SDS TECHNICAL INFORMATION

MODEL 92 COMPUTER THEORY OF OPERATION MANUAL (PRELIMINARY)

SDS 900864A June 1965

SCIENTIFIC DATA SYSTEMS

MODEL 92 COMPUTER
THEORY OF OPERATION
MANUAL
(PRELIMINARY)

SDS 900864A June 1965

SCIENTIFIC DATA SYSTEMS/1649 Seventeenth Street/Santa Monica, California/UP 1-0960

Printed in U.S.A.

CONTENTS

CONTENTS (Continued)

PREFACE

Model 92 Computer is a high-speed, general purpose digital computer designed for real-time systems control, direct digital control, message switching, and repetitive, high-speed computation. The computer is completely modular, utilizing monolithic integrated circuits.

This preliminary manual describes the hardware logic and operation of the central processor, memory, and control console. Additional information on programming, logic, and circuits can be found in the following publications:

SDS 900505B	SDS 92 Computer Reference Manual
SDS 900925A	Model 92 Computer Logic Equations, Main Frame and Memory
SDS 900921A	Model 92 Computer General Reference Drawings
SDS 900922A	92 Computer Module Reference Data

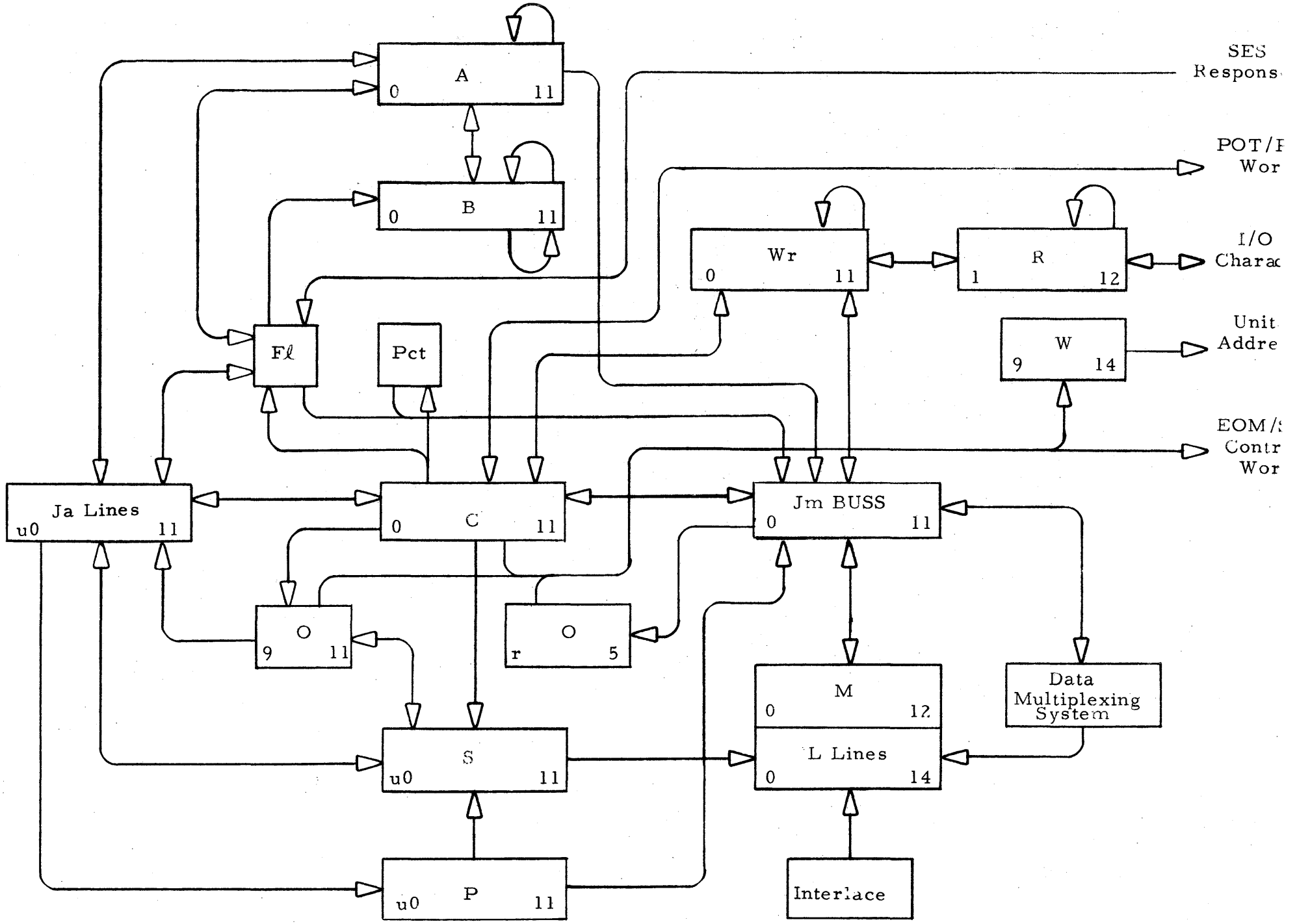
Model 92 Computer

CHAPTER 1

Introduction to the Hardware

1.1 Hardware characteristics

1.2 Hardware organization



BLOCK DIAGRAM

1.1 Hardware characteristics

All flip-flops used in the 92 main frame have the following hardware characteristics:

1. All flip-flops are the usual RS type. However, they also receive a common clock. All changes of state are made on the falling edge of this clock. [This means that the contents of two flip-flops may be directly interchanged; e.g.

$$sA1 = Axb\ B1$$
$$rA1 = Axb\ \overline{B1}$$

$$sB1 = Axb\ A1$$
$$rB1 = Axb\ \overline{A1}$$

When the gating term Axb is true, the falling edge of the clock will swap A1 and B1.]

2. If the set term and the reset term are both true, the flip-flop will set on the falling edge of the clock. [This means the example above may be simplified to:

$$sA1 = Axb\ B1$$
$$rA1 = Axb$$

$$sB1 = Axb\ A1$$
$$rB1 = Axb]$$

Logically, any flip-flop in the 92 main frame falls into one of two groups:

1. Standard RS type. These are recognizable in the equations as those flip-flops which have both a SET equation and a RESET equation; e.g.,

sA1 = Axb B1

rA1 = Axb

2. Repeater type. These flip-flops will automatically reset if there is no set input. However, these flip-flops will not set or reset unless they receive an enable signal. [This is accomplished by holding the reset true and using the enable to gate the common clock.] These flip-flops are recognizable in the equations as those which have a SET equation (only) and an ENABLE equation; e.g.,

sB1 = A1

Bg = Axb

Most of the logic is implemented via AND-OR-BUFFER or AND-OR-INVERTER hardware. The outstanding exceptions are the terms which enable the proper inputs to the adder. These have been implemented via NANDS because of speed considerations.

1.2 Hardware organization

A REGISTER

FUNCTION:

The A register is the index register. A also defines the block length for block I/O opcodes (50, 51, 54, 55). A may also be used as an auxillary accumulator.

IMPLEMENTATION:

12 repeater flip-flops designated A0, . . . , A11.

CONTROL TERMS:

Ag - enables the repeater flip-flops that make up A

Axas - shifts A left one binary bit position

Axb - interchanges A and B

Axja - gates Ja into A

Axjas - gates Ja, shifted right one binary bit position, into A

B REGISTER

FUNCTION:

The B register is the main accumulator.

IMPLEMENTATION:

12 repeater flip-flops designated B0,...,B11.

CONTROL TERMS:

Bg - enables the repeater flip-flops that make up B

Axb - interchanges B and A

Axjas - shifts B right one binary bit position

Bxbsℓ - shifts B left one binary bit position

C REGISTER

FUNCTION:

The C register acts as the main exchange register between memory (M) and both the internal logic and the input/output logic.

IMPLEMENTATION:

12 repeater flip-flops designated C0,...,C11.

CONTROL TERMS:

Cg - enables the repeater flip-flops that make up C

Cxi - gates a PIN input word into C

Cxja - gates Ja into C

Cxjm - gates Jm into C

Cxw - gates Wr into C

Fℓ

FUNCTION:

Fℓ holds arithmetic overflow information. Fℓ also holds the result (0 or 1) of a sense or compare instruction.

IMPLEMENTATION:

1 RS flip-flop designated Fℓ . However, the implementation is such that Fℓ appears to be a repeater flip-flop.

CONTROL TERMS:

Fℓg - enables the quasi-repeater flip-flop Fℓ.

Fℓs - is the set signal to the quasi-repeater flip-flop Fℓ.

Ja LINES

FUNCTION:

The Ja LINES are the outputs of the adder.

IMPLEMENTATION:

15 lines designated Jau0, Jau1, Jau2, Ja0,..., Ja11.

CONTROL TERMS:

Gpxa - gates A onto Ja

Gpxad - gates A + C onto Ja

Gpxam - gates A-1 onto Ja

Gpxc - gates C or C + 1 onto Ja

Gpxeo - gates A $\oplus$ C onto Ja

Gpxex - gates A $\wedge$ C onto Ja

Gpxs - gates S onto Ja

Gpxsi - gates S + 1 onto Ja

Gpxsu - gates A-C onto Ja

Gnnac - gates C-A onto Ja

Prbu - merges the control panel SET BUTTONS with the current contents of Ja

Jm BUSS

FUNCTION:

The Jm BUSS is the memory buss.

IMPLEMENTATION:

12 lines designated Jm0, ..., Jm11.

CONTROL TERMS:

Jmxa - gates A onto Jm

Jmxc - gates C onto Jm

Jmxp - gates P0-P11 onto Jm

Jmxpu - gates [Fℓ, Pct, 0, ..., 0, Pu0, Pu1, Pu2] onto Jm.

Jmxwr - gates Wr onto Jm

$\overline{\text{Mw}}$ - gates M onto Jm

Jmxz - gates the Data Multiplexing Systems input word onto Jm

L LINES

FUNCTION:

The L LINES contain the address of the current memory reference.

IMPLEMENTATION:

15 lines designated L0,...,L14.

CONTROL TERMS:

$\overline{Ts}$ - gates S onto L

Ts $\overline{Dmc}$ - gates the interlace's memory reference address onto L

Dmc - gates the Data Multiplexing System's memory reference address onto L

M REGISTER

FUNCTION:

The M register holds the contents of (read) or for (write) the currently referenced memory location.

IMPLEMENTATION:

13 flip-flops designated M0,...,M12.

M12 is the parity bit.

CONTROL TERMS:

Mw - gates Jm to M0 through M11 and even parity to M12.

O REGISTER

FUNCTION:

The O register holds the current opcode.

IMPLEMENTATION:

6 repeater flip-flops designated Or, O1,..., O5.

CONTROL TERMS:

Og - enables the repeater flip-flops that make up O

Tp End - gates the next opcode into O

O9, O10, O11

FUNCTION:

O9-O11 provide temporary storage for the most significant three bits of a 15-bit operand address. O9-O11 also hold part of an EOM/SES control word.

IMPLEMENTATION:

3 RS flip-flops designated O9, O10, O11.

CONTROL TERMS:

ϕ0 Tp Lp - gates Su0-Su2 into O9-O11.

ϕ0 Tp $\overline{\text{Lp}}$ - gates C9-C11 into O9-O11.

P REGISTER

FUNCTION:

The P register holds the address of the next instruction.

IMPLEMENTATION:

15 repeater flip-flops designated Pu0, Pu1, Pu2, P0, . . . , P11.

CONTROL TERMS:

Pg - enables the repeater flip-flops that make up P

Pxbu - gates the control panel set buttons into P

Pxja - gates Ja into P

Pxp - recirculates P

Pct

FUNCTION:

Pct gates the normal execution of the full-word I/O opcodes (10/50, 11/51, 14/54, 15/55). $\overline{\text{Pct}}$ causes these commands to trap.

IMPLEMENTATION:

1 RS flip-flop designated Pct.

R REGISTER

FUNCTION:

The R register receives/presents the input/output character from/to the connected peripheral.

IMPLEMENTATION:

12 repeater flip-flops designated R1,...,R12.

CONTROL TERMS:

Rg - enables the repeater flip-flops that make up R

W4 W9 - gates the output precessing (i.e. intershifting) of R and Wr

W4 $\overline{\text{W9}}$ - gates the input precessing (i.e. intershifting) of R and Wr

$\overline{\text{W5}}$ W6 $\overline{\text{W9}}$ - gates the merging of an input character into R

S REGISTER

FUNCTION:

The S register holds the address of the current memory reference by the main frame.

IMPLEMENTATION:

15 repeater flip-flops designated Su0, Su1, Su2, S0, . . . , S11.

CONTROL TERMS

Sg - enables the repeater flip-flops that make up S

Sxcl - gates C7-C11 into S7-S11

Sxcm - gates O9-O11 into Su0-Su2 and C0-C6 into S0-S6

Sxcs - gates 1 into S7 and C9-C11 into S8-S10

Sxja - gates Ja into S

Sxp - gates P into S

Int - gates the proper interrupt address into S (S2-S10)

Tr - gates the proper trap address into S (S5-S10)

W REGISTER

FUNCTION:

The W register holds the unit address of the currently connected peripheral.

IMPLEMENTATION:

5 repeater flip-flops designated W9, W11, . . . , W14

1 RS flip-flop designated W10

CONTROL TERMS:

Wg - enables the repeater flip-flops that make up W

Wc - clears W

Ws - gates C6-C11 into W

T1 $\overline{\text{Ta}}$ Wh - clears W

Wr REGISTER

FUNCTION:

The Wr register provides a one-computer-word buffer between the main frame logic (C) and the input/output character (R).

IMPLEMENTATION:

12 repeater flip-flops designated Wr0, . . . , Wr11.

CONTROL TERMS:

Wrg - enables the repeater flip-flops that make up Wr.

$\overline{\text{Iw}}$ Wx - gates C into Wr

W4 - gates the precessing (i.e. intershifting) of Wr and R

Wrxjm - gates Jm into Wr

CHAPTER 2

Introduction to Programming

2.1 Memory word format

2.2 Instruction word formats

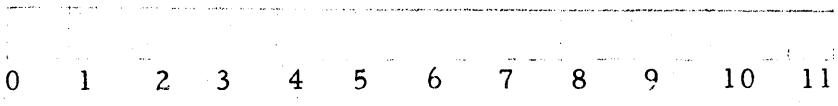
2.3 Operand word format

2.4 Description of opcodes

2.5 Memory allocation

2.1 Memory word format

The 92 computer word is 12 binary bits long.



The bits will be numbered from left to right as shown above.

2.2 Instruction word formats

The 92 allows 6 different modes of addressing. Some of these modes need only one computer word to define both the opcode and either the effective address (i.e. the address of the operand) or the indirect address (i.e. the address at which addressing is reinitiated). The remaining addressing modes require two contiguous computer words to define both the opcode and a computer address.

Addressing Type: Immediate

Instruction Length: One Word

Addressing Area: Next Location

Location	Computer Word
L	Opcode 1 0 0 0 0 0
	0 1 2 3 4 5 6 7 8 9 10 11
	Operand
L + 1	0 1 2 3 4 5 6 7 8 9 10 11
L + 2	Next Instruction

Comments: This addressing mode should not be used with the following opcodes

EXU

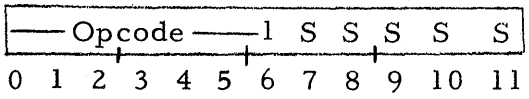
EMC

BRM

Addressing Type: Direct Scratch Pad

Instruction Length: One Word

Addressing Area: Scratch Pad (00001_8-00037_8)

Location	Computer Word
L	
L + 1	Next Instruction

Comments: The operand is taken from location

$\cdot 0000000000SSSSS_2$

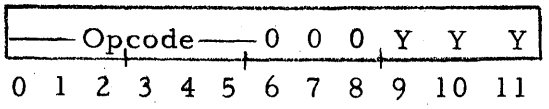
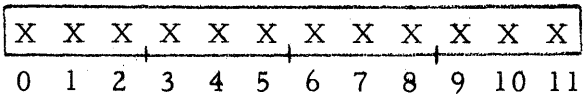
where

SSSSS $\neq$ 00000

Addressing Type: Full Direct

Instruction Length: Two Words

Addressing Area: Full Memory

Location	Computer Word
L	
L + 1	
L + 2	Next Instruction

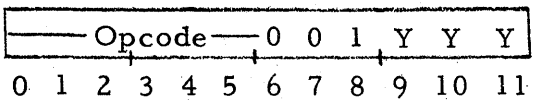
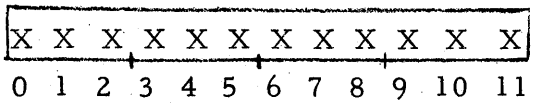
Comments: The operand is taken from location

$YYYXXXXXXXXXXXX_2$

Addressing Type: Index

Instruction Length: Two Words

Addressing Area: Full Memory

Location	Computer Word
L	
L + 1	
L + 2	Next Instruction

Comments: The operand is taken from location

$$EEEEEEEEEEEEEEE_2$$

where

$$YYY\ XXX\ XXX\ XXX\ XXX_2$$

$$-000\ AAA\ AAA\ AAA\ AAA_2$$

$$EEE\ EEE\ EEE\ EEE\ EEE_2$$

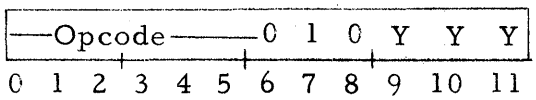
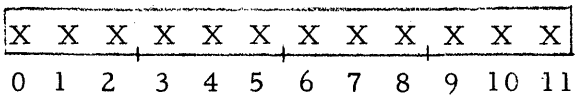
and the contents of the A register is given by

$$AAA\ AAA\ AAA\ AAA_2$$

Addressing Type: Full Indirect

Instruction Length: Two Words

Addressing Area: Full Memory

Location	Computer Word
L	
L + 1	
L + 2	Next Instruction

Comments: The opcode, as given in location L, is saved; but addressing is reinitiated at location

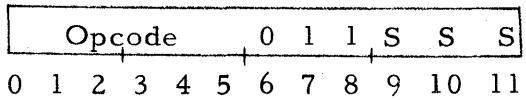
$$YYYXXXXXXXXXXXX_2.$$

Any mode of addressing may be specified at this indirect address.

Addressing Type: Indirect Scratch Pad

Instruction Length: One Word

Addressing Area: Upper, Even Scratch Pad (00020_8-00036_8)

Location	Computer Word
L	
L + 1	Next Instruction

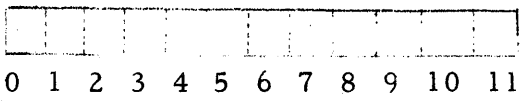
Comments: The opcode, as given in location L, is saved; but addressing is reinitialized at location

$$00000000001SSS0_2.$$

Any mode of addressing may be specified at this indirect address.

2.3 Operand word format

All operands are treated as 12-bit unsigned (i.e. positive) binary integers.



The most significant bit is bit 0. The least significant bit is bit 11.

2.4 Description of opcodes

The most significant 6 bits of (the first word of) an instruction specify the opcode. All opcodes will be written as 2-digit octal numbers.

Many pairs of opcodes perform the same function; only the accumulator referenced (A or B - as specified by the most significant bit of the opcode, Or) differs. In these cases the opcode pair will be discussed as one - with all references to an accumulator made by the ambiguous letter X.

The effective address will be denoted by E; the fifteen bits of the effective address will be numbered E0 through E14.

00-EOM

40-EOM

[Or, E] = 16-bit EOM control word

01-SES

41-SES

[Or, E] = 16-bit SES control word

no response $\Rightarrow$ clear Fℓ

response $\Rightarrow$ set Fℓ

02-CYB

42-CYA

E10 = 0 (c.f. CYD)

(X) is left, circular shifted $\overline{E11\text{-}E14}$ bit positions

02-CYD

42-CYD

E10 = 1 (c.f. CYX)

(A, B) is left, circular shifted $\overline{E11\text{-}E14}$ bit positions

03-CFB

43-CFA

E10 = 0 (c.f. CFI and CFD)

(Fℓ, X) is left, circular shifted $\overline{E11\text{-}E14}$ bit positions

03-CFI

E10 = 1 (c.f. CFB)

(Fℓ, B, A) is left, circular shifted $\overline{E11\text{-}E14}$ bit positions

43-CFD

E10 = 1 (c.f. CFA)

(Fℓ, A, B) is left, circular shifted $\overline{E11\text{-}E14}$ bit positions

04-STB

44-STA

(X)→(E)

05-COB

45-COA

(X) $\wedge$ (E) = 1 anywhere ⇒clear Fℓ

(X) $\wedge$ (E) = 0 everywhere ⇒set Fℓ

06-CEB

46-CEA

(X) = (E) ⇒clear F

(X) ≠ (E) ⇒set F

07-CMB

47-CMA

(X) < (E) ⇒clear F

(X) ≥ (E) ⇒set F

10-POT

(E) = POT output word

50-BPO

(E) = POT output word

(E + (A)) = POT output word

11-WOT

(E)→Wr

51-ROT

(E) →Wr

(E + (A))→Wr

12-DVB

(B, A) ÷ (E)→B

remainder→A

52-DVA

(A, B) ÷ (E)→B

remainder→A

13-MUA

53-MUB

(X) x (E)→(A, B)

14-PIN

PIN input word→(E)

54-BPI

PIN input word→(E)

PIN input word→(E + (A))

15-WIN

Wr→(E)

55-RIN

Wr→(E)

Wr→(E + (A))

16-MPO

(E) + 1→(E)

no carry⇒clear Fℓ

carry⇒set Fℓ

56-MPF

(E) + (Fℓ)→(E)

no carry⇒clear Fℓ

carry⇒set Fℓ

17-XMF

$(E)_0 \longleftrightarrow F\ell$

57-LDF

$(E)_o \rightarrow F\ell$

20-SUB

60-SUA

$(X)-(E)\rightarrow(X)$

no borrow $\Rightarrow$ clear $F\ell$

borrow $\Rightarrow$ set $F\ell$

21-SCB

61-SCA

$(X)-(E)-(F\ell)\rightarrow(X)$

no borrow $\Rightarrow$ clear $F\ell$

borrow $\Rightarrow$ set $F\ell$

22-ADB

62-ADA

$(X) + (E)\rightarrow(X)$

no carry $\Rightarrow$ clear $F\ell$

carry $\Rightarrow$ set $F\ell$

23-ACB

63-ACA

$(X) + (E)+(F\ell)\rightarrow(X)$

no carry $\Rightarrow$ clear $F\ell$

carry $\Rightarrow$ set $F\ell$

24-LDB

64-LDA

$(E)\rightarrow(X)$

25-ANB

65-ANA

(E) $\wedge$ (X)$\rightarrow$(X)

26-EOB

66-EOA

(E) $\oplus$ (X)$\rightarrow$(X)

27-ORB

67-ORA

(E) $\vee$ (X)$\rightarrow$(X)

30-BAX

(A)$\longleftrightarrow$(B)

Take next instruction from E

70-BDA

(A)-1$\rightarrow$(A)

(A) = 7777_8 $\Rightarrow$Take next instruction in sequence

(A) $\neq$ 7777_8 $\Rightarrow$Take next instruction from E

31-BFF

(Fℓ) = 1 $\Rightarrow$Take next instruction in sequence

(Fℓ) = 0 $\Rightarrow$Take next instruction from E

71-BFT

(Fℓ) = 0 $\Rightarrow$Take next instruction in sequence

(Fℓ) = 1 $\Rightarrow$Take next instruction from E

32-BRC

Load Fℓ

Load Pct

Clear currently active interrupt level of highest priority

Take next instruction from E

72-EXU

Execute the instruction at E

33-BRL

Load Fℓ

Load Pct

Take next instruction from E

73-BRU

Take next instruction from E

34-XMB

74-XMA

(X)$\leftrightarrow$(E)

35-MAB

75-MAA

(X) $\wedge$ (E)$\rightarrow$(E)

36-MPB

76-MPA

(X) + (E)$\rightarrow$(E)

no carry $\Rightarrow$clear Fℓ

carry $\Rightarrow$set Fℓ

37-BMC

[Fℓ, Pct, 0,...,0, Pu0, Pu1, Pu2]→(E)

[P0, P1, P2,..., P8, P9, P10, P11]→(E + 1)

Clear Fℓ

Set Pct

Take next instruction from E + 2

77-BRM

[Fℓ, Pct, 0,..., 0, Pu0, Pu1, Pu2]→(E)

[P0, P1, P2,..., P8, P9, P10, P11]→(E + 1)

Take next instruction from E + 2

2.5 Memory allocation

00000		Unassigned
00001	- 00037	Scratch Pad
00040	- 00077	Unassigned
00100	- 00117	(First four) DSC Interlace control word pairs
00120		Trap-12
00122		Trap-52
00124		Trap-13
00126		Trap-53
00130		Trap-10
00132		Trap-50
00134		Trap-11
00136		Trap-51
00140		Trap-14

	00142	Trap-54
	00144	Trap-15
	00146	Trap-55
Is	00150	Interrupt-power on (always armed)
	00152	Interrupt-power off (always armed)
	00154	Interrupt-main frame memory parity (armed via console switch)
	00156	Interrupt-Data Multiplexing System memory parity (armed via console switch)
	00160	Unassigned
	00162	Unassigned
	00164	Interrupt-clock sync (always armed)
	00166	Interrupt-clock pulse (arm furnished-Ij type)
	00170	Interrupt-I1 (arm furnished) } standard I/O channel
	00172	Interrupt-I2 (arm furnished) } standard I/O channel
Ir	00174	Unassigned
	00176	Unassigned
	00200-01176	System interrupts (up to 256 levels - any may be of Ij type if desired)

Ij ⟹ Single instruction interrupt

Ir ⟹ Interrupt system must be enabled before interrupt may go active

Is ⟹ Interrupt may always proceed from waiting to active

CHAPTER 3

Timing

3.1 Common clock

3.2 Clock counter

3.3 Phases

3.4 Cycle alternation

3.5 Summary

3.1 Common clock

All flip-flop changes of state occur on the falling edge of a common clock. This clock is derived from a 1.7143 megacycle crystal - making the clock period 583 nanoseconds. During one clock time (one period of the clock - measured from falling edge to falling edge) the clock will be symmetrically low (false) through the first half and high (true) through the last half.

3.2 Clock counter

One machine cycle is 1.75 microseconds. This means that there are exactly 3 clock times in each machine cycle. These clock times have been named

T1, T0, Tp

and three flip-flops have been used to logically distinguish these three clock times.

3.3 Phases

As a further aid in decoding the current state of the internal logic, eight phases

$\phi 0, \ldots, \phi 7$

have been defined by the binary count in three phase control flip-flops

F1, F2, F3.

These three phase control flip-flops change state only at Tp time (i.e. only on the trailing edge of the common clock which rises while Tp is true). Thus, to every machine cycle corresponds one of the eight possible phases.

3.4 Cycle alternation

An additional timing flip-flop

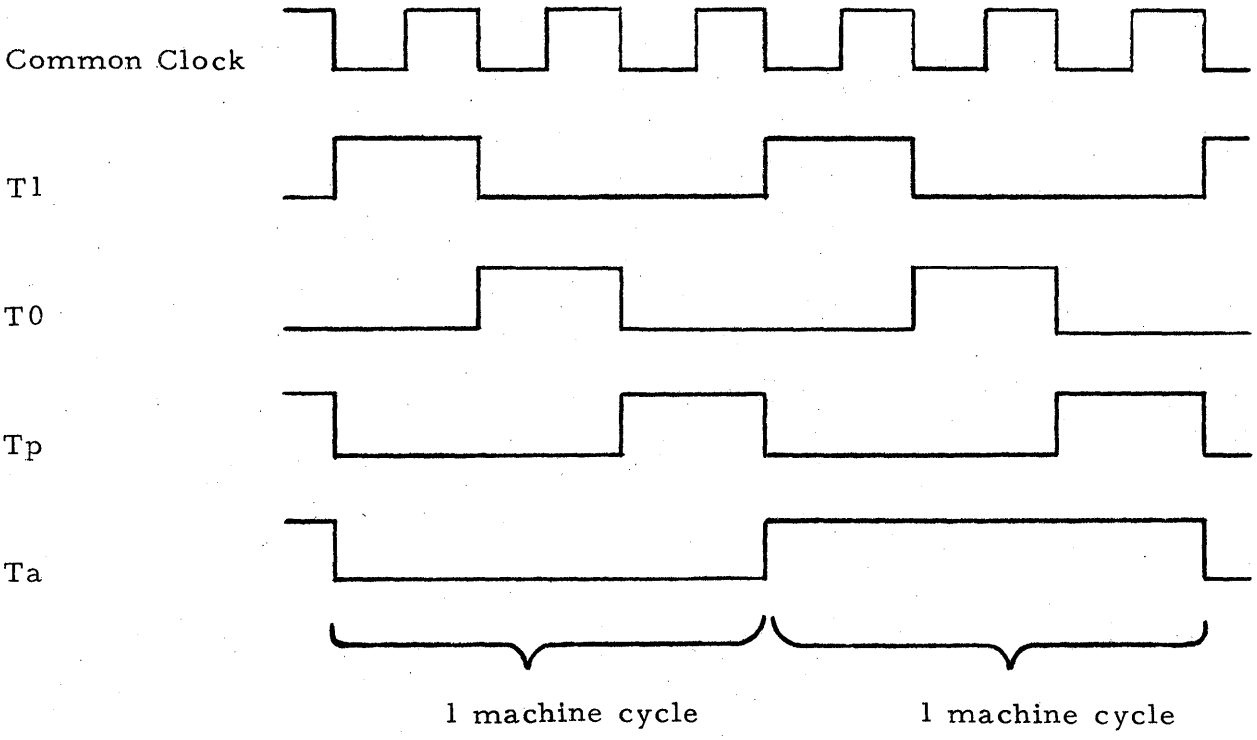
Ta

toggles at every Tp time. Ta essentially defines 3.5 microsecond

machine cycles (from T1 $\overline{Ta}$ through Tp Ta) which are used in parts of the I/O logic.

3.5 Summary

The contents of this chapter are epitomized by the following:



CHAPTER 4

Memory

4.1 Basic operation

4.2 Parity

4.3 Timeshare

4.1 Basic operation

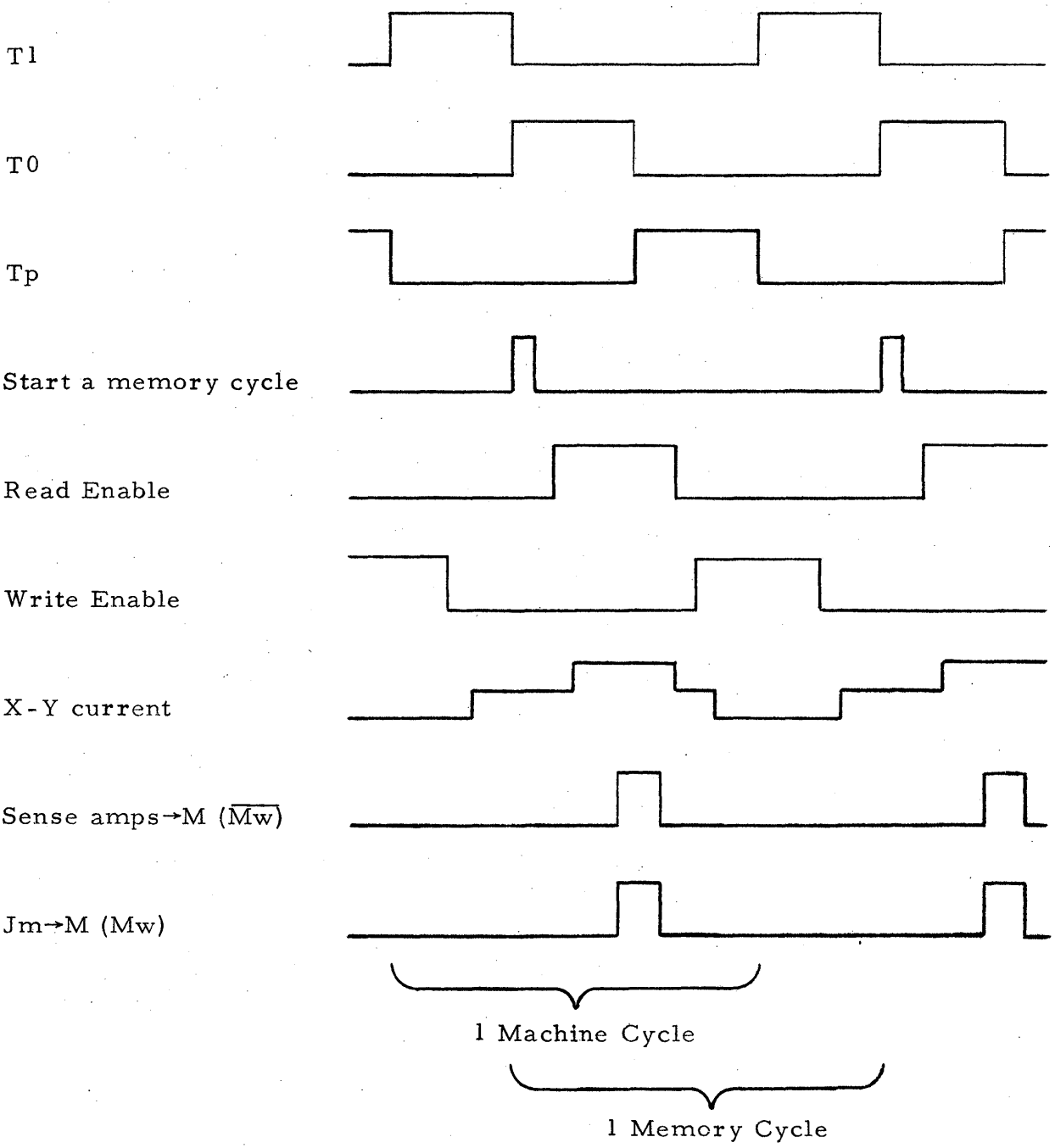
A memory cycle, like a machine cycle, takes 1.75 microseconds. However, unlike a machine cycle, a memory cycle starts at the beginning of T0. If the memory is doing a read cycle ($\overline{Mw}$), the data will be available (on Jm) at Tp. If the memory is doing a write cycle (Mw), the data should be presented to the memory (via Jm) from the start of T0 through Tp.

A memory cycle occurs as follows:

The main frame sends a signal T0m (which is just a copy of T0). On the leading edge of this signal the M register is cleared and Mgm (another signal from the main frame) is inspected. If Mgm is false, nothing further happens. If Mgm is true, a memory cycle is initiated. This memory cycle will address the location given by L (the memory address lines); thus L (and hence S) must be stable from the start of T0 through T1 whenever a memory reference is made. If the memory cycle is a read cycle - signalled by $\overline{Mw}$ (another signal from the main frame) - the memory logic will read the data from memory into M and then place M on the bi-directional memory buss, Jm. If the memory cycle is a write cycle - signalled by Mw - the memory logic will read the data from Jm into M and then write the data word into memory from M.

The memory logic also provides a signal, Tem, which indicates that the temperature of the memory stack is above some certain operating minimum.

A basic memory cycle is epitomized by the following:



4.2 Parity

If the memory parity option is installed, the memory logic will determine the parity of the Jm buss. During a write cycle; if the parity of Jm (i.e. the word to be written into memory) is odd, M12 will be set. This will write a one into the parity bit of the memory word. During a read cycle; if the parity of Jm (i.e. the word read from memory) is unequal to M12

(i.e. the parity bit read from memory), a signal (viz. $\overline{\text{Jme}}$) will be sent from the memory logic to the main frame logic. This signal denotes a parity error.

$\overline{\text{Jme}}$ is not gated by $\overline{\text{Mw}}$. Furthermore, every word read from memory should not be parity checked (e.g. the word at the shift count address). Therefore, the main frame logic must look at $\overline{\text{Jme}}$ only when the parity of a memory read cycle is to be checked. This is effected by the parity enable flip-flop, Cpe. Cpe will be set at Tp time when parity is to be checked. During T1 time, Cpe will gate $\overline{\text{Jme}}$. Cpe will always be reset at the end of T1 time.

When a parity error is recognized the affect depends upon a 3-position console switch:

1. HALT — Go immediately (T1 time) to idle (ϕ1) and remain interlocked until the parity error indication (O10) is cleared - by either the RESET button or the PARITY CONTINUE switch.
2. CONTINUE — The parity error is ignored and the program continues. Any parity error indication is cleared.
3. INTERRUPT — The program continues. However, one of two possible interrupt signals
    1) Cp $\overline{\text{Dmc}}$ Kpi (the parity incorrect read was made under the control of the main frame or standard I/O channel interlace)
    2) Cp Dmc Kpi (the parity incorrect read was made under control of the Data Multiplexing System)

    is sent to the interrupt logic.

Following is a list of the memory references during which parity is not checked:

1. During a write cycle
2. During idle (ϕ1) unless the console function INCREMENT P, MEMORY OUT, STEP, or RUN is being performed.
3. When accessing the effective address of an EOM (00/40), SES (01/41), or SHIFT (02/42, 03/43) instruction
4. When accessing the word following the last word in a defined output block (10/50, 11/51)
5. When accessing sequential scratch pad locations 00001 through 00013 while executing DVX (12/52)
6. When reaccessing the multiplicand while executing a MUX (13/53)
7. When accessing the instruction at the branch-to address but the branch is not taken (70, 31/71)
8. When accessing the instruction at the branch-to address of a BRC (32) for the first (ϕ0 Lp) or second (ϕ4) time.

4.3 Timeshare

Although the 92 main frame is a constant user of the memory, it is possible for other sources to have direct access to the memory. When another source wishes to Timeshare the memory, processing in the main frame is halted. The main frame will resume its operations only when the memory is again available for its use.

The 92 main frame is able to Timeshare memory with two other controllers:

1. Standard I/O Channel Interlace
2. Data Multiplexing System

A request for a Timeshare

Tsq

is sampled at T1 time. Tsq must be stable during this clock time. Tsq will cause the Timeshare flip-flop

Ts

to set. Ts will remain set for the duration of the timeshared memory cycle (from the start of T0 through T1). Ts will block most of the processing in the main frame by blocking the various phase (ϕ) signals. Other operations are blocked directly by Ts. Any double-cycle I/O operation (see Ta) must continue to conclusion-even though a Timeshare occurs during the second machine cycle of the operation.

The Timeshare user controls his memory cycle via

1. L (the memory address lines)
2. Mw (write/read cycle)
3. Jm (write/read data word)

The 92 main frame will monitor the parity of all read cycles ($\overline{Mw}$) and take appropriate action (as described above) in case of a parity error.

Since there must be some way of distinguishing between the two Timeshare users, the Data Multiplexing System must bring up a signal

Dmc

whenever it has control of memory (from the start of T0 through T1). Then

Dmc $\Rightarrow$ Data Multiplexing System Timeshare

Ts $\overline{Dmc}$ $\Rightarrow$ Standard I/O Channel Interlace Timeshare

$\overline{Ts}$ $\Rightarrow$ No Timeshare (the 92 main frame has control of the memory)

From the above it may be concluded that:

1. A Timeshare request always takes precedence over the main frame's use of memory
2. A Data Mulitplexing System's Timeshare request always takes precedence over a Standard I/O Channel Interlace's Timeshare request.

CHAPTER 5

Adder

5.1 Introduction

5.2 Operations

5.1 Introduction

The adder has fifteen output stages

Jau0, Jau1, Jau2, Ja0, . . . , Ja11

The most significant 3 bits (Jau0, Jau1, and Jau2) are only used in three of the adder's multitude of operations

1. S or S + 1
2. Indexing
3. Zero

and are hence formed directly. The remainder of this introduction will be concerned with the least significant 12 bits (Ja0, . . . , Ja11).

We begin with a few definitions:

1. Gn0, . . . , Gn11 - the "generate carry" term for a given stage of the adder
2. Pr0, . . . , Pr11 - the "propagate carry" term for a given stage of the adder
3. K0, . . . , K11 - the "carry" into a given stage of the adder
4. Ku2 - the "carry" out of the most significant stage of the (12 bit) adder

The logic may directly control:

1. the general form of Gn:

$$Gn = g_1 AC + g_2 A\overline{C} + g_3 \overline{A}C$$

2. the general form of Pr:

$$Pr = g_4 AC + g_5 A\overline{C} + g_6 \overline{A}C + g_7 \overline{A}\,\overline{C} + g_8 S$$

3. K11

where the g (j) are gating terms.

The logic has no direct control over:

1. $Ku2 = Gn0 + Pr0\ K0$

2. $K(j) = Gn(j+1) + Pr(j+1)\ K(j+1)\ ;\ j = 0,1\ldots,10$

3. $Ja(j) = Pr(j) \oplus K(j)\ ;\ j = 0,1,\ldots,11$

5.2 Operations

The various operations of the adder may now be described:

1. Addition (A + C)

   $Gn = AC$

   $Pr = A\overline{C} + \overline{A}C$

   $K11 = 0$

   Ja = answer

   $Ku2 \Longrightarrow$ carry out

2. Addition with carry (A + C + 1)

   $Gn = AC$

   $Pr = A\overline{C} + \overline{A}C$

   $K11 = 1$

   Ja = answer

   $Ku2 \Rightarrow$ carry out

3. Subtraction (A-C)

   $Gn = A\overline{C}$

   $Pr = AC + \overline{A}\overline{C}$

   $K11 = 1$

   Ja = answer

   $\overline{Ku2} \Longrightarrow$ borrow out

4. Subtraction with borrow (A-C-1)

   $Gn = A\overline{C}$

Pr = $AC + \overline{A}\,\overline{C}$

Kll = 0

Ja = answer

$\overline{Ku2} \Longrightarrow$ borrow out

5. Indexing (C-A)

Gn = $\overline{A}C$

Pr = $AC + \overline{A}\,\overline{C}$

Kll = 1

Ja = answer

Ku2 $\Longrightarrow$ (Jau0-Jau2) = (O9-O11)

$\overline{Ku2}$ $\Longrightarrow$ (Jau0-Jau2) = (O9-O11) minus 1

6. A

Gn = 0

Pr = $AC + A\overline{C} = A$

Kll = 0

Ja = answer

7. A-1

Gn = $AC + A\overline{C} = A$

Pr = $\overline{A}C + \overline{A}\,\overline{C} = \overline{A}$

Kll = 0

Ja = answer

$\overline{Ku2} \Longrightarrow Ja = 7777_8$

8. C

Gn = 0

Pr = $AC + \overline{A}C = C$

Kll = 0

Ja = answer

9. C + 1

Gn = 0

Pr = $AC + \overline{A}C = C$

Kll = 1

Ja = answer

Ku2 $\Longrightarrow$ Ja = 0000_8

10. S

Gn = 0

Pr = S

Kll = 0

Ja = answer

(Jau0-Jau2) = (Su0-Su2)

11. S + 1

Gn = 0

Pr = S

Kll = 1

Ja = answer

$\overline{Ku2}$ $\Longrightarrow$ (Jau0-Jau2) = (Su0-Su2)

Ku2 $\Longrightarrow$ (Jau0-Jau2) = (Su0-Su2) plus 1

12. Extract (AC)

Gn = 0

Pr = AC

Kll = 0

Ja = answer

13. Exclusive or ($A \oplus C$)

Gn = 0

Pr = $A\overline{C} + \overline{A}C = A \oplus C$

Kll = 0

Ja = answer

14. Inclusive or ($A \vee C$)

Gn = 0

Pr = $AC + A\overline{C} + \overline{A}C = A + C$

Kll = 0

Ja = answer

15. Compare ones ($AC = 1$ anywhere?)

Gn = AC

Pr = $A\overline{C} + \overline{A}C + \overline{A}\overline{C} = \overline{AC}$

Kll = 0

$\overline{Ku2} \Rightarrow$ No

Ku2 $\Rightarrow$ Yes

16. Compare equal ($A = C$?)

Gn = $A\overline{C} + \overline{A}C = A \oplus C$

Pr = $AC + \overline{A}\overline{C} = \overline{A \oplus C}$

Kll = 0

Ku2 $\Rightarrow$ No

$\overline{Ku2} \Rightarrow$ Yes

17. Compare magnitude ($A \geq C$?)

Gn = $A\overline{C}$

Pr = $AC + \overline{A}\overline{C}$

Kll = 1

$\overline{Ku2} \Rightarrow$ No

Ku2 $\Rightarrow$ Yes

18. Zero

Gn = 0

Pr = 0

Kll = 0

Ja = 0000_8

(Jau0-Jau2) = 0_8

19. One

Gn = 0

Pr = 0

Kll = 1

Ja = 0001_8

CHAPTER 6

Basic Internal Operations

6.1 Introduction to timing charts

6.2 End

6.3 Operand assembly

6.4 Trap

6.5 Basic opcodes

6.1 Introduction to timing charts

The opcodes will be described by means of timing charts. A timing chart is divided into machine cycles. Each machine cycle is headed by an identifying logical expression opposite which are listed the events that occur throughout that cycle. The three individual clock times (T1, T0, Tp) then follow-opposed by those events peculiar to the given clock time. Both hardware implicit in the performance of an event and the timing of some of the signals are listed in parentheses following the event. Explanatory notes are bracketed and appear, indented, immediately underneath the event they expound.

In order to obtain a complete picture of an opcode it will be necessary to mentally superimpose the End timing chart and the appropriate operand assembly timing chart upon the timing chart of the given opcode.

6.2 End

During the last phase of every opcode, preparations must be made for the next instruction. These preparations are effected by the signal

End.

Most opcodes hold End true throughout the last phase of their execution. However, those opcodes which change the instruction sequence (viz. BRANCH instructions, EXECUTE instructions, and TRAPPING instructions) obviously will not gate

P→S.

This is avoided by having these exceptional opcodes bring up End only at Tp time of their last phase (which, in fact, is always $\phi 0$).

End

$\overline{\text{End}}$ $\overline{\text{Int}}$

T1 P→S

[Access the next instruction from the address in P]

T0

Tp $\overline{Or} \Rightarrow A \longleftrightarrow B$

[Restore A and B - During operand assembly ($\phi 0$), $\overline{Or}$ caused A and B to be swapped]

M→C (Jm)

[i.e. next instruction→C]

(M0-M5)→O (Jm)

[i.e. next opcode→O]

M6→Lp (Jm)

[Set up Lp for $\phi 0$]

Set Fp

[Set Fp for $\phi 0$]

Clear O10

[Clear the memory parity error indicator - in case a transfer to IDLE is gated (see below).]

Clear O11

[Set up for a possible transfer to IDLE (see below)]

$(\overline{Ht} + Ip) \Rightarrow$ Set Cpe

[Check parity of the next instruction]

Go to $\phi 0$

[Perform the next instruction]

Ht $\overline{Ip} \Rightarrow$ Go to $\phi 1$ (more precisely, $\phi 1$ $\overline{O11}$ Ht)

[IDLE - Note that the HALT flip-flop, Ht, must be set and this ($\overline{Ip}$) must not be the End phase of an EXU opcode or a trapping opcode.]

6.3 Operand assembly

The initial decoding of every instruction is similiar. This similarity extends from the read-out of the instruction to the referencing of memory

at the effective address. The term OPERAND ASSEMBLY will be used to refer generally to the whole breadth of this initial decoding.

Operand assembly takes place in $\phi 0$; conversely, $\phi 0$ is only entered for operand assembly.

Some of the general purpose flip-flops used in $\phi 0$ include:

Fp - Fp signals that the current cycle through $\phi 0$ is processing the first word of a (possible) instruction word pair. $\overline{Fp}$ signals that the current cycle through $\phi 0$ is processing the second word of an instruction word pair.

Lp - Lp signals that the current cycle through $\phi 0$ will conclude operand assembly.

Ip - Ip is examined at $\phi 0$ T1. During $\phi 0$ Fp, it will block any change of P; during $\phi 0$ $\overline{Fp}$, it will gate indexing (as opposed to no indexing).

O9-O11 - O9, O10, and O11 will temporarily (during $\phi 0$ $\overline{Fp}$ T1) hold the most significant 3 bits of any 15 bit address.

O10 - O10 is also used (at $\phi 0$ Fp Tp) to effect (via S + 1→P) updating (effectively P + 2→P) of P for double word instructions.

O11 - O11 is also used (at $\phi 0$ Lp Tp) to gate the conclusion (viz; S→P, End) of the opcodes which change the instruction sequence.

The adder (Ja) is used at T1 and Tp times by the operand assembly logic. The adder is reserved at $\phi 0$ T0 time for use by the particular opcodes. These $\phi 0$ T0 uses of the adder, as well as all other $\phi 0$ events

peculiar to certain opcodes, are described under the particular opcode.

Operand Assembly
(Immediate Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

ϕ0 Fp Lp

T1 — S + 1→S (Ja, Pr, K11)

[Access operand from next location]

Clear O10

$\overline{Ip}$ ⇒Set O10

[O10 will cause P to be incremented at Tp]

Clear O11

[Opcodes which change the instruction sequence will set O11 at T0]

Clear Ip

[Opcodes which temporarily leave the instruction sequence will set Ip]

$\overline{Or}$ ⇒A⟷B

[Instructions which operate on B actually effect their operation in A]

T0

Tp — O10 $\overline{O11}$ ⇒S + 1→P (Ja, Pr, K11)

[i.e P + 2→P]

Clear O9-O11

[For use during the execution phases of certain opcodes]

Clear Lp

[For use during the execution phases of certain opcodes]

Leave Fp set

[For use during the execution phases of certain opcodes]

Operation Assembly

(Direct Scratch Pad Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

ϕ0 Fp Lp

T1 — $\overline{Ip} \Rightarrow S + 1 \rightarrow P$ (Ja, Pr, K11)

[i.e. P + 1→P]

(C,...,C, C7,...,C11)→S

[Access operand from scratch pad]

Clear O10

[O10 would gate the incrementing of P at Tp]

Clear O11

[Opcodes which change the instruction sequence will set O11 at T0]

Clear Ip

[Opcodes which temporarily leave the instruction sequence will set Ip]

$\overline{Or} \Rightarrow A \leftrightarrow B$

[Instructions which operate on B actually effect their operation in A]

T0

Tp — Clear O9-O11

[For use during the execution phases of certain opcodes]

Clear Lp

[For use during the execution phases of certain opcodes]

Leave Fp set

[For use during the execution phases of certain opcodes]

Operand Assembly

(Full Direct Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

$\phi 0$ Fp $\overline{Lp}$

T1 — S + 1→S (Ja, Pr, K11)

[Access bottom 12 bits of the effective address]

Clear O10

$\overline{Ip}$ ⇒Set O10

[O10 will gate the incrementing of P at Tp]

Clear O11

[O11 would gate a change in the instruction sequence at Tp]

Clear Ip

[$\overline{Ip}$ gates the proper setup of S at $\phi 0$ $\overline{Fp}$ Lp T1]

T0

Tp — O10⇒S + 1→P (Ja, Pr, K11)

[i.e. P + 2→P]

(C9-C11)→(O9-O11)

[Save the upper 3 bits of the effective address]

M→C (Jm)

[The bottom 12 bits of the effective address go to C]

Set Lp

[The next cycle through ϕ0 will be the last]

Clear Fp

[The next cycle through ϕ0 will be to process the second word of an instruction word pair]

Set Cpe

[Check parity of these bottom 12 effective address bits]

ϕ0 $\overline{Fp}$ Lp

T1 $\overline{Ip}$ ⇒(O9, O10, O11, C0, . . . , C11)→S

[Access the operand]

Clear O10

[O10 would gate the incrementing of P at Tp]

Clear O11

[Opcodes which change the instruction sequence will set O11 at T0]

Clear Ip

[Opcodes which temporarily leave the instruction sequence will set Ip]

$\overline{Or}$ ⇒A⟷B

[Instructions which operate on B actually effect their operation in A]

T0

Tp — Clear O9-O11

[For use during the execution phases of certain opcodes]

Clear Lp

[For use during the execution phases of certain opcodes]

Set Fp

[For use during the execution phases of certain opcodes]

Operand Assembly

(Index Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

$\phi 0$ Fp $\overline{Lp}$

T1 — S + 1→S (Ja, Pr, K11)

[Access the bottom 12 bits of the base address]

Clear O10

$\overline{Ip}$ ⇒Set O10

[O10 will gate the incrementing of P at Tp]

Clear O11

[O11 would gate a change in the instruction sequence at Tp]

Set Ip

[Ip gates the proper setup of S at $\phi 0$ $\overline{Fp}$ Lp T1]

T0

Tp — O10 ⇒S + 1→P (Ja, Pr, K11)

[i.e. P + 2→P]

(C9-C11)→(O9-O11)

[Save the upper 3 bits of the base address]

M→C (Jm)

[The bottom 12 bits of the base address go to C]

Set Lp

[The next cycle through ϕ0 will be the last]

Clear Fp

[The next cycle through ϕ0 will be to process the second word of an instruction word pair]

Set Cpe

[Check parity of these bottom 12 base address bits]

ϕ0 $\overline{Fp}$ Lp

T1 Ip ⇒ {(O9, O10, O11, C0, . . . , C11)-(0, 0, 0, A0, . . . , A11)} →S

[Access the operand]

Clear O10

[O10 would gate the incrementing of P at Tp]

Clear O11

[Opcodes which change the instruction sequence will set O11 at T0]

Clear Ip

[Opcodes which temporarily leave the instruction sequence will set Ip]

$\overline{Or}$ ⇒A⟷B

[Instructions which operate on B actually effect their operation in A]

T0

Tp Clear O9-O11

[For use during the execution phases of certain opcodes]

Clear Lp

[For use during the execution phases of certain opcodes]

Set Fp

[For use during the execution phases of certain opcodes]

Operand Assembly

(Full Indirect Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

$\phi 0$ Fp $\overline{Lp}$

T1 — S + 1→S (Ja, Pr, K11)

[Access bottom 12 bits of the indirect address]

Clear O10

$\overline{Ip}$ ⇒Set O10

[O10 will gate the incrementing of P at Tp]

Clear O11

[O11 would gate a change in the instruction sequence at Tp]

Clear Ip

[$\overline{Ip}$ gates the proper setup of S at $\phi 0$ $\overline{Fp}$ $\overline{Lp}$ T1]

T0

Tp — O10⇒S + 1→P (Ja, Pr, K11)

[i.e. P + 2→P]

(C9-C11)→(O9-O11)

[Save the upper 3 bits of the indirect address]

M→C (Jm)

[The bottom 12 bits of the indirect address go to C]

Leave Lp clear

[The next cycle through ϕ0 will not be the last]

Clear Fp

[The next cycle through ϕ0 will be to process the second word of an instruction word pair]

Set Cpe

[Check parity of these bottom 12 indirect address bits]

ϕ0 $\overline{Fp}$ $\overline{Lp}$

T1 $\overline{Ip} \Rightarrow$ (O9, O10, O11, C0, . . . , C11)→S

[Access the indirect instruction]

Clear O10

[O10 would gate the incrementing of P at Tp]

Clear O11

[O11 would gate a change in the instruction sequence at Tp]

Set Ip

[Ip will block any change to P during the next ϕ0]

T0

Tp M→C (Jm)

[Indirect instruction goes to C]

M6→Lp (Jm)

[Re-initialize Lp]

Set Fp

[Re-initialize Fp]

Set Cpe

[Check parity of the indirect instruction]

Stay in $\phi 0$

[Operand assembly begins again]

Operand Assembly

(Indirect Scratch Pad Addressing)

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

$\phi 0$ Fp $\overline{Lp}$

T1 — $\overline{Ip} \Rightarrow$ S + 1→P (Ja, Pr, K11)

[i.e. P + 1→P]

(0, . . . , 0, 1, C9, C10, C11, 0)→S

[Access the indirect instruction]

Clear O10

[O10 would gate the incrementing of P at Tp]

Clear O11

[O11 would gate a change in the instruction sequence at Tp]

Set Ip

[Ip will block any changes to P during the next $\phi 0$]

T0

Tp — M→C (Jm)

[Indirect instruction goes to C]

M6→Lp (Jm)

[Re-initialize Lp]

Leave Fp set

[Re-initialize Fp]

Set Cpe

[Check parity of the indirect instruction]

Stay in ϕ0

[Operand assembly begins again]

6.4 Trap

Logical provisions have been made to TRAP certain opcodes instead of executing them normally. When trapping a given opcode, all normal operations are inhibited; the P register is not incremented. Instead, the instruction (pair) at a uniquely defined location pair is executed. The instruction at the trap address will normally be a BMC/BRM to a trap subroutine; since the P register contains the address of the first word of the trapping instruction (pair), proper linkage between the trap subroutine and the trapping instruction is established.

DVX (12/52) and MUX (13/53) are optional instructions. If the option is installed, these instructions will never trap. If the option is not installed, these instructions will always trap.

POT/BOT (10/50), WOT/ROT (11/51), PIN/BPI (14/54), and WIN/RIN (15/55) may operate normally (Pct) or trap ($\overline{\text{Pct}}$). The program controls the operation via Pct.

The trap address pairs have been defined as follows:

Opcode	Address
POT (10)	00130
BPO (50)	00132
WOT (11)	00134
ROT (51)	00136

Opcode	Address
DVB (12)	00120
DVA (52)	00122
MUA (13)	00124
MUB (53)	00126
PIN (14)	00140
BPI (54)	00142
WIN (15)	00144
RIN (55)	00146

End — M→C; (M0-M5)→O; M6→Lp; Set Fp; Set Cpe

$\phi 0$ Tr — $Tr = \overline{Pct}\ \overline{O1}\ O2\ \overline{O4} + \overline{Option}\ \overline{O1}\ O2\ \overline{O3}\ O4$

T1 — Block all normal transfers to S and P.

[Thus P remains pointing to the trapping instruction]

TRAP ADDRESS→S

[Access the instruction at the trap location]

Set Ip

[Ip will block any change of P during the next $\phi 0$. Thus the instruction at the trap address is truly executed (a la mode de EXU)]

Block the possible interchange of A and B

[$\phi 0$ Lp $\overline{Or}$ would have gated this interchange]

T0 — Block the possible clearing of A

[($\phi 0$ Lp) ($\overline{O1}$ O2 $\overline{O3}$ O4 O5) would have gated this clearing. But ($\overline{O1}$ O2 $\overline{O3}$ O4 O5) is held at ground when the MUX/DVX option is absent.]

Tp End (only at Tp)

[End gates the preparation for the next instruction]

Block the possible increment of P

[ϕ0 O10 would have gated this increment]

Block the possible interchange of A and B

[End $\overline{\text{Or}}$ would have gated this interchange]

Block all set pulses to Lp that are not gated by Jm6

[End will, as always, transfer Jm6 to Lp]

6.5 Basic opcodes

CYX, CYD, CFX, CFI, CFD

(02/42, 03/43)

The shift commands (02/42, 03/43) have the capability to effect both single-register and double-register shifts. All shift commands have the following common general structure:

1. All shifts are left circular.
2. $0 \leq$ shift count $\leq 17_8$
3. The least significant four bits of the effective address (E11-E14) determine the shift count-these four bits should contain the 1's complement of the desired shift count.
4. The fifth least significant bit of the effective address (E10) determines whether the shift will be single-register or double-register:

   $\overline{\text{E10}} \Rightarrow$ single-register shift

   E10 $\Rightarrow$ double-register shift
5. Thus, the operation of a particular shift opcode is completely determined by the least significant five bits on the effective address (E10-E14).

$\phi 0$ Lp

T1

T0 (S0-S11)→C (Ja, Pr)

[i.e. Complemented shift count→(C8-C11)

Shift indicator→C7]

Tp

$\phi 3$ (C8-C11 ≠ 1111_2) ⇒C + 1→C (Ja, Pr, K11)

[Increment shift count if terminus has not yet been reached]

Shift A left one binary position

[A is shifted during all SHIFT opcodes]

(C8-C11 ≠ 1111_2) C7 ⇒Also shift B left one binary position

[B is shifted only during double-register SHIFT opcodes]

(C8-C11 ≠ 1111_2) CYX ⇒A0→A11

[X is shifted left circular]

(C8-C11 ≠ 1111_2) CFX ⇒ A0→F(

F(→A11

[F(, X) is shifted left circular]

(C8-C11 ≠ 1111_2) CYD ⇒B0→A11

A0→B11

[(A, B) is shifted left circular]

(C8-C11 ≠ 1111_2) CFI ⇒B0→A11

A0→F(

F(→B11

[($F\ell$, B, A) is effectively shifted left circular]

(C8-C11 $\neq$ 1111_2) CFD $\Rightarrow$B0$\rightarrow$A11

A0$\rightarrow$Fℓ

F$\ell\rightarrow$B11

[($F\ell$, A, B) is shifted left circular]

T1

T0

Tp (C8-C11 $\neq$ $111X_2$) $\Rightarrow$Stay in ϕ3

[The shift is not finished - and will not be finished on this clock time. Continue shifting]

(C8-C11 = $111X_2$) $\Rightarrow$Go to ϕ7

[The shift is finished-or will be finished on this clock time]

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

STX

(04/44)

ϕ0 Lp

T1

T0 Mw (through Tp)

[The memory reference of the effective address thus becomes a write cycle]

A→M (Jm-through Tp)

[A will be written into memory]

Tp

ϕ6

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

COX, CEX, CMX

(05/45, 06/46, 07/47)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

. . [i.e. operand→C]

Set Cpe

[Check parity of the operand]

$\phi 6$ COX $\Rightarrow$ AC $\rightarrow$ Gn

$\overline{AC} \rightarrow Pr$

[Thus any AC will result in Ku2]

CEX $\Rightarrow$ A $\oplus$ C $\rightarrow$ Gn

$\overline{A \oplus C} \rightarrow Pr$

[Thus a difference of bits in any corresponding position of A and C will result in Ku2]

CMX $\Rightarrow$ Effect a normal subtract (Gn, Pr, K11)

[Thus A $\geq$ C will result in Ku2]

T1 COX $\Rightarrow \overline{Ku2} \rightarrow F\ell$

[Whence $F\ell \Rightarrow (X) \wedge (E) \neq 1$ anywhere]

CEX $\Rightarrow$ Ku2 $\rightarrow F\ell$

[Whence $F\ell \Rightarrow (X) \neq (E)$]

CMX $\Rightarrow$ Ku2 $\rightarrow F\ell$

[Whence $F\ell \Rightarrow (X) \geq (E)$]

End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

The divide operation is entirely straightforward. In essence the internal logic performs the division

$$(A, B) \div (C)$$

The logic will initially assume that

$$(A) < (C)$$

the division can then be effected by 12 (trial) subtract operations. These subtractions will always take place in A. This means that (A, B) must be shifted left one binary position before each subtraction. This allows the quotient to be inserted, a bit at a time, into B from the least significant end (B11). In fact, the left shift of A will preceed the left shift of B. This will allow the quotient bit to be inserted into B at the time of B's left shift.

A divide step consists of:

1. Shift A left one binary position

   A0→O9

   B0→A11

2. Try subtracting C from (O9, A). (The logic need only subtract C from A.) The subtraction will be possible either if O9 = 1 or Ku2 = 1 ($\overline{Ku2}$ is the borrow out of the subtract operation A-C).
3. Shift B left one binary position (quotient bit = O9 + Ku2→B11). If the subtract is possible (i.e. quotient bit = O9 + Ku2 = 1), then replace A with the new partial remainder (viz. A-C).

If the above sequence is done a total of 12 times (this divide step count will be made in S), the results will be:

1. The final quotient, properly shifted, in B
2. The final remainder in A

The 12 divide steps are somewhat differentiated by Fp and Lp:

Fp	Lp	Divide Step
1	0	1st
0	0	2nd through 11th
0	1	11th
1	1	12th

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand - divisor→C]

Set Cpe

[Check parity of the operand]

ϕ3

T1 Shift A left one binary position

[This is the start of a divide step]

A0→O9

[Save the most significant bit of A for the ensuing subtraction]

B0→A11

[This consummates the left shift of A]

Fp $\overline{\text{Lp}}$ ⇒1→S (Ja, K11)

[Fp was left set and Lp was left reset by ϕ0. This initializes the divide step count to 1]

$\overline{Fp}\ \overline{Lp} \Rightarrow S + 1 \rightarrow S$ (Ja, Pr, K11)

[This increments the divide step count]

Fp Lp ⇒ End (through Tp)

[This is the last divide step - End gates the preparation for the next instruction]

T0 Effect a normal subtract operation (through Tp - Ja, Gn, Pr, K11)

[Ku2 will be examined at Tp to see if this subtraction is possible]

$(S8\text{-}S11 = 1X11_2) \Rightarrow$ Set Lp

[This is the penultimate divide step - Fp Lp will gate End throughout the last divide step]

Tp Shift B left one binary position

[A had already been shifted at T1]

(O9 + Ku2)→B11

[i.e. quotient bit→B11. This consummates the left shift of B]

(O9 + Ku2) ⇒ Ja→A

[i.e. A-C→A. This places a new (partial) remainder in A]

Clear Fp

[Fp $\overline{Lp}$ had gated the initialization of the divide step count]

Lp ⇒ Set Fp

[This is the end of the penultimate divide step and concludes the preparations that will allow End to be true throughout the last divide step]

End ⇒ Block the possible interchange of A and B

[This interchange, normally gated by End $\overline{Or}$, would have interferred with other A and B register transfers.]

The multiply operation is achieved, quite directly, by 12 additions. In essence the internal logic performs the multiplication

$$(B) \cdot (C)$$

by the following steps:

1. Clear the partial product (A)
2. Examine the least significant bit of the multiplier (B11)
3. If B11 = 1, add C to A and place the sum in A.
   If B11 = 0, do nothing.
4. Shift the partial product (A) right one bit position. If B11 = 1, place the carry from the above addition (A + C) in A0. If B11 = 0, place zero in A0.
5. The bit shifted out of the least significant end of A is the least significant bit of the final answer. It can not be changed by any further additions.
6. Shift the multiplier (B) right one bit position. Place the final answer bit (that was shifted out of A) in B0. The former content of B11 (the least significant multiplier bit) are lost; it has been used and is no longer needed.
7. Since both the multiplier and partial product have been shifted right one bit position, we are in a position to return to step 2 to process the second least significant bit of the original multiplier.

By performing steps 2-6 (above) a total of 12 times the multiplication is accomplished. The final answer appears in (A, B).

Because of shifted transfer paths, the internal logic can perform a complete addition and shift in one clock time. Therefore, after the multiplicand has been accessed, only 4 machine cycles are needed to complete the

Fp and Lp:

Fp	Lp	Count
1	0	1
0	0	2
0	1	3
1	1	4

$\phi 0$ Lp

T1

T0 0 → A

[This initializes the partial product to zero. It is effected by directly pulsing the enable, Ag]

Tp M → C (Jm)

[i.e. operand = multiplicand → C]

Set Cpe

[Check parity of the operand]

$\phi 3$ B11 ⟹ Effect a normal add operation (Ja, Gn, Pr)

[i.e. A + C → Ja]

$\overline{B11}$ ⟹ A → Ja (Pr)

[Note that, in this case, Ku2 = 0]

Ja (shifted right one binary position) → A

Ku2 → A0

[This gives us a new partial product]

Shift B right one binary position

[This repositions the next multiplier bit at B11 and makes room for a final product bit at B0]

Ja11→B0

[i.e. final product bit→B0]

T1 Fp Lp ⇒ End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp $\overline{Lp}$ ⇒ Clear Fp

[This changes the count from 1 to 2]

$\overline{Fp}$ ⇒ Set Lp

[This changes the count from 2 to 3]

Lp ⇒ Set Fp

[This changes the count from 3 to 4]

End ⇒ Block the possible interchange of A and B

[This interchange, normally gated by End $\overline{Or}$, would have interferred with the Ja (right shift)→A and B (right shift)→B transfers at this time]

MPO/MPF

(16/56)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

ϕ4		MPO $\Rightarrow$1$\rightarrow$K11
		MPF $\Rightarrow$F$\ell\rightarrow$K11
	T1	C + K11$\rightarrow$C (Ja, Pr, K11)
		[The operand has been properly incremented]
		Ku2$\rightarrow$Fℓ
		[i.e. carry out$\rightarrow$Fℓ]
	T0	Mw (through Tp)
		[This memory cycle will be a write cycle]
		C$\rightarrow$M (Jm - through Tp)
		[The incremented operand will now be returned to memory]
	Tp	
ϕ7		
	T1	End (through Tp)
		[End gates the preparation for the next instruction]
	T0	
	Tp	

XMF, LDF

(17/57)

ϕ0	Lp	
	T1	
	T0	
	Tp	M$\rightarrow$C (Jm)
		[i.e. operand$\rightarrow$C]

Set Cpe

[Check parity of the operand]

ϕ4

T1 C→C (Ja, Pr)

[This is a hardware quirk]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

XMF ⇒(Fℓ, C1-C11)→M (Jm-through Tp)

[Fℓ will be written into the most significant bit of the operand (along with the other original eleven bits)]

LDF ⇒C→M (Jm - through Tp)

[Thus LDF (needlessly) rewrites the original operand back into memory]

Tp

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0 C0 →Fℓ

[The most significant bit of the operand has been loaded into Fℓ]

Tp

SUX, SCX

(20/60, 21/61)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

ϕ6 SUX ⇒1→K11

SCX ⇒ $\overline{FL}$→K11

T1 A-C-$\overline{K11}$→A (Ja, Gn, Pr, K11)

[The subtraction has been properly performed]

$\overline{Ku2}$→FL

[i.e. borrow out→FL]

End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

ADX, ACX

(22/62, 23/63)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

$\phi 6$ ADX $\Rightarrow$ 0→K11

ACX $\Rightarrow$ Fℓ→K11

[A hardware quirk causes Fℓ→K11 to be also gated by $\overline{O11}$; however, O11 was left cleared by $\phi 0$ Lp]

T1 A + C + K11→A (Ja, Gn, Pr, K11)

[The addition has been properly performed]

Ku2→Fℓ

[i.e. carry out→Fℓ]

End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

LDX, ANX, EOX, ORX

(24/64, 25/65, 26/66, 27/67)

$\phi 0$ Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

$\phi 6$

T1 LDX $\Rightarrow$ C→A (Ja, Pr)

[The LOAD has been performed]

ANX ⇒ A∧C→A (Ja, Pr)

[The AND has been performed]

EOX ⇒ A ⊕ C→A (Ja, Pr)

[The EXCLUSIVE OR has been performed]

ORX ⇒ A∨C→A (Ja, Pr)

[The INCLUSIVE OR has been performed]

End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

BAX

(30)

ϕ0 Lp

T1 A↔B

[This is gated, as always, by ϕ0 Lp $\overline{\text{Or}}$]

T0 Set O11

[O11 will gate the completion of this opcode at Tp]

Tp O11 ⇒ End (only at Tp)

[End gates the preparation for the next instruction-which is actually the operand currently being read from memory]

S→P (Ja, Pr)

[i.e. effective address→P; the branch is made]

End ⇒ Block the interchange of A and B

[By blocking this interchange, normally gated by End $\overline{\text{Or}}$, A and B remain swapped (see T1 above)]

ϕ0 Lp

T1

T0 A-1→A (Ja, Gn, Pr)

[This decrements A]

Ku2 ⇒ Set O11

[If the decremented contents of A are unequal to 7777_8, then the branch is taken (gated at Tp by O11)]

$\overline{Ku2}$ ⇒ Leave O11 clear

[If the decremented contents of A are equal to 7777_8, then the next instruction in sequence will be taken]

Tp O11 ⇒ End (only at Tp)

[End gates the preparation for the next instruction-which is actually the operand currently being read from memory]

S→P (Ja, Pr)

[i.e. effective address→P; the branch is taken]

$\overline{End}$ ⇒ Go to ϕ7

[The branch was not taken - the next instruction in sequence must be accessed]

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

ϕ0 Lp

T1

T0 $BFF \wedge \overline{F\ell} + BFT \wedge F\ell \Rightarrow$ Set O11

[If $F\ell$ is in the condition being tested for, the branch is taken (gated at Tp by O11)]

$BFF \wedge F\ell + BFT \wedge \overline{F\ell} \Rightarrow$ Leave O11 clear

[If $F\ell$ is not in the condition being tested for, the next instruction in sequence will be taken]

Tp O11 $\Rightarrow$ End (only at Tp)

[End gates the preparation for the next instruction - which is actually the operand currently being read from memory]

S→P (Ja, Pr)

[i.e. effective address→P; the branch is taken]

$\overline{End} \Rightarrow$ Go to ϕ7

[The branch was not taken-the next instruction in sequence must be accessed]

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

EXU

(72)

ϕ0 Lp

T1 Set Ip

[Ip will block any change of P during the next ϕ0. Thus the instruction at the effective address is truly EXECUTED]

T0 Set O11

[O11 will gate the completion of this opcode at Tp]

Tp O11 ⇒ End (only at Tp)

[End gates the preparation for the next instruction- which is actually the operand currently being read from memory]

Block the transfer of S to P

[Ip will actually block this transfer. This transfer (normally gated by O11) must be blocked because EXU only leaves the instruction sequence to execute this one instruction]

BRL, BRU

(33/73)

ϕ0 Fp

T1

T0 BRL ⇒ CO → F.

[This loads F. from bit 0 of the first word of an instruction (either direct or indirect)]

C1→Pct

[This loads Pct from bit 1 of the first word of an instruction (either direct or indirect)]

Tp

ϕ0 Lp — Note that this machine cycle could be concurrent with ϕ0 Fp above.

T1

T0 — Set O11

[O11 will gate the completion of this opcode at Tp]

Tp — O11 ⟹ End

[End gates the preparation for the next instruction- which is actually the operand currently being read from memory]

S→P (Ja, Pr)

[i.e. effective address→P; the branch is taken]

XMX

(34/74)

ϕ0 Lp

T1

T0

Tp — M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

ϕ4

T1 C→C (Ja, Pr)

[This is a hardware quirk]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

A→M (Jm - through Tp)

[The register is stored at the effective address]

Tp

ϕ7

T1 C→A (Ja, Pr)

[The operand is stored in the register]

End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

MAX

(35/75)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

ϕ4

T1 A∧C→C (Ja, Pr)

[The AND has been performed]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm- through Tp)

[The AND result is returned to memory]

Tp

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

MPX

(36/76)

ϕ0 Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

$\phi 4$

T1 A + C→C (Ja, Gn, Pr)

[The addition has been performed]

Ku2→F(

[i.e. carry out→F(]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm - through Tp)

[The sum is returned to memory]

Tp

$\phi 7$

T1 End (through Ip)

[End gates the preparation for the next instruction]

T0

Tp

BMC, BRM

(37/77)

$\phi 0$ Lp

T1

T0 Mw (through Tp)

[The memory reference of the effective address thus becomes a write cycle]

(Fℓ, Pct, 0,...,0, Pu0, Pu1, Pu2)→M (Jm - through Tp)

[The first word of the mark is stored at the effective address]

Tp

ϕ4

T1 S + 1→S (Ja, Pr, K11)

[The ensuing memory reference will be at the effective address + 1]

Mw (through Tp)

[The ensuing memory cycle will be a write cycle]

(P0-P11)→M (Jm - through Tp)

[The second word of the mark is stored at the effective address + 1]

T0

Tp S + 1→P (Ja, Pr, K11)

[i.e. effective address + 2→P; the branch is made]

BMC ⇒ Clear Fℓ

Set Pct

ϕ7

T1 End (through Tp)

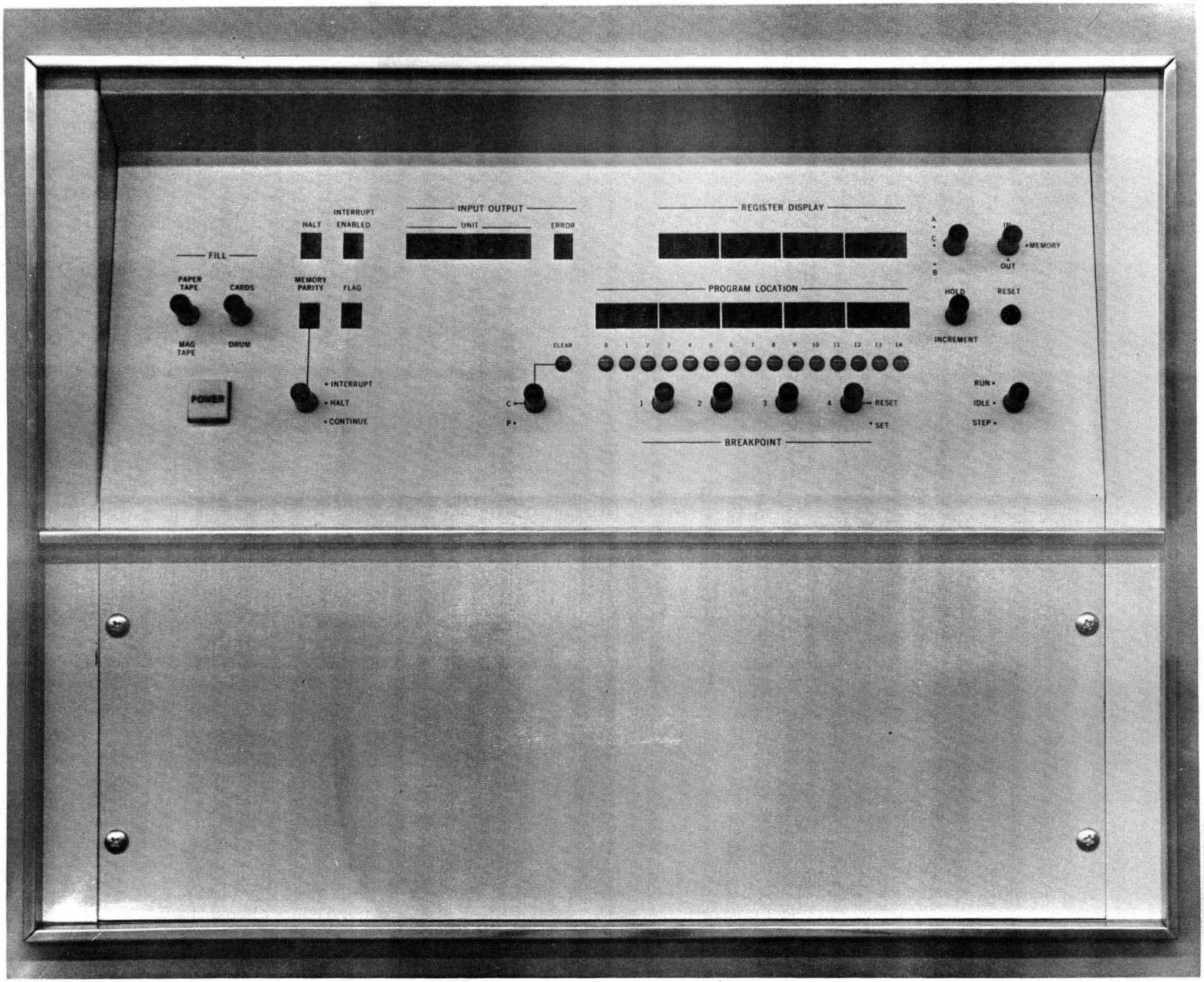
[End will gate the preparation for the next instruction-which will be located in the effective address + 2 (P was set to this address at ϕ4 Tp)]

T0

Tp

CHAPTER 7

Console Operations



Model (Computer Control Panel

7.1 Introduction

A unique phase is entered and remained in when not performing opcodes. $\phi 1$ is this IDLE phase. The computer may enter IDLE because of any of the following:

1. The operator depresses the RESET button. [St will gate an immediate IDLE.]
2. The operator moves the RUN-IDLE-STEP switch from RUN to IDLE. [Ht will gate an IDLE at the completion of the current instruction (End Tp).]
3. The operator STEP's an instruction. [Ht will gate a return to IDLE at the completion of the instruction (End Tp).]
4. A HALT instruction is executed by the program. [Ht will gate an IDLE at the completion of the HALT instruction (End Tp).]
5. A memory parity error halt occurs. [Cp Kp will gate an immediate IDLE.]

$\phi 1$ has been divided into four subphases:

$\phi 1$ $\overline{O11}$ Ht - This is the subphase of IDLE that is always entered first. This is basically an interlocking subphase; there is no way to leave the subphase unless certain internal and external conditions are met.

$\phi 1$ O11 Ht - This is the subphase of IDLE which allows the registers to be displayed and their contents changed. This is also the subphase which recognizes and instigates any console function.

$\phi 1$ O11 $\overline{Ht}$
$\phi 1$ $\overline{O11}$ Ht - These two subphases of IDLE execute every console function.

End Ht $\overline{Ip}$ M→C; Clear O10; Clear O11

$\phi 1$ $\overline{O11}$ Ht Clear O

[This clears out the opcode and initializes Or for register display (see φ1 O11 Ht - $\overline{Or} \Rightarrow$ A and B are home)]

$\overline{Fp} \Rightarrow$ Set Fp

[This initializes Fp for register display (see φ1 O11 Ht- $\overline{Fp}$ would cause A and B to swap)]

Clear O9

[This forces C to be displayed on the REGISTER DISPLAY lights during this subphase]

Clear Ip

[This initializes Ip for a return to instruction execution]

Clear Lp

[This initializes Lp for a possible FILL operation]

$\overline{O9} \Rightarrow$ C→REGISTER DISPLAY lights (Ja, Pr)

[The REGISTER DISPLAY lights are driven from Ja]

O10 ⇒MEMORY PARITY light

$\overline{O10} \Rightarrow$ HALT light

[A memory parity error halt will set O10]

T1

T0 (St + $\overline{Tem}$) ⇒$\overline{Mgm}$

[If either the computer is being reset (St) or the memory stack is not up to a minimum operating temperature ($\overline{Tem}$), then a memory reference will not be made]

Tp $\overline{O10}$ $\overline{Kg}$ $\overline{Ks}$ $\overline{Kmi}$ $\overline{Kmo}$ $\overline{Kpu}$ $\overline{St} \Rightarrow$ Set O11

[If there is no memory parity error indication ($\overline{O10}$), all console function switches are unactivated, and the computer is not being reset ($\overline{St}$); then go to subphase φ1 O11 Ht.]

7.2 Register display/alteration

ϕ1 O11 Ht is the IDLE subphase which allows the contents of A, B, C, and P to be examined and changed.

A, B, and C are all displayed via the REGISTER DISPLAY lights. These lights are actually driven from Ja. A or C may be directly displayed. Since there is no way to put B on Ja, B is displayed by interchanging A and B and then actually displaying B from the A register. This operation requires two flip-flops:

Or - The Or flip-flop indicates if A and B are swapped (Or) or home ($\overline{Or}$).

Fp - The Fp flip-flop will syncronize the REGISTER DISPLAY SELECT switch (A or B) with the internal display logic. [$\overline{Fp}$ signals a change of the REGISTER DISPLAY SELECT switch.]

A third flip-flop is used to control the actual register (A or C) currently being displayed via Ja:

O9 - $\overline{O9} \Rightarrow$ display C

O9 $\Rightarrow$ display A

The additional logic needed to alter the currently displayed register (A, B, or C) are minor. The set buttons will also be placed on Pr; thus ORing them, at Ja, with the register currently being displayed. Finally, Ja will be read back into the particular register. To clear the given register, it is only necessary to block the gating that places the register on Ja; then, when Ja is recirculated back into the register, all zeros will be read in.

The P register display/alteration operates on the same principles as above. However, Ja is not used. The PROGRAM LOCATION lights are driven directly from P. P is directly recirculated. The set buttons are gated directly into P.

Some of the pertinent console signals are:

Kb	-	The REGISTER DISPLAY SELECT switch is requesting B
$\overline{Kb}$	-	The REGISTER DISPLAY SELECT switch is requesting A or C
Kc	-	The REGISTER DISPLAY SELECT switch is requesting C
$\overline{Kc}$	-	The REGISTER DISPLAY SELECT switch is requesting A or B
Krp	-	The clear and set buttons should affect the PROGRAM LOCATION (P)
$\overline{Krp}$	-	The clear and set buttons should affect the REGISTER DISPLAY (A, B, or C)
Krc	-	The clear button
Kru0-Kru2 Kr0-Kr11	-	The fifteen set buttons

ϕ1 O11 Ht

Clear O9

$\overline{Kc} \Rightarrow$ Set O9

[O9 will gate the display of A; $\overline{O9}$ will gate the display of C]

Clear Ip

[This initializes Ip for a return to instruction execution]

Clear Lp

[This initializes Lp for a possible FILL operation]

$\overline{O9}$ ⇒ {C ($\overline{Krc}$ + Krp) + (SET BUTTONS) $\overline{Krp}$ Tp} →Ja→C (Ja, Pr)

O9 ⇒ {A ($\overline{Krc}$ + Krp) + (SET BUTTONS) $\overline{Krp}$ Tp} →Ja→A (Ja, Pr)

Ja→REGISTER DISPLAY lights

[$\overline{O9}$ gates the display of C; O9 gates the display of A]

{P ($\overline{Krc}$ + $\overline{Krp}$) + (SET BUTTONS) Krp} →P

P→PROGRAM LOCATION lights

[P is displayed directly]

T1 P→S

[The ensuing memory reference will thus be at the address currently in P]

(Kb ⊕ Or) ⇒Clear Fp

[$\overline{Fp}$ will gate a swap of A and B at T0]

T0 $\overline{Tem}$ ⇒ $\overline{Mgm}$

[If the memory stack is not up to a minimum operating temperature ($\overline{Tem}$), then a memory reference will not be made]

$\overline{Fp}$ ⇒A⟷B

[Swap A and B. The possible Ja→A transfer at this time must be blocked.]

Toggle Or

[Or indicates whether A and B are swapped (Or) or home ($\overline{Or}$)]

Set Fp

[The A and B registers now correspond to the REGISTER DISPLAY SELECT switch]

Tp Tem (Kg + Kmi + Kmo + Kpu + Ks)⇒Clear Ht

[The memory stack is at an operating temperature (Tem) and a console function has been requested. Go to ϕ1 O11 $\overline{Ht}$]

7.3 Console functions

There are, basically, five console functions. Each function generates two signals from its console switch. From the names given the two signals, each signal of a pair appears to be the logical inverse of the other. However, this is not the case. Whenever either signal of the pair bounces true, its complement must already be stably false. These signal pairs obviate any logical elimination of switch bounce.

Kg, $\overline{Kg}$ - Kg ⇒ The RUN/IDLE/STEP switch is at RUN.
This will cause a return to the instruction sequence beginning with the instruction in the location specified by P.

Kmi, $\overline{Kmi}$ - Kmi ⇒ the MEMORY switch is at IN.
This will cause the contents of C to be stored in the location specified by P.

Kmo, $\overline{Kmo}$ - Kmo ⇒ The MEMORY switch is at OUT.
This will cause the contents of the location specified by P to be read into C (and parity checked).

Kpu, $\overline{Kpu}$ - Kpu ⇒ The PROGRAM LOCATION switch is at INCREMENT
This will cause P to be incremented by one. Then the contents of the location specified by this new P will be read into C (and parity checked).

Ks, $\overline{Ks}$ - Ks ⇒ The RUN/IDLE/STEP switch is at STEP
This will cause the execution of one instruction- the one in the location specified by P- followed by a return to IDLE.

RUN

$\overline{(\overline{Kg})}$

$\phi 1$ O11 $\overline{Ht}$

T1 S→S (Ja, Pr)

S→P (Ja, Pr)

[These are hardware quirks; P→S occured at every T1 time of $\phi 1$ O11 Ht]

Or ⇒ Clear Fp

[Or ⇒ A and B were swapped; $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp}$ ⇒A↔B

[Swap A and B]

Toggle Or

[i.e. Clear Or - thus O is now completely clear]

Set Fp

[A and B are now home]

Tp M→C (Jm)

[This is a hardware quirk- the next instruction is read into C and parity checked a whole machine cycle before it is needed. In fact, End will gate a repeat of these operations during the next machine cycle, $\phi 1$ $\overline{O11}$ $\overline{Ht}$]

Set Cpe

[Check parity of the next instruction]

Clear O11

[i.e. Go to $\phi 1$ $\overline{O11}$ $\overline{Ht}$]

$\phi 1\ \overline{O11}\ \overline{Ht}$

T1 End (through Tp)

[End gates the preparation for the next instruction - note that O is cleared, thus an interrupt could be recognized at this time]

T0

Tp End ⇒ Block the interchange of A and B

[End $\overline{Or}$ would have gated this interchange]

End $\overline{Ht}$ ⇒ Go to $\phi 0$

[Enter the instruction sequence and begin program execution]

MEMORY IN

$(\overline{Kmi})$

$\phi 1\ O11\ \overline{Ht}$

T1 S→S (Ja, Pr)

S→P (Ja, Pr)

[These are hardware quirks; P→S occured at every T1 time of $\phi 1$ O11 Ht]

Or ⇒ Clear Fp

[Or ⇒ A and B were swapped; $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp}$ ⇒ A ⟷ B

[Swap A and B]

Toggle Or

[i.e. Clear Or - thus O is now completely cleared]

Set Fp

[A and B are now home]

Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm-through Tp)

[C will be written into memory]

Tp C→C (Jm)

[This is a hardware quirk]

Clear O11

[i.e. Go to $\phi 1\ \overline{O11}\ \overline{Ht}$]

$\phi 1\ \overline{O11}\ \overline{Ht}$

T1

T0

Tp $\overline{End}$ ⇒ Set Ht

[i.e. Return to $\phi 1\ \overline{O11}$ Ht]

MEMORY OUT

$\overline{(\overline{Kmo})}$

$\phi 1$ O11 $\overline{Ht}$

T1 S→S

S→P

[These are hardware quirks; P→S occured at every T1 time of $\phi 1$ O11 Ht]

Or ⇒ Clear Fp

[Or⇒A and B were swapped; $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp} \Rightarrow A \longleftrightarrow B$

[Swap A and B]

Toggle Or

[i.e. Clear Or-thus O is now completely cleared]

Set Fp

[A and B are now home]

Tp M→C (Jm)

[i.e. operand→C; this performs the desired memory read-out]

Set Cpe

[Check parity of the operand]

Clear O11

[i.e. Go to $\phi 1\ \overline{O11}\ \overline{Ht}$]

$\phi 1\ \overline{O11}\ \overline{Ht}$

T1

T0

Tp $\overline{End} \Rightarrow$ Set Ht

[i.e. Return to $\phi 1\ \overline{O11}\ Ht$]

PROGRAM REGISTER INCREMENT

$(\overline{\overline{Kpi}})$

$\phi 1\ O11\ \overline{Ht}$

T1 S + 1→S (Ja, Pr, K11)

S + 1→P (Ja, Pr, K11)

[Both P and S are incremented by one. Note that when entering this machine cycle S = P was assured since P→S occured at every T1 time of $\phi 1\ O11\ Ht$]

Or ⇒ Clear Fp

[Or ⇒ A and B were swapped; $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp}$ ⇒ A ⟷ B

[Swap A and B]

Toggle Or

[i.e. Clear Or-thus O is now completely cleared]

Set Fp

[A and B are now home]

Tp M→C (Jm)

[i.e. operand→C; this performs the desired memory read-out]

Set Cpe

[Check parity of the operand]

Clear O11

[i.e. Go to ϕ1 $\overline{O11}$ $\overline{Ht}$]

ϕ1 $\overline{O11}$ $\overline{Ht}$

T1

T0

Tp $\overline{End}$ ⇒ Set Ht

[i.e. Return to ϕ1 $\overline{O11}$ Ht]

STEP

$\overline{(Ks)}$

ϕ1 O11 $\overline{Ht}$

T1 S→S (Ja, Pr)

S→P (Ja, Pr)

[These are hardware quirks; P→S occurred at every T1 time of ϕ1 O11 Ht]

Or ⇒ Clear Fp

[Or ⇒ A and B were swapped; $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp}$ ⇒ A ⟷ B

[Swap A and B]

Toggle Or

[i.e. Clear Or-thus O is now completely cleared]

Set Fp

[A and B are now home]

Tp M→C (Jm)

[This is a hardware quirk-the next instruction is read into C and parity checked a whole machine cycle before it is needed. In fact, End will gate a repeat of these operations during the next machine cycle, ϕ1 $\overline{O11}$ $\overline{Ht}$]

Set Cpe

[Check parity of the next instruction]

Clear O11

[i.e. Go to ϕ1 $\overline{O11}$ $\overline{Ht}$]

ϕ1 $\overline{O11}$ $\overline{Ht}$

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block the recognizing of any interrupts

[No interrupts are recognized while STEPPING]

T0

Tp End ⟹ Block the interchange of A and B

[End $\overline{\text{Or}}$ would have gated this interchange]

End $\overline{\text{Ht}}$ ⟹ Go to $\phi 0$

[Enter the instruction sequence and execute one instruction. $\overline{\phi 1}\ \overline{\text{Kg}}$ will set Ht so that IDLE is re-entered at the End of this instruction]

7.4 Fill

It is possible to automatically read in a nine word program. This bootstrap process is called FILL. The nine word program will usually be a small loader to bring in and execute a much bigger program. A FILL is effected as follows:

1. Momentarily depress the RESET button.
2. Engage the appropriate FILL switch, corresponding to the peripheral from which it is desired to FILL. The FILLING peripheral may be
   a. PAPER TAPE
   b. MAG TAPE
   c. CARDS
   d. DRUM
3. While keeping the appropriate FILL switch engaged, throw the RUN/IDLE/STEP switch from IDLE to RUN.

FILL operates as follows:

1. In $\phi 1$ O11 Ht, Kg will gate a transfer to $\phi 1$ O11 $\overline{\text{Ht}}$.
2. S (the address to store the first input word) and P (the address at which execution of the program begins) are both set to 00000_8.

3. Set A = 0010_8. This gives a word count of 9 for the ensuing, automatic RIN instruction.

4. Set up an EOM opcode (i.e. O = 00_8) with a control word of $013XX_8$ (i.e. O9-O11 = 0_8; C = $13XX_8$) where

   XX = 04_8 for a FILL from PAPER TAPE

   XX = 06_8 for a FILL from CARDS

   XX = 10_8 for a FILL from MAG TAPE

   XX = 26_8 for a FILL from DRUM

5. Set Ip

6. Go to ϕ5

7. ϕ5 will gate the transmission of the EOM to the indicated peripheral. A two character per word, binary input is specified.

8. Upon completion of the EOM, ϕ5 Ip will block End. Instead, a RIN opcode will be forced into O (O = 55_8) and control will be transferred to ϕ2.

9. Once in ϕ2 the operation is similar to any RIN command: Nine words (A = 0010_8) will be read into memory starting at location 00000_8 (S = 00000_8); then the instruction sequence is begun at location 00000_8 (P = 00000_8).

Note that if, instead of throwing the RUN/IDLE/STEP switch from IDLE to RUN, the operator throws the RUN/IDLE/STEP switch from IDLE to STEP; the above 9 steps will still be performed. However, the computer will halt after the RIN opcode-instead of executing instructions starting at location 00000_8. This could prove useful to the maintenance man.

The console signals apposite to a FILL are:

Kfc - The CARDS FILL switch is engaged.

Kfd - The DRUM FILL switch is engaged.

Kfm - The MAG TAPE FILL switch is engaged.

Kfp - The PAPER TAPE FILL switch is engaged.

Kf - Any one of the FILL switches is engaged.

The ϕ1 FILL operations will be described on the following timing chart. The ϕ5 operations are described on the EOM timing chart. The ϕ2 (and following) operations are identical to those of any RIN opcode.

FILL

$(\overline{Kg}\ Kf)$

ϕ1 O11 $\overline{Ht}$ Clear O9

[This initializes O9 for a FILL]

Clear Lp

[This initializes Lp for a FILL]

T1 0 → S (Ja)

[Thus the first data word will be stored at 00000_8]

0 → P (Ja)

[Thus instruction execution will begin at 00000_8]

Or ⟹ Clear Fp

[Or ⟹ A and B were swapped. $\overline{Fp}$ will gate them home at T0]

T0 $\overline{Fp}$ ⟹ A ⟷ B

[Swap A and B]

Toggle Or

[i.e. Clear Or-thus O is now completely clear]

Set Fp

[A and B are now home. This also initializes Fp for a FILL]

Tp $10_8 \rightarrow A$

[The RIN will read in 9 words]

Correct EOM control word→C

[This control word will depend on Kfc, Kfd, Kfm, and Kfp]

Clear O11

[O9 is also being cleared; O10 must already be clear (ϕ1 O10⇒MEMORY PARITY)]

Set Ip

[Ip will gate a direct transfer from ϕ5 to ϕ2 following the conclusion of the EOM. Ip will also force a RIN opcode (55_8) into O when this transfer is made]

Go to ϕ5 Fp $\overline{Lp}$

[Note that O = 00_8 = EOM]

7.5 Miscellaneous switches

The PROGRAM LOCATION HOLD switch will allow no change to P- except via the console clear and set buttons.

$\overline{Kr}$ - The PROGRAM LOCATION HOLD switch is engaged.

The consequences of a MEMORY PARITY error are determined by a 3-position console switch:

CONTINUE ($\overline{Kp}$, $\overline{Kpi}$) - The MEMORY PARITY is ignored. Any MEMORY PARITY indication (ϕ1 O10) is cleared.

HALT (Kp, $\overline{Kpi}$) - The MEMORY PARITY causes an immediate halt (i.e. transfer to ϕ1 $\overline{O11}$ Ht). The IDLE logic will

remain interlocked until the MEMORY PARITY indication (O10) is cleared - either by RESET or MEMORY PARITY CONTINUE.

INTERRUPT ($\overline{Kp}$, Kpi) - The parity error interrupts are logically armed when the MEMORY PARITY switch is in this position. Thus any MEMORY PARITY will result in one of two possible interrupts:

1) The main frame or standard I/O channel interlace was using memory.
2) The data multiplexing system was using memory.

The position (RESET or SET) of each of the four BREAKPOINT switches on the console may be individually tested:

Kb1 - BREAKPOINT #1 is SET.

Kb2 - BREAKPOINT #2 is SET.

Kb3 - BREAKPOINT #3 is SET.

Kb4 - BREAKPOINT #4 is SET.

The RESET button on the console causes a clear of the internal logic via the signal St. St does the following:

1. Holds the logic in ϕ1 $\overline{O11}$ Ht
2. Clears any MEMORY PARITY indication (O10)
3. Blocks any memory references (via $\overline{Mgm}$)
4. Clears Fℓ
5. Sets Pct
6. Clears the interrupt enable flip-flop, En
7. Clears the standard I/O channel interrupt arms, Aiw1 and Aiw2
8. Clears the standard I/O channel (via Wc)

St is also made available to both external devices (via the POT connector) and the interrupt logic (via the interrupt connector).

7.6 Lights

The actual sources of all the indicator lights on the console follows:

REGISTER DISPLAY	-	Ja
PROGRAM LOCATION	-	P
FLAG	-	Fℓ
HALT	-	ϕ1 $\overline{O11}$ Ht $\overline{O10}$
MEMORY PARITY	-	ϕ1 $\overline{O11}$ Ht O10
INTERRUPT ENABLE	-	En
INPUT-OUTPUT UNIT	-	W9 through W14
INPUT-OUTPUT ERROR	-	We

CHAPTER 8

Interrupts

8.1 Introduction

A priority interrupt level has three states:

INACTIVE - No interrupt signal has been received into the level and none is currently being processed by its interrupt servicing subroutine.

WAITING - An interrupt signal has been received into the level, but is not yet being processed by its interrupt servicing subroutine. (This situation may be due to an interrupt of higher priority being processed at this time.)

ACTIVE - An interrupt signal has been received into the level and is currently being processed by its interrupt servicing subroutine. This means the main frame has recognized the interrupt's presence by executing the instruction in its assigned memory location (which is usually a BMC, 37, to the body of its interrupt servicing subroutine).

Some flexibility is provided in the transferring between interrupt states:

INACTIVE→WAITING - Interrupt levels which do not have the arming feature will automatically proceed from the INACTIVE state to the WAITING state whenever an interrupt signal is recognized. Interrupt levels which have the arming feature will change states upon recognizing the interrupt signal only if the corresponding arm is set. If the arm is reset, no change of states will occur and all record of the interrupt

signal is lost.

WAITING→ACTIVE - Some interrupt levels will automatically proceed from the WAITING state to the ACTIVE state if/as soon as there are no interrupts of higher priority in either the WAITING state or the ACTIVE state. Other interrupt levels will change states only if there are no interrupts of higher priority in the WAITING or ACTIVE state and the interrupt system is enabled (i.e. En is set).

ACTIVE→INACTIVE - For those interrupt levels whose interrupt servicing subroutine consists of a single instruction (the hardware defines the single instruction interrupt levels by a special signal), the interrupt level will automatically proceed from the ACTIVE state to the INACTIVE state at the End of the execution of the instruction in its assigned memory location. For the other interrupt levels, the change of states will occur at the conclusion of the interrupt servicing subroutine (signalled by a BRC instruction).

Each interrupt level has two flip-flops (Is, Ip) to decode the three states described above.

$\overline{Is}\ \overline{Ip} \Rightarrow$ INACTIVE state

$Is\ \overline{Ip} \Rightarrow$ WAITING state

$Is\ Ip \Rightarrow$ ACTIVE state

The signals from the main frame to the interrupt logic include:

Ti - Ti clocks the set of the Is flip-flop of each interrupt level. Thus, the change from the INACTIVE state to the WAITING state occurs on the trailing edge of Ti.

Ie - Ie clocks the set of the Ip flip-flop associated with the currently WAITING (Is $\overline{Ip}$) interrupt level of highest priority. Thus, the change from the WAITING state to the ACTIVE state occurs on the trailing edge of Ie.

Ib - Ib clocks the clear of both the Is and the Ip flip-flops associated with the currently ACTIVE (Is Ip) interrupt level of highest priority. Thus, the change from the ACTIVE state to the INACTIVE state occurs on the trailing edge of Ib.

Each of the main frame signals to the interrupt logic is normally two clock times long

Ti : $\overline{T0}$

Ie : $\overline{T1}$

Ib : $\overline{T1}$

Ie/Ib signal only during the entering/leaving of an interrupt servicing subroutine. Ti is normally signalling during every machine cycle. However, Ti is not allowed to drop during any machine cycle in which Ie signals. Since it is logically impossible to enter two different interrupt servicing subroutines on two consecutive machine cycles, Ti is maximally five clock times long. This leads to: ANY EXTERNAL INTERRUPT SIGNAL TO THE INTERRUPT LOGIC SHOULD BE AT LEAST 3 CYCLES LONG.

The signals from the interrupt levels to the main frame include:

and all interrupt levels of higher priority are in the INACTIVE state. This interrupt level may proceed to the ACTIVE state whether the interrupt system is enabled or not.

Ir - An interrupt level is currently in the WAITING state and all interrupt levels of higher priority are in the INACTIVE state. This interrupt level may proceed to the ACTIVE state only if the interrupt system is enabled (i.e. En is set).

Ij - The currently ACTIVE interrupt level of highest priority is a single-instruction interrupt.

N6-N14 - The interrupt address associated with the currently WAITING interrupt level of highest priority. These address lines will be shifted left one binary position to define a unique pair of memory locations.

8.2 Recognition

During the End phase of most opcodes, Ir and Is are examined. If a WAITING interrupt level may go ACTIVE, the next instruction is not accessed. Instead, the instruction at the assigned memory location is accessed and executed-with the P register still containing the address of the next instruction in the main instruction sequence. [This executed instruction will normally be a BMC, 37, to the body of the interrupt servicing subroutine for that interrupt level.]

The following opcodes never allow interrupts during their End cycle:

EOM	(00/40)
POT/BPO	(10/50)
WOT/ROT	(11/51)
PIN/BPI	(14/54)
WIN/RIN	(15/55)
BAX	(30)
BRL/BRU	(33/73)

If the following opcodes branch, then interrupts will not be allowed during their End cycle:

BDA	(70)
BFF/BFT	(31/71)

Certain conditions will also block interrupt recognition:

1. When STEPPING (Ks)
2. When halting (Ht)
3. When executing the instruction of a single-instruction interrupt subroutine (Ij). [If interrupts were not blocked, the automatic Ib might occur after a new Ie.]
4. When signalling a memory parity interrupt (Cp Kpi). [The two memory parity interrupt signals are only one clock time (T1) long and their falling edges will gate their respective interrupt levels from the INACTIVE state to the WAITING state (even though Ti remains true). If interrupts were not blocked, this change of state might interfere with Ie.]

END

End Int

T1 N→S

[Access the next instruction from the assigned interrupt address]

Set Ip

[Ip will block any change of P during the next ϕ0. Thus the instruction at the interrupt address is truly EXECUTED.]

Set Ie

[Ie will gate the currently WAITING interrupt level of highest priority to the ACTIVE state]

Do not clear Ti

[No new interrupt levels will be gated from the INACTIVE state to the WAITING state while we are gating this interrupt level from the WAITING state to the ACTIVE state (via Ie)]

T0

Tp $\overline{Or} \Rightarrow A \longleftrightarrow B$

[Restore A and B-during operand assembly ($\phi 0$), $\overline{Or}$ caused A and B to be swapped]

M→C (Jm)

[i.e. next instruction→C]

(M0-M5)→O (Jm)

[i.e. next opcode→O]

M6→Lp(Jm)

[Set up Lp for $\phi 0$]

Set Fp

[Set Fp for $\phi 0$]

Clear Ie

[This drops the Ie signal to the interrupt logic]

Ip ⇒Set Cpe

[Check parity of the next instruction]

Go to $\phi 0$

[Perform the next instruction]

8.3 BRC opcode

For normal interrupt levels ($\overline{Ij}$), the interrupt servicing subroutine

will control the change from the ACTIVE state to the INACTIVE state. A special instruction, BRC (32), will gate (via Ib) the currently ACTIVE interrupt level of highest priority to the INACTIVE state. Unless special programming precautions are taken; the BRC instruction should only be used as the last instruction in the interrupt servicing subroutine (i.e. the instruction that exits from the interrupt servicing subroutine back to the main program). The BRC instruction has been made a 3-cycle instruction so that another interrupt may be recognized during its End phase.

BRC

(32)

ϕ0 Fp

T1

T0 C0→Fℓ

[This loads Fℓ from bit 0 of the first word of an instruction (either direct or indirect)]

C1→Pct

[This loads Pct from bit 1 of the first word of an instruction (either direct or indirect)]

Tp

ϕ0 Lp Note that this machine cycle could be concurrent with ϕ0 Fp above.

T1

T0 Set O11

[O11 will gate S to P at Tp]

Ib (through Tp)

[Ib will gate the currently ACTIVE interrupt level of highest priority to the INACTIVE state]

Tp O11 ⇒ S→P (Ja, Pr)

[i.e. effective address→P; the branch is taken]

Ib ⇒ Block End

[O11 would normally have gated End at ϕ0 Tp. BRC will have two extra machine cycles to allow the recognition of interrupts during its End phase]

ϕ4

T1

T0

Tp

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

8.4 Leaving IDLE

If a HALT instruction is executed but the RUN/IDLE/STEP switch is left in RUN, any interrupt (En Ir + Is) will be properly processed. Upon entering the interrupt servicing subroutine, P contains the address of the instruction following the HALT command. This usually means that, after processing the interrupt, the instruction sequence will be re-entered at

the instruction following the HALT command.

End Ht $\overline{Ip}$ M→C; Clear O10; Clear O11

ϕ1 $\overline{O11}$ Ht Clear O

[This clears out the opcode]

Clear O9

[This forces C to be displayed on the REGISTER DISPLAY lights]

$\overline{O9}$ ⟹ C→REGISTER DISPLAY lights (Ja, Pr)

[The REGISTER DISPLAY lights are driven from Ja]

$\overline{O10}$ ⟹ HALT light

T1

T0 $\overline{Tem}$ ⟹ $\overline{Mgm}$

[If the memory stack is not up to a minimum operating temperature ($\overline{Tem}$), a memory reference will not be made]

Tp $\overline{O10}$ $\overline{St}$ Tem Kg (En Ir + Is) ⟹ Clear Ht

[i.e. Go to ϕ1 $\overline{O11}$ $\overline{Ht}$; this is only effected if there is no parity error indication ($\overline{O10}$), RESET is not being actuated ($\overline{St}$), the memory stack temperature is minimal (Tem), and the RUN/IDLE/STEP switch is in RUN (Kg)]

ϕ1 $\overline{O11}$ $\overline{Ht}$

T1 $\overline{\overline{Kg}}$ ⟹ End (through Tp)

[End gates the preparation for the next instruction-note that O is cleared thus allowing the interrupt to always be recognized]

T0

Tp End⟹Block the interchange of A and B

[End $\overline{Or}$ would have gated this interchange]

End ($\overline{Ht}$ + Ip) ⇒ Go to ϕ0

[EXECUTE the instruction at the interrupt address]

8.5 Single-instruction interrupts

Some interrupt levels can be completely processed with one instruction (e.g. real time clock pulse). Hardware provisions have been made to handle these single-instruction interrupts:

1. A signal, Ij, will be held true by the interrupt logic whenever a single-instruction interrupt level is ACTIVE.
2. Ib will be automatically sent to the interrupt logic during the End phase of the single-instruction interrupt servicing subroutine.
3. No new interrupts will be allowed during the End phase of the single-instruction interrupt servicing subroutine.

Only the following opcodes will be meaningfully interpreted as single-instruction interrupt servicing subroutines:

EOM	(00/40)
POT, PIN	(10/14)
WOT, WIN	(11/15)
MPO	(16)
BMC, BRM	(37/77)
EXU	(72) - Only the allowable opcodes above may be EXECUTED

From the above, an earlier rule may be expanded:

ANY EXTERNAL INTERRUPT SIGNAL TO THE INTERRUPT LOGIC SHOULD BE BETWEEN 3 AND 4 CYCLES LONG.

End Ij

T1 Block any interrupt recognition

[End would normally have gated such recognition]

P→S

[Access the next instruction from the address in P]

T0 Ib (through Tp)

[Ib will gate the currently ACTIVE interrupt level of highest priority (i.e. the single-instruction interrupt level) to the INACTIVE state. Note that (since interrupt recognition at T1 time was blocked) no other interrupt levels are currently being gated (via Ie) from the WAITING state to the ACTIVE state]

Tp Ip ⟹ Block Ib

[If the single-instruction interrupt subroutine is an EXU, Ib will not be signalled until the End of the EXECUTED instruction]

$\overline{Or} \Longrightarrow A \longleftrightarrow B$

[Restore A and B-During operand assembly (ϕ0), $\overline{Or}$ caused A and B to be swapped]

M→C (Jm)

[i.e. next instruction→C]

(M0-M5)→O (Jm)

[i.e. next opcode→O]

M6→Lp (Jm)

[Set up Lp for ϕ0]

Set Fp

[Set Fp for ϕ0]

Clear O10

[Clear the memory parity error indicator-in case a transfer to IDLE is gated (see below)]

Clear O11

[Set up for a possible transfer to IDLE (see below)]

$(\overline{Ht} + Ip) \Rightarrow$ Set Cpe

[Check parity of the next instruction]

Go to $\phi 0$

[Perform the next instruction]

Ht $\overline{Ip} \Rightarrow$ Go to $\phi 1$ (more precisely, $\phi 1$ $\overline{O11}$ Ht)

[IDLE-note that the HALT flip-flop, Ht, must be set and this $(\overline{Ip})$ must not be the End phase of an EXU opcode]

MPO (16) is significantly altered when executed as a single-instruction interrupt servicing subroutine. In fact, the use of MPO in this manner is fairly well limited to the real time clock pulse interrupt level.

MPO
(16 and Ij)

$\phi 0$ Lp

T1

T0

Tp M→C (Jm)

[i.e. operand→C]

Set Cpe

[Check parity of the operand]

$\phi 4$

T1 C + 1→C (Ja, Pr, K11)

[The operand has been properly incremented]

Do not alter Fℓ

[The carry out of the adder, Ku2, usually goes to Fℓ]

Ku2 ⇒Clear Fp

[Thus $\overline{Fp}$ implies that the incremented operand now equals 0000_8 (Fp was left set by ϕ0 Lp)]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm-through Tp)

[The incremented operand will now be returned to memory]

$\overline{Fp}$ ⇒Ski (through ϕ7 Tp)

[An interrupt signal is sent to the real time clock sync interrupt level if the incremented, restored operand equals 0000_8]

Tp

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

T0

Tp

CHAPTER 9

Alert and Test I/O Equipment

9.1 Introduction

9.2 EOM opcode

9.3 SES opcode

9.1 Introduction

The EOM opcodes (00/40) provide one form of output from the main frame. Correspondingly, the SES opcodes (01/41) provide one form of input to the main frame. Both EOM and SES present a 16 bit control word (in fact, only the least significant 12 bits of the control word are presented at some I/O connectors) and various timing signals at the I/O connectors. Additionally, the response or lack of response to the control word will cause SES to set or clear Fℓ.

The effective address (E0-E14) provides 15 of the 16 bits in an EOM/SES control word. [This effective address is actually presented from O9-O11 and C.] The remaining bit is provided by the most significant bit of the opcode, Or. The 16 bits of the EOM/SES control word are labled

C1, C9, C10,..., C22, C23

at the I/O connectors. All of the above may be summarized as follows:

Control Word	Logical Source	Actual Source
C1	Or	Or
C9	E0	O9
C10	E1	O10
C11	E2	O11
C12	E3	C0
C13	E4	C1
C14	E5	C2
C15	E6	C3
C16	E7	C4
C17	E8	C5
C18	E9	C6
C19	E10	C7
C20	E11	C8
C21	E12	C9
C22	E13	C10

Control Word	Logical Source	Actual Source
C23	E14	C11

In practice, EOM's are generally used to alert an external device for an ensuing I/O operation (either via the standard I/O channel or via POT/PIN); SES's are generally used to test operating states and conditions within the external device.

Several general remarks may be made concerning the execution of either an EOM or an SES opcode:

1. The actual execution (i.e. presentation of the control word and timing pulses) occurs in $\phi 5$.
2. The actual execution will last exactly two contiguous machine cycles.
3. The actual execution will start at T1 $\overline{Ta}$ - and hence run through Tp Ta.
4. The actual execution will in no way be altered by any intervening TIMESHARE.

The above is effected by using Fp and Lp to divide $\phi 5$ into four subphases.

Fp	Lp	Subphase
1	0	This is a pre-execution subphase that is remained in until the first execution subphase, $\phi 5$ Fp Lp, may start at T1 $\overline{Ta}$ [$\phi 5$ Fp $\overline{Lp}$ is only entered if necessary]
1	1	This is the first execution subphase.
0	1	This is the second execution subphase.

Fp	Lp	Subphase
0	0	This is a post-execution subphase that is remained in until TIMESHARE is concluded and the next instruction may be accessed [$\phi 5\ \overline{Fp}\ \overline{Lp}$ is only entered if necessary]

9.2 EOM opcode

The timing signals

Q1, Q2

are as shown below for any EOM.

The control signal

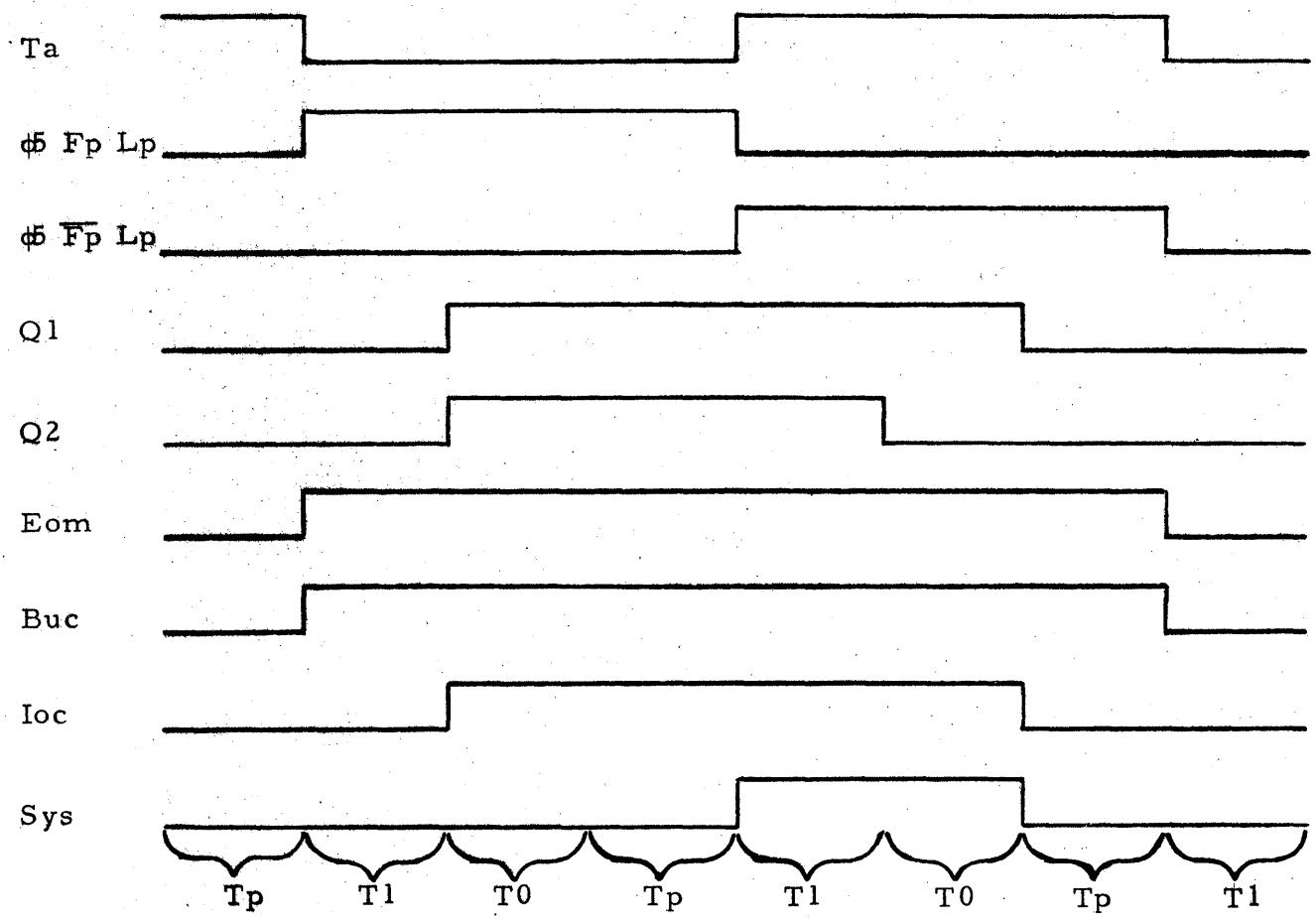
Eom

is as shown below for any EOM.

One of the control signals

Buc, Ioc, Sys

may also be signalling, as shown below, for a given EOM (depending on C1, C9, C10, and C11).



Some internal operations are effected via EOM. The associated control words have been chosen so as to allow direct scratch pad addressing. $\overline{C1}\ \overline{C10}\ \overline{C11}\ \overline{C17}$ signals one of these internal functions. In these cases only C19 through C23 need be further examined:

C19	C20	C21	C22	C23	Function
0	0	0	0	X	HALT
X	X	0	1	0	Clear Fℓ
X	X	1	0	0	Set Fℓ
X	X	1	1	0	Toggle Fℓ
X	1	X	X	0	Clear En
X	1	X	X	1	Set En
1	X	X	X	0	Clear Pct
1	X	X	X	1	Set Pct

EOM

(00/40)

ϕ0 Lp

T1

T0 (S0-S11)→C (Ja, Pr)

[The least significant 12 bits of the effective address will be presented from C]

Tp (Su0-Su2)→(O9-O11)

[The most significant 3 bits of the effective address will be presented from O9, O10, and O11]

Ta⇒Block the clearing of Lp

[i.e. Go to ϕ5 Fp Lp and begin the actual execution of the EOM]

$\overline{Ta}$ ⇒Allow Lp to clear

[i.e. Go to ϕ5 Fp $\overline{Lp}$ and wait one machine cycle before beginning the actual execution of the EOM]

ϕ5 Fp $\overline{Lp}$

T1

T0

Tp Ta⇒Set Lp

[i.e. Go to ϕ5 Fp Lp and begin the actual execution of the EOM]

ϕ5 Fp Lp

T1 Eom (through ϕ5 $\overline{Fp}$ Lp Tp)

[This control signal is not gated by any special EOM control word(s)]

Eom $\overline{C1}$ $\overline{C10}$ $\overline{C11}$⇒Buc (through ϕ5 $\overline{Fp}$ Lp Tp)

[Internally this is equivalent to Eom $\overline{Or}$ $\overline{O10}$ $\overline{O11}$]

T0 Q1 (through ϕ5 $\overline{Fp}$ Lp T0)

[This timing signal is not gated by any special opcode(s)]

Q2 (through ϕ5 $\overline{Fp}$ Lp T1)

[This timing signal is not gated by any special opcode(s)]

Eom $\overline{C1}$ $\overline{C10}$ C11⇒Ioc (through ϕ5 $\overline{Fp}$ Lp T0)

[Internally this is equivalent to Eom $\overline{Or}$ $\overline{O10}$ O11]

Tp Clear Fp

[i.e. Go to ϕ5 $\overline{Fp}$ Lp, where the actual execution of the EOM is concluded]

ϕ5 $\overline{Fp}$ Lp

T1 Eom $\overline{C9}$ C10 C11⇒Sys (through ϕ5 $\overline{Fp}$ Lp T0)

[Internally this is equivalent to Eom $\overline{O9}$ O10 O11]

$\overline{Ip}\ \overline{Tsq}$ ⇒End (through Tp)

[End gates the preparation for the next instruction- if a TIMESHARE is not about to begin ($\overline{Tsq}$) and this is not the automatic EOM of a FILL operation ($\overline{Ip}$)]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp ($\phi 5$ Tp $\overline{End}$ $\overline{Fp}$ $\overline{Ts}$)⇒FILL⇒55_8→O

[i.e. RIN→O]

Set Fp

[This initializes Fp for $\phi 2$]

Clear Lp

[This initializes Lp for $\phi 2$]

Clear Ip

[This initializes Ip for the End of RIN]

Go to $\phi 2$

[The RIN opcode sequence is entered at $\phi 2$]

Ts⇒Clear Lp

[i.e. Go to $\phi 5$ $\overline{Fp}$ $\overline{Lp}$ to await the conclusion of TIMESHARE]

$\phi 5$ $\overline{Fp}$ $\overline{Lp}$

T1 $\overline{Ip}\ \overline{Tsq}$⇒End (through Tp)

[End gates the preparation for the next instruction- if a TIMESHARE is not about to begin ($\overline{Tsq}$) and this is not the automatic EOM of a FILL operation ($\overline{Ip}$)]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp (ϕ5 Tp $\overline{\text{End}}$ $\overline{\text{Fp}}$ $\overline{\text{Ts}}$)⇒FILL⇒55_8→O

[i.e. RIN→O]

Set Fp

[This initializes Fp for ϕ2]

Leave Lp clear

[This initializes Lp for ϕ2]

Clear Ip

[This initializes Ip for the End of RIN]

Go to ϕ2

[The RIN opcode sequence is entered at ϕ2]

Ts⇒Do nothing

[i.e. Remain in ϕ5 $\overline{\text{Fp}}$ $\overline{\text{Lp}}$ awaiting the conclusion of TIMESHARE]

9.3 SES opcode

The timing signals

Q1, Q2

are as shown below for any SES.

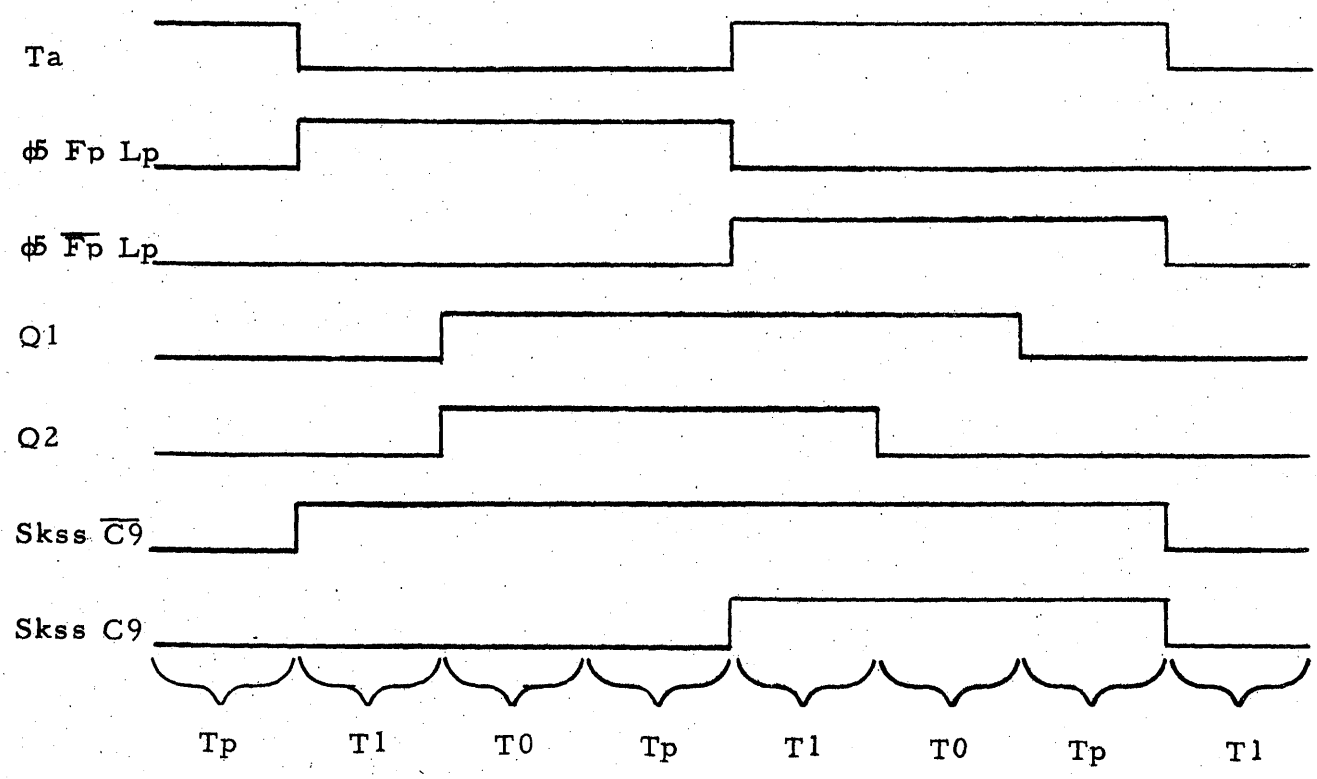
The control signal

Skss

is as shown below. However, note that there are two possible timings-depending upon C9 of the SES control word. The response to the control word (this response is stored in Fℓ) may come, externally, from either of

Sio, Ssc

depending on C1, C9, C10 and C11.


Ta
ϕ5 Fp Lp
ϕ5 $\overline{Fp}$ Lp
Q1
Q2
Skss $\overline{C9}$
Skss C9
Tp
T1
T0
Tp
T1
T0
Tp
T1

Some internal tests are made via SES. The associated control words have been chosen so as to allow direct scratch pad addressing. $\overline{C1}$ $\overline{C9}$ $\overline{C10}$ $\overline{C11}$ $\overline{C17}$ signals one of these internal tests. In these cases only C19 through C23 need be further examined:

C19	C20	C21	C22	C23	Test
X	X	1	0	0	BREAKPOINT #1 (Kb1→Fℓ)
X	X	1	0	1	BREAKPOINT #2 (Kb2→Fℓ)
X	X	1	1	0	BREAKPOINT #3 (Kb3→Fℓ)
X	X	1	1	1	BREAKPOINT #4 (Kb4→Fℓ)
X	1	X	X	X	Interrupt enable (En→Fℓ)
1	X	X	X	X	Program controlled trap (Pct→Fℓ)

SES

(01/41)

ϕ0 Lp

T1

T0 (S0-S11)→C (Ja, Pr)

[The least significant 12 bits of the effective address will be presented from C]

Tp (Su0-Su2)→(O9-O11)

[The most significant 3 bits of the effective address will be presented from O9, O10, and O11]

Ta⇒Block the clearing of Lp

[i.e. Go to ϕ5 Fp Lp and begin the actual execution of the SES]

$\overline{Ta}$⇒Allow Lp to clear

[i.e. Go to ϕ5 Fp $\overline{Lp}$ and wait one machine cycle before beginning the actual execution of the SES]

$\phi 5$ Fp $\overline{\text{Lp}}$

T1

T0

Tp Ta $\Rightarrow$ Set Lp

[i.e. Go to $\phi 5$ Fp Lp and begin the actual execution of the SES]

$\phi 5$ Fp Lp

T1 $\overline{C9} \Rightarrow$ Skss (through $\phi 5$ $\overline{\text{Fp}}$ Lp Tp)

[Internally this is equivalent to $\overline{O9}$]

T0 Q1 (through $\phi 5$ $\overline{\text{Fp}}$ Lp T0)

[This timing signal is not gated by any special opcode(s)]

Q2 (through $\phi 5$ $\overline{\text{Fp}}$ Lp T1)

[This timing signal is not gated by any special opcode(s)]

Tp Clear Fp

[i.e. Go to $\phi 5$ $\overline{\text{Fp}}$ Lp, where the actual execution of the SES is concluded]

$\phi 5$ $\overline{\text{Fp}}$ Lp

T1 C9 $\Rightarrow$ Skss (through $\phi 5$ $\overline{\text{Fp}}$ Lp Tp)

[Internally this is equivalent to O9]

$\overline{\text{Tsq}} \Rightarrow$ End (through Tp)

[End gates the preparation for the next instruction - if a TIMESHARE is not about to begin ($\overline{\text{Tsq}}$)]

T0

Tp Ses $\rightarrow$ Fℓ

[Ses is, logically, the response to the SES control word]

Ts $\Rightarrow$ Clear Lp

[i.e. Go to $\phi 5$ $\overline{\text{Fp}}$ $\overline{\text{Lp}}$ to await the conclusion of TIMESHARE]

$\phi 5\ \overline{Fp}\ \overline{Lp}$

T1 $\overline{Tsq}$ ⇒ End (through Tp)

[End gates the preparation for the next instruction- if a TIMESHARE is not about to begin ($\overline{Tsq}$)]

T0

Tp Ts ⇒ Do nothing

[i.e. Remain in $\phi 5\ \overline{Fp}\ \overline{Lp}$ awaiting the conclusion of TIMESHARE]

CHAPTER 10

Parallel I/O

10.1 Introduction

There are two independent I/O paths on all 92 computers:

1. Standard I/O Channel
    a. Character I/O
    b. Buffered
    c. Oriented towards standard peripherals
2. POT/PIN
    a. Full word I/O
    b. Unbuffered
    c. Oriented towards systems applications

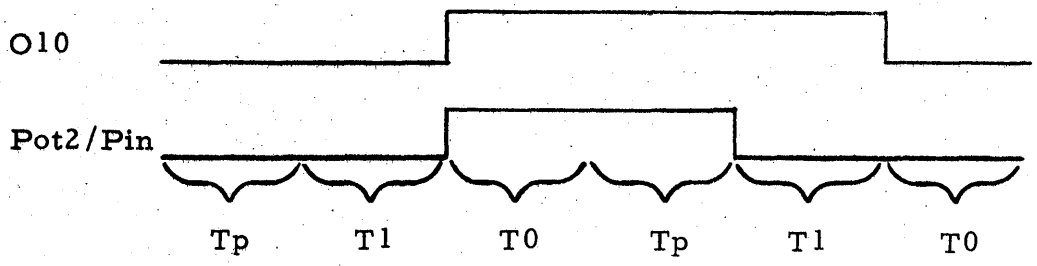
This chapter is concerned with the POT/PIN system. The POT (parallel output) operation is very similar to the PIN (parallel input) operation; hence the two operations will be discussed simultaneously.

The actual execution of the POT/PIN occurs during $\phi 2$. The logic will hang up in $\phi 2$ waiting for a READY signal from the external equipment. Upon receiving the READY signal, the main frame will send a reply (Pot2/Pin). If this is a POT operation, the output word will be available at the output connector (C12 through C23) while Pot2 is true. If this is a PIN operation, the input word will be sampled at the input connector (Cd12 through Cd23) on the trailing edge of Pin.

There are two speeds of POT/PIN available. There are two possible READY signals (Rt and Rtf) from the external equipment to the main frame. The external equipment will select the high speed mode (Rtf) or the low speed mode (Rt) by the READY signal which it sends.

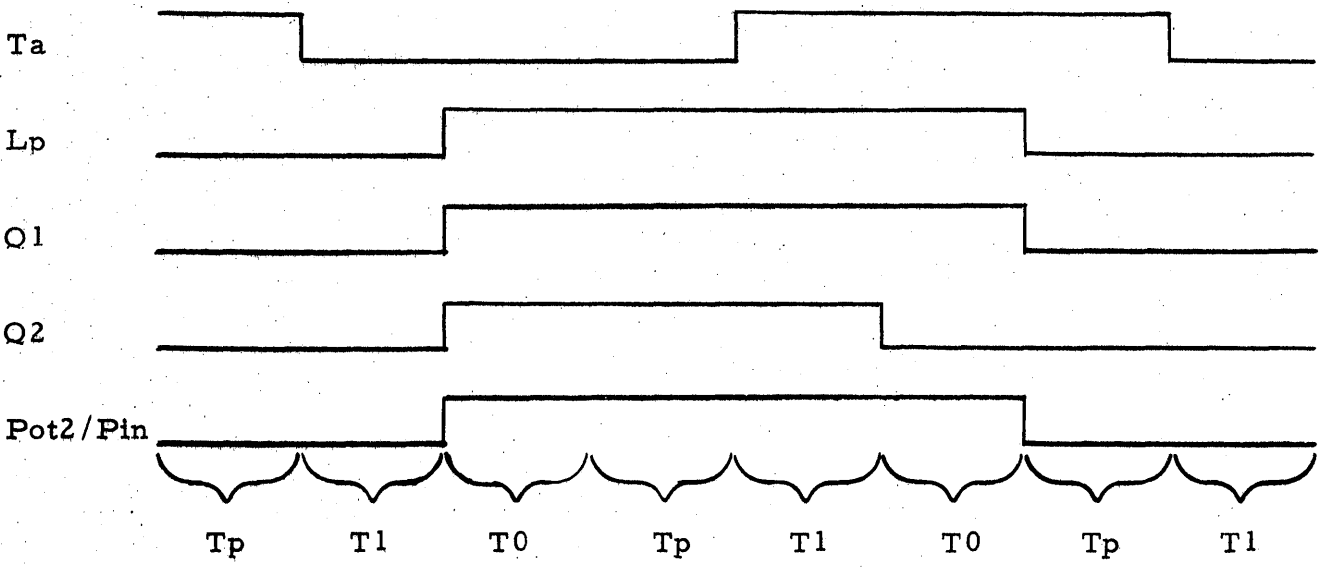
HIGH SPEED MODE

Rtf causes O10 to set at any T1 time. O10 then gates the execution of a POT/PIN operation in one machine cycle. The timing of the pertinent signals is:



LOW SPEED MODE

Rt causes Lp to set at any T1 $\overline{Ta}$ time. Lp then gates the execution of a POT/PIN operation in two machine cycles; naturally this execution will not be altered in any way by an intervening TIMESHARE. The timing of the pertinent signals is:



The POT (10) and PIN (14) opcodes have the block transfer equivalents BPO (50) and BPI (54). The logic stays in $\phi 2$ throughout the block transfer. As each word in the block is transferred, the A register will be decremented by one. The block transfer is terminated either by program control (the A register decrements through zero - thus the original contents of A equaled the block length minus one) or by external control (the external equipment signals BLOCK TERMINATE, Bt). In the high speed mode, BPO/BPI may (depending on Rtf) transfer a word every machine cycle. In the low speed mode, BPO/BPI may (depending on Rt) transfer a word every two machine cycles.

A POT (BPO) operation provides one additional signal, Potl, which is true as long as the logic remains in $\phi 2$. A PIN (BPI) operation provides one additional pulse, Rti, which is true for two clock times when the logic leaves $\phi 2$.

Some of the general purpose flip-flops used in $\phi 2$ of the execution of POT/BPO and PIN/BPI include:

Fp - During BPO operations Fp gates the next output word into C, the parity checking of this new output word, and the incrementing of S so that the following word in the output block may be accessed.

During BPI operations $\overline{Fp}$ gates the incrementing of S so that the current input word may be stored into the next word of the input block.

Lp - Lp will gate the transfer of one word in the low speed mode (Rt).

O9 - O9 gates the termination of POT/BPO or PIN/BPI. (i.e. O9 gates the exit from $\phi 2$). O9 may be set because:

1. A word has been transferred and the operation was not a block transfer (i.e. the opcode was POT or PIN)
2. All words in the defined block have been transferred (i.e. A has been decremented to 7777_8)
3. The external device has signalled the end of the block (via Bt)

O10 - O10 will gate the transfer of one word in the high speed mode (Rtf).

O11 - During BPI operations $\overline{\text{OIT}}$ signals that the last input word has been stored in memory.

10.2 Connectors

Following are the logic signals on the POT/PIN connectors. Note also the many EOM/SES signals on these connectors.

	POT CONNECTOR	PIN CONNECTOR	
1.	Pot1	Pin	1.
2.	Pot2		2.
3.	Ioc	$\overline{\text{Sio}}$	3.
4.	Buc		4.
5.	Sys		5.
6.	Eom		6.
7.	Q1		7.
8.	Q2	$\overline{\text{Rti}}$	8.
9.	$\overline{\text{Rtf}}$	$\overline{\text{Rtf}}$	9.
10.	$\overline{\text{Rti}}$		10.

	CONNECTOR	CONNECTOR	
11.	$\overline{Bt}$		11.
12.	Q2		12.
13.	Skss	Skss	13.
14.	$\overline{Sio}$		14.
15.	$\overline{Ssc}$	$\overline{Ssc}$	15.
16.	$\overline{Rt}$	$\overline{Rt}$	16.
17.	St		17.
18.	C17	$\overline{Bt}$	18.
19.			19.
20.			20.
21.	C1		21.
22.			22.
23.			23.
24.			24.
25.			25.
26.			26.
27.			27.
28.			28.
29.	C9		29.
30	C10		30.
31.	C11		31.
32.	C12	$\overline{Cd12}$	32
33.	C13	$\overline{Cd13}$	33.
34.	C14	$\overline{Cd14}$	34.
35.	C15	$\overline{Cd15}$	35.
36.	C16	$\overline{Cd16}$	36.
37.	C17	$\overline{Cd17}$	37.
38.	C18	$\overline{Cd18}$	38.

	POT CONNECTOR	PIN CONNECTOR	
39.	C19	$\overline{Cd19}$	39.
40.	C20	$\overline{Cd20}$	40.
41.	C21	$\overline{Cd21}$	41.
42.	C22	$\overline{Cd22}$	42.
43.	C23	$\overline{Cd23}$	43.

10.3 POT/BPO opcodes

POT/BPO - LOW SPEED

(10/50)

$\phi 0$ Lp

T1

T0

Tp — Set Cpe

[Check parity of the first output word]

NOTE: Fp is set; O9, O10, O11, and Lp are cleared; Go to $\phi 2$ $\overline{Ta}$ or $\phi 2$ Ta - depending on Ta.

$\phi 2$ $\overline{Ta}$ — Potl (through $\phi 2$)

[Potl is true throughout a POT/BPO operation]

T1 — Fp $\overline{Ts} \Rightarrow M \rightarrow C$ (Jm)

[This takes the next (first) output word to C]

S + 1→S (Ja, Pr, K11)

[Thus the following word in the output block will be accessed]

Clear Fp

[The logic is now ready to output a word from C]

Rt $\overline{O9}$ ($\overline{Fp}$ + $\overline{Ts}$) ⇒ Set Lp

[The low speed READY signal, Rt, will be recognized if the logic is not gated to terminate the output ($\overline{O9}$) and either the C register already has the next output word ($\overline{Fp}$) or this next output word is currently being gated to C ($\overline{Ts}$)]

T0 Lp ⇒ Pot2 (through T0 Ta)

[Pot2 indicates, to the external equipment, that the data lines (C12-C23) may be strobed]

Tp Lp BPO ⇒ A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transmitted]

Lp $\overline{Ku2}$ ⇒ Set O9

[The POT/BPO operation will be concluded following this transmission either because it was a single word transmission (POT ⇒ $\overline{Ku2}$) or because the BPO block length has been reached (decremented A = 7777_8 ⇒ $\overline{Ku2}$).]

Fp $\overline{O9}$ $\overline{Ts}$ ⇒ Set Cpe

[Check parity of the next word in the block]

($\overline{O9}$ + Lp) ⇒ Stay in ϕ2

[The transmission is not completed]

O9 $\overline{Lp}$ ⇒ Go to ϕ7

[The transmission is complete]

ϕ2 Ta

T1 Fp $\overline{Ts}$ ⇒ M→C

[This takes the next (first) output word to C]

S + 1→S (Ja, Pr, K11)

[Thus the following word in the output block will be accessed]

Clear Fp

[The logic is now ready to output a word from C]

Lp⟹Set Fp

[Fp gates the preparation for the next output word]

T0 Clear Lp

[This concludes the transmission of a data word]

Tp Fp $\overline{O9}$ $\overline{Ts}$ ⟹Set Cpe

[Check parity of the next word in the block]

$\overline{O9}$ ⟹Stay in ϕ2

[The transmission is not completed]

O9 ⟹Go to ϕ7

[The transmission is complete]

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp

POT/BPO - HIGH SPEED

(10/50)

ϕ0 Lp

T1

T0

Tp Set Cpe

[Check parity of the first output word]

NOTE: Fp is set; O9, O10, O11, and Lp are cleared

ϕ2 Pot1 (through ϕ2)

[Pot1 is true throughout a POT/BPO operation]

T1 Fp $\overline{Ts} \Rightarrow M \rightarrow C$ (Jm)

[This takes the next (first) output word to C]

S + 1→S (Ja, Pr, K11)

[Thus the following word in the (possible) output block will be accessed]

Clear Fp

[The logic is now ready to output a word from C]

Clear O10

[This concludes the transmission of a data word]

Rtf $\overline{O9}$ $\overline{Tsq}$ ⇒Set O10

[The high speed READY signal, Rtf, will be recognized if the logic is not gated to terminate the output ($\overline{O9}$) and the next word in the (possible) block may be accessed at the conclusion of this word transmission ($\overline{Tsq}$)]

T0 O10 ⇒Pot2 (through Tp)

[Pot2 indicates, to the external equipment, that the data lines (C12-C23) may be strobed]

Set Fp

[Fp gates the preparation for the next output word]

Tp O10 BPO ⇒A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transmitted]

O10 $\overline{Ku2}$ ⇒Set O9

[The POT/BPO operation will be concluded following this transmission either because it was a single word transmission (POT ⇒$\overline{Ku2}$) or because the BPO block length has been reached (decremented A = 7777_8 ⇒ $\overline{Ku2}$)]

$\overline{O9}$ O10 Ku2 ⇒Set Cpe

[Check parity of the next word in the block]

$(\overline{O9} + O10) \Longrightarrow$Stay in $\phi 2$

[The transmission is not completed]

$O9\ \overline{O10} \Longrightarrow$Go to $\phi 7$

[The transmission is complete]

$\phi 7$

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp

10.4 PIN/BPI opcodes

PIN/BPI - LOW SPEED
(14/54)

$\phi 0$ Lp

T1

T0

Tp NOTE: Fp is set; O9, O10, O11, and Lp are cleared; Go to $\phi 2\ \overline{Ta}$ or $\phi 2$ Ta - depending on Ta.

$\phi 2\ \overline{Ta}$ Mw

[Every memory cycle in $\phi 2$ will be a write cycle]

C→M (Jm)

[The input word in C will be written into memory]

T1 Rt $\overline{O9}$ $(\overline{O11} + \overline{Tsq}) \Longrightarrow$Set Lp

[The low speed READY signal, Rt, will be recognized if the logic is not gated to terminate the output ($\overline{O9}$) and either the previous input word has already been stored in memory ($\overline{O11}$) or this previous input word may be stored during the ensuing memory reference ($\overline{Tsq}$).]

T0 Lp ⇒ Pin (through T0 Ta)

[Pin indicates, to the external equipment, that the data lines (Cd12-Cd23) will be strobed (on the trailing edge of Pin)]

$\overline{Ts}$ ⇒ Clear O11

[The current memory reference is storing the last input word]

Tp Lp BPI ⇒ A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transmitted]

Lp $\overline{Ku2}$ ⇒ Set O9

[The PIN/BPI operation will be concluded following this transmission either because it was a single word transmission (PIN ⇒ $\overline{Ku2}$) or because the BPI block length has been reached (decremented A = 7777_8 ⇒ $\overline{Ku2}$)]

($\overline{O9}$ + Lp) ⇒ Stay in ϕ2

[The transmission is not completed]

O9 $\overline{Lp}$ ⇒ Go to ϕ4

[The transmission is complete]

ϕ2 Ta Mw

[Every memory cycle in ϕ2 will be a write cycle]

C→M (Jm)

[The input word in C will be written into memory]

T1 Lp $\overline{Fp}$ ⇒ S + 1→S (Ja, Pr, K11)

[The address within the data block is incremented to store the currently incoming data word]

Lp ⇒ Clear Fp

[$\overline{Fp}$ gates the increment of S (see above) before storing every input word except the first one]

T0 $Lp \Rightarrow Cd \rightarrow C$

[The input word is gated into C]

$\overline{Ts}\ \overline{Lp} \Rightarrow$ Clear O11

[The current memory reference is storing the last input word]

$Lp \Rightarrow$ Set O11

[A new input word has been gated into C and must be stored in memory - the current memory reference is storing unpredictable results since C was changed in the middle of the memory cycle]

Clear Lp

[This concludes the transmission of a data word]

Tp $\overline{O9} \Rightarrow$ Stay in $\phi 2$

[The transmission is not completed]

$O9 \Rightarrow$ Go to $\phi 4$

[The transmission is complete]

$\phi 4$

T1 $C \rightarrow C$ (Ja, Pr)

[This is a hardware quirk]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

$C \rightarrow M$ (Jm-through Tp)

[The last input word will be written into memory]

Rti (through Tp)

[Rti indicates, to the external equipment, the completion of a PIN/BPI operation]

Tp

$\phi 7$

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp

PIN/BPI - HIGH SPEED

(14/54)

ϕ0 Lp

T1

T0

Tp NOTE: Fp is set; O9, O10, O11, and Lp are cleared.

ϕ2 Mw

[Every memory cycle in ϕ2 will be a write cycle]

C→M (Jm)

[The input word in C will be written into memory]

T1 O10 $\overline{Fp}$ ⇒S + 1→S (Ja, Pr, K11)

[The address within the data block is incremented to store the newly strobed input word]

O10 ⇒Clear Fp

[$\overline{Fp}$ gates the increment of S (see above) before storing every input word except the first one]

Clear O10

[This concludes the transmission of a data word]

Rtf $\overline{O9}$ $\overline{Tsq}$ ⇒Set O10

[The high speed READY signal, Rtf, will be recognized if the logic is not gated to terminate the output ($\overline{O9}$) and the previous input word may be stored during the ensuing memory reference ($\overline{Tsq}$)]

T0 O10 ⇒Pin (through Tp)

[Pin indicates, to the external equipment, that the data lines (Cd12-Cd23) will be strobed (on the trailing edge of Pin)]

Tp O10 ⇒Cd→C

[The input word is gated into C]

O10 BPI ⇒A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transmitted]

O10 $\overline{Ku2}$ ⇒Set O9

[The PIN/BPI operation will be concluded following this transmission either because it was a single word transmission (PIN⇒$\overline{Ku2}$) or because the BPI block length has been reached (decremented A = 7777_8⇒ $\overline{Ku2}$)]

($\overline{O9}$ + O10)⇒Stay in $\phi 2$

[The transmission is not completed]

O9 $\overline{O10}$⇒Go to $\phi 4$

[The transmission is complete]

$\phi 4$

T1 C→C (Ja, Pr)

[This is a hardware quirk]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm-through Tp)

[The last input word is (unnecessarily) re-written into memory]

Rti (through Tp)

[Rti indicates, to the external equipment, the completion of a PIN/BPI operation]

Tp

ϕ7	
T1	End (through Tp)
	[End gates the preparation for the next instruction]
	Block any interrupt recognition
	[End would normally have gated such recognition]
T0	
Tp	

CHAPTER 11

Standard I/O Channel

11.1 Initialization

11.2 Character transmission and precessing

11.3 Parity

11.4 Channel error

11.5 Termination

11.6 Channel tests

11.7 Interrupts

11.8 Mag-tape SCAN

11.9 Interlace

11.10 Connectors

11.11 Channel timing charts

11.12 Channel opcodes

11.13 WOT/ROT opcodes

11.14 WIN/RIN opcodes

11.1 Initialization

The standard I/O channel operates, basically, in the following manner. A buffer control ($\overline{C1}\ \overline{C10}\ \overline{C11} \Rightarrow$Buc) EOM with C17 true will initialize both the channel and the selected peripheral for an I/O operation. The initializing of the channel is performed in two stages:

1. Wc clears the channel
2. Ws gates the set-up of the channel according to the EOM control word:
    a. C13 is examined on output operations to determine if leader is desired. $\overline{C13}$ gates leader-basically by forcing an all-zero (including zero parity) output character before transmitting the program-defined output characters.
    b. C16, which defines 2 character/word mode, is "permanently" stored in Wt and used to initialize the character counter, W8.
    c. C18-C23 are stored in the unit address register, W9-W14. Note that the most significant bit of the unit address defines the direction of the transmission (i.e. $\overline{C18} = \overline{W9} \Rightarrow$ INPUT; $C18 = W9 \Rightarrow$ OUTPUT).

11.2 Character transmission and precessing

After initialization the transmission of I/O characters may begin:
The channel contains a 12-bit buffer register, Wr0-Wr11, and a 12-bit character register, R1-R12. All input characters are strobed from the input character lines, Zw1-Zw12, into R. All output characters are presented to the output character lines, Rw1-Rw12, from R. The Wr register acts as a one word buffer between the main frame and the

peripheral. During input operations, characters are assembled in Wr until the defined computer word is complete (1 or 2 characters/word). $\overline{Wf}$ signals that Wr is full. The main frame must then store this word in memory; meanwhile, however, the channel could be strobing another input character into R. During output operations, characters are disassembled from Wr until the defined computer word is exausted (1 or 2 characters/word). $\overline{Wf}$ signals that Wr is empty. The main frame must then load Wr with another word from memory; meanwhile, however, the channel is presenting the last, disassembled output character from R.

The actual transmission of I/O characters is controlled by character clocks, Ecw, from the peripherals. The character clock is syncronized (and voltage spikes eliminated) within the channel by two flip-flops, W6 and W5.

$\overline{W6}\ \overline{W5}$ - The channel is awaiting a character clock. If this is an output operation, a character is currently being presented on the output lines (from R)

W6 $\overline{W5}$ - The channel has recognized the character clock. If this is an output operation, a character is currently being presented on the output lines (from R). If this is an input operation, the input lines are being strobed (into R)

W6 W5 - The channel has recognized the dropping of the character clock. If this is an output operation, the channel is attempting to disassemble the next output character (from Wr) for presentation (the current contents of the output lines are unpredictable).

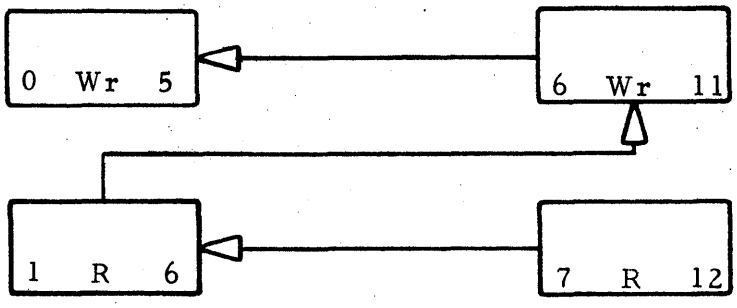
If this is an input operation, the channel is attempting to assemble the just-received input character (into Wr).

If this disassembly/assembly can be effected (Wf) the channel will return to the $\overline{W6}$ $\overline{W5}$ state.

$\overline{W6}$ W5 - The channel could not effect a disassembly/assembly ($\overline{Wf}$). The channel must wait, in this state, until the main frame presents (to Wr) a new data word (for the output operation) or stores (from Wr) the assembled data word (from the input operation). This action by the main frame (signalled by Wx) will then allow the disassembly/assembly to be effected (Wx will set Wf) and the channel will return to the $\overline{W6}$ $\overline{W5}$ state. [During output operations, all output lines (including parity) will be held clear while the channel remains in this state.]

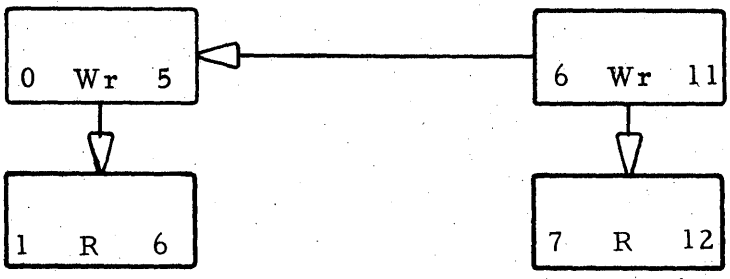
The disassembly/assembly of characters between Wr and R has been given a special name, precessing. Precessing is gated by W4. Precessing is accomplished by an interchange of Wr and R. The actual interchange effected depends on the operation (input or output) being performed.

During input operations, precessing gates the following shift paths:



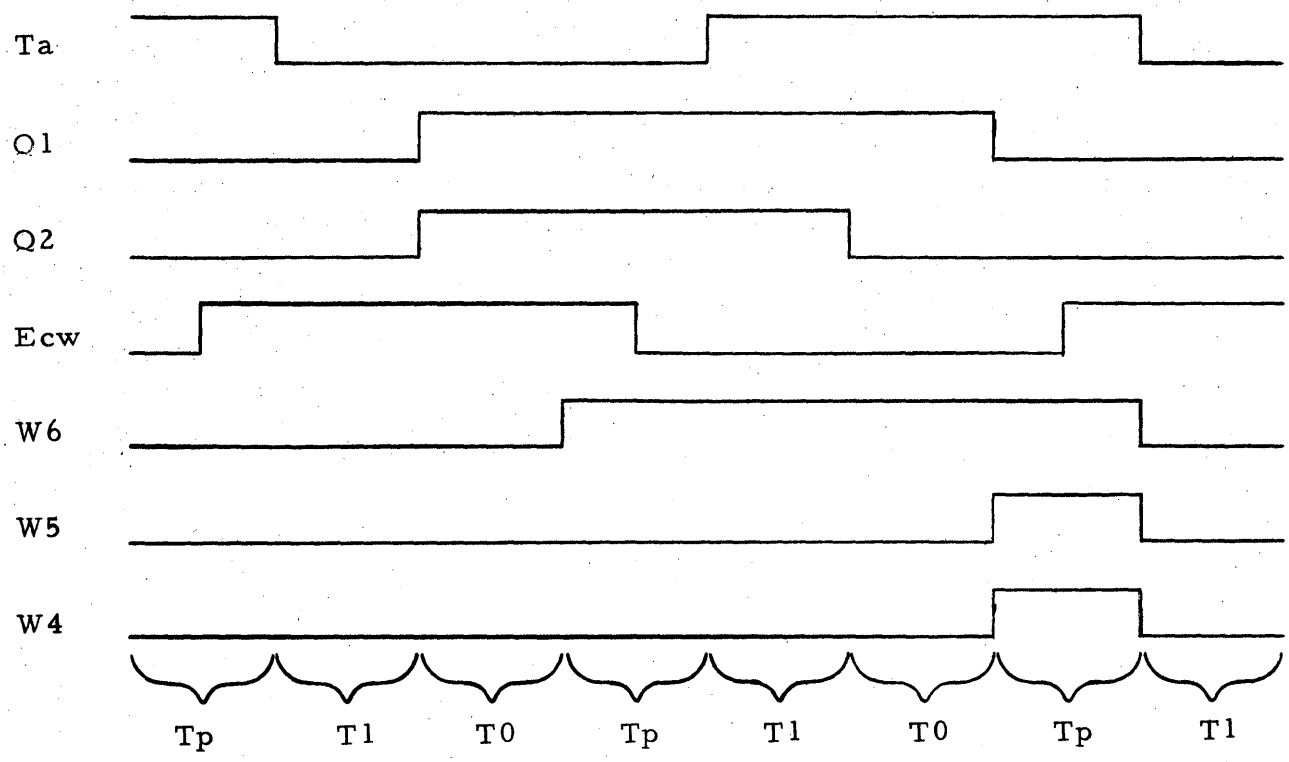
where the 1 character/word mode must always effect two precessing steps (on two contiguous clock times).

During output operations, precessing gates the following shift paths:



where there is no difference between the 1 character/word mode and the 2 character/word mode.

Like most other I/O operations, two machine cycles are (minimally) needed to complete one character transmission. The timing of the pertinent signals is:



11.3 Parity

Thus far, no logical differences have been noted between 6-bit character transmission and 12-bit character transmission. The only programming difference is that (naturally) all 12- bit character transmissions must be done in the 1 character/word mode (6-bit character transmissions may be done in either the 1 character/word mode or the 2 character/word mode). The only logical difference is in the handling of the parity- which will now be discussed.

All output characters include an odd parity bit. There are actually two parity bits available at the output connectors:

Rwp - Rwp will be true if R1-R6 contain an even number of bits. R1-R6 and Rwp constitute a 6-bit character.

Rwe - Rwe will be true if R1-R12 contain an even number of bits. R1-R12 and Rwe constitute a 12-bit character.

All input characters include a parity bit, Zwp. This parity bit is strobed into Rp. Then, when the character is precessed, the total number of bits of R1-R12 and Rp is checked. If this total number of bits is even, a channel parity error has occurred (and the channel error flip-flop, We, will be set). There need be no difference between 6-bit input characters (Zw1-Zw6 and Zwp) and 12- bit input characters (Zw1-Zw12 and Zwp), since 6-bit input peripherals will not attempt to alter Zw7-Zw12 (and all input lines are normally false).

11.4 Channel error

During channel operation, the channel error flip-flop (We) may be set

in three possible way:

1. The peripheral can detect an error and signal the channel (via Wes).
2. The channel can detect a parity error in an input character. [The peripheral can signal the channel (via $\overline{Np}$) to delete the parity checking of input characters.]
3. The channel can detect a RATE error if an incoming character clock (Ecw T0 $\overline{Ta}$) can not be processed (because the channel is in the $\overline{W6}$ W5 state).

11.5 Termination

An I/O channel operation may be terminated by the peripheral. The peripheral halt signal, Whs, is received in the channel halt flip-flop, Wh. Wh will then gate a disconnect (i.e. the unit address will be cleared-no peripheral will have a unit address of 00_8) and the I/O operation is completed-as far as the peripheral is concerned. The main frame may still have input characters (in Wr- and even R) to store in memory.

During a paper tape input operation, the I/O channel actually generates the halt signal, Wph. Wph will also set Wh and the termination of the operation will be as above. This paper tape halt signal, Wph, is generated when an all-zero input character (including zero parity) is received into R. This all-zero, halt character is not further treated as an input character (i.e. unlike the meaningful data characters, it will not be precessed into Wr).

An I/O channel operation may also be terminated by program. An input/output control ($\overline{C1}$ $\overline{C10}$ C11$\Rightarrow$Ioc) EOM with

C13 = 1

C17 = 1

C19 = 0

C20 = 0

C21 = 0

C22 = 0

C23 = 0

will effect this program termination (TIP ⇒Terminate input;

TOP ⇒Terminate output)

During the initialization for an input operation, the terminate input flip-flop (Wtr) was cleared. When the program terminates an input operation, Wtr will be set. Wtr will hold Wf set. This means that the Wr register will never again appear "full". Input characters will be received normally. But these characters will be shuffled through Wr and lost; the main frame will not be gated to store any more data words. When a halt signal (Whs or Wph) sets Wh, the channel will conclude its operation normally.

During the initialization for an output operation, the terminate output flip-flop (W0) was set. When the program terminates an output operation, W0 will be cleared. When the last remaining character in Wr has been transmitted, the channel will enter the $\overline{W6}$ W5 $\overline{W0}$ state. If the channel is not doing a mag-tape operation, the $\overline{W6}$ W5 $\overline{W0}$ state will directly set Wh and the channel will conclude its operation normally. If the channel is doing a mag-tape operation, the channel will remain in the $\overline{W6}$ W5 $\overline{W0}$ state until the mag-tape sets Wh (via Whs). The channel will then conclude its operation normally. [The W0 flip-flop has a different function during input operations. W0 is cleared during the initialization for an input operation.

W0 will be set (and remain set) when one computer word of input characters (1 or 2 characters as defined in the initializing EOM) has been received (in R) by the channel.]

All termination sequences (both input and output - both peripheral initiated and program initiated) will clear W0 at some time in the sequence. $\overline{W0}$ gates the conclusion of any termination sequence:

1. $\overline{W0}$ will block any TIMESHARE requests by the interlace logic.
2. $\overline{W0}$ will block any I1w interrupt signals.
3. $\overline{W0}$ will block any action by a WOT/ROT or WIN/RIN opcode.
4. $\overline{W0}$ will gate the I2w interrupt signal (which must be 2 cycles long, since Ti may remain true throughout any one given cycle).

11.6 Channel tests

Two conditions in the channel may be program tested. A buffer control ($\overline{C1}$ $\overline{C9}$ $\overline{C10}$ $\overline{C11}$) SES with C17 true will effect these test:

1. The channel error flip-flop, We, is tested by also having C2 true. Then

$$We \Longrightarrow \text{Set } F\ell$$

$$\overline{We} \Longrightarrow \text{Clear } F\ell$$

2. The current operating condition of the channel (ACTIVE or INACTIVE) is tested by also having C0 true. Then

$$\text{INACTIVE} \Rightarrow \text{Set } F\ell$$

$$\text{ACTIVE} \Rightarrow \text{Clear } F\ell$$

The channel is INACTIVE only if the unit address is clear (i.e. W9-W14 = 00_8) and no input characters are remaining to be stored following the termination of an input operation.

11.7 Interrupts

Two interrupt levels, with arms, are available for use in conjunction with an I/O operation.

The I1w interrupt (armed by setting Aiw1) signals that the Wr register is full (during input operations) or empty (during output operations).

The I2w interrupt (armed by setting Aiw2) signals that the channel has gone from the ACTIVE state to the INACTIVE state.

These interrupts are armed and disarmed by an input/output control ($\overline{C1}\ \overline{C10}\ C11 \Longrightarrow$Ioc) EOM with

C14 = 1

C17 = 1

C19 = 0

C20 = 0

C21 = 0

C22 = 0

C23 = 0

C15 will be copied into Aiw2; C16 will be copied into Aiw1.

11.8 Mag-tape SCAN

The I/O channel participates in the execution of one, very special peripheral function: A mag-tape SCAN. When SCANNING, Wf is blocked from resetting. This means that input characters are received normally into R but are shuffled through Wr and lost. The main frame will not be gated to store any input words. When the tape unit reaches the end of the tape record, the mag-tape gap signal (Mtg) will become true. This will force Wf to clear. The main frame

is now gated to store a data word (the last two characters in the tape record). [An Ilw interrupt will also (if armed) be generated when Mtg clears Wf.]

SCANNING is only allowed in the 2 character/word mode. The character counter, W8, has long since gone to zero ($\overline{W8}$) when Mtg first becomes true. When the main frame stores the data word (the last two characters in the tape record), the character count will be reinitialized (this means, in the 2 character/word mode, that W8 will be set). W8 will be used to block Mtg (which is still true) from again clearing Wf. [If Wf were again cleared, another Ilw interrupt would be generated (if armed).]

A mag-tape READ operation may be converted to a SCAN operation at any point in the tape block. This change is effected by an input/output control ($\overline{C1}$ $\overline{C10}$ C11 $\Rightarrow$Ioc) EOM with

C12 = 1

C17 = 1

C19 = 0

C20 = 0

C21 = 0

C22 = 0

C23 = 0

11.9 Interlace

An interlace option is available for the I/O channel. Basically, the interlace will automatically service Wr (when Wf clears) by directly referencing memory (via TIMESHARE). During output operations, M will be gated directly to Wr (via Jm). During input operations, Wr will be gated directly to M (via Jm). The address of the memory reference is given by the Iwa register

in the interlace logic. The interlace also has a block length register, Iwℓ. After each interlace memory reference (signalled by the trailing edge of $\overline{\text{Dmc}}$ Ts), the interlace will increment Iwa by one and decrement Iwℓ by one. [Note that the interlace will never TIMESHARE two contiguous memory cycles (the interlace logic will gate the set of Wf too late in the TIMESHARED memory cycle for any precessing during that memory cycle). Thus the interlace may perform the increment and decrement at leisure.]

The interlace informs the main frame of its current state by three signals:

1. Ewl + Ew2 - The interlace has been alerted to receive or is in the process of receiving the starting address (for Iwa) and the block length (for Iwℓ - this is actually the block length minus one).
   [The interlace is alerted for this set-up by any buffer control ($\overline{C1}$ $\overline{C10}$ $\overline{C11}$ ⇒ Buc) or input/output control ($\overline{C1}$ $\overline{C10}$ C11 ⇒ Ioc) EOM which has C9 true. The actual set-up is accomplished by three POT's to the interlace logic.]
2. Iw - The interlace has been set-up and is ready to service Wr, by TIMESHARES, as needed.
3. Iwf - The block length count, Iwℓ, has been decremented to 7777_8. This means that the originally given block length has been reached.

When the block length has been reached (as signalled by Iwf), all further

memory references by the interlace logic (normally gated by Iw) must be blocked. One of two possible events will be gated by Iwf:

1. If Ilw is armed (i.e. Aiwl is set), an Ilw interrupt will occur. [The usual Ilw interrupts have been blocked by Ewl + Ew2 and Iw.]
2. If Ilw is not armed (i.e. Aiwl is reset), then an automatic termination (TIP or TOP) of the operation will occur. This means that: during input operations, Wtr will be set; during output operations, W0 will be cleared.

11.10 Connectors

Following are the logic signals on the channel connectors. There are three different connectors:

1. AUX - All peripherals-except mag-tape-plug into this connector.
2. MAG - All mag-tapes plug into this connector. It differs from the AUX connector only at pin 12.
3. EXT - All peripherals which transmit/receive characters of more than 6-bits must plug into this connector-as well as either AUX or MAG.

Note, also, the many EOM/SES signals on these connectors.

	AUX	MAG	EXT	
1.	$\overline{Zw1}$	$\overline{Zw1}$	R7	1.
2.	$\overline{Zw2}$	$\overline{Zw2}$	R8	2.
3.	$\overline{Zw3}$	$\overline{Zw3}$	R9	3.
4.	$\overline{Zw4}$	$\overline{Zw4}$	R10	4.
5.	$\overline{Zw5}$	$\overline{Zw5}$	R11	5.

	AUX	MAG	EXT	
6.	$\overline{Zw6}$	$\overline{Zw6}$	R12	6.
7.	$\overline{Zwp}$	$\overline{Zwp}$		7.
8.	$\overline{Ecw}$	$\overline{Ecw}$		8.
9.	$\overline{Whs}$	$\overline{Whs}$		9.
10.	$\overline{Sio}$	$\overline{Sio}$		10.
11.	Buc	Buc		11.
12.	Np	Mtg		12.
13.	$\overline{Wes}$	$\overline{Wes}$		13.
14.	W0	W0		14.
15.	W5	W5		15.
16.	W6	W6		16.
17.	Q2	Q2		17.
18.	Ioc	Ioc		18.
19.	W9	W9	Rwe	19.
20.	W10	W10		20.
21.	W11	W11		21.
22.	W12	W12		22.
23.	W13	W13		23.
24.	W14	W14		24.
25.	R1	R1	$\overline{Zw7}$	25.
26.	R2	R2	$\overline{Zw8}$	26.
27.	R3	R3	$\overline{Zw9}$	27.
28.	R4	R4	$\overline{Zw10}$	28.
29.	R5	R5	$\overline{Zw11}$	29.
30.	R6	R6	$\overline{Zw12}$	30.
31.	Rwp	Rwp		31.
32.	C12	C12		32.
33.	C13	C13		33.
34.	C14	C14		34.

	AUX	MAG	EXT	
35.	C15	C15		35.
36.	C16	C16		36.
37.	C17	C17		37.
38.	C18	C18		38.
39.	C19	C19		39.
40.	C20	C20		40.
41.	C21	C21		41.
42.	C22	C22		42.
43.	C23	C23		43.

11.11 Channel timing charts

Buc $\overline{Ta}$	Wc ⇒	Clear W0
		[W0 will be set when one computer word (1 or 2 characters) has been received (in R).]
		Clear W5
		Clear W6
		[The channel is initialized to accept character clocks]
		Clear W8
		[The character counter is cleared-see Ws below]
		Clear W9-W14
		[The unit address is cleared-see Ws below]
		Clear We
		[Any channel error indication is cleared]
		Clear Wh
		[Any channel halt gating is cleared]
		Clear Wt
		[The characters/word flip-flop is cleared-see Ws below]
		Clear Wtr
		[Any terminate input gating is cleared]
		Set Wf
		[Wf will be cleared each time that a computer word has been assembled (in Wr)]
T1		
T0		
Tp		

Buc Ta

T1

T0 $Ws \Rightarrow (C6\text{-}C11) \rightarrow (W9\text{-}W14)$

[The unit address is set-up from C18-C23]

$C4 \rightarrow Wt$

[The characters/word flip-flop is set-up from C16]

$C4 \rightarrow W8$

[The character counter is initialized from C16]

Tp

INPUT CHARACTER PROCESSING

{2 characters/word (Wt)-first character (W8)}

$\overline{Ta}$ Assume that W6 and W5 are clear-the logic is then ready to recognize a character clock, Ecw.

T1

T0 $\overline{W6}\ \overline{W5}\ \overline{W9} \Rightarrow$ Clear R

Clear Rp

[The character register is cleared]

$\overline{W5}$ Ecw $\Rightarrow$ Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp $W6\ \overline{W5} \Rightarrow (R + Zw) \rightarrow R$

$(Rp + Zwp) \rightarrow Rp$

[The input lines are logically OR'ed into the input character register]

•
•
•

Ta Reviewing conditions: W6 is set, W5 is reset, and the

input lines are being sampled every clock time (i.e. R + Zw→R, Rp + Zwp→Rp)-the logic is now ready to recognize the dropping of the character clock, Ecw.

T1 W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

T0 W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

W6 $\overline{Ecw}$ ⇒Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒Clear W6

[The logic will enter either the $\overline{W6}$ W5 state (if the precessing below could not be effected) or the $\overline{W6}$ $\overline{W5}$ state (if the precessing below could be effected)]

W5 Wf Tp ⇒W4 ⇒Clear W5

[i.e. Enter the $\overline{W6}$ $\overline{W5}$ state and allow new character clocks to be recognized]

Clear W8

[Decrement the character counter]

(R1-R6)→(Wr6-Wr11)

[Precess the character into Wr]

Pe12 ⇒Set We

[i.e. Check the parity of the input character]

{2 characters/word (Wt)-second character $\overline{W8}$}

$\overline{Ta}$ Assume that W6 and W5 are clear-the logic is now ready to recognize a character clock, Ecw.

T1

T0 $\overline{W5}$ $\overline{W6}$ $\overline{W9}$ ⇒Clear R

Clear Rp

[The character register is cleared]

$\overline{W5}$ Ecw ⇒Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

W6 $\overline{W5}$ $\overline{W8}$ ⇒Set W0

[During input operations, W0 signifies that enough characters have been received to assemble into a computer word]

•
•
•

Ta Reviewing conditions: W6 is set, W5 is reset, and the input lines are being sampled every clock time (i.e. R + Zw→R, Rp + Zwp→Rp)-the logic is now ready to recognize the dropping of the character clock, Ecw.

T1 W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

T0 W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

character register]

W6 $\overline{\text{Ecw}}$ ⇒Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒Clear W6

[The logic will enter the $\overline{\text{W6}}$ $\overline{\text{W5}}$ state (since the precessing below can always be effected)]

W5 Wf Tp ⇒W4 ⇒Clear W5

[i.e. Enter the $\overline{\text{W6}}$ $\overline{\text{W5}}$ state and allow new character clocks to be recognized]

(Wr6-Wr11)→(Wr0-Wr5)

(R1-R6)→(Wr6-Wr11)

[Precess the character into Wr; a computer word has now been assembled in Wr]

Pe12 ⇒Set We

[i.e. Check the parity of the input character]

Clear Wf

[$\overline{\text{Wf}}$ implies that Wr is full]

INPUT CHARACTER PROCESSING

{1 character/word ($\overline{\text{Wt}}$ $\overline{\text{W8}}$)}

$\overline{\text{Ta}}$ Assume that W6 and W5 are clear-the logic is now ready to recognize a character clock, Ecw.

T1

TC $\overline{\text{W5}}$ $\overline{\text{W6}}$ $\overline{\text{W9}}$ ⇒Clear R

Clear Rp

[The character register is cleared]

$\overline{W5}$ Ecw $\Rightarrow$Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp W6 $\overline{W5}$ $\Rightarrow$(R + Zw)$\rightarrow$R

(Rp + Zwp)$\rightarrow$Rp

[The input lines are logically OR'ed into the input character register]

W6 $\overline{W5}$ $\overline{W8}$ $\Rightarrow$Set W0

[During input operations, W0 signifies that enough characters have been received to assemble into a computer word]

•
•
•

Ta Reviewing conditions: W6 is set, W5 is reset, and the input lines are being sampled every clock time (i.e. R + Zw$\rightarrow$R, Rp + Zwp$\rightarrow$Rp) - the logic is now ready to recognize the dropping of the character clock, Ecw.

T1 W6 $\overline{W5}$ $\Rightarrow$(R + Zw)$\rightarrow$R

(Rp + Zwp)$\rightarrow$Rp

[The input lines are logically OR'ed into the input character register]

T0 W6 $\overline{W5}$ $\Rightarrow$(R + Zw)$\rightarrow$R

(Rp + Zwp)$\rightarrow$Rp

[The input lines are logically OR'ed into the input character register]

W6 $\overline{Ecw}$ $\Rightarrow$Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒ Clear W6

[The logic will enter either the $\overline{W6}$ W5 state (if the precessing below could not be effected) or the $\overline{W6}$ $\overline{W5}$ state (if the precessing below could be effected)]

W5 Wf Tp ⇒ W4 ⇒ Clear W5

[i.e. Enter the $\overline{W6}$ $\overline{W5}$ state and allow new character clocks to be recognized]

(R1-R6)→(Wr6-Wr11)

(R7-R12)→(R1-R6)

[Precess the character]

Pe12 ⇒ Set We

[i.e. Check the parity of the input character]

Set W8

[W8 will gate another precessing at the next clock time]

Clear Wf

[$\overline{Wf}$ implies that Wr is full]

T1 W8 $\overline{Wt}$ ⇒ (Wr6-Wr11)→(Wr0-Wr5)

(R1-R6)→(Wr6-Wr11)

[Precess the character again-a computer word has now been assembled in Wr]

Clear W8

[The precessing of the character is completed]

T0

Tp

$\overline{W9}$ $\overline{Wf}$ — $\overline{Wf}$ signifies that Wr is full. One more input character may be processed into R-but no further precessing (W4) is allowed (and thus no more character processing since W5 remains set) until Wf is set.

W0 $\overline{Wf}$ $\overline{Iw}$ Aiw1 ⇒Ilw

[Ilw is an interrupt signal to the interrupt logic. It signifies, during non-interlace operations ($\overline{Iw}$), that Wr contains an assembled computer word]

T1 W0 $\overline{Wf}$ Iw $\overline{Iwf}$ ⇒Tsq

[This requests a TIMESHARE so that the interlace logic (Iw) may store Wr in memory]

T0 $\overline{Dmc}$ Ts ⇒Mw (through T1)

[The interlace memory cycle will be a write cycle]

Wr→M (Jm-through T1)

[Wr will be written into memory]

Tp (WIN + RIN) ⇒Wr→C

[If the channel is being serviced directly by the program (either a WIN or RIN opcode), Wr will be taken to C (from whence it will be stored in memory)]

($\overline{Dmc}$ Ts + WIN + RIN) ⇒Wx ⇒Wt→W8

[Reinitialize the character counter]

Set Wf

[Wr has been emptied-allow precessing to proceed normally]

INPUT TERMINATION SEQUENCE

·Signal from the peripheral (Whs)

$\overline{Ta}$

T1 Wh ⇒Clear W9-W14

[Clear the unit address register]

Whs⇒Set Wh

[The halt signal initiates the sequence by setting Wh-note that the unit address register will not be cleared until two machine cycles later]

T0 Wh ⇒Block the possible set of W6

[i.e. Ignore further character clocks]

$\overline{W5}$ Wf Wh (W8 + $\overline{Wt}$ + mag tape scan)⇒Set Wtr

[During the input termination sequence, Wtr signifies that there are no more meaningful data words to be stored. Note that Wtr will hold Wf set]

Tp Wh ⇒Block the possible set of W0

[W6 $\overline{W5}$ $\overline{W8}$ would have gated this set]

Block the normal precess (W4) gating

[W5 Wf Tp would have gated a precess]

Wf W0 Wh Tp ⇒W4 ⇒Effect a normal precessing

[This will precess either a received input character (W5) or a dummy character ($\overline{W5}$) into Wr. In the latter case, parity checking of the (dummy) input character must be blocked]

Ta

T1

T0 $\overline{W5}$ Wf Wh (W8 + $\overline{Wt}$ + mag tape scan) ⇒Set Wtr

[During the input termination sequence, Wtr signifies that there are no more meaningful data words to be stored. Note that Wtr will hold Wf set]

Tp Wh ⇒Block the possible set of W0

[W6 $\overline{W5}$ $\overline{W8}$ would have gated this set]

Block the normal precess (W4) gating

[W5 Wf Tp would have gated a precess]

Wf W0 Wh Tp ⇒W4 ⇒Effect a normal precessing

[This will precess either a received input character (W5) or a dummy character ($\overline{W5}$) into Wr. In the latter case, parity checking of the (dummy) input character must be blocked]

NOTE: The logic will precess every Tp until Wf is cleared. The logic will then wait for the meaningful data word to be stored in memory. This sequence will be repeated until Wtr is set (see T0 time above).

Wh Wtr $\overline{Ta}$

T1 Clear W9-W14

[Clear the unit address register (if it is not already clear)]

Clear W0

[$\overline{W0}$ will gate the conclusion of the termination sequence]

T0 Wh ⇒Block the possible set of W6

[i.e. Ignore further character clocks]

Wh $\overline{W0}$ ⇒I2w (through T1 $\overline{Ta}$)

[I2w is an interrupt signal to the interrupt logic. It signifies that the channel is going from the ACTIVE state to the INACTIVE state]

Tp

Wh $\overline{W0}$ Ta

T1

T0

Tp

Wh $\overline{W0}$ $\overline{Ta}$

T1 (W9-W14 = 0) ⇒Wc ⇒Clear W6

Clear W5

[The logic is forced into the $\overline{W6}$ $\overline{W5}$ state]

Clear Wh

[This drops the I2w interrupt signal and concludes the termination sequence]

Clear W9-W14

[Clear the unit address register (if it is not already clear)]

T0

Tp

INPUT TERMINATION SEQUENCE

{All-zero paper tape input character}

Ta — Assume that a paper tape input unit ($\overline{W9}$ $\overline{W10}$ $\overline{W11}$ W12 $\overline{W13}$) is currently connected to the channel. Assume further that the character clock has been recognized-so that W6 is set, W5 is reset, and the input lines are being sampled every clock time (i.e. R + Zw→R, Rp + Zwp→Rp). The logic is now ready to recognize the dropping of the character clock, Ecw.

T1 — W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

T0 — W6 $\overline{W5}$ ⇒(R + Zw)→R

(Rp + Zwp)→Rp

[The input lines are logically OR'ed into the input character register]

W6 $\overline{Ecw}$ ⇒Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp — W5 ⇒Clear W6

[The logic will enter the $\overline{W6}$ $\overline{W5}$ state as the all-zero character will not be treated like a normal input character]

(R1-R8 = 0) $\overline{Rp}$ W5 ⇒ Wph ⇒ Clear W5

[i.e. Enter the $\overline{W6}$ $\overline{W5}$ state]

Block precessing

[The all-zero character is not considered a data character]

Set Wh

[It now appears, logically, as if the peripheral had sent a halt signal (Whs) -and the remainder of the termination sequence is entirely similiar to the Whs termination sequence (q.v.)]

INITIATE OUTPUT OPERATION

{With leader ($\overline{CI3}$)}

Buc $\overline{Ta}$ Wc ⇒ Clear W0

[But W0 will be set by Ws (see below)]

Clear W5

Clear W6

[The channel is initialized to accept character clocks]

Clear W8

[The character counter is cleared]

Clear W9-W14

[The unit address is cleared - see Ws below]

Clear We

[Any channel error indication is cleared]

Clear Wh

[Any channel halt gating is cleared]

Clear Wt

[The characters/word flip-flop is cleared-see Ws below]

Clear Wtr

[Any terminate input gating is cleared]

Set Wf

[But Wf will be cleared by Ws (see below)]

$\overline{W5}$ $\overline{W9}$ ⟹Clear R1-R12

[This sets up the all-zero output character (see Rp below) that signals leader to the peripheral]

T1

T0

Tp

Buc Ta

T1

T0 Ws ⟹(C6-C11)→(W9-W14)

[The unit address is set-up from C18-C23]

C4→Wt

[The characters/word flip-flop is set-up from C16]

Set W0

[$\overline{W0}$ will gate the output termination sequence]

Set Rp

[This forces the parities (Rwp and Rwe) of the all-zero leader character to also be zero]

Clear Wf

[Wf is cleared whenever there are no output

characters remaining in the buffer register (Wr)]

Tp

INTITATE OUTPUT OPERATION

{Without leader (C13)}

Buc $\overline{Ta}$ Wc⇒Clear W0

[But W0 will be set by Ws (see below)]

Clear W5

Clear W6

[But W5 will be set by Ws (see below) to force the $\overline{W6}$ W5 state]

Clear W8

[The character counter is cleared]

Clear W9-W14

[The unit address is cleared-see Ws below]

Clear We

[Any channel error indication is cleared]

Clear Wh

[Any channel halt gating is cleared]

Clear Wt

[The characters/word flip-flop is cleared-see Ws below]

Clear Wtr

[Any terminate input gating is cleared]

Set Wf

[But Wf will be cleared by Ws (see below)]

$\overline{W5}$ $\overline{W9}$ ⇒Clear R1-R12

[This sets up an all-zero output character (see Rp below) since a legitimate output character has not yet been provided (in Wr)]

T1

T0

Tp

Buc Ta

T1

T0 Ws ⟹ (C6-C11)→(W9-W14)

[The unit address is set-up from C18-C23]

C4→Wt

[The characters/word flip-flop is set-up from C16]

Set W0

[$\overline{W0}$ will gate the output termination sequence]

Set W5

[This forces the $\overline{W6}$ W5 state-which implies that there is not a legitimate output character in R]

Set Rp

[This forces the output parity lines (Rwp and Rwe) to zero, since there is no output character available (R1-R12 are also zero)]

Clear Wf

[Wf is cleared whenever there are no output characters remaining in the buffer register (Wr)]

Tp

OUTPUT CHARACTER PROCESSING

{2 characters/word (Wt)-first character $\overline{(W8)}$}

$\overline{Ta}$ Assume that W6 and W5 are clear and an output character is being presented (from R)-the logic is then ready to recognize a character clock, Ecw.

T1

T0 $\overline{W5}$ Ecw ⇒ Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp

•
•
•

Ta Reviewing conditions: W6 is set, W5 is reset, and an output character is being presented (from R)-the logic is now ready to recognize the dropping of the character clock, Ecw.

T1

T0 W6 $\overline{Ecw}$ ⇒ Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒ Clear W6

[The logic will enter the $\overline{W6}$ $\overline{W5}$ state (since the precessing below can always be effected)]

W5 Wf Tp ⇒ W4 ⇒ Clear W5

[i.e. Enter the $\overline{W6}$ $\overline{W5}$ state and allow new character clocks to be recognized]

(Wr0-Wr5) → (R1-R6)

[The next output character is precessed into R-and presented on the output lines]

Clear Wf

[$\overline{Wf}$ implies that Wr is empty]

OUTPUT CHARACTER PROCESSING

{2 characters/word (Wt)-second character (W8)}

$\overline{Ta}$ Assume that W6 and W5 are clear and an output character is being presented (from R) - the logic is then ready to recognize a character clock, Ecw.

T1

T0 $\overline{W5}$ Ecw ⇒Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp

⋮

$\overline{Ta}$ Reviewing conditions: W6 is set, W5 is reset, and an output character is being presented (from R)-the logic is now ready to recognize the dropping of the character clock, Ecw.

T1

T0 W6 $\overline{Ecw}$ ⇒Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒Clear W6

[The logic will enter either the $\overline{W6}$ W5 state (if the precessing below could not be effected) or the $\overline{W6}$ $\overline{W5}$ state (if the precessing below could be effected)]

W5 Wf Tp ⇒W4 ⇒Clear W5

[i.e. Enter the $\overline{W6}$ $\overline{W5}$ state and allow new character clocks to be recognized]

(Wr0-Wr5)→(R1-R6)

(Wr6-Wr11)→(Wr0-Wr5)

[The next output character is precessed into R-and presented on the output lines]

Clear Rp

[$\overline{Rp}$ will allow the correct parities to be presented on the output parity lines (Rwp and Rwe)]

Clear W8

[Decrement the character counter]

OUTPUT CHARACTER PROCESSING

{1 character /word ($\overline{Wt}$ $\overline{W8}$)}

$\overline{Ta}$ Assume that W6 and W5 are clear and an output character is being presented (from R)-the logic is then ready to recognize a character clock, Ecw.

T1

T0 $\overline{W5}$ Ecw ⇒ Set W6

[W6 $\overline{W5}$ signals that a character clock has been recognized]

Tp

•

•

•

Ta Reviewing conditions: W6 is set, W5 is reset, and an output character is being presented (from R)-the logic

is now ready to recognize the dropping of the character clock, Ecw.

T1

T0 W6 $\overline{\text{Ecw}}$ ⇒Set W5

[W6 W5 signals that the dropping of the character clock has been recognized]

Tp W5 ⇒Clear W6

[The logic will enter either the $\overline{\text{W6}}$ W5 state (if the precessing below could not be effected) or the $\overline{\text{W6}}$ $\overline{\text{W5}}$ state (if the precessing below could be effected)]

W5 Wf Tp ⇒W4 ⇒Clear W5

[i.e. Enter the $\overline{\text{W6}}$ $\overline{\text{W5}}$ state and allow new character clocks to be recognized]

(Wr0-Wr11)→(R1-R12)

[The next output character is precessed into R-and presented on the output lines]

Clear Rp

[$\overline{\text{Rp}}$ will allow the correct parities to be presented on the output parity lines (Rwp and Rwe)]

Clear Wf

[$\overline{\text{Wf}}$ implies that Wr is empty]

FETCHING AN OUTPUT WORD

W9 $\overline{\text{Wf}}$ $\overline{\text{Wf}}$ signifies that Wr is empty. One more output character may be processed from R-but no further precessing (W4)

is allowed (and thus no more character processing since W5 remains set) until Wf is set.

W5 $\overline{W6}$ W9 ⇒Clear R1-R12

Set Rp

[This holds all output lines-including the parities (Rwp and Rwe)-at zero]

W0 $\overline{Wf}$ $\overline{Iw}$ Aiwl ⇒Ilw

[Ilw is an interrupt signal to the interrupt logic. It signifies, during non-interlace operations ($\overline{Iw}$), that Wr contains no more characters for disassembly]

T1 W0 $\overline{Wf}$ Iw $\overline{Iwf}$ ⇒Tsq

[This requests a TIMESHARE so that the interlace logic (Iw) may fetch another output word (for Wr) from memory]

T0

Tp (WOT + ROT) ⇒C→Wr

[If the channel is being serviced directly by the program (either a WOT or a ROT opcode), Wr is reloaded from C]

($\overline{Dmc}$ Ts)⇒M→Wr (Jm)

[If the channel is being serviced by the interlace, Wr is reloaded directly from memory]

($\overline{Dmc}$ Ts + WOT + ROT) ⇒Wx⇒Wt→W8

[Reinitialize the character counter]

Set Wf

[Wr has been filled-allow precessing to proceed normally]

OUTPUT TERMINATION SEQUENCE

$\overline{Ta}$

T1 — Whs ⇒Set Wh

[The peripheral signal, Whs, initiates the output termination sequence]

$\overline{W0}$ $\overline{W6}$ W5 ($\overline{\text{mag tape operation}}$) ⇒Set Wh

[The output termination sequence may be initiated, under program control (TOP clears W0), when all available output characters have been transmitted ($\overline{W6}$ W5)]

T0 — Wh ⇒Block the possible set of W6

[i.e. Ignore further character clocks]

Tp — Wh W9 ⇒Block any precess (W4) gating

[W5 Wf Tp would have gated normal precessing; Wf W0 Wh Tp would have gated halt sequence precessing]

Ta

T1

T0

Tp — Wh W9 ⇒Block any precess (W4) gating

[W5 Wf Tp would have gated normal precessing; Wf W0 Wh Tp would have gated halt sequence precessing]

Wh $\overline{Ta}$

T1 — Wh ⇒Clear W9-W14

[Clear the unit address register]

Clear W0

[W0 is cleared (if it is not already clear). $\overline{W0}$ will gate the conclusion of the termination sequence]

T0 Wh ⇒ Block the possible set of W6

[i.e. Ignore further character clocks]

Wh $\overline{W0}$ ⇒ I2w (through T1 $\overline{Ta}$)

[I2w is an interrupt signal to the interrupt logic. It signifies that the channel is going from the ACTIVE state to the INACTIVE state]

Tp Wh $\overline{W0}$ ⇒ Block any precess (W4) gating

[This includes both normal precessing and halt sequence precessing]

Wh $\overline{W0}$ Ta

T1

T0

Tp Wh $\overline{W0}$ ⇒ Block any precess (W4) gating

[This includes both normal precessing and halt sequence precessing]

Wh $\overline{W0}$ $\overline{Ta}$

T1 (W9-W14 = 0) ⇒ Wc ⇒ Clear W6

Clear W5

[The logic is forced into the $\overline{W6}$ $\overline{W5}$ state]

Clear Wh

[This drops the I2w interrupt signal and concludes the termination sequence]

Clear W9-W14

[Clear the unit address register (if it is not already clear)]

T0

Tp

11.12 Channel opcodes

Two opcodes have been provided to directly service the I/O channel:

WOT (11) - WOT loads Wr with a data word when Wr becomes empty during an output operation.

WIN (15) - WIN stores Wr in memory when Wr becomes full during an input operation.

These two opcodes also have record transfer equivalents, ROT (51) and RIN (55). Throughout the record, the logic remains in $\phi 2$. Whenever Wr becomes empty/full, the logic will reload/store it. As each word in the record is transferred, the A register will be decremented by one. The record transfer is terminated either by program control (the A register decrements through zero-thus the original contents of A equalled the record length minus one) or by peripheral control (the peripheral signals a halt via Whs; the peripheral is disconnected by the channel; and no input characters remain to be stored).

Some of the general purpose flip-flops used in $\phi 2$ of the execution of WOT/ROT and WIN/RIN include:

Fp - During ROT operations Fp gates the next output word into C and the incrementing of S so that the following word in the output record may be accessed.

During RIN operations $\overline{Fp}$ gates the incrementing of S so that the current input word may be stored (from C) into the next word of the input record.

O9 - O9 gates the termination of WOT/ROT or WIN/RIN (i.e. O9 gates the exit from $\phi 2$). O9 may be set because:

1. A word has been transferred and the operation was not a record transfer (i.e. the opcode was

WOT or WIN).

2. All words in the defined record have been transferred (i.e. A has been decremented to 7777_8).

3. The channel has disconnected ($\overline{W9}$ $\overline{W10}$ $\overline{W11}$ $\overline{W12}$ $\overline{W13}$ $\overline{W14}$) and no more (possible) input words remain to be stored ($\overline{W0}$).

O10 - O10 will gate the actual transfer of one input/output word between Wr and C. (input $\Rightarrow$ Wr$\rightarrow$C; output $\Rightarrow$ C$\rightarrow$Wr).

11.13 WOT/ROT opcodes

WOT/ROT

(11/51)

ϕ0 Lp

T1

T0

Tp Set Cpe

[Check parity of the first output word]

NOTE: Fp is set; O9, O10, O11, and Lp are cleared

ϕ2

T1 Fp $\overline{Ts} \Rightarrow$M$\rightarrow$C

[This takes the next (first) output word to C]

S + 1$\rightarrow$S (Ja, Pr, K11)

[Thus the following word in the (possible) record will be accessed]

Clear Fp

[The logic is now ready with the next (first) output word in C]

Clear O10

[This concludes the transfer of one output word]

$\overline{Wf}$ W0 $\overline{O9}$ $\overline{Tsq}$ ⇒Set O10

[The fact that Wr is empty ($\overline{Wf}$) will be recognized if the channel is not gated to terminate the transmission (W0), the logic is not gated to terminate the operation ($\overline{O9}$), and the next word in the (possible) record may be accessed at the conclusion of this word transfer ($\overline{Tsq}$)]

T0 O10 ⇒Set Fp

[Fp gates the preparation for the next output word]

Tp O10 ROT ⇒A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transferred]

O10 $\overline{Ku2}$ ⇒Set O9

[The WOT/ROT operation will be concluded following this transfer either because it was a single word transmission (WOT ⇒$\overline{Ku2}$) or because the ROT record length has been reached (decremented A = 7777_8 ⇒ $\overline{Ku2}$)]

O10 ⇒Wx

[Wx will actually gate C→Wr as well as gating certain channel housekeeping functions]

$\overline{O9}$ O10 Ku2 ⇒Set Cpe

[Check parity of the next word in the record]

($\overline{O9}$ + O10) ⇒Stay in $\phi 2$

[The operation is not completed]

O9 $\overline{O10}$ ⇒Go to $\phi 7$

[The operation is complete]

$\phi 7$

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp

11.14 WIN/RIN opcodes

WIN/RIN

(15/[illegible])

$\phi 0$ Lp

T1

T0

Tp NOTE: Fp is set; O9, O10, O11, and Lp are cleared.

$\phi 2$ Mw

[Every memory cycle in $\phi 2$ will be a write cycle]

C→M (Jm)

[The input word in C will be written into memory]

T1 O10 $\overline{Fp}$ ⇒S + 1→S (Ja, Pr, K11)

[The address within the record is incremented to store the newly strobed (from Wr) input word]

O10 ⇒Clear Fp

[$\overline{Fp}$ gates the increment of S (see above) before storing every input word except the first one]

Clear O10

[This concludes the transfer of one input word]

$\overline{Wf}$ W0 $\overline{O9}$ $\overline{Tsq}$ ⇒Set O10

[The fact that Wr is empty ($\overline{Wf}$) will be recognized if the channel is not gated to terminate the transmission (WC), the logic is not gated to terminate the operation ($\overline{O9}$), and the previous input word may be stored during the ensuing memory reference ($\overline{Tsq}$).]

T0

Tp O10 RIN ⇒A-1→A (Ja, Gn, Pr)

[A is decremented by one as each word is transferred]

O10 $\overline{Ku2}$ ⇒Set O9

[The WIN/RIN operation will be concluded following this transfer either because it was a single word transmission (WIN ⇒ $\overline{Ku2}$) or because the RIN record length has been reached (decremented A = 7777_8 ⇒ $\overline{Ku2}$)]

O10 ⇒Wr→C

[Wr is emptied into C-from whence it will be stored in memory]

O10 ⇒Wx

[Wx will gate certain channel housekeeping functions]

($\overline{O9}$ + O10) ⇒Stay in ϕ2

[The operation is not completed]

O9 $\overline{O10}$ ⇒Go to ϕ4

[The operation is complete]

ϕ4

T1 C→C (Ja, Pr)

[This is a hardware quirk]

T0 Mw (through Tp)

[This memory cycle will be a write cycle]

C→M (Jm-through Tp)

[The last input word is (unnecessarily) re-written into memory]

Tp

ϕ7

T1 End (through Tp)

[End gates the preparation for the next instruction]

Block any interrupt recognition

[End would normally have gated such recognition]

T0

Tp