memory location. At this point, unless the device generates a signal called CA INCR INH the DB interface will increment the number it has obtained from memory. This is the address at which the transfer will take place. The DB interface now enters the BREAK major state and proceeds as with a standard single cycle data break.

Note that even though this break took three cycle rather than one, it still only managed to transfer a single word. The two additional cycles were necessary in order to have the word count and current address taken care of in the C.P.'s memory.

ONE LAST THOUGHT Since there are two types of data break one additional signal must be sent along with the others. That is a signal called CYCLE SELECT which is used to tell the DB interface whether the break is to be single or three cycle. If this signal is grounded that indicates a three cycle break, and single cycle if high (3 volts).

DATA BREAK SIGNAL DEFINITIONS

BRK RQST This signal is generated by the device wishing to make a direct memory access.

DATA ADDRESS These 12 lines carry the address at which the transfer is to take place for a single cycle break. For three cycle break this is the address in memory at which the word count value will be found. In either case this address is sent from the device to the DB interface.

CYCLE SELECT This signal generated by the device, specifies whether the break is to be single or three cycle. If it is at ground, it indicates 3 cycle. If it is a +3 volts it indicates single cycle.

DATA IN This is used by the device to specify to the DB interface the direction of data transfer. If it is at ground it says data into the C.P. If it is at +3 volts it means data out of the C.P.

MB INCREMENT When this is asserted by the device it can have one of two meanings. If it is asserted in conjunction with DATA IN being at ground, the result will be that a data word is taken from the device and added to a word in memory at the address specified by the device. If it is used with DATA IN at +3 volts, it will cause the word in the memory location specified by the device to be incremented.

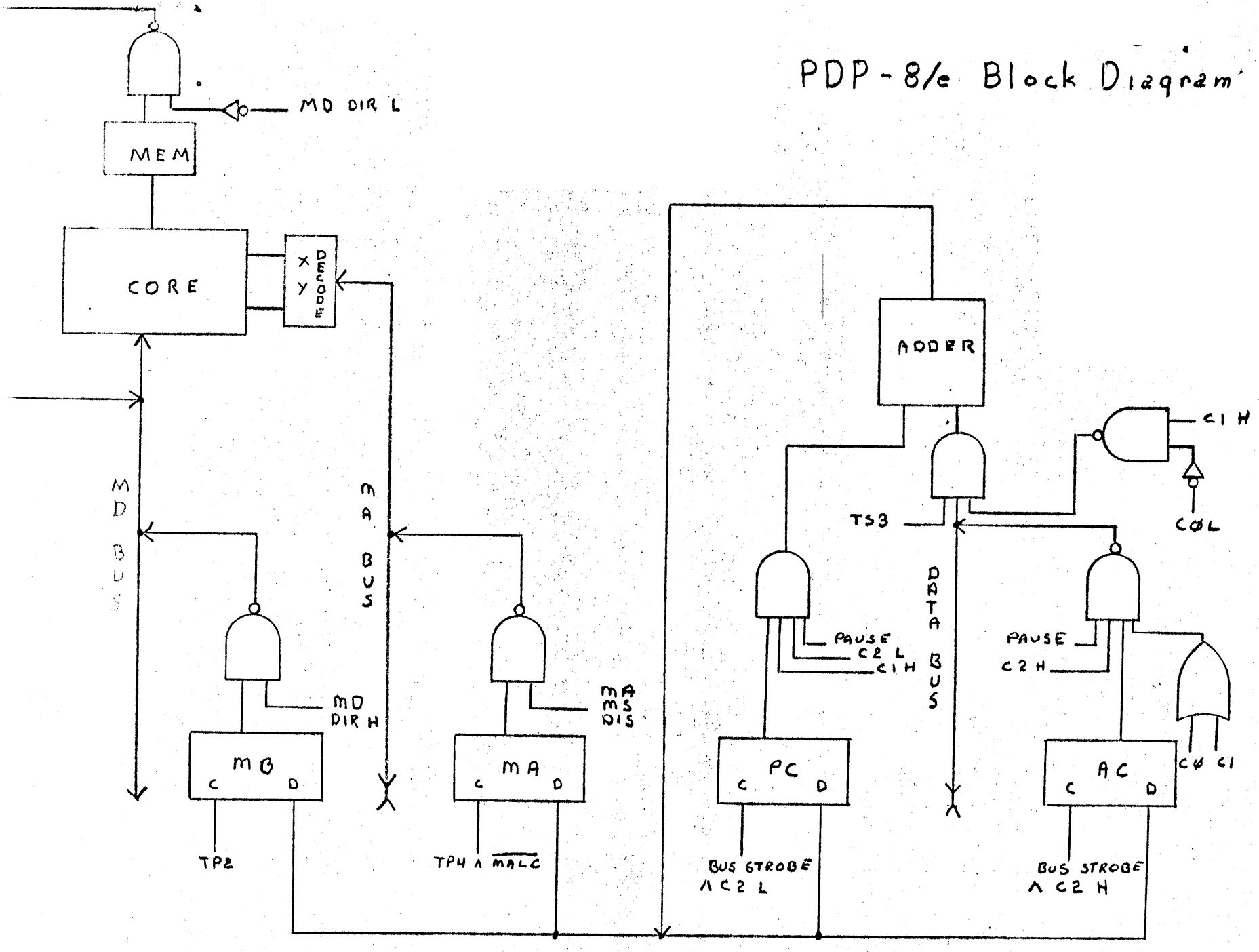
DATA These are the 12 lines that carry data from the device to the DB interface which in turn passes it on to memory. To receive data from the C.P., the device must obtain its data from the MD BUS.

ADDR ACC This signal is generated by the DB interface at the time when it accepts the address from the device. It is normally used by the device to clear the BREAK REQUEST flag. In the case of single cycle break peripherals, it is also used to indicate that the word count, and data address registers must be updated.

W C OVERFLOW There are two ways that this signal gets generated. It can be generated in the word count major state of a three cycle break if the word count goes to zero. It can also be generated during the BREAK major state of any data break if the MB INCREMENT signal is being used, and if the data being incremented, or added to, should produce a 13th bit.

CA INCR INH This signal is only used by three cycle break. It is used whenever the device does not want the address being taken from memory during the CURRENT ADDRESS major state to be incremented.

PDP-8/e Block Diagram



INT - 1

APPENDIX A

CABLE LISTINGS

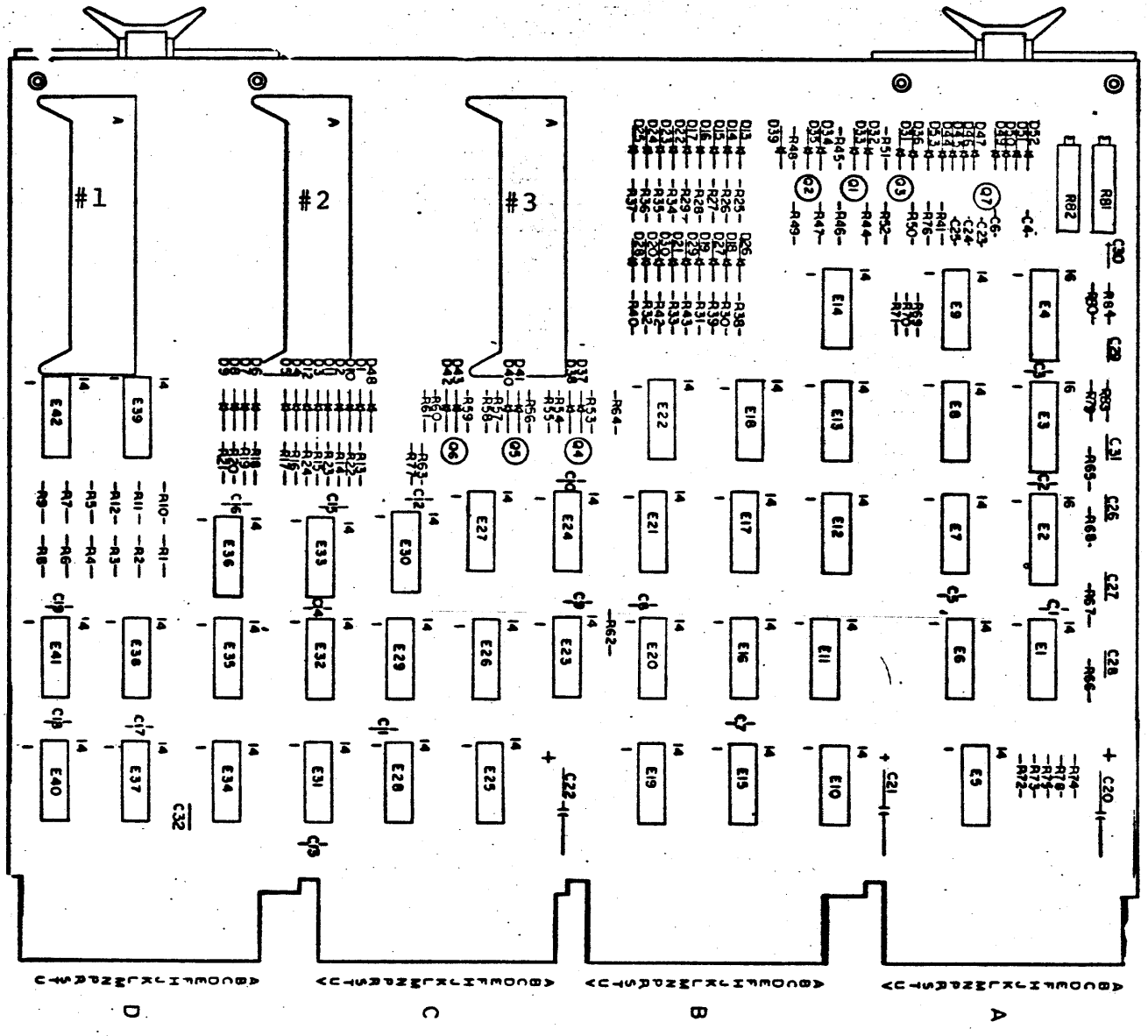
M8350 POSITIVE I/O BUS CABLES

SIGNAL	#1 BERG	M953A
MD 00 H	D	B1
MD 01 H	J	D1
MD 02 H	N	E1
MD 03 H	X	J1
MD 03 L	T	H1
MD 04 H	FF	M1
MD 04 L	BB	L1
MD 05 H	RR	S1
MD 05 L	LL	P1
MD 06 H	L	E2
MD 06 L	F	D2
MD 07 H	V	K2
MD 07 L	R	H2
MD 08 H	DD	P2
MD 08 L	Z	M2
MD 09 H	JJ	S2
MD 10 H	NN	T2
MD 11 H	TT	V2

SIGNAL	#2 BERG	M953A
AC 00	D	B1
AC 01	J	D1
AC 02	N	E1
AC 03	T	H1
AC 04	X	J1
AC 05	BB	L1
AC 06	FF	M1
AC 07	LL	P1
AC 08	RR	S1
AC 09	F	D2
AC 10	L	E2
AC 11	R	H2
TS 3	JJ	S2
TS 1	NN	T2
INIT	TT	V2
BIOP 1	V	K2
BIOP 2	Z	M2
BIOP 4	DD	P2

SIGNAL	#3 BERG	M953A
DATA 00	D	B1
DATA 01	J	D1
DATA 02	N	E1
DATA 03	T	H1
DATA 04	X	J1
DATA 05	BB	L1
DATA 06	FF	M1
DATA 07	LL	P1
DATA 08	RR	S1
DATA 09	F	D2
DATA 10	L	E2
DATA 11	R	H2
RUN	JJ	S2
$\overline{\text{AC CLEAR}}$ BUS	DD	P2
$\overline{\text{INT RQST}}$	Z	M2
$\overline{\text{SKIP}}$ BUS	V	K2

M8350 POSITIVE I/O BUS MODULE

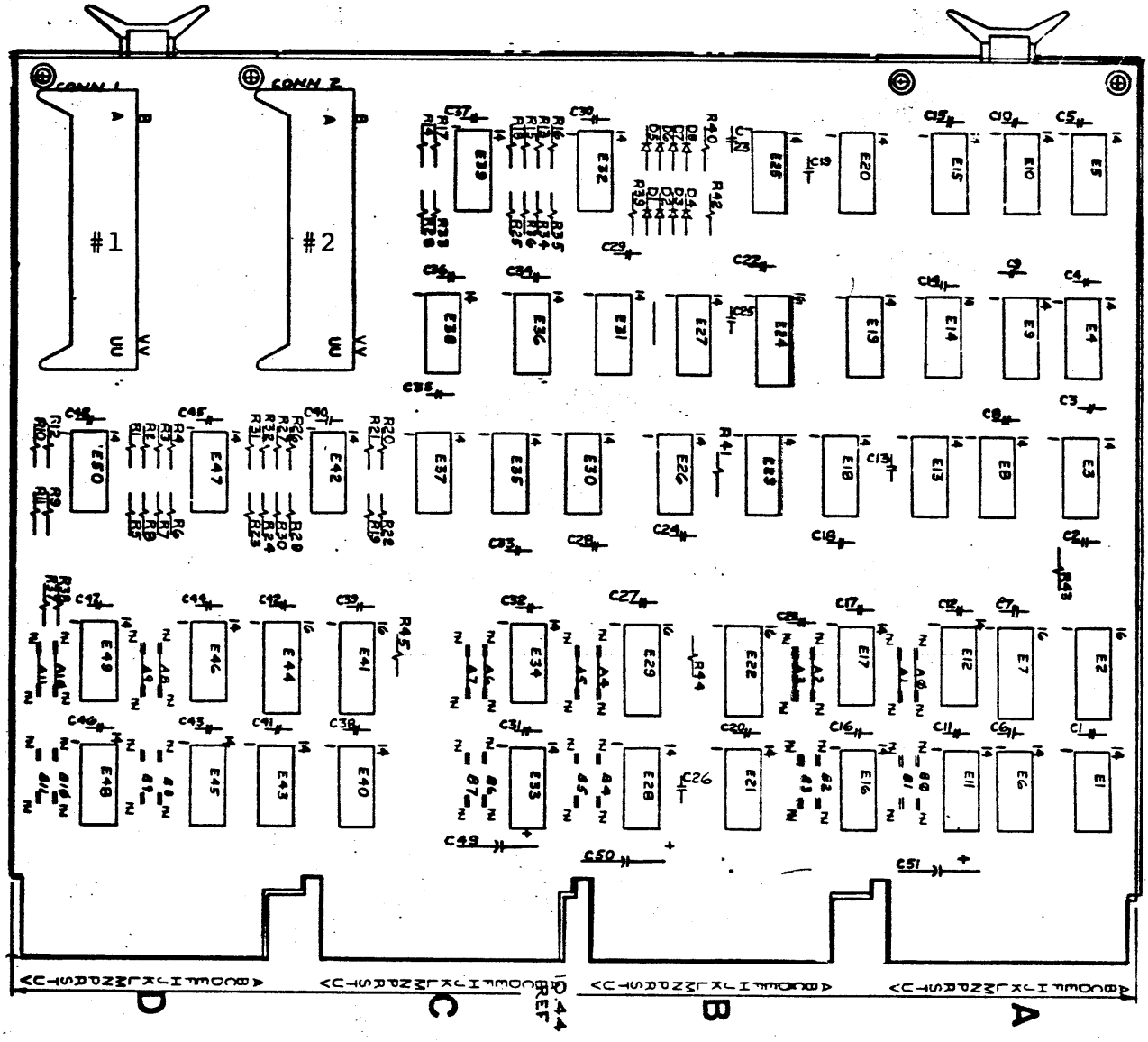


M8360 DATA BREAK CABLES

SIGNAL	#1 BERG	M953A
DATA ADD 00	D	B1
DATA ADD 01	J	D1
DATA ADD 02	N	E1
DATA ADD 03	T	H1
DATA ADD 04	X	J1
DATA ADD 05	BB	L1
DATA ADD 06	FF	M1
DATA ADD 07	LL	P1
DATA ADD 08	RR	S1
DATA ADD 09	F	D2
DATA ADD 10	L	E2
DATA ADD 11	R	H2
INIT	TT	V2
BREAK	DD	P2
ADD ACC	JJ	S2
BRK RQST	V	K2
DATA IN	Z	M2
MB INCR	NN	T2

SIGNAL	#2 BERG	M953A
BRK DATA 00	D	B1
BRK DATA 01	J	D1
BRK DATA 02	N	E1
BRK DATA 03	T	H1
BRK DATA 04	X	J1
BRK DATA 05	BB	L1
BRK DATA 06	FF	M1
BRK DATA 07	LL	P1
BRK DATA 08	RR	S1
BRK DATA 09	F	D2
BRK DATA 10	L	E2
BRK DATA 11	R	H2
WC OVERFLOW	DD	P2
CYCLE SELECT	V	K2
CA INC INH	Z	M2
EXT DATA ADD 0	TT	V2
EXT DATA ADD 1	NN	T2
EXT DATA ADD 2	JJ	S2

M8360 DATA BREAK INTERFACE MODULE



APPENDIX B

DEVICE CODE DISTRIBUTION

PDP 8/e DEVICE CODES

00	CENTRAL PROCESSOR
01, 02	HIGH SPEED READER/PUNCH
03, 04	TELETYPE
05, 07	VC8-E VIDEO DISPLAY
10	KP8-E POWER FAIL
11	DP8-EP REDUNDANCY CHECK CONTROL
12	CARD PUNCH AND CONTROL
13	REAL TIME CLOCK
14-17	CUSTOMER'S USE
20-27	EXTENDED MEMORY
30-35	RESERVED
36, 37	GENERAL PURPOSE INTERFACE BB08
40-47	SYNCHRONOUS DATA INTERFACE
50, 51	RESERVED
52	AMB-EA
53	AD8-EA A/D CONVERTER
50-57	DR8-EA BUFFERED DIGITAL I/O
60-62	DF-32D DISK
60-62, 64	RS08 DISK
63,67	CARD READER
65	X-Y PLOTTER
66	LINE PRINTER
70-72	MAGNETIC TAPE TRANSPORT
70-77	INCREMENTAL TAPE TRANSPORT
73-75	RK8 DISK
76,77	DECTAPE CONTROL