

TITLE:KS10 REFERENCE MATERIAL

SOME OF THE INFORMATION IN THIS DOCUMENT WAS GATHERED FROM EXISTING DOCUMENTATION AND CONDENSED HERE. SOME OF IT IS NEW AND PRELIMINARY, INTENDED FOR THE PILOT TRAINING SEMINAR. SOME OF THE MATERIAL MAY BE OF A PROPRIETARY NATURE SO THE DOCUMENT SHOULD NOT BE DISSEMINATED TO NON-DIGITAL PERSONNEL. SUGGESTIONS, CORRECTIONS AND CONTRIBUTIONS (WHICH WILL BE ACKNOWLEDGED) ARE WELCOME AND SHOULD BE DIRECTED TO THE KS10 FIELD SERVICE DOCUMENTATION TEAM.

JOHN SWAN
DAVE ROCKWELL
RICK GRADY
CARY DEVAN
ART JOHNSON

**COMPANY CONFIDENTIAL
DO NOT DUPLICATE**

CONTENTS

GENERAL (SYSTEMS) INFORMATION

KS10 Backplane	SYS-1
KS10 Parity Generation and Checking	SYS-2
KS10 BUS	SYS-3

CPU

Data Path Block Diagram	CPU-1
CRA Block Diagram (2 sheets)	CPU-2
CRM Block Diagram	CPU-4
Internal Register Bit Formats (APRID/RDAPR)	CPU-5
CPU Board Descriptions	CPU-6
Microcode Bit Definitions (2 sheets)	CPU-21
Microcode Bits Sorted by Physical bit	CPU-23
Microcode Bits Sorted by Function	CPU-26
CPU Block Diagram	CPU-29
CPU Major Data Paths	CPU-30
DPE Microcode Control Signals	CPU-31
DPM Microcode Control Signals	CPU-32
CRA Microcode Control Signals	CPU-33
CRAM Address Logic	CPU-34
CPU Clocks	CPU-35
Basic CPU Operation (3 sheets)	CPU-36
Page Fail Handler (9 sheets)	CPU-39
KS10 Microcode	
Memory Fetch Description	CPU-48
Cache Hit Description	CPU-50

Dispatch ROM Definitions	CPU-52
CRA6 Memory Function	CPU-53
Microcode Fields (SPEC, DISPATCH, SKIP)	CPU-54
Power-up Sequence, Microflow	CPU-58
Dump the 2901 Registers, Microflow	CPU-60
MOVE Instruction (From Console), Microflow	CPU-61
RDIO Instruction (User Mode), Microflow	CPU-63
Priority Interrupt, Microflow	CPU-67
Add Instruction, Microflow	CPU-72
SO U JE Instruction (User Mode), Microflow	CPU-74
TLNE Instruction (User Mode), Microflow	CPU-76
MOVE Instruction (User Mode), Microflow	CPU-78
MOVEM Instruction (User Mode0, Microflow	CPU-79
JRST Instruction (User Mode), Microflow	CPU-81
IDLB Instruction (User Mode), Microflow	CPU-82

CONSOLE

Console Module Block Diagram	CSL-1
General Information	CSL-2
Console Commands	CSL-3
KS10 Bus Arbitrator	CSL-5
Console Data (Instruction) Register	CSL-5
Interrupts	CSL-5
8080 Error Codes	CSL-5

CONSOLE (Cont)

Console Flow Diagrams

Deposit Bus	CSL-6
Deposit CRAM	CSL-8
Examine CRAM	CSL-9
Master Reset	CSL-11
Examine Memory	CSL-12
Deposit Memory	CSL-14
Start the Microcode	CSL-15
Execute KS10 Instruction	CSL-16
Boot Command Sequence <i>UART SWITCH CONFIGURATIONS</i>	CSL-17
Console (8080) Register, Bit Format (7 sheets)	CSL-18

MEMORY

MOS Memory Block Diagram	MEM-1
Status Register [#] Bit Definitions	MEM-2
Logic Description	
Write Cycle	MEM-5
Read-Pause-Write	MEM-7
Refresh Cycle	MEM-8
Read Cycle	MEM-9

UBA

Unibus to KS10 Adapter	UBA-1
<i>UBA WRAP-AROUND TEST</i>	<i>UBA-5</i>
<i>UNIBUS SIGNAL PINS</i>	<i>UBA-6</i>

POWER SYSTEM

KS10 Power Distribution - Simplified Block Diagram	PS-1
L. H Power Supply (H7130)	PS-2

MISCELLANEOUS

Device Address and Vector Assignments for KS10	MISC-1
External Registers	MISC-2
RH11/RP04/5/6/ ^{RH43} Register Format and Definitions	MISC-4
RH11/TM02 Register Bit Format and Definition	MISC-6
Drive Command Function Codes	MISC-9

KS10 BACKPLANE

THE BACKPLANE WILL CONSIST OF TWO 9 SLOT BACKPLANES THAT ARE WIRED TOGETHER.

SLOT 1-9 WILL CONTAIN THE MEMORY MODULES.

	1	9	11	19	1	9
A		C			U	U
B		O			N	N
C		N			I	I
D		T				
E	ARRAYS	O		S		
F		L		M		
					RH11	
				1		
				0		

SLOT

19	UNIBUS ADAPTER
18	CONSOLE
17	RESERVED
16	I/O CONTROLLER
15	I/O CONTROLLER
14	DPM
13	DPE
12	CRA
11	CRM
9	MEMORY CONTROLLER
8	ARRAY 64K
7	ARRAY 128K
6	ARRAY
5	ARRAY
4	ARRAY
3	ARRAY
2	ARRAY
1	ARRAY 512K

KS10 BUS

BUS SIGNALS

DEVICES A MAXIMUN OF EIGHT

REQ N REQUEST BUS, ONE PER DEVICE

GRANT N GRANTS BUS TO THE REQUESTING DEVICE

COM/ADR CYCLE MASTER TRANSMITS THIS SIGNAL WHEN IT SENDS THE
 COMMAND ADDRESS

DATA CYCLE MEMORY TRANSMITS THIS SIGNAL WHEN IT HAS BEEN ^{ADDRESSED} ~~ADDRESS~~
 AND HAS ITS READ DATA AVAILABLE. MASTERS TRANSMIT
 THIS SIGNAL WHEN THEY HAVE THEIR WRT DATA AVAILABLE.
 FOR MEMORY.

BAD DATA CYCLE GENERATED WITH DATA CYCLE IF MEMORY
 DATA WAS UNCORRECTABLE, ECC DETECTED A 2 BIT ERROR

I/O DATA CYCLE ADDRESSED I/O DEVICE TRANSMITS THIS SIGNAL TO CPU
 WITH THE DATA FOR AN I/O READ OPERATION

MEMORY BUSY ASSERTED TO PREVENT FURTHER ARBITRATION OF THE BUS
 BY THE MEMORY WHEN IT IS BUSY.

I/O BUSY ONE OF THE DEVICES IS BUSY PERFORMING
 AN I/O WRITE READ

D00-D17, PARL LEFT HALF OF DATA WORD PARITY BIT

D18-D35, PARR RIGHT HALF OF DATA WORD PARITY BIT

D00-D35 USED FOR ADDRESS AND DATA

COMMAND/ADR CYCLE D00 ASSERTED MEANS THE CYCLE IS AN I/O
 FUNCTION
 D01 ASSERTED MEANS THE CYCLE IS A READ
 FUNCTION
 D02 ASSERTED MEANS THE CYCLE IS A WRITE
 FUNCTION

MEMORY D01&D02 ASSERTED MEANS THE CYCLE IS A
 READ-PAUSE WRITE FUNCTION (MEMORY REFERENCE ONLY)
 D14-D35 MEMORY ADDRESS

I/O D14-17 INDICATES DEVICE # DURING I/O CYCLE
 D4 READ PI REQUEST LEVEL SPECIFIED BY D15-D17
 D5 READ INTERRUPT VECTOR FROM UNIT SPECIFIED BY D14-17
 D6 INDICATES AN I/O DATAB OPERATION TO THE UNIBUS
 D18-D35 DEVICE REGISTER
 D18 INDICATES UNIBUS ADAPTER REFERENCE FOR I/O CYCLE
 D19&-18 AND UNIT 0 INDICATES CONSOLE ADDRESS REGISTER
 D20 &-19 &-18 AND UNIT 0 INDICATES MEMORY I/O REGISTER

T CLOCK _____

TRANSMIT CLK -----

 T0 75 NS PULSE 150 NS CYCLE TIME

R CLOCK

RECIEVE CLK USED TO LATCH BUS DATA 75 NS PULSE

INTERRUPT N ASSERTED BY AN I/O DEVICE AND SENT TO CPU PI LOGIC.
THE 7LEVELS ARE PROGRAMABLE.

AC LOW
RESET

1. REQUESTING THE BUS

ANY DEVICE MAY REQUEST THE BUS. THE REQUEST MUST BE CLOCKED BY THE T CLOCK
THE D INPUT OF THE FLOP MUST BE STABLE BY T CLK TIME. THE REQUEST FF MUST BE CLEARED BY THE END OF THE FIRST CYCLE UNLESS A SECOND TRANSFER IS DESIRED.

2. GRANTING THE BUS

THE BUS MONITOR WILL GRANT THE BUS TO THE REQUESTING DEVICE OF HIGHEST PRIORITY, EXCEPT UNDER THE FOLLOWING CONDITIONS:
A) MEMORY BUSY IS ASSERTED
B) THE CURRENT CYCLE IS A COMMAND ADDRESS CYCLE OR THE CYCLE FOLLOWING IF THERE WAS A COMMAND ADDRESS CYCLE.

A DEVICE HAS CONTROL OF THE BUS FOR THREE CYCLES IF THE FIRST CYCLE WAS A COMMAND ADDRESS CYCLE OTHER WISE ONLY TWO CYCLES.

THE GRANT SIGNAL IS STABLE 50 NS BEFORE AND 15 N S AFTER T CLOCK.

3. USING THE BUS

WHEN THE BUS IS GRANTED, THE MASTER MAY EITHER:
A) USE THE BUS VIA A COMMAND ADDRESS CYCLE.
B) NOT USE THE BUS
C) SEND I/O DATA TO THE CPU
THAT IS, THE NORMAL FIRST CYCLE IS A COM/ADR CYCLE, BUT THE BUS MASTER MAY AT HIS OPTION, DECIDE NOT TO USE THIS CYCLE (EG BECAUSE OF A CACHE HIT). NOT USING A GRANTED CYCLE IS EQUIVALENT TO GIVING UP THE BUS. A REQUEST MUST BE MADE TO OBTAIN THE BUS AGAIN.

4. MEMORY READ CYCLE

A READ CYCLE IS INITIATED BY THE BUS MASTER AFTER BEING GRANTED THE BUS. THE FIRST CYCLE MUST BE A COMMAND ADDRESS CYCLE. THE MEMORY WILL ASSERT BUSY IF THE ADDRESS IS VALID AND FREEZE THE BUS. THE MEMORY WILL SEND THE DATA AND DATA CYCLE.
BUSY WILL BE DROPPED BY THE MEMORY WHEN IT IS FREE TO RECIEVE ANOTHER COMMAND/ADDRESS CYCLE.

5. WRITE CYCLE

THE WRITE CYCLE IS INITIATED IN EXACTLY THE SAME WAY AS A READ CYCLE. THE MEMORY ASSERTS BUSY UNTIL IT IS FREE

sys-4

TO RECIEVE ANOTHER COMMAND/ADDRESS CYCLE. THE DATA AND DATA CYCLE IS SENT ON ONE OF THE FOLLOWING CYCLES,

6. READ-PAUSE-WRITE

THE MEMORY STARTS A READ CYCLE AND SENDS THE DATA TO THE MASTER AND THEN WAITS FOR DATA. BUSY IS HELD TRUE FOR THE COMPLETE CYCLE, UNTIL THE MEMORY IS FREE TO RECIEVE ANOTHER COMMAND/ADDRESS CYCLE.

7. I/O READ

AFTER THE CPU SENDS OUT THE ADDRESS THE DEVICE CAN SEND THE DATA OR RETRIEVE THE DATA, REQUEST THE BUS AND SEND THE DATA WITH I/O DATA DURING THE FIRST CYCLE OR THE SECOND CYCLE

8. I/O WRITE

AFTER THE CPU SENDS OUT THE COMMAND ADDRESS THE DATA IS SENT ON THE SECOND OR THIRD CYCLE.

9. BUS LOGIC LEVELS

LOGICAL	VOLTAGE
0	3.4 V
1	0 +.8V

10. BUS TERMINATION

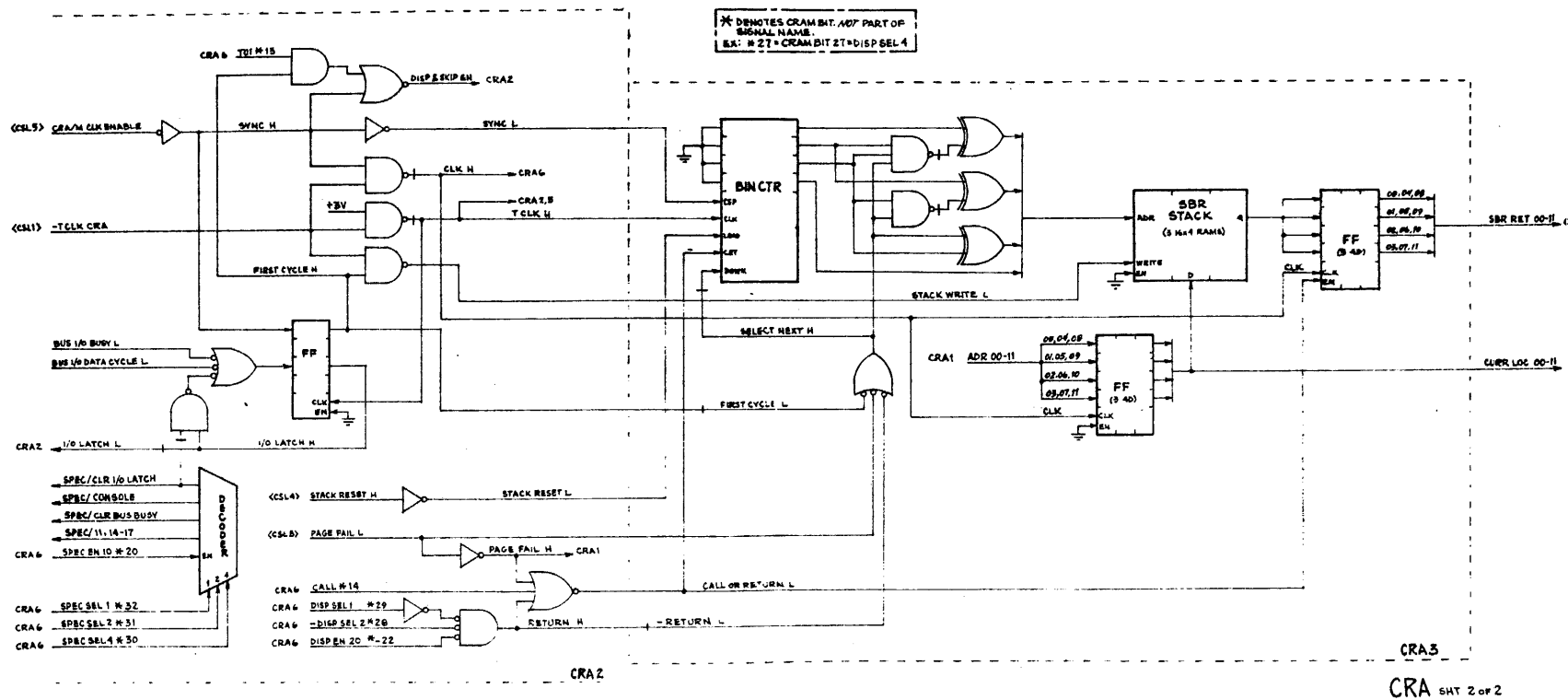
Z= 120 OHMS
BUS TERMINATION WILL BE AT BOTH ENDS OF THE BUS.
THE TERMINATORS WILL BE ON THE CONSOLE MODULE AND THE MEMORY CONTROL MODULE
ONLY 8648 TRANCIVERS CAN BE USED.

11. BUS PARITY ERROR

WHEN DETECTED BY THE CONSOLE THE CPU CLOCK WILL STOP. THE CONSOLE WILL NOTIFY THE CONSOLE TERMINAL.

LF

Sys-5





INTERNAL REGISTER BIT FORMATS -----

APRID 700000

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
RH(E)	*-----MICROCODE OPTIONS-----*-----MICROCODE VERSION NUMBER-----*																	

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
RH(E)	*-----HDWR OPTIONS-----*-----PROCESSOR SERIAL NUMBER-----*																	

WRAPR(CONO APR)
700200

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
E	*-----IORES EN *DIS CLR SET * PWRF *NXM HERR SERR *ITIM INTRQ*PI4 PI2 PI1 *-----																	
19	IORES IO RESET(RESET INTERNAL DEVICES AND GENERATE UNIBUS INIT																	
20	EN ENABLE CONDITIONS SPECIFIED BY BITS 24-31 TO CAUSE INTERRUPTS																	
21	DIS DISABLE INTERRUPTS FOR CONDITIONS SPECIFIED BY BITS 24-31																	
22	CLR CLEAR FLAGS SPECIFIED BY 24-35																	
23	SET SET FLAGS SPECIFIED BY 24-35																	
26	PWRF POWER FAIL																	
27	NXM NON-EXISTENT MEMORY ERROR																	
28	HERR HARD MEMORY ERROR(UNCORRECTABLE)																	
29	SERR SOFT MEM ERROR (CORRECTABLE)																	
30	ITIM INTERVAL TIMER																	
32	INTRQ GENERATE INTERRUPT REQUEST																	
33-35	PI PRIORITY INTERRUPT ASSIGNMENT																	

RDAPR(CONI APR)
700240

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
LH(F)	*-----NOT USED-----*PWRFE*NXME HERR SERR ITIME -----NOT USED-----*																	

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
RH(E)	*-----NOT USED-----*PWRF *NXM HERR SERR *TDONE INTRQ*PI4 PI2 PI1 *-----																	

BITS 8,9,10,11,12=SELECTED FLAGS ENABLED

26	PWRF	POWER FAIL
*27	NXM	NON EXISTENT MEMORY
*28	HERR	HARD MEM ERROR
29	SERR	SOFT MEM ERROR(CORRECTED DATA RETURNED ON BUS)
30	TDONE	INTERVAL TIMER DONE
32	INTRQ	INTERUPT REQUEST
33-35	PI	PRIORITY INTERRUPT LEVEL

*NOTE: PAGE FAIL OCCURS IF ERROR IS RESULT OF CPU MEMORY REQUEST
NXM ALSO SETS IN UNIBUS DEVICE IF ERROR IS RESULT OF
UNIBUS MPR REQUEST

CPU-5

CPU BOARD DESCRIPTIONS

General - All four boards are synchronized by the clock signals produced on CSL1. These clocks come as T CLKS (transmit) and R CLKS (receive) and are very carefully deskewed with the logic on CSL1. With the exception of DPM, the four CPU boards only use T CLKS.

CSL1 R CLK

CSL1 T CLK

DPE5 T CLK

CSL5 CLK ENABLE (L)

DPE5 CLK

CPU CYCLE

Refer to the logic on DPE5. Note how the DPE5 CLK signals are kept in sync with the DPE5 T CLK signals by timing all outputs of the 74S37's at E312 and E314 with CSL1 T CLK DPE. This board only uses one T CLK (DPE5 T CLK A) but generates seven DPE5 CLK signals including DPE5 CLK DP RIGHT, DPE5 CLK DP LEFT, and DPE5 WRITE PARITY. These latter three signals are also under microcode control, but if generated, are done so at DPE5 CLK time.

Similar circuitry is found on CRA2, CRM2, and DPMA. The point being made here is that the CPU CLKS, whenever generated, are deskewed and true (low) in the final 75 ns of a CPU cycle. This is also true for RAMFILE writes, CACHE write, PAGE write, STACK write regardless of the board.

CRA BOARD

The CRA board contains the Control RAM Address logic as well as 36 bits of the 96 bit microcode word. Of the 36 bits, the 12-bit J field, 6-bit DISP field, and 6-bit SKIP field are used internally on CRA, the J field not leaving the board,

(Reference should be made here to the three pages CONTROL RAM LOGIC, SKIP/FUNCTIONS, DISP/FUNCTIONS in the handout accompanying the print set).

CRA6 - Here the 36 bits of the RAM outputs are applied to the CRA6 CRAM register, bits 24 thru 35 applied twice to give both A and B copies. This is merely to save stubbing on the backpanel. Note that bits 00 thru 11 do not leave the board. These are the 12-bit J field. Apart from these, and the SKIP and DISP bits used on the board, all other bits are defined by the table at left of CRA6 and are still (July 1977) subject to change. The parity checking logic looks for even parity, the output signals go to CSL to stop the machine in case of failure. The strange arrangement of bits is due to layout considerations. Note also that CRA2 RESET derived from the 8080 CSL, may clear the entire CRAM word at any time. The cleared word of all zeros still has correct even parity.

CRA1. CRA6 CRAM 00 thru CRA6 CRAM 11 is the J field, and is OR'ed here with any DISP or SKIP outputs to generate the 12-bit CRA1 ADR 00 (=11) address. One copy leaves here to drive CRM, the second copy is used here on CRA. The top four mixers are driven by CRA6 DISP EN 20; the lower four by CRA6 DISP EN 10. It should be clear from these prints, and reference to DISP/FUNCTIONS sheets, that to dispatch on the DIAG ADR register, or the SBR RET register, requires both mixers to be enabled. Thus, DISP/SBR RET requires DISP function of 100 001. 100 generates DISP EN 20 and DISP EN 10 (zeros to enable) and 001 selects input #1.

CRA6 DISP EN 20 only modifies the upper 8 bits of the address, and is usually accompanied by one of the other two enable lines to operate on the lower 4 bits. CRA6 DISP EN 40 selects one of eight dispatch inputs on DPEA; these return to the OR gates at the lower row on CRA1 to also modify address bits 00 thru 11.

The skip functions, most of which are also selected on DPEA, are applied here to modify only the least significant bit of the address. Note that the SKIP enables may be given one, two, or three at a time to enable skipping on one of three possible causes, and may also be given simultaneously with DISP/ functions, to give, for example, a conditional skip during a subroutine return.

CRA2 - The CLK circuitry here has already been described. CRA2 STACK WRITE is generated in sync with the first T CLK of the multiple T CLK CPU CYCLE, to write the current microcode address in the next location on the stack, just in case this may be a CALL instruction.

During the last tick of a CPU cycle, and during the first tick of a 3 tick cycle, CRA2 DISP & SKIP EN is evoked. This signal is used to enable registers on this page that the microcode may SKIP or DISP on. We require these signals not to change during the last two ticks of a microcode cycle.

The NICOND dispatch circuitry shown here is a priority encoder. Note that all input signals come from the backpanel so that priority may be changed by backpanel wiring. It is not clear to me if traps should be initiated, for example, before entering the halt loop requested by the console. Such decisions are therefore deferred by putting them in backpanel wiring.

SKIP/functions 10-17 and SPEC/functions 10-17 are also decoded on this print.

CRA3 - CRA3 shows the subroutine stack. This hardware is rather complex and the reader will probably need to read this description a couple of times.

To start with, we have CSL 8080 generated CRA2 STACK RESET to initialize the pointer at power up, or when issued an MR request. Note that the counter is clocked with a T CLK, as a CPU CLK does not occur during RESET time. However, counting is only enabled during CPU CLK time by CRA2 SYNC to the carry in pin 7. Also, counting is only enabled during CALL or RETURN microcode steps - we don't want the stack to change otherwise. When these conditions are met, then neither PAGE FAIL, OR, not RETURN will cause the stack to increment. (RETURN ,AND, not PAGE FAIL will cause it to decrement.) FIRST CYCLE is of course, not active; it is the last cycle when SYNC is active.

During the FIRST CYCLE (should be called FIRST TICK) of a CPU cycle, CRA3 SELECT NEXT is true. This signal complements bit 2; if bit 2 is true complements bit 4; and if bit 2 and bit 4 are both true, complements bit 8. This signal is thus adding two to the pointer.

If the counter contains N, then during the first cycle (TICK) of a CPU CYCLE, we write the current location into word N+2 of the stack.

Assuming that the current microcode word neither calls nor returns from a subroutine, nothing further happens.

If calling a subroutine, or, if page fail is evoked, CRA3 SELECT NEXT is held true, addressing word N+2 of the RAM until the end of the CPU cycle. At this point, this word becomes the top of the stack, and is therefore loaded into the SBR RET register, and the pointer is incremented.

When a DISP/RETURN is decoded from the current microcode word, the hardware on CRA1 dispatches on the contents of CRA3 SBR RET. CRA3 SELECT NEXT drops at the end of the FIRST CYCLE, to address word N of the stack. Again, at the end of the CPU CYCLE, this word is loaded into SBR RET, because it is now at the top of the stack, and the stack is decremented.

Note that STACK RESET and CRA/M RESET are separate signals from CSL4. It is thus possible to stop the microcode, perhaps examine another microcode word, then continue single stepping from the new location, while maintaining the stack. Loading microcode, and examining microcode functions should not clear the stack. The reset is available for diagnostic purposes, XOR testing purposes, etc.

CRA4 - The CSL board has the ability to examine the outputs of the CRAM register, as well as the CURR LOC, SBR RET, and CRA1 ADR lines. This selection logic is shown on CRA4. The diagnostic hardware here is also used by CRM, and the read function is forced to seven by CSL4 CRAM READ during diagnosis of CRM.

CRA5 - The 12 least significant backpanel BUS lines are received here by the 8646 transceiver latches, under control of the 8080. The console may load the CRA5 DIAG ADR register, for addressing any CRAM location, and then may load the RAM, one 12-bit word at a time, via the CRA4 WRITE pulses. CRA5 also buffers the bus data, for loading the remaining 60 bits on CRM.

Note that when the console reads any 12 bits of data from CRA, the 8646 at E116 (CRA5) also drives BUS DATA lines 21, 22, and 23 with the odd parity generated by the other three transceivers. Thus the board drives 15 data lines, three of which are purely driven to insure even parity on the bus.

CRA7, CRA8, CRA9 - The remaining prints merely shows the address buffers and the 1K RAMS. There are six rows of 12 RAM dips each, therefore, we generate six copies of the address lines for each of board layout, even though a cheaper board would need fewer address buffers, but require care in layout.

CRM Board

The CRM board contains the remaining 2K x 60 bits that wouldn't fit on CRA.

CRM1 - Here we have the address lines from CRA1. Address line 00 is also brought in, unused, for aesthetic reasons. Address lines 01-05 cross the top of the board, while lines 06-11 cross the bottom of the board. At each row of RAM dips, the address lines are buffered, and the buffered copies go to the 24 RAM dips in that row.

CRM2 - The CLK circuitry here has already been described. The second, unused, 74S37 is here to keep any skew here the same as on the other boards.

The CRAM register here (CRM2 CRAM 36-95) is the output that drives the rest of the CPU. As with the 36 bits on CRA6, this register may be cleared via CSL4 CRA/M RESET from the console, (giving even parity). Also, as on CRA6, the table at left is used to equate the CRAM bit numbers with the specific functions that the rest of the CPU knows the bits by,

CRM3 - Across the top of the page, and in the lower right corner, the parity circuitry continuously checks for even parity. The CSL board continuously monitors this signal and can, under 8080 control immediately stop the clock if an error occurs. Note that the MARK BIT is fed back from the backpanel so that it is effectively not used. This same MARK bit is brought to a test point near the module handle, for ease of oscilloscope synchronization.

The diagnostic logic is self explanatory, and is identical and works with, that logic on CRA4.

CRM4 thru CRM8 - These prints are effectively identical, each showing one row of 24 RAM disks, or 12 bits of microcode.

DPE BOARD

The DPE board is the major data path board of the CPU, and the path itself consists of the 2901 4-bit slices (DPE7 & 8). The board of course also contains logic to drive these paths, as well as the IR register DROM, SKIP and DISP logic, and the PI logic.

The actual data flow is shown on the page DPE in the small handout. The internal flow of the 2901 4-bit slice is shown on the page AMD 2901.

DPE1 and DPE2 - The 2901's contain the major working registers, plus ALU's (Arithmetic-Logic Units) and control for operating on the contents of these registers. A 3-bit source field selects two operands; a 3-bit function field determines the function to be performed; and a 3-bit destination field determines where the results go. Refer to sheet AMD 2901. All these fields are provided by the microcode from CRM2. In addition, the source fields are separate for the left and right half of the words. One of the source operands may be selected as the input DBUS.

Note that bit 2 of the destination field is complemented. Thus the eight 2901 destination codes are in a different sequence in the actual machine.

DPE1 also contains the control logic to control the end conditions of the shift paths, and is under control of the three SEL bits of the SPEC/function field. Refer to SPEC/FUNCTIONS sheet for a further description of the shift paths.

Note also that the 2901's define a 40-bit data path. Some instructions (e.g., MUL, DIV, ROT,ASH[left]) require that the data to be shifted is first shifted right into nine 4-bit slices before the appropriate shift paths can be activated, due to the fact that the 36-bit word initially ends in the middle of the end 4-bit slices.

DPE2 also contains the carry look-ahead logic. This should be self explanatory. Note that the microcode can control carry from the left half to right half word (via DPE5/SPEC/CRY 18 INH).

DPE3 and DPE4 - These two prints contain the DBUS mixers. Under control of the microcode DBUS SEL, this mixer can select any of four inputs.

DBUS SEL = 0	DBUS = PC FLAGS,,VMA
= 1	= DP (2901 outputs)
= 2	= RAMFILE
= 3	= DBM (from DPM board)

Note the hardware on DPE3 that will force a selection of DBM back to RAMFILE via DPM5 FORCE RAMFILE. This signal is evoked upon MB WAIT when a cache hit has occurred.

In the lower left of DPE3 is the BYTE DISP logic. This circuitry examines a byte pointer, to determine if we have a 7-bit byte in one of the five standard positions in the word, as shown below.

DP	POSITION						SIZE						DISP		
	00	01	02	03	04	05	06	07	08	09	10	11			
Byte 1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	1
Byte 2	0	1	0	1	1	0	0	0	0	1	1	1	0	1	0
Byte 3	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0
Byte 4	0	0	1	0	0	0	0	0	0	1	1	1	1	0	1
Byte 5	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1

Knowing that the required byte is one of these five, the microcode may use special hardware provided in the SCAD logic (DPM3) and DBM logic (DPM1 & 2) to handle these bytes much more efficiently.

The parity logic, shown on both DPE3 and DPE4, generates ODD PARITY for each half word. These signals are used both in storing RAMFILE parity if loading the ramfile, and storing DP parity, if loading the 2901's from the DBUS. This latter logic is shown on DPE4, where a 16 x 4 bit RAM is shown. This RAM is loaded whenever the CRM2 DEST field shows that we are loading the B register of the 2901, and two bits of microcode are also loaded into the RAM at this time if this parity bit is valid (ie, if the B register is loaded from DBUS, unmodified, then parity is valid). The two bits are one for each half word.

If the DBUS mixer finds that parity is valid, and if the microcode also agrees to check parity, then bits are set in the 74LS175 flip flops. Also, if bad parity was detected, a PARITY ERR signal is sent to the CSL board to halt the clocks. (The operator may also disable the clock halt via an instruction to the 8080.)

DPE5 - The CLK logic has been described elsewhere. However, there are a few unique conditions here. Firstly, the microcode can evoke CRM2 DP HOLD RIGHT to inhibit the generation of DPE5 CLK DP RIGHT to prevent clocking of the right half of the 2901 data path; and similar the left half. Note also the generation of DPE5 WRITE PARITY at the same time as a CPU CLK if enabled by CRM2 DEST 04 or NOT.02.

Some of the SPEC/functions are decoded here. Note that SPEC/IR LOAD and SPEC/XR LOAD are level so that both may be evoked together (SPEC/41 and SPEC/21 come up together via SPEC/61). Also SPEC/ASH TEST must be on level 4 to be evoked during the correct LEFT SHIFT path.

Loading of the VMA is under control of CRA6 MEMORY FUNCTION. (Refer to CRA6 MEMORY FUNCTION page in the handout.) CRM2 #14 unconditionally loads VMA, and #12 loads VMA during the AREAD function only if the particular macro instruction requires it. (Generates DPEA DROM VMA EN.) SPEC/SET SWEEP also requires that the VMA be loaded. The main VMA register is on DPM4, but 10 bits of VMA are duplicated on DPE5 to save backpanel pins. These 10 signals are used in addressing the ramfile (cache).

DPE5 FUNC 01 is used by the 2901's as the least significant function bit. Among other things, this bit changes function 0 ($S + R$) into function 1 ($S - R$). During division, rather than skip on carry out and selecting a microcode step that either does $S+R$ or $S-R$, we merely control the function via CRM2 DIVIDE. This saves 150 ns per divide step in the division subroutine. Similarly, during multi precision, the microcode can cause each step to be the same function as the last, and automatically propagate a previous carry.

DPE6 - The RAMFILE ADDRESS logic is completely contained on DPE6. The table shows the various addressing modes available to the microcode. The Ramfile itself is a 1K ram, of which the lower 128 words are a 512 word cache, and the intermediate 384K words are available as workspace to the microcode.

Function 1 (AC ,FN, #) may need further explanation. By use of the 74LS181 ALU, and this function, AC's such as $AC+1$, $AC+2$, $AC-1$, $AC17$ may be addressed, all in the current block.

All address lines are produced in a high and low true version, and have 4.7K ohm pullup resistors attached, due to the fact that the LS151's don't quite provide enough high drive. Note also that the CRM2 RAM ADR nn lines are buffered by S series and that the 2 x 4 MIX at E402 is also S series for speed-up reasons.

DPE7, DPE8 - These prints show the ramfile itself. The low address lines drive the dips in the ninth row, and the high address lines drive those in the eighth row. Note that each half word has its own parity bit. Each half word has even parity.

DPE9 - The PC flags are under complete control of the microcode via DPE5 SPEC/PC FLAGS and the CRM2 # field. (Note that the # field enters the DPE board only via selection via DBM mixer. CRM2 # nn therefore appears on DPE as either DPM1 DBM nn or DPM2 DBM nn + 18.) The table at the upper left of the print describes the method of control of these flags, and the operation should be self-evident.

However, some things require further description. The carry flags, for example, are driven by SPEC/PC FLAGS and #17, to generate DPE9 CRY0 FLAG, DPE9 CRY1 FLAG, DPE9 OR FLAG (and DPE9 TRAP 1 FLAG with OV FLAG, conditionally, using the DPE9 DP OVERFLOW and DPE9 DP OVERFLOW generated from the DP signals. Note that the carry signals are generated from the carry logic on DPE2, and is only defined during arithmetic functions. Also the DPE1 DP SIGN and DPE1 DP 00 are only the result of the arithmetic function if the microcode makes them so (eg., it is possible to add $C(A) + C(B)$ and put $C(A)$ onto DP lines instead of the sum - this might be found useful for some strange purpose but is not the intended mode of operation).

Note also, that when adding two 36 bit numbers in a 40-bit data path, the correct answer cannot overflow 40 bits. Thus DPE1 DP SIGN has the correct sign, and need only be compared with DPE1 DP 00 to test for overflow. Note also that CRY1 is the exclusive - OR of OVERFLOW and CARRY OUT.

DPE9 SPEC/EXP TEST is to be generated when the exponent is being put back into the data path. At this point, the fraction in the data path is positive, (i.e., it has yet to be negated if it is to be a negative number). As with the carry flags above, the SCAD logic on DPM3 and DPM4 has two extra bits at the left, and cannot have overflowed yet. The problem is merely to test if the 10-bit exponent in SCAD can now fit into the 8-bit field available in the floating point word. Also, it should be a positive number. Thus, for overflow testing SPEC/EXP TEST merely has to test DPM3 SCAD 01, and for underflow testing, DPM3 SCAD 00.

DPE9 SPEC/ASH TEST is generated during a left arithmetic shift. This function (SPEC/44) must be 14, 24, or 44 as the SPEC field select bits also control the shift path (4 for ASH left). The instruction must start with a single bit right shift, to get all sign bits in the left most 2901 4-bit slice. These sign bits are then merely rotated during the ASH left shift, and compared with the bits shifted out to set overflow. Finally, the instruction must finish with a normal (40-bit) left shift.

DPEA - This print contains the IR, XR and AC registers, the DROM driven from IR, and more microcode SKIP and DISP logic.

The SKIP logic contains SKIP functions 20-27 and 40-47. Functions 20 and 40 are both zero (GND) because these are the default cases after MASTER RESET. Note also, that many like functions are on the same level, such that the microcode may test them simultaneously if required. For example, SKIP/14 (001100) enables both 40 and 20, to generate SKIP/-USER and SKIP/USE I/O, which effectively generates SKIP/I/O LEGAL.

The DISP logic is here to save pins on the backpanel. As with the SKIP logic, the J disp and AREAD disp must be on levels 2 and 3, as the most significant eight bits of these dispatches are generated on these levels on CRA1 via DISP EN 20. The remaining dispatches are in random order.

The DROM (Dispatch ROM) consists of three programmable ROM's to give 24 bits unique to each PDP-10 instruction. The eight-bit J field gives 256 possible entry points into the execute code. The four-bit A field is intended to give 16 possible ways of fetching operands before dispatching to the execute code. Similarly, the B field is intended as giving multiple methods of storing operands generated by common code. For example, ADD, ADDM, ADDB go through identical microcode to generate the required sum; the microcode then dispatches on the B field, different for these three instructions, dispatching three different ways to store the answer. The DROM at E113 contains control bits used mainly to control the AREAD and J dispatches, and the AREAD MEM function. DPEA DROM AC DISP is used for instructions, such as JRST, that further decode to AC field to give unique instructions; this is the way the microcode enters unique execute code for them. Similarly DPEA DROM A#J is set for those instructions, such as those with immediate operands, that dispatch directly to the execute code rather than fetching operands. Note that this bit leaves the board - it has the same function to perform on the eight most significant bits of the AREAD dispatch on CRA1.

Special hardware has been added to generate DPEA JRST, 0 which goes also to CRA1 and is used during the EA MOD dispatch. Thus the microcode determines very early (at the same time that it detects indirect and index bits) that it has a JRST, 0 instruction, and is able to save one microcode step on this time-critical instruction.

DPER - This print contains the PI hardware, or most of it. At the bottom of the print are the PI ACTIVE, PI SOFT REQ, and PI CURRENT registers. Their operation is quite straightforward, the signals labels are here for clarity only; the signals do not go anywhere else. (Not even used for reading PI status - the microcode merely remembering in a 2901 register what it loaded these with).

The BUS PI REQ nn signals are synchronized to the CPU CLK, AND'ed with with the PI ACTIVE nn signals, OR'ed with the PI SOFT REQ nn signals, and, if enabled by DPEB PI ON prioritized to generate the highest priority of these on the DPEB PI NEW nn signals. This level is compared with the current PI level to check if an interrupt is required. This interrupt request goes to DPM6, to generate a page fail on the next appropriate memory reference.

Also shown is the hardware required to put the current PI level on the backpanel bus during the I/O WRU (Who-are-you) cycle. Note how the odd-parity of these three bits is placed on the fourth bus line to maintain even bus parity.

Four bits of the APR register are also appropriately located here - the TRAP EN bit, and the three PI level bits. (The rest of APR is located on DPMB). DPMB APR INT REQ is shown enabling the 74LS156 decoder. This has an open-collector output to drive these bus lines. These lines are also pulled up here, with 1K to +5V.

DPM BOARD

The DPM board contains all the data path and logic necessary to make memory requests (via the backpanel bus), which includes the VMA register, hardware PAGE TABLE and CACHE DIRECTORY as well as all bus interface, mixer and timing logic. The 10-bit SCAD (Step-Counter Adder) logic, used for shift, normalize, and exponent arithmetic; for byte pointer updating and for storing 7-bit bytes. The APR register is also on DPM.

DPM1 and DPM2 - These two bits contain the 8-way 36-bit wide DBM mixer. This mixer is under almost complete control of a three bit microcode field (CRM2 DBM SEL nn), the DBM selection shown in a table on DPM1. The single exception, also under microcode control, occurs when selecting DBM/DP while simultaneously selecting one of five bytes via the CRA6 SPEC SEL nn field. This will disable the DPM1 SEL 2 bit to seven of the 36 mixers, selecting the required byte on those mixers, thus depositing a seven-bit byte from the SCAD logic into the data path word.

DPM3 - Here is the 10-bit SCAD logic. Refer to DPM SCAD in the handout. The functions available to the microcode are shown in a table, plus the selections available to both inputs of the adder.

Exponent arithmetic is always done assuming positive floating point numbers - but to save a microcode step, the hardware complements the exponent field on input from a negative number. The exponent is input in the excess 200(8) mode, however, and the microcode must keep track of this.

The microcode may select one of the five 7-bit byte fields in standard ASCII representation. (And put this byte back on any one of the five standard positions - see DPM1 description above).

For shift instructions, the shift inputs (bits 18, 28-35 of the PDP-10 instruction) are also available as an input.

And finally, for updating byte pointer, the position and size fields of the byte pointer are available, though not immediately obvious. Selecting BYTE 1 gives the position field, plus bit 6 of the DP on input A of the adder; selecting SIZE gives the size field plus a zero on input B. Selecting functions POS=SIZE (A=B) thus subtracts the size from the first six bits of byte 1, which is the position field, and the microcode can then put this result back into the pointer via BYTE 1 selection on DBM.

If overflow occurs on this operation, the next microcode step selects 44=SIZE, to subtract the size field from 44. The input 44 also has DP 06 as the seventh bit, for the same reason.

Finally, a ten-bit number may be generated entirely by the microcode.

Also note that the entire SCAD logic is under control of the microcode # field, and cannot be used whenever the # field is being used for other purposes. The final register (FE and SC on DPM4) are controlled, however, by unique microcode bits.

DPM4 - The VMA register is contained on DPM4, as well as the SC and FE registers. The VMA is enabled only during MEM functions, and during the SWEEP function. Refer to CRA6 MEMORY FUNCTION in the handout.

To start a memory reference, the VMA must be loaded with the address, and the VMA flags loaded with the information to control the reference, not necessarily at the same time. Those flags obviously associated with the contents of the VMA, such as USER and PHYSICAL, are loaded with the VMA, are shown on this page, and are prefixed DPM4 VMA. Note that I/O references are controlled by DPM4 VMA flags, and thus, must be initiated at VMA EN time, in this case via DP functions.

The DPM4 VMA AC REF logic is also shown here.

DPM5 - In contrast, those flags associated with the type of reference, rather than the address, are to be found on DPM5, and are prefixed DPM5 MEM. Examples are DPM5 MEM READ and DPM5 MEM WRITE. These are loaded via CRA6 MEMORY FUNCTION, and may or may not be accompanied by loading of the VMA.

A memory cycle (whether MOS RAM, or I/O cycle) is initiated when the microcode generates CRA6 MEMORY FUNCTION. The function is either unconditional (#16), or dependent upon DPEA DROM COND FUNC (#17). In either case, DPM5 MEM EN is generated, to enable loading the MEM flags. However, the cycle is only started (DPM5 START CYCLE) if DPM5 READ EN or DPM5 WRITE EN is generated, and the cycle in progress is not a READ-PAUSE-WRITE cycle (DPM5 RPW CYCLE). Thus CRA6 MEMORY FUNCTION, without #03 (READ) or #05 (WRITE) will merely act as an MB WAIT, but will also clear the MEM FLAGS. (The preferred MB WAIT is via DPMA SPEC/MEM WAIT, which leaves the MEM flags unchanged.)

Note that DPM5 START CYCLE is timed with DPMA SYNC E, which is merely the final tick of a CPU CYCLE. Start Cycle is used to generate DPM5 BUS REQUEST, unless we are initiating a write cycle in SS (Single Step) mode (which brings up DPM5 DLYD WRITE REQ) or else we have DPM5 STOP MAIN MEMORY and are not loading VMA. We don't wish to make the request if we know we are going to get AC REF, PAGE FAIL, or CACHE HIT.

Also, at START CYCLE time, we generate READ DELAY EN, or WRITE DELAY EN. These are used later to enable READ DELAY or WRITE DELAY, to delay the CPU CLK on a subsequent MEM WAIT cycle. DPM5 PAGE FAIL DELAY may also occur at this time. The DELAY enables are turned off as soon as the hardware recognizes that the delay is no longer required.

DPM5 WRITE EN is inhibited during SS MODE if READ EN is brought up too. Thus RPW cycles, during SS MODE, are converted to READ cycles. RPW cycles must always be followed by WRITE cycles; if the RPW cycle was not changed, the START CYCLE for the following WRITE cycle is merely inhibited.

Also, during SS MODE, write cycles bring up DLYD WRITE REQ. DLYD WRITE REQ does not generate BUS REQUEST until the subsequent MEM WAIT cycle. Otherwise, during Single Step Mode we may generate a COM/ADR cycle many milliseconds (or even seconds) before the data cycle. DLYD WRITE REQ merely delays the cycle until the data is ready to be sent, at which time both COM/ADR cycle and DATA CYCLE are both sent.

The final signal on DPM5 is DPM5 FORCE RAMFILE. This signal is generated to convert DBUS/DPM to DBUS/RAMFILE on the data cycle following a READ access. The hardware recognizes that the required data is in the RAMFILE.

DPM6 - This print contains the hardware PAGE TABLE, which is a 512 word x 16 bit ram, made up of eight 286 x 4 bit rams as shown.

During SWEEP cycles, it is intended that DPE2 DP 18 be clear, to reset the VALID bit. During this time, all eight chips will be enabled via pin 17 so that SWEEPing may occur in 256 steps. At all other times, only one chip of each pair will be enabled.

When a memory cycle is initiated, and only then, both DPMC 1 MSEC and DPEB INTERRUPT REQ are loaded into a register to possibly cause a page fail. Both are inhibited by VMA PHYSICAL or MEM WRITE, stopping interrupts from happening during physical memory references (including I/O references), or during writes to memory. This latter is to prevent interrupts from occurring during the second half of a RPW cycle, or from wasting time interrupting just before storing answers away in memory after long multiplication or division. (The microcode still has the ability to check for interrupts elsewhere.) Either of these two interrupts, or NXM or BAD DATA errors, will force PF DISP 10 low, which also inhibits the priority encoder via DPM6 PAGING OK. The encoder outputs then default high, allowing NXM ERR or BAD DATA ERR to modify the PF DISP code. Any of these causes generate PAGE FAIL.

The DPM6 PF DISP nn code is selected via DBM/0 and placed via the 2901's on DP 18-21. The microcode dispatches via DISP/DP 18-21.

DPM7 - The cache directory is a 512 x 12 bit virtual directory. When the VMA is loaded, VMA bits 27-35 address the line number of the cache directory the contents of which is compared with the virtual page number in VMA 18-26, as well as VMA USER. DPM7 CACHE HIT is only generated when all of the following are true,

1. DPMB PAGE EN - Cache enabled via monitor, when paging is turned on.
2. DPM5 MEM READ - Cache hits do not occur on write cycles. We "Write-thru" the cache.
3. -DPM5 MEM CACHE INH - Microcode may inhibit cache hits during read cycles.
4. -DPM4 VMA PHYSICAL - Physical references never generate hits.
5. DPM6 PAGE CACHEABLE - Whole pages may be marked non-cacheable.
6. CSL3 CACHE EN - (Via 74585 at E924) The console may disable the cache.

As with the page table on DPM6, during SWEEP cycles all six of the rams are enabled, to enable turning off the valid bits in only 256 steps. At all other times, only three of the six rams are enabled.

DPM8 and DPM9 - These two prints show the backpanel bus connections, and the mixers that drives them.

The logic on both prints is basically the same. During BUS REQUEST, the COM/ADR data is selected via the mixer, the address itself may however, be directly from the VMA or from the PAGE table, depending upon DPM6 PAGING OK (See generation of DPM9 PAGED REF). At non BUS REQUEST time, the DP data is selected.

Parity is generated for each half of the bus. Outputs of the page table may be sent to the bus 150 nsec after loading VMA, and the parity generation is very time critical. Thus, the DPM9 MIX signals 18-27 directly enter the 93848 parity generator, rather than use the transmit parity output of the 8646 transceivers. Similarly DPM8 MIX 18 & 19 enter the parity chip directly.

DPMC PAGE WRITE EN disables DPM9 MIX signals 19, 20, 23, 24. During page write time, we are not communicating with the backpanel bus, but we still use the parity generator to generate the parity bit for the page table. These four data bits are not used in the page table. We could also have required the microcode to inhibit these four bits, but it didn't cost any hardware.

Both DPM8 MBL PARITY and DPM9 MBR PARITY leave this board, and are checked under microcode control at MEM WAIT time via the hardware on DPE3 and DPE4.

DPMA - Again, the clock circuitry is very similar to that on all other boards, and has been described elsewhere.

DPMA CACHE WRITE and DPMA PAGE WRITE are both generated under hardware control, synchronously with CPU CLKs. PAGE WRITE is generated purely via the microcode (SPEC/PAGE WRITE). CACHE WRITE is generated either during SWEEP cycles (SPEC/SET SWEEP) or during a memory wait of a cycle that did not generate CACHE HIT, nor, VMA AC REF. Simply stated, all cycles that caused a reference to main memory (and were not aborted), or an I/O reference, generate CACHE WRITE. Thus CACHE WRITES occur on data cycles of both read and write cycles to main memory. If VMA PHYSICAL is set, the directory entry is marked invalid. As well as the above CACHE WRITE cycles, MEMORY WRITE cycles to AC references generate DPMA RAMFILE WRITE. It is up to the microcode, of course, to have the correct data on the DBUS when RAMFILE WRITE is generated.

Some of the SPEC/FUNCTIONS are also generated on DPMA. This logic is obvious, with the exception perhaps of SPEC/MEM WAIT. This function is required quickly, and must alone not be gated by SYNC. (it is, of course, SPEC/16).

Also, SPEC/SET SWEEP and SPEC/PAGE WRITE must be capable of being simultaneously evoked, and the same for SPEC/PXCT OFF and SPEC/PXCT EN. To turn PXCT off requires PXCT EN.

At the foot of the page, TRANSMIT 00-17 and TRANSMIT 18-35 are both generated either by COM/ADR EN or DPMA WRITE CYCLE. DPMA WRITE CYCLE is generated at MEMORY DELAY time of a WRITE CYCLE (=DPM5 MEM READ), synchronized either with DPMA SYNC (normally), or by COM/ADR + 1 during write cycles initiated during Single Step Mode.

DPMB - This print contains the APR register, except for the PI bits contained on DPER. There are eight flags, each capable of causing an interrupt and eight enables. Any of the flags, and any of the enables may be individually set or cleared via the microcode by the exec-mode program,

To clear or set the APR EN flags at the bottom of the print, the microcode keeps a copy of these bits in a 2901 register. The contents of this register is modified, then sent to these flags via SPEC/APR EN.

The eight flags at the top of the page may also be individually set or cleared. To do this, however, requires that the microcode read the register via the DBM mixer. As the data passes through the 2901's, the microcode sets or clears the selected bits, then loads them back into the APR register via SPEC/APR flags. This could all happen in one microcode cycle.

Setting bit 32 causes an APR INT REQ. Reading bit 32 (via DBM or DPM2) indicates that APR is interrupting.

DPMC - On DPMC we have the remaining bus access logic, plus the 4.096 MHz oscillator and 1 MSEC interrupt counter. The counter clears at reset time, and, when it overflows, generates DPMC 1 MSEC (if enabled by CSL4 1 MSEC EN). This signal can cause a page fail on the next memory reference, or can be checked at any time by the microcode.

The error flags, NXM and BAD DATA, and the BUS CYCLE flags, are set when the correct inputs are enabled. All flags recirculate the same way, and all are reset via DPM5 CLEAR MEM CYCLE, which is generated either via SPEC/MEM CLEAR, or by the start of a new memory cycle.

FIELD
SIGNAL
A WORD BIT
C RAM BIT
SLOT
PIN
DEFAULT

J (0:11)											
J00	J01	J02	J03	J04	J05	J06	J07	J08	J09	J10	J11
0	1	2	3	4	5	6	7	8	9	10	11
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
12											
(INTERNAL TO CRA BOARD)											

CCODE	FUNCTION	SIMILAR CODES
00 A+Q	ADD	01 05 06
02 Q+Q=Q	PASS	03 04 07
11 B-A+B-A-.25	SUB (1/4 CPL)	10 15 16 20 21 25 26
13 B-Q=B-.25	DECR	12 14 27
24 Q-A=-A-.25	1/4 CPL	22 23 17
35 DVA	OR	30 31 34
37 QYD=D	PASS	32 33 34
46 DAQ	AND	40 41 45
42 QAQ=Q	ZERO	43 44 47
50 AAG	MASK	51 55 56
53 QAB=1AB=B	PASS	52 54
57 DAQ=Q	ZERO	-

<00:35>

ALU FUNCTIONS (12:23)											
LSRC						RSEC					
04	02	01	04	02	01	04	02	01	04	02	01
12	13	14	15	16	17	18	19	20	21	22	23
40	41	42	43	44	45	46	47	48	49	50	51
11											
AR2	BH2	BR2	CH2	CR2	DH2	DR2	EM2	BS2	CJ2	CE2	
D=44						F=LSRC					
0	R	+	S			0	A	Q			
1	R	-	S			1	A	B			
2	R	+	S			2	Q	Q			
3	R	OR	S			3	Q	B			
4	R	AND	S			4	Q	A			
5	R	AND	S			5	D	A			
6	R	XOR	S			6	D	Q			
7	R	XNOR	S			7	D	Q			

61 AYB EX-OR 60 65 66
64 QYA=A PASS 62 63 67
75 DYA=D=EA EX-NOR 70 71 76
77 DFB=D INVERT 72 73 74

FIELD
SIGNAL
A WORD BIT
C RAM BIT
SLOT
PIN
DEFAULT

RAMFILE ADDRESS AND D-BUS (84:44)											
RAM ADR			N.U.	DBUS SEL			DBM SEL				
04	02	01	X	2	1	4	2	1			
06	07	08		40	41	42	43	44			
08	09	10		72	73	69	70	71			
11				11							
AJ2	AS2	BJ2	AJ1	AS1	ER2	FH2	FR2				
D=4			D=1			D=7					

0=AC# AC NUMBER
 1=AC** AC FN#
 2=XR# INDEX REGISTER
 3
 4=VMA VIRTUAL REFERENCE
 5
 6=RAM VMA SELECTS RAM ADR
 7=# ABSOLUTE AC REF

0=PC FLAGSD0-17, NEWPI LEV19-21, VMA27-35
 1=DP DATA PATH
 2=RAM CACHE, ACS, WORKSPACE
 3=DBM DBM MIXER

0=SCARD DIAG0-17, PAGE FAIL DISP18-21, APR FLAGSD27-35
 1=BYTE DATA PATH
 2=BIT LNKEXFOUNT, RH=TIME FRACTION
 3=DP DATA PATH
 4=DP SWAP DATA PATH SWAPPED
 5=VMA VMA FLAGS,, VMA
 6=MEM MEMORY BUFFER
 7=# MAGIC NUMBER IN BOTH HALVES

<36:62>

PARITY GENERATION AND HALF WORD CONTROL (45:56)					
CLKL	GENL	CHKL	CLKR	GENR	CHKR
45	46	47	48	49	50
78	79	80	79	93	51
11					
DJ1	ES2	BE1	DE1	ES2	BN1
D=1	D=K		D=1	D=K	
CLKL=CLOCK LEFT HALF OF MACHINE					
GENL= STORE PARITY FOR 2001 LEFT					
CHKL= CHECK LEFT 1/2 OF DBUS PARITY					
CLKR= SIMILAR TO ABOVE					
GENR= FOR					
CHKR= RIGHT HALF					
* READ PARITY CK (10B) D=0					

SPEC (51:56)												DECODE # BITS	
ENABLE						SELECT						CLR 1MS NCOND FLAG	
40	20	10	4	2	1	11=CLRCLK							
51	52	53	54	55	56	12=CLR 30 LATCH							
18	19	20	30	31	32	13=CLR 10 BUSY							
						14=LD PAGE						WRITE PAGE TABLE	
						15							
						16=LD PRC						LOAD PRC FLAG	
AF1 AN1 BF1 AK1/2 AS1/2 BK1/2						17= WAIT						MEM WAIT	
D=00						20= PREV						FORCE PREVIOUS CONTEXT	
DP SELECTED BY DBM, DBM GETS:						21=LOAD PXR						{ LOAD XRF, PACT FIELD SELECTS AC BLOCK	
DP BITS SCAD 1-7						22							
0=	ALL					23=APR FLAG						LOAD APR FLAG	
1=	00-35					24=CLRCSH						CLEAR CACHE	
2=	00-04, 14-35					25=APR EN						SET APR ENABLES	
3=	00-13, 21-35					26							
4=	00-20, 28-35					27=MEM CLR						{ CLR PAGE FAULT CONDITION	
5=	00-27, 35					30							
6=	ALL					31							
7=	ALL					32							
SHSTYLE:						33							
0=	NORM					34=SWEEP						SET SWEEP	
1=	ZERO					35							
2=	ONES					36=PXCT OFF						TURN OFF EFFECT OF PXCT	
3=	ROT					37							
4=	ASHC					40=INHCRYIB						INHIBIT CARRY INTO LEFT 1/2	
5=	LSHC					41=LOADIR							
6=	DIV					42							
7=	ROTC					43=LDPI						LOAD PI SYSTEM	
BYTE:						44=ASHOV						TEST RESULT OF ASH	
1=	BYTE 1					45=EXPTST						TEST RESULT OF FLOAT PT	
2=	BYTE 2					46=FLAGS						CHANGE PC FLAGS	
3=	BYTE 3					47=LDACBLK						LOAD AC BLK NUMBERS	
4=	BYTE 4					50-60							
5=	BYTE 5					61=LDINST						LOAD INSTRUCTION	

DISPATCH					
ENABLE			SELECT		
41	20	10	4	2	1
51	52	59	60	61	62
2	22	23	27	26	29
12					
BN1	CF1	CN1	BN/R12	CF12	CN/R12
D=70					

11: CONSOLE	CONSOLE DISPATCH
2: "	
3: DROM	
3: A READ	
4: 30	
3: DP LEFT	DP18-21
32	
33	
34: NORM	NORMALIZE
35: DP	DP 32-35
36: RDISP	DROM A FIELD
37: BDISP	DROM B FIELD
46	
4: RETURN	
4: 6	
42: N IL	MULTIPLY
43: PAGE FAIL	
44: NICOND	NEXT INSTRUCTION DISC =
45: BITE	BITE SIZE AND POS = 40
46: SCADDE	EFFECTIVE ADDRESS HERE
47: SCADDB	J12 IF SCAD BIT = 1

Microcode Bit Definitions (Sheet 1 of 2)

CPU-21

FIELD
SIGNAL
ADDRESS BIT
CARRY BIT
SLOT
PIN
DEFAULT

SKIP (63:68)									
ENABLE					SELECT				
40	20	10	4	2	1				
63	64	65	66	67	68				
15	16	17	33	34	35				

12
BM2 CE2 CM2 BS1/2 CS1/2
D=70

04-03
04= IOL6L { NOT USER, USER IOT,
05-11 { CONSOLE MODE
12= LLE AD LEFT L.E. 0
13-30
31= CRY0 AD CRY-2
32= ADLEG0 ADDER LEFT=0
33= ADREG0 ADDER RIGHT=0
34= KERNEL NOT USER
35= FPD FIRST PART DONE
36= AC0 AC NUMBER IS 0
37= INT INTERRUPT REQUEST
40
41
42= LE AD SIGN! AD EQ. 0
43-50
51= CRY2 AD CRY 02
52= DP0 AD SIGN

TIME CONTROL (69:71)									
T00 T01					T02 T03				
69	70				72	73	74	75	76
12	13				29	30	31	32	33

AEZ AMZ
F=*

0=2T
1=3T
2=4T
3=5T
*DT=(109:111)0=0
0=2T
1=3T
2=4T
3=5T

53= DP1B AD P1 5
54= IOT USER IOT
55= JFCL
56= CRY1 AD CRY 1
57= TXXX TEST INSTRUCTION SHOULD SKIP
60-61

RANDOM CONTROL BITS (72:80)									
CRY	LOAD	LOAD	FM	DIV	MULTI	MULTI			
38	39	40	41	42	43	44	45	46	47
72	73	74	75	76	77	78	79	80	
29	30	31	32	33	34	35	36	37	38

AN/R2 D2Z D52 AE1 BF/2 CE1 CN1 BE2

CRY 36 INJECT A CARRY INTO 2501 ADDER
LOADSC LOAD STEP COUNTER FROM SCAD
LOADFE LOAD FE REGISTER FROM SCAD
FMWRITE WRITE RAM FILE
MEM START/COMPLETE MEMORY CYCLE
UNDER MAGIC NUMBER CONTROL
DIVIDE THIS INST IS DOING A DIVIDE
MULTI PREC MULTI PRECISION DIVIDE STEP
MULTI SHIFT FAST SHIFT
CALL THIS IS A CALL

62= ADEG0 AD EQ. 0
63= SC SC SIGN BIT
64= EXECUTE CONSOLE EXECUTE MODE
65= I/O BUSY NOT I/O BUSY
66= CONTINUE CONTINUE

(81:89)
N10
B1:89

MAGIC NUMBER (90:107)									
#00	#01	#02	#03	#04	#05	#06	#07	#08	#09
#10	#11	#12	#13	#14	#15	#16	#17	#18	#19
#20	#21	#22	#23	#24	#25	#26	#27	#28	#29
#30	#31	#32	#33	#34	#35	#36	#37	#38	#39
#40	#41	#42	#43	#44	#45	#46	#47	#48	#49

DE1 D41 EE1 EN1 FE1 FN1 40 2 AM1/2 BD1/2 BM1/2 CD1/2 DM1/2 ED1/2 EM1/2 FD1/2 FM1/2

STATE REGISTER CONTROL
STATE (90:107)
000000 SIMPLE SIMPLE INSTRUCTION CWS
000001 BIT BIT IN PROGRESS
000002 MAP MAP IN PROGRESS
000003 SRC MOVE STRING SRC IN PROGRESS
000004 DST MOVE STRING DST IN PROGRESS
000005 SRC+DST MOVE STRING DEST IN PROGRESS
000006 DST FILLING DEST IN PROGRESS
000007 CVTDB CONVERT DEC TO B N
000010 CCMPT-DST COMPARE DEST NOT CN
000011 EDIT-SRC EDIT SOURCE
000012 EDIT-DST EDIT DESTINATION
000013 EDIT-SHD BOTH SRC AND DST IN PROGRESS

AC CONTROL
ACALU (98:103)
25 B
62 AC+N
ACN (104:107) AC NAMES FOR STRINGS IN EXTENDED CWS
0 SRCLEN SOURCE LENGTH
1 SRCP SOURCE POINTER
3 DLEN DEST LENGTH
4 DSTP DEST POINTER
3 MARK POINTER TO MARK
3 BINC HIGH WORD OF BINARY
4 BIN1 LOW WORD OF BINARY

HALT CODE
HALT (90:107)
00-77= NORMAL HALTS
0 POWER POWER UP
1 HALT HALT INSTRUCTION
2 CBL CONSOLE HALT
100-117= SOFTWARE ERRORS
101 IOPF I/O PAGE FAIL
102 ILLI ILLEGAL INSTRUCTION
103 ILLIT BAD PTR TO UBUS IN INSTRUCTION VECTOR
1000-1117= HARDWARE ERRORS
1000 BWM1 ILLEGAL B-WRITE FROM (BAD ADDR)
1004 NICND ILLEGAL NICND FROM
1005 MULERR VALUE COMPUTED FOR 10+2 WRONG
1777 PAGEF PAGE FAIL IN SMALL CODE

SCAD CONTROL
SCAD (90:92) SCADA (93:95) SCAT (96:99)
0 AXZ 0 SC 0 FE
1 ACRB 1 SH 1 EIT
2 A-B-1 2 PTR 44 (44:48) 2 SHFT
3 A-B 3 BYTE1 3 SIZE
4 A+B 4 BYTE2
5 AANDB 5 BYTE3
6 A-1 6 BYTE4
7 A 7 BYTES
S* (96:107)

WORKSPACE ADDRESS
WORK (96:107)
200 MUL TEMP FOR MULTIPLY
201 DIV TEMP FOR DIVIDE
210 SVMA SAVE VMA
211 SVAR SAVE AR
212 SVARX SAVE ARX
213 SVBR SAVE BR
214 SVBRX SAVE BRX
215 SBR SPT BASE REGISTER
216 CBR CST BASE ADDRESS
217 CSTM CST MASK
220 PUR PROCESS USER REGISTER
221 ADJP "P" FOR ADJBP
222 ADJS "S" FOR ADJBP
223 ADJPTR BYTE POINTER FOR ADJBP
224 ADJG1 TEMP FOR ADJBP
225 ADJG2 TEMP FOR ADJBP
226 ADJBPW BYTES/WORDS FOR ADJBP
227 HSBADR LAST STATUS BLOCK ADDRESS
230 APR APR ENABLER
EXTEND INSTRUCTION
200 EC ORIGINAL EFFECTIVE ADDRESS
241 E1 EA OF WORD AT EC
242 SLEN SOURCE LENGTH
243 MSK BYTE MASK
244 FILL FILL BYTE
245 CWS SRC BYTE IN STRING COMPARE
246 FSG SAVE ARX WHILE STORING
247 BDH BINARY BEING CONVERTED
250 BDL TO DECIMAL
TIMER STUFF
300 TIMEC HIGH ORDER 36 BITS OF TIME
301 TIME1 LOW ORDER 36 BITS OF TIME
302 PERIOD INTERRUPT PERIOD
303 TTG TIME TO GO TO NEXT INTERRUPT
DDIV STUFF
319 ACC
315 AC1
316 AC2
317 AC3
320 DDIV SIGN
321 DVSRH
322 DVSLR
POWERS OF TEN
304 DECLO LOW WORD
375 DECHI HIGH WORD
422 YSAVE Y OF LAST INDIRECT POINTER
APR ID DATA
MICROCODE OPTIONS (90:98) OPT=C
MICROCODE VERSION (99:107) UCY=C3C

FUNCTION									
PC FLAG CONTROL					MEMORY CYCLE CONTROL				
90	SETOV	SET ARITH OVERFLOW	FORCE USER	FORCE USER MODE REF	PI.ZER (90:92)	ZERO	PI.NBI (90:93)	MUST BE ZERO	
91	SETFOV	SET FLOAT OVERFLOW	FORCE EXEC	FORCE EXEC MODE REF (PACT)	PI.P1	PI1 IN PROG	PI.DIR	DROP INTERRUPT REQUESTS	
92	SETNDV	SET NO DIVIDE	FETCH	THIS IS AN INST FETCH	PI.P2		PI.CLR	CLEAR SYSTEM	
93	CLRFPD	CLR FIRST PART DONE	READ CYCLE	START A READ CYCLE	PI.P3				
94	SETFPD	SET FIRST PART DONE	WRITE TEST	PAGE FAIL IF NOT WRITTEN	PI.P4				
95	HOLDUSER	IF-THEN PREVENTS: SET USER IOT/CLR USER IN USER MODE	WRITE CYCLE	START A MEM WRITE CYCLE	PI.P5				
96	SPARE		SPARE		PI.P6				
97	TRAP2	SET TRAP 2	PHYSICAL	DONT LOCK IN CACHE	PI.P7				
98	TRAP1	SET TRAP 1	PHYSICAL	DONT INVOKE PAGING HWDR	PI.P8				
99	LDPCU	LD PCU FROM USER	PACT (99:101)	WHICH PACT BITS TO LOOK AT	PI.P9				
100	SPARE		0= CURRENT	4= B15-12-10-8-6-4	PI.P10	SYSTEM ON	PI.P11	TURN SYSTEM ON	
101	SPARE		1= E1	5= E2	PI.P11	CHAN 1 ON	PI.P12	SELECT CHANNEL 1	
102	SPARE		2	6= B15-12-10-8-6-4	PI.P12		PI.P13		
103	SPARE		3= D1	7= D2	PI.P13		PI.P14		
104	JFCLFLG	DO A JFCL INST	LDVMA	LOAD THE VMA	PI.P14		PI.P15		
105	LD FLGS	LD FLGS FROM DP	EXT ADR	LOAD VMA BITS 16 AND 17	PI.P15		PI.P16		
106			WAIT	START A MEM WRITE CYCLE	PI.P16		PI.P17		
107	ADFLGS	UPDATE CARRY FLGS	RWRITE	START A MEM WRITE CYCLE IF FROM ADRS	PI.P17		PI.P18		

ONLY DURING A DP FUNCTION (90:100)

99 PREVIOUS
100 10 CYCLE THIS IS AN INST CYCLE
101 WRD CYCLE WHO ARE NO CYCLE
102 VECTOR CYCLE READ INTERRUPT VECTOR
103 10 BYTE BYTE CYCLE
104

FLG BITS
(94) FLG.W WBIT FROM PAGE MAP
(95) FLG.PI PI CYCLE
(96) FLG.C CACHE BIT FROM PAGE MAP

Microcode Bit Definitions (Part 2 of 2)

KS10 MICROWORD BITS SORTED BY PHYSICAL BIT

CRA6 CRAM 00 H	CRA6 J 00 H	NC
CRA6 CRAM 01 H	CRA6 J 01 H	NC
CRA6 CRAM 02 H	CRA6 J 02 H	NC
CRA6 CRAM 03 H	CRA6 J 03 H	NC
CRA6 CRAM 04 H	CRA6 J 04 H	NC
CRA6 CRAM 05 H	CRA6 J 05 H	NC
CRA6 CRAM 06 H	CRA6 J 06 H	NC
CRA6 CRAM 07 H	CRA6 J 07 H	NC
CRA6 CRAM 08 H	CRA6 J 08 H	NC
CRA6 CRAM 09 H	CRA6 J 09 H	NC
CRA6 CRAM 10 H	CRA6 J 10 H	NC
CRA6 CRAM 11 H	CRA6 J 11 H	NC
CRA6 CRAM 12 H	CRA6 T00 H	A12E2
CRA6 CRAM 13 H	CRA6 T01 H	A12M2
CRA6 CRAM 14 H	CRA6 CALL H	B12E2
CRA6 CRAM 15 H	CRA6 SKIP EN 40 L	B12M2
CRA6 CRAM 16 H	CRA6 SKIP EN 20 L	C12E2
CRA6 CRAM 17 H	CRA6 SKIP EN 10 L	C12M2
CRA6 CRAM 18 H	CRA6 SPEC EN 40 H	A12F1
CRA6 CRAM 19 H	CRA6 SPEC EN 20 H	A12N1
CRA6 CRAM 20 H	CRA6 SPEC EN 10 H	B12F1
CRA6 CRAM 21 H	CRA6 DISP EN 40 L	B12N1
CRA6 CRAM 22 H	CRA6 DISP EN 20 L	C12F1
CRA6 CRAM 23 H	CRA6 DISP EN 10 L	C12N1
CRA6 CRAM 24A H	CRA6 CRAM PARITY BIT A H	A12F2
CRA6 CRAM 24B H	CRA6 CRAM PARITY BIT B H	A12J2
CRA6 CRAM 25A H	CRA6 CARRY IN A H	A12N2
CRA6 CRAM 25B H	CRA6 CARRY IN B H	A12R2
CRA6 CRAM 26A H	CRA6 MEM FUNCTION A H	B12F2
CRA6 CRAM 26B H	CRA6 MEM FUNCTION B H	B12J2
CRA6 CRAM 27A H	CRA6 DISP SEL 4A H	B12N2
CRA6 CRAM 27B H	CRA6 DISP SEL 4B H	R12R2
CRA6 CRAM 28A H	CRA6 DISP SEL 2A H	C12F2
CRA6 CRAM 28B H	CRA6 DISP SEL 2B H	C12J2
CRA6 CRAM 29A H	CRA6 DISP SEL 1A H	C12N2
CRA6 CRAM 29B H	CRA6 DISP SEL 1B H	C12R2
CRA6 CRAM 30A H	CRA6 SPEC SEL 4A H	A12K1
CRA6 CRAM 30B H	CRA6 SPEC SEL 4B H	A12K2
CRA6 CRAM 31A H	CRA6 SPEC SEL 2A H	A12S1
CRA6 CRAM 31B H	CRA6 SPEC SEL 2B H	A12S2
CRA6 CRAM 32A H	CRA6 SPEC SEL 1A H	B12K1
CRA6 CRAM 32B H	CRA6 SPEC SEL 1B H	B12K2
CRA6 CRAM 33A H	CRA6 SKIP SEL 4A H	B12S1
CRA6 CRAM 33B H	CRA6 SKIP SEL 4B H	B12S2
CRA6 CRAM 34A H	CRA6 SKIP SEL 2A H	C12K1
CRA6 CRAM 34B H	CRA6 SKIP SEL 2B H	C12K2
CRA6 CRAM 35A H	CRA6 SKIP SEL 1A H	C12S1
CRA6 CRAM 35B H	CRA6 SKIP SEL 1B H	C12S2

CRM2 BIT 36A H
 CRM2 BIT 36B H
 CRM2 BIT 37A H
 CRM2 BIT 37B H
 CRM2 BIT 38A H
 CRM2 BIT 38B H
 CRM2 BIT 39A H
 CRM2 BIT 39B H
 CRM2 BIT 40A H
 CRM2 BIT 40B H
 CRM2 BIT 41A H
 CRM2 BIT 41B H
 CRM2 BIT 42A H
 CRM2 BIT 42B H
 CRM2 BIT 43A H
 CRM2 BIT 43B H
 CRM2 BIT 44A H
 CRM2 BIT 44B H
 CRM2 BIT 45A H
 CRM2 BIT 45B H
 CRM2 BIT 46A H
 CRM2 BIT 46B H
 CRM2 BIT 47A H
 CRM2 BIT 47B H
 CRM2 BIT 48H
 CRM2 BIT 49 H
 CRM2 BIT 50 H
 CRM2 BIT 51 H
 CRM2 BIT 52 H
 CRM2 BIT 53 H
 CRM2 BIT 54 H
 CRM2 BIT 55 H
 CRM2 BIT 56 H
 CRM2 BIT 57 H
 CRM2 BIT 58 H
 CRM2 BIT 59 H
 CRM2 BIT 60 H
 CRM2 BIT 61 H
 CRM2 BIT 62 H
 CRM2 BIT 63 H
 CRM2 BIT 64 H
 CRM2 BIT 65 H
 CRM2 BIT 66 H
 CRM2 BIT 67 H
 CRM2 BIT 68 H
 CRM2 BIT 69 H
 CRM2 BIT 70 H
 CRM2 BIT 71 H
 CRM2 BIT 72 H
 CRM2 BIT 73 H
 CRM2 BIT 74 H
 CRM2 BIT 75 H
 CRM2 BIT 76 H
 CRM2 BIT 77 H

CRM2 # 06A H
 CRM2 # 06B H
 CRM2 # 07A H
 CRM2 # 07B H
 CRM2 # 08A H
 CRM2 # 08B H
 CRM2 # 09A H
 CRM2 # 09B H
 CRM2 # 10A H
 CRM2 # 10B H
 CRM2 # 11A H
 CRM2 # 11B H
 CRM2 # 12A H
 CRM2 # 12B H
 CRM2 # 13A H
 CRM2 # 13B H
 CRM2 # 14A H
 CRM2 # 14B H
 CRM2 # 15A H
 CRM2 # 15B H
 CRM2 # 16A H
 CRM2 # 16B H
 CRM2 # 17A H
 CRM2 # 17B H
 CRM2 RAMFILE WRITE H
 CRM2 SPARE 49 H
 CRM2 PARITY INH LEFT H
 CRM2 PARITY INH RIGHT H
 CRM2 DIVIDE H
 CRM2 MULTI PRECISION H
 CRM2 # 00 H
 CRM2 # 01 H
 CRM2 # 02 H
 CRM2 # 03 H
 CRM2 # 04 H
 CRM2 # 05 H
 CRM2 FUNC 04 H
 CRM2 FUNC 02 H
 CRM2 FUNC 01 H
 CRM2 SOURCE L 04 H
 CRM2 SOURCE L 02 H
 CRM2 SOURCE L 01 H
 CRM2 SOURCE R 04 H
 CRM2 SOURCE R 02 H
 CRM2 SOURCE R 01 H
 CRM2 DBM SEL 4 H
 CRM2 DBM SEL 2 H
 CRM2 DBM SEL 1 H
 CRM2 DBUS SEL 2 H
 CRM2 DBUS SEL 1 H
 CRM2 DP B ADR 10 H
 CRM2 DP B ADR 04 H
 CRM2 DP B ADR 02 H
 CRM2 DP B ADR 01 H

A11D1
 A11D2
 A11M1
 A11M2
 B11D1
 B11D2
 B11M1
 B11M2
 C11D1
 C11D2
 C11M1
 C11M2
 D11D1
 D11D2
 D11M1
 D11M2
 E11D1
 E11D2
 E11M1
 E11M2
 F11D1
 F11D2
 F11M1
 F11M2
 A11E1
 A11N1
 B11E1
 B11N1
 C11E1
 C11N1
 D11E1
 D11N1
 E11E1
 E11N1
 F11E1
 F11N1
 A11H2
 A11R2
 B11H2
 B11R2
 C11H2
 C11R2
 D11H2
 D11R2
 E11M2
 E11R2
 F11H2
 F11R2
 A11J1
 A11S1
 B11J1
 B11S1
 C11J1
 C11S1

CRM2 BIT 78 H
CRM2 BIT 79 H
CRM2 BIT 80 H
CRM2 BIT 81 H
CRM2 BIT 82 H
CRM2 BIT 83 H
CRM2 BIT 84 H
CRM2 BIT 85 H
CRM2 BIT 86 H
CRM2 BIT 87 H
CRM2 BIT 88 H
CRM2 BIT 89 H
CRM2 BIT 90 H
CRM2 BIT 91 H
CRM2 BIT 92 H
CRM2 BIT 93 H
CRM2 BIT 94 H
CRM2 BIT 95 H

CRM2 CLK EN LEFT H
CRM2 CLK EN RIGHT H
CRM2 DP A ADR 10 H
CRM2 DP A ADR 04 H
CRM2 DP A ADR 02 H
CRM2 DP A ADR 01 H
CRM2 RAM ADR 04 H
CRM2 RAM ADR 02 H
CRM2 RAM ADR 01 H
CRM2 DEST 04 H
CRM2 DEST 02 H
CRM2 DEST 01 H
CRM2 SC EN H
CRM2 FE EN H
CRM2 PARITY CHECK LEFT H
CRM2 PARITY CHECK RIGHT H
CRM2 CRAM PARITY BIT H
CRM2 MARK BIT H

D11J1
D11S1
E11J1
E11S1
F11J1
F11S1
A11J
A11S2
B11J2
B11S2
C11J2
C11S2
D11J2
D11S2
E11J2
E11S2
F11J2
F11S2

KS10 MICRO WORD BITS SORTED BY FUNCTION

CRM2 BIT	54	H	CRM2 #	00	H	D11E1
CRM2 BIT	55	H	CRM2 #	01	H	D11N1
CRM2 BIT	56	H	CRM2 #	02	H	E11E1
CRM2 BIT	57	H	CRM2 #	03	H	E11N1
CRM2 BIT	58	H	CRM2 #	04	H	F11E1
CRM2 BIT	59	H	CRM2 #	05	H	F11N1
CRM2 BIT	36A	H	CRM2 #	06A	H	A11D1
CRM2 BIT	36B	H	CRM2 #	06B	H	A11D2
CRM2 BIT	37A	H	CRM2 #	07A	H	A11M1
CRM2 BIT	37B	H	CRM2 #	07B	H	A11M2
CRM2 BIT	38A	H	CRM2 #	08A	H	B11D1
CRM2 BIT	38B	H	CRM2 #	08B	H	B11D2
CRM2 BIT	39A	H	CRM2 #	09A	H	B11M1
CRM2 BIT	39B	H	CRM2 #	09B	H	B11M2
CRM2 BIT	40A	H	CRM2 #	10A	H	C11D1
CRM2 BIT	40B	H	CRM2 #	10B	H	C11D2
CRM2 BIT	41A	H	CRM2 #	11A	H	C11M1
CRM2 BIT	41B	H	CRM2 #	11B	H	C11M2
CRM2 BIT	42A	H	CRM2 #	12A	H	D11D1
CRM2 BIT	42B	H	CRM2 #	12B	H	D11D2
CRM2 BIT	43A	H	CRM2 #	13A	H	D11M1
CRM2 BIT	43B	H	CRM2 #	13B	H	D11M2
CRM2 BIT	44A	H	CRM2 #	14A	H	E11D1
CRM2 BIT	44B	H	CRM2 #	14B	H	E11D2
CRM2 BIT	45A	H	CRM2 #	15A	H	E11M1
CRM2 BIT	45B	H	CRM2 #	15B	H	E11M2
CRM2 BIT	46A	H	CRM2 #	16A	H	F11D1
CRM2 BIT	46B	H	CRM2 #	16B	H	F11D2
CRM2 BIT	47A	H	CRM2 #	17A	H	F11M1
CRM2 BIT	47B	H	CRM2 #	17B	H	F11M2
CRA6 CRAM	14	H	CRA6 CALL		H	B12E2
CRA6 CRAM	25A	H	CRA6 CARRY IN A		H	A12N2
CRA6 CRAM	25B	H	CRA6 CARRY IN B		H	A12R2
CRM2 BIT	78	H	CRM2 CLK EN LEFT		H	D11J1
CRM2 BIT	79	H	CRM2 CLK EN RIGHT		H	D11S1
CRA6 CRAM	24A	H	CRA6 CRAM PARITY BIT A		H	A12F2
CRA6 CRAM	24B	H	CRA6 CRAM PARITY BIT B		H	A12J2
CRM2 BIT	94	H	CRM2 CRAM PARITY BIT		H	F11J2
CRM2 BIT	71	H	CRM2 DBM SEL 1		H	F11R2
CRM2 BIT	70	H	CRM2 DBM SEL 2		H	F11H2
CRM2 BIT	69	H	CRM2 DBM SEL 4		H	E11R2
CRM2 BIT	73	H	CRM2 DBUS SEL 1		H	A11S1
CRM2 BIT	72	H	CRM2 DBUS SEL 2		H	A11J1
CRM2 BIT	89	H	CRM2 DEST 01		H	C11S2
CRM2 BIT	88	H	CRM2 DEST 02		H	C11J2
CRM2 BIT	87	H	CRM2 DEST 04		H	B11S2
CRA6 CRAM	23	H	CRA6 DISP EN 10		L	C12N1
CRA6 CRAM	22	H	CRA6 DISP EN 20		L	C12F1
CRA6 CRAM	21	H	CRA6 DISP EN 40		L	B12N1
CRA6 CRAM	29A	H	CRA6 DISP SEL 1A		H	C12N2
CRA6 CRAM	29B	H	CRA6 DISP SEL 1B		H	C12R2
CRA6 CRAM	28A	H	CRA6 DISP SEL 2A		H	C12F2
CRA6 CRAM	28B	H	CRA6 DISP SEL 2B		H	C12J2
CRA6 CRAM	27A	H	CRA6 DISP SEL 4A		H	B12N2
CRA6 CRAM	27B	H	CRA6 DISP SEL 4B		H	B12R2

CRM2 BIT	52	H	CRM2 DIVIDE H	C11E1
CRM2 BIT	83	H	CRM2 DP A ADR 01 H	F11S1
CRM2 BIT	82	H	CRM2 DP A ADR 02 H	F11J1
CRM2 BIT	81	H	CRM2 DP A ADR 04 H	E11S1
CRM2 BIT	80	H	CRM2 DP A ADR 10 H	E11J1
CRM2 BIT	77	H	CRM2 DP B ADR 01 H	C11S1
CRM2 BIT	76	H	CRM2 DP B ADR 02 H	C11J1
CRM2 BIT	75	H	CRM2 DP B ADR 04 H	B11S1
CRM2 BIT	74	H	CRM2 DP B ADR 10 H	B11J1
CRM2 BIT	91	H	CRM2 FE EN H	D11S2
CRM2 BIT	62	H	CRM2 FUNC 01 H	B11H2
CRM2 BIT	61	H	CRM2 FUNC 02 H	A11R2
CRM2 BIT	60	H	CRM2 FUNC 04 H	A11H2
CRA6 CRAM	00	H	CRA6 J 00 H	NC
CRA6 CRAM	01	H	CRA6 J 01 H	NC
CRA6 CRAM	02	H	CRA6 J 02 H	NC
CRA6 CRAM	03	H	CRA6 J 03 H	NC
CRA6 CRAM	04	H	CRA6 J 04 H	NC
CRA6 CRAM	05	H	CRA6 J 05 H	NC
CRA6 CRAM	06	H	CRA6 J 06 H	NC
CRA6 CRAM	07	H	CRA6 J 07 H	NC
CRA6 CRAM	08	H	CRA6 J 08 H	NC
CRA6 CRAM	09	H	CRA6 J 09 H	NC
CRA6 CRAM	10	H	CRA6 J 10 H	NC
CRA6 CRAM	11	H	CRA6 J 11 H	NC
CRM2 BIT	95	H	CRM2 MARK BIT H	F11S2
CRA6 CRAM	26A	H	CRA6 MEM FUNCTION A H	B12F2
CRA6 CRAM	26B	H	CRA6 MEM FUNCTION B H	B12J2
CRM2 BIT	53	H	CRM2 MULTI PRECISION H	C11N1
CRM2 BIT	92	H	CRM2 PARITY CHECK LEFT H	E11J2
CRM2 BIT	93	H	CRM2 PARITY CHECK RIGHT H	E11S2
CRM2 BIT	50	H	CRM2 PARITY INH LEFT H	B11E1
CRM2 BIT	51	H	CRM2 PARITY INH RIGHT H	B11N1
CRM2 BIT	86	H	CRM2 RAM ADR 01 H	B11J2
CRM2 BIT	85	H	CRM2 RAM ADR 02 H	A11S2
CRM2 BIT	84	H	CRM2 RAM ADR 04 H	A11J
CRM2 BIT	48	H	CRM2 RAMFILE WRITE H	A11E1
CRM2 BIT	90	H	CRM2 SC EN H	D11J2
CRA6 CRAM	17	H	CRA6 SKIP EN 10 L	C12M2
CRA6 CRAM	16	H	CRA6 SKIP EN 20 L	C12E2
CRA6 CRAM	15	H	CRA6 SKIP EN 40 L	B12M2
CRA6 CRAM	35A	H	CRA6 SKIP SEL 1A H	C12S1
CRA6 CRAM	35B	H	CRA6 SKIP SEL 1B H	C12S2
CRA6 CRAM	34A	H	CRA6 SKIP SEL 2A H	C12K1
CRA6 CRAM	34B	H	CRA6 SKIP SEL 2B H	C12K2
CRA6 CRAM	33A	H	CRA6 SKIP SEL 4A H	B12S1
CRA6 CRAM	33B	H	CRA6 SKIP SEL 4B H	B12S2
CRM2 BIT	65	H	CRM2 SOURCE L 01 H	C11R2
CRM2 BIT	64	H	CRM2 SOURCE L 02 H	C11H2
CRM2 BIT	63	H	CRM2 SOURCE L 04 H	B11R2
CRM2 BIT	68	H	CRM2 SOURCE R 01 H	E11M2
CRM2 BIT	67	H	CRM2 SOURCE R 02 H	D11R2
CRM2 BIT	66	H	CRM2 SOURCE R 04 H	D11H2
CRM2 BIT	49	H	CRM2 SPARE 49 H	A11N1
CRA6 CRAM	20	H	CRA6 SPEC EN 10 H	B12F1

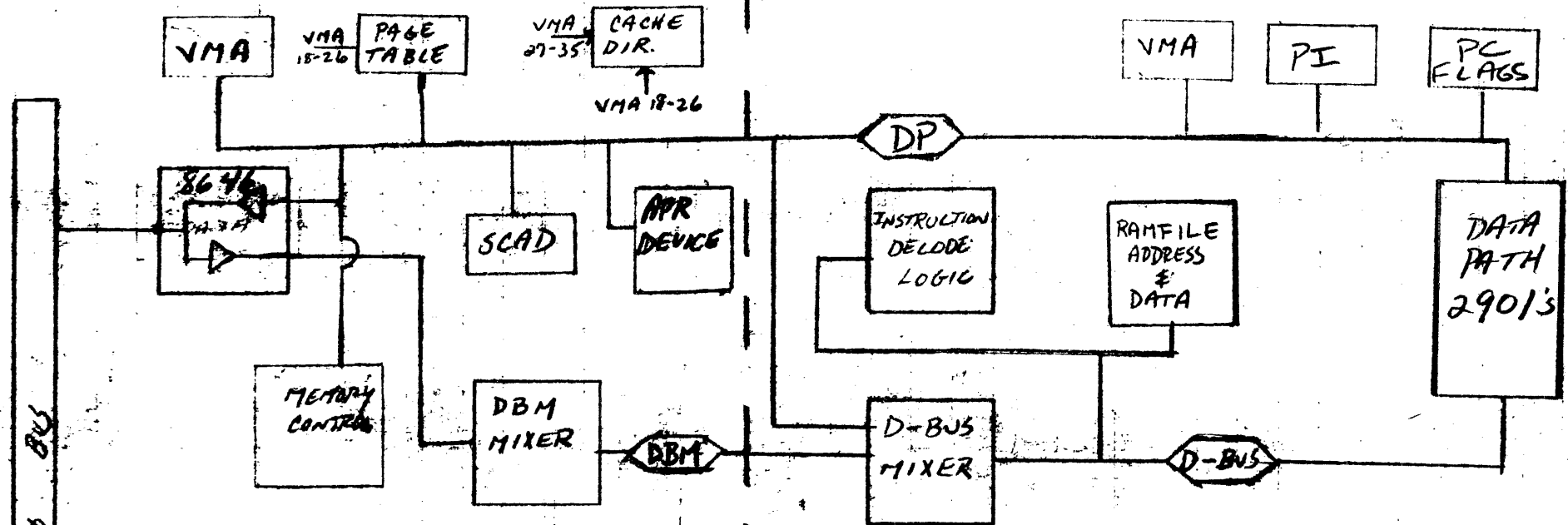
CRA6 CRAM 19 H
CRA6 CRAM 18 H
CRA6 CRAM 32A H
CRA6 CRAM 32B H
CRA6 CRAM 31A H
CRA6 CRAM 31B H
CRA6 CRAM 30A H
CRA6 CRAM 30B H
CRA6 CRAM 12 H
CRA6 CRAM 13 H

CRA6 SPEC EN 20 H
CRA6 SPEC EN 40 H
CRA6 SPEC SEL 1A H
CRA6 SPEC SEL 1B H
CRA6 SPEC SEL 2A H
CRA6 SPEC SEL 2B H
CRA6 SPEC SEL 4A H
CRA6 SPEC SEL 4B H
CRA6 T00 H
CRA6 T01 H

A12N1
A12F1
B12K1
B12K2
A12S1
A12S2
A12K1
A12K2
A12E2
A12M2

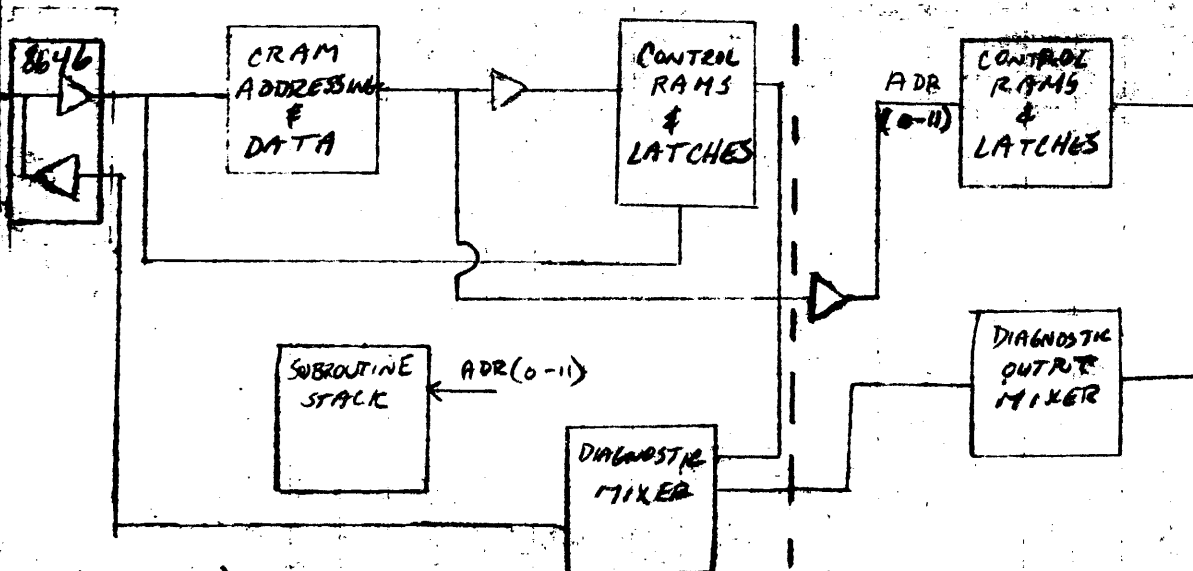
CPU BLOCK DIAGRAM

1 MSEC
CLOCK



DPM (8621)

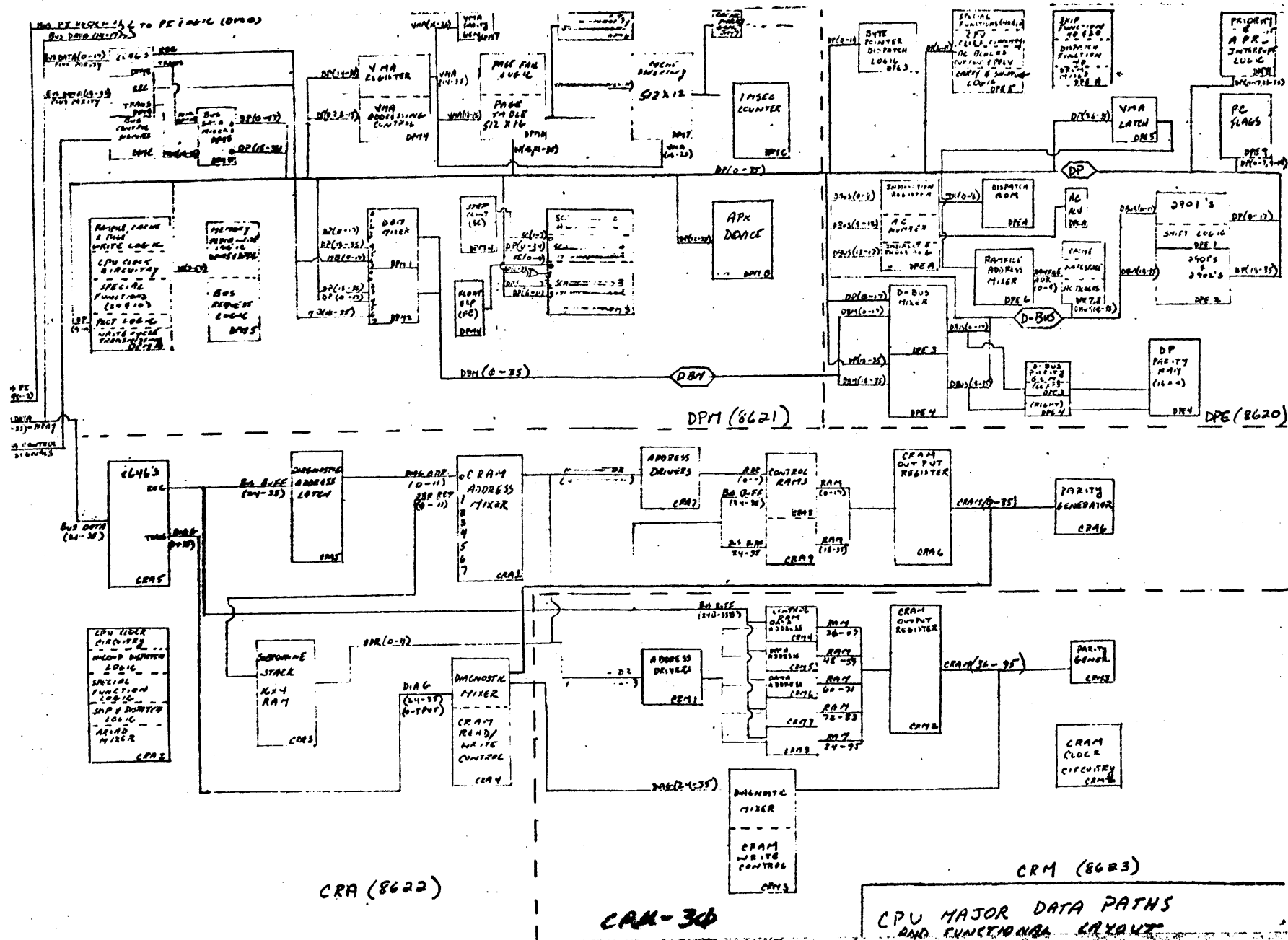
DPE (8620)

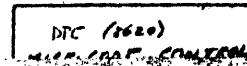


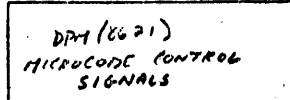
CPU-29

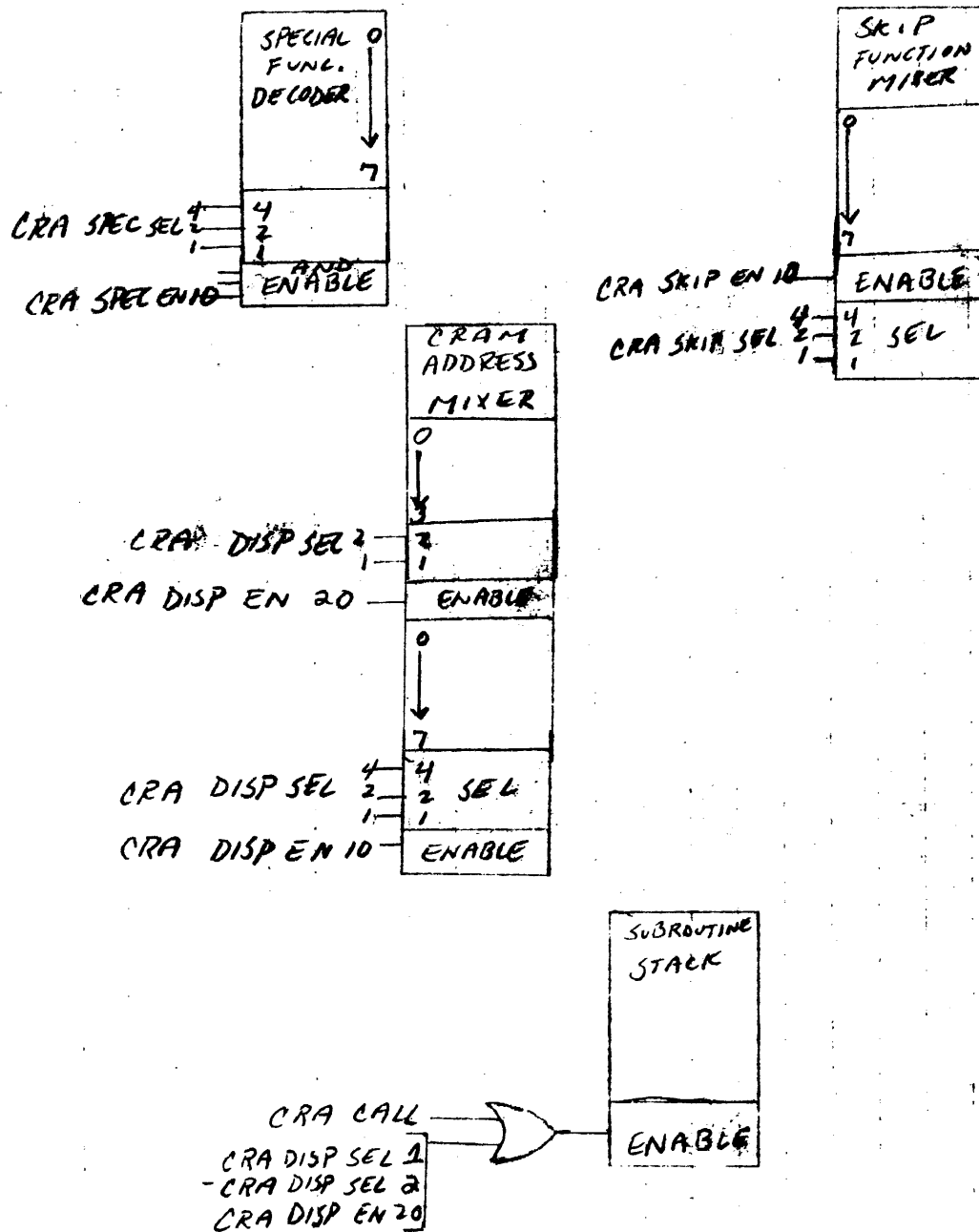
CRM (8623)

ADR (8622)





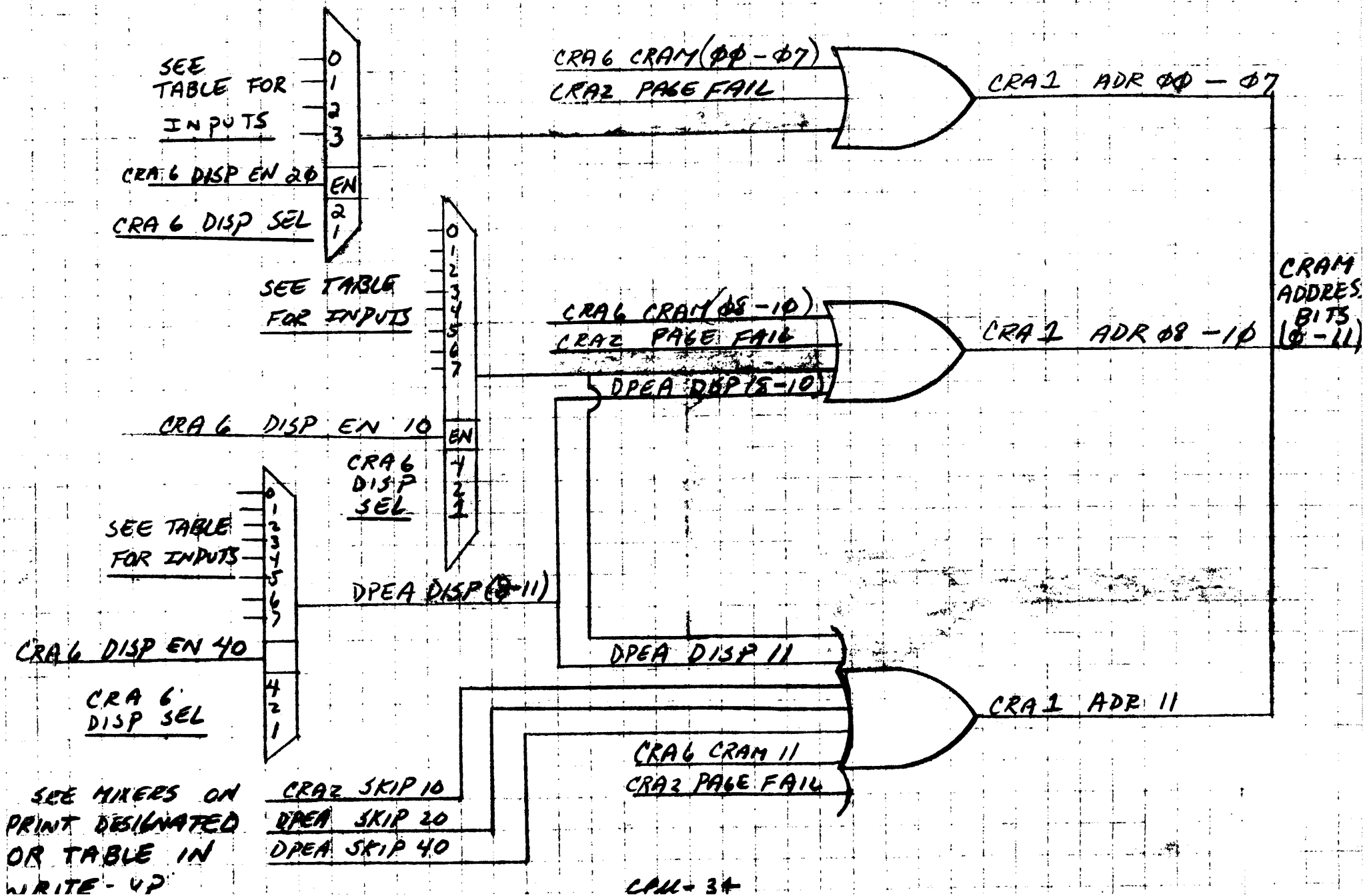




THIS IS INTENDING
TO SHOW THE MICROCODE
SIGNALS THAT ARE
USED IN CONTROLLING
MIXER'S & DECODERS
ON THE CRA BOARD.
IT IS TO BE USED
WITH THE CPU
MAJOR DATA PATHS
DIAGRAM TO SEE
HOW IT FITS IN
TO CONTROL THE
CPU.

CRA (8622)
MICROCODE CONTROL

CRAM ADDRESSING (SEE CRA1)



T CLK
(H)

R CLK (H)

CRA/M CLK ENABLE
DPE/M CLK ENABLE
(L)

3 TICK INSTRUCTIONS
(450 M) TFIELD OF 1

2 TICK
(300 M) TFIELD OF 0

DPE, DPM &
CRA TCLK'S (H)

(TRUE FOR LAST
150 MSEC OF CPU CYCLE)
CRA SYNC (H)

CRA & CRM
CLK'S (A-D)
(D-J) (H)

DPE CLK
(BCFH) CLK
DPM CLK
(DE)

CPU CYCLE
3 TICK INSTR.

FIRST (H)
CYCLE CRA2
TRUE FOR
FIRST 150 MS OF
CPU CYCLE

DP CLK
RIGHT & LEFT
AS LONG AS
CRA2, DP CLK
EN RIGHT &
LEFT ARE TRUE
(H)

NOTE: GATE PROPAGATION
DELAY IS NOT
INCLUDED

CPU CLOCKS
(ABOVE)

STACK WRITE L WRITES THE
LAST M-CODE LOC-
ONTO THE STACK.

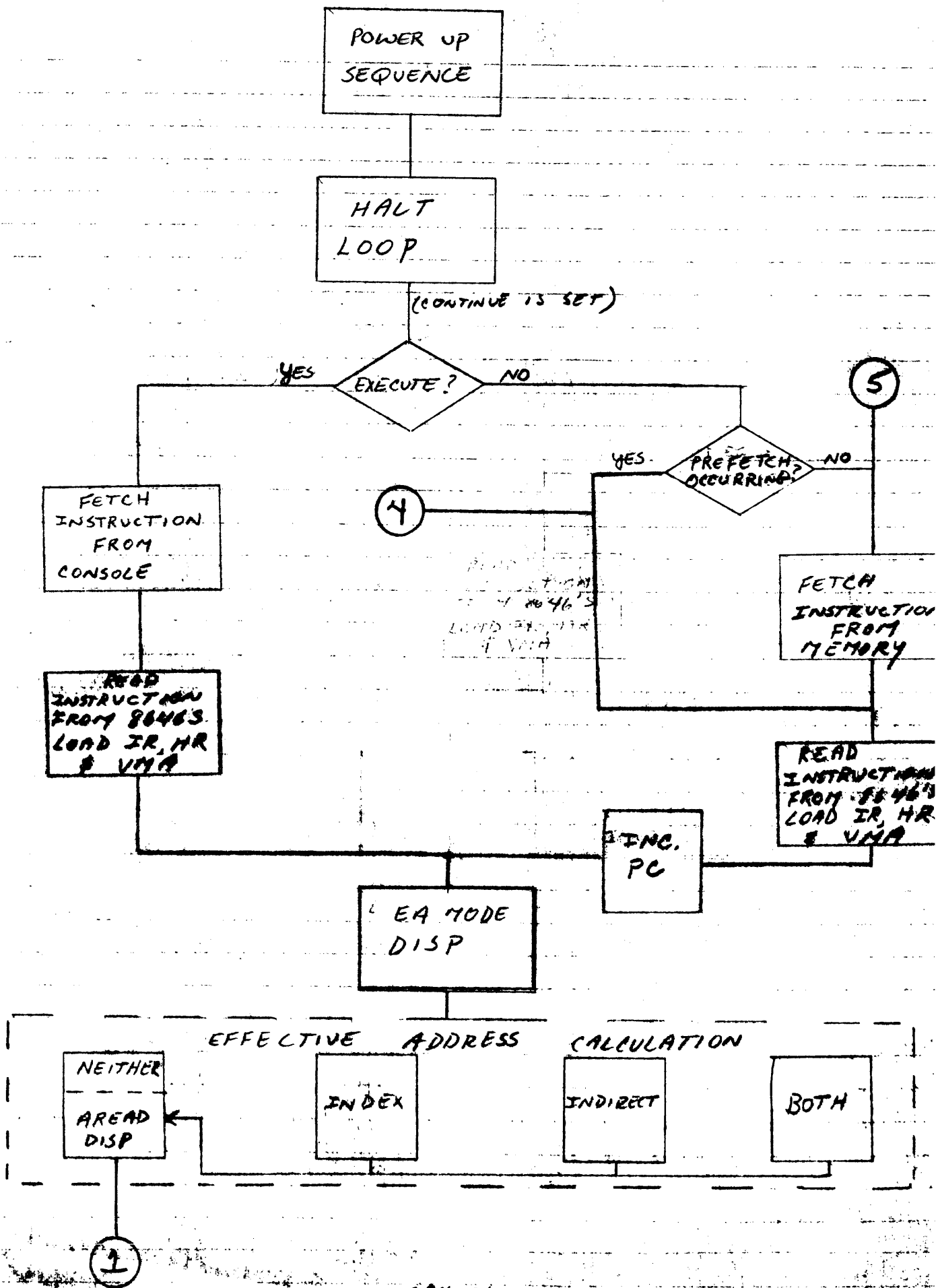
ON WRITES & READS
TO MEMORY & ON
WRITES TO THE ACS
THE M-CODE INSTR WHICH
DOES THE MEM WRIT
GENERATES

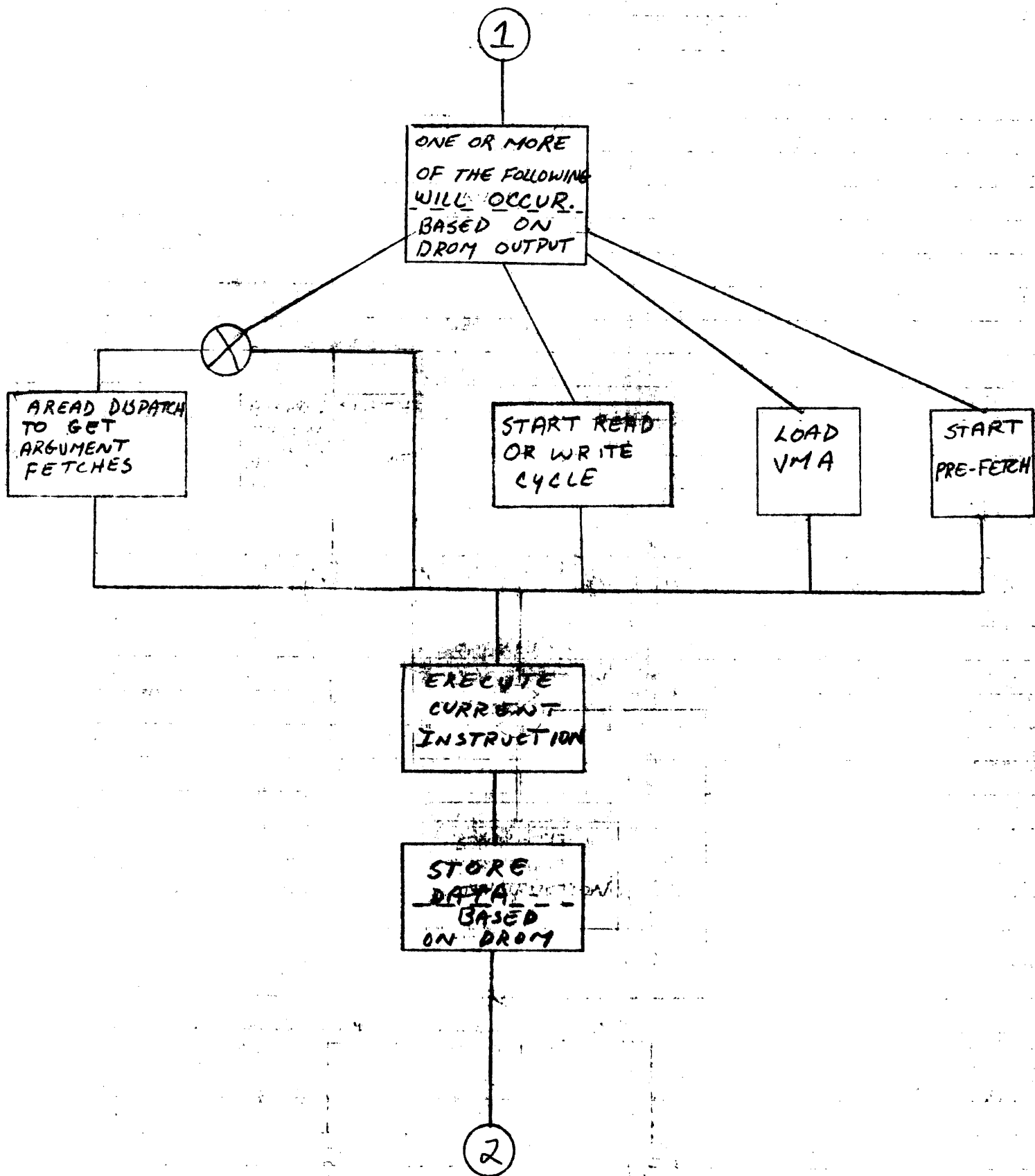
CACHE WRITE L
AND PAGE WRITE L
(AS LONG AS PAGE WRITE
EN & ISSCT
ON THE LAST PART OF

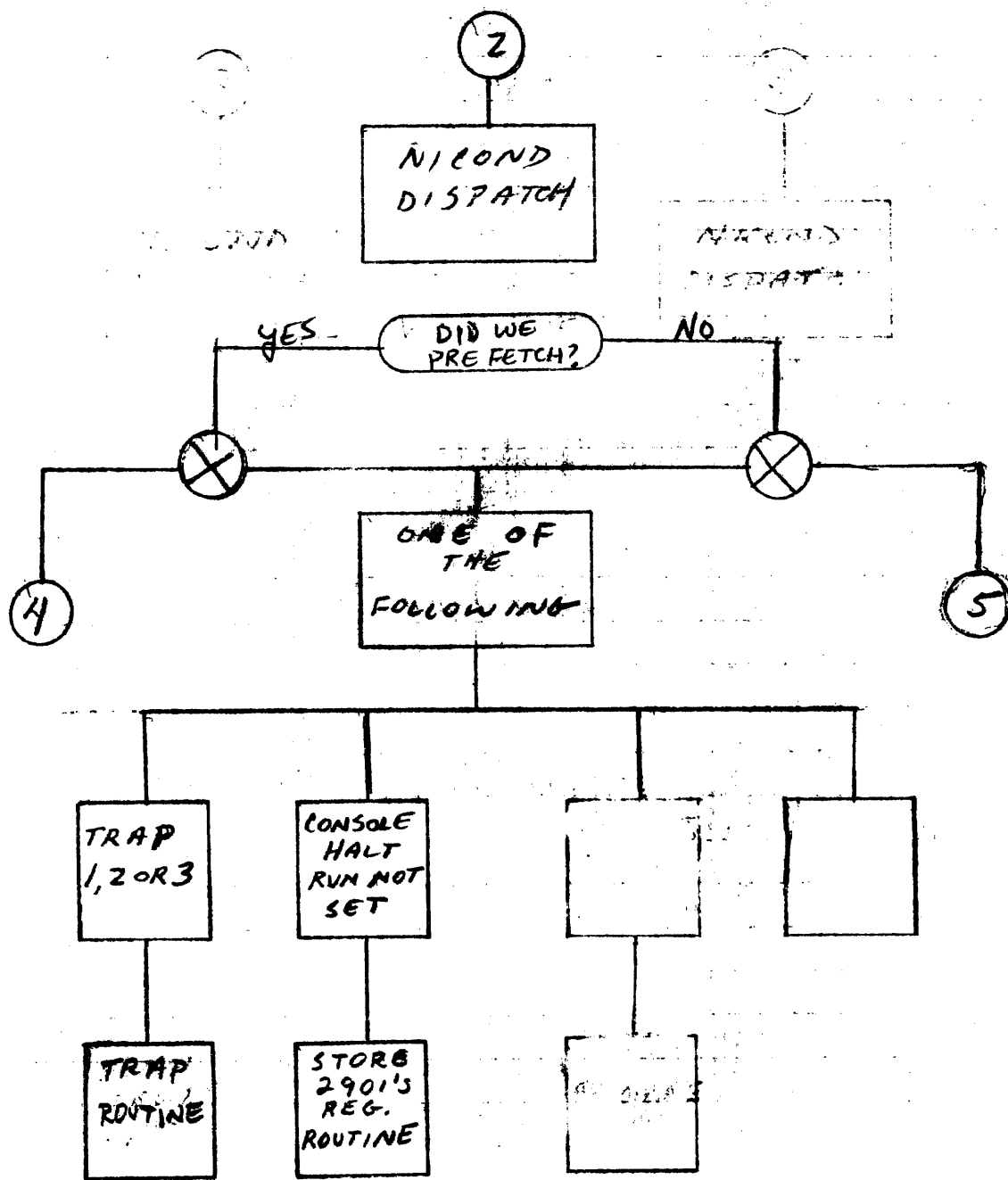
CPH-35

STACK, CACHE &
PAGE TABLE WRITES
IN RELATION TO
CPU CLOCKS.

Basic CPU Operation







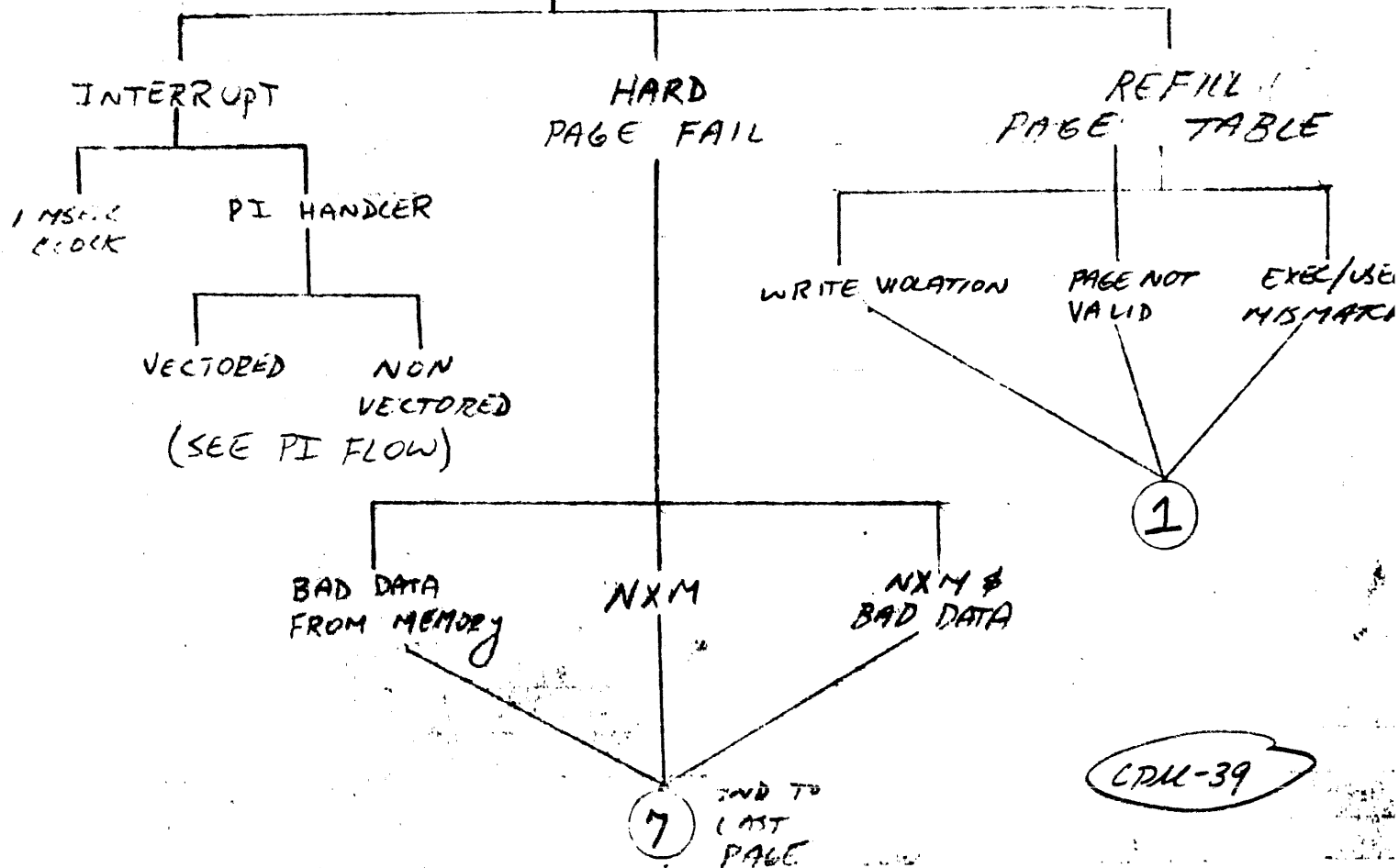
PAGE FAIL HANDLER (KI MODE)

PAGE FAIL
OCCURRED

SAVE 2901
REGISTER
(AR, ARX, BR & BRX)
PUT ADDRESS WHICH PAGE FAILED IN THE BRX

BRING PAGE FAIL
CODE TO
2901'S

DISPATCH
ACCORDING TO
PAGE FAIL CODE



1

CLEAR ANY PAGE
FAILS
(LOAD VMA WITH ZEROES)

CAUSES "AC REF."
CLEARS DPM

ARE WE DOING
A WRITE TEST? (CHECK IF BIT 5 SET IN BRX)
WHERE BRX CONTAINS WORD
WHICH PAGE FAILED.

YES

NO

SET BITS
IN BRX

SAVE THREE
BITS IF THEY
WERE SET
(USER ADDR, WRITE REF
& PAGED REF
& THE VIRTUAL ADDRESS
& PUT IN THE BRX

QUESTION
LEWINE
ABOUT
THIS

IF IT IS
SET IN 4

AND
CLEARS OTHER
BITS

SHIFT THE VIRTUAL ADDR.
9 PLACES TO THE RIGHT
SO THE PAGE # IS IN THE
RIGHTMOST 9 BITS
MASK OUT ALL OTHER BITS
PUT IT IN THE BR [0, PAGE #)

IS THIS KI MODE
REFILL?
(CHECK IF THE 40 BIT IN
THE LEFT HALF OF EBR
IS SET)

NO

YES

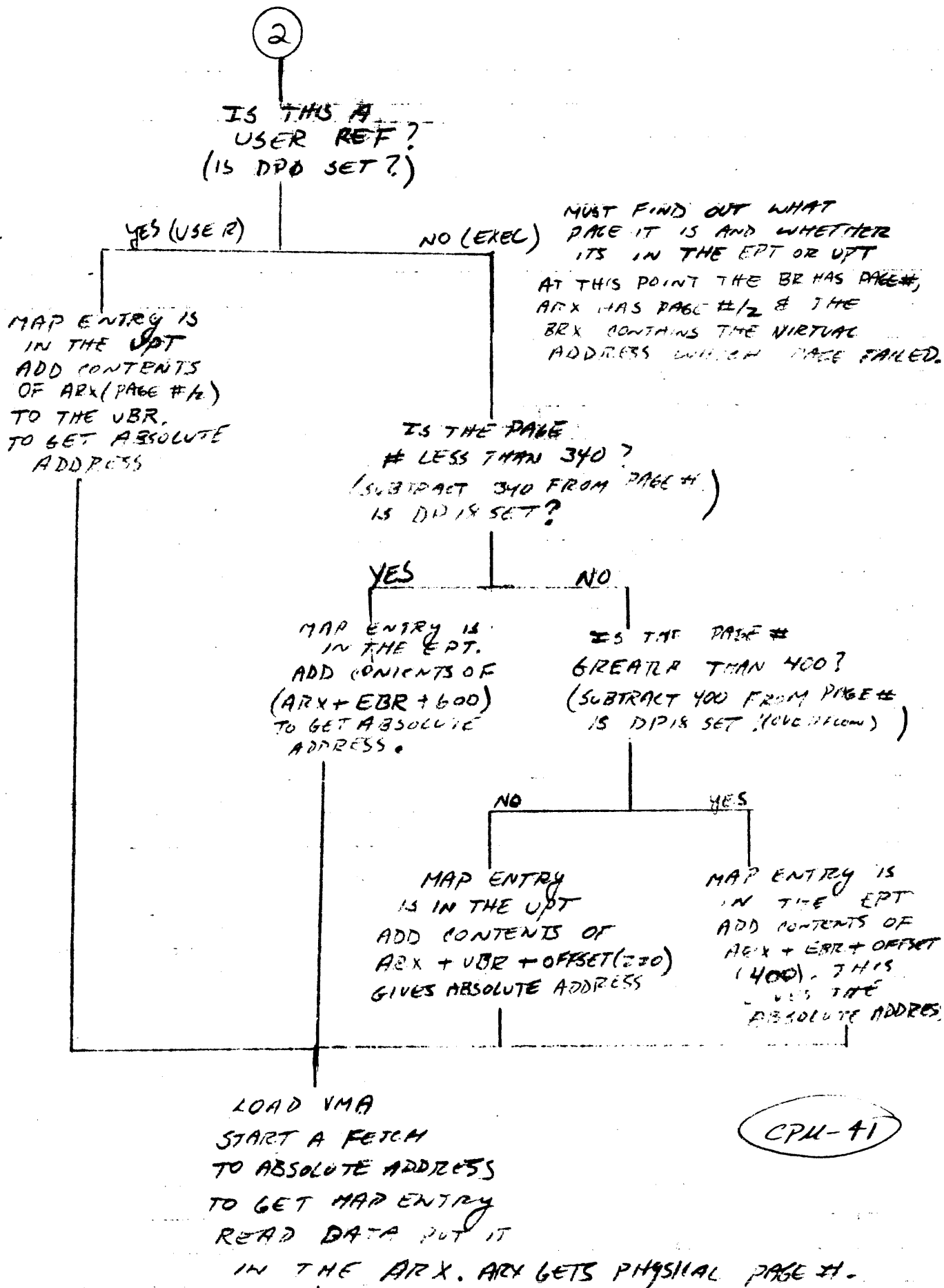
CPU-40

KL MODE
(NOT COVERED YET)

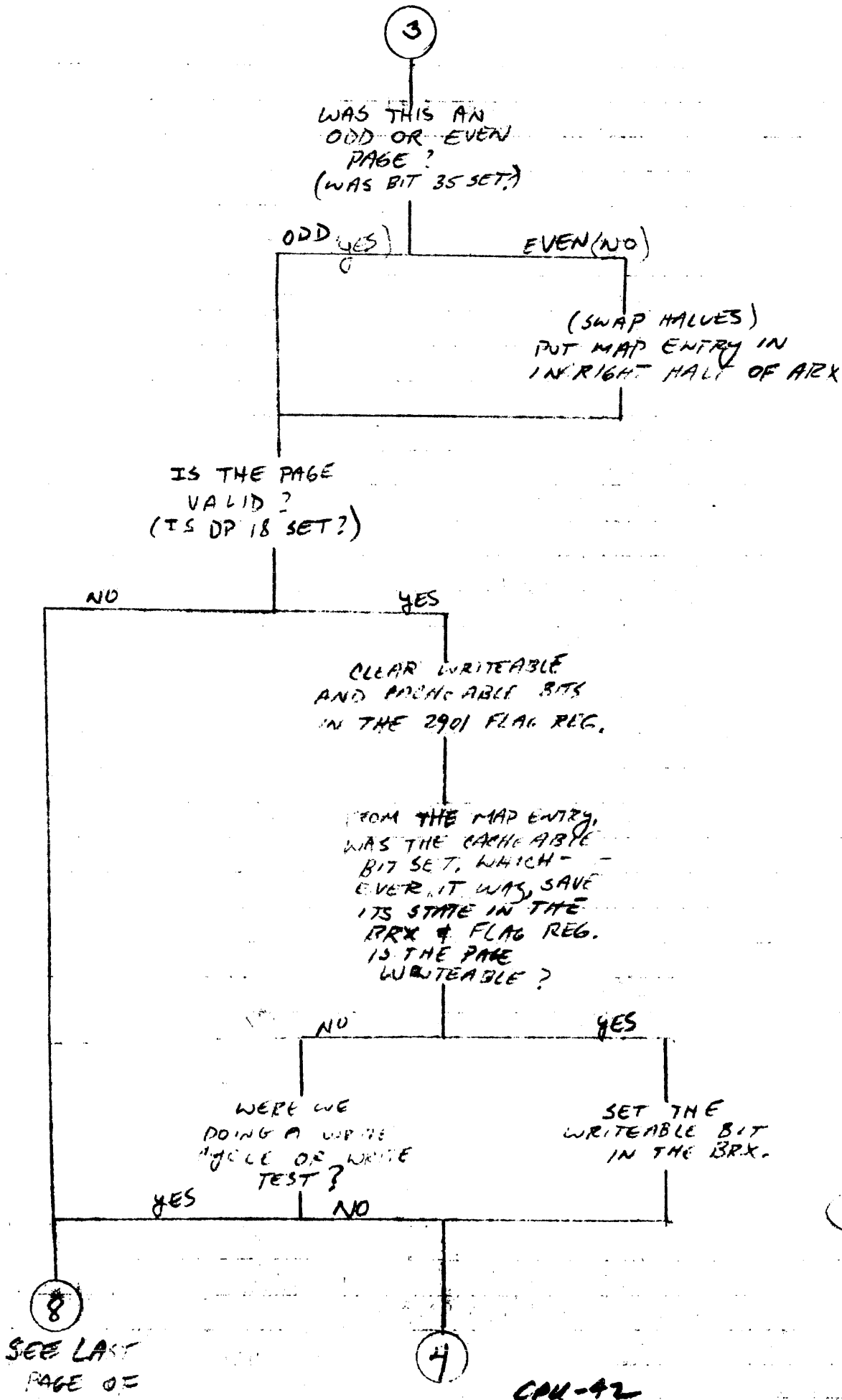
KI MODE
DIVIDE PAGE NUMBER
BY 2 AND PUT
IN THE ARX.

2

FIND OUT WHERE IN UPT OR EPT IT LIES



CPU-41



4

ARX CONTAINS PHYSICAL
PAGE # & BRX HAS THE
CACHEABLE, WRITABLE & USER
BITS FOR THE PAGE

IS THIS A
MAP INSTRUCTION?
(IS THE 400000 BIT SET
IN THE FIG REG?)

NO

YES

RESTORE VMA
FROM MICROCODE
WORK SPACE
PUT IT IN THE AR
& THE BR

EXECUTE MAP
INSTRUCTION

STORE DATA

NICOND DISPATCH

CLEAR THE
READ, WRITE &
WRITE TEST BITS
IN THE RH OF
THE BR.

THESE BITS WOULD
START A CYCLE IF
SET & ON 16 IN THE

RELOAD THE CONTENTS
OF THE VMA, WITH THE
READ, WRITE & WRITE TEST
BITS NOT SET, FROM THE DP.

MASK THE ARX SO
ALL THAT REMAINS
IS THE PHYSICAL
PAGE #.

SET THE VALID BIT
AND PUT THAT & ~~W~~ THE
PAGE # IN THE BRX

5

CPM-43

5

QUESTION: THE BRX AT
THIS POINT CONTAINS
THE CACHEABLE & WRITEABLE
BITS NO NEED TO LOAD
THEM AGAIN.
(DO IN KL MODE)

CHECK THE CACHEABLE
& WRITEABLE BITS
IN THE FLAG REG. IF
THEY ARE SET, SET
THEM IN THE BRX
ALSO.

READ THE CONTENTS
OF THE BRX INTO
THE DP AND LOAD
THE PAGE TABLE.
THE VMA WAS LOADED
PREVIOUSLY, ADDRESSING
THE PAGE WHICH ~~IS~~ ^{IS A PAGE OF}
PAGE TABLE (NOT VALID, WRITE VIOLATION OR PAGE USER MISMATCH)

WE WRITE INTO THE PAGE TABLE
VIA SPEC/PAGE.WRTE, LOADING THE ADDRESSED
WITH THE APPROPRIATE VALID, WRITEABLE,
CACHEABLE & USER BITS AND THE
PHYSICAL PAGE NUMBER VIA THE
DP.

SAVE ADDR OF BRX
IN MICROCODE ADDRESS SPACE

RELOAD THE VMA FROM A.R
THIS TIME WITH THE FLAGS
FOR THE (READ, WRITE OR WRITE TEST)
WE LOAD THIS VIA THE DP.
WAIT FOR THE PREVIOUS CYCLE TO COMPLETE

CPM-44

6

6

START THE CYCLE
(AND HOPE WE DON'T
PAGE FAIL AGAIN)

WE WILL THEN
RETURN TO THE
MICROCODE INSTRUCTION
WHICH WE PAGE FAILED
ON, TO BEGIN WITH 6
(BECAUSE THAT U-CODE ADD.
WAS PUT ON TOP OF THE
STACK)

WE HAVE THUS
COMPLETED THE PAGE
REFILL.

7

HERE FOR HARD PAGE FAILS
(IE NEXT OR/AND BAD DATA)

SAVE BR
IN MICROCODE
WORKSPACE

PUT THE CONTENTS
OF THE VMA INTO
THE BR

CLEAR ANY
PAGE FAILS
BY LOADING THE
VMA WITH Φ .

SAVE THE FLAGS (IN THE BR)
OF THE INSTRUCTION
BEING EXECUTED
(I.E. FETCH, WRITE CYCLE
AND/OR PHYSICAL REF.)

THE LEFT HALF OF
THE BRX CONTAINS THE
PAGE FAIL CODE.
IN THE RIGHT HALF
PUT THE ADDRESS WHICH
CAUSED THE
PAGE FAIL
THE ENTIRE PAGE FAIL
WORD IS NOW IN THE BRX

8

CPM-46

8

GOING TO DO A
PAGE FAIL TRAP
ARE WE DOING A
PI CYCLE?
(IS THE PI BIT ON)
IN PI REG

NO

YES

WERE WE
DOING A FETCH
(WAS THE FETCH BIT
SET IN THE VMA FLAGS)

YES

NO

BACK-UP
PC

DO CLEANUP
DISPATCH

WRITE THE
PAGE FAIL WORD
IN UBR+500

PUT OLD PC
IN (UBR+500+1)

READ THE EA
OUT OF (UBR+500+2)
THIS IS THE NEW PC (KI MODE)
WE RETURN THE CONTENTS OF
THE EA WHICH IS USED BY
THE MONITOR TO DO AS IT
WISHES.

HERE WITH HALT
CODE IN THE AR

IO PAGE FAIL
(LOAD) VMA WITH Φ , THIS
STOPS ANY CYCLES IN PROGRES

STORE HALT CODE
IN LOCATION Φ
AND PC IN LOC. 1

WRITE THE HALT
STATUS BLOCK
TELL THE CONSOLE
WE HAVE HALTED

RETURN TO
THE HALT LOOP

CPU-4T

KS10 MEMORY FETCH DESCRIPTION

A FETCH FROM MEMORY WILL OCCUR WHEN THE CPU WANTS TO FETCH AN INSTRUCTION, FETCH AN ARGUMENT NEEDED FOR INSTRUCTION EXECUTION OR ON A PAGE TABLE REFILL (OCCURS WHEN PAGE IS INVALID). THE MEMORY READ CYCLE IS INITIATED BY THE MICROCODE INSTRUCTION WHICH CONTAINS EITHER THE "START READ" OR "FETCH" MACRO. THE VMA WOULD BE LOADED AT THIS TIME OR HAS BEEN LOADED PREVIOUSLY. WHEN THE MACRO IS USED THE MICROCODE WILL HAVE CRA6 MEMORY FUNCTION AND CRM2 #16 SET WITHIN THE INSTRUCTION. THE OTHER WAY OF STARTING THE CYCLE IS BY A "AREAD DISP". AT AREAD DISP TIME, WHICH OCCURS DURING EA CALC, THE SIGNAL DPEA DROM COND FUNC OUT OF THE DROM MUST BE GENERATED BY THE OP CODE OF THE INSTRUCTION BEING EXECUTED. CRA6 MEM FUNCTION AND CRM2 #17 WILL ALSO BE SET WITHIN THE U-CODE INSTRUCTION DOING THE AREAD DISP. REFERRING TO DPM5, THE RESULT OF STARTING THE CYCLE IN EITHER FASHION IS THAT DPM5 READ DELAY EN IS PRODUCED. THE LAST 75 NS OF THE U-INSTRUCTION THAT GENERATED DPM5 MEM EN, WILL, WITH

DPM5 SYNC E L, SET DPM5 START CYCLE, ON THE NEXT T CLK, BECAUSE OF DPM5 START CYCLE-----DPM5 READ DELAY EN, DPM6 MEM CYCLE, AND DPM5 REQUEST CPU WILL BE SET. DPM5 BUS REQUEST GOES TO THE ARBITRATION LOGIC ON THE CONSOLE BOARD(CSL2). CSL2 BUS GRANT CPU IS RETURNED FROM THE CONSOLE BOARD. ON THE NEXT T CLK IT WILL CLEAR OUT DPM5 BUS REQUEST AND GENERATE COM/ADR CYCLE (SEE DPMC). REFERRING TO DPM 8 & 9 THE PROPER SELECTION OF THE 4*2 MIXER IS MADE DEPENDING ON WHETHER IT IS A PHYSICAL OR PAGED REFERENCE. DPM5 BUS REQUEST WILL STILL BE SET WHEN WE START THE COM/ADR CYCLE. ASSUMING IT IS A VALID PAGED REF, THE DATA SIGNALS WHICH ARE TRANSMITTED ONTO THE BUS WITH THE COM/ADR CYCLE INCLUDE: BUS DATA 01, FROM DPM5 MEM READ, SIGNIFYING A MEMORY READ CYCLE; BUS DATA(16-26) FROM DPM6 PAGE(16-26) WITH THE PHYSICAL PAGE NUMBER OUT OF THE PAGE TABLE; BUS DATA(27-35) FROM DPM4 VMA(27-35) WITH THE LINE NUMBER. THIS SHOULD ALL BE SEEN WITH THE BUS COM/ADR SIGNAL. THE MEMORY SHOULD RESPOND BY ISSUING MEMORY BUSY ON THE SECOND T CLK AFTER THE COM/ADR CYCLE. IF THE MEMORY DOES NOT RESPOND WITHIN COM/ADR+2, DPMC NXM ERR WILL OCCUR GENERATING AN APR INTERRUPT ON DPMB.

ASSUME WE DID GET AN ADDRESS MATCH WITH MEMORY AND THE DATA WASN'T BAD. THE DATA WILL BE TRANSMITTED BACK TO THE CPU WITH BUS DATA CYCLE. THIS WILL OCCUR ON THE SECOND OR THIRD CYCLE AFTER COM/ADR, DEPENDING ON THE WHETHER THE DATA BEING FETCHED WAS GOOD OR CORRECTABLE. IF BUS BAD DATA CYCLE IS TRANSMITTED FROM MEMORY TO THE CPU IT WILL CAUSE AN APR INTERRUPT(DPMA). THE MICROCODE INSTRUCTION WHICH CONTAINS THE "MEM READ" MACRO IS USED TO COMPLETE THE MEMORY CYCLE. THE INSTRUCTION WILL HAVE CRM2#16 AND CRA6 MEM FUNCTION SET, PRODUCING DPM5 MEM EN AGAIN. THIS, WITH DPM6 MEMORY CYCLE WILL GENERATE DPM5 MEMORY DELAY. THIS SIGNAL IS USED TO HOLD UP THE CPU CLOCK AND WAIT FOR MEMORY. FOR THE READ CYCLE, DPM5 READ DELAY IS SENT TO THE CONSOLE BOARD(CSL5) TO STOP THE NEXT CPU CLOCK. THE MEMORY COMES BACK WITH BUS DATA CYCLE. THIS WILL CAUSE DPMC MEM STROBE TO CLEAR OUT DPM5 READ DELAY EN. ONCE THIS SIGNAL IS CLEARED OUT IT WILL FREE-UP THE CPU CLOCK AND WE WILL LOAD AND EXECUTE THE NEXT MICROCODE INSTRUCTION.

THE DATA HAS BEEN LATCHED INTO THE 8646'S (DPM8&9) FROM THE BUS AS DPM(8&9) MB(00-35). THE DATA WILL GO THROUGH THE DPM MIXER(DPM1&2) AS DPM(1&2) DBM(00-35). IT FLOWS INTO THE D-BUS MIXER (DPE1&2) AND OUT AS D-BUS(00-35). THE FINAL DESTINATION, 2901 REG OR AC, IS SELECTED BY THE MICROCODE INSTRUCTION DOING THE "MEM READ". THIS COMPLETES THE MEMORY FETCH. NOTE: ANYTIME WE DO FETCH FROM MEMORY WE WRITE IT INTO THE CACHE. CACHE AND RAMFILE WRITE ARE GENERATED ON DPMA. THE VIRTUAL PAGE ADDRESS(VMA18-26) WILL BE WRITTEN INTO

THE CACHE DIRECTORY IN THE LOCATION SPECIFIED BY THE LINE NUMBER (VMA27-35). IT IS STROBED IN WITH DPMA CACHE WRITE (SEE DPM7). THE DATA WE FETCHED FROM MEMORY WILL BE WRITTEN INTO THE CACHE (DPE7&8). THE ADDRESS TO BE WRITTEN INTO COMES FROM DPE6 RAMFILE ADR (00-09). IT WAS PRODUCED FROM DPE5 VMA (26-35) WHICH IS INPUT TO THE RAMFILE MIXER ON DPE6. WE STROBE THE DATA FETCHED INTO THE CACHE WITH DPE5 RAMFILE WRITE. THE DATA ENTERS THE CACHE FROM DPE (3&4) D-BUS (00-35).

KS10 CACHE HIT DESCRIPTION

A CACHE HIT CAN BE GENERATED ON A READ FROM MEMORY. A FETCH FROM MEMORY WILL OCCUR WHEN THE CPU WANTS TO FETCH AN INSTRUCTION, FETCH AN ARGUMENT NEEDED FOR INSTRUCTION EXECUTION OR ON A PAGE TABLE REFILL (OCCURS WHEN PAGE IS INVALID). THE MEMORY READ CYCLE IS INITIATED BY THE MICROCODE INSTRUCTION WHICH CONTAINS EITHER THE "START READ" OR "FETCH" MACRO. THE VMA WOULD BE LOADED AT THIS TIME OR HAS BEEN LOADED PREVIOUSLY. WHEN THE MACRO IS USED THE MICROCODE WILL HAVE CRA6 MEMORY FUNCTION AND CRM2 #16 SET WITHIN THE INSTRUCTION. THE OTHER WAY OF STARTING THE CYCLE IS BY A "AREAD DISP". AT AREAD DISP TIME, WHICH OCCURS DURING EA CALC, THE SIGNAL DPEA DROM COND FUNC OUT OF THE DROM MUST BE GENERATED BY THE OP CODE OF THE INSTRUCTION BEING EXECUTED. CRA6 MEM FUNCTION AND CRM2 #17 WILL ALSO BE SET WITHIN THE U-CODE INSTRUCTION DOING THE AREAD DISP. REFERRING TO DPM5, THE RESULT OF STARTING THE CYCLE IN EITHER FASHION IS THAT DPM5 READ DELAY EN IS PRODUCED. THE LAST 75 NS OF THE U-INSTRUCTION THAT GENERATED DPM5 MEM EN, WILL, WITH DPM5 SYNC E L, SET DPM5 START CYCLE, ON THE NEXT T CLK, BECAUSE OF DPM5 START CYCLE-----DPM5 READ DELAY EN, DPM6 MEMORY CYCLE, AND DPM5 REQUEST CPU WILL BE SET. DPM5 BUS REQUEST GOES TO THE ARBITRATION LOGIC ON THE CONSOLE BOARD (CSL2). IF THE REQUESTED DATA IS IN CACHE, A CACHE HIT WILL OCCUR AT THIS TIME.

ASSUMING THE DESIRED ARGUMENT IS IN CACHE, THE LOCATION BEING ADDRESSED HAS AN ENTRY IN THE PAGE TABLE CONSISTING OF THE PHYSICAL PAGE NUMBER, VALID BIT AND CACHEABLE BIT BOTH SET. THE HARDWARE SEES THE PAGE IS VALID AND CHECKS THE CACHE DIRECTORY (ADDRESSED BY THE VIRTUAL LINE #) TO SEE IF THE LINE IS IN CACHE. THE CACHE DIRECTORY CONTAINS VALID, PARITY, AND USER BITS IN ADDITION TO THE VIRTUAL PAGE NUMBER FOR THAT LINE. IF THE OUTPUTS OF THE CACHE DIRECTORY MATCH THE VIRTUAL PAGE ADDRESS, IF THE ENTRY IS VALID AND IF THE PARITY IS GOOD (ON THE VIRTUAL PAGE ADDRESS BITS) DPM 7 CACHE HIT EN WILL BE TRUE.

****NOTE:** SINCE A HIT DID OCCUR WE KNOW THAT THE DATA IN CACHE HAS BEEN FETCHED BEFORE, WE ALSO KNOW THAT IT IS THE LINE THAT WE NEED (I.E. THE SAME LINE NUMBER FROM DIFFERENT PAGES CANNOT COEXIST IN CACHE). THIS IS SO BECAUSE WE ADDRESS THE CACHE DIRECTORY WITH LINE # ADDRESS BITS. THIS IS A DEPARTURE FROM THE KL10 CACHE WHICH IS A FOUR PART STRUCTURE.

- DPM6: VMA18-26 ADDRESS THE PAGE TABLE YIELDING OUTPUTS PAGE VALID, WRITEABLE, CACHEABLE, USER, PARITY, AND PHYSICAL PAGE ADDRESS BITS 16-26.
- DPM7: VMA 27-35 ADDRESS THE CACHE DIRECTORY YIELDING OUTPUTS CACHE VALID, USER, PARITY, AND VIRTUAL PAGE ADDRESS BITS (AS CACHE 18-26). NAND GATE AT E513 WILL GENERATE CACHE HIT EN IF THE FOLLOWING CONDITIONS ARE MET: THERE ARE NO PAGE FAILS, PAGING IS ENABLED, WE ARE DOING A READ CYCLE, THE CACHE IS NOT INHIBITED, THE ADDRESS IS CACHEABLE, AND THE PAGE NUMBER IN THE CACHE DIRECTORY MATCHES THE VIRTUAL ADDRESS WE WANT TO REFERENCE. AS LONG AS THE VMA IS NOT BEING LOADED THE OUTPUT DPM7 CACHE HIT WILL BE TRUE.
- DPM5: THE MEM CYCLE ALREADY BEGUN MUST BE ABORTED. CACHE HIT EN CLEARS THE KS10 BUS REQUEST. CACHE HIT GENERATES DPM5 STOP MAIN MEMORY. THESE GO TO DPMC TO ABORT THE CYCLE AS SOON AS POSSIBLE AND GENERATE MEM CYCLE ABORT TO THE MEMORY CONTROLLER.

PREVIOUSLY, WHEN DPM4 VMA WAS LOADED FROM DP, ANOTHER COPY WAS LOADED ON DPE5 VIA DP 26-35. THESE ADDRESS THE CACHE DIRECTORY. THE MICROCODE IS SELECTING VMA 26-35 AS THE INPUTS TO THE RAMFILE ADDRESS MIXER AT THIS TIME, ANTICIPATING A CACHE HIT. RAMFILE ADR 0-11 WILL ADDRESS THE CACHE ON DPE7 AND DPE8, YIELDING RAMFILE 00-35. DPM7 CACHE HIT EN GENERATED DPM5 FORCE RAMFILE WHICH FORCES THE D-BUS

CPH-50

SELECT SIGNALS ON DPE3 TO GATE RAMFILE 00-35 ,RATHER THAN MEMORY DATA (DBM), ON TO THE D-BUS,THE D-BUS DATA 00-35 IS INPUT TO THE 2901'S ON DPE1 & DPE2 AND STROBED INTO THE 2901 REGISTER OR AC SELECTED BY THE MICROCODE INSTRUCTION DOING THE "MEM READ".

"DISPATCH ROM DEFINITIONS"

```

        .DCODE
A/= <2:5>                                ;OPERAND FETCH MODE
        READ=0                            ;READ
        WRITE=1                           ;WRITE
        DREAD=2                           ;DOUBLE READ
        DBLAC=3                           ;DOUBLE AC
        SHIFT=4                           ;SIMPLE SHIFT
        DSHIFT=5                          ;DOUBLE SHIFT
        FPI=6                             ;FLOATING POINT IMMEDIATE
        FP=7                              ;FLOATING POINT
        RD=PF=10                           ;READ, THEN START PREFETCH
        DFP=11                            ;DOUBLE FLOATING POINT
        IOT=12                            ;CHECK FOR IO LEGAL THEN SAME AS I

B/= <8:11>                                ;STORE RESULTS AS
        SELF=4                            ;SELF
        DBLAC=5                           ;DOUBLE AC
        DBLB=6                            ;DOUBLE BOTH
        AC=15                             ;AC
        MEM=16                            ;MEMORY
        BOTH=17                           ;BOTH

;B=FIELD WHEN USED IN FLOATING POINT OPERATIONS
ROUND/= <8>                               ;ROUND THE RESULT
MODE/= <9>                                ;SEPERATE ADD/SUB & MUL/DIV ETC.
FL=B/= <10:11>                            ;STORE RESULTS AS
        AC=1                              ;AC
        MEM=2                             ;MEMORY
        BOTH=3                            ;BOTH

J/= <12:23>                               ;DISPATCH ADDRESS (MUST BE 1400 TO 1775)

ACDISP/= <24>                             ;DISPATCH ON AC FIELD
I/= <25>                                  ;IMMEDIATE DISPATCH, DISP/AREAD DOES A DISP/DROM
; IF THIS BIT IS SET,
READ/= <26>                               ;START A READ AT AREAD
TEST/= <27>                               ;START A WRITE TEST AT AREAD
WRITE/= <28>                              ;START A MEMORY CYCLE ON BWRITE
VMA/= <29>D,1                             ;LOAD THE VMA ON AREAD
A WRITE/= <30>                             ;START A WRITE ON AREAD

```

CRA6 MEMORY FUNCTION

THIS ONE MICROCODE BIT IS USED TO INITIATE A MEMORY CYCLE OR TO LOAD THE VMA REGISTER. THE 18 BIT MAGIC # FIELD IS FURTHER DECODED TO EVOKE THE REQUIRED FUNCTION.

WHEN #13 IS TRUE, THE CYCLE IS CONTROLLED BY DP BITS. THIS IS USED TO REPEAT A REQUEST THAT PAGE FAILED, OR LOADING THE VMA DURING IO INSTRUCTIONS. WHEN #12 IS TRUE THE CYCLE IS CONTROLLED BY THE AREAD FUNCTION WHICH CONTROLS DROM OUTPUTS.

#	MEMORY FUNC	DP FUNC	A READ FUNC
0	FORCE USER	FORCE USER	--
1	FORCE EXEC	--	--
2	FETCH	FETCH	--
3	READ	READ	DROM READ CYCLE
4	WRITE-TEST	WRITE-TEST	DROM WRITE TEST
5	WRITE	WRITE	DROM WRITE CYCLE
6	--	--	--
7	INH CACHE	INH CACHE	--
8	PHYSICAL(NO PAGING)	PHYSICAL	--
9	PXCT(SEE UCODE)	PREVIOUS	--
10	PXCT(")	I/O CYCLE	--
11	PXCT(")	WRU CYCLE	--
12	A READ FUNC	VECTOR CYCLE	--
13	DP FUNC	I/O BYTE CYCLE	--
14	LOAD VMA	--	--
15	EXTENDED ADR	--	--
16	UNCONDITIONAL CYCLE	--	--
17	CONDITIONAL CYCLE(DROM)	--	--

"MICROCODE FIELDS -- SPEC"

THE FOLLOWING SPECIAL FUNCTION ARE DECODED ON DPE1, DPE5, AND DPMA:

1S1	EFFECT	!	CRA6 SPEC	!	CRA6 SPEC	!	CRA6 SPEC
1P1	ON SHIFT	!	EN 40	!	EN 20	!	EN 10
1E1	PATHS	!	E102 ON DPE5	!	E101 ON DPE5	!	E410 ON DPMA
1C1	(SEE DPE1)	!		!	E411 ON DPMA	!	E113 ON CRA2
121	NORMAL	!	CRY 18 INH	!	PREVIOUS	!	
111	ZERO	!	TR LOAD	!	XR LOAD	!	CLR 1 MSEC
121	ONES	!	<SPARE>	!	<SPARE>	!	CLR 10 LATCH
131	ROT	!	PI LOAD	!	APP FLAGS	!	CLR 10 BUSY
141	ASHC	!	ASH TEST	!	SET SWEEP	!	PAGE WRITE
151	LSHC	!	EXP TEST	!	APP EN	!	<SPARE>
161	DIV	!	PC FLAGS	!	EXCT OFF	!	EXCT EN
171	ROIC	!	AC BLOCKS EN	!	MEA CLP	!	WAIT

THE DPM BOARD USES THE SPEC FIELD TO CONTROL THE DBM MIXER, AS FOLLOWS:

1	5	!		!
1	2	!	ACTION WHEN DPM	!
1	3	!	SELECTS DP	!
1	0	!	GET DP BITS	!
1		!	GET SCAD 1-7	!
1	4	!	ALL	!
1		!	NONE	!
1	1	!	8-35	!
1		!	0-6	!
1	2	!	10-6 AND 14-35	!
1		!	7-13	!
1	3	!	10-13 AND 21-35	!
1		!	14-20	!
1	4	!	10-20 AND 29-35	!
1		!	21-27	!
1	5	!	10-27 AND 35	!
1		!	28-31	!
1	6	!	SAME AS ZERO	!
1	7	!	SAME AS ZERO	!

;THE SPEC FIELD IS DEFINED AS A 6-BIT FIELD, THE TOP 3 BITS
 ; ARE SPEC SEL A, SPEC SEL B, AND SPEC SEL C, THE LOW 3 BITS ARE
 ; THE SELECT CODE,

SPEC/= <51;56>D,0

#=10	;DECODE # BITS
CLRCLK=11	;CLEAR 1MS NICOND FLAG
CLR IO LATCH=12	;CLEAR IO LATCH
CLR IO BUSY=13	;CLEAR IO BUSY
LDPAGE=14	;WRITE PAGE TABLE
LDPXCT=16	;LOAD PXCT FLAGS
WAIT=17	;MEM WAIT
PREV=20	;FORCE PREVIOUS CONTEXT
LOADXR=21	;LOAD XR #, USES PXCT FIELD TO SELECT ; CORRECT AC BLOCK
APR FLAGS=23	;LOAD APR FLAGS
CLRC SH=24	;CLEAR CACHE
APR EN=25	;SET APR ENABLES
MEMCLR=27	;CLEAR PAGE FAULT CONDITION
SWEEP=34	;SET SWEEP
PXCT OFF=36	;TURN OFF THE EFFECT OF PXCT
INHCRY18=40	;INHIBIT CARRY INTO LEFT HALF
LOADIR=41	;LOAD THE IR
LDPI=43	;LOAD PI SYSTEM
ASHOV=44	;TEST RESULT OF ASH
EXPTST=45	;TEST RESULT OF FLOATING POINT
FLAGS=46	;CHANGE PC FLAGS
LDACBLK=47	;LOAD AC BLOCK NUMBERS
LDINST=61	;LOAD INSTRUCTION

;THE SPEC FIELD IS REDEFINED WHEN USED FOR BYTE MODE STUFF
 BYTE/= <54;56>

BYTE1=1
 BYTE2=2
 BYTE3=3
 BYTE4=4
 BYTE5=5

;THE SPEC FIELD IS REDEFINED WHEN USED TO CONTROL SHIFT PATHS
 SHSTYLE/= <54;56>

NORM=0	;2 40-BIT REGISTERS
ZERO=1	;SHIFT ZERO INTO 36 BITS (ASH TOP 2901)
ONES=2	;SHIFT IN ONES
ROT=3	;ROTATE
ASHC=4	;ASHC
LSHC=5	;LSHC
DIV=6	;SPECIAL DIVIDE
ROTC=7	;ROTATE DOUBLE

```

"MICROCODE FIELDS == DISPATCH"
|=====|
| D |      CRA1 |      CRA1 |      DPEA |
| I |      DISP |      DISP |      DISP |
| S |      10   |      20   |      40   |
| P |           |           |           |
|=====|
| 0 |    DIAG ADR |    DIAG ADR |      0    |
|=====|
| 1 |    RETURN   |    RETURN   |    DP 18-21 |
|=====|
| 2 |  MULTIPLY   |      J      |      J      |
|=====|
| 3 |   PAGE FAIL |   A-READ    |   A-READ    |
|=====|
| 4 |   NICOND    |  NOT USABLE |   NORM      |
|=====|
| 5 |    BYTE     |  NOT USABLE |    DP 32-35 |
|=====|
| 6 |   EA MODE   |  NOT USABLE |   DROM A    |
|=====|
| 7 |    SCAD     |  NOT USABLE |   DROM B    |
|=====|
;NOTE: DISP EN 40 & DISP EN 10 ONLY CONTROL THE LOW 4 BITS OF THE
;      JUMP ADDRESS, DISP EN 20 ONLY CONTROLS THE HI 7 BITS, TO 00
;      SOMETHING TO ALL 11 BITS BOTH 20 & 40 MUST BE ENABLED.

```

DISP/=<57:62>D,70

CONSOLE=00	;CONSOLE DISPATCH
DROM=12	;DROM
AREAD=13	;A-READ
DP LEFT=31	;DP 18-21
NORM=34	;NORMALIZE
DP=35	;DP 32-35
ADISP=36	;DROM A FIELD
BDISP=37	;DROM B FIELD
RETURN=41	;RETURN
MUL=62	;MULTIPLY
PAGE FAIL=63	;PAGE FAIL
NICOND=64	;NEXT INSTRUCTION DISPATCH
BYTE=65	;BYTE SIZE AND POSITION
EAMODE=66	;EFFECTIVE ADDRESS MODE
SCAD0=67	;J12 IF SCAD BIT 0 = 1

```

"MICROCODE FIELDS == SKIP"
|=====|
| S |      CRA2      |      DPEA      |      DPEA      |
| K |      SKIP      |      SKIP      |      SKIP      |
| I |      10        |      20        |      40        |
| P |                |                |                |
|=====|
| 0 |      0          |      0          |      0          |
|-----|
| 1 |      <SPARE>    |      CRY 02    |      CARRY OUT  |
|-----|
| 2 |      AD=0        |      ADL SIGN   |      ADL=0       |
|-----|
| 3 |      SC SIGN     |      ADR SIGN   |      ADR=0       |
|-----|
| 4 |      EXECUTE     |      USER IOT  |      -USER       |
|-----|
| 5 |      -BUS IO BUSY |      JFCL SKIP  |      FPD FLAG    |
|-----|
| 6 |      -CONTINUE   |      CRY 01    |      AC # IS ZERO |
|-----|
| 7 |      -1 MSEC      |      TXXX       |      INTERRUPT REQ |
|=====|

```

SKIP/= <63:68>D,70

```

IOLGL=04      ;(.NOT.USER)!(USER IOT)!(CONSOLE MODE)
LLE=12        ;AD LEFT ,LE, 0
CRY0=31       ;AD CRY -2
ADLEQ0=32     ;ADDER LEFT = 0
ADREQ0=33     ;ADDER RIGHT = 0
KERNEL=34     ;.NOT. USER
FPD=35        ;FIRST PART DONE
AC0=36        ;AC NUMBER IS ZERO
INT=37        ;INTERRUPT REQUEST
LE=42         ;(AD SIGN)!(AD,EQ,0)
CRY2=51       ;AD CRY 02
DP0=52        ;AD SIGN
DP18=53       ;AD BIT 18
IOT=54        ;USER IOT
JFCL=55       ;JFCL
CRY1=56       ;AD CRY 1
TXXX=57       ;TEST INSTRUCTION SHOULD SKIP
ADEQ0=62      ;AD,EQ,0
SC=63        ;SC SIGN BIT
EXECUTE=64    ;CONSOLE EXECUTE MODE
-IO BUSY=65   ;.NOT. I/O BUSY
-CONTINUE=66  ;CONTINUE

```

SUMMARY POWER UP MICROCODE SEQUENCE

INITIAL CONDITIONS;

1.1 MSEC CLOCK IS OFF

START MICROCODE AT ZERO(SM=SM0)

!

PUT #/377777 IN MASK REG,SHIFT IT
LEFT ONCE,'OR' A #/1 INTO BIT 35 TO
YIELD 1'S IN 36 BITS, 0 IN -2,-1,36,37

!

SHIFT THE 36 BITS OF ONES RIGHT
ONCE TO YIELD THE MAG REG(ZERO
IN BIT 0,1'S IN THE FOLLOWING 35.

!

LOAD THE XWD1 REG WITH A ONE IN
BOTH HALVES VIA #/1

!

TAKE THE RIGHT HALF OF XWD1(#1)
AND PUT IT INTO THE ONE REG

!

DO A CALL WHICH LOAD S THE TOP OF
THE STACK,WE WILL NEVER RETURN TO
LOCATION ZERO OF THE STACK AGAIN.

!

PUT THE #/376000(ADR OF THE HALT
STATUS BLOCK)IN THE AR,THEN PUT
THIS INTO THE MICROCODE ADDRESS
SPACE SPECIFIED BY #.

!

LOAD THE UBR WITH ZERO AND LOAD THE
VMA WITH ZERO CLEARING THE MEM
CONTROL LOGIC

!

CLEAR EBR AND LOAD THE PREVIOUS AND
CURRENT AC BLOCKS WITH ZERO

!

CLEAR FLAGS REG AND DISABLE ALL APR
CONDITIONS AND ZERO ENABLES

CPU-58

DUMP THE 2901 REGISTERS MICROFLOW

INITIAL CONDITIONS:

1. SM1 CONSOLE COMMAND IS GIVEN
2. CACHE DISABLED, 1 MSEC CLK OFF

FLOW:

----- SAVE ARX -----	WORK[SV,ARX]_[ARX]	SAVE THE ARX IN THE MICROCODE WORKSPACE ADDRESS SPECIFIED BY #/212
! !		
----- SAVE VMA -----	WORK[SV,VMA]_[ARX] [ARX]_WORK[HSBADR]	SAVE THE VMA IN THE UCODE WORKSPACE ADDRESS SPECIFIED BY #/210
! !		
----- GET ADR OF HALT [ARX]_WORK[HSBADR] STATUS BLOCK -----		READ ADR OF HALT STATUS BLK FROM UCODE WORKSPACE #/227
! !----- DUMP REGS -----	ABORT MEM CYCLE, CALL [DUMP]	WE ABORT MEM CYCLE TO CLEAR OUT ANY PAGE FAILS WHICH MAY BE OCCURRING
! !		
----- REWRITE AR, ARX [AR]_WORK[SV,AR] AND VMA VMA_[ARX] ----- [ARX]_WORK[SV,ARX] ! ,NOT,[PI],LOAD PI ! RETURN[6] -----		WE WRITE THE CONTENTS OF THE 2901 REG INTO THE MEMORY LOCATIONS SPECIFIED BY THE HALT STATUS BLK ADDRESS
----- JUMP TO HLTLOOP -----		

MOVE INSTRUCTION MICROFLOW(FROM CONSOLE)

INITIAL CONDITIONS:

```

1.CPU IN HALT LOOP
2.1 MSEC CLOCK OFF
3.CACHE IS OFF
4.CONTINUE SET
5.EXECUTE SET
6.20/DATA
7.EX 200000000020 FROM CONSOLE

```

FLOW:

FROM EXECUTE SET

```

|
|

```

```

-----
SET UP 2901
REGISTER
-----

```

```

[AR]_VMA IO READ
[AR]_[AR].OR.#,
#/200000,HOLD LEFT

```

PUT FLAGS IN THE AR TO FORCE EXEC,ST
AN IO READ CYCLE,PHYSICAL ADDRESS AN
UNCONDITIONAL CYCLE(VIA #/241200)
THEN PUT THE ADDRESS OF THE CSL REGI
SPECIFIED BY MAGIC #/200000
(WHERE THE INSTRUCTION IS)IN THE RIG
HALF OF THE AR

```

-----
LOAD THE VMA
START THE FETCH
-----

```

VMA_[AR]WITH FLAGS

LOAD THE VMA FROM THE DATA PATH AND
START THE FETCH FROM THE CONSOLE REG
(VIA #/36,DP FUNC,LOAD VMA,EXT ADR,
AND UNCONDITIONAL CYCLE.

```

-----
FINISH FETCH
-----

```

```

MEM READ,[HR]_MEM
LOAD INST

```

READ INSTRUCTION FROM 8646'S AND LOA
THE IR AND HR

```

-----
DISPATCH ON
INDEXING OR @
-----

```

EAMODE DISP,J/EACALC

```

-----
NOMOD
-----

```

```

EACALC:[AR]_EA,
PXCT DATA,AREAD

```

HERE DUE TO NO INDEXING OR @
AREAD DISP BASED ON DROM
1.DISP TO U50(40(HARDWIRED ON CRA2)
+DROM A FIELD=10)
2.LOAD VMA WITH 20=EA(VIA DROM VMA E
CRA MEM FUNC,CRM #12)
3.START READ OF ARGUMENT(VIA DROM RE

```

-----
READ ARGUMENT
THEN PREFETCH
-----

```

```

MEM READ,      READ ARGUMENT
[AR]_MEM THEN  FETCHED FROM 8646'S INTO THE A
INST DISP      **NOTE:THE PREFETCH IN THIS CASE RES

```

IN SOME RANDOM VALUE BECAUSE THE PC
AN ARBITRARY VALUE.WE ARE EXECUTING
SINGLE INSTRUCTION.

```

-----
STORE DATA
NICOND DISP
-----

```

AC_[AR],NEXT INST

THE DROM B FIELD DIRECTS US TO STORE
THE CONTENTS OF EA(20) INTO THE AC.
DO THE NEXT INSTRUCTION DISPATCH
THERE ISN'T ONE SO WE HALT

CONSOLE HALT
RUN IS RESET

SAVE THE PC
& HALT CODE

GET HLTSB ADR

WRITE HALT
STATUS BLOCK

RETURN TO HALT LOOP

THE CONSOLE CAUSES THE HALT
(I.E, DROPS RUN)

WE PUT THE CODE WHICH CAUSED THE HALT
(IN THIS CASE 2,SAYING CONSOLE) IN
MEM LOCATION 0 AND PC IN 1.WE THEN
PROCEED TO SAVE THE VMA AND AR IN THE
UCODE WORKSPACE,

WE RETRIEVE THE FIRST HLTSB ADR FROM
THE UCODE WORKSPACE(227) AND DUMP
THE CONTENTS OF THE 2901 REGS IN MEM

**NOTE:THE LOCATIONS IN THE UCODE
WORKSPACE ARE ADDRESSED VIA THE
FILED WHETHER READING OR WRITING

RDIO INSTRUCTION MICROFLOW(USER MODE)

INITIAL CONDITIONS:

1.CONTINUE SET
 2.EXECUTE NOT SET
 3.PREVIOUS INSTRUCTION DIDN'T PREFETCH
 4.RUN IS SET
 5.NO TRAPS OR INTERRUPTS OCCURRING
 6.AC1/20
 6A.100/RDIO 0,00(1)
 6B.20/1776700 THIS IS THE CONTROL REG IN RH11
 7. 101/ANY PDP10 INSTRUCTION
 8.PC REG CONTAINS 100
 9.CACHE DISABLED, 1 MSEC CLK IS OFF
 10/READING THRU UBA#1

START FETCH

START:VMA_[PC],
 J/XCTGO

;THE VMA IS LOADED WITH 100
 AND WE START A READ CYCLE TO
 FETCH THE INSTRUCTION #/1400
 FIELD DEFINES(FETCH,READ,VMA
 EN,UNCONDITIONAL CYCLE)

INSTR HAS BEEN FETCHED

XCTGO:
 MEM READ,
 [HR]_MEM,LOAD
 INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'
 LOAD IR AND HR

INCREMENT PC

INCPC:VMA_[PC]+1,
 EA MODE DISP,TURN
 OFF PXCT,J /EACALC

INCREMENT PC,LOAD VMA
 DO EA MODE DISP

BOTH

[HR]_[HR]+XR,
 LOAD VMA,
 PXCT EA,
 START READ,
 J/FETIND

IT IS BOTH INDEXED AND INDIRECT,INDEX FIRST.
 ADD THE CONTENTS OF THE XR(20) TO EA.RESULT
 (20) IS PUT INTO THE HR AND VMA,BEGIN READ
 CYCLE.(#/040112=>START READ,LOAD VMA AND
 UNCONDITIONAL MEM CYCLE.)

FETCH INDIRECT WORD

FETIND:
 MEM READ,
 [HR]_MEM,
 HOLD LEFT,
 LOAD INST EA

RETRIEVES THE CONTENTS OF 20, PUTS IT IN THE
 HR AND LOADS THE INDEX AND INDIRECT REGISTER
 IT WILL BE CHECKED FOR FURTHER INDEX AND 0.

ANY MORE EA CALCULATION?

VMA_[PC],
 EA MODE DISP,
 J/ EACALC

DISPATCH ON INDEXING OR INDIRECTION
 OF THE FETCHED DATA,THERE WAS NEITHER

NOMOD

[AR]_EA,
PXCT DATA,

AREAD DISPATCH ACCORDING TO DROM(40
(HARDWIRED ON CRA2)DROM AREAD(12)=52)

IO LEGAL?

SKIP IO LEGAL

WE ARE LOOKING FOR EITHER EXECUTE, USER IOT
OR -USER FLAG TO BE SET IN ORDER TO SKIP.
(SEE DPE A) IF NOT IN A LEGAL MODE TO DO
THIS INSTRUCTION UUO OUT, IF LEAGAL
WE DO AN INSTRUCTION DISPATCH WHICH
CAUSES US TO GO TO THE EXECUTE CODE
VIA 1400(HARDWIRED) +DROM J FIELD(60)

YES NO

UUO

INST DISP

RECOMPUTE
EA

CALL [IORD]
TL[HR],#20,
CLR IO BUSY,
CALL [IOEA]

CHECK FOR INDIRECTION AGAIN VIA#20 AND
CLEAR IO BUSY(VIA CRA2 SPEC/CLR BUS BUSY)
WE DON'T SKIP SINCE THERE IS INDIRECTION.

GET EA

IOEA:
[AR]_WORK[YSAVE]WORKSPACE,PUT IT INTO THE AR AND
J/IOEA

WE GET THE SAVED EA FROM THE UCODE
CONTINUE EA CALC.

LOOK FOR
INDEXING

IOEA:
READ[HR],
DBUS/DP,
LOAD INSTR EA
TL[HR],#/17

PUT THE CONTENTS OF THE HR ONTO THE
DBUS AND LOAD THE XR(DPEA) ,CHECKING FOR
INDEXING(USING #17,CHECKS FOR ANY BITS
SET)THERE WAS INDEXING TO AC1 IN THIS CASE.

ADD INDEX
VALUE

[AR]_[AR]+XR,
HOLD LEFT

WE ADD THE CONTENTS OF THE XR(20) TO THE EA(
WHICH IS IN AR,PUT THE RESULT(20) INTO THE A

FETCH DATA
WORD

VMA_[AR],
START READ
MEM READ,
[AR]_MEM,
CLR IO LATCH,
RETURN [1]

LOAD THE VMA(VIA AR),FETCH CONTENTS OF 20
/1776700,READ THRU MB'S INTO AR. CLEAR
THE IO BUSY LATCH(CRA2) VIA
CRA2 SPEC/CLR IO LATCH

CHECK FOR
BYTE
MODE

B DISP
[BR]_VMA IO READ

DISPATCH
BASED ON DROM B(J FIELD(527)+DROM B FIELD
(10)=537,IT IS NOT BYTE MODE BUT WOR
MODE,SO WE PUT THE FLAGS IN THE BR VIA #/241
TO FORCE EXEC,START A READ CYCLE,PHYSICAL RE
AND DO AN IO CYCLE.

LOAD VMA AND

VMA_[AR].OR.[BR] WITH FLAGS

LOAD THE VMA FROM DP(#13)

CPH-64

DO THE
IO READ

WITH THE "OR" OF THE CONTENT
OF THE AR AND BR, (1776700 AND
FLAGS RESPECTIVELY, WE THEN START THE IO FETCH
CYCLE TO READ THE RH11 REG(1776700)

IO BUSY SET?

IOWAIT:

WE WAIT FOR UBA1 TO SET IO BUSY
THE STEP COUNTER IS LOADED VIA #20, IF IO BUSY
IS NOT SET WE SKIP

SC_S#,

YES NO

SKIP/-IOBUSY

[1]

SC_SC-1,
SET YET? SCAD DISP,

THE STEP COUNTER WILL CONTINUE TO BE DECREASED
UNTIL EITHER IO BUSY IS SET OR THE STEP COUNTER
GOES NEGATIVE

3T,

--YES-1 NO SKIP/-IOBUSY

SC NEGATIVE?

YES

NO

[3]

[1]

[3] A HARD PAGE FAIL WILL OCCUR
SEE PAGE FAIL FLOW,.....
THE CPU HAS TIMED OUT.

IO BUSY
RESET?

IOW2:
SC_S#/S#/100,
SKIP/-IOBUSY

RELOAD THE SHIFT COUNTER WITH 100
AND SKIP WHEN IO BUSY IS RESET

YES NO

[4]

RESET YET?

IOW3:
CLR IO LATCH,
STEP SC, J/IOW4:

ATTEMPT TO CLEAR THE LATCHED IO
BUSY, DECREMENT THE STEP COUNTER, TRY
TO CLEAR IT AGAIN AND CHECK IF SET

--YES-1 NO

SC NEGATIVE?

IOW4:
CLR IO LATCH,
SKIP/-IO BUSY
J/IOW3:

ONCE RESET KNOW THAT THE UNIBUS
DEVICE BEING ADDRESSED HAS RESPONDED
TO THE UBA WITH SSYNC AND THE DATA
WHICH RESET IO BUSY, IF BUSY REMAINS

YES

NO

[3]

[4]

AND THE SC GOES NEG PAGE FAIL .

READ WORD
FROM MB'S

MEM READ,
[BR]_MEM,
B DISP
TR[AR], #1,
J/IORD2

CAME HERE BECAUSE IO BUSY RESET AND THE UBA
PUT THE DATA ON THE KS10 BUS WITH IO DATA CYC
READ THE WORD FROM THE 8646'S AND PUT IT INTO
THE BR AND CHECK FOR BYTE MODE, (OURS ISN'T)

PUT INTO AR

[AR],[BR],RETURN 3

STORE DATA

AC_[AR],
J/DONE

THE CONTENTS OF THE RH11 REGISTER (776700)
IS PUT INTO AC(0),THEN JUMP TO DONE.

DONE
NICOND DISP

DONE:
VMA_[PC],
NEXT INSTR FETCH,
FETCH

LOAD THE PREVIOUSLY INCREMENTED PC INTO
THE VMA AND START A FETCH OF THE NEXT INSTR.

!
(2)

PRIORITY INTERRUPT MICROFLOW

COMMENTS: THE FOLLOWING IS THE MICROCODE FLOW FOR HANDLING INTERRUPTS. INTERRUPTS CAN OCCUR WHENEVER THE MACHINE IS IN A MEM CYCLE THAT IS A PHYSICAL REFERENCE OR WRITE CYCLE OR WHEN DOING SPECIAL MEM WAIT A CLEAR FUNCTIONS, THESE ARE "OR"ED TOGETHER TO PRODUCE A SIGNAL CALLED WAIT, MEM WAIT ENABLES PAGE FAILS FROM THE 1MSEC CLOCK, AN APR INTERRUPT OR A CONTROLLER INTERRUPT ON THE KS10 BUS, THE PAGE FAIL STARTS THE INTERRUPT ROUTINE.

INITIAL CONDITIONS:

1. CPU IS DOING A MEM CYCLE
2. THE MEM CYCLE IS NOT A WRITE CYCLE OR PHYSICAL REF (PAGING HARDWARE NOT USED)
3. CACHE AND 1MSEC CLOCK ARE OFF

FLOW:

```

-----
MEM WAIT SET                                DPM5: MEM EN MEM EN TRUE DUE TO UCODE EXECUTION, MEM EN
-----                                GENERATES MEM WAIT.
!
-----
INTERRUPT REQ                                DPM6: MEM WAIT ENABLES INTERRUPT REQUEST TO CAUSE
-----                                A PAGE FAIL AS LONG AS VMA JUST LOADED IS NOT TRUE.
-----
PAGE FAIL                                    CRA3: THE ADDRESS OF THE MICROWORD DOING THE MEM WAIT
-----                                IS SAVED ON THE TOP OF THE STACK.
!
!
-----
SAVE AR, BRX,                                U3777: PAGE FAIL                                BEGIN THE PAGE FAIL, SAVE SOME REGISTER
VMA, ARX                                WORK[SV, AR]_[AR]
-----                                WORK[SV, BRX]_[BRX]
!                                [BRX]_VMA
!                                WORK[SV, VMA]_[BRX]
!                                WORK[SV, ARX]_[ARX]
!
-----
READ PF DISP                                PFD: (PAGE FAIL DISP)                                READ THE PF DISP BITS(10,4,2,1) FROM
BITS(10,4,2,1)                                DBM/PF DISP, DBUS/DBM                                DPM6 INTO THE 2901'S. PF DISP 1 IS TR
-----                                AD/D, DEST/PASS, 3T,                                DPEA; THESE ARE INPUT VIA DP 18-21 ON
!                                DISP/DP LEFT, J/PFD                                DPEA PRINT TO GENERATE DISPATCH BITS
!                                WHICH GO TO CRA1 PRINT TO AFFECT THE
!                                LOWER C-RAM ADDRESS BITS.
-----
DISPATCHED DUE                                WORK[SV, BR]_[BR],                                SAVE BR IN THE UCODE WORKSPACE AND
TO INTERRUPT                                J/PFPI1                                JUMP TO FIND OUT WHAT CAUSED THE PF
-----
!
!
-----
CHECK IF CLOCK                                PFPI1:                                WE CHECK TO SEE WHETHER THE PAGE FAIL WAS CA
OR INTERRUPT                                SKIP IRPT,                                BY THE 1MSEC CLOCK OR AN INTERRUPTING DEVICE
-----                                CALL PISUB,                                IF INTERRUPT REQ IS UP (OUR CASE), IT WILL
!                                J/PFT1                                GENERATE DPEA SKIP 40 L WHICH WILL TURN ON
!                                CRA1 C-RAM ADR BIT 11 TO CAUSE A SKIP.
!

```

CPH-67
CPH-17

DEVICE ON BUS DATA 15-17(14 IS PARITY FOR THIS)
ON DPB,LOAD THE VMA WITH FLAGS VIA DP.(DP WAS
DESIGNATED BY #13,THE #36=>LOAD VMA,EXTENDED
AND UNCONDITIONAL CYCLE)

----- MEM READ,
ANY RESPONSE? [AR]_MEM,3T,
----- SKIP SDR,EQ,0

LOOK FOR SOMEONE TO RESPOND FROM THE PI LEVEL
WE TRANSMITTED(I.E. ANY BITS 18-35 COME BACK
WITH IO DATA CYCLE?),SKIP IF NONE SET.

!YES !NO

NON-VECTORED
INTERRUPT

[AR]_[BR]*2
[AR]_[AR]+*,
#/40,HOLD LEFT
[AR]_[AR]+[EBR]
J/PI40

NO BITS(18-35)SET, ASSUME THE INTERRUPT
WAS FROM APR DEVICE(DPEB),TAKE THE VAL
OF THE INTERRUPT LEVEL,SHIFT LEFT ONCE
ADD #40 AND ADD THE RESULT TO THE
CONTENTS OF THE EBR TO YIELD AN ABSOL
ADDRESS OF THE INTERRUPT INSTRUCTION.
LOAD THE VMA AND FETCH THE INSTRUCTION

LOAD VMA
FETCH INSTR

VMA_[AR],
VMA PHYSICAL,
MEM READ,[AR]_MEM,
LOAD VMA,3T,FORCE EXEC

[1]

VECTORED [ARX]_0,
INTERRUPT J/VECINT

THE UBA WITH THE PI LEVEL THAT WAS TRANSMITTED
FROM THE CPU WILL DRIVE ITS HARDWIRED DATA LINE
AS LONG AS THE INTERRUPT REQ IS UP,THE DATA LINE
IT DRIVES IS SLOT DEPENDENT,UBA#1=19,UBA#2=20
ETC.,THE CPU REALIZES AT THIS POINT THAT IT IS
A VECTORED INTERRUPT AND IT MUST RETRIEVE THE
VECTOR ADDRESS FROM THE INTERRUPTING DEVICE.

----- VECINT:
WHICH CONTROLLER [AR]_[AR]*2,
INTERRUPTED?? 3T,SKIP DP18
----- [ARX]_[ARX]+[XWD1],

LOOKING FOR THE DATA LINE WHICH IS ASSERTED BY
THE UBA,TAKE THE DATA,SHIFT LEFT UNTIL THE BIT
SET IS AT DP18,THEN SKIP,THE UNIT # WILL BE 1
BOTH HALVES OF THE ARX.

SET UP VECTOR [AR]_VMA IO READ,
CYCLE VECTOR CYCLE/1

SET UP FOR VECTOR CYCLE,LOAD THE AR
WITH FLAGS FOR CYCLE SPECIFIED BY
#/241240=>FORCE EXEC,READ CYCLE,PHYSIO
REF,IO CYCLE,VECTOR CYCLE.

----- [AR]_[AR].OR.AR
LOAD VMA AND VMA_[AR] WITH FLAGS
START VECTOR
CYCLE

THE UNIT # OF THE INTERRUPTING DEVICE
FROM ARX TO BITS 14-17 IN THE AR,LOAD
THE VMA VIA DP#13 AND START THE VECTO
CYCLE TO GET THE VECTOR ADDRESS,THIS
IS AN IO READ CYCLE TO THE
APPROPRIATE UBA WITH BIT 5 ON(DURING
COM/ADR CYCLE)SIGNIFYING A VECTOR CYC

THE UBA BEING ADDRESSED SHOULD SET IO BUSY AND HOLD I
SET UNTIL IT HAS THE VECTOR ADDRESS OF THE INTERRUPTI
DEVICE,THE CPU WILL EFFECTIVELY GIVE UP THE KS10 BUS
AFTER THE THIRD TICK,IO BUSY WILL NOT LOCK THE
ARBITRATOR AS MEM BUSY DOES,WHEN THE UBA RECIEVES THE
VECTOR ADR FROM THE UNIBUS WITH SSYNC IT MUST REQUEST
THE KS10 BUS AND TRANSMIT THE ADDRESS WITH IO DATA CY

----- EXECUTE XPCW -----!	[BR]_FLAGS AD/ZERO,LOAD FLAGS J/PIXPCW	PUT FLAGS IN BR,EA OF XPCW POINTS TO A FOUR WORD BLOCK,EA=>EA+3.
! ----- STORE FLAGS & OLD PC -----	PIXPCW:VMA_[AR], START WRITE, MEM WRITE, MEM_[BR]	WRITE FLAGS IN ADDRESS SPECIFIED BY
! ! ! ! !	VMA_[AR]+1,LOAD VMA, START WRITE MEM WRITE, MEM_[PC]	INCREMENTS EA AND WRITE THE OLD PC I EA+1
----- FETCH FLAGS & NEW PC -----	[AR]_[AR]+1,START READ, J/XJRSTF MEM READ,[BR]_MEM [AR]_[AR]+1,LOAD VMA, START READ MEM READ,[PC]_MEM, HOLD LEFT READ[BR],LOAD FLAGS, UPDATE USER	FETCH CONTENTS OF EA +2,WHICH CONTAI NEW FLAGS,PUT THEM IN THE BR. INCREMENT THE AR TO EA+3. LOAD VMA AND DO A FETCH OF THE NEW PC. PUT IT INTO THE PC REG. NOTE:HOLD LEFT BECAUSE ONLY SECTION ZERO USED.
[2]		

[3] ! ----- FETCH INSTR NICOND DISP ----- ! ----- EXECUTE INTERRUPT PROGRAM ----- ! ! ----- MONITOR RETRIEVES OLD PC AND FLAGS ----- ! ! ----- INSTRUCTION FETCH-NICOND DISPATCH ----- ! ! ----- CONTINUE NORMAL OPERATIONS -----	DONE:VMA_[PC],LOAD VMA, FETCH,NEXT INSTR FETCH	FETCH THE FIRST INSTRUCTION OF THE INTERRUPT HANDLER
---	---	---

 INITIAL CONDITIONS;

1. CONTINUE SET
 2. EXECUTE NOT SET
 3. PREVIOUS INSTRUCTION DIDN'T PREFETCH
 4. RUN IS SET
 5. NO TRAPS OR INTERRUPTS OCCURRING
 6. 100/ADD 0,20
 7. 101/ANY PDP10 INSTRUCTION
 8. PC REG CONTAINS 100
 9. CACHE DISABLED, 1 MSEC CLK IS OFF
 10. 20/DATA

 START FETCH

START:VMA_[PC],
 J/XCTGO

;THE VMA IS LOADED WITH 100
 AND WE START A READ CYCLE TO
 FETCH THE INSTRUCTION
 #FIELD DEFINES(FETCH,READ,VMA
 EN,UNCONDITIONAL CYCLE)

2-----

 INSTR HAS BEEN FETCHED

XCTGO:
 MEM READ,
 [HR]_MEM,LOAD
 INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'S
 LOAD IR AND HR

 INCREMENT PC

INPC:VMA_[PC]+1,
 EA MOSE DISP,TURN
 OFF PXCT,J EACALC

;THE ADD INSTRUCTION
 WILL PREFETCH BUT
 IT MUST FETCH ARGUMENTS
 FIRST.INCREMENT THE
 PC AND LOAD THE VMA

 NOMOD

EACALC:[AR]_EA,PXCT
 DATA,AREAD

EA PUT IN AR
 INSTR,AREAD DISP
 BASED ON DROM
 1. DISPATCH TO U50 (40
 (HARDWIRED ON CRA2)+DROM
 A FIELD=10)
 2.EA=20 LOADED INTO VMA VIA
 (DROM VMA EN,CRA MEM FUNC,CRA
 3.START READ OF ARGUMENT(VIA
 DROM READ

 READ ARGUMENT THEN
 PREFETCH

MEM READ,[AR]_MEM
 THEN FETCH,INST DISP

READ CONTENTS OF 20 FROM 864
 LOAD VMA FROM PC(101). START
 PREFETCH VIA #140012. DISPATCH
 DISP TO EXECUTE CODE VIA
 DROM J FIELD

 EXECUTE THE ADD 0,20

[AR]_[AR]+AC,

AR CONTAINED CONTENTS OF 20

----- AD FLAGS,3T,

ADD CONTENTS OF AR AND AC

STORE THE DATA

STAC,AC_[AR],

STORE DATA INTO AC(0)

DO NICOND DISPATCH

NEXT INSTR

.DESTINATION
SELECTED BY THE DROM B
FIELD(15)

1

THE NICOND DISPATCH HARDWARE
(CRA2) WILL SHOW THIS
MEMORY CYCLE IN PROGRESS. THE
INSTR DID A PREFETCH, DISPATCH
TO XCTGO

1

V

TO READ IT FROM THE 8646'S

2

SOJE INSTRUCTION MICROFLOW(USER MODE)

INITIAL CONDITIONS:

1. CONTINUE SET
2. EXECUTE NOT SET
3. PREVIOUS INSTRUCTION DIDN'T PREFETCH
4. RUN IS SET
5. NO TRAPS OR INTERRUPTS OCCURRING
6. 100/SOJE 0,20
7. 20/ANY PDP10 INSTRUCTION
8. PC REG CONTAINS 100
9. CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
J/XCTGO

;THE VMA IS LOADED WITH 100
AND WE START A READ CYCLE TO
FETCH THE INSTRUCTION
#FIELD DEFINES(FETCH,READ,VMA
EN,UNCONDITIONAL CYCLE)

2-----

INSTR HAS BEEN FETCHED

XCTGO:
MEM READ,
[HR]_MEM,LOAD
INST,3T,J/INCP

;READ INSTRUCTION FROM 8646'S
LOAD IR AND HR

INCREMENT PC

INPC:VMA_[PC]+1,
EA MODE DISP,TURN
OFF PXCT,J EACALC

DO A EA MODE DISP
INC THE PC,
LOAD THE VMA

NOMOD

EACALC:[AR]_EA,
PXCT DATA,AREAD

AR GETS EA=20,AREAD DISP BASED ON
DROM. 1.DIRECT TO EXECUTE CODE VIA
DROM J FIELD DUE TO CONTROL SIGNAL
DROMA=J BEING TRUE
2.LOAD VMA WITH EA=20(VIA DROM VMA
EN,CRA MEM FUNC,CRM #12)

EXECUTE
INSTRUCTION

[BR]_AC=1,AD FLAGS,
3T,JUMP DISP

SUBTRACTS ONE FROM THE CONTENTS
OF THE AC AND DISPATCHES ACCORDING
TO THE DROM B FIELD=2(J FIELD+
DROM B FIELD(1500+2=1502))

[AC]=0??

AC_[BR],TEST,
SKIP AD.EQ,0,J/JUMPA

TEST IF RESULT=0

!NO

!YES

FETCH NEXT INSTR
NICOND DISPATCH

JUMPA:PC_[AR],
HOLD LEFT
LOAD VMA,FETCH,
NEXT INSTR FETCH

THE EA(20)WAS IN THE
AR,WE LOAD THE VMA
AND FETCH THE CONTENT
OF EAVIA #/140012,

V

CPH-74

!
!
!

2

FETCH NEXT INSTR
NICOND DISPATCH

DONE:VMA_[PC],LOAD VMA
FETCH,NEXT INST FETCH

THE VMA IS LOADED AND
WE START A READ CYCLE
TO FETCH THE CONTENTS
OF PC 101 VIA #140012

!
!
V
2

XCTGO:

GET THE NEXT INSTR FROM THE 8646'S

CPH-75

TLNE INSTRUCTION MICROFLOW(USER MODE)

INITIAL CONDITIONS:

- 1,CONTINUE SET
- 2,EXECUTE NOT SET
- 3,PREVIOUS INSTRUCTION DIDN'T PREFETCH
- 4,RUN IS SET
- 5,NO TRAPS OR INTERRUPTS OCCURRING
- 6,100/TLNE 0,MASK
- 7, 101/ANY PDP10 INSTRUCTION
- 7A,102/ANY PDP10 INSTRUCTION
- 8,PC REG CONTAINS 100
- 9,CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
J/XCTGO

;THE VMA IS LOADED WITH 100
AND WE START A READ CYCLE TO
FETCH THE INSTRUCTION
#FIELD DEFINES(FETCH,READ,VM
EN,UNCONDITIONAL CYCLE)

2-----

INSTR HAS BEEN FETCHED

XCTGO:
MEM READ,
[HR]_MEM,LOAD
INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'
LOAD IR AND HR

INCREMENT PC

INPC:VMA_[PC]+1,
EA MODE DISP,TURN
OFF PXCT,J EACALC

INCREMENT PC,PUT INT
VMA &DO EA MODE DISP

NOMOD

EACALC:AR_[EA],
PXCT,AREAD

AR GETS THE MASK,AREAD DISP BASED ON
DROM
1.GOES DIRECT TO EXECUTE CODE(VIA
DROM J FIELD AND DROM A=J)

EXECUTE
INSTRUCTION

[AR]_[AR]SWAP
[BR]_[AR],AND.AC
TEST DISP

THE MASK IS PUT IN THE LEFT HALF OF
AR.IT IS ANDED WITH THE AC,THEN WE
DISPATCH ACCORDING TO THE 'OR' OF
(J FIELD AND DROM B FIELD)

[BR]=0??

READ [BR],TXXX TEST,
3T,J/DONE

WE ARE LOOKING FOR THE RESULT IN BR
TO BE ZERO IN ORDER TO SKIP.

!NO !YES

V
1

FETCH NEXT
INSTRUCTION

VMA_[PC]+1,FETCH
NEXT INST FETCH

WE CAME HERE ON THE SKIP.WE INC THE
FOR THE SECOND TIME(THE FIRST WAS A

ICOND DISP

!-----

1
!
!
!
V

!
V
2

FETCH NEXT DONE;VMA_[PC],
INSRTUCTION LOAD VMA,FETCH,
NICOND DISPATCH NEXT INSTR FETCH

!
!
!
V
2

INCPC)WHICH ESSENTIALLY DOES THE SK
WE LOAD THE VMA WITH 102 AND
FETCH ITS CONTENTS

THE MASKED RESULT WAS NOT EQUAL TO
SO WE LOAD VMA WITH PC=101 AND FETC
CONTENTS

MOVE INSTRUCTION MICROFLOW(USER MODE)

INITIAL CONDITIONS:

1. CONTINUE SET
2. EXECUTE NOT SET
3. PREVIOUS INSTRUCTION DIDN'T PREFETCH
4. RUN IS SET
5. NO TRAPS OR INTERRUPTS OCCURRING
6. 100/MOVE 0,20
7. 20/ANY PDP10 INSTRUCTION
8. PC REG CONTAINS 100
9. CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
J/XCTGO

;THE VMA IS LOADED WITH 100
AND WE START A READ CYCLE TO
FETCH THE INSTRUCTION
#FIELD DEFINES(FETCH,READ,VM)
EN,UNCONDITIONAL CYCLE)

2

INSTR HAS BEEN FETCHED

XCTGO:
MEM READ,
[HR]_MEM,LOAD
INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'S
LOAD IR AND HR

INCREMENT PC

INPC:VMA_[PC]+1,
EA MODE DISP,TURN
OFF PXCT,J EACALC

INC PC,LOAD VMA
DO EA MODE DISP

NOMOD

EACALC:[AR]_EA,
PXCT DATA,AREAD

AREAD DISPATCH BASED
ON DROM

1. DISPATCH TO U50(40(HARDWIR
ON CRA2)+DROM A FIELD(10))
2. LOAD VMA WITH 20(=EA)
3. START READ OF ARGUMENT

READ ARGUMENT THEN
PREFETCH

MEM READ,
[AR]_MEM THEN
FETCH,INST DISP

READ THE CONTENTS OF 20 FROM
8646'S INTO AR,LOAD VMA FROM
START THE PREFETCH

STORE THE DATA
NICOND DISPATCH

AC_[AR],NEXT INST

THE DROM B FIELD TELLS US TO
STORE THE CONTENTS OF 20
INTO AC.

2

NICOND DISP SHOWS MEM CYCLE IN PROGRESS
BECAUSE OF THE PREFETCH

MOVEM INSTRUCTION MICRO FLOW(USER MODE)

INITIAL CONDITIONS:

- 1,CONTINUE SET
- 2,EXECUTE NOT SET
- 3,PREVIOUS INSTRUCTION DIDN'T PREFETCH
- 4,RUN IS SET
- 5,NO TRAPS OR INTERRUPTS OCCURRING
- 6,100/MOVEM 0,20
- 7,101/ANY PDP10 INSTRUCTION
- 8,PC REG CONTAINS 100
- 9,CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
J/XCTGO

;THE VMA IS LOADED WITH 100
AND WE START A READ CYCLE TO
FETCH THE INSTRUCTION
#FIELD DEFINES(FETCH,READ,VMA
EN,UNCONDITIONAL CYCLE)

2-----

INSTR HAS BEEN FETCHED

XCTGO:
MEM READ,
[HR]_MEM,LOAD
INST,3T,J/INCP

;READ INSTRUCTION FROM 8646'S
LOAD IR AND HR

INCREMENT PC

INPC:VMA_[PC]+1,

THE MOVEM WILL NOT DO
A PRE-FETCH BECAUSE
IT MUST WRITE MEMORY.
DO THE EA
MODE DISP,PC+1=101
LOAD VMA

EA MODE DISP,TURN
OFF PXCT,J/ EACALC

NOMOD

EACALL:[AR]_EA,PXCT DATA,
AREAD

AREAD DISPATCH BASED ON DROM
1,DISPATCH TO U41
(U41=40(HARDWIRED ON CRA2)
+DROM A FIELD(1))
2,START A WRITE VIA DROM WRIT
3,LOAD VMA WITH EA(20)VIA
(DROM VMA EN,CRA MEM FUNC,CRM
#12)

FETCHES ARGUMENT

[AR]_AC,INST DISP

THE CONTENTS OF THE AC ARE
PUT IN THE AR
DISP ACCORDING TO DROM B

STORE DATA

MEM WRITE,MEM_[AR],
J/DONE

DO A DATA CYCLE SENDING DATA
FROM AR TO MEM,THIS COMPLETES
THE MEMORY WRITE CYCLE

FETCH NEXT INST
NCOND DISPATCH

DONE:VMA_[PC],LOAD VMA,
FETCH,NEXT INST FETCH

HERE WE LOAD VMA FROM THE
PREVIOUSLY INCREMENTED PC
AND FETCH THE CONTENTS OF 101

!
!
V
2

XCTGO:

CPV-8φ

JRST INSTRUCTION MICROFLOW(USER MODE)

THE JRST CASE IS UNIQUE IN THAT HARDWARE DETECTS IT AND NO EXECUTE UCODE IS REQUIRED TO GET THE JOB DUN

INITIAL CONDITIONS:

- 1,CONTINUE SET
- 2,EXECUTE NOT SET
- 3,PREVIOUS INSTRUCTION DIDN'T PREFETCH
- 4,RUN IS SET
- 5,NO TRAPS OR INTERRUPTS OCCURRING
- 6,100/JRST 0,200
- 7,200/ANY PDP10 INSTRUCTION
- 8,PC REG CONTAINS 100
- 9,CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
J/XCTGO

;THE VMA IS LOADED WITH 100
AND WE START A READ CYCLE TO
FETCH THE INSTRUCTION
#FIELD DEFINES(FETCH,READ,VMA
EN,UNCONDITIONAL CYCLE)

2-----

INSTR HAS BEEN FETCHED

XCTGO:
MEM READ,
[HR]_MEM,LOAD
INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'S
LOAD IR AND HR

INCREMENT PC

INCPC:VMA_[PC]+1,
EA MODE DISP,TURN
OFF PXCT,J EACALC

DO EA MODE DISP
INC PC,LOAD VMA

JRST CASE
NOMOD
NICOND DISPATCH

[PC]_[HR],HOLD LEFT,
LOAD VMA,FETCH,
NEXT INST FETCH

GET NEW PC(200)
FROM RIGHT HR,FROM
FIELD WE LOAD VMA
AND BEGIN A FETCH OF
CONTENTS OF 200

V
2

IDLB INSTRUCTION MICROFLOW (USER MODE)

INITIAL CONDITIONS:

1,CONTINUE SET
 2,EXECUTE NOT SET
 3,PREVIOUS INSTRUCTION DIDN'T PREFETCH
 4,RUN IS SET
 5,NO TRAPS OR INTERRUPTS OCCURRING
 6,40/010700000020
 6A,100/ILDB 0,40
 6B,20/DATA
 6C,21/377000000000
 7, 101/ANY PDP10 INSTRUCTION
 8,PC REG CONTAINS 100
 9,CACHE DISABLED, 1 MSEC CLK IS OFF

START FETCH

START:VMA_[PC],
 J/XCTGO

;THE VMA IS LOADED WITH 100
 AND WE START A READ CYCLE TO
 FETCH THE INSTRUCTION
 #FIELD DEFINES(FETCH,READ,VM
 EN,UNCONDITIONAL CYCLE)

2-----

INSTR HAS BEEN FETCHED

XCTGO:
 MEM READ,
 [HR]_MEM,LOAD
 INST,3T,J/INCPC

;READ INSTRUCTION FROM 8646'
 LOAD IR AND HR

INCREMENT PC IN CASE

INCPC:VMA_[PC]+1, EA MODE DISP
 EA MOSE DISP,TURN INC PC & LOAD VMA
 OFF PXCT,J EACALC
 PUT THE EA(40) INTO THE AR.DO A DROM AREAD
 DISPATCH, 1.DISPATCH TO 40(HARDWIRED BY
 VIRTUE OF AREAD DISP(CRA2)+DROM A FIELD=0
 2,LOAD VMA WITH EA(40) DUE TO DROM VMA EN,
 CRA MEM FUNC,CRM #12)
 3,START A WRITE TEST VIA DROM WRITE TEST
 4,START A READ OF 40

NOMOD

EACALC:
 [AR]_EA,
 PXCT DATA,
 AREAD

READ DATA

MEM READ,
 [AR]_MEM,
 INST DISP

FETCHING THE CONTENTS OF 40 YIELDS THE
 BYTE SIZE(7),BYTE POSITION(1'ST)
 AND BYTE MEM LOCATION(21)REMEMBER
 THAT THE INCREMENTING PORTION WILL OCCUR
 FIRST,THE CONTENTS OF 40 GO THRU THE 8646'S
 WE DO INSTRUCTION DISPATCH
 ACCORDING TO DROM J FIELD TO GET TO EXECUTE

BEGIN
 EXECUTING
 INSTRUCTION

CALL IBP

THIS IS A MACRO WHICH CAUSES US TO INCREMENT
 THE BYTE POINTER,WE SUBTRACT THE SIZE FROM
 THE POSITION FIELD AND LOOK FOR THE OVERFLOW
 (SCAD 00) AND FIRST PART DONE FLAG,WE WILL
 DO A Y WAY DISPATCH ON THESE TWO SIGNALS,OUR

CPV-82

CASE HAS OVERFLOW BUT NO FPD FLAG.

OVERFLOW
(SCAD 00)
SET

SET P TO 36-S,
J/NXTWRD

THE POSITION FIELD WAS POINTING TO THE LAST
BYTE IN LOCATION 20, SUBTRACTING SIZE FROM
POSITION CAUSED OVERFLOW AS IT SHOULD IN
THIS CASE, WE WILL BUMP THE ADDRESS ONE AND
SUBTRACT THE SIZE(7) FROM 44(OCTAL) TO POINT
AT THE THE FIRST BYTE IN LOCATION 21.

INCREMENT
ADDRESS OF
POINTER

NXTWRD;
[AR]_[AR]+1,
HOLD LEFT,
START WRITE

MEM WRITE,
MEM_[AR], RETURN [4]

THIS IS WHERE WE BUMP THE DATA ADDRESS, WE
START A WRITE(VIA #4,5,16) TO THE RIGHT HALF
OF THE EA(40).
NOTE: THE VMA WAS PREVIOUSLY
LOADED WITH 40 WHEN WE FETCHED THE POINTER
INFORMATION.

EA CALCULATION

LDB: READ[AR],
LOAD BYTE EA,
FE_P, 3T,
CALL [BYTEA]

READ THE INCREMENTED POINTER INFORMATION
FROM THE AR AND LOAD THE XR TO CHECK FOR
INDEXING AND INDIRECTION.
THE FE REG IS LOADED WITH THE INCREMENTED
POSITION FIELD.(35)

INDEXING?

BYTEA:

SET FPD,
EA MODE DISP

THE FPD PC FLAG IS SET VIA #4(SEE PC FLAGS
ON DPE) AND WE ADDRESS
THE RAMFILE WITH XR, IN THIS CASE THERE IS NO
INDEXING(XR=0), WE DISP ACCORDINGLY(I.E. CRAM
ADR 10 IS SET)

LOAD VMA

VMA_[AR],
START READ,
PXCT BYTE DATA,
J/BYTFET

THE VMA IS LOADED FROM THE AR(21), WHICH WE
PREVIOUSLY INCREMENTED AND LOADED, WE THEN ST.
A READ OF 21 VIA #3,14,16(READ, LOAD VMA AND
UNCONDITIONAL MEM CYCLE RESP)

COMPLETE THE
DATA FETCH

BYTFET;
MEM READ,
[BR]_MEM, AND, MASK,
RETURN[1]

PUT THE WORD IN THE BR AND MASK THE SIGN BIT
OUT.

START NEXT
INSTR FETCH

VMA_[PC],
FETCH

LOAD THE VMA FROM PC(101) AND START A FETCH
OF THE NEXT INSTRUCTION

BYTE DISP

FE_FE, AND, S#, S#/0770,
READ [AR],
BYTE DISP,
CALL [LDB1]

WE DISP ACCORDING TO WHICH BYTE WE WANT (I I
THIS CASE) AND GO LOAD THE BYTE INTO AC.
THE AR CONTAINS THE POINTER, BR HAS DATA, FE H
THE POSITION OF THE BYTE, WE MASK OUT THE
JUNK IN THE FE.

CPU-83

LOAD BYTE

LDB1:
7-BIT LDB,
SCADA/BYTE 1,
J/LDB7

THE "7-BIT LDB" MACRO CAUSES US TO TAKE THE DATA FROM BR(SCADA/BYTE 1 SELECTS THE CORRECT BYTE) PUT IT ON THE SCADA LINES AND THEN BECAUSE OF DPM1 HARDWARE, WE LOAD THIS THRU DPM MIXER B1 28-34(35 IS JUNK), THRU THE D BUS MIXER INTO THE BR.

LDB7:AD/ZERO,
RSRC/DA,DBUS/DBM
DBM/*,*/376,A/BR,
B/BR,DEST/AD*,5,3T,
RETURN[2]

MASK OUT THE SELECTED BYTE AND SHIFT IT ONE PLACE TO THE RIGHT,

STORE DATA
NICOND DISP

AC_[AR],
CLR FPD,
NEXT INSTR

PUT THE BYTE INTO AC0, CLEAR THE FPD FLAG AND TO READ THE PREFETCHED INSTRUCTION FROM THE THE 8646'S

[2] XCTGO;

CONSOLE

The console (CSL) board, contains the main clock source, and arbitrates access to the backpanel bus, as well as the 8080 based console hardware. The operator controls the machine via console functions typed in via the console terminal to the 8080; he may load and check microcode; read and write memory; stop and start the clock; single step the clock; halt the machine; start at a given location; execute an instruction, etc. He does not have quite the power of the KL10 console, in that he does not have all the diagnostic features; he cannot, for example, read the PI status at any instant, nor can he at any given instant, read the VMA or PC registers or read the AC's. (Many registers, in fact, are internal to the 2901, and are not available even were the board on an extender module and an oscilloscope available.) [However, when entering the halt loop, either upon executing a halt instruction or when requested to stop by the 8080, the microcode deposits the PC and current block of AC's in main memory, to make these available to the console, as well as the reason for halting, before interrupting the 8080. Thus the operator does have this information available via the microcode, if not by diagnostic hardware.

The console has the same access to the backpanel as the CPU, and can therefore read or write any memory locations, and initiate any I/O that the CPU can (except I/O to the internal CPU registers PI and APR). These it may also do, if the microcode is loaded and the CPU is functional, by executing the required I/O instruction.

The console may also load microcode (2K x 96 bits), read it back to check it, and has access to the same diagnostic points as are available to the KL10 console. Thus the operator can read the current microcode location, next location, current top of stack, etc.

At power-up, the console may load microcode automatically from the disk, or may be programmed to wait for console commands to do so, or may load microcode from the console terminal (which is a direct line to system 1065 on the prototype). It is intended that any unused microcode space will be used to run module once-only diagnostics at power-up time.

CONSOLE

THE CONSOLE MODULE INTERFACES A SERIAL LINE, CTY OR KLINIK, TO THE KS10 BUS. THERE ARE TWO MODES OF OPERATION, USER AND CONSOLE. THE CONSOLE WILL ONLY RECOGNIZE SPECIFIC THE CHARACTERS LISTED BELOW WHEN IN CONSOLE MODE. WHEN IN USER MODE THE CONSOLE WILL ONLY RECOGNIZE ^/ . ALL OTHER CHARACTERS ARE PASSED TO THE MONITOR.

SK XX START 8080 AT XX

BC BOOT AND CHECK. CHECK MEMORY AND VERIFY CRA/M DATA LOADED

EX XX EXECUTE INSTRUCTION XX

MK XX MARK MICRO CODE WORD XX IS ADDRESS SPECIFIED

UM XX UNMARK MICRO CODE

TR TRACE UNTIL ANOTHER CHAR IS RECIEVED

PD X 0 DISABLE PARITY DETECTION
 3 ENABLE ALL PARITY DETECTION
 2 DISABLE DP PARITY DETECTION

COMMANDS CURRENTLY IMPLEMENTED

```

;***** CSL V0.104 *****
^Z        ;ENTER USER MODE
^/        ;ENTER CONSOLE MODE
LA XX     ;SET MEMORY ADDRESS
LI XX     ;SET I/O ADDRESS
LK XX     ;SET 8080 ADDRESS
LC XX     ;SET C-RAM ADDRESS TO BE WRITTEN AND/OR READ
EM XX     ;EXAMINE MEMORY LOCATION XX
EM        ;EXAMINE MEMORY
EN        ;EXAMINE NEXT, LAST FUNCTION(EK,EM,EI)
EB        ;EXAMINE BUS & 8080 CNTRL REGS
EI XX     ;EXAMINE I/O REGISTER XX
EI        ;EXAMINE I/O
EK XX     ;EXAMINE 8080 LOCATION XX
EK        ;EXAMINE 8080 LOC
DM XX     ;DEPOSIT MEMORY WITH XX DATA
DN XX     ;DEPOSIT NEXT LOCATION WITH XX, LAST FUNCTION (DK,DM,DI)
DB XX     ;DEPOSIT XX DATA ONTO THE KS10 BUS
DI XX     ;DEPOSIT I/O REGISTER WITH XX DATA
DK XX     ;DEPOSIT 8080 LOC(ONLY RAM LOCATIONS STICK)
MR        ;MASTER RESET
CS        ;CPU CLK START
CH        ;CPU CLK HALT
CP XX     ;CPU CLK PULSE. XX IS HOW MANY CLOCK PULSES. NO ARG GIVES 1 CP
SI        ;SINGLE INSTRUCTION
LF        ;SET DIAG FUNCTION TO WRITE
DF XX     ;WRITE DATA XX WITH DIAG FUNC FROM "LF" CMD
EC XX     ;EXAMINE C-RAM AT ADDR XX
EC        ;EXAMINE C-RAM.,CURRENT CNTRL REG, NO CLKS.,CURRENT LOC AS ADDR

```

DC XX ;DEPOSIT C-RAM WITH XX (IN 32 OCTAL CHARS
 EX XX ;EXECUTE INSTRUCTION XX
 ST XX ;START KS10 AT ADDRESS XX
 SM XX ;START MICRO-CODE AT XX
 HA ;HALT THE PROCESSOR(EXECUTE HALT INSTR)
 CO ;CONTINUE THE PROCESSOR
 PE X ;PARITY ENABLE 00=DISABLE ALL,,7=ENABLE ALL,1=EN "DP" PAR,2=EN "CRM"
 CE X ;CACHE ENABLE, 0= DISABLE CACHE,1=ENABLE CACHE,<CR> CURRENT STATE?
 TE X ;1 MSEC ENABLE, 0= DISABLE 1 MSEC,1=ENABLE 1 MSEC,<CR> CURRENT STATE?
 LT ;LAMP TEST, LIGHT THE THREE LAMPS ON THE FRONT PANEL
 RC ;READ C-RAM DIRECT,FUNCTIONS 0-17(NO RESETS,NO LOAD DIAG ADDR,NO CPU C
 EJ ;EXAMINE JUMPS , PRINT C-RAM ADDR SIGNALS(CUR,NXT,J,SUB)
 TR XX ;TRACE = REPEATEDLY ISSUE "CP"& "EJ" CMDS TILL ANY CHAR TYPED
 ;XX(IF TYPED) WILL BE A C-RAM ADDRESS AND THE TRACE COMMAND WILL
 ;TRACE UNTIL IT GETS TO ADDRESS XX, WHEREUPON IT WILL STOP
 PM ;PULSE MICRO-CODE = ISSUE SINGLE "CP",EJ"
 ZM ;ZERO MEMORY = ZERO KS10 MOS MEMORY
 RP ;REPEAT, REPEATS LAST LEGITIMATE COMMAND UNTILL ANY CHAR IS TYPED.
 ;CAN ALSO BE LAST COMMAND ON A LINE, AND WILL REPEAT THAT LINE, IE.
 ;EM 0,EK 0,EC 0,RP WILL REPEATEDLY PRINT AN EM0,EK0 AND EC0.,
 ;TO STOP OUTPUT ON THE TTY TYPE "CONTROL=0",.TO RESUME OUTPUT TO
 ;THE TTY TYPE "CONTROL=0" AGAIN
 ;---NOT IMPLIMENTEDSD ;SELECT DEVICE, COMMAND TO SELECT OR CHANGE DEFAULT ;
 BT ;BOOT SYSTEM,,MICRO-CODE IS READ FROM DISK AND LOADED INTO THE
 ;CONTROL STORE,THEN THE MONITOR BOOT IN BLOCK/TRACK/SECTOR 0
 ;IS LOADED INTO MEMORY AT LOCATION 600 AND STARTED
 LB ;LOAD BOOTSTRAP,, LOADS ONLY THE BOOTSTRAP 10 CODE FROM BLOCK 0
 ;TRACK AND SECTOR 0 OF THE SELECTED DISK
 *U ;RUB-OUT CURRENT LINE
 *O ;STOP TYPE-OUT TO THE TTY
 *S ;STOPS TTY OUTPUT & HANGS 0080 WAITING FOR CNTRL-Q
 *Q ;RESUMES TTY OUTPUT
 RUB-OUT ;RUB OUT THE PREVIOUS CHARACTER TYPED

;NOTE::

SEVERAL COMMANDS MAY BE PUT ON A SINGLE LINE, WITH COMMANDS
 SEPARATED BY COMMAS

KS10 BUS ARBITRATOR

CONTROL OF THE KS10 BUS MASTERSHIP OCCURS ON PRINT CSL1. A BUS MASTER ALWAYS HAS CONTROL OF THE BUS FOR AT LEAST TWO TCLK CYCLES. THE ARBITRATOR WILL GRANT THE BUS TO THE REQUESTING DEVICE OF THE HIGHEST PRIORITY UNLESS
1/MEMORY BUSY IS ASSERTED, IN WHICH CASE THE ARBITRATOR IS DISABLED UNTILL THE END OF THE CYCLE OR
2/THE CURRENT CYCLE IS A COMMAND ADDRESS CYCLE
3/THE CYCLE IMMEDIATELY FOLLOWING A COMMAND ADDRESS CYCLE.

CONSOLE DATA REG

A CPU COMMAND ADDRESS IO READ 200000 WILL READ THE CONTENTS OF CSL IO REGISTERS 102,104,106,110 AND 112. THESE REGISTERS (8 BITS EACH) CONTAIN THE LAST INSTRUCTION EXECUTED FROM THE CONSOLE OR DATA FOR A DATA CYCLE. THIS FEATURE IS USED WHEN STARTING UP THE OPERATING SYSTEM. THE 8080 CODE WILL WRITE A JRST ADR IN THE DATA REGISTER AND ISSUE EXECUTE AND CONTINUE. THE IO EXECUTES THE JRST TO THE START OF THE BOOT PROGRAM AND CONTINUES TO COME UP.

THE DECODE OF THE COMMAND ADDRESS DATA LINES OCCURS ON CSL4 AND SETS CSL4 IO READ WHICH GENERATES CSL4 DATA CYCLE. THIS SIGNAL OPENS THE READ GATE ON THE 4X4 REGISTER MEMORIES WHICH PRECEDE THE 8646 TRANSCEIVERS. AT THE NEXT T CLK THE CONTENTS OF 102,104,106,110,112 WILL APPEAR ON THE BUS WITH IO DATA CYCLE.

INTERUPTS

KS10 CAN INTERRUPT THE CONSOLE BY ASSERTING DPMB CSL INTERRUPT L
PIN=F18L1

THE CONSOLE CAN INTERRUPT THE KS10 BY ASSERTING CSL4 INT 10 L
PIN=F18T2

8080 CONSOLE ERROR CODES

?ILL ILLEGAL COMMAND
?NI NOT IMPLIMENTED COMMAND
?UI UNKNOWN INTERRUPT (CONSOLE GOT INT BUT NO CHARACTER)
?BFO BUFFER OVERFLOW CONSOLES INPUT BUFFER FILLED UP
?D CYC DATA CYCLE FAILURE
?C CYC COMMAND /ADDRESS CYCLE FAILURE
?BN BAD NUMBER WAS TYPED
?NBR NO BUS RESPONSE (GOT NO GRANT FOR MY REQUEST)
?NDA NO DATA ACKNOWLEDGE (WAS RECVD BY CONSOLE AFTER A REQ)
?IA ILLEGAL ADDRESS-OUT OF RANGE
?RA REQUIRES ARGUMENT
?DNF DID NOT FINISH (IF EXECUTE DOES NOT CLR CONTINUE)
?DNC DID NOT COMPLETE (NO HALT LOOP AFTER SM OR HA)
?NAR NO ARGUMENT REQUIRED

DEPOSIT THE BUS FROM THE CONSOLE

 COMMAND: DB XX (36 BIT DATA ARGUMENT) DEPOSIT DATA PATTERN TO KS10 BUS
 DESC: THIS COMMAND PERFORMS AN INTERNAL LOOPBACK FUNCTION TO CHECK
 THE ABILITY OF THE CONSOLE BOARD TO WRITE DATA THRU THE KS10
 BUS TRANSCEIVERS AND READ IT BACK WITHOUT ERRORS. THE COMMAND
 IS SPLIT INTO TWO PARTS WHICH WRITE AND READ THE ODD AND EVEN
 IO REGISTERS LOCATED ON CSL 6,7,8. THE FIRST PART WRITES THE ODD
 REGISTERS NORMALLY USED FOR COMMAND ADDRESS CYCLES, READ THE DATA
 BACK AND COMPARES IT TO THE DATA SENT. THE SECOND PART WRITES THE EVE
 REGISTERS NORMALLY USED FOR DATA ARGUMENTS (AND FOR A JRST INSTRUCTIO
 TO START KS10), DOES THE READ AND COMPARES. THE READ OF BUS DATA
 OCCURS THRU THE D BUS MIXERS ON CSLA PRINT. THESE RETURN INVERTED
 DATA TO THE DBUS. A FAILURE IN THE FIRST PART WILL YIELD A ?C CYC
 ERROR CODE, A FAILURE IN THE SECOND PART WILL YIELD A ?D CYC
 ERROR CODE.

FLOW:

 WRITE THE ODD IO REGS

;IO WRT 103,105,107,111,113

!
 !

 SET CONSOLE REQ & T ENB

;IO WRT 210/141

!
 !

 MAKE SURE BUS REQ CAME UP

;IO READ 301 (LOOK FOR 20)
 IF IT DIDN'T PRINT NBR ERR MSG
 BUS ARBITRATOR ON CSL1 IS BROKE

!
 !

 READ THE DATA AND COMPARE IT

;IO READ 0,1,2,3,103-ITS WRONG FLAVOR
 SO 8080 CODE MUST INVERT---IF THE COMPARE
 FAILS PRINT ?C CYC ERROR CODE--CHECK THE
 DATA PATH, CSL OR KS10 BUS IS BROKE

!
 !
 !

 WRITE THE EVEN IO REGS

;IO WRT 102,104,106,110,112

!
 !

 SET DATA CYCLE BIT

;IO WRT 114/1

!
 !

 SET CONSOLE REQ, T ENABLES
 AND LATCH DATA SENT

;IO WRT 210/163--DATA GOES OUT

!
 !

 CHECK FOR BUS REQ AS ABOVE

CSL-6

!

CHECK FOR DATA ACKN

;SHOULD BE THERE AS A RESULT OF DATA CYCLE
ON CSL4 PRINT

!
!

READ AND COMPARE THE DATA
AS ABOVE

;SAME AS ABOVE BUT IF DIE HERE PRINT ?D CYC
ERROR MSG

DEPOSIT CRAM FROM CONSOLE

 COMMAND:DC XX(96 BITS) INTO ADR SPECIFIED BY PREVIOUS EC,LC COMMAND
 DF XX(12 BITS) INTO ADR SPEC BY PREVIOUS EC,LC,LF
 DESC: CRA CONNECTS TO LEAST SIGNIFICANT 12 BITS OF KS10 BUS,THE CONSOLE
 CAN DEP/EXAMINE THE CRAM REGISTER,CURRENT LOCATION,THE TOP
 OF THE SUBRTN STACK AND THE CRAM ADDRESS LINES,THE SELECTION
 OF THE 12 BIT SEGMENT TO BE READ/Written IS DONE BY
 CSL4 DIAG 4,2,1 H. CSL4 DIAG 10 H DETERMINES WHETHER CRA OR
 CRM IS INVOLVED,36 BITS MICRO STORE IS ON CRA,60 BITS ON CRM,
 A BUFFERED COPY OF THE 12 BITS IS AVAILABLE TO CRM FROM CRA5.
 BUS DATA 21,22,23 ARE DRIVEN BY THE PARITY OF THE 12 BITS TO
 MAINTAIN EVEN PARITY ON THE KS10 BUS,THE DF COMMAND IS
 PROBABLY THE BETTER TROUBLESHOOTING AID BECAUSE THE PREVIOUS
 LF COMMAND WILL DEFINE KNOWN DIAG BITS AND 12 BIT DESTINATION
 IN MICRO STORE.

FLOW:

```

-----
8080 IO WRITE 210/144          ;8080 SETS CONSOLE REQ,T ENB
                                TO ENABLE TRANCIEVERS, AND
                                CRA R CLK TO LATCH THE DATA
                                TO CRA
|
|
|
-----
CSL4 DIAG 1,2,4,10 H          ;8080 DOES IO WRT 205 WITH DESIRED
                                DBUS BITS TO SET DESIRED 12 BIT SEGMENT
|PINS:F18C1
|      F18D1
|      F18E1
|      F18F1
-----
CSL4 CRAM WRITE H              ;8080 DOES IO WRITE 204/5 TO SET THIS
|PINS:F18A1                    |
|                              |
|                              |
-----
CRA4 WRITE XX-XX              CRA5 BUS DATA 24-35 L    ;DATA ARGUMENT FROM KS10
                                BUS LATCHED AS CRA5 BUS
                                BUFF 24-35;AVAILABLE TO
                                CRM AS BUS DATA 24B-35B
                                12 BITS STOBED TO RAMS
  
```

EXAMINE CRAM FROM CONSOLE

 COMMAND: EC XXXX(0-3777) OR EC
 DESC: PRINTS THE CONTENTS OF LOCATION XXXX IN MICRO STORE
 IF NO ARGUMENT, LAST CRAM ADDRESS EXAMINED OR DEPOSITED
 OR ADDRESS OF LAST LC COMMAND

FLOW:

 CLEAR THE CRAM REGISTER

| PINS: F18K1=
 | CSL4 CRAM RESET H
 |
 |

 LOAD THE CRAM ADDRESS

| PINS: C18D1=
 | CSL4 CRAM ADR LD H
 |
 |
 |
 |
 |
 |
 |
 |
 |

 PULSE THE CLOCK TO LOAD
 THE CRAM REGISTER

| PINS: D18C1=
 | CSL4 SS MODE H
 | C18U1=
 | CSL5 CRA/M CLK EN L
 |
 |

 READ THE CRAM REGISTER

| PINS: F18C1=CSL4 DIAG 10 H
 | F18D1=CSL4 DIAG 4 H
 | F18E1=CSL4 DIAG 2 H
 | F18F1=CSL4 DIAG 1 H
 |
 |
 |

 READ ALL CRAM BITS YET?

| YES | NO
 |-----|
 |
 |

 VERIFY A AND B COPIES

| YES | NO
 |-----|
 |
 |

;1/8080 IO WRT 204/1=SETS CRAM RESET WHICH
 GOES TO CRA6 AND CRM2 TO CLEAR THE CRAM REG
 2/8080 IO WRT 204/0 CLEARS THE RESET BIT

;1/8080 IO WRT 103,105,107,111,113 LOADS
 THE ADDRESS ARGUMENT INTO THE IO REGISTER
 2/8080 IO WRT 115/0 NO BUS CYCLES
 3/8080 IO WRT 210/144 =SETS CONSOLE REQ,
 T ENB,CRA R CLK
 4/8080 IO WRT 204/20 SETS CRAM ADR LD=LOADS
 THE DIAG ADR REG ON CRA5 FROM BUS BITS 24-
 35;THRU INPUT 0 OF MIXERS ON CRA1 THE CRAM
 ADR GOES TO INVERTERS ON CRA7;THE ENABLE CRA6
 DISP EN 20 L IS TRUE DUE TO PREVIOUS CRAM REG
 RESET
 5/8080 IO WRT 204/0 TO CLEAR THE CRAM ADR LD

;1/8080 IO WRT 204/010 SETS SINGLE STEP
 2/8080 IO WRT 206/2 SETS THE SINGLE CLK
 THIS ENABLES THE NEXT T CLK TO GATE THE
 ADDRESSED MICRO WORD INTO THE CRAM REG

;1/8080 IO WRT 205/FNC=(SEE TABLE)
 2/8080 IO WRT 210/111=SETS CONSL REQ,
 CRA T CLK,R CLK EN
 3/8080 IO RD 0,1,2,3,103 READS DATA FROM
 RECVR LATCHES TO 8080 RAM LOCATIONS
 4/8080 IO WRT 210/0 CLOSSES THE LATCHES

;CONTINUE IN THIS LOOP READING
 12 BITS AT A TIME OVER KS10 BUS 24-35
 UNTIL ALL CRAM BITS ARE IN 8080 RAM

;WHERE TWO COPIES EXIST COMPARE THEM IN 8080

CSL-9

MASTER RESET FROM CONSOLE

FLOW:

SET 0'S TO RUN,EXECUTE,CONTINUE

;8080 WRT IO 212/0 TO CLEAR THESE

! PINS: E18C1=RUN H
! E18S1=EXECUTE H
! D18V2=CONTINUE L
!

INSURE CPU IS STOPPED

;CLEAR SOFTWARE RUN FLAG
SET SINGLE STEP VIA IO WRT 204/10
CLR SINGLE CLK AND CLK RUN VIA
IO WRT 206/0

! PINS: D18C1=CSL4 SS MODE H
!
!

ISSUE RESET

;VIA 8080 IO WRT 100 WITH BIT 7

INSURE CLEAR 10 INT IS LOW

;VIA 8080 IO WRT 205/0

SET UP CURRENT PARITY SETTINGS

;VIA 8080 WRITE 100 +DESIRED BITS

ISSUE DP RESET,CRAM RESET

;VIA 8080 IO WRT 204/5

EXAMINE MEMORY FROM CONSOLE

DESC: THE CONSOLE MUST ISSUE A COMMAND ADDRESS CYCLE ON THE KS10
BUS WITH THE MEMORY ADDRESS. IT THEN WAITS FOR MEMORY TO
RESPOND WITH DATA AND THE SIGNAL DATA CYCLE WHICH LATCHES
THE DATA IN THE RECIEVERS.

FLOW:

SAVE THE ARGUMENT

;VIA 8080 IO WRT 103,105,107,111,113
SAVES 36 BITS (19 LS BITS MAX USED)

SET COM/ADR CYCLE

;VIA 8080 IO WRT 114/4 SETS THE BIT
IN THE 4 WORD MEM ON CSL8 PRINT

START THE COM /ADR CYCLE
PRESET NXM TIMER ON CSL3
ENABLE RECIEVER LATCHES

!PINS: B18J2=BUS COM/ADR CYCLE

;VIA 8080 IO WRT 210/343 COM/ADR
CYCLE BEGINS ON LEADING EDGE T CLK
AFTER BUS REQUEST AND BUS GRANT
THIS BUS REQUEST DOES NOT GO TO
BACKPANEL BECAUSE CONSOLE IS THE
BUS ARBITRATOR.

MAKE SURE BUS REQ WENT OUT

;VIA 8080 IO READ 301(LOOKING FOR 20)

!YES

!NO

BUS ARBITRATOR FAILED

;PRINT NO BUS RESPONSE ERROR MSG
SEE PRINT CSL1

MAKE SURE DID NOT GET NXM

;VIA IO READ 301(ABSENCE OF 100)
NXM TIMER WILL TIME OUT 4 R CLK
TICKS AFTER START OF COM/ADR IF NO MEM BUSY

!YES

!NO

BAD ADDRESS OR
HARDWARE FAILURE

;PRINT NXM MSG
LOOK AT DATA BITS ON
BUS AT COM/ADR TIME

LOOK FOR DATA ACKN

;VIA IO READ 301(LOOK FOR BIT 0)
SHOULD BE THERE AS A RESULT OF DATA CYCLE
COMING BACK FROM MEMORY

!PINS: B18L2=

!BUS DATA CYCLE L

!YES

!NO

MEM DID NOT RESPOND IN
15 MICRO SECS

;IF OTHER EXAMINE
COMMANDS WORK THE
MEMORY IS PROBABLY
BROKE

CSL-12

GET THE LATCHED DATA AND
PRINT IT

;VIA 8080 IO READ 0,1,2,3,103
IO WRT 210/0 TO ENABLE R CLKS AGAIN

CSL-13

DEPOSIT MEMORY FROM CONSOLE

CONDITIONS: THE DESIRED ADDRESS TO BE WRITTEN MUST HAVE BEEN LOADED
BY A PREVIOUS LA OR EM COMMAND
DESC; STORE THE ADDRESS AND DATA INTO 8080 IO REGISTERS, SET THE
APPROPRIATE BITS FOR THE COMMAND/ADDRESS CYCLE AND DATA CYCLE
AND DO IT BY ENABLING T CLKS.

FLOW:

STORE THE ADDRESS INTO IO REGISTERS

;VIA IO WRT FROM RAM TO
8080 REGS 103,105,107,111,113
ADDRESS ARGUMENT IS FROM PREV:
EM OR LA COMMAND

STORE THE FUNCTION BIT FOR COM/ADR
MEMORY WRITE

;VIA 8080 IO WRT 113/2

STORE THE COM/ADR CYCLE BIT

;VIA 8080 IO WRT 115/4

STORE THE DATA ARGUMENT IN
IO REGISTERS

;VIA 8080 IO WRT 102,104,106,108

STORE THE BIT FOR DATA CYCLE

;VIA 8080 IO WRT 114/1

DO IT
SET BITS FOR CHECK NXM, CONSOLE REQUEST
T ENABLES

;VIA 8080 IO WRT 210/360
ISSUE COMMAND ADDRESS CYCLE ON
FOLLOWED BY DATA CYCLE ON NEXT
T CLK

MAKE SURE BUS REQUEST WAS GENERATED
INTERNAL ON CSL BOARD

;VIA IO READ 301 (LOOK FOR 20)
;ARBITRATOR ON CSL1 PROBABLY
FAILED IF NO BUS RESPONSE MSG

MAKE SURE DID NOT GET NXM

;VIA IO READ 301 (100=NXM)
MEM SHOULD RESPOND ON THIRD T
AFTER COM/ADR WITH MEM BUSY TO
CLEAR THE COUNTER

CSL-14

COMMAND:SM XX UCODE NORMALLY STARTS AT ZERO WHICH IS DEFAULT ARGUMENT

ADDRESS ARGUMENT ZERO??

! NO

STORE THE ADR

VIA DEPOSIT MEMORY CODE-SEE DEP MEM FLOWCHART

VIA MASTER RESET CODE-SEE FLOWCHART

```

;ZERO IS DEFAULT--ADDRESS IS WRITTEN INTO
THE DIAGNOSTIC ADDRESS REGISTER ON CRA5
1/CRAM RESET VIA IO WRT 204/1
2/STORE CRAM ADR IN 8080 IO ADR REGISTER
VIA IO WRT 103,105,107,111,113
3/SHIP IT TO CRA5 DIAG ADR REG
VIA IO WRT 210/144 AND 204/20

```

1/CRAM RESET VIA IO WRT 204/1
2/STORE CRAM ADR IN 8080 IO ADR REGISTER
VIA IO WRT 103,105,107,111,113
3/SHIP IT TO CRA5 DIAG ADR REG
VIA IO WRT 210/144 AND 204/20

2/STORE CRAM ADR IN 8080 IO ADR REGISTER
VIA IO WRT 103,105,107,111,113
3/SHIP IT TO CRA5 DIAG ADR REG
VIA IO WRT 210/144 AND 204/20

BY SETTING SINGLE CLK AND CLOCK RUN
VIA IO WRT 206/3

VIA IO RD 300(LOOKING FOR 10)

MICROCODE WILL SET A FLOP ON THE CSL4
PAGE WHEN IT ENTERS THE HALT LOOP.
IF ITS NOT ON PRINT AN ERROR MESSAGE

THIS IS WHERE THE MICROCODE WILL STORE
THE HALT STATUS WORD NORMALLY ZERO ON STARTUP
SEE THE EX MEM FLOW IF PROBLEMS HERE

```

;EXAMINE MEM LOC 1 TO GET THE PC
MICROCODE PUTS IT THEREAS WELL AS
IN HALT STATUS BLK

```

EXECUTE KS10 INSTRUCTION FROM THE CONSOLE

 COMMAND: EX XX(36 BITS)

DESC: THE COMMAND EXECUTES A 10 INSTRUCTION, THE MICROCODE MUST BE STARTED
 AND SITTING IN THE HALT LOOP, THE CPU ISSUES AN IO READ TO THE
 CONSOLE IN RESPONSE TO EXECUTE AND CONTINUE. THE INSTRUCTION IS
 PASSED TO THE CPU OVER THE KS10 BUS AND THE MICROCODE EXECUTES IT.

FLOW:

 STORE THE ARGUMENT

;VIA 8080 IO WRT 102,104,106,110,112
 THIS IS 8080 DATA REGISTER

!
 !

 SET THE BIT FOR IO DATA CYCLE

;VIA 8080 IO WRT 114/2=THIS WILL STROBE
 THE INSTRUCTION TO THE CPU WHEN UCODE ASKS

!
 !

 SET THE BITS FOR EXECUTE
 AND CONTINUE

;VIA 8080 IO WRT 212/3

 !PINS: D18T2=CSL4 EXECUTE L
 ! D18V2=CSL4 CONTINUE L
 ! B18N2=BUS IO DATA CYCLE L
 !

 CONTINUE CLEARED BY CPU YET?

;VIA IO READ 300(SHOULD BE 0)

 !YES

!NO

;MICRO CODE SHOULD CLEAR CONTINUE

!
 !

PRINT ERR MSG

DNF=DID NOT FINISH

!
 !

 DONE

;IF WE HAD DONE A ST COMMAND TO START THE KS10
 WE WOULD HAVE JUST FINISHED EXECUTING A JRST
 NOW ALL WE NEED TO DO IS ISSUE CONTINUE AND RUN
 AND WE'R IN BUSINESS.

MICRO CODE ACTIVITY-THE CPU IS SITTING IN THE HALT LOOP(4-5)
 WAITING FOR EXECUTE AND CONTINUE THE SEQUENCE FOLLOWS
 NOTE: LOCATIONS MAY CHANGE WITH VERSIONS-THIS IS VER 10

LOC: MACRO:

COMMENT:

4

HALT LOOP

2225 [AR]_VMA IO READ

;SET UP THE COMMAND ADDRESS BITS

3500 [AR]_[AR],OR. #, #/200000, H LEFT

;ADDRESS OF CSL REGISTER

3501 VMA_[AR] WITH FLAGS

;START THE IO READ

3502 MEM RD, [HR]_MEM, LOAD INST

;IR GETS THE INSTRUCTION

3503 EA MODE DISP, J/EACALC

;GO DO IT-CLEAR CONTINUE WHEN DUN

CONSOLE BOARD UARTS--SWITCH CONFIGURATIONS

=====

CTY--BOTTOM BERG CONNECTOR--SWITCH PACK E508 FOR BAUD RATE SELECTION

KLINIK--TOP BERG CONNECTOR--SWITCH PACK E509 FOR BAUD RATE SELECTION

BAUD RATE	SWITCHES					
	1	2	3	4	5*	6
=====						
110	0	0	0	0	0	0

150	0	0	0	1	0/1	0

300	0	0	1	0	1	0

600	1	0	0	0	1	0

1200	0	1	0	0	1	0

1800	0	1	0	1	1	0

2400	0	0	1	1	1	0

4800	0	1	1	0	1	0

9600	0	1	1	1	1	0

*NOTE: SWITCH 5=STOP BIT SELECTION 0=2 STOP BITS 1=1 STOP BITS

READ I/O 0

```
*****
| DBUS7 | DBUS6 | | DBUS5 | DBUS4 | DBUS3 | | DBUS2 | DBUS1 | DBUS0 |
| | | | | | | | | |
| R DATA 28 | R DATA 29 | | R DATA 30 | R DATA 31 | R DATA 32 | | R DATA 33 | R DATA 34 | R DATA 35 |
| | | | | | | | | |
*****
```

READ I/O 1

```
*****
| DBUS7 | DBUS6 | | DBUS5 | DBUS4 | DBUS3 | | DBUS2 | DBUS1 | DBUS0 |
| | | | | | | | | |
| R DATA 20 | R DATA 21 | | R DATA 22 | R DATA 23 | R DATA 24 | | R DATA 25 | R DATA 26 | R DATA 27 |
| | | | | | | | | |
*****
```

READ I/O 2

```
*****
| DBUS7 | DBUS6 | | DBUS5 | DBUS4 | DBUS3 | | DBUS2 | DBUS1 | DBUS0 |
| | | | | | | | | |
| R DATA 12 | R DATA 13 | | R DATA 14 | R DATA 15 | R DATA 16 | | R DATA 17 | R DATA 18 | R DATA 19 |
| | | | | | | | | |
*****
```

READ I/O 3

```
*****
| DBUS7 | DBUS6 | | DBUS5 | DBUS4 | DBUS3 | | DBUS2 | DBUS1 | DBUS0 |
| | | | | | | | | |
| R DATA 4 | R DATA 5 | | R DATA 6 | R DATA 7 | R DATA 8 | | R DATA 9 | R DATA 10 | R DATA 11 |
| | | | | | | | | |
*****
```

READ I/O 100

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
REC PE	SPARE PE	ADPT	CRAM	MEM	DP PARITY	CWA PARITY	SPARE PARITY
L	L	PARITY ERR	PARITY ERR	PARITY ERR	ERR L	ERR L	ERR L

READ I/O 101

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
PI REQ 1	PI REQ 2	PI REQ 3	PI REQ 4	PI REQ 5	PI REQ 6	PI REQ 7	MMC REF
							ERR B
							H

READ I/O 102

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
R AC LO	R PESET	R MEM BUSY	R I/O BUSY	R BAD DATA	R COM ADR	R I/O DATA	R DATA

READ I/O 103

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
SPARE PE	SPARE PE	R PAR RIGHT	R PAR LEFT	R DATA 0	R DATA 1	R DATA 2	R DATA 3
L	L						

READ I/O 300

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
CTY	CTY CHAR	KLINIK	KLINIK	DPM	RUN (1)	EXECUTE	CONTINUE
BIT #	LENGTH	BIT #	LENGTH	HALT LOOP	H	H	H
(SW)	(SW)	(SW)	(SW)				

READ I/O 301

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
10	WXM	PANEL	BUS	PE (1)	CONSOLE	ROOT	DATA ACK
INTERRUPT	H	LOCAL	REQ	H	ENABLE H	H	H
H		H			(SW)	(SW)	

READ I/O 302

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
0	0	0	0	I.D. H	KLINIK	TERMINAL	KLINIK
					ENABLE	CARRIER	CARRIER

READ I/O 303

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
0	0	0	0	0	0	0	0

WRT I/O 204

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
?	?	CRAM WRITE	CRAM ADR LOAD	SS MODE	DP RESET H	STACK RESET H	CRAM RESET H

WRT I/O 205

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
?	?	CLR 10 INTERRUPT L	TRAP ENB	CRAM READ	DIAG 4	DIAG 2	DIAG 1

WRT I/O 206

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
?	?	?	?	?	?	SINGLE CLK H	CLK RUN H

WRT I/O 210

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
CHECK NXM	CONSOLE REQ	T ENB FOR COM/ADR	T ENB FOR DATA CYCLE	CRA T CLK	CRA R CLK	LATCH DATA SENT	R CLK FNR L

WRT I/O 212

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
?	?	?	?	?	RUN	EXECUTE	CONTINUE

```

WRT 100
*****
DBUS7    DBUS6    DBUS5    DBUS4    DBUS3    DBUS2    DBUS1    DBUS0
RFSFT    PE DETECT  CRM PE DETECT  DP PE DETECT  CACHE ENABLE  1 MSEC 0    0
          ENABLE          DISAB H
*****

```

```

WRT 101
*****
DBUS7    DBUS6    DBUS5    DBUS4    DBUS3    DBUS2    DBUS1    DBUS0
0        0        0        0        MODEM TEST  CARRIER FAULT
          DTR
*****

```

```

WRT 102/103 DATA/ADR
*****
DBUS7    DBUS6    DBUS5    DBUS4    DBUS3    DBUS2    DBUS1    DBUS0
DATA 28  DATA 29  DATA 30  DATA 31  DATA 32  DATA 33  DATA 34  DATA 35
*****

```

```

WRT 104/105 DATA/ADR
*****
DBUS7    DBUS6    DBUS5    DBUS4    DBUS3    DBUS2    DBUS1    DBUS0
DATA 20  DATA 21  DATA 22  DATA 23  DATA 24  DATA 25  DATA 26  DATA 27
*****

```

```

WRT 106/107 DATA/ADR
*****
DBUS7    DBUS6    DBUS5    DBUS4    DBUS3    DBUS2    DBUS1    DBUS0
DATA 12  DATA 13  DATA 14  DATA 15  DATA 16  DATA 17  DATA 18  DATA 19
*****

```

WRT 110/111 DATA/ADR

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
DATA 4	DATA 5	DATA 6	DATA 7	DATA 8	DATA 9	DATA 10	DATA 11

WRT 112/113 DATA/ADR

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
0	0	0	0	DATA 0	DATA 1	DATA 2	DATA 3

WRT 114/115 ADR/DATA

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
0	0	0	0	RAD DATA CYCLE	COM/ADR CYCLE	I/O DATA CYCLE	DATA CYCLE

WRT 116

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
0	0	0	0	0	0	0	CSL INTERRUPT THE 10

WRT I/O 200 CTY UART WRITE STATUS REGISTER(DATA BUFFER IS I/O 201)

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
HUNT MODE	UART	REQUEST	RESET	SEND	RECEIVE	TERMINAL	TRANSMIT
ON(SYNC)	RESET	TO SEND L	ERRORS	BREAK CHAR	ENABLE	READY	ENABLE
				H		L	

WRT I/O 200 CTY UART READ STATUS REGISTER(DATA BUFFER IS I/O 201)

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
DATA SFT	SYNC	FRAMING	OVERPUN	PARITY	TRANSMITTER	RECEIVER	TRANSMITTER
READY	DETECT	ERROR	ERROR	ERROR	EMPTY	READY	READY

WRT I/O 202 REMOTE UART WRITE STATUS REGISTER(DATA BUFFER IS I/O 203)

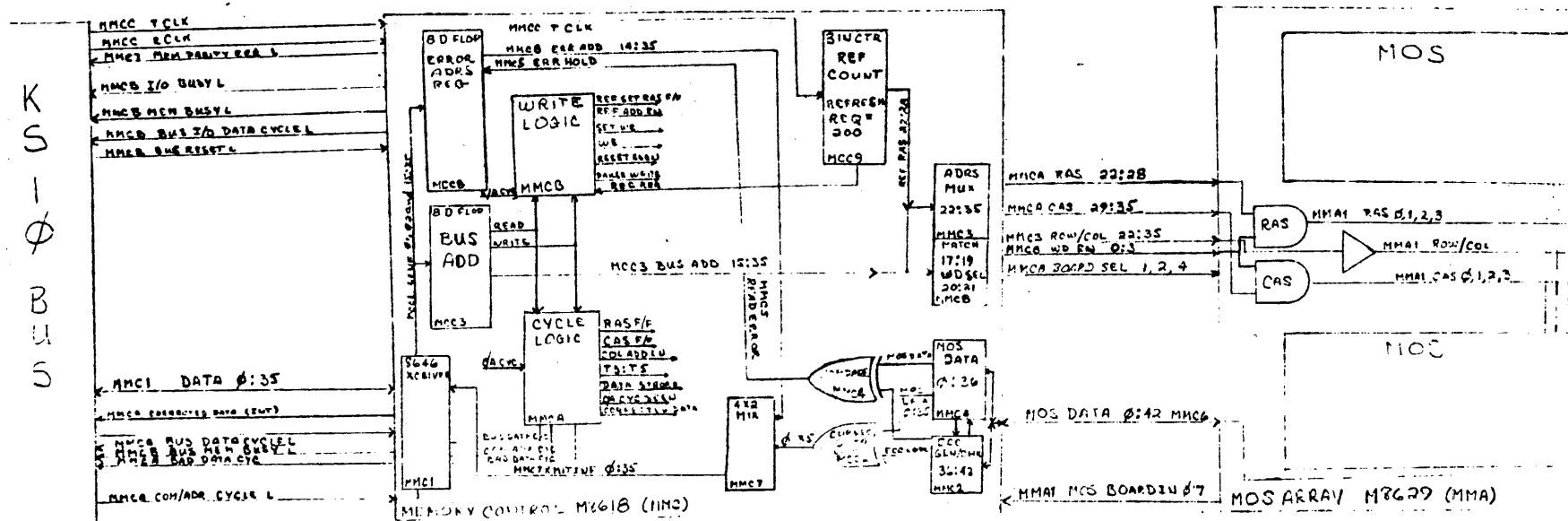
DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
HUNT MODE	UART	REQUEST	RESET	SEND	RECEIVE	TERMINAL	TRANSMIT
ON(SYNC)	RESET	TO SEND L	ERRORS	BREAK CHAR	ENABLE	READY	ENABLE
				H		L	

WRT I/O 202 REMOTE UART READ STATUS REGISTER(DATA BUFFER IS I/O 203)

DBUS7	DBUS6	DBUS5	DBUS4	DBUS3	DBUS2	DBUS1	DBUS0
DATA SET	SYNC	FRAMING	OVERRUN	PARITY	TRANSMITTER	RECEIVER	TRANSMITTER
READY	DETECT	ERROR	ERROR	ERROR	EMPTY	READY	READY

LF

K
S
-
Φ
B
U
S



WRITE STATUS
REGISTER

MMC1 LINE 28 M	MMC1 LINE 29 H	MMC1 LINE 30 H	MMC1 LINE 31 H	MMC1 LINE 32 H	MMC1 LINE 33 H	MMC1 LINE 34 H	MMC1 LINE 35 H
MMC2 FORCE C0L	MMC3 FORCE C40L	MMC3 FORCE C00L	MMC3 FORCE C10L	MMC3 FORCE C4L	MMC3 FORCE C2L	MMC3 FORCE C1L	MMC3 FORCE C00L

MMC-6

MMC-5	MMC-4	MMC-3	MMC-2	MMC-1	MMC-0	MMC-7	MMC-6	MMC-5	MMC-4	MMC-3	MMC-2	MMC-1	MMC-0
ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR	ERR
10	11	12	13	14	15	16	17	18	19	20	21	22	23

EEP ADDR. BITS

MOS MEMORY
MEM-1

BIT ---	FUNCTION -----	
00	MMC5 ERR HOLD	SET WHEN ANY ERROR CONDITION IS DETECTED BY THE MEMORY CONTROLLER. CLEARED BY WRITING A 1 INTO BIT 00 OF THE CONTROL STATUS REGISTER.
01	MMC4 UNCOR ERR HOLD	SET INDICATES THAT THE ERROR BEING HELD WAS UNCORRECTABLE OR THAT THE WORD THAT HAD THE ERROR HAD MORE THAN ONE BIT WRONG.
02	MMC9 REF ERR	REFRESH ERR IS SET 5.5 U SEC. AFTER A WRITE OR READ PAUSE WRITE COMMAND ADDRESS IS ISSUED IF NO WRITE DATA IS RECEIVED. REFRESH ERR CLEARS OUT ANY CYCLE AND PREVENTS ANY MORE TRANSFERS TO OR FROM THE MOS CHIPS. THE CONTROLLER WILL CONTINUE TO REFRESH THE MOS CHIPS SO NO LOSS OF MEMORY WILL OCCUR. CLEARED BY WRITING A 1 INTO BIT 02 OF THE STATUS REGISTER, OF MR.
03	MMC7 PARITY ERR ASSERTED	WHEN THE MEMORY CONTROLLER HAS DETECTED A PARITY ERROR ON THE KS10 BUS. THIS ERROR CAN BE SET BY WRITING A 1 INTO BIT 03 OF THE STATUS REGISTER. PARITY ERR CAN BE CLEARED BY WRITING A 0 INTO BIT 03 OF THE STATUS REGISTER OF MR.
04	MMC3 ECC ON	ASSERTED WHEN THE CORRECTION LOGIC IS ON. WITH THIS BIT OFF THE MEMORY CONTROLLER WILL STILL DETECT ERRORS BUT NO CORRECTIONS WILL BE MADE. ECC IS TURNED ON OR WRITING A 0 INTO BIT 04 OF THE STATUS REGISTER.
05	MMC4 ERR C5	PARITY BIT OF THE CORRECTION CODE.
06	MMC4 ERR C40	CORRECTION CODE BIT C40.
07	MMC4 ERR C20	CORRECTION CODE BIT C20
08	MMC4 ERR C10	CORRECTION CODE BIT C10
09	MMC4 ERR C4	CORRECTION CODE BIT C4
10	MMC4 ERR C2	CORRECTION CODE BIT C2

11	MMCH ERR C1	CORRECTION CODE BIT C1
12	MMCH POWER FAILED	SET IF THE MEMORY HAS LOST POWER OR IF THE BATTERY BACK-UP OPTION IS IN AND THE BATTERY IS NOT CHARGED OR, CLEARED BY WRITING 0 INTO BIT 12.
13	UNUSED	ALWAYS UNASSERTED
14	MMCH ERR ADDR 14	BIT 14 OF THE ERROR ADDRESS REGISTER.
15	MMCH ERR ADDR 15	BIT 15 OF THE ERROR ADDRESS REGISTER.
16	MMCH ERR ADDR 16	BIT 16 OF THE ERROR ADDRESS REGISTER.
17	MMCH ERR ADDR 17	BIT 17 OF THE ERROR ADDRESS REGISTER.
18	MMCH ERR ADDR 18	BIT 18 OF THE ERROR ADDRESS REGISTER.
19	MMCH ERR ADDR 19	BIT 19 OF THE ERROR ADDRESS REGISTER.
20	MMCH ERR ADDR 20	BIT 20 OF THE ERROR ADDRESS REGISTER.
21	MMCH ERR ADDR 21	BIT 21 OF THE ERROR ADDRESS REGISTER.
22	MMCH ERR ADDR 22	BIT 22 OF THE ERROR ADDRESS REGISTER.
23	MMCH ERR ADDR 23	BIT 23 OF THE ERROR ADDRESS REGISTER.
24	MMCH ERR ADDR 24	BIT 24 OF THE ERROR ADDRESS REGISTER.
25	MMCH ERR ADDR 25	BIT 25 OF THE ERROR ADDRESS REGISTER.
26	MMCH ERR ADDR 26	BIT 26 OF THE ERROR ADDRESS REGISTER.
27	MMCH ERR ADDR 27	BIT 27 OF THE ERROR ADDRESS REGISTER.
28	MMCH ERR ADDR 28	BIT 28 OF THE ERROR ADDRESS REGISTER.
29	MMCH ERR ADDR 29	BIT 29 OF THE ERROR ADDRESS REGISTER.
30	MMCH ERR ADDR 30	BIT 30 OF THE ERROR ADDRESS REGISTER.
31	MMCH ERR ADDR 31	BIT 31 OF THE ERROR ADDRESS REGISTER.
32	MMCH ERR ADDR 32	BIT 32 OF THE ERROR ADDRESS REGISTER.
33	MMCH ERR ADDR 33	BIT 33 OF THE ERROR ADDRESS REGISTER.
34	MMCH ERR ADDR 34	BIT 34 OF THE ERROR ADDRESS REGISTER.
35	MMCH ERR ADDR 35	BIT 35 OF THE ERROR ADDRESS REGISTER.

THE MEMORY WILL HOLD THE ERROR ADDRESS, THE ECC CODE AND THE TYPE OF ERROR FOR THE FIRST ERROR THAT IS DETECTED. BIT 00, THE ERROR HOLD BIT MUST BE CLEARED TO ALLOW THE MEMORY TO HOLD A NEW ERROR.

ADDRESS BITS

- 14-16 SELECT WHICH MEMORY CONTROLLER
- 17-19 SELECT WHICH MEMORY ARRAY BOARD
- 20-21 SELECTS ONE OF 4 ADDRS ON THE SELECTED ARRAY BOARD.
- 22-28 7 ROW ADDRESS LINES
- 29-35 7 COLUMN ADDRESS LINES

MOS MEMORY

WRITE CYCLE

A write cycle is started when the controller receives a write command address word from the KS10 Bus. The format is as follows:

D0 = 0	Memory Operation
D1 = 0	Not a read operation (MMC3)
D2 = 1	Write Operation (MMC3)
D14-16	Memory Controller (always zero)
D17-19	Select Array Board (MMC3)
D20-21	4 Word Select on Array Board (MMC3)
D22-28	Row Address (MMC3)
D29-35	Column Address (MMC3)

When the Command Address Cycle L (MMCA) starts the operation the controller compares the address portion of the command address to see if it is an existing address giving an Address Match (MMC8). Address Match and - Ref. Add En (MMCB) with (MMCC)R Clk causes RAS F/F (MMCA) (Row Address Strobe) to set. This sends RAS to selected array board the RAS signal strobes the 7 (seven) address lines into one of four words of MOS chips on the selected array board.

On the next "R" Clk we switch the address lines with Column Address Enable L (MMCA) and 75 nsec later "R" Clk B L (MMCC) sets CAS (Column Address Strobe MMCA) sending the column address to the array boards. The CAS signal strobes the 7 address lines into the column part of the MOS chips. This conditions one of 16k word for the data to be written into memory. This part of the write operation is the same as the read operation.

Bus Data Cycle L (MMCA BUS Data H) tell the controller when the data to be written has arrived from the KS10 BUS. All timing is stopped until BUS DATA H (MMCA) generates WR DATA L (MMCB) at which time "R" Clk. are enabled again (MMCB). The data is transmitted on MOS DATA

(MMC6) 0-35 and into the control board. The signal GEN CHECK BITS H(MMCB) causes 7(seven) bit check code to be generated (CP, C40, C20, C10, C4, C2, C1) on 4x2 mixer (MMC4). This check code goes out and becomes MOS DATA 36-42 (MMC6) and is store in memory with MOS DATA 00-35 (MMC6). The 43 bits of data are written into the selected word of the MOS RAMS when the write signal WE H (MMCB) is sent to the selected array board.

The signal MEM BUSY (MMCB) get set at data cycle time or 2"T" clks after C/A cycle (MMCA).

MOS MEMORY

READ PAUSE WRITE

A Read Pause Write is started when the controller receives a write and Read command address word from the KS10 Bus. The format is as follows:

D0 = 0	Memory Operation
D1 = 1	Read Operation (MMC3)
D2 = 1	Write Operation (MMC3)
D14 = 16	Memory Controller (Always Zero)
D17-19	Select Array Board (MMC3)
D20-21	4 Word Select on Array Board (MMC3)
D22-28	Row Address (MMC3)
D29-35	Column Address (MMC3)

The generation of RAS & CAS is the same as the read and write operations. Because a read and write was decoded on MMC3, MMCA generates Pause. The signal Read Pause 2nd Half (MMCB) let the data read go down KS10 bus at which time any correction or detection would have been done. The read pause 2nd half signal generates Pause Write (MMCB) which causes GEN CHECK BITS H (MMCB) and BUS DATA H (MMCA) to write the MOS DATA 00 - 35 and 36-42 into the selected word of the MOS RAMS when W E H (MMCB) is transmitted. The first bus data cycle (MMCA) transmits the data read from the memory to the master of the KS10 Bus. The second data cycle (MMCA) tell the controller when the data to be written has arrived from the KS10 Bus master.

jp

MOS MEMORY

REFRESH CYCLE

Each location in the MOS IC's must be refreshed every 2 msec. and take approximately 15 usec. for each row. The refresh address and refresh count are done (MMC9) by 4 binary counters which are controlled by "T" clocks. The first two counters track the refresh address. The next two count to 200 at which time a refresh request is made and the refresh address is dated to the address mixer by the Refresh Address Enable (MMCB) signal. This causes Refresh Set RAS (MMCB) which enables the row address strobe to all MOS chips on all array boards.

If there is only a count of 16 left and refresh request (MMC9) is still set then Refresh Error F/F will set (MMC9). This causes the memory to lock out. The only thing that will happen is memory will constantly refresh to preserve data integrity in MOS memory.

jp

MOS MEMORY

READ CYCLE

A read cycle is started when the controller receives a read command address word from the KS10 Bus. The format is as follows:

D0 = 0	Memory Operation
D1 = 1	Read Operation (MMC3)
D14 - 16	Memory Controller (Always Zero)
D17 - 19	Select Array Board (MMC3)
D20 - 21	4 Word Select on Array Board (MMC3)
D22 - 28	Row address (MMC3)
D29 - 35	Column Address (MMC3)

The command address cycle 1 (MMCA) starts the operation. The controller compares the address portion of the command address word to see if it is an existing address giving a address match (MMC8). Address Match and - Ref Add En (MMCB) with (MMCC)R Clk causes RAS (MMCA)Row address strobe to set And sends RAS to selected array board the RAS (MMCA) signal strobes the 7(seven) address lines into one of four words of chips on the selected array board.

On the next "R" Clk we switch the address lines with Column Address Enable 1 (MMCA) and 75 nsec later "R"ClkBL (MMCC) sets CAS (Column Address Strobe MMCA) sending the column address to the array boards. The CAS signal strobes the 7 address lines into the column part of the MOS RAMS and starts reading from one of the 16k words. The data arrives at the output of the MOS RAMS in about 160 nsec later and proceeds down the MOS data lines to the Control board. A delay is set up by F/F T3 and T4. The setting of F/F T4 (MMCA) starts the transmit (MMCA) data down the KS10 Bus to the Bus Master. This data has not been checked yet for single or double bit errors. If there are no single or double bit errors the word is transmitted again with the signal data cycle 1 (MMCA) pin BL2. Checking for single and double bit is done by comparing the MOS DATA bits (MMC4) with the seven ECC bit (MMC4) which should cancel each other out. If a single bit is in error correct enable is generated (MMC5) which causes the placement of the correct bit (MMC2) on the KS10 bus. This corrected word is transmitted for two cycles. Memory Busy (MMCB) will stay set until data cycle occurs. If there is a double bit

detected Uncorrectable Err H (MMC5) is generated which causes Bad Data XMIT (MMCA) to send signal BUS BAD DATA CYCLE L. (MMCA) Bad data cycle will produce a page fail in the CPU if the CPU requested the word. Bad data cycle will produce a time out if the UBA or some other I/O device requested the word.

() = Page number

jp

MEM-14

UNIBUS TO KS10 ADAPTER

THE ADAPTER IS AN INTERFACE BETWEEN THE UNIBUS AND THE KS10 BUS
ADDITIONAL ADAPTERS MAY BE PRESENT ON THE KS10 BUS ,UP TO 4 TOTAL
THE UNIBUS SIDE WILL HAVE A BUS ARBITRATOR,
THE KS10 SIDE WILL APPEAR AS AN I/O DEVICE
THE UNIBUS ARBITRATOR WILL HAVE THE FOLLOWING REQUEST:

INT	INTERRUPT
NPR	DATO TO MEMORY DATI TO MEMORY DATOB TO MEMORY
CPU	I/O READ I/O WRITE
NPR	TRANSFERS DATA TO AND FROM MEMORY EVEN UNIBUS WORD ADDRESS WRITES WILL ZERO THE LOWER HALF WORD IN MEMORY UNLESS THE READ PAUSE WRITE BIT IS SET WHICH FORCES A READ PAUSE WRITE CYCLE ODD UNIBUS WORD ADDRESS WRITES WILL CAUSE A READ PAUSE WRITE CYCLE ADDRESSES WILL BE MAPPED VIA 64 REGISTERS
BR7-4	WILL INITIATE AN INTERRUPT REQUEST TO THE CPU PI LOGIC BR6&7 ARE PROGRAMMABLE AS ONE LEVEL WHILE BR5&4 ARE PROGRAMMABLE AS ANOTHER LEVEL.
CPU	CAN PERFORM A UNIBUS I/O REGISTER READ OR WRITE. THEY WILL CAUSE A CPU REQUEST TO BE ISSUED TO THE UNIBUS ARBITRATOR, I/O READ DATA WILL BE LATCHED WITH SSYNC WHICH WILL REQUEST THE KS10 BUS. THE CPU WILL RECIEVE I/O DATA WHEN THE ADAPTER BECOMES THE KS10 BUS MASTER. I/O WRITE DATA IS SEND TO THE DEVICE WITH MSYNC. THE CPU CAN ALSO ADDRESS MEMORY ON THE UNIBUS IF IT IS IN THE RANGE OF 64-124K
INT	INITIATED BY AN I/O READ WITH BIT 5 SET WHICH CAUSES AN INT REQUEST TO THE UNIBUS ARBITRATOR. WHEN GRANTED THE MODULE WILL LATCH THE VECTOR ADDRESS AND REQUEST THE KS10 BUS TO FINISH THE I/O READ.

INTERNAL REGISTERS

763100 STATUS REGISTER

7630XX 64 MAPING REGISTERS LOADED VIA SM10 BUS

STATUS REGISTER 763100

READ/WRT BIT 18 INDICATES UNIBUS ARBITRATOR TIMEOUT
NO SACK RECEIVED WITHIN 5 MICRO SEC. OR
CP ADDRESSED NON EXISTANT DEVICE REGISTER
OR DEVICE ADDRESSED NON EXISTENT MEMORY
1 CLEARS BIT

READ/WRT BIT 19 INDICATES BAD MEMORY DATA ON XFR TRANSFER.
MASTER WILL TIME OUT ON BAD MEMORY DATA IF
BIT 28 IS SET
1 CLEARS BIT

READ/WRT BIT 20 INDICATES SM10 BUS PARITY ERROR
1 CLEARS BIT

READ/WRT BIT 21 CPU ADDRESSED NON EXISTENT DEVICE
1 CLEARS BIT

READ BIT 24 BR6 OR 7 INTERRUPT REQUEST

READ BIT 25 BR 5 OR 4 INTERRUPT REQUEST

READ BIT 26 AC OR DC LOW
CLEARED ON REGISTER WRITE

RD/WRT BIT 28 DISABLE TRANSFER ON UNCORRECTABLE DATA

WRT ONLY BIT 29 ISSUE UNIBUS INITIALIZE

RD/WRT BIT 30-32 PT LEVEL OF BR6,7 AND

RD/WRT BIT 33-35 PT LEVEL OF BR5 AND4

```

*****
 18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
TIME  BAD  KS10  CPU          BR6  BR5  AC          DISABLE INIT (PT LE
                                1    32    33    34    35
                                VEL HIGH) (PT LEVEL LOW)
*****

```

MAINTENANCE REGISTER 763101

WRT ONLY BIT 35 CHANGE MPR REG

WRT ONLY BIT 34 CHANGE MPR WRAP AROUND

MAPPING REGISTERS 763200-763277

LOADED VIABITS 21-35

BIT 21 IS VALID BIT

BIT 22 IS 36 BIT FAST XFER ENABLE (EVEN * WORDS REQUIRED)

BIT 19 IS ENABLE UPPER TWO BITS ON XFER

BIT 18 IS MAINTENANCE PAUSE FOR EVEN AND ODD MEMORY WRITES

READ VIA BITS 16-26 AND

BIT 9 IS PAGING RAM PARITY COMPUTED ON READ

BIT 8 IS VALID BIT

BIT 7 IS 36 BIT FAST XFER ENABLE

BIT 6 IS DISABLE UPPER 2 BITS

BIT 5 IS MAINTENANCE PAUSE

BIT 4 IS RAM PARITY

MPR ADDRESS CONVERSION UNIBUS--GML

11 ADDRESS

16 15 14 13 12 11 10

2 1 0

11 BYTE
--HALF WORD

UNIBUS ADDRESS BITS 16-11 ADDRESS THE PAGING RAM WHICH SUPPLIES
THE UPPER ELEVEN BITS OF 10 ADDRESS

LF

UBA-3

THE 11 SIDE OF THE UNIBUS ADAPTER RESPONDS TO EVERY MSYNC EXCEPT FOR ADDRESSES GREATER THAN 64K IF THE ASSOCIATED VALID BIT IS SET AND THE PAGING RAM PARITY MAY BE PRESENT ON THE UNIBUS IF ITS ADDRESS IS BETWEEN 64 AND 124K

UNIBUS DATA FORMAT:		PB PA 17		0
UNIBUS DATA CONVERSION:		ADD 0	A	B
		ADD 2	C	D
	0		17 18	35
10 WORD		A	B	C D

I/O REGISTER CONVERSION SM10-- UNIBUS

10 ADDRESS	18	35
11 ADDRESS	17	00

UNIBUS ADAPTER WRAP AROUND TEST

```

=====
EI UBA#,763000      ;ADR OF PAGING RAM FOR LOC ZERU
DI 40000 (VALID),140000(36 BIT,VALID),240000(18 BIT,VALID)
EI UBA#,763101      ;ADR OF UBA MAINT REG
DI 1                ;BIT 35 FOR WRAPAROUND
EI UBA#,ADR         ;LOAD MEM ADR TO BE WRITTEN
                    ;LOW ORDER 2 UNIBUS ADDRESS BITS ARE DISCARDED ,WE ARE DOING
                    ;A 16 BIT WRAPAROUND;C00 SAYS 8 BIT UNIBUS BYTE MODE OR NOT
                    ;C1 SAYS RH OR LH
                    ;EXAMPLES      3,,0=LH MEM LOC 0
                                    3,,2=RH MEM LOC 0
                                    3,,4=LH MEM LOC 1
                                    3,,6=RH MEM LOC 1
DI XXXXXXXX(DATA PATTERN)      ;WILL LOOP THRU UBA TO MEM LOCATION
EM MEM LOC                      ;DID IT GET THERE??

```

UBA-5

UNIBUS SIGNAL PIN FINDER

=====

UNIBUS SIGNAL NAME	BACKPANEL PIN ASSIGNMENT	M9014 CONNECTOR PIN ASSIGNMENT JACK PIN
--------------------------	--------------------------------	---

=====

A00L	BH2	J1	D
A01L	BH1	J1	B
A02L	BJ2	J1	J
A03L	BJ1	J1	F
A04L	BK2	J1	N
A05L	BK1	J1	L
A06L	BL2	J1	T
A07L	BL1	J1	R
A08L	BM2	J1	X
A09L	BM1	J1	V
A10L	BN2	J1	BB
A11L	BN1	J1	Z
A12L	BP2	J1	FF
A13L	BP1	J1	DD
A14L	BR2	J1	LL
A15L	BR1	J1	JJ
A16L	BS2	J1	RR
A17L	BS1	J1	NN
ACLOL	BF1	J2	VV
BBSY L	AP2	J3	BB
BG4 H	BE2	J3	X
BG5 H	BR1	J3	T
BG6 H	BA1	J3	N
BG7 H	AV1	J3	J
BR4 L	BD2	J3	V
BR5 L	BC1	J3	R
BR6 L	AU2	J3	L
BR7 L	AT2	J3	F
C0 L	BU2	J3	NN
C1 L	BT2	J1	TT
D00 L	AC1	J2	B
D01 L	AD2	J2	F
D02 L	AD1	J2	D
D03 L	AE2	J2	L
D04 L	AF1	J2	J
D05 L	AF2	J2	R
D06 L	AF1	J2	N
D07 L	AH2	J2	V
D08 L	AH1	J2	T
D09 L	AJ2	J2	Z
D10 L	AJ1	J2	X
D11 L	AK2	J2	DD
D12 L	AK1	J2	BB
D13 L	AL2	J2	JJ
D14 L	AL1	J2	FF
D15 L	AM2	J2	NN

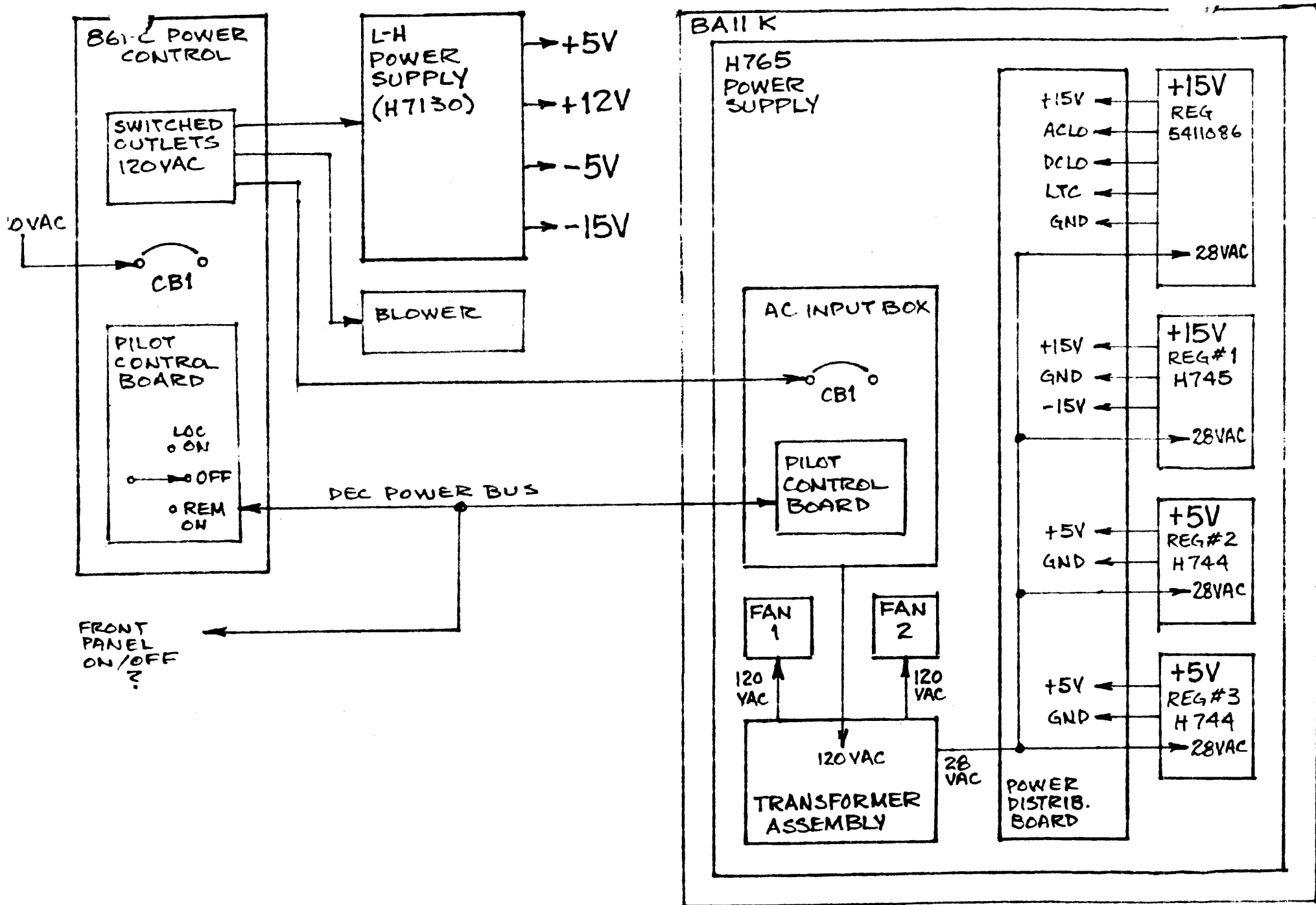
GND	AB2	THE FOLLOWING PINS		
GND	AC2	ON J1 J2 J3 ARE		
GND	AN1	GROUNDED		
GND	AP1	A	SS	HH
GND	AR1	C	U	KK
GND	AS1	E	W	MM

URA-6

GND	AT1	H	Y	PP
GND	AV2	K	AA	SS
GND	BB2	M	CC	UU
GND	BC2	P	EE	

GND	BD1	J1	VV
GND	BE1	J2	TT
GND	BT1	J3	DD
GND	BV2	J3	RR
INIT L	AA1	J3	Z
INTR L	AB1	J3	FF
MSYN L	BV1	J3	LL
NPG H	AU1	J3	D
NPR L	AS2	J3	B
PA L	AM1	J2	LL
PB L	AN2	J2	RR
+5V	AA2	"	"
+5V	BA2	"	"
SACK L	AR2	J3	TT
DCLO L	BF2	J3	VV
SSYN L	BU1	J3	JJ

UBA-7



KS-10 Power Distribution
Simplified Block Diagram

NOT VERIFIED

PS-1

LH POWER SUPPLY (H7130?)

INPUT:

90-128 VAC or 180-256 VAC

47-63 Hz.

12A @ 120V normal maximum current

16A maximum

OUTPUT:

V	A	P-P Noise *
+ 5.0	0-25	50 mV
+ 5.0	0-70	50 mV
+ 12.0	0-10	120 mV
- 5.0	0-1	50 mV
- 15.0	0-2	150 mV

* 50 KHz < Noise < 30 MHz. Ripple and drift not to exceed tolerance limits. Tolerance for all outputs is $\pm 1.5\%$.

BACKUP:

The LH interfaces with a DEC 48V Battery Pack. 30 seconds of backup are provided for: +5.0V/25A; +12.0V/10A; and -5.0V/1A. Batteries will not supply power to the regulators with the unit off.

The following conditions cause battery disconnect from the regulators:

- Power Switch from the Console OFF
- AC Line not low.
- 30 second backup expired.
- Low battery voltage.
- Open circuit breaker.

POWER FAIL SIGNAL:

PS-2

Upon removal of AC input, Power Fail does not go true (0 Volts) for 14-16 msec. The output of the LH will remain in regulation for at least 4 msec

after Power Fail is asserted. Line voltage interrupt of less than 14 msec. will not cause Power Fail to be asserted.

OVERLOAD:

Overload ($>110\%$; $<140\%$ of rated output load) is current limited to within 10% of rated current for a short circuit condition. Removal of the fault condition causes recovery from overload.

OVERVOLTAGE:

Overvoltage ($>110\%$; $<120\%$ of nominal output) is crowbarred. Crowbar reacts within 50 μ sec. to minimum 2 μ sec overvoltage pulse. Recovery is effected by manually cycling OFF then ON.

OVER TEMPERATURE:

Thermal shutdown is provided.

OUTPUT SEQUENCING:

-5.0V must exceed magnitude of 4.0V before +12.0V turns on.

+12.0V must drop to less than 1.0V before the magnitude of the -5.0V is less than 2.0V during turn off.

If -5.0V is short circuited the +12.0V ~~is~~ is crowbarred.

DEVICE ADDRESSES AND VECTOR ASSIGNMENTS FOR KS10

DEVICE	#	ADDRESS	VECTOR	UNIT	BR LEVEL	JUMPERS/SWITCHES
RH11	1	776700	254	1	5	
RH11	2	772440	224	3	5	
LP20	1	775400	750	3	4	
LP20	2	775420	754	3	4	
DZ11	1	760010	340	3	5	A/1 ON V/4,5,6,OFF
	2	760020	350	DITTO	DITTO	A/2 ON V/2,4,5,6 OFF
	3	760030	360	FOR THE	REST	A/1,2 ON V/3,4,5,6 OFF
	4	760040	370			A/3 ON V/2,3,4,5,6 OFF
		+10 ETC	+10 ETC			

KMC11	1	760540	540	A/3,4,6 OFF	V/1,2,5 OFF
	2	760550	550	A/1,3,4,6 OFF	V/2,5 OFF
		+10 ETC	+10 ETC		

NOTE: LOWEST ACCEPTABLE CS REV D.
 NOTE: EACH BACKPLANE SLOT CONTAINING AN M8204 MUST HAVE WIRE CA1 TO CB1 REMOVED.
 SHOULD THE MODULE BE REMOVED FOR ANY REASON THE WIRE SHOULD BE REINSTALLED TO
 INSURE THAT THE NPG SIGNAL WILL BE PASSED ALONG THE UNIBUS.

DUP11	1	760300	570	A/4,5 OFF	V/2 OFF
	2	760310	600	A/1,4,5 OFF	V/1,2 ON
		+10 ETC	+10 ETC		

NOTE: LOWEST ACCEPTABLE CS REV F. STANDARD FACTORY SET JUMPERS (FOR BELL 201
 COMPATABILITY) ARE AS FOLLOWS: W1 IN W5 OUT
 W2 OUT W6 IN
 W3 IN W7 IN
 W4 IN

EXTERNAL REGISTERS

TO ACCESS AN EXTERNAL REGISTER THE EXTERNAL IO INSTRUCTIONS GENERATE AN EXTENDED 30 BIT ADDRESS WHICH IS USED AS AN IO ADDRESS. BITS 14-17 ARE IO CONTROLLER NUMBERS AND BITS 18-35 ARE USED AS THE REGISTER ADDRESS. AN EXTENDED ADDRESS IS REQUIRED TO ADDRESS CONTROLLERS OTHER THAN ZERO, I.E. THE PROGRAMMER MUST USE INDEXING OR INDIRECTION TO YIELD THE DESIRED EFFECTIVE ADDRESS. FOR EXAMPLE, TO WRITE THE FIRST CONTROL STATUS REGISTER IN THE RH11 WITH A WRITE FUNCTION CODE OF 61, ASSUMING THE RH11 IS ON THE FIRST IO CONTROLLER, THE PROGRAMMER COULD EITHER

1/WRTIO AC,0(XR) WHERE XR(INDEX REGISTER)CONTAINS 1,,776700

AND AC CONTAINS 61 OR

2/WRTIO AC,0DRPCS1 WHERE THE LOCATION DRPCS1 CONTAINS 1,776700

AND AC CONTAINS 61 OR

3/WRTIO AC,776700(XR) WHERE XR CONTAINS THE CONT # IN BITS 14-17

AN ADDRESSING SCHEME SUCH AS THIS COULD BE USEFUL FOR USING COMMON CODE AMONG DIFFERENT IO CONTROLLERS.

BYTE INSTRUCTIONS WILL TRANSFER ONLY 8 BITS OF DATA FROM THE LSB'S OF AC. CURRENTLY THERE ARE FOUR SLOTS ALLOCATED FOR IO CONTROLLERS, THESE ARE NOW UNIBUS ADAPTER MODULES. THE CONTROLLER NUMBERS ARE HARDWIRED ON THE BACKPLANE.

SLOT	I/O CONTROLLER#
19	1
17	2
16	3
15	4

EXTERNAL REGISTERS BIT FORMATS

UBA PAGING RAM IO ADDRESS 763000-763077

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	

WRTIO(713)
AC,7630XX

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
MAINT EN16 EN36 INVALID <---NOT USED--> <---PAGE BITS--- LOADR 16 THRU 26--- >																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	

RDIO(712)
AC,7630XX

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	
RAM BITS 10 ADR 16-26--->>>---NOT USED--->																	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																	

RAM P
EN 16
VALID

RAM PARITY BIT
DISABLE UPPER 2 BITS ON UNIBUS TRANSFERS
PAGE IS VALID

MAINT
EN 36
RDPAK

MAINTENANCE PAUSE FOR EVEN/ODD MEM WRITES
ENABLE 36 BIT FAST MODE TRANSFERS
PAGING RAM PARITY COMPUTED ON READ

MISC-2

UBA STATUS REGISTER IO ADDRESS 763100

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
C TIMIC RADIC PAR C NEDIC NOT USED ON WRT ID XFR INIT* <---PIH--->* <---PIL--->*																	

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
* TIMI RADIC PAR* NEDIC INT HINT LIPWR L* IDXFR ! <---PIH--->* <---PIL--->*																	

18	R/W	TIM	UNIBUS ARBITRATOR TIMEOUT=NO SACK IN 5 MICRO SEC=CP ADDRESSED NON EXISTENT DEV REG
19	R/W	BAD	OR DEV ADDRESSED NON EXISTENT MEMORY 1 CLEARS THE BIT
20	R/W	PAR	BAD MEM DATA ON NPR TRANSFER=MASTER WILL TIME OUT ON BAD MEM DATA IF 28 SET=WRT CLEARS THE BIT
21	R/W	NED	KSIO BUS PARITY ERROR=WRITE CLEARS THE BIT
24	RONLY	INT H	CPU ADDRESSED NON EXISTENT DEVICE=WRT CLEARS THE BIT
25	RONLY	INT I	INTERRUPT REQUEST ON BR6 OR BR7
26	RONLY	PWR L	INTERRUPT REQUEST ON BR5 OR BR4
28	R/W	DXFR	AC LOW OR DC LOW=CLEARED ON REGISTER WRITE
29	RONLY	DNIT	DISABLE TRANSFER ON UNCORRECTABLE DATA
30-32	R/W	PIH	ISSUE UNIBUS INIT
33-35	R/W	PIL	PI LEVEL OF BR6,BR7
			PI LEVEL OF BR4,BR5

UBA MAINTENANCE REG IO ADDRESS 763101

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
* <---NOT USED---> CNPRG EWRAP *																	

WONLY BIT 34 CNPRG CHANGE NPR REG
WONLY BIT 35 EWRAP ENABLE NPR WRAP AROUND

MEMORY STATUS REGISTER IO ADDRESS 100000

THE MEMORY WILL HOLD THE ERROR ADDRESS, THE ECC CODE AND THE TYPE OF ERROR FOR THE FIRST ERROR DETECTED. BIT00, THE ERROR HOLD BIT, MUST BE CLEARED TO ALLOW THE MEMORY TO CONTINUE AND THE CAPTURE THE NEXT ERROR IN THE STATUS REGISTER.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
* HOLD BAD DIR ERR* P ERR ECC CON CP * C40 C20 C10 * C04 C02 C01 * PFAIL 1 ADR14*ADR15 ADR16 ADR17 *																	
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
* ADR18 ADR19 ADR20*ADR21 ADR22 ADR23*ADR24 ADR25 ADR26*ADR27 ADR28 ADR29*ADR30 ADR31 ADR32*ADR33 ADR34 ADR35 *																	

00 HOLD MMC5 ERR HOLD SET ON ERROR CONDITION DETECTED BY CONTROLLER-CLEARED WHEN WRITTEN
01 BAD D MMC4 UNCOR ERR HOLD SET SAYS ERROR IN REG WAS UNCORRECTABLE
02 R ERR MMC9 REF ERR REFRESH ERR SET 5.5 USEC AFTER COM/ADR IF NO DATA RECEIVED, CLEARS ANY CYCLE IN PROGRESS AND REFRESHES THE MOS ARRAYS, CLEARED WHEN WRITTEN OR MR.

03	P ERR	MMC7 PARITY ERR ASSERTED	MMC RECIEVED A PARITY ERR ON THE KS10 BUS, SET BY WRITING 1 TO BIT 3;CLEARED BY WRITING 0 TO BIT 3 OR MR.
04	ECCON	MMC3 ECC ON	THE ERROR CORRECTION LOGIC IS ENABLED WHEN ON,WHEN OFF ERRORS WILL STILL BE DETECTED BUT NO CORRECTIONS MADE,TURNED ON BY MR OR IO WRITE TO BIT 35 OF THE STATUS REG
05	CP	MMC4 ERR CP	PARITY BIT OF THE CORRECTION CODE
06	C40	MMC4 ERR C40	CURR CODE BIT C40
07	C20	MMC4 ERR C20	CORRECTION CODE BIT C20
08	C10	MMC4 ERR C10	CORR CODE BIT C10
09	C04	MMC4 ERR C04	CORR CODE C04
10	C02	MMC4 ERR C02	CORR CODE BIT C02
11	C01	MMC4 ERR C01	CORR CODE BIT C01
12	P FAIL	MMC5 POWER FAILED	SET IF MEM HAS LOST POWER OR BATTERY BACKUP FAILED,CLEARED WHEN WRITTEN.
13	1	UNUSED	ALWAYS ASSERTED
14-35	ADPXX	MMCR ERR ADR14-35	BITS 14-35 OF THE ERROR ADDRESS REGISTER.

ADDRESS BITS

14-16	SELECT CONTROLLER
17-19	SELECT ARRAY BOARD
20-21	SELECTS ONE OF FOUR WORDS ON THE SELECTED BOARD
22-28	7 ROW ADDRESS LINES
29-35	7 COLUMN ADDRESS LINES

RH11 RP44/5/6 REGISTER BIT FORMAT		UNIBUS																	
MRUS:DCI	IRH11	ADR																NOTE	
ADR	PRINT	NAME																	
10			120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	
MBUS			C15	C14	C13	C12	C11	C10	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00	
00	IRG3RG6	IRPCS	ISC	IRE	MCPE	10	IDVA	ISEL	IA17	IA16	IRDY	IE	IF4	IF3	IF2	IF1	IF0	IGO	17767001W
--		IRPWC	WC15	WC14	WC13	WC12	WC11	WC10	WC09	WC08	WC07	WC06	WC05	WC04	WC03	WC02	WC01	WC00	17767021
--		IRPRA	RA15	RA14	RA13	RA12	RA11	RA10	RA09	RA08	RA07	RA06	RA05	RA04	RA03	RA02	RA01	RA00	17767041
05	ISS3SS6	IRPDA	10	10	10	ITA16	ITA08	ITA04	ITA02	ITA01	10	10	10	ISA16	ISA08	ISA04	ISA02	ISA01	17767061W
--		IRPCS2	IDIT	IWCE	IUPE	INED	INEM	IPGF	IXF	INDPF	IOP	IIR	ICLR	IPAT	IBAI	IU2	IU1	IU0	17767101
01*	IPG6	IRPDS	IATA	IERP	ITPT	IMOL	IMPL	ILST	IPGM	IDPR	IDRY	IVV	IDR1	IDL64	IGRV	IDIGR	IDF20	IDF5	17767121
													10	10	10	10	10	10	
02	IPG2	IRPEP1	IDCK	IUNS	IOP1	IDTE	IWLF	IAE	IAOE	IMCRC	IMCE	IECH	IWCF	IFER	IPAR	IRMR	IILP	IILF	17767141W
04	IDP	IRPAS	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767161W
07	IDP6	IRPLA	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767201
--		IRPDR	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR09	IDR08	IDR07	IDR06	IDR05	IDR04	IDB03	IDB02	IDB01	IDB00	17767221
03	IEC1RG3	IRPMR	10	10	10	10	10	IMCT	ISBD	IZDT	IDEN	IECCE	IMWR	IMRD	IMSCLK	IMIND	IMCLK	IDMD	17767241W
06	IEC8	IRPDT	INFA	ITAP	IMOH	10	IDRO	10	10	10	10	10	10	10	10	10	10	10	17767261
10	IEC8	IRPSU	ISN39	ISN34	ISN32	ISN31	ISN29	ISN24	ISN22	ISN21	ISN18	ISN14	ISN12	ISN11	ISN08	ISN04	ISN02	ISN01	17767301
11	IRG1	IRPOF	ISGCH	10	10	IFWT22	IECCI	IMCI	10	10	10	10	10	10	10	10	10	10	17767321W
12*	ISS1	IRPDC	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767341W
13*	ISS1	IRPCC	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767361
14*	IEC6	IRPER2	ICUNS	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767401W
			INC																
15*	IEC7	IRPER3	ICYL	ISKI	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767421W
16	IEC2	IRPEC1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767441
17	IEC1	IRPEC2	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	17767461

1. SHARED REGISTER C15-C13, C10-C06 IN RH11 *WHERE TWO ENTRIES TOP=RP04 BOTTOM=RP06 **READ/WRITE ---=RH11 REGISTER
FIRST STORED IN DRIVE

		RH11 RM03 REGISTER BIT FORMAT																UNIBUS1	
MBUS	RH11	PRINT																ADR	NOTE
ADR	NAME	LOC																	
10			120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	
MBUS1			C15	C14	C13	C12	C11	C10	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00	
00	RMCS1		ISC	TRE	MCPE	10	1DVA	IPSEL	IA17	IA16	IRDY	IE	IF4	IF3	IF2	IF1	IF0	IGO	17767001R/W
--	RMWC		WC15	WC14	WC13	WC12	WC11	WC10	WC09	WC08	WC07	WC06	WC05	WC04	WC03	WC02	WC01	WC00	17767021R/W
--	RMBA		BA15	BA14	BA13	BA12	BA11	BA10	BA09	BA08	BA07	BA06	BA05	BA04	BA03	BA02	BA01	BA00	17767041R/W
05	RMDA		10	10	10	10	10	ITA04	ITA02	ITA01	10	10	10	ISA16	ISA08	ISA04	ISA02	ISA01	17767061R/W
--	RMCS2		DLT	WCE	UPE	INED	INEM	IPGE	IMXF	MDPE	IOR	IIR	ICLR	IPAT	IBAI	IU2	IU1	IU0	17767101R/W
01	RMDS		ATA	ERR	PIP	MOL	WRL	LB1	IPGM	IDPR	IDRY	IVV	10	10	10	10	10	10M	17767121R
02	RMER1		ICK	UNS	IOPI	ITE	WLE	IAE	IAOE	HCRC	HCE	IECH	WCF	IFER	IPAR	IRMR	IIR	ILF	17767141R/W
04	RMAS		10	10	10	10	10	10	10	ATA7	ATA6	ATA5	ATA4	ATA3	ATA2	ATA1	ATA0		17767161R/W
07	RMLA		10	10	10	10	10	ISC16	ISC8	ISC4	ISC2	ISC1	10	10	10	10	10	10	17767201R
--	RMDB		DB15	DB14	DB13	DB12	DB11	DB10	DB09	DB08	DB07	DB06	DB05	DB04	DB03	DB02	DB01	DB00	17767221R
03	RMMR1		IOCC	IRLG	IRL	IREXC	ESRC	PLFS	ECRC	IPDA	IPHA	ICONT	IPS	IEFCC	IWD	ILS	ILS&T	IDMD	17767241R
			DBCLK	DREN	DEBL	IMSTOP	MCLK	MRD	MURDY	MOCYL	MSERR	MOF	MSCLK	INC	IMWP	IMIND	MSC	IDMD	17767241W
06	RMDT		10	10	IMOH	10	IDRQ	10	10	IDT8	IDT7	IDT6	IDT5	IDT4	IDT3	IDT2	IDT1	IDT0	17767261R
10	RMSN		ISR000	IS4000	IS2000	IS1000	IS800	IS400	IS200	IS100	IS80	IS40	IS20	IS10	ISN08	ISN04	ISN02	ISN01	17767301R
11	RMOF		10	10	10	IFMT16	ECI	ICI	10	10	IOFF	D10	10	10	10	10	10		17767321R/W
12	RMDC		10	10	10	10	10	10	10	DC512	DC256	DC128	DC64	DC16	DC08	DC04	DC02	DC01	17767341R/W
13	RMHR		10	10	10	10	10	10	10	01	01	01	01	01	01	01	01	01	17767361UNUSED
14	RMHR2		IREQA	IREQB	ITAG	ITEST	ICCTAG	ICHTAG	IBB9	IBB8	IBB7	IBB6	IBB5	IBB4	IBB3	IBB2	IBB1	IBB0	17767401R
15	RMER2		10	ISKI	IOPE	IIVC	ILSC	ILBC	10	10	DEVCK	10	10	10	IDPE	10	10	10	17767421R/W
16	RMEC1		10	10	10	IP4096	IP2048	IP1024	IP512	IP256	IP128	IP64	IP32	IP16	IP8	IP4	IP2	IP1	17767441R
17	RMEC2		10	10	10	10	10	IPAT11	IPAT10	IPAT9	IPAT8	IPAT7	IPAT6	IPAT5	IPAT4	IPAT3	IPAT2	IPAT1	17767461R
1,--	RMBAE		10	10	10	10	10	10	10	10	10	10	10	IA21	IA20	IA19	IA18	IA17	17767501R/W
--	RMCS3		IAPE	IDPEHI	IDPELO	WCEHI	WCELO	IDBL	10	10	10	IE	10	10	IPCK3	IPCK2	IPCK1	IPCK0	17767521R/W

SHARED REGISTER C15-C13,C10-C06 IN RH11 REST STORED IN DRIVE * --RH11 REGISTER

MISC-4A

RM03/RP06/RH11 REGISTER BITS IN PLAIN ENGLISH-REFER

30VU	30 VOLTS UNSAFE
35VF	35 VOLT FAILURE
A	UNIBUS ADDRESS EXTENTION BITS
ABS	ABNORMAL STOP
ACL	AC LOW
ACU	AC UNSAFE
AOE	ADDRESS OVERFLOW ERROR
APE	ADDRESS PARITY ERROR
ATA	ATTENTION ACTIVE
BA	BUS ADDRESS
BAI	UNIBUS ADDRESS INCREMENT INHIBIT
BBX	BUS IN LINES -FUNCTION DEPENDENT ON TAG LINES
BIT	BURST
BLC	BURST LOCATION COUNT
CC	CURRENT CYLINDER
CCTAG	CONTROL SELECT OR CYCLINDER SELECT TAG
CHTAG	CONTROL SELECT OR HEAD SELECT TAG
CLR	CONTROLLER CLEAR
CONT	CONTINUE
CSF	CURRENT SINK FAILURE
CSU	CURRENT SWITCH UNSAFE
DB	DATA BUFFER
DBCLK	DEBUG CLOCK
DBEN	DEBUG CLOCK ENABLE
DBL	DOUBLE WORD OPERATION
DC	DESIRED CYLINDER
DCK	DATA CHECK
DCL	DC LOW
DCU	DC UNSAFE
DE1	DIFFERENCE EQUALS ONE
DEBL	DIAGNOSTIC EBL
DEN	DATA ENVELOPE
DEVCK	DEVICE CHECK(DC OR HEAD SELECT FAULT)
DF20	DRIVE FORWARD 20 INCHES PER SEC
DF5	DRIVE FORWARD 5 INCHES PER SEC
DIGB	DRIVE TO INNER GUARD BAND
DL64	DIFFERENCE LESS THAN 64
DLT	DATA LATE
DMD	DIAGNOSTIC MODE
DPE	DATA PARITY ERROR RECIEVED
DPEHI	DATA PARITY ERROR-ODD WORD
DPELO	DATA PARITY ERROR EVEN WORD
DPR	DRIVE PRESENT
DRQ	DRIVE REQUEST REQUIRED
DRY	DRIVE READY
DT	DRIVE TYPE
DTE	DRIVE TIMING ERROR
DVA	DRIVE AVAILABLE
EBL	END OF BLOCK LEVEL
ECCE	ECC ENVELOPE
ECH	ECC HARD ERROR
ECI	ECC INHIBIT
ECRC	ENABLE CRC OUT
EECC	ENABLE ECC OUT
ERR	ERROR
ESRC	ENABLE SEARCH
EXT	EXTENTION
F	FUNCTION

MISC-5

FEN	FAILSAFE ENABLED
FER	FORMAT ERROR
FMT16	16 BITS/WORD FORMAT(MEANS THE SAME AS FMT22)
FMT22	FORMAT 22 SECTORS
GO	PERFORM OPERATION
GRV	GO REVERSE
HCE	HEADER COMPARE ERROR
HCI	HEADER COMPARE INHIBIT
HCRC	HEADER CRC ERROR
IAE	INVALID ADDRESS ERROR
IE	INTERUPT ENABLE
ILF	ILLEGAL FUNCTION
ILR	ILLEGAL REGISTER
IPCKX	INVERTED PARITY CHECK BITS
IR	INPUT READY
IVC	INVALID COMMAND
IXE	INDEX ERROR
LBC	LOSS OF BIT CHECK
LBT	LAST BLOCK TRANSFERRED
LS	LAST SECTOR
LS&T	LAST SECTOR AND TRACK
LSC	LOSS OF SYSTEM CLOCK
LST	LAST SECTOR TRANSFERRED
MCLK	MAINTENANCE CLOCK
MCPE	MASSBUS CONTROL BUS PARITY ERROR
MDF	MAINT DRIVE FAULT
MDPE	MASSBUS DATA BUS PARITY ERROR
MHS	MULTIPLE HEAD SELECT
MIND	MAINTENANCE INDEX
MOCYL	MAINT ON CYL
MOH	MOVING HEAD
MOL	MEDIUM ON LINE
MRD	MAINTENANCE READ
MS	MAINT SECTOR PULSE
MSC	MAINTENANCE SECTOR COMPARE
MSCLK	MAINTENANCE SECTOR CLK
MSE	MOTOR SEQUENCE ERROR
MSERR	MAINT SEEK ERR
MSTOD	MAINT SEARCH TIME OUT DISABLE
MURDY	MAINT UNIT READY
MWP	MAINT WRITE PROTECT
MWR	MAINTENANCE WRITE
MXF	MISSED TRANSFER
NBA	NOT BLOCK ADDRESSED
NC	NOT CONNECTED
NED	NONEXISTENT MEMORY
NEM	NON EXISTENT MEMORY
NHS	NO HEAD SELECTION
OCYL	OFF CYLINDER
OES	OFFSET INFORMATION
OFF D	OFFSET DIRECTION
OM	OFFSET MODE
OPE	OPERATOR PLUG ERROR
OPI	OPERATION INCOMPLETE
OR	OUTPUT READY
PAR	PARITY ERROR
PAT	PARITY TEST
PATXX	PATTERN
PDA	PACK DATA AREA
PGE	PROGRAM ERROR

MISC-5A

PGM	PROGRAMMABLE
PHA	PACK HEADER AREA
PIP	POSITIONING IN PROGRESS
PLFS	PACK LOOKING FOR SYNC
PLU	PLO UNSAFE
PS	PROM STRORE
PSEL	PORT SELECT
PSU	PACK SPEED UNSAFE
PXXXXX	ECC POSITION REGISTER BITS
R&G	RUN AND GO
RAW	READ AND WRITE
RDY	READY
REQA	REQUEST ON PORTA
REQB	REQUEST ON PORTB
REXC	RECIEVED EXCEPTION
RMR	REGISTER MODIFICATION REFUSED
SA	SECTOR ADDRESS
SBD	SYNC BYTE DETECTED
SC	SPECIAL CONDITION
SCG	SIGN CHANGE
SKI	SEEK INCOMPLETE
SN	SERIAL NUMBER
SXXX	SERIAL NUMBER
TA	TRACK ADDRESS
TAG	CONTROL SELECT TAG
TAP	TAPE DRIVE
TDF	TRANSITIONS DETECTOR FAILURE
TEST	COMMAND SEQUENCER IS BRANCHING
TRE	TRANSFER ERROR
TUF	TRANSITIONS UNSAFE
U	UNIT SELECT
UNS	UNSAFE
UPE	UNIBUS PARITY ERROR
UWR	ANY UNSAFE EXCEPT READ/WRITE
VUF	VELOCITY UNSAFE
VV	VALID VOLUME
WAO	WRITE AND OFFSET
WC	WORD COUNT
WCE	WRITE CHECK ERROR
WCEHI	WRITE CHECK ERROR=ODD WORD
WCELO	WRITE CHECK ERROR=EVEN WORD
WCF	WRITE CLOCK FAIL
WCU	WRITE CURRENT UNSAFE
WD	WRITE DATA
WLE	WRITE LOCK ERROR
WRL	WRITE LOCK
WRU	WRIRE READY UNSAFE
WSU	WRITE SELECT UNSAFE
ZDT	ZERO DETECT

MISC-5B

4H11 TM02 REGISTER BIT FORMAT

MBUS1	ADR	NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	NOTE	
10					120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135																			
MBUS1					C151	C141	C131	C121	C111	C101	C091	C081	C071	C061	C051	C041	C031	C021	C011	C001																			
00	CONTROL	1							IDVA*								IF5	IF4	IF3	IF2	IF1	IF0/G01																R/W	
01	STATUS			DATA	ERR	PIP	MOD	WRU	EOT	INC	OPP	DRY	SSC	IFES	ISDN	IDB	ITM	ROT	SLA																			R	
02	ERROR			CR/	UNS	OPI	DTE	NEF	CS/	IFCE	NSG	PEF/	INC/	IDPAR	IFMT	ICPAR	IPMF	IILR	IILF																			R	
				CR					ITM			LRC	IVPE																										
03	MAINTENANCE	1		IMDF8	IMDF7	IMDF6	IMDF5	IMDF4	IMDF3	IMDF2	IMDF1	IMDF0	ISWC	IMC	IMOP3	IMOP2	IMOP1	IMOP0	IMM																			R/W	
04	ATTENTION			INC	INC	INC	INC	INC	INC	INC	INC	INC	ATA7	ATA6	ATA5	ATA4	ATA3	ATA2	ATA1	ATA0																		R/W	
05	FRAME COUNT	1		IFC15	IFC14	IFC13	IFC12	IFC11	IFC10	IFC09	IFC08	IFC07	IFC06	IFC05	IFC04	IFC03	IFC02	IFC01	IFC00																				R/W
06	DRIVE TYPE	1		INSA	ITAP	IMDH	17 CH	IDPQ	ISPR	INC	-----DRIVE TYPE 00-0R-----																												R
07	CHECK CHARACTER			INC	INC	INC	INC	INC	INC	INC	ICRCP	ICRC7	ICRC6	ICRC5	ICRC4	ICRC3	ICRC2	ICRC1	ICRC0	INDZ																			R
07	CHECK CHARACTER			INC	INC	INC	INC	INC	INC	INC	IDT4	IDT7	IDT6	IDT5	IDT3	IDT9	IDT1	IDT8	IDT2	PE																			R
11	TAPE CONTROL	1		ICCL	ICFS	ICW	ICAO	INC	IDEN	IDEN	IDEN	IFMT	IFMT	IFMT	IFMT	IEV	ISS2	ISS1	ISS0																				R/W
							IDTE		12	11	10	ISEL3	ISEL2	ISEL1	ISEL0	IPAR																							
10	SERIAL NUMBER			ISN15	ISN14	ISN13	ISN12	ISN11	ISN10	ISN09	ISN08	ISN07	ISN06	ISN05	ISN04	ISN03	ISN02	ISN01	ISN00																				R

*HARDWIRED SET

MAGTAP REGISTER BITS IN PLAIN ENGLISH

NAME	BIT#	ORIGIN	DESCRIPTION
7CH	12		7 CHANNEL UNIT -NEGATED ON 9CH DRIVE OR POWER LOSS
ACCL	15		ACCELERATION-SET WHEN TRANSPORT NOT ACTIVELY READING OR WRITING
ATA	15	"	ATTENTION ACTIVE-ATTN L INTERFACE SIGNAL ASSERTED
ATAG-7			ATTENTION SUMMARY PSEUDO REGISTER(1 BIT PER TM02) BITS
BOT	01	SS	BEGINNING OF TAPE
COR/CPC	15		CORRECTABLE DATA ERROR(PE)OR CRCC READ DOES NOT MATCH COMPUTED CRCC(NRZI)
CPAR	03		CONTROL BUS PARITY-INCORRECT PARITY DETECTED
CRCC-7			CRCC BITS FOR LAST TRANSFER IN NRZI MODE
CS/ITM	12		CORRECTABLE SKEW(PE) OR ILLEGAL TAPE MARK(NRZI)
DEN0-2	00-10		SPECIFIES TAPE CHARACTER DENSITY
DPAR	05		DATA BUS PARITY ERROR
DRP	08	"	DRIVE PRESENT-HARDWIRED SET
DRQ	11		DRIVE REQUEST REQUIRED-NEGATED TO INDICATE SINGLE PORT UNIT
DRY	07	"	DRIVE READY-TM02 AND SELECTED TRANSPORT READY FOR COMMAND
DT0-7			DEAD TRACKS-TRACK MAY HAVE DROPPED ONE OR MORE BITS DURING THE TRANSFER
DT00-00	00-00		DRIVE TYPE 011=TU16 012=TU45
DTE	12		DRIVE TIMING ERROR-CLASS B
DVA	11		DRIVE AVAILABLE HARDWIRED SET
EAODTE	12		ENABLE ABORT ON DATA TRANSFER OPERATIONS-ENABLES ABORT ON ERROR REG BITS15,7,6,5
EOT	10	SS	END OF TAPE
ERR	14	"	COMPOSITE ERROR-ANY BIT IN ERROR REG IS SET
EVPAR	03		EVEN PARITY -EVEN PARITY READ/WRITTEN DURING NRZI OPERATION-IGNORED IN PE MODE
FC00-15			FRAME COUNT REGISTER BITS-COUNTS TAPE EVENTS
FCE	09		FRAME COUNT ERROR-CLASS A
FCS	13		FRAME COUNT STATUS-SET AT END OF WRITE IO FRAME COUNT REG-RESET ON FC REG OVERFLOW
FMT	04		FORMAT-DATA TRANSFER WITH INCORRECT FORMAT CODE DETECTED
FMT SEL0-3			FORMAT SELECT CODE--SEE BELOW
FX	0-5		FUNCTION CODE BITS
IDB	03	"	IDENTIFICATION BURST(PE) DETECTED-ASSERTED TILL NEXT TAPE MOTION COMMAND
ILF	00		ILLEGAL FUNCTION-CLASS B
ILR	01		ILLEGAL REGISTER-CLASS A
INC/VPE	06		INCOORECTABLE DATA ERROR OR VERTICAL PARITY ERROR-VPE ON NRZI READ OR DEAD TRACK ERRORS AND /OR SKEW OVERFLOW ON PE READ
MC	05		MAINTENANCE CLOCK
MDFA-R	07-15		MAINTENANCE DATA FIELD-BUFFERS DATA ON WRAP OPERATION-LPC AT END OF NRZI TRANSFERS
MM	00		MAINTENANCE MODE
MOH	13		NEGATED TO INDICATE NON -MOVING HEAD UNIT
MOL	12	SS	MEDIUM ON LINE-TAPE LOADED AND ON-LINE
MOP0-3	01-04		MAINT OPERATION CODE
NC			NOT CONNECTED-UNUSED
NEF	11		NONEXECUTABLE FUNCTION-CLASS B
NSA	15		NOT SECTOR ADDRESSED-ALWAYS ASSERTED
NSG	08		NONSTANDARD GAP TAPE CHAR DETECTED DURING FIRST HALF OF EOP
OPI	13		OPERATION INCOMPLETE-CLASS B
PEF/LRC	07		PE FORMAT ERROR OR NRZI CHECK CHAR ERROR
PES	05	SS	PHASE ENCODED STATUS-DRIVE CONFIGURED FOR PE OPERATION
PIP	13	"/SS	POSITIONING IN PROGRESS-ASSERTED BY TM02 ON SPACE OR DRIVE ON REWIND
RMR	02		REGISTER MODIFICATION REFUSED-REG WRITE ATTEMPTED DURING TAPE OPERATION
SDWN	04	SS	SETTLE DOWN--TAPE MOTION IS STOPPING
SIA	00	SS	SLAVE ATTENTION-SFL SLAVE HAS COME ON LINE
SN00-15			SERIAL NUMBER OF TRANSPORT-BCD REPRESENTATION(4 BITS PER DIGIT)
SPR	10		SLAVE PRESENT
SS0-2	00-02		SLAVE SELECT BITS-SPECIFIES UNIT # OF TRANSPORT TO BE USED
SRC	06	S	SLAVE STATUS CHANGE
SWC	06		SELECTED SLAVE CLOCK-WRT CLOCK SIGNAL GENERATED BY SEL SLAVE

TAP	14		ASSERTED TO INDICATE TAPE TRANSPORT
TCW	14		TAPF CONTROL WRITE-SET ON TC REG WRITE-RESET ON MOTION COMMAND
TM	02	v	TAPF MARK DETECTED
UNS	14		UNSAFE-CLASS B
WRL	11	SS	WRITE LOCK=WRITE LOCK RING INSTALLED

SS=SELECTED TRANSPORT
S=ANY TRANSPORT
H=TM02 LOGIC

FORMAT SELECT CODES

SYSTEM	CODE	DESC
PDP10	0000	CORE DUMP
PDP10	0001	7TRK-NOT USED
PDP10	0010	ASCII
PDP10	0011	COMPATIBLE
PDP11	1100	NORMAL
PDP11	1101	CORE DUMP
PDP11	1110	15 NORMAL

DRIVE COMMAND FUNCTION CODES

COMMAND CODE(OCTAL)	FIXED HEAD DSK	MOVING HEAD DSK	MAGNETIC TAPE
01	NOOP	NOOP	NOOP
03		UNLOAD	REWIND, OFF-LINE
05		SEEK	
07		RECALIBRATE	REWIND
11	DRIVE CLEAR	DRIVE CLEAR	DRIVE CLEAR
13		RELEASE	
15		OFFSET	
17		RETURN TO CENTERLINE	
21	READ IN PRESET	READIN PRESET	READIN PRESET
23		PACK ACKNOWLEDGE	
25			ERASE
27			WRITE FILE MARK
31	SEARCH	SEARCH	SPACE FORWARD
33			BACKSPACE
51	WRITE CHECK DATA	WRITE CHECK DATA	WRITE CHECK FORWARD
53		WRITE CHECK HEADER/DATA	
57			WRITE CHECK REVERSE
61	WRITE DATA	WRITE DATA	WRITE FORWARD
63		WRITE HEADER AND DATA	
71	READ DATA	READ DATA	READ FORWARD
73		READ HEADER AND DAT	
77			READ REVERSE

```

;TO GET A DZ TO INTERRUPT MAYBE???
DI 760020(CSR)/110
DI 760032(TDR)/105
; READ THE INTERRUPT VECTOR DO WRU CYCLE
EI 200050000000
;THEN IO READ TO GET THE VECTOR
EI 100010000000
;THE RIGHT MOST BITS SHOULD BE THE VECTOR

```

KS10 OPTIONS JUMPER AND SWITCH CONFIGURATIONS

----- LP20-LINE PRINTER CONTROLLER -----

M8566 CONTROL LOCATION ABCDEF02

JUMPER	STATE	FUNCTION
W1	IN(#1),OUT(#2)	ADR BIT 4
W2	IN	ADR BIT 5
W3	IN	ADR BIT 6
W4	IN	ADR BIT 7
W5	OUT	ADR BIT 8
W6	OUT	ADR BIT 9
W7	IN	ADR BIT 10
W8	OUT	ADR BIT 11
W9	OUT	ADR BIT 12

W10	OUT(#1),IN(#2)	VEC BIT 2
W11	IN	VEC BIT 3
W12	OUT	VEC BIT 4
W13	IN	VEC BIT 5
W14	IN	VEC BIT 6
W15	IN	VEC BIT 7
W16	IN	VEC BIT 8

W1-9 CORRESPOND TO BASE ADR 775400(UNIT #1) AND 775420(UNIT #2)
 W10-W16 CORRESPOND TO VECTOR 750(#1) AND 754 (#2)
 DIP SITE E6 CONTAINS PRIORITY PLUG BR4 54-08776

M8587 DATA PATHS

W1	OUT	INSTALL TO ENABLE PARITY
W2	IN	INSTALL FOR DAVFU

CABLE

CABLE IS BC06R,CONNECT FROM BERG CONNECTOR (J1) ON THE M8587 MODULE,R18 SIDE
 TOWARDS MODULE(RED LINE AWAY FROM HANDLE),TO CENTER SLOT OF RECEPTICLE HOUSING
 MOUNTED IN THE CONNECTOR TRAY(RED LINE TOWARDS THE CONNECTOR FASTENER,

MISC-14

2. The only interrupt function implemented for devices is the dispatch function (i.e., interrupt vector). Unlike the KL10-B, which dispatches to and executes the instruction in the EPT location specified by the vector address, the KS10 first references an EPT location determined by the UBA number ($EPT + 100 + \text{CONTROLLER \#}$). It then uses this word as an exec-virtual address of a table and executes the instruction at $TABLE + VECTOR/4$.
3. The KS10 implements two levels of PIA for I/O (Unibus) devices, one PIA can have a higher priority than the other. The PI level (1-7) assigned to Unibus devices interrupting on BR levels 7 and 6 is set by loading a high level PIA (bits 30-32 of UBA status register). The PI level (1-7) for devices interrupting on BR levels 5 and 4 is set by loading a low level PIA (bits 33-35 of UBA status register).

Table 4-2 lists the hard-wired interrupt vectors and BR levels for the various KS10 I/O (Unibus) devices in a fully configured system. It also indicates which PIA, high or low level, is associated with each device.

2. The only interrupt function implemented for devices is the dispatch function (i.e., interrupt vector). Unlike the KL10-B, which dispatches to and executes the instruction in the EPT location specified by the vector address, the KS10 first references an EPT location determined by the UBA number ($EPT + 100 + \text{CONTROLLER \#}$). It then uses this word as an exec-virtual address of a table and executes the instruction at $TABLE + VECTOR/4$.
3. The KS10 implements two levels of PIA for I/O (Unibus) devices, one PIA can have a higher priority than the other. The PI level (1-7) assigned to Unibus devices interrupting on BR levels 7 and 6 is set by loading a high level PIA (bits 30-32 of UBA status register). The PT level (1-7) for devices interrupting on BR levels 5 and 4 is set by loading a low level PIA (bits 33-35 of UBA status register).

Table 4-2 lists the hard-wired interrupt vectors and BR levels for the various KS10 I/O (Unibus) devices in a fully configured system. It also indicates which PIA, high or low level, is associated with each device.