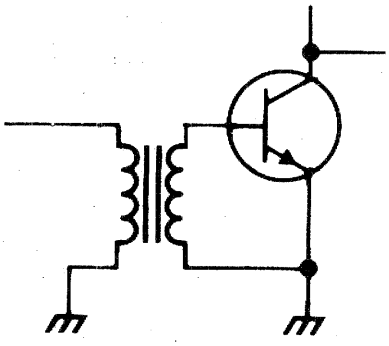
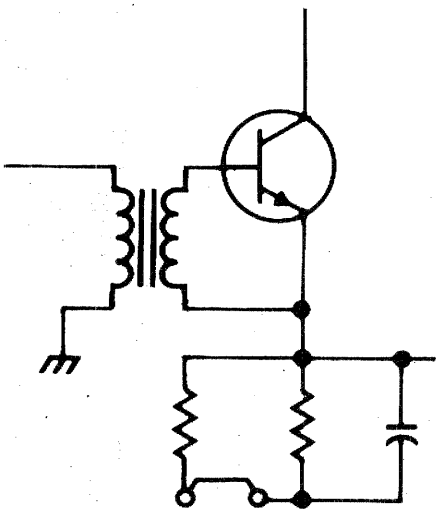
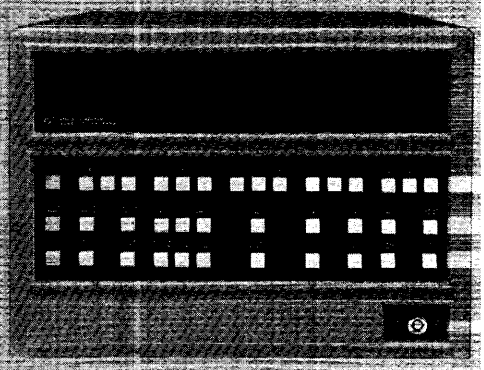






data systems
training center

2100 Computer Hardware Training Manual

HEWLETT-PACKARD

COMPUTER MAINTENANCE COURSE

2100 Computer Hardware Training Manual

STOCK NO. (02100-90159)

-NOTICE-

The information contained in this manual is for training purposes only. Consult the Hewlett-Packard documentation supplied with the computer for current information concerning the specific computer system furnished.

11000 Wolfe Road, Cupertino, California 95014 Area Code 408 257-7000 TWX 910-338-0221

Printed in USA April, 1973

LESSON INDEX

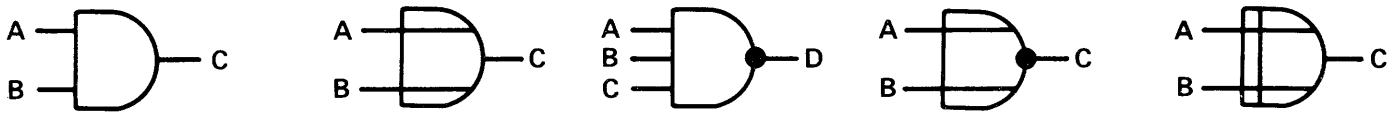
introduction to hardware 1

LESSON 1

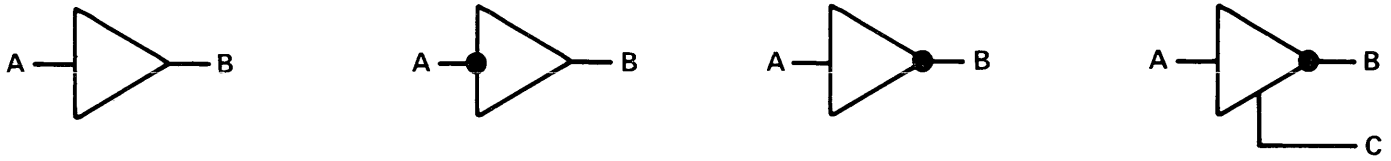
INTRODUCTION TO HARDWARE

FOUR MAJOR CLASSES OF HP LOGIC SYMBOLS

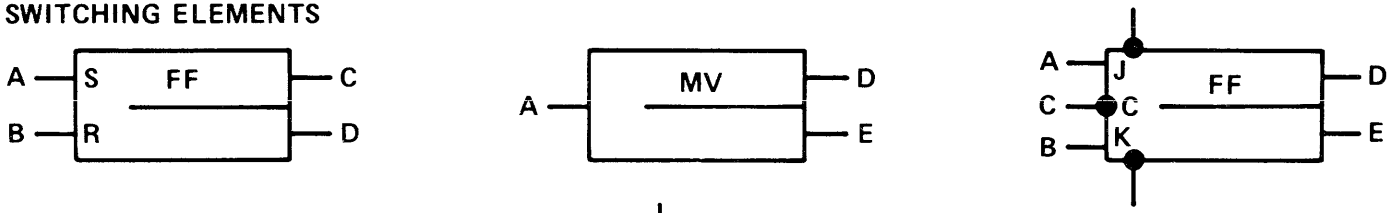
1. GATES



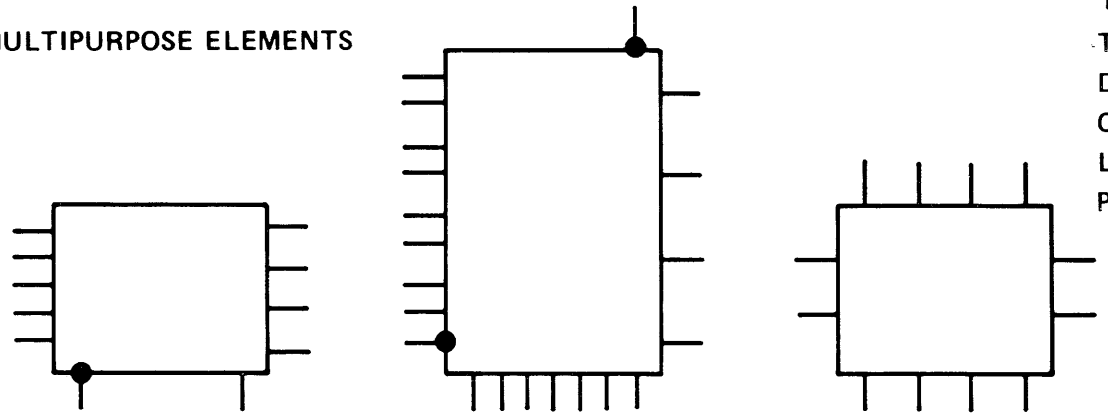
2. AMPLIFIERS



3. SWITCHING ELEMENTS



4. MULTIPURPOSE ELEMENTS



4-BIT SHIFT REGISTER

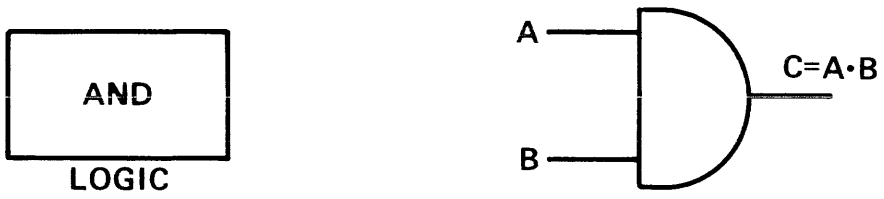
ARITH. LOGIC FUNCTION GENERATOR

FOUR 2-BIT WORD RAM

THE SYMBOL INCLUDES A DESCRIPTIVE NAME INDICATING THE OVERALL LOGIC FUNCTION PERFORMED.

THE "AND" GATE

"AND" PROVIDES LOGICAL MULTIPLICATION



THE OUTPUT "C" IS TRUE (POSITIVE) ONLY
WHEN "A" AND "B" ARE BOTH TRUE (POSITIVE)

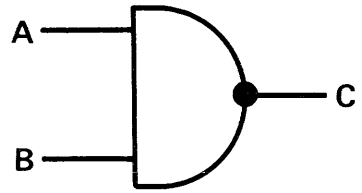
THE ARITHMETIC MULTIPLICATION SIGN (•) ALWAYS INDICATES "AND" IN LOGIC EQUATIONS AND CIRCUITS. THE TERM C=A•B IS A LOGIC EQUATION MEANING BOTH "A" AND "B" MUST BE TRUE FOR "C" TO BE TRUE.

TRUTH TABLE

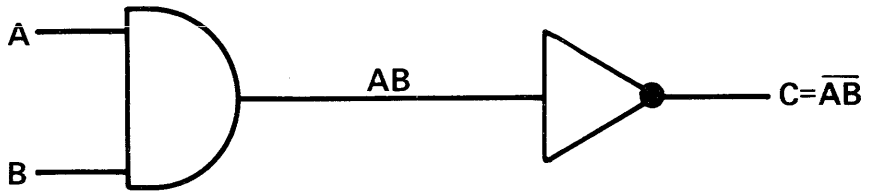
A	B	OUTPUT C=A•B
0	0	0
1	0	0
0	1	0
1	1	1

THE "NAND" GATE

PROVIDES THE LOGICAL OPERATION "NOT-AND"



THE OUTPUT "C" IS FALSE (NOT) ONLY WHEN BOTH "A" AND "B" ARE TRUE.
THUS THE "NAND" GATE CAN BE CONSIDERED AN "AND" GATE
FOLLOWED BY AN INVERTER.

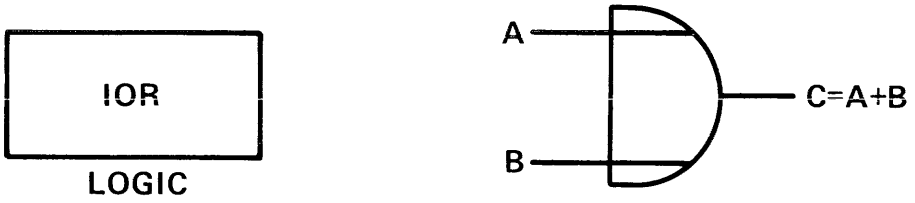


TRUTH TABLE

A	B	C
0	0	1
1	0	1
0	1	1
1	1	0

THE "IOR" GATE (INCLUSIVE "OR")

"IOR" PROVIDES LOGICAL ADDITION



THE OUTPUT "C" IS TRUE (POSITIVE) WHEN
"A" OR "B" OR "BOTH" ARE TRUE (POSITIVE).

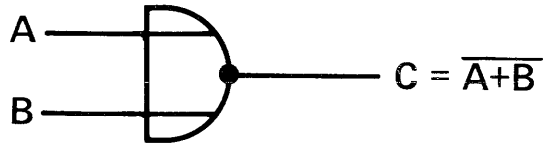
THE ARITHMETIC ADDITION SIGN (+) ALWAYS INDICATES "OR" IN LOGIC EQUATIONS AND CIRCUITS. THE TERM C=A+B IS A LOGIC EQUATION MEANING EITHER "A" OR "B", OR "BOTH", MUST BE TRUE FOR "C" TO BE TRUE.

TRUTH TABLE

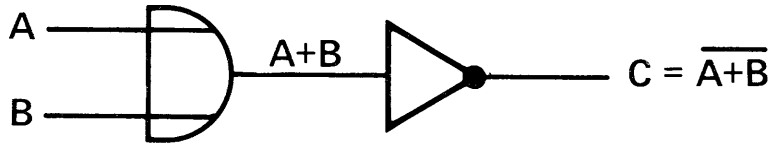
A	B	OUTPUT C=A+B
0	0	0
1	0	1
0	1	1
1	1	1

THE "NOR" GATE

THE "NOR" GATE PROVIDES THE LOGICAL OPERATION "NOT-OR"



THE OUTPUT "C" IS FALSE (NOT) ONLY WHEN EITHER "A" OR "B" ARE TRUE. THUS THE "NOR" GATE CAN BE CONSIDERED AS AN OR GATE FOLLOWED BY AN INVERTER.

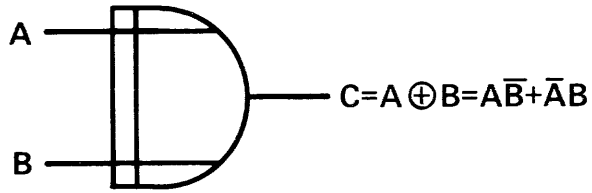


TRUTH TABLE

A	B	C
0	0	1
1	0	0
0	1	0
1	1	0

THE "XOR" GATE (EXCLUSIVE "OR")

PROVIDES THE LOGICAL OPERATION "EXCLUSIVE OR"



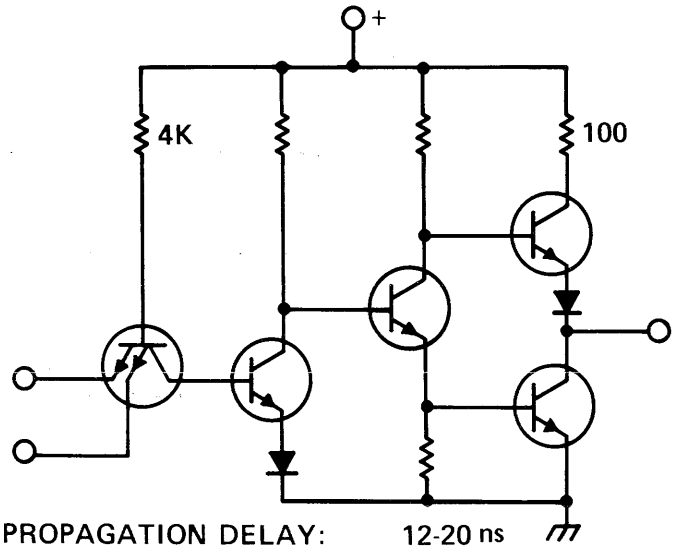
THE OUTPUT "C" IS TRUE (POSITIVE) ONLY
WHEN "A" OR "B" BUT NOT BOTH ARE TRUE.

TRUTH TABLE

A	B	C=AB̄+ĀB
0	0	0
1	0	1
0	1	1
1	1	0

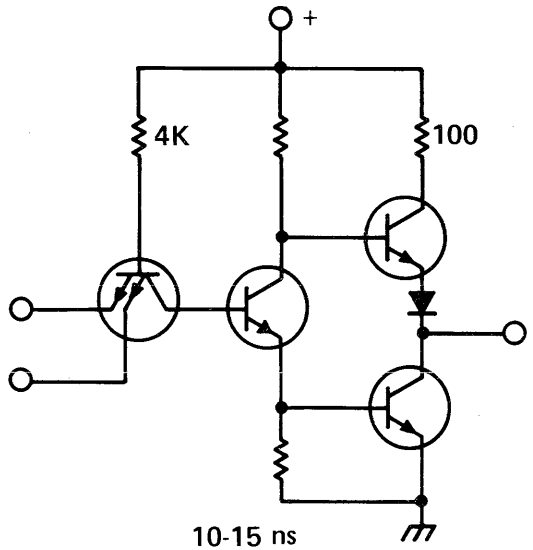
COMPARISON OF COMMON TTL AND CTL GATES

TTL: AND-GATE



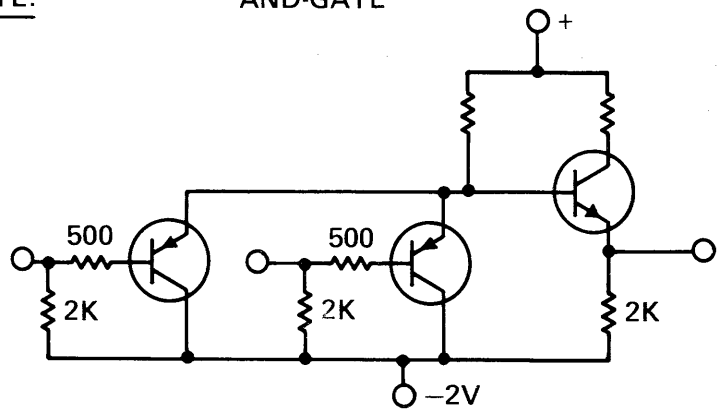
TYP. PROPAGATION DELAY: 12-20 ns
POWER REQUIREMENT: 10mW

NAND-GATE



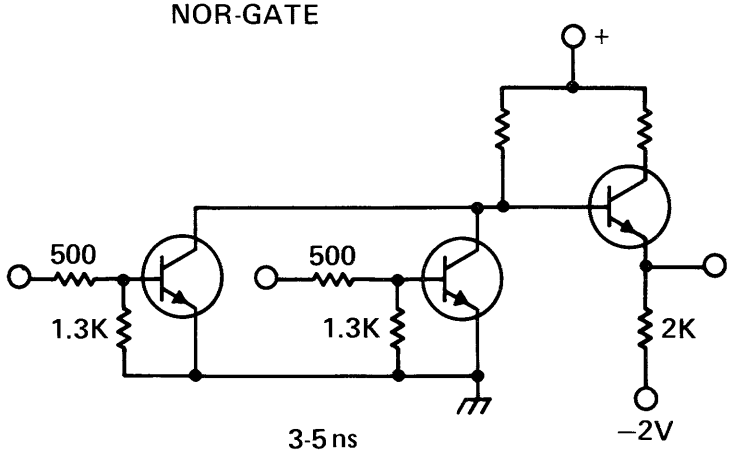
10-15 ns
10mW

CTL: AND-GATE



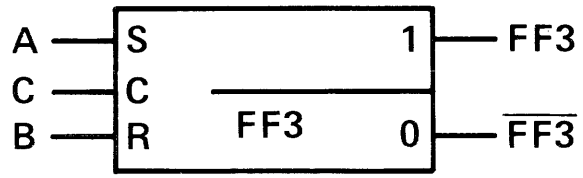
TYP. PROPAGATION DELAY: 3-5 ns
POWER REQUIREMENT: 250mW

NOR-GATE



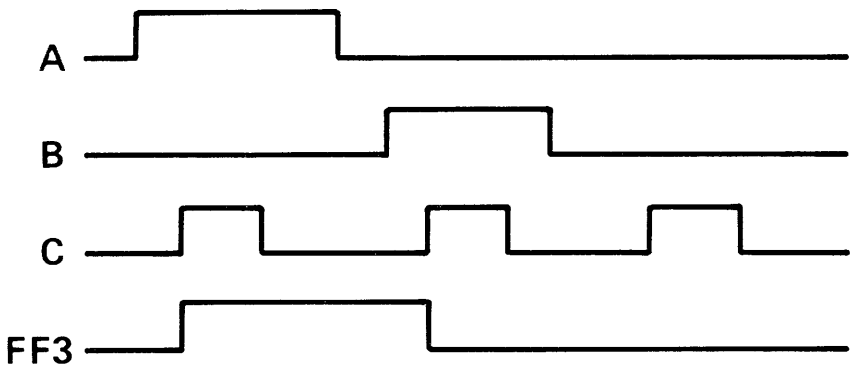
3-5 ns
250mW

THE R-S FLIP FLOP WITH CLOCK



"A" AND "C" ARE REQUIRED TO SET (FF3 TRUE)

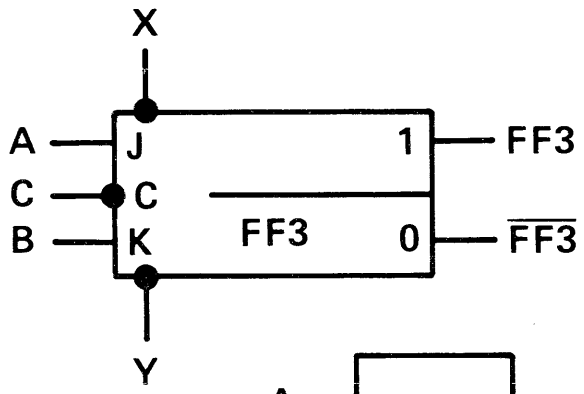
"B" AND "C" ARE REQUIRED TO RESET ($\overline{FF3}$ TRUE)



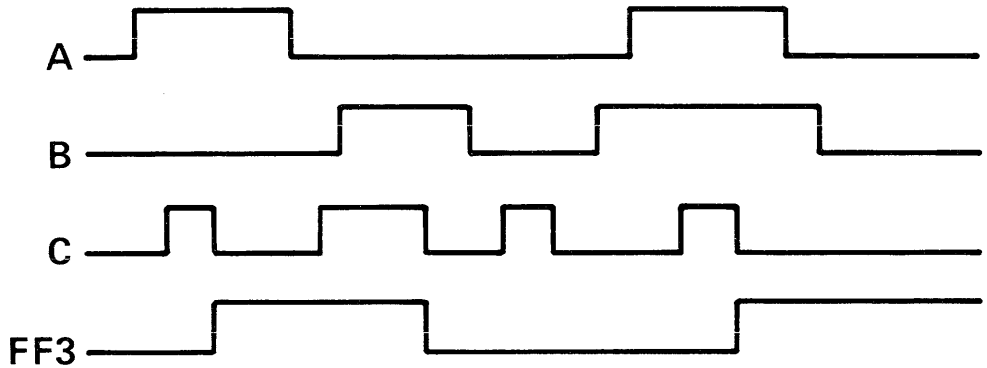
TRUTH TABLE

INPUT		OUTPUT	
A	B	FF3	$\overline{FF3}$
1	0	1	0
0	1	0	1
0	0	NO CHANGE	
1	1	UNDETERMINED	

THE J-K FLIP FLOP



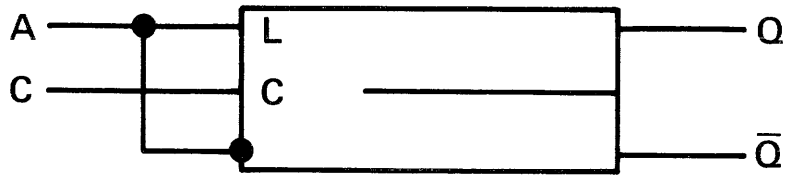
SIMULTANEOUS INPUTS "A" AND "B" REVERSE THE EXISTING STATE FF3 → $\overline{FF3}$. DIRECT SET (X) OR DIRECT RESET (Y) WITHOUT CLOCK



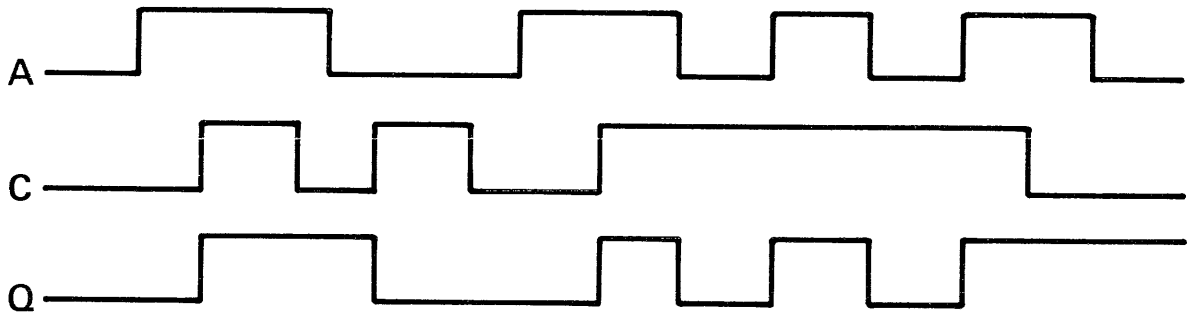
TRUTH TABLE

INPUTS		INITIAL STATE		FINAL STATE	
A	B	FF3	$\overline{FF3}$	FF3	$\overline{FF3}$
1	0	ANY		1	0
0	1			0	1
1	1	0	1	1	0
1	1	1	0	0	1
0	0	ANY		NO CHANGE	

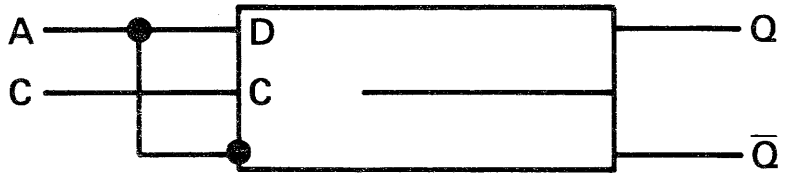
LATCHING FLIP FLOP



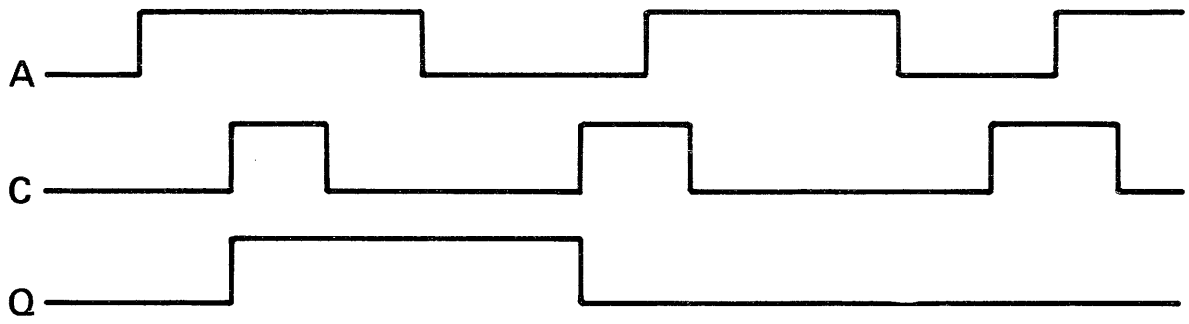
THIS FLIP FLOP WILL FOLLOW INPUT "A" AS LONG AS THE CLOCK INPUT (C) IS TRUE. THE FLIP-FLOP WILL LATCH INTO THE STATE EXISTING AT"'A" WHEN THE CLOCK PULSE AT C TRANSITS FROM TRUE TO FALSE.



DELAY FLIP FLOP

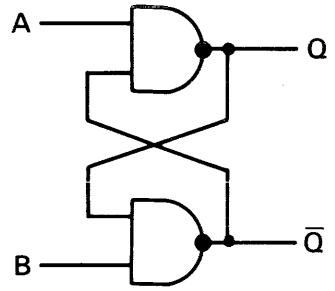


THE CONDITION PRESENT AT THE INPUT "A" WILL BE LOADED INTO THE FLIP FLOP AT THE POSITIVE GOING CLOCK PULSE. CONTRARY TO THE L-FLIP FLOP THE DELAY FLIP FLOP WILL NOT CHANGE ITS STATE WHEN THE CLOCK IS IN A STEADY STATE CONDITION.

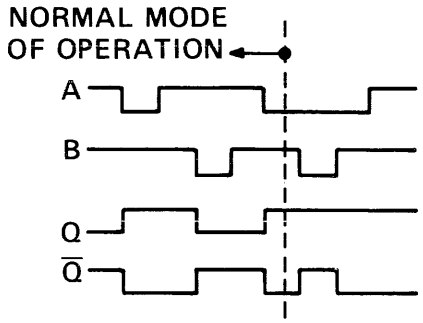


CROSS COUPLED "NAND" AND "NOR" GATES

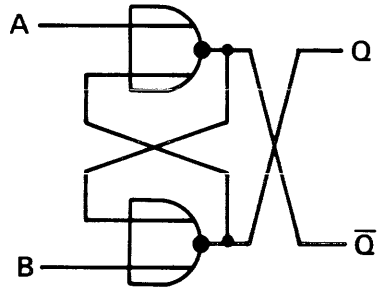
CROSS COUPLED "NAND" GATES:



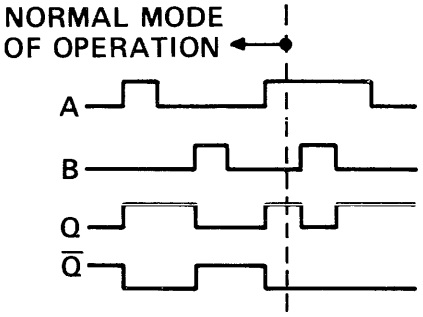
NORMALLY THE TWO INPUTS A & B ARE HIGH. WHEN THE "A" INPUT GOES LOW, THE FLIP FLOP SETS. WHEN "B" GOES LOW, THE FLIP FLOP CLEARS.



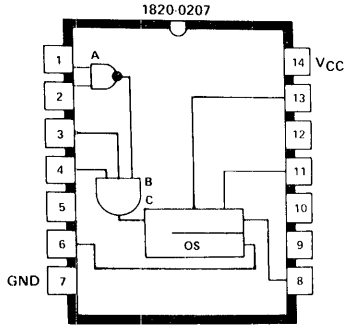
CROSS COUPLED "NOR" GATES:



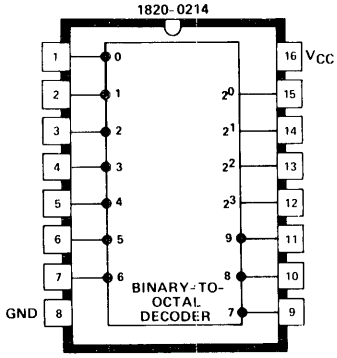
NORMALLY THE TWO INPUTS "A" & "B" ARE LOW. WHEN THE "A" INPUT GOES HIGH THE FLIP FLOP WITH THE CROSSED OUTPUT (TO ALIGN THE SET AND CLEAR OUTPUT WITH THE INPUTS) SETS. WHEN "B" GOES HIGH THE FLIP FLOP CLEARS.



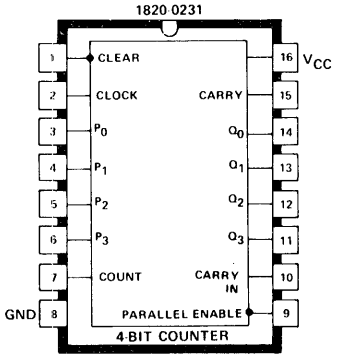
1820-0207



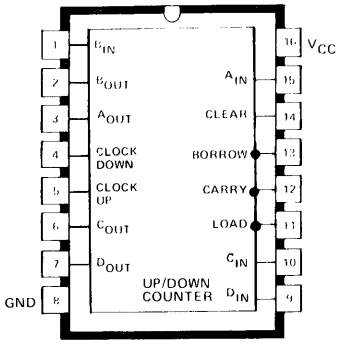
1820-0214



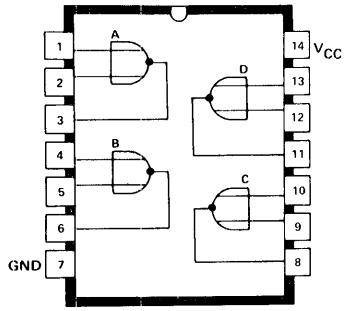
1820-0231



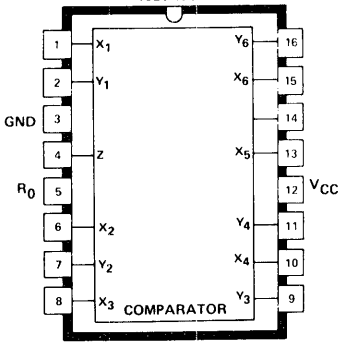
1820-0233



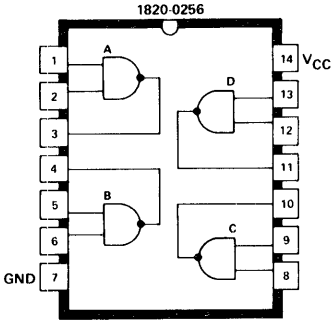
1820-0239



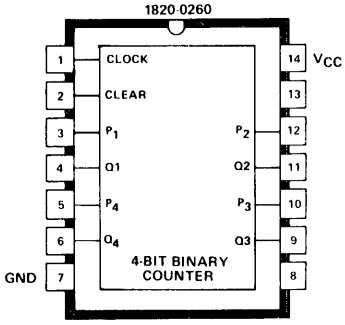
1820-0250



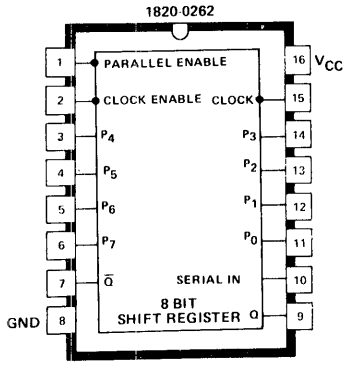
1820-0256



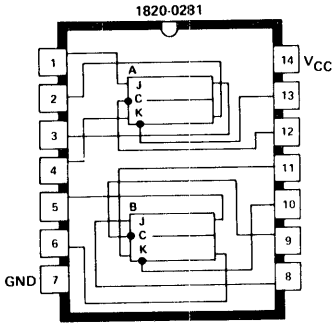
1820-0260



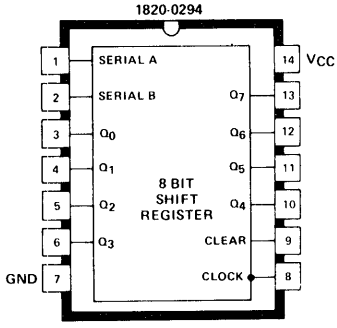
1820-0262



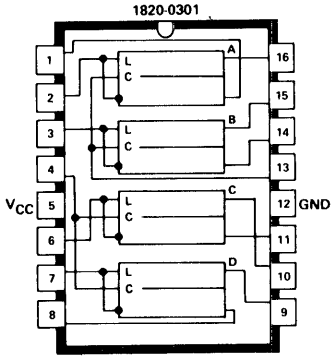
1820-0281



1820-0294



1820-0301



INTEGRATED CIRCUIT DIAGRAMS

BOOLEAN ALGEBRA IS

A SET OF LOGICAL PROPOSITIONS SIMILAR TO ORDINARY ALGEBRA WHICH IS A SET OF MATHEMATICAL PROPOSITIONS.

MATHEMATICAL SYMBOLOGY			BOOLEAN SYMBOLOGY		
1	=	ABSOLUTE NUMBER	1	=	YES (TRUE)
ø	=	ZERO	ø	=	NO (FALSE)
$\overline{A}$	=	LENGTH	$\overline{A}$	=	"NOT A"
$\longrightarrow$	=	APPROACHES LIMIT	$\longrightarrow$	=	REPLACES
•	=	MULTIPLY	•	=	"AND"
+	=	ADD	+	=	"OR"
( )	=	INNER PRODUCT	( )	=	INNER "AND"
[ ]	=	AGGREGATE PRODUCT	[ ]	=	AGGREGATE "AND"
A+B	=	ADD A AND B	A+B	=	A "OR" B
AB	=	MULTIPLY A BY B	AB	=	A "AND" B

EIGHT BASIC IDENTITIES

THESE HOLD FOR BOTH ORDINARY AND BOOLEAN ALGEBRA.

1. $0 + A = A$
2. $0 \bullet A = 0$
3. $1 \bullet A = A$
4. $A + B = B + A$
5. $AB = BA$
6. $A + (B + C) = (A + B) + C$
7. $A\ (BC) = AB(C)$
8. $A\ (B + C) = AB + AC$

ALL THESE EXPRESSIONS MAY BE CONSTRUCTED USING BASIC "AND–OR" GATES.

TEN IDENTITIES UNIQUE TO BOOLEAN ALGEBRA

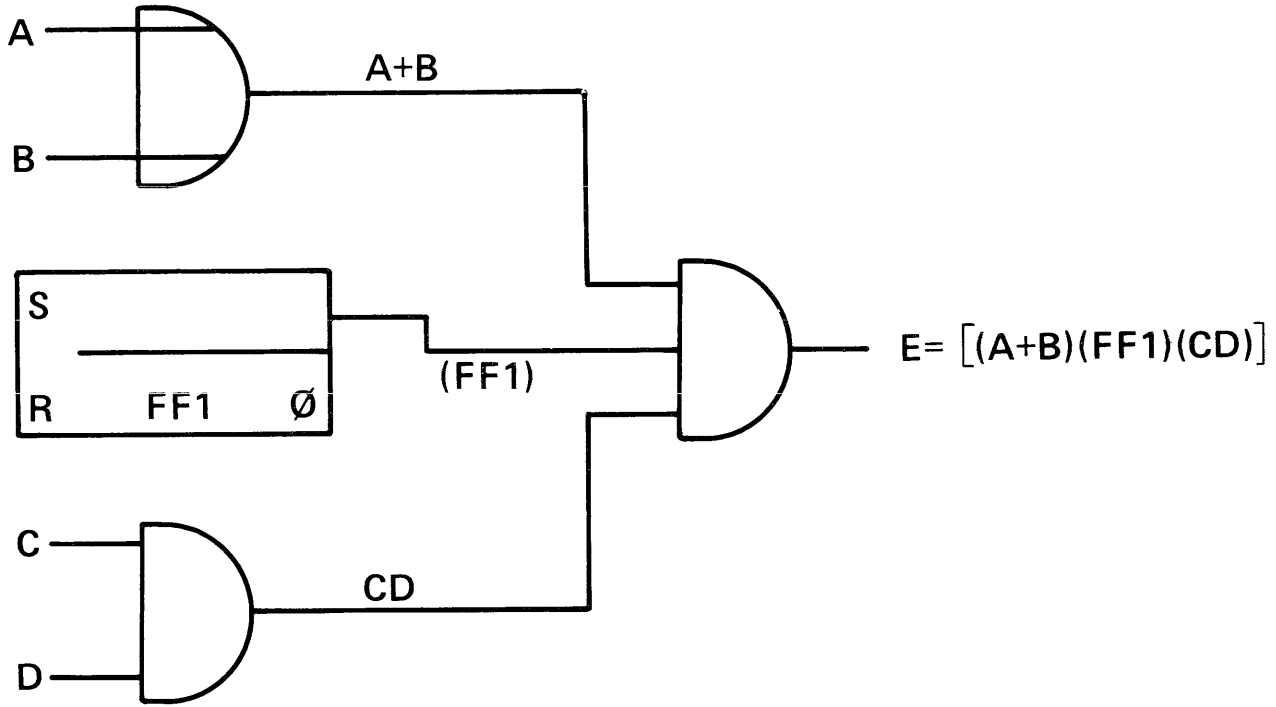
WHICH MAKES SIMPLIFICATION OF COMPLEX CIRCUITRY ECONOMICALLY FEASIBLE.

1. $A + A = A$
2. $A \bullet A = A$
3. $A + 1 = 1$
4. $A \bullet \overline{A} = \emptyset$
5. $A + \overline{A} = 1$
6. $\overline{A + B} = \overline{A} \bullet \overline{B}$
7. $\overline{AB} = \overline{A} + \overline{B}$
8. $A + AB = A$
9. $A\,(A + B) = A$
10. $A + BC = (A + B)\,(A + C)$

ALL THESE EXPRESSIONS MAY BE CONSTRUCTED USING BASIC "NOT", "AND", "OR" "NOR", AND "NAND" GATES.

THE LOGIC EQUATION

COMPLETELY DEFINES A LOGIC CIRCUIT



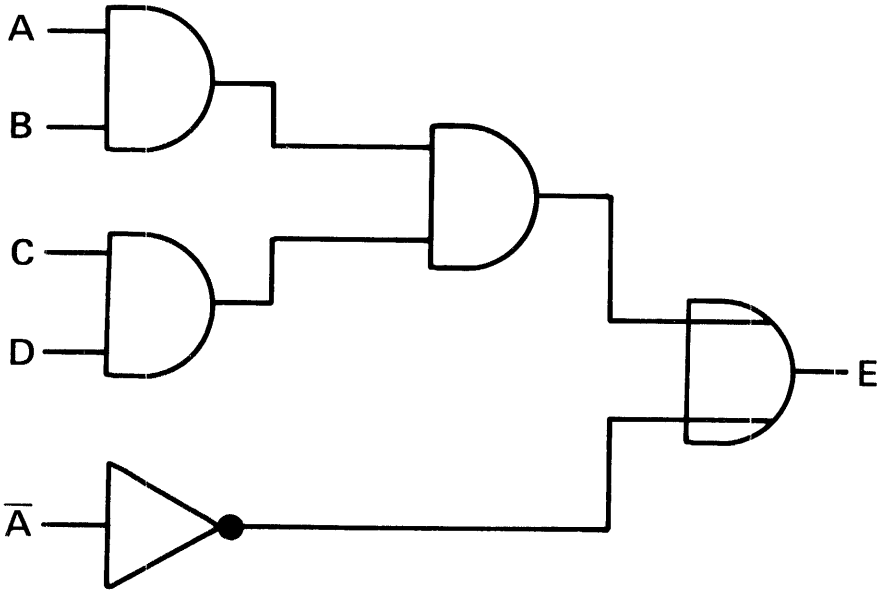
IF "E" IS NOT TRUE, THE CONTROL FUNCTIONS A,B,FF1,C AND D MAY BE CHECKED TO ISOLATE A FAULTY COMPONENT.

PROBLEMS

1. REDUCE TO SIMPLEST FORM USING BOOLEAN IDENTITIES:

   A (BC + AC)

2. DETERMINE E FROM THE FOLLOWING CIRCUIT AND REDUCE TO THE SIMPLEST FORM:



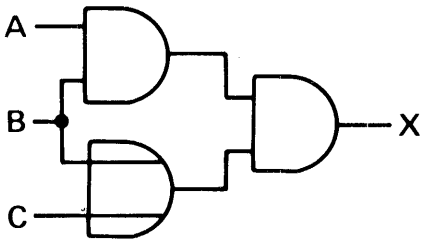
3. CONSTRUCT A SIMPLE CIRCUIT FROM THE FOLLOWING LOGICAL EQUATION:

   E = AB (C + D) + FF1

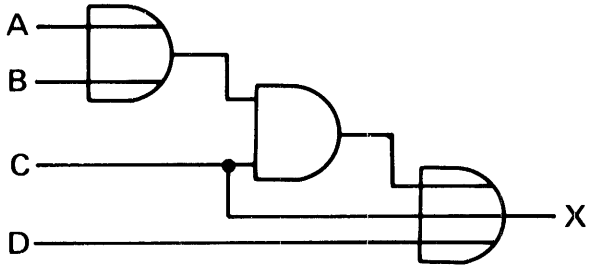
LOGIC EQUATIONS

WRITE THE LOGIC EQUATIONS FOR X, REDUCE IT TO THE SIMPLEST FORM AND DRAW AN EQUIVALENT CIRCUIT.

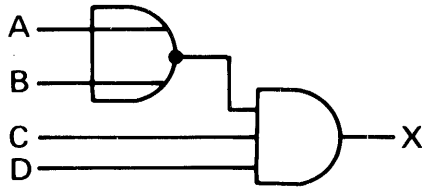
1.



2.

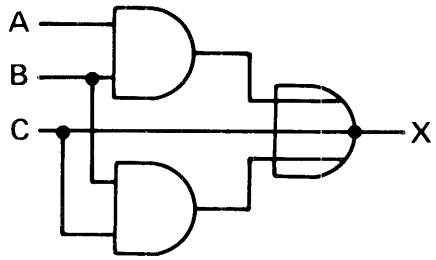


3.

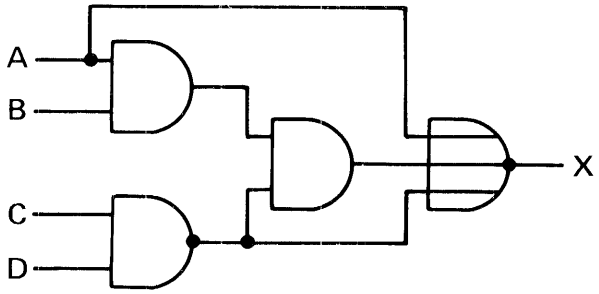


LOGIC EQUATIONS (CONT.)

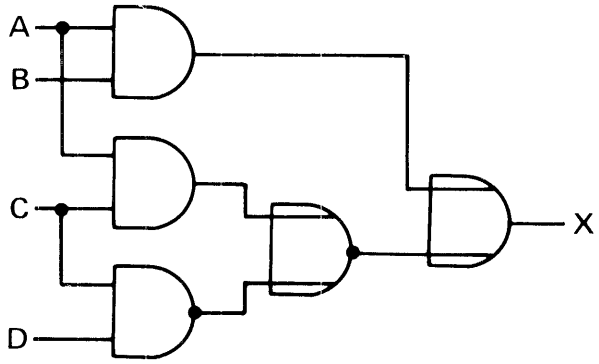
4.



5.



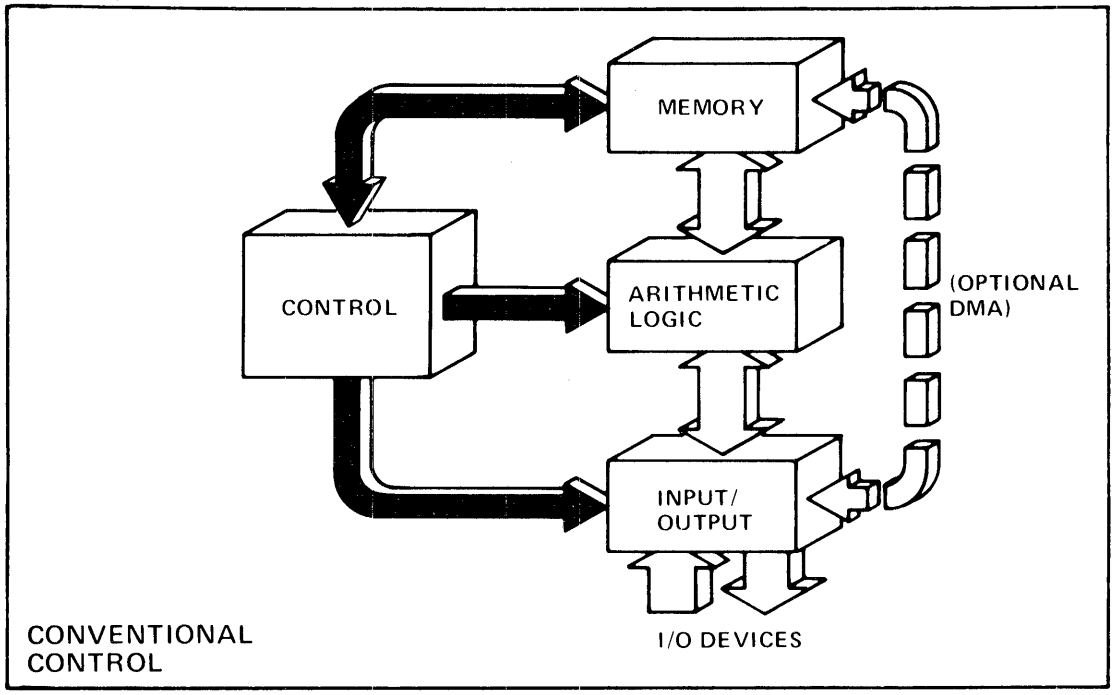
6.

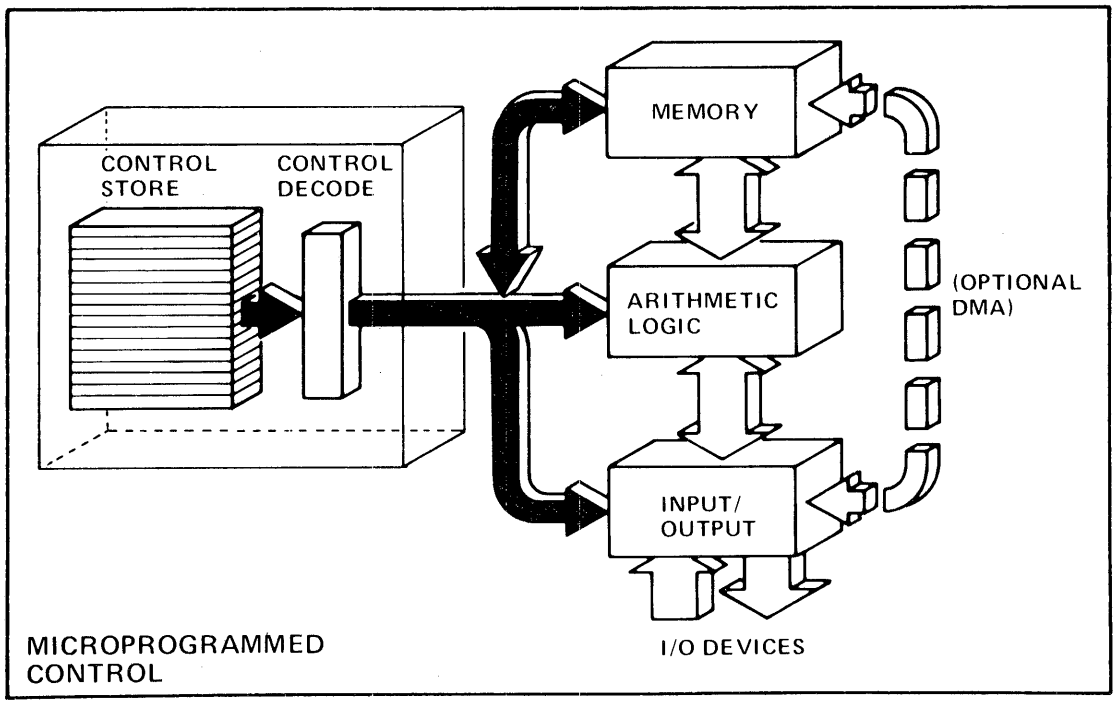


2100A architecture and block diagrams 2

LESSON 2

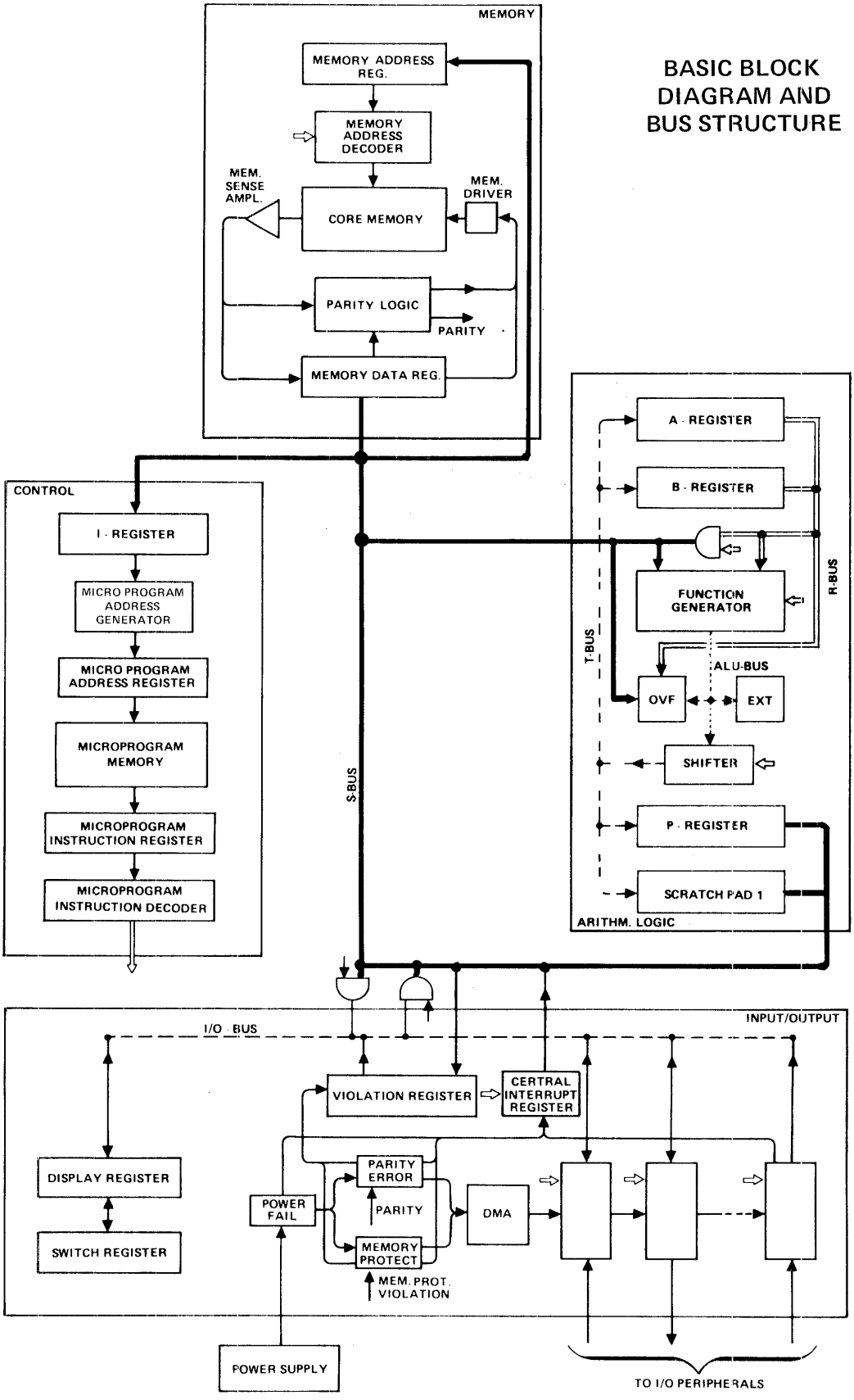
2100A ARCHITECTURE AND BLOCK DIAGRAMS



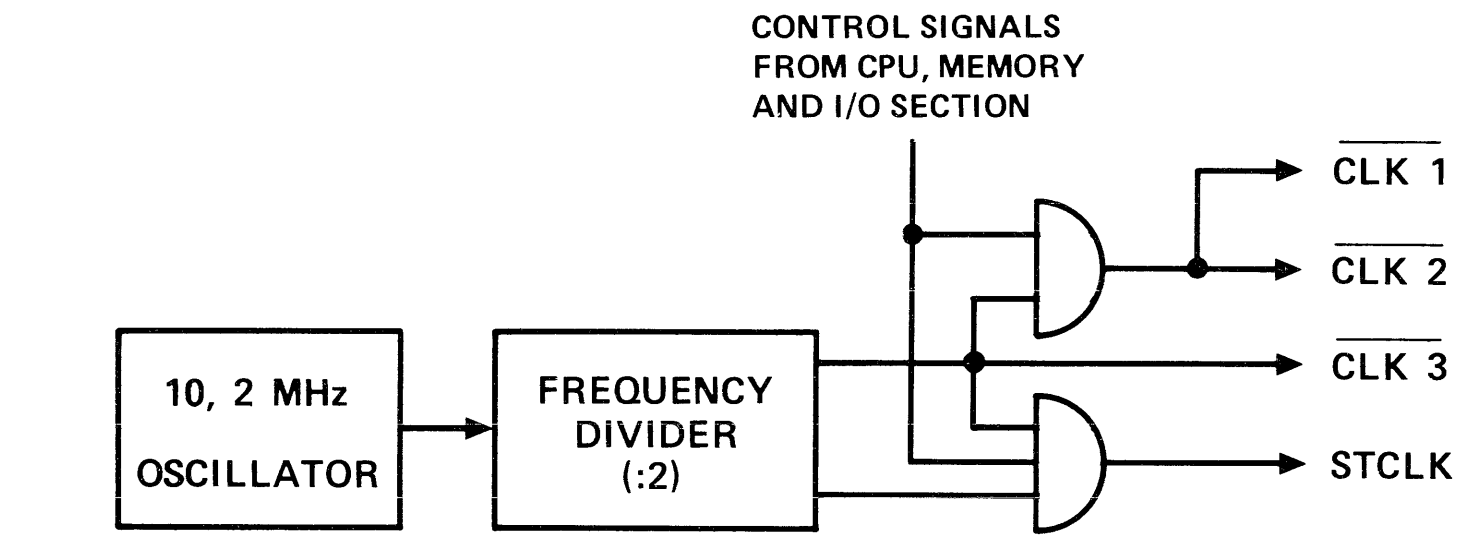


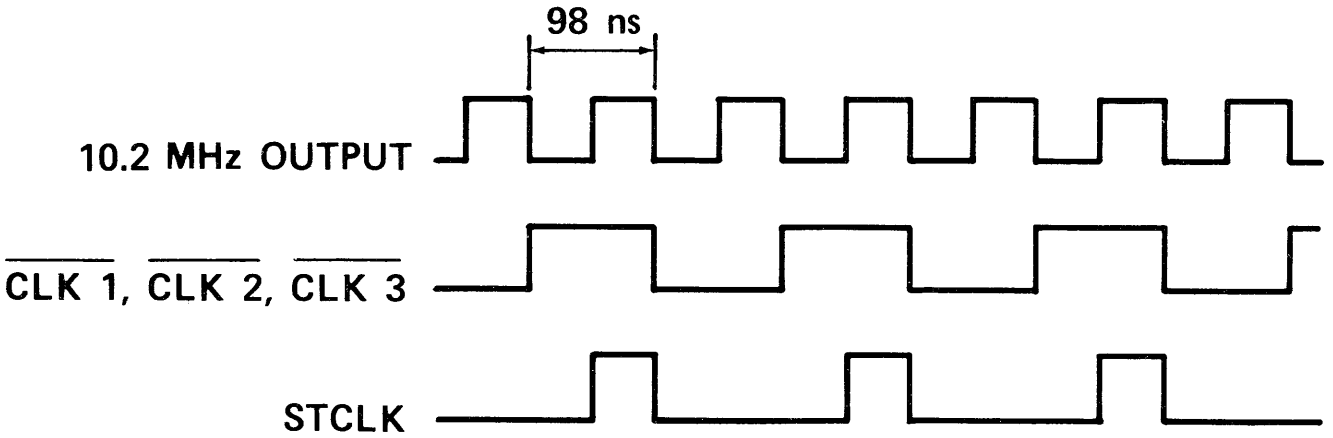
COMPARISON BETWEEN GENERAL ARCHITECTURE OF STANDARD AND ROM CONTROLED COMPUTERS

BASIC BLOCK DIAGRAM AND BUS STRUCTURE



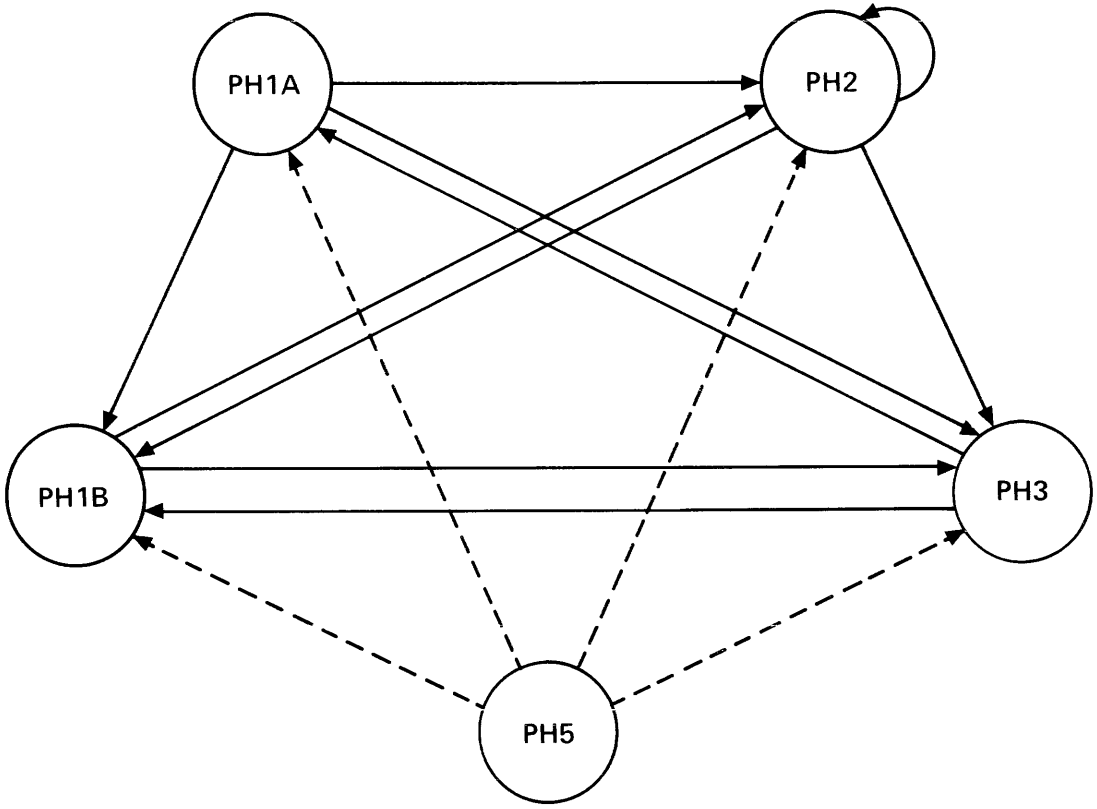
CLOCK GENERATOR

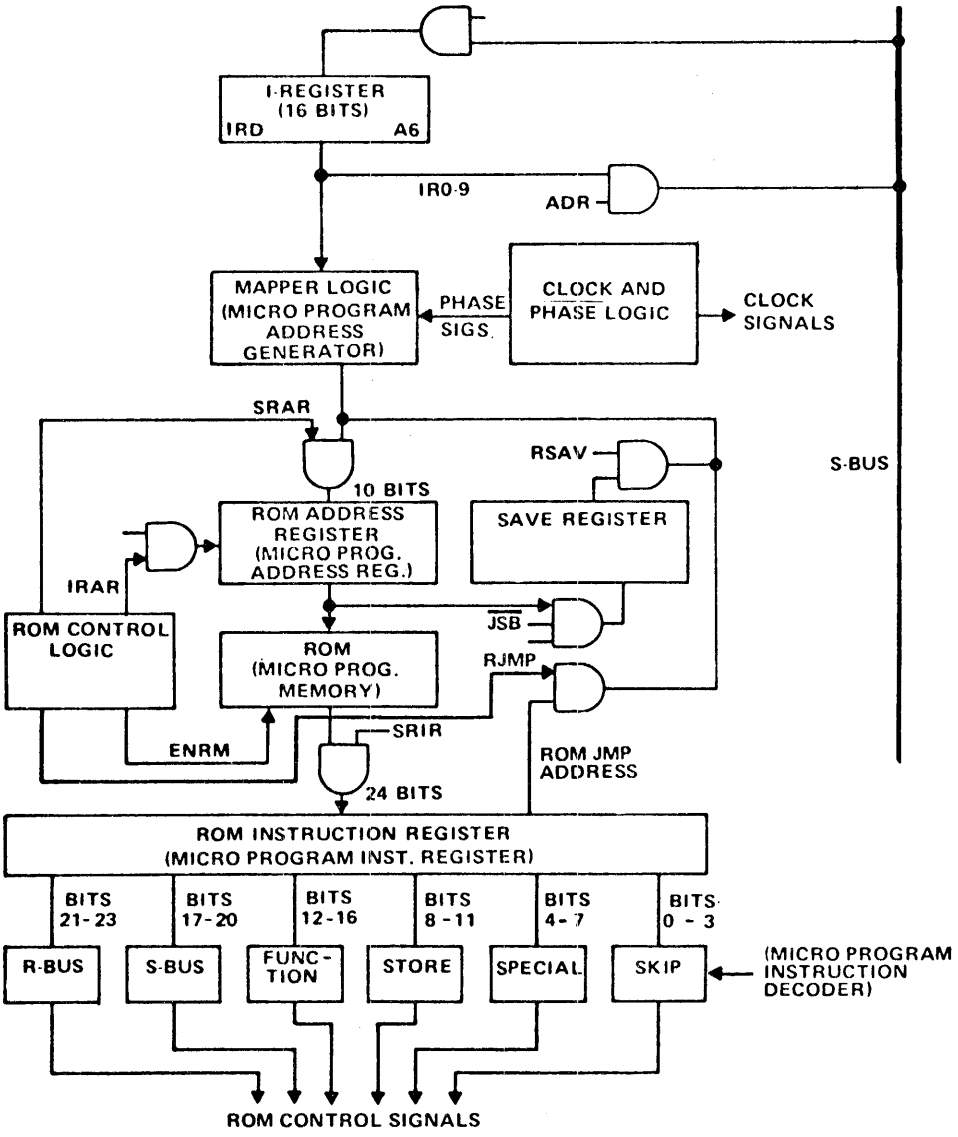




PHASE CONTROL

- AT ANY GIVEN TIME THE COMPUTER WILL BE IN ONE PHASE ONLY.
- ONE SFECIFIC MICROORDER WILL SPECIFY THAT THE NEXT MICROINSTRUCTION WILL BE THE LAST ONE OF THE PRESENT PHASE. IF THIS CONDITION IS REACHED THE PHASE LOGIC WILL ISSUE ONE OF FOUR "SET PHASE SIGNALS" WHICH IS REQUIRED TO CHANGE ANY PHASES.





BLOCK DIAGRAM OF CONTROL UNIT

MICROPROGRAMMING

A BINARY CODE IS ASSIGNED TO ALL OF THE CONTROL SIGNALS IN THE COMPUTER AND THE REQUIRED CODES ARE STORED IN ROM TO BE ACTIVATED BY THE MACHINE INSTRUCTIONS.

HORIZONTAL MICROPROGRAMMING USES A WIDE ROM WORD, AND EACH ROM BIT IS ASSIGNED A GATE.

THE 2100A USES VERTICAL MICROPROGRAMMING WHERE THE ROM WORD IS SHORT AND DECODERS ARE USED TO BREAK DOWN THE BINARY CODES TO THE GATE LEVEL.

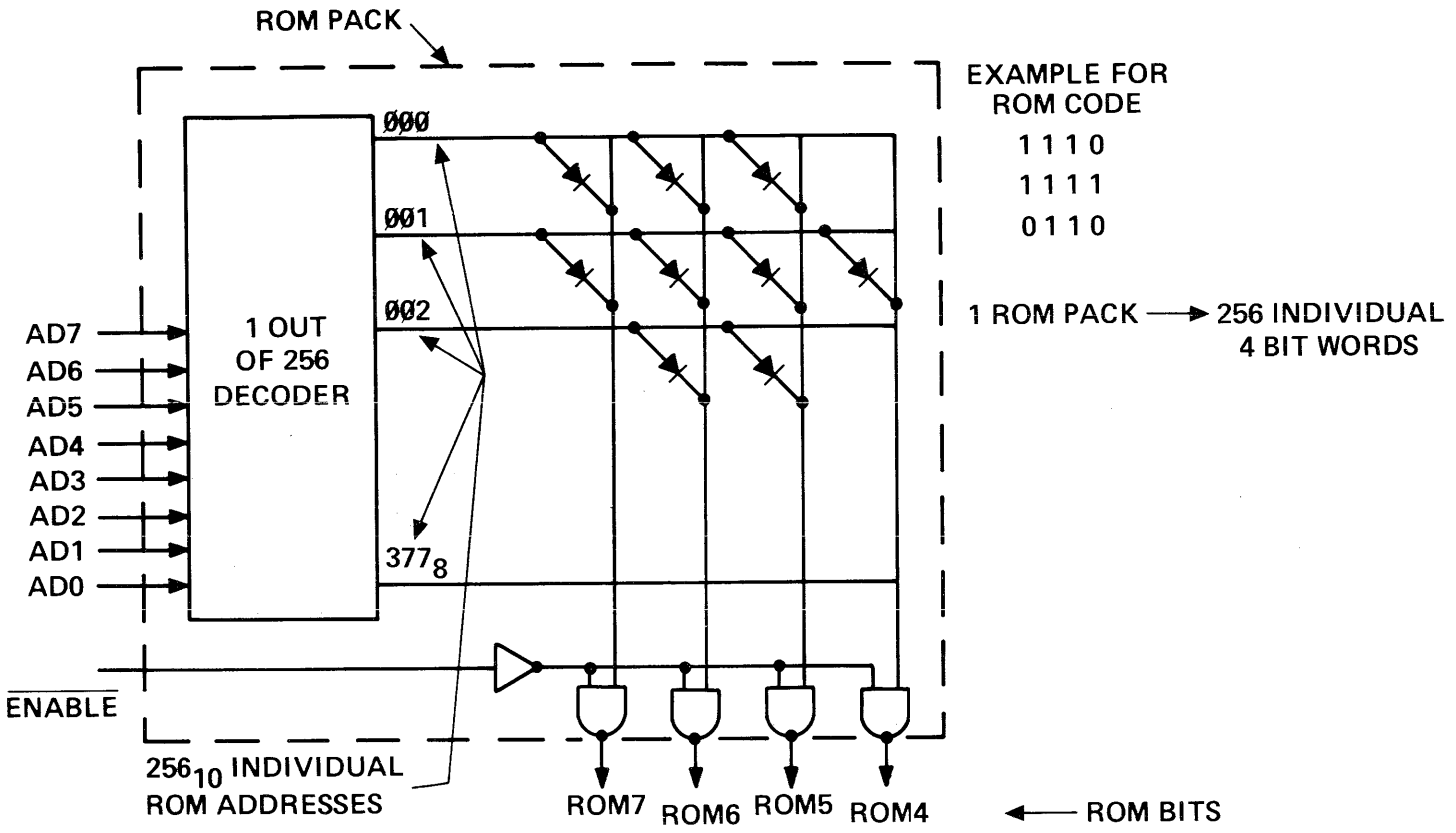
ROM ADDRESS MAPPER

THE ROM ADDRESS MAPPER DECODES THE CURRENT MACHINE INSTRUCTION AND PHASE LOGIC OUTPUT TO PROVIDE THE STARTING ADDRESS IN THE ROM MICROPROGRAM FOR THE NEXT PHASE. THIS ADDRESS IS LOADED INTO THE ROM ADDRESS REGISTER (RAR).

ROM ADDRESS REGISTER

THE ROM ADDRESS REGISTER IS A 10 BIT REGISTER WITH PARALLEL INPUT. THE REGISTER CAN BE INCREMENTED. IT ALWAYS HOLDS THE ADDRESS OF THE NEXT ROM MICROINSTRUCTION TO BE EXECUTED.

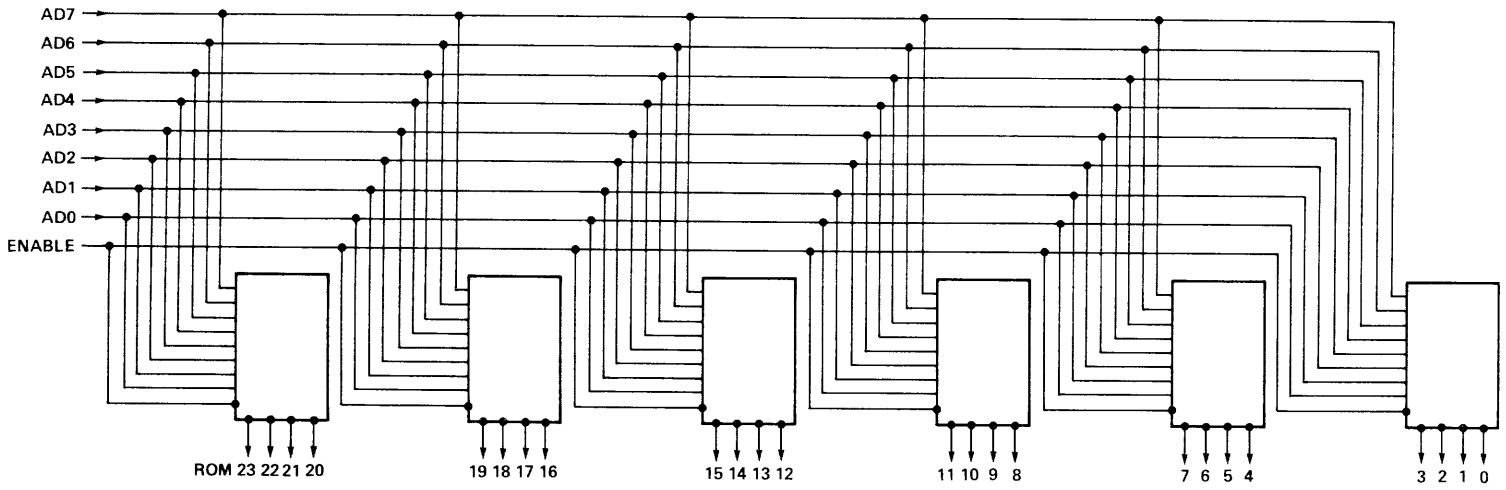
THE ROM PACK



THE 8 BINARY ROM ADDRESS INPUTS ARE DECODED INTO 256_{10} INDIVIDUAL ROM ADDRESSES (256_{10} ROM WORDS)

EACH ROM ADDRESS CREATES A PREDEFINED 4-BIT ROM WORD OUTPUT. THE ROM ACCESS TIME IS $<$ 75ns.

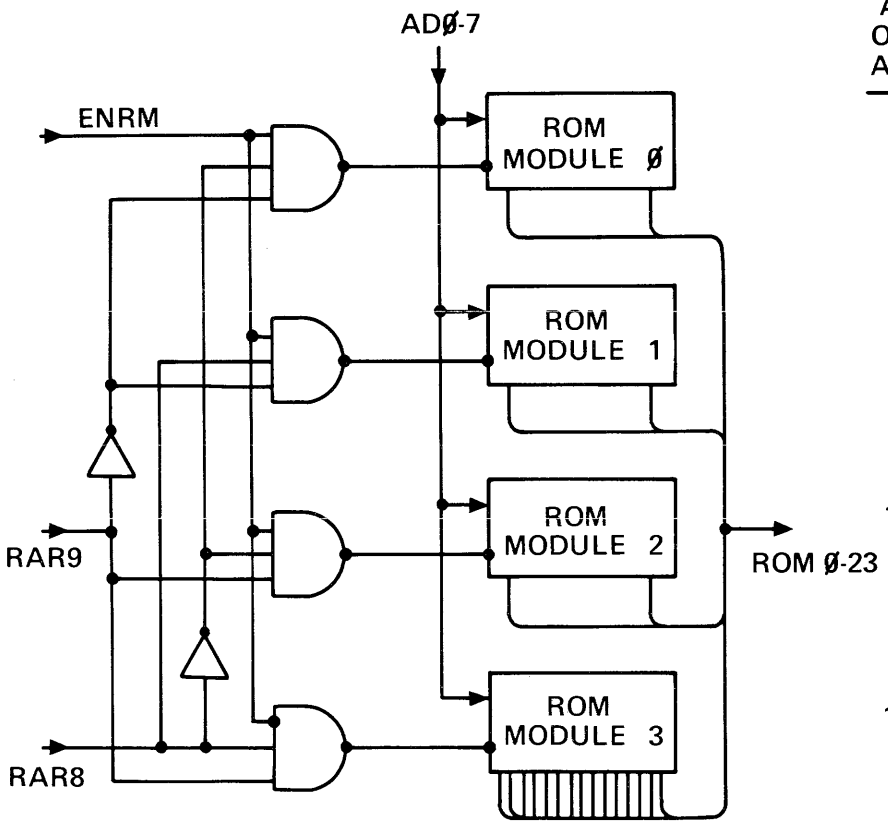
THE 24 BIT ROM



6 INDIVIDUALLY BURNED ROM PACKS WITH THE 8-BINARY ADDRESS INPUTS CONNECTED IN PARALLEL FORM THE 24-BIT ROM WORD.

256 DIFFERENT 24-BIT ROM WORDS IS CONSIDERED TO BE ONE MODULE.

THE ROM MODULES



ABSOLUTE OCTAL ROM ADDRESSES	PURPOSE
0-377	ACCOMODATES THE STANDARD HP2100A MICROINSTRUCTION SET IN 6 ROM PACKS
400-777	ACCOMODATES USUALLY THE FLOATING POINT OPTION HP12901A IN 6 ROM PACKS
1000-1377	ACCOMODATES CUSTOM BUILT ROM, PROM OR WCS
1400-1777	ACCOMODATES CUSTOM BUILT ROM, PROM OR WCS

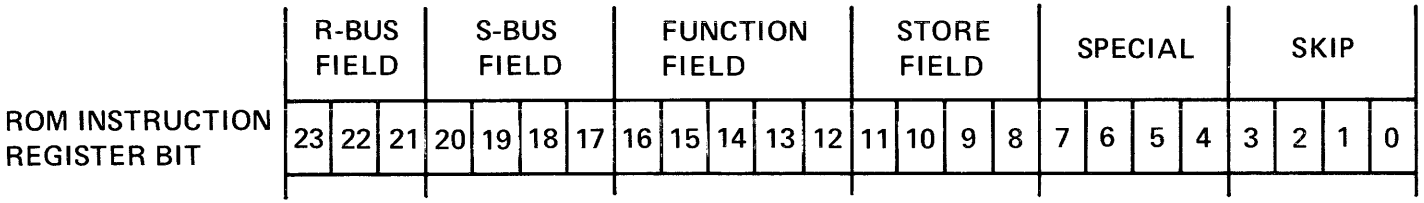
- ANY COMBINATION OF MODULES IS PERMISSIBLE. HOWEVER, MODULE 0 HAS TO BE PRESENT.

ROM INSTRUCTION REGISTER

- THE ROM INSTRUCTION REGISTER (RIR) PROVIDES A 24 BIT BUFFER STORAGE FOR THE ROM MICROINSTRUCTION.
- IT TRANSFERS THE MICROINSTRUCTION PARALLEL TO THE ROM INSTRUCTION DECODER.
- THE ROM INSTRUCTION REGISTER ALLOWS THE CURRENT MICRO-INSTRUCTION TO BE EXECUTED WHILE THE NEXT ONE IS ACCESSED IN ROM; THEREBY REDUCING EXECUTION TIME. THE RIR HOLDS THE CONTENTS OF THE PREVIOUSLY HELD ADDRESS IN THE RAR.

ROM INSTRUCTION DECODER

- THE ROM INSTRUCTION DECODER DECIPHERS THE CONTENTS OF THE ROM INSTR. REG.
- THE ROM INSTRUCTION DECODER IS DIVIDED INTO 6 SECTIONS.
- EACH SECTION DECODES ONE OF THE 6 FIELDS:



R-BUS FIELD: SPECIFIES THE SOURCE OF THE INFORMATION TO BE READ ONTO THE R-BUS.

S-BUS FIELD: SPECIFIES THE SOURCE OF THE INFORMATION TO BE READ ONTO THE S-BUS.

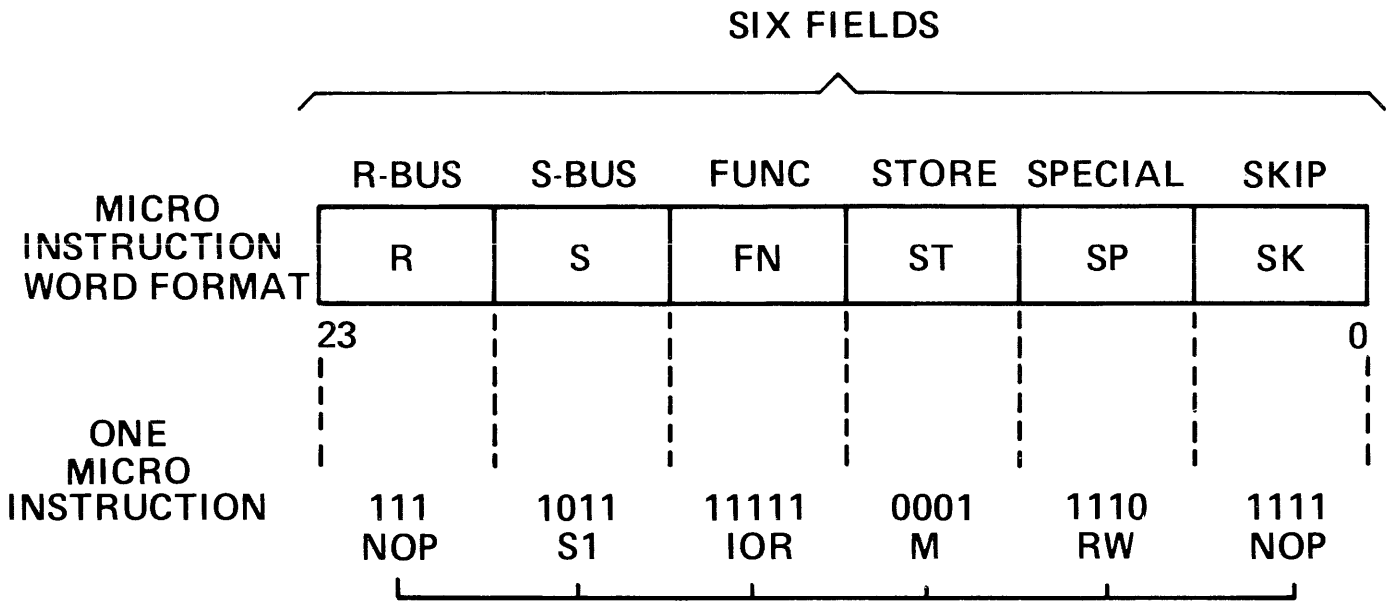
FUNCTION FIELD: SPECIFIES THE ARITHMETIC- OR LOGIC FUNCTIONS, BIT SHIFTING IN REGISTERS OR SPECIAL CONTROL FUNCTIONS WHEN COMBINING R- AND S-BUS CONTENTS ONTO THE T-BUS. CONDITIONAL AND UNCONDITIONAL ROM JMPS OR -JSB CAN ALSO BE DECODED.

STORE FIELD: SPECIFIES THE DESTINATION OF THE INFORMATION ON THE T- OR S-BUS.

SPECIAL FIELD: ENABLES ALTER SKIP, SHIFT-ROTATE AND INPUT/OUTPUT GROUP INSTRUCTIONS, INITIATES MEMORY READ/WRITE OR CLEAR/WRITE OPERATIONS AND SPECIFIES SOME SPECIAL FUNCTIONS.

SKIP FIELD: DECODES CONDITIONAL AND UNCONDITIONAL ROM SKIPS.

MICROPROGRAM DEFINITIONS



A MICROINSTRUCTION (24 BITS LONG) CONSISTS OF 6 MICRO-ORDERS
A MICROPROGRAM CONSISTS OF SEVERAL MICROINSTRUCTIONS.
EACH MICRO-PROGRAM HAS AN ENTRY POINT LABEL.

MICROPROGRAM	R	S	FN	ST	SP	SK	COMMENTS
ENTRY POINT LABEL → ADD	NOP	S1	IOR	M	RW	NOP	FETCH OPERAND
	NOP	T	IOR	S2	NOP	EOP	PUT IN SP2
	A	S2	ADD	A	NOP	NOP	ADD TO A-REG

THREE MICROINSTRUCTIONS

MICROINSTRUCTION CODING

CODE 5-Bit Field / 4-Bit Field / 3-Bit Field	R-Bus Field 3 Bits	S-Bus Field 4 Bits	Function Field 5 Bits	Store Field 4 Bits	Special Field 4 Bits	Skip Field 4 Bits
1 1 1 1 1	NOP	NOP	IOR	NOP	NOP	NOP
1 1 1 1 0	CO	P	SOV	A	RW	UNC
1 1 1 0 1	AAB	CL	CLO	B	IOG1	EOP
1 1 1 0 0	CAB	CR	SFLG	AAB	CW	NAAB
1 1 0 1 1	F	S1	CFLG	CAB	ASG2	AAB
1 1 0 1 0	Q	S2	LWF	Q	ASG1	NMPV
1 1 0 0 1	B	S3	***	F	ECYN	CTR
1 1 0 0 0	A	S4	ARS	P	ECYZ	CTRI
1 0 1 1 1		COND	CRS	S1	LEP	TBZ
1 0 1 1 0		ADR	LGS	S2	AAB	FLG
1 0 1 0 1		CNTR	RSB	S3	SRG2	OVF
1 0 1 0 0		RRS	CJMP	S4	SRG1	COUT
1 0 0 1 1		M	JMP	IR	CNTR	NEG
1 0 0 1 0		T	JMP	T	R1	ODD
1 0 0 0 1		IOI	JSB	M	L1	RPT
1 0 0 0 0		CIR	JSB	IOO	RSS	ICTR
0 1 1 1 1			**			
0 1 1 1 0			XOR			
0 1 1 0 1			NOR			
0 1 1 0 0			AND			
0 1 0 1 1			ADD			
0 1 0 1 0			ADDO			
0 1 0 0 1			INC			
0 1 0 0 0			INCO			
0 0 1 1 1			**			
0 0 1 1 0			DEC			
0 0 1 0 1			SUB			
0 0 1 0 0			DIV			
0 0 0 1 1			MPY			
0 0 0 1 0			PIA			

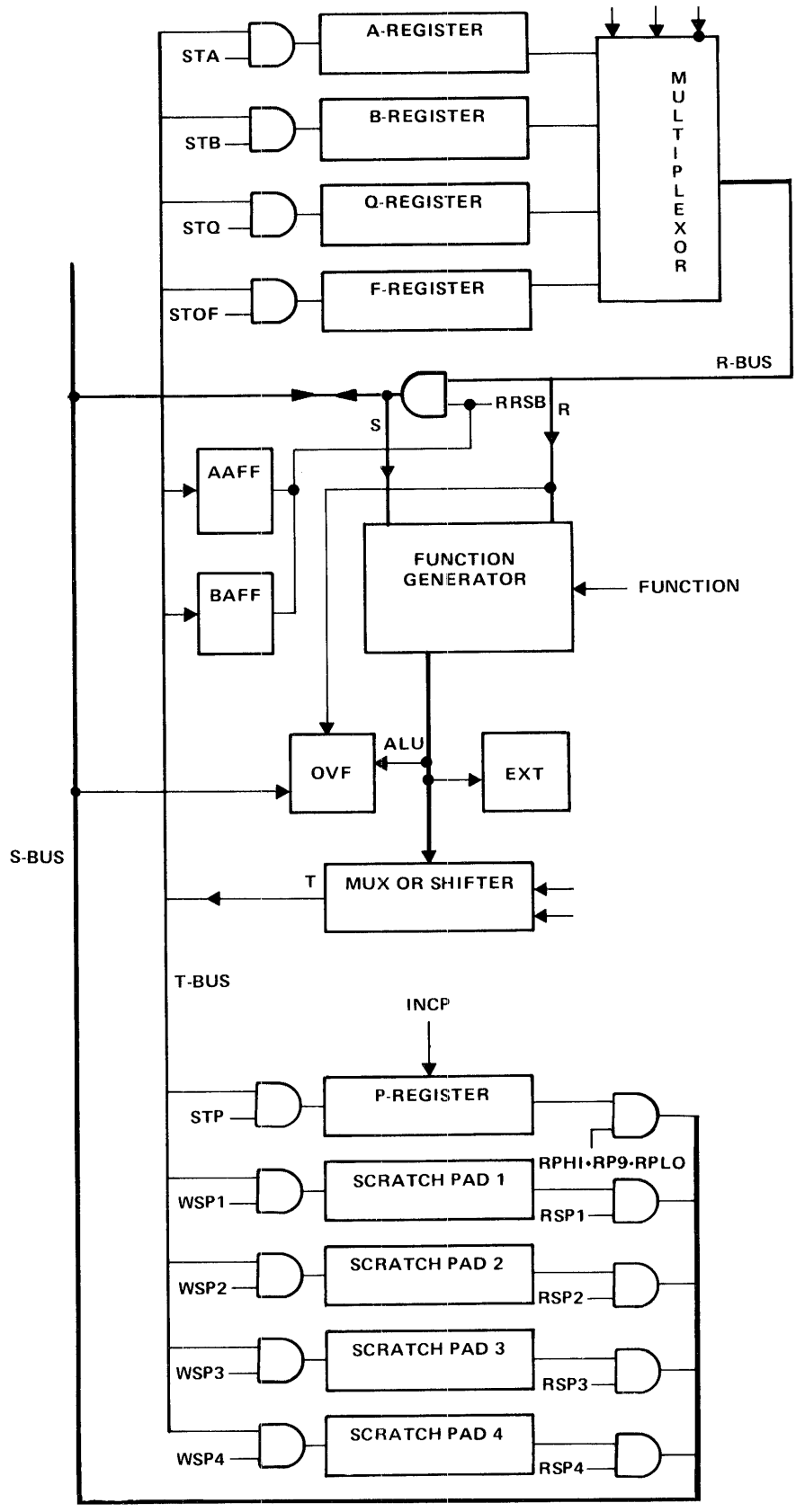
**Undefined codes
***CJMP in later machines (A4 part No. 02100-60022)

CONDITIONAL MICROINSTRUCTIONS

THE EXECUTION OF SOME MICROINSTRUCTIONS IS DEPENDENT ON CERTAIN STATUS SIGNALS FROM MEMORY OR I/O. THE ASYNCHRONOUSLY RUNNING CPU HAS TO WAIT FOR MEMORY- OR I/O- SIGNALS TO BE SUPPLIED AT A PREDEFINED TIME.

IF THE CONDITIONS ARE NOT SATISFIED, THE CPU FREEZES (SUSPENDS EXECUTION OF THE MICROPROGRAM BY ELIMINATING CERTAIN TIMING SIGNALS) UNTIL THE CONDITION IS SATISFIED.

MICROORDER	FIELD	CONDITION
RW / CW	SPECIAL	T6
T / COND	S-BUS	DTRY
M	STORE	$\overline{\text{MBSY}}$
IOG1	SPECIAL	T2



BLOCK DIAGRAM OF ARITHMETIC LOGIC UNIT

F-REGISTER AND Q-REGISTER

- THE F– AND Q– REGISTER ARE TWO ADDITIONAL 16-BIT REGISTERS.
- BOTH REGISTERS ARE FED FROM THE T-BUS AND OUTPUT VIA THE MULTIPLEXER TO THE R-BUS.
- THE Q-REGISTER (QUOTIENT-REGISTER) IS NOT ACCESSIBLE THROUGH SOFTWARE BY THE PROGRAMMER.
- THE F-REGISTER (FENCE-REGISTER) CAN ONLY BE LOADED WITH AN OTA 05 OR OTB 05 SOFTWARE INSTRUCTION.
- BOTH REGISTERS ARE USED BY FIRMWARE (MICROPROGRAMMING) FOR THE EXECUTION OF THE EAG-INSTRUCTION:

ASL

LSL

RRL

DIV

FUNCTION GENERATOR

- THE FUNCTION GENERATOR COMBINES THE CONTENTS ON THE R-BUS WITH THE ONE ON THE S-BUS WITH A SPECIFIED ARITHMETIC OR LOGIC FUNCTION. THE RESULT APPEARS ON THE ALU-BUS.

LOGIC FUNCTIONS		ARITHMETIC FUNCTIONS	
ALU = R+S	(IOR)	ALU = R+S	(ADD)
ALU = $\overline{R}$	(NOT)	ALU = R+S+1	(INCREMENT)
ALU = R $\oplus$ S	(EOR)	ALU = R	(TRANSFER)
ALU = $\overline{R+S}$	(NOR)	ALU = R-S-1	(DECREMENT)
ALU = R·S	(AND)	ALU = R-S	(SUBTRACT)

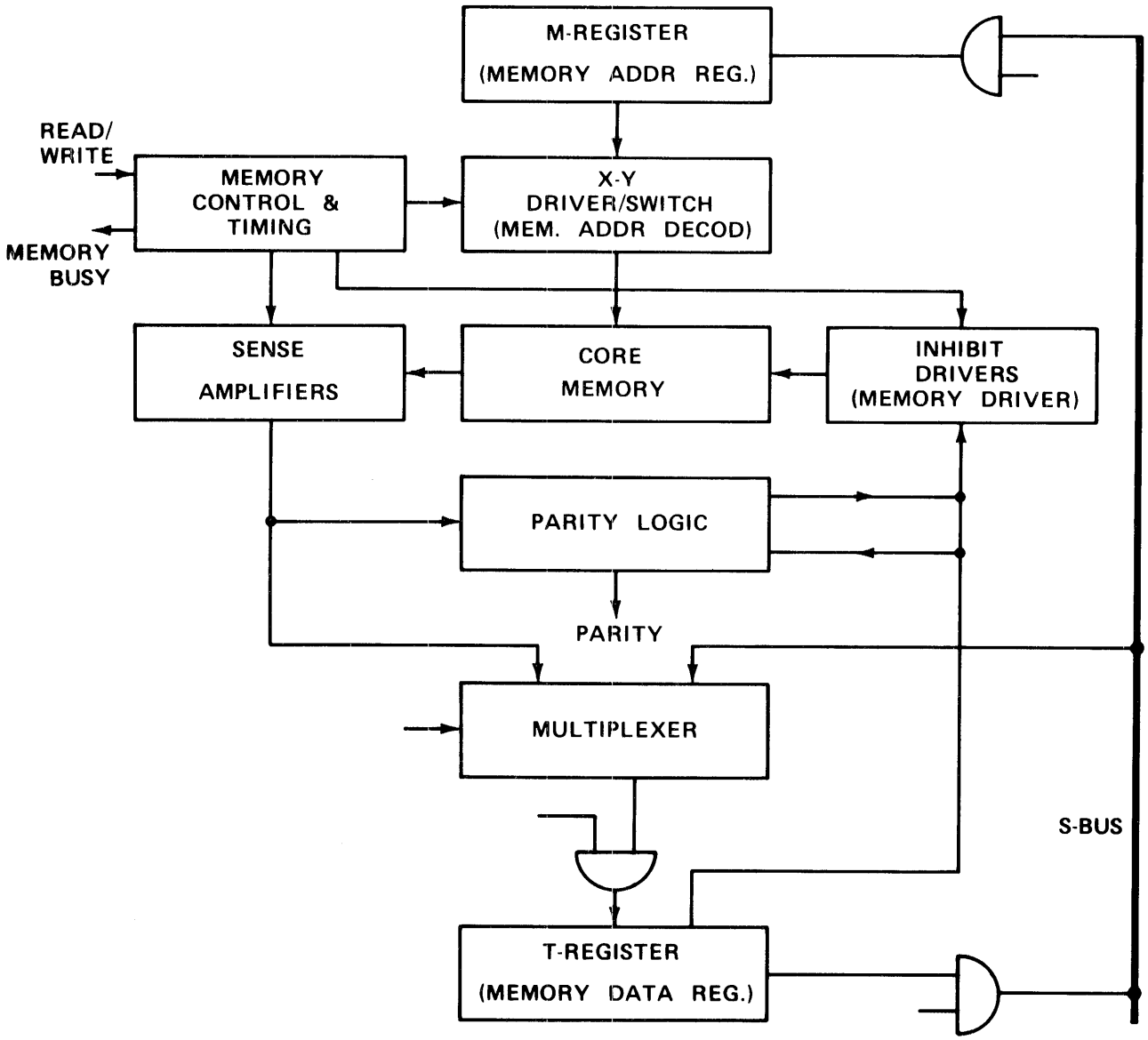
- THE ARITHMETIC OR LOGIC FUNCTION IS DETERMINED BY THE SOFTWARE INSTRUCTION AND DECODED BY THE MICRO-ORDER IN THE FUNCTION FIELD.

THE SCRATCH PADS

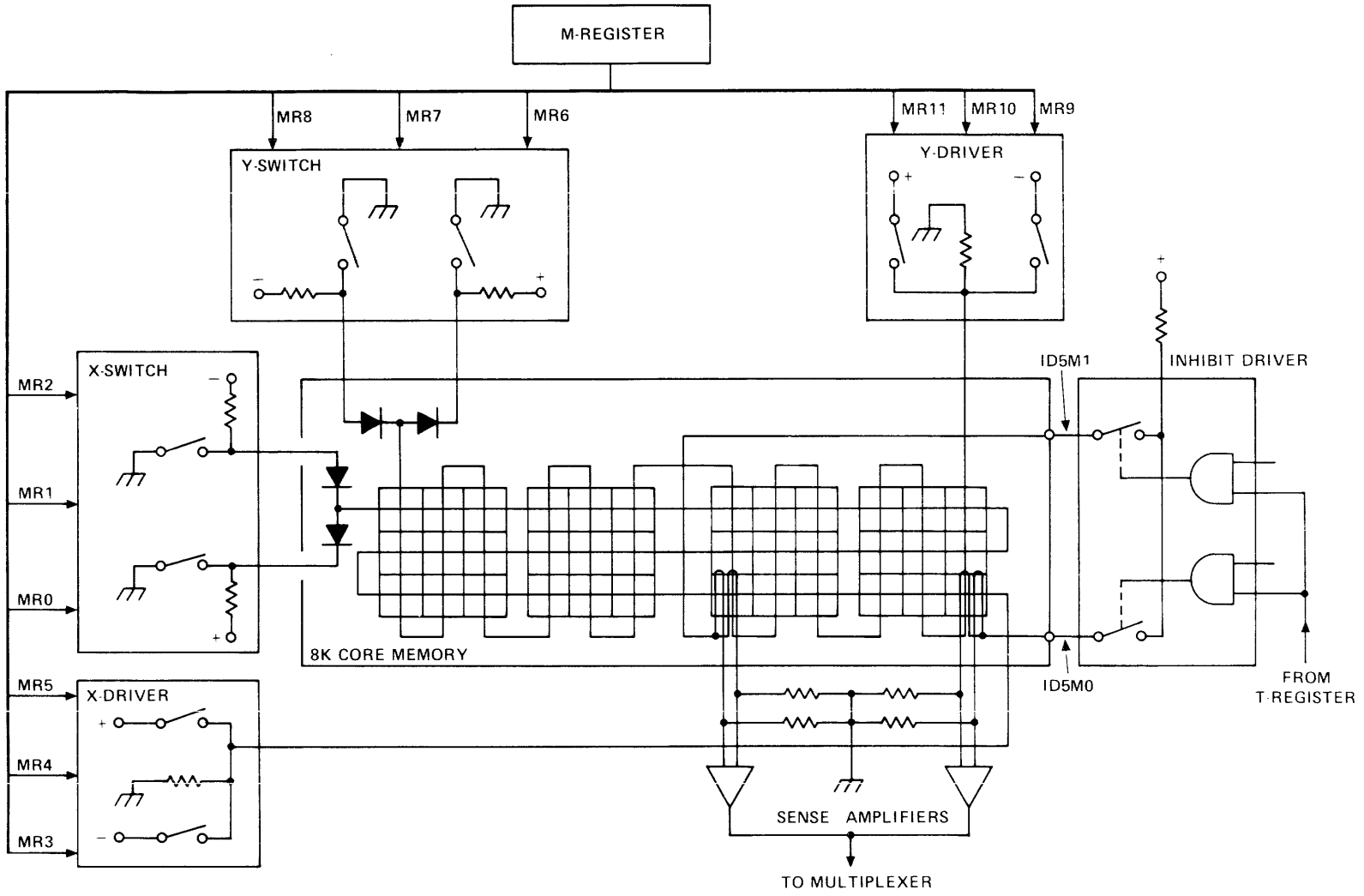
- FOUR 16 BIT REGISTERS FOR TEMPORARY STORAGE OF ADDRESSES AND DATA ARE DESIGNATED SCRATCH PAD 1-4.
- ALL FOUR SCRATCH PAD REGISTERS ARE LOADED FROM THE T-BUS AND OUTPUT TO THE S-BUS.
- THE REGISTERS ARE EXCLUSIVELY ACCESSIBLE BY MICROPROGRAMMING.
- THE GENERAL PURPOSE OF THE SCRATCH PADS IS AS FOLLOW:

SCP1	SCP2	SCP3	SCP4
HOLDS OPERAND ADDRESS OF MRG-INSTRUCTIONS & IS USED DURING DIV-INSTRUC.	HOLDS OPERAND DATA OF MRG-INSTRUCTIONS & IS USED DURING DIV-INSTRUC.	HOLDS F-REG. CONTENTS FOR ASL-, LSL- & RRL-INSTRUC-TIONS & IS USED DURING DIV-INSTRUC.	HOLDS MULTIPLICAND FOR MPY-INSTRUCTION & F-REG CONTENT DURING DIV-INSTRUCTION

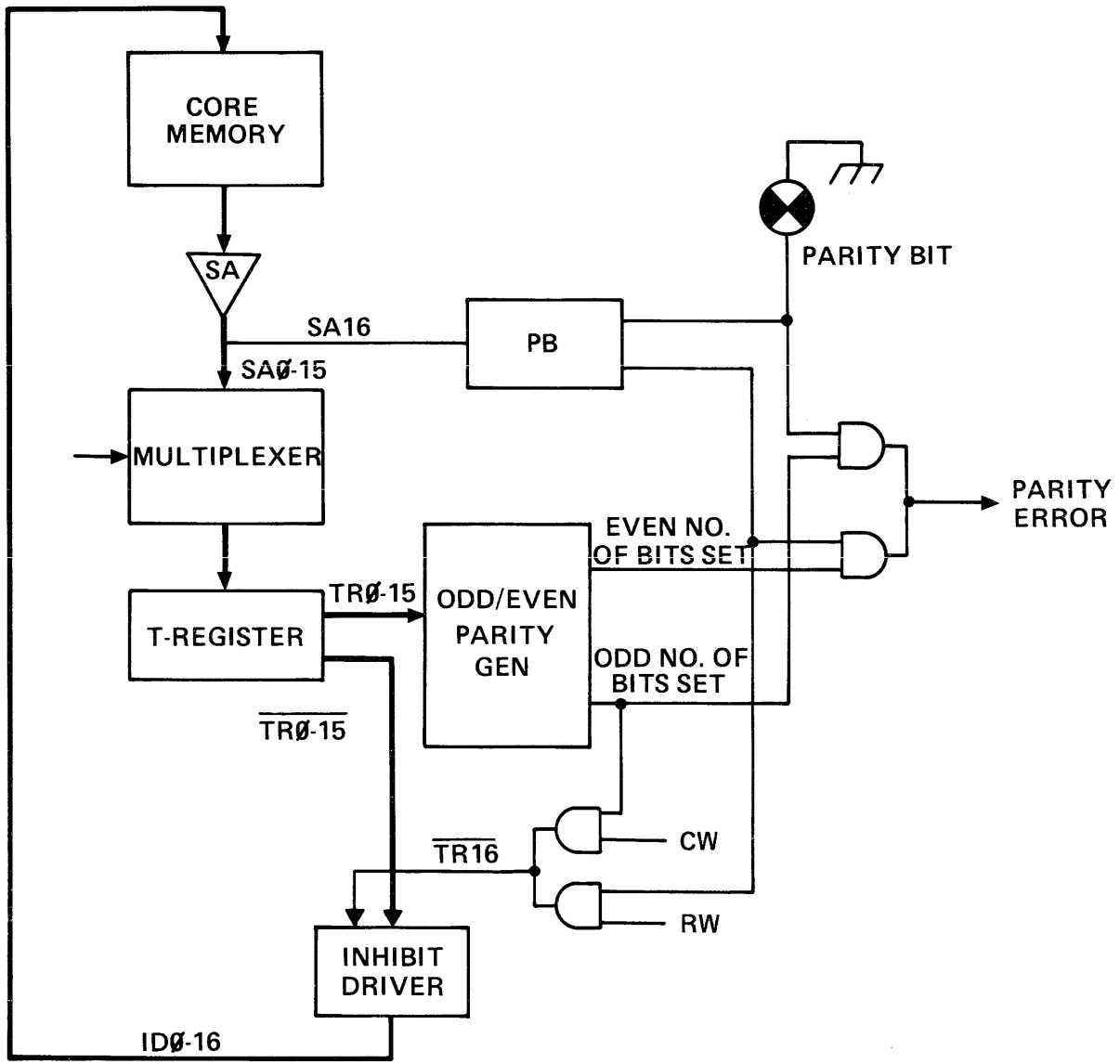
BLOCK DIAGRAM OF MEMORY

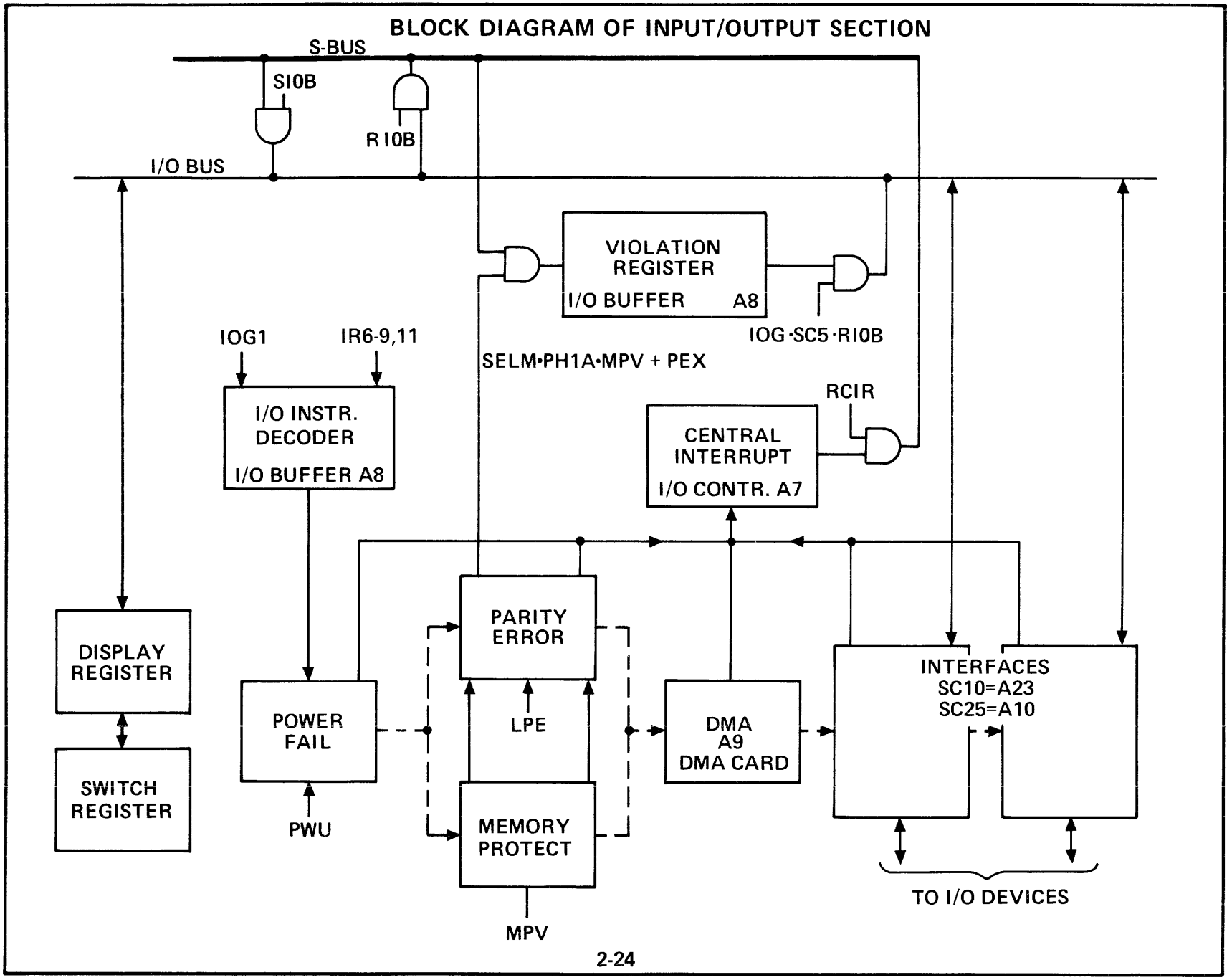


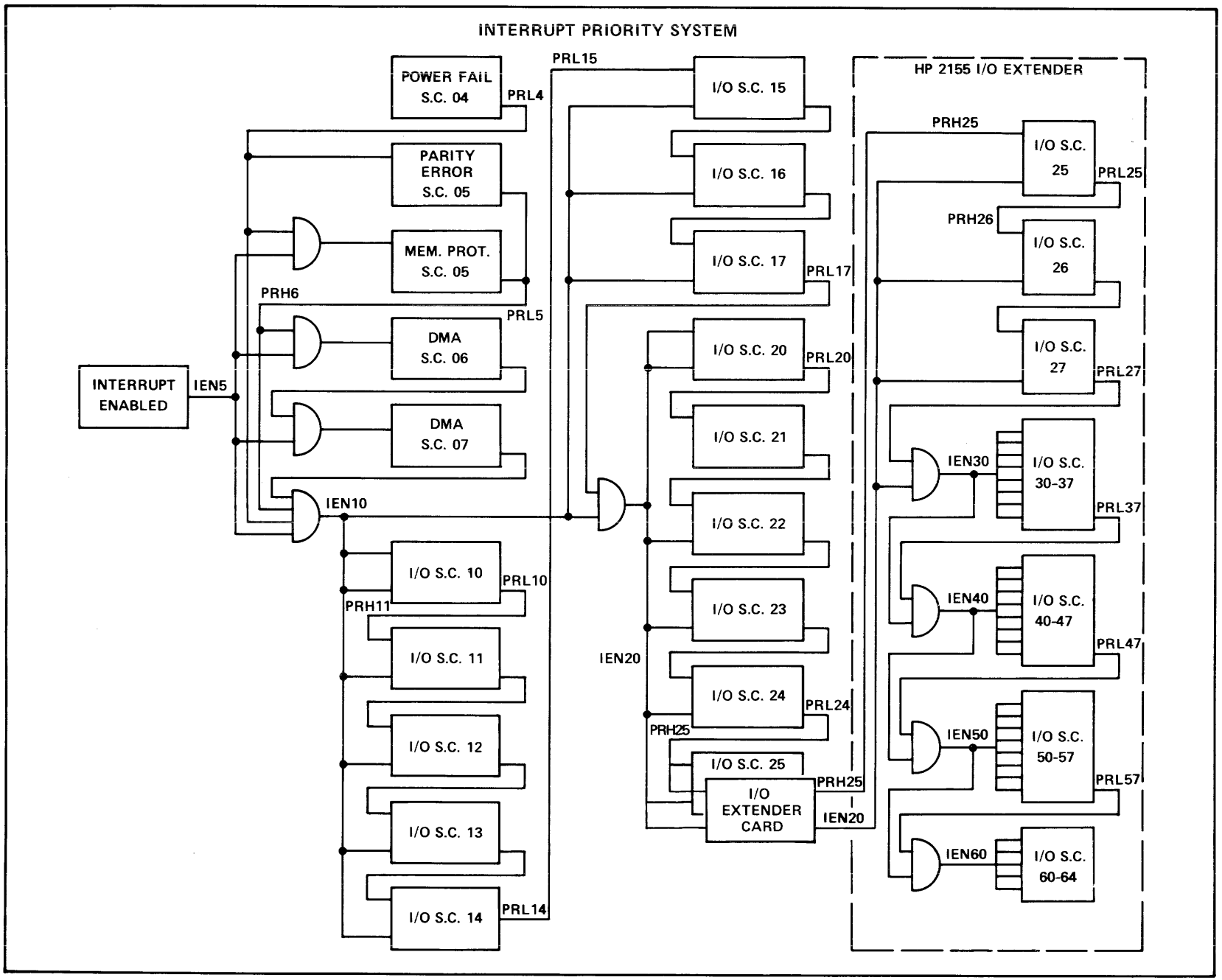
ADDRESSING AND READ/WRITE CYCLE IN CORE MEMORY



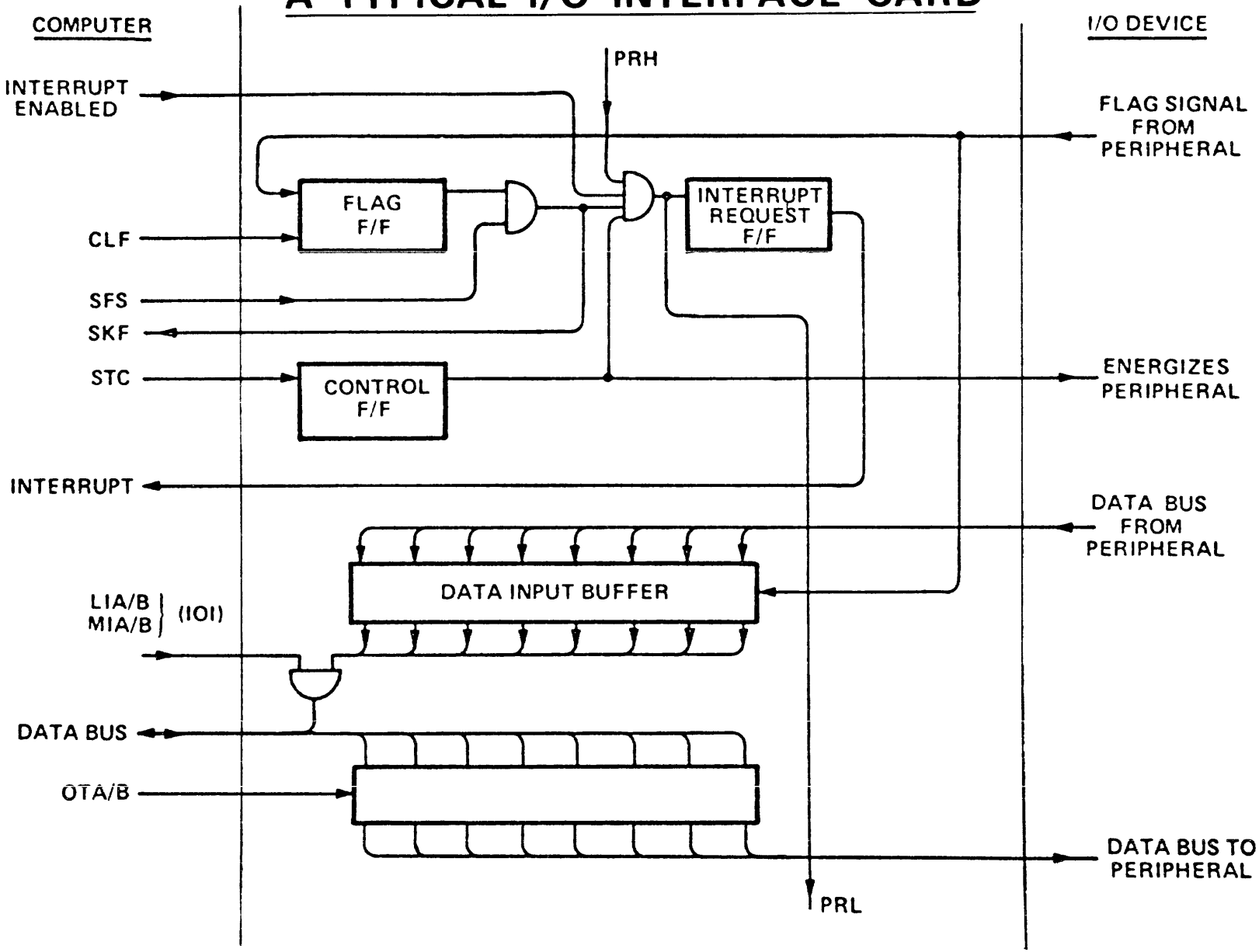
PARITY LOGIC

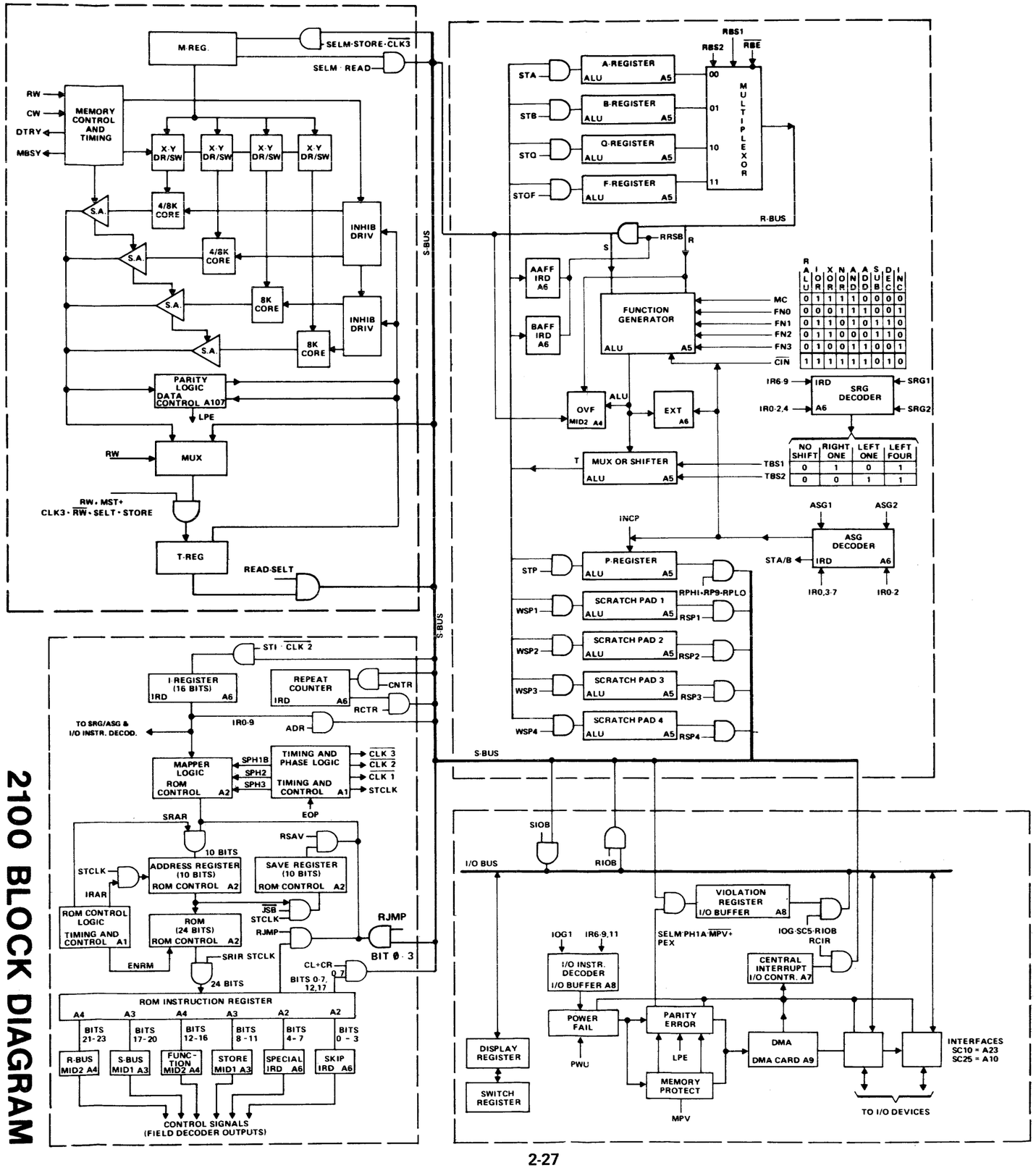



BLOCK DIAGRAM OF INPUT/OUTPUT SECTION
S-BUS
SIOB
RIOB
I/O BUS
VIOLATION REGISTER
I/O BUFFER A8
IOG·SC5·RIOB
IOG1
IR6-9,11
SELM·PH1A·MPV + PEX
RCIR
I/O INSTR. DECODER
I/O BUFFER A8
CENTRAL INTERRUPT
I/O CONTR. A7
DISPLAY REGISTER
SWITCH REGISTER
PARITY ERROR
POWER FAIL
LPE
DMA A9 DMA CARD
INTERFACES SC10=A23 SC25=A10
PWU
MEMORY PROTECT
MPV
TO I/O DEVICES


INTERRUPT PRIORITY SYSTEM
POWER FAIL S.C. 04
PRL4
PARITY ERROR S.C. 05
MEM. PROT. S.C. 05
PRH6
PRL5
DMA S.C. 06
DMA S.C. 07
INTERRUPT ENABLED
IEN5
IEN10
I/O S.C. 10
PRL10
PRH11
I/O S.C. 11
I/O S.C. 12
I/O S.C. 13
I/O S.C. 14
PRL14
PRL15
I/O S.C. 15
I/O S.C. 16
I/O S.C. 17
PRL17
I/O S.C. 20
PRL20
I/O S.C. 21
I/O S.C. 22
I/O S.C. 23
IEN20
I/O S.C. 24
PRL24
PRH25
I/O S.C. 25
I/O EXTENDER CARD
PRH25
IEN20
HP 2155 I/O EXTENDER
PRH25
I/O S.C. 25
PRL25
PRH26
I/O S.C. 26
I/O S.C. 27
PRL27
IEN30
I/O S.C. 30-37
PRL37
IEN40
I/O S.C. 40-47
PRL47
IEN50
I/O S.C. 50-57
PRL57
IEN60
I/O S.C. 60-64

A TYPICAL I/O INTERFACE CARD





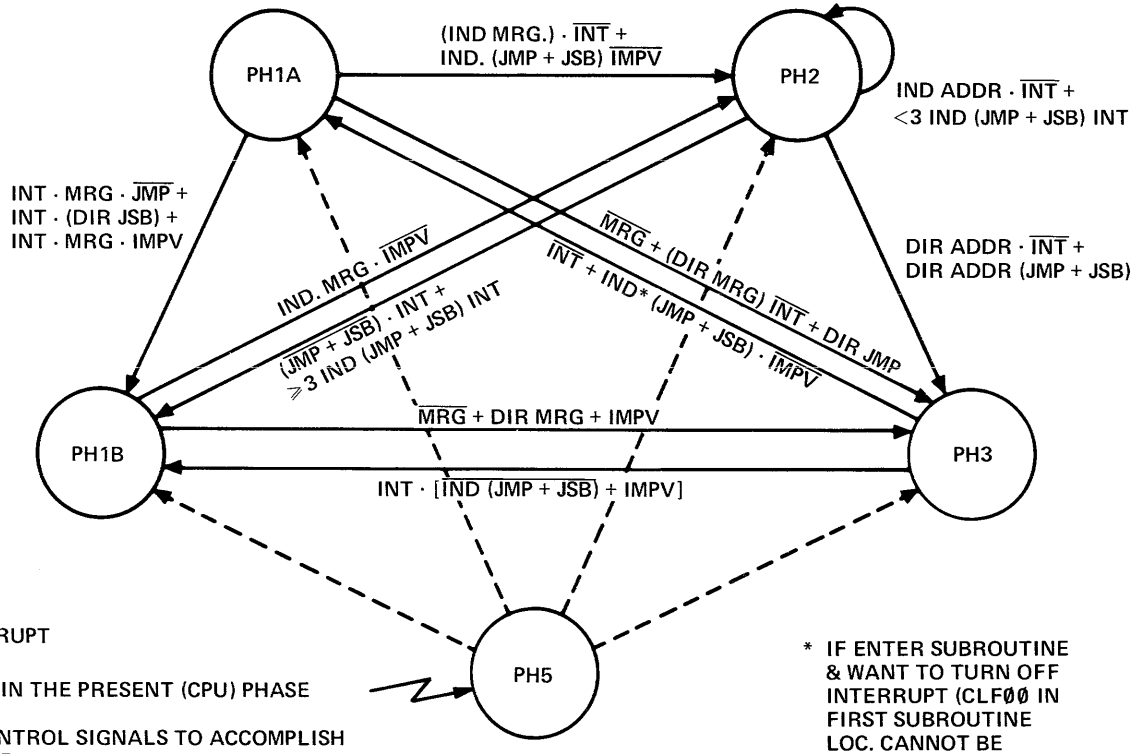
2100 BLOCK DIAGRAM

microprogramming and central processor unit 3

LESSON 3

MICROPROGRAMMING AND CENTRAL PROCESSOR UNIT

PHASE CONTROL LOGIC



A WAITING DMA INTERRUPT
IMPOSES A PH5 WHICH:

1) INHIBITS A CHANGE IN THE PRESENT (CPU) PHASE
2) FREEZES THE CPU
3) CREATS ITS OWN CONTROL SIGNALS TO ACCOMPLISH A DMA DATA TRANSFER

* IF ENTER SUBROUTINE & WANT TO TURN OFF INTERRUPT (CLFØØ IN FIRST SUBROUTINE LOC. CANNOT BE INTERRUPTED

BOOLEAN EQUATIONS FOR PHASE CONTROL

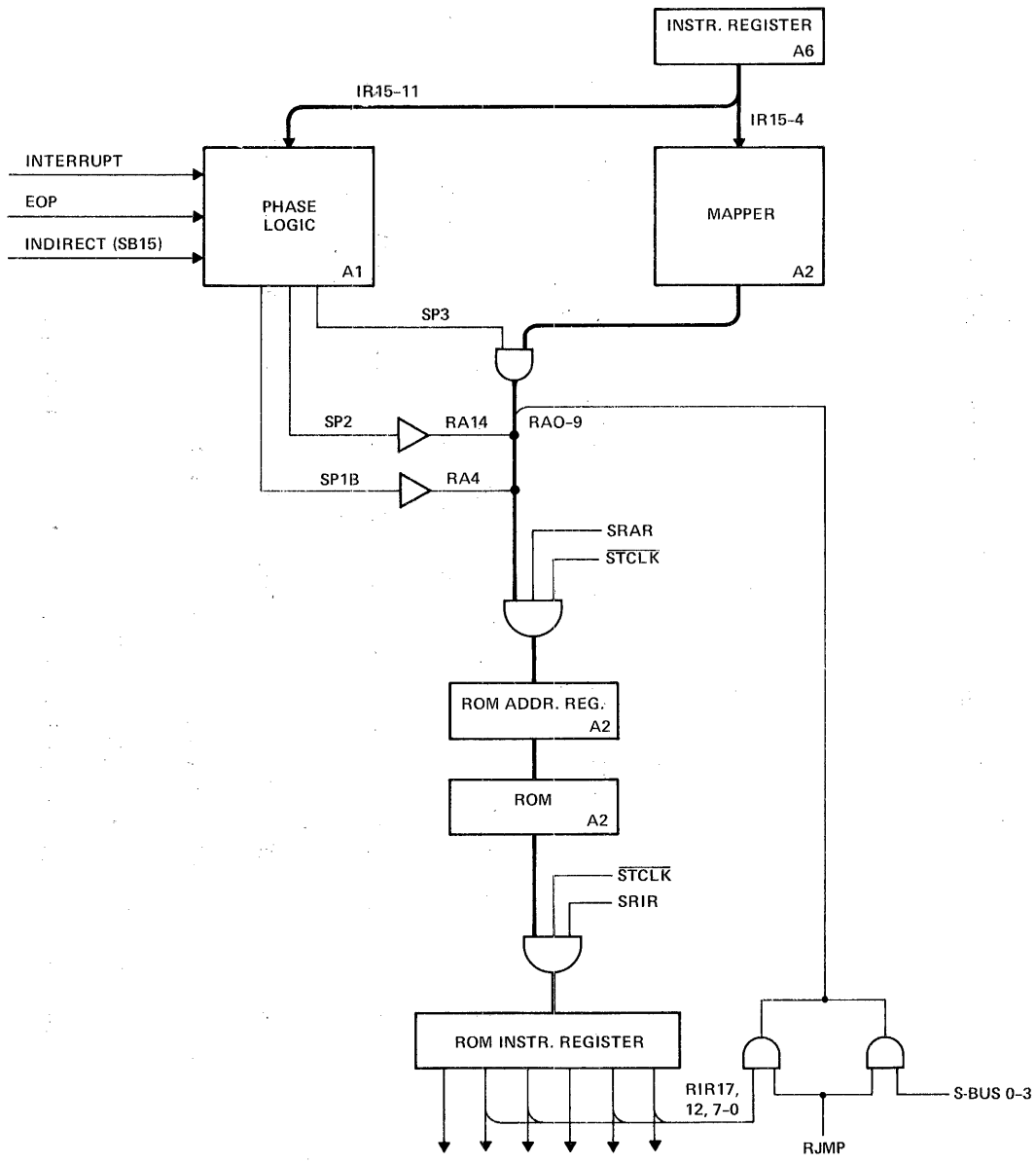
$$SPH1A = PH3 \cdot EOP \cdot \overline{LOOP} \cdot [INT + IR15 \cdot \overline{IMPV}\ (JMP + JSB)]$$

$$\begin{aligned} SPH1B = {} & PH2 \cdot EOP \cdot \overline{LOOP} \cdot INT\ (SB15 \cdot CT3 + \overline{JMP + JSB}) + \\ & PH3 \cdot EOP \cdot \overline{LOOP} \cdot INT\ (\overline{JMP + JSB} + \overline{IR15} + IMPV) + \\ & PH1A \cdot EOP \cdot \overline{LOOP} \cdot INT \cdot MRG\ [\overline{JMP + JSB} + \overline{JMP}\ (\overline{IR15} + IMPV)] \end{aligned}$$

$$\begin{aligned} SPH3 = {} & PH1A \cdot EOP \cdot \overline{LOOP}\ [\overline{INT}\ (\overline{IR15} + IMPV) + JMP\ (\overline{IR15} + IMPV) + \overline{MRG}] + \\ & PH2 \cdot EOP \cdot \overline{LOOP}\ [\overline{INT} \cdot \overline{SB15} + (JMP + JSB) \cdot \overline{SB15}] + \\ & PH1B \cdot EOP \cdot \overline{LOOP}\ [\overline{IR15} + IMPV + \overline{MRG}] \end{aligned}$$

$$\begin{aligned} SPH2 = {} & MRG \cdot EOP \cdot \overline{LOOP}\ \{PH1A \cdot IR15 \cdot \overline{IMPV}\ (\overline{INT} + JMP + JSB) + \\ & \qquad PH2 \cdot SB15\ [\overline{INT} + (JMP + JSB) \cdot \overline{CT3}] + \\ & \qquad PH1B \cdot IR15 \cdot \overline{IMPV}\} \end{aligned}$$

MAPPING THE ROM ADDRESS



BOOLEN EQUATIONS FOR MAPPER OUTPUT

RA0 = (ASG + ASR + ASL) · SPH3

RA1 = (LSR + LSL) · SPH3

RA2 = SP1B + SPH2 + [(STC + CLC + LI*) + (JMP + AD* + CP* + LD* + ST*) + (RRR + RRL) + (CL* + CC*)] SPH3

RA3 = SPH2 + [(OT* + STC + CLC) + (MPY + DLD) + (AND + IOR + JSB + ISZ + CP* + ST*) + (CM* + CC*)] SPH3

RA4 = [(DIV + DST) + (XOR + IOR + JMP + ISZ + LD* + ST*) + (MI* + LI* + OT* + STC + CLC) + ASG] SPH3

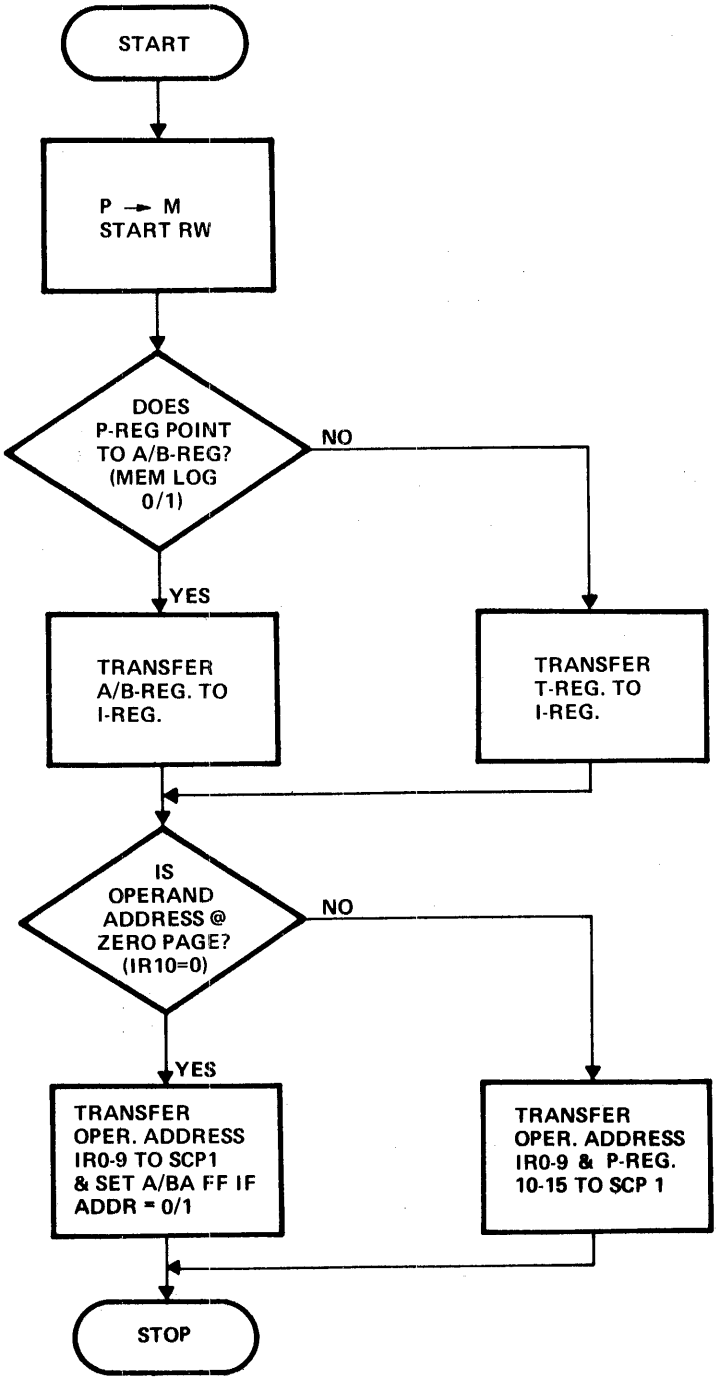
RA5 = [(ASR + LSR + RRR) + (JSB + ISZ) + (LD* + ADD) + IOG] SPH3

RA6 = [(DST + DLD) + (MRG + SRG)] SPH3

RA7 = $EAG_{(FIX\ POINT)}$ · SPH3

RA8 = (FAD + FSB + FMY + FDV + FIX + FLT) · SPH3

FETCH PHASE


START
P → M
START RW
DOES P-REG POINT TO A/B-REG? (MEM LOG 0/1)
NO
YES
TRANSFER A/B-REG. TO I-REG.
TRANSFER T-REG. TO I-REG.
IS OPERAND ADDRESS @ ZERO PAGE? (IR10=0)
NO
YES
TRANSFER OPER. ADDRESS IR0-9 TO SCP1 & SET A/BA FF IF ADDR = 0/1
TRANSFER OPER. ADDRESS IR0-9 & P-REG. 10-15 TO SCP 1
STOP

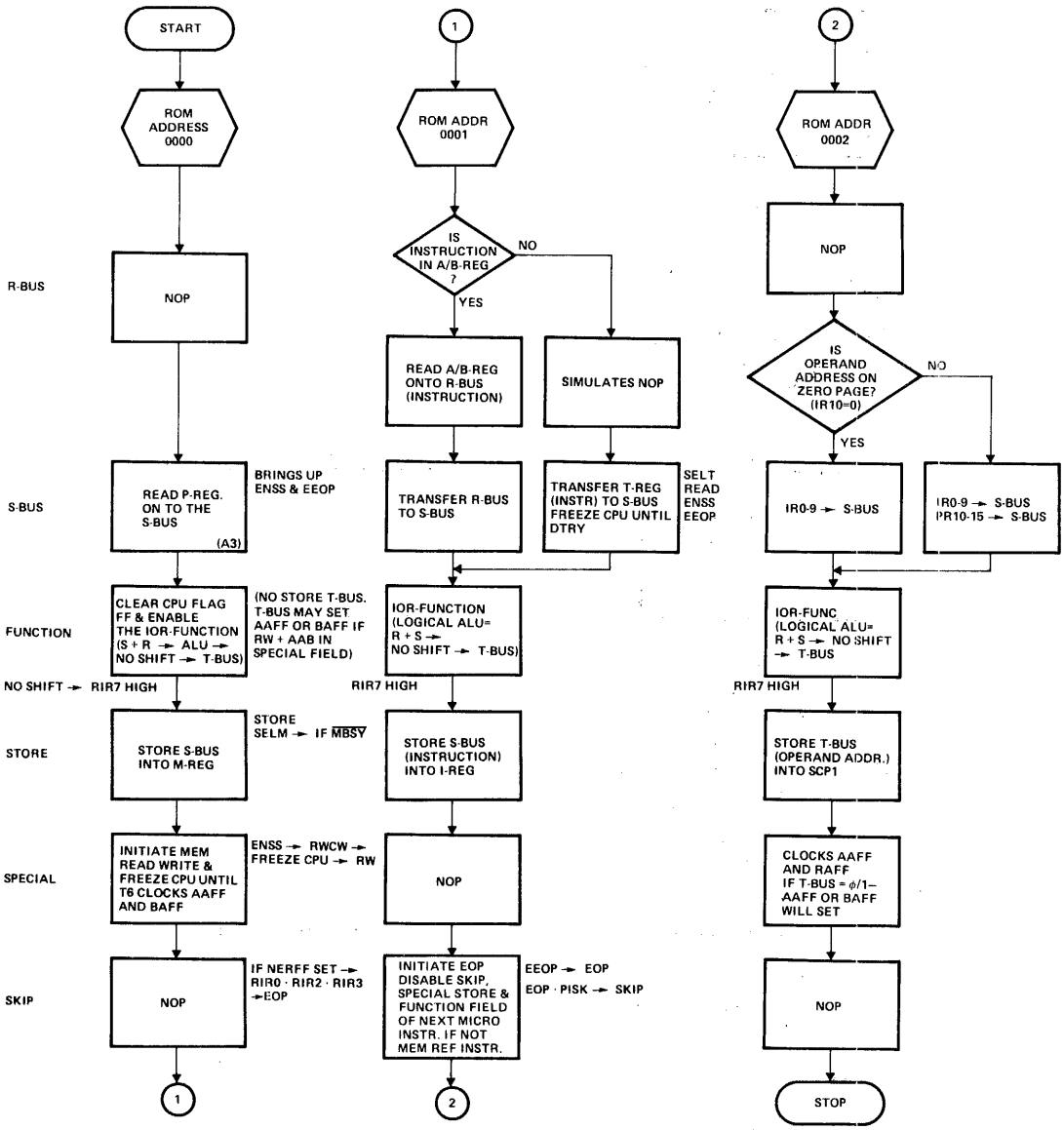
PH1A FLOW CHART

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0000	77330757	PH1A	–	P	CFLG	M	RW	–
0001	53771775		AAB	COND	IOR	IR	–	EOP
0002	73373557		–	ADR	IOR	S1	AAB	–

R	S	FUNC	ST	SP	SK
111	111 0	11 011	000 1	11 10	1 111

R	S	FUNC	ST	SP	SK
101	011 1	11 111	001 1	11 11	1 101

R	S	FUNC	ST	SP	SK
111	011 0	11 111	011 1	01 10	1 111



CPU FREEZE

WHEN A CPU FREEZE OCCURS ALL OF THE CLOCK SIGNALS WITH THE EXCEPTION OF $\overline{\text{CLK3}}$ ARE INHIBITED. THIS PREVENTS THE RAR AND RIR FROM BEING CHANGED AND INHIBITS EXECUTION OF THE ROM MICROINSTRUCTION.

FREEZE SOURCES:

PH5

MEM PROTECT

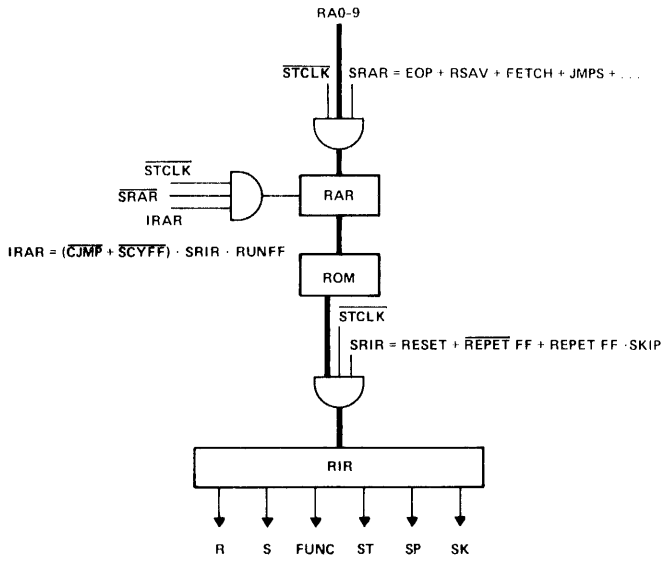
T OR COND IN S BUS AND $\overline{\text{ABFF}}$ → FREEZE UNTIL DTRY

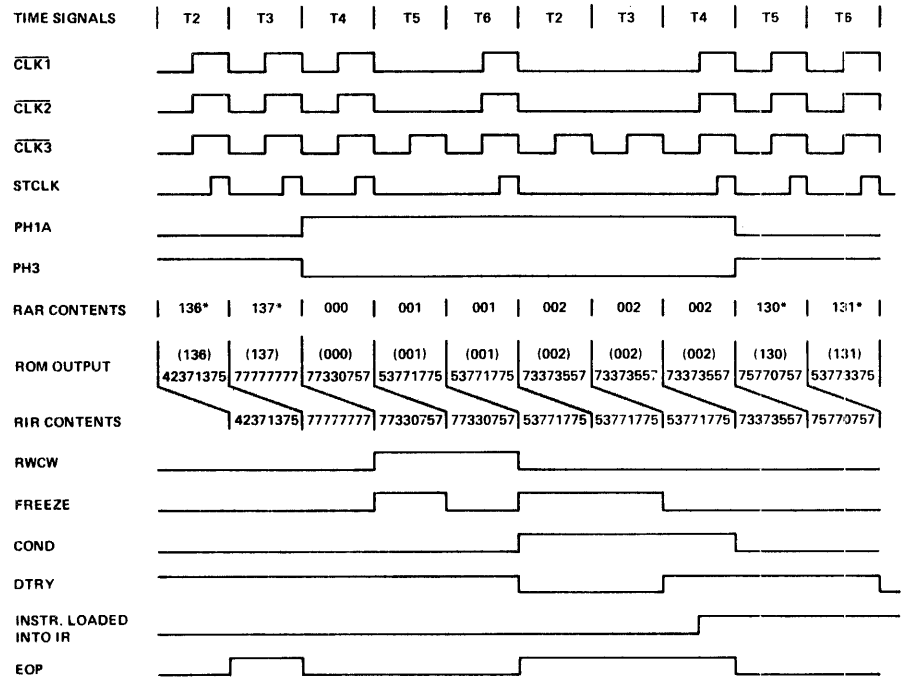
RW OR CW AND $\overline{\text{T6}}$

MBSY AND M IN STORE FIELD

DIV AND $\overline{\text{DT}}$

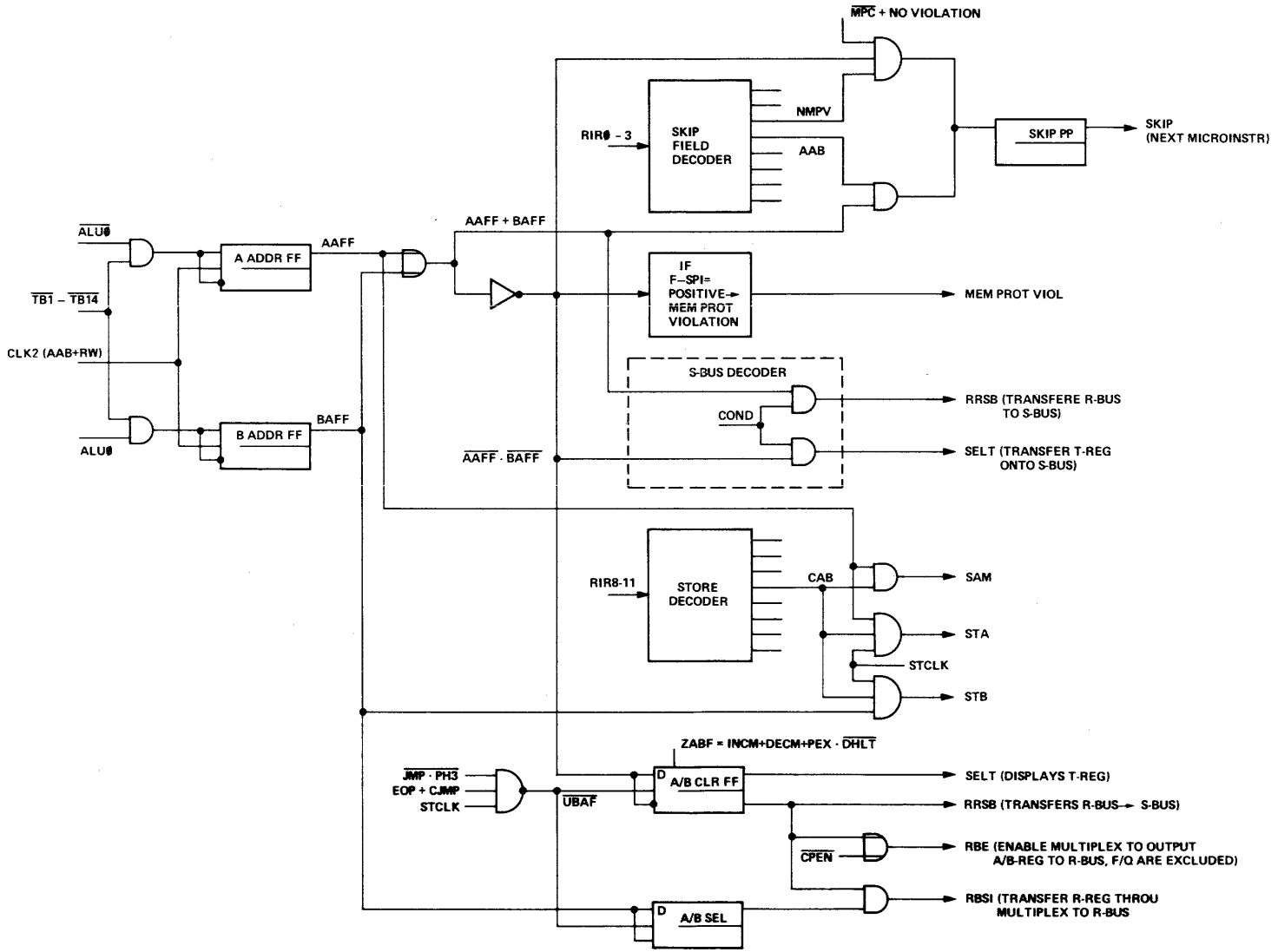
TIMING DIAGRAM OF PH1A
(INSTRUCTION DOES NOT RESIDE IN A/B-REGISTER)



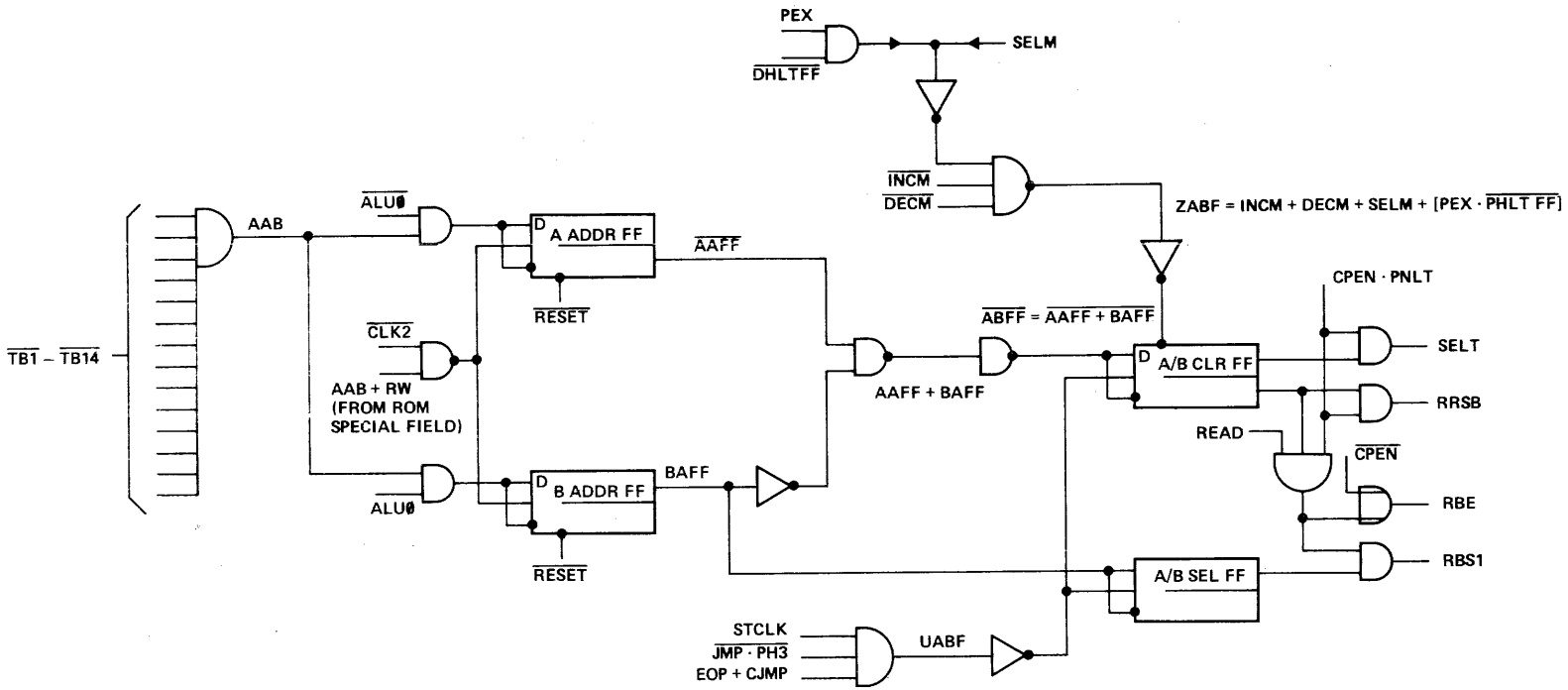


*THESE RAR CONTENTS ARE ARBITRARILY CHOSEN

SIMPLIFIED DIAGRAM OF A/B ADDRESSABLE LOGIC



A/B ADDR. FF, A/B CLRFF & A/B SEL FF. LOGIC IN HALT/SINGLE CYCLE MODE



1.) A/B CLR FF SETS IF NEITHER A– NOR B–ADDR. FF'S HAVE BEEN SET. ∴ SELT · READ → TRANSFERS T-REG TO S-BUS IF COMPUTER IN HLT MODE.

2.) A/B CLR FF RESET IF A– OR B–ADDR. FF IS SET. ∴ RBE ENABLES MULTIPL., $\overline{RBS1}$ → A-REG TO R-BUS, RBS1 → B-REG TO R-BUS. RRSB → TRANSFERS A/B-REG CONTENTS FROM R- TO S-BUS.

3.) S-BUS CONTENTS → I/O-BUS WITH SIOB (FROM FRONT PANEL) → TO DISPLAY REGISTER.

4.) CJMP IS INCLUDED IN THE EQUATION OF UABF TO ENABLE THE PROPER DISPLAY OF EAG–MEMORY REFERENCE INSTRUCTIONS (MPY, DIV, DST, DLD) IF IN SINGLE CYCLE MODE.

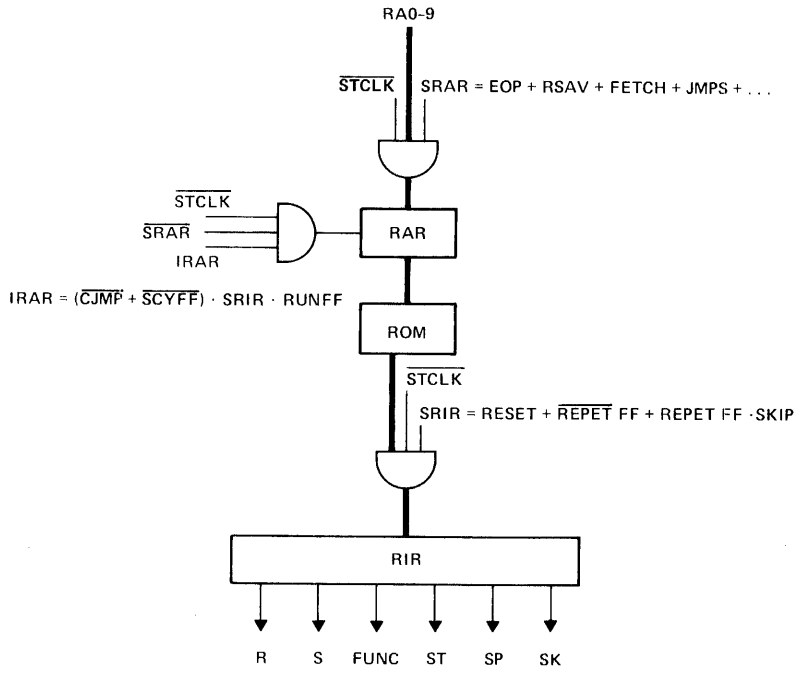
5.) UABF IS DISABLED AT THE END OF JMP · PH3. THIS WILL INHIBIT THE A/B CLR FF TO BE SET ($\overline{ABFF}$ IS HIGH AT JMP · PH3 DUE TO MICRO INSTRUCTION @ ROM ADDRESS 124) ∴ $\overline{SELT}$ → THE JMP INSTRUCTION IS DISPLAYED DURING PH1A AND PH3.

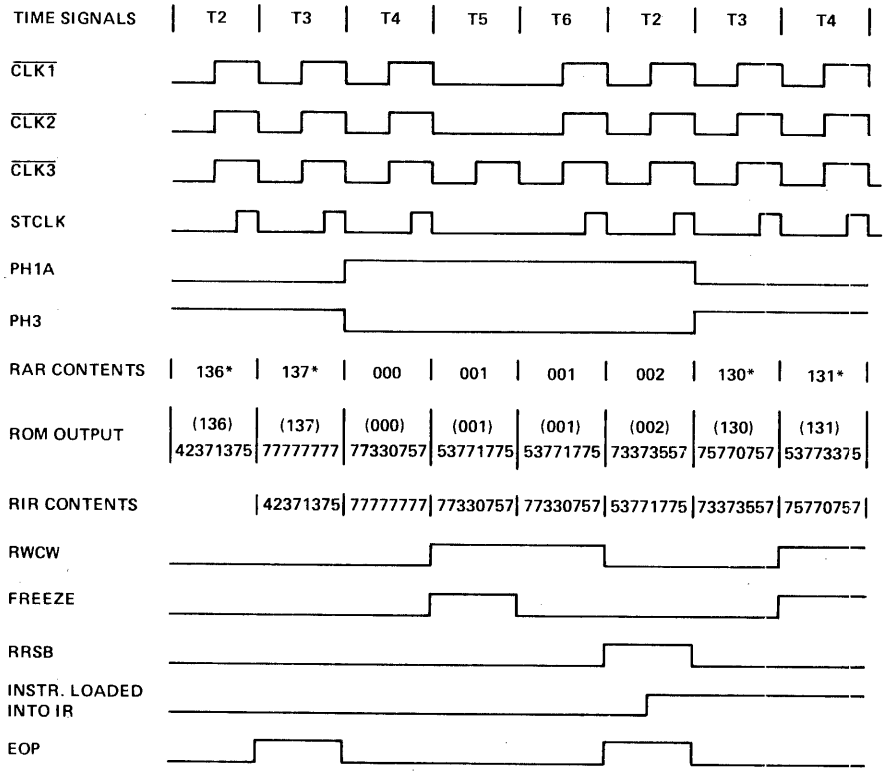
ROM MICROINSTRUCTION FUNCTION FIELD DECODING AND EQUATIONS
CREATED BY ARITHMATIC LOGIC FUNCTION GENERATOR

ROM INSTRUCTION REGISTER BITS					FUNCTION FIELD (5 BITS)	MC	FN0	FN1	FN2	FN3	CIN	OPERATION	EQUATION OF ARITHMATIC LOGIC FUNCTION GENERATOR
16	15	14	13	12									
1	1	1	1	1	IOR	1	0	1	1	1	0	LOGIC OPERATIONS	ALU = R+S
1	1	1	1	0	SOV	1	0	1	1	1	0		
1	1	1	0	1	CLO	1	0	1	1	1	0		
1	1	1	0	0	SFLG	1	0	1	1	1	0		
1	1	0	1	1	CFLG	1	0	1	1	1	0		
1	1	0	1	0	LWF	1	0	1	1	1	0		
1	1	0	0	1	—	1	0	1	1	1	0		
1	1	0	0	0	ARS	1	0	1	1	1	0		
1	0	1	1	1	CRS	1	0	1	1	1	0		
1	0	1	1	0	LGS	1	0	1	1	1	0		
1	0	1	0	1	RSB	1	0	1	1	1	0		
1	0	1	0	0	CJMP	1	0	1	1	1	0		
1	0	0	1	1	JMP	1	0	1	1	1	0		
1	0	0	1	0	JMP	1	0	1	1	1	0		
1	0	0	0	1	JSB	1	0	1	1	1	0		
1	0	0	0	0	JSB	1	0	1	1	1	0		
0	1	1	1	1	—	1	0	0	0	0	0		ALU = $\overline{R}$
0	1	1	1	0	XOR	1	0	1	1	0	0		ALU = R ⊕ S
0	1	1	0	1	NOR	1	1	0	0	0	0		ALU = $\overline{R+S}$
0	1	1	0	0	AND	1	1	1	0	1	0		ALU = R·S
0	1	0	1	1	ADD	0	1	0	0	1	0	ARITHMATIC OPERATIONS	ALU = R+S
0	1	0	1	0	ADDO	0	1	0	0	1	0		ALU = R+S
0	1	0	0	1	INC	0	1	0	0	1	1		ALU = R+S+1
0	1	0	0	0	INCO	0	1	0	0	1	1		ALU = R+S+1
0	0	1	1	1	—	0	0	0	0	0	0		ALU = R
0	0	1	1	0	DEC	0	0	1	1	0	0		ALU = R-S-1
0	0	1	0	1	SUB	0	0	1	1	0	1		ALU = R-S
0	0	1	0	0	DIV	0	0	1	1	0	1		ALU = R-S
0	0	0	1	1	MPY	0	1*	0	0	1*	0		ALU = R+S
0	0	0	1	0	P1A	0	0	0	0	0	0		ALU = R
0	0	0	0	1	RFE	0	0	0	0	0	0		ALU = R
0	0	0	0	0	RFI	0	0	0	0	0	0		ALU = R

*IF A-REG BIT Ø SET

TIMING DIAGRAM OF PH1A
(INSTRUCTION RESIDES IN A/B-REGISTER)





*THESE RAR CONTENTS ARE ARBITRARILY CHOSEN

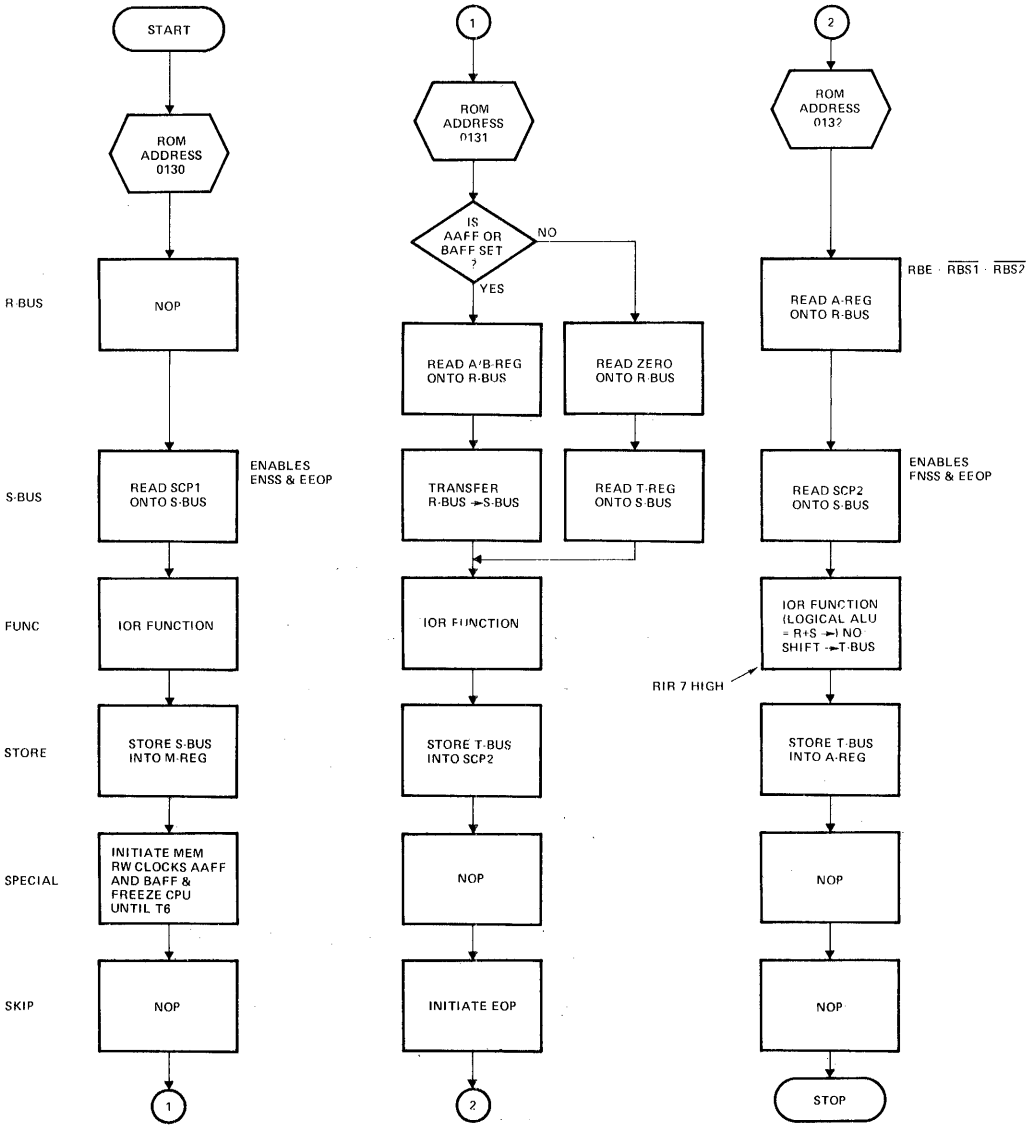
FLOW CHART OF IOR -- INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0130	75770757	IOR	--	S1	IOR	M	RW	--
0131	53773375		AAB	COND	IOR	S2	--	EOP
0132	05377377		A	S2	IOR	A	--	--

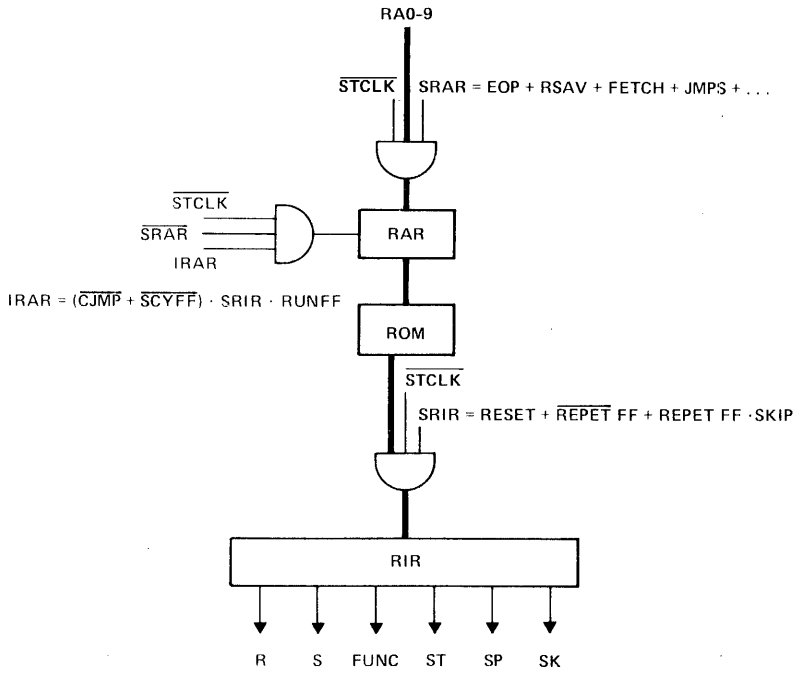
R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 111

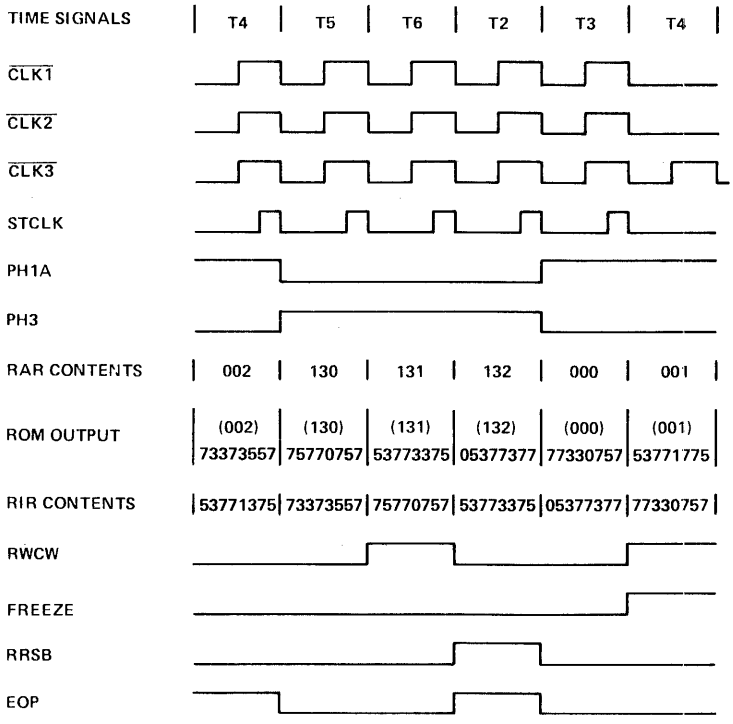
R	S	FUNC	ST	SP	SK
101	011 1	11 111	011 0	11 11	1 101

R	S	FUNC	ST	SP	SK
000	101 0	11 111	111 0	11 11	1 111

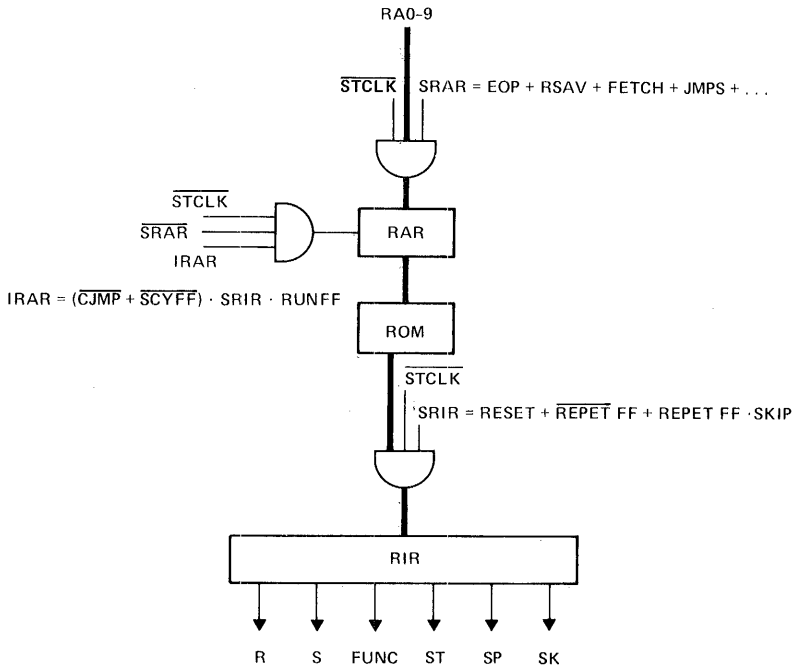


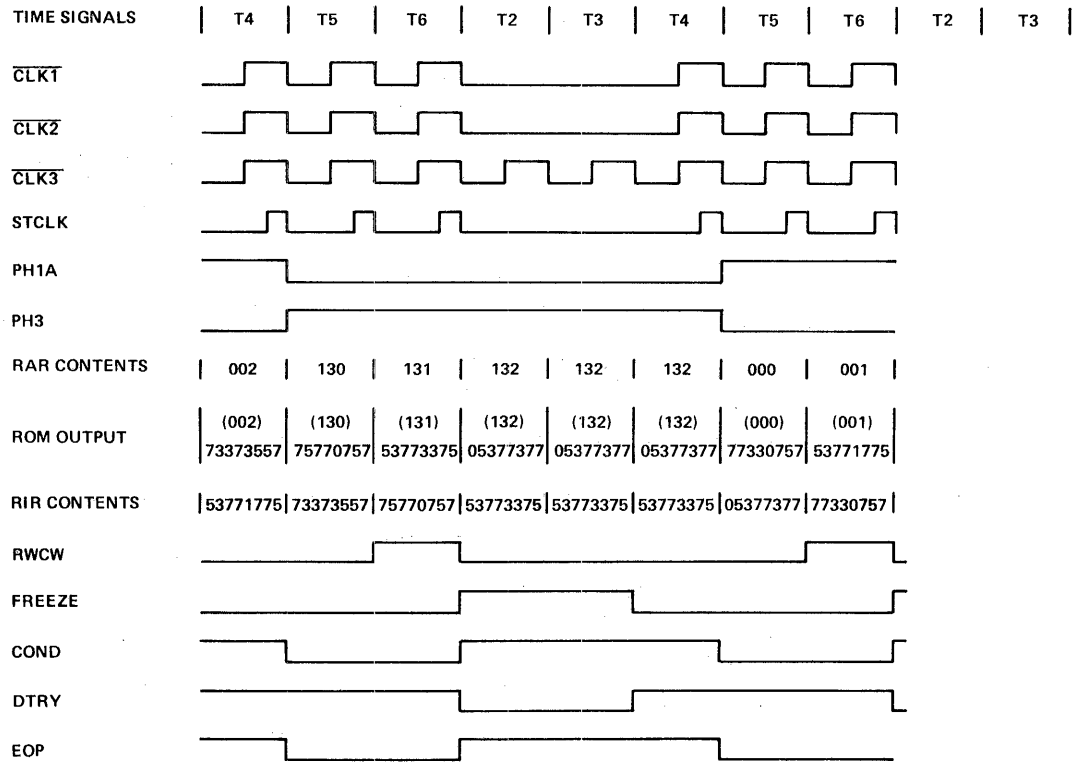
TIMING DIAGRAM OF PH3 IOR INSTRUCTION
(OPERAND IS IN A/B-REGISTER)



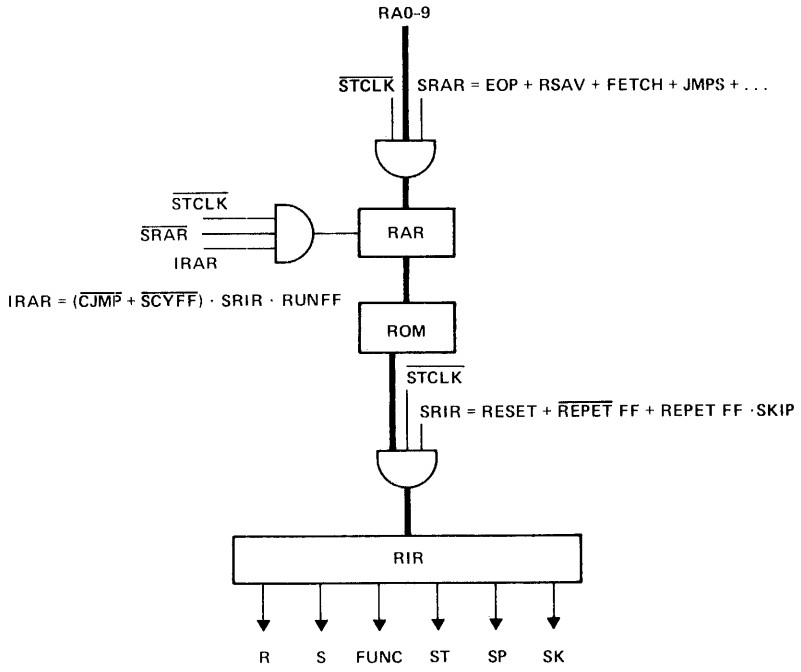


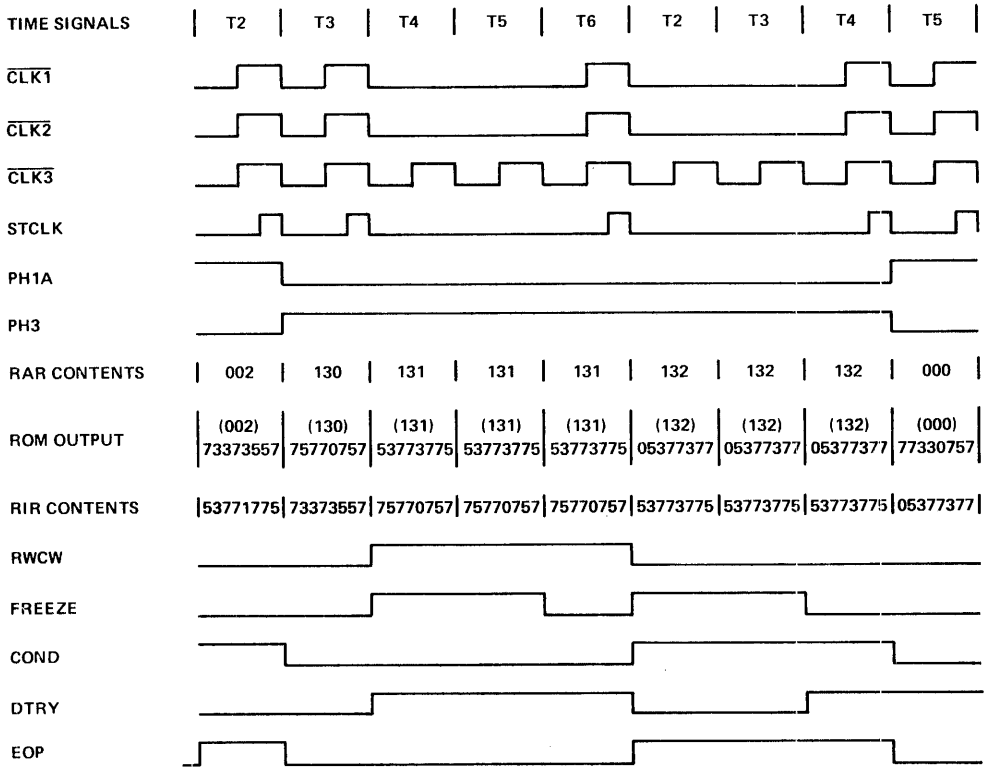
TIMING DIAGRAM OF PH3 IOR INSTRUCTION
(OPERAND IS NOT IN A/B-REGISTER)





TIMING DIAGRAM OF PH3 IOR INSTRUCTION
(INSTRUCTION RESIDES IN A/B-REG. & OPERAND IS NOT IN A/B-REGISTER)





FLOW CHART OF AND/XOR – INSTRUCTION

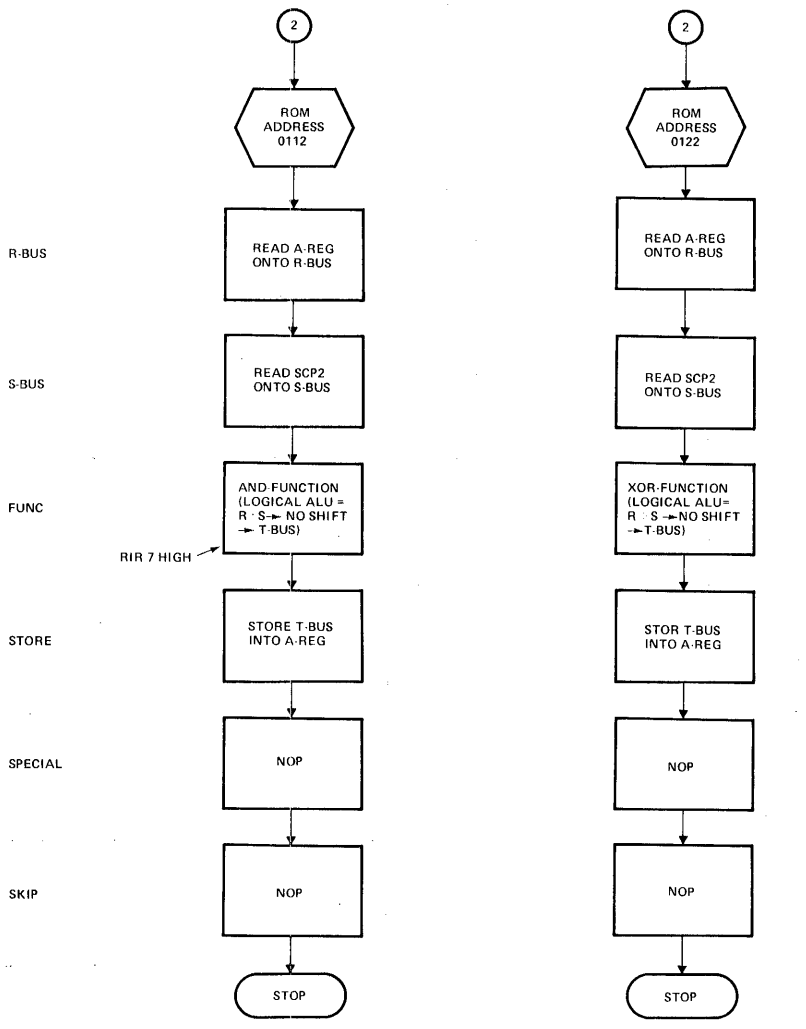
ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0110	75770757	AND	–	S1	IOR	M	RW	–
0111	53773375		AAB	COND	IOR	S2	–	EOP
0112	05147377		A	S2	AND	A	–	–
0120	75770757	XOR	–	S1	IOR	M	RW	–
0121	53773375		AAB	COND	IOR	S2	–	EOP
0122	05167377		A	S2	XOR	A	–	–

AND – INSTRUCTION

R	S	FUNC	ST	SP	SK
000	101 1	01 100	111 0	11 11	1 111

XOR – INSTRUCTION

R	S	FUNC	ST	SP	SK
000	101 0	01 110	111 0	11 11	1 111



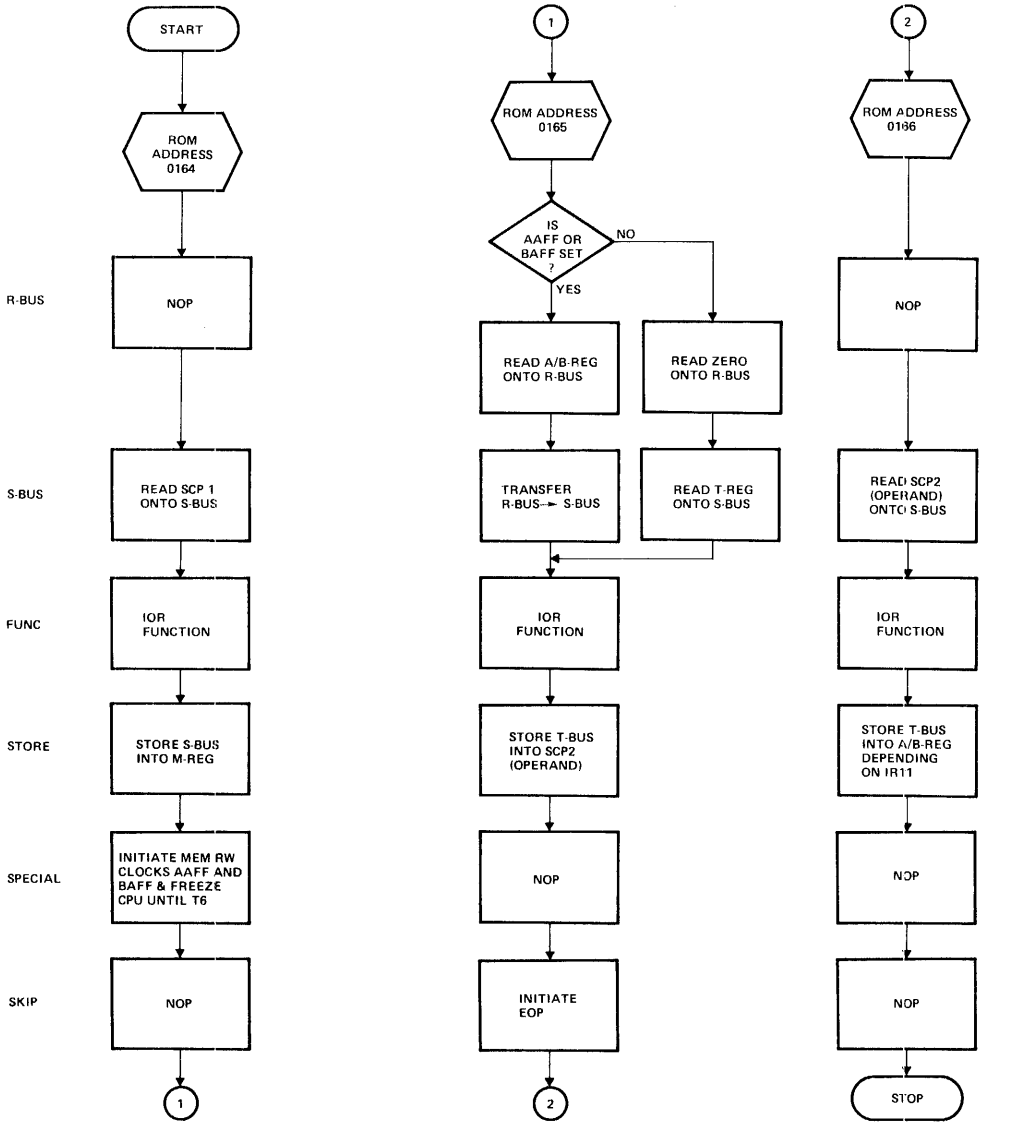
FLOW CHART OF LD* – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0164	75770757	LD*	–	S1	IOR	M	RW	–
0165	53773375		AAB	COND	IOR	S2	–	EOP
0166	75375777		–	S2	IOR	CAB	–	–

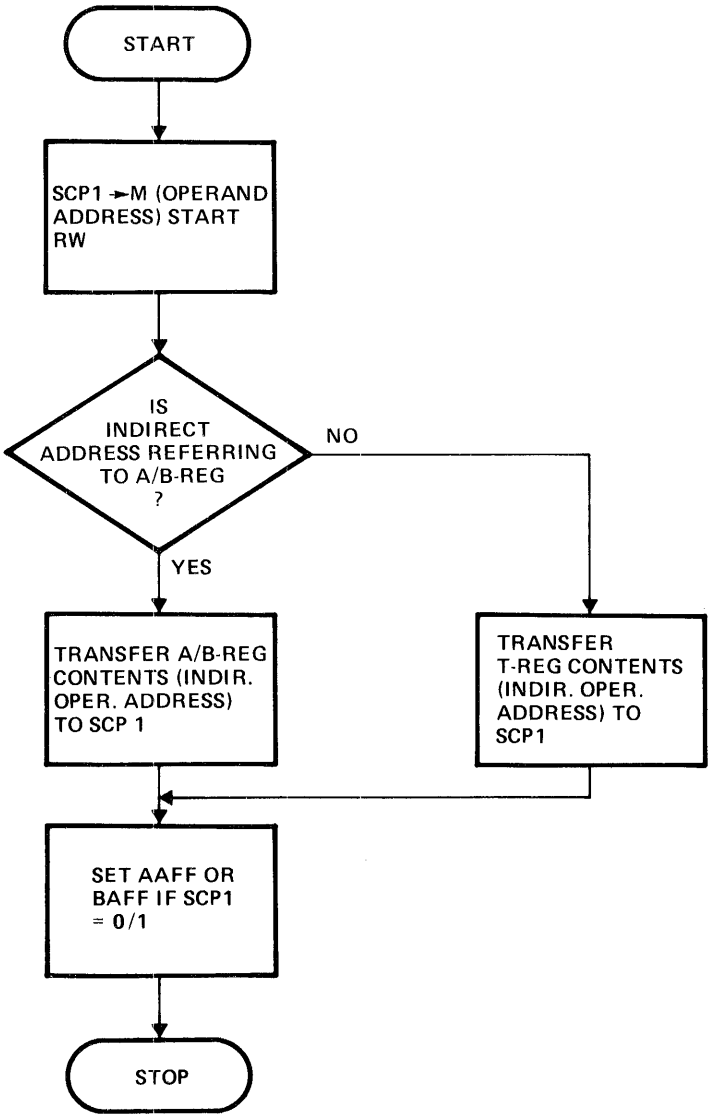
R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 111

R	S	FUNC	ST	SP	SK
101	011 1	11 111	011 0	11 11	1 101

R	S	FUNC	ST	SP	SK
111	101 0	11 111	101 1	11 11	1 111



INDIRECT PHASE



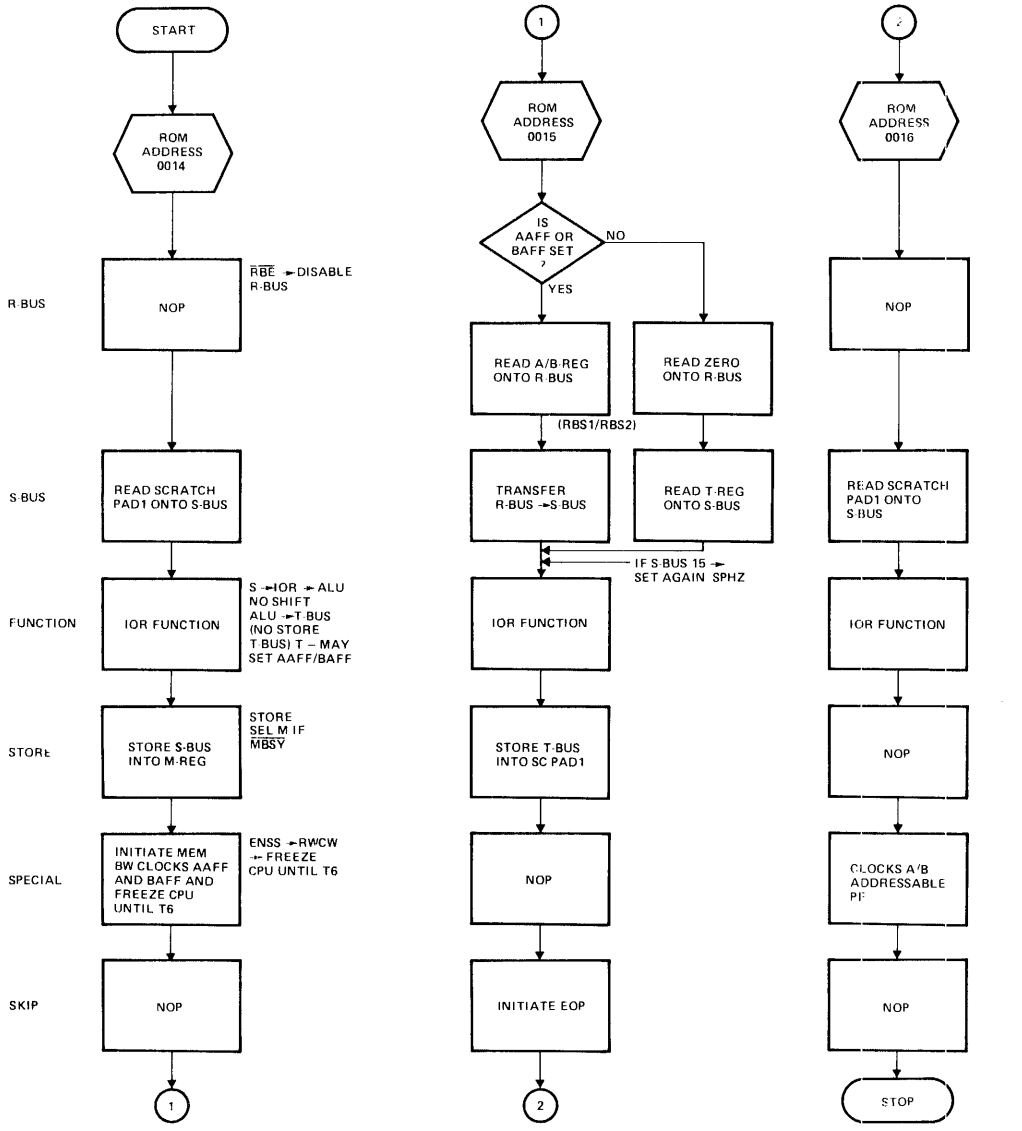
FLOW CHART OF PH2

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0014	75770757	PH2	–	S1	IOR	M	RW	–
0015	53773775		AAB	COND	IOR	S1	–	EOP
0016	75777557		–	S1	IOR	–	AAB	–

R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 111

R	S	FUNC	ST	SP	SK
101	011 1	11 111	011 1	11 11	1 101

R	S	FUNC	ST	SP	SK
111	101 1	11 111	111 1	01 10	1 111



(CAN NOT LOOP ON PHASE 2 BECAUSE $SCP1_{(STORE)}$ IS OVERWRITTEN BY AAB/COND $SCP1_{(S\ BUS)}$ IS ALTERED)

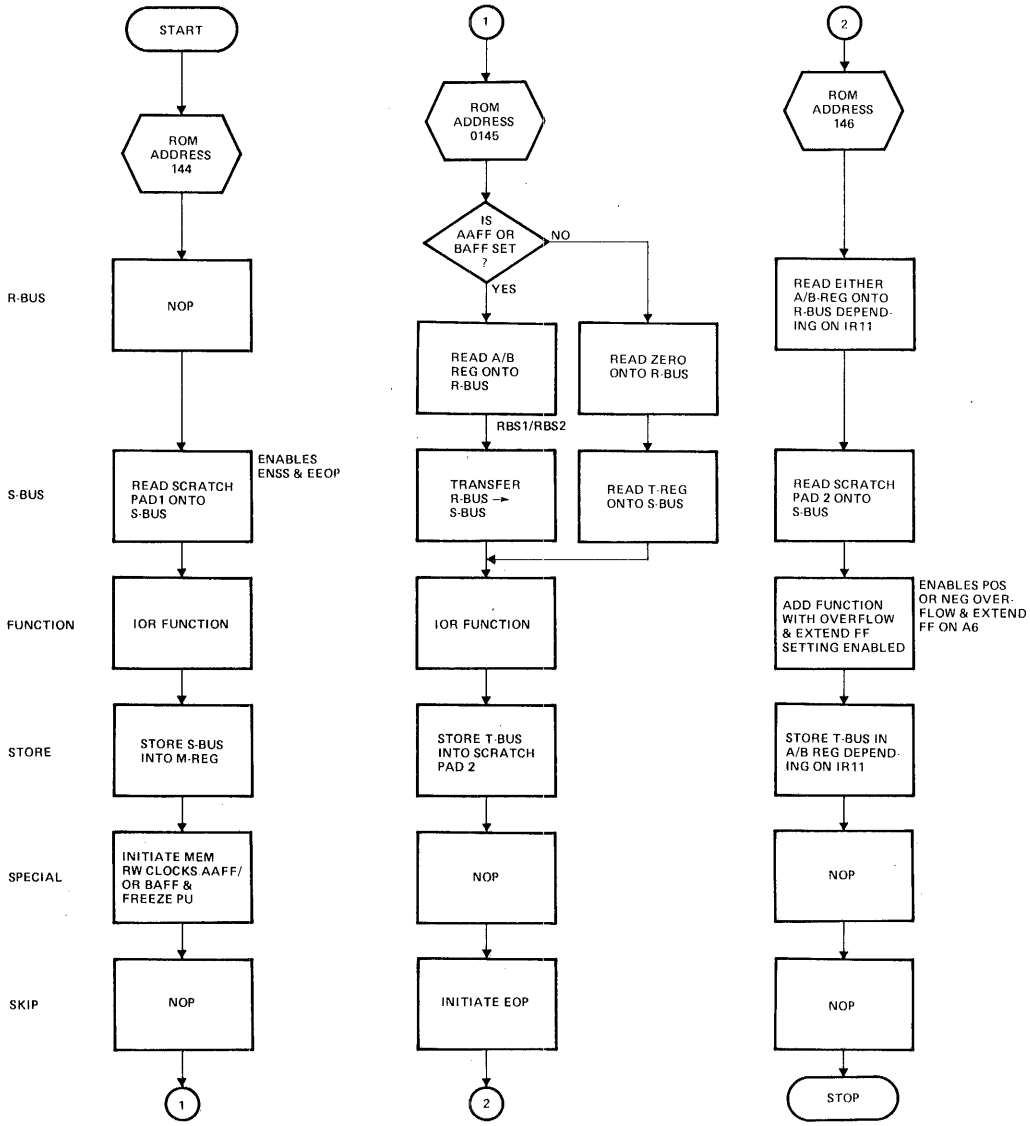
FLOW CHART OF ADD – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0144	75770757	ADD	–	S1	IOR	M	RW	–
0145	53773375		AAB	COND	IOR	S2	–	EOP
0146	45125777		CAB	S2	ADDO	CAB	–	–

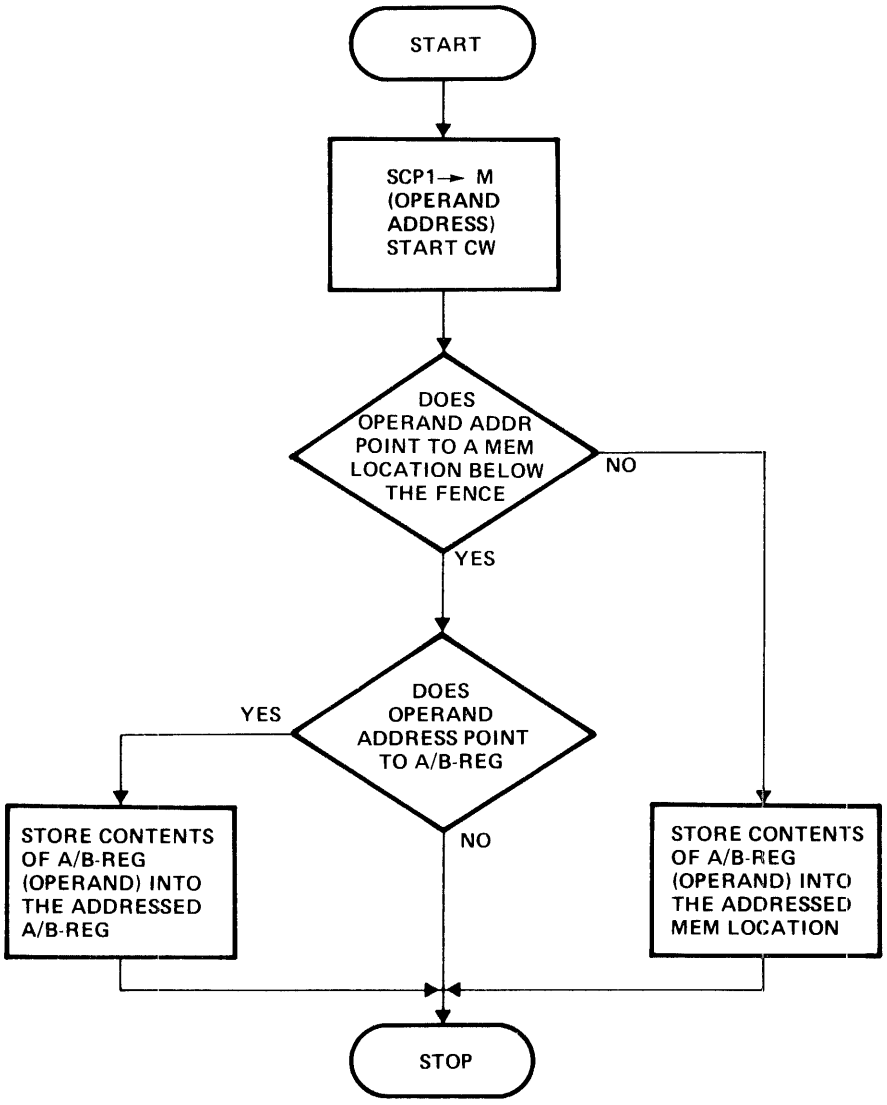
R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 111

R	S	FUNC	ST	SP	SK
101	011 1	11 111	011 0	11 11	1 101

R	S	FUNC	ST	SP	SK
100	101 0	01 010	101 1	11 11	1 111



STA/B – INSTRUCTION



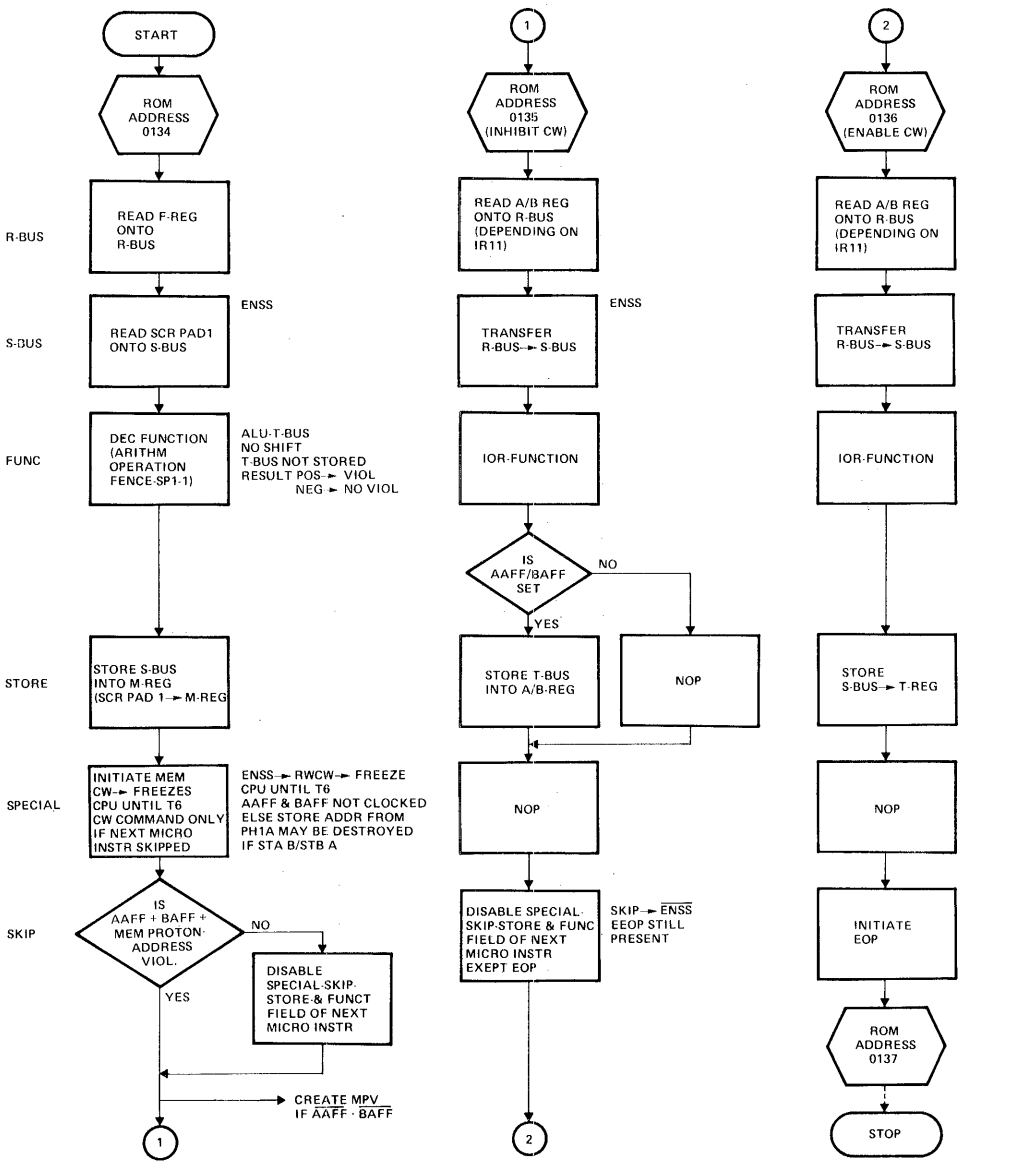
FLOW CHART OF ST* – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0134	35460712	ST*	F	S1	DEC	M	CW	NMPV
0135	42376376		CAB	RRS	IOR	AAB	–	UNC
0136	42371375		CAB	RRS	IOR	T	–	(EOP)
0137	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
011	101 1	00 110	000 1	11 00	1 010

R	S	FUNC	ST	SP	SK
100	010 0	11 111	110 0	11 11	1 110

R	S	FUNC	ST	SP	SK
100	010 0	11 111	001 0	11 11	1 101



ROM SKIPS

IF A ROM SKIP CONDITION IS SATISFIED, IT RESULTS IN THE NULLING OF THE NEXT INSTRUCTION. TO NULL THE INSTRUCTION, THE DECODING OF THE STORE, SPECIAL, AND SKIP FIELDS IS INHIBITED. THE DECODING OF SOME FUNCTIONS IS ALSO INHIBITED. (JMP, JSB, MPY, SFLG, ETC.)

THE SEQUENCE OF RAR AND RIR CONTENTS IN A MICROPROGRAM IS THE SAME WHETHER AN INSTRUCTION IS SKIPPED OR NOT!

EOP AND LEP CAN NEVER BE SKIPPED!

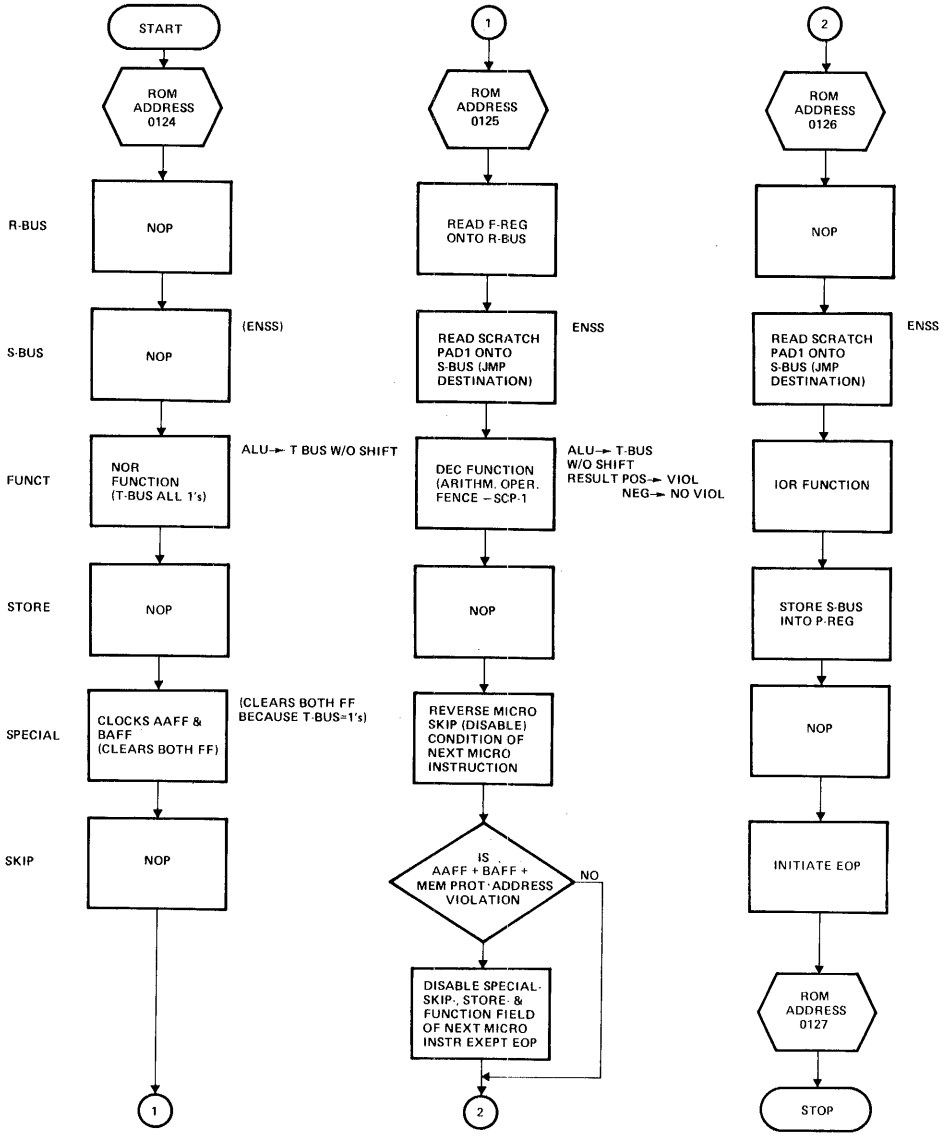
FLOW CHART OF JMP – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0124	77557557	JMP	–	–	NOR	–	AAB	–
0125	35467412		F	S1	DEC	–	RSS	NMPV
0126	75774375		–	S1	IOR	P	–	EOP
0127	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
111	111 1	01 101	111 1	01 10	1 111

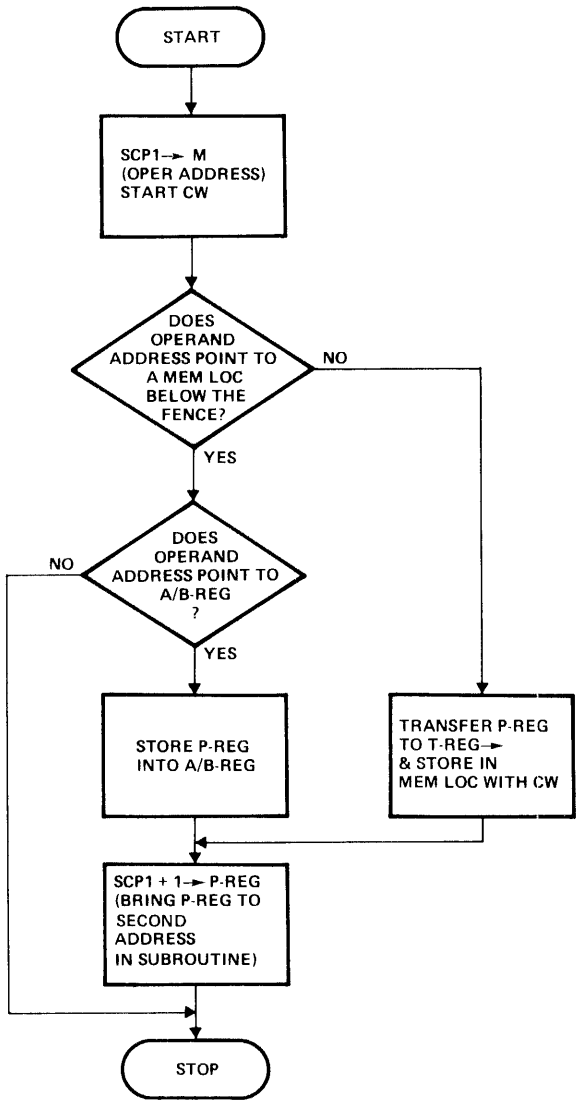
R	S	FUNC	ST	SP	SK
011	101 1	00 110	111 1	00 00	1 010

R	S	FUNC	ST	SP	SK
111	101 1	11 111	100 0	11 11	1 101



JSB – INSTRUCTION

MAIN PROGR.			SUBROUTINE	
LOC.	OP. CODE		LOC.	OP CODE
125	LDA		200	127 NOP
126	JSB	200	201	SZA
127	ADA		202	INA
130	HLT		203	JMP 200, I



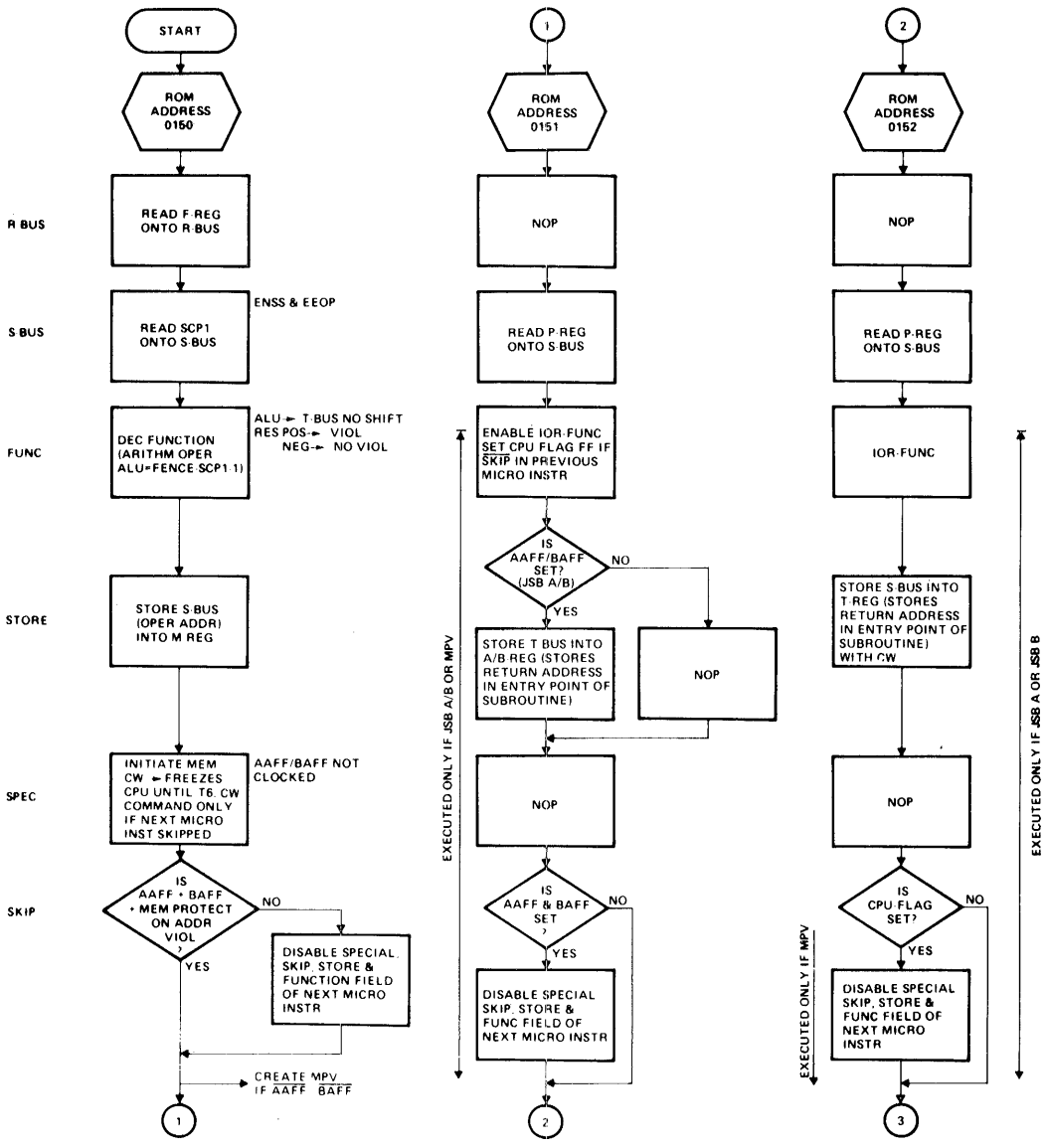
FLOW CHART OF JSB INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R BUS 23 21	S BUS 20 17	FUNCTION 16 12	STORE 11 8	SPECIAL 7 4	SKIP 3 0
0150	35460712	JSB	F	S1	DEC	M	CW	NMPV
0151	77346373		–	P	SFLG	AAB		AAB
0152	77371366		–	P	IOR	T		FLG
0153	75514375			S1	INC	P		EOP
0154	77777777				IOR			

R	S	FUNC	ST	SP	SK
011	101 1	00 110	000 1	11 00	1 010

R	S	FUNC	ST	SP	SK
111	111 0	11 100	110 0	11 11	1 011

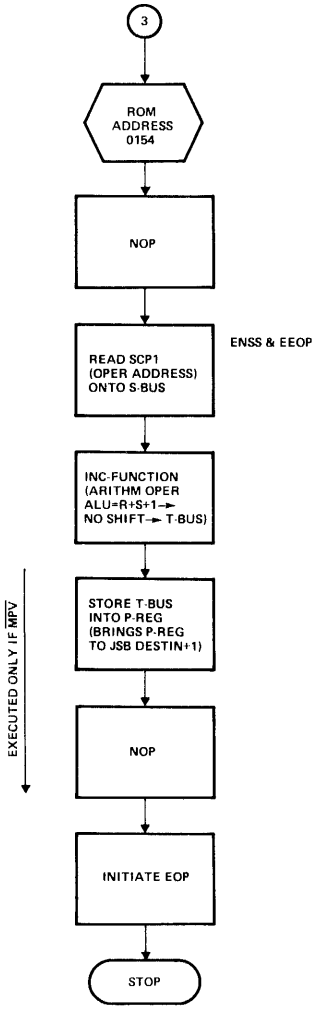
R	S	FUNC	ST	SP	SK
111	111 0	11 111	001 0	11 11	0 110



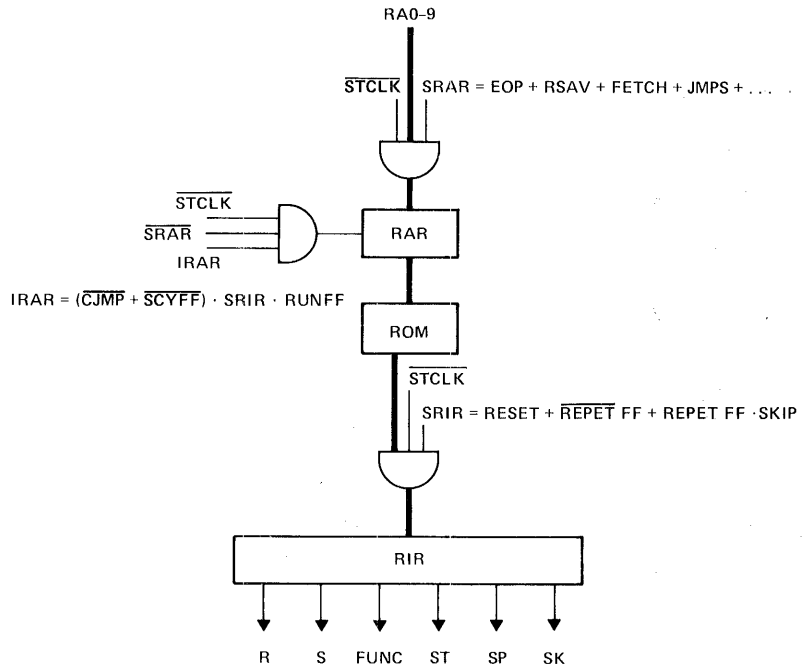
FLOW CHART OF JSB – INSTRUCTION (CONTINUE)

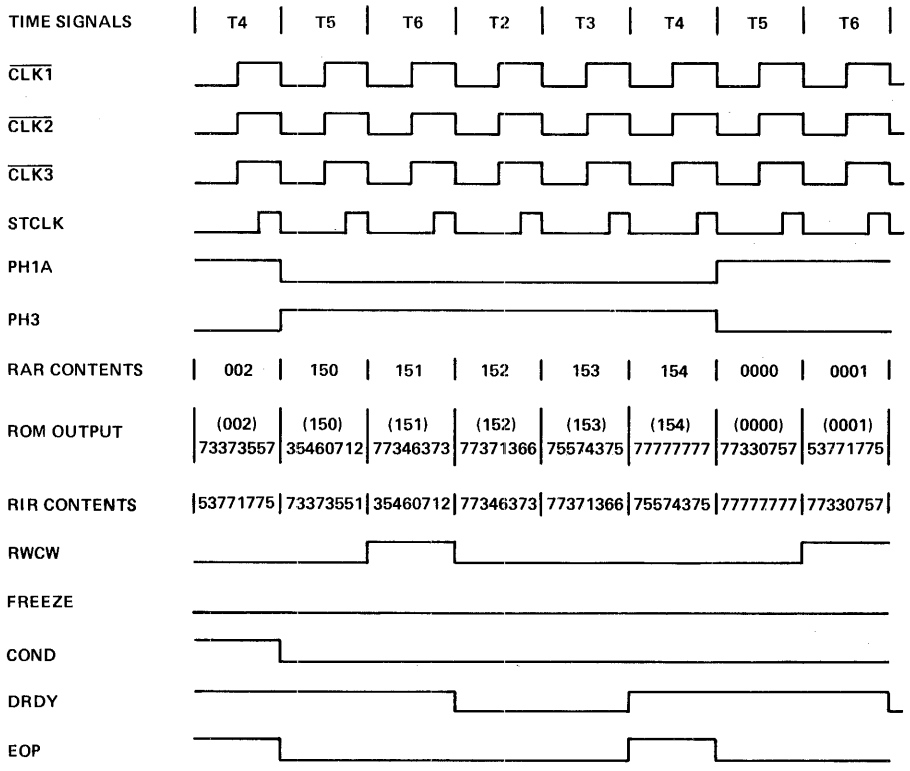
ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0150	35460712	JSB	F	S1	DEC	M	CW	NMPV
0151	77346373		–	P	SFLG	AAB	–	AAB
0152	77371366		–	P	IOR	T	–	FLG
0153	75514375		–	S1	INC	P	–	EOP
0154	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
111	101 1	01 001	100 0	11 11	1 101

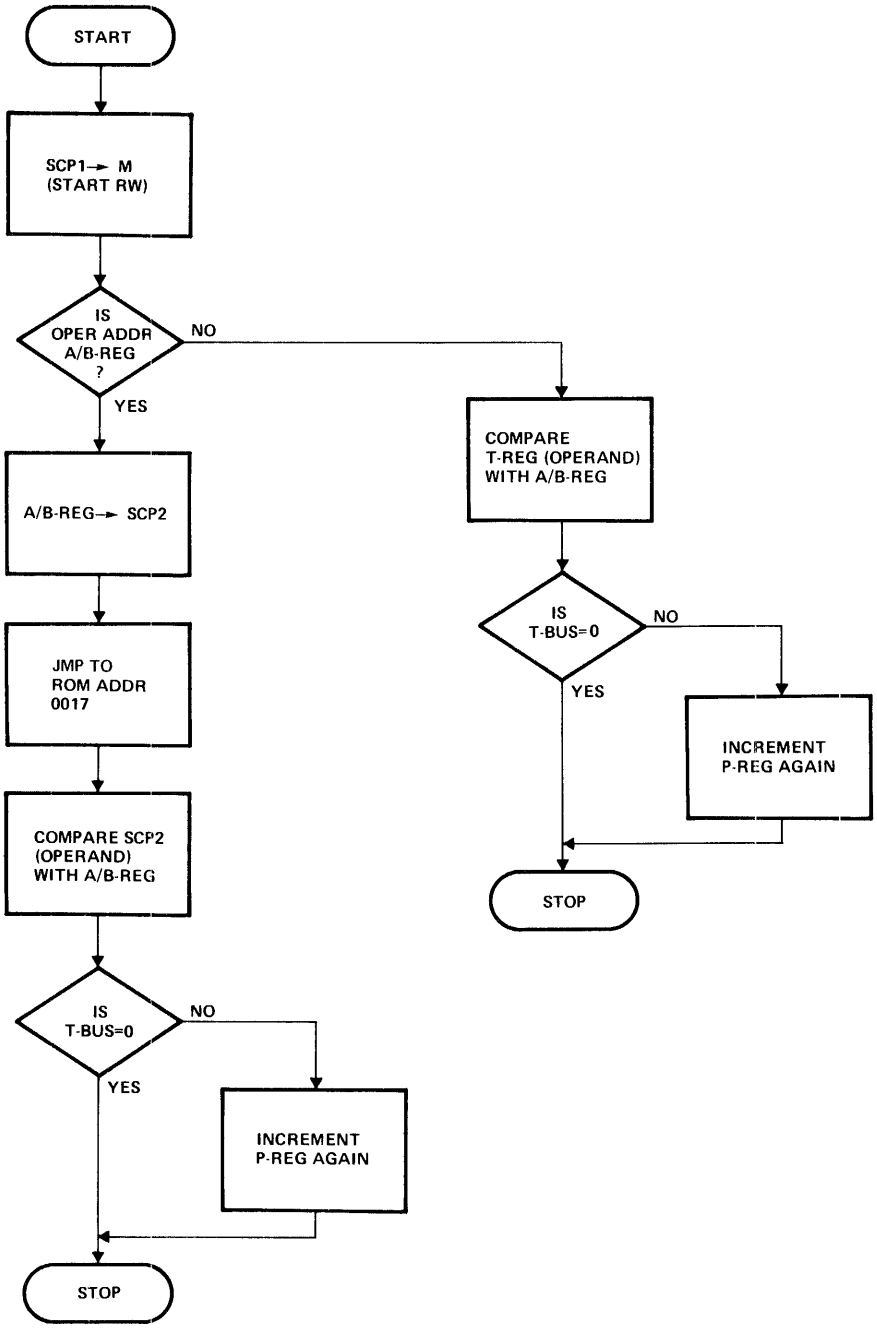


TIMING DIAGRAM OF PH3 JSB-INSTRUCTION





COMPARE INSTRUCTION



ROM JMP, JSB OR CJMP EXECUTIONS

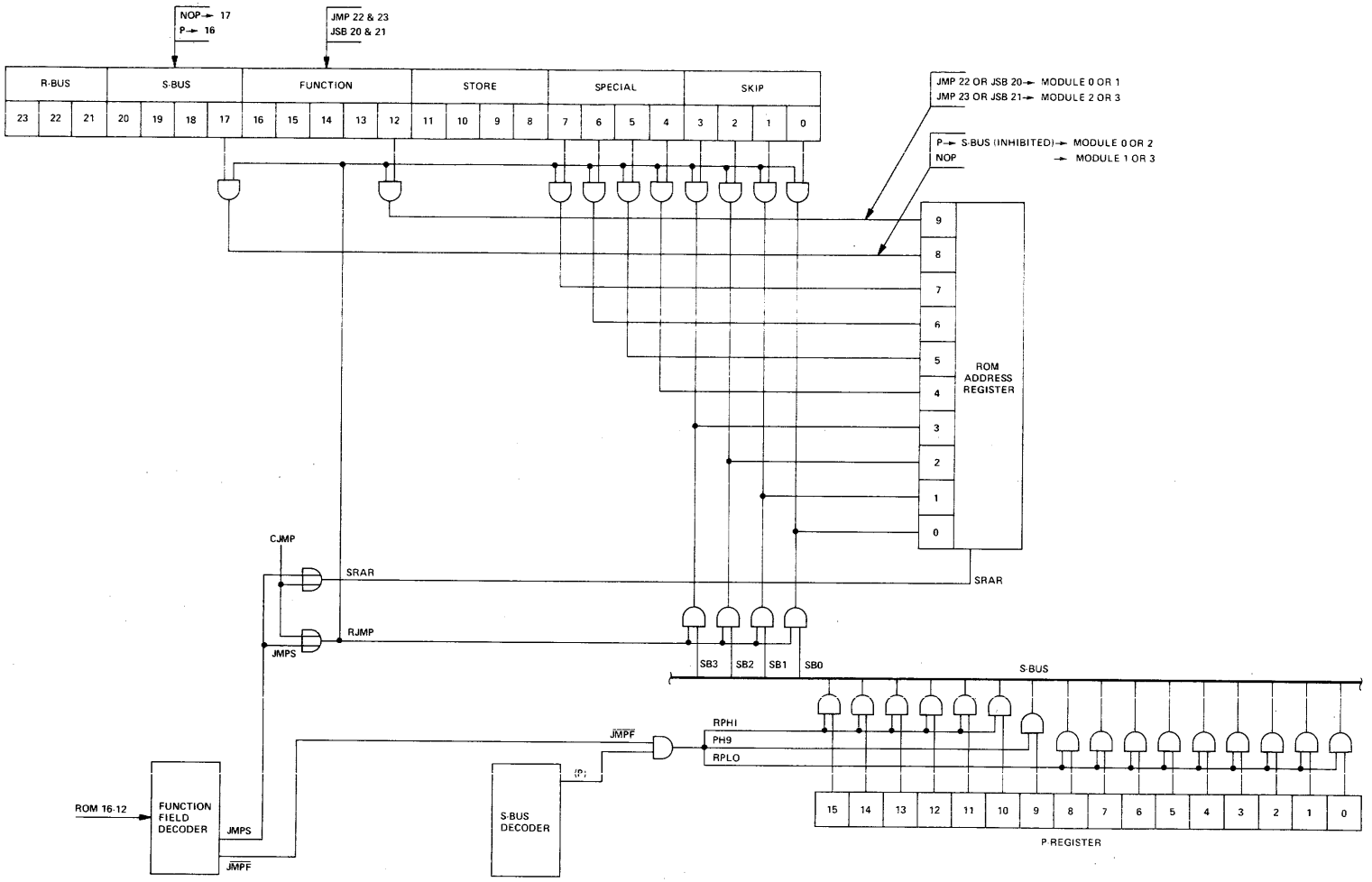
WHEN MICROINSTRUCTIONS CONTAINING JMP, JSB OR CJMP ARE EXECUTED ENRM GOES LOW AND THE OUTPUT FROM ROM IS ALL 1'S (NOP). AT THE FOLLOWING $\overline{\text{STCLK}}$ (END OF JMP, JSB OR CJMP-MICROORDER) TWO MAJOR OPERATIONS ARE PERFORMED:

1. THE NOP IS STORED INTO THE RIR AND THEREFORE NOP'S THE MICRO INSTRUCTION FOLLOWING THE JMP, JSB OR CJMP.
2. THE JMP, JSB OR CJMP DESTINATION ADDRESS (RIR 17, 12, 7–0) IS TRANSFERED TO THE RAR.

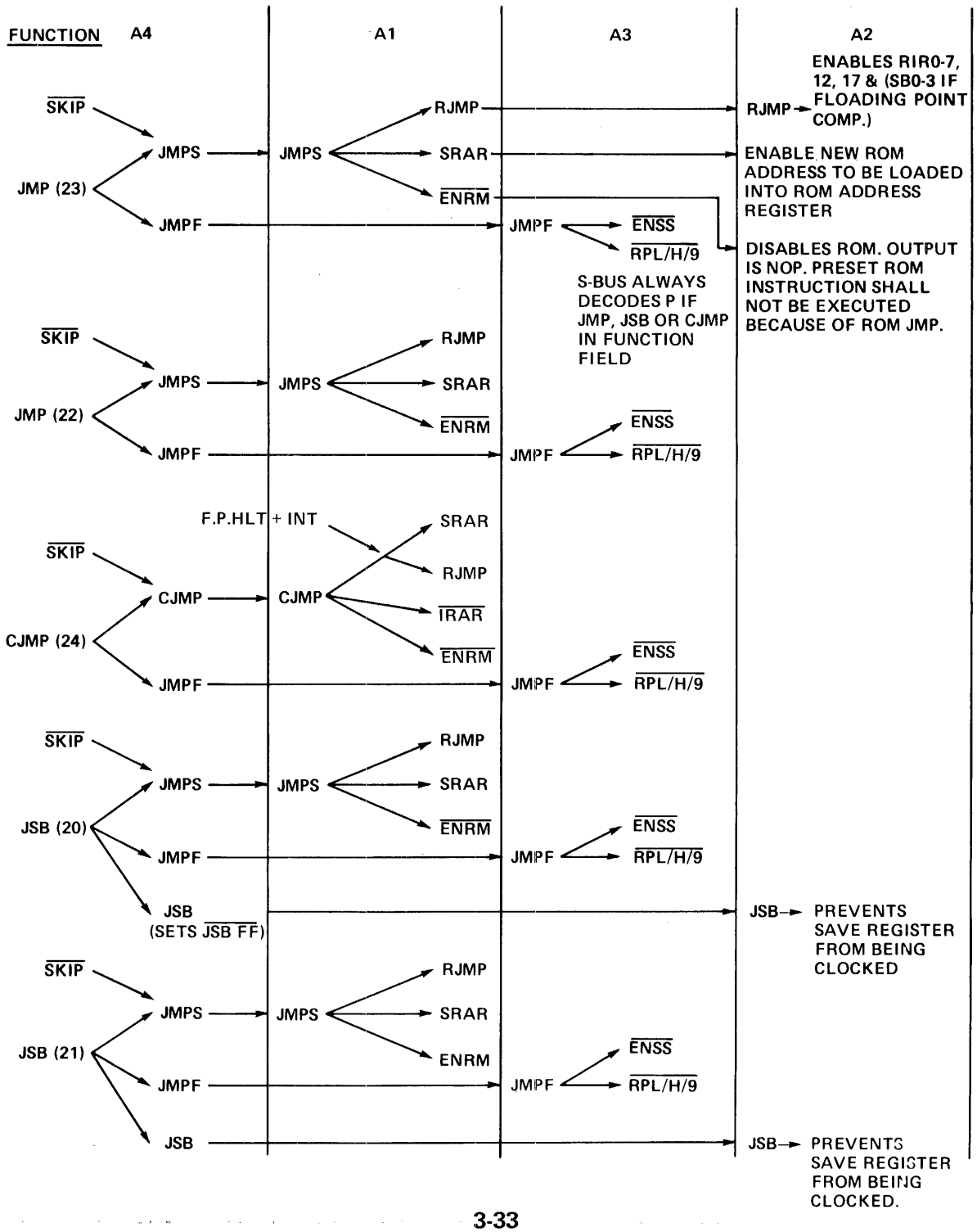
RIR 12	⟶	RAR 9
RIR 17	⟶	RAR 8
RIR 7–0	⟶	RAR 7–0

200 ns LATER (WITH THE NEXT $\overline{\text{STCLK}}$) THE ROM CONTENTS OF THE ADDRESSED ROM LOCATION (IN PARAGRAPH 2) IS CLOCKED INTO THE RIR.

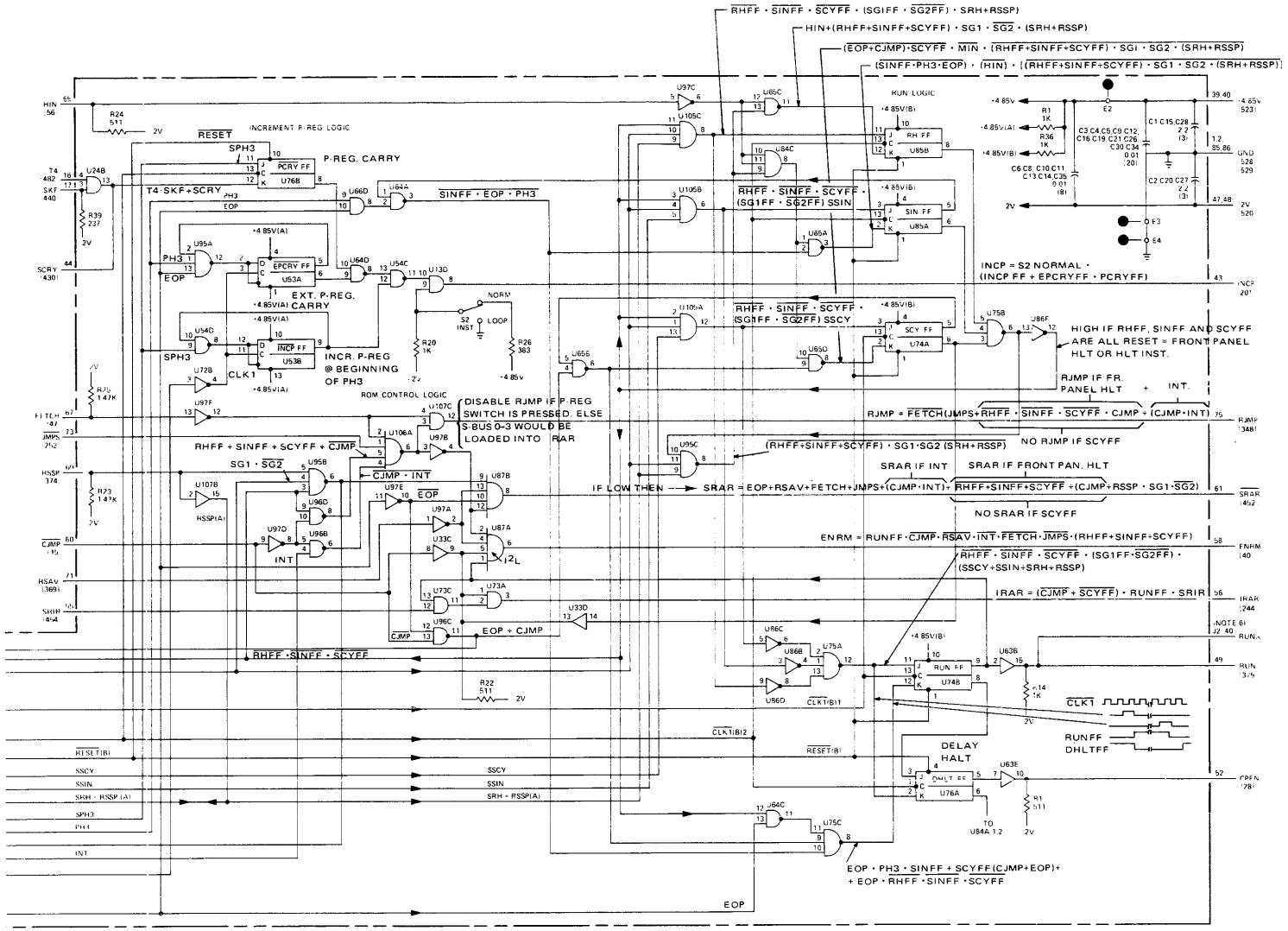
ROM JMP AND JSB CONTROL SIGNALS



HARDWARE CONTROL SIGNALS CREATED BY ROM JMP, JSB AND CJMP



ROM CONTROL LOGIC



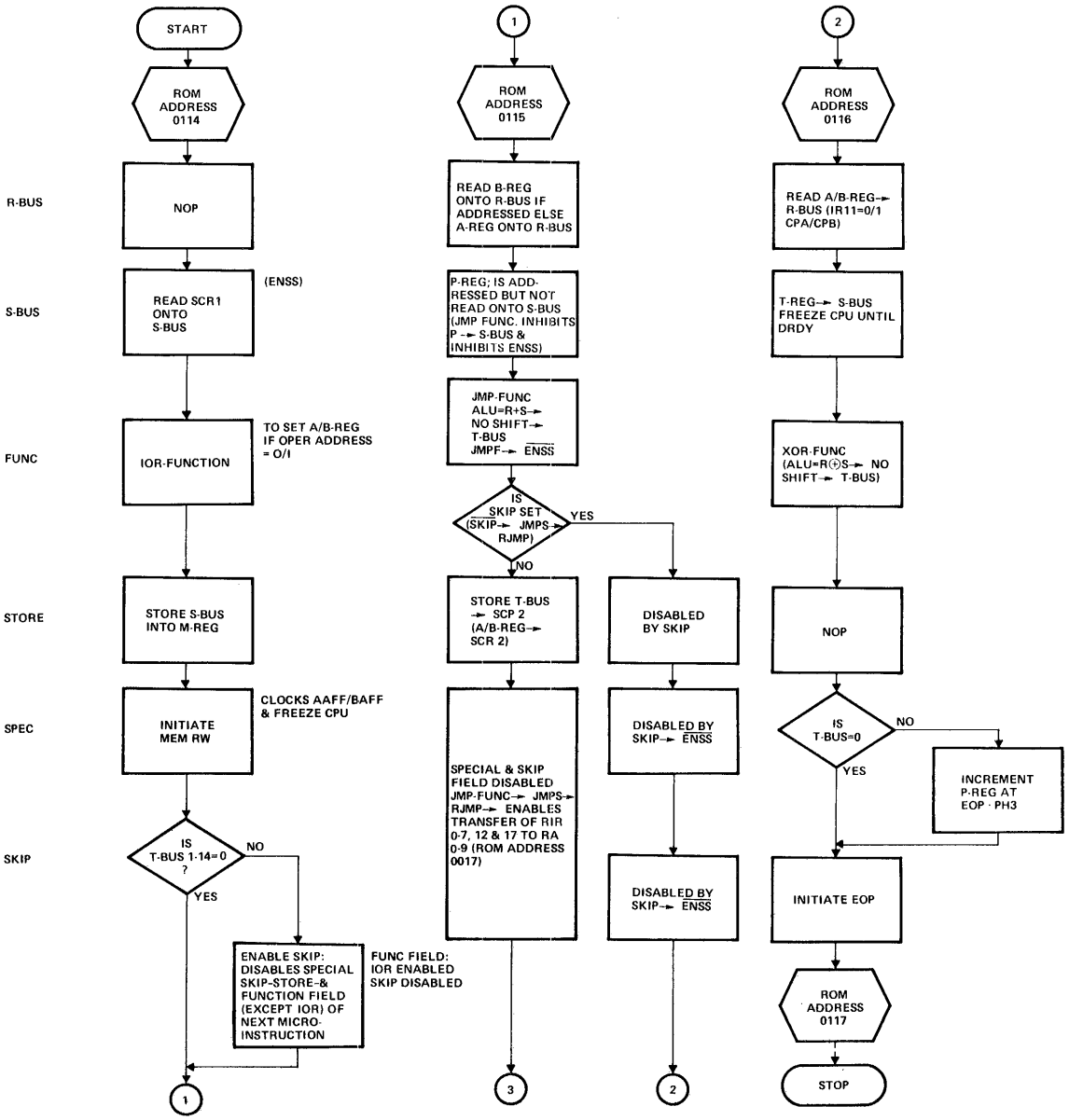
FLOW CHART OF CP* – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0114	75770754	CP*	–	S1	IOR	M	RW	NAAB
0115	57223017		AAB	P	JMP	S2	XX	–
0116	41167635		CAB	T	XOR	–	ECYN	EOP
0117	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 100

R	S	FUNC	ST	SP	SK
101	111 0	10 010	011 0	00 00	1 111

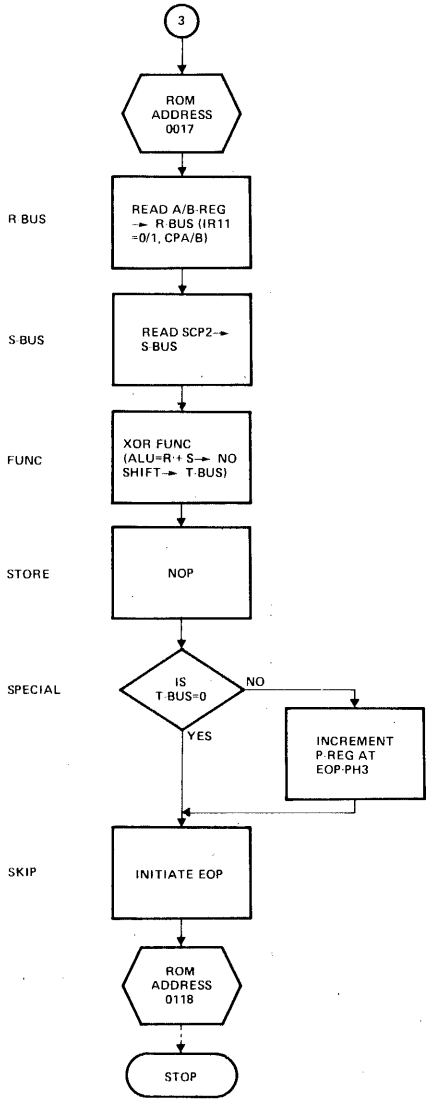
R	S	FUNC	ST	SP	SK
100	001 0	01 110	111 1	10 01	1 101



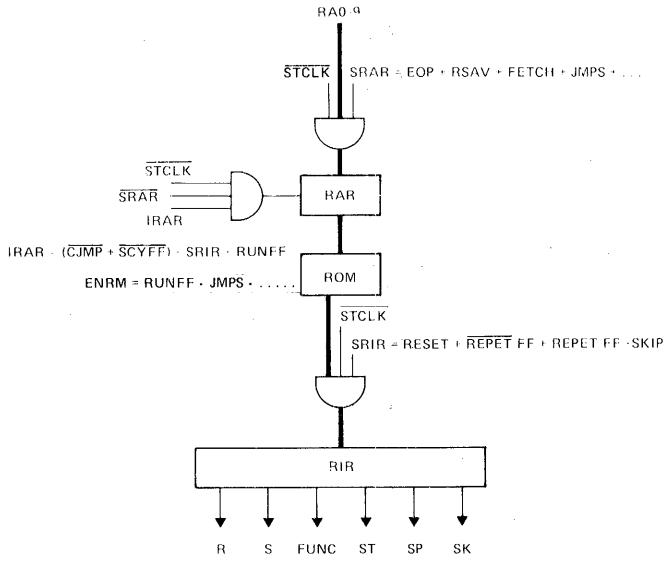
FLOW CHART OF CP* – INSTRUCTION (CONTINUE)

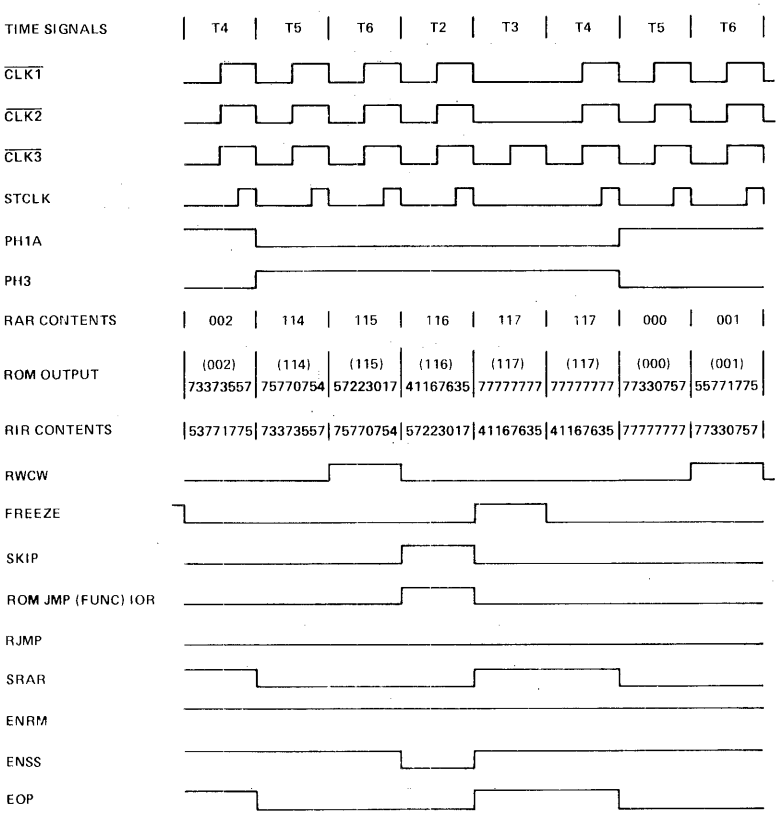
ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0017	45167635	XX	CAB	S2	XOR	–	ECYN	EOP
0020	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
100	101 0	01 110	111 1	10 01	1 101

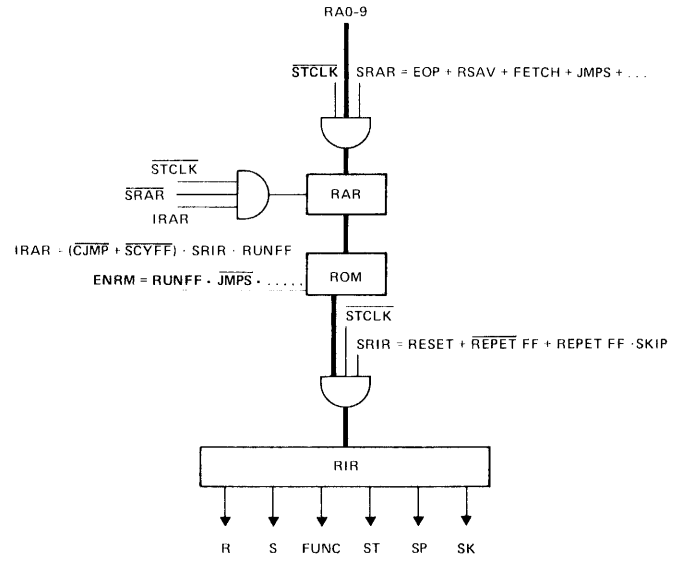


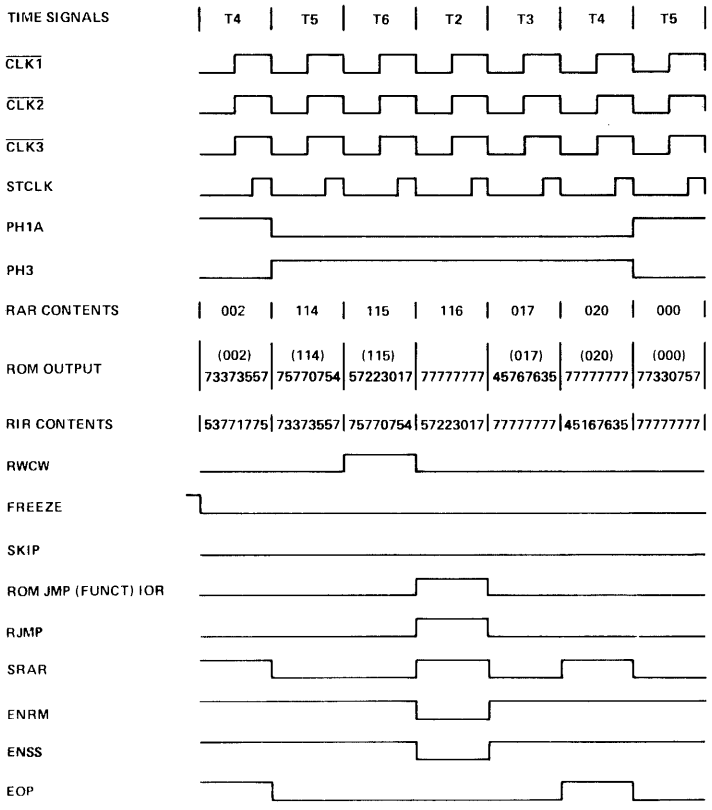
TIMING DIAGRAM OF PH3, CP*-INSTRUCTION
(OPERAND ADDRESS IS NOT IN A/B-REGISTER)



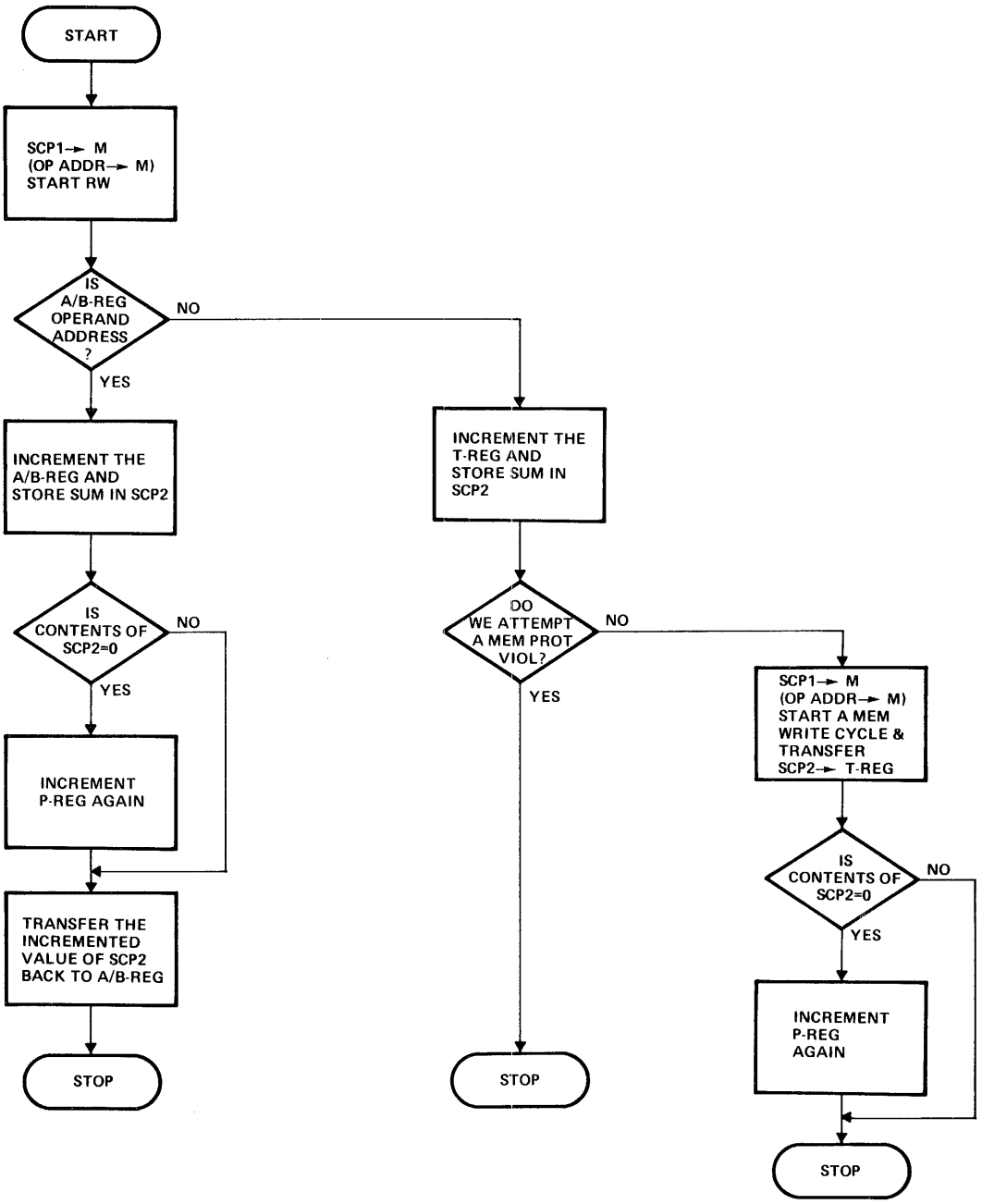


TIMING DIAGRAM OF PH3, CP*-INSTRUCTION
(OPERAND ADDRESS IS IN A/B-REGISTER)





ISZ INSTRUCTION



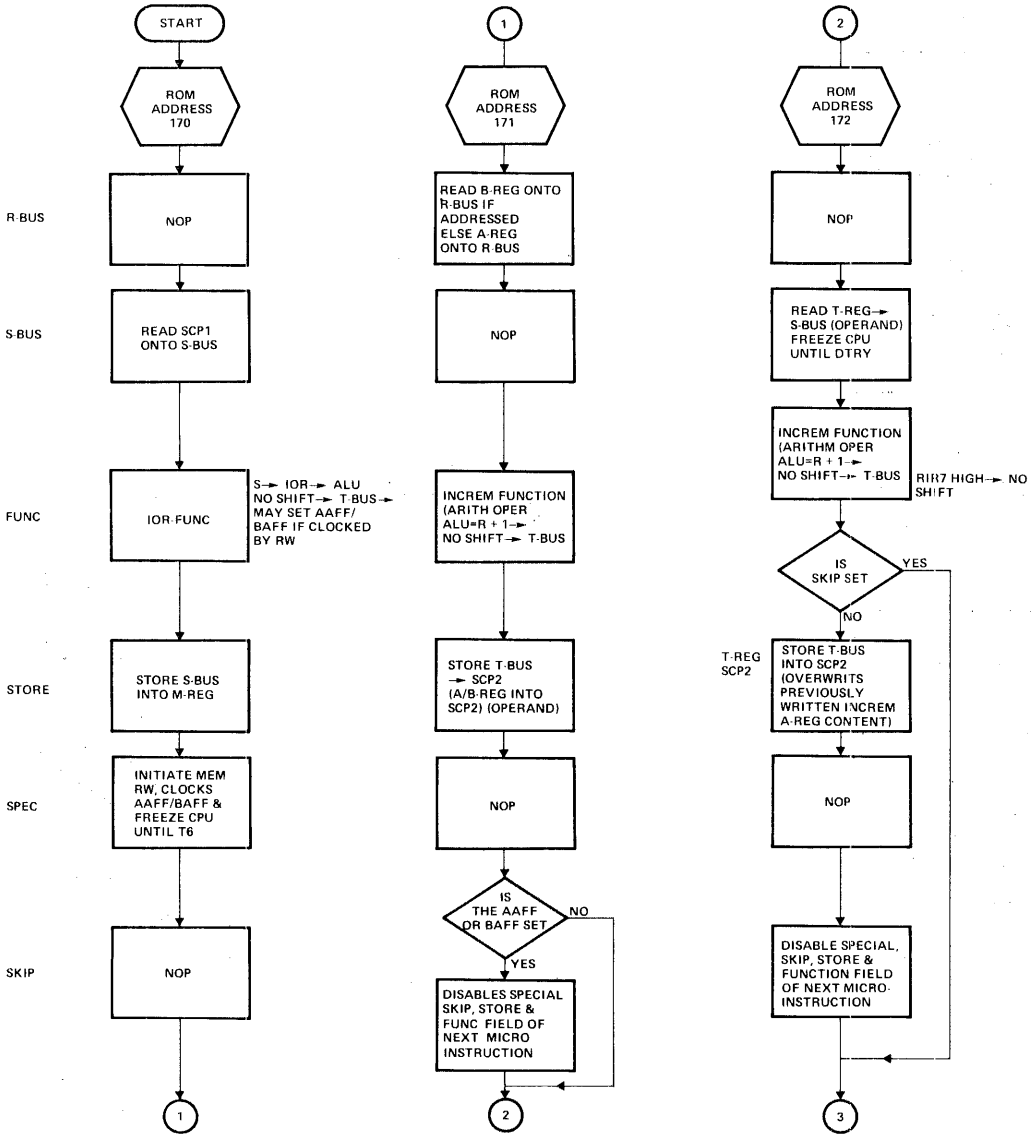
FLOW CHART OF ISZ – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0170	75770757	ISZ	–	S1	IOR	M	RW	–
0171	57513373		AAB	–	INC	S2	–	AAB
0172	71113376		–	T	INC	S2	–	UNC
0173	75377616		–	S2	IOR	–	ECYZ	UNC
0174	35460712		F	S1	DEC	M	CW	NMPV
0175	75376376		–	S2	IOR	AAB	–	UNC
0176	75371215		–	S2	IOR	T	ECYZ	EOP
0177	77777777		–	–	IOR	–	–	–
0200	77777775		–	–	IOR	–	–	EOP

R	S	FUNC	ST	SP	SK
111	101 1	11 111	000 1	11 10	1 111

R	S	FUNC	ST	SP	SK
101	111 1	01 001	011 0	11 11	1 011

R	S	FUNC	ST	SP	SK
111	001 0	01 001	011 0	11 11	1 110



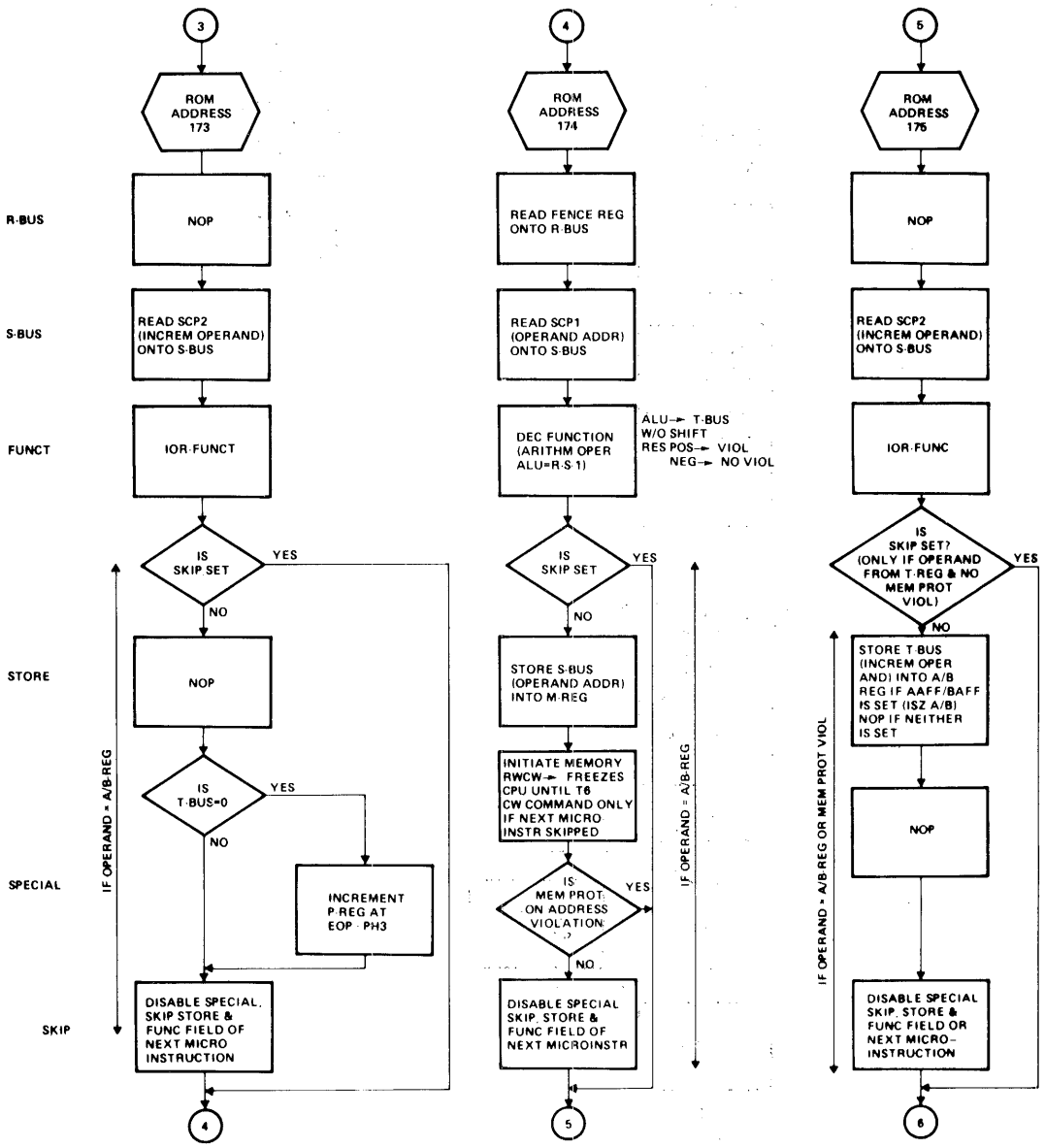
FLOW CHART OF ISZ – INSTRUCTION (CONTINUE 1)

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0170	75770757	ISZ	–	S1	IOR	M	RW	
0171	57513373		AAB	–	INC	S2		AAB
0172	71113376		–	T	INC	S2		UNC
0173	75377616		–	S2	IOR	–	ECYZ	UNC
0174	35460712		F	S1	DEC	M	CW	NMPV
0175	75376376		–	S2	IOR	AAB	–	UNC
0176	75371215		–	S2	IOR	T	ECYZ	EOP
0177	77777777		–	–	IOR	–	–	
0200	77777775		–	–	IOR	–	–	EOP

R	S	FUNC	ST	SP	SK
111	101 0	11 111	111 1	10 00	1 110

R	S	FUNC	ST	SP	SK
011	101 1	00 110	000 1	11 00	1 010

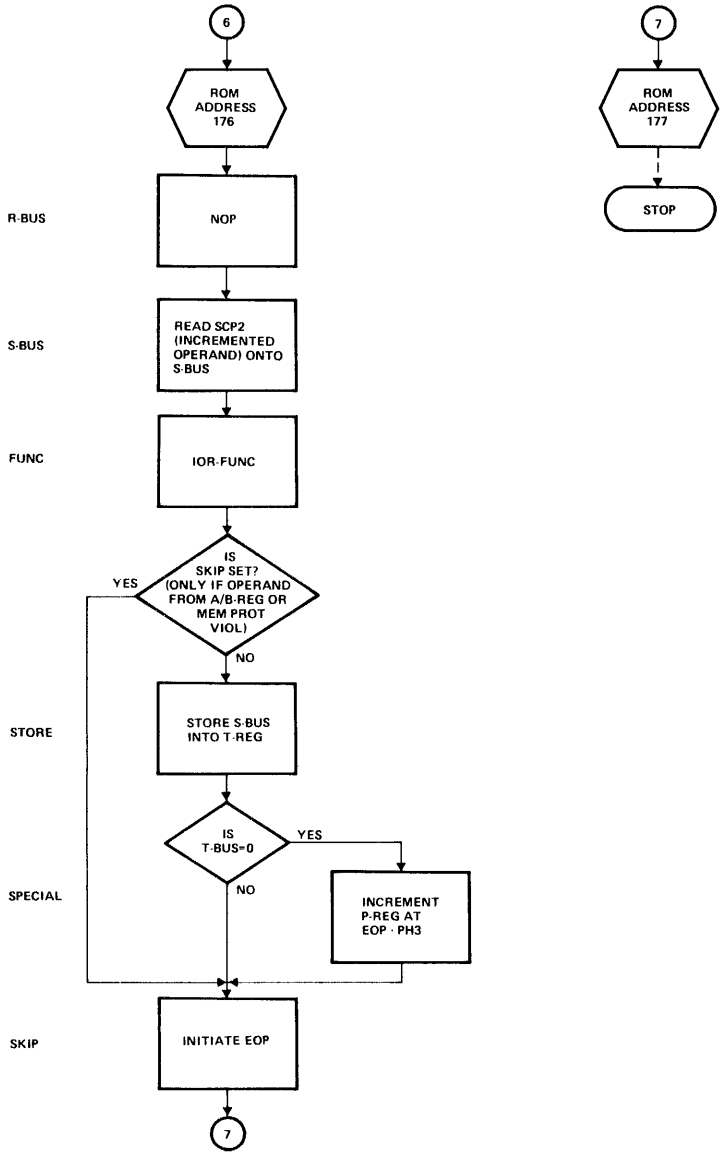
R	S	FUNC	ST	SP	SK
111	101 0	11 111	110 0	11 11	1 110



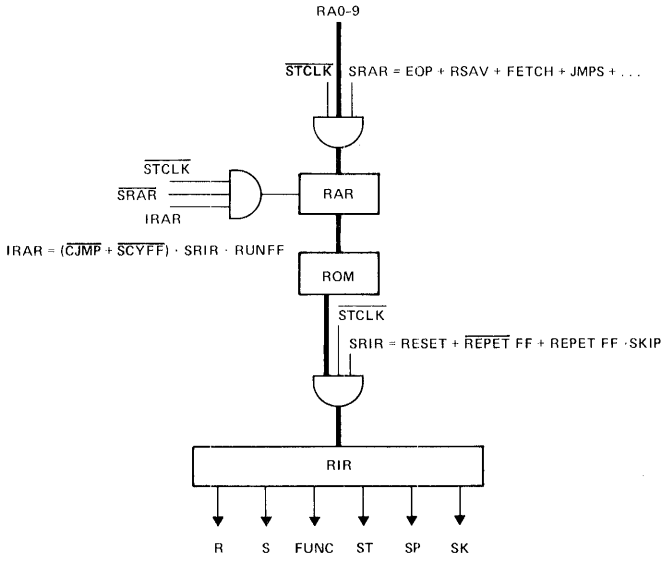
FLOW CHART OF ISZ – INSTRUCTION (CONTINUE 2)

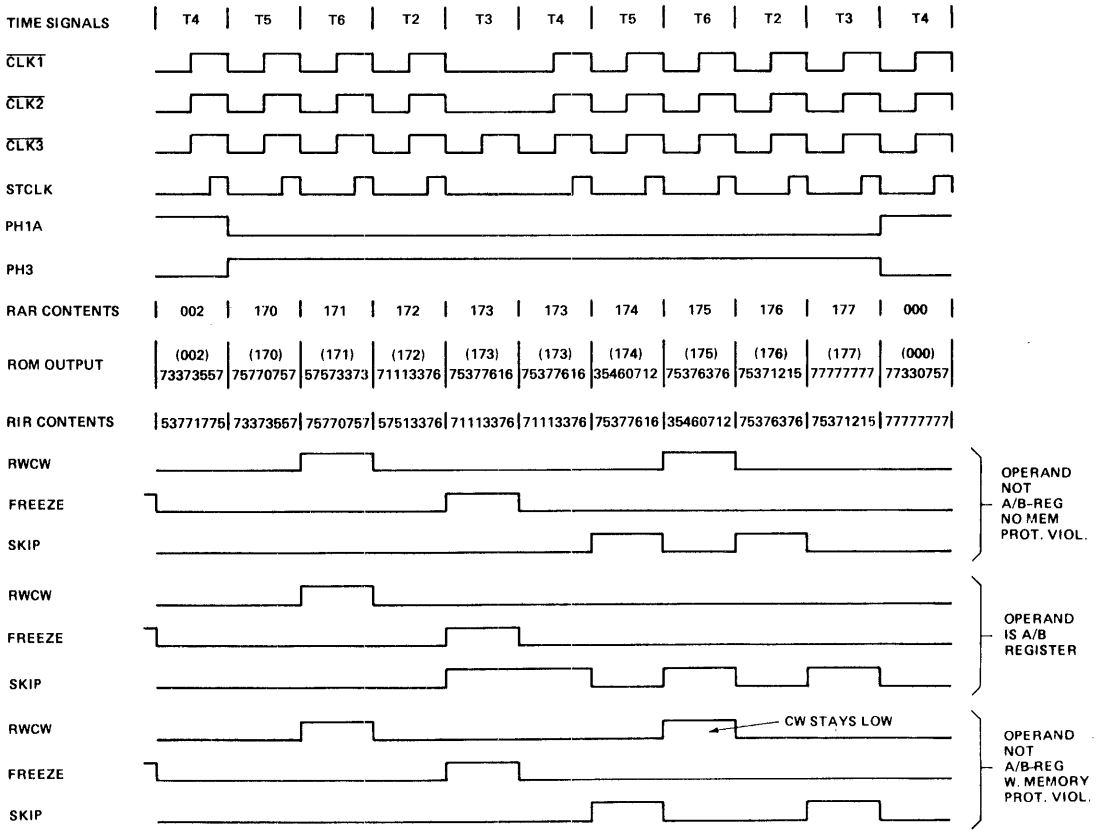
ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0170	75770757	ISZ	–	S1	IOR	M	RW	–
0171	57513373		AAB	–	INC	S2	–	AAB
0172	71113376		–	T	INC	S2	–	UNC
0173	75377616		–	S2	IOR	–	ECYZ	UNC
0174	35460712		F	S1	DEC	M	CW	NMIPV
0175	75376376		–	S2	IOR	AAB	–	UNC
0176	75371215		–	S2	IOR	T	ECYZ	EOP
0177	77777777		–	–	IOR	–	–	–
0200	77777775		–	–	IOR	–	–	EOP

R	S	FUNC	ST	SP	SK
111	101 0	11 111	001 0	1000	1 101



TIMING DIAGRAM OF PH3, ISZ INSTRUCTION





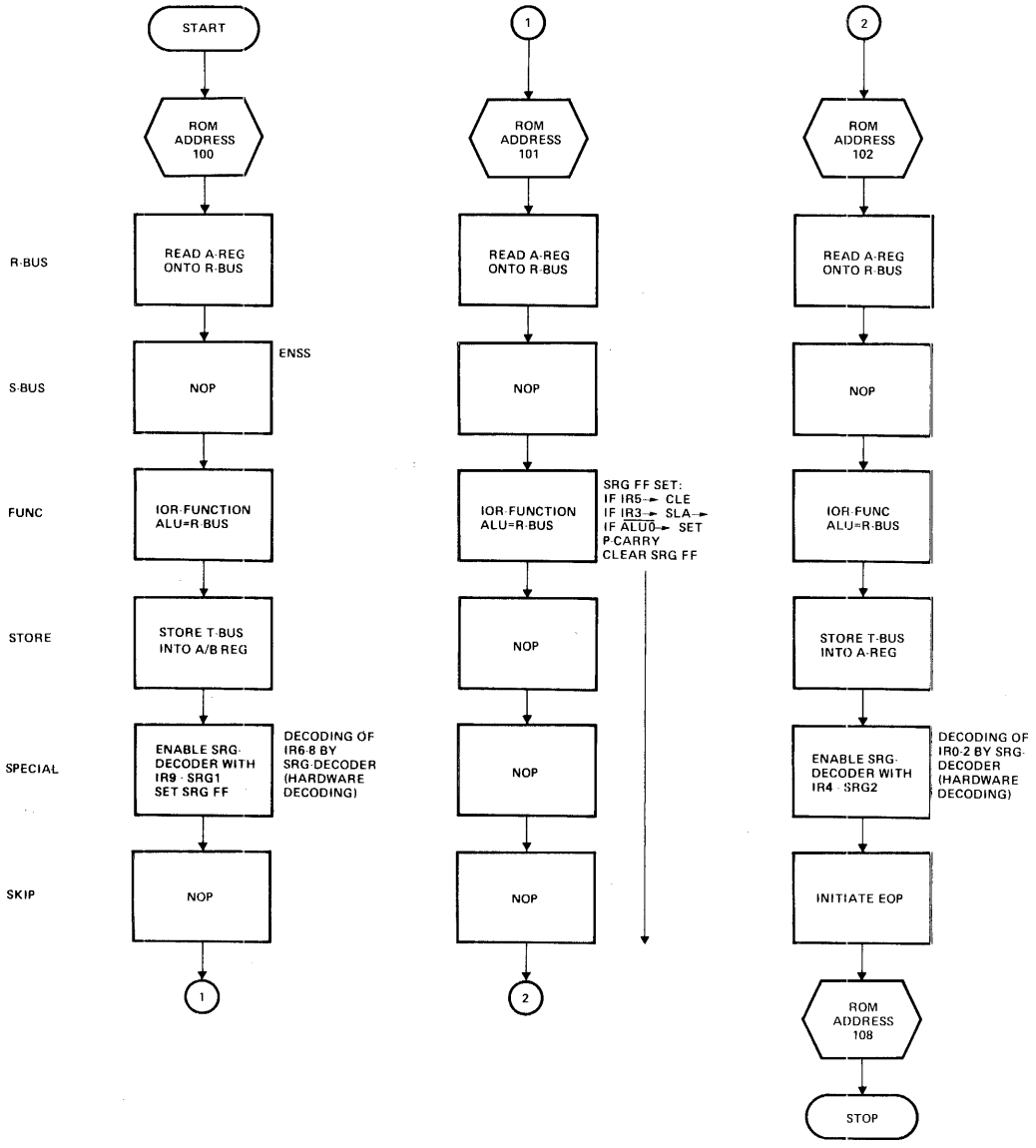
FLOW CHART OF SRG – INSTRUCTIONS

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0
0100	07777117	SRGA	A	–	IOR	A	SRG1	–
0101	07777777		A	–	IOR	–	–	–
0102	07777135		A	–	IOR	A	SRG2	EOP
0103	77777777		–	–	IOR	–	–	–

R	S	FUNC	ST	SP	SK
000	111 1	11 111	111 0	01 00	1 111

R	S	FUNC	ST	SP	SK
000	111 1	11 111	111 1	11 11	1 111

R	S	FUNC	ST	SP	SK
000	111 1	11 111	111 0	01 01	1 101



HARDWARE SIGNALS TO IMPLEMENT SRG-INSTRUCTIONS

I-REGISTER BIT 9876 4210	1000	1001	1010	1011	1100	1101	1110	1111
INSTRUCTION	*LS	*RS	R*L	R*R	*LR	ER*	EL*	*LF
SL1	X		X		X		X	
SR1		X		X		X		X
SL4								X
TBS1		X		X		X		X
TBS2	X		X		X		X	X
ALX14→ (ALU15→ TB15)	X							
(ALU14→ TB15)			X				X	
ALX16→ (ALU15→ TB15)		X						
(ALU0→ TB15)				X				
(EXT→ TB15)						X		
LSI → (ALU15→ TB0)			X					
(EXT→ TB0)							X	
SRG1/2→ (ALU15→ EXT)							X	
(ALU0→ EXT)						X		

ASG-INSTRUCTIONS

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0021	47777657	ASGD	CAB	–	IOR	–	ASG1	–
0022	47525675		CAB	–	ADDO	CAB	ASG2	EOP
0023	77777777		–	–	IOR	–	–	–
0025	77775657	ASGA	–	–	IOR	CAB	ASG1	–
0026	47525675		CAB	–	ADDO	CAB	ASG2	EOP
0027	77777777		–	–	IOR	–	–	–
0031	47555657	ASGB	CAB	–	NOR	CAB	ASG1	–
0032	47525675		CAB	–	ADDO	CAB	ASG2	EOP
0033	77777777		–		IOR	–	–	–
0035	77555657	ASGC	–	–	NOR	CAB	ASG1	–
0036	47525675		CAB	–	ADDO	CAB	ASG2	EOP
0037	77777777		–	–	IOR	–	–	–

A/B-REG (DEPENDING ON IR11) → R-BUS→ ALU-BUS FOR POSSIBLE EXECUTION OF SL*/SS* - INSTRUCTIONS

CLA/B (DEPENDING ON IR11)→ ALU-BUS FOR POSSIBLE EXECUTION OF SL*/SS* - INSTRUCTIONS

CMA/B (DEPENDING ON IR11)→ ALU-BUS FOR POSSIBLE EXECUTION OF SL*/SS* - INSTRUCTIONS

CCA/B (DEPENDING ON IR11)→ ALU-BUS FOR POSSIBLE EXECUTION OF SL*/SS* - INSTRUCTIONS

INA/B & SZA/B IF PROGRAMMED

FUNCTION OF ASG1 & ASG2

THE MICROORDERS ASG1 & ASG2 ENABLE THE EXECUTION OF THE FOLLOWING ASG-INSTRUCTIONS:

			IC-NR ON A6
ASG1:	SEZ	SEZ · RSS	U17A & B
	CLE		U73C & U97B
	CME		U97C, U91C, U73B & U63B
	CCE		U97A & 63C
	RSS		U54 DECODER
	SL*	SL* · RSS	
	SS*	SS* · RSS	
	SS* · SL*	SS* · SL* · RSS	
ASG2:	IN*		U107A
	SZ*	SZ* · RSS	U61B, C & D, U91A, U32B & D

DECODING ASG INSTRUCTIONS

- THE MAPPER, WHEN DECODING AN ASG-INSTRUCTION WILL OUTPUT ROM ADDRESS 21, 25, 31 OR 35 DEPENDING ON INSTRUCTION REG. BIT 8 & 9 ∴ CL*, CM* OR CC* ARE EXECUTED UNDER ROM CONTROL

$\overline{IR9} \cdot \overline{IR8}$	→	$\overline{CL}^* \cdot \overline{CM}^* \cdot \overline{CC}^*$	→	ROM ADDRESS 21
$\overline{IR9} \cdot IR8$	→	CL*	→	" " 25
$IR9 \cdot \overline{IR8}$	→	CM*	→	" " 31
$IR9 \cdot IR8$	→	CC*	→	" " 35

- ASG1 ENABLES DECODING OF CLE, CME, CCE, SEZ, SS*, SL* & RSS BY HARDWARE

- ASG2 ENABLES DECODING OF IN* & SZ* BY HARDWARE

EXECUTION PRIORITY OF CONCATENATED ALTER-SKIP GROUP INSTRUCTIONS

1.) EXTEND REGISTER

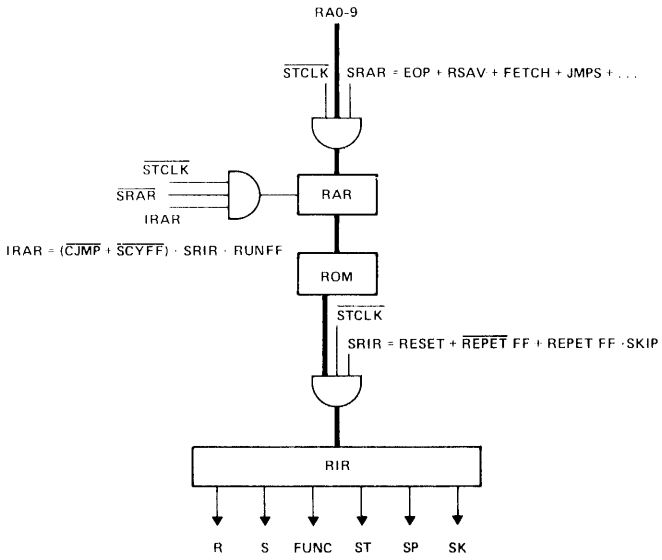
SEZ OR SEZ · RSS IS EXECUTED BEFORE CLE, CME OR CCE

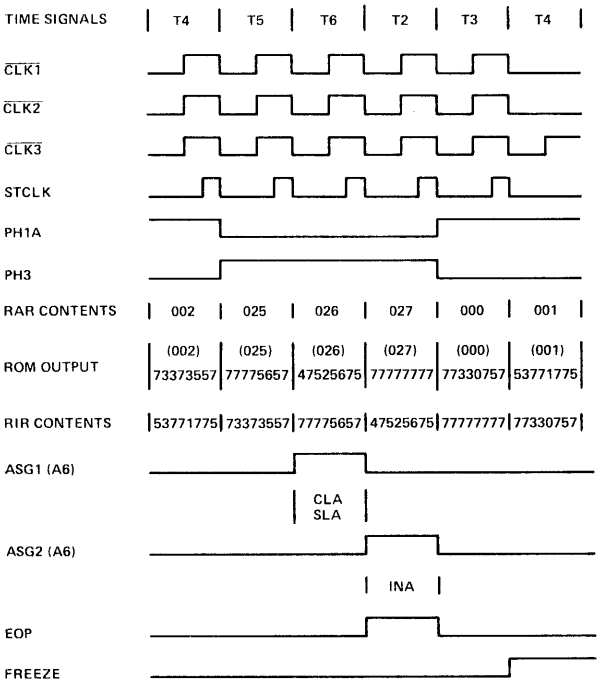
2.) A– OR B–REGISTER

THE FOLLOWING SEQUENCE APPLIES TO THE A– OR B–REGISTER RELATED ASG-INSTRUCTIONS:

1. CL* OR CM* OR CC*
2. SS* OR SL* (RSS)
3. IN*
4. SZ* (RSS)

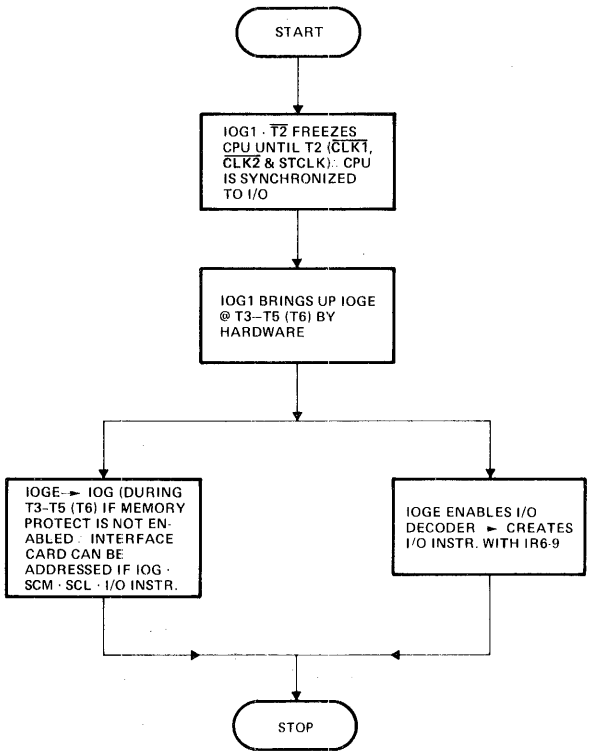
TIMING DIAGRAM OF PH3; CLA, SLA, INA – INSTRUCTION



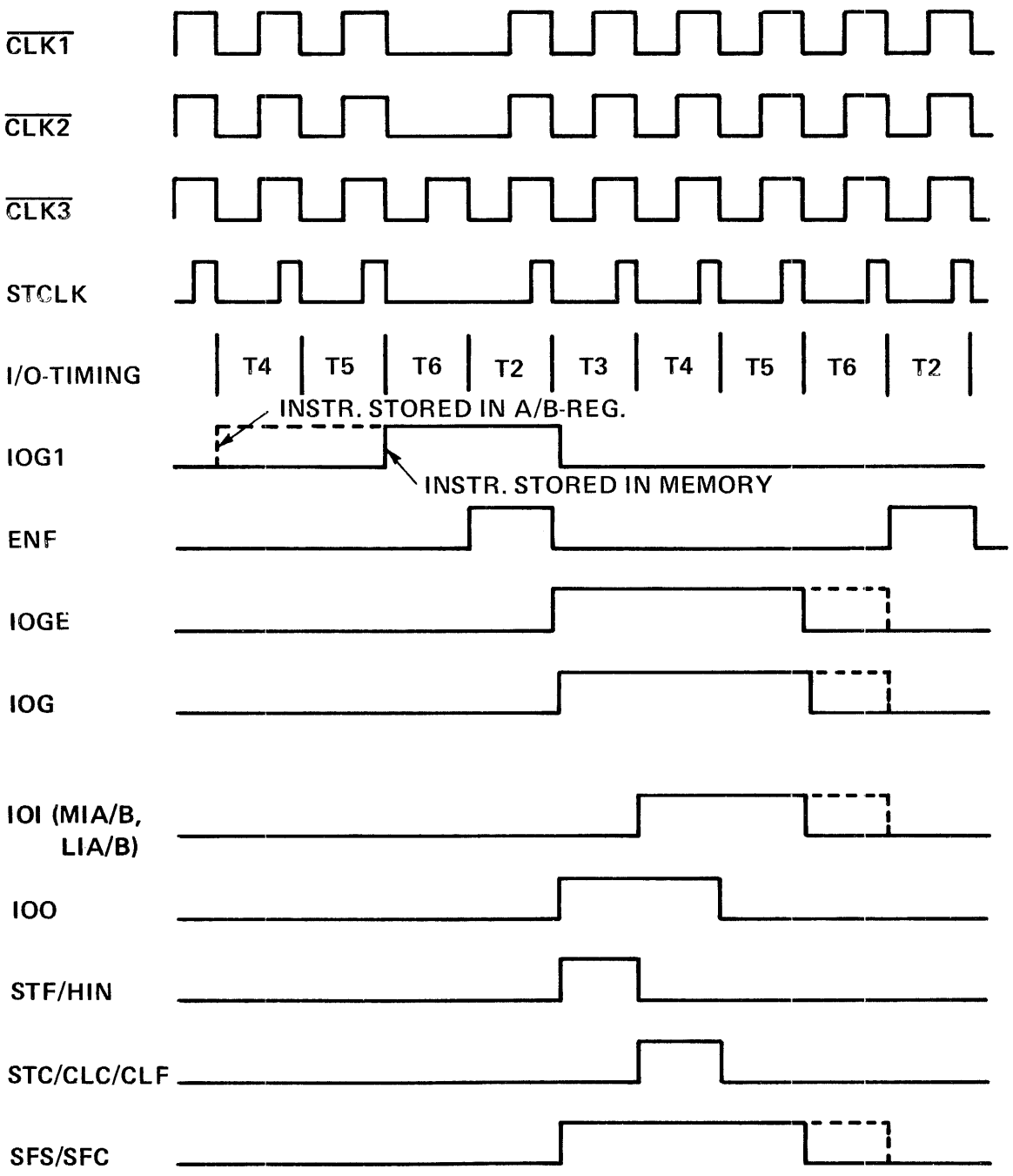


STC, CLC, STF, CLF, SFS, SFC & HLT – INSTRUCTIONS

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0040	77777737	FLAG	–	–	IOR	–	IOG1	–
0041	77777777		–	–	IOR	–	–	–
0042	77777775		–	–	IOR	–	–	EOP
0043	77777777		–	–	IOR	–	–	–
0074	77777737	CTRL	–	–	IOR	–	IOG1	–
0075	77777777		–	–	IOR	–	–	–
0076	77777775		–	–	IOR	–	–	EOP
0077	77777777		–	–	IOR	–	–	–

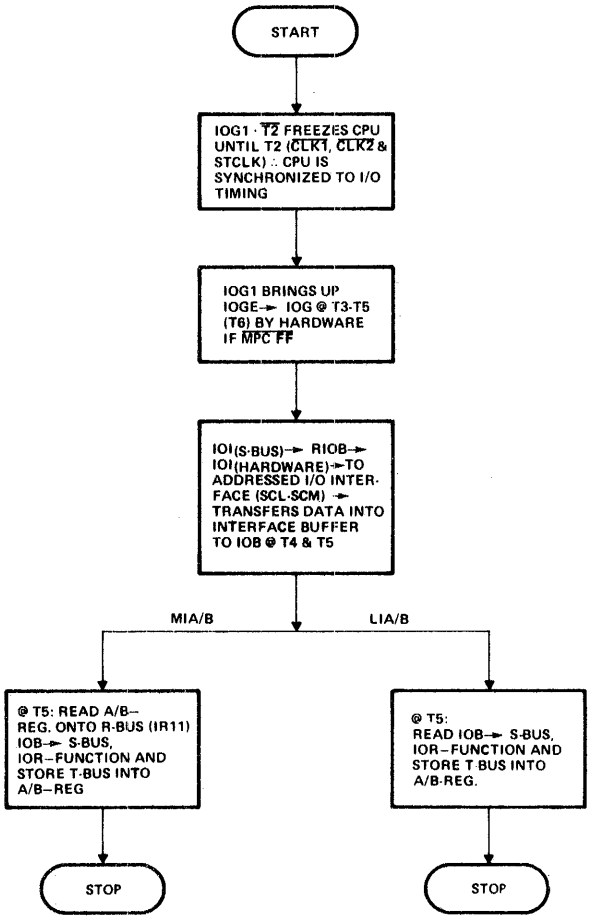


TIMING DIAGRAM FOR I/O – INSTRUCTIONS



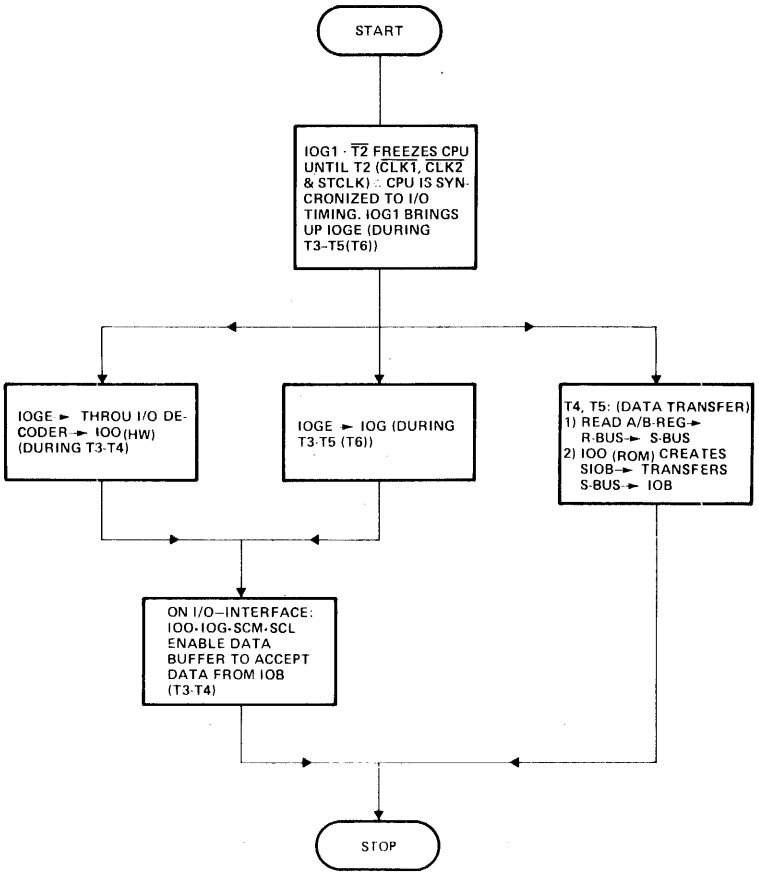
MI*/LI* – INSTRUCTIONS

OM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0060	77777737	MI*	–	–	IOR	–	IOG1	–
0061	77777777		–	–	IOR	–	–	–
0062	70777775		–	IOI	IOR	–	–	EOP
0063	40775777		CAB	IOI	IOR	CAB	–	–
0064	77777737	LI*	–	–	IOR	–	IOG1	–
0065	77777777		–	–	IOR	–	–	–
0066	70777775		–	IOI	IOR	–	–	EOP
0067	70775777		–	IOI	IOR	CAB	–	–



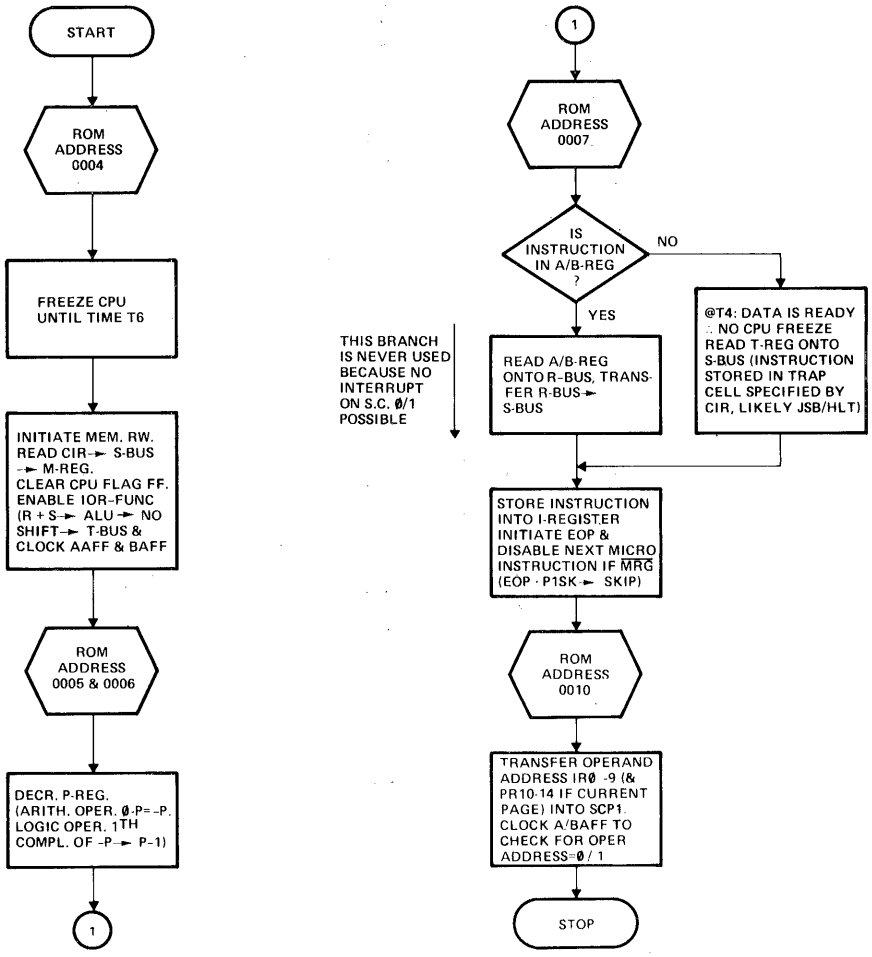
OT* – INSTRUCTION

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0070	77777737	OT*	–	–	IOR	–	IOG1	–
0071	42377777		CAB	RRS	IOR	–	–	–
0072	42370375		CAB	RRS	IOR	IOO	–	EOP
0073	42370377		CAB	RRS	IOR	IOO	–	–

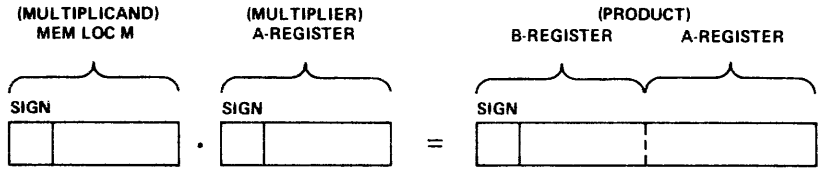


INTERRUPT PHASE

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0004	70330757	PH1B	–	CIR	CFLG	M	RW	–
0005	77054377		–	P	SUB	P	–	–
0006	77154377		–	P	NOR	P	–	–
0007	53771775		AAB	COND	IOR	IR	–	EOP
0010	73373557		–	ADR	IOR	S1	AAB	–



THE BINARY MULTIPLICATION



MULTIPLICAND POS MULTIPLIER POS	MULTIPLICAND POS MULTIPLIER NEG	MULTIPLICAND NEG MULTIPLIER POS	MULTIPLICAND NEG MULTIPLIER NEG
$6_{10} \cdot 3_{10}$	$6_{10} \cdot -3$	$-6_{10} \cdot 3_{10}$	$-6_{10} \cdot -3_{10}$
0110·0011	0110·1101	1010·0011	1010·1101
0110	0110	1010	1010
0110	0000	1010	0000
0000	0110	0000	1010
0000	0110	0000	1010
00010010 → $+18_{10}$	01001110 → $+78_{10}$	00011110 → $+30_{10}$	10000010 → -126_{10}

HOW CAN AN INTEGER MULTIPLICATION BE IMPLEMENTED BY HARD/FIRMWARE?

THERE ARE TWO WELL KNOWN ALGORITHMS:

1. ALGORITHM

1. CHECK MULTIPLIER. IF NEGATIVE: STORE THE SIGN BIT AND CONVERT MULTIPLIER TO POSITIVE EQUIVALENT.

2. CHECK MULTIPLICAND. IF NEGATIVE: STORE THE SIGN BIT AND CONVERT MULTIPLICAND TO POSITIVE EQUIVALENT.

3. EXECUTE BINARY MULTIPLICATION

4. EXCLUSIVE OR THE TWO STORED SIGN BITS. IF RESULT IS "1" THE PRODUCT SHOULD BE NEGATIVE. ∴ CONVERT THE PRODUCT TO THE NEGATIVE EQUIVALENT.

2. ALGORITHM

1. EXECUTE BINARY MULTIPLICATION (REGARDLESS OF SIGN OF MULTIPLICAND AND MULTIPLIER).

2. IF THE MULTIPLIER IS NEGATIVE SUBTRACT THE MULTIPLICAND FROM THE 16 MOST SIGNIFICAND BITS OF THE PRODUCT.

3. IF THE MULTIPLICAND IS NEGATIVE SUBTRACT THE MULTIPLIER FROM THE 16 MOST SIGNIFICAND BITS OF THE PRODUCT.

THE HP 2100 A MICROPROGRAM UTILIZES THE 2. ALGORITHM TO PERFORM A MULTIPLICATION.

WHY DOES THE 2. ALGORITHM EXECUTE A BINARY MULTIPLICATION CORRECTLY?

EXAMPLE: $0110 \cdot 1101$

```
  0110
 0110
0110
01001110
```

EXPECTED COMPUTATION: $6_{10} \cdot (-3_{10}) = -18_{10}$

$X \cdot M_e = P_e$

PERFORMED COMPUTATION: $6_{10} \cdot 13_{10} = 78_{10}$

$X \cdot M_I = P_I$

THE MULTIPLICAND IS TOO BIG BY A VALUE OF $16_{10} = 2^n$

(n = NR. OF BITS OF MULTIPLICAND)

$$M_I = M_e + 2^n$$

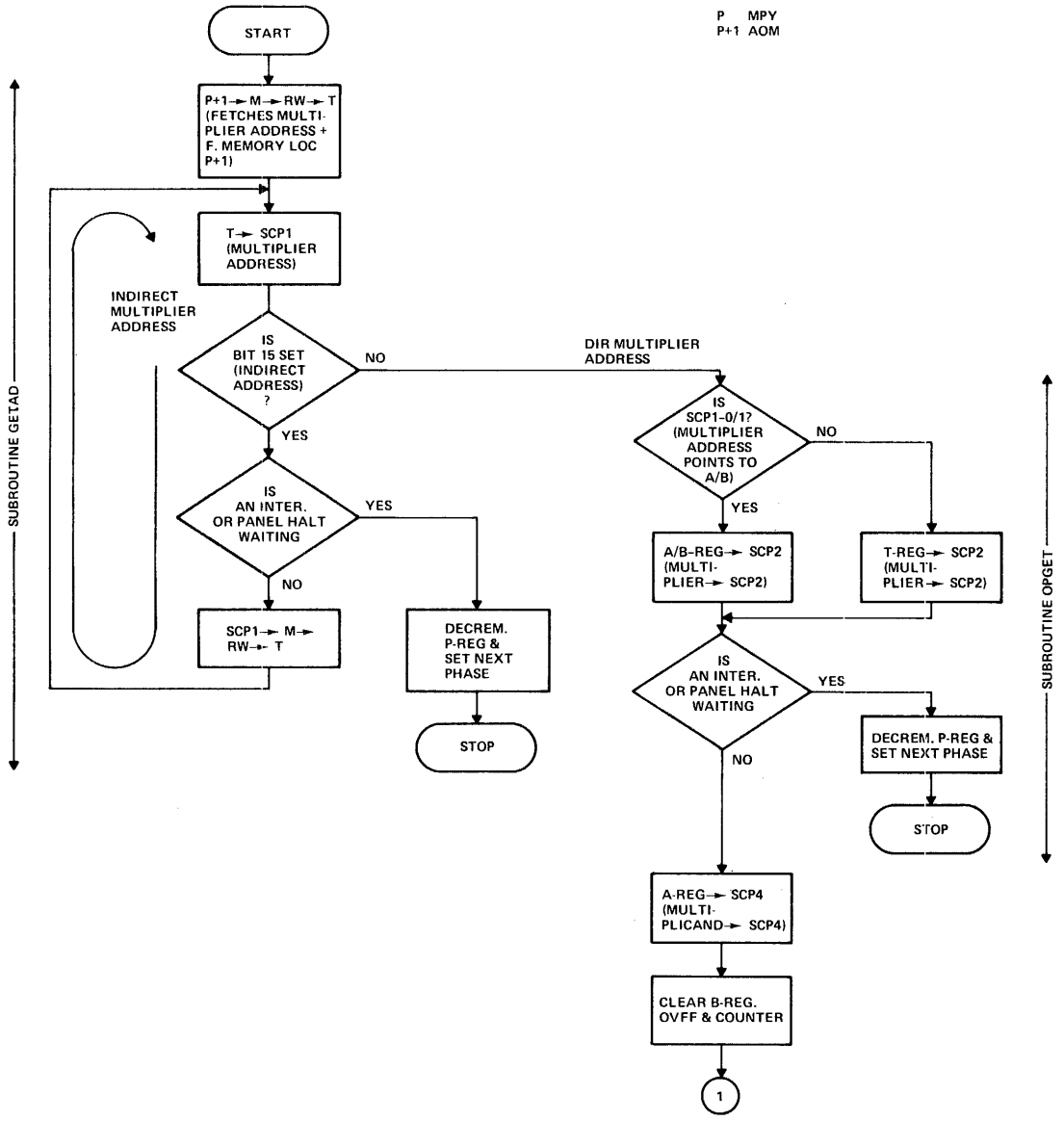
$$\therefore P_I = X \cdot M_I = X \cdot (M_e + 2^n) = P_e + X \cdot 2^n$$

$$\underline{P_e = P_I - X \cdot 2^n}$$

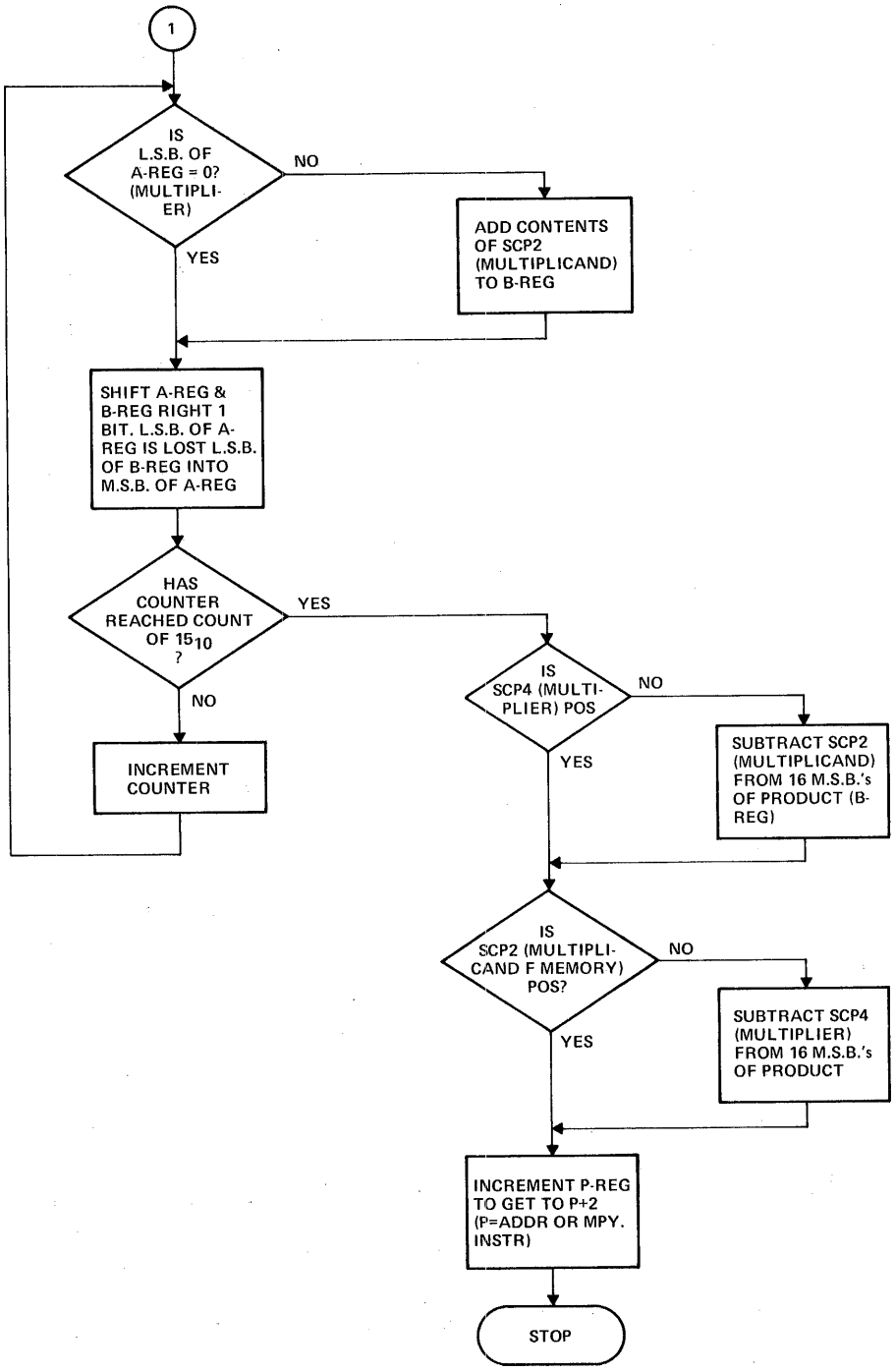
01001110	←	P_I
- 01100000	←	$-(X \cdot 2^n)$
11101110	→	$P_e = -18_{10}$

X = MULTIPLICAND
M_e = EXPECTED MULTIPLIER
M_I = INTERPERED MULTIPLIER
P_e = EXPECTED PRODUCT
P_I = INTERPERED PRODUCT

MPY – INSTRUCTION



MPY – INSTRUCTION (CONTINUE)



MPY – MICROPROGRAM

ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS						COMMENTS
			R-BUS 23-21	S-BUS 20-17	FUNCTION 16-12	STORE 11-8	SPECIAL 7-4	SKIP 3-0	
0210	77777577	MULT	–	–	IOR	–	LEP	–	Legal entry point for MPY. Execute next line.
0211	77207762		–	P	JSB	–	GETAD	–	Execute GETAD subroutine. Puts multiplier address in S1.
0212	77207632		–	P	JSB	–	OPGET	–	Execute OPGET subroutine. Puts multiplier in S2.
0213	07222155		A	P	JMP	S4	MPY	–	Save multiplicand in S4. Jump to 0155.
0214	77777777		–	–	IOR	–	–	–	Unused.
0215	77777777		–	–	IOR	–	–	–	Unused.
0216	77777775		–	–	IOR	–	–	EOP	Unused.
0217	77777777		–	–	IOR	–	–	–	Unused.
0362	77370757	GETAD	–	P	IOR	M	RW	–	Send P-register address to memory, start read cycle.
0363	53773403	ONEMO	AAB	COND	IOR	S1	RSS	NEG	Normally fetches contents of a software DEF instruction. Put fetched word into S1. Skip next line if not indirect (i.e., if bit 15 = 0).
0364	77227766		–	P	JMP	–	IND	–	Jump to 0366.
0365	75657557		–	S1	RSB	–	AAB	–	Set AAF/BAF if S1 contents = 0 or 1. Return to calling subroutine.
0366	77247411	IND	–	–	CJMP	–	PDEC	–	Jump to 0011 if interrupt or panel halt. Else continue.
0367	75770757		–	S1	IOR	M	RW	–	Send operand address to memory and start read cycle.
0370	77227763		–	–	JMP	–	ONEMO	–	Jump to 0363.
0232	75770757	OPGET	–	S1	IOR	M	RW	–	Send operand address to memory and start read cycle.
0233	53773377		AAB	COND	IOR	S2	–	–	Put operand data into S2.
0234	77247411		–	P	CJMP	–	PDEC	–	Jump to 0011 if interrupt or panel halt. Else continue.
0235	77657777		–	–	RSB	–	–	–	Return to calling routine.
									Continued from 0213:
0155	77756461	MPY	–	–	CLO	B	CNTR	RPT	Clear counter, B-register, and Overflow. Set repeat mode.
0156	15036450		B	S2	MPY	B	R1	CTRI	Execute MPY on B and S2 16 times. Result in B, A-registers.
0157	14377403		–	S4	IOR	–	RSS	NEG	Skip next line if multiplicand (was A-register) is positive.
0160	15056777		B	S2	SUB	B	–	–	Subtract multiplier from high order word of result.
0161	75377403		–	S2	IOR	–	RSS	NEG	Skip next line if multiplier (from memory) is positive.
0162	14056775		–	S4	SUB	B	–	EOP	Subtract multiplicand from high word. Set next phase.
0163	77114377		–	P	INC	P	–	–	Increment P-register past the DEF software instruction.
0011	77054375	PDEC	–	P	SUB	P	–	EOP	Decrement the P-register and set next phase: P→ -P
0012	77154377		–	P	NOR	P	–	–	One's comp of -P = P-1

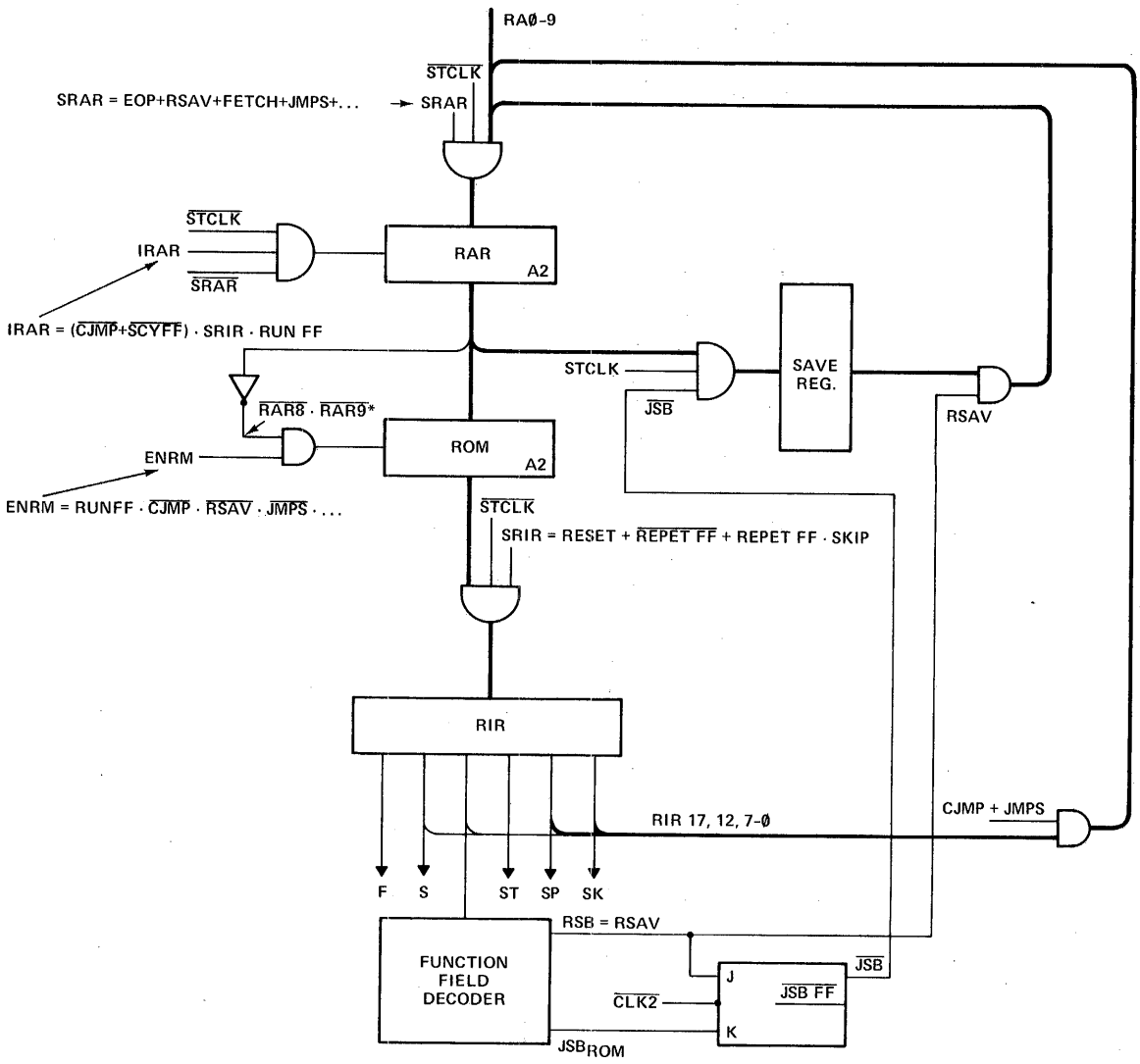
SPECIAL MICROORDERS REQUIRED FOR THE EXECUTION OF EAG-MEMORY REF. INSTRUCTIONS

ROM ADDR	MICROORDER	BIT CONF	COMMENT
0210	LEP	0111	MACRO-CODE (IR15 · $\overline{IR14}$ · $\overline{IR13}$ · $\overline{IR12}$ · $\overline{IR10}$ · $\overline{IR11}$ · $\overline{IR9}$) @ SPH 3 CLEARS LEP FF ∴ ALL MICROINSTR. WILL BE SKIPPED. LEP SETS LEP FF ∴ INHIBITS SKIP. (ON A2)
0211	JSB GETAD	11110010	JSB TO RA 362 IN ROM MODULE Ø. (SEE FIG. 3-33 & 3-34) SETS $\overline{JSB\ FF}$ (A4)→ ∴ SAVE REGISTER (A2) WILL NO MORE BE CLOCKED→ ∴ PRESENT ADDRESS IN SAVE REGISTER IS LOCKED IN UNTIL $\overline{JSB\ FF}$ CLEARS.
0363	RSS NEG	00000011	SKIP FIELD DECODER (A6) U24A & U92B→ SKIP = RSS · NEG · $\overline{ALU15}$→ SKIP NEXT MICROINSTR. IF NOT INDIRECT ($\overline{ALU15}$).
0366	CJMP PDEC	00001001	COND JMP TO RA11. (SEE FIG. 3-33 & 3-34) IF RUN MODE AND FRONT PANEL HLT SWITCH PRESSED (A24 S29→ RELEASES THE RESET SWBN FF→ THROUGH DELAY CIRCUIT Q1 & Q2 SETS SWBN FF→ SETS HLT SW FF THROUGH U41B & U43B→ HIN TO A1 RESETS RH FF, SIN FF & SCY FF THROUGH U75B, U86F TO U96D WHERE ANDED WITH CJMP→ RJMP, SRAR, $\overline{ENRM}$, $\overline{IRAR}$ ∴ EXECUTES RJMP. TO ROM ADDR. 11 IF RUN MODE AND INTERRUPT (A1, PIN 22) TO U96B WHERE ANDED WITH CJMP→ RJMP, SRAR, $\overline{ENRM}$, $\overline{IRAR}$ ∴ EXECUTES RJMP TO ROM ADDR. 11 IF RUN MODE AND ANY OTHER MICROORDER→ $\overline{RJMP}$, $\overline{SRAR}$, $\overline{ENRM}$ AND $\overline{IRAR}$ ∴ EXECUTES NOP. IF SINGLE CYCLE MODE: $\overline{SCY}$ FF (A24) CLEARS→ SSCY TO A1 THROUGH U105A SETS SCY FF UNTIL CJMP THROUGH U65B RESETS SCY FF @ NEXT CLK1→ $\overline{IRAR}$, $\overline{ENRM}$, $\overline{RJMP}$ AND $\overline{SRAR}$ (ROM JMP IS NOT EXECUTED). U105A ALSO SETS RUN FF DURING SCY FF IS SET. DHLT FF RESETS DURING RUN FF SET + 200 NS. (HALTS MACHINE)

SPECIAL MICROORDERS REQUIRED FOR THE EXECUTION OF EAG-MEMORY REF. INSTRUCTIONS (CONTINUE)

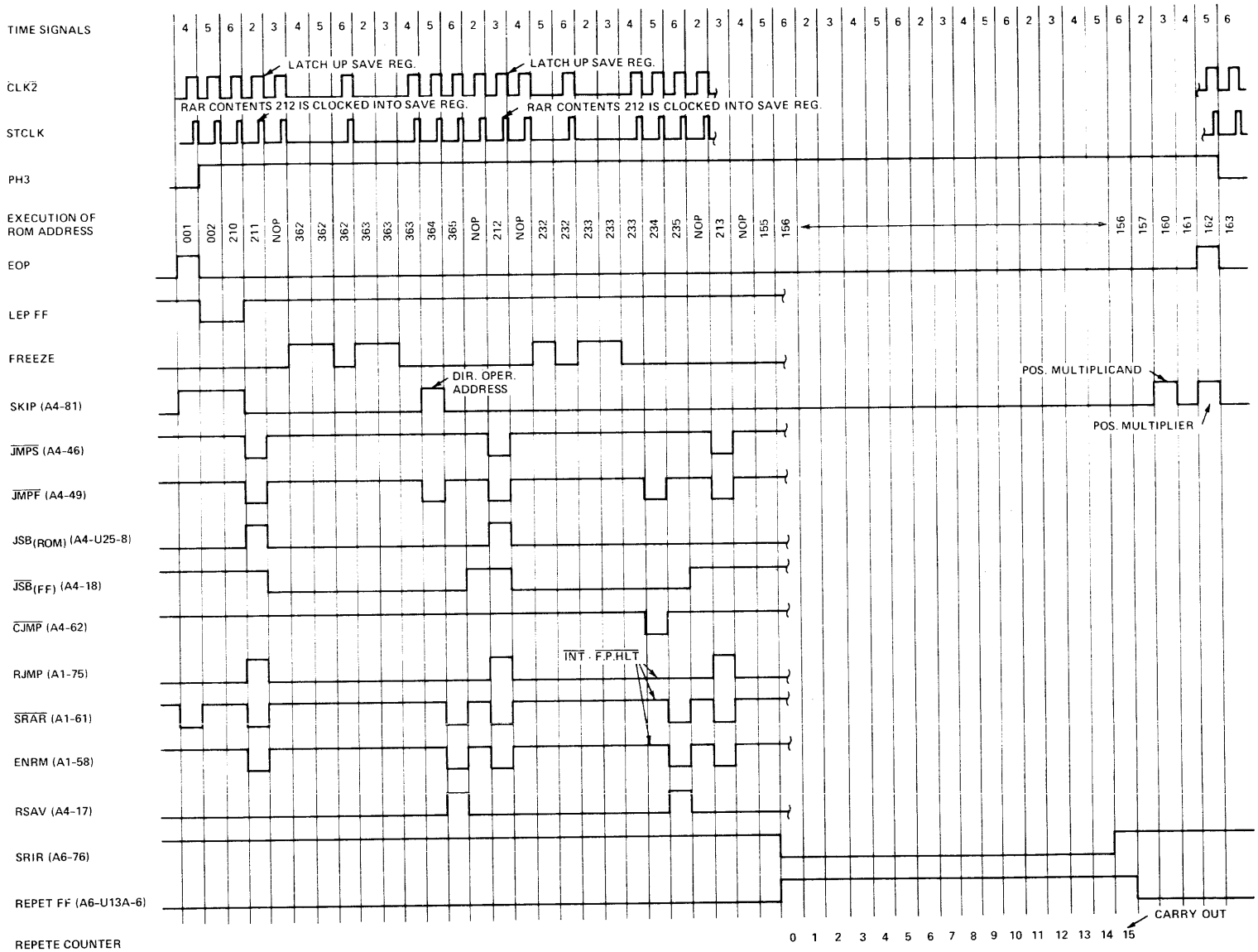
ROM ADDR	MICROORDER	BIT CONF	COMMENT
365	RSB	10101	RSB BRINGS UP RSAV→ ENABLES READ OUT FROM SAVE REG TO RAO-9 AND STORES RETURN ADDRESS INTO RAR. RSB ALSO SETS $\overline{\text{JSB FF}}$→ ENABLING THE REGISTER TO FOLLOW THE RAR OUTPUT AGAIN.
155	CNTR, RPT	00110001	ENABLES SBØ–3 TO BE READ INTO THE REPEAT COUNTER. PRESENTLY Ø'S ON S-BUS ∴ CLEARS REPEAT COUNTER.
			RPT CLEARS $\overline{\text{RPT FF}}$ ∴ SRIR GOES LOW AND INHIBITS CLOCKING OF $\overline{\text{SKIP FF}}$.

ROM JSB

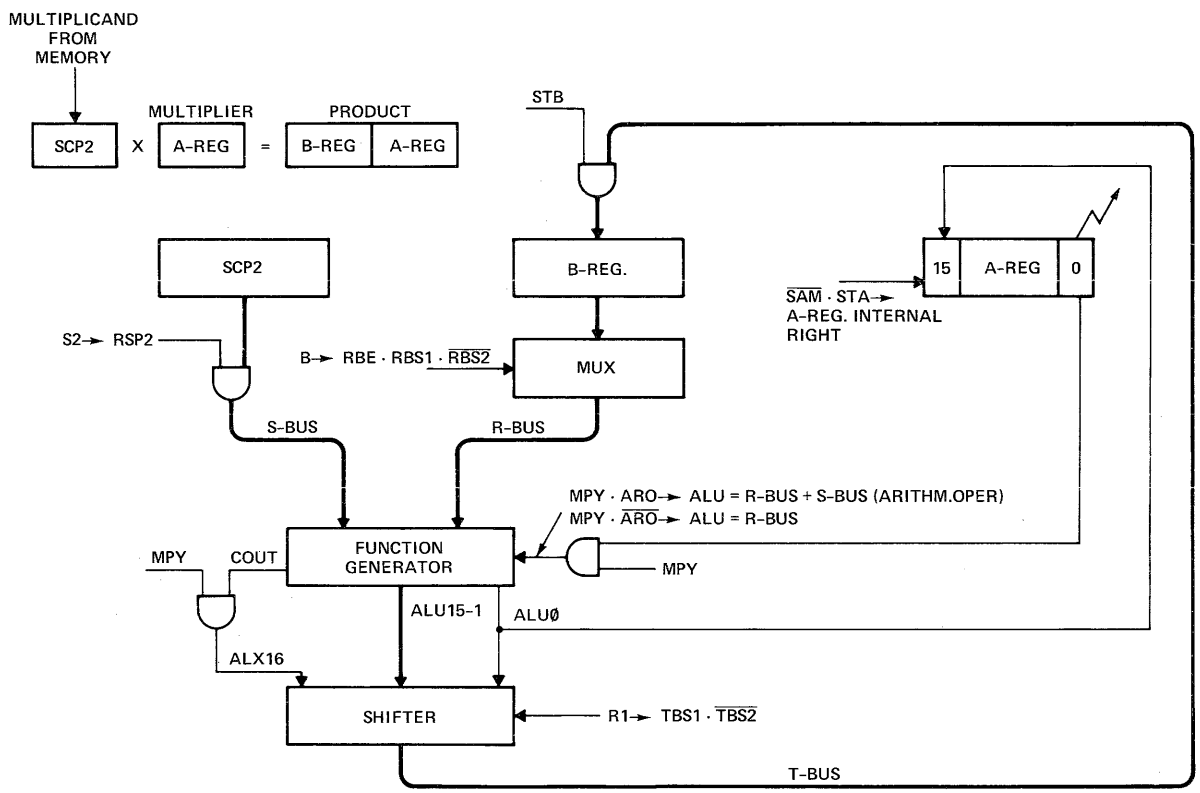


*CONSIDER ROM MODUL Ø ONLY

TIMING DIAGRAM OF PH3, DIR. MPY – INSTR. (OP. ADDR NOT IN A/B-REG.)



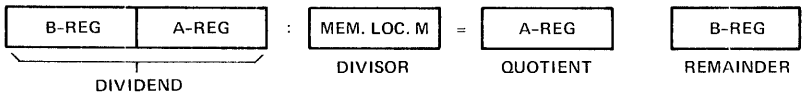
EXECUTION OF MULTIPLICATION

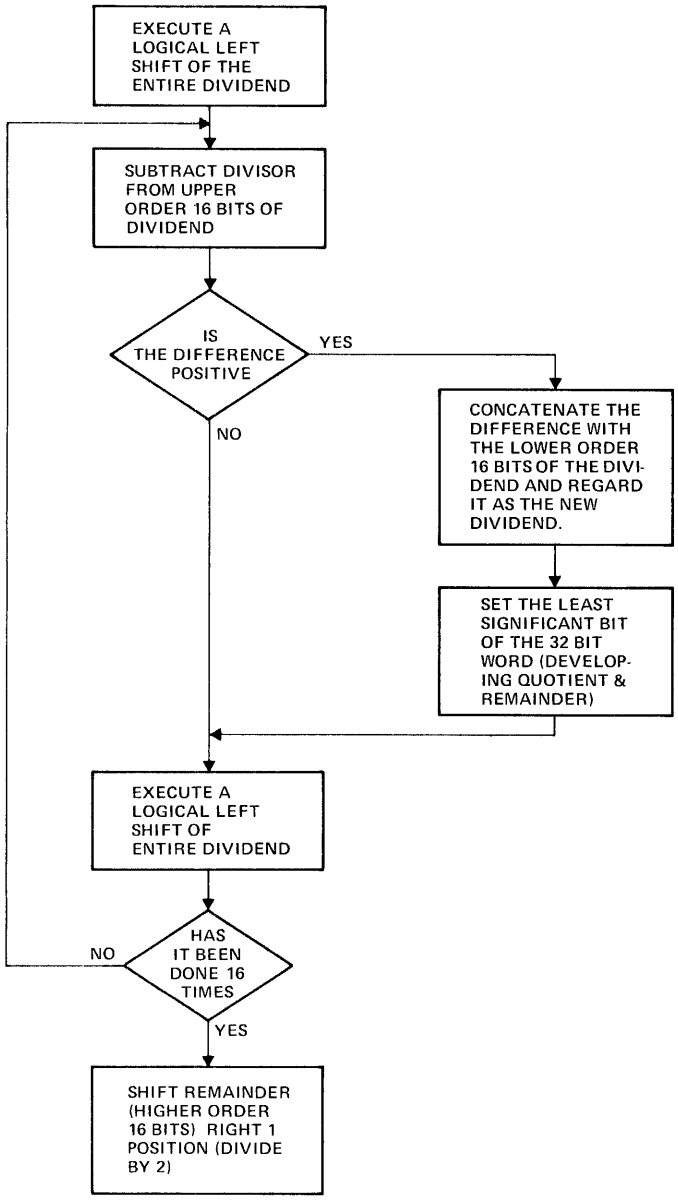


ROM ADDRESS	ROM WORD (OCTAL)	ENTRY POINT LABEL	FIELD CONTENTS					
			R-BUS	S-BUS	FUNCTION	STORE	SPECIAL	SKIP
0156	15036450		B	S2	MPY	B	R1	CTRI

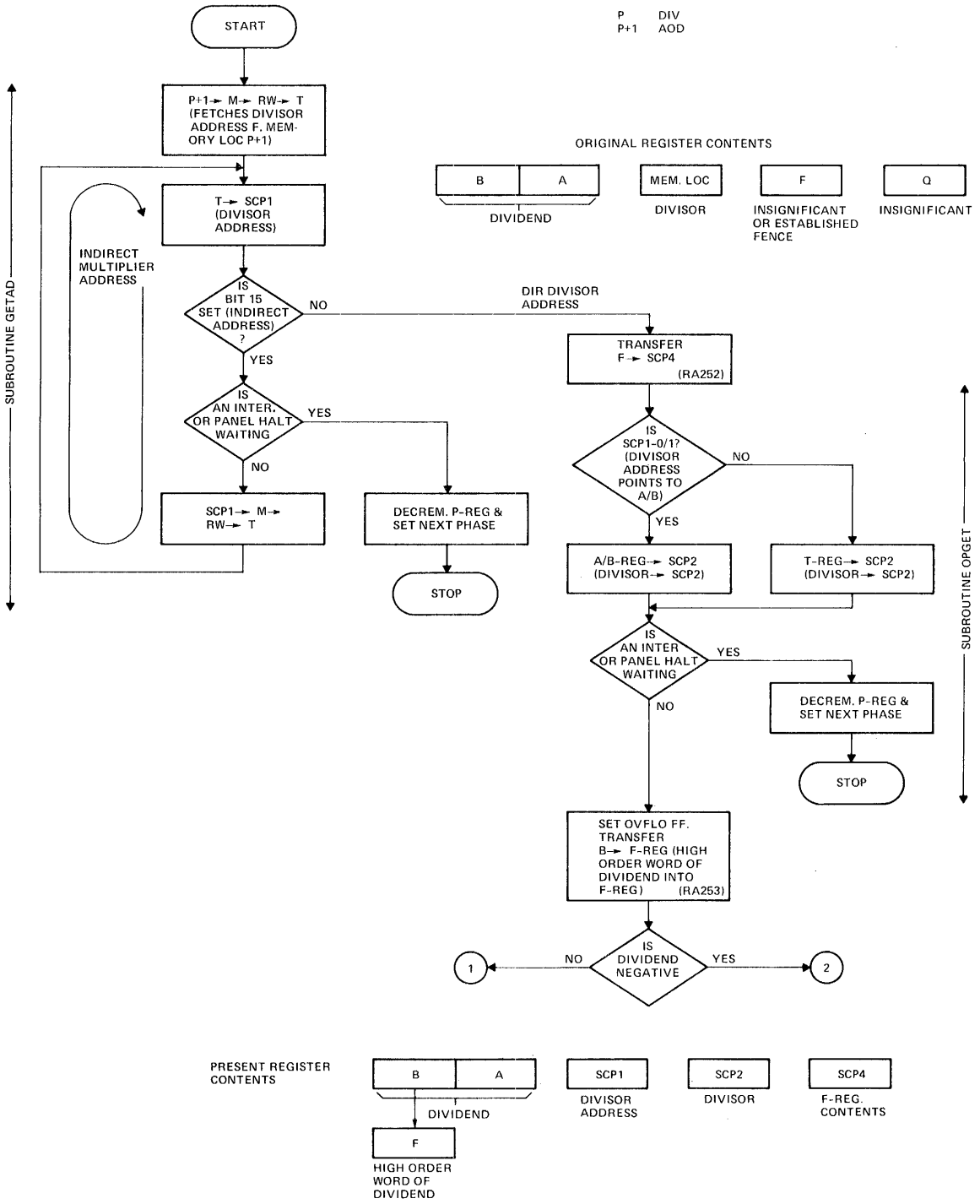
THE BINARY DIVISION

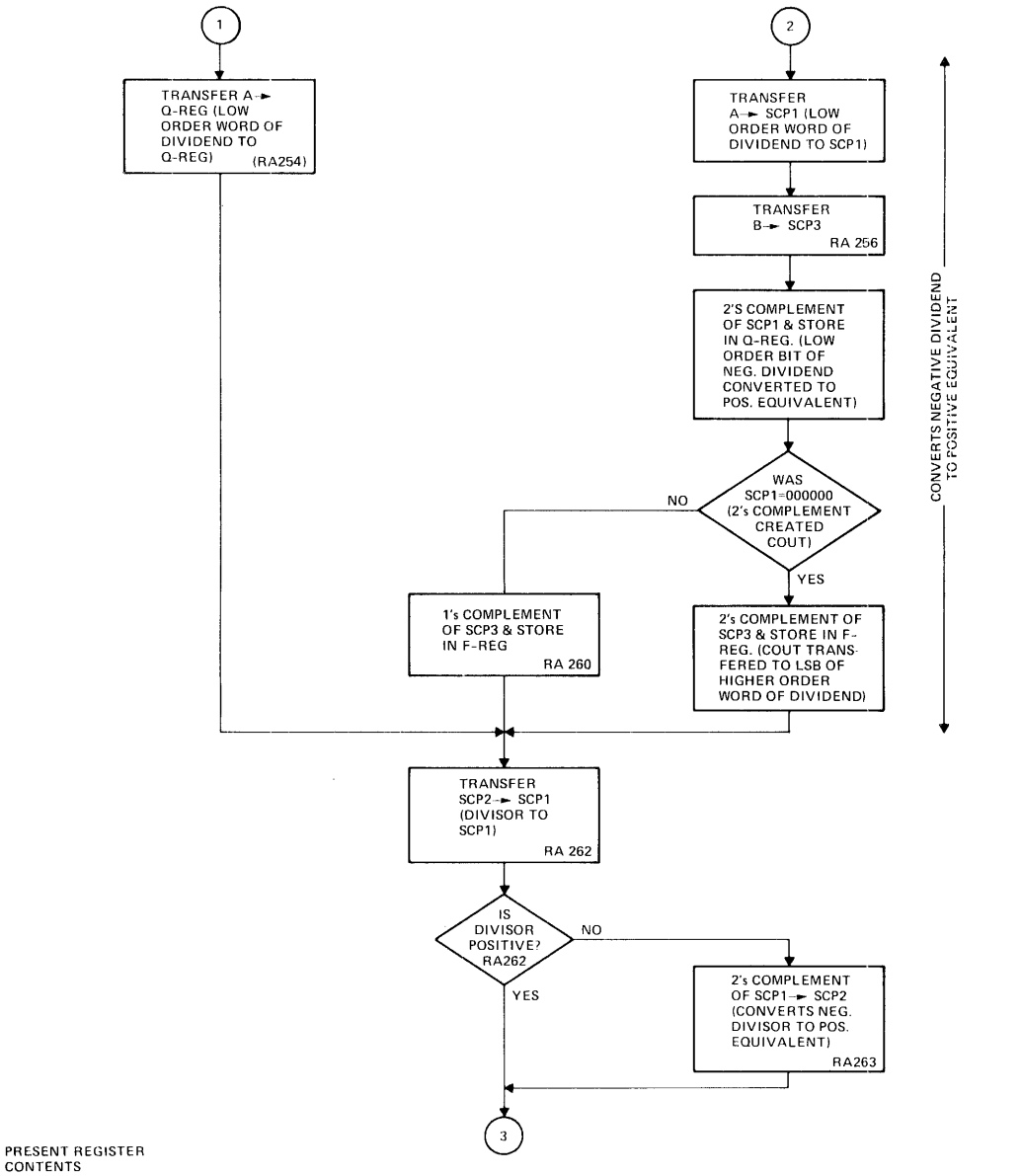
- TO EXECUTE A BINARY DIVISION CORRECTLY DIVIDEND AS WELL AS DIVISOR HAVE TO BE POSITIVE FOR THE SAME REASON AS ILLUSTRATED IN THE MULTIPLY INSTRUCTION FIGURE

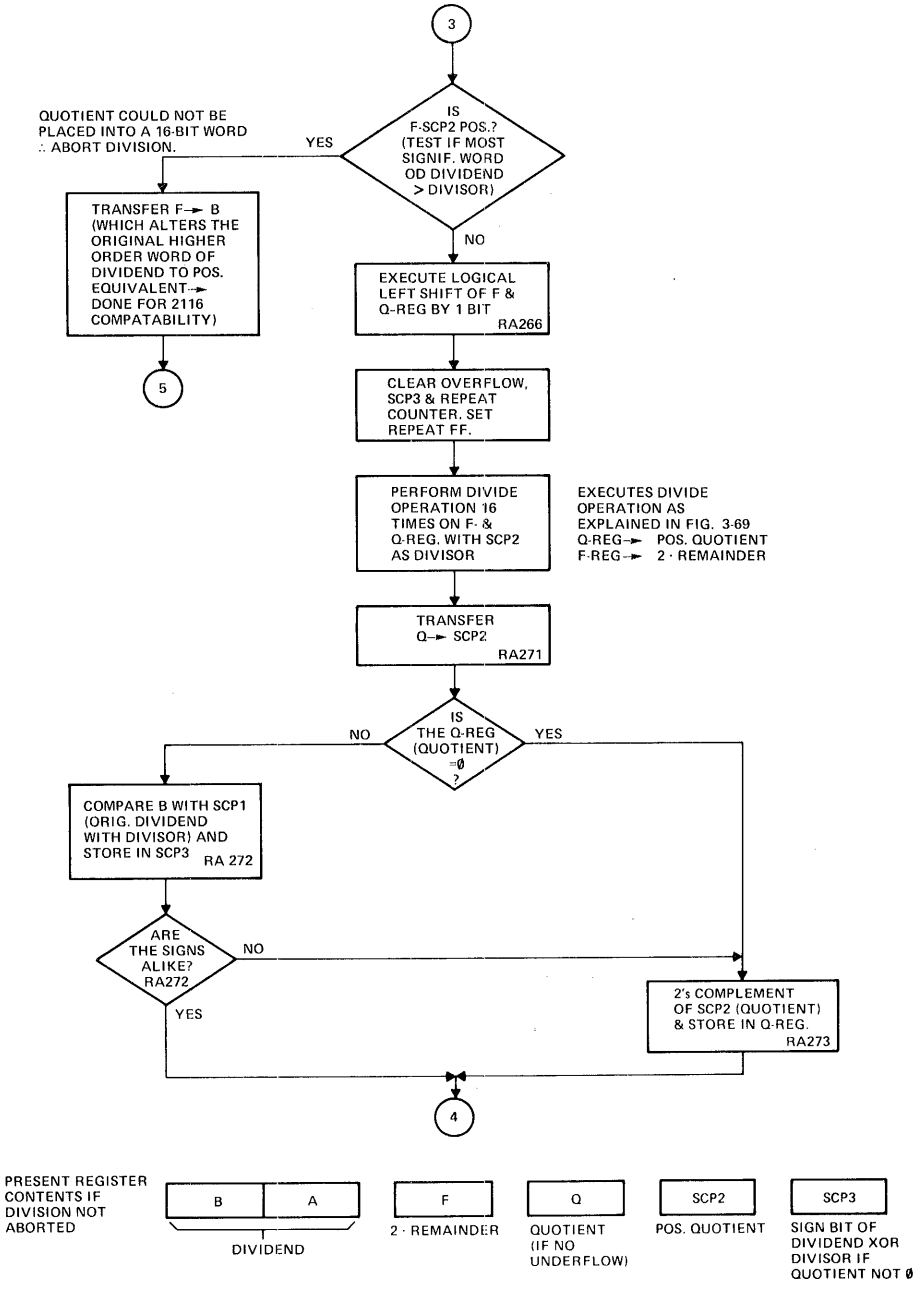


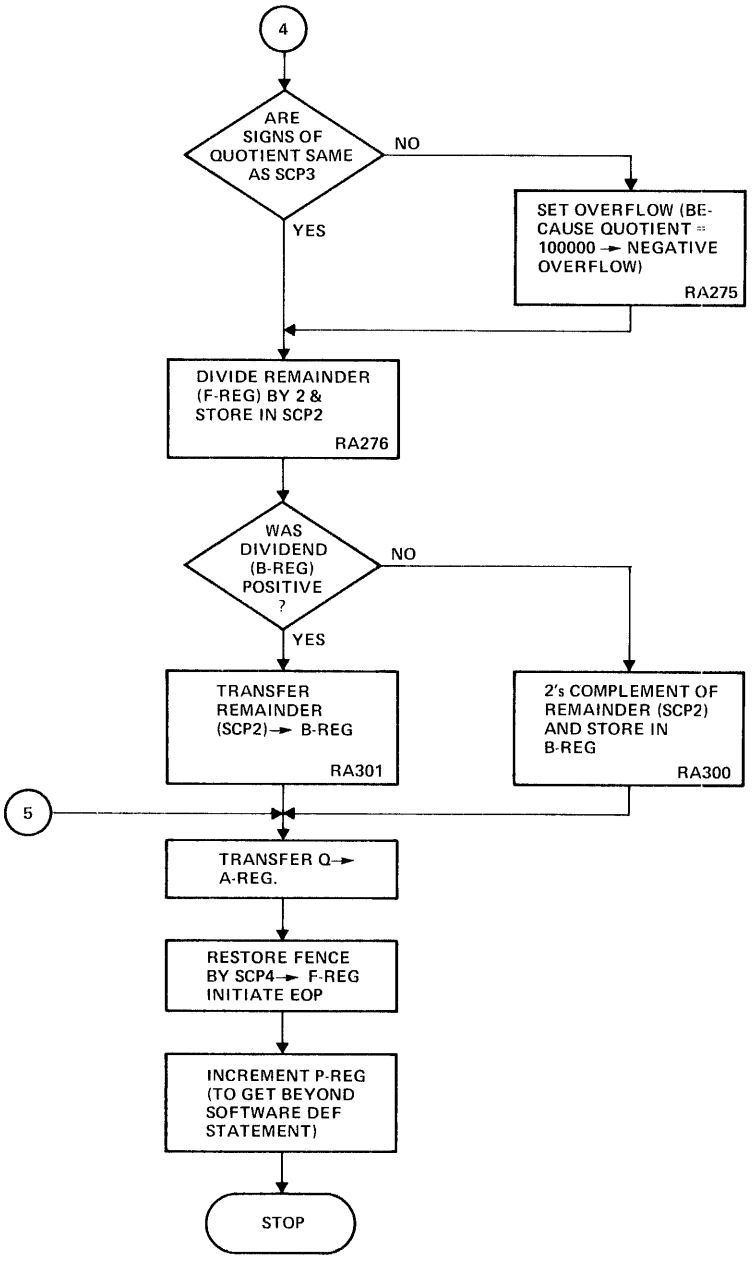


DIV – INSTRUCTION

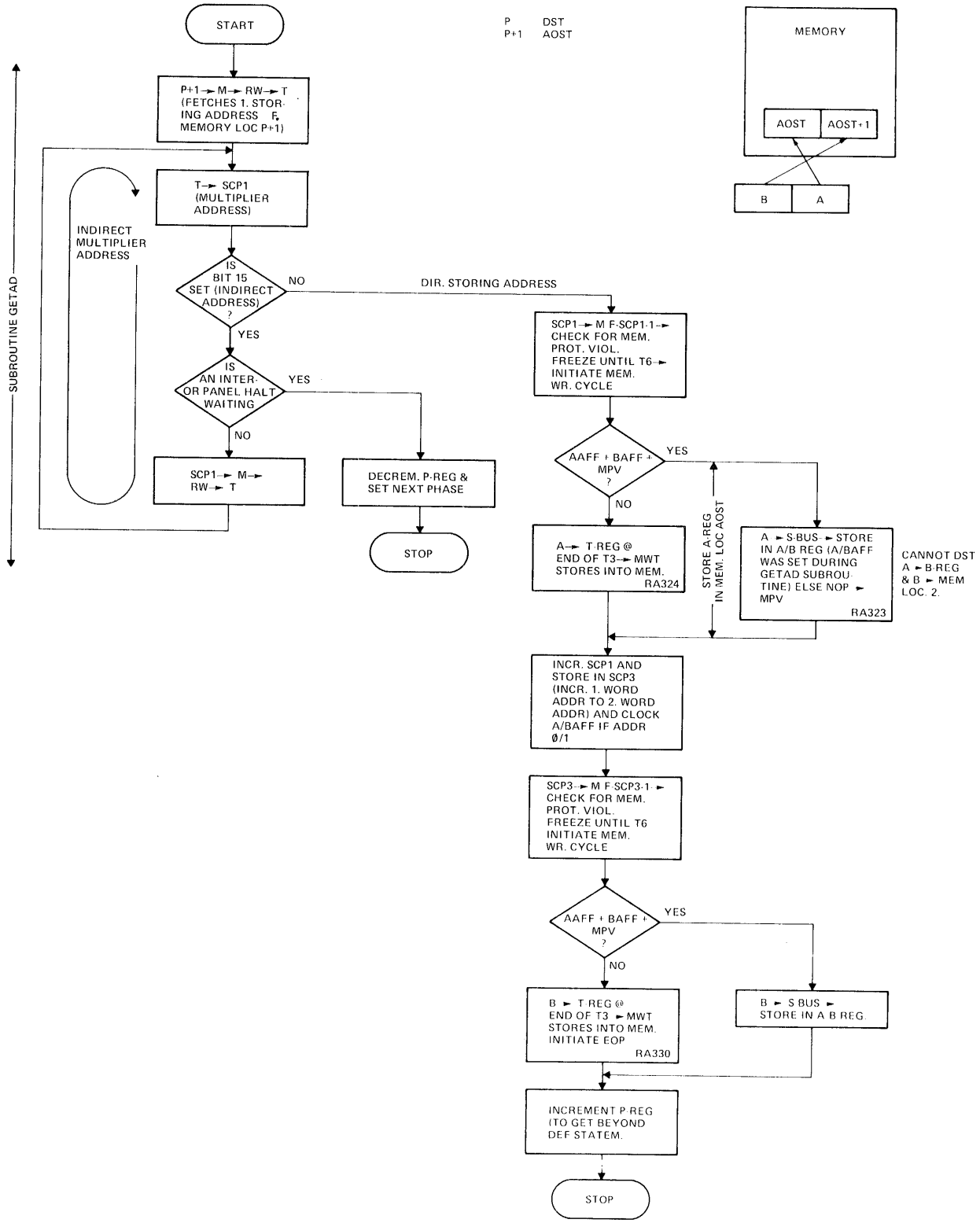



1
TRANSFER A→ Q-REG (LOW ORDER WORD OF DIVIDEND TO Q-REG) (RA254)
2
TRANSFER A→ SCP1 (LOW ORDER WORD OF DIVIDEND TO SCP1)
TRANSFER B→ SCP3 RA 256
2'S COMPLEMENT OF SCP1 & STORE IN Q-REG. (LOW ORDER BIT OF NEG. DIVIDEND CONVERTED TO POS. EQUIVALENT)
WAS SCP1=000000 (2's COMPLEMENT CREATED COUT)
NO
YES
1's COMPLEMENT OF SCP3 & STORE IN F-REG RA 260
2's COMPLEMENT OF SCP3 & STORE IN F-REG. (COUT TRANSFERED TO LSB OF HIGHER ORDER WORD OF DIVIDEND)
CONVERTS NEGATIVE DIVIDEND TO POSITIVE EQUIVALENT
TRANSFER SCP2→ SCP1 (DIVISOR TO SCP1) RA 262
IS DIVISOR POSITIVE? RA262
NO
YES
2's COMPLEMENT OF SCP1→ SCP2 (CONVERTS NEG. DIVISOR TO POS. EQUIVALENT) RA263
3
PRESENT REGISTER CONTENTS
B
A
DIVIDEND
F
Q
POSITIVE EQUIVALENT OF DIVIDEND
SCP1
DIVISOR
SCP2
POS. EQUIV. OF DIVISOR
SCP3
B-REG CONTENTS
SCP4
F-REG CONTENTS

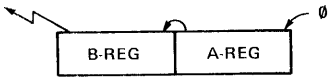

3
IS F-SCP2 POS.? (TEST IF MOST SIGNIF. WORD OD DIVIDEND > DIVISOR)
YES
QUOTIENT COULD NOT BE PLACED INTO A 16-BIT WORD ∴ ABORT DIVISION.
TRANSFER F→ B (WHICH ALTERS THE ORIGINAL HIGHER ORDER WORD OF DIVIDEND TO POS. EQUIVALENT→ DONE FOR 2116 COMPATABILITY)
5
NO
EXECUTE LOGICAL LEFT SHIFT OF F & Q-REG BY 1 BIT
RA266
CLEAR OVERFLOW, SCP3 & REPEAT COUNTER. SET REPEAT FF.
PERFORM DIVIDE OPERATION 16 TIMES ON F- & Q-REG. WITH SCP2 AS DIVISOR
EXECUTES DIVIDE OPERATION AS EXPLAINED IN FIG. 3-69
Q-REG→ POS. QUOTIENT
F-REG→ 2 · REMAINDER
TRANSFER Q→ SCP2
RA271
IS THE Q-REG (QUOTIENT) =Ø ?
NO
YES
COMPARE B WITH SCP1 (ORIG. DIVIDEND WITH DIVISOR) AND STORE IN SCP3
RA 272
ARE THE SIGNS ALIKE? RA272
NO
YES
2's COMPLEMENT OF SCP2 (QUOTIENT) & STORE IN Q-REG.
RA273
4
PRESENT REGISTER CONTENTS IF DIVISION NOT ABORTED
B
A
DIVIDEND
F
2 · REMAINDER
Q
QUOTIENT (IF NO UNDERFLOW)
SCP2
POS. QUOTIENT
SCP3
SIGN BIT OF DIVIDEND XOR DIVISOR IF QUOTIENT NOT Ø

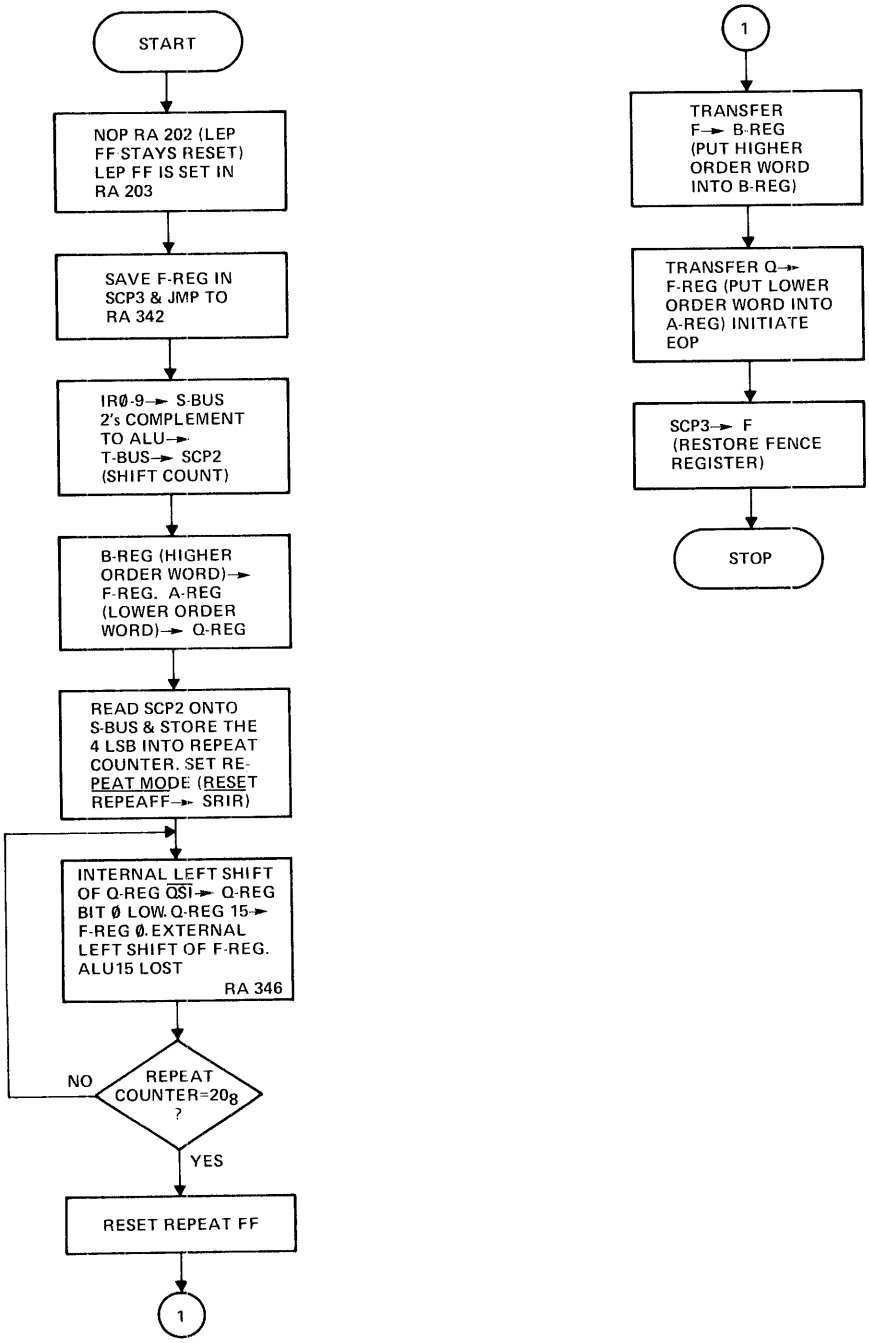

4
ARE SIGNS OF QUOTIENT SAME AS SCP3
NO
YES
SET OVERFLOW (BECAUSE QUOTIENT = 100000 → NEGATIVE OVERFLOW)
RA275
DIVIDE REMAINDER (F-REG) BY 2 & STORE IN SCP2
RA276
WAS DIVIDEND (B-REG) POSITIVE ?
NO
YES
TRANSFER REMAINDER (SCP2)→ B-REG
RA301
2's COMPLEMENT OF REMAINDER (SCP2) AND STORE IN B-REG
RA300
5
TRANSFER Q→ A-REG.
RESTORE FENCE BY SCP4→ F-REG INITIATE EOP
INCREMENT P-REG (TO GET BEYOND SOFTWARE DEF STATEMENT)
STOP

DST – INSTRUCTION



LSL – INSTRUCTION



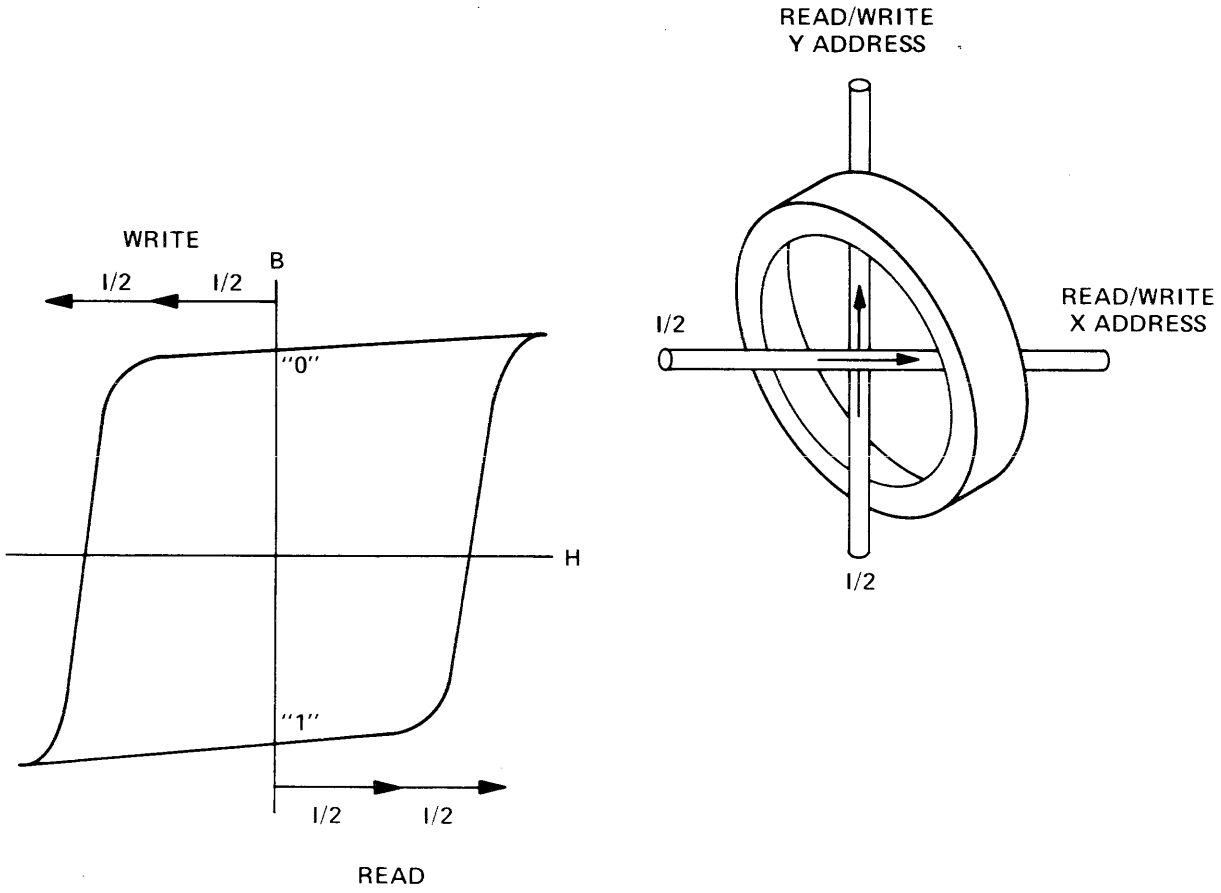


memory 4

LESSON 4

MEMORY

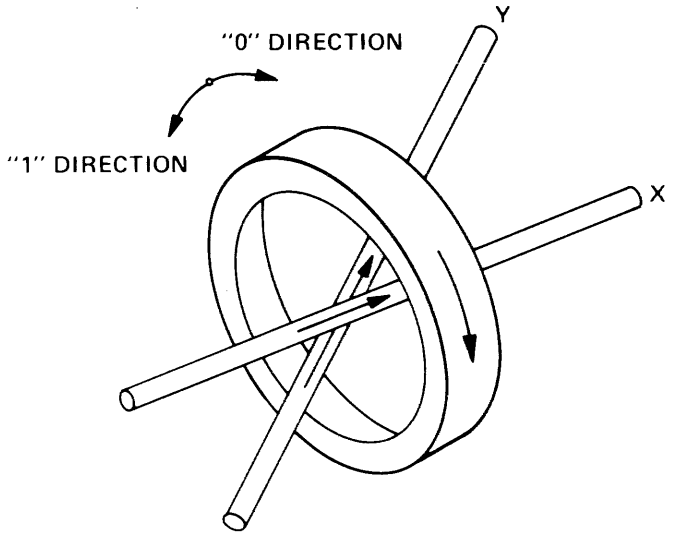
THE MAGNETIC CORE



THE CORE IS ALWAYS SATURATED WHEN IN A STATIC STATE (NRZI)
A CHANGE OF STATE INDICATES READING OR WRITING A "1"

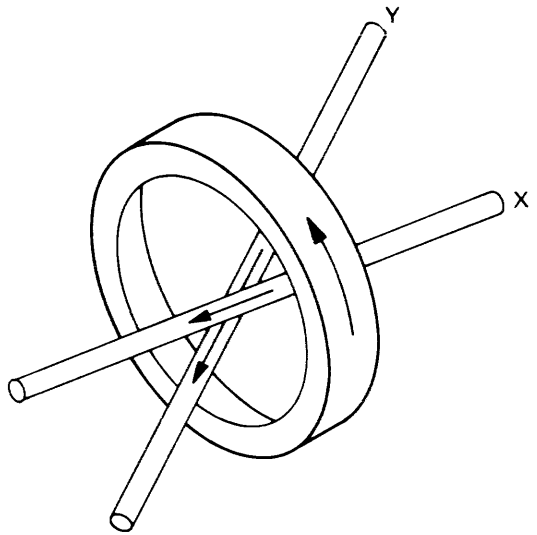
READ/WRITE CURRENTS VS CORE STATUS

READ CYCLE



CORE ORIGINALLY
MAGNETIZED IN "1" DIRECTION,
READ CURRENT SWITCHES CORE
(RESETS CORE TO ZERO DIRECTION)

WRITE CYCLE

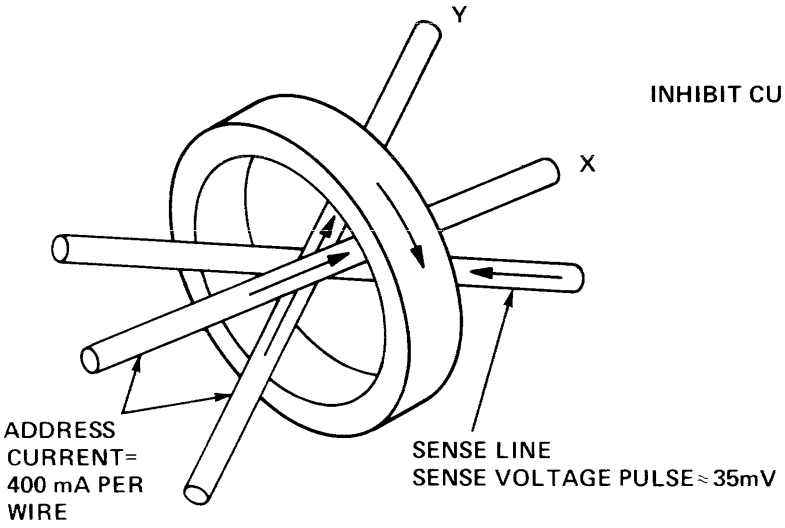


MAGNETIZED IN "0" DIRECTION,
WRITE CURRENT SWITCHES CORE
TO "1" DIRECTION

CURRENT DURING WRITE CYCLE
IS IN OPPOSITE DIRECTION THAN
DURING READ

DETERMINING CORE STATUS

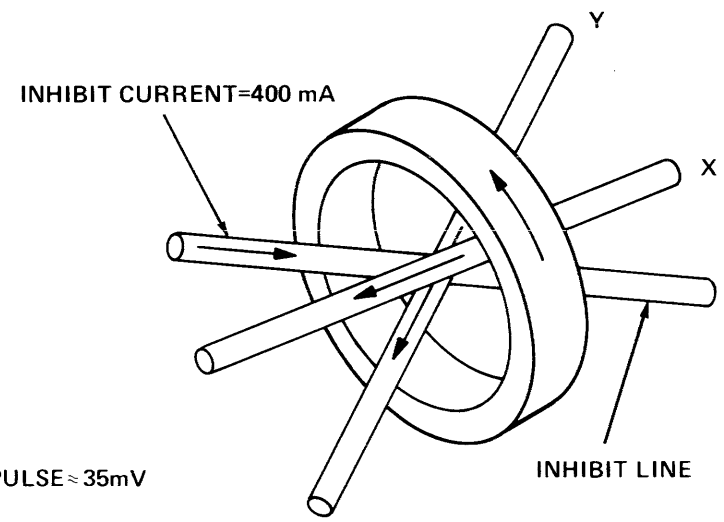
READ/SENSE



SENSE "1"

READ CURRENT SWITCHES CORE
IF MAGNETIZED IN CLOCK DIRECTION→
NO CHANGE→ NO SENSE PULSE

WRITE/INHIBIT

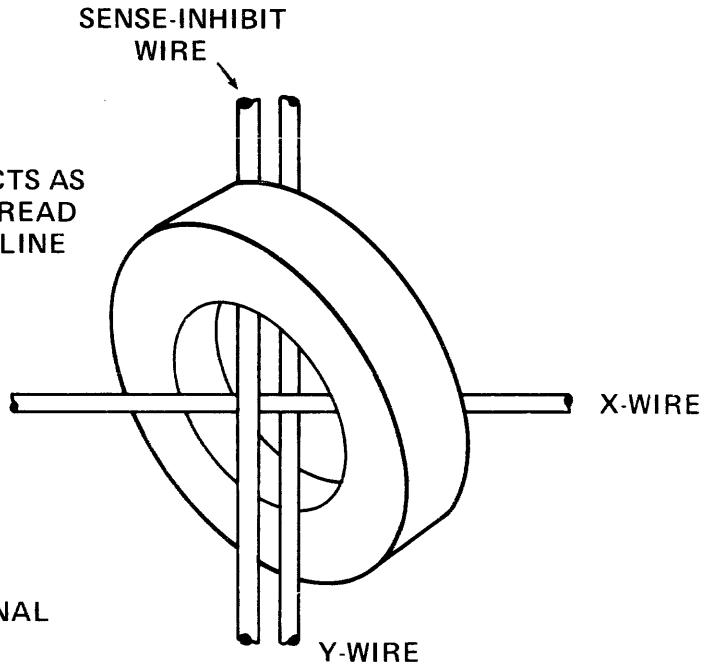


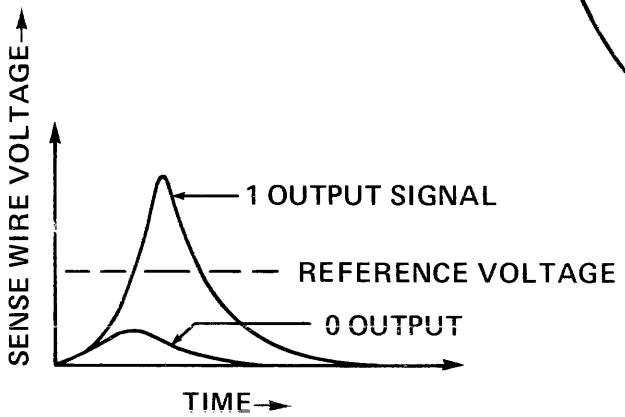
WRITE "0"

INHIBIT CURRENT PREVENTS
CORE FROM SWITCHING

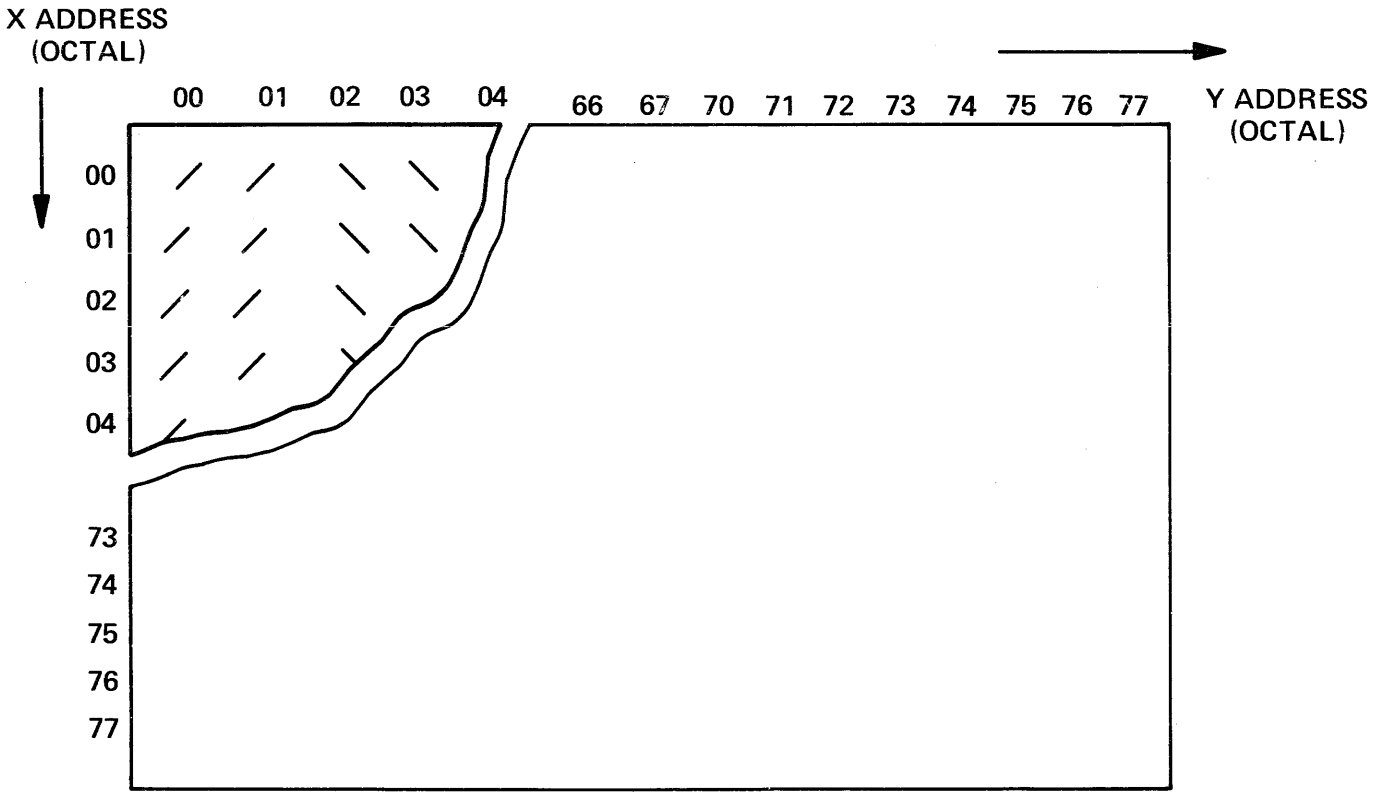
THE 3-WIRE SYSTEM

THE SENSE-INHIBIT WIRE ACTS AS A SENSE LINE DURING THE READ CYCLE, AND AS AN INHIBIT LINE DURING THE WRITE CYCLE.





THE 4K BIT MATRIX



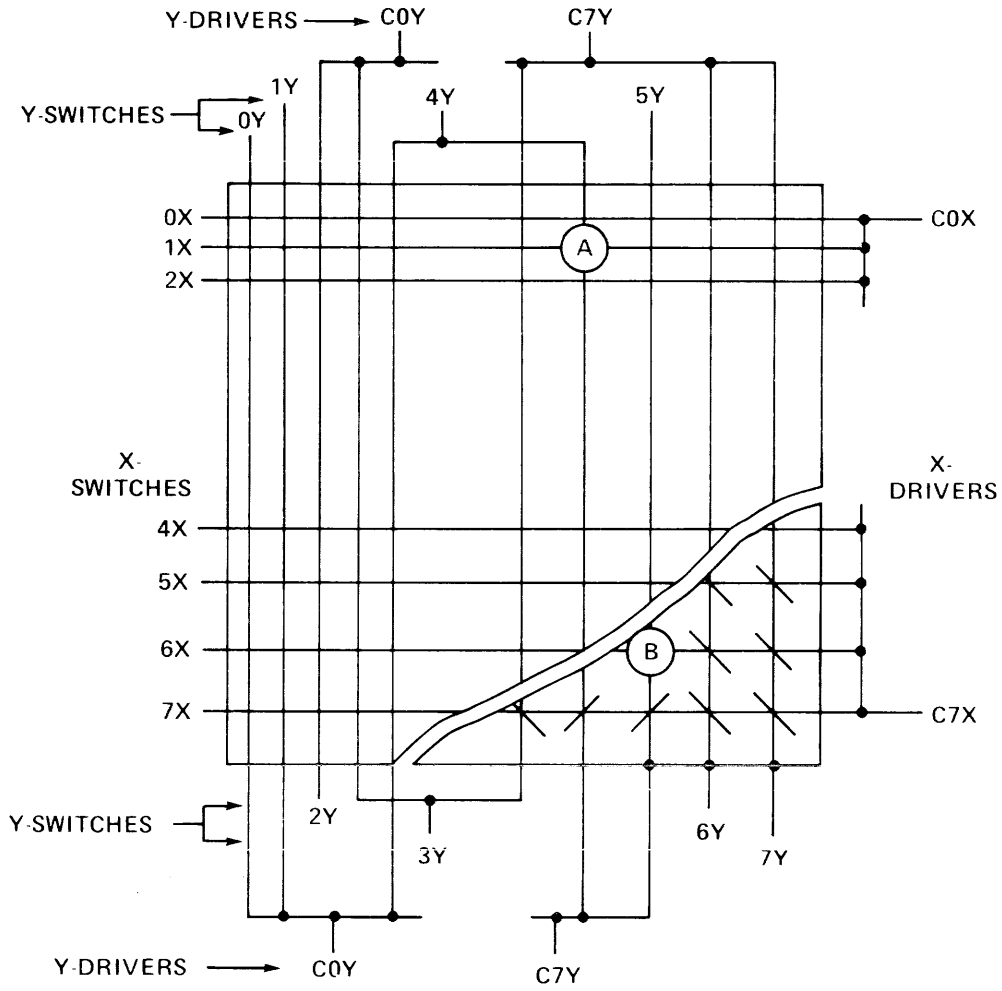
4K BIT MATRIX CONTAINS 64 X 64 = 4096_{10} ($10{,}000_8$) CORES

MEMORY ADDRESS IDENTIFICATION

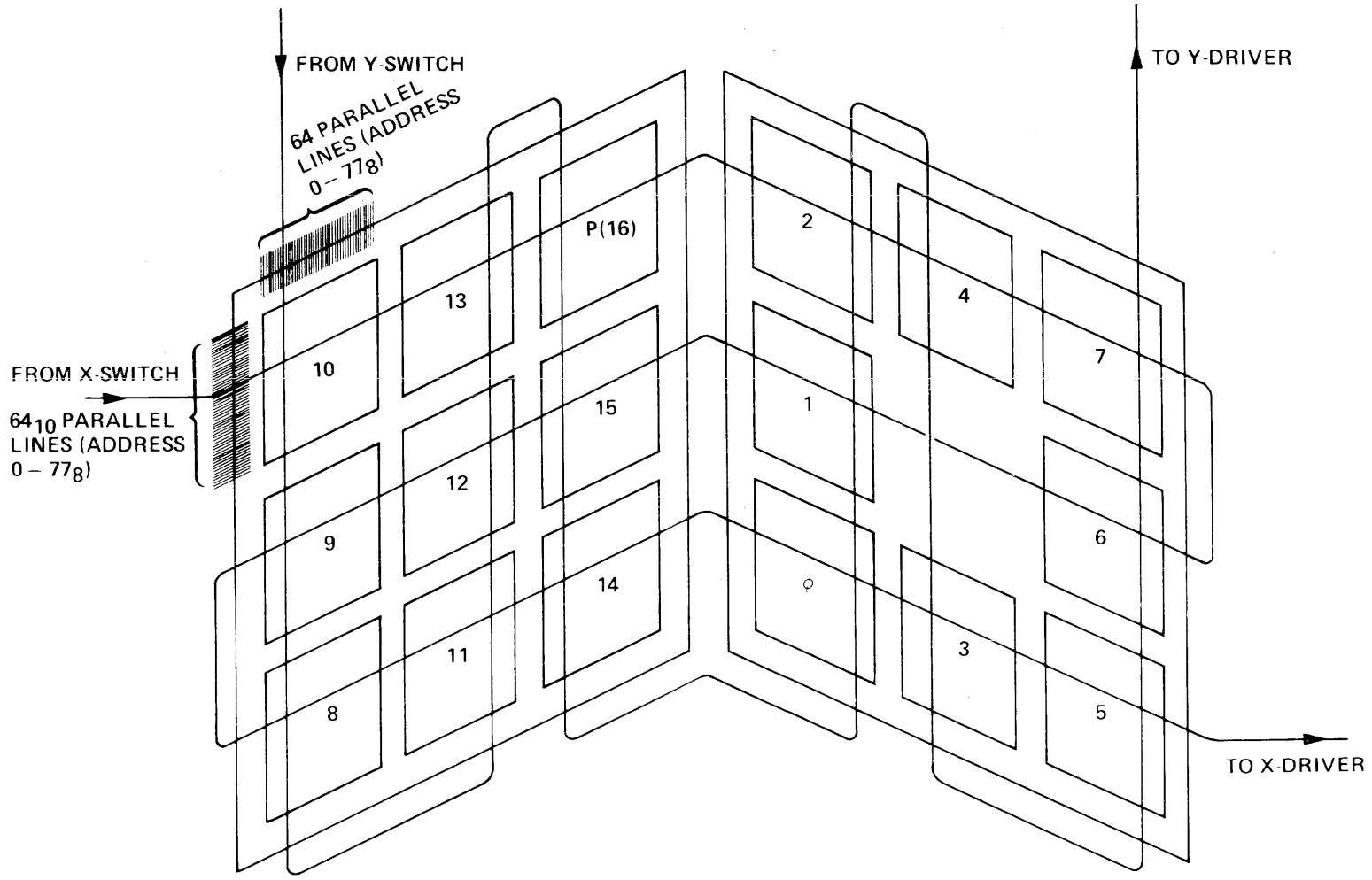
M-REGISTER

	Y		X	
	DR.	SW.	DR.	SW.
A	7	4	0	1
B	7	5	7	6

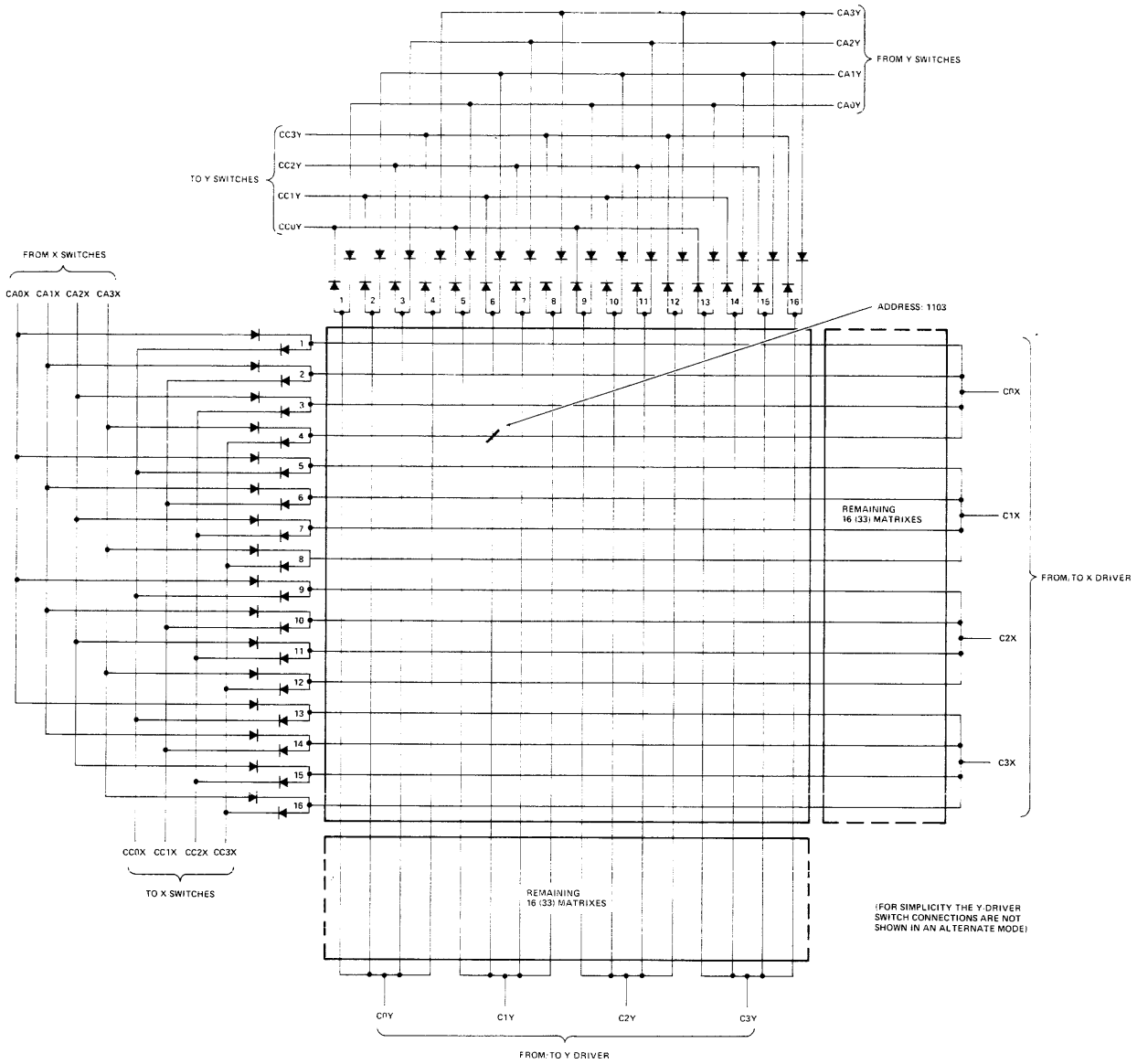
C0 – C7 ARE DRIVERS
0 – 7 ARE SWITCHES



4K CORE STACK

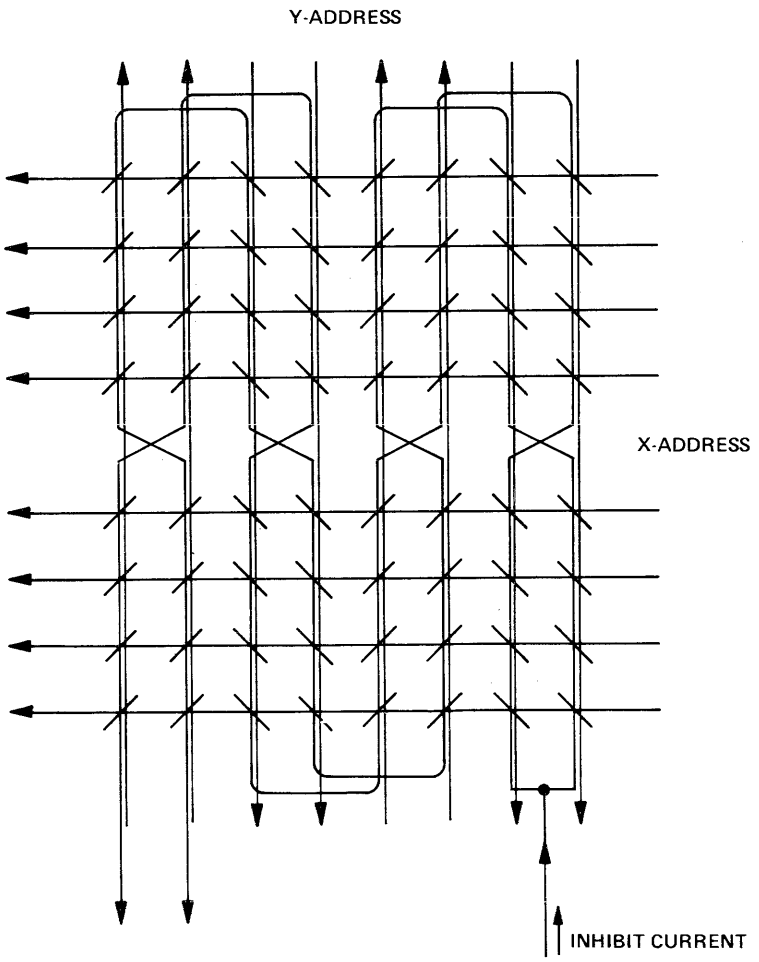


ISOLATION DIODES ON A 16 X 16 MATRIX

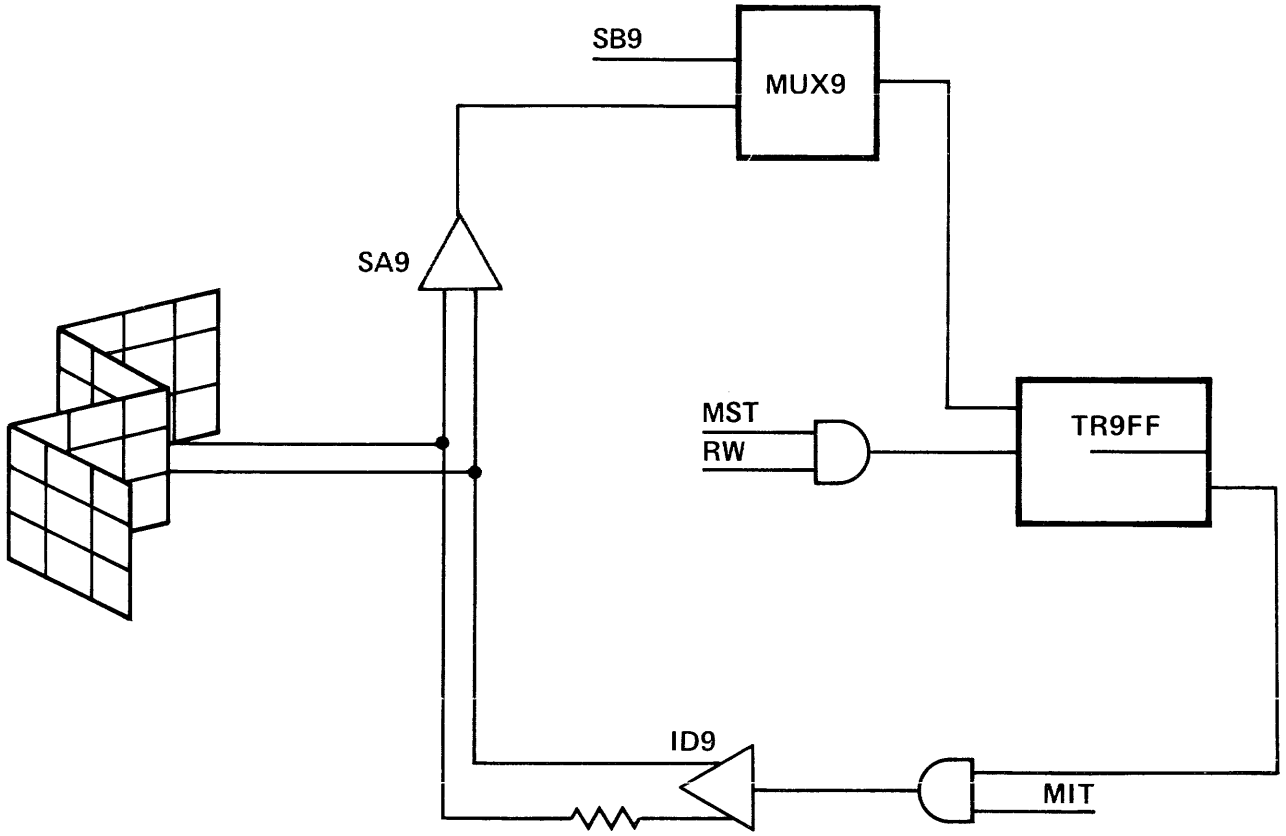


WIRE LAY OUT IN 3-WIRE MEMORY

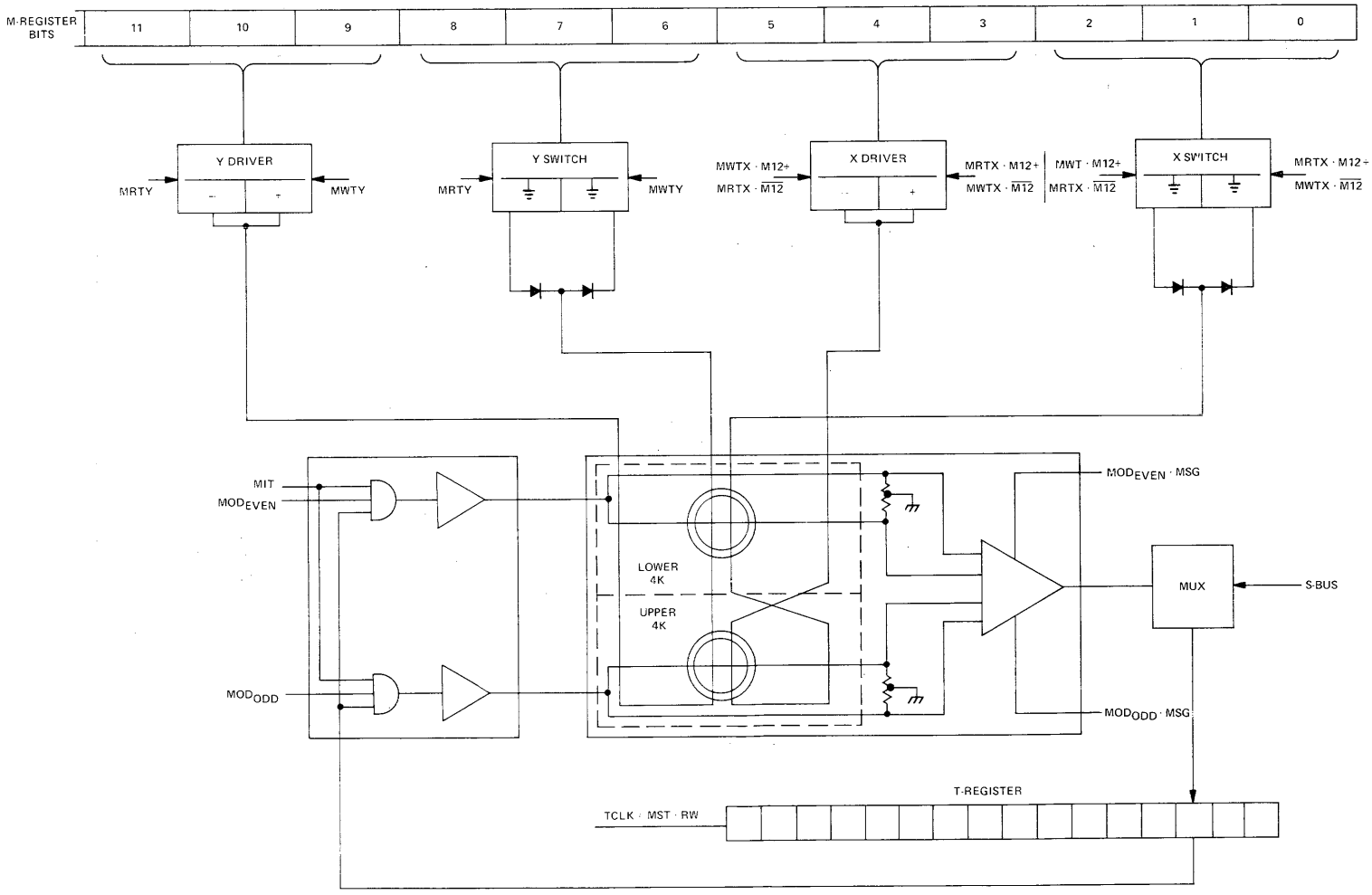
(8 X 8 MATRIX)



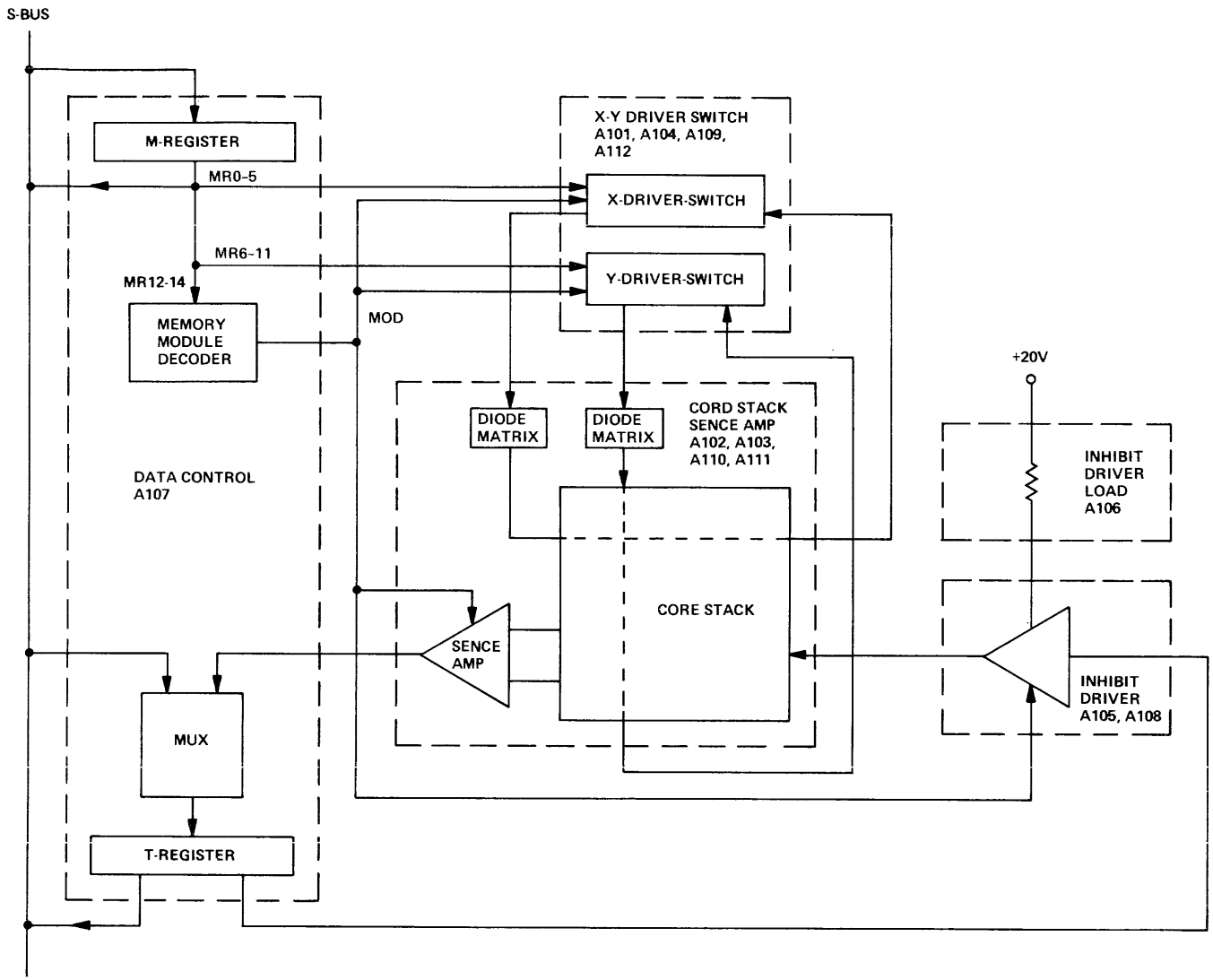
TYPICAL DATA SENSE AND INHIBIT



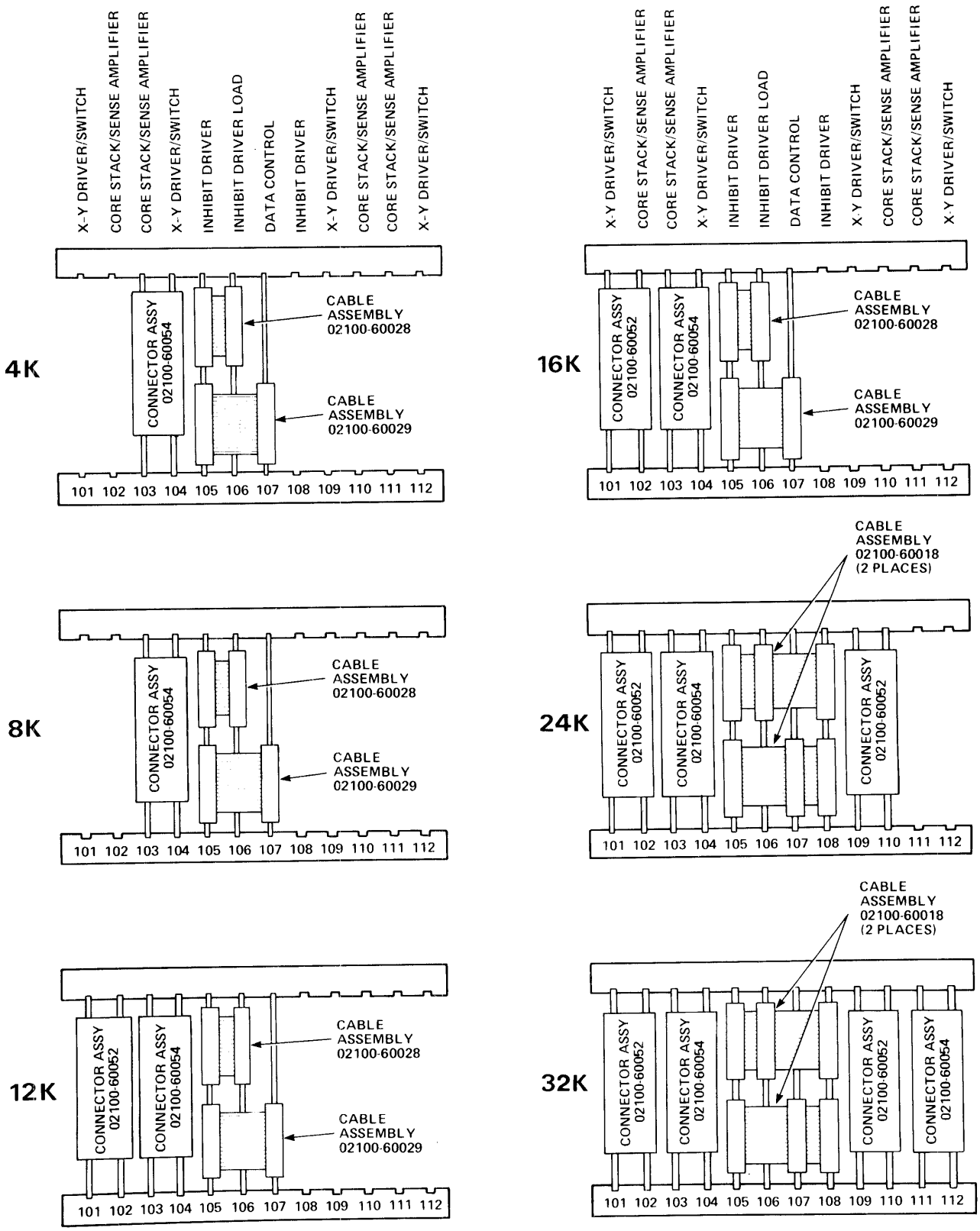
SIMPLIFIED MEMORY ADDRESSING

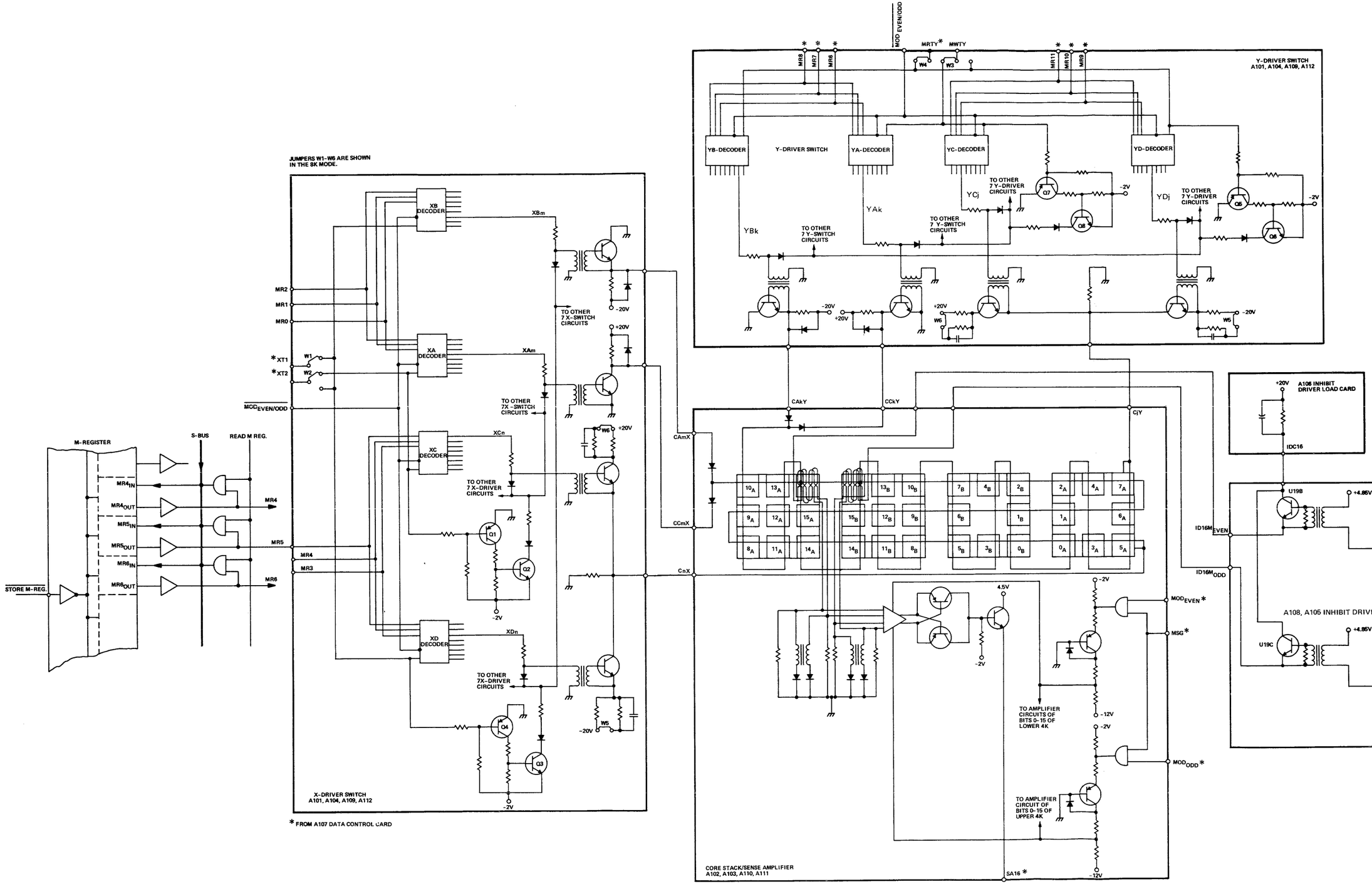


MEMORY BLOCK DIAGRAM

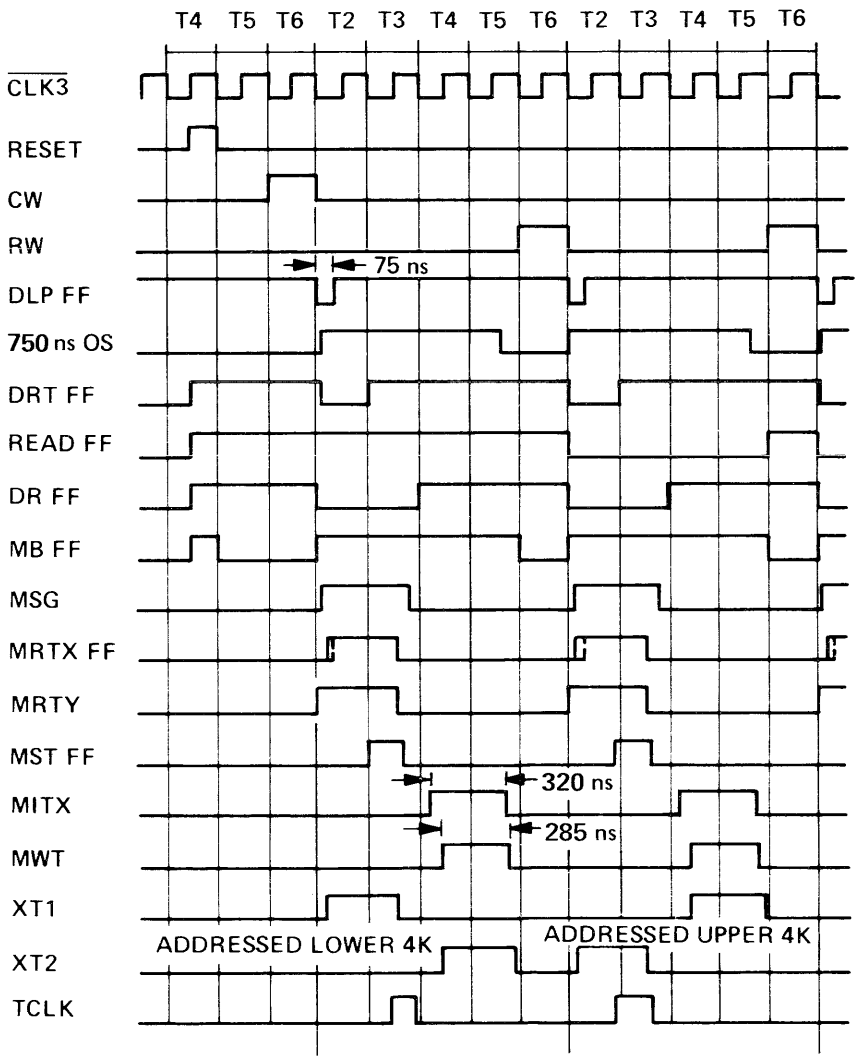


MEMORY SECTION CARD CAGE LOADING CONFIGURATIONS




JUMPERS W1-W6 ARE SHOWN IN THE 8K MODE.
M-REGISTER
S-BUS
READ M REG.
STORE M-REG.
MR4IN
MR4OUT
MR5IN
MR5OUT
MR6IN
MR6OUT
MR4
MR5
MR6
MR2
MR1
MR0
MR3
*XT1
*XT2
W1
W2
MOD EVEN/ODD
X8 DECODER
XA DECODER
XC DECODER
XD DECODER
XBm
XAm
XCn
XDn
TO OTHER 7 X-SWITCH CIRCUITS
TO OTHER 7X -SWITCH CIRCUITS
TO OTHER 7X-DRIVER CIRCUITS
TO OTHER 7X-DRIVER CIRCUITS
+20V
-20V
-2V
W6
W5
Q1
Q2
Q3
Q4
X-DRIVER SWITCH
A101, A104, A109, A112
* FROM A107 DATA CONTROL CARD
MOD EVEN/ODD
MRTY*
MWTY
W4
W3
MR8
MR7
MR6
MR11
MR10
MR9
Y-DRIVER SWITCH
A101, A104, A108, A112
YB-DECODER
Y-DRIVER SWITCH
YA-DECODER
YC-DECODER
YD-DECODER
YBk
YAk
YCj
YDj
TO OTHER 7 Y-SWITCH CIRCUITS
TO OTHER 7 Y-SWITCH CIRCUITS
TO OTHER 7 Y-DRIVER CIRCUITS
TO OTHER 7 Y-DRIVER CIRCUITS
Q7
Q8
Q5
Q6
-2V
+20V
-20V
W6
W5
CAkY
CCkY
CjY
CAmX
CCmX
CnX
CORE STACK/SENSE AMPLIFIER
A102, A103, A110, A111
4.5V
-2V
-12V
MOD EVEN *
MSG *
MOD ODD *
SA16 *
TO AMPLIFIER CIRCUITS OF BITS 0-15 OF LOWER 4K
TO AMPLIFIER CIRCUIT OF BITS 0-15 OF UPPER 4K
A106 INHIBIT DRIVER LOAD CARD
+20V
IDC16
ID16M EVEN
ID16M ODD
U19B
U19C
+4.85V
TO INHIBIT DRIVER CIRCUIT OF BITS 0-15 OF LOWER 4K
TO INHIBIT DRIVER CIRCUIT OF BITS 0-15 OF UPPER 4K
MOD0,4 *
IDEMS 3X *
TR16 *
MITX *
MOD1,5 *
A108, A105 INHIBIT DRIVER CARD

HP 2100 DATA CONTROL CARD SIGNALS (A107)

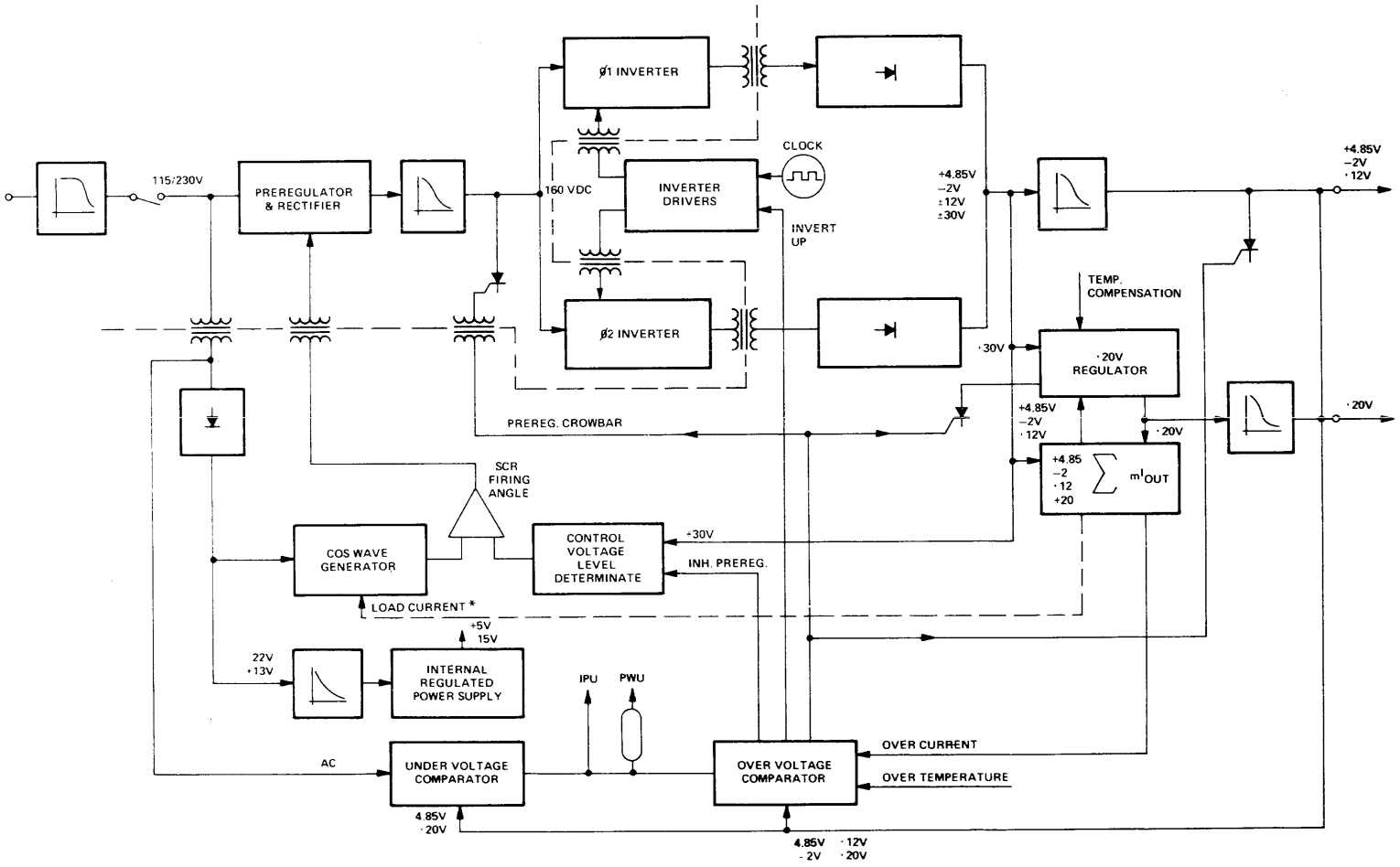


power supply 5

LESSON 5

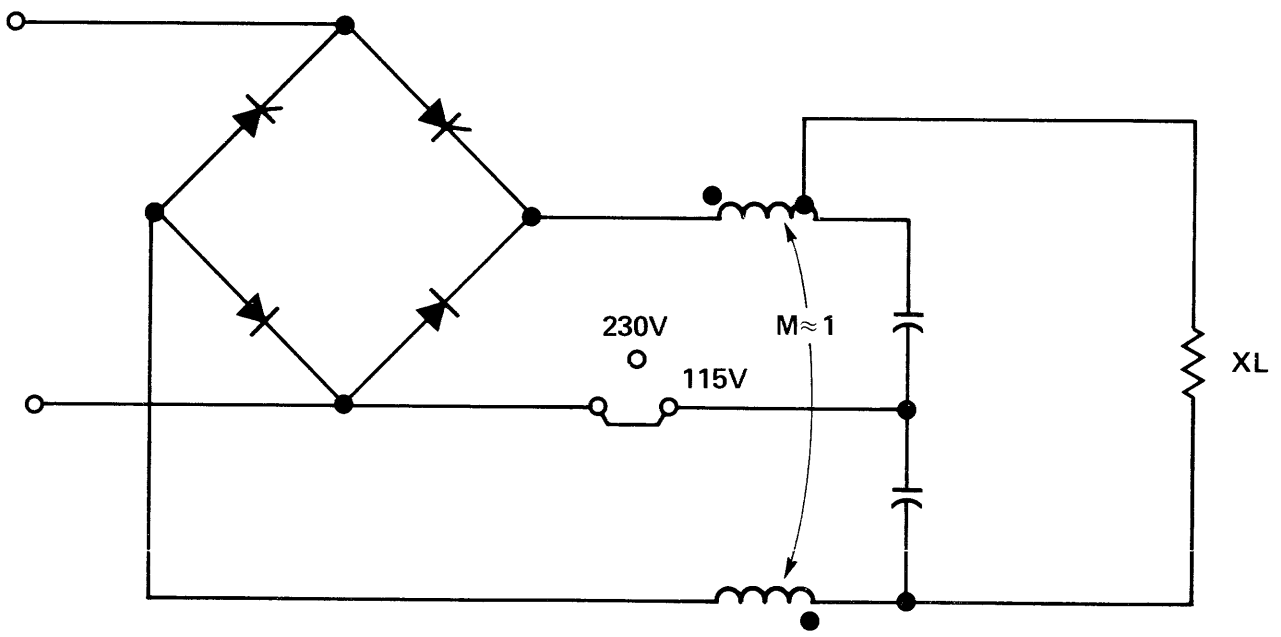
POWER SUPPLY

2100 POWER SUPPLY, BLOCK DIAGRAM

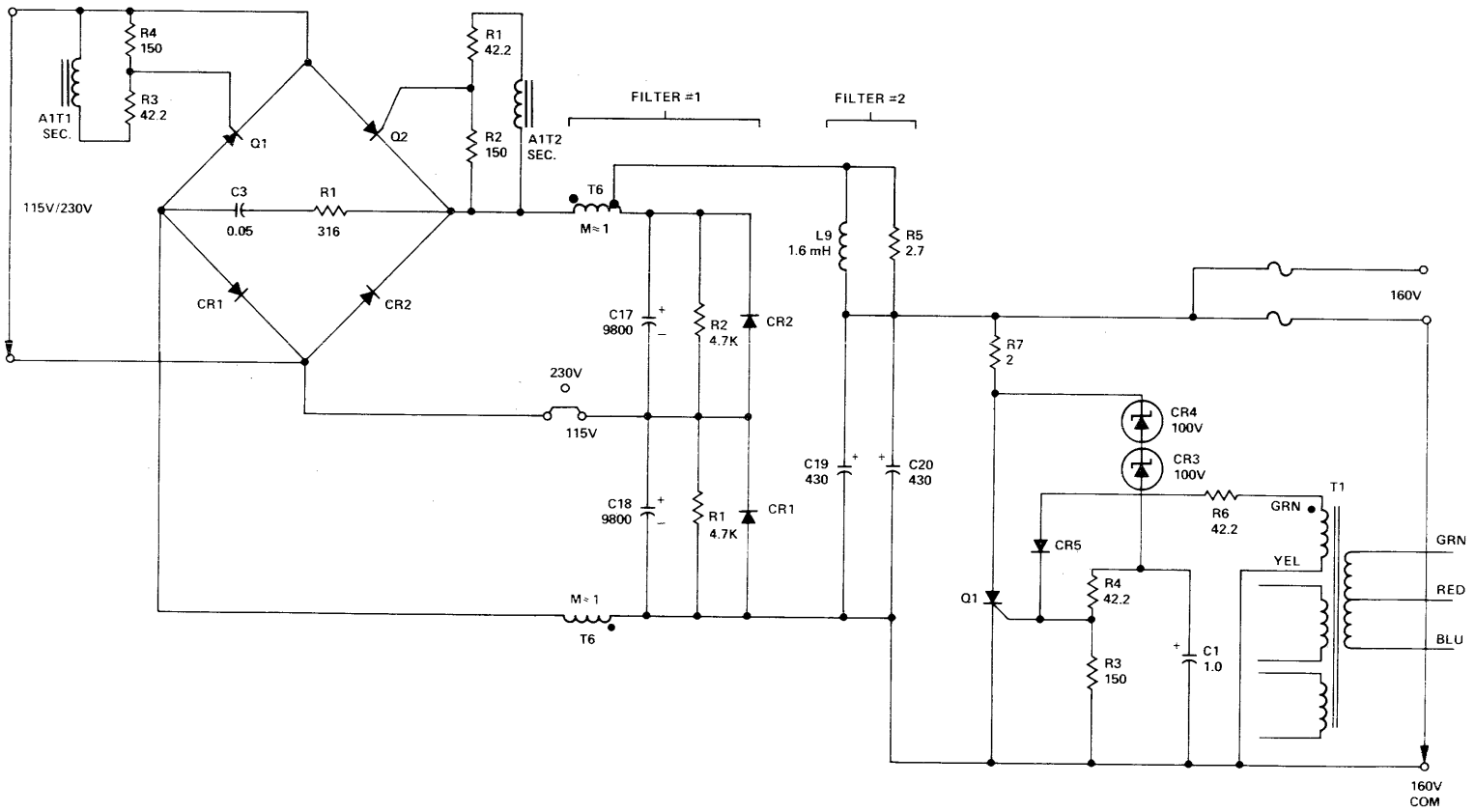


* LOAD CURRENT FEEDBACK TO COS WAVE GENERATOR IN POWER SUPPLIES PRIOR TO REV. 1240.

2100 POWER SUPPLY, SIMPLIFIED PREREGULATOR

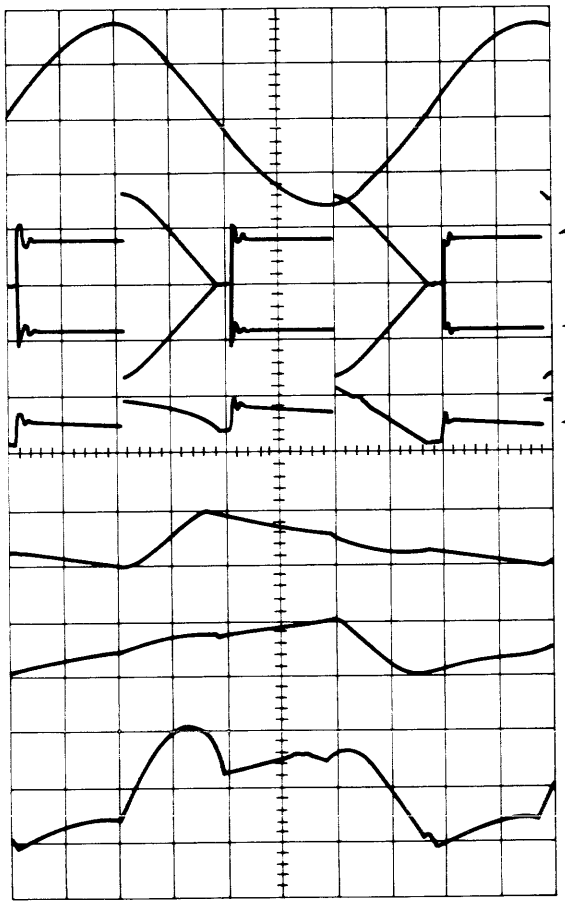


2100 POWER SUPPLY, PREREGULATOR



VOLTAGES ON PREREGULATOR & 160V OUTPUT BOARD

TIME SCALE: 2 ns/cm



115V LINE VOLTAGE MEASURED BETWEEN TB1-5 AND TB1-6
SCALE: 100 $V_{DC/CM}$

OUTPUT VOLTAGE OF PREREGULATOR BRIDGE MEASURED BETWEEN KATHODE OF CR2 AND TB1-6
SCALE: 100 $V_{DC/CM}$

OUTPUT VOLTAGE OF PREREGULATOR BRIDGE MEASURED ANODE OF CR1 AND TB1-6
SCALE: 100 $V_{DC/CM}$

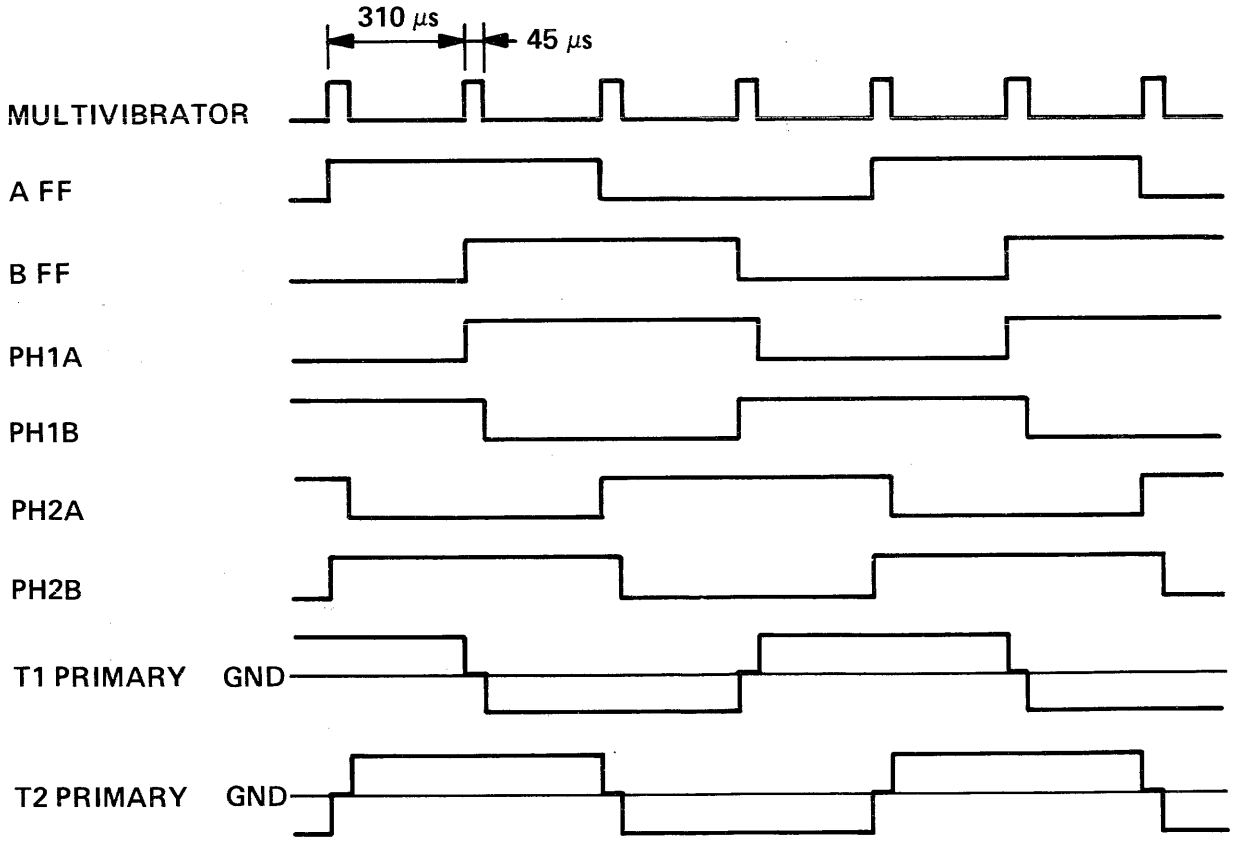
RIPPLE VOLTAGE ON TAP OF TRANSFORMER T6 MEASURED BETWEEN TAP AND TB1-6
SCALE: 5 $V_{AC/CM}$; +80 V_{DC}

RIPPLE VOLTAGE ACROSS FILTER CAPACITOR C17 (GND=TB1-6)
SCALE: 2 $V_{AC/CM}$; +80 V_{DC}

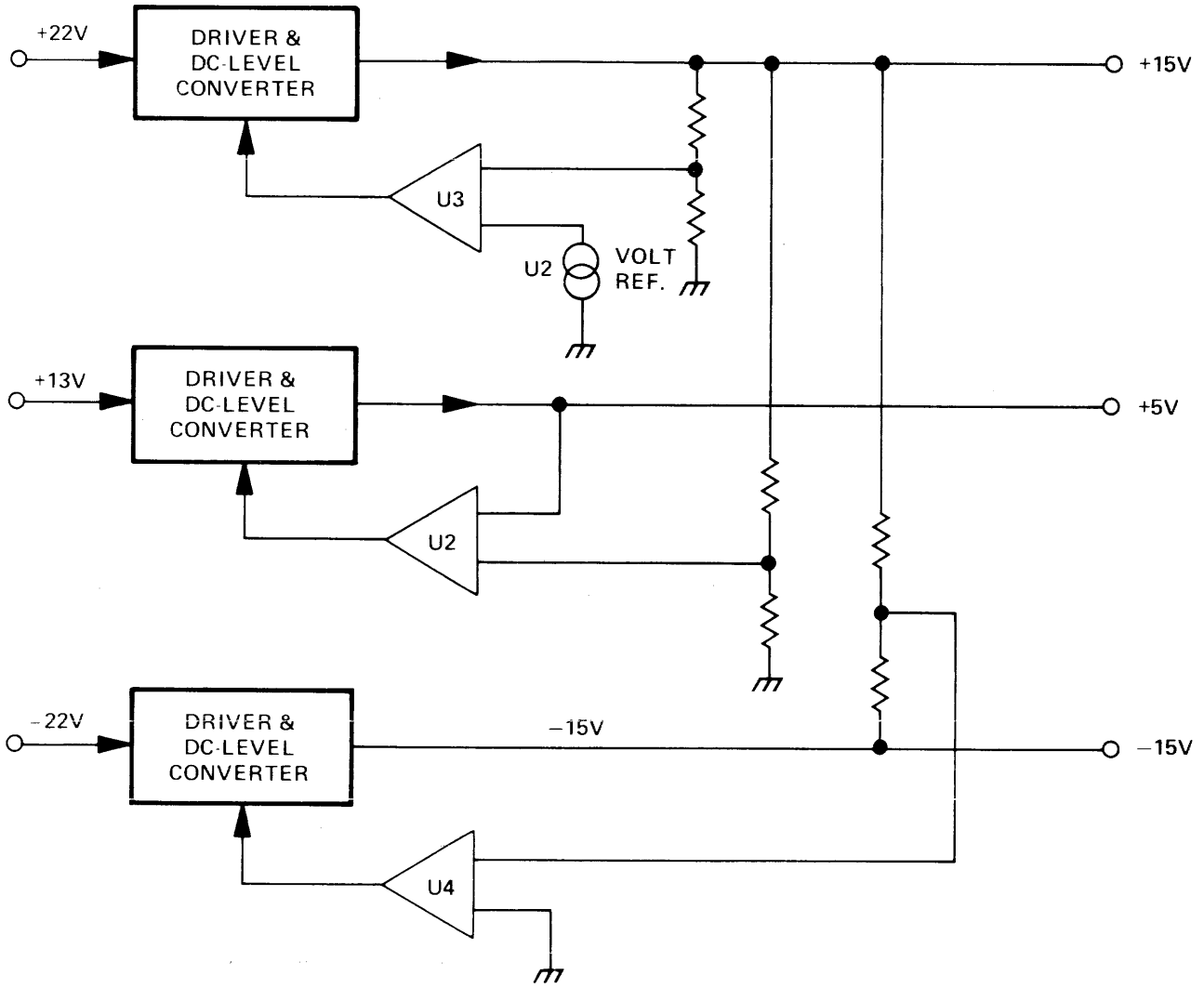
RIPPLE VOLTAGE ACROSS FILTER CAPACITOR C18 (GND=TB1-6)
SCALE: 2 $V_{AC/CM}$; -80 V_{DC}

RIPPLE VOLTAGE ON +160 V. MEASURED BETWEEN TERMINAL E52 AND TB1-6
SCALE: 1 $V_{AC/CM}$; +80 V_{DC}

2100 POWER SUPPLY, INVERTER DRIVER

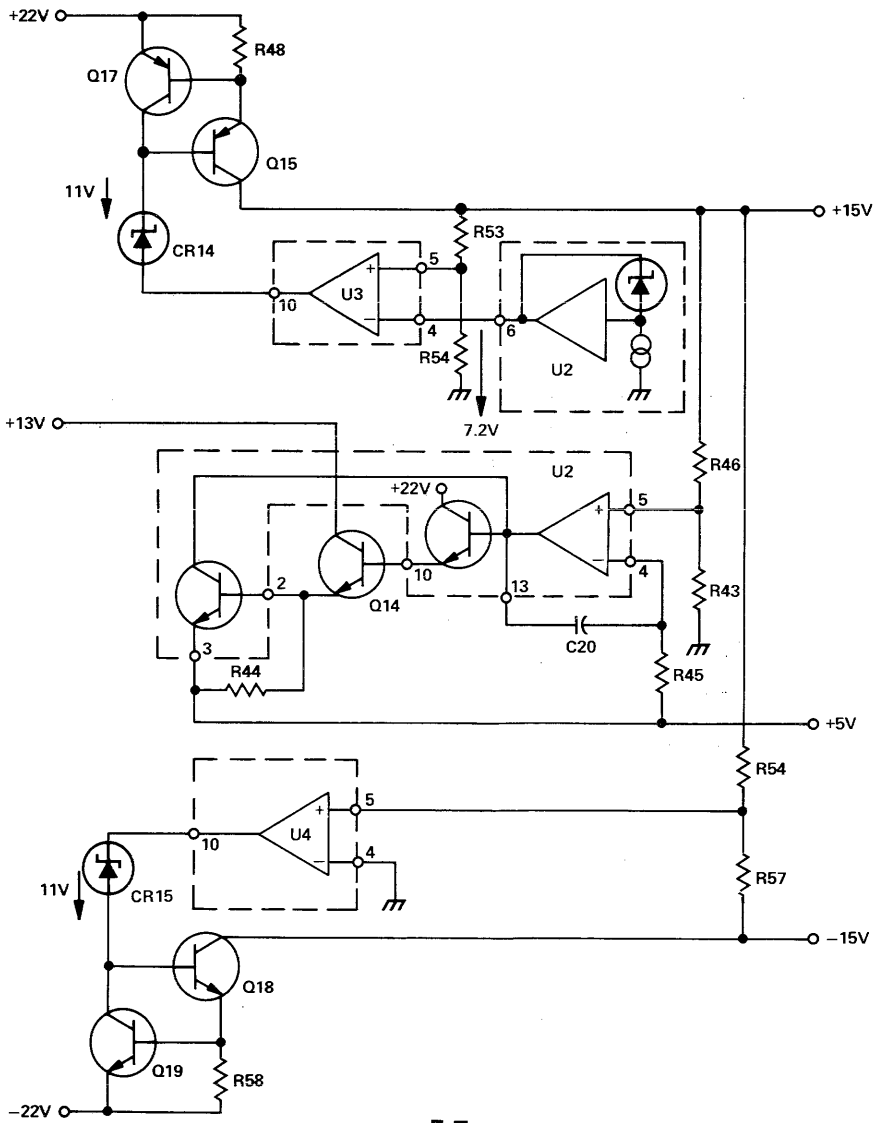


2100 POWER SUPPLY, BLOCK DIAGRAM OF INTERNAL VOLTAGE SUPPLY REGULATION

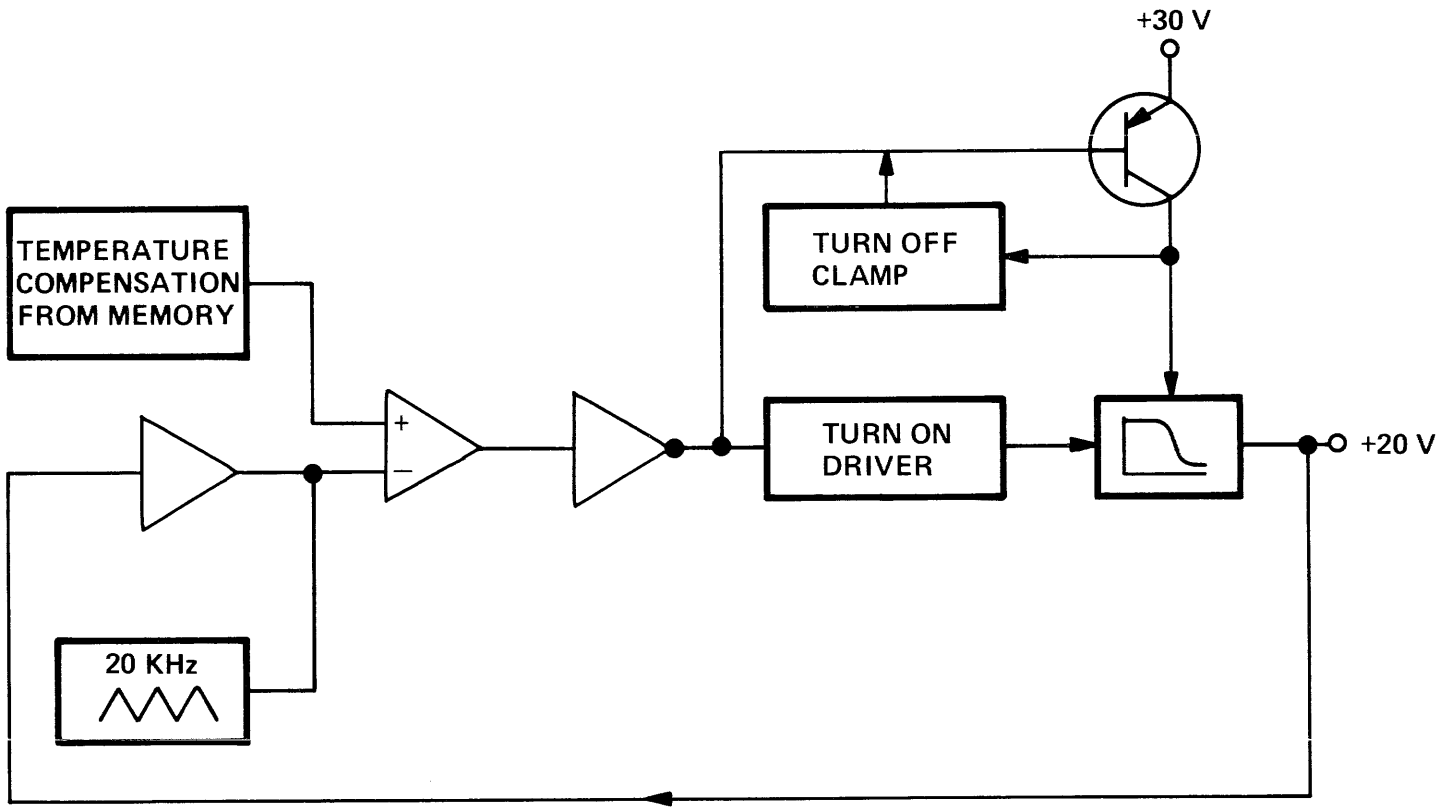


2100 POWER SUPPLY

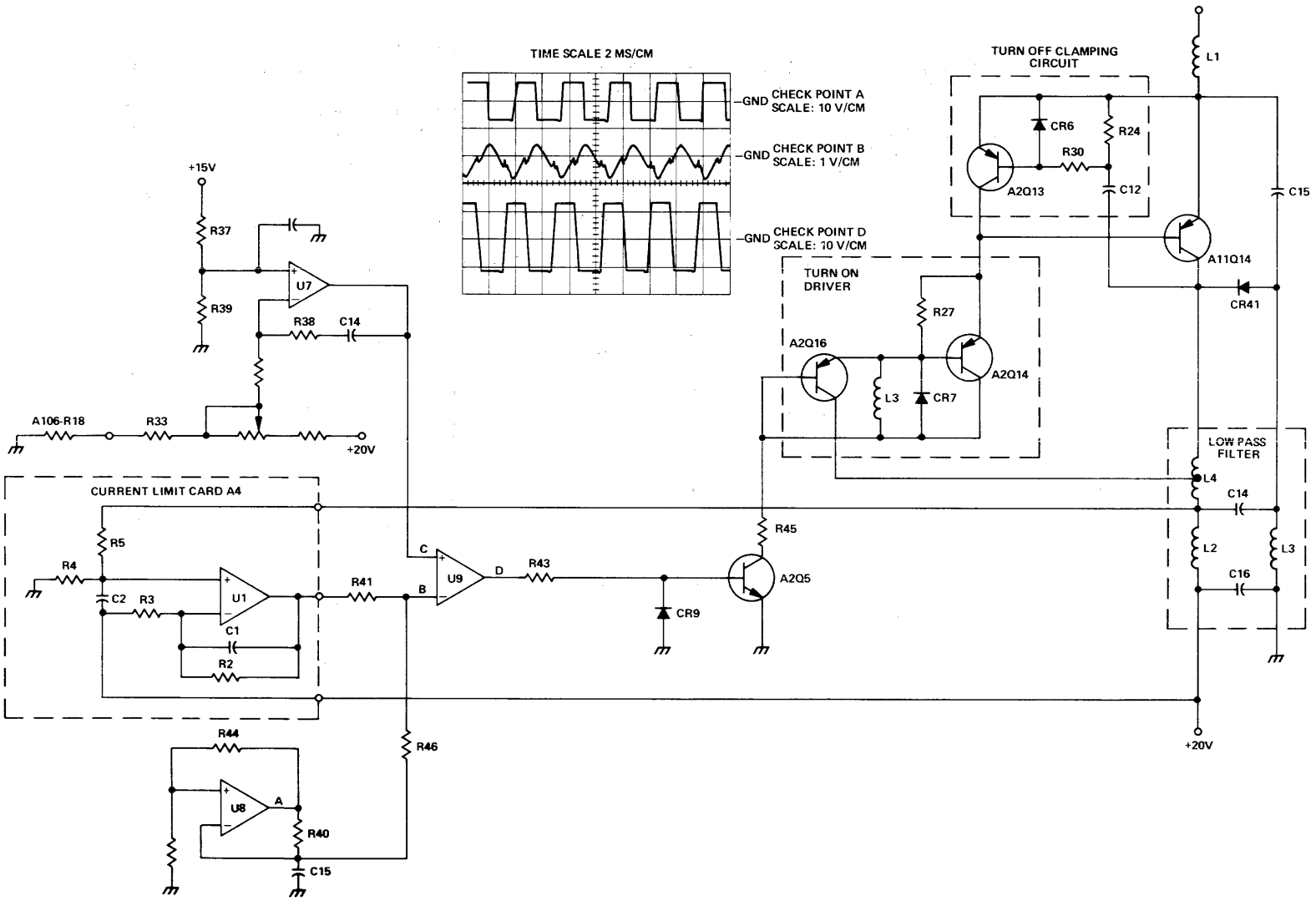
INTERNAL VOLTAGE SUPPLY REGULATOR



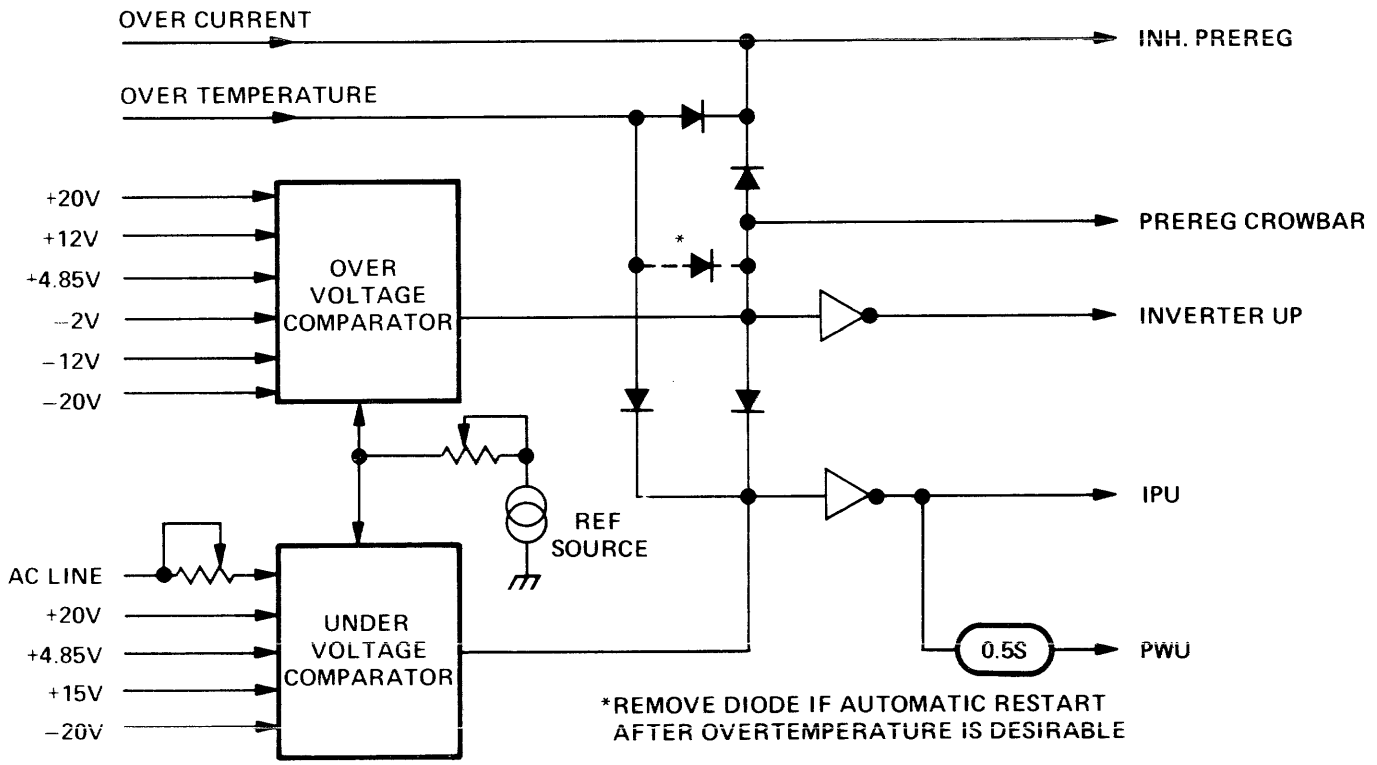
BLOCK DIAGRAM +20V REGULATOR



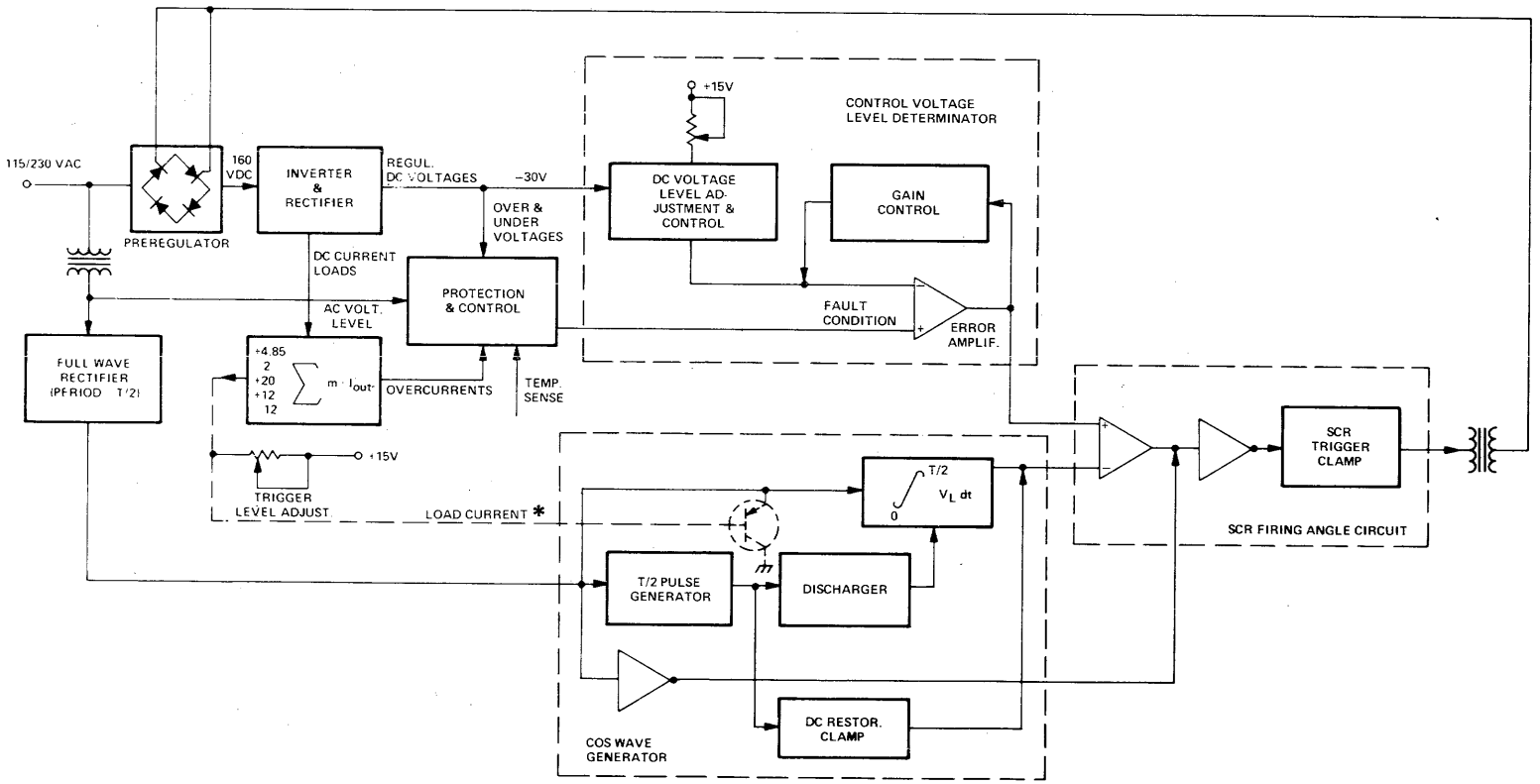
+20V REGULATOR



2100 POWER SUPPLY, BLOCK DIAGRAM PROTECTION & CONTROL BOARD



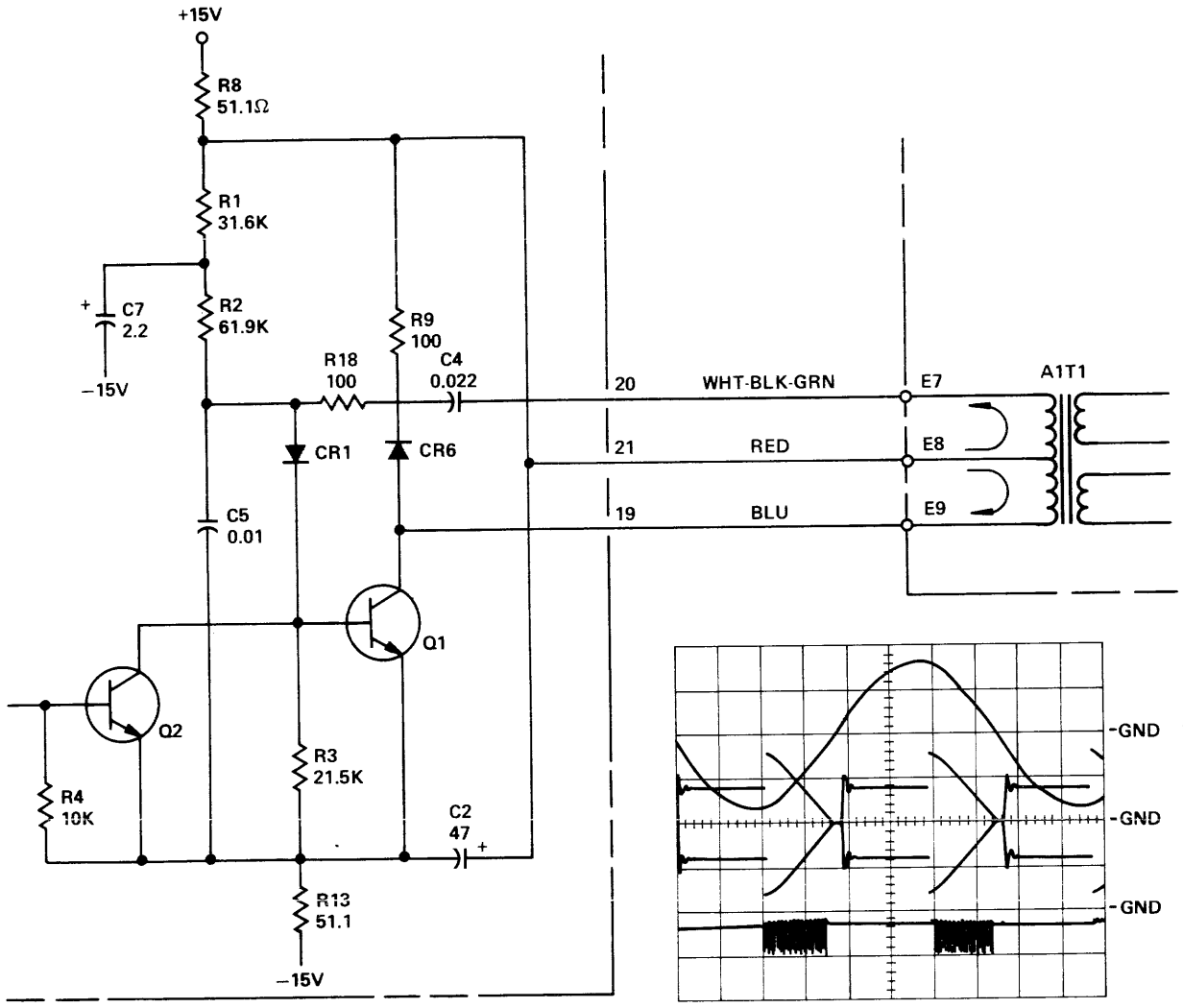
2100 POWER SUPPLY, PREREGULATOR CONTROL BLOCK DIAGRAM



* LOAD CURRENT FEEDBACK TO COS WAVE GENERATOR IN POWER SUPPLIES PRIOR TO REV. 1240.

TRIGGER CLAMP

(REGENERATIVE PULSE GENERATOR)



115V LINE VOLTAGE

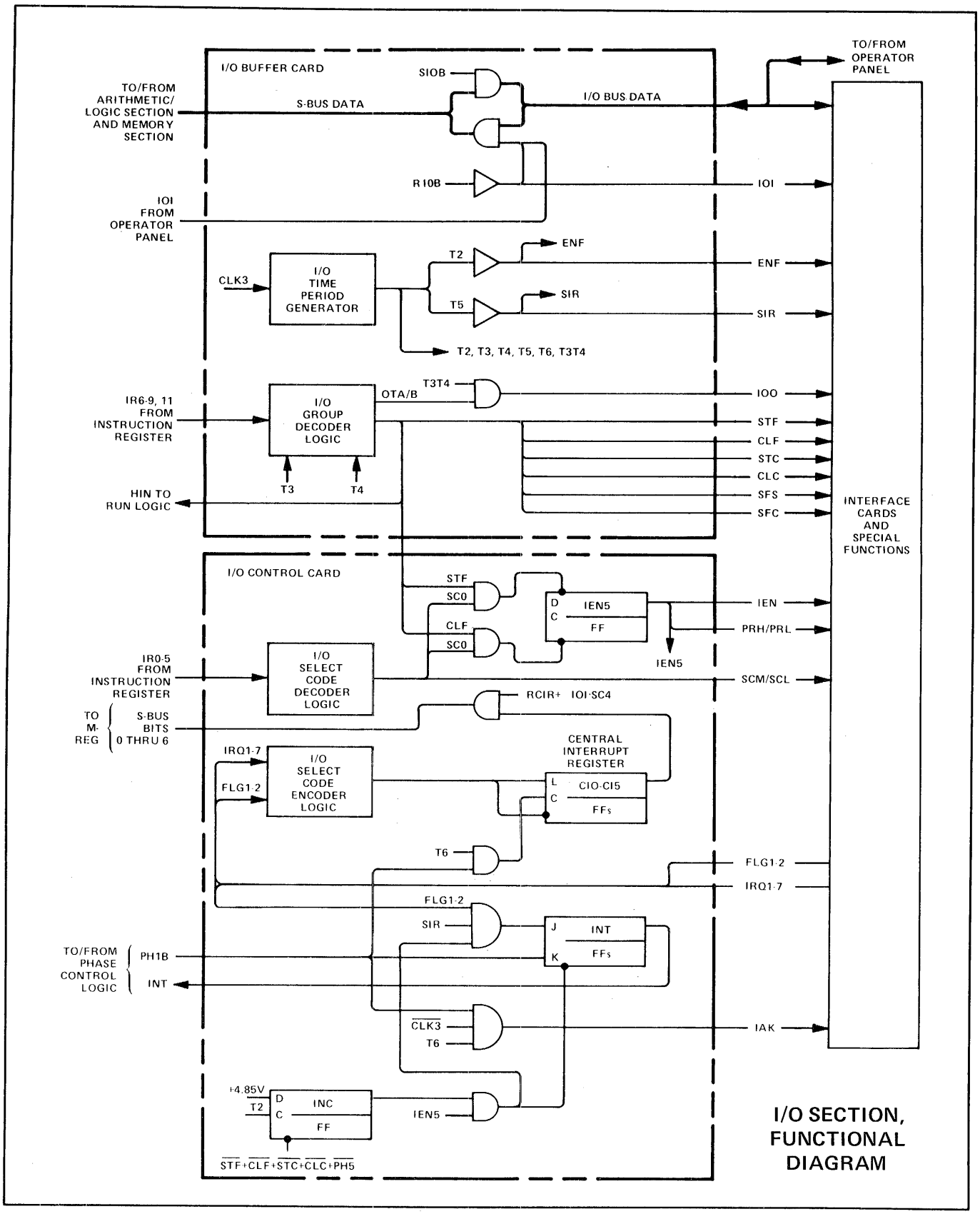
OUTPUT VOLTAGE OF PREREGULATOR

TRIGGER CLAMP OUTPUT COLLECTOR OF Q1 SCALE 20V/CM

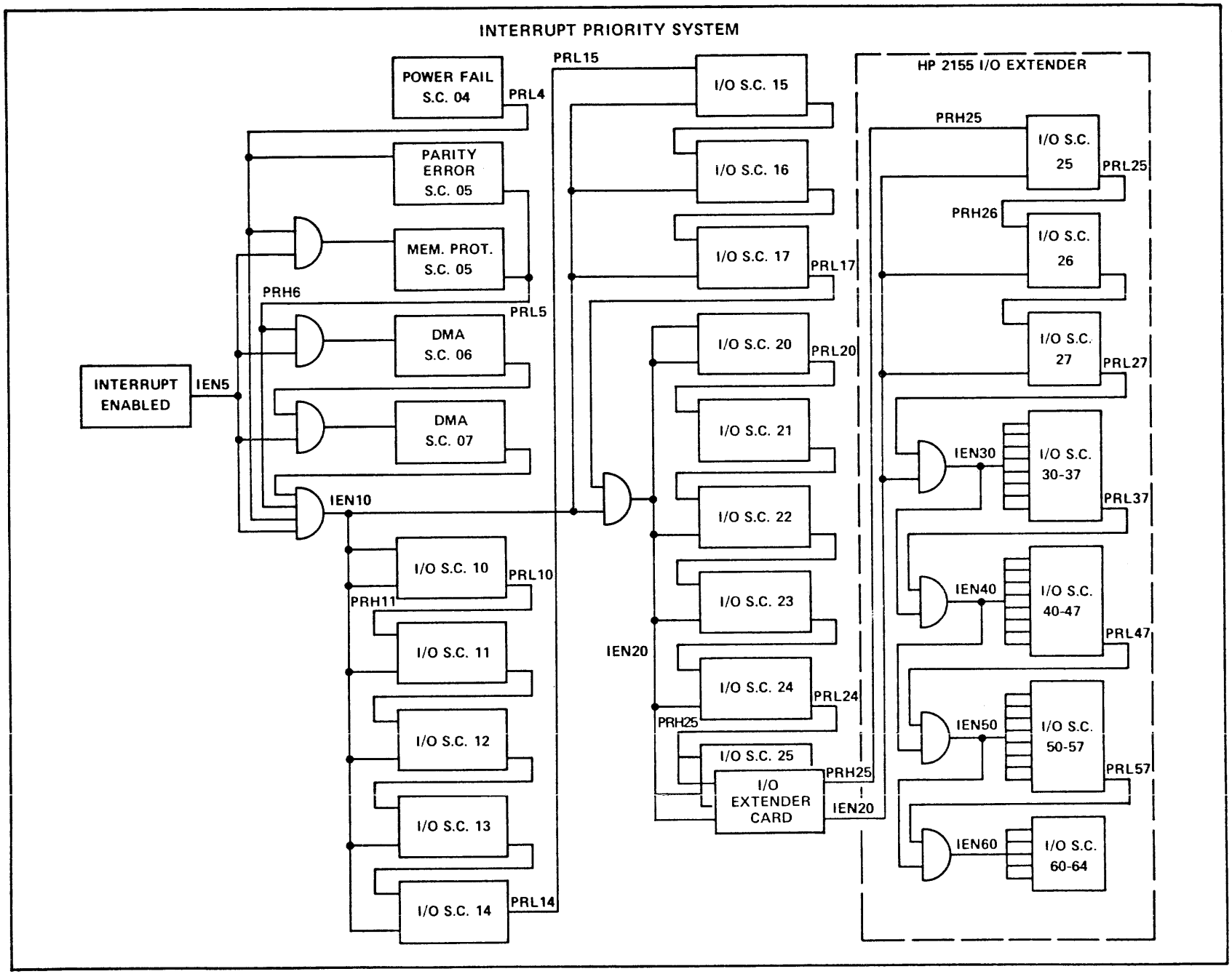
input/output and direct memory access 6

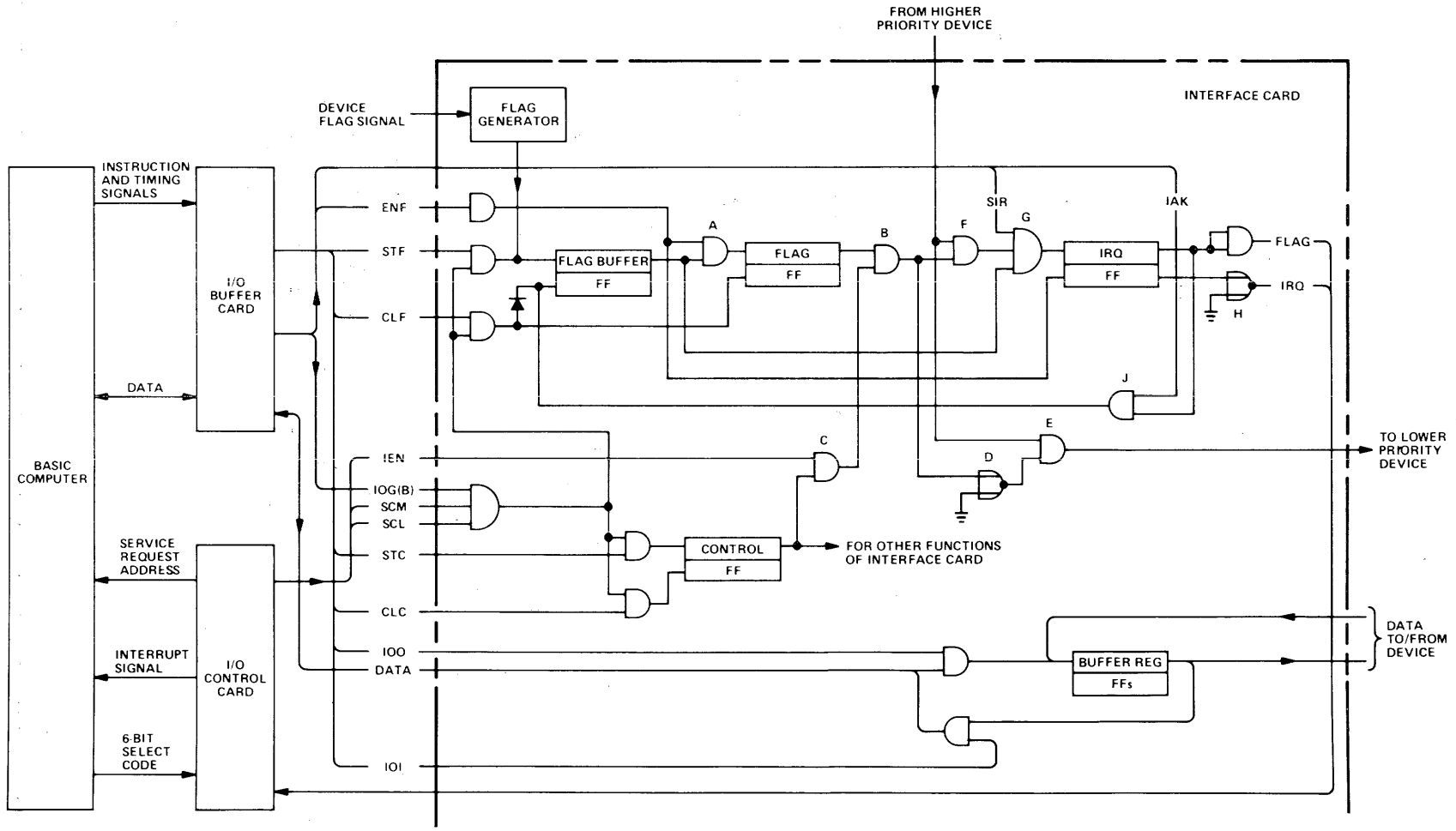
LESSON 6

INPUT/OUTPUT AND DIRECT MEMORY ACCESS



I/O SECTION, FUNCTIONAL DIAGRAM

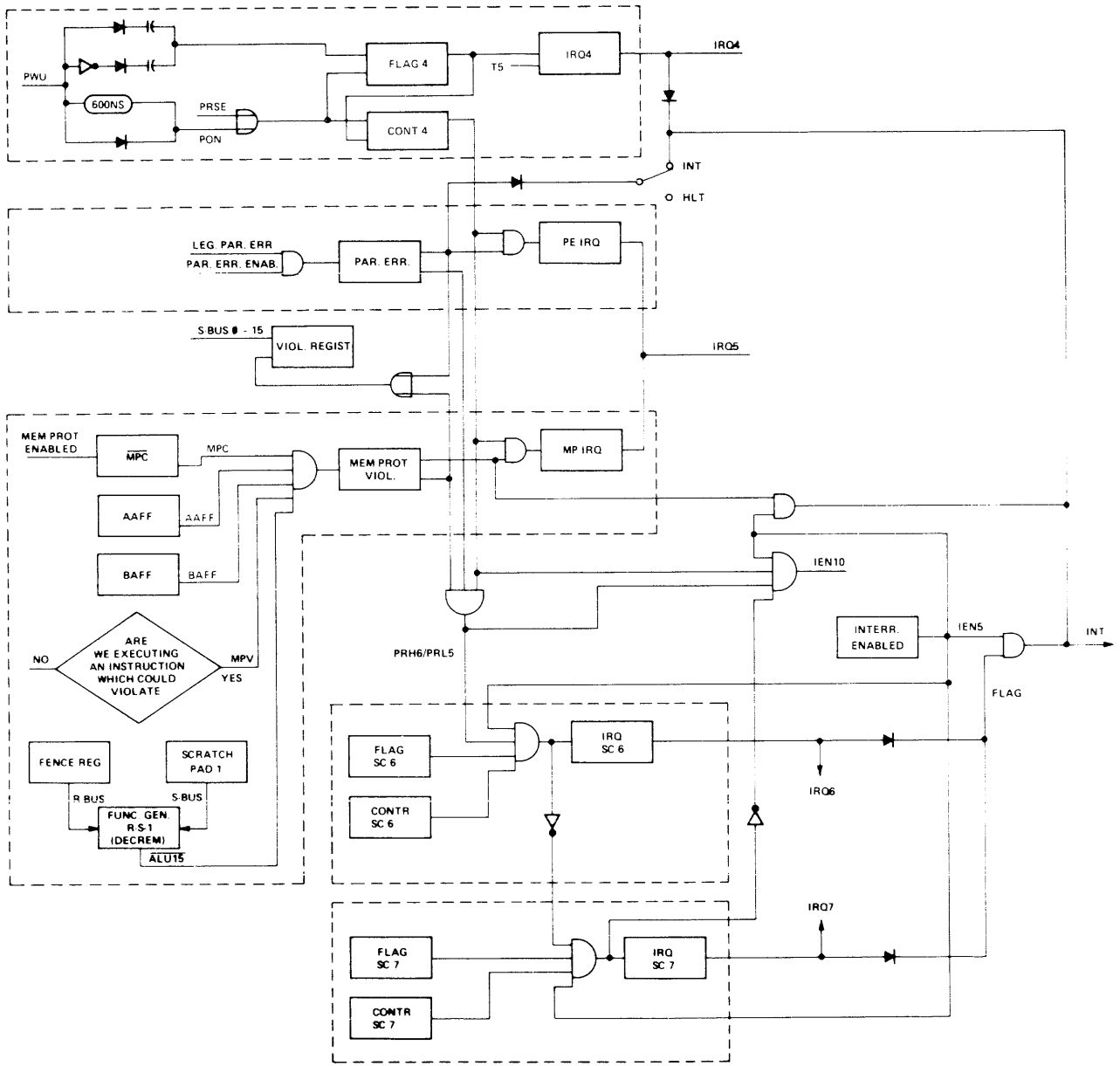

INTERRUPT PRIORITY SYSTEM
POWER FAIL S.C. 04
PRL4
PARITY ERROR S.C. 05
MEM. PROT. S.C. 05
PRH6
PRL5
DMA S.C. 06
DMA S.C. 07
INTERRUPT ENABLED
IEN5
IEN10
I/O S.C. 10
PRL10
PRH11
I/O S.C. 11
I/O S.C. 12
I/O S.C. 13
I/O S.C. 14
PRL14
PRL15
I/O S.C. 15
I/O S.C. 16
I/O S.C. 17
PRL17
I/O S.C. 20
PRL20
I/O S.C. 21
I/O S.C. 22
I/O S.C. 23
IEN20
I/O S.C. 24
PRL24
PRH25
I/O S.C. 25
I/O EXTENDER CARD
PRH25
IEN20
HP 2155 I/O EXTENDER
PRH25
I/O S.C. 25
PRL25
PRH26
I/O S.C. 26
I/O S.C. 27
PRL27
IEN30
I/O S.C. 30-37
PRL37
IEN40
I/O S.C. 40-47
PRL47
IEN50
I/O S.C. 50-57
PRL57
IEN60
I/O S.C. 60-64



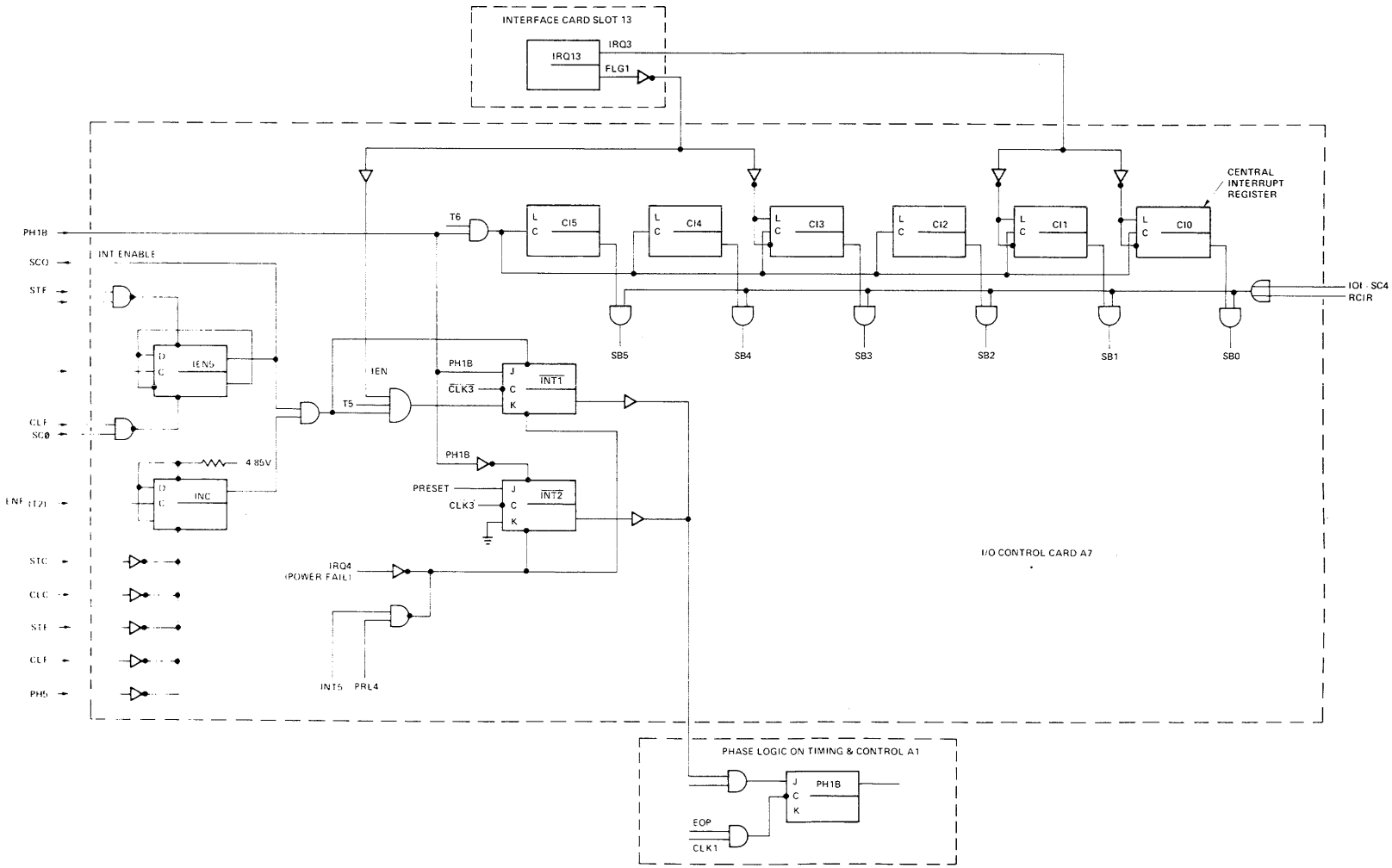
6-3

TYPICAL I/O INTERFACE CARD

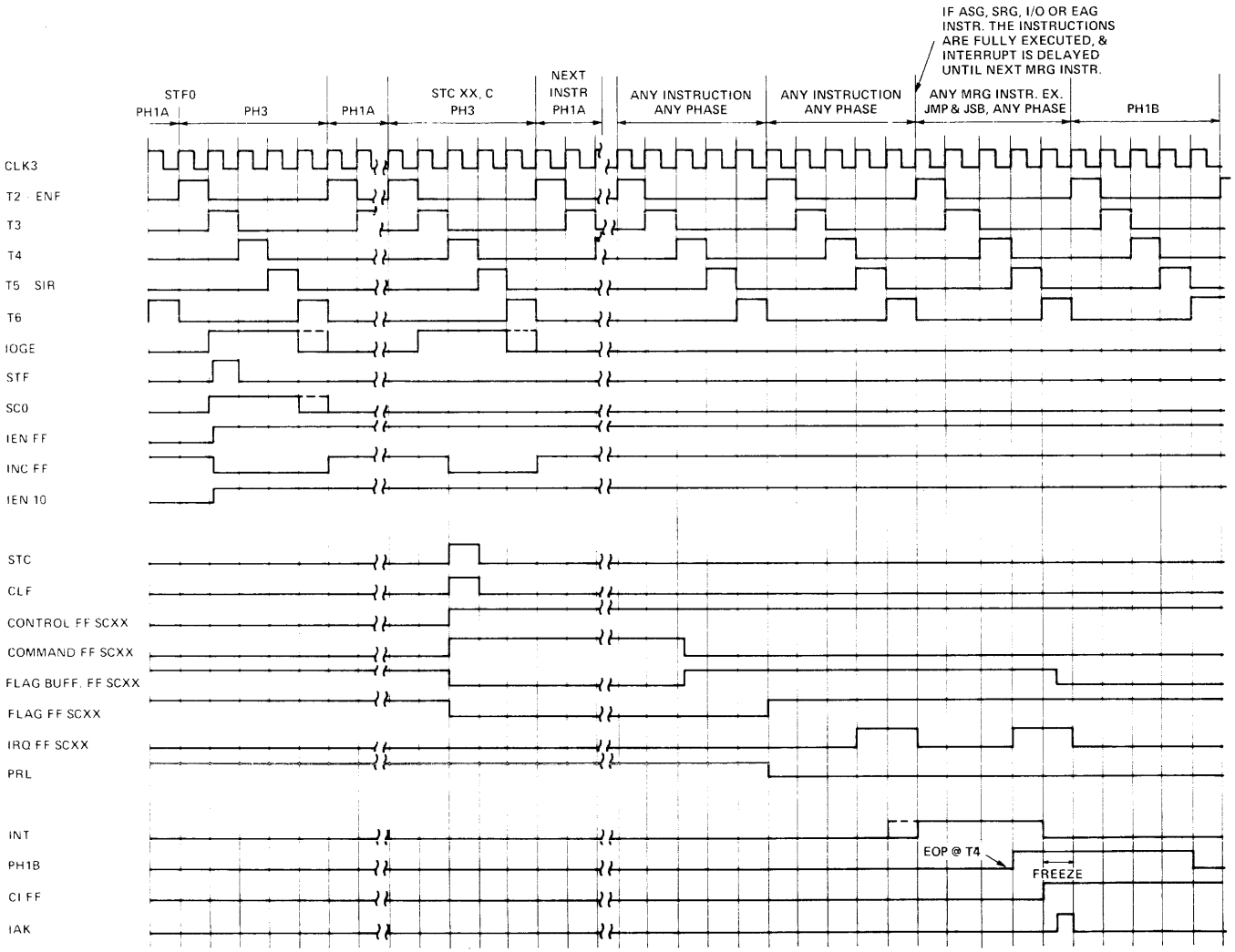
BLOCK DIAGRAM OF I/O INTERRUPT PRIORITY



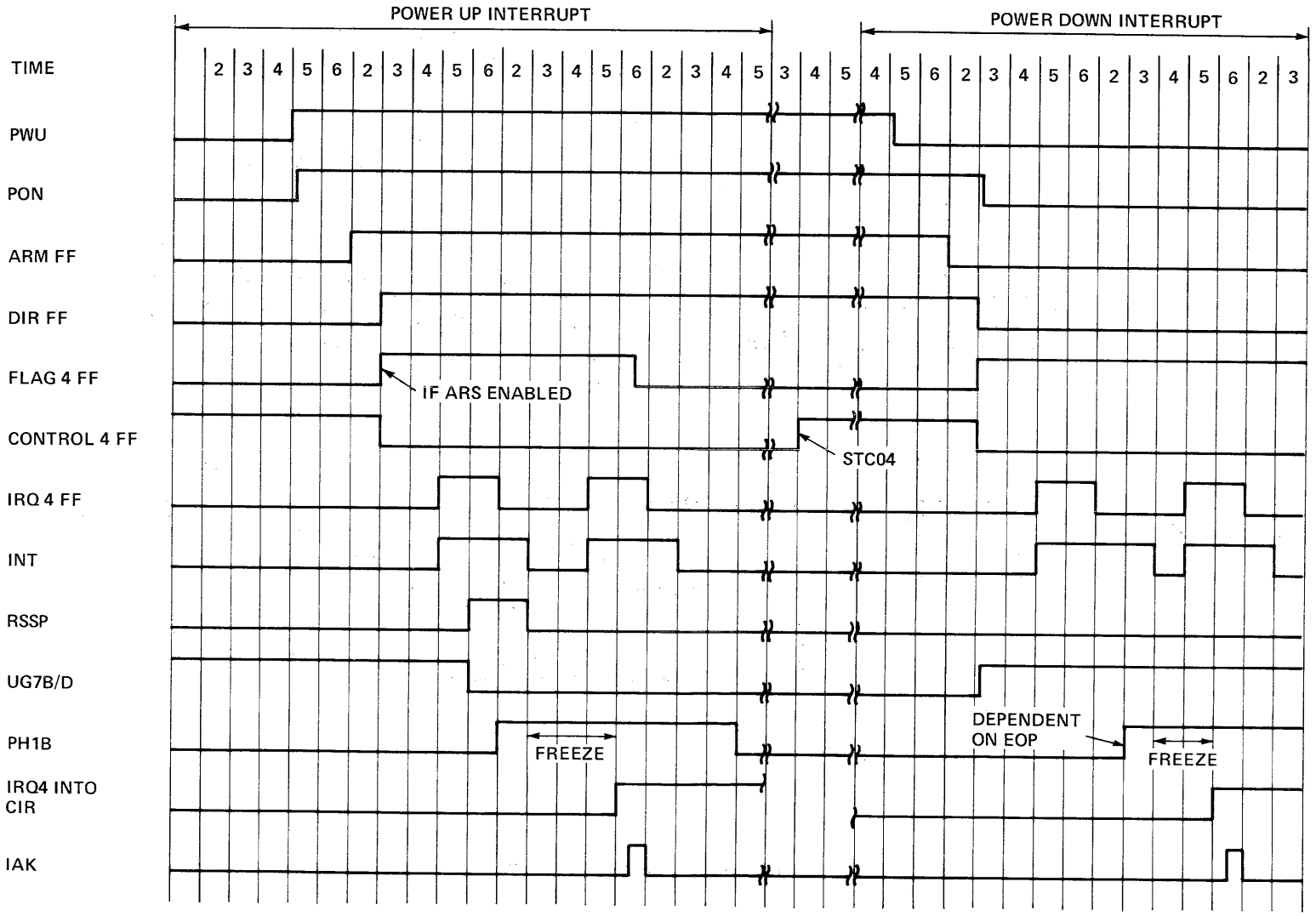
INTERRUPT SEQUENCE FROM IRQ TO PH1B



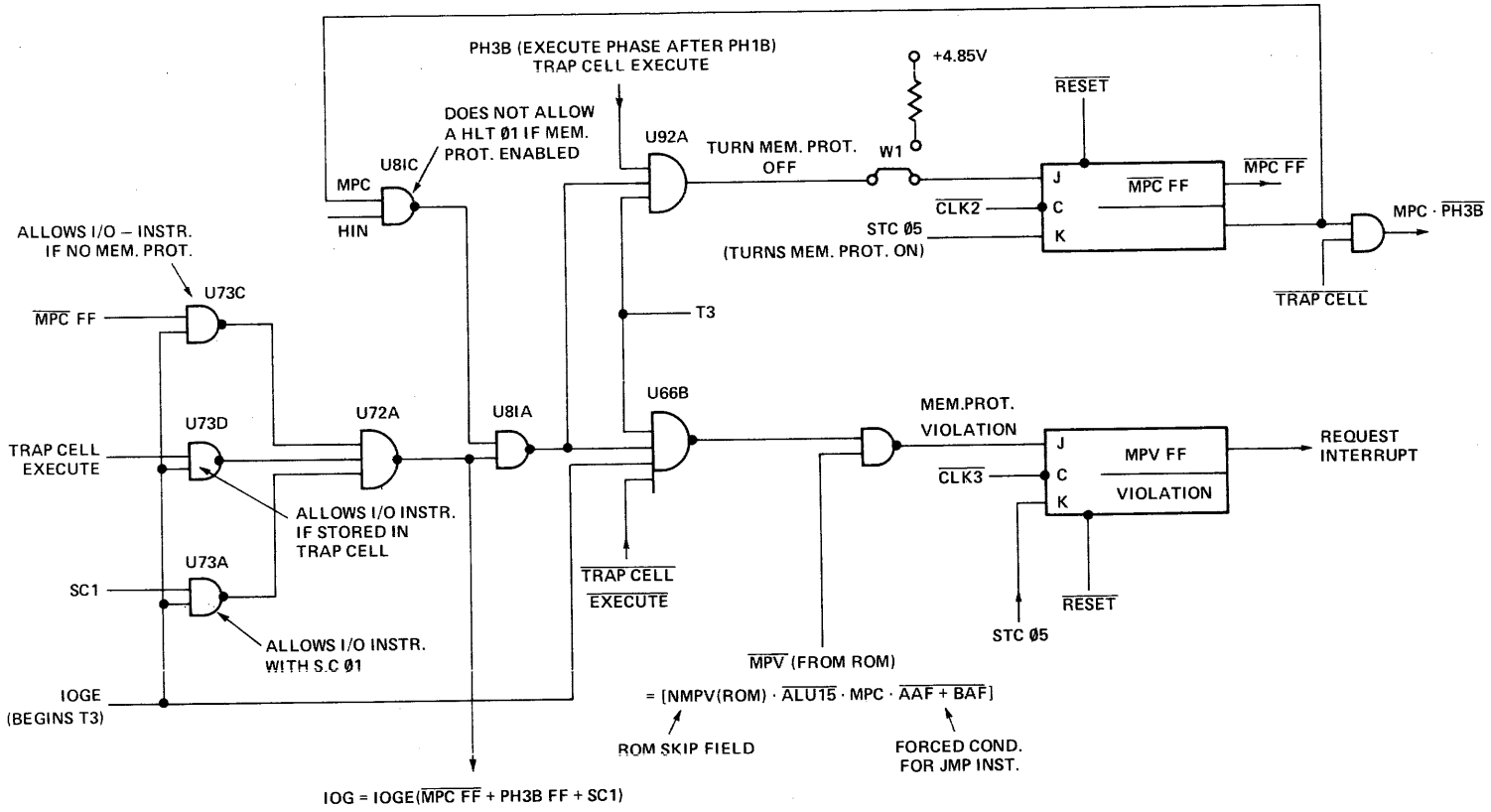
2100 I/O INTERRUPT SYSTEM TIMING



TIMING DIAGRAM FOR POWER FAIL W. AUTOM. RESTART



SIMPLIFIED MEMORY PROTECT



DIRECT MEMORY ACCESS

- DMA PROVIDES A DIRECT TRANSFER OF AN ENTIRE DATA BLOCK FROM MEMORY TO A PREASSIGNED PERIPHERAL (OUTPUT) OR PERIPHERAL TO MEMORY (INPUT).

- DMA ACCOMPLISHES A WORD TRANSFER BY STEALING A MEMORY CYCLE (IMPOSES A PH5) FROM THE CPU IF THE PERIPHERAL VIA ITS INTERFACE IS READY TO PARTICIPATE IN SUCH A TRANSFER.

- DMA INTERRUPTS ONLY AT THE END OF AN ENTIRE DATA BLOCK TRANSFER IF INTERRUPT IS ENABLED.

- THE TWO DMA-CHANNELS CARRY SC.Ø6 AND SC.Ø7.

- BEFORE A DMA TRANSFER TAKES PLACE IT HAS TO BE INITIALIZED BY A SPECIAL ROUTINE.

WHICH PARAMETERS HAVE TO BE SPECIFIED IN THE INITIALIZATION ROUTINE?

1. SELECT CODE OF PERIPHERAL TO BE SERVICED BY DMA.
2. SHALL WE HAVE A CLC ON THE DEVICE I/O CHANNEL AFTER THE LAST WORD OF THE BLOCK HAS BEEN TRANSFERRED?
3. SHALL WE HAVE A STC ON THE DEVICE I/O CHANNEL AFTER EACH WORD IN THE DATA BLOCK HAS BEEN TRANSFERRED (REINITIATE PERIPHERAL)?

→ DMA CONTROL WORD (CW)

4. MEMORY STARTING ADDRESS OF FIRST WORD TO BE TRANSFERRED.
5. DIRECTION OF DATA TRANSFER.

→ DMA ADDRESS WORD (AW)

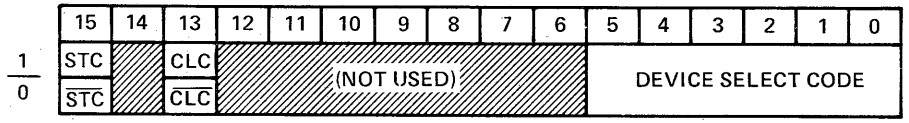
6. BLOCK LENGTH.

→ DMA BLOCK LENGTH WORD (BL)

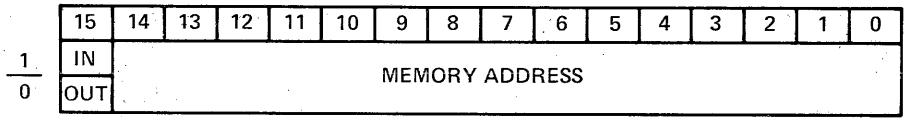
7. FOR RANDOM ACCESS MEMORIES (DISC/DRUM): DIRECTION, TRACK- AND SECTOR ADDRESS TO DISC CONTROL INTERFACE BOARD.

→ DISC/DRUM CONTROL WORD (DCW)

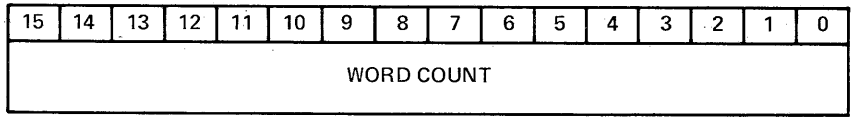
DMA INITIALIZATION PARAMETERS



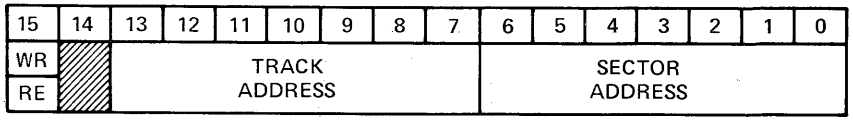
DMA CONTROL WORD (CW)



DMA ADDRESS WORD (AW)



DMA BLOCK LENGTH (BL)
(IN 2' COMPL. FORM)

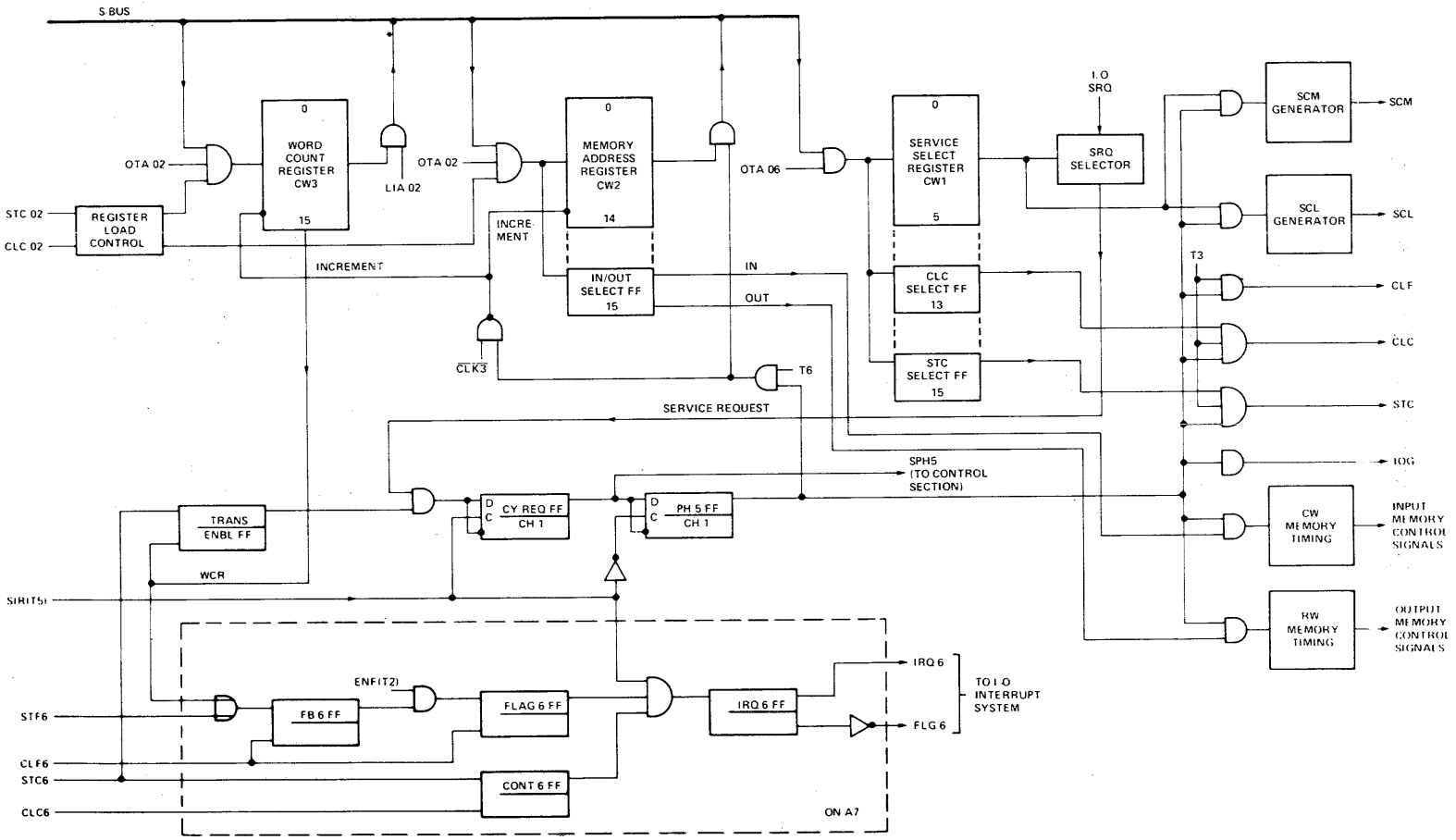


DISC/DRUM CONTROL WORD (DCW)

PROGRAM TO INITIALIZE DMA

LABEL	OPCODE	OPERAND	COMMENTS
INIT	LDA	CW	FETCHES CONTROL WORD (CW) FROM MEMORY AND LOADS IT IN A-REGISTER.
	OTA	6	OUTPUTS CW TO DMA CHANNEL 1.
	CLC	2	PREPARES MEMORY ADDRESS REGISTER TO RECEIVE ADDRESS WORD (AW).
	LDA	AW	FETCHES AW FROM MEMORY AND LOADS IT IN A-REGISTER.
	OTA	2	OUTPUTS AW TO DMA CHANNEL 1.
	STC	2	PREPARES WORD COUNT REGISTER TO RECEIVE BLOCK LENGTH WORD (BL).
	LDA	BL	FETCHES BL FROM MEMORY AND LOADS IT IN A-REGISTER.
	OTA	2	OUTPUTS BL TO DMA CHANNEL 1.
STRT	STC	10B,C	START INPUT DEVICE.
	STC	6B,C	ACTIVATE DMA CHANNEL 1.
	SFS	6	WAIT WHILE DATA TRANSFER TAKES PLACE OR, IF INTERRUPT PROCESSING IS USED, CONTINUE PROGRAM.
	JMP	*−1	
.	.	.	
.	.	.	
.	.	.	
	HLT		HALT
CW	OCT	120010	ASSIGNMENT FOR DMA CHANNEL 1; SPECIFIES I/O CHANNEL SELECT CODE ADDRESS (10_8), STC AFTER EACH WORD IS TRANSFERRED, AND CLC AFTER FINAL WORD IS TRANSFERRED.
AW	OCT	100200	MEMORY ADDRESS REGISTER CONTROL. DMA CHANNEL 1 SPECIFIES MEMORY INPUT OPERATION AND STARTING MEMORY ADDRESS (200_8).
BL	DEC	−50	WORD COUNT REGISTER CONTROL. DMA CHANNEL 1 (WCR1); SPECIFIES THE 2'S COMPLEMENT OF THE NUMBER OF CHARACTER WORDS IN THE BLOCK OF DATA TO BE TRANSFERRED (50_{10}).

SIMPLIFIED DMA LOGIC

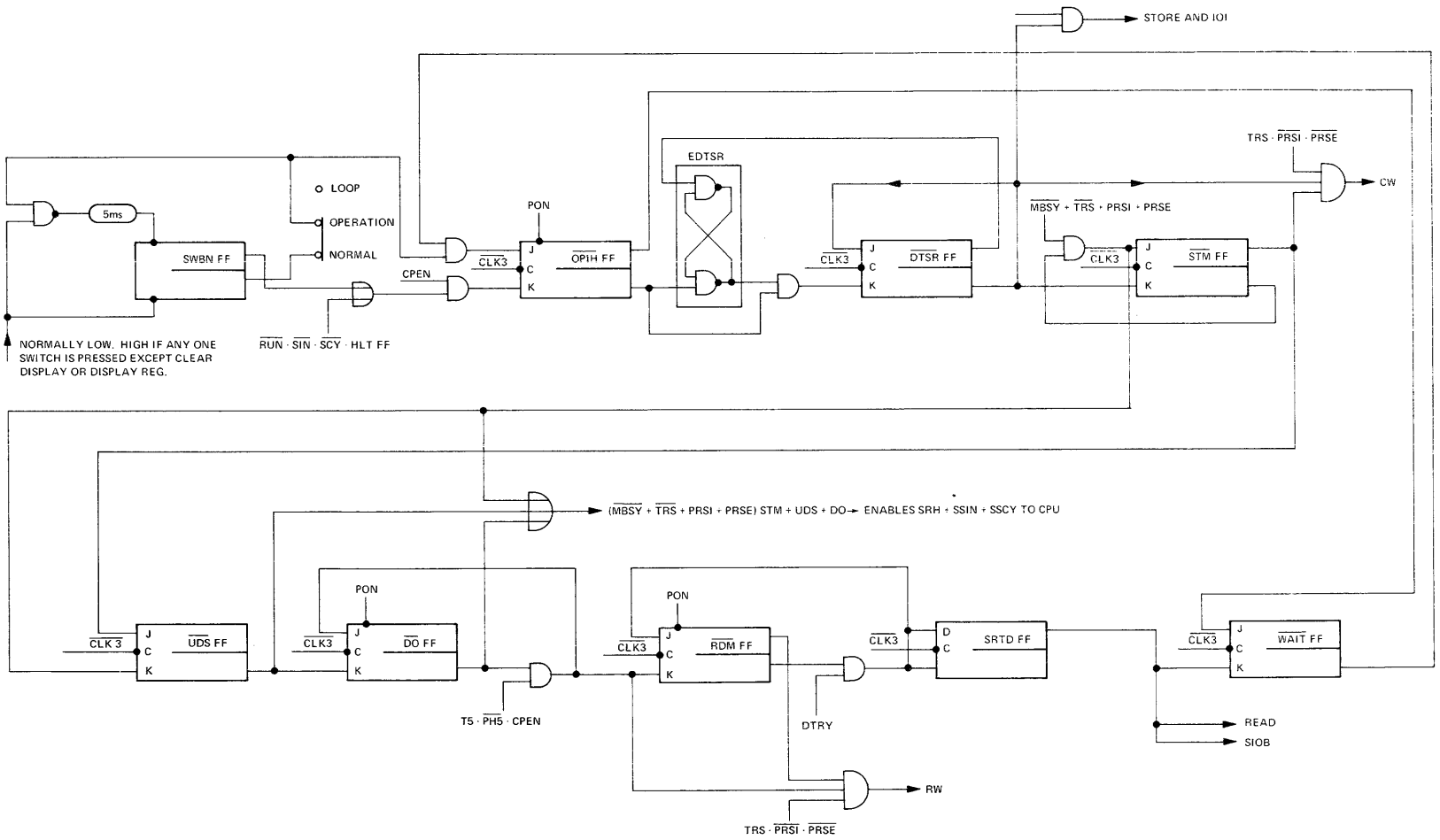


front panel 7

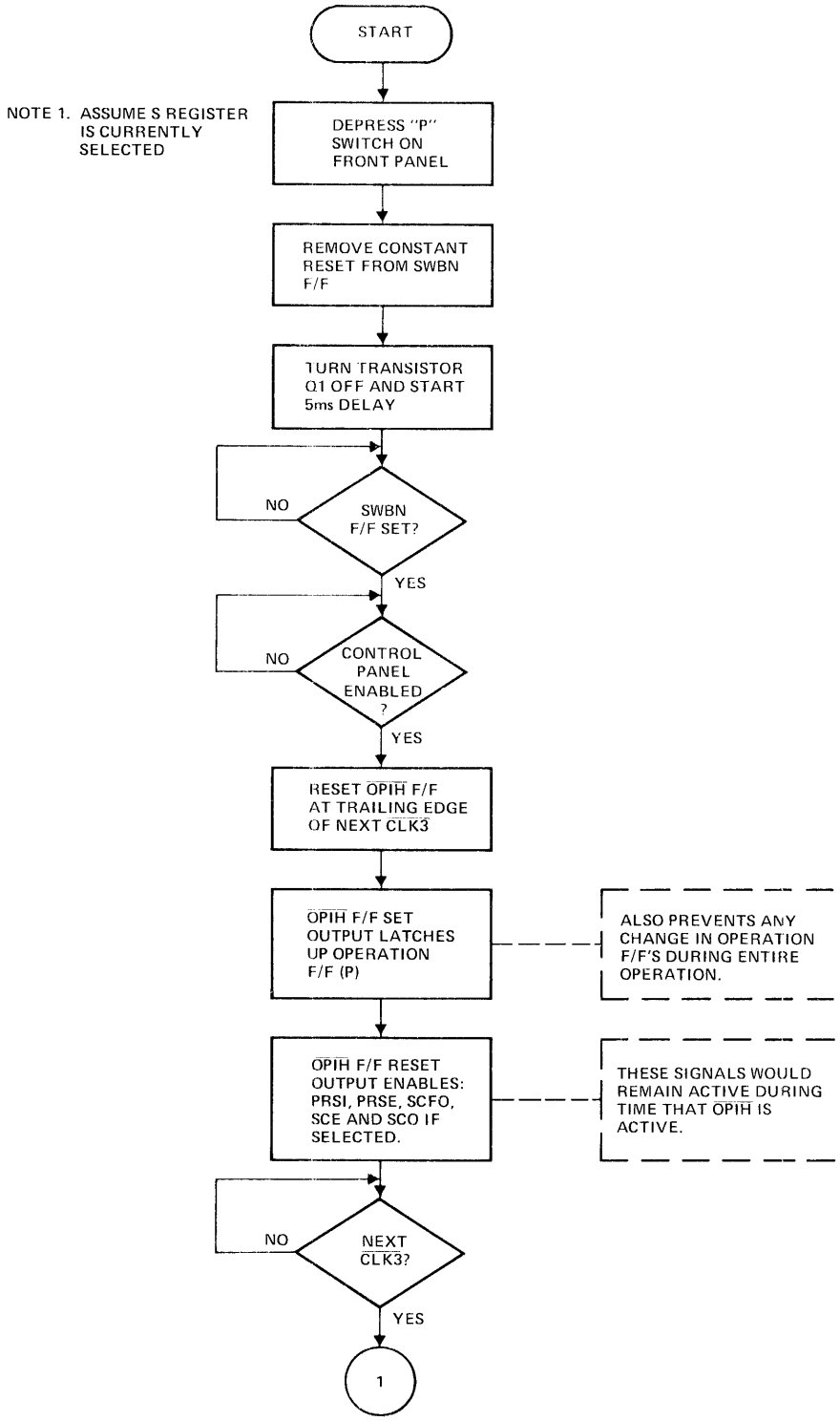
LESSON 7

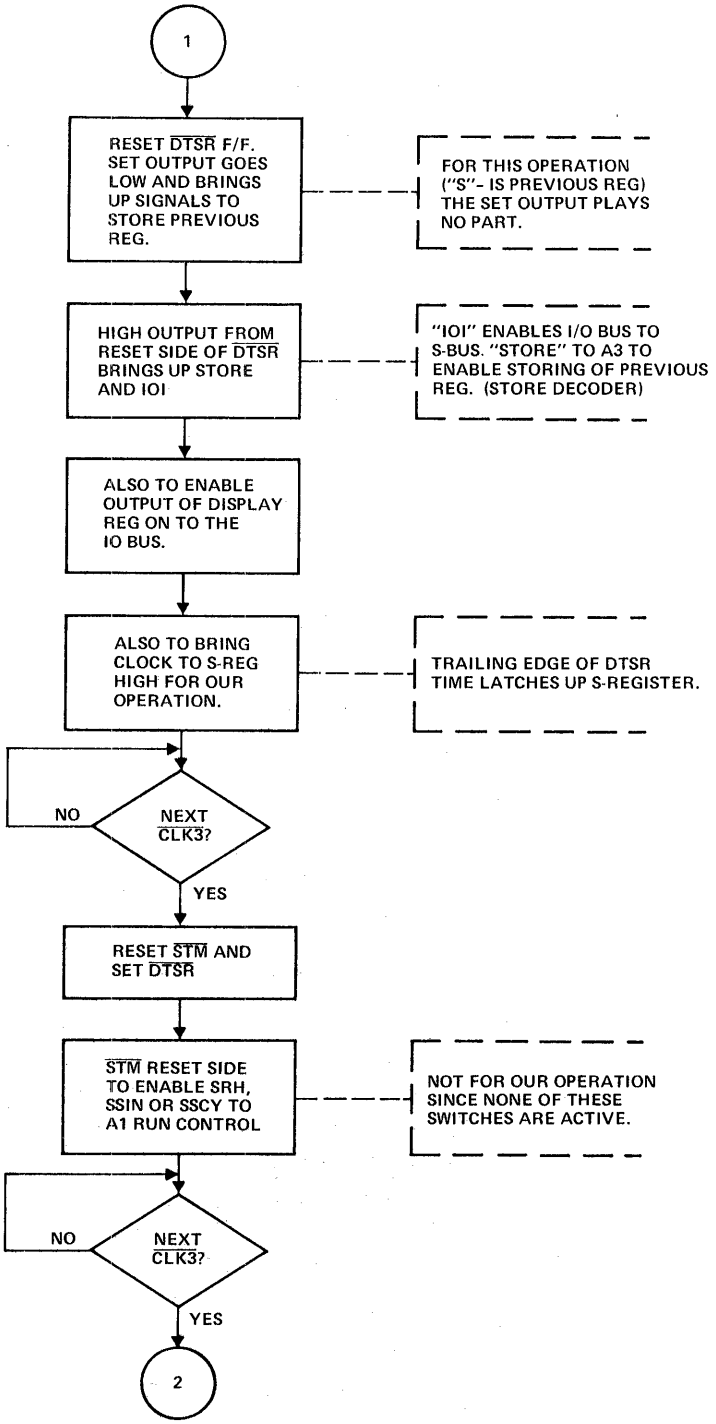
FRONT PANEL

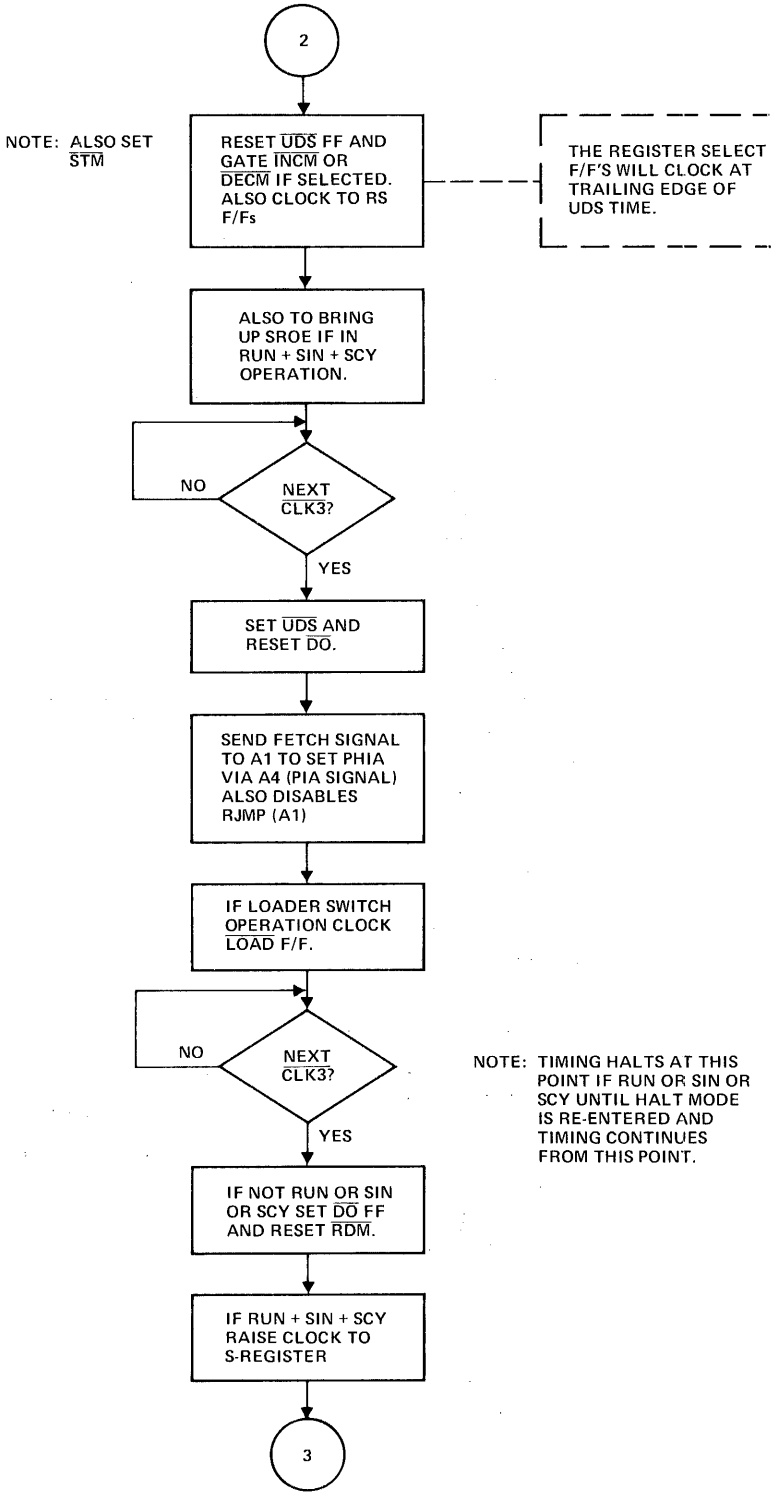
OPERATOR PANEL CONTROL LOGIC

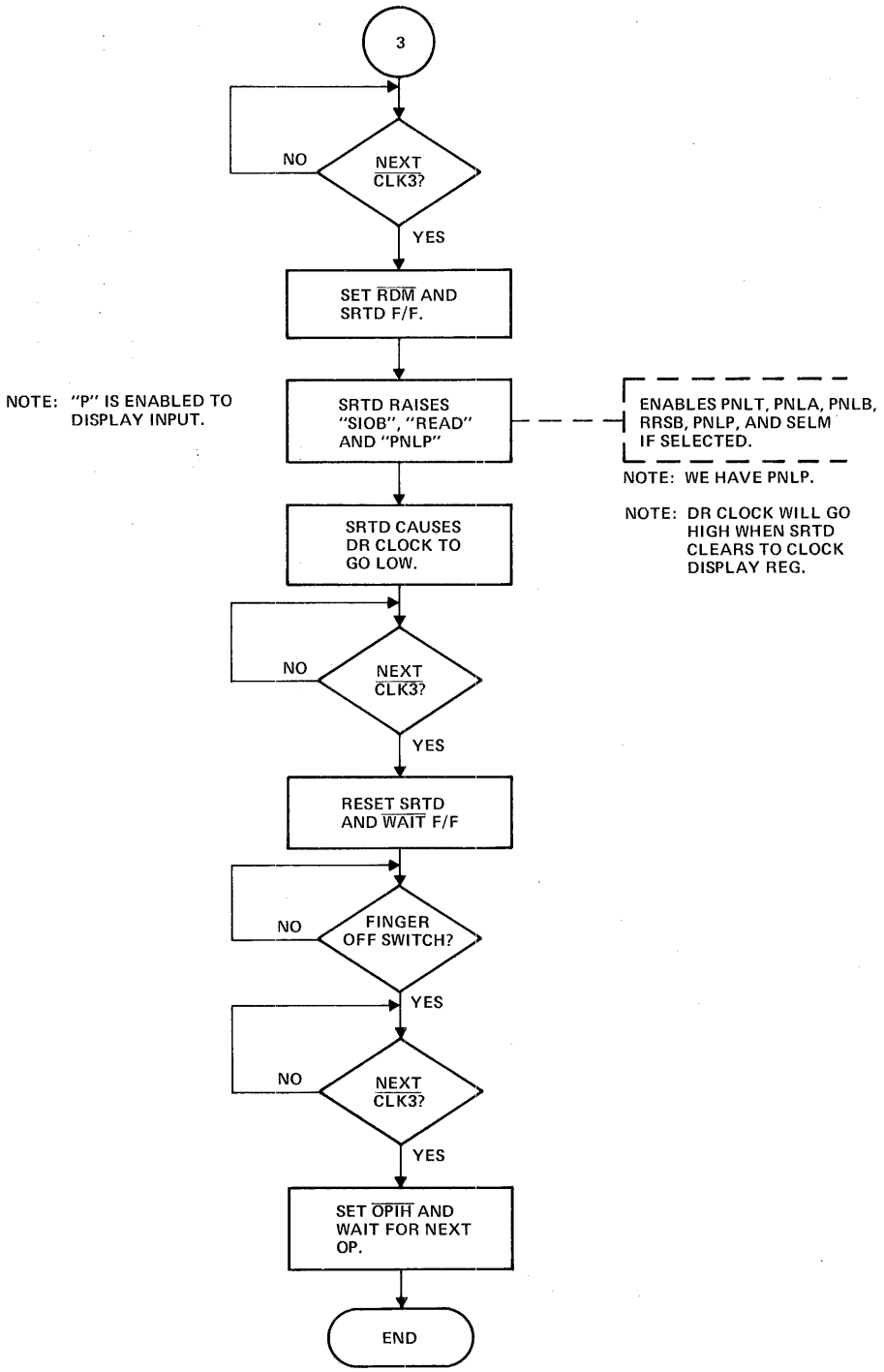


FRONT PANEL "P" OPERATION

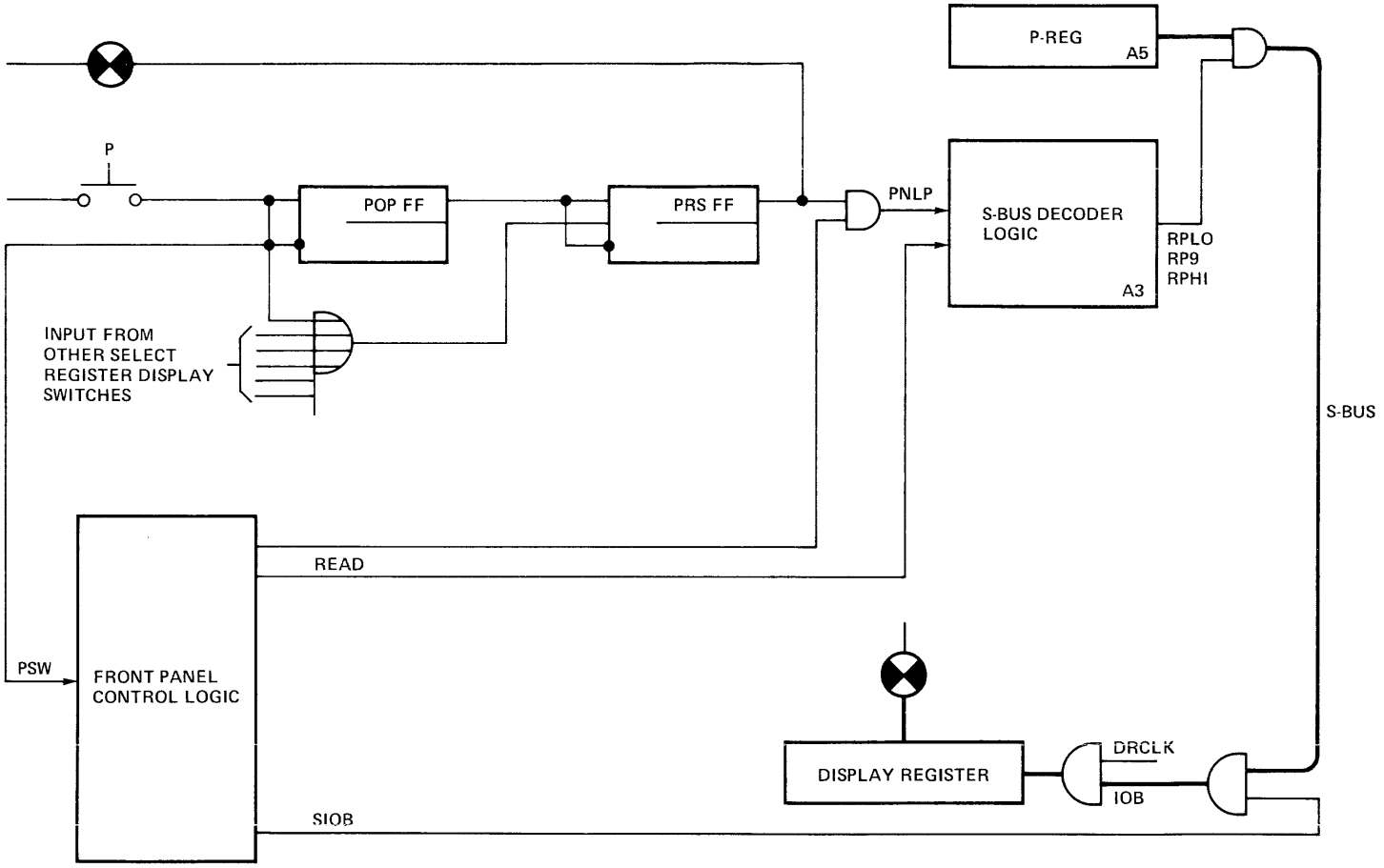



1
RESET DTSR F/F. SET OUTPUT GOES LOW AND BRINGS UP SIGNALS TO STORE PREVIOUS REG.
FOR THIS OPERATION ("S"- IS PREVIOUS REG) THE SET OUTPUT PLAYS NO PART.
HIGH OUTPUT FROM RESET SIDE OF DTSR BRINGS UP STORE AND IOI
"IOI" ENABLES I/O BUS TO S-BUS. "STORE" TO A3 TO ENABLE STORING OF PREVIOUS REG. (STORE DECODER)
ALSO TO ENABLE OUTPUT OF DISPLAY REG ON TO THE IO BUS.
ALSO TO BRING CLOCK TO S-REG HIGH FOR OUR OPERATION.
TRAILING EDGE OF DTSR TIME LATCHES UP S-REGISTER.
NO
NEXT CLK3?
YES
RESET STM AND SET DTSR
STM RESET SIDE TO ENABLE SRH, SSIN OR SSCY TO A1 RUN CONTROL
NOT FOR OUR OPERATION SINCE NONE OF THESE SWITCHES ARE ACTIVE.
NO
NEXT CLK3?
YES
2

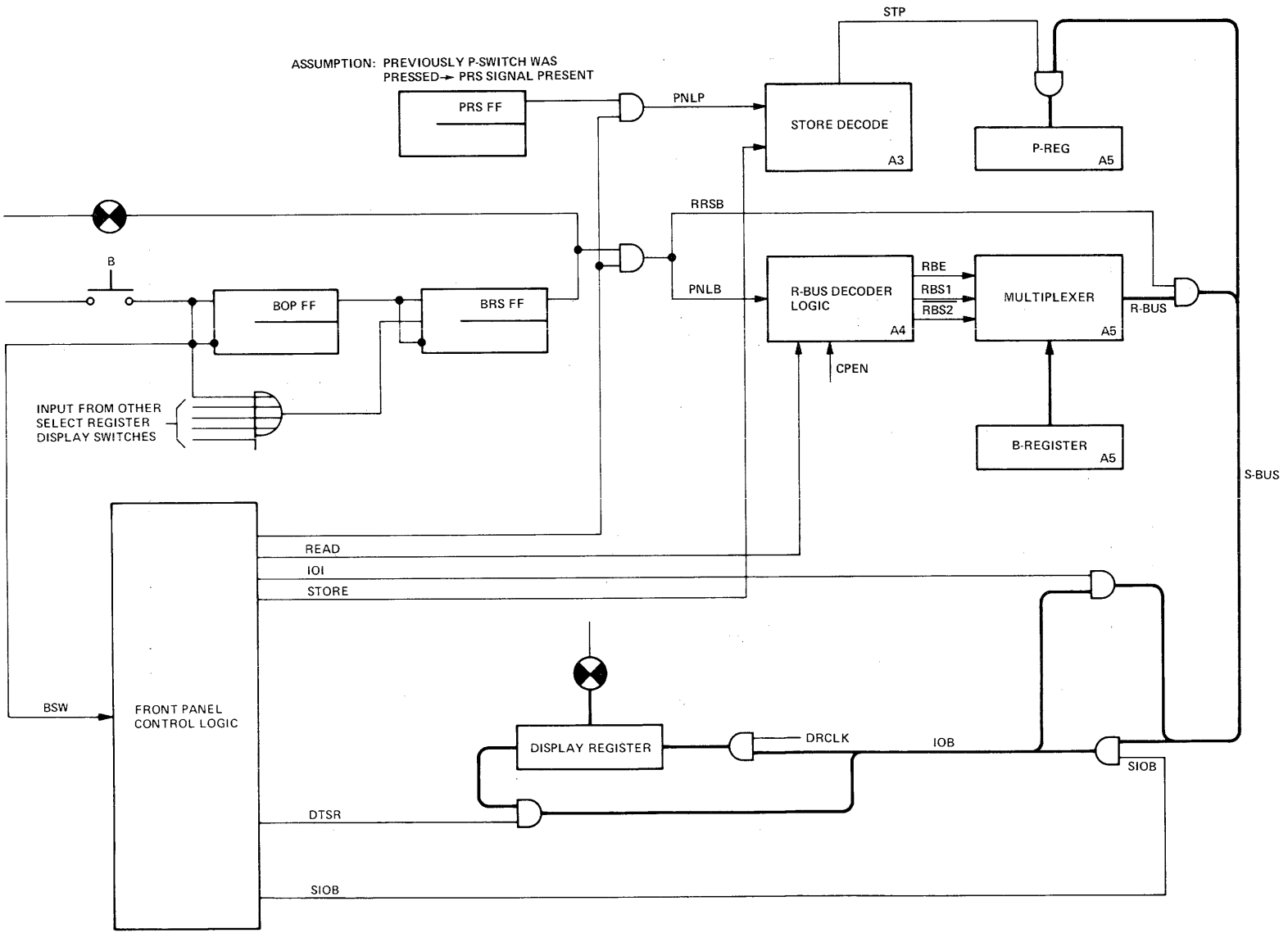

2
NOTE: ALSO SET STM
RESET UDS FF AND GATE INCM OR DECM IF SELECTED. ALSO CLOCK TO RS F/Fs
THE REGISTER SELECT F/F'S WILL CLOCK AT TRAILING EDGE OF UDS TIME.
ALSO TO BRING UP SROE IF IN RUN + SIN + SCY OPERATION.
NO
NEXT CLK3?
YES
SET UDS AND RESET DO.
SEND FETCH SIGNAL TO A1 TO SET PHIA VIA A4 (PIA SIGNAL) ALSO DISABLES RJMP (A1)
IF LOADER SWITCH OPERATION CLOCK LOAD F/F.
NO
NEXT CLK3?
YES
NOTE: TIMING HALTS AT THIS POINT IF RUN OR SIN OR SCY UNTIL HALT MODE IS RE-ENTERED AND TIMING CONTINUES FROM THIS POINT.
IF NOT RUN OR SIN OR SCY SET DO FF AND RESET RDM.
IF RUN + SIN + SCY RAISE CLOCK TO S-REGISTER
3


3
NEXT CLK3?
NO
YES
SET RDM AND SRTD F/F.
SRTD RAISES "SIOB", "READ" AND "PNLP"
ENABLES PNLT, PNLA, PNLB, RRSB, PNLP, AND SELM IF SELECTED.
NOTE: WE HAVE PNLP.
NOTE: "P" IS ENABLED TO DISPLAY INPUT.
SRTD CAUSES DR CLOCK TO GO LOW.
NOTE: DR CLOCK WILL GO HIGH WHEN SRTD CLEARS TO CLOCK DISPLAY REG.
NEXT CLK3?
NO
YES
RESET SRTD AND WAIT F/F
FINGER OFF SWITCH?
NO
YES
NEXT CLK3?
NO
YES
SET OPIH AND WAIT FOR NEXT OP.
END

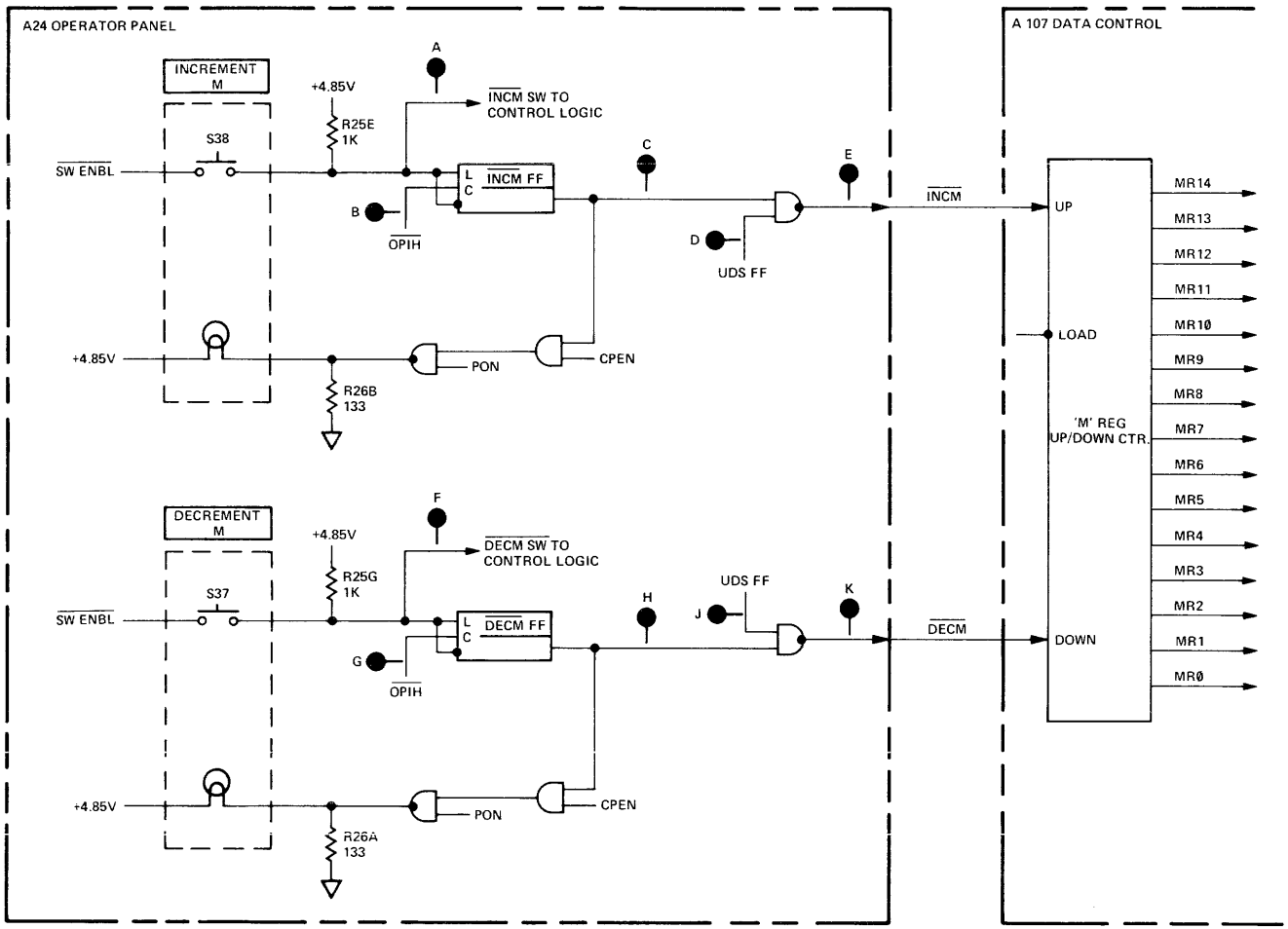
BLOCK DIAGRAM FOR DISPLAY P-REG.



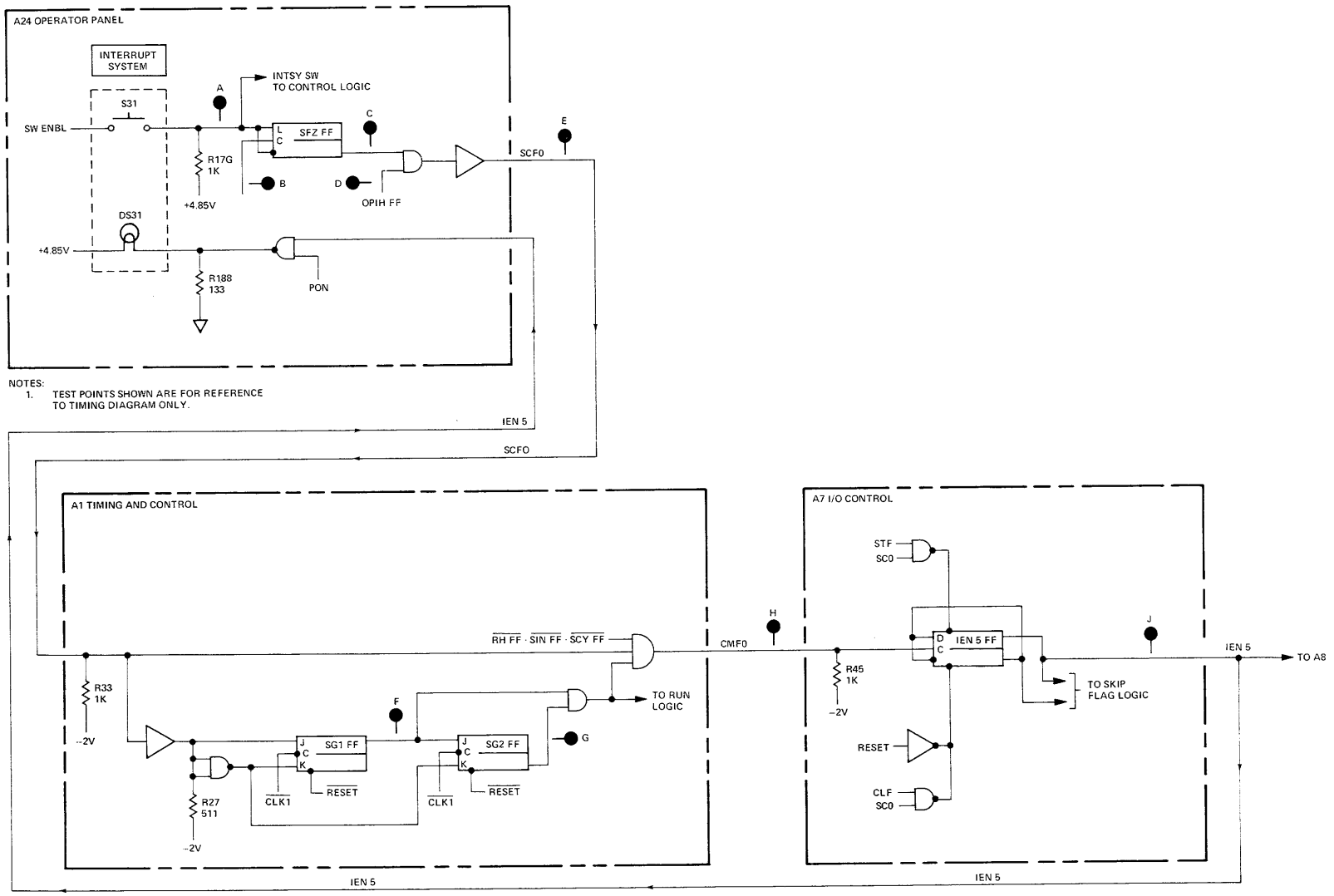
BLOCK DIAGRAM FOR DISPLAY B-REG.
(1. P-REG IN DISPLAY→ P-REG. 2. B-REG TRANSFERED TO DISPLAY REG.)



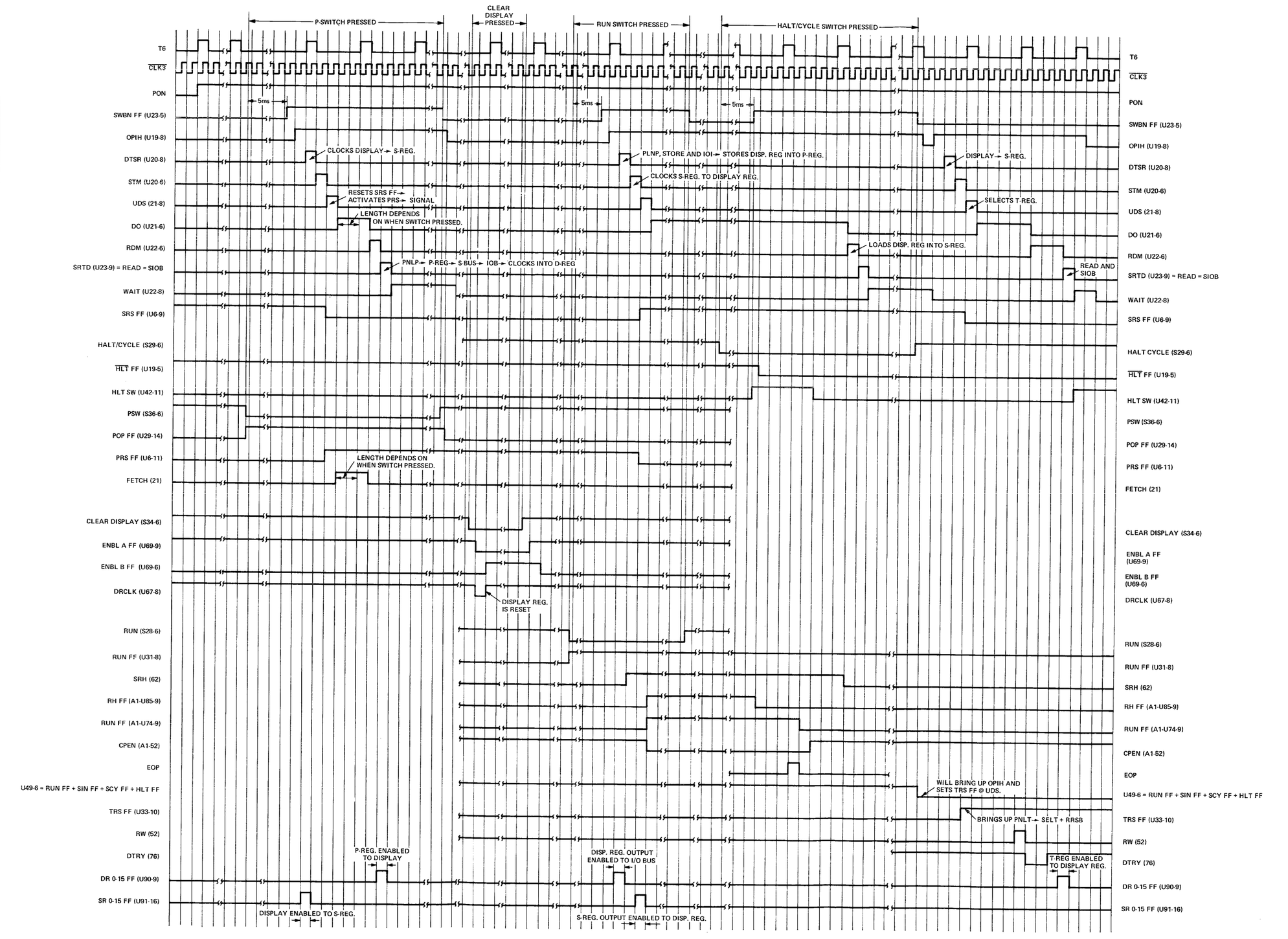
BLOCK DIAGRAM: INCREMENT/DECREMENT 'M' REGISTER SWITCHES
(OPERATOR'S PANEL)



SIMPLIFIED DIAGRAM OF 'INTERRUPT SYSTEM'
SWITCH ON OPERATOR'S PANEL



FRONT PANEL TIMING DIAGRAM



Printed in U.S.A.

HP 02100-90159