

EY-D5283-SG-002

DIGITAL  
**VAX 11/750**  
MAGIC BOOK

Rev. 8 JAN-1-1983

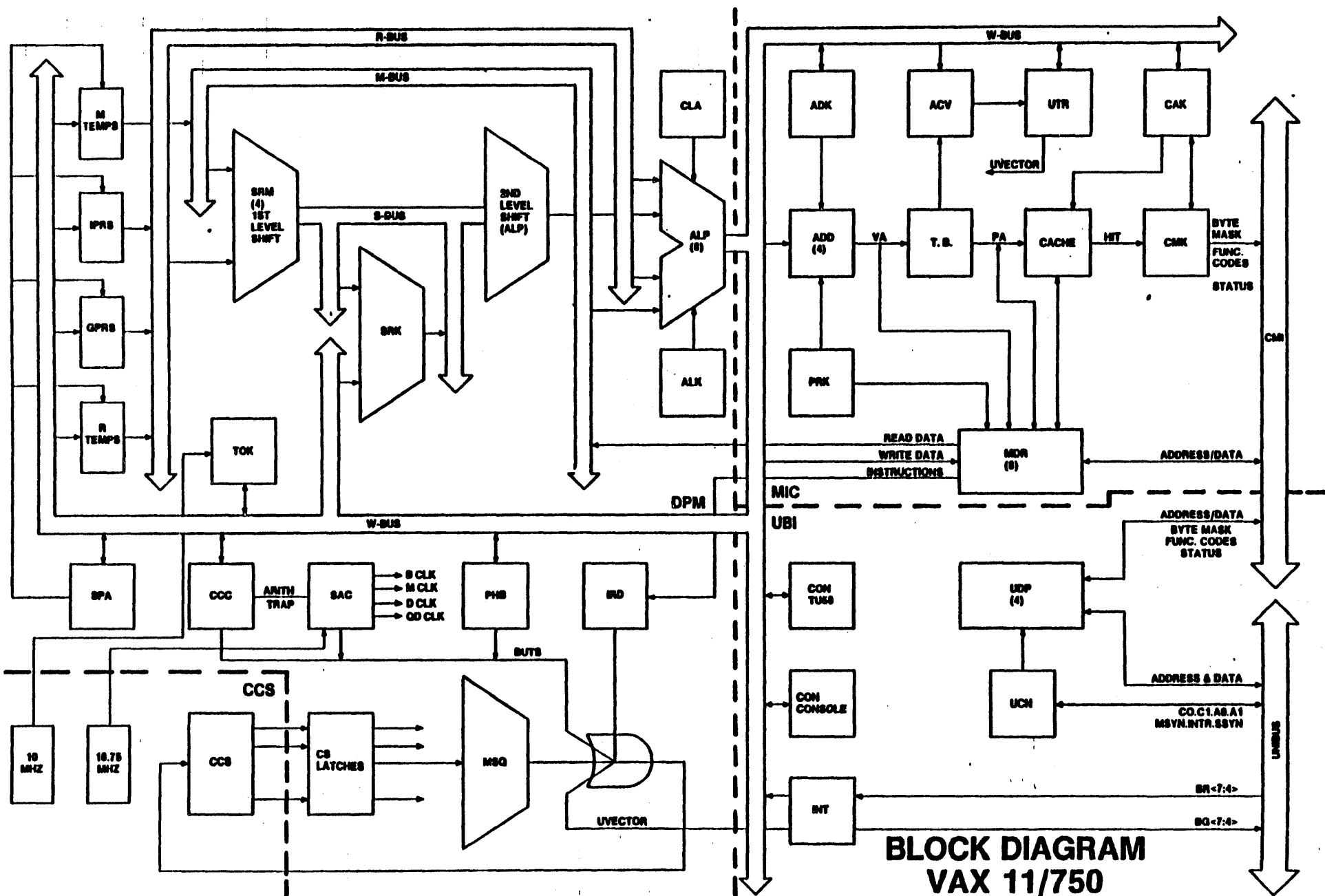
VAX 11/750 MAGIC BOOK  
FOR INTERNAL USE ONLY

THIS TEXT WAS CREATED IN AN ATTEMPT TO CENTRALIZE THE ESSENTIAL INFORMATION REQUIRED TO MAINTAIN THE 11/750 AT A BRANCH LEVEL. CONTAINED IN THIS TEXT IS INFORMATION CONCERNING BOARD LOCATIONS, GATE ARRAYS LOCATED ON EACH BOARD, BASIC FUNCTIONS OF THE CHIPS, PART NUMBERS, AND MISC. OTHER INFORMATION YOU MIGHT FIND USEFUL WHEN INSTALLING OR MAINTAINING THE VAX 11/750 SYSTEMS.

THE INTENT OF THIS GUIDE IS NOT TO BECOME A STEP BY STEP TROUBLESHOOTING TOOL, ONLY TO MAKE SOME USEFUL INFORMATION AVAILABLE IN A SINGLE PACKAGE.

INDEX OF MAIN TOPICS

CONTENTS:		PAGE #	
MAINTENANCE PHILOSOPHY		2	
SIMPLIFIED BLOCK DIAGRAM		3	
MICROCODE BIT FIELD CHART		4	
MAJOR BUS DEFINITIONS		5	
BACKPLANE PIN COUNTS		6	
POWER SYSTEM BLOCK DIAGRAM		7	
PHYSICAL ADDRESS ORGANIZATION		8	
VMS SHUTDOWN PROCEEDURE		9	
FLOATING POINT ACCELERATOR	FPA L0001	10	SLOT 1
DATA PATHS MODULE	DPM L0002	19	SLOT 2
MEMORY INTERCONNECT MODULE	MIC L0003	37	SLOT 3
UNIBUS INTERCONNECT MODULE	UBI L0004	49	SLOT 4
SECOND UNIBUS ADAPTER MODULE	SUB L0010	57	SLOT 7,8,9
CPU/WRITEABLE CONTROL STORE	CCS/WCS L0005	69	SLOT 5
REMOTE DIAGNOSTIC MODULE	RDM/MTM L0006	73	SLOT 6
OPTION SLOTS		76	SLOT 7,8,9
RH750 MASSBUS ADAPTER MODULE	MBA L0007	77	SLOT 7,8,9
CPU MEMORY CONTROLLER MODULE	CMC L0011/L0016	85	SLOT 10
UNIBUS EXERCISOR/TERMINATOR	UET M9313	91	
DIAGNOSTIC TRANSFER/NAMES/AUTOSIZER/ATTACHES		95	
BASIC CONSOLE AND COPY COMMANDS		104	
WRITEBOOT UTILITY		107	
REVCON REVISION CONTROL DOCUMENT		108	
BACKPLANE CABLES AND JUMPERS		130	
SYE ERROR LOGGER		132	
SDA SYSTEM DUMP ANALYZER		135	
BOOTSTRAP PROCESS DESCRIPTION		137	
INSTRUCTION DECODE PROCESS		142	
MACHINE AND BUGCHECKS		152	
UNIBUS ADDRESS TRANSLATION WORKSHEET		171	
EDT EDITOR KEYPAD DIAGRAMS		172	



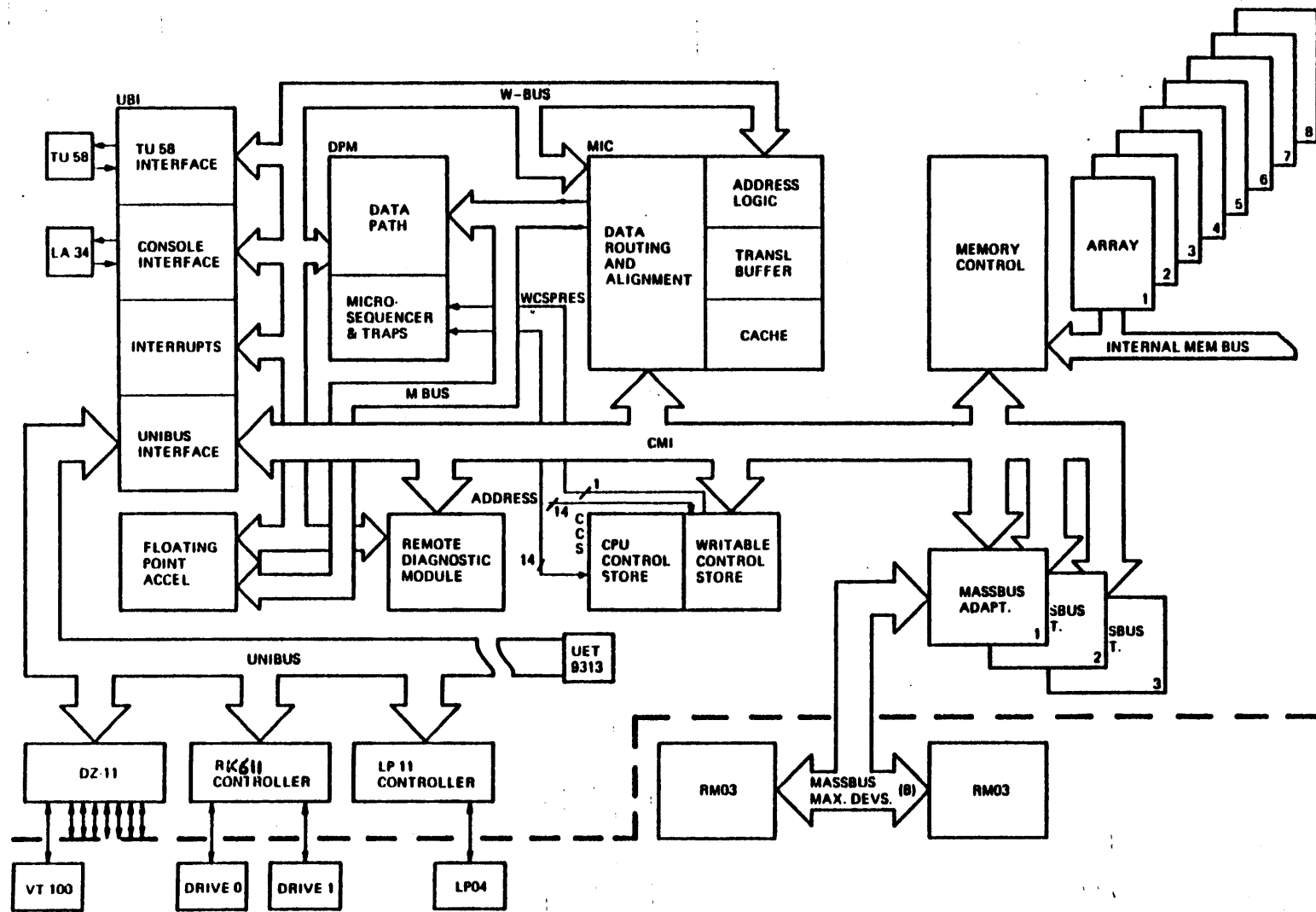
**BLOCK DIAGRAM  
VAX 11/750  
(GATE ARRAY LEVEL)**

VAX 11/750 MAINTENANCE PHILOSOPHY  
-----

- o THE CUSTOMER IS REQUIRED TO PROVIDE A VOICE GRADE TELEPHONE LINE AND CONNECTOR FOR DDC (DIGITAL DIAGNOSTIC CENTER) COMMUNICATION. (THIS REQUIREMENT IS INCLUDED WITH POWER AND ENVIRONMENTAL REQUIREMENTS IN THE VAX 11/750 SITE PREPARATION GUIDE. P/N EK-CORP-SP-003 ) (REV. IS SUBJECT TO CHANGE)
- o THE RDM OPTION WILL BE INSTALLED IN THE BACKPLANE OF ALL VAX 11/750 SYSTEMS DIGITAL INSTALLS, TO PROVE THE VALUE OF RD TO THE CUSTOMER DURING THE WARRANTY PERIOD. IT WILL BE LEFT IN THE BACKPLANE FOR ALL CUSTOMERS WITH THE STANDARD RD MAINTENANCE CONTRACT.

BASIC FLOW:

- o THE CUSTOMER CALLS THE DDC "TOLL FREE NUMBER" WHEN THERE IS A PROBLEM.  
  
(NOTE: NUMBERS ARE SUBJECT TO CHANGE)  
1-800-525-6570 FOR DDC CONNECTION  
1-303-599-4000 FOR ENGINEER ASSISTANCE (NOT TOLL FREE)  
1-303-593-7890 U.S.F.S. LIBRARY D.E.C. EMPLOYEES ONLY
- o MAIL STOP CX/DDC COLORADO SPRINGS, COLORADO
- o THE DDC PERFORMS REMOTE SUBSYSTEM ISOLATION.
- o THE DDC IDENTIFIES THE FAILING OPTION TO THE BRANCH OFFICE.
- o THE BRANCH OFFICE SEND THE RIGHT ENGINEER WITH THE RIGHT PARTS TO FIX THE PROBLEM.
- o FOR CPU PROBLEMS:  
  
THE ENGINEER TAKES THE CPU SPARES AND RDM TOOL TO THE SITE.  
  
THE ENGINEER RUNS THE TU58 MICRODIAGNOSTIC CASSETTE TAPES.  
  
FOR CUSTOMERS WITH NON-RD CONTRACTS, THE ENGINEER INSTALLS THE RDM TOOL INTO THE VAX 11/750 BACKPLANE, AND REMOVES IT WHEN HE/SHE COMPLETES THE WORK.
- o ON CPU LOGIC MODULES, FAULTS ARE ISOLATED TO A SPECIFIC MODULE AND SIMULTANEOUSLY TO A STRING OF CHIPS (AVERAGE OF TWO GATE ARRAYS).
- o THE ENGINEER PERFORMS COMPONENT LEVEL REPLACEMENT (CLR) BY REPLACING THE INDICATED GATE APRAYS.
- o THE FIX SHOULD THEN BE VERIFIED WITH THE DDC CENTER TO ASSIST THEM WITH BUILDING A CASE HISTORY OF FAILURES FOR THE 11/750. THIS IS IMPORTANT!!!
- o WHEN CLR DOES NOT CORRECT THE FAULT ON CPU LOGIC MODULES, AND ALL OTHER CPU FAILURES, THE FAILING MODULE OR ASSEMBLY IS REPLACED.



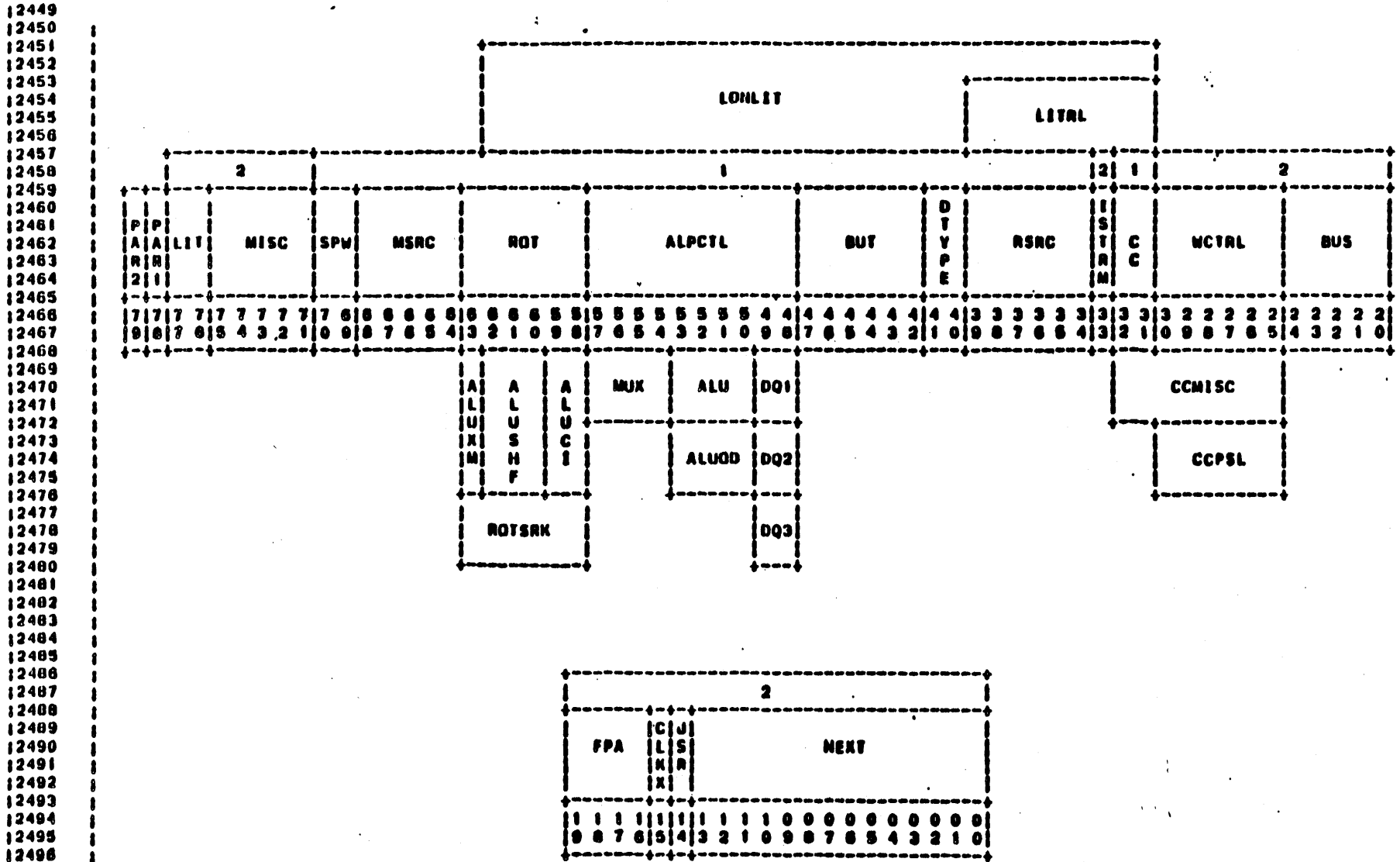
3

VAX-11/750 Simplified System Block Diagram

CM1062.MCR [130,134]  
 DEFIN.MIC [130,134]

MICRO2 IF(12) 19-Jan-81 15:27:55 CLOXX Rev 6.00, Clock rate = 160ns  
 Comet Micro Word Chart

12448 .TOC " Comet Micro Word Chart"



MAJOR BUS DEFINITIONS  
-----

THE MAJOR COMPONENTS OF THE PROCESSOR ARE INTERCONNECTED VIA TWO 32 BIT "LOW TRUE" BUSES CALLED THE MEMORY BUS (MBUS) AND THE WRITE BUS (WBUS).

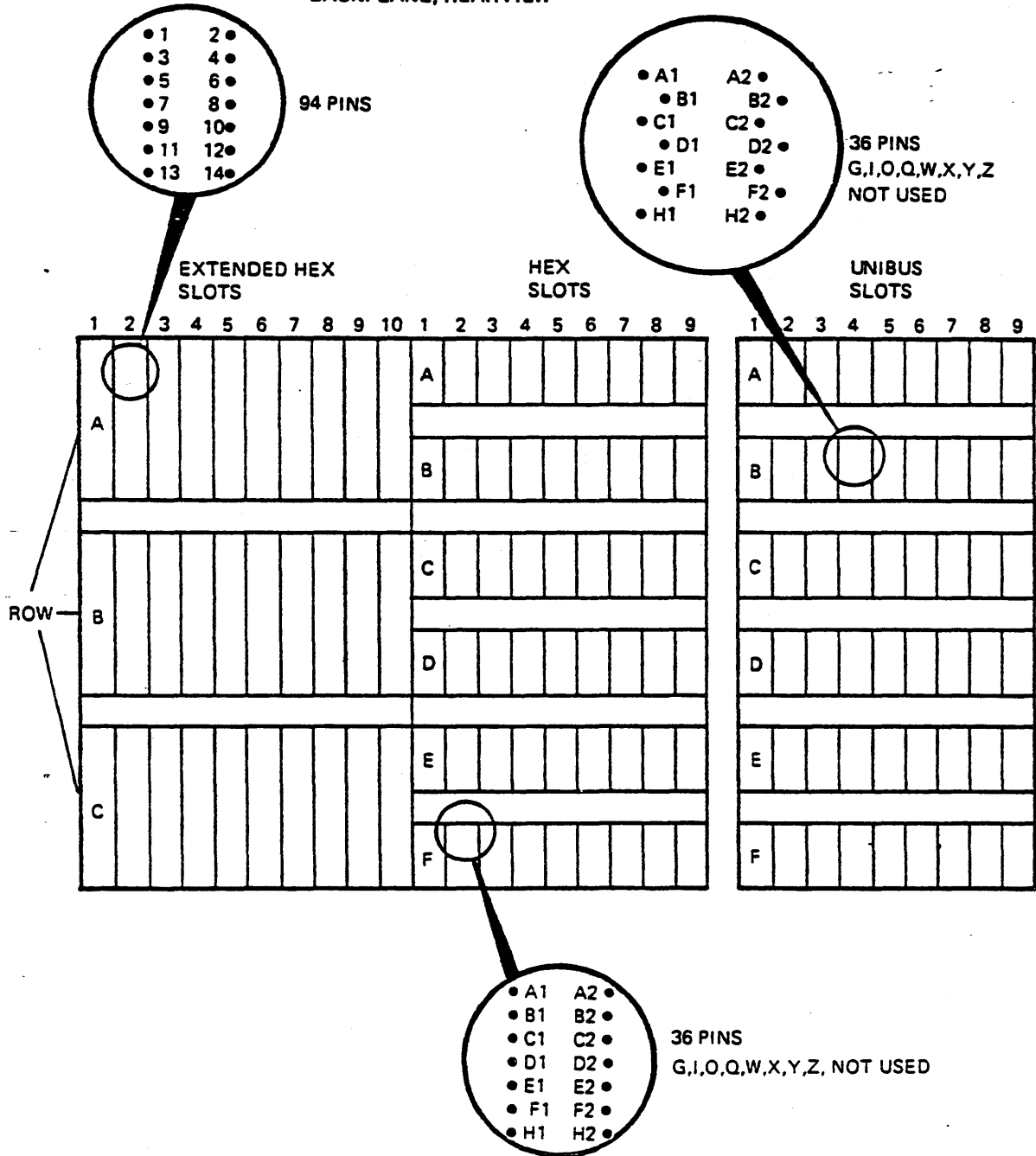
**MBUS:** THE MEMORY BUS IS PRIMARILY USED WHEN SOURCING PROGRAM INSTRUCTION OPERAND DATA FROM MEMORY, THROUGH THE MIC MODULE AND TO THE DPM FOR PROCESSING. IT MAY ALSO BE UTILIZED WHEN THE OPTIONAL FLOATING POINT ACCELERATOR REQUIRES OPERAND DATA FROM THE MIC MODULE OR MEMORY. THE MBUS IS NORMALLY SOURCED FROM THE MIC MODULE BUT IT CAN ALSO BE LOADED BY THE MTEMP REGISTERS ON THE DPM MODULE. CONTROL OF THE MBUS IS ACCOMPLISHED BY MICROCODE FIELDS AND CANNOT BE DIRECTLY ACCESSED BY THE CONSOLE TERMINAL.

**WBUS:** THE WRITE BUS IS THE BASIC INTERCONNECTION BETWEEN FOUR OF THE MAJOR CPU MODULES (DPM, MIC, UBI, FPA). THE WRITE BUS ACTIVITY IS CONTROLLED VIA MICROCODE FIELDS AND CAN BE UTILIZED BY MOST COMPONENTS INTERNAL TO THE CPU KERNEL. THE WBUS LIKE THE MBUS CANNOT BE DIRECTLY ACCESSED BY THE CONSOLE TERMINAL.

**CMI:** THE CPU MEMORY INTERCONNECT BUS IS THE MAJOR CENTRAL BUS. IT IS A TRI-STATE BUS (SOME SIGNALS ARE LOW TRUE AND OTHERS ARE HI TRUE) WHICH PROVIDES THE HIGH SPEED TRANSFER OF DATA BETWEEN CPU, MEMORY, AND DEVICE ADAPTERS (I.E. RH750, DW750, FP750, DR750, CI750 ETC.).

**NOTE:** INDIVIDUAL MODULES AND GATE-ARRAYS MAY HAVE THEIR OWN INTERNAL BUS STRUCTURES BUT THEY WILL BE DEALT WITH AS WE ENCOUNTER THEM IN THIS TEXT.

BACKPLANE, REARVIEW





POWER SUPPLY PART NUMBERS

OLD PART #'S

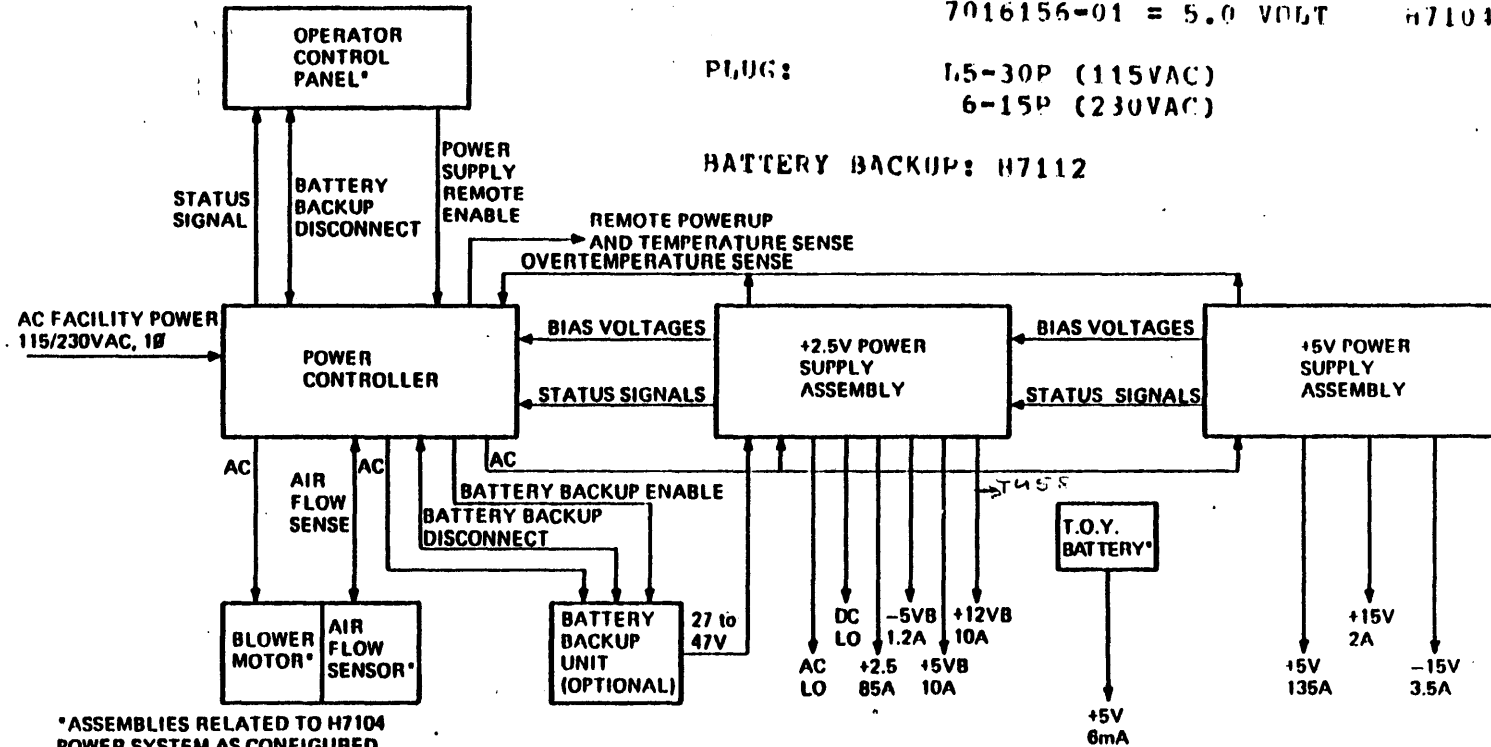
NEW PART #'S

CONTROLLERS: 7015929-00 = 115 VOLT  
7015929-01 = 230 VOLT

SUPPLIES: 7016157-01 = 2.5 VOLT H7104-C  
7016156-01 = 5.0 VOLT H7104-D

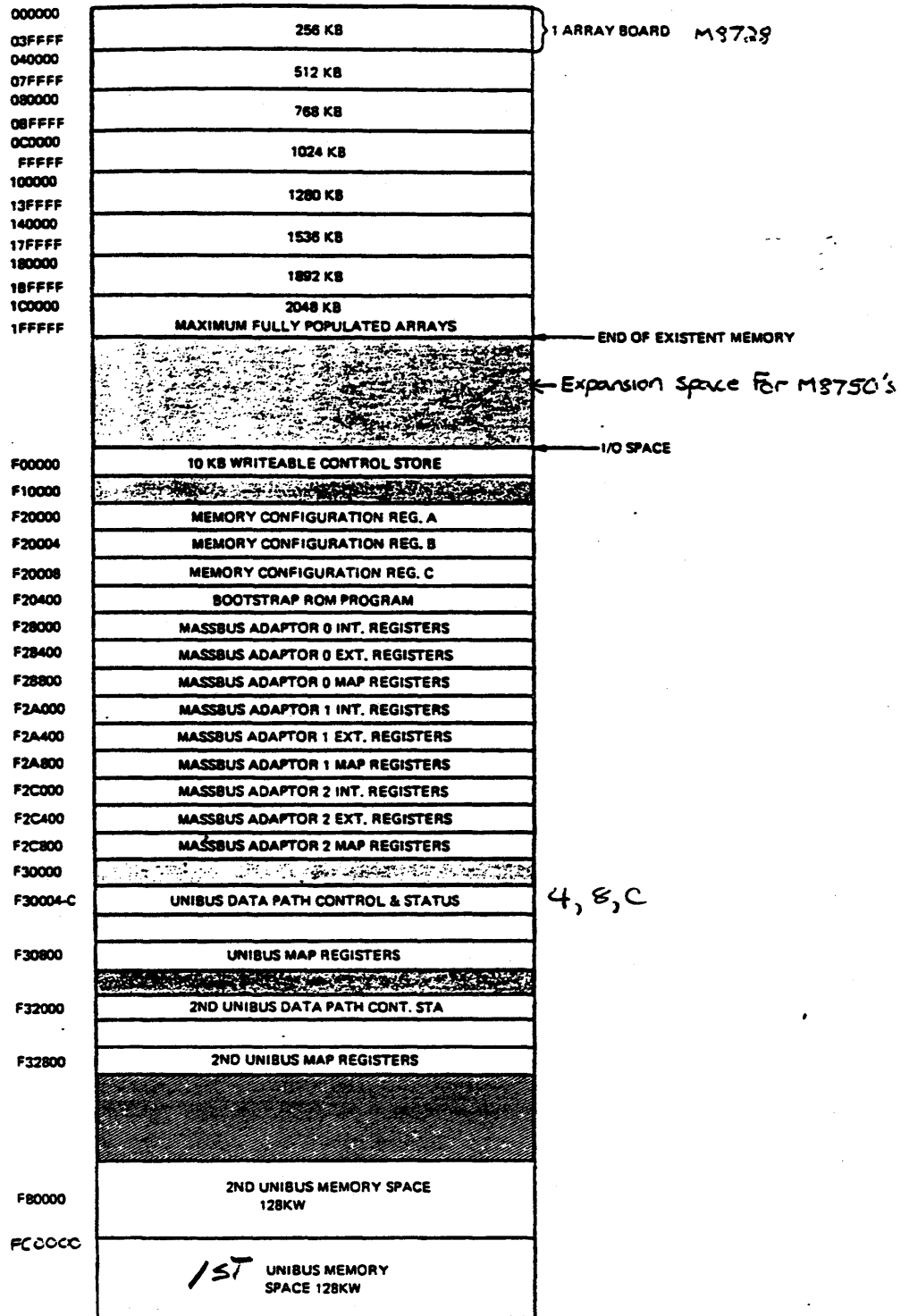
PLUG: 15-30P (115VAC)  
6-15P (230VAC)

BATTERY BACKUP: H7112



\*ASSEMBLIES RELATED TO H7104 POWER SYSTEM AS CONFIGURED FOR OPERATION WITH THE VAX-11/750

H7104 Power System Block Diagram



VAX-11/750 Physical Memory Organization

## VMS SHUTDOWN PROCEDURE

TO BRING THE VAX/VMS OPERATING SYSTEM DOWN, ONE MUST HAVE THE PROPER PRIVILEGES. THESE CAN BE HAD BY LOGGING INTO THE SYSTEM MANAGERS ACCOUNT. THE NORMAL FIELD SERVICE ACCOUNT MAY NOT HAVE THE PRIVILEGES TO BRING THE SYSTEM DOWN.

SO LOGOUT FROM THE ACCOUNT YOU ARE IN, IF YOUR IN, AND LOG INTO THE SYSTEM MANAGERS ACCOUNT AS SHOWN BELOW (UNDERLINED).

(OF COURSE IN THE FIELD YOU PROBABLY WILL NOT HAVE THE PASSWORD)

USERNAME:SYSTEM  
-----

PASSWORD:MANAGER  
-----

NOTE THE PASSWORD IS NOT DISPLAYED.

AFTER YOU HAVE THE "\$" PROMPT, THEN TYPE THE UNDERLINED RESPONSES.

Welcome to VAX/VMS Version VX.X

\$ @SYS\$SYSTEM:SHUTDOWN OR \$ @[SYSEXE]SHUTDOWN  
-----

System shutdown command procedure.

23-MAR-1980 09:35:23

How many minutes until shutdown?: 10 (or whatever)  
--

Reason?: PM (or <CR> if no message is desired)  
--

Do you want to spin down the disks?: YES (or <CR> if not)  
---

Expected uptime? (<CR> if not known):

Enable automatic reboot?:

YOU HAVE NOW STARTED THE SHUTDOWN PROCEDURE. YOU HAVE GIVEN IT TEN MINUTES TO DO THIS, ALSO GIVEN THE REASON AS SYSTEM PM, AND TOLD IT THAT YOU WANTED TO SPIN DOWN THE USER PACKS (NOT THE SYSTEM PACK). THE SYSTEM WILL SEND OUT A WARNING AT PREDETERMINED TIMES. IT WILL STOP ALL QUEUES, LOG EVERYONE OUT AND FINALLY COME UP WITH THE FOLLOWING MESSAGE:

SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM

NOW YOU CAN TYPE A CONTROL "P" TO GET BACK TO CONSOLE COMMAND LANGUAGE MODE WITH THE PROMPT ">>>".

10

**L0001 FPA**

## FP750

The Floating Point Accelerator (FPA) -  
An optional high-speed processor extension  
to the Vax-11/750 CPU. FP750.

## A. Purpose:

1. To increase the speed at which the Vax-11/750 can execute certain floating point instructions.
  - a. Single-precision floating
  - b. Double-precision floating
  - c. Extended Modulus (EMOD)
  - d. Polynomial (POLY)
2. To enhance the execution of integer multiply instructions
3. It will not accelerate execution of grand (G) or huge (H) floating instructions.

## B. Characteristics:

1. Extended-Hex Module
2. 28 gate arrays
3. 64-bit fraction data path
4. No internal diagnostics
5. 80-bit micro-word
6. Operates independently of the CPU
  - a. Uses CPU data cache for data fetch
  - b. Uses the CPU Instruction buffer for instruction fetch
  - c. While FPA is executing, the CPU can;
    - 1) Calculate memory addresses
    - 2) Fetch data
    - 3) Prepare to transmit data

## 4) Store FPA results

7. Can operate on numbers from  $.29 \times 10$  to  $1.7 \times 10$

8. Can operate on signed integers from  $2^2$  to  $-1$ .

## C. Interfacing to the CPU.

1. M-Bus

2. W-Bus

3. Miscellaneous

a. Control store address lines from CCS.

b. Exceptions/Interrupts to UBI.

c. Clock signals from the DPM.

d. Others, to and from all of the above.

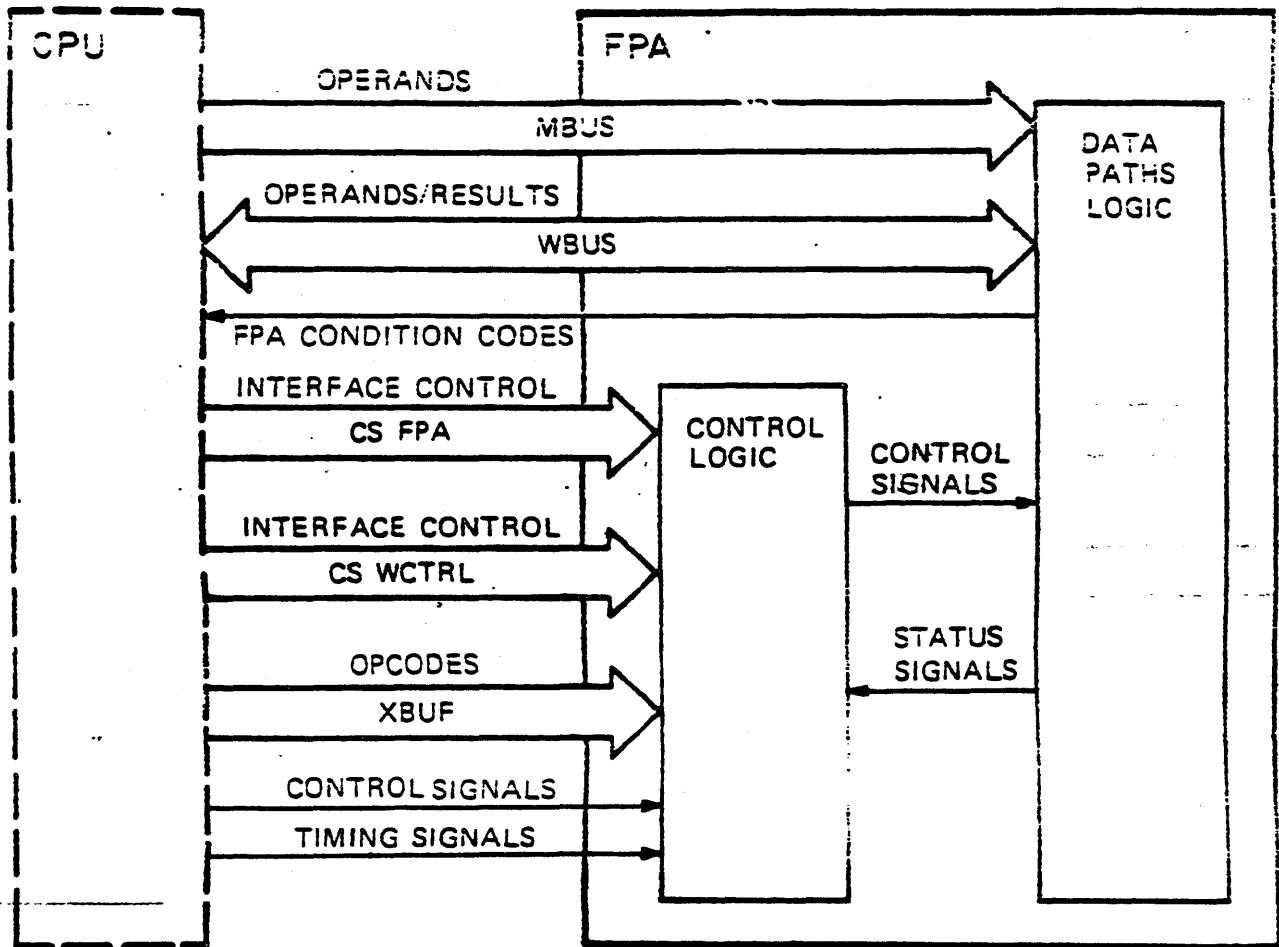
THE MICROFICHE LISTINGS FOR ECKAB.EXE AND ECKAC.EXE  
 GIVE BOARD LAYOUTS AND LOCATIONS OF EACH GATE ARRAY.

```

-----
|      (TOP)
|      THE FPA MODULE #L0001
|-----|-----|-----|-----|
| FCC | | FGA | | FEX | | FEX |
|-----|-----|-----|-----|

O> LED (FPA ENABLED)

|-----|-----|-----|
| FFA | | FFA | | FIO |
|-----|-----|-----|
| FFA | | FCS | | FIO |
|-----|-----|-----|
| CLA | | FCS | | F4R |
|-----|-----|-----|
| FFA | | FCS | | F4R |
|-----|-----|-----|
| FFA | | FIO | | FIO |
|-----|-----|-----|
| FFA | | FIO | | FIO |
|-----|-----|-----|
| FFA | | CLA | | FIO |
|-----|-----|-----|
| FFA | | FIO | | FIO |
|-----|-----|-----|
  
```



FPA I/O CPU SIGNAL INTERFACE



\*\*\* CAUTION \*\*\*

THE L0001 MODULE, AS ALL 11/750 MODULES, CONTAINS ELECTROSTATIC DISCHARGE SENSITIVE DEVICES (ESDS). THE USE OF THE VELOSTAT KIT IS ESSENTIAL TO PREVENT DAMAGE WHICH MAY NOT BECOME IMMEDIATELY APPARENT. (VELOSTAT KIT NUMBER A2-W0299-10)

- 1) Run system shutdown or the equivalent (@SYSSSYSTEM:SHUTDOWN)
2) Verify that the hardware revision level is Rev 3 or higher, and microcode revision level is equal to or greater than 94 decimal (step 4).
3) The FP750 will only work correctly in 11/750 systems that contains MINIMUM CPU Microcode revision 94. To verify revision level of the 11/750 system examine the System IDENTIFICATION Register (SID).
4) Place the keyswitch to the LOCAL position, HALT system (^P) and type the following:

>>>E/I 3E

(printout) I 0000003E 02005EXX



- XX = 03 L0011 controller/old backplane
XX = 30 L0011 controller/old backplane/SID switch
XX = 38 L0016 controller/new backplane/SID switch

VJIE: VAX750-R-003 FCO (REV 94 Micro code) is a prerequisite before the installation of a FP750 option. This FCO consists of rework to the L0004 UBI module and replacement of the L0005 CCS module, this will bring both modules (CCS and UBI) to REV "H".

This FCO can be ordered using EQ-01128-01 number.

\*\*\* DO NOT INSTALL THE L0001 MODULE WITHOUT PROPER MICROCODE LEVEL \*\*\*

5) Place the frontpanel Action on Power Switch to the HALT position, and remove power from the system.

6) Unpack the VeloStat tool from its container, open package, and attach the 15' pot ground cord to the VeloStat snap fastener, which attached to the wrist strap. Attach the end with the alligator clip to a reliable electrical ground on the 11/750 system.

7) Install the L0001 FPA module in slot #1 of the CMI backplane. If the RDM or DM L0006 module is not already resident in the CPU, install it in slot #6, move the console and IU58 cables from the left side to the right side of slot #6 on the processor backplane (looking at the back of the processor).

8) Reapply primary power by turning the keyswitch to local position, on the console panel. The system will come up in the HALT state, with the console prompt >>>.

9) Test the FPA's ON/OFF (enable/disable) capability by using the following console commands:

```
>>> D/I 28 0
>>> E/I 28
```

```
(print out)          00000028  00000000
```

Depositing 0 to IPR# 28 disables FPA (GREEN LED will not be lit)

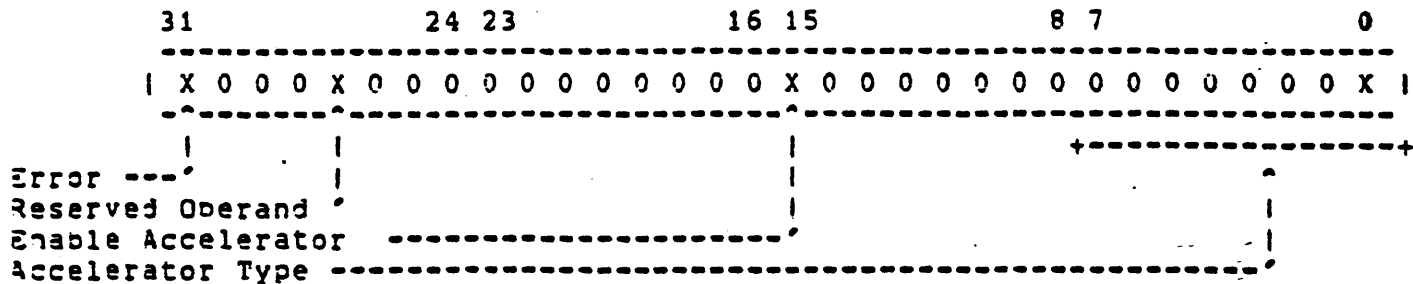
```
>>> D/I 28 8000
>>> E/I 28
```

```
(print out)          00000028  00000001
```

Depositing 8000 to IPR# 28 enables FPA (GREEN LED will be lit)

NOTE: FP750 uses the Accelerator Status/Control register IPR #28

The following diagram describes the bit position of the Accelerator Control/Status Register (ACCS)



Bit <15> - FPA Enable (WO)

Bit <7:0> - Accelerator Type

- 0 = No accelerator (or disabled FPA)
- 1 = Enabled FPA
- 2-255 = reserved

Note: ACCS <15> always reads as 0. In order to determine if an 11/750 has an FPA you must first write a 1 to ACCS <15> then read ACCS <0>. If it reads 0 there is no FPA present, if it reads back as an 1 there is an FPA and it is now Enabled.

Diagnostic Acceptance  
 #####

The VAX Architectural and Floating Point Instruction Exercisers are used to verify the integrity of the FP750 option. These instruction exercisers will run under the Diagnostic Supervisor stand-alone or on-line, but to properly test the FP750 they should be run stand-alone. (In on-line mode, they cannot disable and enable the FP750). The enhanced DPM micro diagnostic is used to verify the logic which interfaces the FP750 from the DPM module of the 11/750 processor.

Version V07.2 or greater of the ECKAB DPM micro diagnostic will verify some of the CPU-FPA interface logic. Verification of this interface logic, which is resident in the DPM module of the CPU and is not activated until the FPA is enabled, is tested under test D2-DC of this diagnostic. These test are listed below:

- D2 The FPA Enable/Disable Function
- D3 FPA Stall/wait Test
- D4 MBUS/WBUS Interface Test
- D5 FPA Reserved Operand Trap Test
- D6 FPA Trap Logic Test
- D7 Condition Code Test (1 of 4)
- D8 Condition Code Test (2 of 4)
- D9 Condition Code Test (3 of 4)
- DA Condition Code Test (4 of 4)
- DB FPA Copy to Condition Codes
- DC FPA Copy of the FU Bit

10) Boot Diagnostic Supervisor (ECSAA) and attach the processor.

```
>>> B/10 [device]
DS> ATT KA750 CMI KAO NO NO YES 0 1
                                     |
                                     | (Accelerator type)
                                     |--- ( 1=FPA 0=No FPA)

DS> SElect ALL
DS> SET IRace
```

11) Run EVKAB

The ARCHITECTURAL instruction exerciser will run first with the FP750 disabled (Green LED on the L0001 Module will not be lit). When this first pass has completed successfully, another pass with the FP750 enabled will be performed.

12) Run EVKAC

The FLOATING POINT instruction exerciser like EVKAB, will also run with the FP750 disabled on the first pass. Again after when successful completion, the FP750 is enabled and tested.

If a failure occurs when running either EVKAB or EVKAC, run ECKAB (DPM micro diagnostic) to verify the integrity of the interface logic on the DPM module. If the failure is not detected when running ECKAB, replace the L0001 module and rerun EVKAB, EVKAC, and ECKAB.

13) Run ECKAB

Run DPM micro diagnostic to verify the interface logic

```
^P          (go to console mode(nalt any macro
              program activity))

^D          (go to RDM console control mode)

RDM>       (get RDM Prompt;install TU58 tape)

RDM> IE    (start test)
```

14) Error Free Passes of EVKAB, EVKAC and ECKAB indicate verification is complete (diagnostic runtime is about 45 minutes).

15) If RDM is to be removed, power down system, remove L0006 module, replace Console and TU58 cables to their original positions.

16) Dis-connect the VelaStat tool and repack it in the kit container.

17) Power up system.

18) Bootstrap customer's operating software.

**L0002 DPM**

SLOT 2). THE DATA PATHS MODULE (DPM) #L0002

IT HOUSES THE GENERAL PURPOSE REGISTERS (GPR'S), INTERNAL PRIVILEGED REGISTERS (IPR'S), MTEMP AND RTEMP MICROCODED REGISTERS, ARITHMETIC LOGIC UNIT (ALU), ROTATOR LOGIC, Q AND D REGISTERS, P AND S LATCHES, SYSTEM CLOCKS, MICROSEQUENCER, IR DECODE ROMS.

THE DPM IS CONNECTED TO THE W BUS AND M BUS.  
(THE MICROSEQUENCER IS CONNECTED TO THE CCS OF COURSE)

NOTE: THERE IS NO PARITY CHECKING AT ALL ON THE DPM MODULE OTHER THAN CONTROL STORE MICROCODE PARITY WHICH IS LATCHED ON THE MIC, UBI, DPM, AND CCS MODULES AND CHECKED ON THE DPM.

THE MICRODIAGNOSTIC ECKAB.FXE TAPE #1 WILL TEST THE DPM.

GATEARRAYS: ALP, ALK, CCC, CLA, IRD, MSQ, PHB, SAC, SPA, SRK, SHM, TOK

GATE-ARRAY MAGIC BOOK PICTURE SYMBOLOGY

-----  
X:X = COMPLETE RANGE OF BITS CONTAINED IN EACH CHIP  
N = NOT APPLICABLE IN THIS CHIP  
<> = BI-DIRECTIONAL (MAJOR BUS)  
o = INVERTED

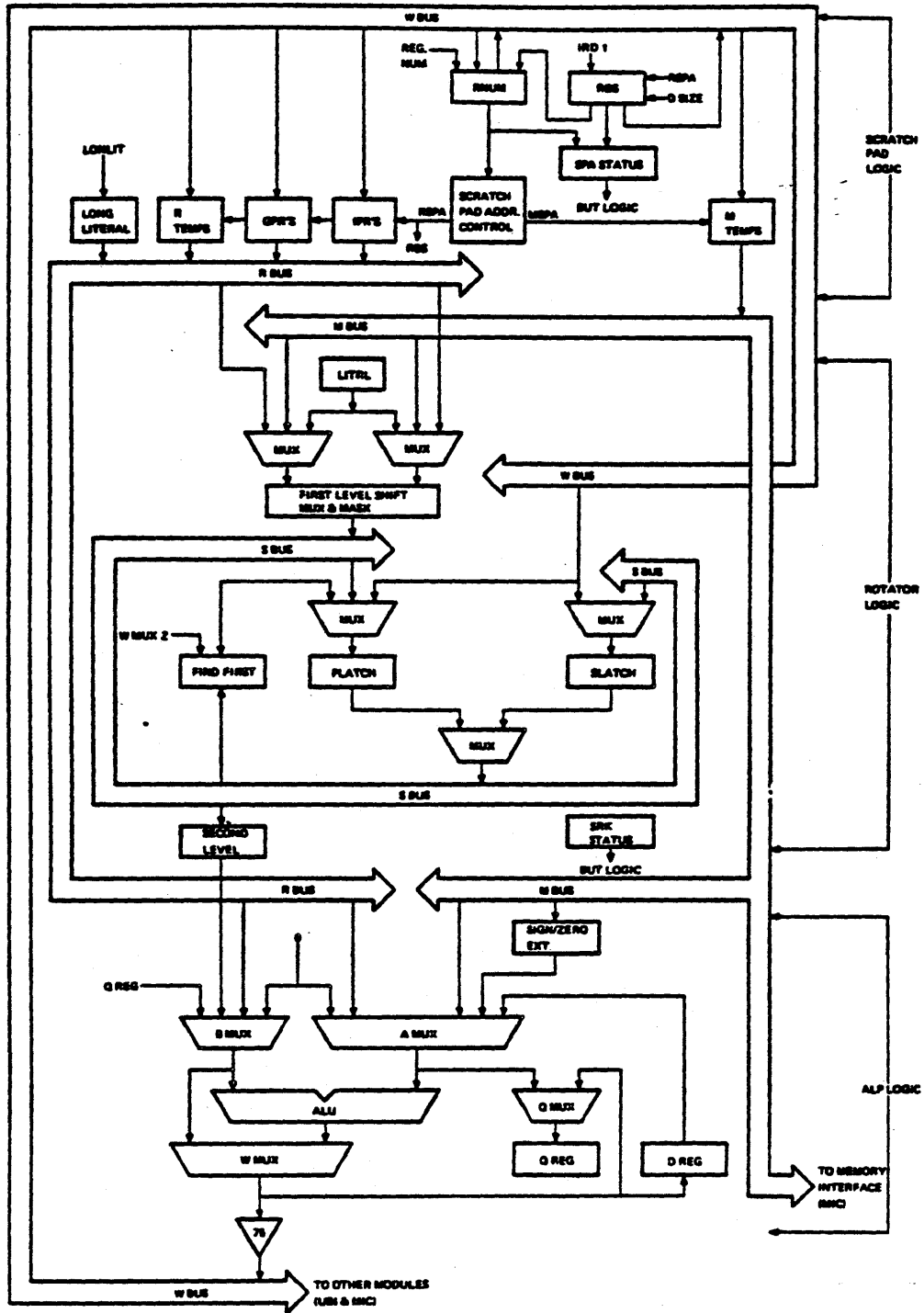
(TOP)

THE DPM MODULE #L0002

PINS&gt;&gt;&gt;&gt;

	PHB	MSG	
	CCC	SAC	
SPA	SRK	IRD	
TOK	CLA	ALK	
	ALP1	ALP5	
	ALP2	ALP6	
	ALP3	ALP7	
	ALP4	ALP8	
SRM1	SRM2	SRM3	SRM4

DATA PATH



Data Path Block Diagram



ALP: ARITHMETIC LOGIC PROCESSOR (ALU)  
 CONTAINS LOGIC TO PERFORM 1094 ARITHMETIC AND LOGICAL  
 FUNCTIONS. ALSO CONTAINS A "SECOND LEVEL" SHIFTER FOR USE  
 WITH THE SUPER ROTATOR MULTIPLEXER. (SEE SRK AND SRM CHIPS)

3 CHIPS:	CHIP	BIT SLICE
	ALP 1	<3-0>
PART NUMBER	ALP 2	<7-4>
19-14682	ALP 3	<11-8>
	ALP 4	<15-12>
	ALP 5	<19-16>
	ALP 6	<23-20>
	ALP 7	<27-24>
	ALP 8	<31-28>

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM                    GATE ARRAY: ALP (1 THROUGH 8)

BUS DEFINITIONS: SB = SBUS (SUPER ROTATOR BUS INTERNAL TO DPM ONLY)  
 USED TO TRANSFER INTERNAL ROTATOR DATA BETWEEN  
 FIRST AND SECOND LEVEL SHIFTERS.  
 RB = RBUS (ROTATOR BUS INTERNAL TO DPM ONLY)  
 USED TO TRANSFER DATA FROM SCRATCH PAD REGISTERS  
 (EXCLUDING MTEMPS) TO ALU.  
 MB = MBUS (MEMORY BUS. SEE MAJOR BUS DEFINITIONS PAGE)  
 WB = WBUS (WRITE BUS. SEE MAJOR BUS DEFINITIONS PAGE)

TERM DEFINITIONS: WMUXZ = W BUS MULTIPLEXER EQUAL TO ZERO

	ALP	N,N,EXT_DATA,EXT_DATA,EXT_DATA
SB(2,6,10,14,18,22,26,30)---1---	o	o_48_EXT_DATA,EXT_DATA,EXT_DATA
Q (0,3,7,11,15,19,23,27)---2_o	o	Io_47_SHF 1
SB (1,5,9,13,17,21,25,29)---3---		l--46_SB (6,10,14,18,22,26,30,34)
QD CLK---4_o		l--45_SB (3,7,11,15,19,23,27,31)
GENERATE CARRY (G X:X)---5_o		Io_44_SHF 0
ALUC(0,3,7,11,15,19,23,27)---6_o		l--43_SB (5,9,13,17,21,25,29,33)
V (N,7,N,15,N,N,N,31)---7---		l--42_SB (4,8,12,16,20,24,28,32)
A (3,7,11,15,19,23,27)---8_o		l--41_SB (0,4,8,12,16,20,24,28)
PROPAGATE CARRY (P X:X)---9_o		Io_40_Q (3,7,11,15,19,23,27,31)
WB(3,7,11,15,19,23,27,31)---10<>		Io_39_RB (3,7,11,15,19,23,27,31)
WB (1,5,9,13,17,21,25,29)---11<>		l--38_GROUND
VGA---12---		Io_37_RB (2,6,10,14,18,22,26,30)
VCC---13---		Io_36_RB (1,5,9,13,17,21,25,29)
WB(2,6,10,14,18,22,26,30)---14<>		l--35_GROUND
WB (0,4,8,12,16,20,24,28)---15<>		Io_34_DP PHASE
A (0,3,7,11,15,19,23,27)---16_o		Io_33_Rb (0,4,8,12,16,20,24,28)
ALPCTL 2 (OPC2)---17---		Io_32_MB (2,6,10,14,18,22,26,30)
ALK OP 4 (OPC4)---13---		Io_31_MB (3,7,11,15,19,23,27,31)
ALK OP 5 (OPC5)---19---		Io_30_Mb (0,4,8,12,16,20,24,28)
ALPCTL 3 (OPC3)---20---		l--29_ALPCTL 9 (OPC9)
WMUXZ B(0,0,1,1,2,2,3,3)---21---		Io_28_Mb (1,5,9,13,17,21,25,29)
ALK OP 6 (OPC6)---22---		l--27_ALPCTL 7 (OPC7)
ALK OP 0 (OPC0)---23---		l--26_ALPCTL 8 (OPC8)
ALK OP 1 (OPC1)---24---	( )	Io_25_+3VDDM,+3VDDH,X(8:15)EN, ----- X(8:15)EN,DSIZE1,DSIZE1,DSIZE1, DSIZE1

THIS SIDE TOWARDS FINGERS ON BOARD

ALK: ARITHMETIC LOGIC CONTROL

CONTROLS THE ALP FUNCTIONS BY DECODING MICROCODE  
INPUTS AND GENERATING THE CONTROL SIGNALS.

PART NUMBER 19-14689

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM                      GATE ARRAY: ALK

TERM DEFINITIONS: (SIO) = SHIFT IN/OUT

	ALK	
Q (SIO) 7__1__o	-----	o_48_Q (SIO) 0
ALU SIO 0__2__o	o	<>47_W8 31
ALU SIO 31__3__o		<>46_W8 30
ALPCTL 0__4__I		o_45_Q (SIO) 15
ALPCTL 1__5__I		o_44_Q (SIO) 31
ALPCTL 6__6__I		__43_ALPCTL 3
ROT 3__7__I		__42_ALK OP 0
ROT 4__8__I		__41_C 31
ROT 2__9__I		__40_ALPCTL 4
DOUBLE ENABLE_10__I		__39_BCD
PSLC_11__I		__38_GROUND
VGA_12__I		__37_ALPCTL 5
VCC_13__I		__36_ALPCTL 2
ALK OP 6_14__I		__35_GROUND
ROT 0_15__I		__34_ALPCTL 3
ALK OP 5_16__I		o_33_LONG LITERAL
ROT 1_17__I		__32_SPW 1
QD CLK_18__o		__31_D SIZE 0
ROT 5_19__I		__30_SPW 0
ALPCTL 7_20__I		__29_SPWB ENABLE (BYTE)
CARRY OUT (COUT)_21__o		__28_SPWL ENABLE (LONGWORD)
ALPCTL 9_22__I		__27_SPWW ENABLE (WORD)
ALK OP 4_23__I		__26_D SIZE 1
ALK OP 1_24__I	( )	o_25_(BYTE) X(3:15)EN

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

CCC:      CONDITION CODE CHIP

CONTAINS PSL(PSW) BITS <C,V,Z,N,IV,FH,OV>

CONTROLS THE SETTING OF ALL CONDITION CODES FOR VAX  
NATIVE AND COMPATABILITY MODE INSTRUCTIONS AT THE  
REQUEST OF THE MICROCODE.

WORKS IN CONJUNCTION WITH THE ALU.

PART NUMBER: 19-14684

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE  
                  MIC MICRO'S ECKAC.EXE

MODULE: DPM                    GATE ARRAY: CCC

TERM DEFINITIONS: CCBR = CONDITION CODE BRANCH

	CCC	
D SIZE 0__1__	-----o	49_FPA PRESENT
D SIZE 1__2__	o	<>47_WB 15
WMUX Z 80 (BYTE) __3__		__46_ALUV 31
IR 6__4__		__45_ALUV 7
IR 4__5__		__44_ALUV 15
IR 7__6__		__43_WMUX Z 31 (BYTE)
IR 5__7__		o_42_ALUC 31
IR 3__8__		o_41_ALUC 10
IR 2__9__		o_40_ALUC 07
IR 1__10__		<>39_WB 31
IR 0__11__		__38_GROUND
VGA__12__		__37_WMUX Z 83 (BYTE)
VCC__13__		<>36_WB 7
FPA Z__14__o		__35_GROUND
ARITHMETIC TRAP__15__o		<>34_WB 5
FPA V__16__o		__33_CCBR 1
PSLC__17__		<>32_WB 6
CCBR 0__18__		__31_WMUX Z 82 (BYTE)
PROC INIT__19__o		<>30_WB 4
BUFF B CLK__20__		__29_CC CTRL 3
WB 0__21<>		__28_CC CTRL 1
WB 3__22<>		__27_CC CTRL 2
WB 1__23<>		__26_CC CTRL 0
WB 2__24<>		__25_D CLK EN
	( )	

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

CLA: CARRY LOOK AHEAD

LOOKS AT ALP CHIPS (P AND G SIGNALS) TO GENERATE AND/OR PROPAGATE CARRIES, AS WELL AS DEALING WITH THE STATE OF THE MOST SIGNIFICANT BITS OF EACH DATA TYPE TO AID THE CCC CHIP WITH DETERMINATION OF POSITIVE OR NEGATIVE CHARACTERISTICS.

PART NUMBER: 19-14686

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM

GATE ARRAY: CLA

TERM DEFINITIONS: BCD = BINARY CODED DECIMAL  
ALUC = ALU CARRY BITS

```

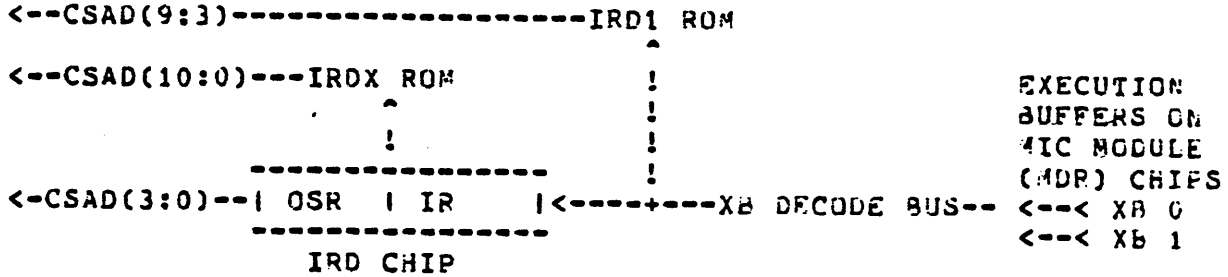
                                CLA
G (12:15)  _1_ 0----- _48_QD CLK
NON BCD  _2_ 1 0      | _47_LONG LIT
ALUC 23 (C6) _3_ 01   | _46_+3V NUM
P (12:15)  _4_ 01   | _45_LITREG CLK
G (04:07)  _5_ 01   | _44_G (31:28)
ALUC 0 (C0) _6_ 01   | _43_P (23:20)
              N _7_ 1   | _42_SCD FROM ALK
              N _8_ 1   | _41_P (19:16)
              N _9_ 1   | _40_ALUC 07 (C2)
ALUC 31    _10_ 01   | _39_P (31:28)
G (03:00) _11_ 01   | _38_GROUND
              VGA _12_ 01 | _37_ALUC 15 (C4)
              VCC _13_ 01 | _36_G (11:08)
G (31:28) _14_ 01   | _35_GROUND
ALUC 31    _15_ 01   | _34_G (19:16)
G (23:20) _16_ 01   | _33_P (03:00)
G (15:12) _17_ 01   | _32_P (27:24)
P (11:08) _18_ 01   | _31_G (07:04)
G (27:24) _19_ 01   | _30_P (07:04)
ALUC 27 (C7) _20_ 01 | _29_G (03:00)
G (11:08) _21_ 01   | _28_G (23:20)
G (27:24) _22_ 01   | _27_CARRY IN FROM ALK
ALUC 19 (C5) _23_ 01 | _26_ALUC 03 (C1)
G (19:16) _24_ 01   | _25_ALUC 11 (C3)
                                -----

```

THIS SIDE TOWARDS FINGERS ON BOARD

IRD: INSTRUCTION REGISTER DECODE CHIP

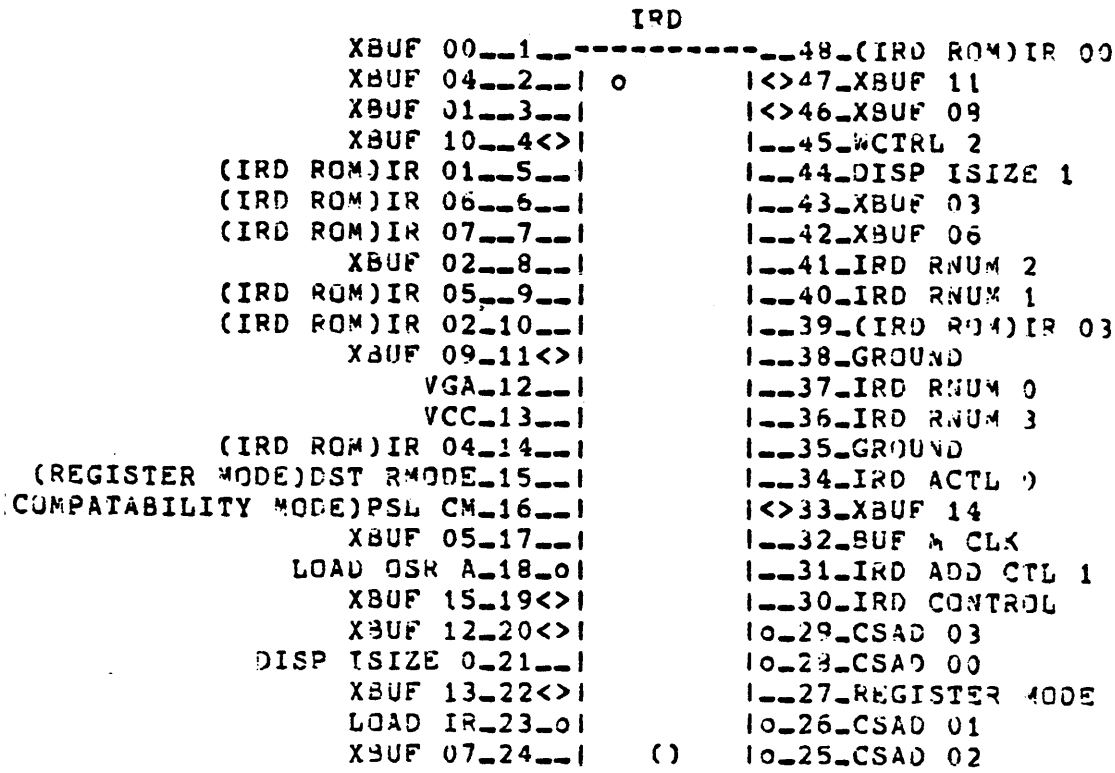
RECEIVES INSTRUCTION STREAM DATA FROM THE EXECUTION BUFFERS,  
HELPS DECODE THE OP CODE AND 1st OPERAND SPECIFIER, AND ASSISTS  
THE IRD1 DECODE ROMS IN GENERATING THE PROPER MICROCODE ADDRESS  
FOR EACH INSTRUCTION AND ADDRESSING MODE. OP CODES ARE LATCHED  
IN IRD UNTIL NEXT IRD1 TIME.



PART NUMBER: 19-14696

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM                    GATE ARRAY: IRD



-----  
THIS SIDE TOWARDS FINGERS ON BOARD

MSQ: MICRO SEQUENCER

SEQUENCES THE MICROCODE, FORMS LOW 6 BITS OF CONTROL STORE ADDRESS, CONNECTED TO THE MICRO STACK AND NEXT FIELD LATCHES.

PART NUMBER: 19-14695

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM

GATE ARRAY: MSQ

```

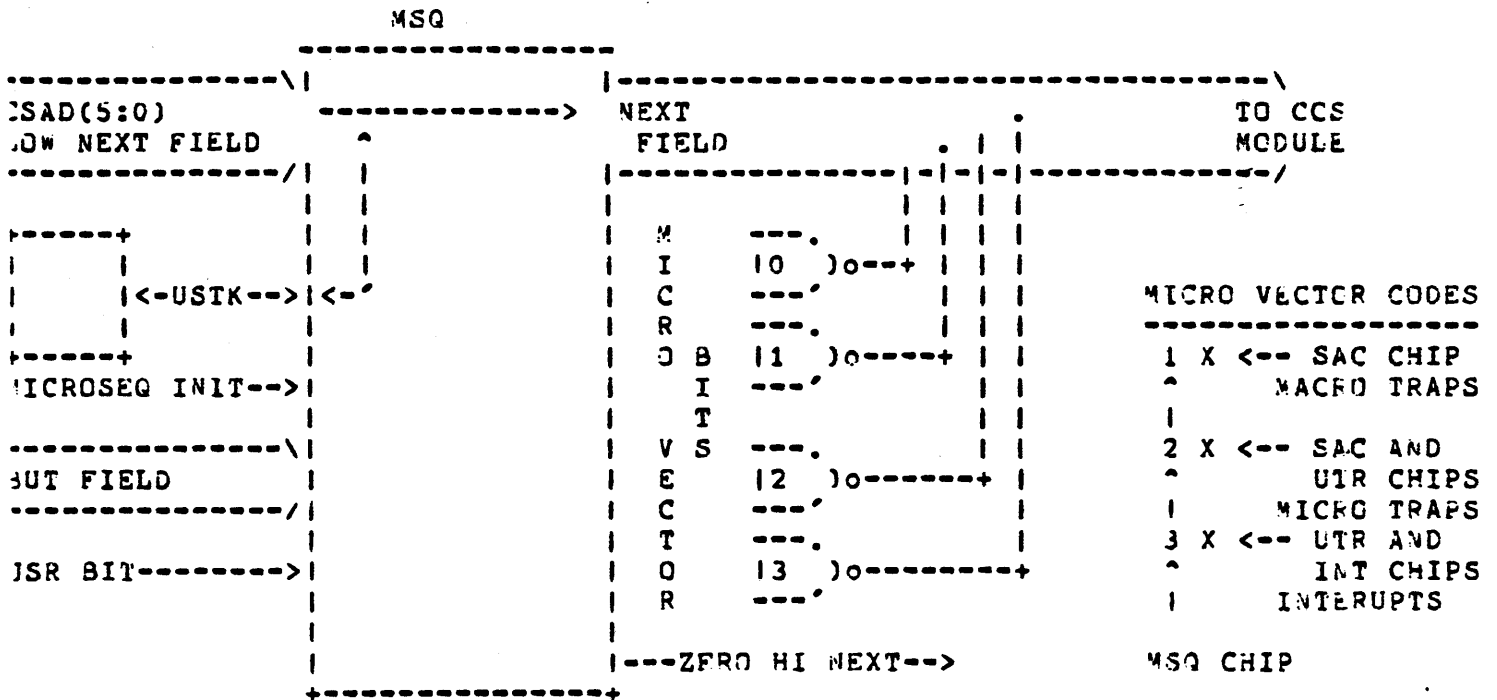
                                MSQ
LIT 1--1-----48_DISABLE HI NEXT
LOAD QSR A--2--o  o          1--47_BUF B CLK
BUT CTRL CODE A--3--|          1o-46_DO SERVICE
    BUT 2--4--|          1o-45_MSEQ INIT (NEGATION OF DCLO)
    BUT 0--5--|          1--44_BUF W CLK
    BUT 1--6--|          1--43_STK 5
    LIT 0--7--|          1o-42_OSIN (FROM OS ROM INVH)
    FPA WAIT--8--o|          1o-41_ZERO HI NEXT
    USTK ADDR 3--9--|          1o-40_MICRO ADDRESS INHIBIT
    USTK ADDR 1-10--|          1o-39_ENABLE IRD RO4 H
    USTK ADDR 2-11--|          1--38_GROUND
        VGA-12--|          1o-37_USTK OUTPUT ENABLE
        VCC-13--|          1--36_JSR
    USTK ADDR 0-14--|          1--35_GROUND
        IRD CTR 1-15--|          1o-34_CSAD 5 (ADC)
        IRD CTR 2-16--|          1--33_NXT 5
        STK 0-17--|          1--32_NXT 4
        STK 1-18--|          1o-31_CSAD 4 (ADC)
        NXT 1-19--|          1--30_STK 3
    .CSAD 0 (ADC)-20--o|          1--29_STK 4
    CSAD 1 (ADC)-21--o|          1--28_NXT 3
        STK 2-22--|          1o-27_CSAD 3 (ADC)
        NXT 2-23--|          1--26_ENABLE MICRO VECTOR
        NXT 0-24--|          ( ) 1o-25_CSAD 2 (ADC)

```

-----  
 THIS SIDE TOWARDS FINGERS ON BOARD

THERE ARE FIVE WAYS OF SEQUENCING THE MICROCODE:

- 1). STRAIGHT THROUGH NEXT FIELD ADDRESSING (NO MODIFICATIONS)
- 2). BRANCHING ON BUT CONDITIONS IN HARDWARE
- 3). PERFORMING MICRO VECTOR OPERATIONS
- 4). JUMPS TO AND RETURNS FROM MICRO SUBROUTINES
- 5). INSTRUCTION DECODE



PHB: PRACTICALLY HALF BUTS

CONTAINS PSL BITS <CM,TP,FPD>, STEP COUNTER, STATUS FLAGS,  
AND ABOUT HALF THE LOGIC TO PERFORM (BUT)BRANCH AN MICFC-  
TEST MICRO-ORDERS.

PART NUMBER: 19-14703

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM                      GATE ARRAY: PHB

TERM DEFINITIONS: PHB GOOD SAM = GOOD SAMARITAN BUS

PHB	
D CLK ENABLE H__1_0	-----0_48_CSAD 1
CSAD 0__2_0   0	__47_MISC CTL 0
WB 00__3<>	0_46_LOAD IR
N__4__	0_45_BUF M CLK
PSL FPD__5__	__44_PHB GOOD SAM 1
WB 03__6<>	<>43_WB 04
INTERUPT__7__	__42_PHB GOOD SAM 2
CSAD 3__8_0	__41_PHB GOOD SAM 0
CSAD 2__9_0	<>40_WB 27
DO SERVICE_10_0	<>39_WB 31
WB 01_11<>	__38_GROUND
VGA_12__	__37_PSL TP
VCC_13__	<>36_WB 30
WB 02_14<>	__35_GROUND
DISABLE CSAD_15__	__34_PSL CM (COMPATABILITY MODE)
BUT 0_16__	0_33_CSAD 04
BUT 1_17__	__32_IRD LOAD RNUM
LONG LITERAL_18_0	__31_IRD ADDR CTL 0
BUT 4_19__	0_30_LOAD OSR A
BUT 5_20__	__29_IRD ADDR CTL 0
BUT 2_21__	__28_MISC CTL 3
BUT 3_22__	__27_MISC CTL 1
WB 05_23<>	__26_MISC CTL 4
CSAD 5_24_0   ( )	__25_MISC CTL 2
	-----

THIS SIDE TOWARDS FINGERS ON BOARD



# SERVICE ARBITRATION AND CLOCKS 31

CONTAINS THE IRD COUNTER (WHICH IS USED PRIMARILY TO TRACK THE NUMBER OF BYTES OF ISTREAM DATA THAT HAS BEEN EXECUTED), SERVICE ARBITRATION REFERS TO THE PRIORITIZING OF TRAPS AND MICROTRAPS, AND FINALLY THE SAC CHIP CREATES ALL SYSTEM CLOCKS. (BASE,B,QD,M,PHASE) FROM THE OSCILLATOR INPUT (B27-B28 ON BACKPLANE), SAC ALSO HAS CONNECTIONS TO THE RDM MODULE TO HANDLE CLOCKING CONTROL AND DISABLES THE CCS BOARD FROM DRIVING THE MICRO ADDRESSES DURING MICRO DIAGNOSTIC EXECUTION. THE SAC CHIP ALSO MONITORS THE FIRST CS PARITY ERROR (DETECTED BY THE ACV CHIP ON THE MIC MODULE), LOOKING FOR ANOTHER ONE BEFORE THE FIRST IS DONE WITH IT'S MICRO-TRAP (IN WHICH CASE IT WILL IMMEDIATELY HALT THE CLOCK.

BASE CLOCK = 160 NS 6.25 MHZ RUNS CONSTANTLY  
 B CLOCK = 160 NS 6.25 MHZ BASIC SYNCHRONIZED CLOCK USED THROUGHOUT THE CPU AND CMI.  
 M CLOCK = 320 NS CAN BE EXTENDED WITH THE CLKX BIT OF MICROCODE TO 480 NS THIS CLOCK IS THE MAIN MICROSEQUENCER CLOCK USED TO STROBE MICROCODE FROM CCS.  
 PHASE CLOCK= SPLITS THE M CLOCK INTO TWO HALFS. PHASE IS HIGH DURING THE FIRST 160 NS DURING WHICH TIME ALL READS OCCUR, THE PHASE IS LOW DURING THE SECOND 160 NS WHEN ALL WRITES OCCUR. THE PHASE CLOCK IS ALSO EXTENDED WITH THE CLKX BIT.  
 QD CLOCK = FOLLOWS M CLOCK, IT IS USED TO CLOCK THE Q (QUOTIENT) AND D (DIVIDEND) REGISTERS INTERNAL TO THE ALU SECTION OF THE DPM MODULE.

PART NUMBER: 19-14691

BEST DIAGNOSTICS: ALL MICRO'S DPM MICRO'S

MODULE: DPM GATE ARRAY: SAC

## SAC

CONSOLE HALT	1_o	48_CLKX (CLOCK EXTEND)
CPU OSC IN(18.75 MHZ)	2_o	47_FF TRAP L
N	3_o	46_ARITHMETIC TRAP
PHASE	4_o	45_FPA STALL
MEM STALL	5_o	44_FPA WAIT
SETC	6_o	43_TIMER SERVICE
BASE CLOCK	7_o	42_CSAD 1
RD ADDR INHIBIT(FROM RDM)	8_o	41_CSAD 0
CS PARITY ERROR	9_o	40_CSAD 2
SL CM (COMPATABILITY MODE)	10_o	39_INTERRUPT PENDING
HALT(STOPS B CLK)	11_o	38_GROUND
VGA	12_o	37_PSL TP H
VCC	13_o	36_LOAD OSR
MICRO TRAP	14_o	35_GROUND
IRD CTR 1	15_o	34_DO SERVICE
BUT 0	16_o	33_ENABLE MICRO VECTOR
BUT 2	17_o	32_DOUBLE ENABLE
BUT 1	18_o	31_QD CLK ENABLE
BUT CTRL CODE A	19_o	30_LATCH MICRO TRAP
BUF M CLK	20_o	29_GEN DEST INHIBIT
IRD CTR 0	21_o	28_M CLK ENABLE
IRD CTR 2	22_o	27_D CLK ENABLE
MICRO SEQUENCER INIT	23_o	26_CLK CTRL 0 (FROM RDM)
INSTRUCTION FETCH	24_o	25_CLK CTRL 1 (FROM RDM)

SPA: SCRATCH PAD ADDRESSING CHIP

THE SPA CONTROLS THE ADDRESSING OF THE 64 SCRATCHPAD REGISTERS AND IT CONTROLS THE REGISTER BACKUP STACK (BACKS UP UP TO 6 GPR REGISTERS), CONTAINS LOGIC TO KEEP TRACK OF THE AUTO-INCREMENTING/DECREMENTING OF THE GPR'S, DEVELOPS IT'S OWN STATUS BITS AND ENABLE SIGNALS FOR THE VARIOUS REGISTERS.

PART NUMBER: 19-14690

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM

GATE ARRAY: SPA

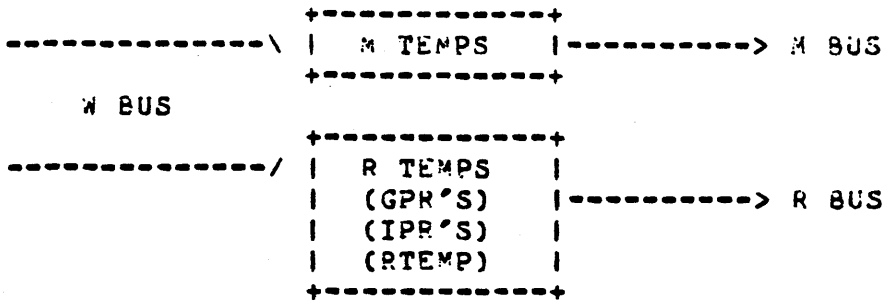
TERM DEFINITIONS: MSPA = MTEMP SCRATCHPAD ADDRESS  
 RSPA = RTEMP SCRATCHPAD ADDRESS  
 RNUM = REGISTER NUMBER BUS (FROM RNUM REGISTER)

## SPA

IRD RNUM 1	1	-----	48	IRD RNUM 3		
D SIZE 0	2	0		47	IRD RNUM 2	
IRD LOAD RNUM (LOAD REGISTER)	3			<>	46	WB 02
D CLK ENABLE	4			<>	45	WB 03
MSRC 3	5			44	IRD RNUM 0	
BUF M CLK	6	0		<>	43	WB 00
PHASE	7			<>	42	WB 01
MSRC 2	8			41	SPA ST0 (STATUS)	
MSRC 1	9			40	SPA ST1 (STATUS)	
MSRC 0	10			39	D SIZE 1	
MSRC 4	11			38	GROUND	
VGA	12			37	INSTR FETCH	
VCC	13			36	RCS GPR (ENABLE)(RSRC/=GPR)	
RSPA 1	14			35	GROUND	
MSPA 1	15			34	RSRC 1	
MSPA 2	16			33	LITREG ENABLE	
RSPA 2	17			32	RSRC 3	
MCS TEMP (ENABLE)(MSRC/=TMP)	18	0		31	DSI RMODE (DEST. IS REG. MODE)	
RSPA 0	19			30	RSRC 4	
MSPA 0	20			29	RSRC 5	
RSPA 3	21			28	LIT 0	
MSPA 3	22			27	RSRC 0	
RSRC 2	23			26	RCS IPR (ENABLE)(RSRC/=IPR)	
SPWM	24	0	( )	25	RCS TMP (ENABLE)(RSRC/=TMP)	

THIS SIDE TOWARDS FINGERS ON BOARD

REGISTERS FALL INTO TWO CATEGORIES RTEMP'S AND MTEMP'S. THE DEFINITION OF THESE CATEGORIES IS AS FOLLOWS:



M TEMP'S = M BUS TEMPORARY REGISTERS USED BY THE MICROCODE FOR TEMPORARY STORAGE OF DATA.  
(CAN ONLY BE ACCESSED BY MICROCODE OR MICRO-MONITOR)

R TEMP'S = CONSIST'S OF ALL REGISTERS THAT FEED THE R BUS;

R BUS TEMPORARY REGISTERS (MICROCODE OR MICRO-MONITOR)  
GPR'S 16 GENERAL PURPOSE REGISTERS  
IPR'S INTERNAL PRIVILEGED REGISTERS (TEMPORARY HOLDING POINT FOR DATA DESTINED FOR MEMSCR'S)

NOTE: RTEMP AND MTEMP 0-7 ARE DUAL PORTED TOGETHER.  
EX. WRITING RTEMP 5 ALSO WRITES MTEMP 5  
READS STILL ONLY AFFECT ONE OF THEM

CONTROLS SUPER ROTATOR MULTIPLEXER OPERATIONS. LOOKS AT ROT FIELD OF MICROCODE AND TELLS THE SRM CHIPS WHAT TO DO. S AND P LATCHES (SIZE AND POSITION) ARE CONTAINED IN THIS CHIP.

NOTE: THE SRK CHIP CONTROLS THE SRM CHIP FUNCTIONS VIA A COMPLEX ARRAY OF SIGNALS CALLED PRI (PRIMARY), SEC (SECONDARY), AND SHF (SHIFT). THESE SIGNALS AND MALFUNCTIONS OF THESE SIGNALS CAN BE QUITE CONFUSING AND ARE BEST DIAGNOSED USING THE MICRO-DIAGNOSTICS.

PART NUMBER: 19-14688

BEST DIAGNOSTICS: DPM MICRO'S ECKA3.EXE

MODULE: DPM

GATE ARRAY: SRK

		SRK		
	ROT 3	1	-----	43_ROT 2
	ROT 0	2	0	10_47_(PRIMARY_FUNCTION)PRI 1
	ROT 5	3		1_46_ROT 1
(SECONDARY_FUNCTION)SEC	3	4	0	10_45_(SECONDARY_FUNCTION)SEC 4
DSIZE	0	5		10_44_(SECONDARY_FUNCTION)SEC 5
DSIZE	1	6		10_43_(SECONDARY_FUNCTION)SEC 2
(PRIMARY_FUNCTION)PRI	0	7	0	10_42_(SECONDARY_FUNCTION)SEC 0
WMUXZ BYTE	1	8		10_41_(SECONDARY_FUNCTION)SEC 1
	ROT 4	9		10_40_QD CLK
WMUXZ BYTE	0	10		10_39_SHF 2(TO SRM 1ST LEVEL SHIFT)
SHF 1(TO ALP 2ND LEVEL SHIFT)		11	0	1_38_GROUND
	VGA	12		1_37_SRK ST1 (STATUS)
	VCC	13		1_36_SRK ST0 (STATUS)
WMUXZ BYTE	3	14		1_35_GROUND
WMUXZ BYTE	2	15		10_34_SHF 4(TO SRM 1ST LEVEL SHIFT)
WB 05	16	<>		10_33_SHF 3(TO SRM 1ST LEVEL SHIFT)
WB 02	17	<>		10_32_SHF 0(TO ALP 2ND LEVEL SHIFT)
WB 07	18	<>		1<>31_SB 04
WB 03	19	<>		1<>30_SB 02
WB 00	20	<>		1<>29_SB 03
WB 06	21	<>		1<>28_SB 01
WB 04	22	<>		1<>27_SB 00
S3 06	23	<>		1<>26_SB 07
SB 05	24	<>	( )	1<>25_WB 01

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

**SRM: SUPER ROTATOR MULTIPLEXER**

4 CHIPS PERFORM THE 64 FUNCTIONS UNDER SRK CONTROL. THESE CHIPS CONTAIN A "FIRST LEVEL" SHIFTER WHICH IS CAPABLE OF TAKING A 64 BIT INPUT FROM ANY COMBINATION OF THE R BUS, M BUS, OR SHORT LITERAL FIELD OF MICROCODE, AND SHIFTING ANY MULTIPLE OF FOUR BITS (NIBBLE). THE 32 BIT OUTPUT FROM THE SHIFTER IS PLACED ON THE SUPER BUS (S BUS) AND SENT TO THE ALU SECTION OF THE DPM MODULE WHERE IF DESIRED IT CAN BE ROTATED FROM 0 TO 3 MORE BITS INSIDE A SECOND LEVEL SHIFTER (COMPLETE WITH BIT BUCKET) INTERNAL TO THE ALP CHIP'S. THE OUTPUT OF THIS SECOND LEVEL SHIFTER IS ONCE AGAIN 32 BITS.

CHIP	BITS
SRM 1	0,4,8,12,16,20,24,28,32
SRM 2	1,5,9,13,17,21,25,29,33
SRM 3	2,6,10,14,18,22,26,30,34
SRM 4	3,7,11,15,19,23,27,31

PART NUMBER: 19-14687

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM

GATE ARRAY: SRM (1 THROUGH 4)

		SRM	
	SHF 2__1_o	-----o_48	SEC 5
	SHF 4__2_o	o	Io_47_+3V NOM,MB 09,MB 10,ME 11
	SHF 3__3_o		Io_46_MB 31 L
RSRC 5,SEC 0,SEC 0,SEC 0	__4__I		Io_45_RB 28,29,30,31
RSRC 1,2,3,4	__5__I		Io_44_RB 20,21,22,23
	SEC 4__6_o		Io_43_RB 16,17,18,19
	SEC 3__7_o		Io_42_RB 24,25,26,27
	PRI 1__9_o		Io_41_RB 12,13,14,15
CC 0,CC 1,ISTRM,RSRC 0	__9__I		Io_40_RB 08,09,10,11
+3V NOM,GRND,GRND,GRND	__10_o		Io_39_RB 00,01,02,03
+3V NOM,+3V NOM,GRND,GRND	__11_o		I__38_GROUND
	VGA__12__I		Io_37_RB 04,05,06,07
	VCC__13__I		I__36_DP PHASE
	PRI 0__14_o		I__35_GROUND
SB 12,13,14,15	__15<>I		Io_34_MB 28,29,30,31
SB 8,9,10,11	__16<>I		Io_33_MB 24,25,26,27
	SEC 1__17_o		Io_32_MB 16,17,18,19
	SEC 2__18_o		Io_31_MB 08,09,10,11
	SEC 0__19_o		Io_30_MB 12,13,14,15
SB 0,1,2,3	__20<>I		Io_29_MB 04,05,06,07
SB 4,5,6,7	__21<>I		Io_28_MB 20,21,22,23
SB 16,17,18,19	__22<>I		Io_27_MB 00,01,02,03
SB 20,21,22,23	__23<>I		Io_26_SB 32,33,34,N
SB 24,25,26,27	__24<>I	( )	Io_25_SB 28,29,30,31

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

TOK: TIMED OPERATION CONTROL

PROGRAMMABLE INTERVAL CLOCK, 1 MICRO SECOND CLOCK, ABLE TO GENERATE INTERRUPTS AT PROGRAMMABLE INTERVALS.

PART NUMBER: 19-14694

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE

MODULE: DPM

GATE ARRAY: TOK

	TOK	
GROUND__1_0_	-----<>	48_WB 12
WB 19__2<>  0	<>	47_WB 15
WB 23__3<>	<>	46_WB 14
WB 24__4<>		45_IOK OSC (1 MHZ FROM CCS)
TIMER SERVICE (TRAP TO REFILL)__5__		44_N
WB 31__6<>	<>	43_WB 13
WB 18__7<>	<>	42_WB 11
WB 20__8<>	<>	41_WB 10
WB 17__9<>	<>	40_WB 08
PROC INIT__10_0	<>	39_WB 09
TIMER INTERRUPT__11_0		38_GROUND
VGA__12__		37_N
VCC__13__	<>	36_WB 07
WB 22__14<>		35_GROUND
WB 21__15<>	<>	34_WB 06
WB 16__16<>	<>	33_WB 05
BUF B CLK__17_0	<>	32_WB 04
N__18__	<>	31_WB 02
N__19__		30_N
D CLK ENABLE__20__	<>	29_WB 01
WCTRL 3__21__	<>	28_WB 03
WCTRL 1__22__	<>	27_WB 00
WCTRL 4__23__		26_WCTRL 2
WCTRL 5__24__  ()		25_WCTRL 0

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

37

**L0003 MIC**

MIC

SLOT 3). THE MEMORY INTERCONNECT MODULE (MIC) #L0003

THE MEMORY INTERCONNECT MODULE IS THE SECOND MAJOR PART OF THE CPU, IT HOUSES THE DATA ROUTING AND ALIGNMENT LOGIC, ADDRESS LOGIC, TRANSLATION BUFFER, DATA CACHE, EXECUTION BUFFERS, SEVERAL PC REGISTERS, VA (VIRTUAL ADDRESS) AND MA (MEMORY ADDRESS) REGISTERS, CMI LATCH, PA (PHYSICAL ADDRESS) MUX, MDR'S (MEMORY DATA REGISTER) AND WDR (WRITE DATA REGISTER)

ALL ADDRESSES AND DATA PASS THROUGH THIS MODULE, IT PROVIDES THE INTERFACE TO AND FROM THE DPM MODULE AND CMI BUS, IT ALSO DETECTS UNIBUS ADDRESSES AND SIGNALS THE UBI MODULE WITH A SIGNAL CALLED "UB REQ H" PIN <C45>.

DATA PARITY, CHECKING AND GENERATING LOGIC FOR BOTH CACHE AND TRANSLATION BUFFERS ARE LOCATED ON THE BOARD.

THE MIC IS CONNECTED TO THE W BUS, MBUS, AND CMI.

THE MICRODIAGNOSTICS ECKAC.EXE TAPE #2, ECKAB.EXE TAPE #1 AND ECKAL.EXE TB AND CACHE DIAGNOSTICS WILL TEST THE MIC MODULE.

GATEARRAYS: ACV,ADD,ADK,CAK,CMK,MDR,PRK,UTR



(TOP)

THE MIC MODULE #L0003

PINS>>>>

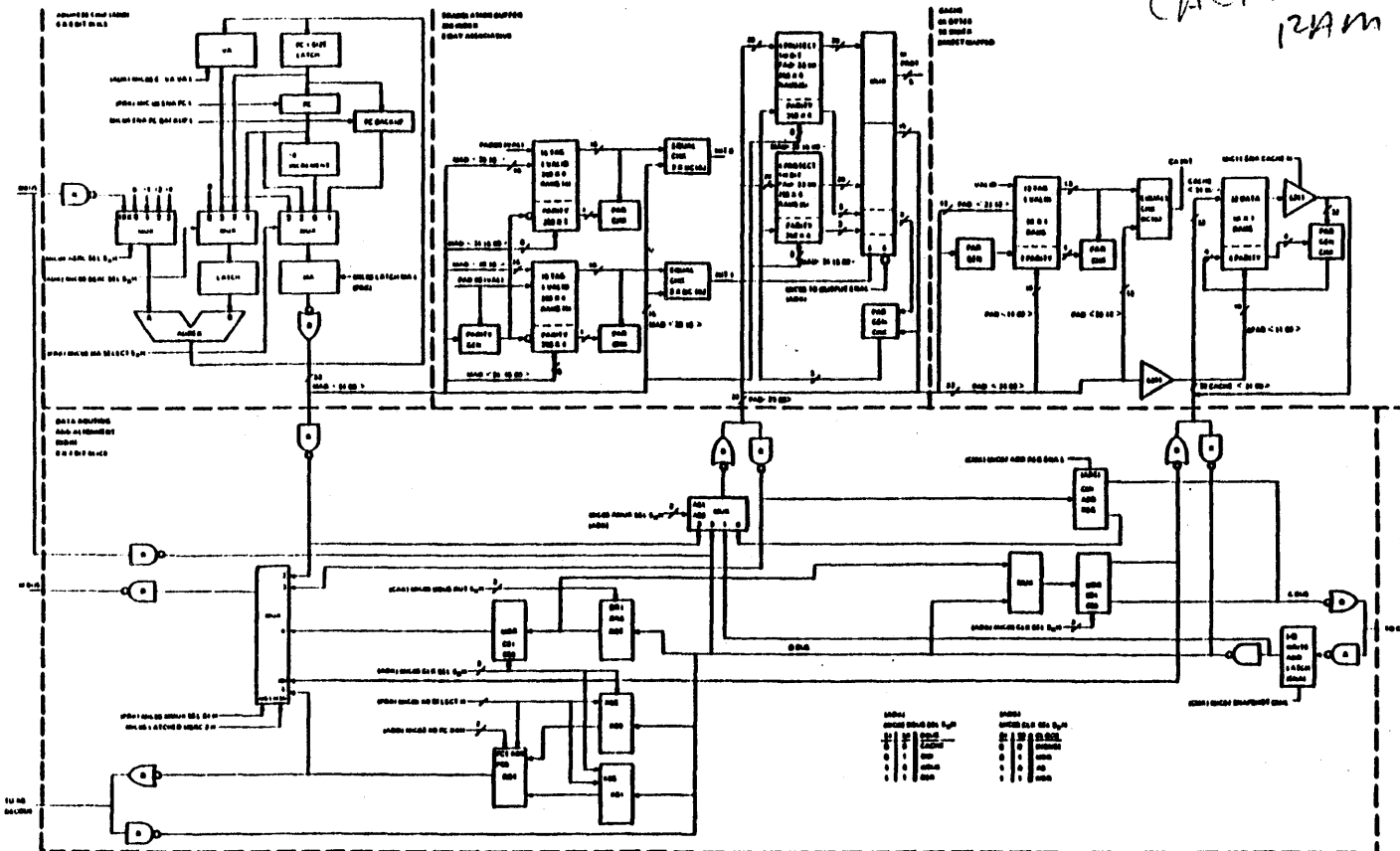
```
-----  
| CAK | | CMK/CML |  
-----  
| ADK | | UTR |  
-----  
| PRK | | ACV |  
-----
```

```
-----  
| ADD1 | | MDR6 | | MDR1 |  
-----  
| ADD2 | | MDR8 | | MDR4 |  
-----  
| ADD3 | | MDR5 | | MDR2 |  
-----  
| ADD4 | | MDR7 | | MDR3 |  
-----
```

4 GATE  
ADD  
LOGIC

T.B.  
RAM

CACHE  
RAM



DATA  
ROUTING  
MDR  
S

MIC Block Diagram

ACV: ACCESS VIOLATION CHIP

CAUSES CS PARITY ERROR MICRO TRAPS, FPA RESERVED OPERANDS, UNALIGNED DATA, PAGE BOUNDARY VIOLATIONS, AND ACCESS CONTROL VIOLATIONS FROM THE TB. THE ACV WORKS WITH THE UTR CHIP TO HANDLE MICRO-TRAPS. CONTAINS THE MEMORY MANAGEMENT ENABLE LATCH. (IF MEM. MGMT. IS OFF, THE ACV WILL MONITOR CS PARITY AND FPA RESERVED OPERANDS ONLY).

PART NUMBER: 19-14699

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE  
DPM MICRO'S ECKAB.EXE  
TB + CACHE ECKAL.EXE

MODULE: MIC

GATE ARRAY: ACV

## ACV

CS BUS 4	1	-----	48_D SIZE 0
M CLK ENABLE	2	0	0_47_B CLK
WB 24	3	<>	46_PROC INIT
PTE ACCESS CODES	4	---> AC 0	45_PAGE BOUNDARY VIOLATION
	5	AC 3	<>44_WB 25
	6	AC 1	<>43_WB 27
	7	---> AC 2	<>42_WB 26
MICRO TRAP	8	_0	41_PHASE 1
ACCESS CONTROL VIOLATION	9	---	40_FORCE 4A 09
LATCHED BUS 0	10	---	39_LATCHED WCTRL 0
LATCHED BUS 3	11	---	38_GROUND
VGA	12	---	37_LATCHED BUS 1
VCC	13	---	36_LATCHED WCTRL 3
TB VALID (V BIT)	14	---	35_GROUND
PTE CHECK OR PROBE	15	---	34_MAD 00
MICRO VECTOR 0	16	---	33_LATCHED WCTRL 1
MICRO VECTOR 1	17	---	32_MAD 01
ENCODED MICRO TRAP 1	18	_0	31_LATCHED WCTRL 5
ENCODED MICRO TRAP 0	19	_0	30_D CLK ENABLE
GROUND	20	---	29_MAD 02
ENCODED MICRO TRAP 2	21	_0	28_LATCHED WCTRL 2
PREFETCH	22	_0	27_LATCHED WCTRL 4
CS PARITY ERROR	23	---	26_D SIZE 1
LATCHED BUS 2	24	---	0_25_FP RESERVED OPERAND

THIS SIDE TOWARDS FINGERS ON BOARD

ADD: ADDRESS CHIPS

CONTAINS THE PC'S AND VA CIRCUIT, ENABLE LINES AND LOAD PATHS, PLUS A AND B SOURCE MUX SELECT CONTROLS. THE ADD CHIPS CONTAIN AN INTERNAL ADDER CAPABLE OF BUMPING THE PC OR VA BY 1, 2, 4, OR FROM THE N BUS.

4 CHIPS	CHIP	BIT SLICE
	ADD 1	<7-0>
	ADD 2	<15-8>
	ADD 3	<23-16>
	ADD 4	<31-24>

PART NUMBER: 19-14683

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE

MODULE: MIC GATE ARRAY: ADD (1 THROUGH 4)

	ADD	
ENABLE VA SAVE	1_0	48_MAD 06,14,22,30
PAGE BOUND, PAGE BOUND, N, N	2_1 0	47_MAD 05,13,21,29
BSRC SELECT S0	3_1	46_ICO, ICO, ICO, N
N	4_1	45_MAD 07,15,23,31
WB 06,14,22,30	5_1	44_MAD 04,12,20,28
WB 05,13,21,29	6_1	43_ENABLE VA
WB 07,15,23,31	7_1	42_ENABLE PC
ASRC SELECT S0	8_1	41_MA SELECT S1
(CARRY GENERATE)CG1, CG1, CG1, N	9_1	40_GRND, GRND, COMP MODE, COMP MODE
CG2, CG2, N, N	10_1	39_MA SELECT S0
BSRC SELECT S1	11_1	38_GROUND
VGA	12_1	37_8 CLK
VCC	13_1	36_ENABLE PC BACKUP
(CARRY PROGAGATE)CP, CP, CP, N	14_1	35_GROUND
WB 04,12,20,28	15_1	34_LATCH MA
ASRC SELECT S2	16_1	33_MAD 03,11,19,27
WB 00,08,16,24	17_1	32_N
WB 03,11,19,27	18_1	31_MAD 00,08,16,24
WB 02,10,18,26	19_1	30_XB PC 01, N, N, N
ASRC SELECT S1	20_1	29_(ICI)+3V, ICO, ICO, ICC
WB 01,09,17,25	21_1	28_MAD 02,10,18,26
(ACI)+3V, CX, CY, CZ	22_0 1	27_MAD 01,09,17,25
N	23_1	26_GRND, FORCE MA 09, GRND, GRND
(ID)GRND, +3V, +3V, +3V	24_1 ( )	25_XB PC 00, N, N, N

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

ADK: ADDRESS CONTROL CHIP

CONTAINS IPR COMPONENTS (MEMSCR'S) FOR TB, MEMORY MANAGEMENT, AND THE RANDOM FLIP FLOP. THE ADK WORKS WITH THE PRK, ADD AND MDR CHIPS. THE ADK DETECTS TB HITS AND MISSES.

NOTE: TO DISABLE HALF OF THE TB ON A LIVE VMS SYSTEM:

- 1). REMOVE ALL USERS FROM THE SYSTEM TEMPORARILY
- 2). TYPE ^P ON THE CONSOLE TERMINAL
- 3). TYPE: >>>D/I 24 D (FOR GROUP 0) OR D/I 24 A (FOR GROUP 1)
- 4). TYPE: >>>C

PART NUMBER: 19-14700

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE

MODULE: MIC

GATE ARRAY: ADK

		ADK	
AMUX SELECT S1	1	-----	0_48_B CLK
DS1	2	0	I__47_AMUX SELECT S0
WB 26	3	<>	I__46_DST RMODE
WB 25	4	<>	I__45_MMUX SELECT S1
WB 24	5	<>	Io_44_WRITE VECTOR OCCURED
WB 27	6	<>	Io_43_SNAPSHOT CMI
TB GROUP 0 WR (HIT)	7		I__42_PSL CH (COMPATABILITY MODE)
TB HIT 0	8		I__41_RTUT DIN4
TB GROUP 1 WR (HIT)	9		Io_40_STATUS VALID
TB HIT 1	10		I__39_M CLK ENABLE
ENABLE VA	11	0	I__38_GROUND
VGA	12		I__37_D CLK ENABLE
VCC	13		I__36_CS BUS 4
LATCHED WCTRL 2	14		I__35_GROUND
BSRC SEL S0	15		I__34_TB PARITY ENABLE
LATCHED WCTRL 5	16		I__33_LATCHED BUS 3
LATCHED WCTRL 0	17		I__32_LATCHED BUS 0
LATCHED WCTRL 3	18		Io_31_PTE CHECK
BSRC SEL S1	19		I__30_LATCHED BUS 1
LATCHED WCTRL 1	20		Io_29_INVALID PREFETCH
LATCHED WCTRL 4	21		I__28_COMPATABILITY MODE
CLK SELECT S0	22		I__27_LATCHED BUS 2
CLK SELECT S1	23		Io_26_TB OUTPUT ENABLE
PHASE 1	24	()	I__25_DEUS SELECT S0

THIS SIDE TOWARDS FINGERS ON BOARD

CAK: CACHE CONTROL CHIP

CONTROLS THE ENABLING AND DISABLING OF CACHE, THE TRANSFER OF DATA TO AND FROM THE MDR CHIPS, AND CACHE HIT VALIDATION.

NOTE: TO DISABLE CACHE TYPE: >>>D/I 25 1  
TO RE-ENABLE CACHE TYPE: >>>D/I 25 0

ON A LIVE VMS SYSTEM:

- 1). REMOVE ALL USERS FROM THE SYSTEM TEMPORARILY
- 2). TYPE ^P ON THE CONSOLE TERMINAL
- 3). TYPE: >>>D/I 25 1 (OFF) OR D/I 25 0 (ON)
- 4). TYPE: >>>C

PART NUMBER: 19-14701

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE  
TB + CACHE ECKAL.EXE

MODULE: MIC GATE ARRAY: CAK

		CAK	
MAD 00	1	-----	48_DBUS ROT S0
WB 27	2	<>  o	I_47_LATCHED #CTRL 0
B CLK	3	o	I_46_LATCHED #CTRL 4
WB 26	4	<>	I_45_LATCHED #CTRL 2
GRND	5		I_44_LATCHED BUS 1
WB 25	6	<>	I_43_LATCHED BUS 0
CACHE INIT	7	o	I_42_LATCHED BUS 2
WB 24	8	<>	I_41_LATCHED BUS 4
CACHE HIT	9		I_40_LATCHED BUS 3
N	10		I_39_LATCHED #CTRL 3
SNAPSHOT CMI	11	o	I_38_GROUND
VGA	12		I_37_LATCHED #CTRL 1
VCC	13		I_36_LATCHED #CTRL 5
CACHE DATA PARITY ERROR	14	o	I_35_GROUND
CACHE TAG PARITY ERROR	15		I_34_D CLK ENABLE
GRND	16		I_33_DST RMODE
N	17		I_32_D SIZE 0
CACHE GROUP 0 WR	18		I_31_D SIZE 1
ENABLE BYTE 0	19	o	I_30_STATUS VALID
CACHE VALID 0	20		I_29_INVALID PREFETCH
ENABLE BYTE 3	21	o	I_28_M CLK ENABLE
ENABLE BYTE 1	22	o	I_27_I/O ADDRESS
ENABLE BYTE 2	23	o	I_26_DBUS ROT S1
MUX SELECT S1	24	( )	I_25_MAD 01

THIS SIDE TOWARDS FINGERS ON BOARD

CMK/CML: CPU MEMORY INTERCONNECT CHIP

MONITORS AND CONTROLS SIGNALS TO AND FROM THE CMI AND STALLS THE MICROCODE ON CERTAIN CONDITIONS.

NOTE: THIS IS THE ONLY CHIP THAT CONTROLS THE CPU'S ACCESS TO THE CMI BUS. OPERATION OF THE CMK/CML CHIP CAN BE VERIFIED BY DOING CONSOLE MODE DEPOSITS AND EXAMINES OF MAIN MEMORY WITH CACHE DISABLED AND AGAIN FROM RDM MODE WHICH DOES NOT USE THE CMK/CML CHIP.

PART NUMBER: 19-14697

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE

MODULE: MIC GATE ARRAY: CMK

		CMK		
	B CLK	1_0	-----	48_D CLK ENABLE
ADDRESS REGISTER ENABLE	2_0	o		I<>47_CMI DATA 25
CMI DATA	27_3	<>		Io_46_INVALID PREFETCH
LATCHED BUS	3_4			Io_45_CORR DATA INTERRUPT
LATCHED BUS	1_5			Io_44_WRITE VECTOR OCCURRED
LATCHED BUS	2_6			I__43_STATUS 0
	N	7_7		Io_42_CMI STATUS 00
CMI DATA	31_8	<>		Io_41_GRANT STALL
CS BUS	4_9			Io_40_CMI STATUS 01
M CLK ENABLE	10_10			Io_39_STATUS VALID
CMI DATA	29_11	<>		I__38_GROUND
VGA	12_12			I<>37_CMI DATA 25
VCC	13_13			I__36_STATUS 1
LATCHED BUS	0_14			I__35_GROUND
CMI DATA	28_15	<>		I__34_INHIBIT CMI
CMI DATA	30_16	<>		Io_33_CACHE INTERRUPT
MAD 01	17_17			I__32_CACHE HIT
MAD 00	18_18			I__31_WAIT
CMI CPU PRIORITY	19_19	o		I__30_PHASE 1
DST RMODE	20_20			I__29_UR INTERRUPT GRANT
DSIZE 0	21_21			Io_28_CMI DBZ
DSIZE 1	22_22			Io_27_ENABLE CMI
MMUX SEL S1	23_23			Io_26_MICRO SEQUENCER INIT
CMI HOLD	24_24	o	( )	Io_25_SNAPSHOT CMI

THIS SIDE TOWARDS FINGERS ON BOARD

**MDR: MEMORY DATA REGISTER CHIPS**

CONTAINS THE EXECUTION BUFFERS, WRITEDATA REGISTER, MEMORY DATA REGISTER, PHYSICAL ADDRESS MUX AND CONTROL LOGIC FOR ROUTING DATA IN AND OUT, TO AND FROM THE CMI, \* BUS AND \* BUS.

8 CHIPS	CHIP	BITS
	MDR 1	0,8,16,24
	MDR 2	1,9,17,25
	MDR 3	2,10,18,26
	MDR 4	3,11,19,27
	MDR 5	4,12,20,28
	MDR 6	5,13,21,29
	MDR 7	6,14,22,30
	MDR 8	7,15,23,31

**BUS DEFINITION: D BUS = DATA BUS (AN INTERNAL BUS INSIDE THE MDR CHIPS)**

**PART NUMBER: 19-14681**

**BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE**

**MODULE: MIC**

**GATE ARRAY: MDR (1 THROUGH 8)**

		MDR		
	CLK SELECT S1	1	-----	48 MBUS ENABLE
	CLK SELECT S0	2	0	Io_47_MBUS 24,25,26,27,28,29,30,31
	B CLK	3	0	Io_46_MBUS 16,17,18,19,20,21,22,23
ADDRESS REGISTER	ENABLE	4	0	I_45_MAD 24,25,26,27,28,29,30,31
	ENABLE CMI	5	0	Io_44_MBUS 08,09,10,11,12,13,14,15
	XB PC 00	6		I_43_MMUX SELECT S1
	XB PC 01	7		Io_42_MBUS 00,01,02,03,04,05,06,07
	XB SELECT	8		I_41_LATCHED MSRC 2
XBUF 00,01,02,03,04,05,06,07		9		I_40_MAD 08,09,10,11,12,13,14,15
	SNAPSHOT CMI	10	0	I_39_MAD 16,17,18,19,20,21,22,23
XBUF 08,09,10,11,12,13,14,15		11		I_38_GROUND
	VGA	12		I_37_MAD 00,01,02,03,04,05,06,07
	VCC	13		I_36_+3V, ALL OTHERS GROUNDED
	DBUS ROT S0	14		I_35_GROUND
	DBUS ROT S1	15		I_34_DBUS SELECT S1
CMI 24,25,26,27,28,29,30,31	<>	16		I_33_DBUS SELECT S0
CMI 16,17,18,19,20,21,22,23	<>	17		I_32_PAD 16,17,18,19,20,21,22,23
CMI 08,09,10,11,12,13,14,15	<>	18		I_31_PAD 08,09,10,11,12,13,14,15
CMI 00,01,02,03,04,05,06,07	<>	19		I_30_N,N,PAD 02,03,04,05,06,07
WB 00,01,02,03,04,05,06,07		20		I_29_AMUX SELECT S1
WB 08,09,10,11,12,13,14,15		21		I_28_AMUX SELECT S0
WB 16,17,18,19,20,21,22,23		22		I_27_CACHE 00,01,02,03,04,05,06,07
WB 24,25,26,27,28,29,30,31		23		I_26_CACHE 08,09,10,11,12,13,14,15
CACHE 24,25,26,27,28,29,30,31		24	( )	I_25_CACHE 16,17,18,19,20,21,22,23

-----  
THIS SIDE TOWARDS FINGERS ON BOARD



PRK: PREFETCH CONTROL CHIP

USED IN CONJUNCTION WITH THE ADD, AND MDR CHIPS TO MONITOR USAGE OF THE EXECUTION BUFFERS. WHEN EXECUTION BUFFERS ARE EMPTY, OR A NEW ADDRESS IS PLACED IN THE PC, THE PRK CHIP FORCES PREFETCHING OF A NEW INSTRUCTION FROM MEMORY USING THE ADDRESS IN THE PC (NEW ADDRESS IN PC) OR PC+4 (EX. BUF. EMPTY).

PREFETCHING IS INDEPENDANT OF THE MICROCODE AND WILL HAPPEN WHENEVER AN XB IS EMPTY AND THERE IS NO BUS CYCLE IN PROGRESS. THE PRK WILL STALL THE M CLOCK WHEN BOTH XB'S ARE EMPTY AND THE CPU ATTEMPTS AN IRD1. (THIS CAN OCCUR WHEN A DEVICE IS TYING UP THE CMI WITH TRANSFERS AND THE PRK HAS TO WAIT FOR COMPLETION BEFORE IT CAN PREFETCH (CPU HAS A PRIORITY OF 0)). THE PRK WILL ALSO HAVE TO STALL WHENEVER THE PC GETS A NEW ADDRESS SUCH AS A BRANCHING INSTRUCTION OR A NEW PROGRAM. THIS ALLOWS TIME TO PERFORM THE FIRST PREFETCH FROM THE NEW PC PRIOR TO THE START OF NORMAL EXECUTION.

PREFETCHES ARE LONGWORDS ONLY!

PART NUMBER: 19-14698

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE

MODULE: MIC

GATE ARRAY: PRK

		PRK		
LATCHED MSRC 1	1	-----	48	LATCHED MSRC 2
SNAPSHOT CMI	2	o	47	LATCHED MSRC 3
B CLK	3		46	MA SELECT S1
PHASE 1	4		45	LATCHED MSRC 0
MMUX SELECT S1	5		44	LATCHED MSRC 4
ENABLE ACV STALL (STOPS M CLK)	6		43	MA SELECT S0
MICRO SEQUENCER INIT	7		42	IRD1
STALL	8		41	ISIZE 0
LATCH MA	9		40	MIC LOAD DSR
PREFETCH	10		39	ISIZE 1
ENABLE VA SAVE	11		38	GROUND
VGA	12		37	XB PC 00
VCC	13		36	DSI RMODE
MICRO TRAP	14		35	GROUND
M CLK ENABLE	15		34	LATCHED BUS 0
D CLK ENABLE	16		33	LATCHED BUS 1
LATCHED BUS 3	17		32	PSL CM (COMPATABILITY MODE)
STATUS VALID	18		31	LATCHED BUS 2
LATCHED WCTRL 1	19		30	XB PC 01
XB 1 IN USE	20		29	LATCHED BUS 4
XB 0 IN USE	21		28	LATCHED WCTRL 0
LATCHED WCTRL 3	22		27	LATCHED WCTRL 2
XB SELECT	23		26	LATCHED WCTRL 5
ENABLE PC	24	( )	25	LATCHED WCTRL 4

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

MONITORS THE MACHINE CONDITIONS THAT CAN CAUSE A MICRO-TRAP. GENERATES MICRO-VECTOR ADDRESSES, AND DECODES THE HIGHEST PRIORITY TRAP CONDITION. THE UTR RECEIVES ENCODED MICRO TRAP INPUTS FROM VARIOUS HARDWARE COMPONENTS AND DECODES THEM INTO THEIR APPROPRIATE MICRO-VECTOR ADDRESSES TO BE PLACED ON THE CONTROL STORE ADDRESS LINES WHEN PERFORMING A MICRO TRAP.

PART NUMBER: 19-14702

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE

MODULE: MIC GATE ARRAY: UTR

	UTR	
ENCODED MICRO TRAP 1__1_0	-----0_48	ADDRESS REGISTER ENABLE
ENCODED MICRO TRAP 2__2_0   0	__47	ACCESS CONTROL VIOLATION
MICRO TRAP__3_0	__46	TB DATA PARITY ERROR
MICRO VECTOR 3__4__	__45	LATCHED BUS 3
PTE CHECK OR PROBE__5__	__44	M BIT
MICRO VECTOR 1__5__	__43	TB PARITY ENABLE
MICRO VECTOR 0__7__	__42	TB HIT 1
MICRO VECTOR 2__8__	__41	TB HIT 0
GENERATE DEST INHIBIT__9_0	__40	TB TAG 1 PARITY ERROR
DU SERVICE__10_0	0_39	WRITE BUS ERROR INTERRUPT
MSRC XB__11__	__38	GROUND
VGA__12__	__37	TB TAG 0 PARITY ERROR
VCC__13__	__36	LATCHED WCTRL 1
ENCODED MICRO TRAP 0_14_0	__35	GROUND
WCTRL HHLXXX__15_0	__34	LATCHED WCTRL 0
XB SELECT__16__	__33	LATCHED WCTRL 2
XB 0 IN USE__17_0	__32	D CLK ENABLE
PROCESSOR INIT__18_0	<>31	WB 24
RTUT DINH__19__	<>30	WB 25
STATUS 0__20__	<>29	WB 26
STATUS 1__21__	<>28	WB 27
STATUS VALID__22_0	__27	PHASE 1
XB 1 IN USE__23_0	0_26	PREFETCH
B CLK__24_0   ( )	__25	INHIBIT CHI

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

**L0004 UBI**

UBI

SLOT 4). THE UNIBUS INTERFACE MODULE (UBI) #L0004

THE UNIBUS INTERFACE MODULE IS MUCH LIKE ANY UNIBUS ADAPTER WITH THE EXCEPTION OF HAVING ADDITIONAL LOGIC ON IT TO HANDLE COMMUNICATIONS FOR THE TU-53 AND CONSOLE AND ALSO HANDLING ALL INTERRUPTS WITHIN THE CPU, ALL UNIBUS AND MASSBUS DEVICES INTERRUPT VIA THE UBI. THE UBI CONTAINS POWER FAIL LOGIC, THE T.O.Y. CLOCK AND CHARGING CIRCUIT, THE UNIBUS DATA LATCH, 3 BUFFERED DATA PATHS, 1 DIRECT DATA PATH, BYTE SWAPPING LOGIC, ADDRESS BUFFERS AND MAPS.

THE UBI IS CONNECTED TO THE W BUS, CMI, AND UNIBUS.

THE DPM MICRO-DIAGNOSTIC ECKA6.EXE WILL TEST THE COMMUNICATIONS SECTIONS AND THE LEVEL 3 DIAGNOSTIC ECCBA.EXE TAPE #6 WILL TEST THE ADAPTER SECTION.

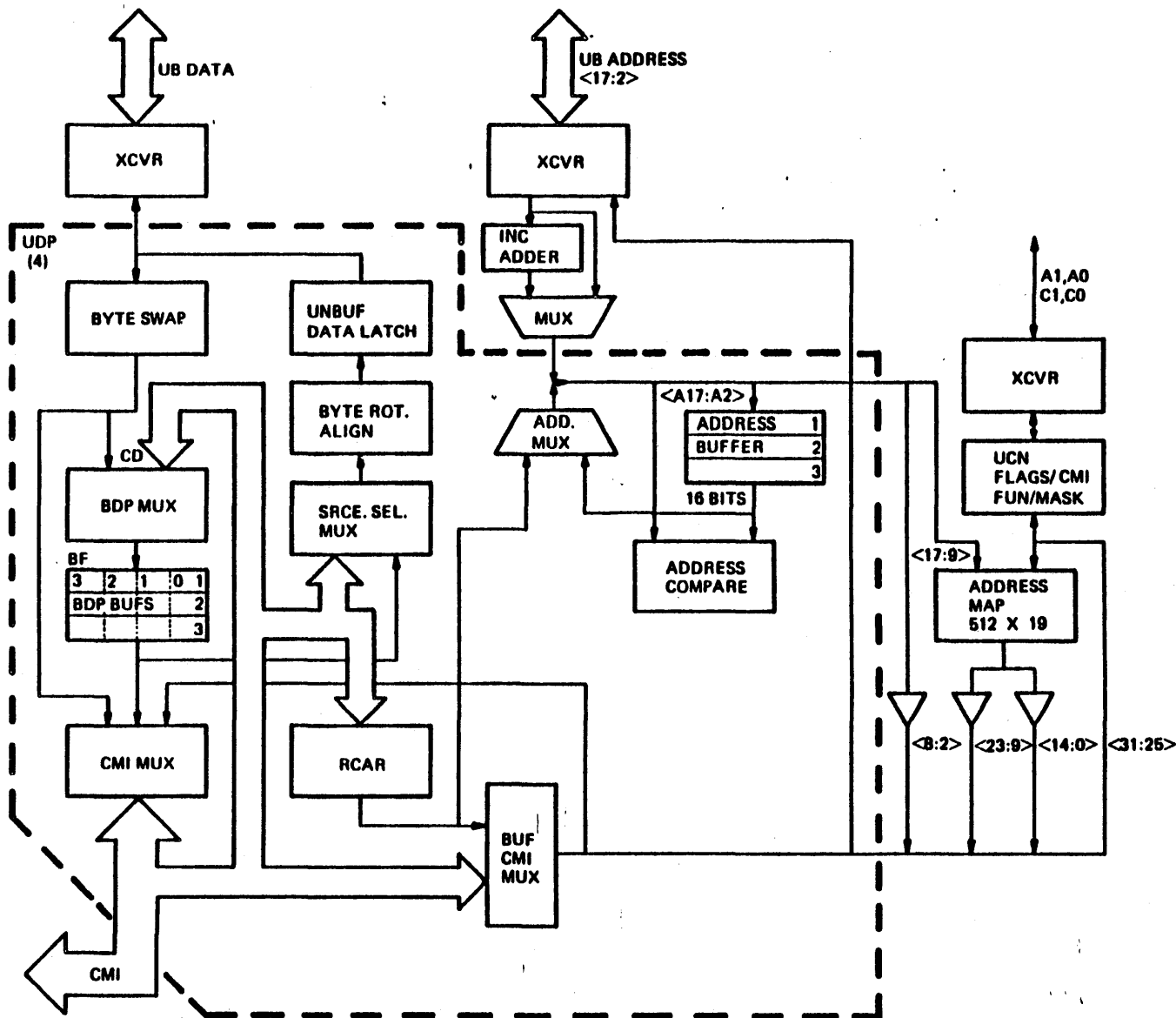
THE SECOND UNIBUS OPTION (SUB MODULE) WILL ALSO BE TESTED BY ECCBA.EXE

GATEARFAYS: CON,INT,UCN,UDP

(TOP)  
THE UBI MODULE #L0004

PINS>>>>

```
-----  
| UCN |  
-----  
-----  
| UDP1 |  
-----  
-----  
| UDP2 |  
-----  
-----  
| UDP4 |  
-----  
-----  
| UDP3 |  
-----  
-----  
| INT |  
-----  
-----  
| CON1 | (TU-58)  
-----  
-----  
| CON2 | (CONSOLE)  
-----
```



UNIBUS Interface Block Diagram

CON: CONSOLE CHIP

THE CON CHIPS CONVERT SERIAL DATA FROM THE TU-58 OR THE  
CONSOLE TERMINAL TO PARALLEL DATA FOR THE W BUS, OR  
PARALLEL TO SERIAL IF ROUTING IN THE OPPOSITE DIRECTION.

PART NUMBER: 19-14685

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE  
DW750 MACRO ECCBA.EXE (LEVEL 3)

MODULE: UBI GATE ARRAY: CON (TU58 AND CONSOLE)

## CON (TU58, CONSOLE)

WCTRL 0,+3V	1	-----	48	TU/CON DONE SYNC
MICRO SEQUENCER INIT	2	o	47	CKDO (INTERNAL CLOCK SIGNALS)
WCTRL 5	3		46	WCTRL 4
WCTRL 2	4		45	+3V
GRND, WCTRL 0	5		44	TU/CON T READY SYNC
N, HALT DET BR SYNC	6		43	CLCO (INTERNAL CLOCK SIGNALS)
TU/CON DONE SYNC H	7	o	42	M CLK
+3V, FRONT PANEL LOCKED	8		41	TU58 INT L, SERIAL LINE INT L
TU/CON SERIAL INPUT	9		40	WB 25
CLD1 (INTERNAL CLOCK SIGNALS)	10		39	WB 24
TU/CON T READY SYNC	11		38	GROUND
VGA	12		37	D CLK ENABLE
VCC	13		36	WCTRL 3
GRND, RD INTERRUPT INHIBIT	14		35	GROUND
WB 16	15	<>	34	CLCI (INTERNAL CLOCK SIGNALS)
N	16	o	33	GRND, HALT DET SYNC
GRND	17		32	WCTRL 1
TU/CON BAUD RATE CLOCK	18		31	M CLK
WB 19	19	<>	30	WB 22
WB 17	20	<>	29	BREAK CLK, N
WB 18	21	<>	28	GRND, INSTR FETECH
WB 21	22	<>	27	SET BREAK, CON HALT
WB 20	23	<>	26	N
WB 23	24	<>	25	EIA TU/CON SERIAL OUTPUT

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

# 54

INT: INTERRUPT CHIP

THE INTERRUPT CHIP ENABLES THE HANDLING OF ALL INTERRUPT REQUESTS BOTH MASSBUS AND UNIBUS, CONTAINS PSL BITS <22-26 and IPL>, PERFORMS INTERRUPT ARBITRATION, ISSUES BUS GRANTS, AND INSERTS VALUES ON THE MICRO-VECTOR LINES.

PART NUMBER: 19-14704

BEST DIAGNOSTICS: DPM MICRO'S ECKAB.EXE  
DW750 MACRO ECC9A.EXE (LEVEL 3)

MODULE: UBI GATE ARRAY: INT

## INT

WRITE BUS ERROR INTERRUPT__1_0	-----	0_48_TIMER INTERRUPT
+3V__2_0   0		0_47_INTERRUPT PENDING
SPFI (SYNC POWER FAIL INT.)__3_0		__46_UB INTERRUPT GRANT
CORRECTED DATA INTERRUPT__4_0		__45_SBR4 (SYNCHRONOUS BR)
WCTRL 4__5__		__44_HPBG5 (HIGHEST PRIORITY BG)
*CTRL 5__6__		__43_HPBG4 (HIGHEST PRIORITY BG)
PHASE 1__7__		__42_SBR5 (SYNCHRONOUS BR)
WB 22(=PSL 22)__8<>		__41_HPBG5 (HIGHEST PRIORITY BG)
WB 23(=PSL 23)__9<>		__40_SBR7 (SYNCHRONOUS BR)
PROCESSOR INIT__10_0		__39_SBR6 (SYNCHRONOUS BR)
WCTRL 2__11__		__38_GROUND
VGA__12__		__37_SYNCHR RESET BG
VCC__13__		0_36_B CLK
WCTRL 1__14__		__35_GROUND
*CTRL 3__15__		__34_M CLK ENABLE
WB 25(=PSL 25)_16<>		<>33_WB 16 (IPL)
WB 24(=PSL 24)_17<>		<>32_WB 17 (IPL)
WB 26(=PSL 26)_18<>		<>31_WB 19 (IPL)
WCTRL 0__19__		0_30_SERIAL LINE INTERRUPT(CONSOLE)
MICRO VECTOR 0__20__		__29_D CLK ENABLE
MICRO VECTOR 2__21__		<>28_WB 18 (IPL)
MICRO VECTOR 1__22__		<>27_WB 20 (IPL)
MICRO TRAP__23_0		__26_PTE CHECK OR PROBE
MICRO VECTOR BRANCH__24__   ( )		0_25_DO SERVICE

-----

THIS SIDE TOWARDS FINGERS ON BOARD



UCN: UNIBUS DATA PATH CONTROL CHIP

THE UCN CHIP CONTROLS THE UDP CHIP FUNCTIONS, ENABLES UBI ARBITRATION FOR THE CMI, ISSUES AND MONITORS UNIBUS CONTROL SIGNALS AND CMI STATUS LINES FOR USE BY THE UBI MICROCODE. THE UBI IS ROM CONTROLLED AND THE UCN CHIP PLACES THE PROPER MICRO-ADDRESS ON THE UBI MICRO-ADDRESS LINES AFTER DECIDING WHAT FUNCTION NEEDS TO BE DONE.

PART NUMBER: 19-14693

BEST DIAGNOSTICS: DW750 MACRO FCCBA.EXE (LEVEL 3)

MODULE: UBI

GATE ARRAY: UCN

	UCN	
UBI BUFF CMI 31__1__	-----	__48_P8
MSYN__2__   0		__47_TIME COUNT (TIMEOUT)
UBI UNIBUS ADDRESS 11__3__		o__46_CMI DBBZ (DATA BUS BUSY)
UBI UNIBUS ADDRESS 08__4__		o__45_CMI STATUS 00
ADDRESS = UNIBUS (ADDU)__5__		o__44_CMI STATUS 01
ADDRESS = CMI (ADDC)__6__		__43_SSYN
ENABLE ARB REQUEST__7__		__42_UNIBUS INIT
C1__8__		__41_UBI BUFF CMI 00
ARBITRATION OK__9__o		__40_UBI BUFF CMI 29
UCR A2(MICRO CONTROL ROM)__10__o		__39_UBI BUFF CMI 30
UBI MATCH__11<>		__38_GROUND
VGA__12__		o__37_UCR A1(MICRO CONTROL ROM)
VCC__13__		__36_INTERRUPT
UBI BUFF CMI 28__14__		__35_GROUND
C0__15__		__34_UBI BUFF CMI 25
UCR A3(MICRO CONTROL ROM)__16__o		__33_BUT 0
A0__17__		__32_UNIBUS ADDRESS 10
A1__18__		o__31_B CLK
BUT 1__19__		__30_UNIBUS ADDRESS 09
BUT 2__20__		o__29_MAP CONTROL OUT ENABLE
UCR A0(MICRO CONTROL ROM)__21__o		__28_UBI LATCH DATA PATH SEL 1
UBI BUFF CMI 27__22__		__27_UBI LATCH DATA PATH SEL 0
SC 1__23__		__26_SC 0
UBI BUF CMI 26__24__   ( )		__25_UBI LATCH OFFSET

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

UDP: UNIBUS DATA PATHS CHIP

COMPLETE UNIBUS DATA PATHS CONTAINED IN THESE CHIPS,  
3 BUFFERED AND 1 DIRECT, BYTE SWAPPING AND OFFSET LOGIC.  
PROVIDES A PATH FOR ALL ADDRESSES AND DATA BETWEEN THE  
CMI AND UNIBUS.

4 CHIPS	CHIP	BITS
	UDP 1	0,1,8,9,16,17,24,25
	UDP 2	2,3,10,11,18,19,26,27
	UDP 3	4,5,12,13,20,21,28,29
	UDP 4	6,7,14,15,22,23,30,31

PART NUMBER: 19-14692

BEST DIAGNOSTICS: DW750 MACRO ECC8A.EXE (LEVEL 3)

MODULE: UBI

GATE ARRAY: UDP (1 THROUGH 4)

UDP (1 THROUGH 4)

UNIBUS DATA 09,11,13,15__1<>	-----<>48_UBI BUFF CMI 08,10,12,14
UNIBUS DATA 08,10,12,14__2<>	o  <>47_UBI BUFF CMI 09,11,13,15
UNIBUS DATA 00,02,04,06__3<>	<>46_CMI DATA 09,11,13,15
UNIBUS DATA 01,03,05,07__4<>	<>45_CMI DATA 08,10,12,14
N,ADDC,ADDC,ADDC(ADDRESS=CMI)__5__	<>44_CMI DATA 01,03,05,07
GRND,+3V,GRND,GRND (ID)__6__	<>43_UBI BUFF CMI 01,03,05,07
B CLK__7__o	<>42_CMI DATA 00,02,04,06
A1__8__	<>41_UBI BUFF CMI 00,02,04,06
A0__9__	__40_BDPC 2 (PORT CONTROL)
BDPC 1(BUFF DATA PATH)__10__	<>39_CMI DATA 16,18,20,22
BDPC 0(BUFF DATA PATH)__11__	__38_GROUND
VGA__12__	<>37_CMI DATA 17,19,21,23
VCC__13__	<>36_UBI BUFF CMI 16,18,20,22
UBI DBBZ__14__o	__35_GROUND
UBI PREV DBBZ__15__	<>34_CMI DATA 25,27,29,31
UNIBUS ADDRESS 03,10,12,14__16<>	<>33_CMI DATA 24,26,28,30
UNIBUS ADDRESS 16,08,05,07__17<>	<>32_UBI BUFF CMI 17,19,21,23
UNIBUS ADDRESS 15,17,04,06__18<>	<>31_UBI BUFF CMI 25,27,29,31
SC 1 (SLAVE CONTROL)__19__	<>30_UBI BUFF CMI 24,26,28,30
UNIBUS ADDRESS 02,09,11,13__20<>	__29_LATCH DATA PATH SELECT 1
SC 0 (SLAVE CONTROL)__21__	__28_ADDC,ADDU,ADDU,ADDU
PRIC 0 (PORT CONTROL)__22__	__27_LATCH DATA PATH SELECT 0
UBI LATCH__23<>	__26_LATCH OFFSET
PRIC 1 (PORT CONTROL)__24__	( )  __25_PRTC 2 (PORT CONTROL)

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

L0010 SUB

(TOP)  
THE SUB MODULE #L0010  
SECOND UNIBUS ADAPTER

PINS>>>>

-----  
UDP2

-----  
UCN

-----  
UDP4

-----  
UDP3

-----  
UDP1

## ACTION 1

INSPECT PARTS

The basic DW750 option consist of the following hardware parts. Check that none are missing or damaged before you procede.

<u>QUANTITY</u>	<u>DESCRIPTION</u>
1	L0010 Second UNIBUS(SUB) Module
1	Ribbon cable assembly consisting of: (3) 40-conductor EC06 ribbon cables, tie wrapped and formed.
1	M9014 Transition module
1	M9302 UNIBUS Terminator

There should be an expansion box, and possibly an expansion cabinet. The expansion box and, or cabinet are not part of the DW750 option, but should have been ordered separately.

## ACTION 2

INSTALL EQUIPMENT

## 2.1 CHECK HARDWARE REVISION LEVEL AND POWER SYSTEM DOWN

2.1.1 If VMS is running bring it down in an orderly fashion by either having the customer bring it down, or with his permission typing the following command.

EX:

```
S @sysssystem:shutdown
~~~~~
```

2.1.2 Examine the CPU hardware revision level to assure compatatibility between the option and the CPU. If the CPU is not at the correct revision level, do not procede with this option installation until the CPU is updated and checked out. The following example shows you how to check the CPU rev.

EX:

```
>>> E/I 3E
~~~~~
```

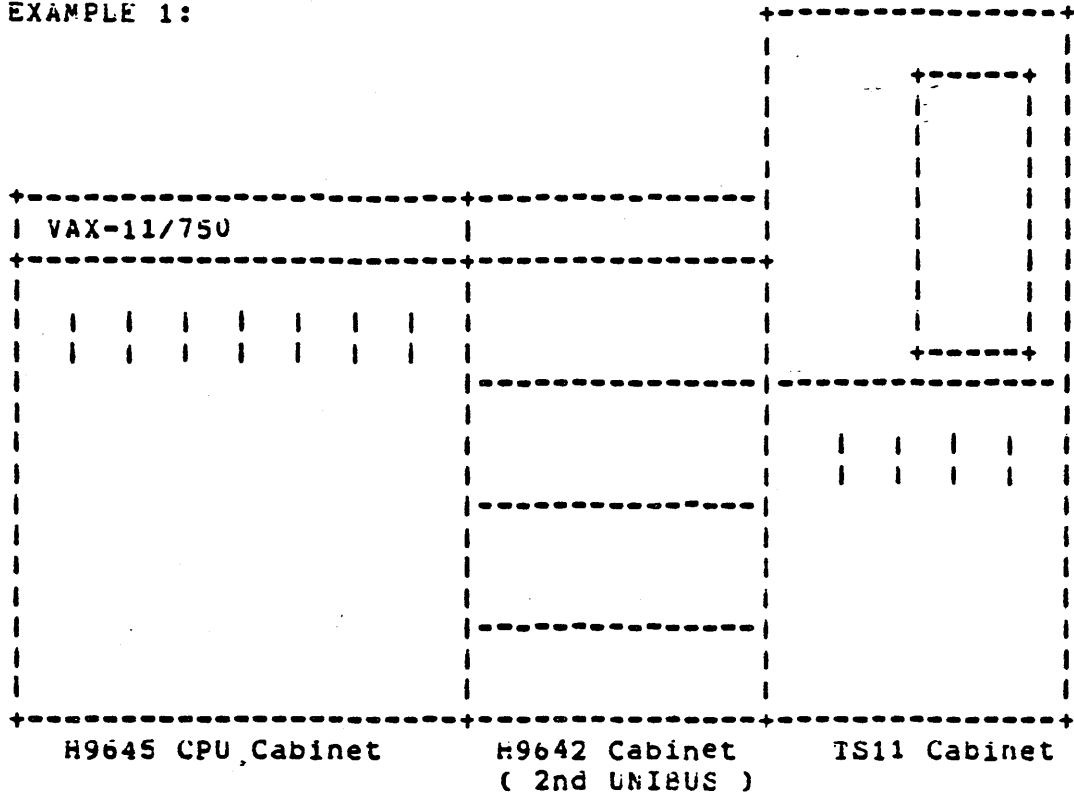
```
      I 0000003E      02005E30
      (for systems with 16K arrays)
      CR
      I 0000003E      02005E48
      (for systems with 1meg arrays)
```

2.1.3 Power the system off using the key switch.

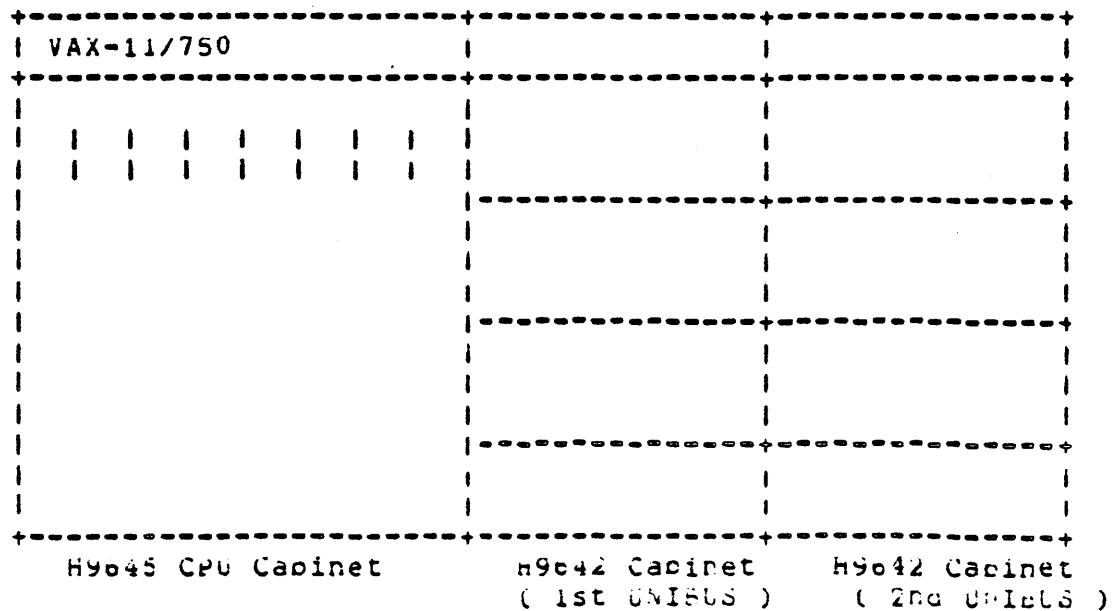
## 2.2 INSTALL EXPANSION BOX OR CABINET

2.2.1 Install expansion box or cabinet per appropriate installation documentation included with the option. Expansion cabinet should be installed to the right of the CPU cabinet per the following examples.

EXAMPLE 1:



EXAMPLE 2:



# 61

- 2.3 SET UP VELOSTAT KIT *PN 29-11762*
- 2.3.1 Unfold the VELOSTAT mat to full size (24x24).
- 2.3.2 Attach the 15' ground cord to the VELOSTAT snap fastener on the mat, and the alligator clip of the ground cord to a good ground on the VAX-11/750.
- 2.3.3 Attach the wrist strap to either wrist and the alligator clip to a convenient portion of the mat.
- 2.4 UNPACK THE L0010 MODULE
- 2.4.1 Place the L0010 module while still in the box on the VELOSTAT mat.
- 2.4.2 Remove the module from the box and protective covering and lay it flat on the VELOSTAT mat. This will bring the module to the same potential as the CPU and eliminate static discharge damage.
- 2.5 INSTALL DW750 OPTION
- 2.5.1 With the wrist strap still attached to your wrist install the L0010 module in a CMI option slot. The first CMI option slot is recommended (VAX-11/750 slot number 7) to alleviate cabling problems.
- 2.5.2 Remove grant jumpers from backplane slot where L0010 is installed. No jumpers need to be added for the DW750 option because it has fixed addresses, and a fixed CMI Arbitration Level of 3.

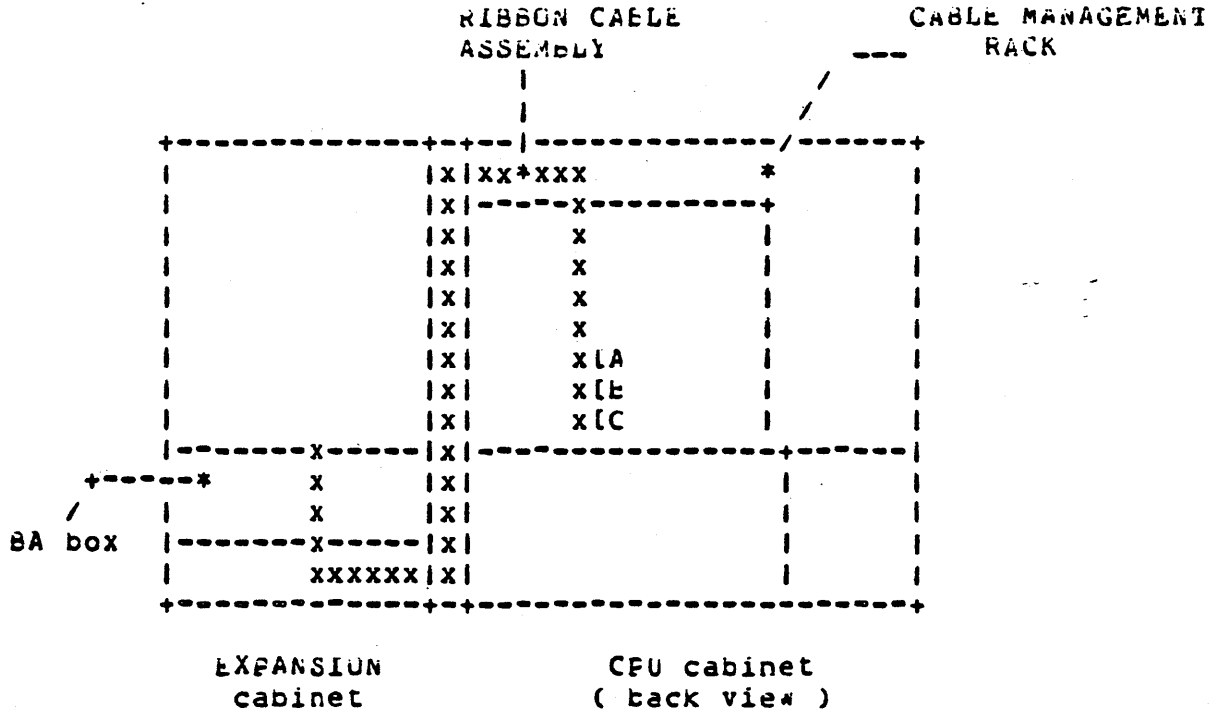
## NOTE ----

RH750s START WITH CMI ARBITRATION LEVEL (3),  
IF YOU HAVE ONE OR MORE IN YOUR SYSTEM YOU  
MUST MOVE THEM DOWN ONE CMI ARBITRATION LEVEL.

### EX:

	WITHOUT DW750 INSTALLED	
RH750#0	ADDRESS F28000	CMI ARB LEVEL 3
RH750#1	ADDRESS F2A000	CMI ARB LEVEL 2
	WITH DW750 INSTALLED	
RH750#0	ADDRESS F28000	CMI ARB LEVEL 2
RH750#1	ADDRESS F2A000	CMI ARB LEVEL 1

- 2.5.3 Connect the three ribbon cables to backplane slots B and C as in the MASSBUS option.
- 2.5.4 Route the cable assembly up the backplane to the cable management rack, and then to the left. Next route it between the VAX-11/750 CPU cabinet, and the expansion cabinet, then across the bottom of the expansion cabinet and up the back to the BA box. See diagram.



- 2.5.5 Install the M9014 on the end of the cable, and install it in the UNIBUS IN slot of the expansion DD11 backplane.
- 2.5.6 Install the UNIBUS options that are going on the second UNIBUS per their installation manuals.
- 2.5.7 Install the M9302 UNIBUS Terminator module in slot A6 of the last DD11 backplane.
- 2.5.8 Install the UNIBUS Exerciser (UBE) module ( M7855 ) from your Field Service Spares Kit into an SFC slot in the expansion DD11 backplane. Remove the MPG jumper wire (CA1 to CB1) in the backplane slot where the UBE is located.

**NOTE**

-----  
 UBE ADDRESS MUST BE SET FOR 770000  
 AND VECTOR SET FOR 510

SWITCHES	ADDRESS (E 125)	VECTOR (E 88)
S1	ON	ON
S2	ON	ON
S3	ON	ON
S4	ON	ON
S5	ON	OFF
S6	ON	OFF
S7	ON	ON
S8	ON	OFF



2.6 CHECK FOR POWER AND GROUND SHORTS IN EXPANSION BOX

2.7 CHECK REMOTE SENSE CABLE

Check that remote sense cable is connected from the CPU to the expansion cabinet.

2.8 POWER SYSTEM ON

2.8.1 Turn on all breakers.

2.8.2 Turn on key switch.

### SECTION 3 HARDWARE CHECKOUT

3.1 EXAMINE THE BUFFER DATA PATH REGISTERS OF SECOND UNIBUS

There are three buffer data path registers, and they are at the following addresses.

CSR1	F32004
CSR2	F32008
CSR3	F3200C

EX:

>>> E/P F32004

The register format of each of the registers is as follows.

```

+---+---+---+-----+-----+-----+-----+-----+-----+
|31|30|29|28|                |                |                |                |
+---+---+---+-----+-----+-----+-----+-----+
|  |  |  |                |                |                |                |
|  |  |  |                |                |                |                |
|  |  |  |                |                |                |                |
|  |  |  |                |                |                |                |
+-----+-----+-----+-----+-----+
(PJR) Purge Request -----+
(UCE) Uncorrectable Error
(NXM) Non Existent Memory
(ERR) Error Flag [OR of bits 29 & 30]

```



3.3.2 The following is a description of the IPEC registers:

```

ADDRESS REGISTER   FRF460
+-----+
115                                                    001
+-----+
  
```

This register contains sixteen of the address bits used during an NPR transfer initiated by control register 1. The upper two bits, 16 and 17, are contained in control register 1.

```

DATA REGISTER     FBF462
+-----+
115                                                    001
+-----+
  
```

This register has a dual function. For an NPR cycle it contains the data either sent or received by the NPR. For a BR cycle it contains the vector passed with the interrupt.

```

CONTROL REGISTER 1  FBF464
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
115|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 I I A A B B B B P T P A A C C N
 N N C C R R R R E G E 1 1 1 0 F
 I T L I 7 6 5 4           7 6           R
 T D C E
   O 1
   N
   E
  
```

NPR - Setting this bit causes the device to go an NPR cycle with the data contained in the address and data registers. If the bit fails to clear, it indicates that the device was unable to become bus master. This bit is also cleared by INIT.

C0, C1 - These bits determine what type of transfer will be done when NPR is set. They are as follows.

C1	C0	
0	0	DATI
0	1	DAIIP
1	0	DATG
1	1	DAIGB

A17, A16 - These bits are the upper two bits of the address register. INIT does not clear these bits.

PB - Setting this bit simulates a memory parity error setting the BUS PB signal on the UNIBUS when the data register is read. This bit is cleared by INIT.

TO - This bit indicates that a UNIBUS transfer timed out and SSYN was not returned. It is reclocked every transfer, and cleared by INIT. READ ONLY.

PE - This bit indicates that BUS PB on the UNIBUS occurred during a DATI. It is reclocked every DATI cycle, and also cleared by INIT. READ ONLY.

BR7-BR4 - These four bits cause the device to assert their respective BR requests, and attempt to interrupt at that level. They may be set in any combination to verify the arbitration logic. Once these bits are set the IPEC will attempt to interrupt until either the bit is cleared or the interrupt has taken place. These bits are not cleared by the interrupt taking place, and must be explicitly cleared by either writing a zero to the appropriate bit position, or by INIT before they can be set again to initiate another interrupt.

ACIE - This bit is ACLO Interrupt Enable. when set, it will cause an interrupt to vector 1E4 on the leading edge of a UNIBUS ACLO signal (power going down) and again approximately 100 ms after the trailing edge of ACLO (power coming up). Cleared by INIT.

ACLG1 - This bit is set by a power fail condition, and causes an interrupt if ACIE is set.  
READ ONLY

INTDONE - This bit indicates an interrupt has taken place that was caused by one of the BR bits being set. The bit is cleared by writing a (1) to it or by INIT.

INIT - This bit will initialize the internal logic of the IPEC when set. The output is undefined when read.

### CONTROL REGISTER 2 FBF46c

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|15|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
E I A B B B B V V V V V V V V V
X N C R R R R 8 7 6 5 4 3 2 1 0
T I L 7 6 5 4
M D C
C O 2
D N
E

```

V8-V0 - These bits specify the vector to be used by an interrupt initiated by Control Register 2. These bits are NOT cleared by INIT.

BR7-BR4 - These bits cause the device to interrupt in the same manner as the BR bits in Control Register 1. Cleared by INIT.

ACLO2 - This bit when set will cause ACLO on the UNIBUS to be asserted for approximately 1.5 ms. The bit is self clearing. This bit is NOT affected by INIT.

INTDONE - This bit works the same as the INTDONE bit in Control Register 1, but for interrupts initiated by the BR bits in Control Register 2. This bit is cleared by writing a one to it or by INIT.

EXIMOD - This bit is reserved for future use, should be zero when read. READ ONLY.

#### 3.4 EXAMINE A UNIBUS EXERCISER REGISTER

Examine location FBF000, this will give you location 770000 on the second UNIBUS. For a description of what the bits in the UBE registers do, consult the UBE Users Manual.

By examining a UBE register you are checking that you can get out to the BA box.

#### 3.5 BOOT UP THE DIAGNOSTIC SUPERVISOR IN STANDALONE MODE

Minimum revision of the Diagnostic Supervisor that can be used is (6.4).

EX:

```
B/10 XXXX
-----
```

where (XXXX) is the boot device.

#### 3.6 ATTACH THE DW750

This can be done in two ways, either by running the Autosizer program EVSBA or by doing a manual attach.

EX:1

```
DS> RUN EVSBA
-----
DS> SELECT ALL
-----
```

EX:2

```

DS> ATTACH DW750 CMI DW1
-----
DS> ATTACH UBE DW1 U20 770000 510
-----
DS> SELECT DW1
-----
DS> SELECT UBO
-----

```

## 3.7 RUN THE UBI/DW750 DIAGNOSTIC

A minimum of two passes of this diagnostic should be run. The program is called ECCBA, and should be REV 1.3 or higher.

EX:

```

DS> RUN ECCBA
-----

```

## 03.8 RUN APPROPRIATE DIAGNOSTICS FOR DEVICES ON THE SECOND UNIBUS

Run whatever other appropriate diagnostics are necessary to verify the peripherals that were added to the DW750'S UNIBUS. These diagnostics can be determined by referring to the installation manuals for the added devices, looking them up in EVNDX, or by using the Diagnostic Supervisor help file as follows.

EX:

```

DS> HELP DEV XXXX
-----

```

Where (XXXX) is the device you want know about.

## 3.9 REMOVE THE UNIBUS EXERCISER MODULE

3.9.1 Remove the UBE module.

3.9.2 Replace the NPG jumper wire on the backplane (wire from pins CA1 to CB1 in slot where M7855 is installed).

3.9.3 Replace the Grant card back in it's original slot (slot D of an SPC slot).

## 3.10 BRING UP VMS AND RUN UETP

For information on setting up and running UETP refer to the VAX/VMS UETP User's Guide (AA-Dc43A-TE)

## 3.11 RETURN SYSTEM TO CUSTOMER

**L0005 CCS/WCS**

CCS

SLOT 5). THE CPU CONTROL STORE (CCS) #60005

THE CPU CONTROL STORE CONTAINS THE MICROCODE ROMS FOR THE SYSTEM (6K BY 80 BITS), A "NEXT" ADDRESS LATCH, AND MISC. BANK SELECT AND PARITY GENERATOR LOGIC (HI NEXT FIELD ONLY).

THE CCS BOARD IS THE MOTHER BOARD FOR THE WCS (WRITEABLE CONTROL STORE) OPTION. THE WCS IS PRESSED ON THE PINS ON THE BOARD AND IF IT MUST BE REMOVED, USE A GRANT CARD AS A PRY BAR TO PREVENT IT FROM CRACKING. THERE IS A JUMPER ON SOME CCS BOARDS TO TIE BIT 13 OF THE "NEXT" FIELD OF THE MICROCODE LOW, IT MUST BE REMOVED TO ENABLE WCS-USE.

THERE IS NO DIAGNOSTIC TO TEST THE CCS ROM CONTENTS, THE BEST WAY TO DETERMINE THE CONDITION OF THE CCS IS TO DO A PARITY CHECK UNDER RDM MODE (RDM>PAR 0) IF YOU GET A PARITY STOP AT CSAD 17FD, IT IS OK. (AS FAR AS WE CAN TELL) KEEP IN MIND THAT DIFFERENT REVS. OF CCS BOARDS WILL CAUSE FAILURES UNDER THE WRONG REV. OF DIAGNOSTICS. THE REV. OF THE CCS CAN BE DETERMINED BY EXAMINING THE IPR 3E BITS 9-15 (SID) THE NUMBER WILL BE IN HEX AND MUST BE CONVERTED TO DECIMAL TO BE ABLE TO INTERPRET THE REV. LEVEL (EXAM: 3E = REV 62) THE DPM MICRODIAGNOSTICS DO SOME MINOR INTEGRITY TESTS ON THE CCS IF YOU WANT TO RUN THEM, RUN ECKA8.EXE DPM MICRO'S.

THE CCS MICROCODE FIELDS CONNECT TO THE DP4, MIC, AND UFI MODULES THROUGH BACKPLANE WIRING.

THE WCS IF INSTALLED IS CONNECTED TO THE CMI.  
THE WCS CAN BE TESTED USING THE LEVEL 3 DIAGNOSTIC ECKA8.EXE.

THE MODULE ALSO CONTAINS TWO OSCILLATORS ONE 19.75 MHZ FOR THE SAC CHIP TO CREATE SYSTEM CLOCKS, AND ONE 1 MHZ FOR THE TOK CHIP'S COUNTER.

THE CCS MODULE HAS NO GATE ARRAYS.



WRITEABLE CONTROL STORE KU750  
-----

THE KU750 WRITEABLE CONTROL STORE OPTION IS DESIGNED TO ALLOW THE LOADING OF OPTIONAL MICROCODE TO HANDLE FLOATING POINT AND HUGE INSTRUCTIONS NOT NORMALLY SUPPORTED BY THE BASIC CPU MICROCODE. THE WCS WILL ALSO ALLOW THE LOADING OF CUSTOMER WRITTEN ROUTINES IN MICROCODE PROVIDED THAT THE CODE IS WRITTEN IN THE PROPER FORMAT AND CARE IS TAKEN TO CALCULATE CORRECT MICRO-FIELD USAGE AND PARITY.

THE WCS IS LOADED VIA THE CMI BUS FROM ADDRESSES BEGINNING FROM F00000. EACH MICRO WORD WILL REQUIRE FOUR WRITES ACROSS THE CMI TO ASSEMBLE ALL 80 BITS. IF THE CODE IS DIGITAL'S FLOATING POINT OPTIONAL PACKAGE THEN A LOADER ROUTINE WILL BE PROVIDED ON A CASSETTE TAPE AND INSTRUCTIONS WILL BE GIVEN EXPLAINING HOW TO MAKE THE LOADING PROCESS A PART OF THE STARTUP COMMAND PROCEDURE.

**INSTALLATION:** THE WCS IS A SMALL PC BOARD WITH RAM CHIPS ON IT AND HAS NO METAL FRAMEWORK OR HANDLES. IT IS INSTALLED "PIGGYBACK" ON THE CONTROL STORE (CCS 00005) MODULE BY FIRST REMOVING THE PUSH ON JUMPER (IF INSTALLED) FROM THE PINS PROTRUDING FROM THE CCS BOARD. THIS JUMPER WAS INSERTED BY MANUFACTURING TO TIE BIT 13 OF THE CONTROL STORE "NEXT" FIELD TO GROUND THUS PREVENTING ACCESS OF MICRO ADDRESSES IN THE WCS RANGE (2000 ^) FROM BEING ACCESSED WITHOUT WCS INSTALLED. NEXT REMOVE (IF INSTALLED) THE PLASTIC PIN GUARDS FROM THE PIN PORTS ON THE CCS. THESE GUARDS SERVED TO PROTECT THE PINS AND ALSO ASSURE THEY ARE STRAIGHT. THE WCS MUST BE CAREFULLY PRESSED DOWN AGAINST THE CCS AND SECURED BY TWO NYLON SCREWS PROVIDED IN THE KIT. FINALLY RE-INSTALL THE CCS MODULE IN THE CPU.

**NOTE:** THIS TEXT IS INTENDED TO BE A REMINDER OF THE BASIC INSTALLATION PROCEDURE AND SHOULD NOT BE SUBSTITUTED FOR THE KU750 INSTALLATION GUIDE IN ANY WAY...

**TESTING AND DIAGNOSIS:** THE WCS CAN BE TESTED BY RUNNING THE LEVEL 3 MACRO DIAGNOSTIC ECKAX.EXE THIS TEST SHOULD BE RUN IN THE MANUAL MODE (DS>E ECKAX/SEC:MANUAL) AND THE WCS FORMATTING TEST WILL GIVE A "LAST ADDRESS" OF WCS. THIS ADDRESS SHOULD BE REMEMBERED AND USED IN THE ATTACH COMMAND FOR THE KA750 CPU. IF THE DIAGNOSTIC FAILS AND THE CPU IS KNOWN TO BE IN GOOD CONDITION, THEN THE WCS MUST BE REPLACED.

**TECH TIP:** WHEN REPLACING THE WCS USE A G727 UNIBUS GRAB CARD AS A PRY BAR TO EVENLY DISTRIBUTE THE PRESSURE OF REMOVAL. IF A SCREWDRIVER IS USED, THERE IS A VERY GOOD CHANCE THAT YOU WILL CRACK THE PC BOARD.

NOTE: THE WCS IS NOT GOING TO BE STOCKED IN YOUR FIELD SPARES KITS DUE TO THE PROJECTED LOW DEMAND FOR THIS OPTION, THEREFORE IT IS LEFT UP TO YOUR INDIVIDUAL OFFICES TO KEEP A SPARE ON HAND IF YOU SUPPORT THE OPTION.

USER ACCESS: FOR A USER TO ACCESS A USER WRITTEN MICROCODE ROUTINE IN THE WCS, HE/SHE MUST FIRST BE SURE THAT THE REQUIRED MICROCODE IS LOADED. IN THE MACRO PROGRAM THAT IS ATTEMPTING ACCESS WCS HE/SHE MUST USE A "XFC" NATIVE MODE INSTRUCTION. THE XFC (EXTENDED FUNCTION CALL) INSTRUCTION WILL SEND US TO SCBB+14 AND THE LOWER TWO BITS OF THE VECTOR ADDRESS AT THAT LOCATION MUST BE EQUAL TO 2. IF THIS IS THE CASE WE WILL TRAP TO WCS LOCATION 2001 AND HOPEFULLY PICK UP THE FIRST MICROWORD OF OUR ROUTINE. TO RETURN TO CCS AFTER EXECUTION OF THE MICROROUTINE WE SIMPLY HAVE TO HAVE A CCS ADDRESS IN THE "NEXT" FIELD OF OUR FINAL MICROWORD IN WCS. OBVIOUSLY SOME CARE MUST BE GIVEN IN THE SELECTION OF A RETURN ADDRESS AND THE USER IN ANY CASE SHOULD RESEARCH THE SUBJECT THOROUGHLY AND REFER TO THE ASSOCIATED DOCUMENTATION BEFORE ATTEMPTING AN ADVENTURE OF THIS MAGNITUDE.

NOTE: CUSTOMER WRITTEN MICROCODE IS NOT SUPPORTED BY D.E.C.

**L0006 RDM/MTM**

RDM/MTM

SLOT 6). THE REMOTE DIAGNOSTIC MODULE (RDM) #L0006 (F.S.)  
 THE MAINTENANCE TOOL MODULE (MTM) #L0006 YA (CUST.)

THE REMOTE DIAGNOSTIC MODULE IS CONSIDERED A DIAGNOSTIC TOOL. IT IS NEVER CUSTOMER OWNED OR REPAIRED ON CUSTOMER TIME! O.E.M. CUSTOMERS CAN PURCHASE A SPECIAL MODULE CALLED THE MTM MODULE WHICH ALLOWS RUNNING MICRO-DIAGNOSTICS BUT DOES NOT ALLOW CONNECTION TO THE ODC CENTERS.

THE RDM ALLOWS RUNNING THE MICRODIAGNOSTICS, AND REMOTE DIAGNOSIS FROM THE ODC CENTERS. WHEN THE RDM IS INSTALLED, THE TU-58 AND CONSOLE SIGNAL CABLES ON THE BACKPLANE MUST BE MOVED TO THE RIGHT SIDE (VIEWED FROM THE REAR) OF THE PINS IN SLOT 6. THIS ALLOWS THE CONSOLE TO TALK DIRECTLY TO THE RDM BOARD, AND THE TU-58 TO LOAD MICRO-DIAGNOSTIC PROGRAMS DIRECTLY INTO THE RAM ON THE RDM BOARD OR LEVEL 4 MACRO-DIAGNOSTICS THROUGH THE RDM INTO MAIN MEMORY.

WHEN IN CONSOLE I/O MODE, (>>>) A ^P THEN ^D WILL ENTER RDM MODE (RDM>). WHEN A MICRODIAGNOSTIC TAPE IS LOADED, THE FIRST PROGRAM TO LOAD IS ECKAA.EXE OUR MICRO MONITOR. ONCE LOADED, THE MICMON HANDLES THE DIAGNOSTIC EXECUTION AND HAS QUITE A FEW OF IT'S OWN UNIQUE COMMANDS. THE DIAGNOSTIC MINI REFERENCE GUIDE LISTS THE COMMANDS UNDER MICMON (MIC>). THE MICMON CAN BE LOADED BY ITSELF UNDER RDM MODE BY THE COMMAND RDM>TE/C <CR>

NOTE: MICRO-DIAGNOSTICS REV. 4.XX AND UP HAVE ECKAF.EXE THE RDM'S OWN INTERNAL DIAGNOSTIC AT THE END OF EACH TAPE. TO RUN THE RDM DIAGNOSTIC YOU MUST USE A NEW COMMAND UNDER MICMON, (MIC>DI DM <CR>)

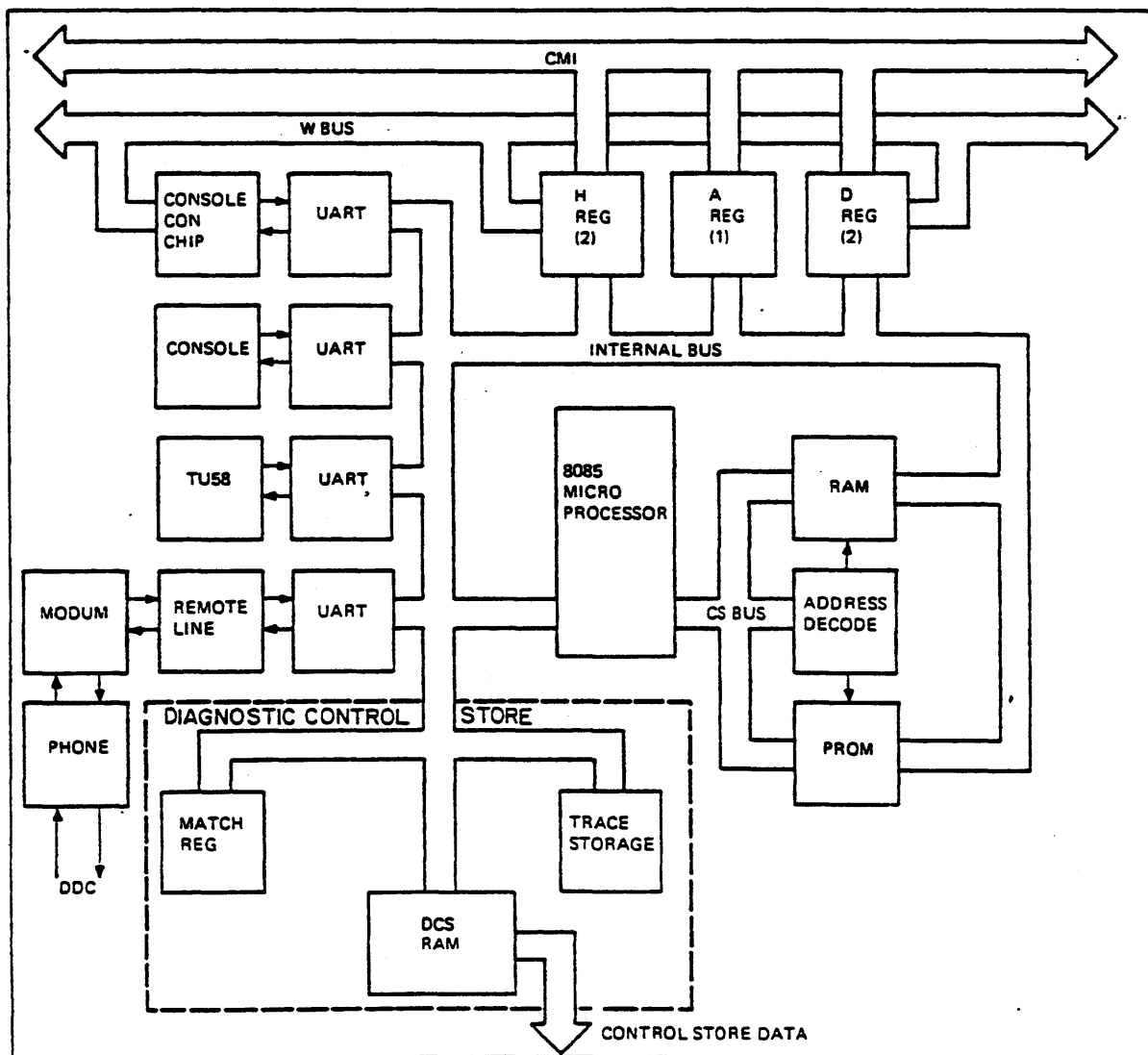
\*\*\* THIS DIAGNOSTIC MUST BE RUN IN A KNOWN GOOD CPU! \*\*\*

THE RDM MODULE CONNECTS DIRECTLY TO THE CMI TO ALLOW TROUBLESHOOTING ANY MODULE LOCATED ON THE CMI BUS.  
 \*\*\* THE RDM HAS THE HIGHEST CMI PRIORITY OF ANY NEXUS \*\*\*  
 IT IS ALSO CONNECTED TO THE W BUS.

THE RDM MODULE HAS NO GATE ARRAYS.

## CONTROL KEY FUNCTIONS, RDM

Control D Enter RDM console mode  
 Control P Enter console mode  
 Control U Abort current command line  
 Control O Inhibit printing of text  
 Control R Retype current command line  
 Control C Cancel current function (repeat console command)  
 Control S Disable CPU output to active terminal  
 Control Q Continue output to terminal after Control S



Remote Diagnostic Block Diagram

## OPTIONS      SLOTS #7,8,9)      OPTION SLOTS

THESE SLOTS CAN BE USED TO INSTALL ANY OF THE CURRENT CPU OPTIONS AVAILABLE. (EXCLUDING THE FPA WHICH MUST BE INSTALLED IN SLOT #1) THE MOST COMMON OPTION IN THESE SLOTS IS THE MASBUS ADAPTER (MBA) OPTION. WHEN AN OPTION IS INSTALLED THE GRANT JUMPERS ON THE BACKPLANE MUST BE REMOVED FOR THAT SLOT AND THE APPROPRIATE ARBITRATION JUMPERS MUST BE SET UP IN ACCORDANCE WITH THE OPTION INSTALLATION GUIDE. OPTIONS AVAILABLE INCLUDE THE MASBUS ADAPTER (MBA), AND A SECOND UNIBUS ADAPTER (SUB), THE SECOND UNIBUS DOES NOT HAVE "CON" CHIPS OR AN "INT" CHIP ON IT, THUS IT IS REFERRED TO AS A "SUB" (SECOND UNIBUS) ADAPTER. THE PURPOSE OF THIS SECOND UNIBUS IS TO ALLOW CONNECTION OF MORE UNIBUS DEVICES.

PLEASE NOTE: THAT THE SYSTEM CANNOT BE BOOTED FROM ANY DEVICES ON THE SECOND UNIBUS.

77

L0007 MBA

THE RH750 MASSBUS ADAPTER IS A GENERAL PURPOSE INTERFACE BETWEEN THE CMI AND THE HIGH SPEED MASSBUS DRIVES. IT INTERFACES THE 32 BIT CMI DATA PATH TO THE 16 BIT DATA PATH OF THE MASSBUS.

#### GATE ARRAYS:

- MDP:** MASSBUS DATA PATH (8 CHIPS)  
RESPONSIBLE FOR ROUTING DATA AND ADDRESS INFORMATION TO AND FROM THE CMI AND MASSBUS. EACH CHIP HANDLES 4 BITS EACH. *ALSO TELL WHICH MBA YOU ARE LOOKING FOR*
- MDC:** MASSBUS DATA PATH CONTROL CHIP (1)  
CONTROLS AND MAINTAINS STATUS ON MAP PARITY AND VALIDITY. IT DETECTS THE BEGINNING AND END OF A DATA TRANSFER, AND MAINTAINS THE STATUS ON THE SUCCESS OF EACH TRANSFER.
- MCI:** MASSBUS CMI INTERFACE CONTROL (1)  
HANDLES ARBITRATION, COMMAND/ADDRESS CONTROL, STATUS GENERATION AND CHECKING INTERRUPTS.
- MRC:** MASSBUS REGISTER CONTROL (1)  
DETECTS THE INITIATION OF DATA TRANSFERS AND PRODUCES THE DATA TRANSFER FUNCTION CODES. IT ALSO PRODUCES THE CONTROL SIGNALS FOR THE MASSBUS CONTROL BUS.
- MSC:** MASSBUS SILO CONTROL CHIP (1)  
CONTAINS THE REGISTERS NEEDED TO ADDRESS THE 32 BYTES OF SILO RAM, AND THE LOGIC USED TO DETECT SILO EMPTY AND SILO FULL. IT CONTROLS THE GENERATION AND CHECKING OF SILO AND MASSBUS DATA BUS PARITY. THE CMI DATA MASK IS GENERATED HERE AS IS THE CONTROL FIELD FOR CONTROLLING THE FLOW OF DATA IN THE MDP CHIPS.

#### THE BUSES:

- CONTROL BUS:** THE MASSBUS CONTROL BUS IS AN ASYNCHRONOUS BUS LINKING THE DRIVE CONTROL/STATUS REGISTERS WITH THE CPU. THE CONTROL BUS IS INDEPENDANT OF THE DATA BUS, ALLOWING NON DATA TRANSFER OPERATIONS TO BE INITIATED WHILE A DATA TRANSFER IS IN PROGRESS. THERE ARE 31 SIGNAL LINES.
- DATA BUS:** THE MASSBUS DATA BUS IS A HIGH SPEED SYNCHRONOUS BUS USE FOR TRANSFERRING BLOCKS OF DATA. THE MBA MUST BUFFER THE DATA AND GENERATE MEMORY ADDRESSES. THERE ARE 23 SIGNAL LINES.
- INTERNAL BUS:** INTERFACES ALL THE INTERNAL SIGNALS THAT CONTROL THE ADAPTER. THERE ARE 32 SIGNAL LINES.



(TOP)  
THE MBA MODULE #L0007

-----  
MSC

-----  
MDP3

-----  
MDP2

-----  
MDP4

-----  
MDP1

-----  
MRC

-----  
MDP8

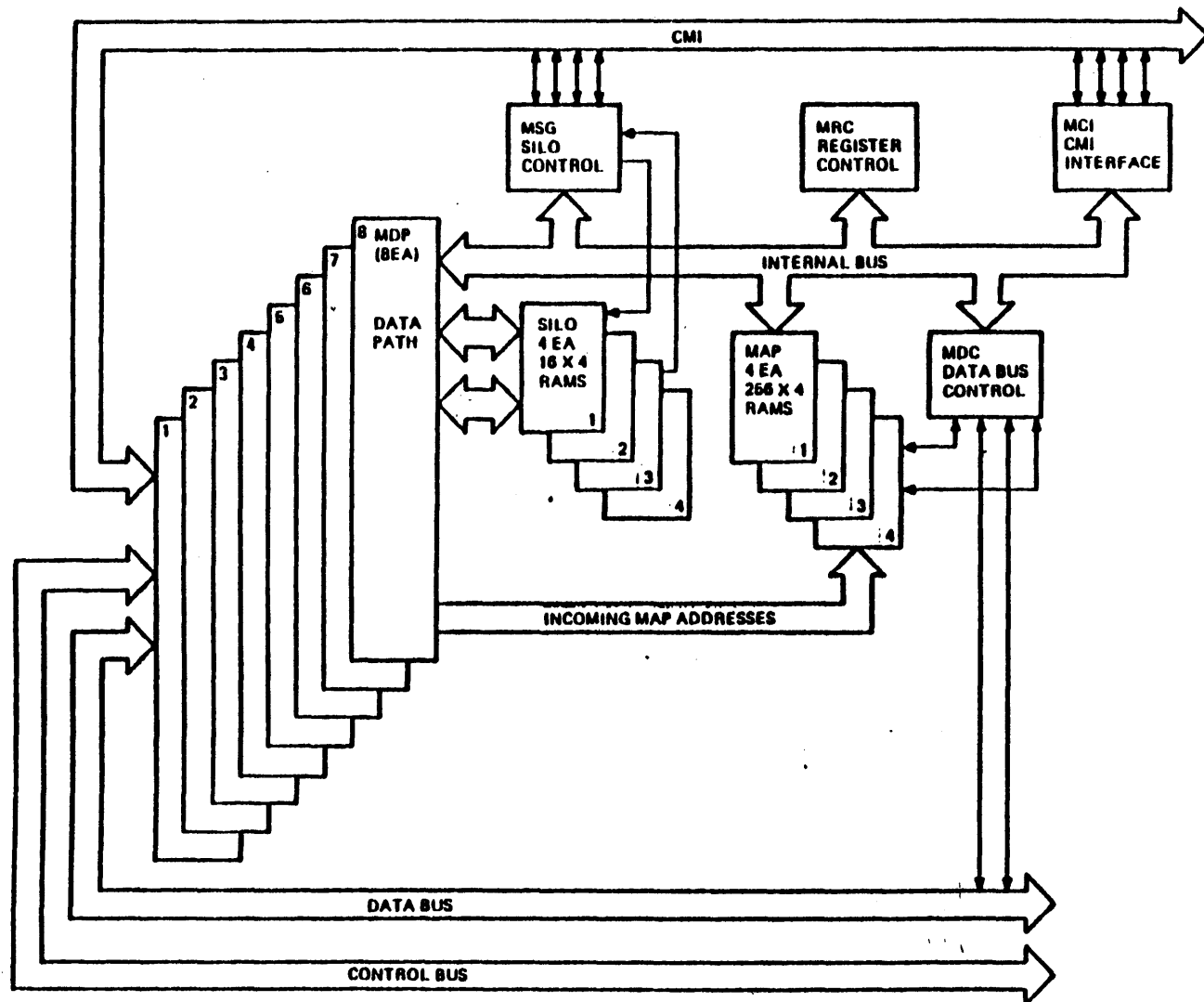
-----  
MDP5

-----  
MCI

-----  
MDP7

-----  
MDP6

-----  
MDC



RH-750 MBA Simplified Block Diagram

MASSBUS ADAPTER INSTALLATION JUMPERS  
-----

- MBA'S MUST BE SET UP FOR PROPER:
1. SILO TRANSFER RATE
  2. MBA NUMBER IDENTIFICATION (MBA0, MBA1 OR MBA2)
  3. CMI ARBITRATION LEVEL (1, 2 OR 3)
1. BEFORE ANY MBA INSTALLATION, FIRST REMOVE ALL BG JUMPERS FROM THE DESIRED SLOT. (PREFERABLY SLOT 9 IF THIS IS THE FIRST MBA)
  2. SILO TRANSFER RATE INVOLVES A SINGLE JUMPER BETWEEN PINS 43 AND 45 OF THE DESIRED SLOT. THIS JUMPER MUST BE INSTALLED IN ALL SLOTS CONTAINING MBA'S. SECTION A OF THE DESIRED SLOT.  
-----
  3. MBA IDENTIFICATION JUMPERS ARE REQUIRED TO INFORM THE CPU WHICH MBA IS IN WHICH SLOT. THE JUMPERS ARE INSTALLED AS FOLLOWS:

ALL JUMPERS INSTALLED IN SECTION A OF THE DESIRED SLOT

MBA0 AS RELATED TO DEVICE CODE DBAX:

JUMPER PINS 51 TO 53 AND 52 TO 54 TO SELECT MBA0

MBA1 AS RELATED TO DEVICE CODE DBX:

JUMPER PINS 51 TO 53 ONLY TO SELECT MBA1

MBA2 AS RELATED TO DEVICE CODE DBCX:

JUMPER PINS 52 TO 54 ONLY TO SELECT MBA2

4. CMI ARBITRATION JUMPERS ARE TO ESTABLISH A CMI PRIORITY LEVEL FOR THE MBA. KEEP IN MIND THAT IN THE 11/750 THE MBA'S USE THE UNIBUS BR5/BG5 LINES TO INTERRUPT THE CPU, THUS SOME THOUGHT MUST BE GIVEN TO WHAT LEVEL TO ASSIGN TO WHICH SLOT.

NOTE: BASICALLY THE HIGHER THE SLOT NUMBER, THE HIGHER THE CMI ARB LEVEL. (THIS IS ONLY A GOOD RULE OF THUMB, NOT A REQUIREMENT.)

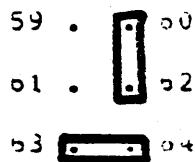
ALSO SOME THOUGHT SHOULD BE GIVEN TO ACCESS OF CABLES ON THE BACKPLANE WHEN INSTALLING OTHER OPTIONS.

THERE ARE THREE CMI ARBITRATION LEVELS ASSIGNED TO MBA'S. THEY ARE AS FOLLOWS:

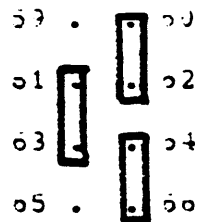
CMI ARB LEVEL 3:     INSTALL JUMPER 62 TO 64  
SECTION A



CMI ARB LEVEL 2:     INSTALL JUMPERS 60 TO 62 AND 63 TO 64  
SECTION A



CMI ARB LEVEL 1:     INSTALL JUMPERS 60 TO 62, 61 TO 63 AND 64 TO 66  
SECTION A



IF OTHER DEVICES ARE TO BE INSTALLED IN THE OPTION SLOTS, THERE ARE TWO MORE CMI ARB LEVELS RESERVED FOR FUTURE USE.

CMI ARB LEVEL 5:   INSTALL JUMPER 55 TO 57  
                          SECTION A

55  . 56  
57  . 58

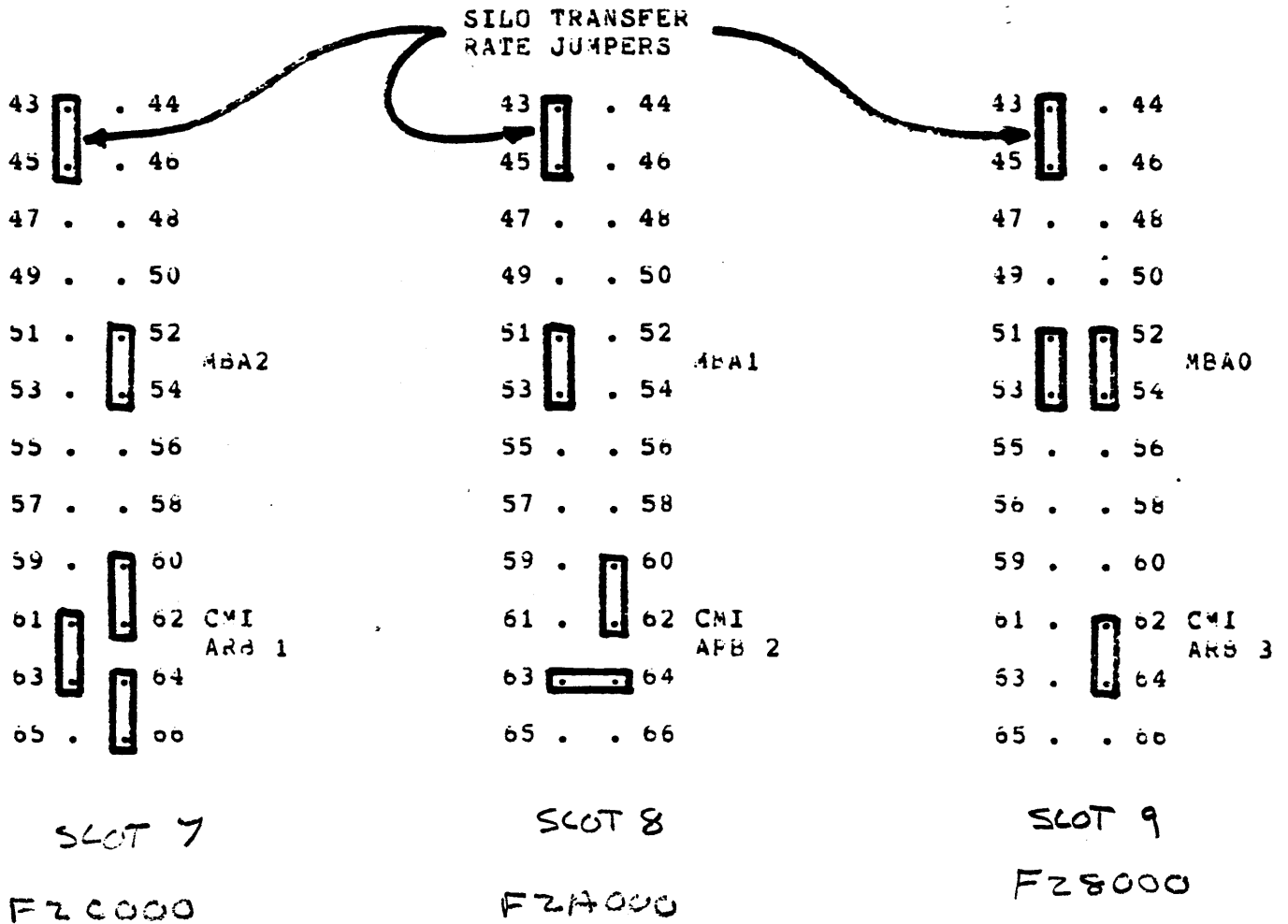
AND CMI ARB LEVEL 6:   INSTALL JUMPER 57 TO 59  
                          SECTION A

57  . 58  
59  . 60

## EXAMPLES OF RECOMMENDED MASSBUS ADAPTER JUMPER CONFIGURATIONS

### ----- SECTION A OF DESIRED SLOT -----

WHEN INSTALLING AN OPTION, ALL BG JUMPERS MUST BE REMOVED FROM THAT SLOT!



**L0011/16 CMC**

CMC

SLOT #10). THE CPU MEMORY CONTROLLER (CMC) #L0011 OR #L0016

FIRST NOTE THAT THE BOARD NUMBER IN THIS SLOT IS L0011  
OR L0016 NOT 10.

THE CPU MEMORY CONTROLLER MODULE CONTAINS IT'S OWN  
MICROCODE TO CONTROL MEMORY REFRESHES, DATA BUFFERING,  
SELECT TIMING (CAS AND RAS). THE BOARD CONTAINS ERROR  
CHECKING AND CORRECTING LOGIC, STATUS REGISTERS,  
CONFIGURATION STATUS LOGIC, AND THE BOOTSTRAP ROMS.

STATUS REGISTERS: CSR0 F20000 (ERROR LATCHING REGISTER)  
CSR1 F20004 (DIAGNOSTIC REGISTER)  
CSR2 F20008 (MEMORY MAP REGISTER)

BOOTSTRAP ROMS: ROM SOCKET A F20400  
ROM SOCKET B F20500  
ROM SOCKET C F20600  
ROM SOCKET D F20700

NOTE: WHEN EXAMINING THE ROMS IN THEIR SOCKETS, THE LOW  
WORD OF THE FIRST LOCATION WILL CONTAIN THE ASCII  
EQUIVALENT OF THE DEVICE TYPE MNEUMONIC FOR THAT ROM.

EXAMPLE: >>>E/L/P F20400

P 00F20400 XX XX 44 44 = DD (DDA0)

THE CMC IS CONNECTED TO THE CMI AND THE MEMORY BUS.

THE LEVEL 3 DIAGNOSTIC ECKAM.EXE TAPE #5 WILL TEST THE  
MEMORY AND CONTROLLER. PLEASE NOTE THAT THE DIAGNOSTIC  
WILL TEST ONLY ARRAYS 1-7, TO TEST ARRAY 0, YOU MUST  
EITHER SWAP POSITIONS WITH ANOTHER BOARD OR RUN THE  
MIC MICRODIAGNOSTIC (ECKAC.EXE TAPE #2).

THE CMC HAS TWO AVAILABLE VERSIONS:

L0011: SUPPORTS M8728 256 KB ARRAYS ONLY (MAX. 2 MB'S)  
L0016: SUPPORTS BOTH M8728 AND M8750 MODULES  
(M8750 IS A 1 MEGABYTE ARRAY GIVING US A MAX.  
OF 8 MB'S)  
IF BOTH TYPES ARE MIXED, THE M8728'S MUST BE  
INSTALLED DIRECTLY AFTER THE 1 MEG BOARDS.

NOTE: THE L0016 CMC MODULE REQUIRES A REV. C BACKPLANE WHICH  
ENABLES THE USE OF BIT 24 OF THE CMI ADDRESS. THE REV.  
CAN BE DETERMINED BY EXAMINING THE SYSTEM I.D. REGISTER

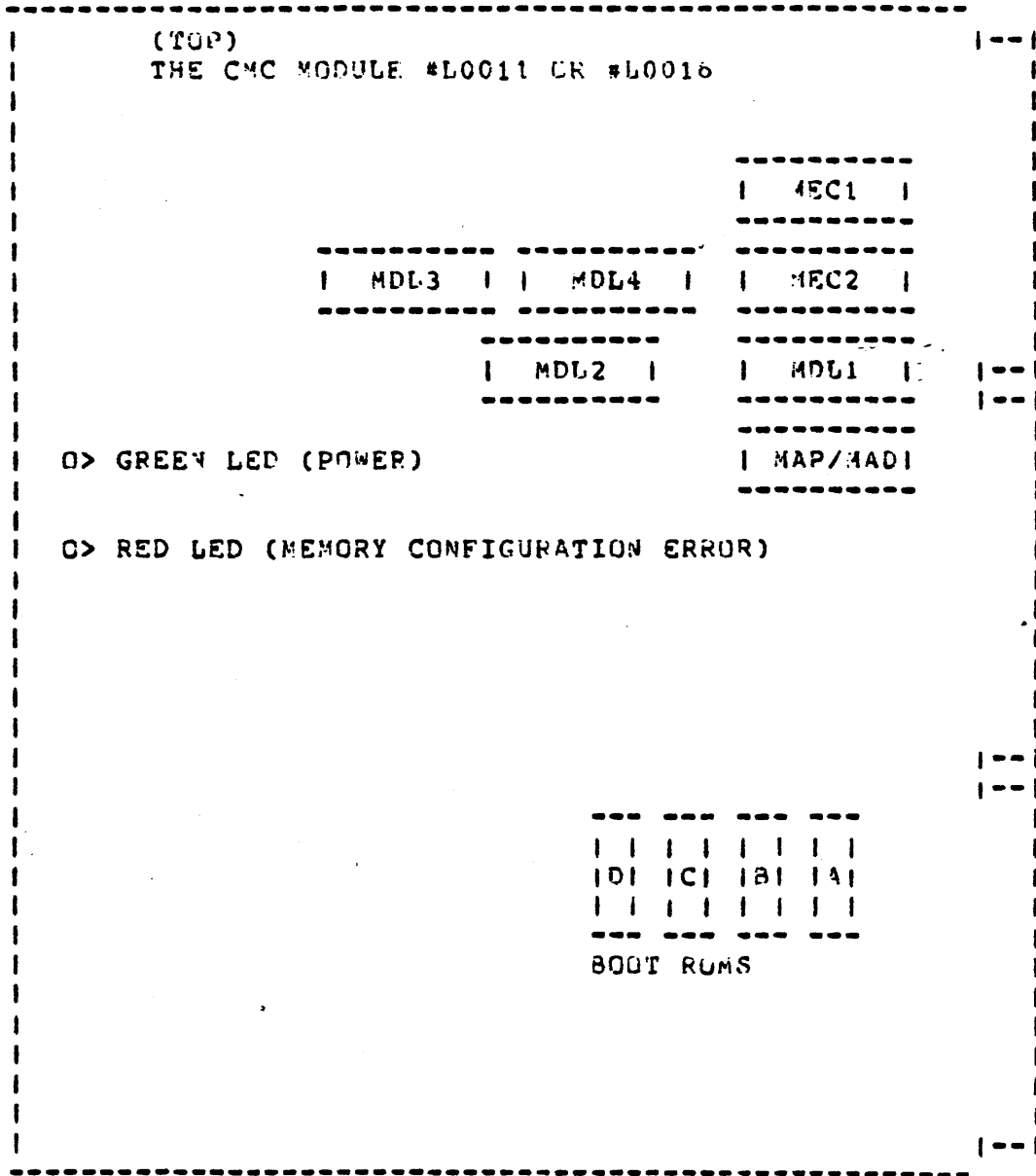
>>>E/I 3E

I 0000003E 02 00 XX X8

^  
A 3 INDICATES A REV C BACKPLANE  
(SEE THE REVCON DOCUMENT)

GATEARRAYS: MAP,MDL,MEC





MAP/MAD: MEMORY ADDRESS PROCESSOR CHIP

MAP = L0011 MAD = L0016

THE MAP/MAD CHIP PERFORMS THE DECODING OF ADDRESS BITS TO ENABLE MEMORY ARRAYS, DETECT NXM'S, SIZE MEMORY ARRAY BOARD POPULATIONS, AND DETERMINE STARTING ADDRESS OFFSET. THE MAP/MAD CHIP ISSUES NXM STATUS ON THE CMI, AND LIGHTS THE RED LED IF IT DETECTS AN ILLEGAL CONFIGURATION OF MEMORY.

PART NUMBER: 19-14706

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE  
MS750 MACRO ECKAM.EXE (LEVEL 3)

MODULE: CMC GATE ARRAY: MAP

MAP	
CMI DATA 19__1__	o_48_CMCK STA 19
CMI DATA 17__2__   o	1o_47_CMCK STA 23
CMI DATA 18__3__	1o_46_CMCK STA 21
CMI DATA 20__4__	1o_45_CMCK STA 20
CMCF LATCH IAR__5_o	1o_44_CMCK STA 17
CMCU MEMORY PRESENT__6_o	1o_43_CMCK STA 22
CMI DATA 22__7__	1o_42_INTERNAL BUS MEMORY PRESENT 7
CMI DATA 21__8__	1o_41_FNGP3 7
CMI DATA 23__9__	1o_40_CMCK STA 19
INTERNAL BUS MEMORY PRESENT 1_10_o	1o_39_INTERNAL BUS MEMORY PRESENT 2
FNGP3 1_11_o	1__38_GROUND
VGA_12__	1o_37_INTERNAL BUS MEMORY PRESENT 4
VCC_13__	1o_36_INTERNAL BUS MEMORY PRESENT 3
INTERNAL BUS MEMORY PRESENT 0_14_o	1__35_GROUND
FNGP3 0_15_o	1o_34_FNGP3 2
INTERNAL BUS ADDRESS MEM SEL2_16_o	1o_33_INTERNAL BUS MEMORY PRESENT 5
INTERNAL BUS ADDRESS MEM SEL1_17_o	1o_32_FNGP3 5
INTERNAL BUS ADDRESS MEM SEL3_18_o	1o_31_FNGP3 6
INTERNAL BUS ADDRESS MEM SEL0_19_o	1o_30_INTERNAL BUS MEMORY PRESENT 6
N_20__	1o_29_FNGP3 4
N_21__	1__28_N
INTERNAL BUS ADDRESS MEM SEL5_22_o	1o_27_FNGP3 3
INTERNAL BUS ADDRESS MEM SEL4_23_o	1__26_CMCF MAR LATCH
INTERNAL BUS ADDRESS MEM SEL6_24_o	( ) 1o_25_INTERNAL BUS ADDRESS MEM SEL7

-----  
THIS SIDE TOWARDS FINGERS ON BOARD

MDL: MEMORY DATA LOOP CHIPS

THE MDL CHIPS FUNCTION LIKE THE MDR CHIPS ON THE MIC MODULE, THEY PASS ALL DATA AND ADDRESSES TO AND FROM THE CMI AND INTERNAL MEMORY BUSES. THEY PROVIDE VARIOUS STATUS REGISTERS (CSR'S), AND A PATH FOR THE BOOTSTRAP ROMS TO MEMORY.

4 CHIPS	CHIP	BIT SLICE
	MDL 1	<7-0>
	MDL 2	<15-8>
	MDL 3	<23-16>
	MDL 4	<31-24>

PART NUMBER: 19-14707

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE  
MS750 MACRO ECKAM.EXE (LEVEL 3)

MODULE: CMC GATE ARRAY: MDL (1 THROUGH 4)

MDL (1 THROUGH 4)	
CMI DATA 06,14,22,30__1<>-----	48_LATCH IAR
CMI DATA 00,08,16,24__2<>  o	Io_47_INIT F
N,A08,A16,N__3__	<>46_CMI DATA 02,10,18,26
LATCH REG 2 MDL 0,1,2,3__4__	<>45_CMI DATA 01,09,17,25
MDL OUTPUT CONTROL 2__5_o	<>44_CMI DATA 07,15,23,31
LATCH AUX MAR__6__	__43_CMI DR ENABLE
N__7__	<>42_CMI DATA 05,13,21,29
A03,A11,A19,CSR 1-27__8__	<>41_CMI DATA 03,11,19,27
MDL OUTPUT CONTROL 1__9_o	__40_LATCH MDR
N,A09,A17,CSR 1-25__10__	<>39_CMI DATA 04,12,20,28
A02,A10,A18,CSR 1-26__11__	__38_GROUND
VGA__12__	Io_37_CSR WR CY
VCC__13__	__36_DR ENABLE 0,1,2,3
N,MDL1 ADD HIT,MDL2 ADD HIT,N__14__	__35_GROUND
VGA,VGA,GRND,GRND (IDENT)__15_o	<>Io_34_INTERNAL BUS DB07,15,23,31 RD
VGA,GRND,VGA,GRND (IDENT)__16_o	<>Io_33_INTERNAL BUS DB06,14,22,30 RD
A04,A12,A20,CSR 1-28__17__	<>Io_32_INTERNAL BUS DB05,13,21,29 RD
A06,A14,A22,N__18__	<>Io_31_INTERNAL BUS DB00,08,18,24 RD
A07,A15,A23,CSR 0-31__19__	<>Io_30_INTERNAL BUS DB01,09,17,25 RD
A05,A13,A21,N__20__	<>Io_29_INTERNAL BUS DB03,11,19,27 RD
LATCH REG 1 MDL 0,1,2,3__21__	__28_ROM DATA 3
N,MDL1 ERR HIT,MDL2 ERR HIT,N__22__	<>Io_27_INTERNAL BUS DB02 RD
ROM DATA 0__23__	__26_ROM DATA 2
INTERNAL BUS DB04,12,20,28 RD__24_o <> ( )	__25_ROM DATA 1

THIS SIDE TOWARDS FINGERS ON BOARD

MEC: MEMORY ERROR CORRECTION CHIPS

DETECT AND CORRECT ALL SINGLE BIT MEMORY ERRORS USING THE MODIFIED HAMMING CODE AND SYNDROME BITS, AND DETECT DOUBLE BIT ERRORS (UNCORRECTABLE).

2 CHIPS CHIP BIT SLICES
MEC 1 <15-0>
MEC 2 <31-16>

PART NUMBER: 19-14705

BEST DIAGNOSTICS: MIC MICRO'S ECKAC.EXE
MS750 MACRO ECKAM.EXE (LEVEL 3)

MODULE: CMC GATE ARRAY: MEC (1 AND 2)

MEC (1 AND 2)
MEC LATCH DATA IN\_\_1\_\_->-o\_48\_INTERNAL\_BUS\_DB10,26 RD
OUTPUT BYTE0 LOWWORD,HIGHWORD\_\_2\_\_| o <>|o\_47\_INTERNAL\_BUS\_DB06,24 RD
MEC LATCH OUTPUT\_\_3\_\_o| <>|o\_46\_INTERNAL\_BUS\_DB15,31 RD
INTERNAL\_BUS\_DB05,21 RD\_\_4\_\_o|<> |o\_45\_MEC LATCH OUTPUT
INTERNAL\_BUS\_DB01,17 RD\_\_5\_\_o|<> <>|o\_44\_INTERNAL\_BUS\_DB12,28 RD
INTERNAL\_BUS\_DB07,23 RD\_\_6\_\_o|<> <>|o\_43\_INTERNAL\_BUS\_DB14,30 RD
N\_\_7\_\_-| <>|o\_42\_INTERNAL\_BUS\_DB13,29 RD
INTERNAL\_BUS\_DB03,19 RD\_\_8\_\_o|<> |\_\_41\_OUTPUT BYTE 1 LOW,HIGH
INTERNAL\_BUS\_DB02,18 RD\_\_9\_\_o|<> <>|o\_40\_INTERNAL\_BUS\_DB09,25 RD
INTERNAL\_BUS\_DB00,16 RD\_\_10\_\_o|<> <>|o\_39\_INTERNAL\_BUS\_DB11,27 RD
INTERNAL\_BUS\_DB04,20 RD\_\_11\_\_o|<> |\_\_38\_GROUND
VGA\_12\_\_-| |o\_37\_CORRECT DISABLE
VCC\_13\_\_-| <>|o\_36\_INTERNAL\_BUS\_CB02 RD,P SYND02
INTERNAL\_BUS\_DB06,22 RD\_\_14\_\_o|<> |\_\_35\_GROUND
N,SINGLE ERROR\_15\_\_o| |\_\_34\_VGA,HIGHWORD OUTPUT CB SYNDROME
N,ERROR\_16\_\_o| |\_\_33\_LOWWORD GEN,GRND
P SYNDROME 1,INTL BUS CB 1 RD\_\_17\_\_o| <>|o\_32\_INTERNAL\_BUS\_CB01 RD,P SYND01
P SYNDROME01,INTL BUS CB01 RD\_\_18\_\_o| <>|o\_31\_INTERNAL\_BUS\_CB08 RD,P SYND08
P SYNDROME04,INTL BUS CB04 RD\_\_19\_\_o| <>|o\_30\_INTERNAL\_BUS\_CB16 RD,P SYND16
P SYNDROME16,INTL BUS CB16 RD\_\_20\_\_o| <>|o\_29\_INTERNAL\_BUS\_CB 1 RD,P SYND T
P SYNDROME02,INTL BUS CB02 RD\_\_21\_\_o| <>|o\_28\_INTERNAL\_BUS\_CB04 RD,P SYND04
P SYNDROME32,INTL BUS CB32 RD\_\_22\_\_o| <>|o\_27\_INTERNAL\_BUS\_CB32 RD,P SYND32
VGA,GRND\_23\_\_o| |\_\_26\_LOWWORD OUTPUT CB BUS,GRND
LOWWORD LATCH CB REGISTER,VGA\_24\_\_o| ( ) |o\_25\_P SYNDROME08,INTL BUS CB08 RD

THIS SIDE TOWARDS FINGERS ON BOARD

91

**M93 13 UET**

NPR READ (DATI) USING MAP 0, BDP 1, PFN 6

```

>>>D/I 37 1           :INIT
>>>D/W/P F30004 1     :PURGE BDP 1
*** >>>D/L/P F30800 80200008 :SET UP MAP 0, VALID, ADD PFN = 1000 PFN = 8
>>>D/P/L 1000 12345678 :DATA
>>>D/W/P FFF460 0     :SET UP UNIBUS ADDRESS
>>>D/W/P FFF464 1     :SET NPR "GO" BIT

```

TEST THE RESULTS:

```

>>>E/W FFF462
      Should get:5678

```

INCREMENT UET ADDRESS REGISTER:

```

>>>D/W/P FFF460 2
>>>D/W/P FFF464 1

```

TEST THE SECOND RESULT:

```

>>>E/W FFF462
      Should get:1234

```

-----

OTHER DATA PATHS AND MAP FIELDS CAN BE USED BY SUBSTITUTING THE DATA AT THE "\*\*\*" FOR THE FOLLOWING: BDP2 = 80400008  
BDP3 = 80600008  
DIRECT DP = 80000008

-----

MANY OTHER EXAMPLES CAN BE TRIED USING THIS FORMAT:

NPR WRITE (DATC or DATOB)

```

A). LOAD ADDRESS REGISTER FFF460
B). LOAD DATA REGISTER   FFF462
C). LOAD CONTROL REGISTER FFF464 WITH THE FOLLOWING:
      CR<0>=1
      CR<2,1>=DATC or DAIG3 (See Chart)
      CR<4,3>=A17,A16      (Address bits 17 and 16)

```

NPR READ (DATI or DATIP)

```

A). LOAD ADDRESS REGISTER FFF460
B). LOAD CONTROL REGISTER FFF464 WITH THE FOLLOWING:
      CR<0>=1
      CR<2,1>=DATI or DATIP (See Chart)
      CR<4,3>=A17,A16      (Address bits 17 and 16)

```

B.R.

```

A). LOAD DATA REGISTER WITH VECTOR ADDRESS
B). LOAD CONTROL REGISTER WITH THE FOLLOWING:
      CR<11-8>=BR LEVEL   (Example:CR<8>=1; = #4)

```

FUNCTION CHART:

```

C1 | C0
-----
0 | 0 = DATI
0 | 1 = DATIP
1 | 0 = DATC
1 | 1 = DATOB

```

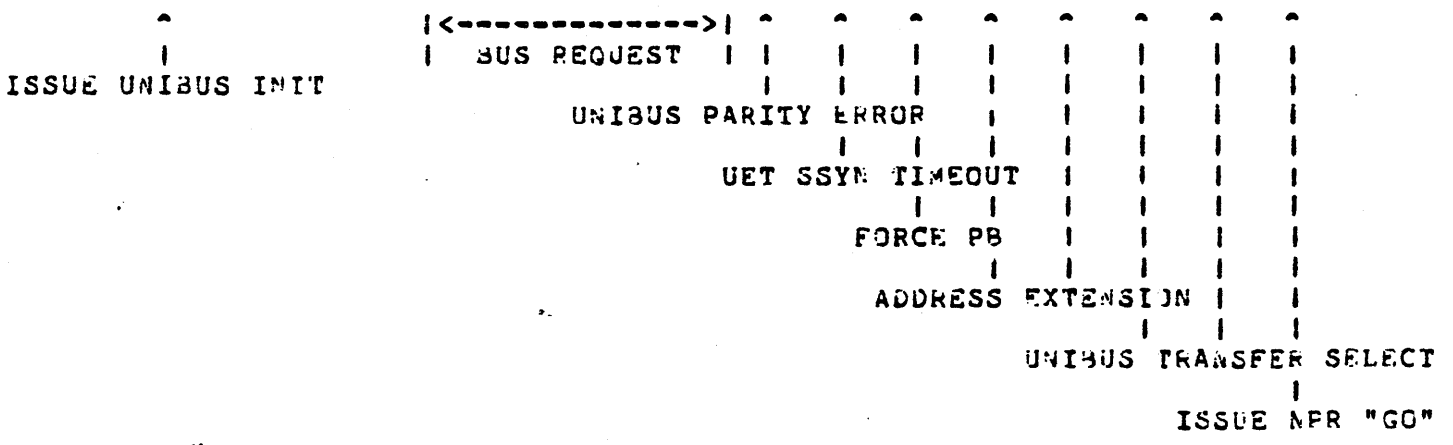
FFF460	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
--------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

UET UNIBUS ADDRESS REGISTER

FFF462	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
--------	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

UET UNIBUS DATA REGISTER

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FFF464	INIT				BR7	BR6	BR5	BR4	PE	TO	PB	A17	A16	C1	CO	NPR



## SUFFERED DATA PATH CSR'S

-----

BDP 1 = F30004            BDP 2 = F30008            BDP 3 = F3000C

-----

31   30   29	BITS 1 THROUGH 28 ARE NOT USED	0

-----

^   ^   ^

|   |   |

ERROR BIT |

(OR OF BITS 30 AND 29)

|   |

NXM |

UCE

^

PURGE BIT

IF UNIBUS DATA: SEND IT TO THE CMI

IF CMI DATA: CLEAR THE BUFFER

## CMI MAP DATA FIELDS

-----

ADDRESSES F30800 THROUGH F30FFC

31,30<----->26,25,24,23,22,21,20<----->15,14<----->0

	NOT USED	NOT USED	NOT USED
V	IO	DP   DP	PAGE FRAME NUMBER

-----

^

VALID BIT

ES THIS MAP VALID

|

BYTE OFFSET BIT

USED TO ACCESS ODD

BYTE BOUNDARIES

^   ^

DATA PATH SELECT BITS

(SEE THE CHART)-----> BIT 22 | 21

|

|

|

|

DIRECT DATA PATH = 0 | 0

BUFF DATA PATH 1 = 0 | 1

BUFF DATA PATH 2 = 1 | 0

BUFF DATA PATH 3 = 1 | 1



**DIAGNOSTICS**

## DIAGNOSTICS AND LEVELS

THE SPECIFIC LEVELS OF ALL DIAGNOSTICS CAN BE FOUND IN  
EVNDX ON MICROFICHE.

DIAGNOSTIC LEVELS  
-----

- LEVEL 1) RUNS ONLINE ONLY, WITHOUT DIAG. SUPERVISOR.  
(UETP, ERRLOG, SDA ETC.)
- LEVEL 2) RUNS ONLINE OR OFFLINE, UNDER DIAG. SUPERVISOR.  
(DISK FORMATTERS OR DEVICE RELIABILITY ETC.)
- LEVEL 2R) RUNS ONLINE ONLY, WITH DIAG, SUPERVISOR.  
(RESTRICTED DUE TO REQUIRED DEVICE DRIVER UNDER VMS)
- LEVEL 3) RUNS OFFLINE ONLY, UNDER DIAG. SUPERVISOR.  
(MAJORITY OF DIAGNOSTICS FOR REPAIR LEVEL)
- LEVEL 4) RUNS OFFLINE ONLY, WITHOUT DIAG. SUPERVISOR.  
(STANDALONE AND BOOTABLE DIAGNOSTICS)
- LEVEL 5) MICRO-DIAGNOSTICS

DISTRIBUTION OF DIAGNOSTICS TO TAPES IS SUBJECT TO VARIATION.

DIAGNOSTICS **97** LEVEL

TAPE AA)	CANCELLED		
TAPE #1)	ECKAA.EXE	4	MICMON
	ECKAB.EXE	4	DPM MICRODIAGNOSTICS
	ECKAF.EXE	4	RDM DIAGNOSTICS
TAPE #2)	ECKAA.EXE	4	MICMON
	ECKAC.EXE	4	MIC MICRODIAGNOSTICS
	ECKAF.EXE	4	RDM DIAGNOSTICS
TAPE #3)	CANCELLED		
TAPE #4)	CANCELLED		
TAPE #5)	ECKAL.EXE	4	CACHE AND TB DIAGNOSTICS
	ECKAM.EXE	3	MAIN MEMORY DIAGNOSTICS
	ECKAX.EXE	3	"CLUSTER" EXERCISER
TAPE #6)	ECSAA.EXE	N/A	DIAG. SUPERVISOR
	ECSAA.HLP		HELP FILE
	EVSBA.EXE	3	VAX AUTOSIZER
	EVSBA.HLP		HELP FILE
	CONFIG.COM or UBIATT.COM (CONFIGURATION FILES FOR THE 11/750 DIAGNOSTICS)		
TAPE #7)	EVKAA.EXE	4	"HARDCORE" INSTRUCTION TESTS
TAPE #8)	EVKAB.EXE	2	VAX-11 ARCHITECTURAL INSTRUCTIONS
	EVKAC.EXE	2	VAX-11 FLOATING POINT INSTRUCTIONS
	EVKAD.EXE	2	VAX-11 COMPATIBILITY MODE INSTRUCTIONS
	EVKAE.EXE	3	VAX-11 PRIVILEGED ARCHITECTURE INSTRUCTIONS
TAPE #9)	EVQDB.EXE	3	LOADABLE DRIVER FOR RP04/5/6
	EVQDR.EXE	3	LOADABLE DRIVER FOR RK03/5
	EVQDM.EXE	3	LOADABLE DRIVER FOR RK06/7
	EVQDL.EXE	3	LOADABLE DRIVER FOR RLO2
	EVAGA.EXE	2R	VAX-11 CR11 DIAGNOSTIC
	EVRAA.EXE	2	VAX-11 RP/RM/RK RELIABILITY DIAGNOSTICS
	EVRAC.EXE	2	VAX-11 RP/RM/RK DISK FORMATTER

TAPE #10)	EVOMA.EXE	3	VAX-11 M9203 REPAIR LEVEL DIAG.
	EVXA.EXE	3	VAX-11 COM IOP REPAIR LEVEL DIAGNOSTIC
	EVDAA.EXE	3	VAX-11 DZ11 8 LINE ASYNC. MUX
TAPE #11)	ECSAA.EXE	N/A	<u>DIAG. SUPERVISOR</u>
	EVREA.EXE	3	VAX-11 RK611 DIAG. PART A
	EVREB.EXE	3	VAX-11 RK611 DIAG. PART B
TAPE #12)	EVREC.EXE	3	VAX-11 RK611 DIAG. PART C
	EVRED.EXE	3	VAX-11 RK611 DIAG. PART D
	EVREE.EXE	3	VAX-11 RK611 DIAG. PART E
TAPE #13)	EVREF.EXE	3	VAX-11 RK611/RK06/7 DRIVE FUNCTIONAL PART 1
	EVREG.EXE	3	VAX-11 RK611/RK06/7 DRIVE FUNCTIONAL PART 2
TAPE #14)	EVGDR.EXE	3	LOADABLE DRIVER FOR RM03/5
	EVROA.EXE	3	VAX-11 RM03/5 DISKLESS DIAG.
	EVROB.EXE	3	VAX-11 RM03/5 FUNCTIONAL DIAG.
TAPE #15)	EVQTS.EXE	3	LOADABLE DRIVER FOR TS-11
	EVMAA.EXE	2	VAX TM03/TS11/TU78 DATA RELIABILITY DIAGNOSTIC
	EVMAO.EXE	3	VAX TS11 SUBSYSTEM REPAIR DIAG.
TAPE #16)	EVRF A.EXE	3	VAX-11 RL02 DISK SUBSYSTEM FUNCTIONAL DIAGNOSTIC
	EV RGA.EXE	3	VAX-11 RM80 DISK FORMATTER
	EV RGB.EXE	3	VAX-11 RM80 DISK DRIVE FUNCTIONAL DIAGNOSTIC
TAPE #17	ECCAA.EXE	3	RH750 (48A) DIAGNOSTICS
	ECCBA.EXE	3	DW750 (USI) DIAGNOSTICS
	ECSAA.EXE	N/A	<u>DIAGNOSTIC SUPERVISOR</u>

-----

THE DIAGNOSTICS LISTED ABOVE ARE ONLY THOSE WHICH APPEAR  
IN THE STUDENT GUIDES, THESE ARE THE MAIN DIAGNOSTICS FOR  
THE 11/750 SYSTEM "PACKAGES". ADDITIONAL DIAGNOSTICS  
FOR PERIPHERAL DEVICES AND COMMUNICATION EQUIPMENT CAN BE  
FOUND IN EVVX MICROFICHE.

-----

COPYING DIAGNOSTIC MEDIA

---

COPYING THE DIAGNOSTIC MEDIA TO THE SYSTEM DEVICE. THE PROCEDURE IS EXPLAINED FOR THE 3 POSSIBLE DIAGNOSTIC MEDIA.

NOTE: VERSION 3.X VMS TAKES UP CONSIDERABLY MORE DISK SPACE THAN PREVIOUS VERSIONS. THERE WILL NOT BE ROOM FOR ALL OF THE VAX SYSTEM DIAGNOSTICS ON THE PACK. IT IS RECOMMENDED THAT YOU BE SELECTIVE OF THE DIAGNOSTICS NEEDED OR PUT ALL DIAGNOSTICS ON A SEPARATE PACK OR MAGTAPE.

1. TU58 DIAGNOSTIC DISTRIBUTION (RK07 PACKAGE SYSTEM)

A. IF THE DIAGNOSTIC UPDATE OR DIAGNOSTIC KIT HAS TO ENTERED INTO THE SYSTEM FROM TU58 CARTRIDGES, THE FLX UTILITY MAY BE USED TO ACCOMPLISH THE TRANSFER.

B. PERFORM THE FOLLOWING COMMANDS ONCE THE NEW VMS SYSTEM HAS BEEN INSTALLED AND BOOTED. LOG INTO THE SYSTEM MANAGER'S ACCOUNT TO PERFORM THIS PROCEDURE.

C.       \$ RUN SYS\$SYSTEM:SYSGEN  
 SYSGEN>CONNECT CONSOLE  
 SYSGEN>EXIT  
 \$ MOUNT CS1:/FOR  
 \$ SET DEF SYS\$MAINTENANCE  
 \$ MCR FLX  
 FLX>/RS=CS1:\*/RT  
 FLX>

D. ALL THE DIAGNOSTICS AND FILES ON THIS TAPE HAVE BEEN TRANSFERRED TO THE [SYSMAINT] DIRECTORY AT THIS POINT.

E. INSERT THE NEXT TAPE AND REPEAT THE FLX> COMMAND TO COPY THE NEXT TAPE.

FLX> /RS=CS1:\*/RT  
 FLX>

F. REPEAT THIS PROCESS UNTIL ALL TAPES HAVE BEEN TRANSFERRED TO THE [SYSMAINT] AREA.

G. A CONTROL Y WILL EXIT FROM THE FILEX UTILITY.

FLX>Y

## 2. RK07 VAXPAX DIAGNOSTIC DISTRIBUTION (DUAL RK07 PACKAGE SYSTEMS)

- A. MOUNT THE VAXPAX DISK CARTRIDGE IN DRIVE 1.
- B. MOUNT THE NEWLY CREATED VMS SYSTEM IN DRIVE 0.
- C. BOOT THE NEW VMS SYSTEM FROM DRIVE 0 AND LOG INTO THE SYSTEM MANAGERS ACCOUNT. PERFORM THE FOLLOWING COMMANDS TO TRANSFER THE VAXPAX DIAGNOSTICS TO THE SYSTEM DEVICE.
- ```
$ MOUNT DMA1: VAXPAX
$ SET DEF SYS$MAINTENANCE
$ COPY DMA1:[SYSMAINT]*.*;* *
$ DIR/FULL DIAGBOOT.EXE,ECSAA.EXE,CONFIG.COM
```
- D. MAKE CERTAIN THAT DIAGBOOT.EXE, ECSAA.EXE AND CONFIG.COM ARE CONTIGUOUS DISK FILES. IF THEY ARE NOT COPY THEM TO THEMSELVES USING THE /CONTIG SWITCH, THEN PURGE THE OLD VERSIONS OUT OF THE DIRECTORY.
- ```
$ COPY/CONTIG DIAGBOOT.EXE,ECSAA.EXE,CONFIG.COM *
$ PURGE DIAGBOOT.EXE,ECSAA.EXE,CONFIG.COM
```
- E. THIS COMPLETES THE DIAGNOSTIC TRANSFER TO THE SYSTEM DEVICE. YOU MAY WISH TO DELETE ANY OF THE DIAGNOSTICS WHICH RELATE TO THE 11/780 AND 11/730.
- ```
$ DELETE ES*.*;*
$ DELETE EN*.*;*
```

## 3. MAGTAPE VAXPAX DIAGNOSTIC DISTRIBUTION (RM03/TS11 OR RM80 PACKAGE SYSTEMS)

- A. THIS PROCEDURE WILL TRANSFER THE DIAGNOSTIC MEDIA FROM THE TS11 MAGTAPE TO THE SYSTEM DEVICE DRA0:.
- B. BOOT THE NEWLY CREATED VMS SYSTEM AND LOG INTO THE SYSTEM MANAGERS ACCOUNT. PERFORM THE FOLLOWING COMMANDS TO TRANSFER THE MAGTAPE DISTRIBUTION TO THE [SYSMAINT] AREA OF THE NEWLY CREATED DISK.
- ```
$ MOUNT MSA0: VAXPAX
$ SET DEF SYS$MAINTENANCE
$ COPY MSA0:*.*;* *
```
- C. THIS WILL TRANSFER ALL THE DIAGNOSTICS ON THE MAGTAPE TO THE [SYSMAINT] AREA OF THE NEW DISK.
- D. MAKE CERTAIN THAT DIAGBOOT.EXE, ECSAA.EXE AND CONFIG.COM ARE CONTIGUOUS DISK FILES. IF THEY ARE NOT COPY THEM TO THEMSELVES USING THE /CONTIG SWITCH, THEN PURGE THE OLD VERSIONS OUT OF THE DIRECTORY.
- ```
$ COPY/CONTIG DIAGBOOT.EXE,ECSAA.EXE,CONFIG.COM *
$ PURGE DIAGBOOT.EXE,ECSAA.EXE,CONFIG.COM
```
- E. THIS COMPLETES THE DIAGNOSTIC TRANSFER TO THE SYSTEM DEVICE. YOU MAY WISH TO DELETE ANY OF THE DIAGNOSTICS WHICH RELATE TO THE 11/780 AND 11/730.
- ```
$ DELETE ES*.*;*
$ DELETE EN*.*;*
```

Title: Autosizer  
 Author:  
 Processor Applicability: VAX Family

Have all your VAX 11/780 and 11/750 systems CONFIGURED for you using EVSBA.EXE released in the Diagnostic Update release 3. This program, available after November 1981, will pass configuration information on to the Diagnostic Supervisor. It builds a series of ATTACH commands based on the hardware it found during its sizing process which is passed on to the Supervisor and may be written to the console load media for later use. You will no longer need to build a configuration command file! It will be built for you!

#### PERTINANT INFORMATION

1. The program is a level 3 standalone program that runs under the Diagnostic Supervisor.
2. It requires 256KB Memory, a Console Terminal and Load Device in working order.
3. The program operates in three modes: Default, Manual, and Selftest.

To select any of these modes type after the DS> prompt:

```

RUN EVSBA.EXE           for Default
RUN EVSBA.EXE/SECTION:MANUAL  for Manual
RUN EVSBA.EXE/SECTION:SELFTEST for Selftest
  
```

- 3.1 Default: In the default mode the program sizes the system passing the configuration information on to the Diagnostic Supervisor. After the execution of the program this information may be seen by the operator by typing SHOW DEVICE after the DS> prompt.

ALWAYS VERIFY this information when using this mode since the program makes educated guesses regarding some necessary information. In particular UNIBUS devices which use floating addresses for their Control Status Registers and vectors may be configured incorrectly for a particular system. (See EVSBA.DOC for further information)

- 3.2 Manual: The Manual option may be executed by typing the following command after the DS> prompt: RUN EVSBA.EXE/SECTION:MANUAL  
 This causes the program to give the following prompt: COMMAND?

Legal responses to this prompt are: ATTACH, CHANGE, EXIT, HELP, LIST, READ, SIZE, and WRITE. Explanations of these commands may be read by accessing the program's help file. (DS> H EVSBA HELP)

note: You must use an ATTACH command to pass the configuration information on to the Supervisor after SIZEing the system in this mode.

Commands READ and WRITE access only the console media.

- 3.3 Selftest: The Selftest option may be executed by typing the following command after the DS> prompt: RUN EVSBA.EXE/SECTION:SELFTST  
As in the Manual mode the COMMAND? prompt is given. Any manual mode command is valid. The primary difference between these two modes is that when using the SIZE command in the Selftest mode all the configuration information is shown to the operator on the console.
4. There is a "QUICK" execution of the program available in all modes. A response of SET FLAG QUICK to the DS> prompt prior to running the program will cause it to ignore the presence of terminals connected to a DZ11. For systems with a large number of terminals this can save a considerable amount of time and the program will proceed very quickly.
  5. The autosizer can be run from either the console subsystem or from the system diagnostic media. This fact can be useful in situations where mass storage devices are inoperable.
  6. A recommended sequence of operation is:
    1. Boot the Diagnostic Supervisor.
    2. DS> SET QUICK if quick execution is desired.
    3. DS> RUN EVSBA/SEC:SELFTST
    4. COMMAND? SIZE (sizes system)
    5. COMMAND? CHANGE (only if configuration information needs change)
    6. COMMAND? WRITE (writes configuration information to console media)
    7. COMMAND? ATTACH (passes configuration information to Supervisor)
    8. COMMAND? EXIT (exit from EVSBA back to the Supervisor)
    9. DS> SHOW DEVICE (to see results of autosizer)
    10. DS> CLEAR QUICK (clears the QUICK FLAG)
    11. DS> SELECT or DESELECT devices for running desired diagnostic
    12. DS> RUN desired diagnostic
    13. To copy CONFIG.COM file created by program to [SYSMAINT] (assuming that you logged into FIELD SERVICE) use the FILEX utility after Booting VMS as follows:
 

\$ MOUNT/FOR CS1:	MC SYSGEN
\$ MCR FLX	SYSGEN> CONNECT CONSOLF
FLX>=CS1:CONFIG.COM/RT	SYSGEN> EX
FLX>*Y	
\$ DISMOUNT CS1:	



UUT-Type	LINK	GENERIC	PARAMETERS	TYPICAL
AA11K	DWn	??an	CSR VCT BR	770460 350 5
AD11K	DWn	??an	CSR VCT BR	770400 xxx 6
CR11	DWn	CRa	CSR VCT BR	777160 230 4
DL11	DWn	??a	CSR VCT BR	
DHC11	DWn	XMa	CSR VCT BR	760050 xxx 5
DMP11	DWn	XDan	CSR VCT BR	
DHR11	DWn	XMan	CSR VCT BR	
DR11B	DWn	??a	CSR VCT BR	772410 124
DR11K	DWn	??a	CSR VCT BR	767770 xxx 4
DR11W	DWn	??a	CSR VCT BR	
DR780	SBI	XFn	TR BR	
DUP11	DWn	XJa	CSR VCT BR	
DV11	DWn	XVa	CSR VCT BR	775000
DW750	CHI	DWn	BR	
DW780	SBI	DWn	TR BR	3 4 (\$1 UBA)
DW780	SBI	DWn	TR BR	4 4 (\$2 UBA)
DZ11	DWn	TTa	CSR VCT BR EIA/20MIL	760100 xxx 5 EIA
KA750	CHI	KAn	G H TOY WCS ACC	NO NO YES 0 0
KA780	SBI	KAn	G H WCS ACC	No No 0 0
KMC11	DWn	XMan	CSR VCT BR	
KW11K	DWn	??a	CSR VCT BR	770404 xxx 6
LA34	TTa	TTan		
LA36	TTa	TTan		
LA38	TTa	TTan		
LA120	TTa	TTan		
LA180	LPa	LPan		
LP05	LPa	LPan		
LP06	LPa	LPan		
LP11	DWn	LPa	CSR VCT BR	777514 200 4
LP14	LPa	LPan		
LP25	LPa	LPan		
LPA11K	DWn	LAan	CSR VCT BR	770460 350 5
MA780	SBI	MAn	TR BR MPH PORT	
MBE	RHn	MBn	DRIVE #	
*MS750	CHI	MSn	BR	
MS780	SBI	MSn	TR	
PCL11	DWn	??a	CSR VCT BR	764200 170 x
RH750	CHI	RHn	BR	5
RH780	SBI	RHn	TR BR	8 5 (RH0)
RH780	SBI	RHn	TR BR	9 5 (RH1)
RK06	DMa	DMan		
RK07	DMa	DMan		
RK611	DWn	DMa	CSR VCT BR	777440 210 5
RL01	DLa	DLan		
RL02	DLa	DLan		
RL11	DWn	DLa	CSR VCT BR	774400 160 5
RM03	RHn	DRan		
RM05	RHn	DRan		
RM80	RHn	DRan		
RP04	RHn	DBan		
RP05	RHn	DBan		
RP06	RHn	DBan		
RP07	RHn	DRan		
RX02	DYa	DYan		
RX211	DWn	DYa	CSR VCT BR	777170 264 5
TE16	MTa	MTan		
TM03	RHn	MTa	DRIVE #	
TM78	RHn	MFa		
TS11	DWn	MSan	CSR VCT BR	772520 224 5
TU45	MTa	MTan		
*TU58 Unibus	DWn	DDan	CSR VCT BR	776500 xxx x
TU77	MTa	MTan		
TU78	MFa	MFan		
UBE	DWn	UBan	CSR VCT BR	
VT50	TTa	TTan		
VT52	TTa	TTan		
VT55	TTa	TTan		
VT100	TTa	TTan		

Note: The typical column is only a partial list because of the great amount of possibilities in configurations - these are by no means any sort of standard.

a = Alpha Character  
n = Numeric Character

## SOME BASIC CONSOLE COMMANDS

-----  
 UNDER CONSOLE I/O MODE >>>

EXAMINE >>>E/X/Y (ADDRESS IN HEX) <CR>

	B=BYTE		P=PHYSICAL
X=	W=WORD	Y=	V=VIRTUAL
	L=LONGWORD		I=IPR (SPECIFY REGISTER #)
			G=GPR (SPECIFY REGISTER #)

DEPOSIT >>>D/X/Y (ADDRESS IN HEX) (DATA IN HEX) <CR>

INITIALIZE >>>I

START >>>S <ADDRESS>

CONTINUE >>>C

BOOT >>>B/(FLAG)/(QUALIFIER) (DEVICE) <CR>

SOME FLAGS: /1 =CONVERSATIONAL BOOT  
 /10 =DIAG.SUPERVISOR  
 /100 =SOLICIT FILE NAME

ENTER RDM MODE FROM CONSOLE I/O MODE = >>>^P  
 >>>^D

ENTER RDM MODE FROM VMS = s@SYSSSYSTEM:SHUTDOWN or s@[SYSEXE]SHUTDOWN

THEN ^P  
 >>>^D

## SOME COPY COMMANDS

-----  
FILEX COPY FROM DISK (DEFAULT DRIVE) TO TU-53  
-----

```

$ RUN SYSSSYSTEM:SYSGEN
SYSGEN>CONNECT CONSOLE
SYSGEN>EXIT
$ MOUNT CS1:/FOREIGN
$ SET DEFAULT [SYSMAINT]
$ MCR FLX
FLX>CS1:/RT/ZE          ZERO OUT EXISTING DIRECTORY ON TAPE
                        (OPTIONAL)
FLX>CS1:/RT/LI          LIST DIRECTORY (OPTIONAL)
FLX>CS1:/RT/XX = {device|filename.ext/RS <CR>
                  {TO}      {FROM}
                  XX = IM = IMAGE MODE (EXE,ULB,SMC,SYS,OLB,TSK)
                        DE = DELETE THE SPECIFIED FILE
                        FB = FORMATTED BINARY (OBJ,STB,BIN,LDA)
                        FA = FORMATTED ASCII (ALL OTHER EXTENSIONS)
                        CO = CONTIGUOUS FILE TO DISK (FILES COMMING
                           FROM DISK TO TAPE ARE ALWAYS CONTIGUCUS)
                        RS = RS-11 FORMAT (SYSTEM)
                        RT = RT-11 FORMAT (TAPE)

```

```

FLX>^Y

```

```

$

```

-----  
TO COPY FROM TUSE TO DISK (RK07)  
-----

```

$ RUN SYSSSYSTEM:SYSGEN
SYSGEN>CONNECT CONSOLE
SYSGEN>EXIT
$ MOUNT CS1:/FOREIGN
$ SET DEFAULT [SYSMAINT]
$ MCR FLX
FLX>DM0:/RS/CO=CS1:filename.ext/RT
                  {TO}      {FROM}
FLX> ^Y
$

```

-----  
TO REBUILD VAB.EXE  
-----

SAME AS ABOVE COPY PROCEDURE BUT

```

$ SET DEF [SYSFXE]

```

NOTE: WHEN USING FILEX UNDER VERSION 3.X VMS, THE DEVICE NEMONICS (DDU) MAY NOT BE RECOGNIZED SINCE VERSION 3.X LIKES LOGICAL NAMES INSTEAD OF DEVICE NAMES. UNFORTUNATELY FILEX DOES NOT LIKE LOGICAL NAMES. SO... TO AVOID UNKNOWN DEVICE ERRORS YOU MUST EITHER SET YOUR DEFAULT TO THE CORRECT DIRECTORY FIRST AND GIVE THE DESTINATION SPEC.

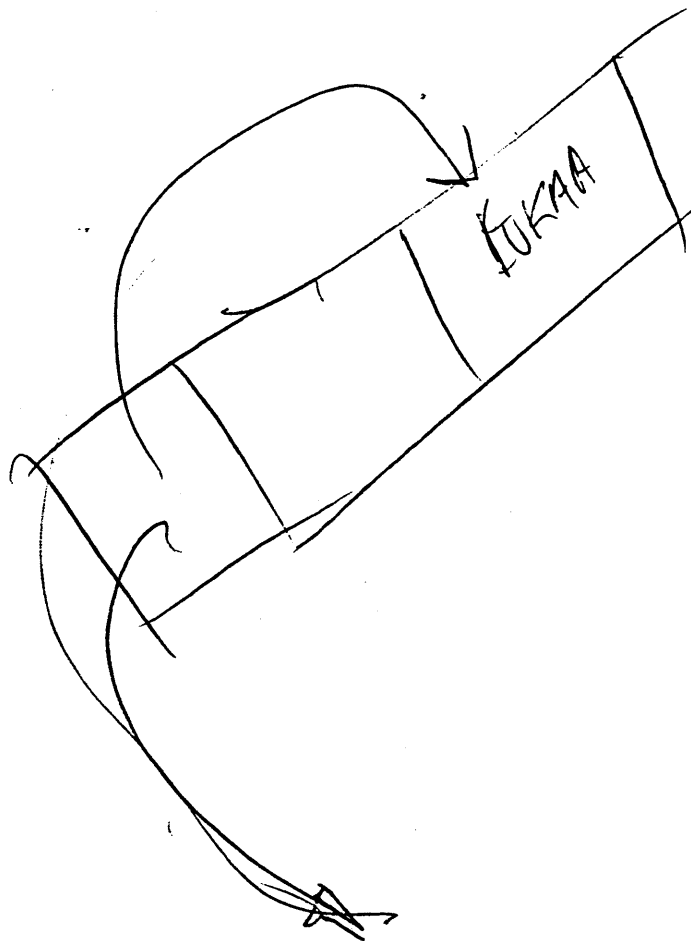
EXAMPLE: S SET DEF SYSSMAINTENANCE  
S MC FLX  
FLX> CSI:/RT=FILENAME.EXT/RS

OR... BE SURE TO USE THE FULL FILESPEC AS REQUIRED BY VERSION 3,

EXAMPLE: FLX> CSI:/RT=DPO:[SYSO.SYSMAINT]FILENAME.EXT/RS

# 106

\$ XFFR VME.EYE  
\$ MCR WRITE



# 107

## TO WRITE A BOOTBLOCK ON TAPE OR DISK

-----

\$ MCR WRITEBOOT

Target System Device (and bootfile if not VMB.EXE)...: DDCU:filename.ext

Enter VBN to Boot File (default is 1)...: 1 or 2

Enter Load Address (default is 200) ...: 10000 or 200 or C000

\$

DDCU = DEVICE NAME, CONTROLLER AND UNIT #  
(CSA1:,DMA0:, ETC.)

LEVEL 4 DIAGNOSTICS AND MONITORS ARE THE ONLY BOOTABLE PROGRAMS

THE VAX 11/750 HAS FOUR BOOTABLE PROGRAMS AT THIS TIME...

PROGRAM NAME	VBN	LOAD ADDRESS	DESCRIPTION
-----	---	-----	-----
EVKAA.EXE	2	200	HARDCORE
ECKAL.EXE	2	200	TB AND CACHE
ECSAA.EXE	2	10000	DIAGNOSTIC SUPERVISOR
BOOT58.EXE	1	C000	BOOT58 MONITOR

NOTE: TO REBUILD A BOOTBLOCK ON THE SYSTEM DISK, SIMPLY SPECIFY THE  
DISK'S NAME (DDCU:) AND STRIKE 3 <CR>'S

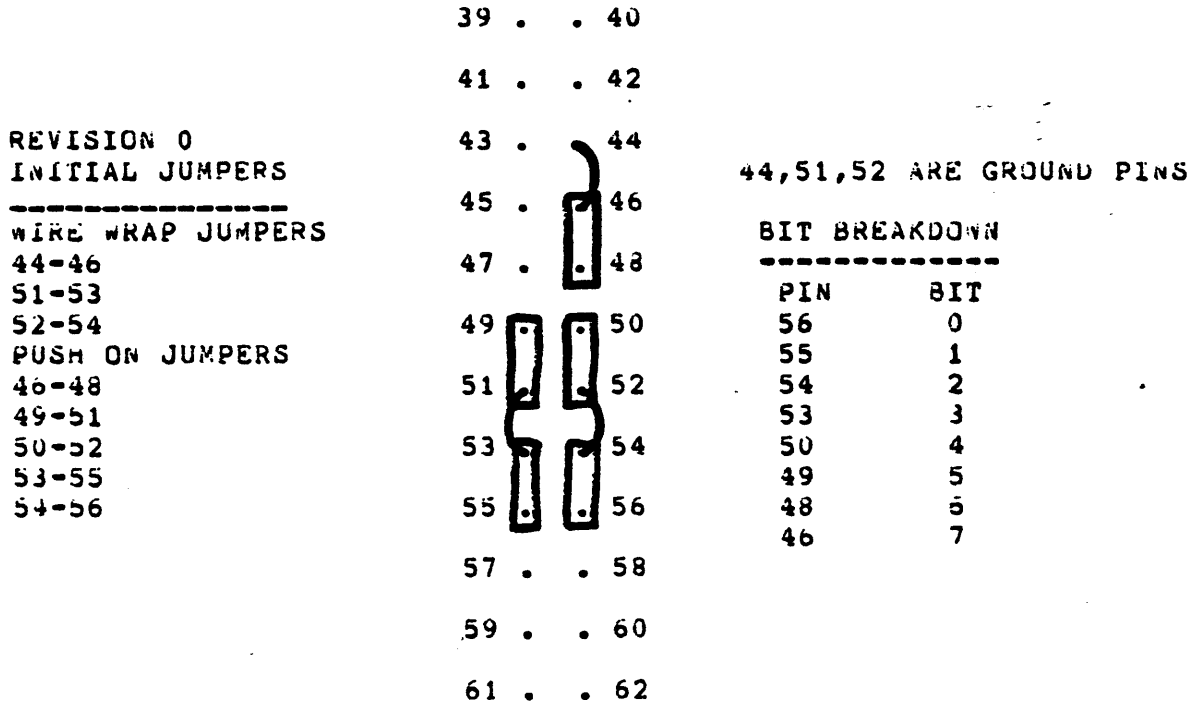
NOTE: WHEN USING WRITEBOOT UNDER VERSION 3.X VMS, THE DEVICE MNEMONICS  
(DDCU) MAY NOT BE RECOGNIZED SINCE VERSION 3.X LIKES LOGICAL  
NAMES INSTEAD OF DEVICE NAMES. SO... TO AVOID UNKNOWN DEVICE  
ERRORS YOU MUST BE SURE TO USE THE CORRECT LOGICAL NAME OR USE  
THE LONG DIALOGUE FOR THE DEVICE AND DIRECTORY SPECS.

EXAMPLE: DRA0:[SYS0.SYSMAINT]FILENAME.EXT

**REVCON**

BACKPLANE WIRING FOR SID REGISTER IPR 3E <7:0>

SLOT 4 SECTION B



44,51,52 ARE GROUND PINS

BIT BREAKDOWN

PIN	BIT
56	0
55	1
54	2
53	3
50	4
49	5
48	6
46	7

- HARDWARE REV. 0 = ALL JUMPERS INSTALLED IPR 3E = 00
  - HARDWARE REV. 1 = REMOVE JUMPER 54-56 (FLOATS BIT 0 HI) IPR 3E = 01
  - HARDWARE REV. 2 = REMOVE JUMPER 53-55 (FLOATS BIT 1 HI) IPR 3E = 02  
AND RE-INSTALL JUMPER 54-56 (GROUNDS BIT 0)
  - HARDWARE REV. 3 = REMOVE JUMPER 54-56 AND 53-55 IPR 3E = 03  
(FLOATS BITS 0 AND 1)
- ETC.

PUSH ON JUMPER PART NUMBER = 12-14314-00

Date: 8/1/92

Updates:

CSSE  
VW01-1/C05

## 11/750 REVISION CONTROL DOCUMENT

-----

```

=====
|CHANGES IN THIS UPDATE:
|-----
|
| . Added compatibility chart (1.1)
| . Added Kernal ID Register and SID Switch Info (3.0)
| . Revised and updated 11750 Kernal Rev history (4.0)
|   - Rev B and Rev C backplane compatibility info
| . Added MS750-CA memory option info (5.2)
| . Updated info on all options (5.0)
|-----
=====
    
```

### TABLE OF CONTENTS

-----

	Page
1.0 INTRODUCTION.....	2
1.1 11/750 Compatibility Chart	
2.0 REVISION CRITERIA.....	3
3.0 11/750 KERNAL REVISION CONTROL.....	3
3.1 Kernal ID Register H/W Rev Level Input Device (SID Switch Pack)	
3.1.1 Electrical Requirements	
3.1.2 Installation Procedures	
3.2 SID Register Problem	
3.3 Microcode Revision	
4.0 11/750 KERNAL REVISION HISTORY.....	8
4.1 KA750 Module Revision Charts	
4.2 KA750-00 Revision Summary (LR)	
4.3 KA750-01 Revision Summary (VAX750-M-0001)	
4.4 KA750-02 Revision Summary (VAX750-R-0002)	
4.5 KA750-03 Revision Summary (VAX750-R-0003)	
4.6 KA750-38 Revision Summary	
5.0 OPTION REVISION CONTROL.....	15
5.1 MS750-AA	
5.2 MS750-CA	
5.3 RH750	
5.4 FP750	
5.5 KU750	
5.6 DW750	
5.7 DR750	



## INTRODUCTION

This document is intended to define the revision history of the VAX 11/750 system for purposes of identifying the compatible diagnostics, firmware, and operating system software. (All references to the 11750 apply to the 11751 as well, unless otherwise noted.)

This document will be updated on a quarterly basis and released to microfiche in the VAX Library Update and the Speed Bulletin.

The following VAX LIBRARY CARDS will be required as reference to the revision control plan. This document will state the source of information necessary for understanding firmware, diagnostic, and system software compatibility.

DOCUMENT	INITIAL STARTING DOCUMENTATION DESCRIPTION
ZZ-EVNDX-W.0	VAX Diagnostic Index
ZZ-ECDAB-1.0	VAX 11/750 Control Store Microcode Listing CMT050
ZZ-ECOAD-1.0	VAX 11/750 Boot Rom Listing TU58 RK07 RL02

### 1.1 11/750 COMPATIBILITY CHART

The following chart summarizes the compatible hardware, software, microcode and diagnostic revision levels for each kernel revision. For information concerning the module revisions for each option revision, see Section 5.0.

11750 KERNAL REV WITH SID	00	10	20	130/38	4X	5X	6X
11750 KERNAL REV w/OUT SID	00	01	02	03			
H/W Options:							
KA750	00	01	02	03			
MS750-AA	00	00	00	01			
MS750-CA	-	-	-	-			
RH750	-	-	00	01			
FP750	-	-	-	01			
KU750	-	-	-				
DR750							
DW750							
DX: * EVNDX	IV - 7	IV - 7	IV - 7	IV - 7			
RELEASE:	2.X	2.X	2.X	2.X,3			
MICROCODE:	050	052	062	094			

\* Check VAX Diagnostic Evaluations (distributed in the Speed Bulletin) for bugs that exist in a particular diagnostic release.

## 2.0 REVISION CRITERIA

Architectural or functional changes in the system must cause a change of the kernel revision level. This includes changes to the hardware that changes performance or operation of the system that is detectable to the diagnostics and operating system software. Modifications to cables, power subsystems, ventilation equipment, or mechanical design should not cause a change to the kernel revision level. The VAX 11/750 revision history for a particular subsystem or option will be designated as follows...

## OPTION-XXN

where, OPTION is the five letter neumonic (i.e. KA750), XX is a functional change description code (i.e. 01), and N is a non-functional change to the option. Non-functional changes to an option include relayout of modules to eliminate rework wires or changes to documentation that do not affect operation of the system. Changes to purchase part numbers at the module level should not cause the hardware revision to be raised either. Functional changes to any of the modules in the kernel subsystem (i.e. KA750 CPU and MS750) increments each module revision, as well as the kernel revision. A brady marker indicating the module revision should be wrapped around the third tab down from the top of the module handle. Manufacturing will be instructed to attach the brady markers as soon as possible.

A switch pack and pull up resistor assembly has been designed and will be installed in all Rev "C" backplanes shipped from manufacturing. The switch pack will be set by manufacturing before it is shipped. An engineering spec has been written which contains instructions as to how the switch pack should be set. This document will be under ECO control, so that when there is an ECO that changes the switch pack setting (i.e. a change to the kernel revision), the document will be ECO'd and the new switch pack settings added. (See Section 3.1 for more detailed information on the switch pack).

## 0 VAX 11/750 KERNAL REVISION LEVEL

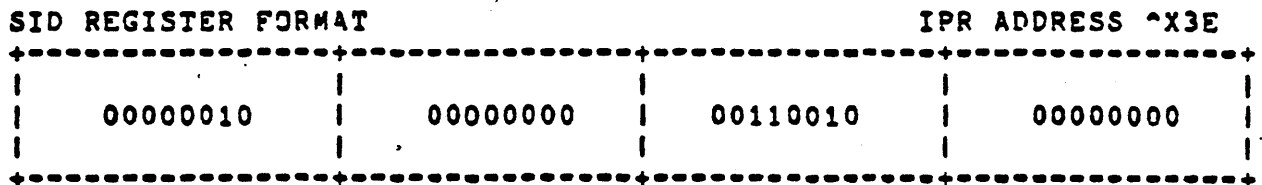
The VAX 11/750 processor has a system identification register (SID) register similiar to the VAX 11/780 processor; however, there are two basic differences between the VAX 11/750 and VAX 11/780 processor SID registers. These are...

- \* VAX 11/750 SID does NOT contain the system serial number
- \* VAX 11/750 SID DOES contain a field describing current control store microcode revision. This is because VAX 11/750 does not have the FPLA to intercept ECO'd microcode locations.

The figure 2-1 illustrates the format of the VAX 11/750 system identification register (SID). This register is accessible to macro code and the console terminal through IPR address ^X3E. Examination of the register shows 4 bytes, 3 of which are functional. The high byte is the processor type code byte used by the diagnostics and operating system to decide which type of processor the software is operating on. Known type codes are listed below.

TYPE CODE BYTE	PROCESSOR
00000000	undefined
00000001	VAX 11/780
00000010	VAX 11/750
00000011	VAX 11/730

Byte 2 of the SID register is always zero. Byte 1 of the SID register is the microcode revision of the control store. This number is generated by the microprogrammer in the REV750.MIC file of the microcode listing ECDAB. There is a MICRO2 assembler directive called .SET/MICROREV=version at the top of the page which is upgraded for each major assembly of the the microcode. In the MFPR microinstruction flows this equated value is substituted into the short literal field of the microinstruction and becomes the microcode revision that appears in byte 1 of the SID register. The number following MICROREV is DECIMAL. You must convert the hex byte to decimal after examining the SID register. Byte 0 of the SID register is the hardware revision of the kernel. The hardware revision is programmable on the CPU backplane. There is a 74LS244 tri-state driver on the UBI module that interfaces to the CPU wBUS. The MFPR microinstruction flows read this byte when referencing the SID register. At Limited Release (LR) this byte should be 00. That means all the bits are grounded on the CPU backplane. (See Section - Electrical Requirements of Kernel Revision Level Input Device.) Each hardware change that changes the functionality of the hardware will INCREMENT the hardware revision by one. The number in byte 0 of the SID register is a BINARY revision level programmed on the SID input device.



TYPE CODE = 2	MUST BE 0	MICROCODE REV	HARDWARE REV
		= 02	2
		= 04	3

Figure 2-1

NOTE: This example shows the type code as 2 (VAX 11/750), Microcode revision is 32 hex or 50 decimal, and the hardware revision level is equal to zero zero.

### 3.1 Kernel Identification Register Hardware Revision Level Input Device Description

The Hardware revision level of the kernel that is visible in byte 0 of the SID register is generated by a 16 pin DIP switchpack consisting of 8 single-pole single-throw switches that ground or open SID bits <7:0> to produce a binary number corresponding to the kernel revision level. Each bit of byte 0 of the SID is pulled up to +5V through a resistor contained in a 14 pin DIP pull-up resistor package.

The Kernal Rev Level Input device is manufactured using a 1.6" X 2.6" standard size fingerless board that has an edge mounted 40 pin AMP connector to press on the backplane of the system on slot 4. There are 2 ICs on this device, they are the DIP switch-pack assembly and DIP pull-up resistor package. This SID switch, part number < ??????? > is manufactured with shelf items listed below.

#### Kernal Hardware Revision Level Input Device Parts List

Qty.	Description	DEC PN
1	40 pin edge connector	12-11620-00
1	DIP rocker-switch 8 sw	12-11164-04
1	DIP 4.7K terminator	13-00005-00
1	PC board 1.6" X 2.6"	50-15141-00
1	Housing backplane conn	12-16821-00

#### 3.1.1 Electrical Requirements of the Kernal Revision Level Input Device

The kernal revision level input device requires +5V and ground to operate properly. The input signals SYS ID <7:0> H must also be interfaced to the input device. The electrical connection is made via the 40 pin AMP connector that is pressed on the backplane. The following list describes signal locations on the backplane and AMP connector. A signal with a dash "-" implies a no connect to the input device.

Signal	AMP pin	11/750 Backplane 400Bxx	AMP pin	Signal
-	A	19	20	B
-	C	21	22	D
GND	E	23	24	F
-	H	25	26	J
-	K	27	28	L
-	M	29	30	N
-	P	31	32	R
-	S	33	34	T
-	U	35	36	V
-	W	37	38	X
-	Y	39	40	Z
-	AA	41	42	BB
-	CC	43	44	DD
-	EE	45	46	FF
-	HH	47	48	JJ
SYS ID 5 H	KK	49	50	LL
-	MM	51	52	NN
SYS ID 3 H	PP	53	54	RR
SYS ID 1 H	SS	55	56	TT
-	UU	57	58	VV
				+5V

Power consumption with all switches closed is approximately equal to 42 milliwatts.

## .1.2 Kernal Revision Level Input Device Installation Procedure

1. Remove primary power from the system by turning CBI to OFF position.
2. Open rear door of the VAX 11750 and remove the backplane cover plate by loosening 4 screws and lifting off.
3. Install the Backplane Connector Housing (12-16821) on slot 4 of the CPU backplane so that the blind holes at each end of the connector cover pins B40017, B40018 on top, and B40059 and B40060 on the bottom.
4. Set the binary revision level on the switch to desired number (see table two page 3) according to the following example...

	Example is for:		Hardware revision level 20 Hex					
SYS ID	<7>	<6>	<5>	<4>	<3>	<2>	<1>	<0>
SWITCH	S8	S7	S6	S5	S4	S3	S2	S1
	ON	ON	OFF	ON	ON	ON	ON	ON

NOTE: Early SID switch modules have the switch pack reversed.  
 ---- Use etched bit position on board for reference. Disregard switch positions marked on switch pack.

When the switch is ON, ground is connected to the input of the 74LS244 on the L0004 (UBI) module producing a "0" data bit in byte 0 of the SID register. If the switch is OFF the current path is removed and the inputs to the 74LS244 are pulled up to +5V causing a "1" to be generated in the that bit position.

5. Install the Kernal Rev Level Input Device in the backplane connector housing with component side (Side 1) facing the right side of the VAX 11750 CPU cabinet (when viewed from the rear).
6. Secure backplane cover plate and rear door of the VAX 11/750 and set the POWER ON ACTION switch to HALT. Turn CBI to the ON position.
7. Verify the hardware revision level by examining the SID register in console mode by typing...(at the console)

```
>>>E/I 3E<CR>
-I 0000003E 02003E20
```

This example shows a Kernal that has CMT062 microcode and a hardware revision of 20. (See Tables 1 and 2 on next page).



## 3.3 Microcode Revision History

A comprehensive revision history of VAX 11/750 microcode is contained in the microcode listing in the VAX LIBRARY fiche set. Revision W of the VAX LIBRARY contains the microcode listing for control store version CMT050. Since there have been numerous changes to the microcode since version CMT050, it is suggested that you read the microcode revision history contained in the REV750.MIC file of the microcode listing. Changes to the VAX 11/750 microcode AFTER CMT062 will be described briefly in this document and what the symptoms and corrective actions were.

## 4.0 11/750 KERNAL REVISION HISTORY

-----

Listed below is the revision history of the 11750 kernal, indicating the compatible module revisions for each hardware revision. Equivalent module revisions are separated by commas.

This history has been separated into two charts:

Chart 1 - 11/750 with a Rev "B" backplane  
SID Switch Setting = xxxxxxx0-7

Chart 2 - 11/750 with a Rev "C" backplane  
SID Switch Setting = xxxxxxx9-F

### KA750 Module Revision Charts

CHART 1

-----

11750 KERNAL REV WITH SID		00		10		20		30		4X		5X	
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
11750 KERNAL REV W/OUT SID		00		01		02		03					
=====	=====	=====	=====	=====	=====	=====	=====	=====	=====	=====	=====	=====	=====
MODULE		SLOT											
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
L0002		2		B		C		C		C,D			
L0003		3		B		C		C,D		C,D			
L0004		4		E		F		H,J		H,J			
L0005		5		D		E		F		H			
L0011		10		D		D		D		D			
L0016		10		*,A		*,A		*,A		*,A			
M8728		11-18		C		C		C		C			
M9313		28 A-B		A		A		A,B		A,B			
TUSB		UNIBUS		F		F		F		F			
TUSB		CON		B		B		B		B			
CONT PANEL				C		C		C		C			
BACKPLANE				A		B		B		B			
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

\* = LR Release

CHART 2

11750 KERNAL REV WITH SID		00	10	20	38	4X	5X
11750 KERNAL REV W/OUT SIDI		00	01	02	03		
MODULE	SLOT						
L0002	2	B	C	C	C,D		
L0003	3	B	C	C,D	C,D		
L0004	4	E	F	H,J	H,J		
L0005	5	D	E	F	H		
L0011	10	D	D	D	N/A		
L0016	10	*,A	*,A	*,A	*,A		
M8728	11-18	C	C	C	C		
M8750	11-18	N/A	N/A	N/A	N/A		
M9313	28 A-B	A	A	A,B	A,B		
TU58	UNI BUS	F	F	F	F		
TU58	CON	B	B	B	B		
CONT PANEL		C	C	C	C		
BACKPLANE		C	C	C	C		

The L0011 or the L0016 controllers are valid for hardware revision 00,01 (10) and 02 (20); however, these revisions will not support the MS750-CA memory option. The minimum acceptable revision for inclusion of the MS750-CA is hardware Rev 48. See MS750-CA option chart (Section 5.2) for requirements.

#### 4.2 KA750-00 REVISION SUMMARY

\* This is the initial introduction of revision control on the 11750 and represents the minimum module revision levels at FCS - October 1980.

\* Compatible Revision Levels:

- . Microcode version at FCS - CMT050.
- . VAX/VMS version at FCS - 2.0.
- . Diagnostics

Current Diagnostic Release is V. Refer to EVNDX and the VAX Diagnostic Evaluations for each Diagnostic release (see Speed Bulletin) for compatibility problems.



```
* Quick Check - >>>E/I 3E
                  I      0000003E  02003200
                  >>>
```

```
* SID Register Switch Pack Setting - Set bits 0 to 7 to the
                                      ON position (See below)
```

	7	6	5	4	3	2	1	0	<--	REG BIT POSITION
O	+-----+									ETCHED ON BOARD
F	1	2	3	4	5	6	7	8	<--	SWITCH NUMBER STAMPED
F	-----									ON SWITCH PACK
	+-----+									

### 4.3 KA750-01/KA750-10 REVISION SUMMARY

- \* This represents VAX750-M-0001 FCD - began shipping from manufacturing 12/1/80. Field implementation began 3/81.
- \* This FCD combines a microcode change and a hardware change to the L0003, L0004, and L0005 Modules. The CPU backplane is also modified. The following ECOs are incorporated...

```
L0003-TW001
L0004-TW003
L0005-TW002
70-16486-TW001
```

This FCD was implemented to inhibit the possible interruption of an instruction that references the Unibus address space performing a DATIP. If the instruction is faulted because of a TB miss and external interrupts are pending, the Unibus is hung until the DATOB is done after the microcode completes the translation. The following instruction would cause this possible conflict:

```
ADDW3 #12, physical translation to @*XFFFF20
```

Part of this FCD also connects some signals from the CPU to the FPA (slot 1 to slot 2) for FPA interfacing and also the drawing set of the FPA is modified for signal name continuity.

\* Compatible Revision Levels:

- Microcode - CMT052

- VAX/VMS Version 2.2 - backwards compatible to 2.0

**Diagnostics**

Current EVNDX release is W. The following diagnostic compatibility problems have been identified:

- ECKAX - Test fails MACHINE CHECK TEST 7 with optional WCS module installed. Diagnostic Bug.
- EVKAB - Intermittant failures on the CVIPL and other instructions. Absence of post-process CLKX bit?
- ECKAC - 4.0 or higher will fail test 7C if this FCD is missing.
- ECKAM - 1.2 fails with 8 array boards present.

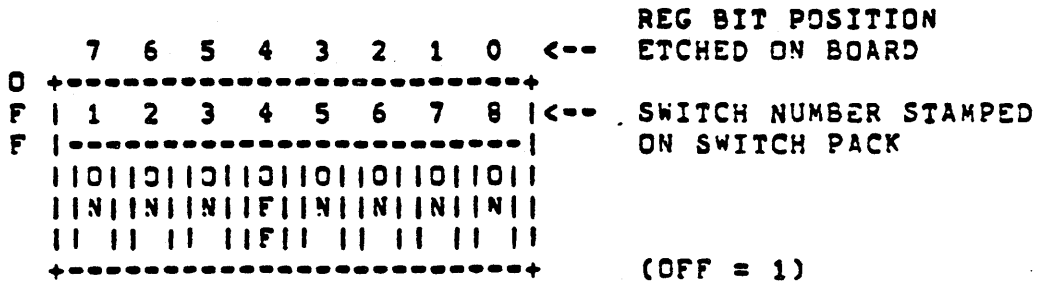
Refer to EVNDX and the VAX Diagnostic Evaluations for each Diagnostic release (see Speed Bulletins) for incompatibilities.

**\* Quick Check -**

- A. >>>E/I 3E
- |     |          |          |                     |
|-----|----------|----------|---------------------|
| I   | 0000003E | 02003401 | (w/out SID)         |
| >>> |          |          |                     |
| >>> | I        | 0000003E | 02003410 (with SID) |

B. Inspect L0003 module and look for a 7427 in chip position E1. Also inspect the backplane assembly and look for a wire from 100B10 to 200B10 (MEM STALL H)

- \* SID Register Switch Pack Setting - Set bits 0 to 3 and 5 to 7 to the ON position; set bit 4 to OFF position**



KA750-02/KA750-20 REVISION SUMMARY  
-----

- \* This represents VAX750-4-0002 FCO. Mfg. began shipping 3/30/81. Field implementation began 6/81.
- \* This FCO corrects a large collection of problems in the microcode and hardware. The following ECDs are incorporated. There is the possibility of Unexpected System Service Exceptions caused by the CON gate array that is fixed with the ECD to L0004 module.

L0005-TW003  
L0004-TW004

Some systems in the field may have L0004 ECD without the L0005 ECD when the FCO is implemented.

## \* Compatible Revision Levels:

- Microcode - CMT062 (replaces CMT052). Refer to REV750.MIC file for a list of all fixes to the microcode.
- VAX/VMS Version 2.2 - backwards compatible to 2.0
- Diagnostics

Current EVNDX Release is Y. The following problems exist in the upgrade to CMT062 as far as diagnostics are concerned:

## • ECKAL 2.0 Fails at PC 5000FFF1 ?

Microcode change makes it impossible to force a TB Miss in both groups of the TB. Diagnostic attempted to read and write the TBDR which now causes a reserved operand fault.

## ECKAX 1.2 to 3.2

Fails at TEST 1. The diagnostic expects a reserved operand fault when accessing IPR ^X3F and it does not occur. This is because of the addition of the IPR ^X3E which is called TBCHK. It allows the programmer to probe the TB at a VA and then branch on the state of the PSL VBIT indicating a TB hit. Diagnostic bug.

## ECKAX 1.2 to 3.2

Fails at TEST 7 with optional KU750 module. Diagnostic forces a machine check in a WCS location with bad parity. Diagnostic bug.

## ECKAM 1.2

Diagnostic does not work properly when there are 8 array boards installed. Does not report correctable errors correctly.

# 122

Refer to EVNDX and the VAX Diagnostic Evaluations for each Diagnostic release (see Speed Bulletins) for incompatibilities.

- \* Quick Check - >>>E/I 3E  
I 0000003E 02003E02 (W/out switch)  
>>>  
>>> I 0000003E 02003E20 (With Switch)

- \* SID Switch Setting -

Set SID register switch bit 0 to 4, 6 and 7 to ON position  
Set SID register switch bit 5 to OFF position. (See below)

	7	6	5	4	3	2	1	0	<--	REG BIT POSITION ETCHED ON BOARD
O	+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
F	1	2	3	4	5	6	7	8	<--	SWITCH NUMBER STAMPED ON SWITCH PACK
F	-----									
	0  0  0  0  0  0  0  0  0									
	N  N  F  N  N  N  N  N  N									
	F									
	+-----+-----+-----+-----+-----+-----+-----+-----+-----+									

## 4.5 KA750-03/KA750-30 REVISION SUMMARY

-----

- \* This represents VAX750-R-0003 FCD. Mfg. began shipping 2/22/82; field implementation began January 1982.
- \* This FCD is based on ECO L0005-TW005  
L0004-TW004

and corrects interface problems with the floating point accelerator FP750 which began shipping Jan-1982.

Also, the layered software product DBMS must have this FCD in order to operate correctly. This product will report to operator if the system does not have this ECO installed.

- \* Compatible Revision levels:

- Microcode - CMT094 replaces CMT052 or CMT062. Refer to REV750.MIC file for a list of all fixes to the microcode.
- VAX/VMS - Current version is 2.4 and is backwards compatible to 2.0.

# 123

## . Diagnostics

Current EVNDX release is 3.0. Refer to EVNDX and the VAX Diagnostic Evaluations for each Diagnostic release (see Speed Bulletins) for incompatibilities.

### \* Quick Check -

Examine the SID register in console mode.

```
>>>E/I 3E
      I 0000003E 02005E03
```

```
>>>      I 0000003E 02005E30
```

Inspect the L0005 module for an IC in location E27 with the part number 932F1.

### \* SID Switch Setting

Set switch bit positions 0 to 3, 6 and 7 to the ON position  
Set switch bit positions 4 and 5 to the OFF position

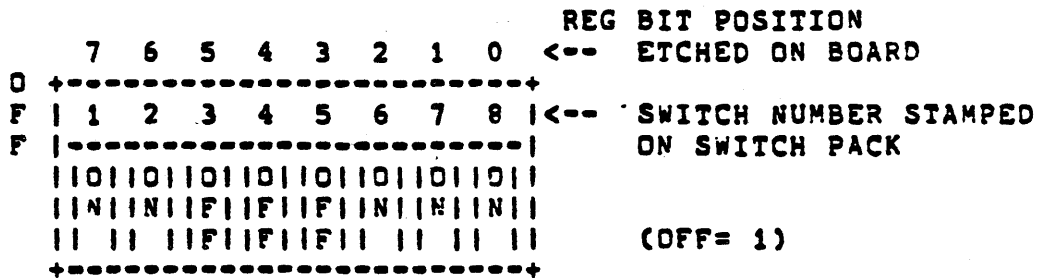
	7	6	5	4	3	2	1	0	REG BIT POSITION
O	+-----+<--								ETCHED ON BOARD
F	1	2	3	4	5	6	7	8	<-- SWITCH NUMBER STAMPED
F	-----								ON SWITCH PACK
									(OFF= 1)
	+-----+<--								

4.6 KA750-38 REVISION SUMMARY

- \* This represents VAX ECO 7016486-TW002, which brings the backplane rev to C. Manufacturing began shipping 6/1/82. This ECO does not increase the hardware revision on the kernal.
- \* ECO was done to expand the backplane addressing capabilities to support the optional 1 megabyte memory array system.
- \* SYS ID Switch has been installed by manufacturing on all Rev C backplanes. This switch identifies both the kernal rev and the backplane addressing capabilities.

Rev C backplane - last two hex digits will be reversed when the SID switch is installed. Last digit is used to indicate the backplane rev. (i.e. 8 = C backplane; 0 = B backplane)

Set switch bit positions 0 to 2, 6 and 7 to the ON position.  
Set switch bit positions 3 to 5 to the OFF position.



5.0 OPTION REVISION CONTROL

The option interfaces and adaptors will have a separate revision history from the CPU. The KA750 CPU Kernal subsystem will include the following integral subsystems.

CPU	KA750	-	70-16486	CPU Backplane Asembly
			L0002	DPM Module
			L0003	MIC Module
			L0004	UBI Module
			L0005	CCS Module
			TU58-XA	Tape Drive Unit
			54-13489	TU58 Tape Controller Unit
			M9313	UET Unibus Terminator/ Exerciser
	MS750-AA		L0011	CMC Module
			M8728	256KB array board
	MS750-CA		L0016	Controller
			M8750	1MB array board
			M8728	256KB array board

# 1.25

OPTION	RH750	L0007	MBA Module
KU750-YG		5413865-C	Add on daughter board
FP750		L0001	FPA Module
DW750		L0010	2nd Unibus Module
DR750		L0014	Interprocessor I/F Module

The internal options of the 11750, with the exception of the MS750-AA and MS750-CA, will be tracked at the unit revision level only. This means that a functional change to the RH750, DW750, FP750 and KU750 will not increment the kernel revision level.

Each option revision summary will indicate any hardware, operating system, diagnostic and microcode constraints. The option will be considered compatible with the kernel hardware, VMS and microcode revisions used during the development of the option. Earlier compatible revisions will be noted only if they have been tested and proven to work.

Pertinent diagnostics to be run for each option (and the required revision, if any) will also be noted.

## MS750-AA OPTION REVISION DESCRIPTION

MS750-AA Revision		00	01	02	03	04	05
MODULE	SLOT						
L0011	10	D	D				
L0016	10	A	N/A				
M8728	11-19	C	C				
BACKPLANE	70-16486	B	C				

## MS750AA-00 REVISION SUMMARY

- \* Creation date is October 1980.
- \* This is the initial introduction of revision control on the MS750-AA and represents the minimum module revision levels at FCS.
- \* Note that only L0011 memory controllers shipped at FCS. Note that the L0016 controller, which will support both the M8728 and the new M82750 memory arrays and will be available in Q1FY83, can also be used in a Rev 00 machine.

- \* Shortly after FCS, VAX750-M-0001 was done which increased the Rev of the backplane from A to B. Only 27 machines were shipped with "A" backplanes.

## MS750AA-01 REVISION SUMMARY

-----

- \* Creation date July 82.
- \* New revision of the backplane is introduced to increase the addressing capabilities.
- \* The L0016 is not valid for Revision 01 - that combination (L0016 and Rev C backplane) is a new option designation - MS750-CA (See below).
- \* The M8750 memory array will not function in an MS750-AA option configuration.
- \* Diagnostics - ECKAC and ECKAM. Run ECKAC first; run ECKAM in QUICK VERIFY mode.

5.2

## MS750-CA OPTION REVISION DESCRIPTION

-----

MS750-CA Revision	00	01	02	03	04	05
MODULE						
L0016	10	A				
M8728	11-18	C				
M8750	11-18	A				
BACKPLANE	70-16486	C				

## MS750CA-00 REVISION SUMMARY

-----

- \* Creation date - projected August 1982.
- \* This is the initial introduction of revision control for the MS750-CA and represents the minimum revision levels required for FCS of this option - July 1982.
- \* Note that the L0011 cannot be used in this option.
- \* Any mixture of M8728 and M8750 arrays will function; however, M8750 arrays must occupy the slots adjacent to the L0016 controller, starting with slot 11.



- \* This option requires VMS V 3.0 or higher.
- \* Minimum 11750 kernal rev to support this option is 48.
- \* Diagnostics - ECKAC (min. rev. 6.2) - EVNDX 7.0 (July 1982)  
                   ECKAM (min. rev. 2.4)                   "

Run ECKAC first; run ECKAM in QUICK VERIFY mode.

### 5.3 RH750      OPTION REVISION DESCRIPTION

```

-----
RH750 Revision      | 00 | 01 | 02 | 03 | 04 | 05 |
-----
MODULE  |  SLOT  |      |      |      |      |      |
-----
L0007   | 7,8 or 9 | A   | A1,B |      |      |      |
-----
    
```

#### RH750-00 REVISION SUMMARY

- \* Creation date is FCS - April 1981.
- \* This represents the initial introduction of revision control for the RH750 and represents the minimum revision level for the L0007 module at FCS.
- \* Diagnostics - ECCAA and EVRAA.

#### RH750-01 REVISION SUMMARY

- \* Creation date is October, 1981.
- \* Represents RH750-R-0001 FCD, which consisted of ECO L0007-TW002 to fix the problem of data lates on multiple MASSBUS systems. Replace 23-909A9 at location E12 with 23-969A9.
- \* FCD done on "C" etch modules only. Etch Revision "D", Module Rev "B" is a relayout of the L0007 module and is equivalent to Etch Rev "C", Module Rev "A1".
- \* Diagnostics - ECCAA and EVRAA.

## 5.4 FP750 OPTION REVISION DESCRIPTION

FP750 Revision	00	01	02	03	04	05
MODULE						
L0001	1	B	C			

## FP750-00 REVISION SUMMARY

- \* Creation date is FCS - December 1981. This represents the minimum module revision level required at FCS.
- \* 11750 kernal rev must be at Rev 3 (Rev 94 microcode required).
- \* Diagnostics - ECKAB (min. rev. 7.2) - EVNDX 4.0 (Jan 1982)  
                   ESCAA (min. rev. 6.4) -           "  
                   EVKAB (min. rev. 2.5) -           "  
                   EVKAC (min. rev. 4.0) -           "

## FP750-01 REVISION SUMMARY

- \* Creation date is March 1981.
- \* Represents FP750-R-0001 FCD consisting of ECD L0001-TW002, which fixes the problem of the FPA not powering up "enabled" due to incomplete initialization of circuitry.
- \* Diagnostics - ECKAB (min. rev. 7.2) - EVNDX 4.0 (Jan 1982)  
                   ESCAA (min. rev. 6.4) -           "  
                   EVKAB (min. rev. 2.5) -           "  
                   EVKAC (min. rev. 4.0) -           "

## 5.5 KU750 OPTION REVISION DESCRIPTION

KU750 Revision	00	01	02	03	04	05
MODULE						
5413865-C attaches to L0005		C	1			

## KU750-00 REVISION SUMMARY

-----

- \* Creation date is FCS - March 82.
- \* This represents the minimum module revision level required at FCS.
- \* Requires KU780-YG microcode rev at 2.0 or higher.
- \* Diagnostic - ECKAX - 11750 Cluster Exerciser - EVNDX 7.0 (July 82) (minimum rev 3.4)

### 5.6 DW750 OPTION REVISION DESCRIPTION

-----

DW750 Revision		00	01	02	03	04	05
MODULE	SLOT						
L0010		?					

## DW750-00 REVISION SUMMARY

-----

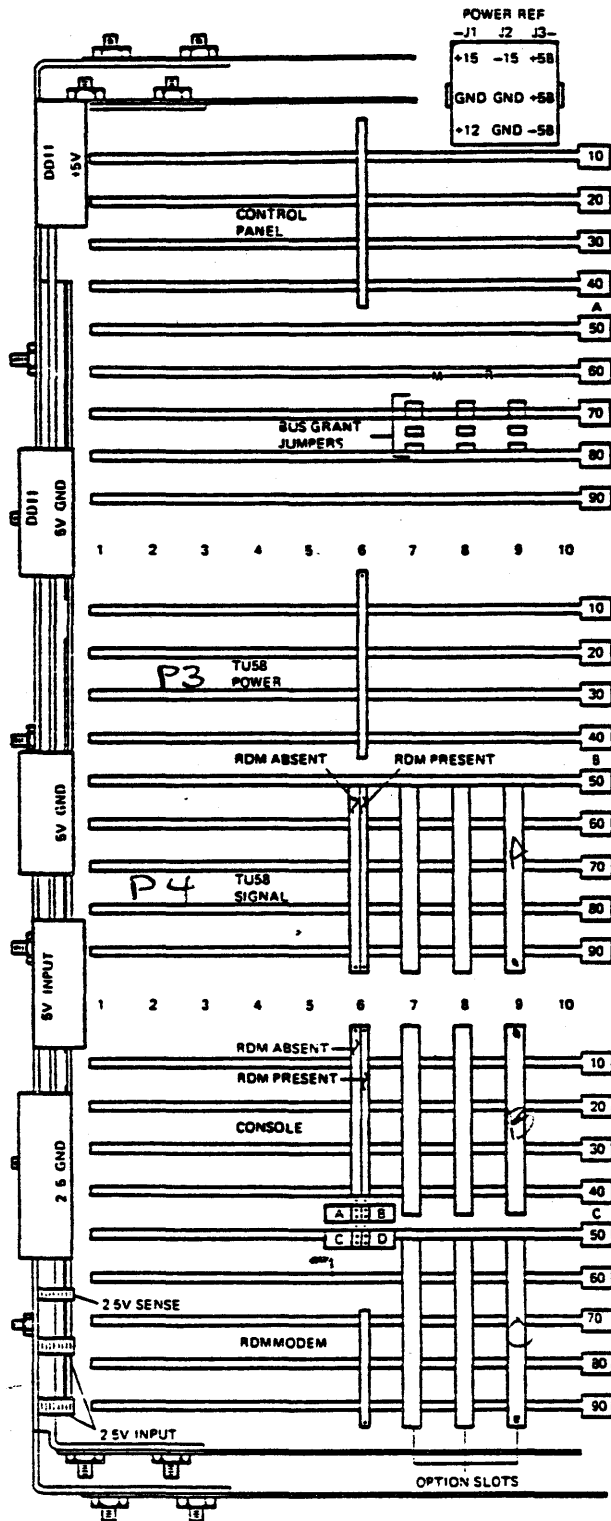
- \* FCS scheduled for September 1982.
- \* 11750 kernal rev must be 30/38 or higher.
- \* Requires Version 3.0 or higher.
- \* Diagnostics - ECSAA - min. rev 6.4 - EVNDX Release 4.0 (Jan 82)  
ECCBA - min. rev 1.3-

### 5.7 DR750 OPTION REVISION DESCRIPTION

-----

DR750 Revision		00	01	02	03	04	05
MODULE	SLOT						
L0014		?					

- \* FCS scheduled for September 1982.
- \* 11750 kernal rev must be 40/48 or higher.
- \* Requires VMS Version 3.0.
- \* Diagnostics - EVDFD - rev 1.0 - EVNDX Release 7.0 (July 82)  
 EVDFE - Rev 1.0  
 EVDFG - rev 1.0  
 EVDFH - rev 1.0  
 ECDFB - rev 1.0  
 ECDFC - rev 1.0  
 ECSAA - min rev 6.7



OPTION SLOT BUS GRANTS	
TO SELECT	REMOVE JUMPER
BG 4	A00X 67
BG 5	A00X 68
BG 6	A00X 73
BG 7	A00X 77

X=SLOT 7.8.9

TEST POINTS		
RDM 19	MATCH PULSE	C00681
RDM 23	SA CLOCK	C00673
RDM 23	SA ST/SP	C00675
DPM 17	M CLOCK	R00205
DPM 17	BASE CLOCK	A00273
DPM 17	B CLOCK	R00208
MIC 04	MEM STALL	R00210
DPM 17	PHASE 1	A00590

HARDWARE REV LEVEL (SYS ID)	
BIT	PIN #
0	R00456
1	R004
2	R00454
3	R00453
4	R00450
5	R00449
6	R00448
7	R00446

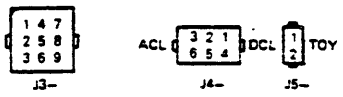
1 = JUMPER IN

CONSOLE BAUD RATE				
RATE	CON SR			
	A	B	C	D
300	0	0	1	0
600	0	1	1	0
1200	1	1	1	0
2400	0	0	0	1
3600	1	0	0	1
4800	1	1	0	1
9600	1	0	1	1
19200	0	1	1	1
38400	1	1	1	1

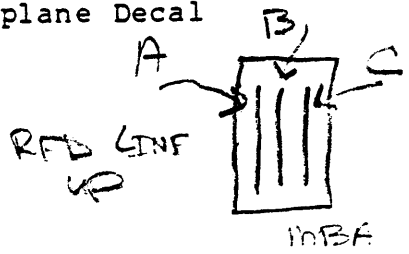
  

PIN #	C00645	C00646	C00649	C00650
JUMPER TO GND	C00643	C00644	C00651	C00652

BAUD RATE JUMPERS FOR CONSOLE



VAX-11/750 Backplane Decal







## OPTION SLOT JUMPERS





### SECTION A OF DESIRED SLOT





BUS GRANT JUMPERS SLOTS 7,8,9 OF THE EXTENDED HEX SECTION

WHEN AN OPTION IS NOT INSTALLED IN A SLOT, ALL BG JUMPERS MUST BE INSTALLED!

WHEN INSTALLING AN OPTION, ALL BG JUMPERS MUST BE REMOVED FROM THAT SLOT!

SLOT 7  
-----  
65 . . 66  
67  68  
69  70  
71 . . 72  
73  74  
75 . . 76  
77  78  
79 . . 80

SLOT 8  
-----  
65 . . 66  
67  68  
69  70  
71 . . 72  
73  74  
75 . . 76  
77  78  
79 . . 80

SLOT 9  
-----  
65 . . 66  
67  68  
69  70  
71 . . 72  
73  74  
75 . . 76  
77  78  
79 . . 80

132

## ERROR LOGGER

ERROR LOG DESCRIPTION AND USE  
-----

THE ERROR LOGGER CONSISTS OF BASICALLY THREE PARTS:

- 1). A SET OF EXECUTIVE ROUTINES THAT DETECT ERRORS AND EVENTS AND RECORDS RELEVANT INFORMATION INTO AN ERROR LOG BUFFER IN MEMORY.
- 2). A PROCESS CALLED ERRFMT.EXE THAT PERIODICALLY EMPTIES THE BUFFERS, TRANSFORMS THE DESCRIPTIONS OF THE ERRORS INTO A STANDARD FORMAT AND STORES THE FORMATTED INFORMATION IN A FILE ON DISK.
- 3). A PROCESS CALLED SYE.EXE THAT GENERATES READABLE REPORTS FROM THE INFORMATION FORMATTED BY ERRFMT.EXE.

THE EXECUTIVE ROUTINES AND ERRFMT.EXE RUN CONTINUOUSLY WITHOUT USER INTERVENTION TO FILL THE BUFFERS WITH RAW DATA ON EVERY DETECTED ERROR AND EVENT. WHEN A BUFFER BECOMES FULL OR A PREDETERMINED TIME HAS EXPIRED, THE BUFFER IS EMPTIED TO A FILE ON DISK. IF A SUDDEN BURST OF ERRORS OCCUR FASTER THAN THEY CAN BE FORMATTED AND STORED, THEY WILL BE ASSIGNED A SEQUENCE NUMBER AND NO OTHER DATA CONCERNING THE EVENT OR ERROR WILL BE LOGGED.

THE FILE WHICH CONTAINS THE ERROR INFORMATION IS CONTAINED IN THE [SYSERR] DIRECTORY AND IS CALLED ERRLOG.SYS. WHEN RUNNING SYE.EXE THE FILE SHOULD BE RENAMED TO PREVENT VERSION NUMBERS FROM ACCUMULATING. ANY NEW ERRORS ENCOUNTERED BY ERRFMT.EXE WILL CAUSE A NEW ERRLOG.SYS TO BE CREATED.

TO RUN SYE.EXE  
-----

```
s SET DEFAULT SYSSDISK:(SYSERP)

s RENAME ERRLOG.SYS  ERRLOG.OLD/NEW_VERSION

s DIR

s RUN SYSSSYSTEM:SYE   or   s #C SYE
```

THE PROGRAM WILL ASK SEVERAL QUESTIONS:

```
INPUT FILE?      (SPECIFY THE EXACT FILE TO BE COMPILED)
                  Example:  ERRLOG.OLD;23
                  Default:  ERRLOG.OLD
```

```
OUTPUT FILE?    (THIS WILL BE THE END RESULT OF SYE)
                  Example:  MYFILE.REW
                  Default:  SYSSOUTPUT
                  LP WILL SEND OUTPUT TO A PRINTER
```

# 134

## OPTIONS?

(SEVERAL OPTIONS ARE AVAILABLE:)

Default: ROLL-UP  
Options: R ROLL-UP  
          B BRIEF  
          C CRYPTIC  
          S STANDARD

### ROLL-UP

A QUICK SUMMARY OF ERRORS FOR EACH FAILING DEVICE WITH NO DETAILS ABOUT THE INDIVIDUAL ERRORS. THE TOTAL WILL EQUAL THE SUM OF HARDWARE AND SOFTWARE ERRORS.

### BRIEF

CONTAINS A BRIEF DESCRIPTION ABOUT EACH ERROR  
INCLUDING: A). TYPE OF ERROR  
          B). DEVICE OR COMPONENT WHICH CAUSED IT  
          C). A SEQUENCE NUMBER  
          D). A TIME WHEN THE ERROR WAS LOGGED

### CRYPTIC

DEVICE AND CPU ERRORS ONLY. THE OUTPUT WILL CONTAIN THE CONTENTS OF ASSOCIATED REGISTERS WITH EVERY ERROR BUT NO EXPLANATION.

### STANDARD

EVERY ERROR HAS AN ENTRY AND A COMPLETE BREAKDOWN OF REGISTERS AND A DESCRIPTION OF WHAT THE REGISTERS ARE.

### DEVICE NAME?

(INDIVIDUAL DEVICE OR <CR> FOR ALL)

CP CPU AND CMI  
CO CONFIGURATION CHANGES  
ME MEMORY AND ALERT  
DEVICES: SY SYSTEM INFORMATION AND BUGCHECKS  
          DMAX RK's  
          DBAX RP's etc.  
          D ALL DISK  
          M ALL TAPES  
          MT  
          MF  
          UNKNOWN UNKNOWN DEVICE ERRORS

YOU CAN ALSO USE A "-" TO DELETE CERTAIN DEVICES

Example: -D EVERYTHING BUT DISKS  
          -/CONFIG EVERYTHING BUT MOUNTS AND DISMOUNTS

AFTER DATE? (DESIRED FIRST DATE OF ENTRY)

BEFORE DATE? (DESIRED LAST DATE OF ENTRY)

          XX-YYY-19ZZ XX:XX:XX.XX  
          DAY MONTH YEAR DELTA TIME IF DESIRED  
Example: 11-SEP-1981 03:22:00.00  
          HRS MIN SEC 100ths

IF YOU DID NOT SPECIFY AN OUTPUT FILE OR DEVICE, THE SYE PROGRAM WILL INSTRUCT YOU TO ALIGN THE PAPER AND STRIKE RETURN.

IF YOU SPECIFIED AN OUTPUT FILE OR DEVICE, YOU SHOULD RECEIVE A SUCCESSFUL COMPLETION MESSAGE, AT THIS TIME YOU COULD PRINT OR TYPE THE OUTPUT FILE.



135

**SDA**

System Dump Analyzer (SDA) Procurement  
-----

This text is intended to demonstrate how to procure a SDA report after a system crash. It is not intended to demonstrate the interpretation of the SDA.

Now normally the crash dump file is contained within the [SYSEXE] directory located on the system disk.

Log in to SYSTEM MANAGER account.

Upon the advent of \$ prompt, obtain a list of files contained within the directory [SYSEXE] The file that must be there is:

SYSDUMP.DMP

```
*****
*
*           DO NOT RENAME THIS FILE           *
*
*
*****
```

Once you have ascertained that the file is present, then type:

\$ MCF SDA

The standard response to that should be:

Enter name of the dump file>

The response to that statement is:

[SYSEXE]SYSDUMP.DMP

The response to typing [SYSEXE]SYSDUMP.DMP <CR>, is a brief description of the dump and then a SDA prompt:

SDA>

After the SDA> prompt, type in the following:

```
SDA> SET OUTPUT SDADUMP.xxx (let xxx be your initials)
SDA> SHOW SUMMARY
SDA> SHOW CRASH
SDA> SHOW STACK
SDA> SHOW PROCESS
SDA> EXAMINE/PO
SDA> EXIT
```

The EXIT should have returned the you back to DCL. Obtain a directory. This directory should contain a file SDADUMP.xxx (xxx should be your initials for file type) Now all you have to do is obtain a hardcopy of the dump.

\$ PRINT SDADUMP.xxx

You also can look at this file at your terminal:

\$ TYPE SDADUMP.xxx

## Bootstrap Process

### BOOTSTRAP PROCESS

The following lists the steps required to obtain a running system on a VAX-11/750 processor:

1. Power up occurs.
2. The VAX-11/750 microcode detects power on and follows the power on strategy selected by the POWER-ON-ACTION switch located on the processor control panel.
  - a. If a restart cannot be done, either an automatic bootstrap from the default bootstrap device or a halt will be done.
  - b. If the machine halts, the microcode program gains control. This program:
    - (1) Issues the console prompt (>>>) at the console terminal
    - (2) Accepts interactive commands to bootstrap the system by means of the default bootstrap device or a user-specified bootstrap device
3. The microcode program looks up and executes the bootstrap device read-only memory (ROM). This ROM is 256 bytes and contains a main routine (at the entry) and a subroutine. The main routine reads block 0 from the bootstrap device and jumps to the boot block entry. The main routine and the boot block routine use the ROM subroutine to read arbitrary blocks from the bootstrap device into memory.
4. The boot block contains the logical block address, size, and entry offset of the program to be executed in the bootstrap process. This program can be either (1) stand-alone BOOT58, when the bootstrap device is the TU58 console drive, or (2) VMB.EXE, when the bootstrap device is the system disk.
  - a. If the bootstrap operation is performed from the console TU58 tape cassette using stand-alone BOOT58, the user types BOOT58 commands to set up register input values and to load and start VMB.EXE.
  - b. If the bootstrap operation is performed directly from the system disk using VMB.EXE, the microcode program derives the register input values.
5. VMB.EXE is the primary bootstrap program, which contains CPU-independent code and CPU-dependent routines. It also contains a set of primitive non-interrupt-driven drivers for

all possible system devices and a primitive file system for locating and reading Files-11 Structure Level 1 and Structure Level 2 files.

VMB.EXE performs the following steps:

- a. Saves the register values and some values calculated from the register values in the restart parameter block (RPB).
  - b. Reads the system identification register to determine the processor type and to select the table of appropriate processor-dependent data and subroutines.
  - c. Determines the amount and pattern of memory. A page frame number (PFN) bitmap is constructed. Unless inhibited by a boot flag, memory is tested for gross, uncorrectable parity errors. VMB.EXE constructs, in the RPB, a table indexed by nexus number of all memory controller and I/O adapter types.
  - d. Based on register values, one of the following occurs:
    - (1) A boot block at the designated logical block number (LBN) will be read into memory and given control.
    - (2) A file named [SYSEXE]SYSBOOT.EXE will be read into memory and given control.
    - (3) A file named [SYSMAINT]DIAGBOOT.EXE will be read into memory and given control.
    - (4) A file specified by the user in response to a prompt will be read into memory and given control.
6. SYSBOOT is the standard secondary bootstrap program. It performs initialization suitable for the unmapped environment. SYSBOOT performs the following steps:
- a. Reads current parameter settings from SYS.EXE.
  - b. Looks up the bootstrap device driver file and stores information about it.
  - c. If register values so indicate, prompts the user to modify current system parameter settings. The user can change the start-up command procedure name and modify system parameters using SET or a previously created parameter file. New parameters become the "current" parameters on the next bootstrap operation.
  - d. Sets up SPT, SYSPHD, SCB, and PFN data structures.
  - e. Reads the resident executive into high physical memory.
  - f. Locates and transfers to INIT code.
7. The system initialization process consists of four stages: INIT, SYSINIT, STARTUP.COM, and SYSTARTUP.COM.
- a. INIT is part of SYS.EXE. It performs the following:
    - (1) Enables mapping and sets the PC to system space.
    - (2) Prints the system announcement message

- (3) If requested by means of the boot flag, stops at the XDELTA breakpoint.
- (4) Initializes the system for paging.
- (5) Deallocates available physical pages (PFN bitmap set up by VMB) to the free page list.
- (6) Initializes the system page table for paged and nonpaged pools.
- (7) Initializes I/O adapters using the list of present adapters generated by VMB.EXE. Initialization consists of mapping adapter register space (only the number of pages actually used are mapped) and calling adapter-specific routines to allocate and set up data structures and to initialize the adapter hardware. In addition, for UNIBUS adapters, the 8K byte I/O page of the UNIBUS is mapped.

Data structures allocated are:

MASSBUS -- adapter control block  
channel request block  
interrupt descriptor block

UNIBUS -- adapter control block

- (8) Performs additional process initialization tasks.
- (9) Transfers the primitive VMB.EXE system device driver into nonpaged pool; and saves the driver entry and boot device control/status register (CSR) as virtual addresses (rather than physical addresses) in the RPB.
- (10) Loads the CPU-dependent code image into nonpaged pool and links it into the system.
- (11) Loads the terminal handler into non-paged pool, and connects the interrupt vectors. Loads the driver image for the system device into nonpaged pool, connects its interrupt vector, and derives the name of the system disk. The rule for the system disk device name is as follows:

device name      Examine the primitive driver, where the device name is stored.

controller      The controller designator is "A," "B," or "C" for the first, second, or third occurrence of this kind of adapter. For example, if the adapter of the system device is the second MASSBUS, the controller is B. (Note that for a generally configured system, it is possible to use the AUTOCONFIGURE command procedure to derive the controller name incompatibly with INIT. Consequently, some care is required when configuring multiple controllers of possible system disks across multiple buses.)

unit              Passed from VMB.EXE input, register R3.

## BOOTSTRAP PROCESS 141

- (12) Adds the prologues of the resident drivers (for example, MB, NL) to the prologue list.
- (13) Performs initialization of resident drivers.
- (14) Moves completion code of INIT into the pool and executes it. The completion code deallocates space occupied by INIT (and optionally XDELTA) to the free page list. The completion code then jumps to the scheduler, which ultimately results in SYSINIT being swapped in and started.

b. SYSINIT performs the following:

- (1) If necessary or requested, prompts for the time of day.
- (2) Writes back system parameters to SYS.EXE.
- (3) Creates some logical names.
- (4) Sets up swapping and paging files.
- (5) Installs the VAX-11 RMS image and system message file as pageable system sections.
- (6) Mounts the system disk (ACP process created).
- (7) Creates the job controller, OPCOM, and ERRFMT.
- (8) Creates the STARTUP process.

c. STARTUP reads input from the start-up command procedure, which causes it to:

- (1) Create logical names.
- (2) Run SYS\$SYSTEM:SYSGEN to configure the I/O system.
- (3) Install known images.
- (4) Invoke [SYSMGR]SYSTARTUP.COM.
- (5) Log out.

d. SYSTARTUP.COM is an empty command procedure distributed by DIGITAL. The system manager can edit SYSTARTUP.COM to perform site-specific start-up functions.

8. SYSGEN is run by STARTUP or at any other time. SYSGEN:

- a. Provides for dynamic loading of and connecting to drivers. (The operator, null, and mailbox drivers are permanently part of the executive image.)
- b. Provides for the creation of new parameter files (which have an encoded format).
- c. Creates paging, swapping, and system dump files.

## Instruction Decode



THIS IS AN ATTEMPT TO DEMONSTRATE THE FLOW OF A MACRO INSTRUCTION THROUGH THE 11/750 DATA PATHS.

INITIAL INPUT ARGUMENTS  
-----

```
>>>D/L/P 100 005261D0          MOVL (R1),R2
                                HALT

>>>D/G 1 1000                  SET UP ADDRESS OF 1000 IN R1

>>>D/L/P 1000 12345678        SOME DATA IN 1000

>>>S 100
```

AND WERE OFF... THE START COMMAND IS DECODED BY THE CONSOLE MICROCODE IN CCS AND WILL FIRST INITIALIZE THE MACHINE. WE KNOW THAT THE CPU WILL PERFORM AN XB FLUSH WHENEVER WE WRITE TO THE PC, AND SINCE WE SPECIFIED A NEW PC INSIDE THE START COMMAND, AN EXECUTION BUFFER FLUSH TAKES PLACE. AN XB FLUSH REMEMBER DOES NOT WRITE ALL ZERO'S TO THE XB'S! THAT WOULD BE SENSELESS.

ANY TIME THAT WE WRITE THE PC, THE PRK CHIP WILL PERFORM A DOUBLE PREFETCH OPERATION BY TAKING THE VALUE SPECIFIED IN THE PC AND PERFORMING A BUS READ FROM MEMORY. SINCE THIS FIRST PREFETCH HAS ONLY FILLED XB0, ANOTHER PREFETCH WILL OCCUR USING THE PC+4 AND THE I-STREAM DATA RETURNED WILL BE PUT IN XB1. NOW THAT WE HAVE THE XB'S FULL OF DATA, THE PRK WILL START MONITORING THE PC BITS 1:0 AND THE 'XB SELECT' LINES FROM THE MDR CHIPS, AND THE BUT FIELD OF THE MICROCODE LOOKING FOR HIS TWO CONDITIONS TO BE MET.

- 1). IS THERE AN EMPTY XB? DETERMINED BY THE PC BITS <1:0> = 3
- 2). IS THERE A BUS CYCLE IN PROGRESS? MONITOR BUS FIELD

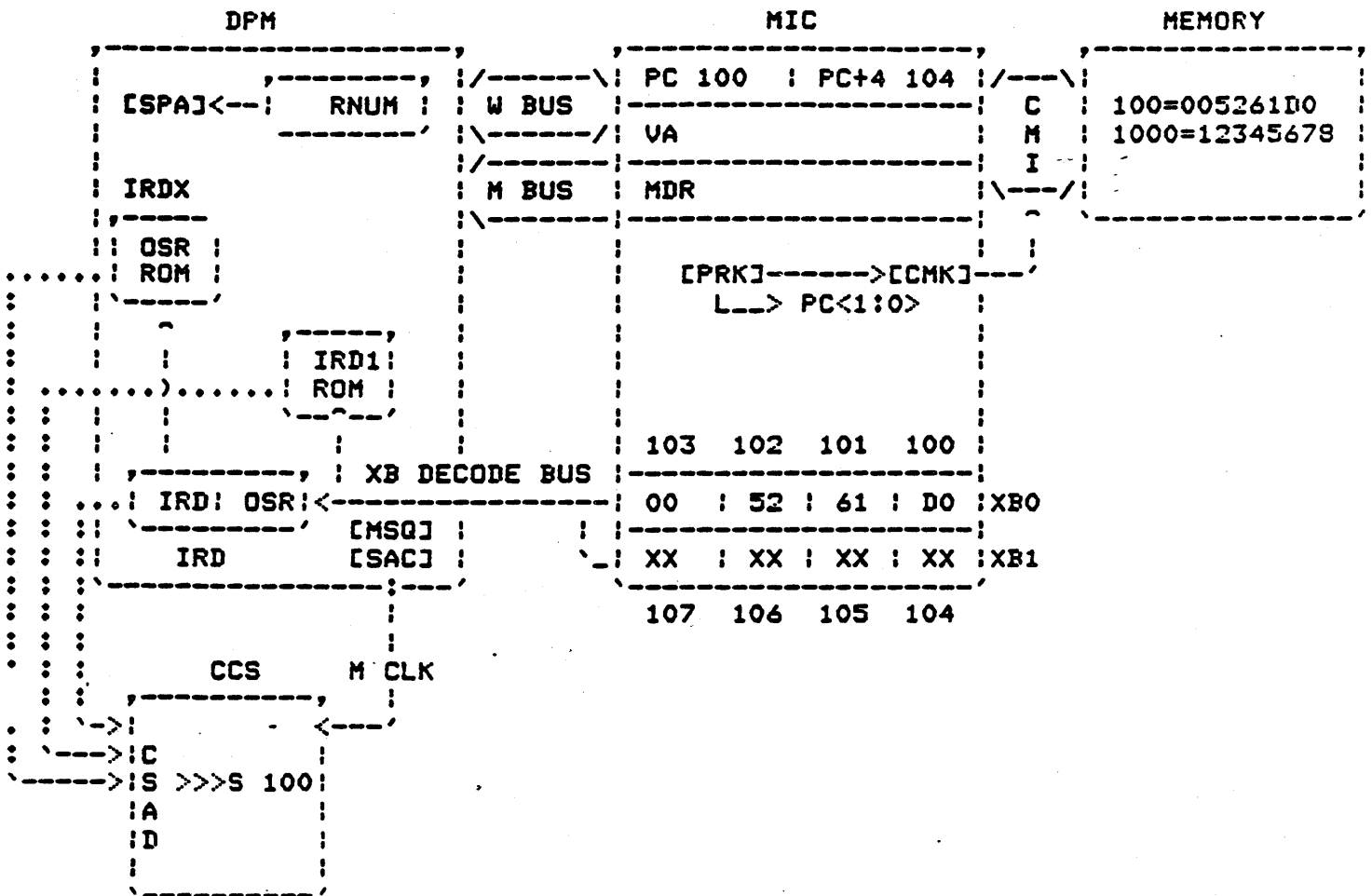
KEEP IN MIND THAT THE PRK IS WORKING TOTALLY TRANSPARENT TO THE MICROCODE AND WILL INITIATE A PREFETCH WHENEVER IT'S CONDITIONS ARE MET OR THE PC GETS REPLACED BY THE USER OR THE USERS PROGRAM.

EXAMPLE: 2\$:: BRB 2\$ THIS BRANCH INSTRUCTION WOULD REPLACE THE PC WITH THE PC PLUS THE BRANCH OFFSET.

FINALLY AFTER THE FIRST XB WAS FILLED, THE MICROCODE ROUTINE FOR THE START COMMAND WILL DO AN IRD1 AND THE WHOLE MESS BEGINS...

I THINK A BLOCK DIAGRAM WOULD BE NICE RIGHT ABOUT NOW...

## BLOCK NUMBER 1



LETS BEGIN... AFTER THE START COMMAND INITIALIZES THE MACHINE AND WRITES THE PC, THE MICROCODED BUT FIELD GETS AN IRD1.

PC	:	PC+4
100	:	104

1. IRD1 OCCURS

ON AN IRD1 WE KNOW THAT TWO BYTES OF I-STREAM DATA WILL BE SOURCED FROM ONE OF THE XB'S OVER THE DECODE BUS TO THE IRD GATE ARRAY. SOURCING THIS DATA, MOVES AN OPCODE AND THE FIRST OPERAND SPECIFIER INTO THE IRD CHIP, AND THE OPCODE IS ALSO SENT TO THE IRD1 ROM FOR DECODING. SINCE TWO BYTES WERE SOURCED, WE BUMP THE PC BY 2.

PC	:	PC+4
102	:	106

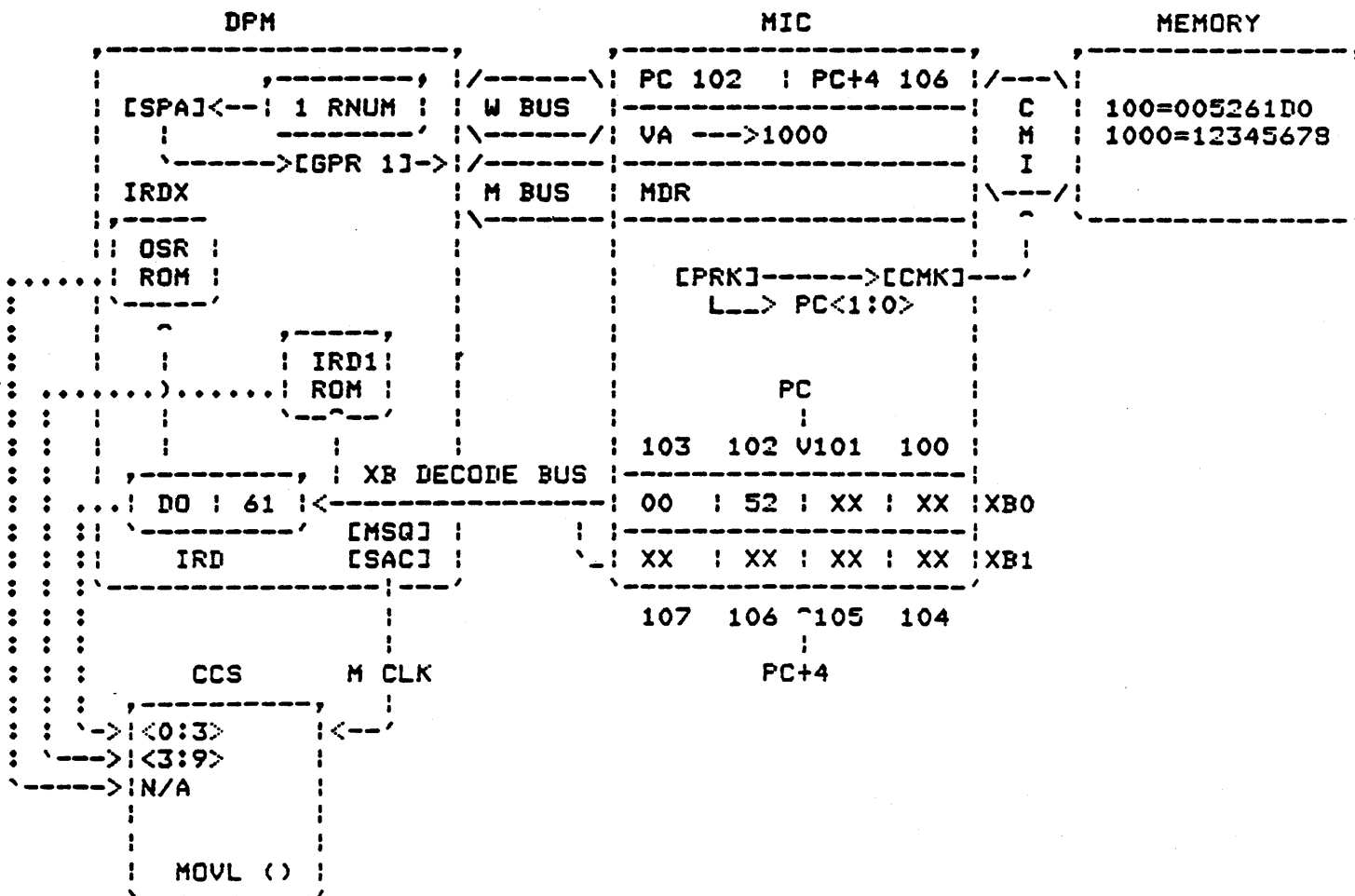
REFER TO BLOCK NUMBER 2

MICROWORD NUMBER 1

AT THIS TIME THE IRD1 ROM WILL LOOK AT THE OPCODE AND WHEN IT DECODES IT AS A MOVL INSTRUCTION, IT WILL OUTPUT BITS 3 THROUGH 9 OF THE BASE CONTROL STORE ADDRESS WHICH WHICH WILL TAKE US TO THE PROPER MICROCODE ROUTINE. ALSO AT THIS POINT THE IRD CHIP WILL EVALUATE THE 1st OPERAND SPECIFIER AND OUTPUT THE CONTROL STORE ADDRESS BITS 0 THROUGH 3 GIVING US A TOTAL CSAD FOR OUR MOVL INSTRUCTION IN REGISTER DEFERRED MODE. THE IRD CHIP WILL OUTPUT THE ENCODED VALUE FOR GPR 1 INTO THE RNUM REGISTER. (OSR DECODE)  
THE MDR WHICH CONTAINS GARBAGE WILL BE BACKED UP IN THE Q REGISTER.

\*\*\* SEE BLOCK NUMBER 2 \*\*\*

BLOCK NUMBER 2



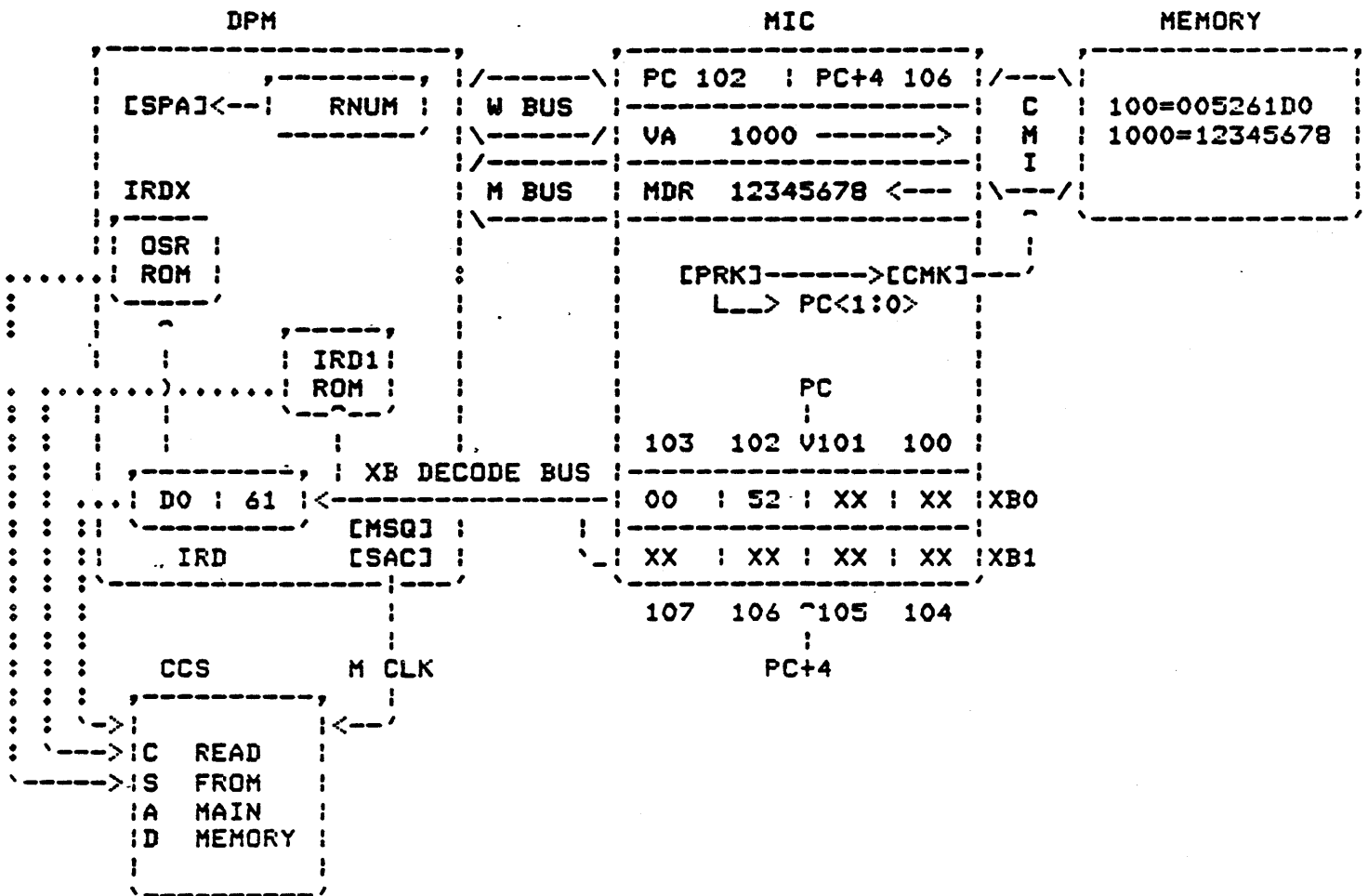
WHEN THE SPA GATE ARRAY SEE'S THE NUMBER IN RNUM, IT WILL SELECT THE CONTENTS OF R1 AND SEND IT OUT ONTO THE R BUS, THROUGH THE B LEG BYPASS OF THE ALU AND OUT ON THE W BUS. THE MICROWORD WILL SET UP THE VA REGISTER TO RECIEVE THE W BUS (WHICH IS CARRYING OUR ADDRESS OF 1000).

MICROWORD NUMBER 2

THE SECOND MICROWORD WILL CAUSE A BUS READ CYCLE TO OCCUR FROM MAIN MEMORY INTO THE MDR.

\*\*\* SEE BLOCK NUMBER 3 \*\*\*

BLOCK NUMBER 3



NOW THAT WE HAVE OUR DATA IN THE MDR, WE NEED SOMEPLACE TO PUT IT. NO MORE CAN BE DONE WITH THE 1st OPERAND, SO THE MICROCODE ROUTINE WILL DO AN IRDX TO BRING IN THE 2nd OPERAND.

AN IRDX WILL SOURCE ONE BYTE FROM THE XB INTO THE IRD CHIP AND ALSO BUMP THE PC BY 1.

PC	:	PC+4	
-----			
103	:	107	IRDX

MICROWORD NUMBER 3

WHEN THE OPERAND HITS THE IRD CHIP IT WILL BE DECODED TO FIND OUT IF REGISTER MODE IS USED AND WHICH REGISTER TO GIVE RNUM IF NEEDED. THE IRD CHIP WILL SEND THE OPCODE TO THE IRDX ROMS TO SUPPLY AN ADDRESS

THE IRDX (OSR) ROM WANTS TO KNOW TWO THINGS:

1. WHAT OP CODE IS IT? FOR A PARTIAL ADDRESS INTO THE ROM.
- 2). WHAT MODE ARE WE IN? REGISTER MODE DETERMINED BY THE UPPER 4 BITS OF THE OPERAND SPECIFIER FROM THE IRD CHIP.

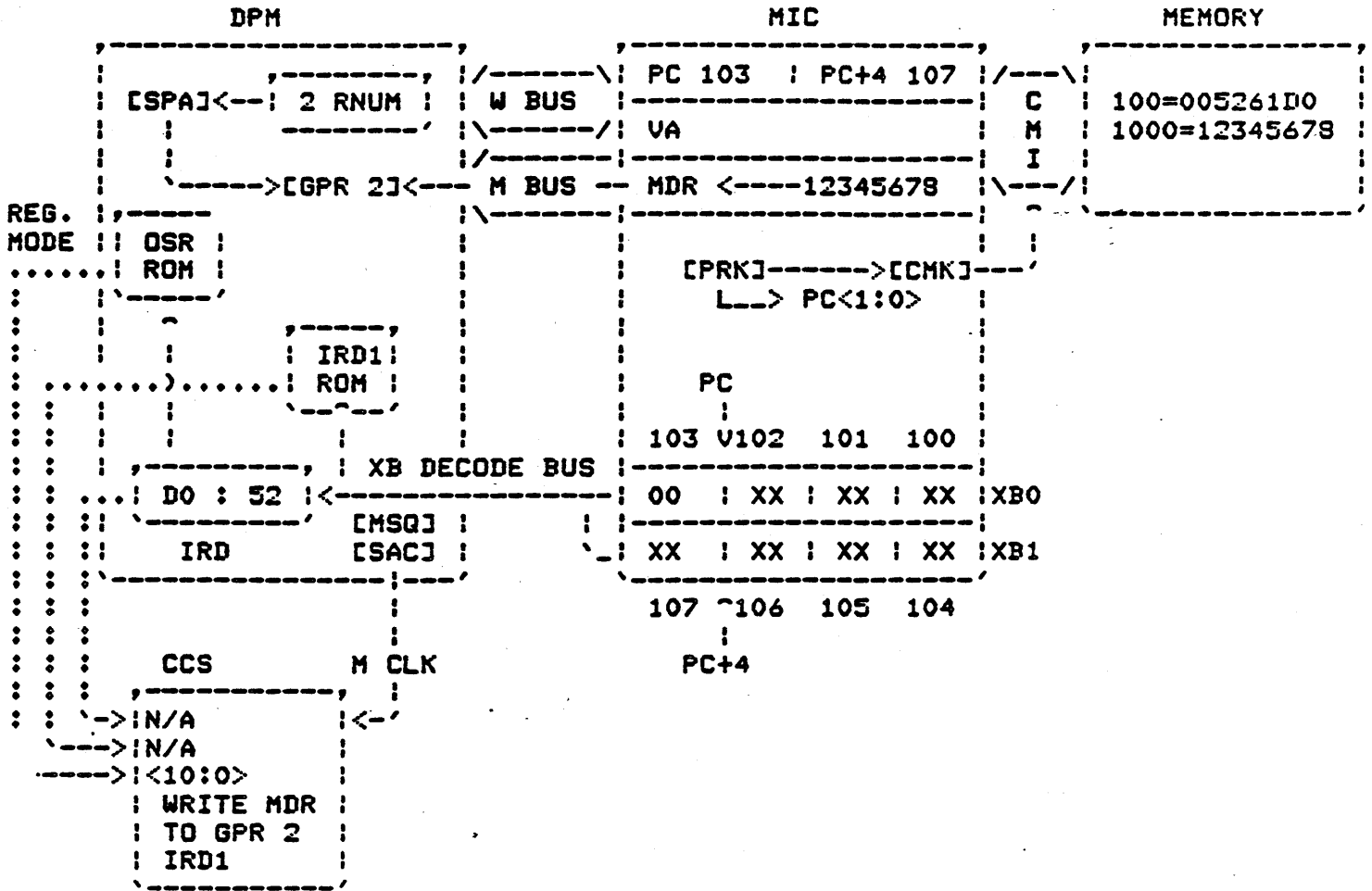
AT THIS TIME THE IRDX ROM WILL OUTPUT AN ADDRESS THAT WILL PLACE US IN THE MICROCODE TO HANDLE THE NEEDED OPERAND SPECIFIER.

THE ENCODED VALUE FOR GPR 2 IS SENT TO THE RNUM REGISTER AND LIKE BEFORE, THE SPA SELECTS THAT REGISTER BUT THIS TIME WE WILL BE WRITING INTO IT.

THE CONTENTS OF THE MDR WILL BE SENT ACROSS THE M BUS, THROUGH THE ALP CHIPS AND ONTO THE WBUS TO BE WRITTEN INTO THE SELECTED GPR AND THE MICRO ROUTINE WILL END UP WITH ANOTHER IRD1 FOR THE NEXT INSTRUCTION.

\*\*\* SEE BLOCK NUMBER 4 \*\*\*

## BLOCK NUMBER 4



HAVING JUST FINISHED THE MOVL INSTRUCTION, THE MICROCODE ROUTINE LEFT US WITH ANOTHER IRD1. AS BEFORE AN IRD1 WILL SOURCE TWO MORE BYTES OF I-STREAM DATA OVER THE XB DECODE BUS, INTO THE IRD CHIP AND ALSO UP TO THE IRD1 ROM. AS BEFORE THE PC WILL BE BUMPED BY 2.

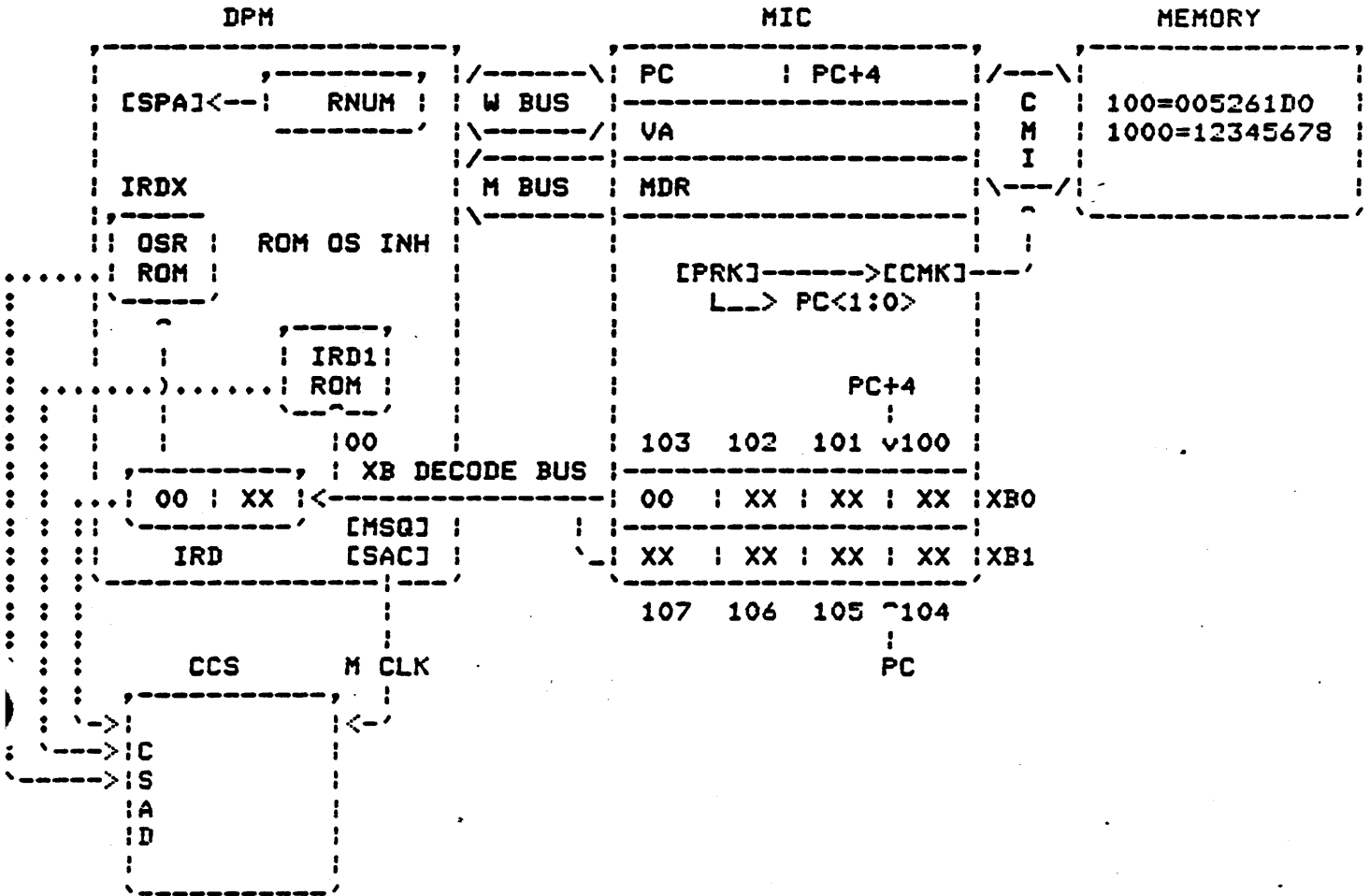
```

PC : PC+4
-----
105 : 109
    
```

\*\*\* SEE BLOCK NUMBER 5 \*\*\*

## BLOCK NUMBER 5

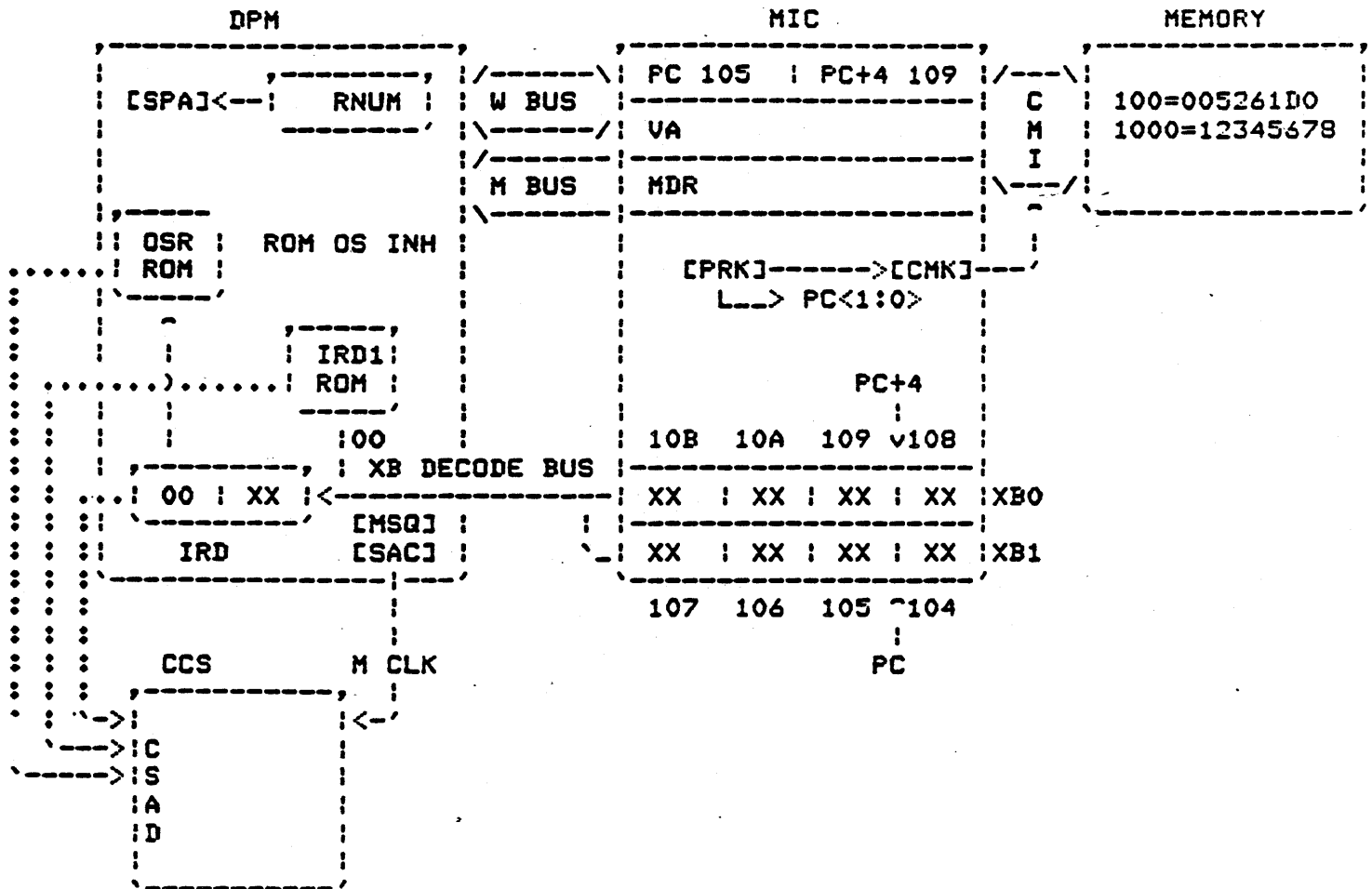
---



NOTICE WHAT HAPPENED TO THE PC AND PC+4... THE PC HAS BEEN BUMPED TO 105 WHICH TELLS THE PRK CHIP THAT WE HAVE USED ALL THE DATA IN XB0. A PREFETCH CYCLE WILL OCCUR USING THE PC+4 AS OUR ADDRESS TO FETCH DATA FROM MAIN MEMORY. IF WE SEND THE ADDRESS OF 109 OVER THE CMI WE WILL GET BACK THE LONGWORD ADDRESS CONTAINING 109. THIS IS DUE TO THE FACT THAT THE CMI IGNORES BITS 0 AND 1 OF THE ADDRESS THUS GIVING US A LONGWORD ADDRESS OF 108 WHICH IS EXACTLY WHAT WE WANT.

\*\*\* SEE BLOCK NUMBER 6 \*\*\*

## BLOCK NUMBER 6



EXECUTION OF THE NEW INSTRUCTION TAKES PLACE SIMULTANEOUSLY WITH THE PREFETCH, BUT NOTICE WHAT INSTRUCTION WE ARE USING... IT IS A HALT INSTRUCTION. WE KNOW THAT A HALT INSTRUCTION HAS NO OPERANDS ONLY AN OPCODE, THEREFORE SOMETHING MUST BE DONE TO PREVENT THE IRD CHIP FROM EVALUATING THE SECOND BYTE AS A 1st OPERAND SPECIFIER. WHAT HAPPENS IS WHEN THE IRD1 ROMS DECODE THE HALT OPCODE, (OR ANY ONE BYTE INSTRUCTION) A SIGNAL NAMED "ROM OS INHIBIT" IS OUTPUTED FROM THE ROM ITSELF AND SENT TO THE MSQ AND SAC CHIPS WHERE IT DISABLES ANOTHER SIGNAL CALLED "LOD OSR A" WHICH WILL PREVENT THE UPDATING OF THE OSR COUNTER. THE SAME SIGNAL TELLS THE SAC CHIP TO TELL THE PHB CHIP NOT TO GENERATE THE SIGNAL "IRD LOD RNUM" WHICH WILL PREVENT THE SPA CHIP FROM LOOKING AT RNUM, AND FINNALLY THE "LOD OSR A" SIGNAL TELLS THE IRD CHIP NOT TO DECODE THE DATA ON THE OSR SECTION OF XB DECODE AS IT IS NOT REALLY AN OPERAND.

THE HALT MICROCODE FLOW WILL NOW TEST THE CURRENT MODE TO SEE IF WE ARE IN KERNAL MODE AS YOU MUST BE TO HALT THE CPU.

ASSUMING THAT WE ARE IN KERNAL MODE, THE MICROCODE ROUTINE WILL ...

- 1). SET UP A HALT CODE OF 06 IN A TEMPORARY REGISTER
- 2). ADD 1 TO THE CURRENT PC GIVING US PC=106 AND PC+4=10A
- 3). VARIOUS OTHER TASKS REQUIRED TO SHUTDOWN THE CPU
- 4). AND FINALLY SEND THE PC TO THE PRINT ROUTINE



THE MICROCODE PRINT ROUTINE WILL ALWAYS SUBTRACT 2 FROM ANY GIVEN PC BEFORE ACTUALLY SENDING IT TO THE CONSOLE.

```

PC= 106          00000104 06
  - 2            >>>
  ---
   104

```

THIS LEAVES US AT A PC OF 104 WHICH IS ONE BYTE AHEAD OF THE ACTUAL OPCODE OF THE HALT INSTRUCTION.

THE REASON FOR THIS IS BECAUSE NOW WE CAN SIMPLY TYPE...

```
>>> C
```

AND CONTINUE ON WITH THE NEXT OPCODE FOLLOWING THE HALT INSTRUCTION.

ONE FINAL NOTE: DURING EXECUTION OF MACRO INSTRUCTIONS, IF ANY GIVEN INSTRUCTION BLOWS UP AFTER BEING DECODED ON AN IRD1, THE PC WOULD HAVE ALREADY BEEN UPDATED BY 2... SO THE PRINT ROUTINE CALLED IF WE WERE TO HALT THE CPU, WOULD SUBTRACT 2 FROM THE PC GIVING US THE CORRECT OPCODE ADDRESS OF THE FAILING INSTRUCTION.

THIS ALSO CLARIFIES WHY WE HAVE TO ADD 2 TO A MICRO-VERIFY ERROR HALT TO GET THE CORRECT FAILURE CODE. THE MICRO-VERIFY ROUTINE IS RESIDENT IN CCS ROM AND IS NOT A MACRO PROGRAM AT ALL! THUS IT DOES NOT UPDATE THE PC IN ANY WAY, BUT IT STILL USES THE SAME PRINT ROUTINE FOR THE ERROR DISPLAY.

THINK YOU'VE GOT THAT DOWN??? NOW TRY TO EXPLAIN IT TO A CLASS FULL OF BEWILDERED ENGINEERS!!!

GOOD LUCK.

**Machine and Bugchecks**



ALL RIGHT WE ARE OFF!!!! WHAT YOU SEE IN THE LOGOUT IS WHAT IS PUSHED ONTO THE STACK WHEN A MACHINE CHECK OCCURS WHILE NORMAL RUNNING OF VMS "AFTER" THE VECTOR ADDRESS IS BROUGHT IN IN FROM SCBB+4 AND THE VECTOR BITS 0 AND 1 ARE CHECKED. WE WILL ATTACK THE STACK DUMP FROM TWO AREAS;

1. INFORMATION RELATING TO LOCATION OF FAULT (PC ETC)
2. CAUSE OF THE FAULT.

LOCATION : AT (SP)+8 IS THE VIRTUAL ADDRESS REGISTER. THIS REGISTER IS USED TO FETCH THE OPERAND DATA NEEDED BY THE INSTRUCTION. SO IT CONTAINS THE OPERAND ADDRESS IF THE MACHINE CHECK OCCURRED WHILE FETCHING OPERAND DATA.

AT (SP)+C IS THE PC AT THE TIME OF THE EXCEPTION. THIS MAY BE USED WITH (SP)+2C WHICH IS THE ADDRESS OF THE OPCODE OF THE FAILING INSTRUCTION. EX: IF YOU ARE PREFETCHING AND USE AN INSTRUCTION AT ADDRESS 1000 AND THAT INSTRUCTION HAS 5 OPERAND SPECIFIERS THE ADDRESS OF THE OPCODE +2 IS STORED IN THE PC BACKUP REGISTER UNTIL THE NEXT OPCODE IS USED.(IRD1 TIME) AS YOU USE THE 5 OPERANDS IN THE INSTRUCTION THE PC (NOT PC BACKUP) IS INCREMENTED TO KEEP TRACK OF EXECUTION BUFFER USAGE. SO IF WE HAVE A MACHINE CHECK INVOLVED WITH EXECUTION BUFFER DATA, WE HAVE PUSHED ONTO THE STACK THE ACTUAL PC (SP+C) AND THE OPCODE OF THE INSTRUCTION (SP+2C).

AT (SP)+30 WE HAVE THE STANDARD PSL.

CAUSE: WE SHOULD FIRST LOOK AT THE SUMMARY PARAMETER CODE AT (SP)+4. GENERALLY SPEAKING YOU WILL ONLY HAVE NUMBERS 1,2,6 OR 7. 1,6 AND 7 ARE BASICALLY THE SAME THING. THESE MEAN A CONTROL STORE PARITY ERROR OCCURRED OR SOMEHOW THE MACHINE WAS SENT TO AN UNUSED IRD OR UNKNOWN ROM LOCATION. THIS COULD HAPPEN FOR A FEW REASONS, OF WHICH THE MOST LOGICAL IS THAT YOU HAVE A BAD CONTROL STORE, BAD MICROSEQUENCER ON THE DPM OR A BAD IRD DECODE ON THE DPM.

THE MOST COMMON AND HARDEST TO FIGURE OUT IS THE CODE OF 2. THIS RELATES TO MEMORY ERROR, TB PARITY TIMEOUT ETC. YOU LIKE THAT ETC. DO YOU. WELL LETS TAKE THE CONFUSION OUT OF THE STATEMENT. IF YOU EVER SEE A 2 FOR A SUMMARY PARAMETER CODE THE FIRST THING YOU SHOULD LOOK AT IS THE MACHINE CHECK ERROR SUMMARY REGISTER; (SP)+28. YOU CAN RELATE THIS REGISTER (MCSR) TO THE ABOVE CHART BECAUSE IT WILL TELL YOU WHAT CAUSED YOU TO GET TO UCODE ADDRESS 28. FIND THE BREAKOUT OF THE MCSR (PAGE 28 IN MINI REF. GUIDE) AND YOU WILL SEE A FOUR BIT REGISTER. LET US MAKE THE NEEDED CHANGE. THERE IS NO LONGER AN UNALIGNED UNIBUS REFERENCE THAT CAUSES A MACHINE CHECK, SO CROSS IT OFF. BIT 0 WILL TELL YOU IF THE MACHINE CHECK OCCURED WHILE DOING A PREFETCH OR OPERAND FETCH. (THIS MAY HELP YOU TO FIGURE ON USING THE VA OR PC FOR LOCATION)

IF BIT 0=0 THEN AN OPERAND FETCH WAS HAPPENING  
IF BIT 0=1 THEN A PREFETCH OF AN INSTRUCTION CAUSED IT.

BITS 2 AND 3 WILL TELL YOU IF IT WAS A TB ERROR OR BUS ERROR AS AN EXAMPLE WE WILL USE THE TB ERROR FIRST.

TB PARITY ERROR WHILE FETCHING AN OPERAND WOULD CAUSE THE REGISTER TO LOOK LIKE THIS WHEN PUSHED ON THE STACK

00000004 BIT 2 SET AND 0 CLEAR.

IF A TB ERROR OCCURED WHILE PREFETCHING IT WOULD BE AS FOLLOWS;

00000005 BIT 2 SET AND 0 SET.

EITHER WAY IF IT IS A TB ERROR YOU SHOULD THEN LOOK AT (SP)+1C OR THE TRANSLATION GROUP REGISTER. THIS WILL TELL YOU WHICH GROUP (0 OR 1) AND IF IT WAS A TAG OR DATA ERROR.

YOU MAY ALSO LOOK AT (SP)+14 WHICH IS THE SAVED MODE REGISTER. THIS WILL TELL YOU THE PROCESSOR ACCESS MODE AND MEMORY MANAGEMENT STATES DURING THE LAST MICROCODE REFERENCE TO MEMORY.

FROM THIS YOU SHOULD KNOW WHAT CAUSED THE MACHINE CHECK AND THE LOCATION.

LET US RETURN TO THE MCSR AND ASSUME IT LOOKED LIKE THIS;

00000008 BIT 3 SET 0 CLEAR  
THIS WOULD MEAN A BUS ERROR HAPPENED DURING AN OPERAND  
FETCH.

IF YOU LOOK AT THE CHART YOU WILL FIND THERE ARE TWO THINGS  
THAT CAN CAUSE A BUS ERROR. TO FIND OUT WHICH ONE IT WAS  
LOOK AT (SP)+24 THE BUS ERROR REGISTER. THE BUS ERROR  
REGISTER IS A FOUR BIT REGISTER IN THE MEMORY INTERCONNECT  
MODULE SLOT THREE. (NOT THE MEMORY CONTROLLER) THE EXAMPLE  
WE WILL USE FIRST IS UNCORRECTABLE DATA CAUSED THE BUS ERROR.

THE BUS ERROR REG. WOULD LOOK LIKE THIS;

00000004

THIS SAYS UNCORRECTABLE DATA CAUSED THE ERROR,  
THERE WERE NO LOST ERRORS (RECEIVED AN OTHER ERROR  
BEFORE THE LAST ONE WAS CLEARED)

!!!!CORRECTED READ DATA DID NOT OCCUR. CORRECTED  
READ DATA CAUSES AN INTERRUPT NOT A MACHINE CHECK!!!

IF YOU LOOK AT THE CHART YOU WILL FIND THAT UNCORRECTABLE  
DATA CAN BE CAUSED BY TWO THINGS;

1. CACHE PARITY ERROR
2. UNCORRECTABLE DATA FROM THE CMI

TO DETERMINE WHICH OF THESE CAUSED THE BUS ERROR LOOK  
AT (SP)+20 WHICH IS THE CACHE ERROR REGISTER. THIS  
REGISTER CONTAINS INFORMATION ON THE DATA CACHE. IT IS  
A FOUR BIT REGISTER ON THE MIC MODULE THAT WILL TELL YOU  
IF THE LAST REFERENCE WAS A HIT; LOST ERROR AGAIN AS BEFORE  
AND IF YOU HAD A CACHE PARITY ERROR. IF THERE WAS NO CACHE  
PARITY ERROR SET IN THE REGISTER THEN THE BUS ERROR WAS  
CAUSED BY THE UNCORRECTABLE DATA FROM THE CMI.

SO; CONTINUING RIGHT ON LET US ASSUME THAT THE BUS ERROR WAS CAUSED BY A NON EXISTANT MEMORY. AS YOU CAN SEE BY THE CHART THAT TWO THINGS CAN CAUSE NXM. FIRST LETS LOOK AT THE BUS ERROR REGISTER. IT EQUALS;

00000008 BIT 3 SET = NXM

THEN WE WOULD LOOK AT THE READ LOCK TIME OUT REGISTER (RLTO) THIS IS A ONE BIT REGISTER THAT IF BIT 0 IS SET A READ LOCK TIME OUT CAUSED THE NXM. WHAT IS A READ LOCK TIME OUT? GOOD QUESTION. IF THE CPU ATTEMPTS TO ACCESS THE CMI DURING A READ LOCK CONDITION A TIMER IS STARTED IN THE CMK GATE ARRAY ON THE MIC MODULE. IF THE TIMER RUNS FOR 64 USEC (USEC IS CORRECT) THEN THE CMK CHIP GENERATES NXM TO THE UTRAP CHIP THAT WILL CAUSE A MACHINE CHECK. IF BIT 0 IS CLEAR IN THIS REGISTER AND THE BUS ERROR REGISTER SAYS A NXM CAUSED THE MACHINE CHECK THEN IT WAS CAUSED BY NXM ON THE CMI.

THE ONLY THING THAT WAS PUSHED ONTO THE STACK THAT WE HAVE NOT TALKED ABOUT IS (SP)+10, THE MEMORY DATA REGISTER (MDR). THIS WILL CONTAIN THE LAST DATA FETCHED FROM CACHE OR MAIN MEMORY.

HOPEFULLY THIS EXPANATION, CHART AND HANDOUT WILL CLEAR UP SOME MISCONCEPTIONS CONCERNING THE 11/750 MACHINE CHECK.

## GENERAL 11/750 MICROCODE FLOW FOR A MACHINE CHECK

---

### 1. MACHINE CHECK EXCEPTION CONDITION OCCURS

THE VARIOUS TYPES ARE AS FOLLOWS:

```

A. BUS ERROR:----> NXM OFF CHI FROM: ----> CMC MODULE
                  | (NON EXISTANT MEMORY) | (MEMORY CONTROLLER)
                  |                         |
                  |                         | ----> UBI MODULE
                  |                         | (UNIBUS INTERFACE)
                  |                         |
                  |                         | ----> MBA MODULE
                  |                         | (MASBUSS ADAPTER)
                  |
                  | ----> UCE -----> UCE FROM CMC
                  | (UNCORRECTABLE ERROR) | (OR OTHER DEVICE)
                  |                         |
                  |                         | ----> CACHE PARITY ERROR
                  |
                  | ----> RLTO
                  | (READ LOCK TIME OUT)
    
```

```

B. TB ERROR:-----> TRANSLATION BUFFER TAG PARITY ERROR
                  |
                  | ----> TRANSLATION BUFFER DATA PARITY ERROR
    
```

THE TWO CATEGORIES OF MACHINE CHECK CONDITIONS CAN BE BROKEN DOWN INTO TWO MORE GROUPS:

```

A. SOURCING DATA FROM I-STREAM:--> MSRC XB TB ERROR
SEE NOTE
                  | *****
                  | * TRANSLATION BUFFER ERROR *
                  | * ENCOUNTERED WHEN SOURCING *
                  | * THE BAD DATA FROM THE *
                  | * EXECUTION BUFFER *
                  | *****
                  |
                  | ----> MSRC XB BUS ERROR
                  | *****
                  | * BUS ERROR ENCOUNTERED *
                  | * WHEN SOURCING THE BAD DATA*
                  | * FROM THE EXECUTION BUFFER *
                  | *****
    
```

NOTE<sup>1</sup> WHEN A TB OR BUS ERROR OCCURS DURING A PREFETCH, THE ERROR IS IGNORED UNTIL WE ATTEMPT TO SOURCE THE BAD DATA FROM THE EXECUTION BUFFER. THIS IS TO PREVENT UNNECESSARY ERROR HANDLING OF DATA THAT MIGHT NOT GET USED ANYWAY. THE DATA IN THE XB IS NOT ALWAYS THE RIGHT DATA TO BE EXECUTED, FOR EXAMPLE: IF THE CURRENTLY EXECUTING INSTRUCTION IS A BRANCHING INSTRUCTION IT WILL MODIFY THE PC THUS CAUSING AN EXECUTION BUFFER FLUSH WHICH CLEARS OUT THE XB AND FILLS IT WITH THE DATA FROM THE NEW PC AND PC+4.



B. ERROR DURING INSTRUCTION DECODE:---->BUT X3 TB ERROR  
(IRD1/IRDX)

```

| *****
| * TB ERROR ENCOUNTERED *
| * DURING AN IRD1 OR IRDX *
| *****
|
---->BUT XB BUS ERROR
*****
* BUS ERROR ENCOUNTERED *
* DURING AN IRD1 OR IRDX *
*****

```

2. MSQ, UTR AND SAC CHIPS SET UP A MICRO VECTOR OF 0028 AT THE OUTPUT OF THE MICROSEQUENCER, SENDING US TO THE PROPER MICRO ADDRESS AND THE MACHINE CHECK MICRO ROUTINE SETS UP OUR SCBB+4 AND BUILDS THE STACK.
3. SCBB+4 CONTAINS OUR MACRO VECTOR ADDRESS
4. USE THE LOWER TWO BITS TO SELECT A STACK:

VECTOR BITS <1> | <0>

-----

```

0 | 0 > USE KERNAL STACK UNLESS <IS> BIT
  |   IS SET IN PSL
0 | 1 > USE INTERRUPT STACK
  |
1 | 0 > TRAP TO WCS ADDRESS 2001 IF WCS
  |   IS NOT PRESENT, TRAP TO 0001 IN CCS
1 | 1 > HALT AT VECTOR PC POINTS TO
    INTERRUPTED OR FAULTED INSTRUCTION.

```

00000000 07  
>>>

5. PUSH PSL, PC AND 11 OTHER LONGWORDS OF INFORMATION ON STACK.
6. LOWER TWO BITS OF VECTOR GET ZEROS WHEN CROSSING CMI ON ADDRESS CYCLE. THE ADDRESS POINTED TO BY THE VECTOR WILL BE THE START OF THE MACRO MACHINE CHECK HANDLER ROUTINE.
7. IRD1 OF MACRO ROUTINE TAKES PLACE.



# 161

IF THE MACHINE CHECK TURNS INTO A BUGCHECK, IT WILL HAVE THE FOLLOWING RESULTS:

	FATAL BUGCHECK CODE	NON-FATAL BUGCHECK CODE	
		BUGCHKFATAL BIT SET	BUGCHKFATAL BIT CLEAR
KERNAL MODE	SHUTDOWN THE CPU	LOG THE ERROR SHUTDOWN THE CPU	LOG THE ERROR THEN REI
EXECUTIVE MODE			
	PROCESS HAS BUGCHECK PRIV.	PROCESS DOES NOT HAVE BUGCHECK PRIV.	
SUPERVISOR MODE	LOG THE ERROR THEN	DO NOT LOG THE ERROR BUT	DISMISS THE ERROR THEN REI
USER MODE	SEXIT_S	SEXIT_S	

## BUGCHECKS

A BUGCHECK is an internal inconsistency within a process or VMS, such as a corrupted data structure or unexpected exception, detected by VMS. Bugchecks can be the result of programming errors or hardware failures. Software related Bugchecks can be quickly isolated from the information saved in the system dump file on a system crash and from source listings. Hardware related Bugchecks are not so easy to isolate because the hardware failure can occur long before VMS detects it. Later on we will look at how to troubleshoot some of the common Bugcheck failures.

Bugchecks are not always fatal to the system. A Bugcheck that occurs while the CPU is in either User or Supervisor mode will result in termination of the process that incurred the Bugcheck, providing the process does not have privilege to cause a Bugcheck. If the Bugcheck is not fatal, VMS will dismiss it and allow the process to continue. Otherwise fatal Bugchecks will not crash the system from User or Supervisor mode.

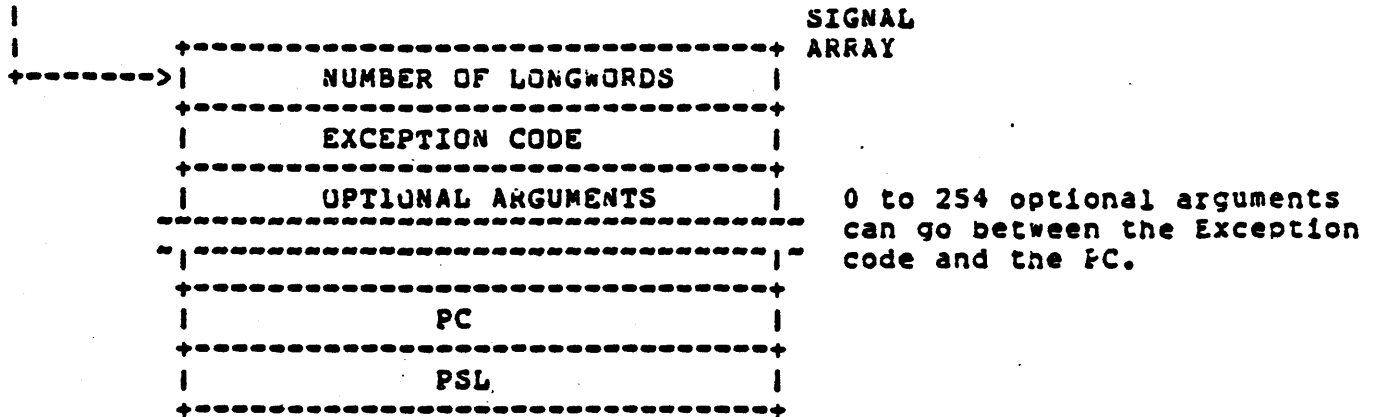
VMS protects itself and its data structures by using the Bugcheck mechanism while in Executive or Kernel mode. Non-fatal Bugchecks which occur in Executive or Kernel mode are dismissed the same as those in Supervisor or User mode, unless the SYSBOOT parameter BUGCHECKFATAL is turned on. Non-fatal Bugchecks will be logged to the Error Log. Fatal Bugchecks will result in the orderly shutdown of the system. A small amount of information describing the Bugcheck is sent to the console terminal, a dump file is written to the disk and then a special code is sent to the CON chip's console transmit data buffer and a HALT instruction is executed. The system will then be rebooted unless the SYSBOOT parameter flag BUGREBOOT is cleared.

The crash dump file can be analyzed using the System Dump Analyzer (SDA). The size of the dump file must be four blocks larger than the number of physical pages in the system. If the space reserved on the disk for the dump file is too small, only the physical pages that can fit in the file will be written. A small dump file will not contain some of the most crucial contents of physical memory (the system page tables) which may make analysis with SDA impossible.



The values contained in the MECHANISM ARRAY are:

- o 00000004 -- the number of longwords in the MECHANISM ARRAY. In a MECHANISM ARRAY, this value is always four.
- o FRAME -- the address of the stack frame.
- o DEPTH -- the stack depth. (FFFFFFFFD to FFFFFFFF) Look for this when the Vector Array is not on the stack.
- o R0 -- the contents of R0 at the time of the exception.
- o R1 -- the contents of R1 at the time of the exception.



As an Example:

The values contained in the SIGNAL ARRAY for an Access Violation Exception which caused a Bugcheck are:

- o 00000005 -- the number of longwords contained in the SIGNAL ARRAY. For access violations this number is always five.
- o EXCEPTION CODE -- a code which identifies the type of exception.
- o REASON MASK -- the longword whose lowest three bits, if set, indicate that the instruction caused a Length Violation (bit 0), referenced the process page table (bit 1), and/or read/modify operation (bit 2).
- o VIRTUAL ADDRESS -- the virtual address that the system tried to reference at the time of the exception.
- o PC -- the Program Counter. The PC contains the address of the instruction that signaled the exception.
- o PSL -- the processor status longword at the time of the exception.

Signal arrays differ in length, from 4 to 258 longwords, depending on the kind of exception the system detects. See the VAX-11 Run Time Library Reference Manual for details.

The Signal Array contains more interesting information about the bugcheck. The format of the Signal Array varies for different Bugchecks. The Exception Code identifies what kind of error led to the Bugcheck. The Exception Code indicates such errors as Access Violation, Opcode Reserved, DEC, etc. Following the Exception Code are optional arguments. These arguments will vary in number and meaning for different Bugchecks. Next on the stack is the PC of the instruction that would have been executed next, if an Exception had not occurred.

Once you have located the Exception Code within the Signal Array, enter the following on a running VAX/VMS system:

```
s @TT<cr>
s _EXIT %X<exception code><cr>
```

For access violations the EXCEPTION CODE is 0000000C.

EXAMPLE:

```
s @TT<cr>
s _EXIT %X0C<cr>
```

```
%SYSTEM=F-ACCVIO, access violation, reason mask=00,
virtual address=0000000C, PC=7FFD3A58, PSL=0004034
```

Now that you know what the Exception Code means, you can look up a short explanation in the VAX/VMS System Messages and Recovery Procedures Manual. For instance, continuing with the Access Violation example, you would look up ACCVIO on page 2-3 and find the following:

```
" ACCVIO, access violation, reason mask=xx, virtual address=location,
PC=location, PSL=xxxxxxxx
```

Facility: VAX/VMS System Services

Explanation: An image attempted to read from or write to a memory location that is protected against the current mode. This message indicates an exception condition and is followed by a register and stack dump to help locate the error.

User Action: Examine the PC and virtual address displayed in the message and check the program listing to verify that instruction operands or procedure call arguments are correct. "

The explanation given in the VAX/VMS System Messages and Recovery Manual will give you an idea of what the software was attempting to do or a description of the Exception condition which led to the Bugcheck. The User Action may give you some idea of how to proceed in examining the crash dump. Remember that this manual was intended for programmers creating program errors and not for analyzing hardware failures, so some of the Explanations and User Actions will not be appropriate to a hardware failure.

Now that you have some idea where the Bugcheck error was detected and what type of an error caused the Bugcheck, you can attempt a bit of analysis using SDA. The above stack information may be available at the console or by using SDA and examining the stack. Exactly how you proceed with SDA will depend on your experience and the type of problem you are troubleshooting.

For instance, suppose you had an access violation caused by a length violation which led to a Bugcheck. The VA that failed can be found in the Signal Array. Try to examine this address using SDA. It will probably not be possible because the page may not have been mapped. Then check the process page table or system page table to find out if the address is mapped and what protection exists. If the VA is an 800xxxxx value, then you can use the system map (SYS.MAP) and locate the VMS module which contains the address. If the address is not mapped, it may indicate that the program calculated the address incorrectly or dropped/picked a bit in the data paths because of a hardware error. Try to figure out what the address should have been and if the VA that was generated is off by a single bit. Maybe one particular register dropped a bit. From a single failure you may not have enough information to isolate the problem to a small enough area of the system to warrant swapping a module. In these cases it is better to wait for additional crashes and collect more information.

Another possibility is that a device could cause an error, such as constant interrupts, which could cause a system crash or hang. Be especially suspicious of the system disk, MBA or Massbus if all of the failures happen while page faulting a page or swapping a process.

A customer written device driver, or for that matter a DEC device driver, could cause a Bugcheck. If the VA or PC which causes the failure is 800xxxxx and you cannot find the module which contains this address in the SYS.MAP, then the address may be within a device driver or other VMS component such as RMS. To find out if it is within a device driver, run SYSGEN and SHOW /DEVICES. The SHOW/DEVICES command will print out a list of address indicating where each device driver is loaded, and addresses where key structures within the I/O data base can be found. The SHOW DEVICE command under SDA could also be used. Just knowing that the address which caused the Bugcheck is associated with a particular device driver will give you some idea of where to start. In the case of a suspected customer written device driver, it would be wise to involve Software Support to help analyze the crash and look at the code of the device driver.



## BUGCHECK ANALYSIS NUMBER ONE

Let's try looking at an example of one Bugcheck which was forced by hardware error and see if we can determine where the problem lies.

\*\*\*\* COMMENTS and SDA COMMANDS are indicated by "\*" \*\*\*\*

SDA> SHOW CRASH

\*\*\*\*\*

VAX/VMS System dump analyzer

Dump taken on 13-JUL-1981 16:19:26.67  
SSRVEXCEPT, Unexpected system service exception

Time of system crash: 13-JUL-1981 16:19:26.67

Version of system: VAX/VMS VERSION V2.3

Reason for BUGCHECK exception: SSRVEXCEPT, Unexpected system service exception

Process currently executing: SYSTEM

Current image file name: \_DRA0:[SYSEXE]DIRECTORY.EXE;3 \*\* GETTING A DIRECTORY

Current IPL: 0 (decimal)

General registers:

\*\*\*\* THE CONTENTS OF REGISTERS R0,R1,SP,PC,& PSL HAVE BEEN \*\*\*\*  
\*\*\*\* MODIFIED BY THE BUGCHECK HANDLER. THE PC IS POINTING \*\*\*\*  
\*\*\*\* TO THE BUGCHECK HANDLER FOR SYSTEM SERVICE EXCEPTION. \*\*\*\*

R0 = 7FFEF35	R1 = 8000A122	R2 = 7FFEC200	R3 = 7FFEA200
R4 = 80070EAO	R5 = 7FFEA838	R6 = 7FFEA8EC	R7 = 00000000
R8 = 7FFEF878	R9 = 7FFEF988	R10 = 7FFEA790	R11 = 7FFEA210
AP = 7FFECD84	FP = 7FFECD6C	SP = 7FFECD6C	PC = 8000A128
PSL = 00000000			

Processor registers:

POBR = 80097000	PCBB = 0001A674	ACCS = 00008001
POLR = 00000000	SCBB = 0007DA00	SBIFS = 00040000
P1BR = 7F89B000	ASTLVL = 00000004	SBISC = 00000000
P1LR = 001FFE87	SISR = 00180000	SBIMT = 00200200
SBR = 0007E400	ICCS = 800000C1	SBIER = 00008002
SLR = 00000700	ICR = FFFFE66B	SBITA = 20000001
	TODR = 738E01C0	SBIS = 00000000

ISP = 8007F000	** ON THIS SYSTEM, THIS IS AN EMPTY STACK **
KSP = 7FFECD6C	** THIS IS THE CURRENT STACK **
ESP = 7FFEDDB0	
SSP = 7FFEF818	
USP = 7FFCC9C8	

SDA&gt; SHOW PROCESS

\*\*\*\*\*

Process status: 00040001 RES,PHDRES

PCB address	80070EA0	JIB address	8007A980
Master PID	00020016	Creator PID	00000000
PID	00020016	Subprocess count	0
PHD address	80096600	Swapfile disk address	00000000
State	CUR	Termination mailbox	0000
Current priority	4	AST's enabled	KESU
Base priority	4	AST's active	NGNE
UIC	[001,004]	AST's remaining	19
Mutex count	0	Buffered I/O count/limit	12/12
waiting EF cluster	0	Direct I/O count/limit	12/12
Starting wait time	1A1B0000	BUFIO byte count/limit	20480/20480
Event flag wait mask	F7FFFFFF	# open files allowed left	20
Local EF cluster 0	C8000001	Timer entries allowed left	20
Local EF cluster 1	00000000	Active page table count	0
Global cluster 2 pointer	00000000	Process WS page count	40
Global cluster 3 pointer	00000000	Global WS page count	55

A> SHOW STACK  
 |\*\*\*\*\*|

Current operating stack (KERNEL):

	7FFECD4C	7FFEF988	
	7FFECD50	7FFEA790	CTLSAG_CLIDATA+580
	7FFECD54	7FFEA210	CTLSAG_CLIDATA
	7FFECD58	7FFECD84	CTLSGL_KSTKBAS+584
	7FFECD5C	7FFECD6C	CTLSGL_KSTKBAS+56C
	7FFECD60	7FFECD64	CTLSGL_KSTKBAS+564
	7FFECD64	8000A128	EXESEXCPIN+006
	7FFECD68	00000000	
SP =>	7FFECD6C	00000000	
	7FFECD70	00000000	
	7FFECD74	7FFEA956	MNGSIMGACTBUF+156
	7FFECD78	7FFECD0C	CTLSGL_KSTKBAS+5C0
	7FFECD7C	80000014	SYSSCALL_HANDL+004
	7FFECD80	80011265	EXESREFLECT+14E
	7FFECD84	00000002	<===== *** THE VECTOR ARR.
	7FFECD88	7FFECDA4	CTLSGL_KSTKBAS+5A4
	7FFECD8C	7FFECD90	CTLSGL_KSTKBAS+590
	7FFECD90	00000004	<===== *** THE MECHANISM
	7FFECD94	7FFECD0C	CTLSGL_KSTKBAS+5C0
	7FFECD98	FFFFFFFFE	
	7FFECD9C	00400005	
	7FFECDA0	7FFEC000	CTLSA_DISPVEC
	7FFECDA4	00000005	<===== *** THE SIGNAL ARR
	7FFECDA8	00000444	<===== *** NOW FIND THE E
	7FFECDAC	00000000	
	7FFECD80	80011A00	EXESIMGSTA+5C2
	7FFECD84	800119FD	EXESIMGSTA+5BF
	7FFECD88	00400000	
	7FFECD8C	8000E39F	EXESIMGACT+CAC
	7FFECD90	00000000	
	7FFECD94	00CC0000	
	7FFECD98	7FFEDDC8	CTLSGL_KSPINI+FC8
	7FFECD9C	7FFECDE4	CTLSGL_KSTKBAS+5E4
	7FFECDD0	80007658	EXESCMKRNL+018
	7FFECDD4	00000000	
	7FFECDD8	7FFEA000	MNGSIMGACTBUF+600
	7FFECDDC	7FFEA8EC	MNGSIMGACTBUF+0EC
	7FFECDE0	00000000	
	7FFECDE4	00000000	
	7FFECDE8	00000000	
	7FFECDEC	7FFEDDC8	CTLSGL_KSPINI+FC8
	7FFECDF0	7FFEDDB0	CTLSGL_KSPINI+FB0
	7FFECDF4	8000A130	EXESEXCPIN+00E
	7FFECDF8	80000096	SYSSCMKRNL+006
	7FFECDFC	01800000	

\*\*\*\* If you have been following the crash so far, you should know that the Exception Code was a 444. Using the methods shown earlier, you should have been able to determine that the exception code indicates a PAGRDERR, that is a PAGRDERR. Now looking that up in the VAX/VMS System Messages and Recovery Procedures Manual you would find the following:

" PAGRDERR, page read error, reason mask=xx, virtual address=location, PC=location, PSL=xxxxxxx

Facility: VAX/VMS system services

Explanation: The system failed to read a page from disk into memory during a page fault operation. This message indicates an exception condition and is usually followed by a display of the condition arguments, registers, and stack at the time of the exception.

User Action: Check the status of the device and repeat the request. If the failure persists, notify the system manager."

Now what do you think would be a good area to examine? While it is not possible from the information above to state conclusively that the Bugcheck was caused by a hardware failure in the disk subsystem, the available evidence is pointing in that direction. This Bugcheck was in fact caused by switching the system disk offline/online and then attempting to perform a DIR command. As you can see, this Bugcheck was fairly straight forward and could be isolated to the disk subsystem. If this Bugcheck occurred again and the hardware was available, you could look at the disk subsystem registers. You would find the Volume Valid bit reset. From this you would then be able to pursue the MBA or Disk drive to determine why the Volume Valid bit was reset on the system disk.



## EDT Version 2 VT100 Keypad

Gold	Help	Fndnxt	Del L
		Find	Und L
Pase	Sect	Append	Del W
Command	Fill	Replace	Und W
Advance	Backup	Cut	Del C
Bottom	Top	Paste	Und C
Word	Eol	Char	Enter
Chnscase	Del Eol	Specins	
	Line	Select	Subs
	Open Line	Reset	

Backspace	Go to beginning of line
Delete	Delete character
Linefeed	Delete to start of word
CTRL/A	Compute tab level
CTRL/D	Decrease tab level
CTRL/E	Increase tab level
CTRL/K	Define key
CTRL/L	Form feed
CTRL/T	Adjust tabs
CTRL/U	Delete to start of line
CTRL/W	Refresh screen
CTRL/Z	Return to line mode

## EDT Version 2 VT52 Keypad

Gold	Help	Del L	Up
		Und L	Replace
Pase	Fndnxt	Del W	Down
Command	Find	Und W	Sect
Advance	Backup	Del C	Right
Bottom	Top	Und C	Specins
Word	Eol	Cut	Left
Chnscase	Del Eol	Paste	Append
	Line	Select	Enter
	Open Line	Reset	Subs

Backspace	Go to beginning of line
Delete	Delete character
Linefeed	Delete to start of word
CTRL/A	Compute tab level
CTRL/D	Decrease tab level
CTRL/E	Increase tab level
CTRL/F	Fill text
CTRL/K	Define key
CTRL/L	Form feed
CTRL/T	Adjust tabs
CTRL/U	Delete to start of line
CTRL/W	Refresh screen
CTRL/Z	Return to line mode





FOR INTERNAL USE ONLY

\*\*\*\*\*  
\* THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT \*  
\* NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL \*  
\* EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES \*  
\* NO RESPONSIBILITY FOR ANY ERRORS WHICH MAY APPEAR IN THIS TEXT.\*  
\* PREPARED BY EDUCATIONAL SERVICES DEPARTMENT INSTRUCTORS OF \*  
\* DIGITAL EQUIPMENT CORPORATION \*  
\*\*\*\*\*

FOR INTERNAL USE ONLY

ANY SUGGESTIONS OR COMMENTS CONCERNING THIS DOCUMENT SHOULD BE DIRECTED TO:

DIGITAL EQUIPMENT CORPORATION  
EDUCATIONAL SERVICES  
VAX 11/750 MAGIC BOOK

12 CROSBY DRIVE BU0/E38  
BEDFORD, MASSACHUSETTS 01730

OR CALL DTN-249-4697  
(617) 276-4697