Alan Ross

EK-VAXV3-HB-001

VAX Maintenance Handbook

VAX-11/750

digital

EK-VAXV3-HB-001

VAX Maintenance Handbook

VAX-11/750

1983 Edition

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, March 1983

Printed in U.S.A.

The manuscript for this book was created on a Word Processing System. Book production was done by Educational Services Development and Publishing in South Lawrence, MA.

CONTENTS

CHAPTER 3 MODULES AND GATE ARRAYS

CHAPTER 4 CONFIGURATION AND CABLING

CHAPTER 6 RDM AND MICRODIAGNOSTICS

CHAPTER 7 BLOCK DIAGRAMS

CHAPTER 8 CMI, UNIBUS, AND MASSBUS

CHAPTER 9 TROUBLESHOOTING AIDS

CHAPTER 10 CHARTS AND MACROS

CHAPTER 1
INTRODUCTION

INTRODUCTION

This handbook is a summary of VAX-11/750 system maintenance information. It is intended to serve as a single source reference for DIGITAL Field Service, and for Engineering, Manufacturing, and Training personnel.

The material complements detailed information available in hardware and service manuals and in maintenance print sets. It contains tables, diagrams, and procedures, and assumes that the reader is familiar with the VAX-11/750 system, its nomenclature and mnemonics.

For further information, refer to the combined VAX-11/780 and VAX-11/750 microfiche library, and to tables included in this chapter that list related hardware manuals.

Hard copy documents may be ordered through the nearest DIGITAL sales office, the Accessories and Supplies Group catalog (Documentation Products Directory), or directly from the following addresses.

Customers and OEMs:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attn: Publishing and Circulation Services, NR2
Customer Services Section

Digital personnel:

Digital Equipment Corporation
10 Forbes Road
Northboro, MA 01532

Attn: Publishing and Circulation Services, NR3

Technical descriptions and service manuals are also available on microfiche. For information on microfiche libraries (including maintenance print sets and diagnostic listings), contact:

Digital Equipment Corporation
Micropublishing Systems, BU/E46
12 Crosby Drive
Bedford, MA 01730

VAX-11/750 SYSTEM HARDWARE MANUALS

Title	Document Number
VAX-11/750 Central Processor Unit Technical Description	EK-KA750-TD
VAX-11/750 UNIBUS Interface Technical Description	EK-UI750-TD
VAX-11/750 Memory System Technical Description	EK-MS750-TD
RH750 MASSBUS Adapter Technical Description	EK-RH750-TD
DW750 Second UNIBUS Interface Technical Description	EK-DW750-TD
FP750 Floating Point Accelerator Technical Description	EK-FP750-TD
VAX-11/750 H7104 Power System Technical Description	EK-PS750-TD
VAX Architecture Handbook	EB-19580-20
VAX Hardware Handbook	EB-21710-20
VAX Software Handbook	EB-21812-20
VAX Systems Site Preparation Guide	ED-22517-20
VAX-11/750 Installation and Acceptance Test Manual	EK-SI750-IN ✓
VAX-11/751 User's Guide	EK-11751-UG
VAX Maintenance Handbook, VAX Systems	EK-VAXV1-HB ✓
VAX Diagnostic System User's Guide	EK-VX11D-UG ✓
VAX-11/750 Diagnostic System Overview Manual	EK-VXD75-UG ✓
VAX-11/750 Diagnostic Mini Reference Guide	EK-KC750-RM
VAX-11/750 Gate Array Chip Reference Manual	EP-GA750-RM*

*Available only on microfiche

Title	Document Number
KC750 Microdiagnostic/Technical Manual	EK-KC750-TM
KC750 Options User Guide	EK-KC750-UG
KC750 Options Installation Guide	EK-KC750-IN
KC750 Technical Change Notice	EK-KC750-N1
VAX Diagnostic Design Guide	EK-1VAXD-TM
VAX-11/750 Microdiagnostic Reference Manual	EK-750YA-RM ✓
VAX-11/750 Self-Maintenance Diagnostic Guide	EK-750YA-UG ✓
VAX-11/750 Change Notice	EK-750YA-01
VAX-11/750 System Map	EN-01570-12
VAX-11/750 PM Worksheets	EK-11750-WS
VAX-11/750 Equipment Care Sheets	EK-11750-EC
Site Preparation Data Sheets	EK-CORP-SP

VAX-11/750 PERIPHERAL MANUALS

Title	Document Number
LA38 DECwriter IV IPB	EK-ØLA38-IP
LA34 DECwriter IV IPB	EK-LA34S-IP
DECwriter IV Technical Manual	EK-LA34S-TM
DECwriter IV Pocket Service Guide	EK-LA34S-PS
LA34 DECwriter IV User Guide	EK-ØLA34-UG
LA34 User's Guide Addendum	EK-ØLA34-N1
LA34 DECwriter IV Reference Guide	EK-ØLA34-RG
LA12Ø Technical Manual	EK-LA12Ø-TM
LA12Ø User's Guide	EK-LA12Ø-UG
LA12Ø User's Guide Addendum	EK-ALA12-UG
LA12Ø Operator Reference Card	EK-LA12Ø-RG
LA12Ø Pocket Service Guide	EK-LA12Ø-SV
LA12Ø Pocket Service Guide Addendum	EK-ALA12-SV
LA12Ø Series Pocket Service Guide	EK-LA12Ø-PS
LA12Ø Customer Equipment Care	EK-LA12Ø-EC
LA12Ø DECwriter III IPB	EK-LA12Ø-IP
TU58 DECtape II Technical Manual	EK-ØTU58-TM
TU58 DECtape II User's Guide	EK-ØTU58-UG
TU58 DECtape II Pocket Service Guide	EK-ØTU58-PS
TU58 DECtape Customer Equipment Care	EK-ØTU58-EC
TU58 Cartridge Tape Drive IPB	EK-ØTU58-IP
TU58-VA DECtape II Configuration	EK-ØTU58-CG
TU58 DECtape Installation Sheets	EK-ØTU58-WS

VAX-11/750 PERIPHERAL MANUALS (CONT)

Title	Document Number
PDP-11 DECdisk Subsystem Register Reference Card	EH-18955-18
RK06/07 Technical Description Manual	EK-RK067-TD
RK06/07 User's Manual	EK-RK067-UG
RK06/07 Disk Drive Service Manual	EK-RK067-SV
RK06/07 FTB Operating/Service Manual	EK-RK67F-OP
RK06/07 Customer Care Sheet	EK-RK067-EC
RK07 Disk Drive IPB	EK-0RK07-IP
RK611 Technical Description Manual	EK-RK611-TM
RK611 Controller IPB	EK-RK611-IP
RM03 Technical Manual	ER-0RM03-TM
RM03 Disk Drive Maintenance Manual	ER-0RM03-MP
RM03 Disk Drive IPB	EK-0RM03-IP
RM02/03 User's Guide	EK-RM023-UG
RM02/03 Disk Subsystem Service Manual	EK-RM023-SV
RM02/03 Customer Equipment Care	EK-RM023-EC
RM02/03 PM Worksheets	EK-RM023-WS
TS11 Subsystem Technical Manual	EK-0TS11-TM
TS11 Subsystem User's Guide	EK-0TS11-UG
TS11 Subsystem Customer Equipment Care	EK-0TS11-EC
TS11 Pocket Service Guide	EK-0TS11-PS
TS11-A Magnetic Tape Subsystem IPB	EK-TS11A-IP
TS11-B Magnetic Tape Subsystem	EK-TSB11-IP
TS11-B Magnetic Tape Subsystem IPB	EK-TS11B-IP
TS11-C Magnetic Tape Subsystem IPB	EK-TS11C-IP

VAX-11/750 PERIPHERAL MANUALS (CONT)

Title	Document Number
RMØ5 Disk Drive Service Manual	EK-ØRMØ5-SV
RMØ5 Disk Subsystem User's Guide	EK-ØRMØ5-UG
RMØ5 Disk Drive IPB	EK-ØRMØ5-IP
RMØ5 PM Worksheets	EK-ØRMØ5-WS
TU77 Magnetic Tape Transport V1	EK-1TU77-TM
TU77 Magtape Technical Manual V2	EK-2TU77-TM
TU77 Technical Update	EK-2TU77-Ø1
TU77 Magtape Transport User's Guide	EK-ØTU77-UG
TU77 Magnetic Tape Addendum	EK-ØTU77-Ø1
TU77 Magnetic Tape Transport IPB	EK-ØTU77-IP
TU77/TU78 Magnetic Transport Equipment Care	EK-TU778-EC
TU77/78 PM Worksheets	EK-TU778-WS
RM8Ø Disk Drive Technical Description	EK-ØRM8Ø-TD
RM8Ø Disk Drive Service Manual	EK-ØRM8Ø-SV
RM8Ø Disk Drive User's Guide	EK-ØRM8Ø-UG
RM8Ø Disk Drive Pocket Guide	EK-ØRM8Ø-PG
RM8Ø Disk Drive Customer Care Package	EK-ØRM8Ø-EC
RM8Ø Disk Drive IPB	EK-ØRM8Ø-IP
TE16/TE1ØW/TE1ØN Maintenance Manual	EK-ØTE16-TM
TE16/TE1ØW/TE1ØN Equipment Care	EK-ØTE16-EC
TE16 DEC Magtape IPB	EK-ØTE16-IP

VAX-11/750 MAINTENANCE PHILOSOPHY

VAX-11/750 system maintenance consists of the repair or replacement of modules, components, or complete hardware units, depending on the type of hardware under test and the level of testing that applies to each unit.

Overall CPU maintenance consists of replacing gate array chips on the defective CPU module as determined by microdiagnostic testing. If replacing the gate array(s) called out by the microdiagnostic test does not fix the problem, the module must be replaced.

Corrective maintenance for internal CPU options such as the floating-point accelerator (FPA) module and the remote diagnostic module (RDM) consists of module replacement if the associated diagnostic indicates a failure.

The MASSBUS adapter (MBA) and second UNIBUS (SUB) modules are also replaced if the associated diagnostic indicates a failure. Attached external devices normally follow field replaceable unit (FRU) fault isolation and are serviced according to their respective maintenance philosophies.

The following tables provide an overview of the corrective action prescribed for devices inside the VAX-11/750 system cabinet.

CPU MODULES

Module Name (Mnem)	Module Number (Slot)	Level of Replacement: Component/ Gate Array	Level of Replacement: Replace Module	Notes/Comments
Data Path (DPM)	LØØØ2 (2)	Primary (micro-diagnostic testing)	Secondary	Replace module if the gate arrays do not fix the problem.
Memory Inter-connect (MIC)	LØØØ3 (3)	Primary	Secondary	Replace module if the gate arrays do not fix the problem.
UNIBUS Inter-face (UBI)	LØØØ4 (4)	Primary	Secondary	Tested by running level 3 diagnostic, ECCBA.EXE, on Tape 6.
CPU Control Store (CCS)	LØØØ5 (5)		Primary	No diagnostic to test this module.

NOTE:
A way to determine the condition of the CCS is to do a parity check as follows. The LØØØ6 remote diagnostic module (RDM) must be in place.

```
>>>  ^P^D        Performing this test produces a
RDM> STO         series of address printouts.
RDM> PAR Ø
      .          If CSAD = 17FD (hex),
      .          the CCS is okay.
      .
CSAD = 17FD
```

The RDM is also known as the LØØØ6YA maintenance tool module (MTM) to customers.

CPU OPTIONS

Module Name (Mnem)	Module Number (Slot)	Level of Replacement: Component/ Gate Array	Level of Replacement: Replace Module	Notes/Comments
Remote Diag-nosis (RDM)	L0006 (6)		Primary	DIGITAL-owned option used to run micro-diagnostics locally or remotely. Test by running ECKAF.EXE
Float-ing Point Acceler-ator (FPA)	L0001 (1)		Primary	Tested by running EVKAC.EXE.
MASSBUS Adapter (MBA)	L0007 (7,8,9)		Primary	Up to three MBAs may exist.
Second UNIBUS Inter-face (SUB)	L0010 (7,8,9)		Primary	SUB typically resides in slot 7. Microcode driven it contains its own NPR arbitrator. UBI handles BR arbitration for first and second UNIBUS and MASSBUS (option slots).
CMI/Memory Control (CMC)		Primary (micro-diagnostic testing)	Secondary	Microcode driven, the CMC controls memory re-freshes error checking, data buffering, select timing and contains ECC logic. Test by running ECKAM.EXE.
CMC 1, 16K RAMs	L0011 (10)			CMC 1 = 256 kilobytes per memory array module
CMC 2, 64K RAMs	L0016 (10)			CMC 2 = 1 megabyte per memory array module
TU58 Control/ Drive			Replace module/ drive unit	Test with sections 8 and 9 of diagnostic ECKAX.EXE.

THE REMOTE DIAGNOSTIC FACILITY

The customer is required to provide a voice grade telephone line and connector for DIGITAL Diagnostic Center (DDC) communication.

The RDM option will be installed in all VAX-11/750 systems during installation to prove its value to the customer during the warranty period. It will be left in the backplane for all customers with the standard RD maintenance contract.

CONTACTING THE DDC

Basic Flow:

1. The customer calls the DDC toll free number when there is a problem.

1-800-525-6570	For DDC connection (toll free)
1-303-599-4000	For Field Service assistance
1-303-593-7890	U.S.F.S Library (DEC employees only)

   The Library mail stop is CX/DDC (Colorado Springs).

2. The DDC performs remote subsystem fault isolation and identifies the failing option to the Branch office.

3. The Branch office sends a Field Service Engineer with parts to correct the problem.

For CPU Problems:

1. The engineer takes the CPU spares and RDM tool to the site.

2. The engineer runs the microdiagnostic cassette tapes.

   For customers with non-RD contracts, the RDM tool is installed in the VAX-11/750 backplane and is removed when work is complete.

   On CPU modules, faults are isolated to a specific module and also to a string of chips (average of two gate arrays).

3. The engineer performs component level replacement (CLR) by replacing the indicated gate arrays.

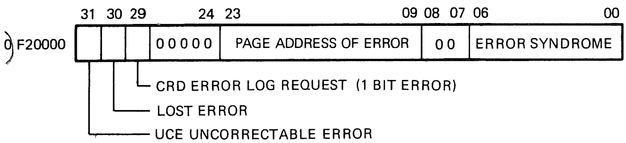
   When CLR does not correct the fault on CPU modules or other CPU-related failures, the failing module or assembly is replaced.

   IMPORTANT: The fix should be verified with the DDC to assist them in building a case history of failures for the VAX-11/750.

CHAPTER 2
SYSTEM REGISTERS

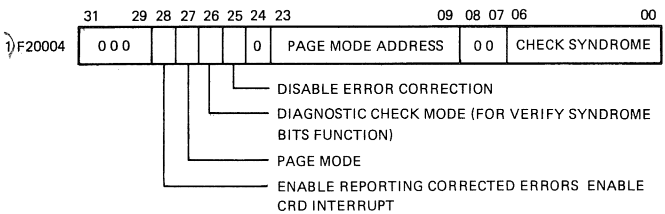
CMC (MS750) REGISTERS

CONTROL/STATUS REGISTER 0 (CSR 0)



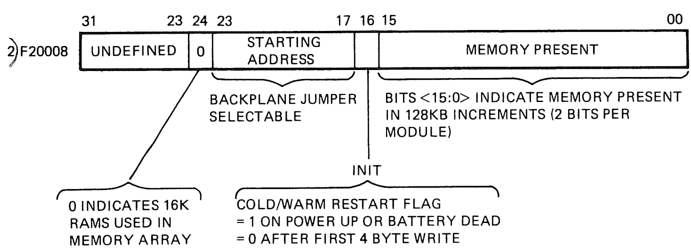
TK-4113

CONTROL/STATUS REGISTER 1 (CSR 1)



TK-4114

CONTROL/STATUS REGISTER 2 (CSR 2)



TK-4115

MIXED BOARDS

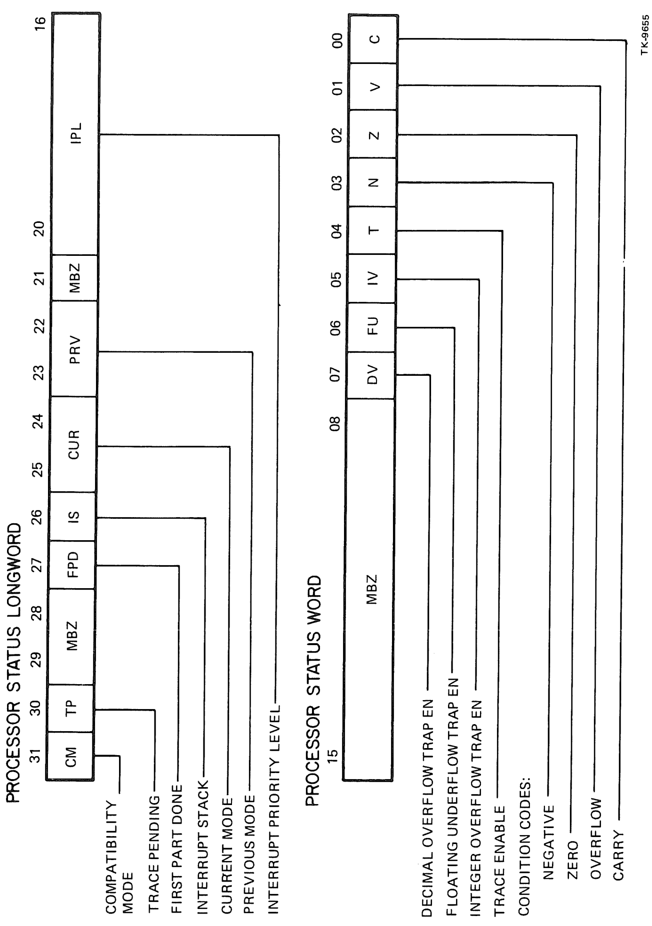
BIT 24=1

256 1M 1M

11 10 10

indicates a 1MByte Board

CPU (KA750) PROCESSOR STATUS LONGWORD (PSL)



INTERNAL PROCESSOR REGISTER (IPR) SUMMARY

Address	Mnemonic	Type*	Name
ØØ	KSP		Kernel Stack Pointer
Ø1	ESP		Executive Stack Pointer
Ø2	SSP		Supervisor Stack Pointer
Ø3	USP		User Stack Pointer
Ø4	ISP		Interrupt Stack Pointer
Ø5-Ø7	Reserved		
Ø8	PØBR		PØ Base Register
Ø9	PØLR		PØ Length Register
ØA	P1BR		P1 Base Register
ØB	P1LR		P1 Length Register
ØC	SBR		System Base Register
ØD	SLR		System Length Register
ØE-ØF	Reserved		
1Ø	PCBB		Process Control Block Base
11	SCBB		System Control Block Base
12	IPL		Interrupt Priority Level
13	ASTR		AST Level Register
14	SIRR	W/O	Software Interrupt Request Register
15	SISR		Software Interrupt Summary Register
16	Reserved		
17	CMIERR	R/O	CMI Error Register
18	ICCS		Interval Clock Control/Status
19	NICR	W/O	Next Interval Count Register
1A	ICR	R/O	Interval Count Register
1B	TODR		Time of Day Register
1C	CSRS		Console Storage Receiver Status
1D	CSRD	R/O	Console Storage Receiver Data
1E	CSTS		Console Storage Transmit Status
1F	CSTD	W/O	Console Storage Transmit Data

*Registers are read/write unless otherwise specified; R/O means read-only; W/O means write-only to perform the specified function(s).

INTERNAL PROCESSOR REGISTER (IPR) SUMMARY (CONT)

Address	Mnemonic	Type*	Name
20	RXCS		Console Receive Control/Status
21	RXDB	R/O	Console Receive Data Buffer
22	TXCS		Console Transmit Control/Status
23	TXDB	W/O	Console Transmit Data Buffer
24	TBDR		Translation Buffer Disable Register
25	CADR		Cache Disable Register
26	MCESR		Machine Check Error Summary Register
27	CAER		Cache Error Register
28	ACCS	R/O	Accelerator Control/Status
29-36	Reserved		
37	IO RESET	W/O	Initialize UNIBUS
38	MME		Memory Management Enable
39	TBIA	W/O	Translation Buffer Invalidate All
3A	TBIS	W/O	Translation Buffer Invalidate Single
3B	TB DATA		Translation Buffer Data
3C	Reserved		
3D	PMR		Performance Monitor Register
3E	SID	R/O	System Identification
3F	TBHP	W/O	Probe Translation Buffer for TB Hit

*Registers are read/write unless otherwise specified; R/O means read-only; W/O means write-only to perform the specified function(s).

INTERNAL PROCESSOR REGISTERS

HEX	NAME	
IPR #00	KSP	**KERNEL STACK POINTER**
IPR #01	ESP	**EXECUTIVE STACK POINTER**
IPR #02	SSP	**SUPERVISOR STACK POINTER**
IPR #03	USP	**USER STACK POINTER**
IPR #04	ISP	**INTERRUPT STACK POINTER**

31 — 00
VIRTUAL ADDRESS OF TOP OF STACK

IPR #08 POBR **PO BASE REGISTER**

RESERVED OPERAND FAULT IF VLA < 2**31

IPR #0A P1BR **P1 BASE REGISTER**

RESERVED OPERAND FAULT IF VLA < 2**31 - 2**21

31 — 02	01 00
VIRTUAL LONGWORD ADDRESS	MBZ

IPR #09 POLR **PO LENGTH REGISTER**

LENGTH OF POPT IN LONGWORDS

IPR #0B P1LR **P1 LENGTH REGISTER**

2**21 - LENGTH OF P1PT IN LONGWORDS

IPR #0D SLR **SYSTEM LENGTH REGISTER**

LENGTH OF SPT IN LONGWORDS
RESERVED OPERAND FAULT IF MBZ ≠0

31 — 22	21 — 00
MBZ	LENGTH IN LONGWORDS

TK-1750

INTERNAL PROCESSOR REGISTERS (CONT)

HEX NAME

IPR #10 PCBB

PROCESS CONTROL BLOCK BASE

RESERVED OPERAND FAULT IF MBZ $\neq$ 0.

31 30	29 02	01 00
MBZ	PHYSICAL LONGWORD ADDRESS OF PCB	MBZ

IPR #11 SCBB

SYSTEM CONTROL BLOCK BASE (See page 19 of pocket Service guide

RESERVED OPERAND FAULT IF MBZ $\neq$ 0.

31 30	29 02	01 00
MBZ	PHYSICAL PAGE ADDRESS OF SCB	MBZ

IPR #12 IPLR

INTERRUPT PRIORITY LEVEL REGISTER

31 05	04 00
MBZ	PSL<20:16>

IPR #13 ASTR

AST LEVEL REGISTER

RESERVED OPERAND FAULT IF NOT VALID I.E., MBZ $\neq$ 0.

31 03	02 00
MBZ	ASTLVL

IPR #OC SBR

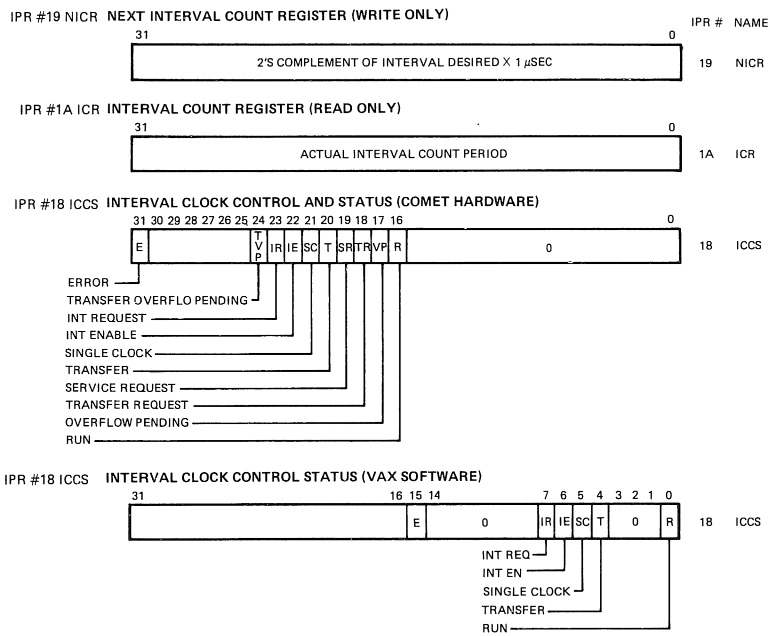
SYSTEM BASE REGISTER

RESERVED OPERAND FAULT IF MBZ $\neq$ 0.

31 30	29 02	01 00
MBZ	PHYSICAL LONGWORD ADDRESS	MBZ

TK-1753

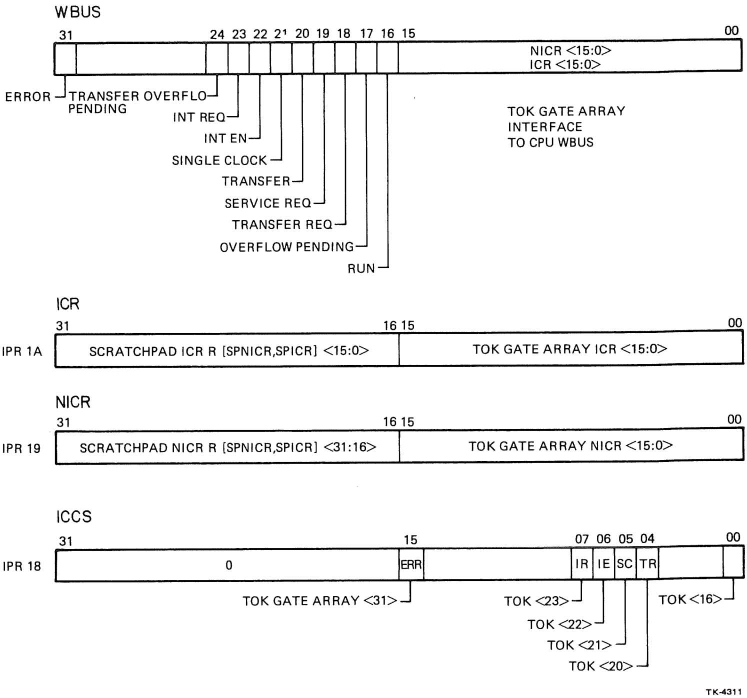
INTERNAL PROCESSOR REGISTERS (CONT)



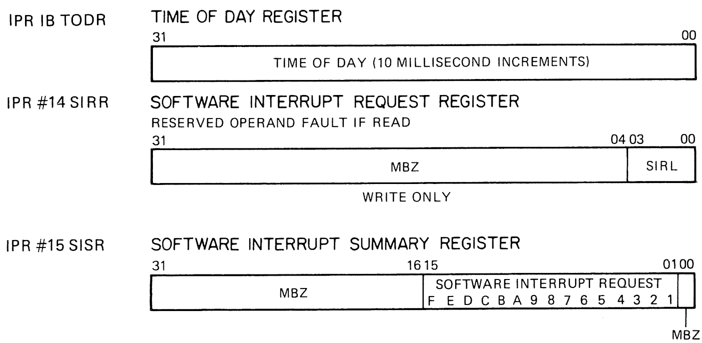
INTERVAL TIMER PROCESSOR REGISTERS

TK-5929

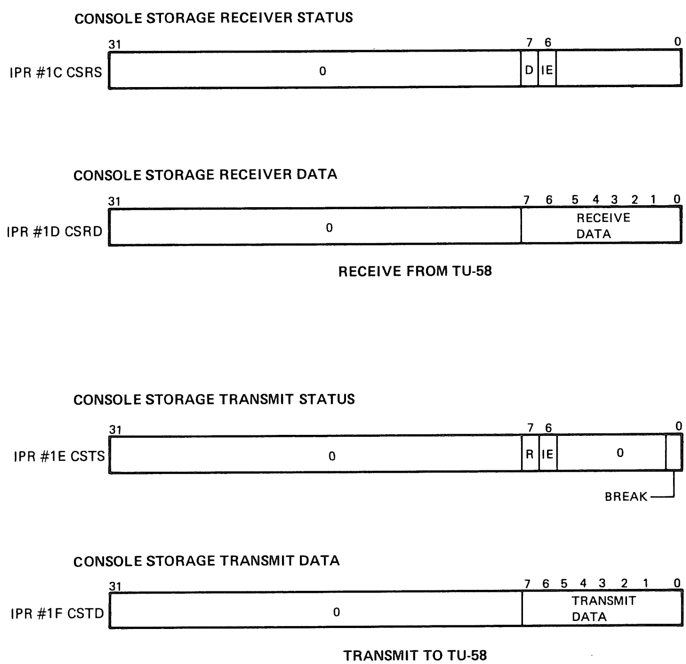
INTERVAL REGISTER HARDWARE USAGE



INTERNAL PROCESSOR REGISTERS (CONT)



TK-1752

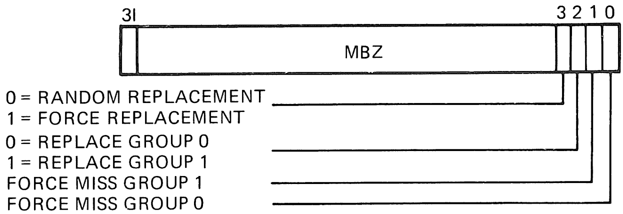


TK-1733

TU-58

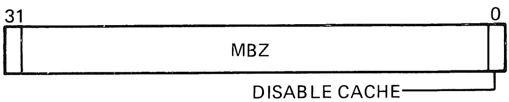
IPR #24 TBGDR — TRANSLATION BUFFER GROUP DISABLE REGISTER — IPR #24

THIS IPR IS READ/WRITE TO ALL BITS



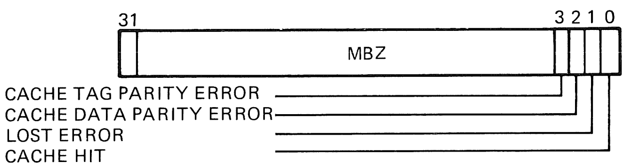
IPR #25 CADR — CACHE DISABLE REGISTER — IPR #25

THIS IPR IS READ/WRITE



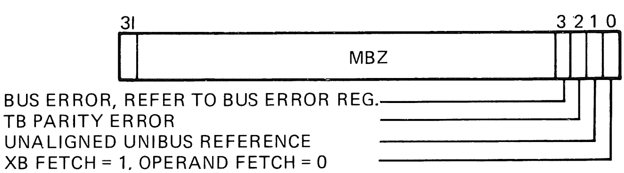
IPR #27 CAER — CACHE ERROR REGISTER — IPR #27

THIS IPR IS READ/WRITE



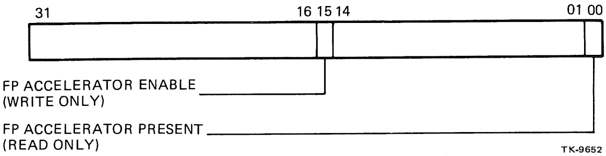
IPR #26 MCESR — MACHINE CHECK ERROR SUMMARY REGISTER — IPR #26

THIS IPR IS READ/WRITE TO ALL BITS. WRITING, A 1 TO BIT 3 CLEARS THE BUS ERROR REGISTER. WRITING A 1 TO BIT 2 CLEARS THE TB GROUP PARITY REGISTER.



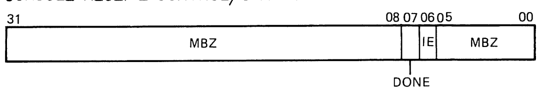
TK-5765

IPR NO. 28 ACCS ACCELERATOR CONTROL/STATUS

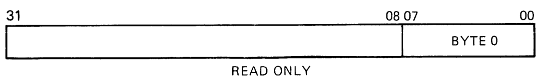


HEX NAME

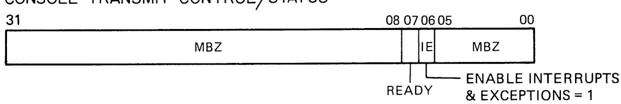
IPR #20 RXCS CONSOLE RECEIVE CONTROL/STATUS



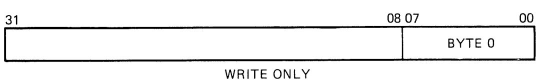
IPR #21 RXDB CONSOLE RECEIVE DATA BUFFER



IPR #22 TXCS CONSOLE TRANSMIT CONTROL/STATUS



IPR #23 TXDB CONSOLE TRANSMIT DATA BUFFER



TK-1749

CONSOLE TERMINAL

INTERNAL PROCESSOR REGISTERS (CONT)

HEX NAME ID#

IPR #38 MME

MEMORY MANAGEMENT ENABLE

WRITE 1 ALSO CAUSES MICROCODE TO INVALIDATE TB.

31 … 01	00
	MME

IPR #39 TBIA

TRANSLATION BUFFER INVALIDATE ALL

RESERVED OPERAND FAULT IF READ

31 … 00
MBZ

WRITE ONLY

IPR #3A TBIS

TRANSLATION BUFFER INVALIDATE SINGLE

RESERVED OPERAND FAULT IF READ

31 … 00
VIRTUAL ADDRESS

WRITE ONLY

IPR #3D PMR

PERFORMANCE MONITOR REGISTER

RESERVED OPERAND FAULT IF >1

31 … 01	00
MBZ	PME

IPR #3E SID

SYSTEM IDENTIFICATION (READ ONLY)

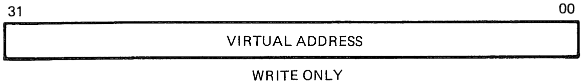
RESERVED OPERAND FAULT IF WRITE

31 … 24	23 … 16	15 … 8	7 … 0
SYSTEM TYPE	0	MICROCODE REVISION LEVEL	HARDWARE REVISION LEVEL
0 UNDEFINED 1 11/780 2 11/750 3 NEBULA 4 - 255 RESERVED		FROM MICRO. WORD LITERAL FIELD	BACKPLANE JUMPERS SLOT 4 OR BACKPLANE SWITCH CARD

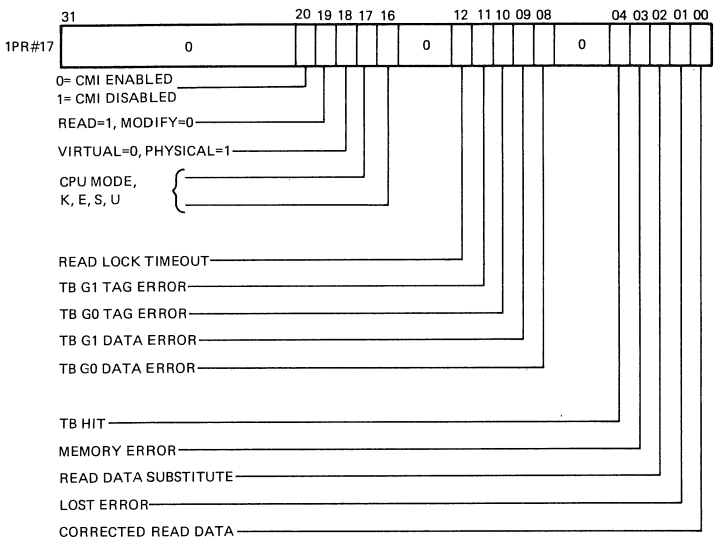
TK-2099

INTERNAL PROCESSOR REGISTERS (CONT)

IPR NO. 3F TBHP PROBE TRANSLATION BUFFER FOR TB HIT



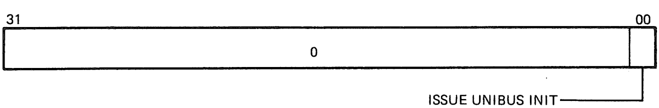
TK-9653



CMI ERROR PROCESSOR REGISTER

TK-3266

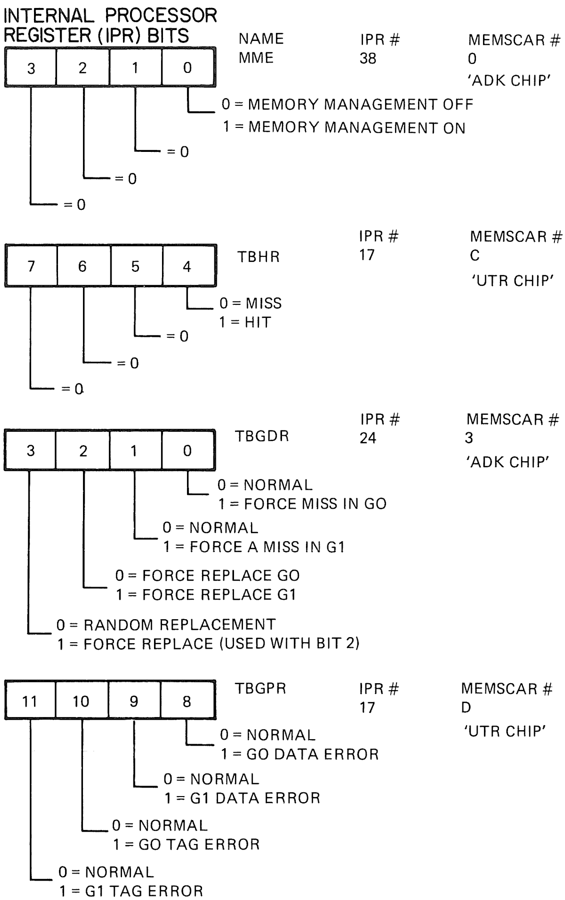
IPR #37 IO RESET INITIALIZE UNIBUS



IO RESET PROCESSOR REGISTER

TK-3267

TB REGISTER BIT FIELDS



TK5769

CACHE REGISTER BIT FIELDS

INTERNAL PROCESSOR REGISTER (IPR) BITS

NAME	IPR #	MEMSCAR #
CAER	27	4

'CAK CHIP'

Bit	Meaning
0	0 = MISS 1 = HIT
1	LOST ERROR
2	0 = NORMAL 1 = DATA ERROR
3	0 = NORMAL 1 = TAG ERROR

CADR

IPR #	MEMSCAR #
25	6

'CAK CHIP'

Bit	Meaning
0	0 = CACHE ON 1 = DISABLE CACHE (FORCE MISS)
1	UNDEFINED
2	UNDEFINED
3	UNDEFINED

CACHE WRITE ONLY REGISTER

IPR #	MEMSCAR #
17	E

'UTR CHIP'

Bit	Meaning
20	0 = CMI ON 1 = DISABLE CMI
21	= 0
22	= 0
23	= 0

TK-5802

TB AND CACHE CONTROL/STATUS REGISTER BIT FIELDS

INTERNAL PROCESSOR REGISTER (IPR) BITS

3	2	1	0

NAME MCESR — IPR # 26 — MEMSCAR # 8 'UTR CHIP'

- Bit 0: 0 = OPERAND FETCH; 1 = XB FETCH
- Bit 1: 0 = NORMAL; 1 = UNALIGNED UNIBUS REFERENCE
- Bit 2: 0 = NORMAL; 1 = TB ERROR (WRITING A ONE CLEARS TBGPR)
- Bit 3: 0 = NORMAL; 1 = BUS ERROR (WRITING A ONE CLEARS BER)

3	2	1	0

BUS ERROR SUMMARY REGISTER — IPR # 17 — MEMSCAR # 9 'UTR CHIP'

- Bit 0: 0 = NORMAL; 1 = CORRECTED READ DATA
- Bit 1: 0 = NORMAL; 1 = LOST ERROR
- Bit 2: 0 = NORMAL; 1=UNCORRECTECTABLE DATA ERROR
- Bit 3: 0 = NORMAL; 1 = NONEXISTENT MEMORY

19	18	17	16

SAVED MODE REGISTER — IPR # 17 — MEMSCAR # 1 'ADK CHIP'

- Bit 16: = MODE <0>
- Bit 17: = MODE <1>
- Bit 18: 0 = VIRTUAL; 1 = PHYSICAL
- Bit 19: 0 = READ – MODIFY; 1 = NORMAL READ

15	14	13	12*

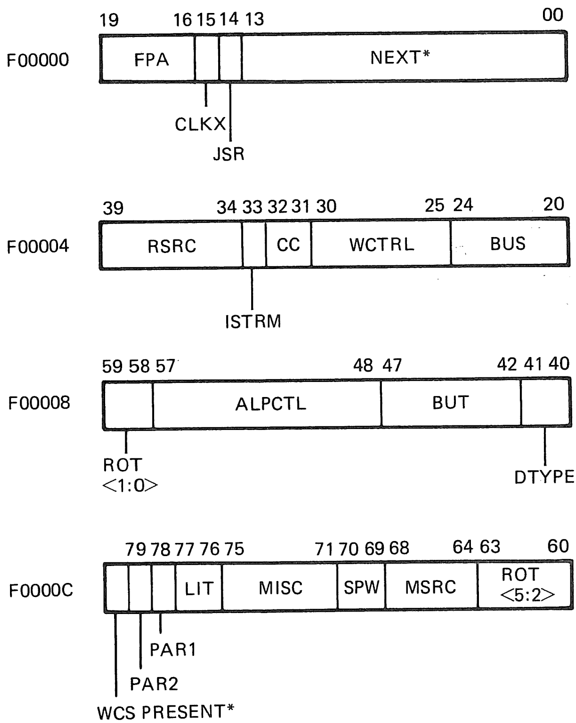
WRITE VECTOR OCCURRED REGISTER — IPR # 17 — MEMSCAR # 2 'ADK CHIP'

- Bit 12: 0 = NORMAL; 1 = VECTOR IN MDR
- Bit 13: = 0
- Bit 14: = 0
- Bit 15: = 0

* ALSO READS AS THE READ LOCK TIMEOUT BIT

TK5770

WCS (KU750) REGISTER DATA WRITE FORMAT



*NOTE: WCS ADDRESS SPACE IS F00000 THROUGH F0FFFC.

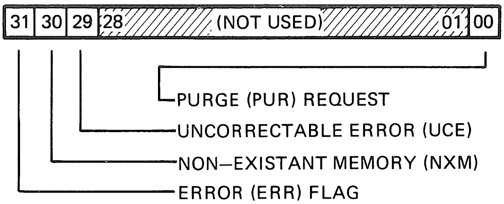
BIT <20> IS WRITTEN AS 1 ON THE LAST WORD WRITTEN TO THE WCS. WCS PRESENT SET ENABLES THE WCS MICROCODE TO BE EXECUTED.

TK-9658

UBI AND SUB REGISTERS

UBI OR SUB (DW750) CONTROL/STATUS REGISTER (CSR)

CMI DATA LONGWORD

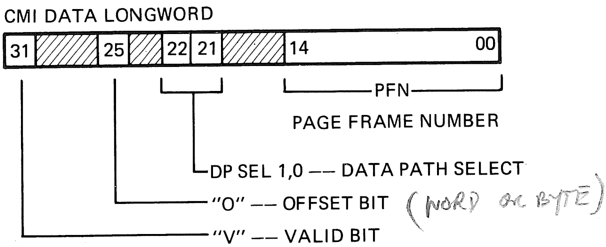


TK-3886

CSR Addresses

Control/Status Register	UBI Address	SUB Address	Buffered Data Path
CSR 1	F30004	F32004	BDP 1
CSR 2	F30008	F32008	BDP 2
CSR 3	F3000C	F3200C	BDP 3

UBI OR SUB MAP REGISTER DATA



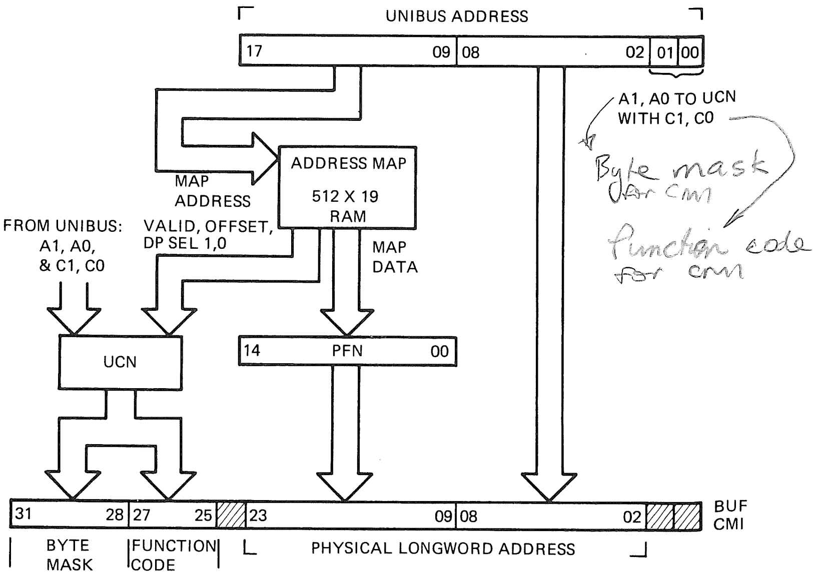
TK-3882

DP SEL Bits 1	Ø	Data Path
Ø	Ø	Direct Data Path (DDP)
Ø	1	Buffered Data Path 1 (BDP 1)
1	Ø	Buffered Data Path 2 (BDP 2)
1	1	Buffered Data Path 3 (BDP 3)

MAP Register Addresses

MAP Register	UBI Address	SUB Address
MAP ØØØ	F3Ø8ØØ	F328ØØ
MAP ØØ4	F3Ø8Ø4	F328Ø4
.	.	.
.	.	.
MAP 7F8	F3ØFF8	F32FF8
MAP 7FC	F3ØFFC	F32FFC

UNIBUS TO CMI MAP ADDRESS TRANSLATION



TK-3883

(See page 178 for Block Diagram)
(" " 210 for Register Summary)

UET REGISTERS

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

UNIBUS ADDRESS REGISTER

FFF462																

UNIBUS DATA REGISTER

UET CONTROL STATUS 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	C0	NPR

- INIT (15): ISSUE UNIBUS INIT
- BR7–BR4 (11–08): BUS REQUEST LEVEL SELECT
- PE (07): UNIBUS PARITY ERROR
- TO (06): UET SSYN TIME OUT
- PB (05): FORCE PB
- A17, A16 (04–03): ADDRESS EXTENSION
- C1, C0 (02–01): UNIBUS TRANSFER SELECT
- NPR (00): GO ISSUE NPR

FFF466	ZERO	ROM DATA

TK-5609

IPEC REGISTERS

UNIBUS Address and Data Registers

	15 … 00
FBF460	UNIBUS ADDRESS

<A15:A00>

	15 … 00
FBF462	UNIBUS DATA

<D15:D00>

TK-8001

Control Register 1 (CR1)

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FBF464	INIT1		ACLO1	ACIE					PE	SSYN TO	PB	A17	A16	C1	CO	NPR

Bit 14: INT DONE1

Bits 11–08: CR1<BR7:BR4>

TK-8002

Control Register 2 (CR2)

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FBF466	ENG OPT															

Bit 14: INT DONE2

Bit 13: ACLO2

Bits 12–09: CR2<BR7:BR4>

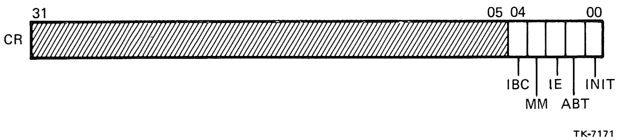
Bits 08–00: <V8:V0>

TK-8003

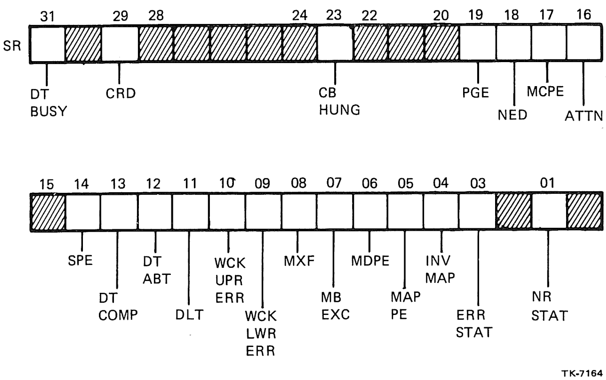
MBA (RH750) REGISTERS

MBA INTERNAL REGISTERS

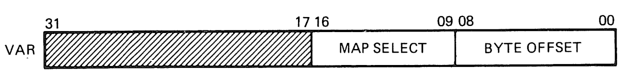
Control Register (CR) – F28004



Status Register (SR) – F28008



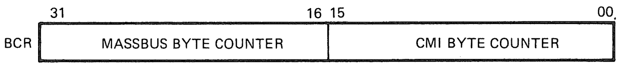
Virtual Address Register (VAR) – F2800C



MBA (RH750) REGISTERS (CONT)

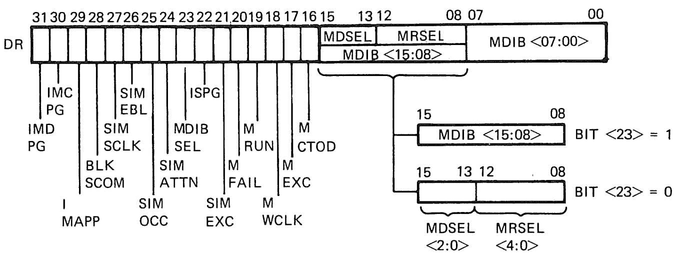
MBA INTERNAL REGISTERS (CONT)

Byte Count Register (BCR) – F28010



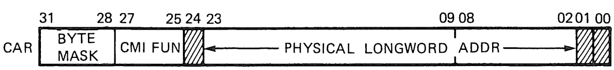
TK-7166

Diagnostic Register (DR) – F28014



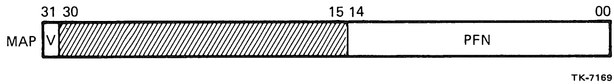
TK-7168

Command Address Register (CAR) – F2801C

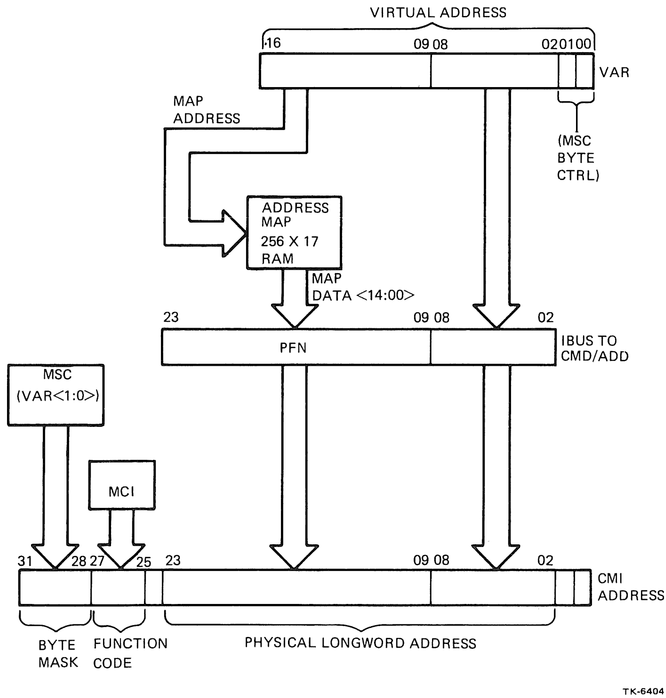


TK-7170

MBA MAP REGISTER DATA – F28800 - F28BFC



MBA TO CMI MAP ADDRESS TRANSLATION



MASSBUS EXTERNAL DEVICE REGISTERS

Drive Control Register (R0) – F28400

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
										F4	F3	F2	F1	F0	GO

TK-7165

Attention Summary Register (R4) – F28410

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
								ATA 7	ATA 6	ATA 5	ATA 4	ATA 3	ATA 2	ATA 1	ATA 0

TK-7167

DZ11 REGISTERS

Register	15 (MSB)	14	13	12	11	10	09	08 (HIGH BYTE)	07 (LOW BYTE)	06	05	04	03	02	01	00 (LSB)	UNIBUS ADDRESS (OCTAL)	SYSTEM PHYSICAL BYTE ADDRESS FOR UBI (HEX)
CONTROL & STATUS (CSR)	RO TX RDY	RW TX INTR ENAB	RO SILO ALARM	RW SILO ALARM ENAB	NOT USED	RO TX LINE C	RO TX LINE B	RO TX LINE A	RO RX DONE	RW RX INTR ENAB	RW MAST SCAN ENAB	RW CLEAR	RW MAINT	NOT USED	NOT USED	NOT USED	760100**	FFE040
RECEIVER BUFFER (RBUF)	RO DATA VALID	RO OVRN	RO FRAM ERR	RO PAR ERR	NOT USED	RO RX LINE C	RO RX LINE B	RO RX LINE A	RO RBUF D7	RO RBUF D6	RO RBUF D5	RO RBUF D4	RO RBUF D3	RO RBUF D2	RO RBUF D1	RO RBUF D0	760102**	FFE042
LINE PARAMETER (LPR)	NOT USED	NOT USED	NOT USED	WO RX ON	WO FREQ D	WO FREQ C	WO FREQ B	WO FREQ A	WO ODD PAR	WO PAR ENAB	WO STOP CODE	WO CHAR LGTH B	WO CHAR LGTH A	WO LINE C	WO LINE B	WO LINE A	760102**	FFE042
TRANSMIT* CONTROL (TCR)	RO DATA TERM RDY 7	RO DATA TERM RDY 6	RO DATA TERM RDY 5	RO DATA TERM RDY 4	RO DATA TERM RDY 3	RO DATA TERM RDY 2	RO DATA TERM RDY 1	RO DATA TERM RDY 0	RO LINE ENAB 7	RO LINE ENAB 6	RO LINE ENAB 5	RO LINE ENAB 4	RO LINE ENAB 3	RO LINE ENAB 2	RO LINE ENAB 1	RO LINE ENAB 0	760104**	FFE044
MODEM* STATUS (MSR)	RO CO 7	RO CO 6	RO CO 5	RO CO 4	RO CO 3	RO CO 2	RO CO 1	RO CO 0	RO RING IND 7	RO RING IND 6	RO RING IND 5	RO RING IND 4	RO RING IND 3	RO RING IND 2	RO RING IND 1	RO RING IND 0	760106**	FFE046
TRANSMIT DATA (TDR)	WO BRK 7	WO BRK 6	WO BRK 5	WO BRK 4	WO BRK 3	WO BRK 2	WO BRK 1	WO BRK 0	WO TBUF 7	WO TBUF 6	WO TBUF 5	WO TBUF 4	WO TBUF 3	WO TBUF 2	WO TBUF 1	WO TBUF 0	760106**	FFE046

* THE HIGH BYTE OF THE TCR (DATA TERMINAL READY) AND THE MSR ARE NOT USED WITH THE 20MA OPTIONS.

** THESE ADDRESSES REFER TO THE FIRST DZ11 ON UBI ONLY FIRST SUBADDRESS IS FBE040 FOR DZ11, FBFF20 FOR RK611

TK-9662

RK611 REGISTERS

UNIBUS ADDRESS (OCTAL) — SYSTEM PHYSICAL BYTE ADDRESS FOR UBI (HEX)

CONTROL AND STATUS REGISTER 1
RKCS1 — READ/WRITE

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
	DI	DCT PAR		CTO	CDT			RDY	IE	0	F4	F3	F2	F1	GO	777440	FFFF20
CERR OCLR			CFMT			BA17	BA16										

WORD COUNT REGISTER
RKWC — R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
WC 15	WC 14	WC 13	WC 12	WC 11	WC 10	WC 09	WC 08	WC 07	WC 06	WC 05	WC 04	WC 03	WC 02	WC 01	WC 00	777442	FFFF22

BUS ADDRESS REGISTER
RKBA — R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
BA 15	BA 14	BA 13	BA 12	BA 11	BA 10	BA 09	BA 08	BA 07	BA 06	BA 05	BA 04	BA 03	BA 02	BA 01	BA 00	777444	FFFF24

DISK ADDRESS (TRACK & SECTOR) REG
RKDA — R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
0	0	0	0	0	TA 2	TA 1	TA 0	0	0	0	SA 4	SA 3	SA 2	SA 1	SA 0	777446	FFFF26

CONTROL AND STATUS REGISTER 2
RKCS2 — R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
DLT	WCE	UPE	NED	NEM	PGE	MDS	UFE	OR	IR		BAI	RLS	DS 2	DS 1	DS 0	777450	FFFF28
										SCLR							

DRIVE STATUS REGISTER
RKDS — READ ONLY

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
	SDA	PIP	0	WRL	0	0	DDT		VV					0	DRA	777452	FFFF2A
SVAL								DRDY		DROT	SPLS	ACLO	OFST				

ERROR REGISTER
RKER — RO

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
DCK	UNS	OPI	DTE	WLE		COE		BSE	ECH				NXF	SKI	ILF	777454	FFFF2C
					IDAE		HVRC			DTYE	FMTE	DRPAR					

ATTENTION SUMMARY AND OFFSET
RKAS/OF — R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS	SYSTEM ADDRESS
ATN 7	ATN 6	ATN 5	ATN 4	ATN 3	ATN 2	ATN 1	ATN 0	OF 7	OF 6	OF 5	OF 4	OF 3	OF 2	OF 1	OF 0	777456	FFFF2E

TK-9694

RK611 REGISTERS (CONT)

SYSTEM PHYSICAL BYTE ADDRESS FOR UBI (HEX)

UNIBUS ADDRESS (OCTAL)

DESIRED CYLINDER REGISTER
RKDC

R/W

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	UNIBUS ADDRESS (OCTAL)	SYSTEM PHYSICAL BYTE ADDRESS FOR UBI (HEX)
0	0	0	0	0	0	DC	DC	DC	DC	DC	DC	DC	DC	DC	DC	777460	FFFF30

UNUSED

																777462	FFFF32

DATA BUFFER

R/W

15							08	07							00		
DB 15	DB 14	DB 13	DB 12	DB 11	DB 10	DB 09	DB 08	DB 07	DB 06	DB 05	DB 04	DB 03	DB 02	DB 01	DB 00	777464	FFFF34

MAINTENANCE REGISTER 1
RKMR1

R/W

15							08	07							00		
RD GATE	WRT GATE	ECCW	PCD	PCA	MEWD	MERD	MCLK	MIND	MSP	DMD	PAT	MS 3	MS 2	MS 1	MS 0	777466	FFFF36

ECC POSITION REGISTER
RKECPS

RO

15							08	07							00		
0	0	0	EPS 12	EPS 11	EPS 10	EPS 09	EPS 08	EPS 07	EPS 06	EPS 05	EPS 04	EPS 03	EPS 02	EPS 01	EPS 00	777470	FFFF38

ECC PATTERN REGISTER
RKECPT

RO

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
0	0	0	0	0	EPT 10	EPT 09	EPT 08	EPT 07	EPT 06	EPT 05	EPT 04	EPT 03	EPT 02	EPT 01	EPT 00	777472	FFFF3A

MAINTENANCE REGISTER 2
RKMR2

RO

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
																777474	FFFF3C

MAINTENANCE REGISTER 3
RKMR3

RO

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
																777476	FFFF3E

TK-9695

DR750 REGISTERS

DR750 COMMAND BLOCK

	BASE DEC	BASE HEX
INPUT QUEUE FORWARD LINK (INPTQ HEAD)	0	0
INPUT QUEUE BACKWARD LINK (INPTQ TAIL)	4	4
TERMINATION QUEUE FORWARD LINK (TERMQ HEAD)	8	8
TERMINATION QUEUE BACKWARD LINK (TERMQ TAIL)	12	C
FREE QUEUE FORWARD LINK (FREEQ HEAD)	16	10
FREE QUEUE BACKWARD LINK (FREEQ TAIL)	20	14
COMMAND PACKET SPACE		

TK-9280

DR750 COMMAND PACKET

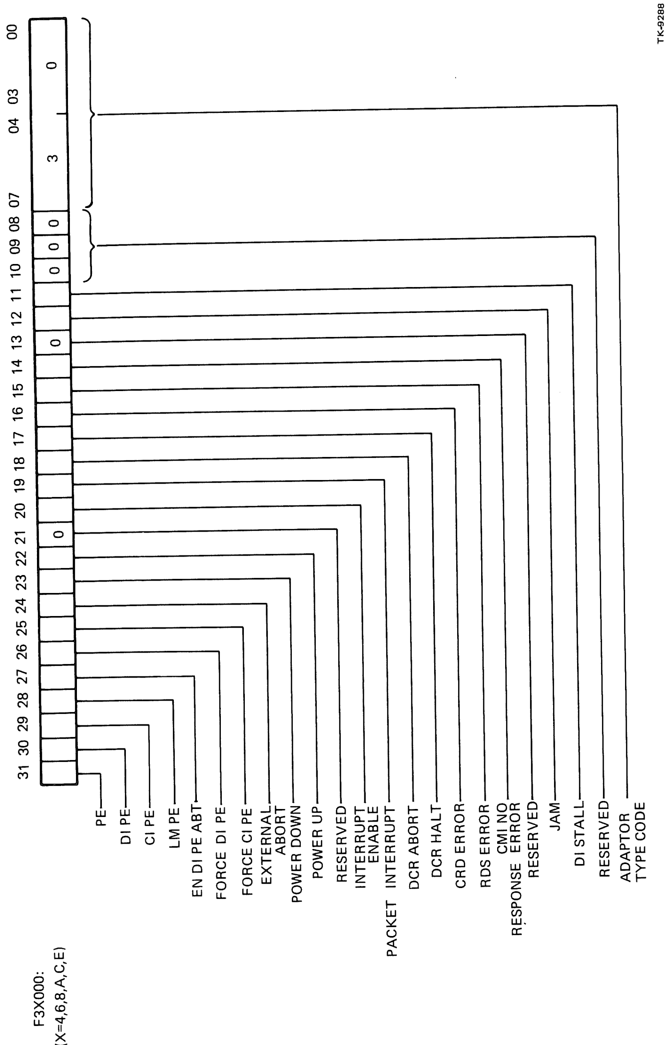
Bit positions: 31 29 24 23 19 16 15 08 07 03 02 00

	SELF RELATIVE FORWARD LINK	000
	SELF RELATIVE BACKWARD LINK	000
PACKET CONTROL LONGWORD	INTRPT CNTRL | SLE | CIS | NOT USED | 0000 | DEVICE CONTROL CODES | LAST REG IN LOG AREA | LENGTH OF DEVICE MESSAGE	
	BYTE COUNT	
	VIRTUAL ADDRESS OF BUFFER	
	RESIDUAL MEMORY BYTE COUNT	
	RESIDUAL DDI BYTE COUNT	
	DR32 STATUS LONGWORD (DSL)	
	DEVICE MESSAGE	
	LOG AREA	

TK-9279

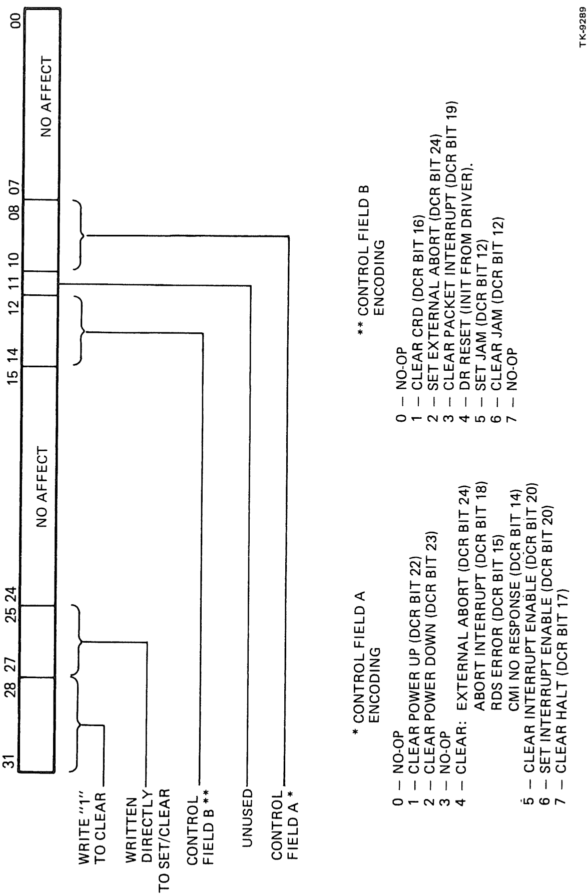
DR750 REGISTERS (CONT)

DR750 DCR REGISTER – READ FORMAT



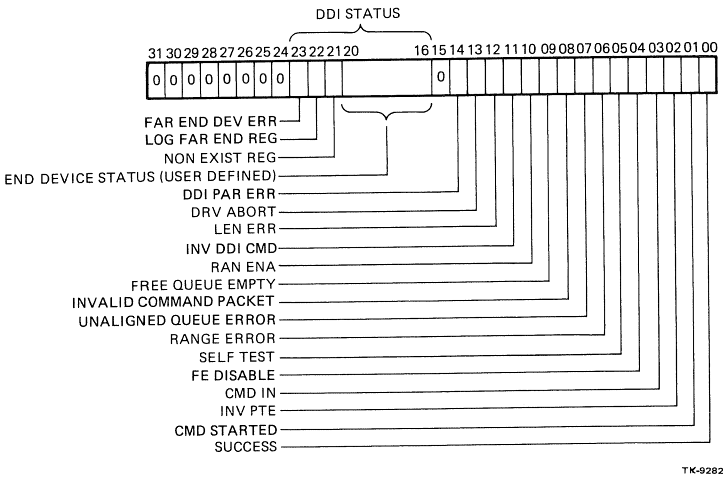
DR750 REGISTERS (CONT)

DR750 DCR REGISTER – WRITE FORMAT



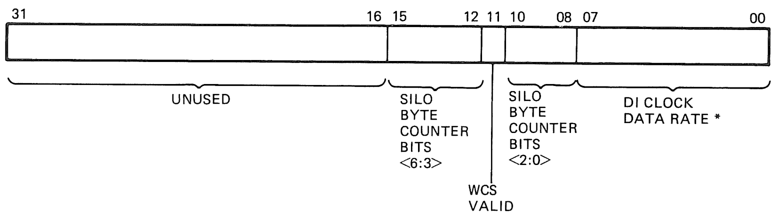
DR750 REGISTERS (CONT)

DR750 STATUS LONGWORD



DR750 REGISTERS (CONT)

DR750 UTILITY REGISTER



* NOTE: FF THRU FD
ARE NOT VALID

TK-9286

DI CLOCK DATA RATE SELECTION

Utility Register <7:Ø> Value	DI Data Rate (Mbyte/sec)
FF to FD*	--
FC	3.12
FB	2.5Ø
FA	2.Ø8
F9	1.78
F8	1.56
F7	1.38
F6	1.25
F5	1.14
F4	1.Ø4
F3	Ø.96
F2	Ø.89
F1	Ø.83
FØ	Ø.78
:	:
:	:
ØØ	Ø.Ø488

*Loading of these registers by the software must be prevented.

DR750 DEVICE VECTORS

Base Address	Slot Number	BR4	BR5	BR6*	BR7
F34000	10	128	168	1A8	1E8
F36000	11	12C	16C	1AC	1EC
F38000	12	130	170	1B0	1F0
F3A000	13	134	174	1B4	1F4
F3C000	14	138	178	1B8	1F8
F3E000	15	13C	17C	1BC	1FC

*Standard interrupt level is BR6.

CHAPTER 3
MODULES AND GATE ARRAYS

VAX-11/750 MODULE UTILIZATION

MICROCODE ADDRESS RANGE
0000 - 17FF

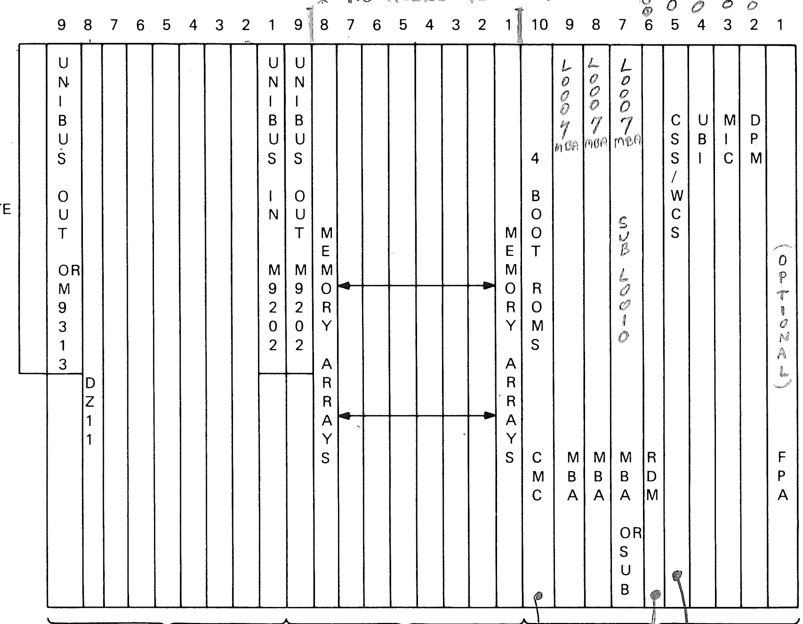
* 1 MByte Boards before 1/4 MByte boards.
* NO HOLES ALLOWED

- SYSTEM TIMING (M-CLOCK) 320nS

DPM - DATA PATH MODULE
- INSTRUCTION EXECUTION HAPPENS HERE ALSO DATA ALIGNMENT / SHIFTING
- G.P.R.'s HERE EXCEPT PROG CNTER.

MIC - CONTAINS P/COUNTER
- MEM. MGNT
- PAGE TABLE ENTRIES
- CACHE (1K LONGWORD)

CCS (PCS) 6K x 80 BITS
- CONTAINS WCS



BACKPLANE CARD LOCATION
FRONT VIEW

TK-9656

SUB = SECOND UNIBUS

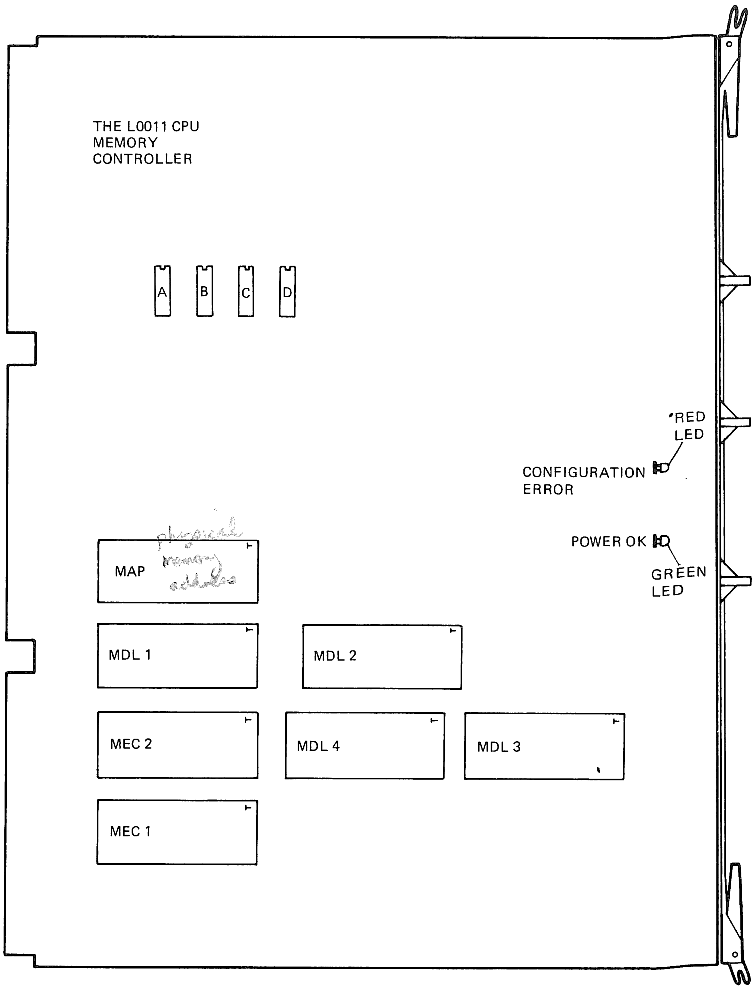
UPDATED TO L0008.

L0011 - 256KB.
L0016 - 1MByte.

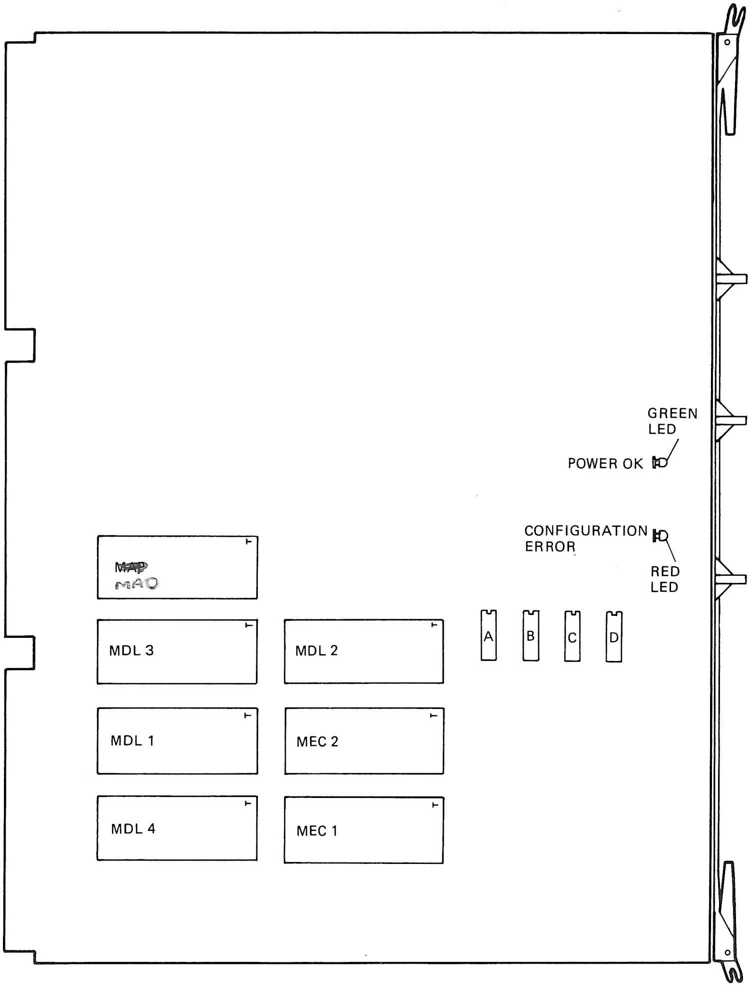
INVOKE-USE WITH ^D. (DM>)

CABLES TO TU-58's CONSOLE MUST BE MOVED WHEN MDM MODULE INSTALLED

CMC GATE ARRAY LAYOUT (L0011)



CMC GATE ARRAY LAYOUT (L0016)



TK-9660

Block diagram p 156

CMC GATE ARRAY DESCRIPTION

MDL - Four memory data loop (MDL) chips make up the data path between the CMI and memory, the CMI and the control/status registers, and from the bootstrap ROMs to memory.

MEC - Two memory error correction (MEC) chips detect double bit errors, and detect and correct single bit errors.

MAP/MAD - One memory address processor (MAP) chip on the L0011 and one memory address decoder (MAD) chip on the L0016. Address bits are decoded to enable memory arrays and determine starting address offsets. Address validity checks are performed and memory array board population is detected.

BOOTSTRAP DEVICE ROMS

ROM Part Number	Device Code	Device Type	Controller
23-908A9-00	RH	RM80	RH750
23-990A9-00		RA80	UDA50
23-294E2-02	DD	TU58 - MRSP*	

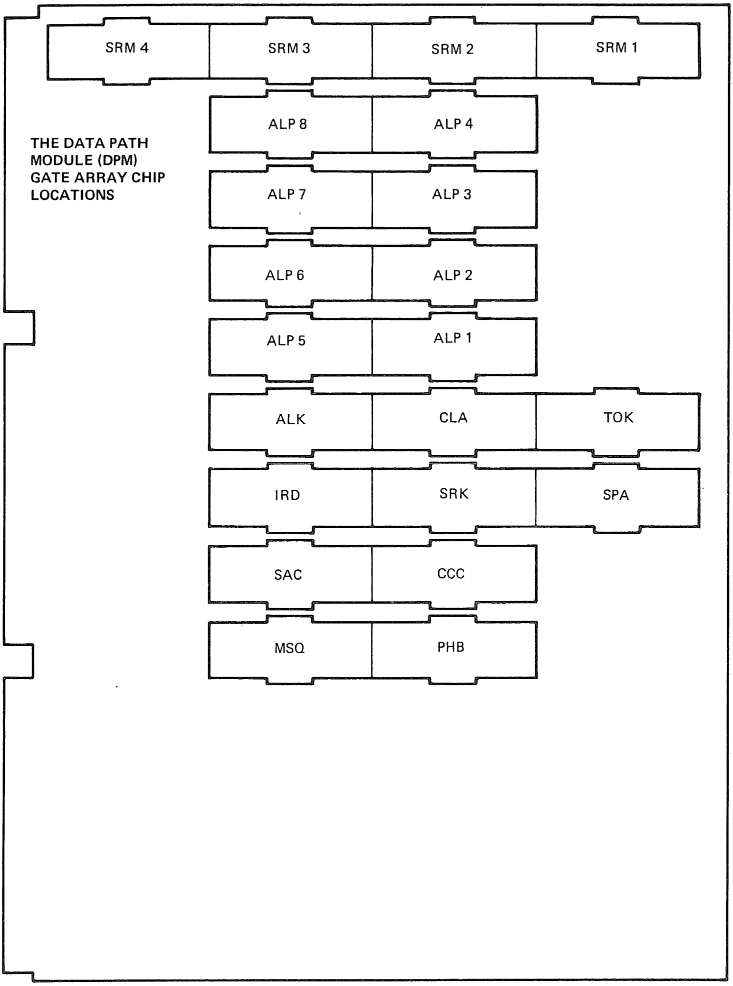
*Modified radial serial protocol (MRSP) is required for UNIX.

ROM Starting Microaddresses

Device ROM	Starting Microaddress
A	FA02
B	FB02
C	FC02
D	FD02

CSR page 15

DPM GATE ARRAY LAYOUT (L0002)



TK-4711

page 159 for block diagram

DPM GATE ARRAY DESCRIPTION

SRM - Four super rotator multiplex (SRM) chips perform 64 functions under the control of the SRK.

SRK - One super rotator control (SRK) chip controls SRM operations.

ALP - Eight arithmetic logic processor (ALP) chips make up the arithmetic logic unit (ALU) of the DPM. The ALU performs most of the data manipulation during the execution of macroinstructions.

ALK - One arithmetic logic control (ALK) chip controls ALP operations by decoding microcode inputs and generating the necessary signals.

CLA - One carry lookahead (CLA) chip contains the logic that the ALU uses to generate and propagate carries.

TOK - One timed operation control (TOK) chip contains the programmable interval time clock.

IRD - One instruction register decode (IRD) chip processes the IR decodes. Receives opcode and operand specifier from the execution buffer decodes (XB), decodes it, and generates signals to select the correct microcode routine.

SPA - One scratchpad addressing (SPA) chip controls the 64 scratchpad register addressing. Contains logic that keeps track of general purpose register (GPR) autoincrements and autodecrements.

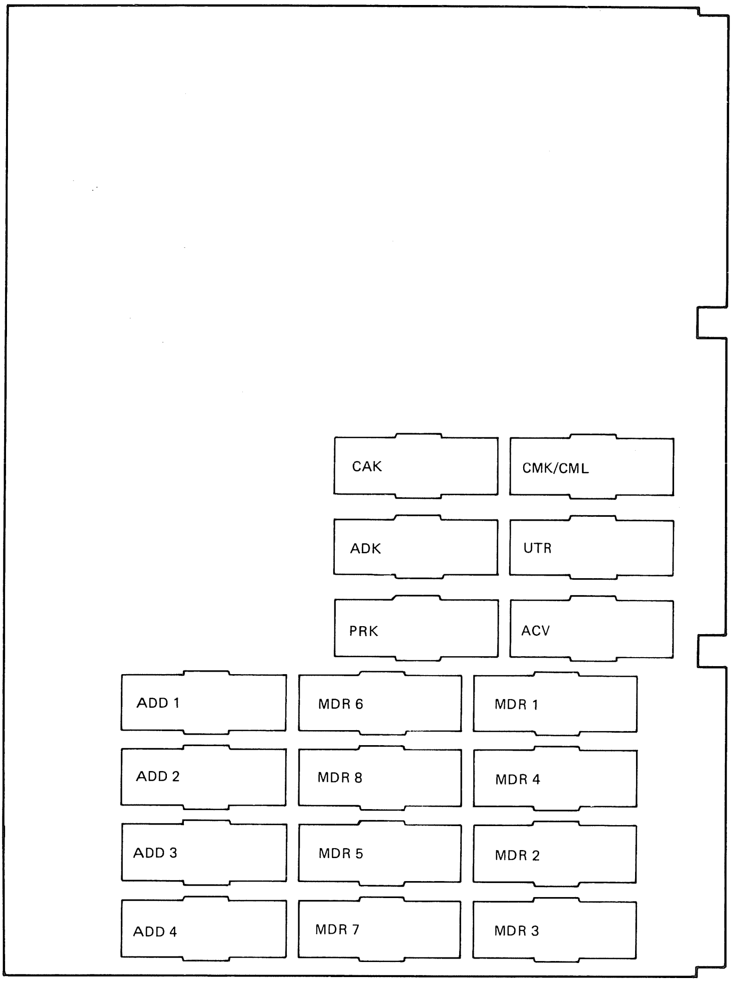
SAC - One service arbitration and clock (SAC) chip is associated with the IRD counter, service arbitration, and system clocks.

CCC - One condition code (CCC) chip is associated with the condition codes for both VAX-11 and compatability mode instructions. It stores PSL bits FU, IV, DV, N, Z, V, and C, and reads the bits at the request of the microcode.

MSQ - One microsequencer (MSQ) chip, together with the CCS module, forms the CPU microsequencer that sequences the CPU microcode.

PHB - One practically half the BUTs (PHB) chip contains PSL bits, status flags, the step counter, and logic that performs about half of the BUT micro-orders.

MIC GATE ARRAY LAYOUT (L0003)



page 164
for block diagram

MIC GATE ARRAY DESCRIPTION

MDR - Eight memory data register (MDR) chips contain logic that routes MIC data to or from the CMI or the M-bus, W-bus, or cache.

PRK - One prefetch control (PRK) chip initially fetches eight instruction (I-stream) bytes from memory starting with the PC address. It then replaces four bytes at a time during program execution.

ADK - One address control (ADK) chip controls the MIC addressing logic.

ADD - Four address (ADD) chips make up the PC and VA registers and their load paths.

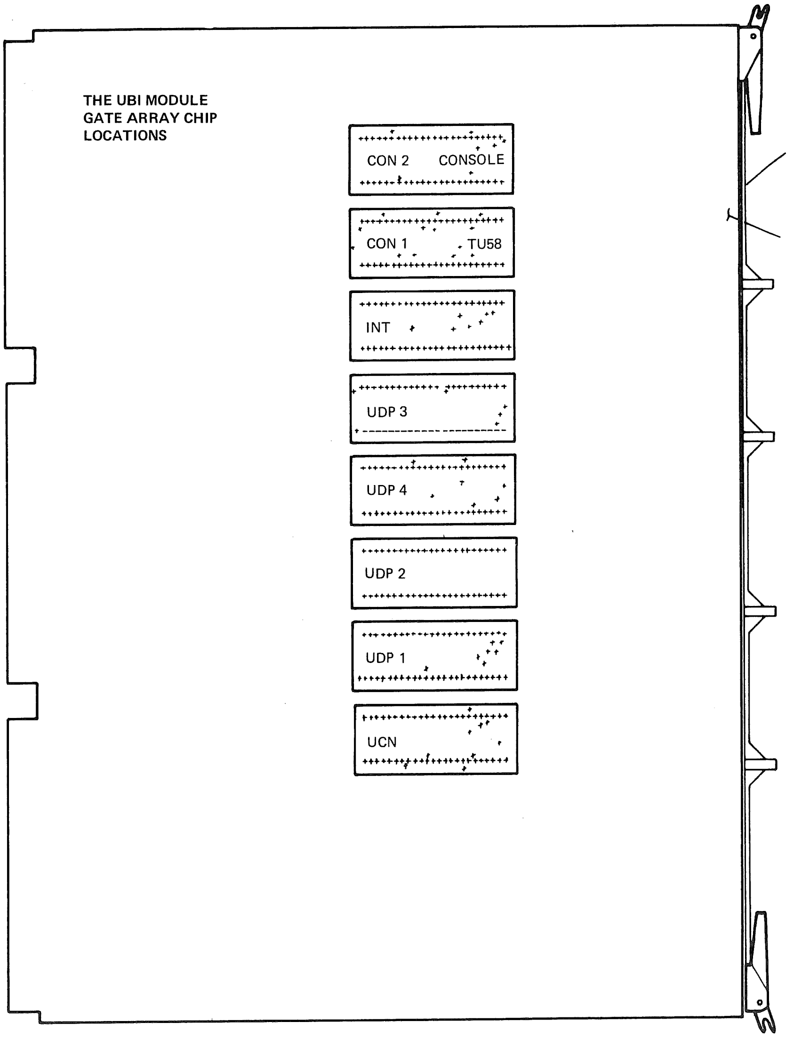
CMK - One CMI control (CMK) chip monitors and transmits CMI control signals. Stalls the microcode under certain conditions. A CML chip is installed in its place for the DR750.

CAK - One cache control (CAK) chip enables or disables cache. Controls the transfer of data to or from the MDR chips.

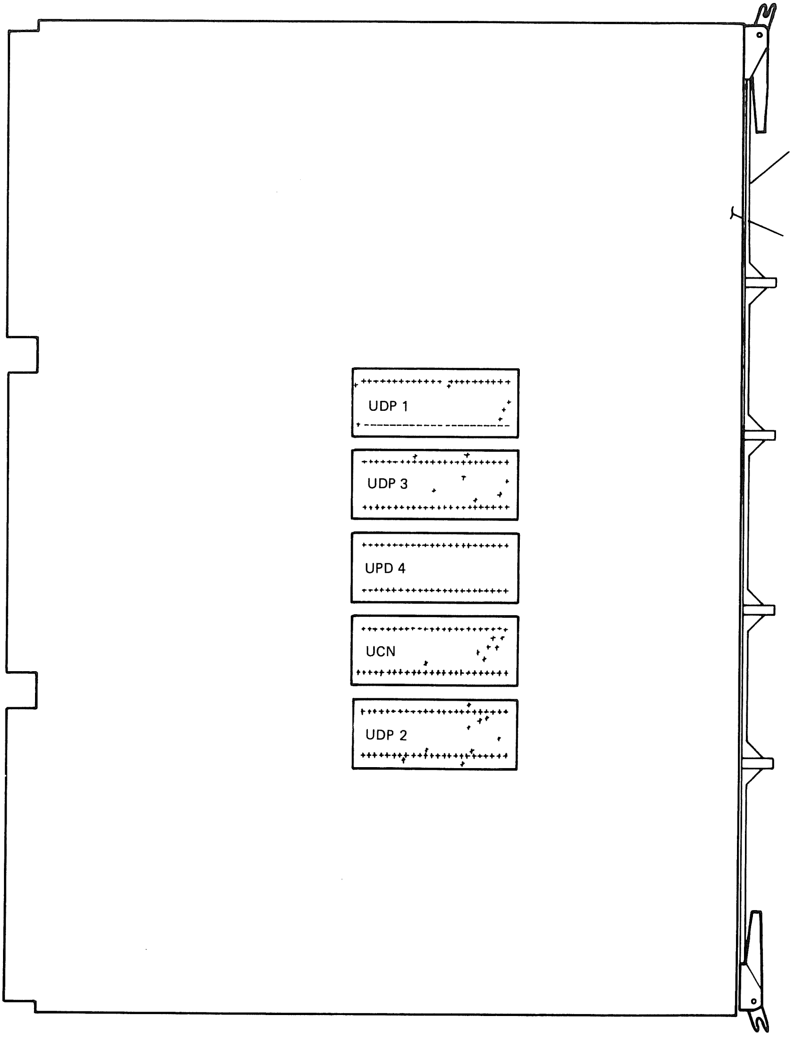
UTR - One microtrap (UTR) chip monitors machine conditions that cause a microtrap.

ACV - One access violation (ACV) chip detects access violations, control store parity errors, FPA reserved operands, unaligned data, and page boundary violations.

UBI GATE ARRAY LAYOUT (L0004)



SUB GATE ARRAY LAYOUT (L0010)



UBI AND SUB GATE ARRAY DESCRIPTION

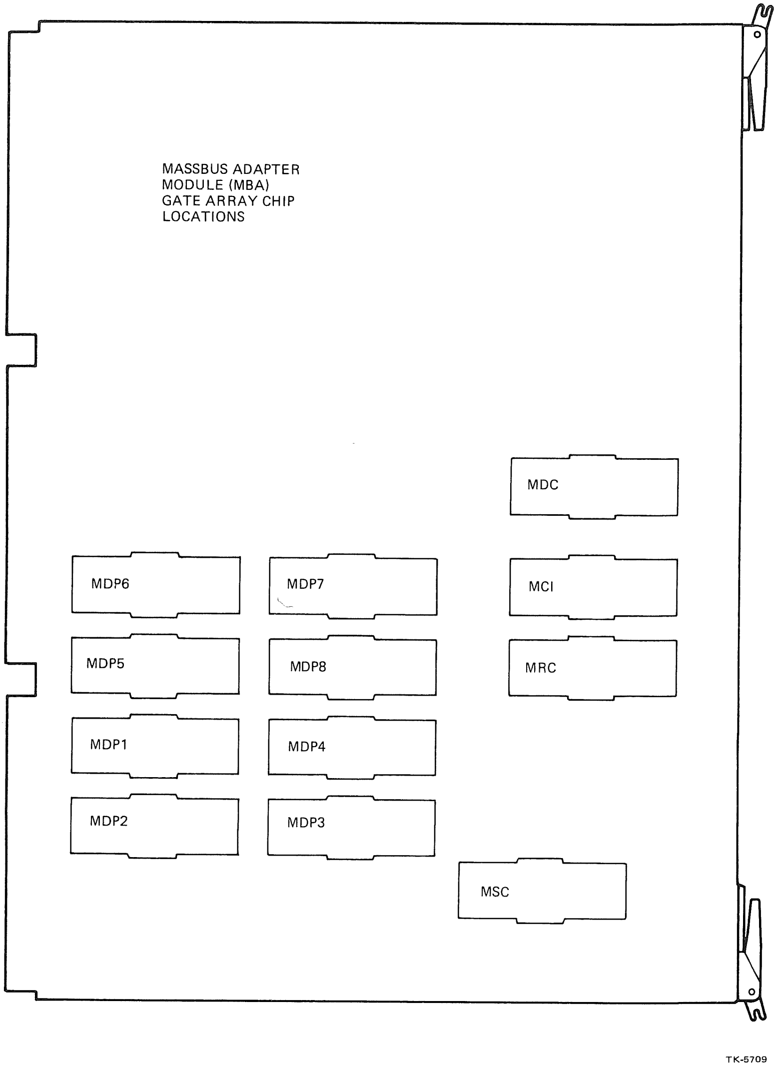
UDP - Four UNIBUS data path (UDP) chips make up the data path logic for the UBI and SUB. All address and data information passes through the UDP between four tristate ports (CMI, BUF CMI, UB data, UB address). Contains address latch and compare logic for the buffered data paths and for CPU access to UBI or SUB and UNIBUS registers.

UCN - One UNIBUS control (UCN) chip works in conjunction with the UBI and SUB control stores to provide microsequencer control of all operations. Contains the byte and error flags for the buffered data paths. Performs interpretation between the CMI and UNIBUS control signals and defines operations.

INT - One interrupt (INT) chip on the UBI performs arbitration to process all interrupt requests and inserts values on the microvector lines to steer the CCS. Contains bits of the PSL, arbitrates the UNIBUS for transactions from the CPU, arbitrates bus requests (BRs) from the backplane, and issues bus grants (BGs).

CON - Two console (CON) chips on the UBI each convert serial/parallel data for communication between the CPU and the TU58 and console terminal via the W-bus. Contain limited console command character recognition.

MBA GATE ARRAY LAYOUT (L0007)



MBA GATE ARRAY DESCRIPTION

MDP - Eight MASSBUS data path (MDP) chips route data between five tristate ports (CMI, IBUS, CBUS, SILO, MBUS). All address and data information is routed through the MDPs. Contain address latch and compare logic for CPU access to MBA and MASSBUS registers.

MDC - One MASSBUS data bus control (MDC) chip controls and maintains status on control bus parity and on MAP parity and validity. Initiates and coordinates the start and end of DMA transfers. Contains and controls bit fields of the VAR and initiates MAP parity/validity checks.

MCI - One MBA CMI interface control (MCI) chip controls MBA and CMI interfacing plus CMI arbitration, status generation and checking, and processor interrupts. Produces the CMI function codes for DMA transfers on the CMI and controls MAP address translations. Slave logic detects CPU transfers with internal or external registers and directs the MRC to perform the transfer.

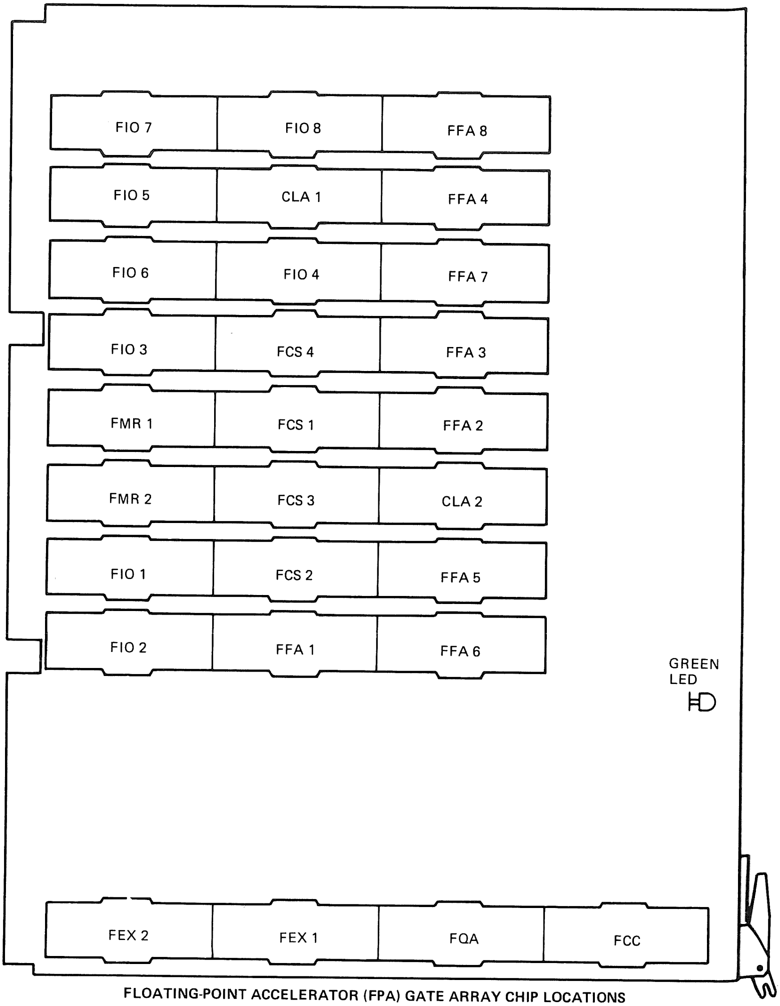
MRC - One MBA register control (MRC) chip responds to the MCI on CPU transfers with internal or external registers. Detects the issue of a valid data transfer command and defines it for the MBA. Produces control signals to the control bus and data transfer function codes that control clocking in the MDP registers.

MSC - One MBA SILO control (MSC) chip produces data transfer function codes that control data routing and alignment between the SILO and the MDP registers. Produces the SILO address, detects SILO full or SILO empty conditions, and generates and checks parity on SILO and MASSBUS data. Produces the CMI data byte mask for DMA transfers on the CMI.

Block diagram p 182,

Jumpers p 207

FPA GATE ARRAY LAYOUT (L0001)



FLOATING-POINT ACCELERATOR (FPA) GATE ARRAY CHIP LOCATIONS

TK-8067

FPA GATE ARRAY DESCRIPTION

FFA - Eight floating fractional arithmetic (FFA) chips form the arithmetic logic unit (ALU) of the FPA. The ALU performs most of the data manipulation during the execution of floating-point instructions. It outputs the 64-bit FP bus to the FIO.

CLA - Two carry look ahead (CLA) chips contain logic used by the ALU and incrementer multiplexer circuits.

FIO - Eight floating input/output (FIO) chips interface operands from the M-bus and W-bus. Store immediate multiple operand results.

FCS - Four floating coarse shifter (FCS) chips perform right or left shifts in multiples of four. Produce a 67-bit output from a 64-bit input.

FQA - One floating quick aligner (FQA) chip positions the FCS for certain operations.

FMR - Two fraction multiplier (FMR) chips contain most of the logic that performs fraction multiplication.

FEX - Two floating exponent (FEX) chips form the exponent data path.

FCC - One floating condition code (FCC) chip processes all operands to obtain condition code status for traps, faults, and the PSL.

GATE ARRAY PART NUMBER/MODULE CROSS-REFERENCE

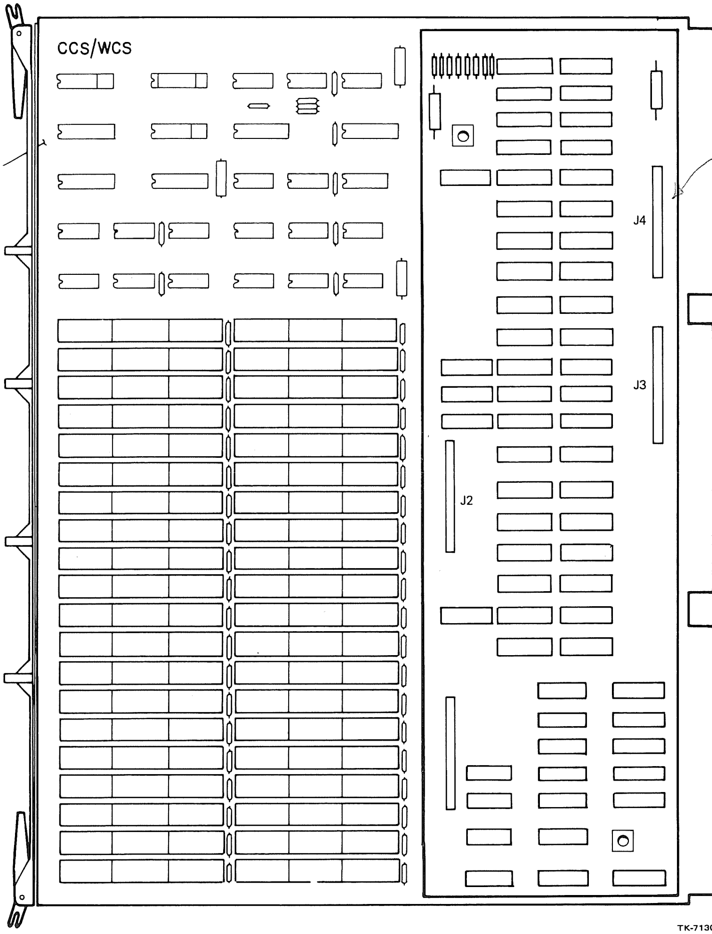
Chip Number	Chip Mnemonic	DIGITAL Part No.	Home Module	Diagram Number	Data String
DC606	TIM	19-14680	DPM	1	(Use TOK chip)
DC607	MDR	19-14681	MIC	1	<00,08,16,24>
				2	<01,09,17,25>
				3	<02,10,18,26>
				4	<03,11,19,27>
				5	<04,12,20,28>
				6	<05,13,21,29>
				7	<06,14,22,30>
				8	<07,15,23,31>
DC608	ALP	19-14682	DPM	1	<03:00>
				2	<07:04>
				3	<11:08>
				4	<15:12>
				5	<19:16>
				6	<23:20>
				7	<27:24>
				8	<31:28>
DC609	ADD	19-14683	MIC	1	<07:00>
				2	<15:08>
				3	<23:16>
				4	<31:24>
DC610	CCC	19-14684	DPM	1	
DC611	CON	19-14685	UBI	1	(TU58)
				2	(Console)
DC612	CLA	19-14686	DPM	1	
	FPA			1	<IC7:IC0>
				2	<C7:C0>
DC613	SRM	19-14687	DPM	1	<00,04,08,12, 16,20,24,28,32>
				2	<01,05,09,13, 17,21,25,29,33>
				3	<02,06,10,14, 18,22,26,30,34>
				4	<03,07,11,15, 19,23,27,31>
DC614	SRK	19-14688	DPM	1	
DC615	ALK	19-14689	DPM	1	
DC616	SPA	19-14690	DPM	1	
DC617	SAC	19-14691	DPM	1	

Chip Number	Chip Mnemonic	DIGITAL Part No.	Home Module	Diagram Number	Data String
DC618	UDP	19-14692	UBI/SUB	1	<00,01,08,09, 16,17,24,25>
				2	<02,03,10,11, 18,19,26,27>
				3	<04,05,12,13, 20,21,28,29>
				4	<06,07,14,15, 22,23,30,31>
DC619	UCN	19-14693	UBI/SUB	1	
DC620	TOK	19-14694	DPM	1	
DC621	MSQ	19-14695	DPM	1	
DC622	IRD	19-14696	DPM	1	
DC623	CMK	19-14697	MIC	1	
DC624	PRK	19-14698	MIC	1	
DC625	ACV	19-14699	MIC	1	
DC626	ADK	19-14700	MIC	1	
DC627	CAK	19-14701	MIC	1	
DC628	UTR	19-14702	MIC	1	
DC629	PHB	19-14703	DPM	1	
DC630	INT	19-14704	UBI	1	
DC631	MEC	19-14705	CMC	1	<15:00>
				2	<31:16>
DC632	MAP	19-14706	CMC	1	(L0011)
DC633	MDL	19-14707	CMC	1	<07:00>
				2	<15:08>
				3	<23:16>
				4	<31:24>
DC636	FIO	19-14710	FPA	1	<35:32,03:00>
				2	<39:36,07:04>
				3	<43:40,11:08>
				4	<47:44,15:12>
				5	<51:48,19:16>
				6	<55:52,23:20>
				7	<59:56,27:24>
				8	<63:60,31:28>

GATE ARRAY PART NUMBER/MODULE CROSS-REFERENCE (CONT)

Chip Number	Chip Mnemonic	DIGITAL Part No.	Home Module	Diagram Number	Data String
DC637	FCS	19-14711	FPA	1	<00,04,08...64>
				2	<01,05,09...65>
				3	<02,06,10...66>
				4	<03,07,11...67>
DC638	FFA	19-14712	FPA	1	<07:00>
				2	<15:08>
				3	<23:16>
				4	<31:24>
				5	<39:32>
				6	<47:40>
				7	<55:48>
				8	<63:56>
DC639	FMR	19-14713	FPA	1	<59:56,51:48...>
				2	<63:60,55:52...>
DC641	FEX	19-14715	FPA	1	<04:00>
				2	<09:05>
DC642	FQA	19-14716	FPA	1	
DC643	FCC	19-14717	FPA	1	
DC645	MDP	19-14719	MBA	1	<17,16,01,00>
				2	<19,18,03,02>
				3	<21,20,05,04>
				4	<23,22,07,06>
				5	<25,24,09,08>
				6	<27,26,11,10>
				7	<29,28,13,12>
				8	<31,30,15,14>
DC646	MSC	19-14720	MBA	1	
DC647	MRC	19-14721	MBA	1	
DC648	MDC	19-14722	MBA	1	
DC649	MCI	19-14723	MBA	1	
DC650	MAD	19-14724	CMC	1	(L0016)
DC651	CML	19-14725	MIC	1	(DR750 usage)

CCS MODULE WITH WCS (L0005)



IF NO WCS INSTALLED JUMPER J4 pins 16 → 18
of BACKPLANE B00544-48

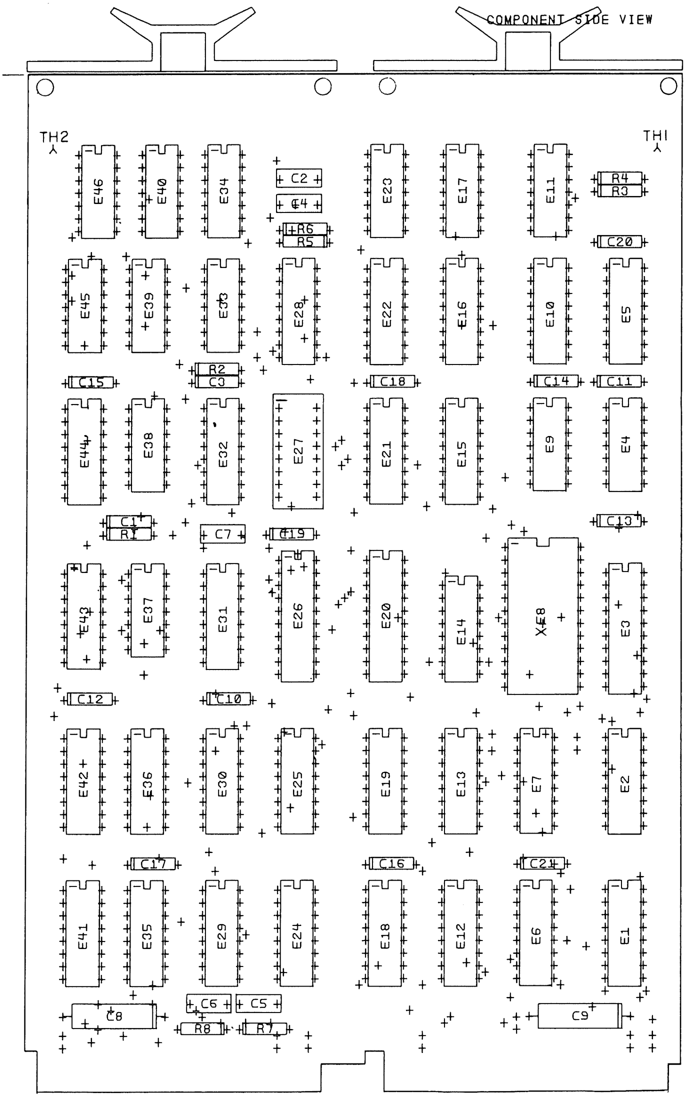
Row 1 2 3 4 5 6

CCS MODULE ROM LAYOUT

```
                                                         +-------+-------+---+
                                                         |       |       |   |
                                                         |  MUX  |  ALU  |DQ1|
                                                         |       |       |   |
            +-+-+---+---------+---+---------+-----------+--------+-------+---+----------+---+
            | | |   |         |   |         |           |                    |          | D |
            |P|P|   |         |   |         |           |                    |          | T |
            |A|A|LIT|  MISC   |SPW|  MSRC   |    ROT    |       ALPCTL       |   BUT    | Y |
            |R|R|   |         |   |         |           |                    |          | P |
            |2|1|   |         |   |         |           |                    |          | E |
            +-+-+---+---------+---+---------+-----------+--------+-------+---+----------+---+
            |7 7 7 7|7 7 7 7|7 7 6 6|6 6 6 6|6 6 6 6|5 5 5 5|5 5 5 5|5 5 4 4|4 4 4 4|4 4 4 4|
            |9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|
+-----------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
| 0000-03FF | E146  | E140  | E134  | E128  | E122  | E116  | E110  | E104  |  E98  |  E92  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0400-07FF | E145  | E139  | E133  | E127  | E121  | E115  | E109  | E103  |  E97  |  E91  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0800-0BFF | E144  | E138  | E132  | E126  | E120  | E114  | E108  | E102  |  E96  |  E90  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0C00-0FFF | E143  | E137  | E131  | E125  | E119  | E113  | E107  | E101  |  E95  |  E89  |
|           |       |       |       |       |       |       |       |       |       |       |
| 1000-13FF | E142  | E136  | E130  | E124  | E118  | E112  | E106  | E100  |  E94  |  E88  |
|           |       |       |       |       |       |       |       |       |       |       |
| 1400-17FF | E141  | E135  | E129  | E123  | E117  | E111  | E105  |  E99  |  E93  |  E87  |
+-----------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

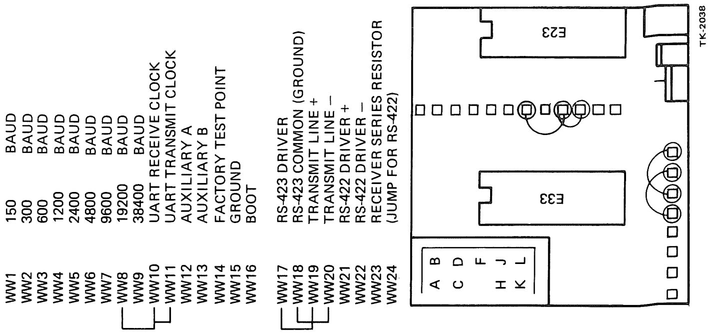
```
            +-----------------+            *Note:  The microcode listing is reversed for these
            |                 |                    bits with respect to the actual PROM pattern.
            |      LITRL      |
            |                 |
            +---------+-+-----+-----------+----------+-------+-+-+-------------------------+
            |         |I|     |           |          |       | | |                         |
            |         |S|     |           |          |       |C|J|                         |
            |  RSRC   |T| C   |   WCTRL   |   BUS    |  FPA  |L|S|          NEXT           |
            |         |R| C   |           |          |       |K|R|                         |
            |         |M|     |           |          |       |X| |<-   Reversed ->|*       |
            +-------+-+-+-+-+-+-----+-----+-+--------+-------+-+-+-----+-------+--+--------+
            |3 3 3 3|3 3 3 3|3 3 2 2|2 2 2 2|2 2 2 2|1 1 1 1|1 1 1 1|1 1 0 0|0 0 0 0|0 0 0 0|
            |9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|
+-----------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
| 0000-03FF |  E86  |  E80  |  E74  |  E68  |  E62  |  E56  |  E50  |  E44  |  E38  |  E32  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0400-07FF |  E85  |  E79  |  E73  |  E67  |  E61  |  E55  |  E49  |  E43  |  E37  |  E31  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0800-0BFF |  E84  |  E78  |  E72  |  E66  |  E60  |  E54  |  E48  |  E42  |  E36  |  E30  |
|           |       |       |       |       |       |       |       |       |       |       |
| 0C00-0FFF |  E83  |  E77  |  E71  |  E65  |  E59  |  E53  |  E47  |  E41  |  E35  |  E29  |
|           |       |       |       |       |       |       |       |       |       |       |
| 1000-13FF |  E82  |  E76  |  E70  |  E64  |  E58  |  E52  |  E46  |  E40  |  E34  |  E28  |
|           |       |       |       |       |       |       |       |       |       |       |
| 1400-17FF |  E81  |  E75  |  E69  |  E63  |  E57  |  E51  |  E45  |  E39  |  E33  |  E27  |
+-----------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

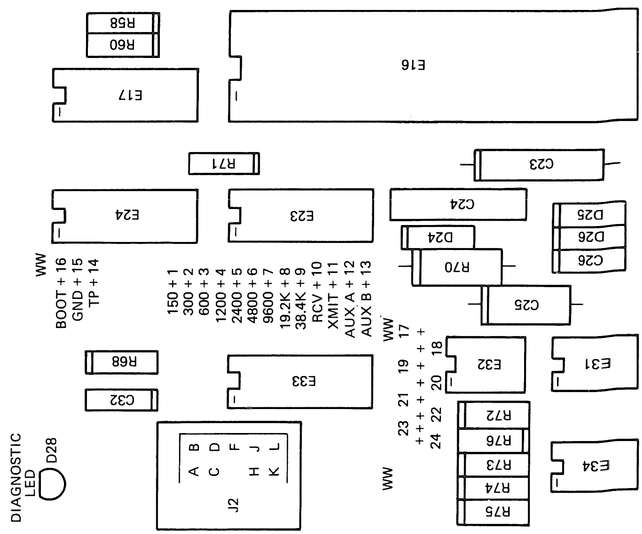
UET COMPONENT LAYOUT



NOTE: For circuit details refer to schematic D-CS-M9313-Ø-4.

TU58 INTERFACE COMPONENT AND JUMPER LAYOUT

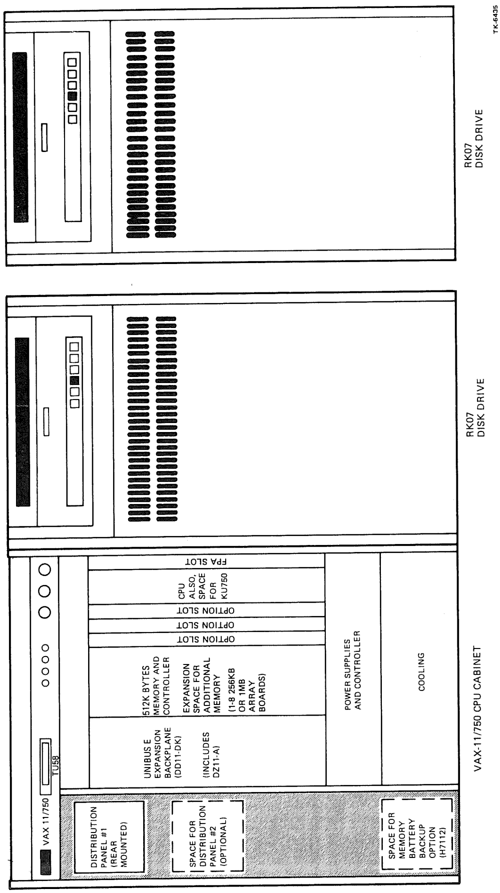




CHAPTER 4
CONFIGURATION AND CABLING

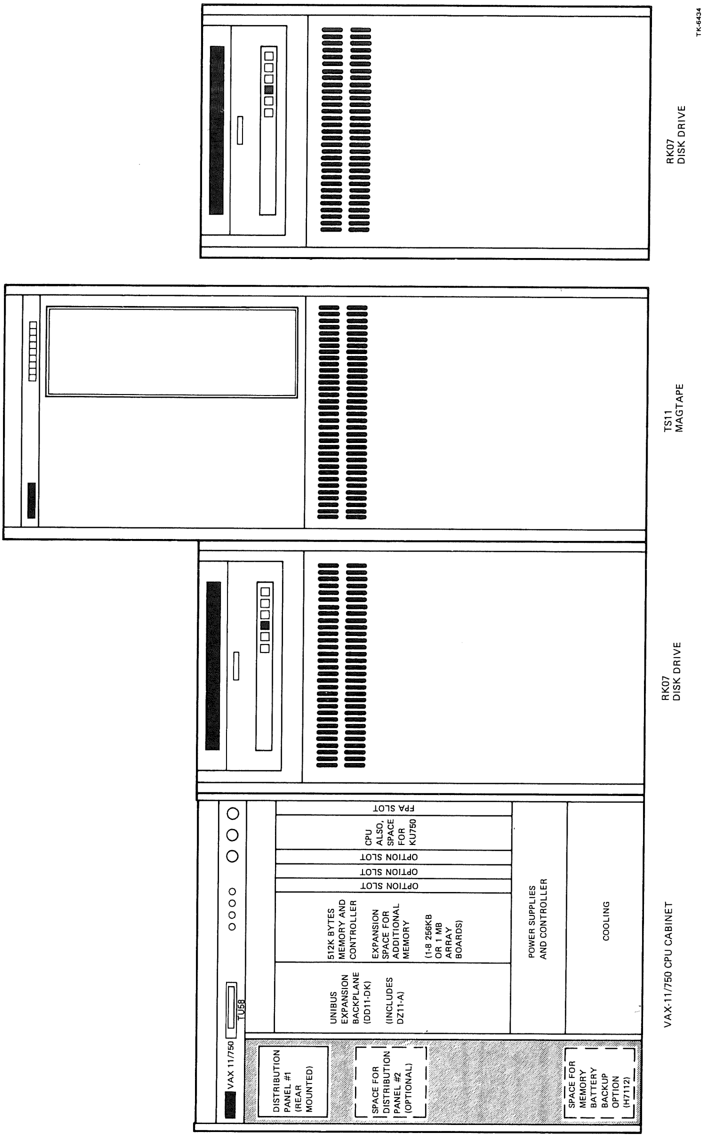
VAX-11/750 SYSTEM CONFIGURATIONS

DUAL RK07 SYSTEM



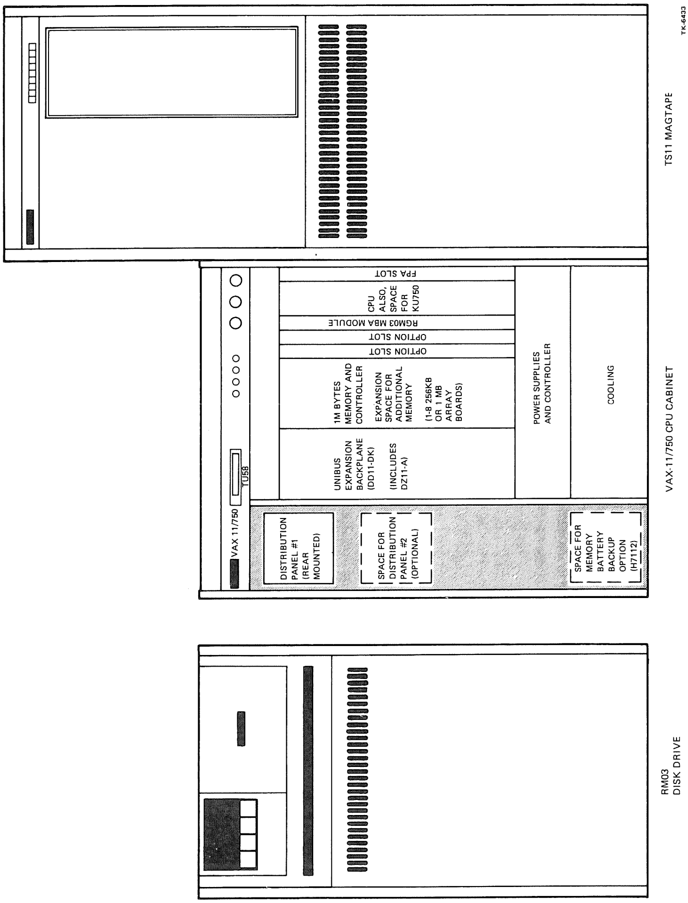
VAX-11/750 SYSTEM CONFIGURATIONS (CONT)

DUAL RK07 SYSTEM WITH TS11



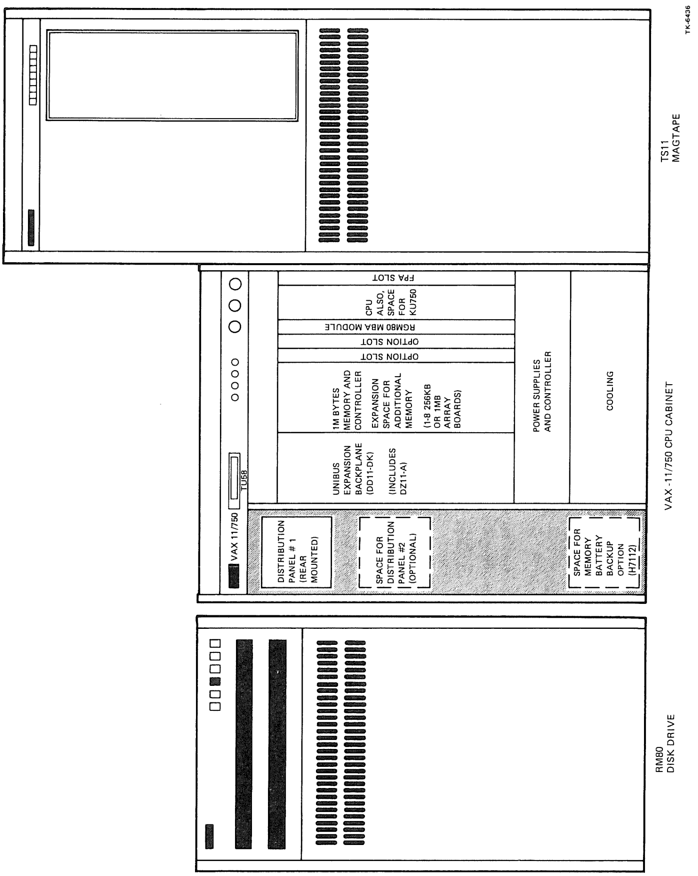
VAX-11/750 SYSTEM CONFIGURATIONS (CONT)

RM03/TS11 SYSTEM



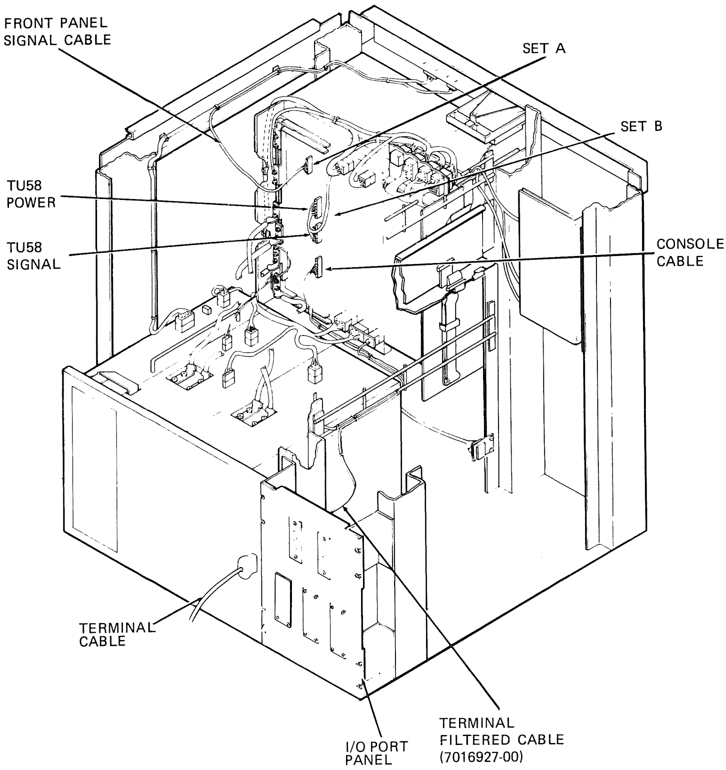
VAX-11/750 SYSTEM CONFIGURATIONS (CONT)

RM80/TS11 SYSTEM



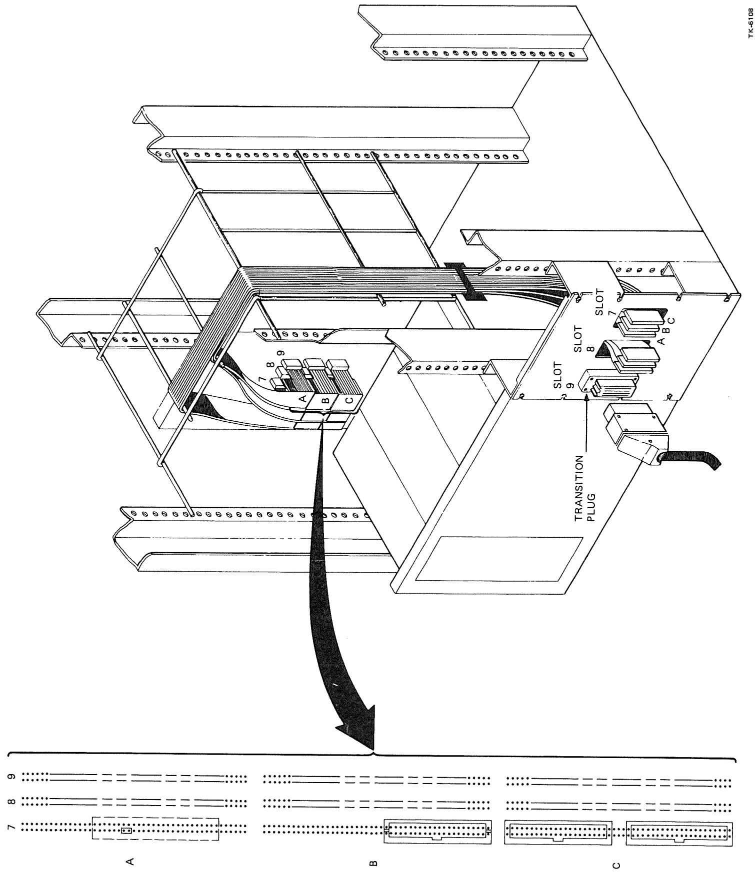
SYSTEM CABLING DIAGRAMS

TU58 AND FRONT PANEL CABLES

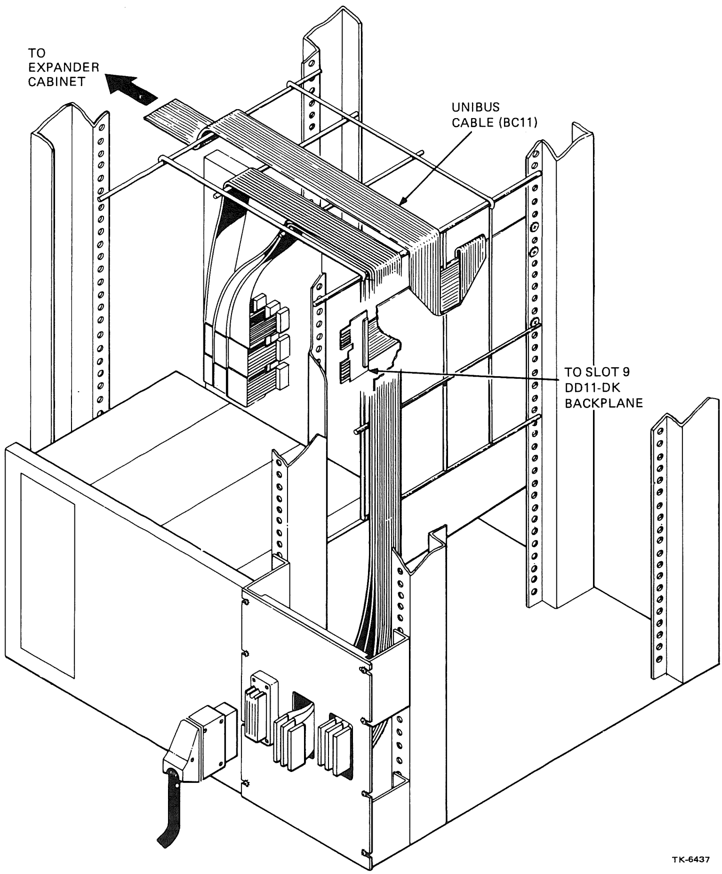


SYSTEM CABLING DIAGRAMS (CONT)

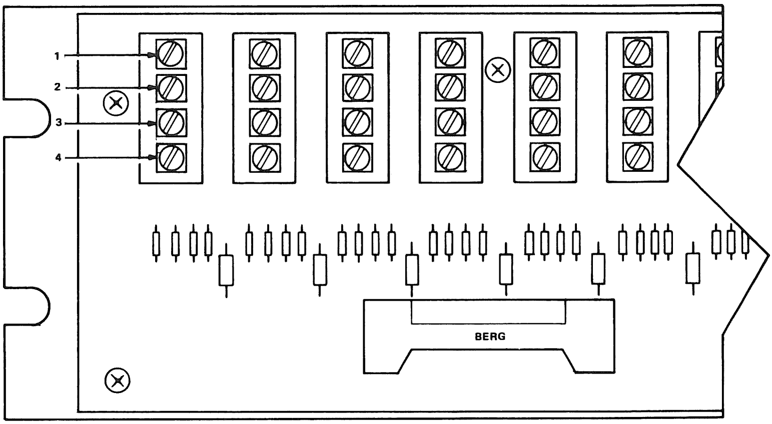
MASSBUS CABLES



UNIBUS CABLES



DZ11 DISTRIBUTION PANEL



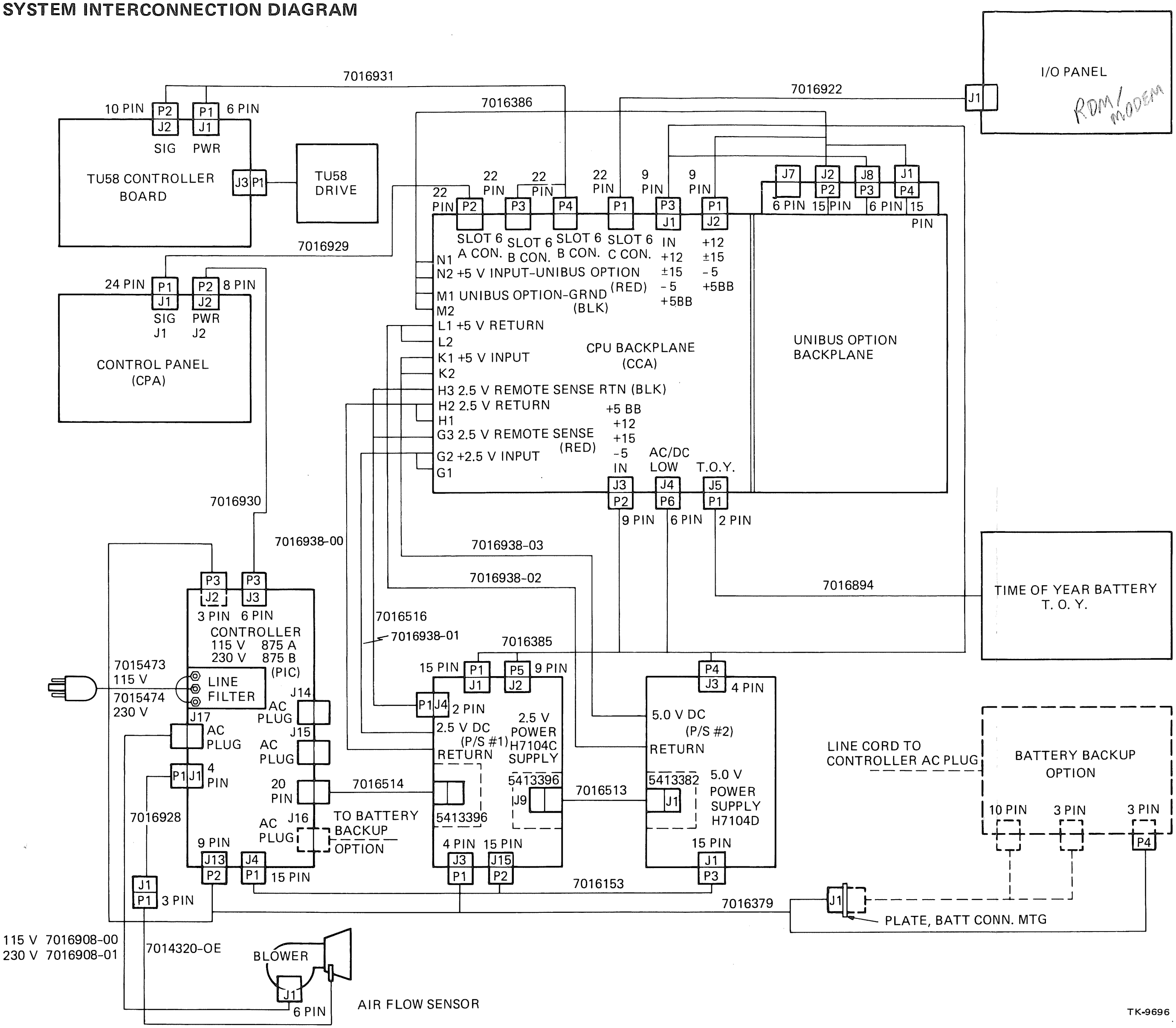
MA-0802

DZ11 DISTRIBUTION TERMINALS

Terminal Number	Signal Name
1	REC +
2	REC -
3	XMIT -
4	XMIT +

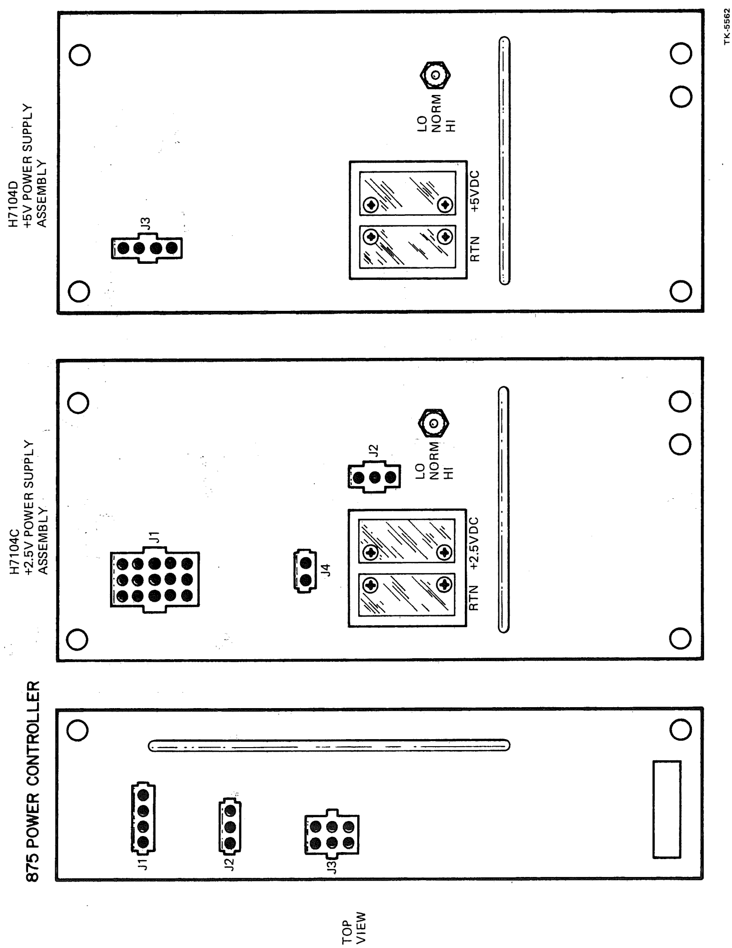
VAX-11/750 POWER SYSTEM

SYSTEM INTERCONNECTION DIAGRAM

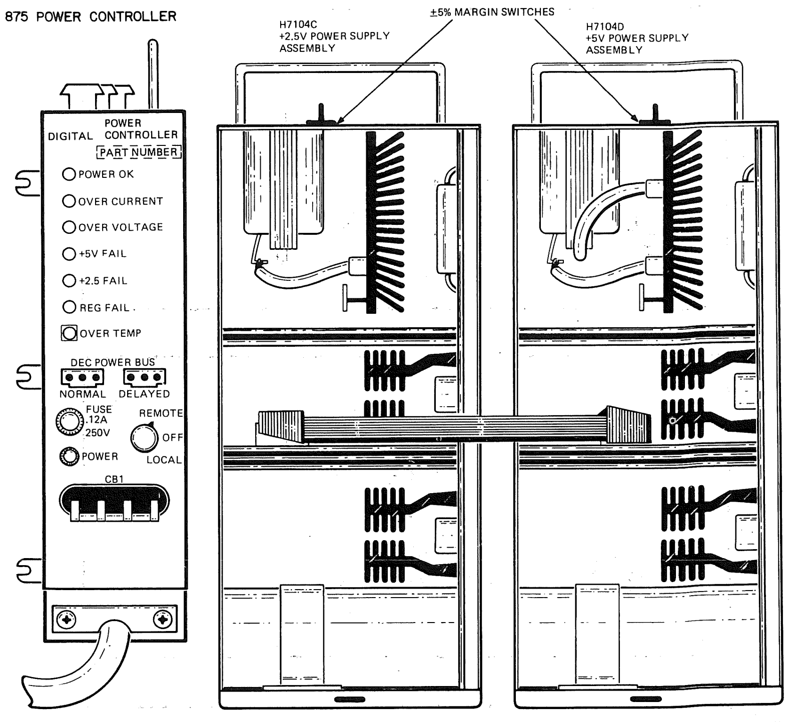


225+203.

VAX-11/750 POWER SUPPLY, PART 1



VAX-11/750 POWER SUPPLY, PART 2



The ac power controller panel consists of eight indicators, one circuit breaker, one REMOTE/LOCAL switch, one .12 A (1/8 A) fuse, and two DIGITAL power bus connectors.

AC POWER CONTROLLER PANEL

Indicator	Definition
DC OK	The green DC OK indicator is on when the power system is functioning correctly. It is off if any fault indicator is on in the power supply system.
OVERVOLTAGE	The red OVERVOLTAGE indicator is on when there is an overvoltage condition in the +2.5 V or +5 V power supply. The affected power supply is indicated by its fail indicator being on.
OVERCURRENT	The red OVERCURRENT indicator is on when there is an overcurrent condition in the +2.5 V or +5 V power supply. The affected power supply is indicated by its fail indicator being on.
+5 V FAIL	The red +5 V FAIL indicator is on when there is a malfunction in the +5 V power supply.
+2.5 V FAIL	The red +2.5 V FAIL indicator is on when there is a malfunction in the +2.5 V power supply.
REG FAIL	The red REG FAIL indicator is on when there is a regulator malfunction in the +/-5 V, +12 V, or +/-15 V power supply.
OVER TEMP	The OVER TEMP indicator is on when there is an overtemperature condition in the +5 V or +2.5 V power supply. The indicator is not lit, however, when the system is shut down by an overtemperature condition inside the controller.
POWER	The POWER indicator is on when the ac power cable is plugged into a live ac power source. It is on whether CB1 is on or off.

REMOTE/LOCAL SWITCH

Position	Definition
REMOTE	This is the normal operating position. Unswitched ac power is applied to the power bus. Switched ac power is controlled by the key operated switch on the CPU front panel.
OFF	No switched ac power can be applied to the system.
LOCAL	Switched ac power is applied to the system regardless of the position of the CPU power keyswitch.

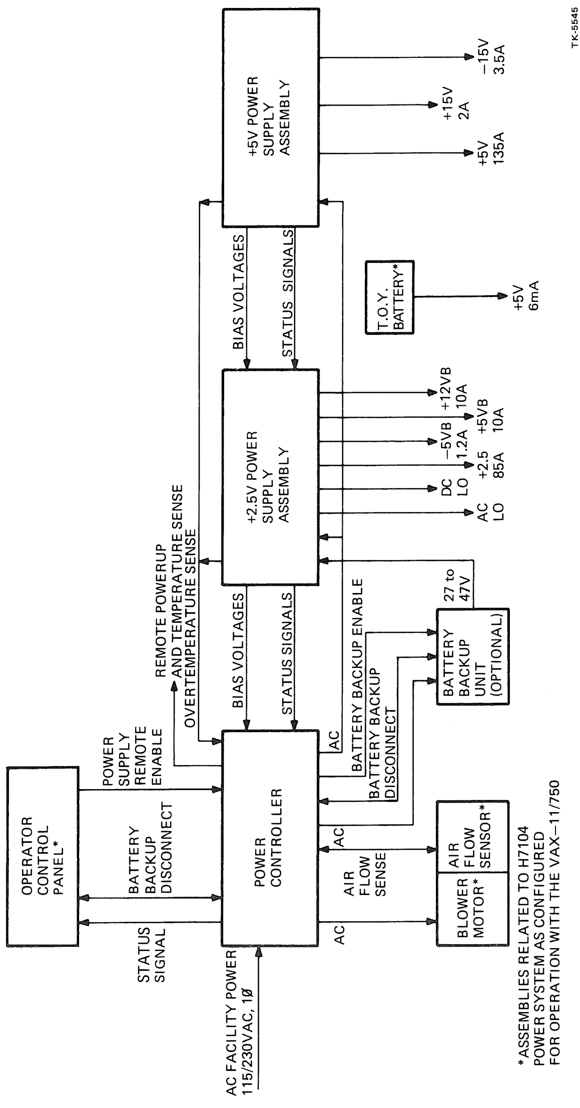
CIRCUIT BREAKER (CB1) — CB1 is the main circuit breaker to the power supply system. When in the ON position, unswitched ac power is applied to the power bus and switched ac power is determined by the position of the REMOTE/LOCAL switch. When in the OFF position, ac power cannot be applied to the power bus.

DEC POWER BUS — There are two DIGITAL power bus connectors: normal and delayed. These connectors provide a POWER UP REQUEST signal that can be interconnected between the H875 power system and remote power systems.

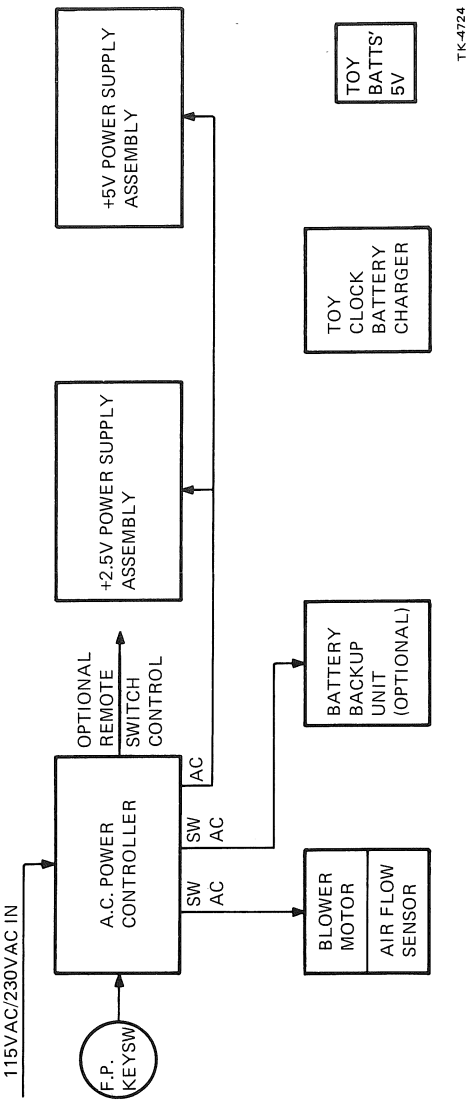
There is a half-second delay on delayed power bus connector output J9.

VAX-11/750 POWER SYSTEM (CONT)

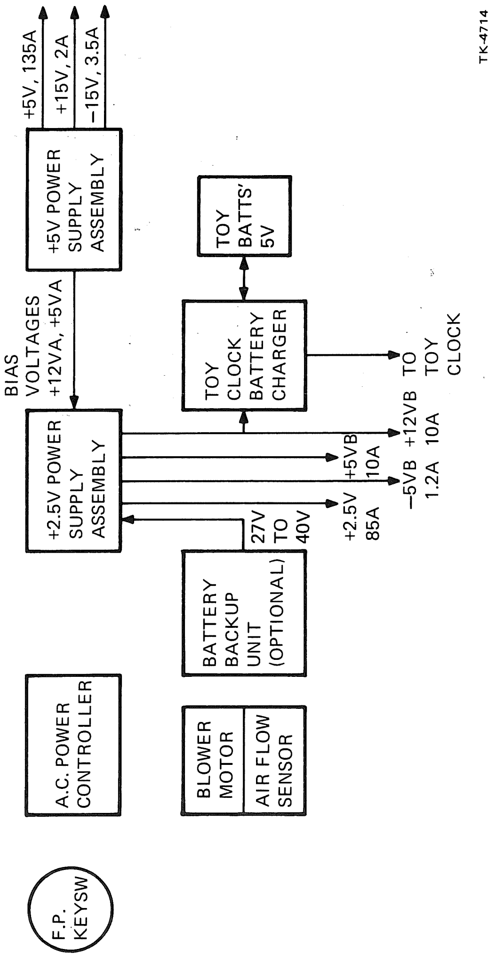
H7104 POWER SYSTEM BLOCK DIAGRAM



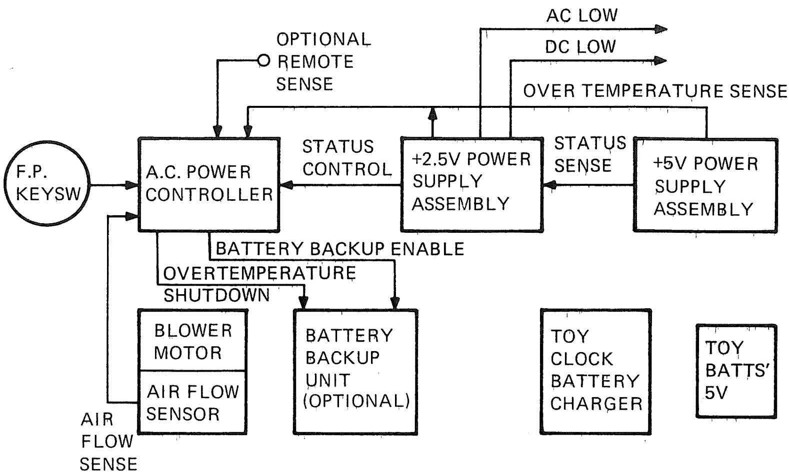
AC POWER DISTRIBUTION



DC POWER DISTRIBUTION

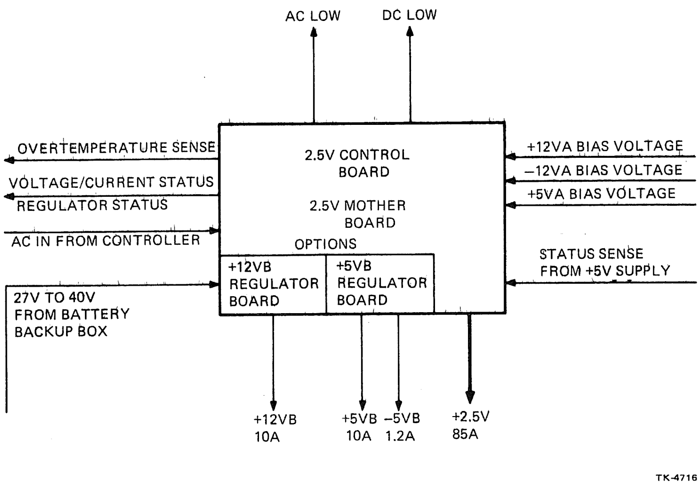


POWER SYSTEM SENSING

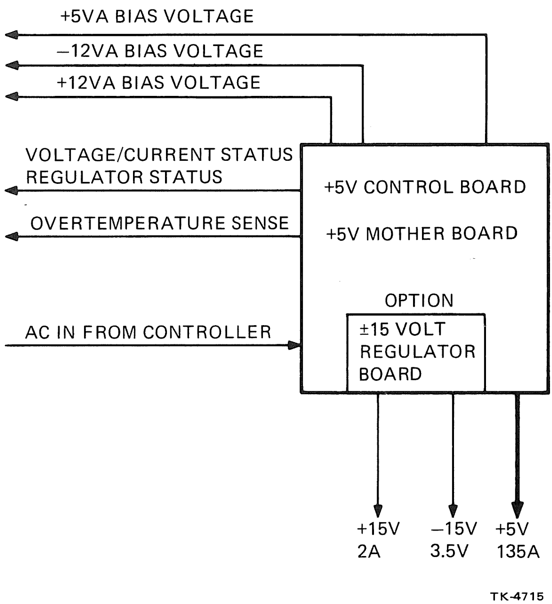


TK-4713

+2.5 V POWER SUPPLY BLOCK DIAGRAM



+5 V POWER SUPPLY BLOCK DIAGRAM



APPLYING SYSTEM POWER

Use the following procedure to correctly apply power to the system.

1. Ensure that the CPU's main circuit breaker (CB1) on the ac power controller is off (down).

2. Verify that the two 5 percent margin switches on the power supply are in the center position (up).

3. Place the REMOTE/LOCAL switch on the ac power controller panel in the REMOTE position. The CPU power key lock switch on the front panel should be in the off position.

4. Connect the CPU cabinet ac power cable to the external ac power source. The power phase indicator on the ac power controller should now be on.

5. Move the CB1 circuit breaker to the on position (up).

6. Power can now be applied to the CPU by the key lock switch on the CPU front panel.

CPU CABINET POWER REQUIREMENTS

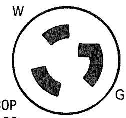
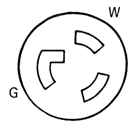
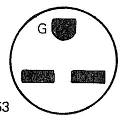
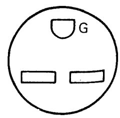
Single Phase	Nominal	Mininum	Maximum
Vac (RMS)	120	90	128
Phase to neutral	120	90	128
Phase to ground*	120	90	128
Neutral to ground	N/A	N/A	N/A
Hertz	60	47	63
Hertz	50	47	63
Current (amperes)			25.0 A at 90 Vac
Vac (RMS)	240	180	256
Phase to neutral	240	180	256
Phase to ground	N/A	N/A	N/A
Neutral to ground	N/A	N/A	N/A
Hertz	60	47	63
Hertz	50	47	63
Current (amperes)			12.5 A at 180 Vac

CAUTION

Expansion cabinets with separate power cables must be powered from the same ac line phase as the CPU or damage may result to the equipment. Also, systems that are interconnected by cables and share the same logic and/or chassis grounding must share the same phase.

VAX-11/750 POWER SYSTEM (CONT)

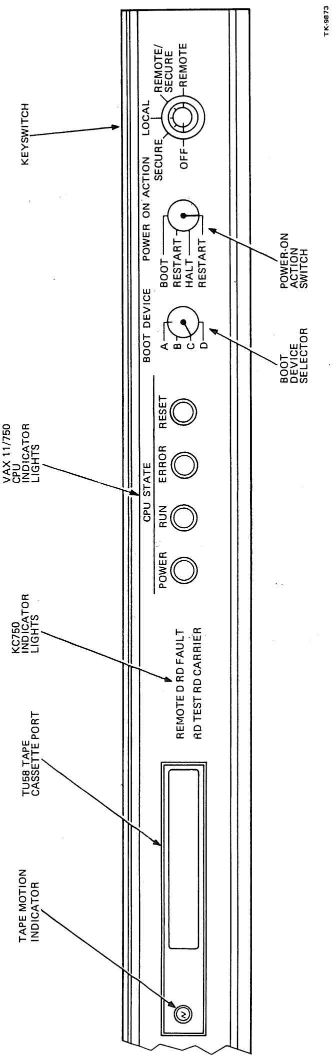
STANDARD POWER PLUGS AND RECEPTACLES

120V 30A 1-PHASE	HUBBEL #2611 NEMA # L5-30P DEC # 12-11193 	#2610 L5-30R 12-11194 
240V 15A 1-PHASE	NEMA # 6-15P DEC # 90-08853 	6-15R 12-11204 

TK-5997

CHAPTER 5
BOOTSTRAPPING AND OPERATION

VAX-11/750 FRONT PANEL



CONSOLE PANEL INDICATORS

Indicator	Description
POWER	Indicates that the console subsystem is supplied with correct voltage. The VAX-11/750 processor can lose partial power and still light the LED and allow diagnostic testing.
RUN	The CPU is in program mode, running a main memory program.
ERROR	Glows dimly to indicate a control store (CCS or WCS) parity error. Fully on indicates a double control store parity error and CPU clock stopped.
RESET Switch	Pressing the RESET switch causes the system to perform the action selected by the POWER ON ACTION switch (initializes the processor).
REMOTE D	Power key switch is in the REMOTE or REMOTE SECURE position.
RD FAULT	RDM logic failure. Lights for about ten seconds during console power-up as part of logic self-test.
RD TEST	DDC host computer is performing RD tests.
RD CARRIER	Carrier signal detected from DDC.
Tape Motion	Steady on during a search or rewind. Blinking indicates a read or write is in progress.

CONSOLE SWITCH FUNCTIONS

Position	Description
Power Key Switch	
OFF	Switched ac power is removed from the system. Unswitched ac power is still applied.
LOCAL	Normal on position. All power is applied and the operator controls the system from the console.
	Console mode - The operator performs console commands.
	Program mode - The operator communicates with the system program. CTRL/P and CTRL/D are not passed to the system program but are recognized by the console subsystem.
LOCAL SECURE	Normal operation as for LOCAL except as follows:
	Console mode - The local terminal and the RESET switch are disabled.
	Program mode - CTRL/P and CTRL/D are ignored by the console subsystem and are passed to the system program.
REMOTE	System responds only to the remote terminal and to a remote CTRL/P or CTRL/D to change processor states.
REMOTE SECURE	System responds only to the remote terminal and remote CTRL/P or CTRL/D are passed to the system program. The remote operator can enable the system TALK state to communicate with the local operator.

VAX-11/750 FRONT PANEL (CONT)

CONSOLE SWITCH FUNCTIONS (CONT)

Position	Description
Power-On Action Switch	
BOOT	The console subsystem performs a power-up bootstrap of the system on a power-up, a fatal error, or when the RESET switch is pressed. The boot is performed from the device selected by the BOOT DEVICE switch.
RESTART/BOOT	On a power-fail restart, the CPU microcode initialize sequence checks for a valid restart parameter block (RPB). If the RPB is valid, the program returns to its previous operating state. If not, the system performs a bootstrap sequence from the device selected by the BOOT DEVICE switch.
HALT	The processor halts and no restart is attempted.
RESTART/HALT	A restart is attempted as for RESTART. If unsuccessful, however, the processor halts.
Boot Device Switch (Typical Devices)	
Device A	TU58 Boot ROM
Device B	System Disk Boot ROM (DB - MASSBUS, DM - UNIBUS)
Device C	Alternate Disk Boot ROM
Device D	Spare

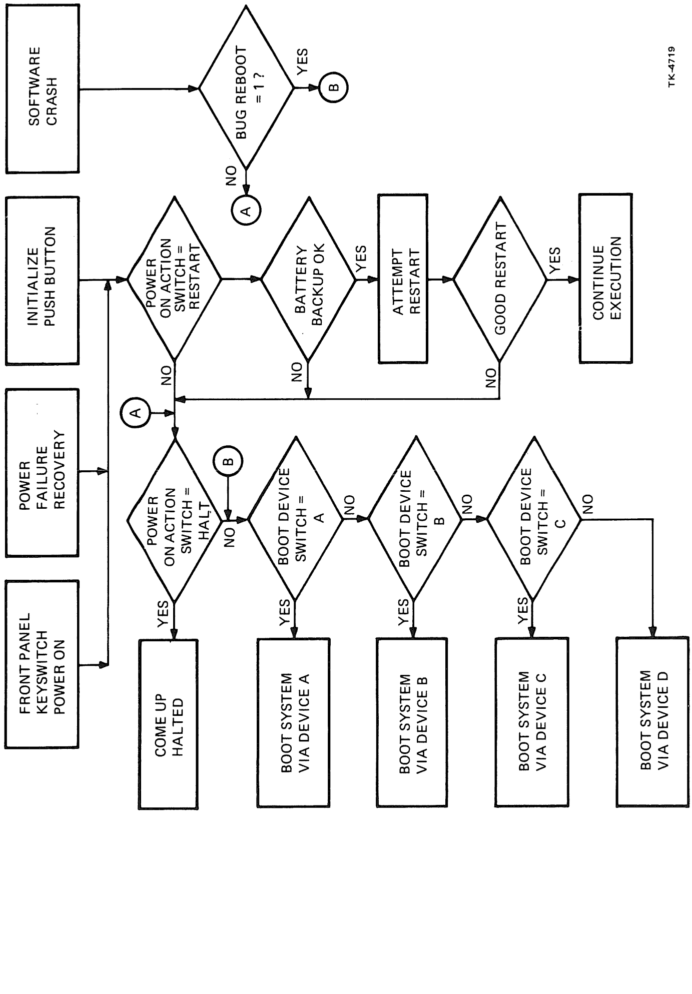
* BOOTROMS ARE ON MEMORY CONTROLLER

* BOOTROM ADDRESSES: (READ LOWER TWO BYTES)

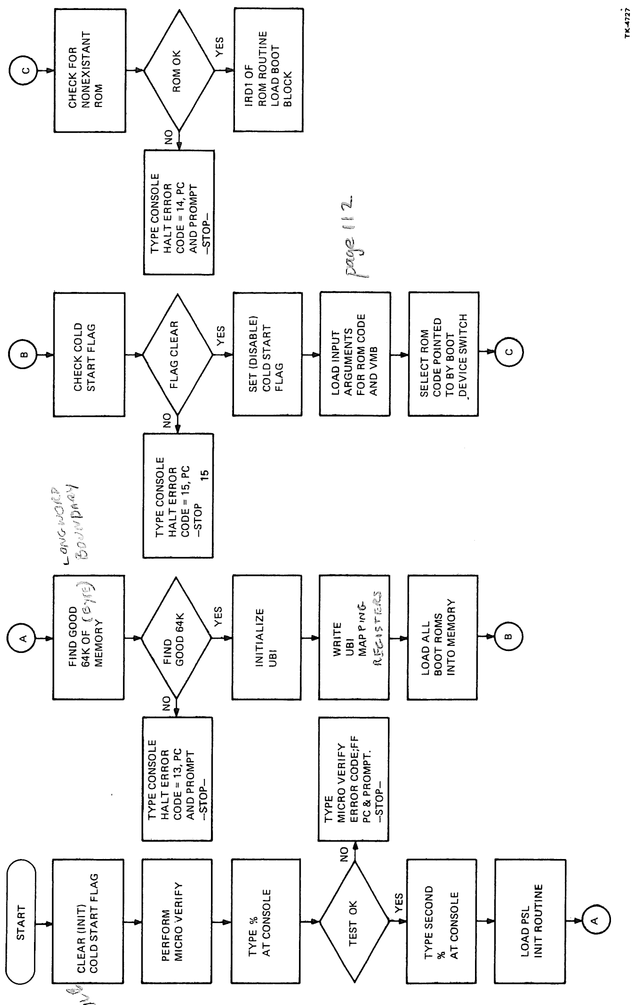
A - F20400
B - F20500
C - F20600
D - F20700

(SEE PAGE 140 (VI) FOR CONVERSION)

CONSOLE BOOTSTRAP FLOW



CONSOLE SUBSYSTEM ACTION ON A BOOT



CSR 2 module
on cmes
page 15

LOADS PSL = 041F0000
(IE. PRIORITY = 32)

BOOTSTRAP SEQUENCE

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

1. The operator powers up the VAX-11/750 system.

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 the microcode cannot perform a restart, it will perform a bootstrap from the default or execute a halt.

   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) standalone 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 standalone 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.

* 750 DIAGNOSTIC SYSTEM OVERVIEW MANUAL CHAPTER 4

5. VMB.EXE is the primary bootstrap program. It 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 as virtual (rather than physical) addresses in the RPB

(1Ø) Loads the CPU dependent code image into nonpaged pool and links it into the system

(11) Loads the terminal handler into nonpaged 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. Therefore, some care is required when configuring multiple controllers of possible system disks across multiple buses.)
unit	Passed from VMB.EXE input, register R3.

(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 controler, 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 to 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 that have an encoded format)

   c. Creates paging, swapping, and system dump files

INPUT ARGUMENTS

The general registers receive the following input arguments from the console subsystem.

R1 - system bus address of a MASSBUS adapter (MBAØ unless otherwise specified in the BOOT command).

R2 - physical address of the UNIBUS I/O page associated with a UNIBUS adapter (UBIØ unless otherwise specified in the BOOT command).

R3 - device unit number (Ø unless otherwise specified in the BOOT command).

R5 - software boot control flags (Ø unless otherwise specified in the BOOT command).

SP - <base address + ^X2ØØ> of the 64K bytes of good memory.

(refer to page 193 for address map.)

See page 238 for load addresses

SOFTWARE BOOT CONTROL FLAGS

Flag	Hex Value	Function
Ø	/1	Conversational boot. Returns the prompt SYSBOOT> to allow alteration of VMS parameter.
1	/2	Debug. This flag is passed through to VMS and causes the code for the executive debugger to be included in the running system.
2	/4	Initial breakpoint. If this flag is set, and the executive debugger code is included (flag bit 1), a breakpoint will occur immediately after executive mode enables mapping.
3	/8	Not used on the VAX-11/75Ø.
4	/1Ø	Diagnostic boot. This flag causes a boot by file name for the diagnostic supervisor.
5	/2Ø	Bootstrap breakpoint. This flag causes the bootstrap to stop at a breakpoint after performing necessary initialization.
6	/4Ø	Image header. If this flag is set, the transfer address from the image header of the boot file will be used. Otherwise, control will transfer to the first byte of the boot file.
7	/8Ø	Memory test inhibit. This flag inhibits the testing of memory during bootstrapping.
8	/1ØØ	File name. Causes the bootstrap to solicit the name of the boot file.
9	/2ØØ	Halt before transfer. Causes a halt instruction to be executed before transfer to the secondary boot file. This option is useful for debugging.

VMB PRIMARY BOOT FAILURES

BOOT is the program name for VMB.EXE.
The 'F' indicates a fatal error and the type of error is reported.

%BOOT-F-Unknown processor	Indicates that the CPU is not a VAX-11/750 or VAX-11/780. Check SID register; if wrong, CCS module is bad.
%BOOT-F-Unexpected exception	Indicates that one of the following exceptions occurred: 1. Access violation 2. Breakpoint op code 3. Reserved operand 4. T-bit trap 5. Page fault (TNV)
%BOOT-F-Unexpected machine check	Indicates that a machine check occurred. Check all apapters using console EXAMINE and DEPOSIT commands. Probably a timeout.
%BOOT-F-Nonexistent drive	Self-explanatory. Check boot command and ensure system disk is drive being booted.
%BOOT-F-Unable to locate boot file	VMB cannot find [SYSEXE]SYSBOOT.EXE or if bit 4 in R5 is set, VMB cannot find [SYSMAINT]DIAGBOOT.EXE.
%BOOT-F-Bootfile not contiguous	Indicates that [SYSEXE]SYSBOOT.EXE or [SYSMAINT]DIAGBOOT.EXE is not contiguous on system disk. Recopy or rebuild.
%BOOT-F-I/O error reading boot file	Indicates a problem reading boot file from disk by $QIO service (VMS system service).

CONSOLE COMMANDS

Command	Description
CTRL/P	Enter console mode, issues >>> prompt.
CTRL/D	Enter RDM console mode, issues RDM> prompt.
>>>E	Examine command
>>>D	Deposit command

Format:

E [QUALIFIER]<SP>[ADDRESS]<CR>

D [QUALIFIER]<SP>[ADDRESS]<SP>[DATA]<CR>

Qualifiers:	/B	Set size to byte
	/W	Set size to word
	/L	Set size to longword
	/P	Physical address space
	/V	Virtual address space
	/I	IPR
	/G	GPR
Address:	nnnn	Hex number of physical or virtual address
	<SP>*	Last address
	<SP>+	Next address (deposit only)
	<SP>P	PSL

>>>H	Processor halt command 780 ONLY
>>>I	Processor initialize command; invalidates TB and cache. Issues processor INIT and UNIBUS INIT.
>>>T	Test command; runs microverify routine.
>>>S nnnn	Start command; performs initialize functions, stores specified hex address in PC, and starts program there.
>>>S<CR>	Start command; performs initialize functions and starts program at current contents of the PC.
>>>C	Continue command; starts program at current contents of the PC without initialize functions.
>>>N	Single-steps the program after the PC is loaded.
>>>B	Boot command; boots from device selected by front panel device switch.

Command Description

Format:

```
B [QUALIFIER]<SP>[DDCU]<CR>
```

Qualifiers:	/X	Inhibit running of microverify.
	/hex number	Stores boot control flags in R5.
	DD	Device
	C	Adapter code
	U	Unit number

Examples:

>>>B DDCU	Boot device specified by DDCU.
>>>B/X DDCU	Boot device specified by DDCU and inhibit microverify.
>>>B/n DDCU	Pass four-digit hex number to R5 and boot device specified by DDCU.

>>>D/G/L F 1000	Stores 1000 in PC.
>>>D/P 1000 001234EF	Stores longword of code in 1000.
>>>E/I 25	Examines cache disable register.
>>>I	Performs processor initialize.
>>>B/10/X DMA0	Boots diagnostic supervisor from DMA0, without microverify.

>>>X Binary load/unload command; reserved for use by manufacturing for automated test device (APT) that communicates with the console to transfer data between itself and memory.

Binary Load:

```
>>>X<SP>[ADDRESS]<SP><0'COUNT><CR><CHKSUM1>[DATA]<CHKSUM2>
```

Address	Starting address of the load
Count	Number of bytes to be transferred (unsigned 30-bit hex number, bit <31> is a zero)
CHKSUM1	Two's complement checksum of the command string
Data	Bytes of binary data
CHKSUM2	Two's complement checksum of the data

Binary Unload:

```
>>>X<SP>[ADDRESS]<SP><1'COUNT><CR><CHKSUM>
```

Address	Starting address of the unload
Count	Number of bytes to be transferred (unsigned 30-bit hex number, bit <31> is a one)
CHKSUM	Two's complement checksum of the command string

CONSOLE COMMANDS (CONT)

CONSOLE COMMAND ERROR CODES

Code	Description
?2Ø	Memory examine or deposit failed: access violation (ACV), translation not valid (TNV), machine check, bus error, TB parity error, CCS/WCS parity error.
?11	Error in accessing IPR or PSL
?3Ø	Checksum error on APT load or unload
?33	Attempt to boot from unrecognized device (DM, DL, DD, or DB)
?34	Controller not A, B, C, or D in BOOT command

CONSOLE HALT CODES

Code	Description
Ø1	Test console command executed
Ø2	CTRL/P halt or single macroinstruction mode (>>>N)
Ø4	Interrupt stack not valid
Ø5	Double bus write error halt
Ø6	Processor halt instruction executed (>>>H)
Ø7	Vector <1:Ø> = 3, halt at vector
Ø8	Vector <1:Ø> = 2, WCS disabled or not present
ØA	Change mode instruction executed on interrupt stack
ØB	Change mode instruction executed, vector <1:Ø> not = Ø

BOOT, POWER-UP, AND INITIALIZATION HALT CODES

Code	Description
Ø6	Halt instruction on console boot command, boot ROM, or VMB.EXE failed
11	Power-up, cannot find RPB, FPS1 at RESTART/HALT
12	Power-up, warm start flag false FPS1 at RESTART/HALT
13	Power-up, cannot find good 64KB of memory → E/G A
14	Power-up and boot, bad or nonexistent boot ROM
15	Power-up, cold start flag set during boot subroutine
16*	Power-up halt FPS1 at HALT position
FF	Microverify test failure

R10 gives base address of a load ca

*Normal halt.

MICROVERIFY ERROR CODES

Code	PC+2	Test Name/Error Message
'@'		BBUS, WBUS test
	ØØØ	Bad bit in DREG or SUPROT
	ØØ1	Bad bit in RBUS or WBUS
'C'		MBUS test
	Ø31	Bad bit in QREG
	Ø32	Bad bit in MBUS
'E'		Scratch pad bit test
	Ø51	Error clearing RTEMP
	Ø52	Error filling RTEMP with ones
	Ø54	Error clearing GPR
	Ø57	Error filling GPR with ones
	Ø58	Error clearing IPR
	Ø5B	Error filling IPR with ones
	Ø5D	Error clearing MTEMP
	Ø5E	Error filling MTEMP with ones
'F'		MTEMP explicit address test
	Ø61	Error addressing MTEMPØ
	Ø62	Error addressing MTEMP1
	Ø64	Error addressing MTEMP2
	Ø67	Error addressing MTEMP4
	Ø68	Error addressing MTEMP8
'I'		RTEMP explicit address test
	Ø91	Error addressing RTEMPØ
	Ø92	Error addressing RTEMP1
	Ø94	Error addressing RTEMP2
	Ø97	Error addressing RTEMP4
	Ø98	Error addressing RTEMP8
'J'		IPR explicit address test
	ØA1	Error addressing IRPØ
	ØA2	Error addressing IPR1
	ØA4	Error addressing IPR2
	ØA7	Error addressing IPR4
	ØA8	Error addressing IPR8
'L'		GPR explicit address test
	OC1	Error addressing RØ
	OC2	Error addressing R1
	OC4	Error addressing R2
	OC7	Error addressing R4
	OC8	Error addressing R8
	OCE	Error addressing dual port

MICROVERIFY ERROR CODES (CONT)

Code	PC+2	Test Name/Error Message
'O'		XB/IR/OSR bit test
	OF1	Error in XB<31:Ø>
	OF2	Error in XB<63:32>
	OF4	Error in IR
	OF7	Error in OSR
'Q'		Source XB PC increment test
	111	Error sourcing one byte from XB
	112	Error sourcing 2 bytes from XB or Incrementing PC by 1
	114	Error sourcing an unaligned longword or incrementing PC by 2
	117	Error incrementing PC by 4
'R'		RNUM/DSIZE test
	121	Error reading DSIZE ROM operand 1
	122	Error loading/reading RNUM
	124	Error reading DSIZE ROM operand 2
	127	Error loading/reading RNUM
	128	Error reading DSIZE ROM operand 3
	12B	Error loading/reading RNUM
	12D	Error reading DSIZE ROM operand 4
	12E	Error loading/reading RNUM
'T'		RNUM/DSIZE test continued
	141	Error reading DSIZE ROM operand 5
	142	Error loading/reading RNUM
	144	Error reading DSIZE ROM operand 6
'X'		Cache parity error test
	181	Failed to get cache parity error
	182	Bad machine check error summary register
	184	Bad cache error register
'['		TB parity error test
	1B1	Failed to get group Ø TB parity error
	1B2	Bad TB group parity error register
	1B4	Bad machine check error summary register
	1B7	Failed to get group 1 TB parity error
	1B8	Bad TB group parity error register
	1BB	Bad machine check error summary register
']'		Control store parity error test
	1D1	Failed to get control store parity error
	1B2	Error in control store parity error
'^'		Cache test
	1E1	Error filling cache with ones. Location not initially = Ø
	1E2	Error filling cache with ones. Unable to write ones

BOOT58 COMMANDS

The following is a description of the commands that can be entered to the standalone BOOT58 program. They are listed in alphabetical order.

BOOT

B[device-name]

Bootstraps the system from the specified device. If you omit the device name, the system is bootstrapped using the default bootstrap command procedure (DEFBOO.CMD). Note that you cannot enter the name of a command procedure to the BOOT command and you cannot specify this command within a command procedure.

DEPOSIT

D[loc-qual,size-qual] location value

Deposits a value in the specified location. The location is interpreted according to the location and size qualifiers. The location qualifier can be expressed as follows:

/G	general register
/I	internal process register
/P	physical memory

The size qualifier can be expressed as follows:

/B	byte
/W	word
/L	longword

If you do not specify the location and size qualifiers, the default values established by a previous command are used.

See page 112 for input arguments.
see page 238 for load addresses of diagnostics

EXAMINE

E[loc-qual,size-qual] location

Displays the contents of the specified location. The location is interpreted according to the location and size qualifiers. The location qualifier can be expressed as follows:

/G general register
/I internal process register
/P physical memory

The size qualifier can be expressed as follows:

/B byte
/W word
/L longword

If you do not specify the location and size qualifiers, the default values established by a previous command are used.

HELP

Displays the BOOT58 help file at the console terminal. You cannot specify this command within a command procedure.

LOAD

LO file-spec [/START:address]

Loads a file from the bootstrap device into memory, starting at the address specified with the /START qualifier. If you omit the /START qualifier, the file is loaded into memory beginning at the first free address.

START

S value

Transfers control to the value specified. You generally use this command with the LOAD command.

@file-spec

Executes the name of the command procedure specified. You cannot specify a command procedure file-spec of more than six characters, nor can you specify nested command procedures.

CHAPTER 6
RDM AND MICRODIAGNOSTICS

RDM INSTALLATION

RDM HARDWARE

The following hardware is supplied with the indicated RDM options.

Option	Hardware (One Each)	Usage
KC750-CA	RDM module - L0006 Freestanding modem Diagnostic kit Filtered interface cable (70-16921) Modem interface cable (BC05D-25) KC750 Options User Guide (EK-KC750-UG)	U.S.
KC750-DA	RDM module - L0006 Diagnostic kit Filter interface cable (70-16921) KC750 Options User Guide (EK-KC750-UG) (Modem and cable supplied by customer)	Foreign

PREINSTALLATION

The following items must take place before installation of the RDM.

1. The customer's system configuration must be evaluated and a system configuration worksheet supplied to the DIGITAL Diagnostic Center (DDC).

2. The customer (U.S. only) must furnish the telephone company with the following information on the modem to be used.

   a. Model number

   b. Manufacturer's number

   c. Ringer equivalence number*

   d. FCC registration number*

   e. Voice jack direct-connect type receptacle

   f. Telephone number of line/RJ11C if already installed

   *To be supplied by the DIGITAL district office.

INSTALLATION PROCEDURE

The following is the procedure for installing the RDM module and hardware.

1. Perform an orderly shutdown of the operating system, then power down the system by turning the console power key switch to OFF.

2. The RDM module is supplied with the remote port baud rate set to 300 baud by jumpers on the module. Earlier modules had a switchpack in the location currently occupied by the jumpers.

   The baud rate is selected by jumpers W4, W5, W6, W7 (or by switch positions E190-1 through 4) as shown below.

   In Europe, remove jumper W-3 on the opposite end of the module from the baud rate jumpers.

Baud Rate	W4 or E190-1	W5 or E190-2	W6 or E190-3	W7 or E190-4
300	On	On	Off	On
400	Off	On	Off	On
600	On	Off	Off	On
1,200	Off	Off	Off	On
2,400	On	On	On	Off
3,600	Off	On	On	Off
4,000	On	Off	On	Off
4,800	Off	Off	On	Off
7,200	On	On	Off	Off
9,600	Off	On	Off	Off
19,200	On	Off	Off	Off

3. Install the L0006 module in slot 6 of the extended hex backplane.

   Refer to the RDM and modem cabling illustrations for the next steps.

4. On the pin side of the CPU backplane, carefully move the following cables from the left side of slot 6 (odd numbered pins) to the right side of slot 6 (even numbered pins), keeping the same vertical placement.

   a. Front panel cable (row A)

   b. TU58 cable (row B)

   c. Console cable (row C)

CAUTION

Make sure there are two open pins between the console cable and the console baud rate plugs.

RDM INSTALLATION (CONT)

5. Plug the RDM filtered cable assembly (70-16921) into the bottom 22 pins of set C on the right side of slot 6. Attach the assembly to the backplane so that pin 1 is on top and pin 22 is on the bottom (pin C06-92).

   CAUTION

   When installing this cable, be sure that it is connected properly and that the two bottom pins C06-93 and C06-94 are left unconnected.

6. Attach the filtered cable assembly to the I/O port panel using two 6-32 Kepnuts provided with the cover plate.

7. Attach the asset tag around the filtered cable using the tie strap and tighten it to prevent movement of the tag.

8. Plug the modem cable cinch plug into the RDM filtered cable connector in the I/O port panel and secure.

9. Route the modem cable as necessary and connect it to the modem. The U.S. modem is a freestanding unit that requires a 115 Vac power source.

   CAUTION

   Modem ac power must not come from the VAX-11/750 internal power distribution system. This violates UL regulations and cabinet power integrity.

INSTALLATION VERIFICATION

Use the following procedure to verify RDM operation.

1. Mount diagnostic software and scratch media.

2. Power up the system by turning the boot device switch to A, the power-on action switch to HALT, and the power key switch to LOCAL.

3. Observe the RDM power-up self-test. After power-up, the fault indicator should turn on for about ten seconds. Inspect the installation if the fault indicator does not turn on or does not turn off, or if there is no console printout or error message with the SYS, ROM, or RAM code. If the installation is correct, remove the RDM. Install another RDM or return the cables to their original positions.

If the power-up test is successful, the console terminal prints the following.

```
% %
00000000 16
>>>
```

4. Perform the RDM installation tests described later in this chapter yourself or ask the DDC by telephone to perform an installation verification.

5. For remote testing by the DDC, turn the console power key switch to REMOTE.

6. Check with the DDC from the console terminal or by telephone for results of the verification.

RDM REMOVAL

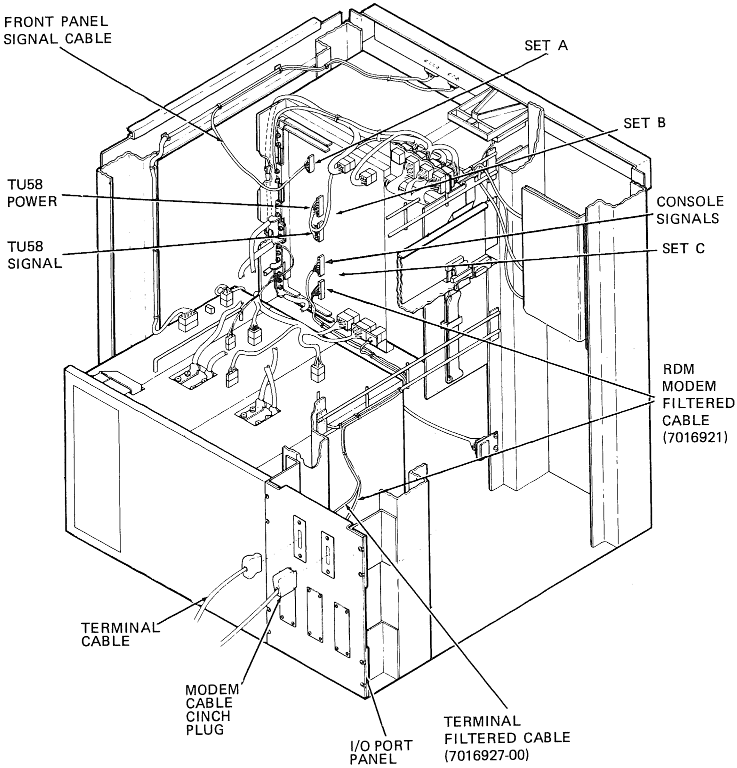
Removing the RDM is the reverse of the installation procedure. If removal is temporary, only installation steps 1 through 4 need to be performed in reverse. The modem cables and connectors are left in place.

CAUTION
Always power down the system before installing or removing the RDM.

INSTALLATION OUTSIDE THE UNITED STATES

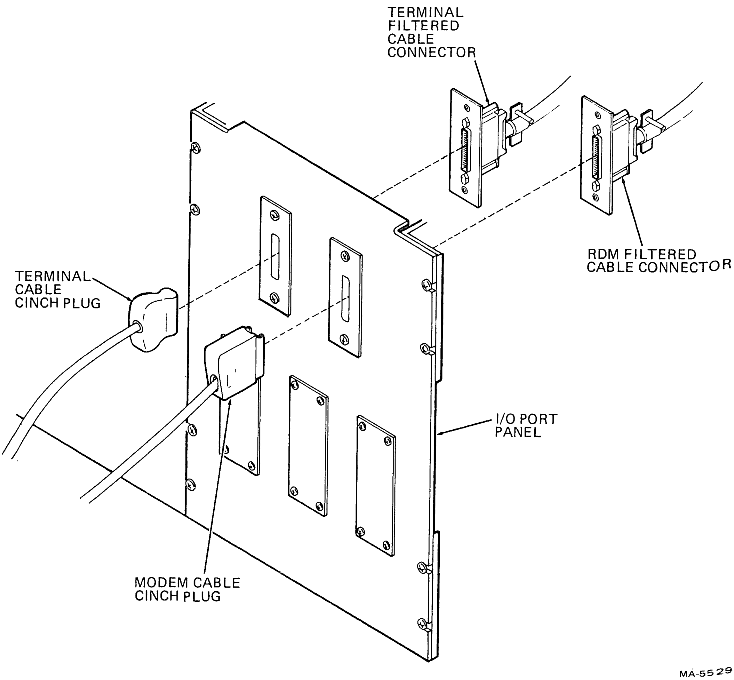
DIGITAL does not provide modems with RDM options outside the United States. The standard practices and procedures of a given country must be followed to obtain a modem there. Refer to the KC750 Options Installation Guide (EK-KC750-IN) for specifications on modems that work with the RDM.

RDM CABLING

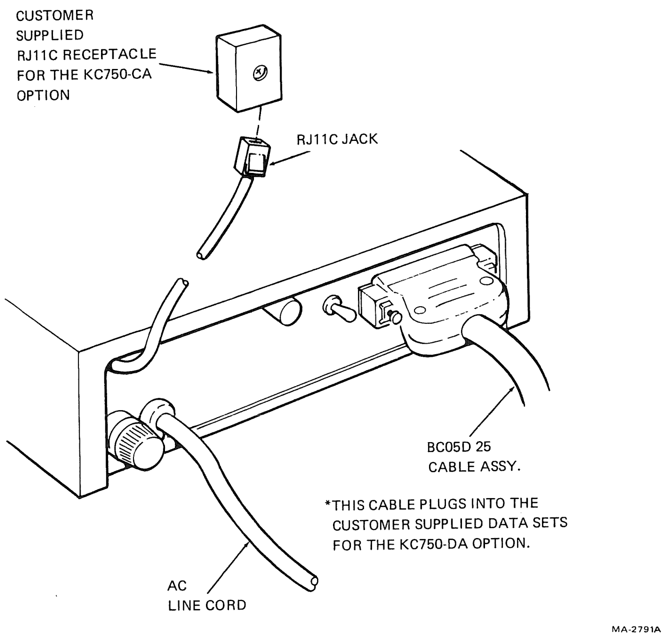


TK-9872

FILTERED CABLE INSTALLATION



MODEM CABLING



RDM INSTALLATION (CONT)

RDM/MODEM SIGNALS

RDM Pin	70-16921 Cable Conn. Pin	EIA (RS232-C) Signal Name	EIA Abbr.	EIA Circuit
C0671	1	Protective Ground	GND	AA
C0672	7	Signal Ground	GND	AB
C0678	2	Transmitted Data	TXD	BA
C0680	3	Received Data	RXD	BB
C0690	4	Request to Send	RTS	CA
C0688	5	Clear to Send	CTS	CB
C0686	6	Data Set Ready	DSR	CC
C0682	20	Data Terminal Ready	DTR	CD
C0692	22	Ring Indicator	RI	CE
C0684	8	Carrier Detect	CD	CF
	(Unused)	Force Busy	FB	CN

SELECTED CPU BACKPLANE SIGNALS

Pin	Signal Name
Local Terminal	
C0624	Ground
C0634	RDM terminal serial out to TU58 serial in
C0632	RDM terminal serial in to TU58 serial out
C0645	Console baud rate A L
C0646	Console baud rate B L
C0649	Console baud rate C L
C0650	Console baud rate D L
TU58	
B0686	TU58 serial out (signals between TU58 and RDM)
B0688	TU58 serial in (signals between TU58 and RDM)
B0685	EIA TU serial out L (signals between RDM and CPU)
B0687	EIA TU serial in L (signals between RDM and CPU)
Front Panel	
A0601	The front panel cable is plugged into set A (top half) with cable pin 1 on top connecting to A0601.
RDM Specific	
C0673	RDM SA CLK L (drive)
C0675	SA ST/SP L (drive)
C0674	RDM RESET L (receive)
C0681	RDM MATCH PULSE (receive)

RDM INSTALLATION TESTS

The following series of tests should be run in the order given.

VAX CPU TEST

To verify CPU operation after the RDM has been installed, mount the TU58 tape that contains the EVKAA diagnostic and type the following.

```
D/I 25 1<RET>
B<RET>
```

Cache is disabled and the tape motion indicator for the TU58 should blink or stay on for about two minutes while EVKAA is loaded. The diagnostic runs repeatedly until it is stopped by CTRL/P. The following dialogue summarizes the procedure.

```
>>>D/I 25 1<RET>
>>>B<RET>
%%

EVKAA - 4.Ø done!
EVKAA - 4.Ø done!
CTRL/P
ØØØØ nnnn Ø2
>>>
```

In this dialogue, n stands for some number. The EVKAA diagnostic should run at least twice before being stopped. (CTRL/P may have to be typed more than once to stop EVKAA.)

Next, use the VAX firmware to write and read test data with VAX memory locations. Type the following to load memory.

```
>>>D/P/L Ø Ø<RET>
>>>D + FFFFFFFF<RET>
>>>D + AAAAAAAA<RET>
>>>D + 55555555<RET>
>>>D 2FFFC 12345678<RET>
>>>D + 87654321<RET>
```

Then, type the following to examine memory.

```
>>>E Ø<RET>
                  P     ØØØØØØØØ     ØØØØØØØØ
>>>E<RET>
                  P     ØØØØØØØ4     FFFFFFFF
>>>E<RET>
                  P     ØØØØØØØ8     AAAAAAAA
>>>E<RET>
                  P     ØØØØØØØC     55555555
>>>E 2FFFC<RET>
                  P     ØØØ2FFFC     12345678
>>>E<RET>
                  P     ØØØ3ØØØØ     87654321
```

If the examined data does not agree with the deposited data, the test has failed.

VAX MEMORY BUS TEST

This test verifies the ability of the RDM hardware to read and write data with VAX memory. (The VAX CPU test should be run before this test.)

To run the test, first enter the RDM console state as follows.

```
>>>CTRL/D
RDM>
```

Examine VAX memory.

```
RDM>E Ø<RET>
                      P     ØØØØØØ     ØØØØØØØØ
RDM>E<RET>
                      P     ØØØØØ4     FFFFFFFF
RDM>E<RET>
                      P     ØØØØØ8     AAAAAAAA
RDM>E<RET>
                      P     ØØØØØC     55555555
RDM>E 2FFFC<RET>
                      P     Ø2FFFC     12345678
RDM>E<RET>
                      P     Ø3ØØØØ     87654321
```

Deposit to memory.

```
RDM>D Ø FFFFFFFF<RET>
RDM>D + Ø<RET>
RDM>D 2FFFC 55AA55AA<RET>
RDM>D + AA55AA55<RET>
```

Reexamine memory.

```
RDM>E Ø<RET>
                      P     ØØØØØØ     FFFFFFFF
RDM>E<RET>
                      P     ØØØØØ4     ØØØØØØØØ
RDM>E 2FFFC<RET>
                      P     Ø2FFFC     55AA55AA
RDM>E<RET>
                      P     Ø3ØØØØ     AA55AA55
RDM>
```

If the data read from memory is not the same as data written into memory, the test has failed.

VAX CONTROL STORE PARITY CHECK

This test exercises the RDM's ability to know whether the VAX clock is running, to stop the VAX clock, and to check VAX control store parity. (The VAX memory bus test should be run before this test.)

If the RDM is working properly, it refuses to check VAX control store parity while the VAX clock is running. In order to see if the RDM can detect the VAX clock, ask it to do a parity check while the VAX clock is running.

```
RDM>PAR<RET>
    CLK RUNNING
RDM>
```

Check that the RDM can stop the VAX clock.

```
RDM>STO<RET>
    CLK STOPPED CSAD Ø96n      NEXT Ø96n
```

In this dialogue, n stands for some number.

Now test the RDM's ability to check the VAX control store parity. (An error has been preset at microaddress 17FD.)

```
RDM>PAR Ø<RET>
PARITY ERROR CSAD 17FD
RDM>
```

If any other address appears in the dialogue, a parity error has been detected in that address of the VAX control store.

MICROBREAK POINT AND TRACE

This test exercises the RDM logic that stops the VAX clock when the CPU executes the microinstruction at a preset microaddress called the microbreak point. The test also exercises the RDM's ability to trace VAX control store addresses. A trace is a listing of the CS addresses of the last 65 microinstructions executed, in the order of their execution. (The VAX control store parity check should be run before this test.)

First, initialize the CPU as follows.

```
RDM>RET<RET>
%%

ØØØØØØØØ 16

>>>CTRL/D
RDM>
```

Set the microbreak point.

```
RDM>SE A2B<RET>
RDM>
```

This microbreak point happens to be the address of the CPU's carriage return microinstruction. When the CPU does a carriage return, the RDM should then stop the CPU clock.

Return to the VAX console state. The CPU does a carriage return in the process.

```
RDM>RET/D<RET>
CLK STOPPED CSAD ØA2B      NEXT nnnn
```

Where n is some hexadecimal number and NEXT indicates the address of the next instruction to be executed.

Next check that the RDM does a VAX control store trace.

```
RDM>TR<RET>

        CSAD    ØA2B    NEXT nnnn
        CSAD    nnnn
        CSAD    nnnn
        ...     ...
        CSAD    Ø965
        CSAD    Ø964
        CSAD    Ø966
        CSAD    Ø964
        CSAD    Ø966
        ...     ...
```

Sixty-five microaddresses should be displayed in all. In the above dialogue, an ellipsis (...) shows that a continuation of address printouts takes place. Also, n represents some hexadecimal number and NEXT is the address of the next microinstruction to be executed.

Note that the address of the last microinstruction executed (ØA2B) is the address of the carriage return microinstruction. The addresses Ø964 and Ø966 are the loop in which the CPU waits for a carriage return.

MICRODIAGNOSTIC RUN TEST

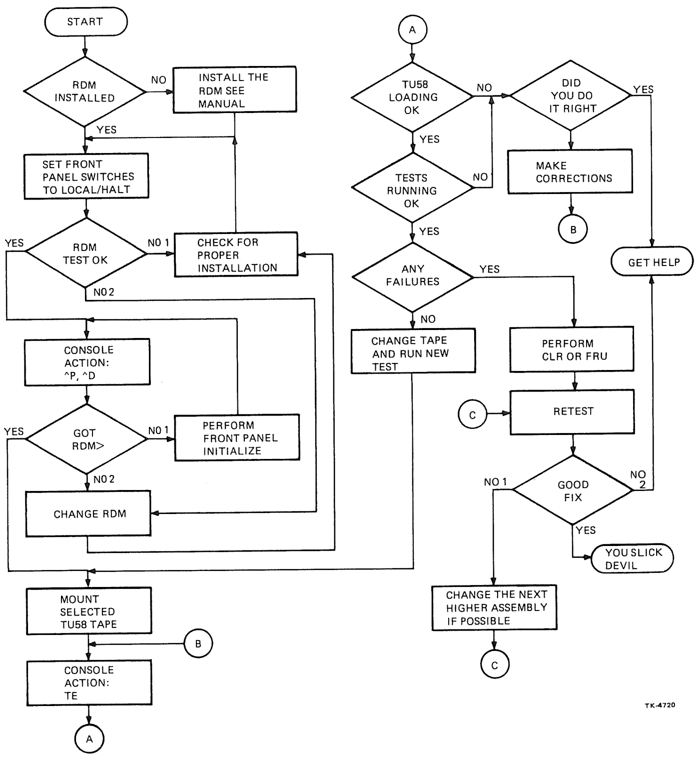
The most RDM hardware is tested when a microdiagnostic is run on the VAX. Before beginning the test, make sure the system is in the RDM console state. The microdiagnostic used here is the data path module (DPM) microdiagnostic ECKAA. Mount the TU58 tape with this program in the VAX front panel. Use the following dialogue to load the microdiagnostic into RDM memory and run it twice.

```
RDM>TE/C<RET>
MIC>DI PA:2<RET>

ECKAA-VØ2.ØØ DPM-VØ2.ØØ
Ø1, Ø2, Ø3, Ø4, Ø5, Ø6, Ø7, Ø8, Ø9, ØA, ØB, ØC, ØD,
ØE, ØF, 1Ø, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A,
1B, 1C, 1D, 1E, 1F, 2Ø, 21, 22, 23,
END OF PASS Ø1

DPM-VØ2.ØØ
Ø1, Ø2, Ø3, Ø4, Ø5, Ø6, Ø7, Ø8, Ø9, ØA, ØB, ØC, ØD,
ØE, ØF, 1Ø, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A,
1B, 1C, 1D, 1E, 1F, 2Ø, 21, 22, 23,
END OF PASS Ø2

MIC>RET<RET>
RDM>
```

Note that the MIC> prompt is displayed when the system is in the microdiagnostic state.

If the printout is not the same as the one just given and the test has been performed correctly, replace the RDM with a spare. If a spare is not available, remove the existing RDM and return the backplane cables to their original positions.

RDM TROUBLESHOOTING FLOW



RDM CONTROL KEY FUNCTIONS

Key	Function
CTRL/D	Enter RDM console command mode
CTRL/P	Enter CPU console command mode
CTRL/U	Abort current command line
CTRL/O	Inhibit printouts during diagnostic test
CTRL/R	Retype current command line
CTRL/C	Cancel current function (halts REP or R command)
CTRL/S	Disable CPU output to active terminal
CTRL/Q	Enable CPU output to active terminal

RDM CONSOLE COMMANDS

Command	Function
RDM>E	Examine - uses the following command switches:

E/B <ADDRESS>	Data size is byte
E/W <ADDRESS>	Data size is word
E/L <ADDRESS>	Data size is longword
E/C <ADDRESS>	Examines RDM status registers and RAM addresses in the range of 8000 to 8FF8
E/N or E +	Examines next address
E -	Examines previous address
E *	Uses data last examined or deposited as the address

Command	Function
RDM>D	Deposit - uses the following command switches:

D/B <ADDRESS> <DATA>	Data size is byte
D/W <ADDRESS> <DATA>	Data size is word
D/L <ADDRESS> <DATA>	Data size is longword
D/N or D + <DATA>	Deposits data in next address
D - <DATA>	Deposits data in previous address
D * <DATA>	Uses data last examined or deposited as the address

Command	Function
RDM>INI	Initialize - simlates a power-fail sequence to recover CPU from a hung condition. Asserts ACLO and DCLO signals. (May be used with R command to troubleshoot power-fail recovery problems.)
RDM>RET	Return - switches system from RDM console/command mode to CPU program mode. Stop-on-micromatch, if set, is disabled.
RDM>RET/D	Return/D - switches system from RDM command mode to RDM control mode. System is at prompt level of the program but, unlike CPU program mode, CTRL/D returns the system to RDM console mode whether the CPU is running or not. Stop-on-micromatch, if set, remains enabled.

RDM CONSOLE COMMANDS (CONT)

Command	Function
RDM>TA	Talk - enables talk mode between local and remote terminal during an RD session.
RDM>SH	Show - displays 8085 CPU operating states.
RDM>SH/V	Show version - displays current revision level of the firmware running in the RDM.
RDM>REP	Repeat - continually repeats last command given until CTRL/C is typed. (CTRL/O stops output to terminal.)
RDM>R <COMMAND>	Repeat next - continually repeats specified command until CTRL/C is typed. (CTRL/O stops output to the terminal.)
RDM>PAR <ADDRESS>	Parity scan - runs CCS parity check from the specified starting address. Stops at preset error address 17FD if no error is found. CPU clock must be stopped to use this command. WCS may be checked by specifying addresses 2000 through 2400 (WCS must be loaded after power-up).
RDM>STO	Stop - stops CPU clock.
RDM>STE	Step - steps through one microinstruction on an M clock tick. Displays address of micro-instruction currently latched in CPU and address of the next microinstruction.
RDM>STE/T	Step tick - advances microinstruction by one B clock tick. Displays current address latched in CPU and data to be latched on the next M clock tick.
RDM>CON	Continue - restarts the CPU clock.
RDM>TR	Trace - displays in reverse order the control store addresses stored in the RDM diagnostic control store (DCS). CTRL/C or CTRL/D halts display of the 65 stored addresses. CPU clock must be stopped to use this command.
RDM>UA <ADDRESS>	Microaddress - halts the CPU and latches selected microaddress in CCS latches. CPU then restarts and checks parity on the address data. (May be used with R command for scope loop.) Stop-on-micromatch function, if set, is disabled.

RDM CONSOLE COMMANDS (CONT)

Command	Function
RDM>UA/C <ADDRESS>	Microaddress/C - temporarily places selected microaddress on CSS address lines until next M clock tick. Used to isolate failures in the CCS latching mechanism. CPU program flow is not changed. Stop-on-micromatch function, if set, is disabled.
RDM>SE <ADDRESS>	Set - enables stop-on-micromatch function. CPU clock is stopped when the CCS address bus equals the selected address. TR command is then used to trace path taken by the CPU to reach that point in the CCS.
RDM>CL	Clear - disables stop-on-micromatch function.
RDM>LIN	Link - used to create executable control file in the RDM RAM with the LNK> prompt. CTRL/C exits file or PER command causes it to execute until CTRL/C is given. Link list is destroyed if system is powered down or if the RET, RET/D, or any TE command is given.
RDM>PER	Perform - causes link list to be continuously executed until CTRL/C is given.
RDM>LO <FILENAME><SP>[ADDRESS]	Load - reads the specified file from TU58 to main memory, starting the load at the specified address. May be used to load files if CPU does not have the functionality to do so, or if the file is not properly hooked to an RT11 boot block. RDM needs only the CPU base clock to achieve the load.
RDM>TE	Test - loads the microdiagnostic monitor (MICMON) from TU58 to RDM RAM and runs micro-diagnostic tests. Testing is canceled if an error is detected or CTRL/C is typed. MICMON then switches to the command state, giving the MIC> prompt.
RDM>TE/C	Test-command - loads MICMON as for TE but places it in the command state with the MIC> prompt.
RDM>TE <FILENAME>	Test-file - loads user-specified program into RDM RAM and runs it. The file name must consist of six characters and the file type of three. For example: RDM>TE MICMON.TST<CR>

RDM CONSOLE ERROR CODES

Code	Description
TU58 Function Errors	
TAP:01	Tape UART - Device timeout
TAP:02	Tape UART - Error from UART
TAP:03	Tape UART - Data set ready dropped
TAP:04	Tape UART - Receive overflow
TAP:05	Tape checksum error received
TAP:06	Tape byte count maximum exceeded
TAP:07	Tape no end packet (invalid operation code)
TAP:08	Tape invalid packet received
TAP:09	Tape file not found
TAP:12	Tape directory error
TAP:13	Tape flag received (not command or data)
TAP:14	Tape read length error (not all records fit)
TAP:C9	Tape bad record number
TAP:D0	Tape bad operation code
TAP:DF	Tape motor stopped
TAP:E0	Tape block not found
TAP:EF	Tape data check error
TAP:F5	Tape write protocol error
TAP:F7	Tape cartridge not present
TAP:F8	Tape bad unit number
TAP:EE	Tape partial operation (end of medium)
TAP:FF	Tape diagnostic failure
Console Terminal Errors	
TRM:01	Terminal UART - Device timeout
TRM:02	Terminal UART - Error from UART
TRM:03	Terminal UART - Data set ready dropped
TRM:04	Terminal UART - Receive overflow
TRM:0A	Terminal CTRL/C received
TRM:0B	Terminal command input buffer overloaded
TRM:0C	Terminal CTRL/D received
TRM:0D	Terminal command input larger than buffer
TRM:0E	Terminal remote line CRC error

RDM CONSOLE ERROR CODES (CONT)

Code	Description
CMI Errors	
CMI:ØØ	Nonexistent memory
CMI:Ø1	Corrected read data
CMI:Ø2	Read data substitute
General Errors	
SYNTAX ERROR	Error entering console command
INVALID COMMAND	RDM does not recognize command
ROM	ROM failed RDM power-up self test
RAM	RAM failed RDM power-up self test
RDM:1Ø	Operation already in progress
RDM:11	Invalid operation code in macro
REM:Ø1	Remote UART - Device timeout
REM:Ø2	Remote UART - Error from UART
REM:Ø3	Remote UART - Data set ready dropped
REM:Ø4	Remote UART - Receive overflow
CPU:Ø1	CPU UART - Device timed out
CPU:Ø2	CPU UART - Error received from UART
CPU:Ø3	CPU UART - Data set ready dropped
CPU:Ø4	CPU UART - Receive overflow

VAX-11/750 DIAGNOSTICS

VAX-11/75Ø diagnostic programs are available under the following levels of operation.

Level 1 - VMS operating system based diagnostic programs that run without the diagnostic supervisor.

UETP, ERRLOG, SPEAR, SDA

Level 2R - Diagnostic supervisor based programs - any diagnostic that requires a VMS based driver.

Peripheral diagnostics not supported by the supervisor in the standalone mode

System diagnostic control program

Level 2 - Diagnostic supervisor based programs that can be run on-line under VMS or in the standalone mode.

Bus interaction program

Formatter and reliability-level peripheral diagnostics

Level 3 - Diagnostic supervisor based programs that can only be run in the standalone mode.

Functional-level peripheral diagnostics
Repair-level peripheral diagnostics
CPU cluster diagnostics

Level 4 - Standalone macrodiagnostics that run without the supervisor.

Hardcore instruction test (tests the basic CPU functions necessary to run the supervisor)

Level 5 - Console based diagnostics that run only in the standalone mode.

Microdiagnostics
Console operating program
Microverify program

VAX-11/750 DIAGNOSTICS (CONT)

Device	Name	Level	Description	TU58 Cassette Number
	ECSAA	3	Diagnostic Supervisor	6,17
	ECKAA	5	Microdiagnostic Monitor	1,2
KA75Ø	ECKAB	5	Micro Data Path Module (DPM)	1
	ECKAC	5	Micro Memory Interface (MIC)	2
	ECKAL	4	Cache/Translation Buffer	5
	EVKAA	4	VAX Hardcore Instruction (Bootable)	7,17,32,37
	EVKAB	2	VAX Architectural Instruction	8,32
(FP75Ø)	EVKAC	2	VAX Floating-Point Instruction	8
	EVKAD	2	VAX Compatability Mode Instruction	8
	EVKAE	3	VAX Privileged Architectural Instruction	8
KC75Ø	ECKAF	5	VAX RDM Microdiagnostic	1,2
MS75Ø	ECKAM	3	VAX-11/75Ø Memory	5
	EVKAM	2R	VAX Memory User Mode	8
TU58	ECKAX	3	VAX-11/75Ø Cluster Exerciser	5
	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,39
RH75Ø	ECCAA	3	VAX-11/75Ø MASSBUS Adapter (MBA)	6,17,33
(DW75Ø)	ECCBA	3	VAX-11/75Ø UNIBUS Interface (UBI) and Second UNIBUS (SUB)	6,17,33
RKØ7	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
	EVRAC	2	VAX Disk Formatter	9
	EVREA	3	VAX RK611 Controller Part A	11
	EVREB	3	VAX RK611 Controller Part B	11
	EVREC	3	VAX RK611 Controller Part C	12
	EVRED	3	VAX RK611 Controller Part D	12
	EVREE	3	VAX RK611 Controller Part E	12
	EVREF	3	VAZ RKØ6/Ø7 Functional Part 1	13
	EVREG	3	VAX RKØ6/Ø7 Functional Part 2	13

NOTE: Refer to the microfiche VAX diagnostic listing index (EVNDX) for the current diagnostic revision and TU58 cassette numbers.

Device	Name	Level	Description	TU58 Cassette Number
RLØ2	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
	EVRFA	3	RLØ2 Subsystem Functional	22
RMØ3/	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
RMØ5	EVRAC	2	VAX Disk Formatter	9
	EVRDA	3	VAX RMØ3/Ø5/8Ø Diskless	14
	EVRDB	3	VAX RMØ3/Ø5 Functional	14
RM8Ø	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
	EVRDA	3	VAX RMØ3/Ø5/8Ø Diskless	14
	EVRGA	3	RM8Ø Formatter	16,39
	EVRGB	3	RM8Ø Functional	16
RPØ4/	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
RPØ5/	EVRAC	2	VAX Disk Formatter	9
RPØ6	EVRBA	3	VAX RPØ4/Ø5/Ø6 Functional	22
	EVRCA	3	VAX DCL/RPØ4/Ø5/Ø6 Repair	22
RPØ7	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
	EVRAC	2	VAX Disk Formatter	9
	EVRHA	3	RPØ7 Front End	25
	EVRHB	3	RPØ7 Functional	25
	EVRHC	3	RPØ7 Dual Port Controller	25
RXØ2	EVRAA	2	VAX RP/RK/RM/RX/TU58 Data Reliability	9,38,39
	EVRAC	2	VAX Disk Formatter	9
	EVRIA	3	VAX RXØ2 Subsystem Repair	22
TS11	EVMAA	2	VAX Magtape Data Reliability	15
	EVMAD	3	VAX TS11 System Repair	15
TE16/	EVMAA	2	VAX Magtape Data Reliability	15
TU45/	EVMAB	3	VAX TE16,TU45/77 Drive Function	15
TU77	EVMAC	3	VAX TE16,TU45/77 Control Logic	15
TU78	EVMAA	2	VAX Magtape Data Reliability	15
	EVMAE	3	TM78/TU78 Control Logic	24

NOTE: Refer to the microfiche VAX diagnostic listing index (EVNDX) for the current diagnostic revision and TU58 cassette numbers.

MICROMONITOR (MICMON) COMMANDS

Command	Function
MIC>RET	Return - system leaves MICMON and returns to RDM command mode.
MIC>DI (QV) QUICK VERIFY	Diagnose - performs TU58 diagnostics from the RDM diagnostic control store (DCS).
MIC>DI PA:<PASS-COUNT>	Diagnose - initializes program control flags and starts execution loop of test programs in DCS for specified number of passes.
MIC>DI TE:<TEST-NUMBER> or MIC>DI TE:<TEST-NUMBER> <TEST-NUMBER>	Diagnose - executes specified test or range of tests. <SP>
MIC>DI TE:<TEST-NUMBER> CO	Diagnose - continually executes specified test until stopped by an error or CTRL/C.
MIC>DI DM	Diagnose - runs diagnostic module ECKAF.EXE.
MIC>CO	Continue - restarts DCS test following error or CTRL/C.
MIC>LO	Loop - sets loop flag and loops on a program detected and reported error.
MIC>SE FL xx	Set flag - sets specified program control flag. (page 149)
MIC>CL FL xx	Clear flag - clears specified program control flag.
MIC>SH FL	Show flags - displays current states of the program control flags.
MIC>SH VB	Show VBUS - displays current signal states on the visibility bus (VBUS).
MIC>SE SO:<DCS-ADDRESS>	Halts execution of DCS program at specified address.

Command	Function
MIC>SE ST CY	Steps through DCS microinstructions on M clock cycle.
MIC>SE ST TI	Steps through DCS microinstructions, stopping twice in each M clock cycle as a function of the phase clock.

pseudo-instructions are 8085 code.

MIC>SE ST IN [:TEST-PC]

Set step instruction - steps through pseudo instructions in current test. If test-pc is specified, step function starts when the instruction at test-pc is ready to be executed. If test-pc is not specified, stepping begins on the next pseudo instruction of the current test following a loop (LO) or continue (CO) command.

MIC>CL SO: <DCS-ADDRESS>

Clear stop-on-micromatch - DCS address, if specified, generates scope sync pulse on slot 6, pin C81 (18ØØ hex is added to desired address). Pulse occurs with M clock when the current address matches DCS address.

MIC>SE CF: <DCS-ADDRESS> <BIT-NUMBER>

Set control file bit - sets specified control file bit at the specified DCS address (bits <95:88>).

MIC>CL CF: <DCS-ADDRESS> <BIT-NUMBER>

Clear control file bit - clears specified control file bit at the specified DCS address.

MIC>EX <REGISTER> or
MIC>EX <REGISTER:NUMBER>

Examines the following processor registers:

VA	Virtual address register (VAR)	(page 165)
MA	Memory address register (MAR)	
PC	CPU program counter (PC)	(" ")
PB	PC backup register (PCB)	
RT:nn	Rtemp registers Ø through 2F	
MT:n	Mtemp registers Ø through F	
PS	Processor status longword (PSL)	
MD	Memory data register (MDR)	(Page 168)
WD	Write data register (WDR)	
SR:n	Status and control registers Ø through E	
SF	Status flags	
ST	Step counter	

MICROMONITOR PROGRAM CONTROL FLAGS

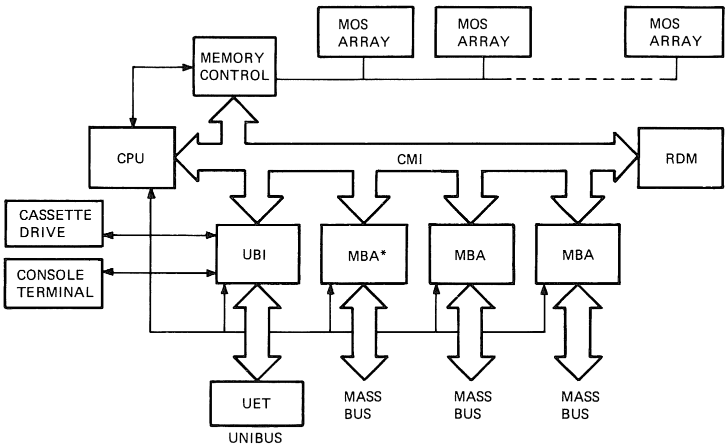
Flag	Function
HALT	Halt on error - returns to the monitor on a program detected error.
LOOP	Loop on error - loops on program detected error. When set by MIC>SE FL LO, the HALT flag must be cleared and the NER flag set for continuous loop. The loop may include pseudo instructions and DCS instructions from the ERRLOOP to the IFERROR instruction in the failing test. If the IB flag is set or a microtrap occurs, the program does not loop on microinstructions in DCS. NER must be clear to inhibit error messages.
NER	No error report - inhibits error messages.
BELL	Bell on error - rings bell on the first, then every fifth occurrence of, an error.
IB	Inhibit burst - when used with LOOP flag, program does not loop only on DCS instructions but loops instead between the ERRLOOP and IFERROR pseudo instructions.
QA	Quality assurance - each test reponds as if an error was detected.
TR	Trace - monitor prints test names and numbers.
~~SA~~ NOT SUPPORTED	Signature analysis - used with signature analyzer to help diagnose faults. Loop occurs on test in progress whether or not an error occurs. Flag provides two sync points on the backplane: Start/stop window, slot 6, pin C75 Clock pulse, slot 6, pin C73 The signature analyzer analyzes test points by displaying a value (signature) if the signal pattern is steady. This value is compared with the value from a known good module to locate failures.

VISIBILITY BUS (VBUS) SIGNALS

Bit Number (Decimal)	Bit Number (Hex)	Signal Name	Print Set Page
ØØ	ØØ	UBIØ3 FORCE TB PE L	UBI15
Ø1	Ø1	UBIØ3 FORCE CACHE PE L	UBI15
Ø2	Ø2	CS HNEXT PAR H	UBI15
Ø3	Ø3	UBIØ3 RTUT DINH L	UBI15
Ø4	Ø4	UBIØ3 BUSF PAR H	UBI15
Ø5	Ø5	UBI15 INT PEND L	UBI15
Ø6	Ø6	UBI15 UB INT GRANT H	UBI15
Ø7	Ø7	UBI12 CON HALT L	UBI15
Ø8	Ø8	MICRO VECTOR 3 H	MICØ4
Ø9	Ø9	MICRO VECTOR 2 H	MICØ4
1Ø	ØA	MICRO VECTOR 1 H	MICØ4
11	ØB	MICRO VECTOR Ø H	MICØ4
12	ØC	MICØ7 GEN DEST INH L	MICØ4
13	ØD	MICØ7 UTRAP L	MICØ4
14	ØE	MICØ4 LATCHED MBUS 15 L	MICØ4
15	ØF	MICØ4 PROC INIT L	MICØ4
16	1Ø	MICØ4 MSRC XB H	MICØ4
17	11	MICØ4 MEM STALL H	MICØ4
18	12	MICØ4 STATUS VALID H	MICØ4
19	13	MICØ5 UB REQ H	MICØ4
2Ø	14	MICØ7 CORR DATA INT L	MICØ4
21	15	MICØ7 WR BUS ERR INT L	MICØ4
22	16	Not used, always Ø	MICØ4
23	17	Not used, always Ø	MICØ4
24	18	DPM17 INSTR FETCH H	DPM21
25	19	DPM17 DO SRVC L	DPM21
26	1A	DPM16 IRD1 H	DPM21
27	1B	DPM14 UVCTR BRANCH H	DPM21
28	1C	DPM14 DISABLE HI NEXT H	DPM21
29	1D	DPM2Ø CS PARITY ERROR H	DPM21
3Ø	1E	DPM14 LD DSR L	DPM21
31	1F	DPM17 PSL CM H	DPM21
32	2Ø	DPM19 ISIZE Ø L	DPM21
33	21	DPM19 ISIZE 1 L	DPM21
34	22	DPM18 DST RMODE H	DPM21
35	23	DPM13 TIMER INT L	DPM21
36	24	DPM19 DSIZE Ø H	DPM21
37	25	DPM19 DSIZE 1 H	DPM21
38	26	DPM11 MCS TMP L	DPM21
39	27	UBI13 MSEQ INIT L	DPM21

CHAPTER 7
BLOCK DIAGRAMS

VAX-11/750 BASIC DIAGRAM

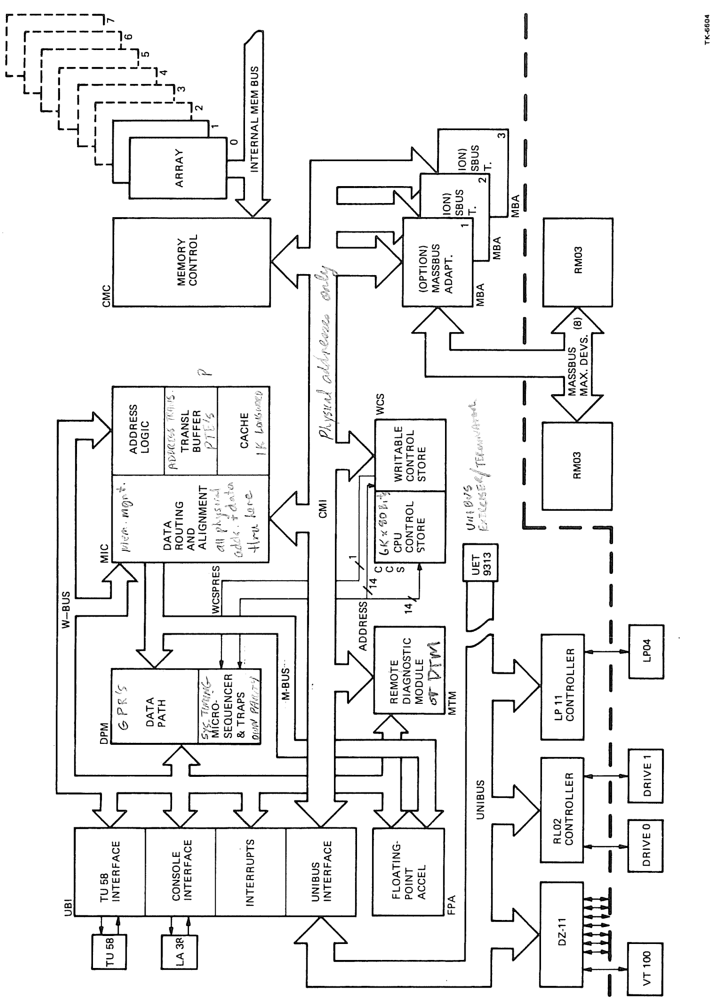


*SECOND UNIBUS INTERFACE IS OPTIONAL

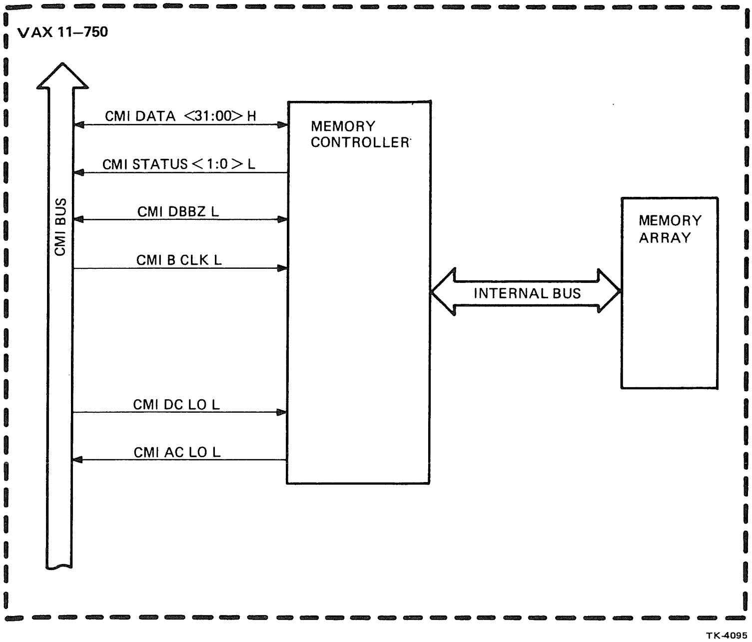
TK-3871

VAX-11/750 SYSTEM DIAGRAM

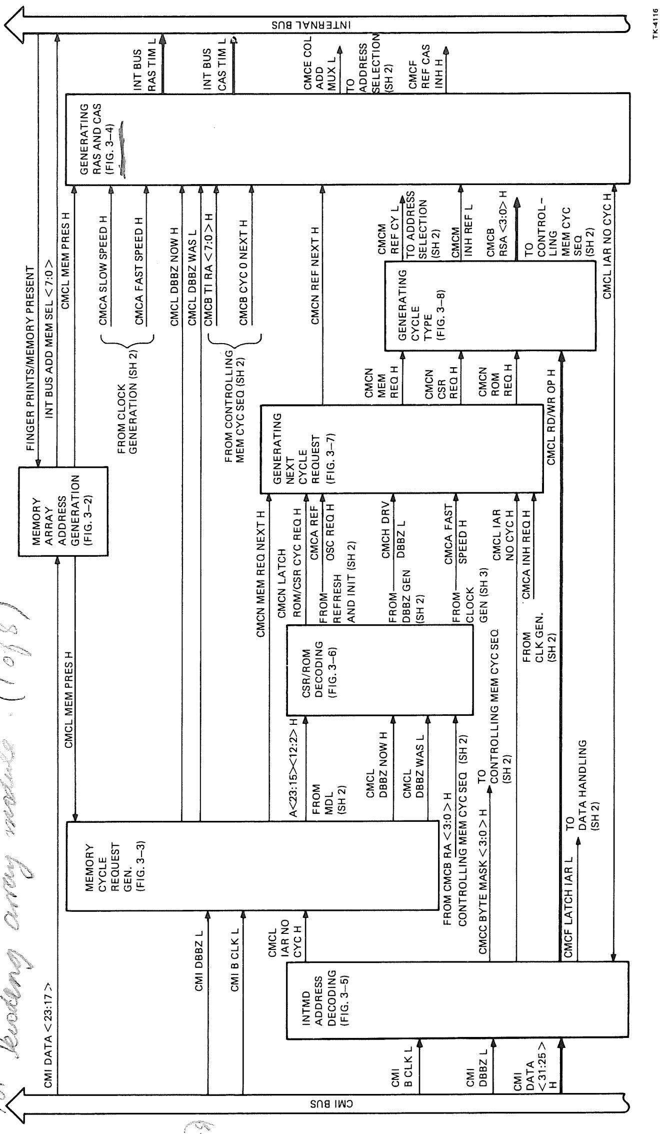
DTM needs only the m-clock to be useable.



CMC (MS750) BASIC DIAGRAM



CMC (MS750) BLOCK DIAGRAM, PART 1



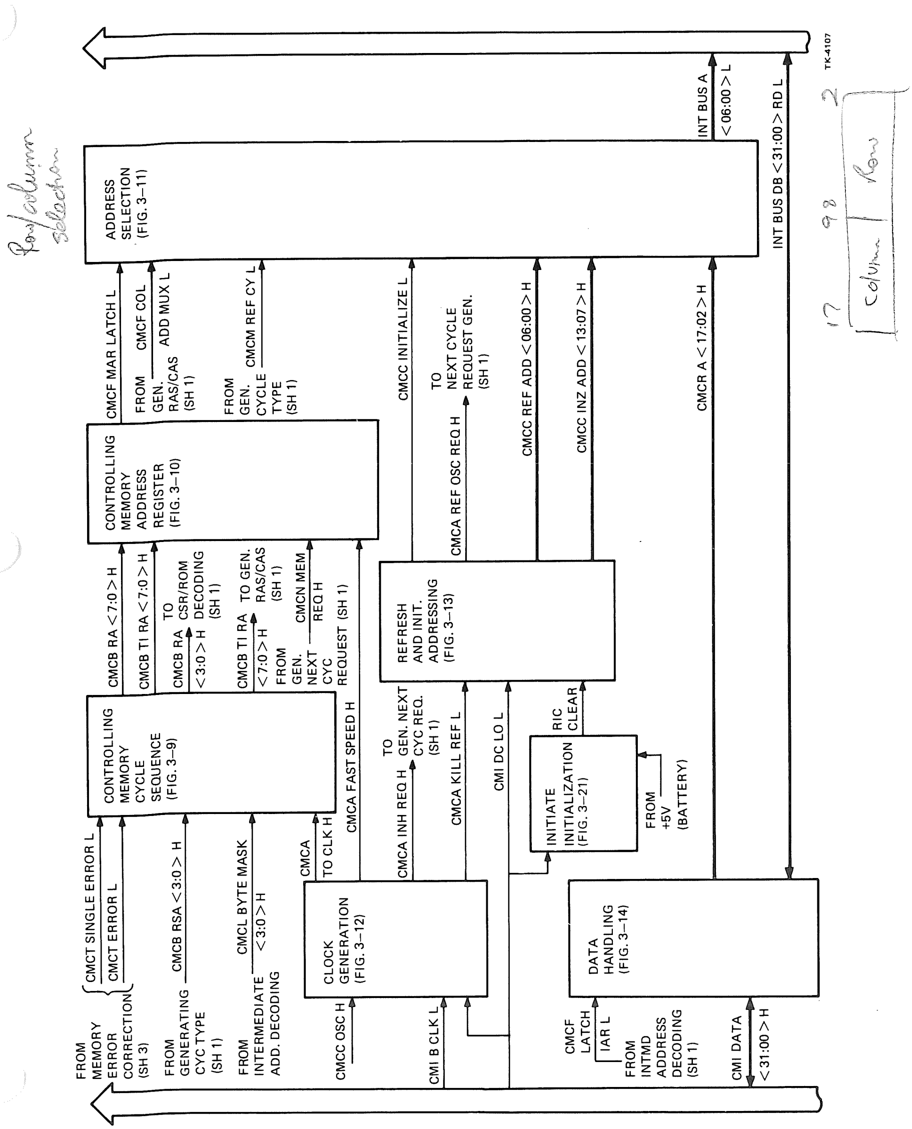
Board layout p. 55.
Gate arrays p. 56

Registers page 15

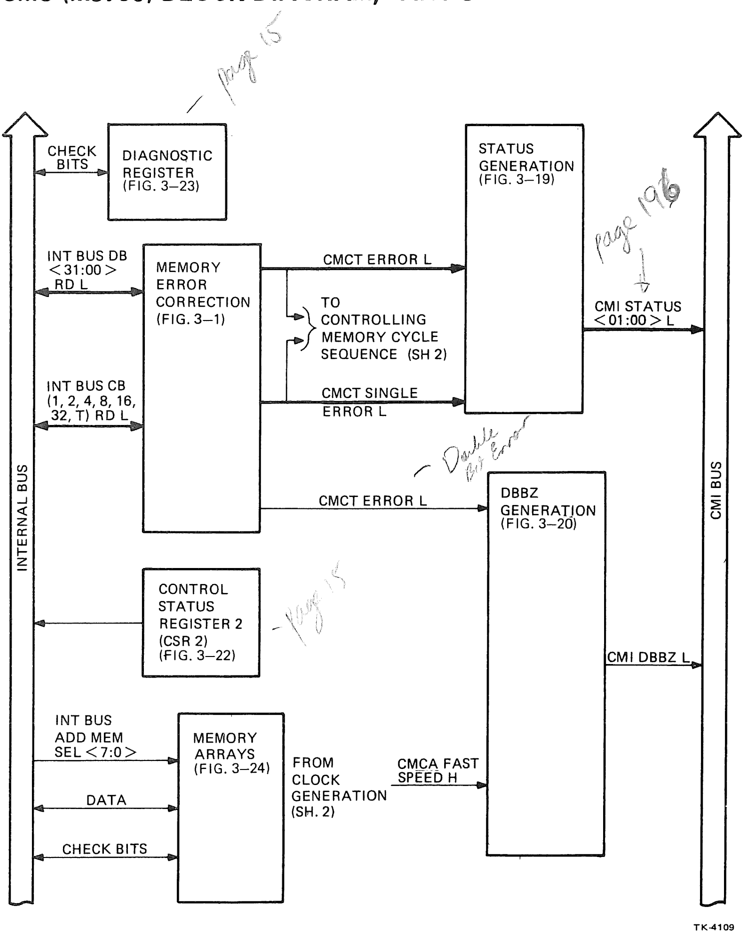
for Decoding array module (1 of 8)

(45 lines)

CMC (MS750) BLOCK DIAGRAM, PART 2

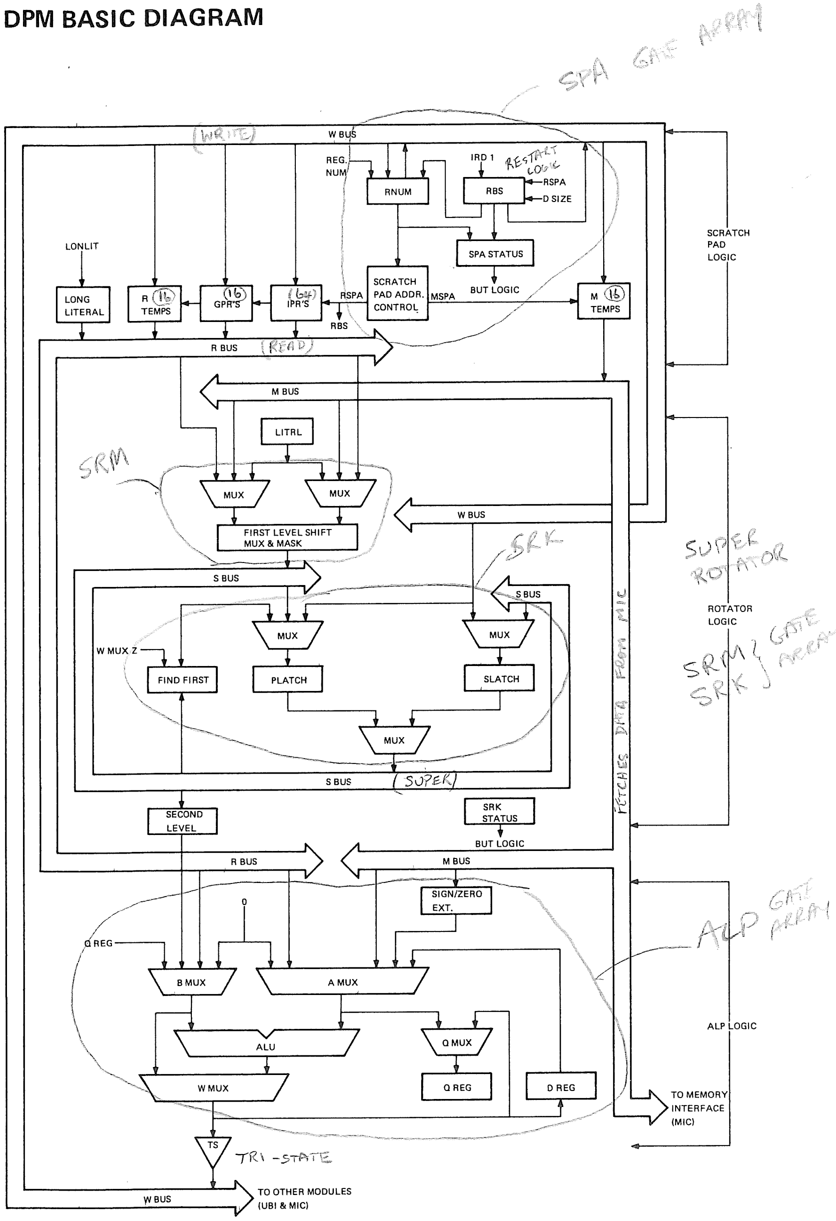


CMC (MS750) BLOCK DIAGRAM, PART 3

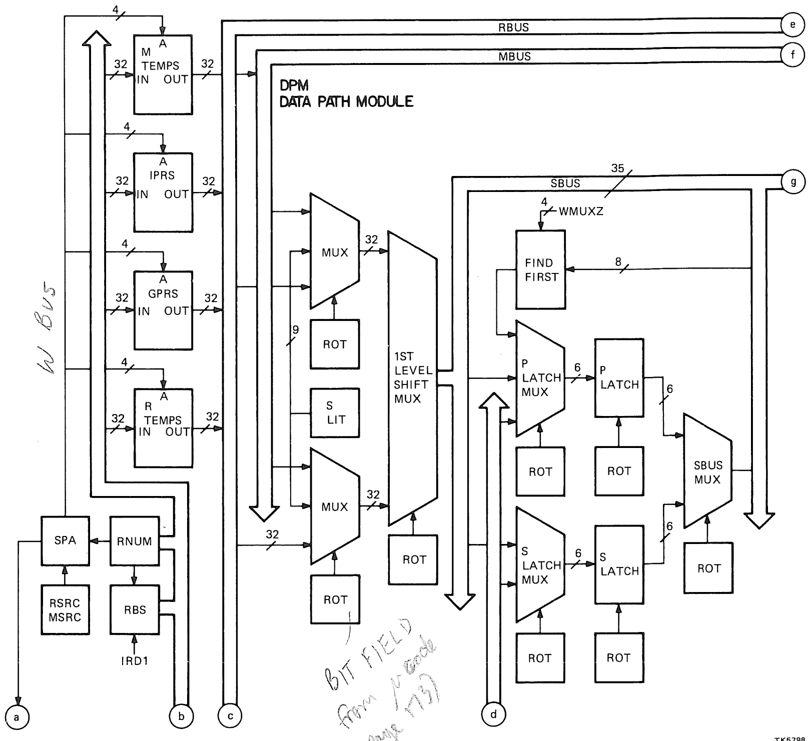


(L0002)

DPM BASIC DIAGRAM



DPM BLOCK DIAGRAM, PART 1

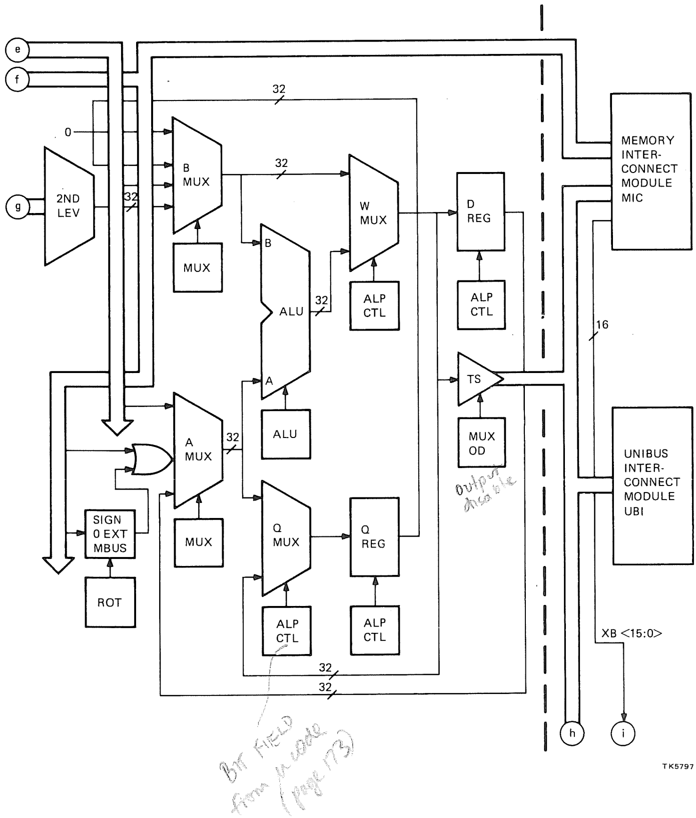


BIT FIELD from μcode (see page 173)

page 57 for Gate array description

SPA GATE ARRAY + SRM & SRK GATE ARRAYS

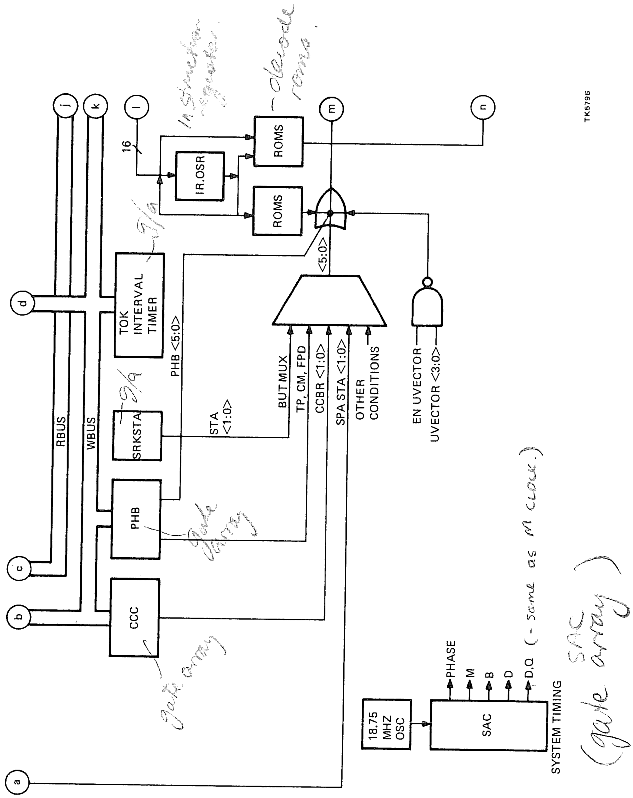
DPM BLOCK DIAGRAM, PART 2



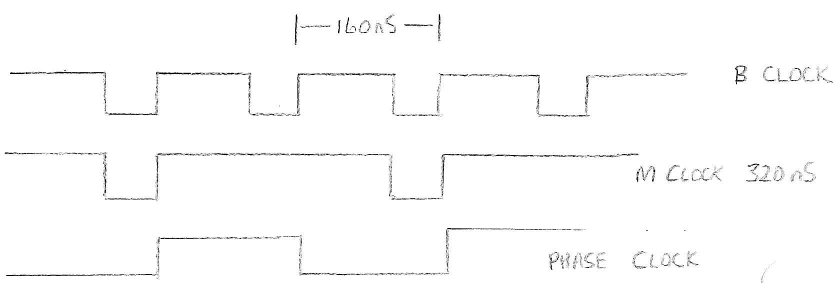
TK5797

ALP
GATE ARRAY

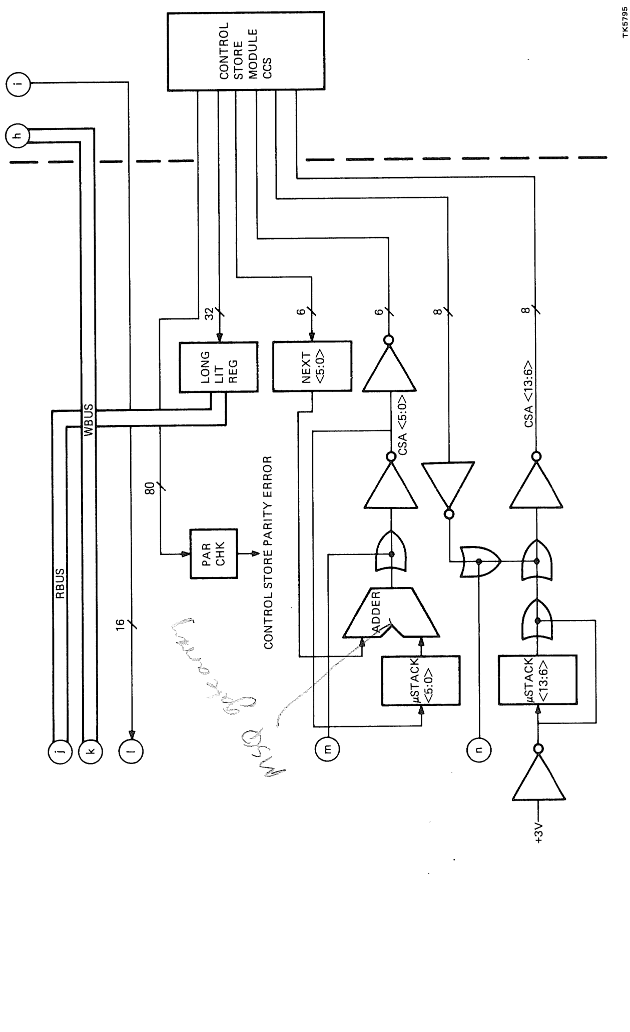
DPM BLOCK DIAGRAM, PART 3



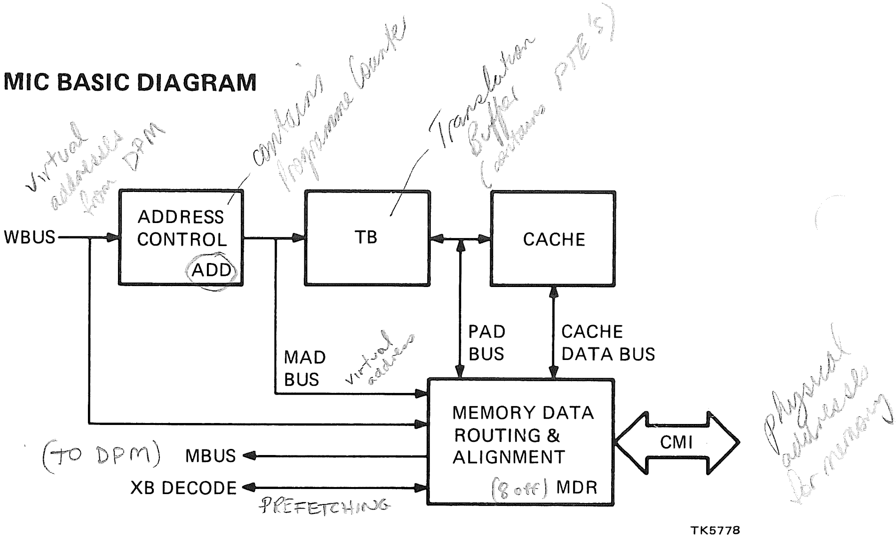
18.75 MHz = 53.3 nS ∴ 3 x Primary clock = 1 Base clock.



DPM BLOCK DIAGRAM, PART 4



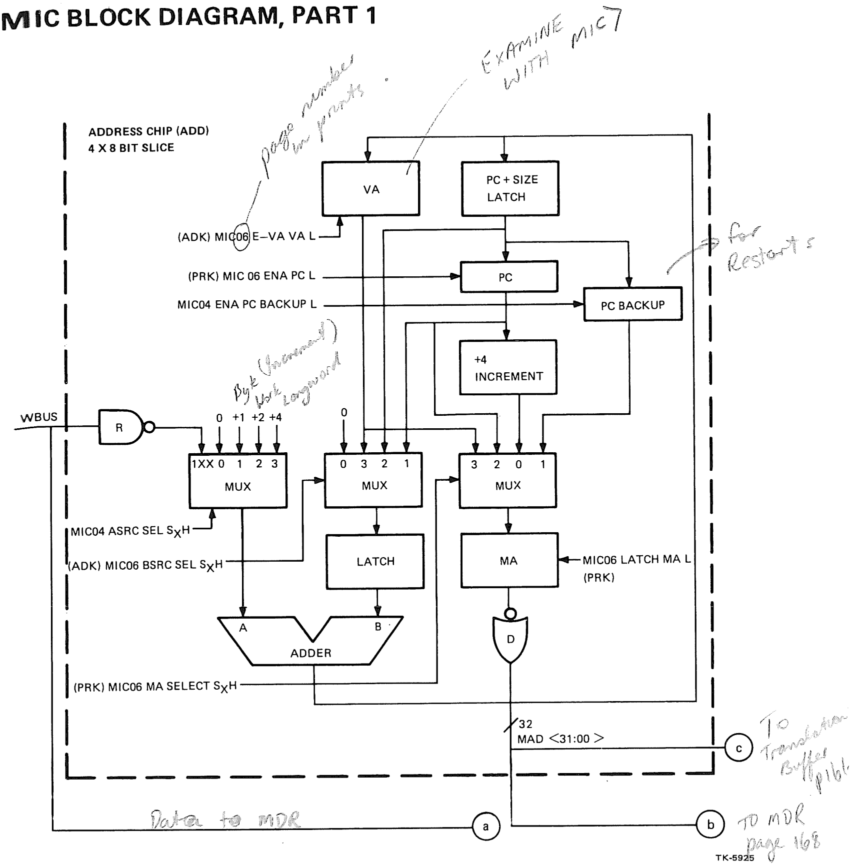
MIC BASIC DIAGRAM



Diagnostics can turn off memory management. ie TB is not used.

page 59 for gate arrays

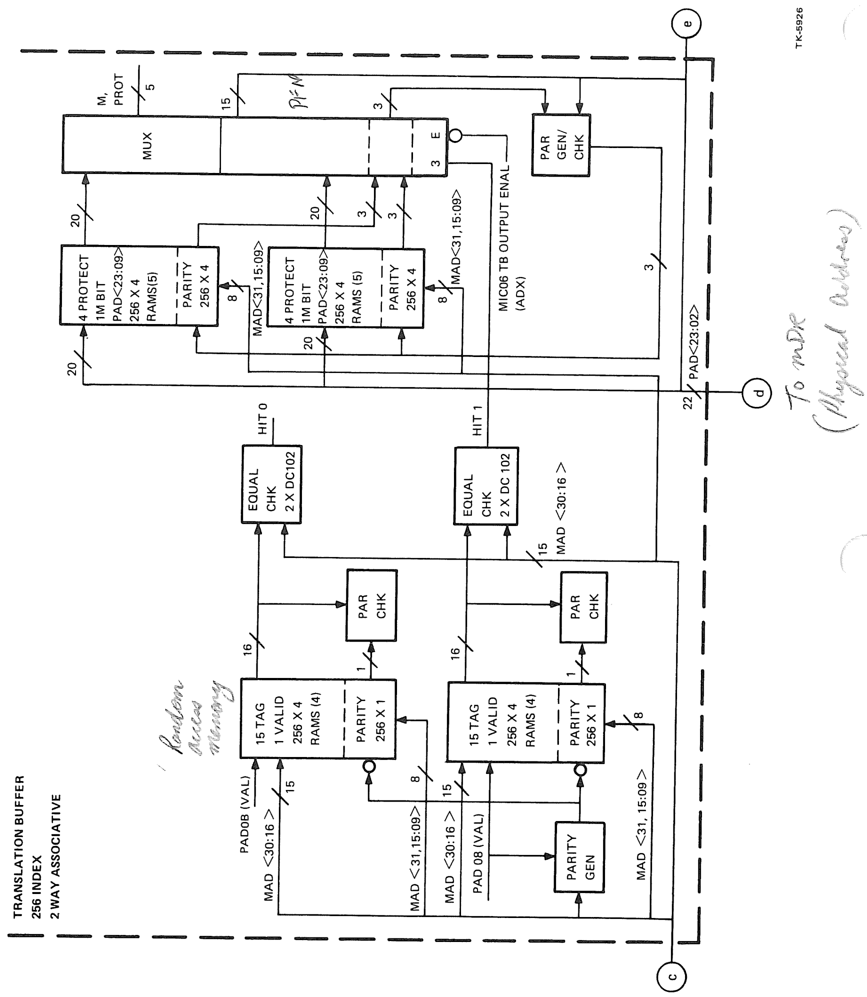
MIC BLOCK DIAGRAM, PART 1



ADD gate array
ADK for control
PRK (instruction prefetch)

MIC BLOCK DIAGRAM, PART 2

To cache page 167



To MDR (Physical Address)

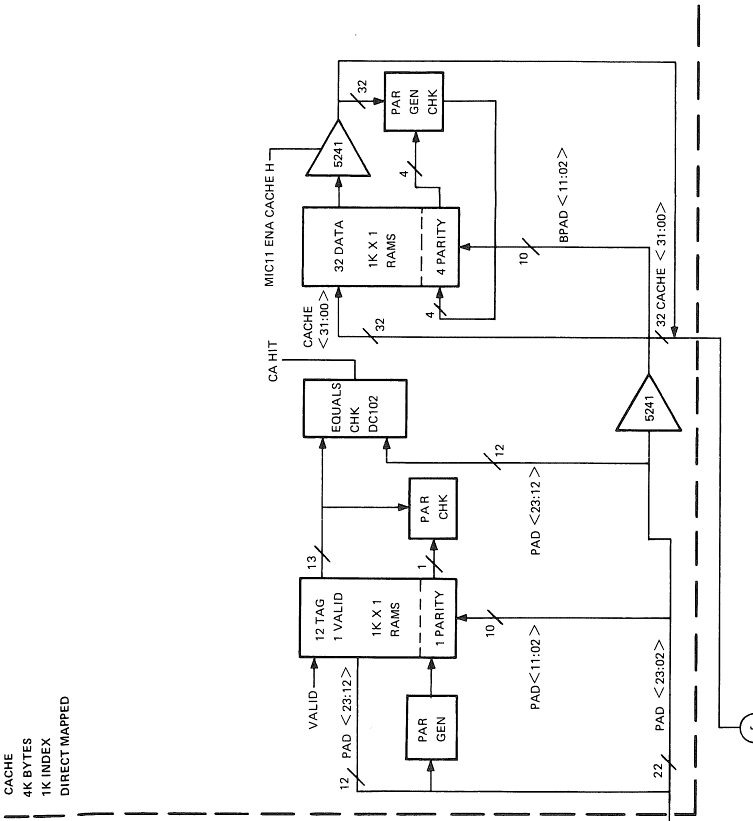
Random Access Memory

virtual address from ADD ... array

TRANSLATION BUFFER

also see page 169

MIC BLOCK DIAGRAM, PART 3



TO MDR
CACHE DATA
page 168

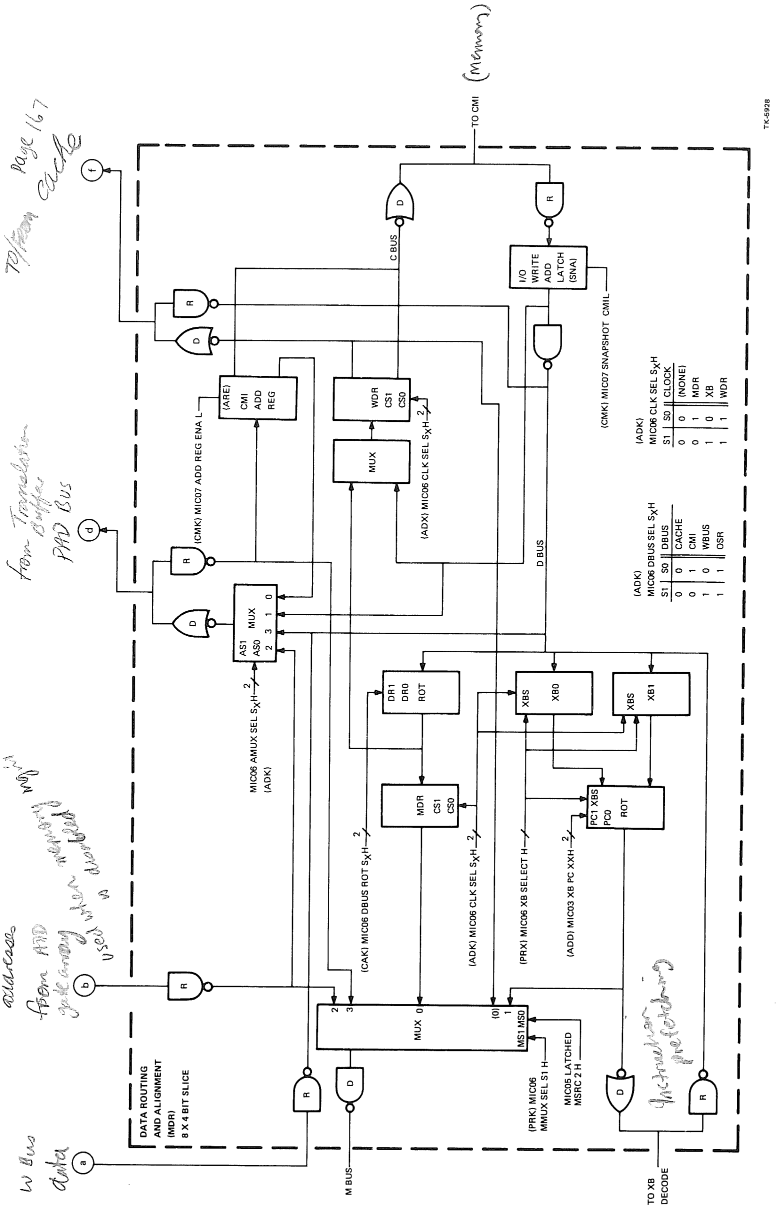
PHYSICAL
ADDRESS
from
Translation
Buffer
(P. 166)

REGISTERS P 27-29

CAK GATE ARRAY

CACHE

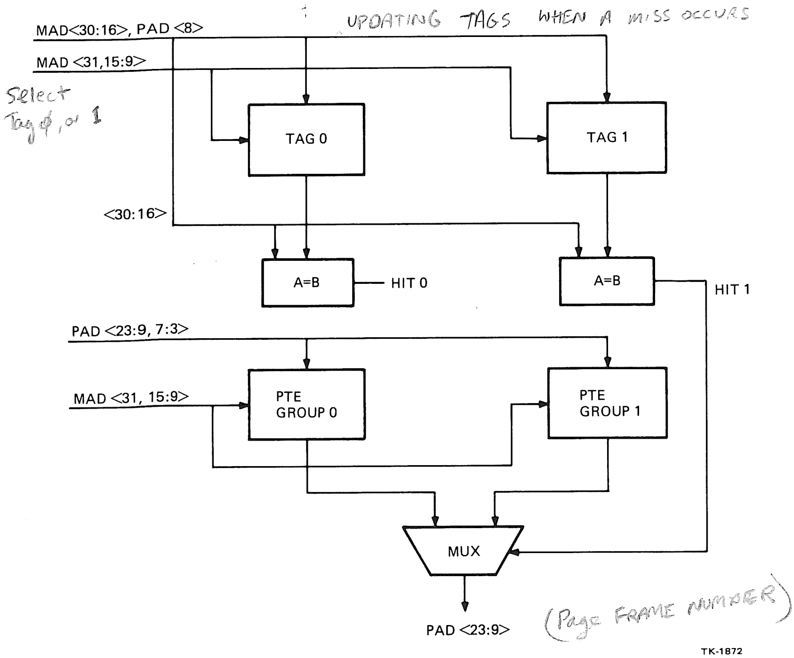
MIC BLOCK DIAGRAM, PART 4



DATA ROUTING

MDR GATE ARRAYS

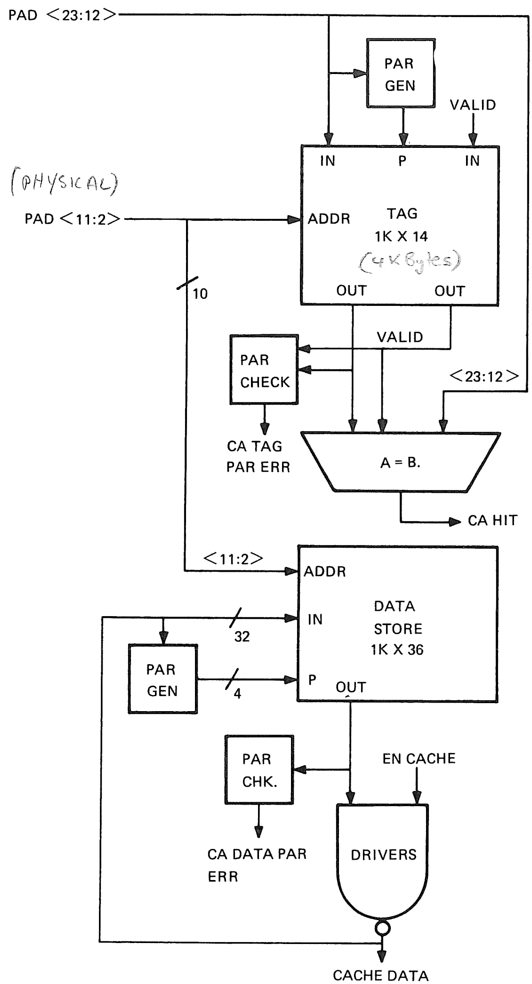
TB FUNCTIONAL DIAGRAM



TRANSLATION BUFFER LOOKS
FOR CACHE HIT.

(USE WITH PAGE 166)

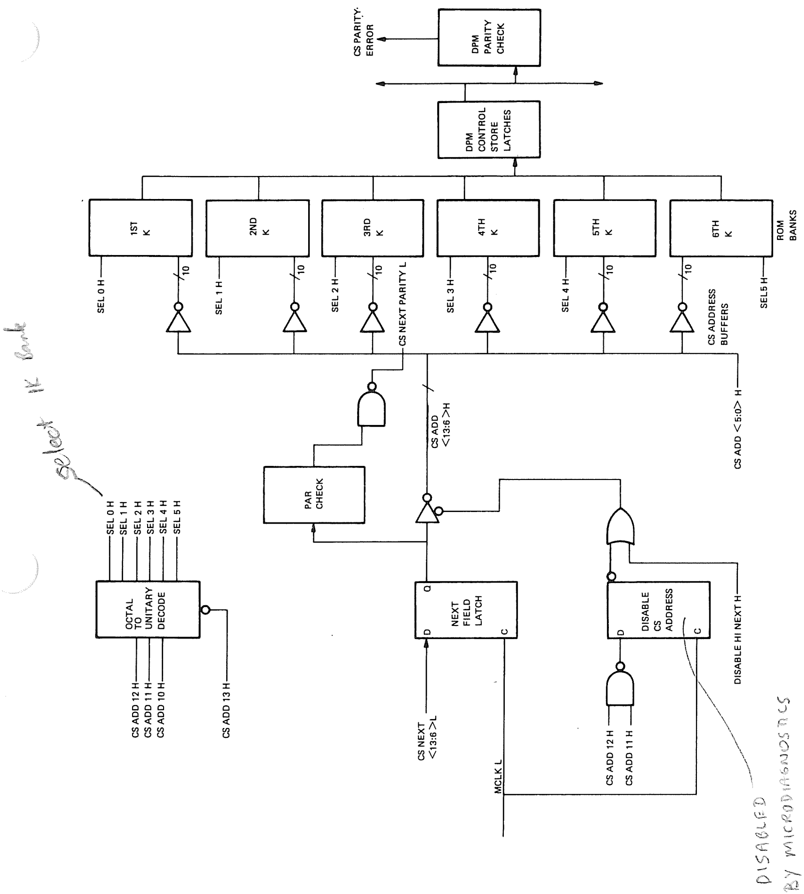
CACHE FUNCTIONAL DIAGRAM



— IF THESE TWO ARE EQUAL = CACHE HIT.

TK-3041

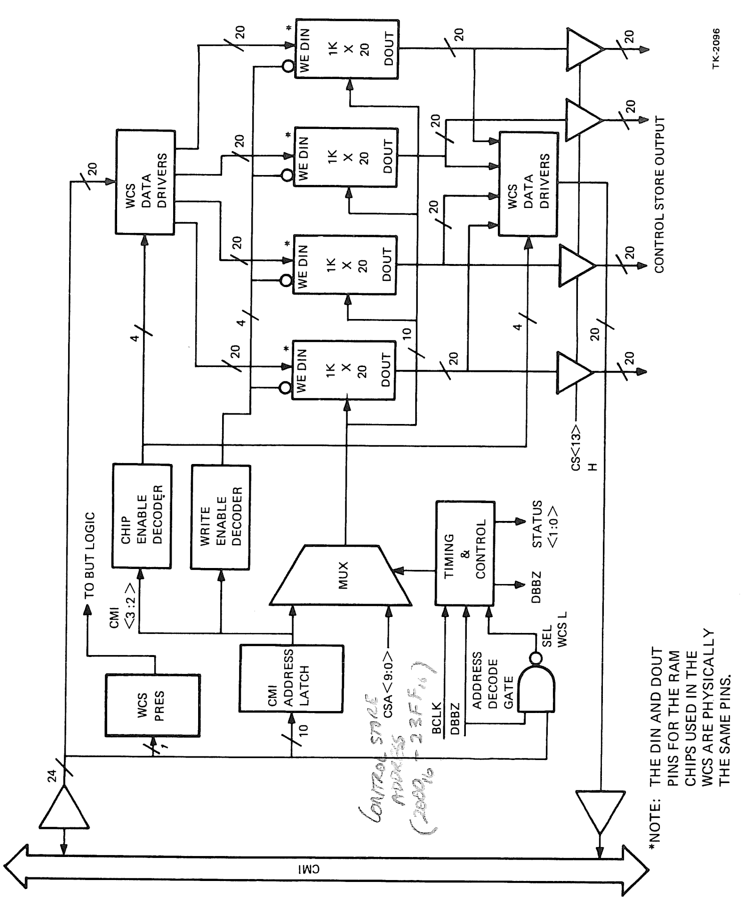
CCS BLOCK DIAGRAM



Layout page 71

WCS (KU750) BLOCK DIAGRAM

TOTAL OF 80 BITS



PHYSICAL START ADDRESS = F00000 - F10000

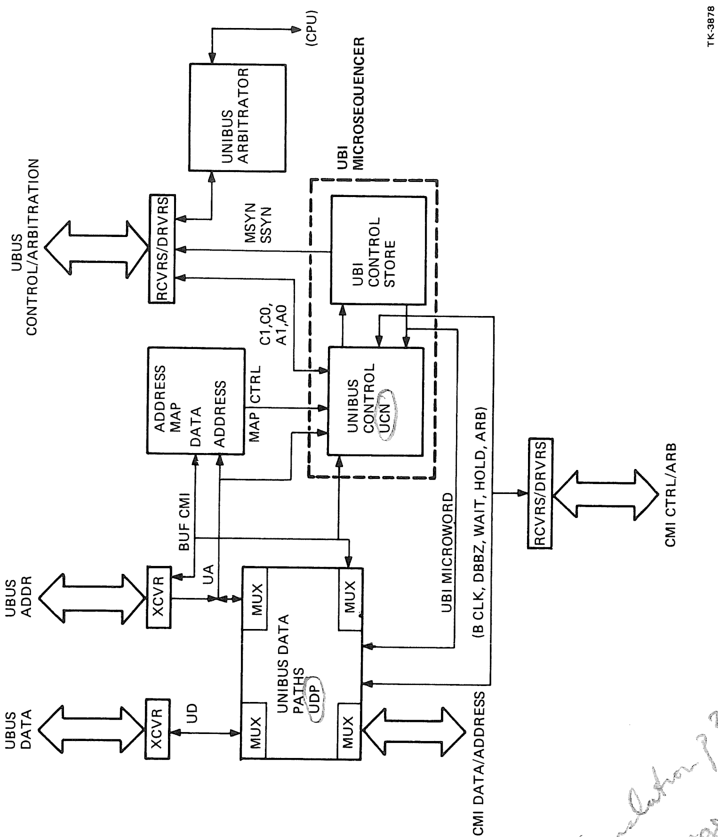
CCS/WCS MICROWORD

```
                                        +---------------------------------------------------------------+
                                        |                                                               |
                                        |             LONGLIT                 +-------------------------+
                                        |                                     |         LITRL           |
    +--------------+--------------------+-------------------------------------+---+---+-----+-----------------------+
    |     PAR2     |                                 PAR1                         |2| 1 |          PAR2           |
+-+-+----+---------+---+------+---------+----------+------------+----------+---+--+-+---+------------+------------+
| | |    |         |   |      |         |          |            |          | D |      |I|   |            |            |
|P|P|    |         |   |      |         |          |            |          | T |      |S|   |            |            |
|A|A|LIT |  MISC   |SPW| MSRC |   ROT   |  ALPCTL  |    BUT     |          | Y | RSRC |T| C |   WCTRL    |    BUS     |
|R|R|    |         |   |      |         |          |            |          | P |      |R| C |            |            |
|2|1|    |         |   |      |         |          |            |          | E |      |M|   |            |            |
+-+-+----+---------+---+------+---------+----------+------------+----------+---+------+-+---+------------+------------+
|7|7|7 7|7 7 7 7 7|7 6|6 6 6 6 6|6 6 6 6 5 5|5 5 5 5 5 5 5 4 4|4 4 4 4 4 4|4 4|3 3 3 3 3 3|3|3 3|3 2 2 2 2 2 2 2 2 2|
|9|8|7 6|5 4 3 2 1|0 9|8 7 6 5 4|3 2 1 0 9 8|7 6 5 4 3 2 1 0 9 8|7 6 5 4 3 2|1 0|9 8 7 6 5 4|3|2 1|0 9 8 7 6 5 4 3 2 1 0|
+-+-+----+---------+---+---------+-----------+-----------------+-----------+---+-----------+-+---+--------------------+
                               | |     |   |        |       |   |                           |              |
                               |A|  A  | A |  MUX   |  ALU  |DQ1|                           |    CCMISC    |
                               |L|  L  | L |        |       |   |                           |              |
                               |U|  U  | U +--------+-------+---+                           +-+------------+
                               |X|  S  | C |        |       |   |                             |            |
                               |M|  H  | I |        | ALUOD |DQ2|                             |   CCPSL    |
                               | |  F  |   |        |       |   |                             |            |
                               +-+-----+---+        +-------+---+                             +------------+
                               |           |                |   |
                               |  ROTSRK   |                |DQ3|
                               |           |                |   |
                               +-----------+                +---+
```

*Note: The microcode listing shows these bits in reverse order.

```
+-------------------------------------------------------------+
|                            PAR2                             |
+---------+-+-+-----------------------------------------------+
|         |C|J|                                               |
|   FPA   |L|S|                  NEXT                         |
|         |K|R|<-          Reversed         ->|*              |
+---------+-+-+-----------------------------------------------+
|1 1 1 1|1|1|1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|9 8 7 6|5|4|3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-------+-+-+---------------------------+
```

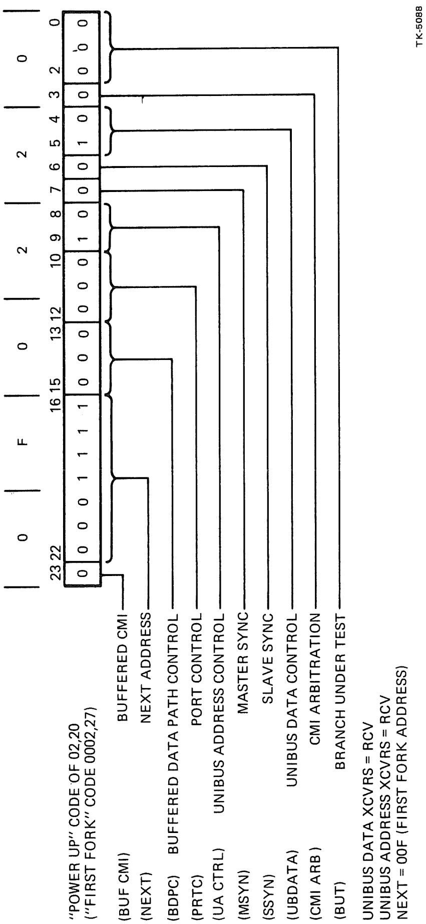
UBI BLOCK DIAGRAM



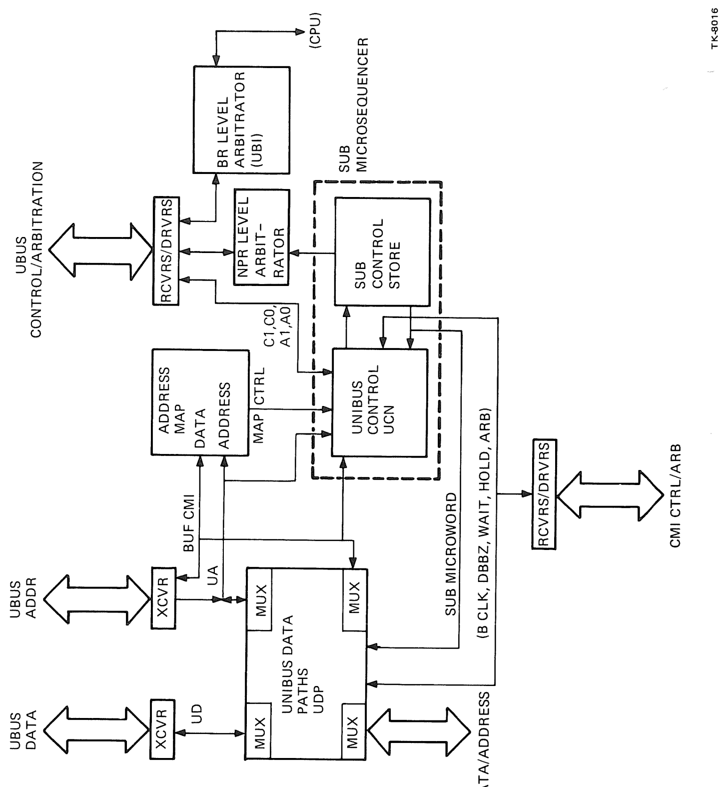
gate arrays page 61

Address Translation p 36
Register Summary page 210.
Data Flows p 178

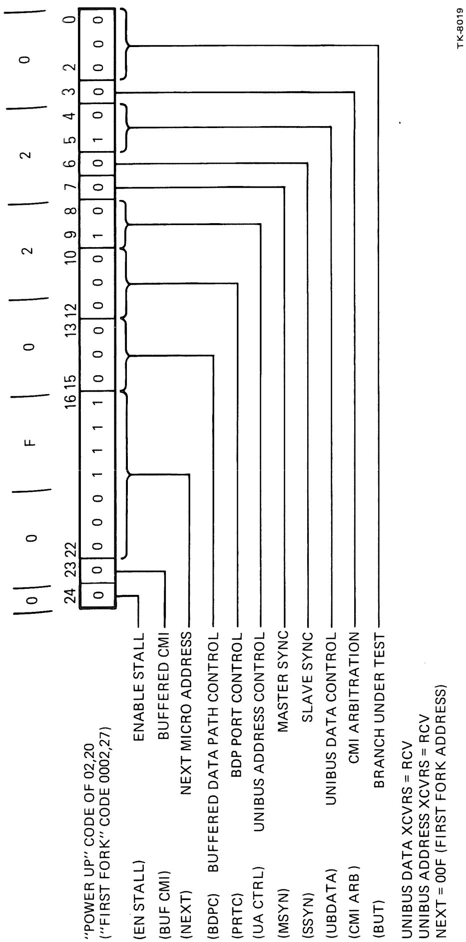
UBI MICROWORD



SUB (DW750) BLOCK DIAGRAM

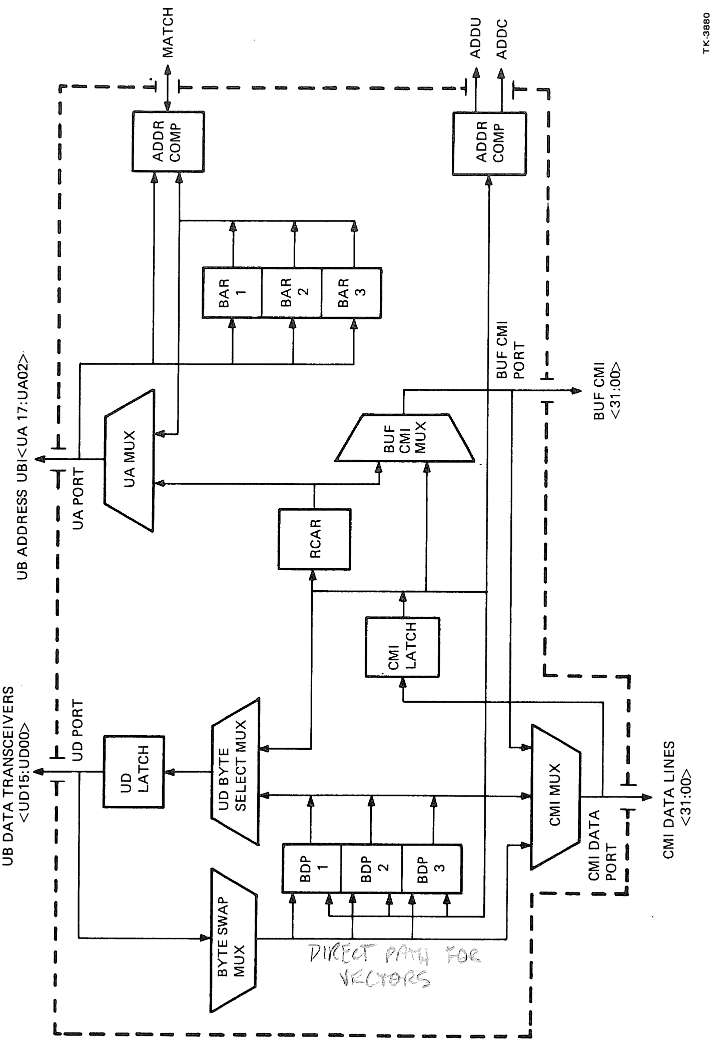


SUB MICROWORD



UDP DATA FLOW BLOCK DIAGRAM

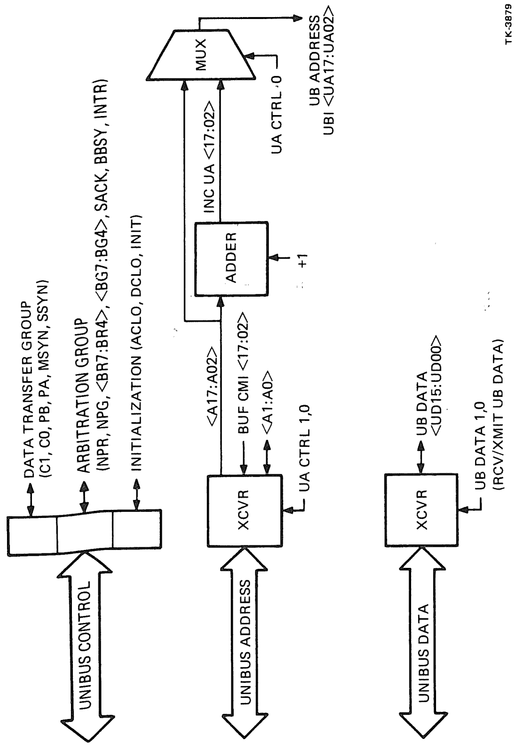
(Better diagram page 48 Diagnostic User)



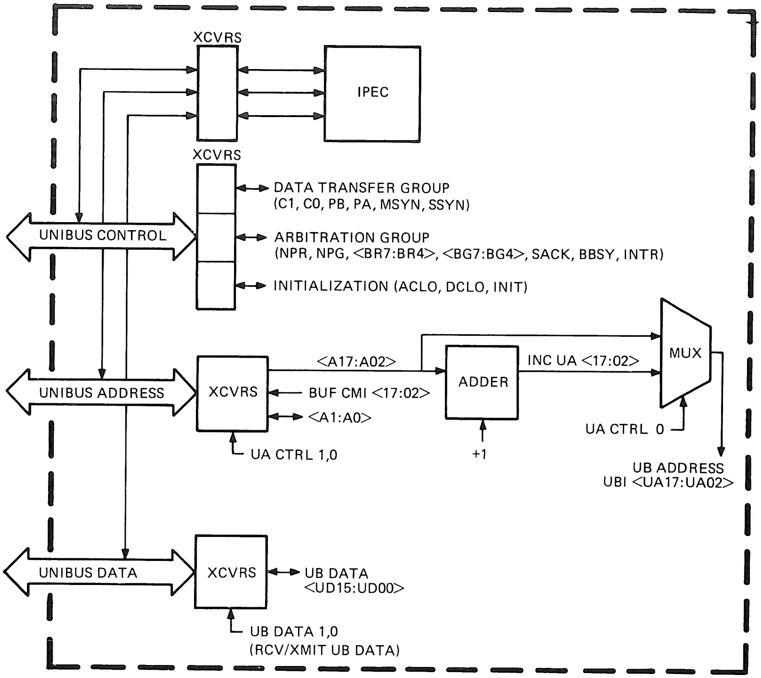
UDP gate array

(See page 34 for address translation.
(Block diagram p 174)

UBI INTERFACE TO FIRST UNIBUS LINES

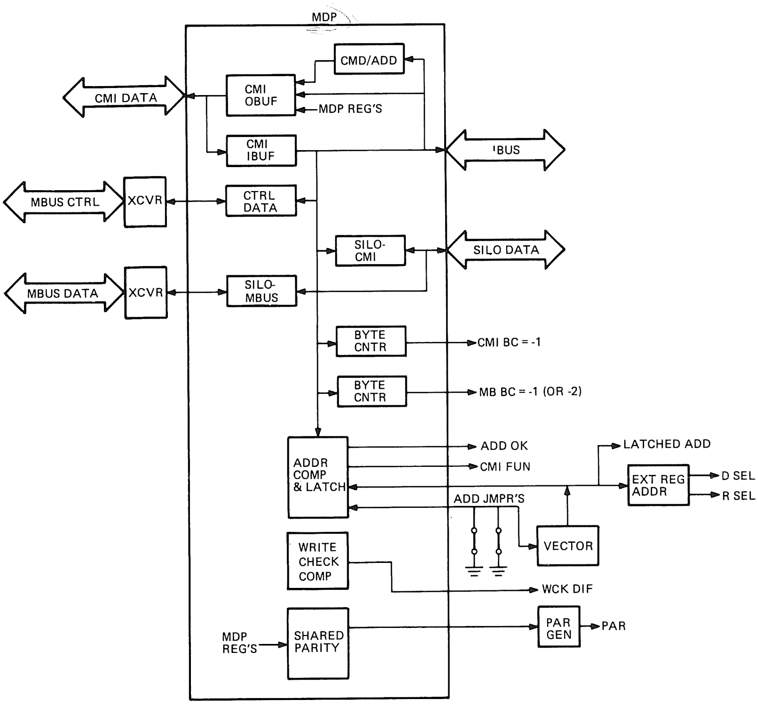


SUB INTERFACE TO SECOND UNIBUS LINES



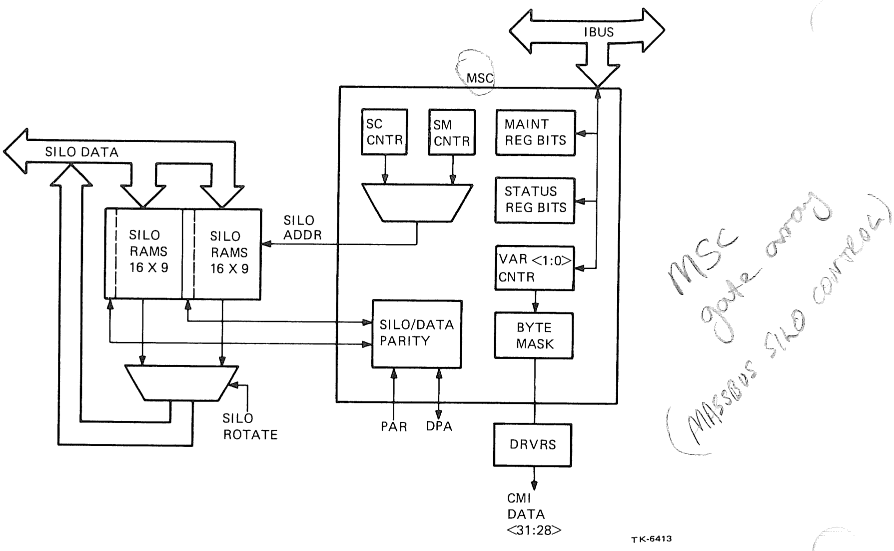
TK-8017

MBA (RH750) BLOCK DIAGRAM, PART 1

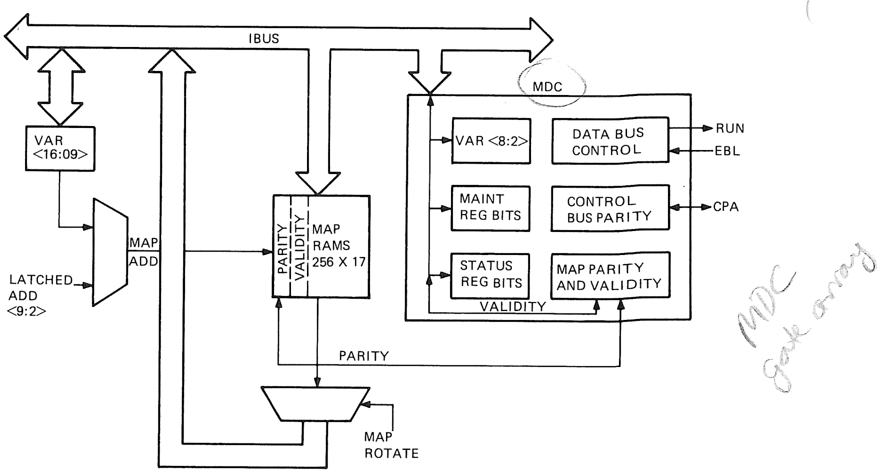


TK-6410

MBA (RH750) BLOCK DIAGRAM, PART 2

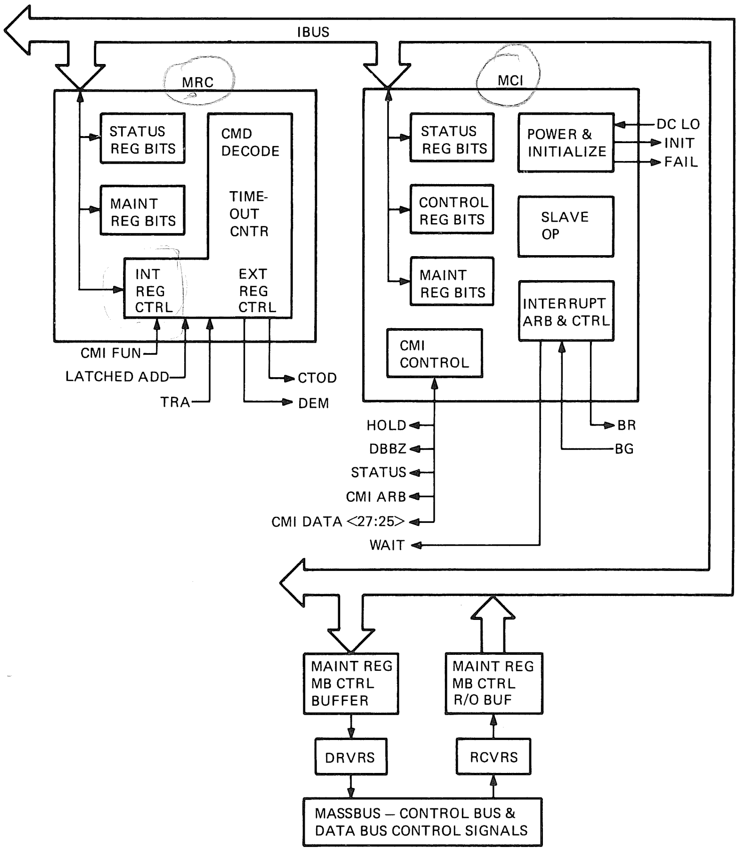


TK-6413



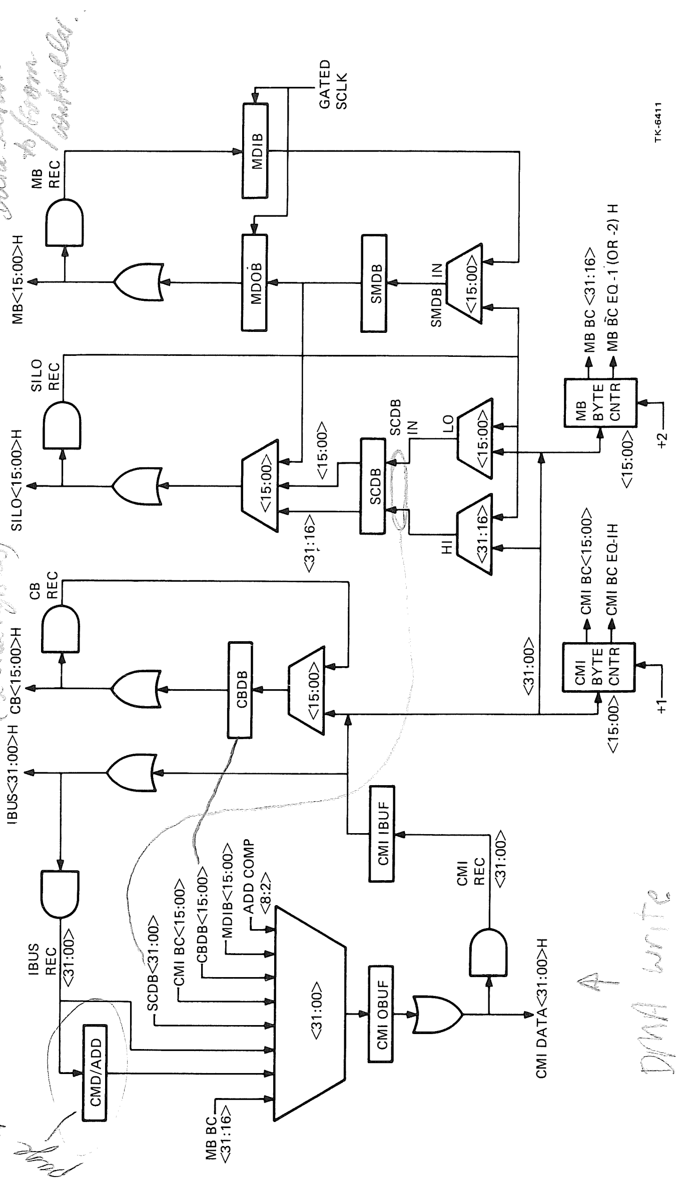
TK-6414

MBA (RH750) BLOCK DIAGRAM, PART 3

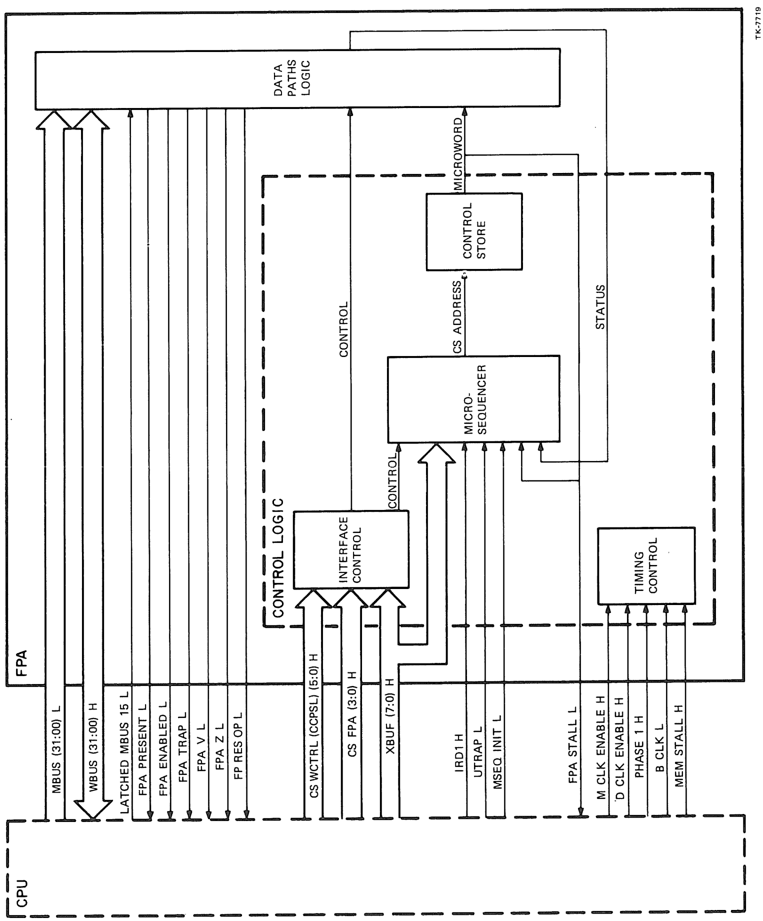


TK-6415

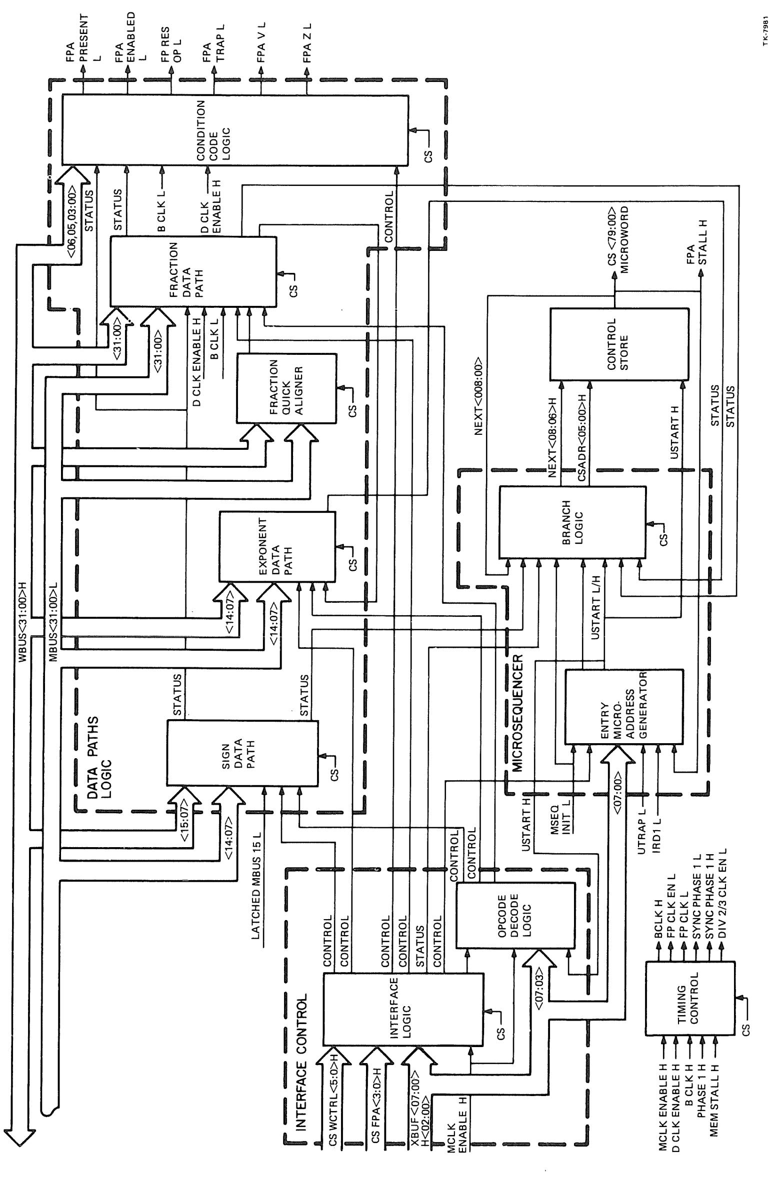
MDP DATA FLOW BLOCK DIAGRAM



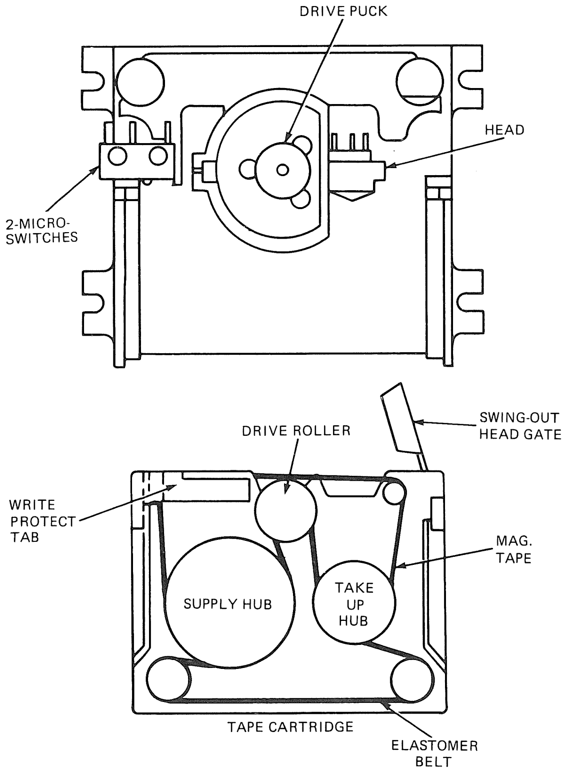
FPA (FP750) BASIC DIAGRAM



FPA (FP750) BLOCK DIAGRAM



TU58 DRIVE/CARTRIDGE DIAGRAM



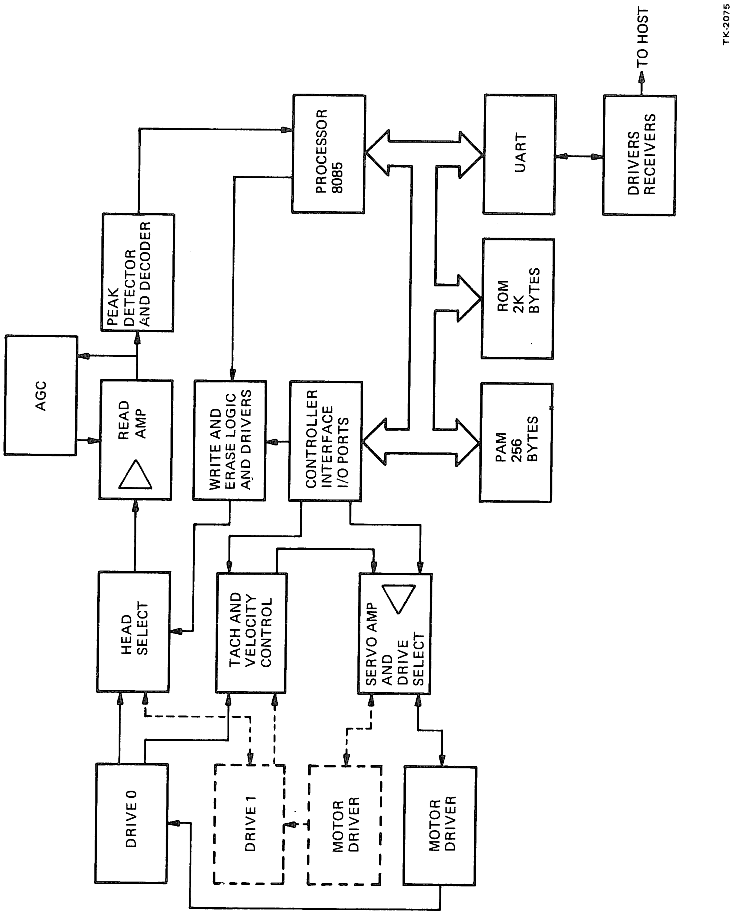
TK-2040

TU58 CASSETTE TAPE BLOCK LOCATIONS

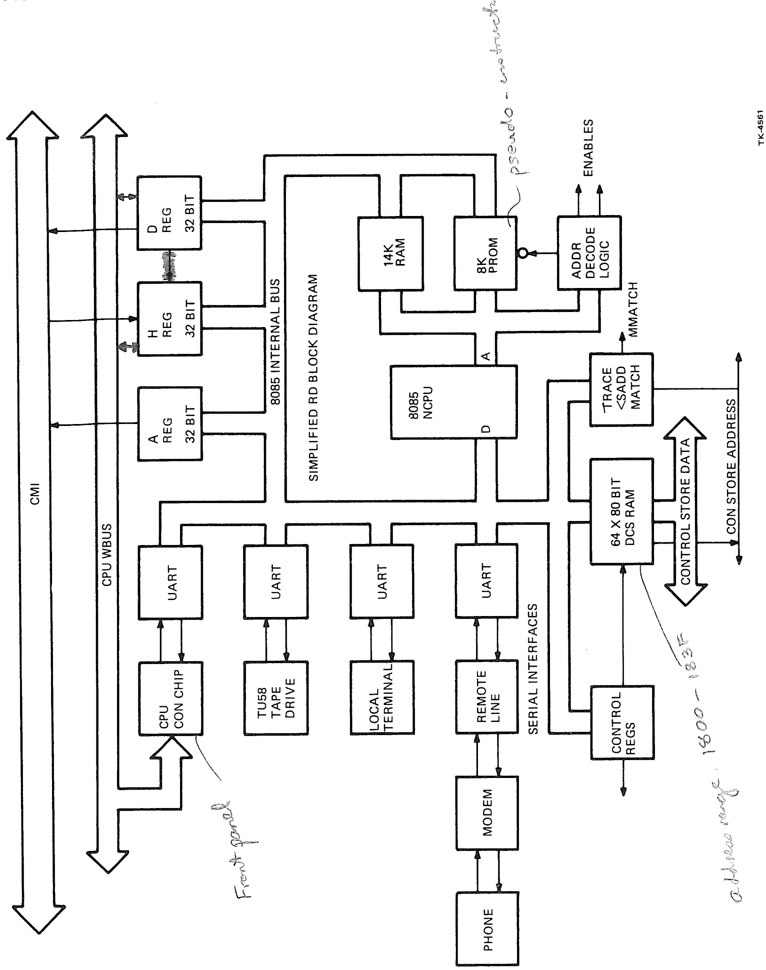
BOT	#256	#384	#257	#385	#258	#386	#259		#382	#510	#383	#511	EOT
BOT	#0	#128	#1	#129	#2	#130	#3		#126	#254	#127	#255	EOT

TK-2080

TU58 BLOCK DIAGRAM



RDM BLOCK DIAGRAM



CHAPTER 8
CMI, UNIBUS, AND MASSBUS

CMI PHYSICAL ADDRESS MAP

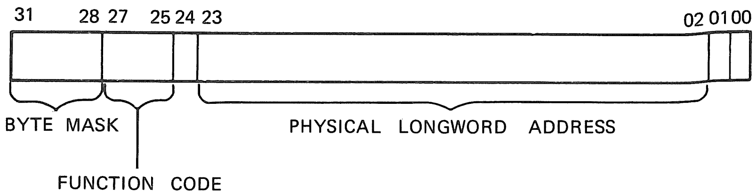
1 MByte array			
000000 0FFFFF	000000 03FFFF	256 KB (1 MB) *	1 ARRAY BOARD
100000 1FFFFF	040000 07FFFF	512 KB (2 MB)	
200000 2FFFFF	080000 0BFFFF	768 KB (3 MB)	
300000 3FFFFF	0C0000 FFFFF	1024 KB (4 MB)	
400000 4FFFFF	100000 13FFFF	1280 KB (5 MB)	
500000 5FFFFF	140000 17FFFF	1536 KB (6 MB)	
600000 6FFFFF	180000 1BFFFF	1892 KB (7 MB)	
700000 7FFFFF	1C0000 1FFFFF	2048 KB (8 MB) MAXIMUM FULLY POPULATED ARRAYS	
	F00000	10 KB USER CONTROL STORE (WCS)	I/O SPACE
	F10000		
	F20000	MEMORY CONTROL/STATUS REG. 0	
	F20004	MEMORY CONTROL/STATUS REG. 1	
	F20008	MEMORY CONTROL/STATUS REG. 2	
	F20400	BOOTSTRAP ROM A	
	F20500	BOOTSTRAP ROM B	
	F20600	BOOTSTRAP ROM C	
	F20700	BOOTSTRAP ROM D	
	F28000	MASSBUS ADAPTOR 0 INT. REGISTERS	
	F28400	MASSBUS ADAPTOR 0 EXT. REGISTERS	
	F28800	MASSBUS ADAPTOR 0 MAP REGISTERS	
	F2A000	MASSBUS ADAPTOR 1 INT. REGISTERS	
	F2A400	MASSBUS ADAPTOR 1 EXT. REGISTERS	
	F2A800	MASSBUS ADAPTOR 1 MAP REGISTERS	
	F2C000	MASSBUS ADAPTOR 2 INT. REGISTERS	
	F2C400	MASSBUS ADAPTOR 2 EXT.REGISTERS	
	F2C800	MASSBUS ADAPTOR 2 MAP REGISTERS	
	F30000-C	1ST UNIBUS/DATA PATH CONTROL & STATUS	
	F30800 F30FFF	1ST UNIBUS/MAP REGISTERS	
	F32000-C	2ND UNIBUS/DATA PATH CONTROL & STATUS	
	F32800 F32FFF	2ND UNIBUS/MAP REGISTERS	
	F80000 FBFFFF	2ND UNIBUS/MEMORY SPACE 131 KW	
	FC0000 FFFFFF	1ST UNIBUS/MEMORY SPACE 131 KW	

* USING L0016 CONTROLLER

TK-5814

CMI ADDRESS AND DATA FORMAT

CMI ADDRESS FORMAT

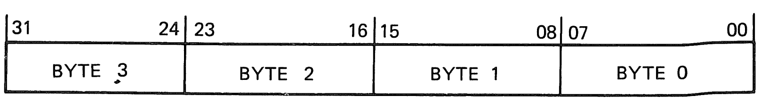


TK-3875

CMI BYTE MASK AND FUNCTION BITS

Byte Mask Bit	Byte(s) Valid for Transfer	Function Bit 27	26	25	CMI Operation
Bit <31>	Byte 3 valid	0	0	0	Read
Bit <30>	Byte 2 valid	0	0	1	Read lock
Bit <29>	Byte 1 valid	0	1	0	Read modify
Bit <28>	Byte 0 valid	0	1	1	(Undefined)
		1	0	0	Write
		1	0	1	Write unlock
		1	1	0	Write vector
		1	1	1	(Undefined)

CMI DATA FORMAT



TK-3876

CMI SIGNAL DESCRIPTION

Signal Lines	Description
Timing Group	
B CLK L (160nS)	B CLK L is generated by the CPU to synchronize all system activities. One B CLK cycle is from one rising edge of B CLK L to the next. B CLK L is low for one-third of the cycle.
Control/Address and Data Group	
CMI DATA <31:00>	The CMI data lines are first asserted by a device that has arbitrated for the CMI and assumed control as master. The new master transmits control and address information to the slave in the CMI address format, and asserts DBBZ for one B CLK cycle. Data is then transferred on the lines in the CMI data format. Bits <01:00> of the CMI address are ignored since four bytes (one longword) of data are represented on the lines and the valid data is selected by the byte mask.

NOTE

CMI DATA signals are asserted on a logical 1 at a nominal +3.5 V (high). All other CMI signals are asserted at a nominal ground (low).

Signal Lines	Description
Data Bus Busy (DBBZ)	DBBZ is first asserted by the master for one B CLK cycle while it places the CMI address of the slave on the CMI data lines. DBBZ is then asserted by the slave until the data transfer is complete. If the slave is immediately ready to respond, it may not assert DBBZ.
HOLD	HOLD is used to temporarily block all CMI arbitration. Cache, for example, requires extra B CLK cycles to perform an invalidation. (Cache is not part of the DMA transaction and does not assert DBBZ.)
WAIT (INTERRUPT)	WAIT is asserted by a CMI subsystem to initiate a processor interrupt. It is held until the write vector operation is performed.

CMI SIGNAL DESCIRPTION (CONT)

Signal Lines	Description
Priority Arbitration Group	
<ARB7:ARB1>	An ARB level is assigned to each subsystem and is used to gain control of the CMI. If HOLD or a higher priority level is not asserted when a subsystem asserts its own, the subsystem assumes control of the CMI, asserting DBBZ and the CMI address of a slave subsystem. If HOLD or a higher priority level is asserted, the subsystem asserts its own priority bit to hold off subsystems of lower priority until it gains control. Priority on the CMI is assigned as follows: ARB7 RDM - highest priority ARB6 Reserved ARB5 Reserved ARB4 UBI ARB3 MBAØ or SUB ARB2 MBA1 or MBAØ (with SUB) ARB1 MBA2 or MBA1 (with SUB) CPU - lowest priority (does not assert an ARB level)
Status Group	
STATUS <1:Ø>	Status is transmitted by a slave to indicate the condition under which it returns data to the master. Status bit combinations are encoded as follows:

page 202, 3 for jumpers

Status Bit

1	Ø	
Ø	Ø	NXM - The master tried to access non-existent memory space to read or write data.
Ø	1	UCE - Read data returned to the master carried an uncorrectable error.
1	Ø	CRD - Corrected read data was returned to the master.
1	1	NO ERR - The read data had no errors.

For slots 7, 8, 9.

CMI SIGNAL PIN ASSIGNMENTS ON OPTION SLOTS

BLOCK A

	A1	A2	+12
	A3	A4	
	A5	A6	
	A7	A8	
	A9	A1Ø	
+2.5	A11	A12	
	A13	A14	
	A15	A16	
	A17	A18	
	A19	A2Ø	+5
	A21	A22	
GND	A23	A24	GND
	A25	A26	
	A27	A28	
+2.5	A29	A3Ø	
	A31	A32	
	A33	A34	
	A35	A36	
	A37	A38	+5
	A39	A4Ø	
SLOT SEL 2 H	A41	A42	SLOT SEL 3 H
GND	A43	A44	GND
	A45	A46	
+2.5	A47	A48	
	A49	A5Ø	
GND	A51	A52	GND
SLOT SEL Ø H	A53	A54	SLOT SEL 1 H
CMI ARB 5 L	A55	A56	
ARB OUT [A] L	A57	A58	+5
CMI ARB 6 L	A59	A6Ø	ARB IN 1 [A] L
ARB IN Ø [A] L	A61	A62	CMI ARB 3 L
CMI ARB 2 L	A63	A64	ARB OUT [A] L
+2.5	A65	A66	CMI ARB 1 L
UBUS BG4 [A] H	A67	A68	UBUS BG4 [A+1] H
UBUS BG5 [A] H	A69	A7Ø	UBUS BG5 [A+1] H
GND	A71	A72	GND
UBUS BG6 [A] H	A73	A74	UBUS BG6 [A+1] H
	A75	A76	+5
UBUS BG7 [A] H	A77	A78	UBUS BG7 [A+1] H
UBUS BR4 L	A79	A8Ø	UBUS BR5 L
UBUS BR6 L	A81	A82	UBUS BR7 L
+2.5	A83	A84	CMI ARB 1 L
CMI ARB 2 L	A85	A86	CMI ARB 3 L
CMI ARB 4 L	A87	A88	CMI ARB 5 L
CMI ARB 6 L	A89	A9Ø	CMI ARB 7 L
-15	A91	A92	CMI HOLD L
CMI WAIT L	A93	A94	-5

For Slots 7, 8, 9

CMI SIGNAL PIN ASSIGNMENTS ON OPTION SLOTS (CONT)

BLOCK B	CMI DATA Ø H	B1	B2	+12
	CMI DATA 1 H	B3	B4	CMI DATA 2 H
	CMI DATA 3 H	B5	B6	CMI DATA 4 H
	CMI DATA 5 H	B7	B8	CMI DATA 6 H
	CMI DATA 7 H	B9	B1Ø	CMI DATA 8 H
	+2.5	B11	B12	CMI DATA 9 H
	CMI DATA 1Ø H	B13	B14	CMI DATA 11 H
	CMI DATA 12 H	B15	B16	CMI DATA 13 H
	CMI DATA 14 H	B17	B18	CMI DATA 15 H
	CMI DATA 16 H	B19	B2Ø	+5
	CMI DATA 17 H	B21	B22	CMI DATA 18 H
	GND	B23	B24	GND
	CMI DATA 19 H	B25	B26	CMI DATA 2Ø H
	CMI DATA 21 H	B27	B28	CMI DATA 22 H
	+2.5	B29	B3Ø	CMI DATA 23 H
	CMI DATA 24 H	B31	B32	CMI DATA 25 H
	CMI DATA 26 H	B33	B34	CMI DATA 27 H
	CMI DATA 28 H	B35	B36	CMI DATA 29 H
	CMI DATA 3Ø H	B37	B38	+5
	CMI DATA 31 H	B39	B4Ø	CMI STATUS Ø L
	CMI STATUS 1 L	B41	B42	CMI DBBZ L
	GND	B43	B44	GND
	DPM17 B CLK L	B45	B46	+5
	+2.5	B47	B48	DW75Ø
	+2.5	B49	B5Ø	+5
	GND	B51	B52	GND
		B53	B54	
		B55	B56	
		B57	B58	
		B59	B6Ø	
		B61	B62	
		B63	B64	
		B65	B66	
		B67	B68	
		B69	B7Ø	
		B71	B72	
		B73	B74	
		B75	B76	
		B77	B78	
		B79	B8Ø	
		B81	B82	
		B83	B84	
		B85	B86	
		B87	B88	
		B89	B9Ø	
		B91	B92	GND
	BUS 2 AVAILABLE H	B93	B94	-5

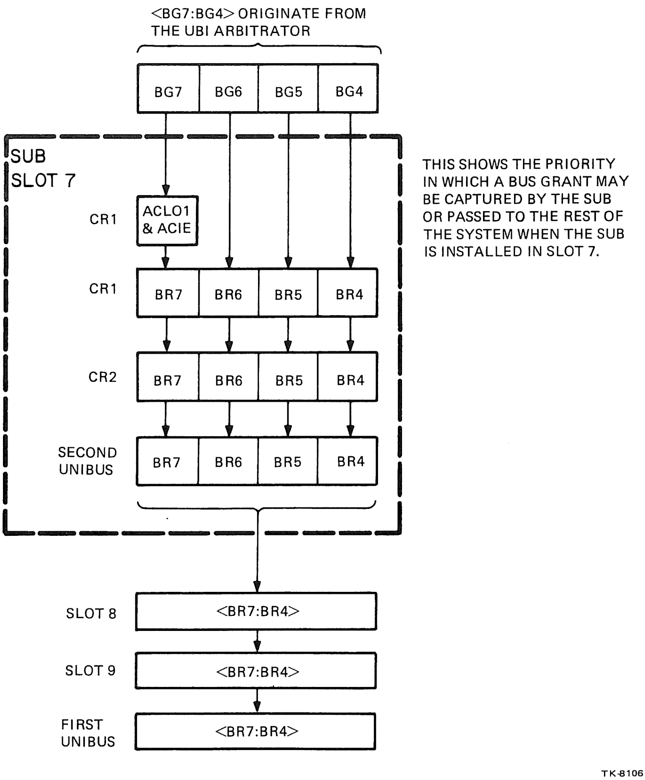
For Slots 7, 8, 9

VAX-11/750 BACKPLANE (CONT)

CMI SIGNAL PIN ASSIGNMENTS ON OPTION SLOTS (CONT)

BLOCK C	DW75Ø	C1	C2	+12
		C3	C4	
		C5	C6	
		C7	C8	
		C9	C1Ø	
		C11	C12	
		C13	C14	
		C15	C16	
		C17	C18	
		C19	C2Ø	
		C21	C22	
		C23	C24	
		C25	C26	
		C27	C28	
		C29	C3Ø	
		C31	C32	
		C33	C34	
		C35	C36	
		C37	C38	
		C39	C4Ø	
		C41	C42	GND
	GND	C43	C44	GND
	UNIBUS ACLO L	C45	C46	+5
	+2.5	C47	C48	UNIBUS INIT L
	+2.5	C49	C5Ø	+5
	GND	C51	C52	GND
		C53	C54	
		C55	C56	
		C57	C58	
		C59	C6Ø	
		C61	C62	
		C63	C64	
		C65	C66	
		C67	C68	
		C69	C7Ø	
		C71	C72	
		C73	C74	
		C75	C76	
		C77	C78	
		C79	C8Ø	
		C81	C82	
		C83	C84	
		C85	C86	
		C87	C88	
		C89	C9Ø	
		C91	C92	GND
	UNIBUS DCLO L	C93	C94	-5

BUS GRANT CHAIN AND CONTINUITY JUMPERS



BUS GRANT CHAIN AND CONTINUITY JUMPERS (CONT)

Each of the three backplane option slots is hard-wired to set CMI arbitration for the SUB at level ARB3 when it is installed and no jumpers must be added.

All of the following continuity jumpers must be in place on an empty slot. They must all be removed from the slot when the SUB or an MBA is installed.

Bus Grant Level	Remove All Jumpers
BG7	A77 to A78
BG6	A73 to A74
BG5	A69 to A70
BG4	A67 to A68

CAUTION

Before the SUB is unpacked for installation, refer to the DW780 installation procedure that specifies the use of a VELOSTAT* mat and ground cord to prevent static discharge damage.

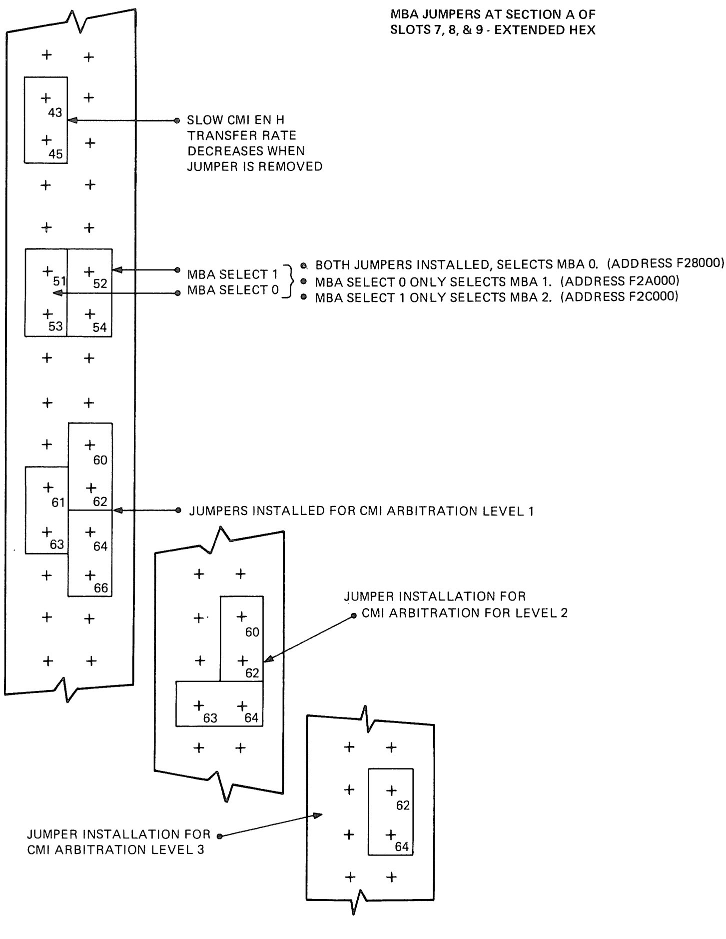
Grounding must first be made between a snap fastener on the mat and a good ground on the system. Connection is then made from a wrist strap worn by the installer to an alligator clip that is fastened to the mat.

The box is placed on the VELOSTAT mat and opened. The module is then removed from the box, the protective covering removed, and the module laid flat on the VELOSTAT mat. This brings it to the same potential as the CPU.

With the wrist strap still in place, the L0010 module is plugged into slot 7 to make cabling easier, and to place the second UNIBUS at first priority in the bus grant chain.

*VELOSTAT is a trademark of the 3M Company.

MBA INSTALLATION JUMPERS



Block diagram 182
gate arrays p65

MBA INSTALLATION JUMPERS (CONT)

MBA Select Jumpers

To Select MBA As Follows	Install These Jumpers
MBA Ø	A53 to A51 (Gnd) A54 to A52 (Gnd)
MBA 1	A53 to A51 (A54 removed)
MBA 2	A54 to A52 (A53 removed)
SLOW CMI Jumper	A45 to A43 (Gnd) normally in place. Removing this jumper reduces the rate at which the SILO is loaded and unloaded on the CMI before RUN is set and after it is cleared

MBA CMI Arbitration Jumpers

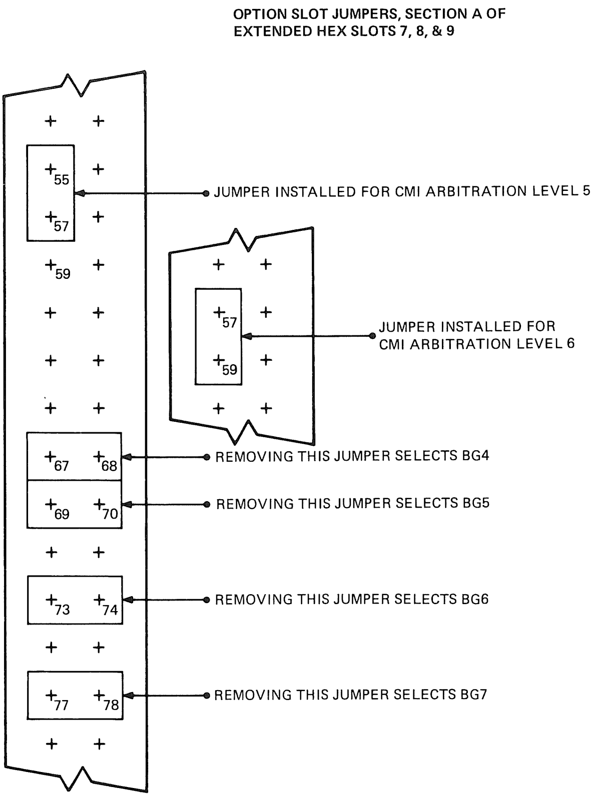
To Select ARB Level	For MBA	Install These Jumpers
ARB 3	MBA Ø	A64 to A62 (MBA ARB to CMI ARB <3>)
ARB 2	MBA 1	A64 to A63 (MBA ARB to CMI ARB <2>) A62 to A6Ø (CMI ARB <3> IN)
ARB 1	MBA 2	A64 to A66 (MBA ARB to CMI ARB <1>) A63 to A61 (CMI ARB <2> IN) A62 to A6Ø (CMI ARB <3> IN)

MBA Interrupt Priority Plug

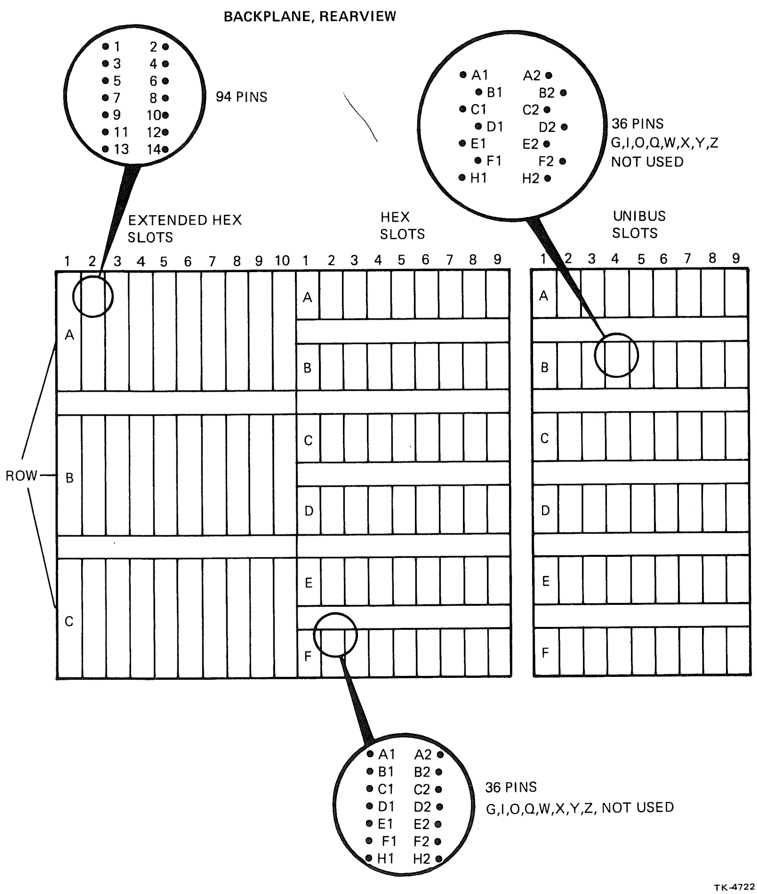
To Set MBA at BR Priority Level	Use Priority Plug Part Number
7	54-Ø8782-ØØ
6	54-Ø878Ø-ØØ
5	54-Ø8778-ØØ*
4	54-Ø8776-ØØ

*Standard BR level plug, supplied

RESERVED ARBITRATION JUMPERS

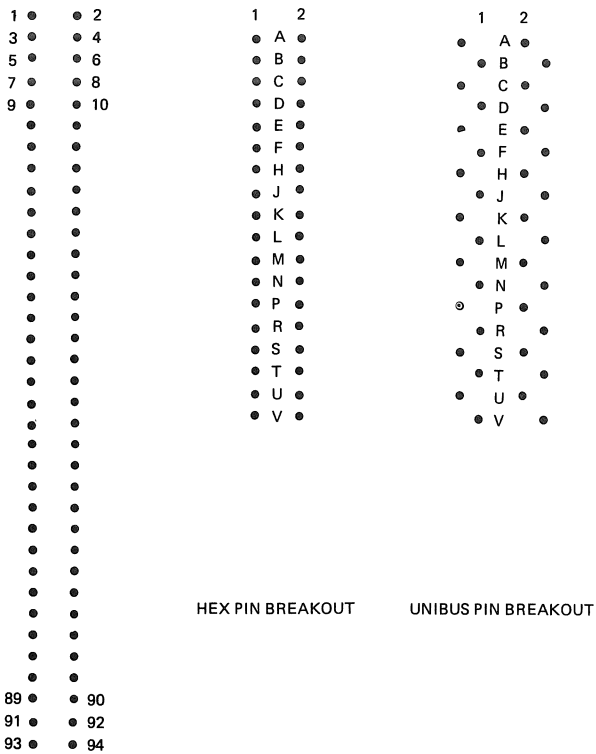


MODULE BLOCK LAYOUT



VAX-11/750 BACKPLANE (CONT)

MODULE PIN BREAKOUTS



EXTENDED HEX PIN BREAKOUT

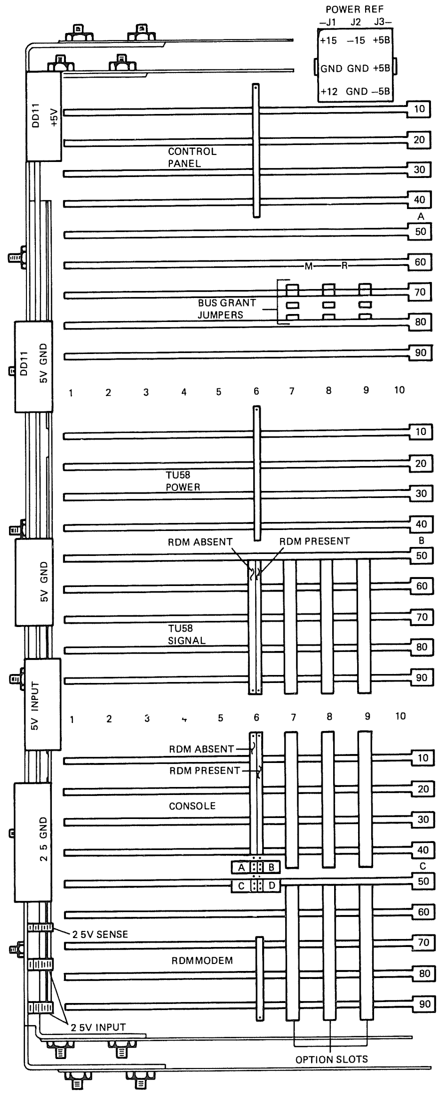
HEX PIN BREAKOUT

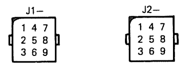
UNIBUS PIN BREAKOUT

TK-3213

VAX-11/750 BACKPLANE (CONT)

CPU BACKPLANE FROM PIN SIDE



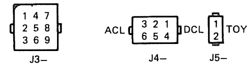


OPTION SLOT BUS GRANTS	
TO SELECT	REMOVE JUMPER
BG 4	A00X 67
BG 5	A00X 69
BG 6	A00X 73
BG 7	A00X 77
X=SLOT 7.8.9	

TEST POINTS		
RDM 19	MATCH PULSE	C00681
RDM 23	SA CLOCK	C00673
RDM 23	SA ST/SP	C00675
DPM 17	M CLOCK	B00205
DPM 17	BASE CLOCK	A00273
DPM 17	B CLOCK	B00209
MIC 04	MEM STALL	B00210
DPM 17	PHASE 1	A00590

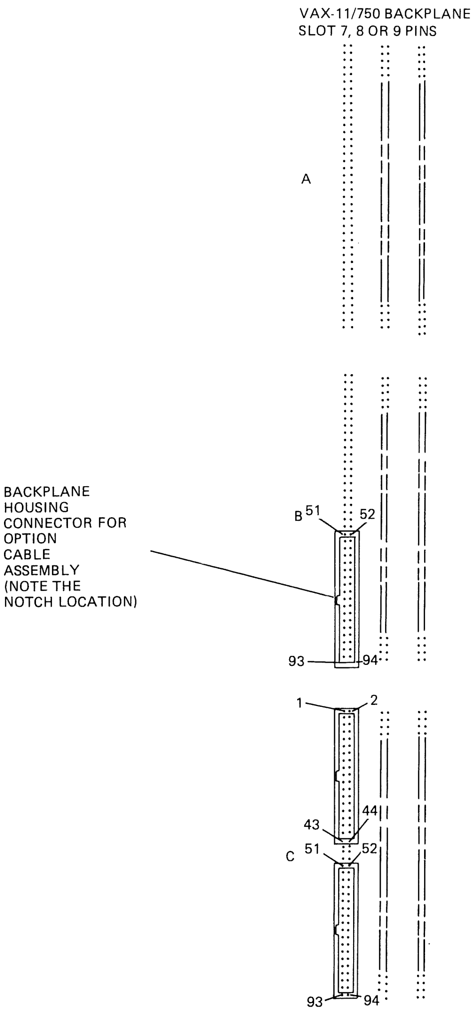
HARDWARE REV LEVEL (SYS ID)	
BIT	PIN #
0	B00456
1	B00455
2	B00454
3	B00453
4	B00450
5	B00449
6	B00448
7	B00446

CONSOLE BAUD RATE				
	CON BR			
RATE	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

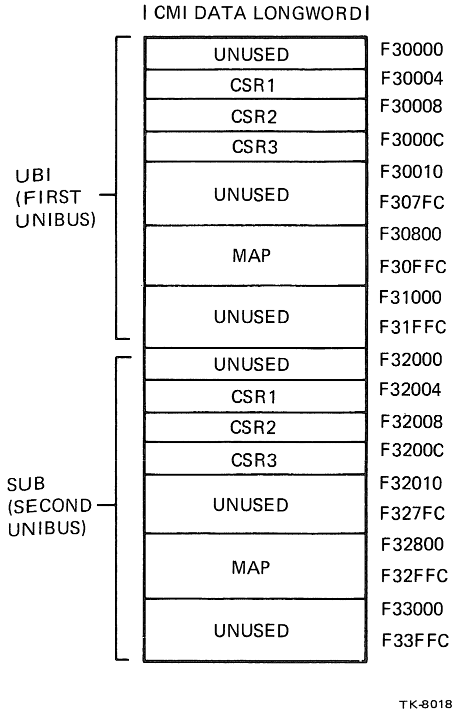


TK-4512

BACKPLANE CONNECTOR HOUSING INSTALLATION DIAGRAM



UBI AND SUB PHYSICAL ADDRESS SPACE



FIRST AND SECOND UNIBUS REGISTER SUMMARY

Block diagram p 174
Address Translation p 34

UBI Address	SUB Address	Register Name	Description/Comments
F30004	F32004	CSR 1	CSR for BDP 1
F30008	F32008	CSR 2	CSR for BDP 2
F3000C	F3200C	CSR 3	CSR for BDP 3
F30800	F32800	MAP 00	Physical base address
F30804	F32804	MAP 04	
F30808	F32808	MAP 08	The 512 MAP addresses are
F3080C	F3280C	MAP 0C	accessed in increments of
F30810	F32810	MAP 10	four through the range of
F30814	F32814	MAP 14	800-FCC (hex).
F30818	F32818	MAP 18	
F3081C	F3281C	MAP 1C	The desired MAP address may
F30820	F32820	MAP 20	be obtained by adding the
.	.	.	MAP register number to the
.	.	.	physical base address.
.	.	.	
.	.	.	
F30FF0	F32FF0	MAP 7F0	
F30FF4	F32FF4	MAP 7F4	
F30FF8	F32FF8	MAP 7F8	
F30FFC	F32FFC	MAP 7FC	
FC0000	F80000	UNIBUS	Physical base address for
.	.	.	UNIBUS I/O devices.
.	.	.	
.	.	.	The CMI address of a device
.	.	.	is obtained by converting
.	.	.	the octal device address to
.	.	.	hexadecimal and adding it to
.	.	.	the physical base address.
.	.	.	
FFFFFC	FBFFFC	UNIBUS	
FFF460	FBF460	UET/IPEC	UB address register
FFF462	FBF462	.	UB data register
FFF464	FBF464	.	UET-CR, IPEC-CR1
FFF466	FBF466	UET/IPEC	IPEC-CR2, UET PROM* data register option

*UET UB address <11:00> are loaded with the PROM address. Reading the PROM data register then returns PROM data <07:00>.

DZ FFE040
RH250 F28000
F2A000
DW750 F30000
nexus number

11/0100 & 9
10/100 .6
11/000

FIXED DEVICE ADDRESS AND VECTOR ASSIGNMENTS

Assigned UNIBUS devices normally occupy vector addresses through 274 octal (000 through 0BC hexadecimal).

Floating UNIBUS devices may be assigned vector addresses through 774 octal (0C0 through 1FC hexadecimal).

UNIBUS Device	Address Base	Address Size	Vector Base	Vector Size	No. of Units/ Comments
AA11 (1)	776750	8	140	4	1st unit
AA11	776400	8			4 extra units
AAV11	770440	4			1 unit
AD01	776770	4	130	2	1
ADF11	770460	8			1
ADV11-A	770400	2	400	4	1
AFC11	772570	4	134	2	1
AR11	770400	8			1
BDV11-CSR	777520	3			1
BDV11-LTC	777546	1			1
BDV11-ROM	773000	256			1
BM792-YA	773000	32			1
BM792-YB	773100	32			1
BM792-YC	773200	32			1
BM792-YH	773300	32			1
BM873-YA	773000	128			1
BM873-YB/YC	773000	256			1
BXV11			154	2	1 UDA class
BMV11	772150	2			1 UDA class
CD/CM/CR11	777160	4	230	2	1
CMR11	764070	4	170	2	1 CSS device
CPU			114	2	
CSR11	764000	4	270	2	1 CSS device
CSS/User	764000	1024			1
DC11	774000	4			32
DC14-D	777360	8			1
Diagnostics	760000	4			1
DIP11			260	2	
DL/DLV11-A/B	777560	4	060		1 - console
DL/DLV11-A/B	776500	4			16
DL11-C/D/E	775610	4			31
DLV11-E	775610	4			31
DLV11-F	776500	4			16
DLV11-J	776500	16			4
DL11-W	777546	1	060	4	1st LTC
DL11-W	777560	4			1 - console
DL11-W	776500	4			16
DM11-BB/BA	770500	4			16 - control
DM11	775000	4			16
DN11-AA	775200	4			16
DN11-DA	775200	1			64

FIXED DEVICE ADDRESS AND VECTOR ASSIGNMENTS (CONT)

UNIBUS Device	Address Base	Address Size	Vector Base	Vector Size	No. of Units/ Comments
DP11	774400	4			-32*
DR11-A/C	767600	4			-16*
DR11-B (1)	772410	4	124	2	2
DR11-B (2)	772430	4			
DRV11	767750	4			-3*
DRV11-B	772410	4	124	2	3
DRV11-J	764120	8			-3*
DS11	775400	67			1
DT11	777420	1			8
DV11	775000	16			4
DX11	776200	16			2
DZ11	760100	4	300	2	8
Floating Address	760010	1020			
Floating Vectors			300	xxx	
FP11	772160	8			1
FPP/FIS Exception		2	244	2	
GT40	772000	4			4
IBV11	760150	2	420	4	1
ICR/ICS11	771000	256	234	2	1
IP11/IP300	771000	128	234	2	2
KE11	777300	8			2
KG11	770700	4			8
KL11	776500	4	060	4	16
KL11	777560	4			1 - console
KPV11	777546	1			1 - LTC
KT11	772200	64			1
KT11-SR3	772516	1			1
KT error			250	2	
KU116-AA	777540	3			1
KW11-A			440	4	
KW11-L	777546	1	100	2	1
KW11-P	772540	4	104	2	1
KW11-W	772400	4			1
KWV11-A	770420	2	440	4	1
LAV11/LPV11	777514	2	200	2	1
LP/LS/LV11	777514	2	200	2	LP0
LP/LS/LV11	764004	2	170	2	LP1
LP/LS/LV11	764014	2	174	2	LP2
LP/LS/LV11	764024	2	270	2	LP3
LP/LS/LV11	764034	2	274	2	LP4
LP/LS/LV11	764044	2			LP5
LP/LS/LV11	764054	2			LP6
LP/LS/LV11	764064	2			LP7
LP20 (1)	775400	32	200	2	2
LP20 (2)			210	2	
LPA11-K	770460	8			1
LPS11	770400	16			1

*Descending assignments

FIXED DEVICE ADDRESS AND VECTOR ASSIGNMENTS (CONT)

UNIBUS Device	Address Base	Address Size	Vector Base	Vector Size	No. of Units/ Comments
M792	773000	32			8
M7930	777510	4			1
M9301-XX	765000	256			1
M9301-XX	773000	256			1
ML11	776400	22			1 - RH70/RH11
MM11	772100	1			16
MR11-DB	773100	64			1
MRV11-11	773000	256			1
MS11/MSV11	772100	1			16
NCV11	772760	8			1
OST	772500	6			1
PA611-R	772600	4	(Fltng)	8	(2 rdrs/PA611-R)
PA611-P	772700	4	(Fltng)	8	(2 puns/PA611-P)
PC11/PCV11	777550	4	070	4	1
PCL11	764200	16	170	4	1 - CSS device
PDP-11	777570	68			1
PDP-11 Reserved			000	2	
PDP-11 CPU Errors			004	2	
PDP-11 Reserved Instructions			010	2	
PDP-11 Breakpoint/ Trace Traps			014	2	
PDP-11 I/O Trap			020	2	
PDP-11 Power Fail			024	2	
PDP-11 EMT Trap			030	2	
PDP-11 TRAP Trap			034	2	
PDP-11 PIRQ			240	2	
PR11	777550	4			1
RC11	777440	8	210	2	1
Reserved	770100	32			1
Reserved	770440	8			1
Reserved	772154	2			1
Reserved	772514	1			1
Reserved	772550	8			1
Reserved	775606	1			1
Reserved	777000	56			1
Reserved	777200	32			1
Reserved	777526	1			1
REV11 (1)	777300	256			1
REV11 (2)	765000	256			1
RF11	777460	8	204	2	1
RH70/RH11-Alt.	776300	32	150	2	Alt. RS/RP/RM/TJ
RK611/RK711	777440	16	210	2	1
RK11/RKV11	777400	8	220	2	1
RL11/RLV11	774400	4	160	2	1
RLV12	774400	8	160	2	1
RM02/03/04/05	776700	22	254	2	1 - RH70/RH11
RP04/05/06	776700	22	254	2	1 - RH70/RH11
RP11	776700	16	254	2	1
RS03/RS04	772040	16	204	2	1 - RH70/RH11

FIXED DEVICE ADDRESS AND VECTOR ASSIGNMENTS (CONT)

UNIBUS Device	Address Base	Address Size	Vector Base	Vector Size	No. of Units/ Comments
RX11/211	777170	4	264	2	1
RXV11/211	777170	4	264	2	1
RSTS/E Crash Dump			144	2	
RSTS/E Statistics Pointer			234	1	
System Software Reserved			110	1	
System Software Reserved			040	8	
TA11/DIP11-A	777500	4	260	2	1
TC11	777340	8	214	2	1
TE16	772440	16	224	2	1 - RH70/RH11
Testers	770000	32			1
TM11/TMB11	772520	8	224	2	1
TR79	764000	4			1
TS11	772520	2	224	2	4
TU16/45/77	772440	16	224	2	1 - RH70/RH11
TU58	776500	4			4
TU78	775400	32	260	2	1 - RH70/RH11
UDA	772150	2	154	2	1 - UDA device
UDC11	771774	2	234	2	1
UDC units	771000	1			256
UET	772140	4			1
UNIBUS Map	770200	64			1
Unused - Reserved for DIGITAL			164	2	
User/CSS Reserved			170	4	
User/CSS Reserved			270	4	
VSV11	772000	4			4
VT48	772000	16			1
VTV01	772600	112			2
XY11	777530	4	120	2	1

FLOATING ADDRESS UNIBUS DEVICES

Rank/ Order	UNIBUS Device	Address Size
1	DJ11	4
2	DH11	8
3	DQ11	4
4	DU/DUV11	4
5	DUP11	4
6	LK11A	4
7	DMC11	4
7	DMR11	4
8	DZ/DZV11	4
8	DZ32	4
9	KMC11	4
10	LPP11	4
11	VMV21	4
12	VMV31	8
13	DWR70	4
14	RL/RLV11	4
15	LPA11-K	8
16	KW11-C	4
17	Reserved	4
18	RX11	4
18	RX211	4
19	DR11-W	4
20	DR11-B	4
21	DMP11-AD	4
22	DPV11	4
23	ISB11	4
24	DMV11-AD	8
25	UNA/QNA	4
26	UDA	2
27	DMF32	16

FLOATING VECTOR UNIBUS DEVICES

Rank/ Order	UNIBUS Device	Size
1	DC11	4
1	TU58	4
2	KL11	4
2	DL11-A	4
2	DL11-B	4
2	DLV11-J	16
2	DLV11/11-F	4
3	DP11	4
4	DM11-A	4
5	DN11	2
6	DM11-BB/BA	2
7	DH11 (ctrl)	2
8	DR11-A	4
8	DRV11-B	4
9	DR11-C	4
9	DRV11	4
10	PA611-R/P	8
11	LPD11	4
12	DT11	4
13	DX11	4
14	DL11-C	4
15	DJ11	4
16	DH11	4
17	GT40	8
18	VSV11	8
19	DQ11	4
20	KW11-W	4
20	KWV11	4
21	DU/DUV11	4
22	DUP11	4
23	DV11 (+ctrl)	6
24	LK11-A	4
25	DWUN	4
26	DMC11	4
27	DMR11	4
28	DZ11	4
28	DZ32/DZV11	4
29	LPP11	4
30	VMV21	4
31	VMV31	4
32	VTV01	4
33	DWR70	4
34	RL/RLV11	2

Rank/ Order	UNIBUS Device	Size
35	TS11	2
36	LPA11-K	4
37	IP11/IP300	2
38	KW11-C	4
39	RX11	2
39	RX211	2
40	DR11-W	2
41	DR11-B	2
42	DMP11-AD	4
43	DPV11	4
44	ISB11	4
45	DMV11-AD	4
46	UNA/QNA	2
47	UDA	2
48	DMF32	16

UNIBUS SIGNAL DESCRIPTION

Gate array layout page 61, 63

Signal Lines — Description

Data Transfer Lines

UB Address <A17:AØØ>
The address lines are asserted by the master device to select a slave (a unique memory or device register address). <A17:AØ1> address a 16-bit word; <AØØ> specifies the upper or lower byte of the word on DATOB transfers.

UB Data <D15:DØØ>
The data lines are enabled to transfer data information between the master and the slave.

Control <C1:CØ>
Two control signals are encoded by the master device to control the slave in one of four data transfer operations. Transfer direction is specified with respect to the master as follows:

C1	CØ	
Ø	Ø	Data In (DATI) - a word of data is transferred to the master from the slave.
Ø	1	Data In Pause (DATIP) - a DATI is followed by a DATO or DATOB to the same location.
1	Ø	Data Out (DATO) - a word of data is transferred from the master to the slave.
1	1	Data Out Byte (DATOB) - a byte of data is transferred from the master to the slave. The upper or lower byte is specified by <AØ>.

Parity <PB:PA>
Two parity signals are asserted by the slave on a DATI transfer as follows (PA is not currently used and is not defined):

PB	PA	
Ø	Ø	No Error
1	Ø	Parity Error
Ø	1	Reserved
1	1	Reserved

Master Sync (MSYN)
MSYN is asserted by the master to indicate to the slave that valid address and control information (along with data on a DATO/B) is present on the UNIBUS.

Slave Sync (SSYN)
SSYN is asserted by the slave in response to MSYN from the master. It indicates to the master that DATO/B data has been clocked from, or that DATI/P data is asserted on, the UB data lines.

UNIBUS SIGNAL DESCRIPTION (CONT)

Signal Lines	Description
Priority Arbitration Lines	
Nonprocessor Request (NPR)	NPR is asserted by an I/O device for a DMA transfer directly with memory. (No processor intervention is required.)
Nonprocessor Grant (NPG)	NPG is the processor response to the NPR. It indicates to the I/O device that it will be the next UNIBUS master when the current master has completed its operation.
Bus Request <BR7:BR4>	BRn is asserted by an I/O device for a processor interrupt. (Processor intervention is requested for service.)
Bus Grant <BG7:BR4>	BGn is the processor response to a BRn. It indicates to the I/O device that it will be the next UNIBUS master. Only the highest request receives a bus grant at any one time.

NOTE

All UNIBUS signals are asserted at a nominal ground (low) except for the grant signals (<NPG,BR7:BR4>) which are asserted at a nominal +3.5 V (high).

Select Acknowledge (SACK)	SACK is asserted by the NPR or BR requesting device that has received a grant. This device becomes the new bus master when the current master completes its operation and releases BBSY.
Bus Busy (BBSY)	BBSY is asserted by the current master to indicate that the UNIBUS is in use.
Interrupt (INTR)	INTR is asserted in place of MSYN by an interrupting device that has received a grant. It informs the CPU that an interrupt vector address is present on the UB data lines. (INTR is cleared on the receipt of SSYN.)

Signal Line	Description
Initialization Lines	
Initialize (INIT)	INIT is asserted by the UBI and is passed to the SUB when DCLO is asserted on the first UNIBUS (on a system power-up) or when CPU INIT is issued. It remains asserted for about 70 milliseconds after the negation of DCLO.
DC Low (DCLO)	DCLO is available from each system power supply and is asserted if an out-of-tolerance condition occurs. DCLO on the first UNIBUS asserts INIT to both the first and second UNIBUS. DCLO on the second UNIBUS returns NXM status on the CMI if the CPU attempts to access a second UNIBUS device.
AC Line Low (ACLO)	ACLO warns of an impending power failure. ACLO on the first UNIBUS initiates the power-fail trap sequence and may be used in peripheral devices to terminate operations and save data. ACLO on the second UNIBUS causes the SUB to interrupt the processor if the ACIE bit in CR1 of the IPEC is enabled.

UNIBUS CABLE PIN ASSIGNMENTS

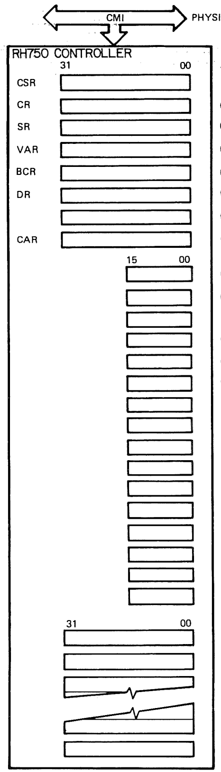
Pin	Signal Name	Pin	Signal Name
AA1	INIT L	BA1	BG6 H
AA2	+ 5 V	BA2	+ 5 V
AB1	INTR L	BB1	BG5 H
AB2	GROUND	BB2	GROUND
AC1	DØØ L	BC1	BR5 L
AC2	GROUND	BC2	GROUND
AD1	DØ2 L	BD1	GROUND
AD2	DØ1 L	BD2	BR4 L
AE1	DØ4 L	BE1	GROUND
AE2	DØ3 L	BE2	BG4 H
AF1	DØ6 L	BF1	ACLO L
AF2	DØ5 L	BF2	DCLO L
AH1	DØ8 L	BH1	AØ1 L
AH2	DØ7 L	BH2	AØØ L
AJ1	D1Ø L	BJ1	AØ3 L
AJ2	DØ9 L	BJ2	AØ2 L
AK1	D12 L	BK1	AØ5 L
AK2	D11 L	BK2	AØ4 L
AL1	D14 L	BL1	AØ7 L
AL2	D13 L	BL2	AØ6 L
AM1	PA L	BM1	AØ9 L
AM2	D15 L	BM2	AØ8 L
AN1	GROUND	BN1	A11 L
AN2	PB L	BN2	A1Ø L
AP1	GROUND	BP1	A13 L
AP2	BBSY L	BP2	A12 L
AR1	GROUND	BR1	A15 L
AR2	SACK L	BR2	A14 L
AS1	GROUND	BS1	A17 L
AS2	NPR L	BS2	A16 L
AT1	GROUND	BT1	GROUND
AT2	BR7 L	BT2	C1 L
AU1	NPG H	BU1	SSYN L
AU2	BR6 L	BU2	CØ L
AV1	BG7 H	BV1	MSYN L
AV2	GROUND	BV2	GROUND

MBA PHYSICAL ADDRESS SPACE

CMI DATA LONGWORD	
MBA #0 INT. REGISTERS	F28000
MBA #0 EXT. REGISTERS	F28400
MBA #0 MAP REGISTERS	F28800
UNUSED	F28C00 F29FFC
MBA #1 INT. REGISTERS	F2A000
MBA #1 EXT. REGISTERS	F2A400
MBA #1 MAP REGISTERS	F2A800
UNUSED	F2AC00 F2BFFC
MBA #2 INT. REGISTERS	F2C000
MBA #2 EXT. REGISTERS	F2C400
MBA #2 MAP REGISTERS	F2C800
UNUSED	F2CC00 F2DFFC

TK-6406

MBA AND MASSBUS REGISTER SUMMARY



REGISTER ADDRESS	MNEMONIC	INTERNAL REGISTERS	RH750 ADDRESS
00	CSR	CONFIGURATION (NOT USED)	F28000
01	CR	CONTROL	004
02	SR	STATUS	008
03	VAR	VIRTUAL ADDRESS	00C
04	BCR	BYTE COUNT	010
05	DR	DIAGNOSTIC	014
06		UNUSED REGISTER	018
07	CAR	COMMAND/ADDRESS	F2801C
	EXTERNAL REGISTERS		
00	ERB + UNIT & REGISTER OFFSETS	DRIVE CONTROL	F28400
01			404
02			408
03			40C
04		ATTENTION SUMMARY	410
05			414
06			418
07			41C
08			420
09			424
0A			428
0B			42C
0C			430
0D			434
0E			438
0F			F2843C
	MAP REGISTERS		
00	MAP + OFFSET (800)		F28800
01		(804)	804
|		|	|
FE		(BF8)	BF8
FF		(BFC)	F28BFC

NOTE: THE ADDRESSES GIVEN HERE ARE FOR THE FIRST MBA AND DRIVE. THE OFFSET VALUES FOR OTHER MBA's AND DRIVE UNITS CAN BE CALCULATED AS SHOWN BELOW.

TO FIND THE OFFSET FOR A PARTICULAR DRIVE UNIT, ADD THE APPROPRIATE VALUE BELOW TO THE EXTERNAL REGISTER ADDRESS:

DR0 = 000	DR4 = 200
DR1 = 080	DR5 = 280
DR2 = 100	DR6 = 300
DR3 = 180	DR7 = 380

TO SELECT A CONTROLLER THIS HEX NO. IS:

MBA0 = 8
MBA1 = A
MBA2 = C

TK-7172

MBA REGISTER OFFSETS

MBA PHYSICAL BASE ADDRESSES

MBA Number	Physical Base Address (Hex)	Vector Address (Hex) BR4	BR5*	BR6	BR7
MBA Ø	F28ØØØ	11Ø	15Ø	19Ø	1DØ
MBA 1	F2AØØØ	114	154	194	1D4
MBA 2	F2CØØØ	118	158	198	1D8

} set by backplane jumpers

*Standard BR level is 5.

MBA INTERNAL REGISTER OFFSETS

To Access Internal Register	Add to MBA Physical Base Address	
ØØ - Configuration/Status (CSR)	ØØØ	Read Only
		(unused, reads as zeros)
Ø1 - Control (CR)	ØØ4	Read/Write
Ø2 - Status (SR)	ØØ8	Read Only
Ø3 - Virtual Address (VAR)	ØØC	Read/Write
Ø4 - Byte Count (BCR)	Ø1Ø	Read/Write
Ø5 - Diagnostic (DR)	Ø14	Read/Write
Ø6 - Unused Register Address	Ø18	Returns NXM status to CPU
Ø7 - Command/Address (CAR)	Ø1C	Read Only

MBA MAP REGISTER OFFSETS

To Access MAP Register	Add to MBA Physical Base Address	
MAP ØØ	8ØØ	
MAP Ø1	8Ø4	
--	--	The 256 MAP register addresses
--	--	are accessed in increments of
--	--	four within the range of 8ØØ
--	--	through BFC (hex).
MAP FE	BF8	
MAP FF	BFC	

Block diagram p183

(controller registers)

MBA EXTERNAL (MASSBUS) REGISTER OFFSETS

Drive Register	Drive Unit Number 0	1	2	3	4	5	6	7
00	400	480	500	580	600	680	700	780
01	404	484	504	584	604	684	704	784
02	408	488	508	588	608	688	708	788
03	40C	48C	50C	58C	60C	68C	70C	78C
04	410	490	510	590	610	690	710	790
05	414	494	514	594	614	694	714	794
06	418	498	518	598	618	698	718	798
07	41C	49C	51C	59C	61C	69C	71C	79C
08	420	4A0	520	5A0	620	6A0	720	7A0
09	424	4A4	524	5A4	624	6A4	724	7A4
0A	428	4A8	528	5A8	628	6A8	728	7A8
0B	42C	4AC	52C	5AC	62C	6AC	72C	7AC
0C	430	4B0	530	5B0	630	6B0	730	7B0
0D	434	4B4	534	5B4	634	6B4	734	7B4
0E	438	4B8	538	5B8	638	6B8	738	7B8
0F	43C	4BC	53C	5BC	63C	6BC	73C	7BC
10	440	4C0	540	5C0	640	6C0	740	7C0
11	444	4C4	544	5C4	644	6C4	744	7C4
12	448	4C8	548	5C8	648	6C8	748	7C8
13	44C	4CC	54C	5CC	64C	6CC	74C	7CC
14	450	4D0	550	5D0	650	6D0	750	7D0
15	454	4D4	554	5D4	654	6D4	754	7D4
16	458	4D8	558	5D8	658	6D8	758	7D8
17	45C	4DC	55C	5DC	65C	6DC	75C	7DC
18	460	4E0	560	5E0	660	6E0	760	7E0
19	464	4E4	564	5E4	664	6E4	764	7E4
1A	468	4E8	568	5E8	668	6E8	768	7E8
1B	46C	4EC	56C	5EC	66C	6EC	76C	7EC
1C	470	4F0	570	5F0	670	6F0	770	7F0
1D	474	4F4	574	5F4	674	6F4	774	7F4
1E	478	4F8	578	5F8	678	6F8	778	7F8
1F	47C	4FC	57C	5FC	67C	6FC	77C	7FC

NOTE: The offset value, when added to the MBA physical base address, is transmitted as the drive unit and register select codes on the MASSBUS control bus.

MASSBUS SIGNAL DESCRIPTION

gate array layout p64

Signal Line	Description
Data Bus Section	
Data Bus Data <D15:D00>	The parallel data path transfers DMA data between the MBA and the drives. The MBA transfers 16 bits (2 bytes) at a time and the active drive must have its "16-bit format" control bit set. Bits <D17:D16> are asserted as zeros by the MBA for writes to drives that perform parity checking.
Data Bus Parity (DPA)	DPA is the odd-parity bit for data bus data. It is asserted by the MBA on write to drive (WTD) transfers. It is asserted by the drive on read from drive (RFD) or write check (WCK) transfers.
RUN	When a data transfer command is written to a drive's control register, the drive connects to the data bus and asserts OCC. RUN is then asserted by the MBA to start transfers. RUN is clocked by the drive at the end of each block or sector on the trailing edge of EBL. If RUN is still asserted, the function continues to the next block or sector. If not, the operation terminates.
Occupied (OCC)	OCC is asserted by the drive that has received a valid data transfer command and is connected to the data bus is active). If the drive is not able to set OCC because of an error condition, the MBA times out and MXF is set in the MBA status register. OCC is negated on the trailing edge of the last EBL of the operation.
End of Block (EBL)	EBL is asserted by the drive for two microseconds after the final SCLK of each block or sector. EBL may be asserted earlier on error conditions where it is necessary to terminate the operation immediately.
Exception (EXC)	EXC is asserted by the active drive when an abnormal condition occurs during DMA transfers. EXC is asserted at or before EBL and is negated on the trailing edge of EBL. Also, on the assertion of EBL, MB EXC is set in the MBA status register. EXC distinguishes errors in the active drive during DMA transfers from error or status changes signaled by the ATTN line. An active drive does not assert ATTN during data transfers (see ATTN description) until the operation has been aborted or terminated and the drive is no longer active.

Signal Line	Description
Sync Clock (SCLK)	SCLK is generated by the active drive to control the data bus data clocking in the MBA. On read from drive (RFD) or write check (WCK) transfers, the drive changes data on the assertion of SCLK. The MBA then clocks data on the negation of SCLK.
Write Clock (WCLK)	The MBA receives SCLK and returns it to the drive as WCLK on write to drive (WTD) transfers. The MBA changes data on the negation of WCLK. The drive then clocks data on the assertion of WCLK.

Control Bus Section

Control Bus Data <C15:CØØ>	The parallel data path transfers control and status information between the MBA and the drive on CPU reads or writes to a drive's control/status logic.
Control Bus Parity (CPA)	CPA is the odd-parity bit for control bus data. It is asserted by the MBA on a CPU write to a drive's control or status register. It is asserted by the drive on a CPU read of a drive's control or status register.
Control to Drive (CTOD)	CTOD is asserted by the MBA when the transfer is a CPU write (from the MBA to the drive). CTOD is negated by the MBA when the transfer is a CPU read (from the drive to the MBA).
Drive Select <DS2:DSØ>	The MBA asserts a three-bit code to select one of eight possible drive units.
Register Select <RS4:RSØ>	The MBA asserts a five-bit code to select a register within the drive.
Demand (DEM)	DEM is asserted by the MBA to perform a control bus transfer with a drive. It is asserted after the RS, DS, and CTOD lines are settled (and the control bus data lines are settled if the transfer is a CPU write).
Transfer (TRA)	TRA is asserted by the drive in response to DEM from the MBA. It is asserted when the drive clocks control to drive data to the selected register on a CPU write. It is asserted after drive to control data is asserted on the control bus data lines from the selected register on a CPU read. It is negated when the negation of DEM has been received from the MBA.

Signal Line	Description
Attention (ATTN)	ATTN is asserted by any inactive drive as a result of a change of status such as the completion of a motion command or an error. ATTN is also asserted by a previously active drive when its GO bit is cleared after a transfer that produces an error. ATTN is a request for service by processor interrupt. The attention active (ATA) status bit for each drive is read from the attention summary register to determine the requesting drive(s).
Initialize (INIT)	INIT is asserted by the MBA to perform a system reset of all drives. It is asserted on a system power-up, or by writing a one to the INIT bit of the MBA control register. It must not be written while RUN is set since the active drive will abort execution of the current command and perform drive clear command functions. In a dual-port drive, the drive accepts the INIT command only from the MBA that has control. It accepts the command from either control when in the idle state.
FAIL	FAIL is asserted by the MBA to indicate that a system power-fail condition exists or that the MBA is in the maintenance mode. While FAIL is asserted, all drives ignore assertion of the INIT or DEM signals.

address translation
p. 39

MASSBUS CABLE PIN ASSIGNMENTS

I/O Cable	Pin*		Polarity	Signal Name
Cable A	A	1	-	MASS DØØ
	B	2	+	
	C	3	+	MASS DØ1
	D	4	-	
	E	5	-	MASS DØ2
	F	6	+	
	H	7	+	MASS DØ3
	J	8	-	
	K	9	-	MASS DØ4
	L	1Ø	+	
	M	11	+	MASS DØ5
	N	12	-	
	P	13	-	MASS CØØ
	R	14	+	
	S	15	+	MASS CØ1
	T	16	-	
	U	17	-	MASS CØ2
	V	18	+	
	W	19	+	MASS CØ3
	X	2Ø	-	
	Y	21	-	MASS CØ4
	Z	22	+	
	AA	23	+	MASS CØ5
	BB	24	-	
	CC	25	-	MASS SCLK
	DD	26	+	
	EE	27	+	MASS RS3
	FF	28	-	
	HH	29	+	MASS ATTN
	JJ	3Ø	-	
	KK	31	-	MASS RS4
	LL	32	+	
	MM	33	-	MASS CTOD
	NN	34	+	
	PP	35	+	MASS WCLK
	RR	36	-	
	SS	37	+	MASS RUN
	TT	38	-	
	UU	39		Spare
	VV	4Ø	GND	Ground

*Alternate pin assignments

NOTE

MASSBUS cables are installed as labeled.

I/O Cable	Pin*		Polarity	Signal Name
Cable B	A	1	-	MASS DØ6
	B	2	+	
	C	3	+	MASS DØ7
	D	4	-	
	E	5	-	MASS DØ8
	F	6	+	
	H	7	+	MASS DØ9
	J	8	-	
	K	9	-	MASS D1Ø
	L	1Ø	+	
	M	11	+	MASS D11
	N	12	-	
	P	13	-	MASS CØ6
	R	14	+	
	S	15	+	MASS CØ7
	T	16	-	
	U	17	-	MASS CØ8
	V	18	+	
	W	19	+	MASS CØ9
	X	2Ø	-	
	Y	21	-	MASS C1Ø
	Z	22	+	
	AA	23	+	MASS C11
	BB	24	-	
	CC	25	-	MASS EXC
	DD	26	+	
	EE	27	+	MASS RSØ
	FF	28	-	
	HH	29	+	MASS EBL
	JJ	3Ø	-	
	KK	31	-	MASS RS1
	LL	32	+	
	MM	33	-	MASS RS2
	NN	34	+	
	PP	35	+	MASS INIT
	RR	36	-	
	SS	37	+	MASS SP1
	TT	38	-	
	UU	39		Spare
	VV	4Ø	GND	Ground

*Alternate pin assignments

NOTE

MASSBUS cables are installed as labeled.

MASSBUS CABLE PIN ASSIGNMENTS (CONT)

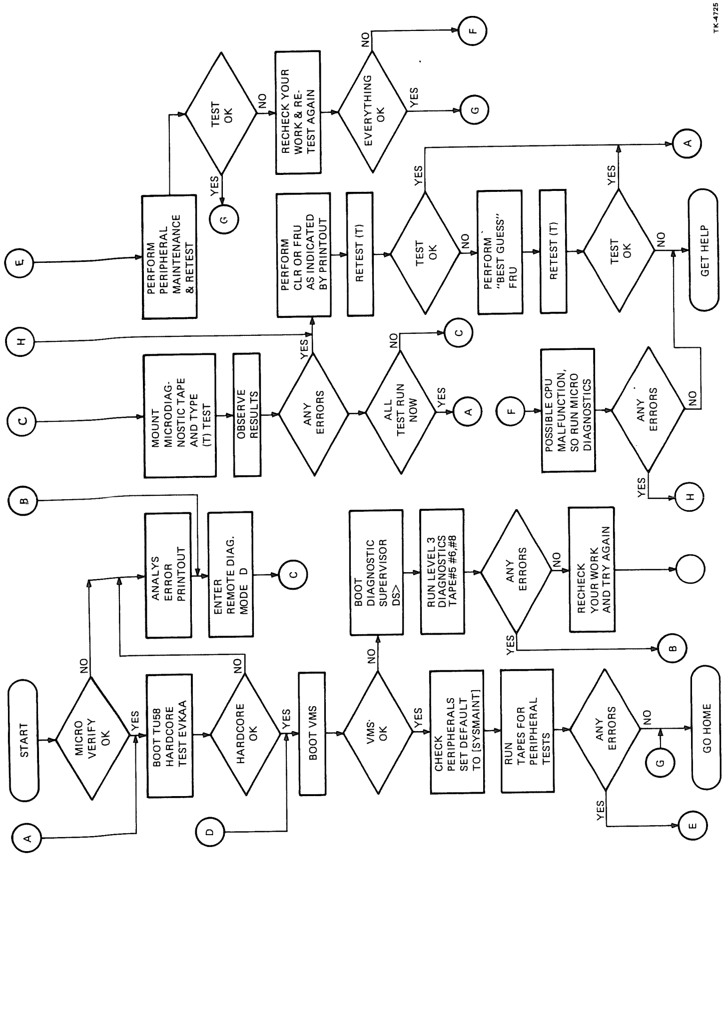
I/O Cable	Pin*		Polarity	Signal Name
Cable C	A	1	-	MASS D12
	B	2	+	
	C	3	+	MASS D13
	D	4	-	
	E	5	-	MASS D14
	F	6	+	
	H	7	+	MASS D15
	J	8	-	
	K	9	-	MASS D16
	L	1Ø	+	
	M	11	+	MASS D17
	N	12	-	
	P	13	-	MASS DPA
	R	14	+	
	S	15	+	MASS C12
	T	16	-	
	U	17	-	MASS C13
	V	18	+	
	W	19	+	MASS C14
	X	2Ø	-	
	Y	21	-	MASS C15
	Z	22	+	
	AA	23	+	MASS CPA
	BB	24	-	
	CC	25	-	MASS OCC
	DD	26	+	
	EE	27	+	MASS DSØ
	FF	28	-	
	HH	29	+	MASS TRA
	JJ	3Ø	-	
	KK	31	-	MASS DS1
	LL	32	+	
	MM	33	-	MASS DS2
	NN	34	+	
	PP	35	+	MASS DEM
	RR	36	-	
	SS	37	+	MASS SP2
	TT	38	-	
	UU	39	H (high)	MASS FAIL H
	VV	4Ø	GND	Ground

*Alternate pin assignments

NOTE
MASSBUS cables are installed as labeled.

CHAPTER 9
TROUBLESHOOTING AIDS

SYSTEM TROUBLESHOOTING FLOW



UNIBUS TROUBLESHOOTING WITH THE UET OR IPEC

(SECOND UNIBUS)

First UNIBUS - The following sequence of console commands causes the UET to execute a single NPR read (DATI transfer) from memory.

1.

>>>D/I 37 1	Initialize CPU (page 27)
>>>D/W/P F3ØØØ4 1	Purge UBI BDP 1 (page 32)
>>>D/L/P F3Ø8ØØ 8Ø2ØØØØ8	Set up UBI MAP address Ø for (p 33) validity, BDP 1, PFN 1ØØØ
>>>D/L/P 1ØØØ 12345678	Load data to memory 1ØØØ
>>>D/W/P FFF46Ø Ø	Set UET bus address to Ø (p. 35)
>>>D/W/P FFF464 1	Set UET NPR (GO) bit

2. Test the results:

>>>E/W FFF462	Examine UET data register (should return 5678 as data)

3. Increment the bus address register:

>>>D/W/P FFF46Ø 2	Set UET bus address to 2
>>>D/W/P FFF464 1	Set UET NPR (GO) bit

4. Test the results:

>>>E/W FFF462	Examine UET data register (should return 1234 as data)

Second UNIBUS - This sequence causes the IPEC to execute a single NPR read (DATI transfer) from memory.

1.

>>>D/I 37 1	Initialize CPU
>>>D/W/P F32ØØ4 1	Purge SUB BDP 1
>>>D/L/P F328ØØ 8Ø2ØØØØ8	Set up SUB MAP address Ø for validity, BDP 1, PFN 1ØØØ
>>>D/L/P 1ØØØ 12345678	Load data to memory 1ØØØ
>>>D/W/P FBF46Ø Ø	Set IPEC bus address to Ø
>>>D/W/P FBF464 1	Set IPEC NPR (GO) bit

2. Test the results:

>>>E/W FBF462	Examine IPEC data register (should return 5678 as data)

3. Increment the bus address register:

>>>D/W/P FBF46Ø 2	Set IPEC bus address to 2
>>>D/W/P FBF464 1	Set IPEC NPR (GO) bit

4. Test the results:

>>>E/W FBF462	Examine IPEC data register (should return 1234 as data)

The following substitutions can be made in either program to test other data path and MAP address combinations.

MAP address:	UBI	SUB	
	F30800	F32800	MAP physical base address
	F30804	F32804	(MAP locations are accessed in increments of four)
	.	.	
	.	.	
	F30FF8	F32FF8	
	F30FFC	F32FFC	
MAP data:	DDP	- 80000008	(Direct data path)
	BDP 1	- 80200008	
	BDP 2	- 80400008	
	BDP 3	- 80600008	
CSR data:	UBI	SUB	
	F30000	F32000	Base (no CSR for the DDP)
	F30004	F32004	CSR 1 for BDP 1
	F30008	F32008	CSR 2 for BDP 2
	F3000C	F3200C	CSR 3 for BDP 3
Exerciser address:	UET	IPEC	
	FFF460	FBF460	Bus address register
	FFF462	FBF462	Bus data register
	FFF464	FBF464	CR1*
	FFF466	FBF466	IPEC CR2, UET PROM data register

*Other operations can be performed using combinations of the following control register bits:

CR<0> = NPR (GO) bit
CR<2:1> = <C1:C0> from control bits
CR<4:3> = <A17:A16> bus address bits

<C1>	<C0>	Function
0	0	DATI
0	1	DATIP
1	0	DATO
1	1	DATOB

To do a DATO or DATOB with either program, load the bus data register in place of the memory deposit with one word of data (>>>D/W/P FFF462 or FBF462).

THE FILEX UTLIITY

FILEX may be used to copy files from TU58 to disk or from disk to TU58. It can be used to build new files or when existing files have been corrupted. All programs must first be in place; the boot block is then added using the WRITEBOOT facility.

Disk to Tape - To copy from the default disk drive to a TU58 cassette, do the following (FILEX uses only 'ddu' and does not use controller codes).

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>CONNECT CONSOLE
SYSGEN>EXIT
$ MOUNT CS1:/FOREIGN
$ SET DEFAULT [SYSMAINT]
$ MCR FLX
FLX>CS1:/RT/LI             (List tape directory - optional)
FLX>CS1:/RT/ZE             (Zero tape directory - optional)
FLX>CS1:/RT/XX = [device]filename.ext/RS
    (to device)     (from device)

        XX Command Codes
        LI - LIST   (or DI)
        IM - Image mode (EXE, ULB, SML, SYS, OLB, TSK)
        DE - Delete specified file
        FB - Formatted binary (OBJ, STB, BIN, LDA)
        FA - Formatted ASCII (all other extensions)
        CO - Contiguous file to disk (file coming from
             disk to tape are always contiguous)
        RS - RS11 format (system)
        RT - RT11 format (tape)
        DO - DOS  (TU77, TU78)
FLX>^Y                (when operation is complete)
$
```

Tape to Disk (RKØ7) - To copy from a TU58 cassette to the default disk, do the following:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>CONNECT CONSOLE
SYSGEN>EXIT
$ MOUNT CS1:/FOREIGN
$ SET DEFAULT [SYSMAINT]
$ MCR FLX
FLX>DMO:/RS/CO = CS1:filename.ext/RT
    (to device)    (from device)

FLX>^Y        (when operation is complete)
$
```

THE WRITEBOOT UTILITY

The WRITEBOOT utility may be used to write a boot block on any bootable disk. A bootstrap operation is normally performed from the the boot block that is block Ø of the system disk. If block Ø is bad, the system can only be booted from the TU58. When the system is running again, the boot block can be rewritten using WRITEBOOT.

> NOTE
> The operator must have LOG_IO privilege to use the WRITEBOOT facility.

To call WRITEBOOT, type the following command.

```
$ RUN SYS$SYSTEM:WRITEBOOT
```

The program then asks three questions.

1. Target system device:

   Enter the name of the device that will have the boot block written and the boot file name if not VMB.EXE (for example: DMAØ or DMAØ: filename.ext). Specify the target system device 'ddcu' as for any other VAX/VMS device.

   ```
   dd -  Device type
    c -  Controller designation
    u -  Unit number
   ```

   An example: 'DMAØ' is RKØ7 unit Ø on RK611 controller A.

2. Enter VBN of boot file code:

   To rewrite the boot block, press RETURN (default is 1).

3. Enter load address of primary bootstrap in hex:

   To rewrite the boot block, press RETURN (default is 2ØØ).

WRITEBOOT then writes the boot file specified on the first line to the address specified on the third line.

WRITEBOOT may also be used to write a boot block on a TU58 cassette tape using these specifications:

Target system device: ddcu: filename.ext

(ddcu = CSA1: for console TU58, or ddcu of UNIBUS drive)

Enter VBN of boot file code: 1 or 2

(default is 1 if RETURN is pressed)

Enter load address of primary bootstrap in hex:

(default is 200 if RETURN is pressed)

Level 4 monitors and diagnostics are the only bootable programs from TU58. Four bootable programs are currently available and are written to a TU58 cassette tape using these parameters:

Filename.ext	VBN	Load Address	Description
EVKAA.EXE	2	200	Hardcore tests
ECKAL.EXE	2	200	TB and cache
ECSAA.EXE	2	10000	Diagnostic supervisor
BOOT58.EXE	1	C000	BOOT58 monitor

Boot 58 commands p 120.

MACHINE CHECKS

The CMIERR bit of IPR 17, when set, indicates multiple TB or cache parity errors. These indications may also appear in operator requested shutdown bugchecks when the system error log is examined, although there is nothing wrong with the system.

CMIERR is only valid when the system halts at the machine check vector after one of the following.

- Memory error
- TB parity error
- Cache parity error
- Control store parity error

CMIERR summarizes the pertinent MEMSCR registers of the L0003 module. The MEMSCR registers are D type latches that follow the input signal level and false indications sometime occur when the macro code does an MFPR from this register. Two forced machine checks are included here for illustration.

TRANSLATION BUFFER OR BUS ERROR

This machine check was caused by mapping a nonexistent page with a $CRMPSC system service and then accessing that page in user mode.

Exception PC	0000045C	
Error PSL	03C00000	Interrupt priority level = 0 Previous mode = User Current mode = User
Summary code	00000002	Translation buffer or bus error
VA last ref	00000600	
PC at error	0000045F	
MDR	00000000	
SMR	0000000B	CPU mode = User Virtual Read
RLTO	00000000	
TBGPR	00000000	
CAER	00000000	
BER	00000008	Memory error
MCESR	00000008	Operand reference Bus error

In the VAX-11/750 machine check logout (compared to the VAX-11/780), no translated address is pushed onto the stack. This is because the physical address latch is contained within gate arrays and there is no direct method of obtaining the physical address.

The MCESR register may be read to determine what type of exception condition occurred. The bits are defined as follows.

	<03>	<02>	<01>	<00>
MCESR	MEM ERR	TB PARITY	0	XB FETCH

If bit <3> is a one in the stack logout, an error occurred referencing memory for read data. The BER register logout may then be read to determine the type of memory error.

The BER register will indicate what type of error occurred. Bit <3> indicates a memory while bit <2> indicates an uncorrectable error. Correctable errors will interrupt the processor (via SCBB+54) and log the error while uncorrectable errors will cause machine checks.

	<03>	<02>	<01>	<00>
BER	MEM ERROR	UNCORR ERROR	LOST ERROR	CORR ERROR

If the uncorrectable error bit is not set (as in this example) it can be assumed that it was a reference to a nonexistent location. If the TB parity error bit <2> in the MCESR register is a one, some type of TB parity error caused the machine check.

The TBGPR can be read to determine what the error was. If machine check was not caused by a memory error or TB parity error, a cache parity error is the only other possibility. The CGPR contains the data or tag parity information.

Several other machine checks may occur in the VAX-11/750.

a. Summary parameter = to 7 — This error occurs only if the instruction decode ROM is not accessed at IRD1 time. The mechanism only works if there is no CS parity error and the NEXT field in an IRD1 ROM state is used instead of the IRD ROM output.

b. Summary parameter = to 6 — This is a microsequencing error that occurs if the microcode jumps to filler words in the control store. R6 contains the error code or filler word accessed. This only happens if there is no CS parity error.

The following description illustrates the last possible occurrence of machine checks, the control store parity error. Note that the VAX-11/750 machine check logout does not have the micro PC where the machine check occurred pushed on the stack. This information cannot be saved because the VAX-11/750 does not have a UPC save register like the VAX-11/780. It is possible, however, to troubleshoot intermittent CS parity error machine checks by using the RDM as follows.

```
^P                ; stop the application
XXXXXXXX Ø2
>>>^D
RDM> SE 2Ø        ; set stop on micro match at CS address ØØ2Ø

RDM> RET/D        ; return to previous mode
>>>C              ; continue running application
```

When the CS parity error occurs, the following message is printed at the console.

```
CLOCK STOPPED CSAD ØØ2Ø  NEXT ØF56

RDM> TR           ; the TRACE command will permit you to see a
                  ; trace of where the micromachine has been

CSAD ØØ2Ø         NEXT ØF56
     XXXX         ; XXXX is the microinstruction causing the CS
     YYYY         ; parity error
     YYYY
      .
      .
      .
     YYYY         ; 64 microinstructions ago

RDM>
```

Repeat this process to be certain that the error occurs in the same place. Swapping the LØØØ5 module will usually prevent this type of problem. The following is a description of the CS parity error machine check.

CONTROL STORE PARITY ERROR

This machine check was forced by clearing a single bit of a microinstruction stored in the WCS. A G and H floating-point mathematics program was performed and when the microinstruction with bad parity was executed, the machine check occurred.

Exception PC	ØØØØØB9C	
Error PSL	Ø3CØØØØ4	Z-bit Interrupt priority level = Ø Previous mode = User Current mode = User
Summary code	ØØØØØØØ1	Control store parity error
VA last ref	ØØØØØ46C	
PC at error	ØØØØØBA8	
MDR	FFFFFFFF	
SMR	ØØØØØØØB	CPU mode = User Virtual Read
RLTO	ØØØØØØØØ	
TBGPR	ØØØØØØØØ	
CAER	ØØØØØØØ1	Cache hit
BER	ØØØØØØØØ	
MCESR	ØØFDØØØØ	Operand Reference Opcode mnemonic = ESCD

On this type of machine check exception, all stack information does not help. The summary parameter indicates that a control store parity error occurred and there is no way to determine what happened.

All that can be shown is in the machine check error summary register (MCESR). Note that byte 2 contains the letters 'FD'. These bits are the opcode of the instruction that caused the machine check. VMS writes the opcode into the image of the MCESR in memory so that SYE can determine the opcode of the instruction that saw the control store parity error.

WRITING SPRS

GENERAL GUIDELINES

Software Performance Reports (SPRs) provide feedback to DIGITAL on problems with software and provide help to customers with difficulties they are having.

The following guidelines cover the information that should be provided with all SPRs. Depending on the problem, this information will vary in quantity and content. Remember that the more pertinent the information included, the easier it will be to resolve the problem.

1. Scenario

   The user should supply a complete scenario, usually in the form of a batch log or console listing, that will show exactly how the problem was produced. Supplying only the output produced by the problem is not enough. The problem may be caused by an interaction between various system events, software packages, devices, SYSGEN parameters, DCL symbols, or logical names. Some or all of the displays generated by the following commands may be required for different problems:

   ```
   $SHOW LOGICAL /ALL
   $SHOW SYMBOL /ALL/GLOBAL

   $RUN SYS$SYSTEM:SYSGEN
   USE ACTIVE
   SHOW /ALL
   SHOW /SPECIAL
   ```

2. Limit Problem Scope

   The user should (as much as possible) eliminate all extra elements from the scenario. For example, if the execution of a very large program causes a problem, the user should shorten the program to include only the code that causes th problem or write a small program that demonstrates the problem. This action has two benefits: first, the user may trap logic errors, and second, the maintainer looking into the problem does not have to understand unnecessary material.

3. Machine Readable Files

If possible, supply any software needed to reproduce the problem. This may include source programs, image files, sample data, or command procedures. If source programs are submitted, also include any libraries or require files referenced. These files must be provided in machine readable format. The console media or an ANSI magtape are the best media to include with the SPR.

If the problem involves a system crash, include the system dump.

The data should be written onto an ODS-2 format disk or an ANSI magtape. For example, the following commands will copy the system dump file to an ANSI magtape:

```
$INIT MTAØ:  DUMPS
$MOUNT MTAØ:  DUMPS
$COPY SYS$SYSTEM:SYSDUMP.DMP MTAØ:
$DISMOUNT MTAØ:
```

To copy files to the console medium, use the following commands:

```
$RUN SYS$SYSTEM:SYSGEN
CONNECT CONSOLE

(At this time, remove the console medium and
place a scratch volume in the console device.)

$INIT CSA1:  SPRDATA
$MOUNT CSA1:  SPRDATA
$CREATE/DIRECTORY CSA1:[DUMP]
$COPY MYDTA.DAT,MYIMAGE.EXE CSA1:[DUMP]
$DISMOUNT CSA1:
```

When machine readable data is provided in another format, the user should include the exact commands that were used to write the data and the commands used to read it. Other formats cause problems and should not be used. For example, using FLX without the /IM switch will create a dump file that is completely unusable, because FLX eliminates all bytes of zero in a file.

All machine readable media submitted with SPRs will be returned to the customer.

4. System Environment

Every computer site runs a different type of workload. Some problems only appear under certain conditions. For example, some sites give different classes of users different base priorities. These sites may have problems that other sites do not. This information can be very important in resolving the problem, especially for system hangs or crashes.

The user should describe any special software packages being used. Any unusual hardware devices or user written drivers should also be mentioned.

Software version numbers should be included. For example, if there is a problem with accessing local symbols during a DEBUG session, the version numbers of DEBUG and all compilers/assemblers should be specified.

If any patches other than from maintenance updates are being used, they should be mentioned in the SPR.

5. User Analysis (Optional)

Optionally, the user may include an analysis of the problem. Any useful miscellaneous information should be included such as, "Without xyz's happening, the problem could not be reproduced" or "On version Vx.y, this problem does not occur."

6. Problem Specific Information to Include

Solving different types of problems will require different kinds of information. The following table shows the information typically needed for different types of problems.

Problem	Information to Include
System bugcheck/crash	A machine readable copy of the system dump file MUST be included.* (Output from the SDA utility should NOT be sent because it usually does not include enough information to resolve the problem.) A copy of the error log at the time of the error should also be included because many system problems are triggered by hardware errors.**
Machine check	On a machine check, include a copy of the error log.** A machine readable copy of the system dump file should also be included.*
System hang	When a system appears "hung" (no response on any terminals), the system should be manually crashed and the system dump file included with the SPR. When the system is shut down in this way, the console listing is very important and should be included with the SPR. On the VAX-11/780 console terminal, enter: ^P HALT @CRASH On the VAX-11/750 console terminal, enter: ^P E P E/I 0 E E E E D/G F FFFFFFFF D P 1F0000 C

This will cause the system to bugcheck in what is recognized by VMS developers as a forced crash.

A description of the currently running workload should also be included.

Executive

If the user suspects a problem with executive code, include the active values of the system parameters. These can be obtained by invoking SYSGEN and entering both the SHOW/ALL and SHOW/SPECIAL commands.

A machine readable copy of the source program showing the problem plus libraries, require files, and build files should also be included, if possible.

Include a copy of the error log at the time of the problem.**

Devices

For any suspected device or device driver error, include a copy of the error log at the time of the problem.**

Files

If the problem appears to be with a file, information (DIRECTORY/FULL) on that file and its directory should be included. If possible, include a machine readable copy of the file itself.

Intermittent

For a problem that is intermittent or that cannot be reproduced, include a copy of the error log at the time of the problem.**

Problem	Information to Include
Command language interpreters	When submitting an SPR on a command language interpreter, it is important to show all symbols (SHOW SYMBOL/ALL/GLOBAL) and logical names (SHOW LOGICAL/ALL) defined on the system.
Job controller	If the job controller aborts, it will print a message on the console and write a file named SYS$SYSTEM:SNAPSHOT.DAT. The user should include the console printout and a machine readable copy of the SNAPSHOT.DAT file.
Librarian	If the user encounters a problem with the LIBRARIAN, include the following information: 1. A machine readable copy of the library itself 2. Machine readable copies of all input files to the library 3. Information (DIRECTORY/FULL) on the library file 4. Information (LIBRARY/LIST/FULL) on the library contents If the problem can be duplicated at will, include the scenario and any command files used.
Linker	If the user encounters a problem with the LINKER, include machine readable copies of the object files and libraries used in the link along with a full map (LINK/MAP/FULL).
DECnet	For a DECnet problem, supply configurations of the systems involved in the problem. This information should include the version numbers of the operating systems and DECnet, the hardware on both systems, and the patch level of the DECnet software on the non-VMS system, if applicable.

Problem	Information to Include
Terminals	If the user suspects a problem with the terminal driver, provide the following information: 1. A list of terminal characteristics (SHOW TERMINAL) 2. The type of terminal 3. The type of modem (if any) 4. Any special front end equipment 5. Any unusual terminal configuration If the problem involves remote file access, it is often useful for the maintainer to know if the same or a similar operation can be performed from a different account, or with the source and destination nodes reserved.
Compiler/assembler	If the user encounters a problem with the assembler or a compiler, include the source program that caused the problem. (It is very important to include all require files and libraries that are referenced by the source program, also.) It is especially important to limit the scope of the problem when submitting SPRs on compilers. Include the version number of the compiler and the version number of the operating system.

*The raw data file (SYS$SYSTEM:SYSDUMP.DMP), not the formatted output from the SDA utility, should be included. Formatted output usually does not include all the information needed to solve the problem.

**The raw data file (SYS$SYSDISK:[SYSERR]ERRLOG.SYS), not the formatted output from the SYE utility, should be included. Formatted output often does not include all the information needed to solve the problem.

The following SPR priority explanations should be used as a guideline for determining the priority of an SPR:

1. Most production work cannot be run. Functions/jobs that are not usable are a major use of system, system will not boot, necessary peripherals cannot be used as intended.

2. Some production work cannot be run. Certain jobs/functions are not usable, performance degraded, installation has insufficient excess capacity.

3. All production work can be run with some impact on user. Significant manual intervention required, extra procedures, performance degraded but installation has excess capacity.

4. All production work can be run with no significant impact on user. Problem can be easily patched, simple bypass procedure exists.

5. No system modifications needed to return to normal production. Suggestion, consultation, documentation error.

GENERAL NOTES

VAX-11/750 TROUBLESHOOTING TIPS

1. When troubleshooting, place the console power-on action switch in the HALT position. If you do not, the CPU will try to reboot if a CPU halt occurs for any reason.

2. If the console prompt does not appear when power is turned on, insert the RDM module and type D. The RDM prompt (RDM>) should appear.

   Stop the clock by typing 'STO' followed by a carriage return (<RET>). The address of the last control store location accessed will be printed. If the address is Ø, dc low is asserted. If the address is X8XX, ac low is probably asserted and the microcode is in a loop waiting for it to deassert.

   First ac low is asserted by the power supply. The memory controller then asserts it until all memory locations have been cleared. Typing 'TRACE<RET>' after the RDM prompt prints a trace of the last 64 control store addresses referenced. This is very useful if the CPU is hung in a loop.

3. Turn off cache (>>>D/I 25 1<RET>), then try to examine and deposit locations in memory and on the UNIBUS from the console prompt (>>>). Also examine and deposit the RH75Ø if one is present.

   If memory cannot be examined from the console prompt, type D and see if it is accessible from the RDM prompt. This should determine whether the CMI and memory are working or not. Remember, UNIBUS locations cannot be accessed from the RDM prompt.

   The UNIBUS exerciser terminator (UET) is always present on the first UNIBUS, so it should be possible to examine or deposit the address register at location FFF46Ø. (The location is FBF46Ø for the internal programmable exerciser control (IPEC) on the second UNIBUS.) The console default should be set to physical addressing and word mode when accessing to UNIBUS address space, which is "1111 11XX XXXX XXXX XXXX XXXX" where X makes up the 18-bit binary address. (See Chapter 2, UBI and SUB Registers.)

4. Run a parity test on the control store by typing 'PAR Ø' from the RDM prompt. This checks control store parity in sequential locations starting with location Ø. A halt with a parity error in any address other than 17FD (hex) indicates a problem in the control store.

5. Always run the DPM microdiagnostics before the microdiagnostics. (See Chapter 6, RDM Installation Tests.)

6. Make sure that the system has the latest revision microcode and the latest FCOs are installed. This can be checked by an examine of the SID register (>>>E/I 3E <RET>). The bit fields of the SID register are as follows. (The revision levels shown are current as of the date of this document.)

   Bits <31:24> = Type code (the VAX-11/750 = 2)
   Bits <15:08> = Microcode revision (currently = 5E)
   Bits <07:00> = Hardware revision (currently = 3)

   Refer to Speed Bulletins 174, 175, and 207 for latest FCOs to date.

7. The current revisions for microdiagnostics are:

ECKAA	-	Rev. 6.0	Microdiagnostic Monitor (MICMON)
ECKAB	-	Rev. 6.1	Micro DPM
ECKAC	-	Rev. 5.0	Micro MIC

8. ECKAM does not test the first memory array module. The first 256-kilobyte array is tested by the MIC microdiagnostics.

9. Earlier versions of ECKAM may not report single bit errors or find some dual addressing errors. Check periodically to find out when the updated version will be available.

10. Cache parity errors in the VAX-11/750 cause a machine check to occur at location SCBB +4. This can be treated as a fatal error. Unlike the VAX-11/780, the VAX-11/750 does not attempt to obtain good data from memory on a cache parity error.

DW750 INSTALLATION

The DW750 is typically installed in slot 7 of the extended hex backplane to make cable routing easier and to place the SUB at first priority in the bus grant chain.

CAUTION

> Refer to the DW750 installation procedure that specifies use of the VELOSTAT mat kit to prevent damage from static discharge. Also see figures and tables in Chapter 8 under the headings Bus Grant Chain and Continuity Jumpers and MBA Installation Jumpers.

If one or more RH750 MASSBUS adapters are already in place, they must be moved down one slot and one CMI ARB level as shown in the following examples.

Example 1: MBAs Before DW750 Installation

Slot	Device	Base Address	CMI ARB Level
7	MBA 0	F28000	ARB 3
8	MBA 1	F2A000	ARB 2
9	--	(Spare)	ARB 1

Example 2: MBAs After DW750 Installation

Slot	Device	Base Address	CMI ARB Level
7	SUB	F32000	ARB 3
8	MBA 0	F28000	ARB 2
9	MBA 1	F2A000	ARB 1

UNIBUS EXERCISER (UBE) ON THE SECOND UNIBUS

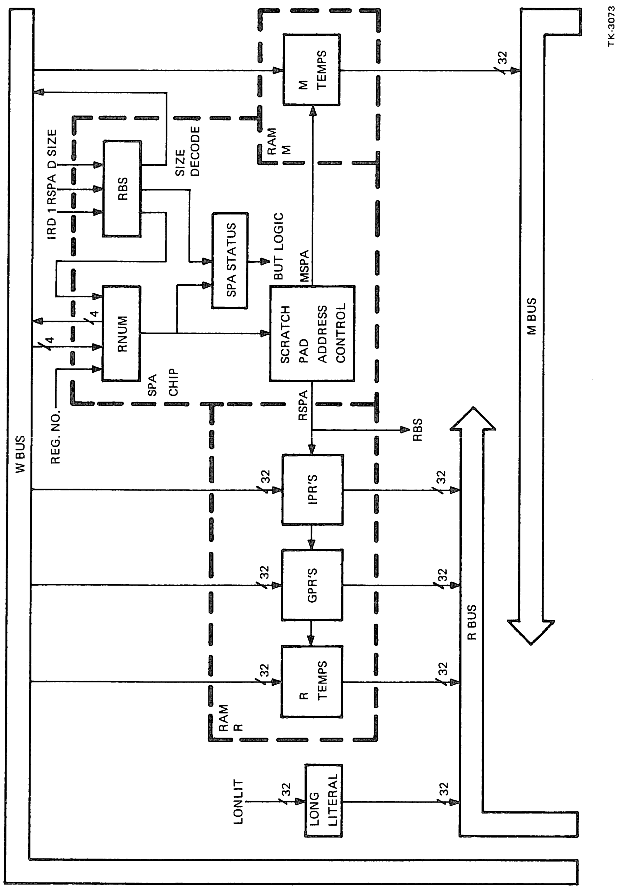
The M7855 UBE module may be plugged into an SPC slot in the second UNIBUS expander box to test cabling and connectors.

UBE Base CMI Address: FBF000 (770000 octal)

UBE Vector: 510

CHAPTER 10
CHARTS AND MACROS

SYSTEM SCRATCHPAD LOGIC



RTEMP AND GPR FUNCTIONS

REG NO	RSRC	ASSIGNMENT/PURPOSE
0	00	DUAL PORT TEMP 0
1	01	DUAL PORT TEMP 1
2	02	DUAL PORT TEMP 2
3	03	DUAL PORT TEMP 3
4	04	DUAL PORT TEMP 4
5	05	DUAL PORT TEMP 5
6	06	DUAL PORT TEMP 6
7	07	DUAL PORT TEMP 7
8	08	R S-PAD TEMP 8
9	09	R S-PAD TEMP 9
10	0A	R S-PAD TEMP 10
11	0B	R S-PAD TEMP 11
12	0C	R S-PAD TEMP 12
13	0D	R S-PAD TEMP 13
14	0E	MEMORY MANAGEMENT TEMP 5
15	0F	MEMORY MANAGEMENT TEMP 1

REG NO	RSRC	ASSIGNMENT/PURPOSE
0	10	GPR 0
1	11	GPR 1
2	12	GPR 2
3	13	GPR 3
4	14	GPR 4
5	15	GPR 5
6	16	GPR 6
7	17	GPR 7
8	18	GPR 8
9	19	GPR 9
10	1A	GPR 10
11	1B	GPR 11
12	1C	GPR 12
13	1D	GPR 13
14	1E	STACK POINTER
15	1F	MICRO CODE TEMPORARY

TK-3068

PRIVILEGED IPR AND MTEMP FUNCTIONS

REG NO	RSRC	ASSIGNMENT/PURPOSE
0	20	KERNEL STACK POINTER
1	21	EXECUTIVE STACK POINTER
2	22	SUPERVISOR STACK POINTER
3	23	USER STACK POINTER
4	24	INTERRUPT STACK POINTER
5	25	PROCESS CONTROL BLOCK BASE
6	26	MEMORY MANAGEMENT TEMP 2
7	27	MEMORY MANAGEMENT TEMP 3
8	28	P0 BASE REGISTER
9	29	P0 LENGTH REGISTER
10	2A	P1 BASE REGISTER
11	2B	P1 LENGTH REGISTER
12	2C	SYSTEM BASE REGISTER
13	2D	SYSTEM LENGTH REGISTER
14	2E	NEXT INTERVAL REGISTER
15	2F	MEMORY MANAGEMENT TEMP 4

REG NO	MSRC	ASSIGNMENT/PURPOSE
0	00	DUAL PORT TEMP 0
1	01	DUAL PORT TEMP 1
2	02	DUAL PORT TEMP 2
3	03	DUAL PORT TEMP 3
4	04	DUAL PORT TEMP 4
5	05	DUAL PORT TEMP 5
6	06	DUAL PORT TEMP 6
7	07	DUAL PORT TEMP 7
8	08	M S-PAD TEMP 8
9	09	M S-PAD TEMP 9
10	0A	M S-PAD TEMP 10
11	0B	ERROR CODE, MEM FAULTS & ARITH TRAP
12	0C	FPD PACK ROUTINE OFFSET
13	0D	MEMORY MANAGEMENT TEMP 0
14	0E	SYSTEM CONTROL BLOCK BASE
15	0F	SOFTWARE INT SUMMARY REGISTER

TK-3069

RSRC ASSIGNMENTS

RSRC <5:0 > HEX	RAM-R REGISTER	OPERATION
00-0D	TEMP0-TEMP13	-
0E	MM.TEMP5	-
0F	MM.TEMP1	-
10-1D	R0-R13	-
1E	SP	-
1F	RTMPGPR	-
20	KSP	-
21	ESP	-
22	SSP	-
23	USP	-
24	ISP	-
25	PCBB	-
26	MM.TMP2	-
27	MM.TEMP3	-
28	P0BR	-
29	P0LR	-
2A	P1BR	-
2B	P1LR	-
2C	SBR	-
2D	SLR	-
2E	SPNICR.SPICR	-
2F	MM.TEMP4	-

TK-3060

RSRC ASSIGNMENTS (CONT)

RSRC <5:0> HEX	RAM-R REGISTER	OPERATION
30	TEMP.R	-
31	DST.R	-
32	IPR.R	-
33	CRP.R	-
34	(TEMP0)	-
35	(TEMP7)	LONLIT
36	(TEMP0)	ZERO
37	(TEMP0)	ZERO.CLRRBSP
38	TEMP.ROR1	-
39	DST.POR1	-
3A	IPR.ROR1	-
3B	GPR.ROR1	-
3C	TEMP.R+1	-
3D	DST.R+1	-
3E	IPR.R+1	-
3F	GPR.R+1	-

TK-3061

MSRC ASSIGNMENTS

MSRC<4:0> HEX	RAM-M REGISTER	OPERATION	DESCRIPTION
00-0A	TEM P0-TEMP10	-	MICROCODE TEMPORARIES
0B	ERRCOD	-	ERROR CODE
0C	F PDOFFSET	-	FPD PACK ROUTINE OFFSET
0D	MM.TEMP0	-	MEMORY MANAGEMENT TEMP
0E	SCBB	-	SYSTEM CONTROL BLOCK BASE
0F	SISR	-	SOFTWARE INT SUMMARY
10	TEMP.R	-	MTEMP INDEXED BY RNUM
11	TEMP.R +1	-	MTEMP INDEXED BY RNUM+1
12	(TEMP0)*	MDR	MBUS < - - MDR
13	(TEMP0)*	WDR	MBUS < - - WDR
14	(TEMP0)	PSHSUB	WRITE - - TO RBS
15	(TEMP0)	PSHADD	WRITE + TO RBS
16	(TEMP0)	WBUS RNUM	WBUS < - - RNUM
17	(TEMP0)*	XB.PC PC+1	MBUS < - - XB, PC < - - PC+I SIZE
18	(TEMP0)*	MA	MBUS < - - MA
19		PC BACK	MBUS < - - PC BACK
1A	(TEMP0)*	PC	MBUS < - - PC
1B	(TEMP0)*	VA	MBUS < - - VA
1C	(TEMP0)	READRBS	READ RBS
1D	(TEMP0)	RNUM WBUS	RNUM < - - WBUS
1E	(TEMP0)	WB RBSP	WBUS < - - RBSP
1F	(TEMP0)*	TB	MBUS < - - TB DATA

TK-3079

```
;	CMT094.MCR		MICRO2  1M(01)	1-SEP-82  13:33:09	CLOKX Rev  9.01, Clock rate = 160ns
;	CHARTS.MIC		CHARTS.MIC

					;494	.TOC "CHARTS.MIC"
					;495	.TOC "REVISION 19.0"
					;496	;	J.Heom, G.Koeckhoven, C. E. MCDOWELL,  P. R. GUILBAULT
					;497
;498	.NOBIN
;499	.TOC	"	Revision History"
;500	; 19	Fill in data for IPR 3F
;501	; 18	Change some documentation and add a few charts
;502	; 17	Correct some documentation
;503	; 16	Add machine check logout and memory control and status registers.
;504	; 15	Initial release.
```

```
;505    .TOC "  Macro Level Charts             : PSL Chart"
;506
;507
;508
;509
;510    ;       +----------------------------------------------------------------+
;511    ;       |                               PSL                              |
;512    ;       +-+-+---+-+-+---+---+-+-+----------+---------------+-+-+-+-+-------+
;513    ;       |3|3|2 2|2|2|2 2|2 2|2|2 1 1 1 1|1 1 1 1 1 0 0|0|0|0|0|0 0 0 0|
;514    ;       |1|0|9 8|7|6|5 4|3 2|1|0 9 8 7 6|5 4 3 2 1 0 9 8|7|6|5|4|3 2 1 0|
;515    ;       +-+-+---+-+-+---+---+-+-+----------+---------------+-+-+-+-+-------+
;516    ;       |C|T| M |F|I| C | P |M|          |               |D|F|I|T|N|Z|V|C|
;517    ;       |M|P| B |P|S| U | R |B|   IPL    |      MBZ      |V|U|V| | | | | |
;518    ;       | | | Z |D| | R | V |Z|          |               | | | | | | | | |
;519    ;       +-+-+---+-+-+---+---+-+-+----------+---------------+-+-+-+-+-------+
;520    ;                                          |             PSW              |
;521    ;                                          +------------------------------+
;522    ;
;523    ;
;524    ;                       BITS            DESCRIPTION
;525    ;
;526    ;                       3:0             Condition Codes
;527    ;
;528    ;                       4               Trace enable
;529    ;
;530    ;                       5               Integer Overflow trap enable
;531    ;
;532    ;                       6               Floating Underflow trap enable
;533    ;
;534    ;                       7               Decimal Overflow trap enable
;535    ;
;536    ;                       15:8            Reserved to Digital, must be zero
;537    ;
;538    ;                       20:16           Interrupt Priority Level
;539    ;
;540    ;                       21              Reserved to Digital, must be zero
;541    ;
;542    ;                       22:23           Previous Mode
;543    ;
;544    ;                       25:24           Current Mode
;545    ;
;546    ;                       26              Interrupt Stack
;547    ;
;548    ;                       27              First Part Done
;549    ;
;550    ;                       29:28           Reserved to Digital, must be zero
;551    ;
;552    ;                       30              Trace Pending
;553    ;
;554    ;                       31              Compatibility Mode
```

```
.TOC "  Macro Level Charts                : Internal Processor Registers"

;  +-----------------------------------------------------------------------------------------------------+
;  |                                          COMET IPR'S                                                |
;  +--------------+--+------------------------------------+--------------+--+----------------------------+
;  |              |  |                                    |              |  |                            |
;  |              |RW|                                    |              |RW|                            |
;  |              |  |                                    |              |  |                            |
;  +--------------+--+------------------------------------+--------------+--+----------------------------+
;  | 00  KSP      |  | KERNEL STACK POINTER               | 20  RXCS     |  | CONSOLE RECEIVE CONTROL/STATUS |
;  | 01  ESP      |  | EXECUTIVE STACK POINTER            | 21  RXDB     |RO| CONSOLE RECEIVE DATA BUFFER    |
;  | 02  SSP      |  | SUPERVISOR STACK POINTER           | 22  TXCS     |  | CONSOLE TRANSMIT CONTROL/STATUS |
;  | 03  USP      |  | USER STACK POINTER                 | 23  TXDB     |WO| CONSOLE TRANSMIT DATA BUFFER   |
;  | 04  ISP      |  | INTERRUPT STACK POINTER            | 24  TBDR     |  | TRANSLATION BUFFER DISABLE REGISTER |
;  | 05  Reserved |  |                                    | 25  CADR     |  | CACHE DISABLE REGISTER         |
;  | 06  Reserved |  |                                    | 26  MCESR    |  | MACHINE CHECK ERROR SUMMARY REGISTER|
;  | 07  Reserved |  |                                    | 27  CAER     |  | CACHE ERROR REGISTER           |
;  | 08  P0BR     |  | P0 BASE REGISTER                   | 28  ACCS     |RO| ACCELERATOR CONTROL/STATUS REG |
;  | 09  P0LR     |  | P0 LENGTH REGISTER                 | 29  Reserved |  |                                |
;  | 0A  P1BR     |  | P1 BASE REGISTER                   | 2A  Reserved |  |                                |
;  | 0B  P1LR     |  | P1 LENGTH REGISTER                 | 2B  Reserved |  |                                |
;  | 0C  SBR      |  | SYSTEM BASE REGISTER               | 2C  Reserved |  |                                |
;  | 0D  SLR      |  | SYSTEM LENGTH REGISTER             | 2D  Reserved |  |                                |
;  | 0E  Reserved |  |                                    | 2E  Reserved |  |                                |
;  | 0F  Reserved |  |                                    | 2F  Reserved |  |                                |
;  | 10  PCBB     |  | PROCESS CONTROL BLOCK BASE         | 30  Reserved |  |                                |
;  | 11  SCBB     |  | SYSTEM CONTROL BLOCK BASE          | 31  Reserved |  |                                |
;  | 12  IPL      |  | INTERRUPT PRIORITY LEVEL           | 32  Reserved |  |                                |
;  | 13  ASTR     |  | AST LEVEL REGISTER                 | 33  Reserved |  |                                |
;  | 14  SIRR     |WO| SOFTWARE INTERRUPT REQUEST REGISTER | 34  Reserved |  |                                |
;  | 15  SISR     |  | SOFTWARE INTERRUPT SUMMARY REGISTER | 35  Reserved |  |                                |
;  | 16  Reserved |  |                                    | 36  Reserved |  |                                |
;  | 17  CMIERR   |RO| CMI ERROR REGISTER                 | 37  IO RESET |WO| INITIALIZE UNIBUS              |
;  | 18  ICCS     |  | INTERVAL CLOCK CONTROL/STATUS      | 38  MME      |  | MEMORY MANAGEMENT ENABLE       |
;  | 19  NICR     |WO| NEXT INTERVAL COUNT REGISTER       | 39  TBIA     |WO| TRANSLATION BUFFER INVALIDATE ALL |
;  | 1A  ICR      |RO| INTERVAL COUNT REGISTER            | 3A  TBIS     |WO| TRANSLATION BUFFER INVALIDATE SINGLE|
;  | 1B  TODR     |  | TIME OF DAY REGISTER               | 3B  TB DATA  |  | TRANSLATION BUFFER DATA        |
;  | 1C  CSRS     |  | CONSOLE STORAGE RECEIVER STATUS    | 3C  Reserved |  |                                |
;  | 1D  CSRD     |RO| CONSOLE STORAGE RECEIVER DATA      | 3D  PMR      |  | PERFORMANCE MONITOR REGISTER   |
;  | 1E  CSTS     |  | CONSOLE STORAGE TRANSMIT STATUS    | 3E  SID      |RO| SYSTEM IDENTIFICATION          |
;  | 1F  CSTD     |WO| CONSOLE STORAGE TRANSMIT DATA      | 3F  TBHP     |WO| Probe tb for tb hit            |
;  +--------------+--+------------------------------------+--------------+--+----------------------------+
;
;                        +---------------------------------------------------------------+
;                        |                     System Identification                     |
;                        +---------------+---------------+---------------+---------------+
;                        |3 3 2 2 2 2 2 2|2 2 2 2 1 1 1 1|1 1 1 1 1 1 0 0|0 0 0 0 0 0 0 0|
;                        |1 0 9 8 7 6 5 4|3 2 1 0 9 8 7 6|5 4 3 2 1 0 9 8|7 6 5 4 3 2 1 0|
;                        +---------------+---------------+---------------+---------------+
;                        |0 0 0 0 0 0 1 0|0 0 0 0 0 0 0 0|   Micro Rev   | Hardware Rev  |
;                        +---------------+---------------+---------------+---------------+
;
```

```
.TOC "  Macro Level Charts              : Memory Status & Control Registers"

;
;                          +---------------------------------------------------------------+
;                          | CMIERR                 CMI ERROR REGISTER             IPR #^X17|
;                          +---------------------------------------------------------------+
;                          |3|3|2|2|2|2|2|2|2|2|2|2|1|1|1|1|1|1|1|1|1|1|0|0|0|0|0|0|0|0|0|0|
;                          |1|0|9|8|7|6|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|
;                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;                          |               | CMIDIS|  SMR  |  RLTO | TBGPR | TBHIT |  BER  |
;                          +---------------+-------+-------+-------+-------+-------+-------+
;                                                | | | | |       | | | | |       | | | | |
;                          Disable CMI = 1 --------+ | | | |       | | | | |       | | | | |
;                          Read = 1, Modify = 0 ----+ | | |       | | | | |       | | | | |
;                          Virtual = 0, Physical = 1 -+ | |       | | | | |       | | | | |
;                                                +--------+ |       | | | | |       | | | | |
;                          Saved CPU Mode <1:0> |          |       | | | | |       | | | | |
;                                                +----------+       | | | | |       | | | | |
;                                                                   | | | | |       | | | | |
;                          Read Lock Timeout = 1 -------------------+ | | | |       | | | | |
;                          TB G1 Tag Parity Error -------------------+ | | |       | | | | |
;                          TB G0 Tag Parity Error --------------------+ | |       | | | | |
;                          TB G1 Data Parity Error ---------------------+ |       | | | | |
;                          TB G0 Data Parity Error -----------------------+       | | | | |
;                                                                                 | | | | |
;                          TB Hit on last reference (G0 or G1) -------------------+ | | | |
;                          Memory Error (Timeout, error) --------------------------+ | | |
;                          Uncorrectable Error ---------------------------------------+ | |
;                          Lost Error ---------------------------------------------------+ |
;                          Corrected Read Data --------------------------------------------+
;
;
;                          +---------------------------------------------------------------+
;                          | MCESR      MACHINE CHECK ERROR SUMMARY REGISTER        IPR #^X26|
;                          +---------------------------------------------------------------+
;                          |3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
;                          |1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
;                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;                          |                           MBZ                         | | | | |
;                          +-------------------------------------------------------+-------+
;                                                                                    | | | |
;                          BUS ERROR, Refer to Bus Error Register or CMIERR  --------+ | | |
;                          TB PARITY ERROR, Refer to CMIERR  ---------------------------+ | |
;                          MBZ ---------------------------------------------------------+ |
;                          XB fetch for ISTRM = 1, Operand Fetch = 0      -----------------+
```

```
+-------------------------------------------------------------+
| TBDR     TRANSLATION BUFFER GROUP DISABLE REGISTER  IPR #^X24|
+-------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MBZ                          | | | | |
+-------------------------------------------------------+-------+
                                                          | | | |
0 = Random Replacement, 1 = Force Replacement   ----------+ | | |
0 = Replace Group 0, 1 = Replace Group 1 -------------------+ | |
1 = Force Miss Group 1 ---------------------------------------+ |
1 = Force Miss Group 0 -----------------------------------------+

+-------------------------------------------------------------+
| CADR                  CACHE DISABLE REGISTER        IPR #^X25|
+-------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MBZ                              | |
+-----------------------------------------------------------+-+
                                                              |
1 = CACHE OFF ------------------------------------------------+

+-------------------------------------------------------------+
| CAER                  CACHE ERROR REGISTER          IPR #^X27 |
+-------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MBZ                          | | | | |
+-------------------------------------------------------+-------+
                                                          | | | |
Cache Tag Parity Error -----------------------------------+ | | |
Cache Data Parity Error ------------------------------------+ | |
Lost Error ---------------------------------------------------+ |
Cache Hit ------------------------------------------------------+
```

```
.TOC "  Macro Level Charts            : Machine Check Logout Stack"
;                                     MACHINE CHECK EXCEPTION STACK LOGOUT TABLE
;
;                          (SP)      LENGTH PARAMETER    00000028            SUMMARY PARAMETER
;                          (SP)+4    SUMMARY PARAMETER   0000000X
;                          (SP)+8    VA                  XXXXXXXX            1 = CS Parity Error
;                          (SP)+C    PC at time of error XXXXXXXX            2 = Memory Error,  Cache Parity,
;                          (SP)+10   MDR                 XXXXXXXX                Time Out, or TBUF parity
;                          (SP)+14   SAVED MODE REG      0000000X            6 = Micro-sequencing error,
;                          (SP)+18   RLTO                0000000X                entered filler micro-code
;                          (SP)+1C   TBGPR               0000000X            7 = Bad IRD
;                          (SP)+20   CAER                0000000X
;                          (SP)+24   BER                 0000000X
;                          (SP)+28   MCESR               0000000X
;                          (SP)+2C   PC                  XXXXXXXX
;                          (SP)+30   PSL                 XXXXXXXX
```

```
+---------------------------------------------------------------+
| RLTO              READ LOCK TIMEOUT REGISTER         ^X18(SP)|
+---------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            MBZ                              | |
+-------------------------------------------------------------+-+
                                                              |
Read Lock Time Out on CMI ------------------------------------+


+---------------------------------------------------------------+
| SMR               SAVED CPU MODE REGISTER            ^X14(SP) |
+---------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            MBZ                        | | | | |
+-------------------------------------------------------+-------+
                                                        | | | |
Read Reference = 1, Modify Reference = 0  --------------+ | | |
Virtual Reference = 0, Physical Reference = 1 ------------+ | |
Saved CPU Mode <1:0>, 00 = Kernel --------------------------+ |
                      01 = Exec   ----------------------------+
                      10 = Supervisor
                      11 = User


+---------------------------------------------------------------+
| BER               BUS ERROR REGISTER                 ^X24(SP) |
+---------------------------------------------------------------+
|3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|
|1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            MBZ                        | | | | |
+-------------------------------------------------------+-------+
                                                        | | | |
Memory Error -------------------------------------------+ | | |
Uncorrectable Error --------------------------------------+ | |
Lost Error -------------------------------------------------+ |
Corrected Data Error -----------------------------------------+
```

```
.TOC "  Macro Level Charts                 : System Control Block"

;    Vector              Description              IPL         I/E

;    SCBB+0              Not Used                 -           -
;    SCBB+4              Machine Check            1F          E
;    SCBB+8              Kernel Stack Invalid     1F          E
;    SCBB+C              Power Fail               1E          I
;    SCBB+10             Reserved Opcode          -           E
;    SCBB+14             Customer Opcode XFC      -           E
;    SCBB+18             Reserved Operand         -           E
;    SCBB+1C             Reserved Address Mode    -           E
;    SCBB+20             Access Violation         -           E
;    SCBB+24             Translation Invalid      -           E
;    SCBB+28             Trace Trap               -           E
;    SCBB+2C             Breakpoint Opcode        -           E
;    SCBB+30             Compatability Mode       -           E
;    SCBB+34             Arithmetic Trap          -           E
;    SCBB+40             CHMK                     -           E
;    SCBB+44             CHME                     -           E
;    SCBB+48             CHMS                     -           E
;    SCBB+4C             CHMU                     -           E
;    SCBB+54             Corrected Read Data      1A          I
;    SCBB+60             Write Bus Error          1D          I
;    SCBB+84             Soft Interrupt           1           I
;    SCBB+88             Soft Interrupt           2           I
;    SCBB+8C             Soft Interrupt           3           I
;    SCBB+90             Soft Interrupt           4           I
;    SCBB+94             Soft Interrupt           5           I
;    SCBB+98             Soft Interrupt           6           I
;    SCBB+9C             Soft Interrupt           7           I
;    SCBB+A0             Soft Interrupt           8           I
;    SCBB+A4             Soft Interrupt           9           I
;    SCBB+A8             Soft Interrupt           A           I
;    SCBB+AC             Soft Interrupt           B           I
;    SCBB+AD             Soft Interrupt           C           I
;    SCBB+B4             Soft Interrupt           D           I
;    SCBB+B8             Soft Interrupt           E           I
;    SCBB+BC             Soft Interrupt           F           I
;    SCBB+C0             Interval Timer           18          I
;    SCBB+F0             TU-58   Receive          14          I
;    SCBB+F4             TU-58 Transmit           14          I
;    SCBB+F8             Console Receive          14          I
;    SCBB+FC             Console Transmit         14          I
;    SCBB+110-1D8        MASSBUS                  14-17       I
;    SCBB+200            Unibus                   14-17       I
;
;    MACRO VECTOR BITS<1:0> = 0 ; Process Interrupt on Kernel Stack unless PSL <IS> = 1
;    MACRO VECTOR BITS<1:0> = 1 ; Process Interrupt or Exception on Interrupt Stack
;    MACRO VECTOR BITS<1:0> = 2 ; Trap to WCS location 2001 for service if WCS is present .
;    MACRO VECTOR BITS<1:0> = 3 ; Halt Processor
```

```
.TOC "  Macro Level Charts            : Massbus and Unibus Vector Generation"

; MASSBUS VECTORS
;     MDR after CMI WRITE VECTOR from a Massbus Adaptor
;
;     +---------------------------------------------------------------+    XX = The encoded priority plug BR. XX = 00 for BR4
;     |3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0|                                          = 01 for BR5
;     |1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0|                                          = 10 for BR6
;     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                          = 11 for BR7
;     |                      MBZ                |0 0 1 X X 0 1 Y Y 0 0|    YY = The MBA address jumper selection.
;     +-----------------------------------------+-----+-------+-------+                                   YY = 00 for MBA 0
;                                                                                                            = 01 for MBA 1
;                                                                                                            = 10 for MBA 2
;
; UNIBUS VECTORS
;     MDR after CMI WRITE VECTOR from UBI 0
;     +-----------------------------------------+---------------------+
;     |                      MBZ                |0 1 Z Z Z Z Z Z Z Z Z|    ZZZZZZZZ = The Unibus vector
;     +-----------------------------------------+-----+-------+-------+
;
;     MDR after CMI WRITE VECTOR from UBI 1
;     +-----------------------------------------+---------------------+
;     |                      MBZ                |1 0 Z Z Z Z Z Z Z Z Z|
;     +-----------------------------------------+-----+-------+-------+
```

```
        .TOC    "       Tables and charts     : Instruction Name vs Op Code"






        ;  ACBB    9D     BGTR    14     CASEW   AF     CVTLP   F9     FFS     EA     MOVQ    7D     resvrd 59
        ;  ACBD    6F     BGTRU   1A     CHME    BD     CVTLW   F7     HALT    00     MOVTC   2E     resvrd 5A
        ;  ACBF    4F     BICB2   8A     CHMK    BC     CVTPL   36     INCB    96     MOVTUC  2F     resvrd 5B
        ;  ACBL    F1     BICB3   8B     CHMS    BE     CVTTP   26     INCL    D6     MOVW    B0     resvrd 77
        ;  ACBW    3D     BICL2   CA     CHMU    BF     CVTPT   24     INCW    B6     MOVZBL  9A     RET    04
        ;  ADAWI   58     BICL3   CB     CLRB    94     CVTPS   08     INDEX   0A     MOVZBW  9B     ROTL   9C
        ;  ADDB2   80     BICPSW  B9     CLRD    7C     CVTRDL  6B     INSQHI  5C     MOVZWL  3C     RSB    05
        ;  ADDB3   81     BICW2   AA     CLRF    D4     CVTRFL  4B     INSQTI  5D     MTPR    DA     SBWC   D9
        ;  ADDD2   60     BICW3   AB     CLRL    D4     CVTSP   09     INSQUE  0E     MULB2   84     SCAN   2A
        ;  ADDD3   61     BISB2   88     CLRQ    7C     CVTWB   33     INSV    F0     MULB3   85     SKPC   3B
        ;  ADDF2   40     BISB3   89     CLRW    B4     CVTW    6D     JMP     17     MULD2   64     SOBGEQ F4
        ;  ADDF3   41     BISL2   C8     CMPB    91     CVTWF   4D     JSB     16     MULD3   65     SOBGTR F5
        ;  ADDL2   C0     BISL3   C9     CMPC3   29     CVTWL   32     LDPCTX  06     MULF2   44     SPANC  2B
        ;  ADDL3   C1     BISPSW  B8     CMPC5   2D     DECB    97     LOCC    3A     MULF3   45     SUBB2  82
        ;  ADDP4   20     BISW2   A8     CMPD    71     DECL    D7     MATCHC  39     MULL2   C4     SUBB3  83
        ;  ADDP6   21     BISW3   A9     CMPF    51     DECW    B7     MCOMB   92     MULL3   C5     SUBD2  62
        ;  ADDW2   A0     BITB    93     CMPL    D1     DIVB2   86     MCOML   D2     MULP    25     SUBD3  63
        ;  ADDW3   A1     BITL    D3     CMPP3   35     DIVB3   87     MCOMW   B2     MULW2   A4     SUBF2  42
        ;  ADWC    D8     BITW    B3     CMPP4   37     DIVD2   66     MFPR    DB     MULW3   A5     SUBF3  43
        ;  AOBLEQ  F3     BLBC    E9     CMPV    EC     DIVD3   67     MNEGB   8E     NOP     01     SUBL2  C2
        ;  AOBLSS  F2     BLBS    E8     CMPW    B1     DIVF2   46     MNEGD   72     POLYD   75     SUBL3  C3
        ;  ASHL    78     BLEQ    15     CMPZV   ED     DIVF3   47     MNEGF   52     POLYF   55     SUBP4  22
        ;  ASHP    F8     BLEQU   1B     CRC     0B     DIVL2   C6     MNEGL   CE     POPR    BA     SUBP6  23
        ;  ASHQ    79     BLSS    19     CVTBD   6C     DIVL3   C7     MNEGW   AE     PROBER  0C     SUBW2  A2
        ;  BB      E1     BLSSU   1F     CVTBF   4C     DIVP    27     MOVAB   9E     PROBEW  0D     SUBW3  A3
        ;  BBCC    E5     BNEQ    12     CVTBL   98     DIVW2   A6     MOVAD   7E     PUSHAB  9F     SVPCTX 07
        ;  BBCCI   E7     BNEQU   12     CVTBW   99     DIVW3   A7     MOVAF   DE     PUSHAD  7F     TSTB   95
        ;  BBCS    E3     BPT     03     CVTDB   68     EDITPC  38     MOVAL   DE     PUSHAF  DF     TSTD   73
        ;  BBS     E0     BRB     11     CVTDF   76     EDIV    7B     MOVAQ   7E     PUSHAL  DF     TSTF   53
        ;  BBSC    E4     BRW     31     CVTDL   6A     EMODD   74     MOVAW   3E     PUSHAQ  7F     TSTL   D5
        ;  BBSS    E2     BSBB    10     CVTDW   69     EMODF   54     MOVB    90     PUSHAW  3F     TSTW   B5
        ;  BBSSI   E6     BSBW    30     CVTFB   48     EMUL    7A     MOVC3   28     PUSHL   DD     XFC    FC
        ;  BCC     1E     BVC     1C     CVTFD   56     ESCD    FD     MOVC5   2C     PUSHR   BB     XORB2  8C
        ;  BCS     1F     BVS     1D     CVTFL   4A     ESCE    FE     MOVD    70     REI     02     XORB3  8D
        ;  BEQL    13     CALLG   FA     CVTFW   49     ESCF    FF     MOVF    50     REMQHI  5E     XORL2  CC
        ;  BEQLU   13     CALLS   FB     CVTLB   F6     EXTV    EE     MOVL    D0     REMQTI  5F     XORL3  CD
        ;  BGEQ    18     CASE    8F     CVTLD   6E     EXTZV   EF     MOVP    34     REMQUE  0F     XORW2  AC
        ;  BGEQU   1E     CASEL   CF     CVTLF   4E     FFC     EB     MOVPSL  DC     resvrd  57     XORW3  AD
```

```
;869    .TOC    "       Tables and charts      : ESCD Instruction Name vs Op Code(Two byte op codes FDxx)"
;870
;871
;872
;873
;874
;875    ;  ACBG    4F    EMODH   74    resvrd 15    resvrd 3C    resvrd 93    resvrd BA    resvrd DF
;876    ;  ACBH    6F    MNEGG   52    resvrd 16    resvrd 3D    resvrd 94    resvrd BB    resvrd E0
;877    ;  ADDG2   40    MNEGH   72    resvrd 17    resvrd 3E    resvrd 95    resvrd BC    resvrd E1
;878    ;  ADDG3   41    MOVAH   7E    resvrd 18    resvrd 3F    resvrd 96    resvrd BD    resvrd E2
;879    ;  ADDH2   60    MOVAO   7E    resvrd 19    resvrd 57    resvrd 97    resvrd BE    resvrd E3
;880    ;  ADDH3   61    MOVG    50    resvrd 1A    resvrd 58    resvrd 9A    resvrd BF    resvrd E4
;881    ;  CLRH    7C    MOVH    70    resvrd 1B    resvrd 59    resvrd 9B    resvrd C0    resvrd E5
;882    ;  CLRO    7C    MOVO    7D    resvrd 1C    resvrd 5A    resvrd 9C    resvrd C1    resvrd E6
;883    ;  CMPG    51    MULG2   44    resvrd 1D    resvrd 5B    resvrd 9D    resvrd C2    resvrd E7
;884    ;  CMPH    71    MULG3   45    resvrd 1E    resvrd 5C    resvrd 9E    resvrd C3    resvrd E8
;885    ;  CVTBG   4C    MULH2   64    resvrd 1F    resvrd 5D    resvrd 9F    resvrd C4    resvrd E9
;886    ;  CVTBH   6C    MULH3   65    resvrd 20    resvrd 5E    resvrd A0    resvrd C5    resvrd EA
;887    ;  CVTDH   32    POLYG   55    resvrd 21    resvrd 5F    resvrd A1    resvrd C6    resvrd EB
;888    ;  CVTFG   99    POLYH   75    resvrd 22    resvrd 77    resvrd A2    resvrd C7    resvrd EC
;889    ;  CVTFH   98    PUSHAH  7F    resvrd 23    resvrd 78    resvrd A3    resvrd C8    resvrd ED
;890    ;  CVTGB   48    PUSHAO  7F    resvrd 24    resvrd 79    resvrd A4    resvrd C9    resvrd EE
;891    ;  CVTGF   33    resvrd  00    resvrd 25    resvrd 7A    resvrd A5    resvrd CA    resvrd EF
;892    ;  CVTGH   56    resvrd  01    resvrd 26    resvrd 7B    resvrd A6    resvrd CB    resvrd F0
;893    ;  CVTGL   4A    resvrd  02    resvrd 27    resvrd 80    resvrd A7    resvrd CC    resvrd F1
;894    ;  CVTGW   49    resvrd  03    resvrd 28    resvrd 81    resvrd A8    resvrd CD    resvrd F2
;895    ;  CVTHB   68    resvrd  04    resvrd 29    resvrd 82    resvrd A9    resvrd CE    resvrd F3
;896    ;  CVTHD   F7    resvrd  05    resvrd 2A    resvrd 83    resvrd AA    resvrd CF    resvrd F4
;897    ;  CVTHF   F6    resvrd  06    resvrd 2B    resvrd 84    resvrd AB    resvrd D0    resvrd F5
;898    ;  CVTHG   76    resvrd  07    resvrd 2C    resvrd 85    resvrd AC    resvrd D1    resvrd F8
;899    ;  CVTHL   6A    resvrd  08    resvrd 2D    resvrd 86    resvrd AD    resvrd D2    resvrd F9
;900    ;  CVTHW   69    resvrd  09    resvrd 2E    resvrd 87    resvrd AE    resvrd D3    resvrd FA
;901    ;  CVTLG   4E    resvrd  0A    resvrd 2F    resvrd 88    resvrd AF    resvrd D4    resvrd FB
;902    ;  CVTLH   6E    resvrd  0B    resvrd 30    resvrd 89    resvrd B0    resvrd D5    resvrd FC
;903    ;  CVTRGL  4B    resvrd  0C    resvrd 31    resvrd 8A    resvrd B1    resvrd D6    resvrd FD
;904    ;  CVTRHL  6B    resvrd  0D    resvrd 34    resvrd 8B    resvrd B2    resvrd D7    resvrd FE
;905    ;  CVTWG   4D    resvrd  0E    resvrd 35    resvrd 8C    resvrd B3    resvrd D8    resvrd FF
;906    ;  CVTWH   6D    resvrd  0F    resvrd 36    resvrd 8D    resvrd B4    resvrd D9    SUBG3   43
;907    ;  DIVG2   46    resvrd  10    resvrd 37    resvrd 8E    resvrd B5    resvrd DA    SUBG2   42
;908    ;  DIVG3   47    resvrd  11    resvrd 38    resvrd 8F    resvrd B6    resvrd DB    SUBH2   62
;909    ;  DIVH2   66    resvrd  12    resvrd 39    resvrd 90    resvrd B7    resvrd DC    SUBH3   63
;910    ;  DIVH3   67    resvrd  13    resvrd 3A    resvrd 91    resvrd B8    resvrd DD    TSTG    53
;911    ;  EMODG   54    resvrd  14    resvrd 3B    resvrd 92    resvrd B9    resvrd DE    TSTH    73
```

```
;912    .TOC    "       Tables and charts      : Op Code vs Instruction Name"
;913
;914
;915
;916
;917
;918    ;  00 HALT      22 SUBP4     48 CVTFB     6E CVTLD     91 CMPB      B7 DECW      DC MOVPSL
;919    ;  01 NOP       23 SUBP6     49 CVTFW     6F ACBD      92 MCOMB     B8 BISPSW    DD PUSHL
;920    ;  02 REI       24 CVTPT     4A CVTFL     70 MOVD      93 BITB      B9 BICPSW    DE MOVAF
;921    ;  03 BPT       25 MULP      4B CVTRFL    71 CMPD      94 CLRB      BA POPR      DE MOVAL
;922    ;  04 RET       26 CVTTP     4C CVTBF     72 MNEGD     95 TSTB      BB PUSHR     DF PUSHAF
;923    ;  05 RSB       27 DIVP      4D CVTWF     73 TSTD      96 INCB      BC CHMK      DF PUSHAL
;924    ;  06 LDPCTX    28 MOVC3     4E CVTLF     74 EMODD     97 DECB      BD CHME      E0 BBS
;925    ;  07 SVPCTX    29 CMPC3     4F ACBF      75 POLYD     98 CVTBL     BE CHMS      E1 BBC
;926    ;  08 CVTPS     2A SCANC     50 MOVF      76 CVTDF     99 CVTBW     BF CHMU      E2 BBSS
;927    ;  09 CVTSP     2B SPANC     51 CMPF      77 resvrd    9A MOVZBL    C0 ADDL2     E3 BBCS
;928    ;  0A INDEX     2C MOVC5     52 MNEGF     78 ASHL      9B MOVZBW    C1 ADDL3     E4 BBSC
;929    ;  0B CRC       2D CMPC5     53 TSTF      79 ASHQ      9C ROTL      C2 SUBL2     E5 BBCC
;930    ;  0C PROBER    2E MOVTC     54 EMODF     7A EMUL      9D ACBB      C3 SUBL3     E6 BBSSI
;931    ;  0D PROBEW    2F MOVTUC    55 POLYF     7B EDIV      9E MOVAB     C4 MULL2     E7 BBCCI
;932    ;  0E INSQUE    30 BSBW      56 CVTFD     7C CLRD      9F PUSHAB    C5 MULL3     E8 BLBS
;933    ;  0F REMQUE    31 BRW       57 resvrd    7C CLRQ      A0 ADDW2     C6 DIVL2     E9 BLBC
;934    ;  10 BSBB      32 CVTWL     58 ADAWI     7D MOVQ      A1 ADDW3     C7 DIVL3     EA FFS
;935    ;  11 BRB       33 CVTWB     59 resvrd    7E MOVAD     A2 SUBW2     C8 BISL2     EB FFC
;936    ;  12 BNEQ      34 MOVP      5A resvrd    7E MOVAQ     A3 SUBW3     C9 BISL3     EC CMPV
;937    ;  12 BNEQU     35 CMPP3     5B resvrd    7F PUSHAD    A4 MULW2     CA BICL2     ED CMPZV
;938    ;  13 BEQL      36 CVTPL     5C INSQHI    7F PUSHAQ    A5 MULW3     CB BICL3     EE EXTV
;939    ;  13 BEQLU     37 CMPP4     5D INSQTI    80 ADDB2     A6 DIVW2     CC XORL2     EF EXTZV
;940    ;  14 BGTR      38 EDITPC    5E REMQHI    81 ADDB3     A7 DIVW3     CD XORL3     F0 INSV
;941    ;  15 BLEQ      39 MATCHC    5F REMQTI    82 SUBB2     A8 BISW2     CE MNEGL     F1 ACBL
;942    ;  16 JSB       3A LOCC      60 ADDD2     83 SUBB3     A9 BISW3     CF CASEL     F2 AOBLSS
;943    ;  17 JMP       3B SKPC      61 ADDD3     84 MULB2     AA BICW2     D0 MOVL      F3 AOBLEQ
;944    ;  18 BGEQ      3C MOVZWL    62 SUBD2     85 MULB3     AB BICW3     D1 CMPL      F4 SOBGEQ
;945    ;  19 BLSS      3D ACBW      63 SUBD3     86 DIVB2     AC XORW2     D2 MCOML     F5 SOBGTR
;946    ;  1A BGTRU     3E MOVAW     64 MULD2     87 DIVB3     AD XORW3     D3 BITL      F6 CVTLB
;947    ;  1B BLEQU     3F PUSHAW    65 MULD3     88 BISB2     AE MNEGW     D4 CLRF      F7 CVTLW
;948    ;  1C BVC       40 ADDF2     66 DIVD2     89 BISB3     AF CASEW     D4 CLRL      F8 ASHP
;949    ;  1D BVS       41 ADDF3     67 DIVD3     8A BICB2     B0 MOVW      D5 TSTL      F9 CVTLP
;950    ;  1E BCC       42 SUBF2     68 CVTDB     8B BICB3     B1 CMPW      D6 INCL      FA CALLG
;951    ;  1E BGEQU     43 SUBF3     69 CVTDW     8C XORB2     B2 MCOMW     D7 DECL      FB CALLS
;952    ;  1F BCS       44 MULF2     6A CVTDL     8D XORB3     B3 BITW      D8 ADWC      FC XFC
;953    ;  1F BLSSU     45 MULF3     6B CVTRDL    8E MNEGB     B4 CLRW      D9 SBWC      FD ESCD
;954    ;  20 ADDP4     46 DIVF2     6C CVTBD     8F CASEB     B5 TSTW      DA MTPR      FE ESCE
;955    ;  21 ADDP6     47 DIVF3     6D CVTWD     90 MOVB      B6 INCW      DB MFPR      FF ESCF
```

CHARTS (CONT)

```
.TOC    "       Tables and charts     : ESCD Op Code vs Instruction Name(Two byte op codes FDxx)"

;  00 resvrd   25 resvrd   4A CVTGL    6F ACBH     91 resvrd   B6 resvrd   DB resvrd
;  01 resvrd   26 resvrd   4B CVTRGL   70 MOVH     92 resvrd   B7 resvrd   DC resvrd
;  02 resvrd   27 resvrd   4C CVTBG    71 CMPH     93 resvrd   B8 resvrd   DD resvrd
;  03 resvrd   28 resvrd   4D CVTWG    72 MNEGH    94 resvrd   B9 resvrd   DE resvrd
;  04 resvrd   29 resvrd   4E CVTLG    73 TSTH     95 resvrd   BA resvrd   DF resvrd
;  05 resvrd   2A resvrd   4F ACBG     74 EMODH    96 resvrd   BB resvrd   E0 resvrd
;  06 resvrd   2B resvrd   50 MOVG     75 POLYH    97 resvrd   BC resvrd   E1 resvrd
;  07 resvrd   2C resvrd   51 CMPG     76 CVTHG    9C CVTFH    BD resvrd   E2 resvrd
;  08 resvrd   2D resvrd   52 MNEGG    77 resvrd   99 CVTFG    BE resvrd   E3 resvrd
;  09 resvrd   2E resvrd   53 TSTG     78 resvrd   9A resvrd   BF resvrd   E4 resvrd
;  0A resvrd   2F resvrd   54 EMODG    79 resvrd   9B resvrd   C0 resvrd   E5 resvrd
;  0B resvrd   30 resvrd   55 POLYG    7A resvrd   9C resvrd   C1 resvrd   E6 resvrd
;  0C resvrd   31 resvrd   56 CVTGH    7B resvrd   9D resvrd   C2 resvrd   E7 resvrd
;  0D resvrd   32 CVTDH    57 resvrd   7C CLRH     9E resvrd   C3 resvrd   E8 resvrd
;  0E resvrd   33 CVTGF    58 resvrd   7C CLRO     9F resvrd   C4 resvrd   E9 resvrd
;  0F resvrd   34 resvrd   59 resvrd   7D MOVO     A0 resvrd   C5 resvrd   EA resvrd
;  10 resvrd   35 resvrd   5A resvrd   7E MOVAH    A1 resvrd   C6 resvrd   EB resvrd
;  11 resvrd   36 resvrd   5B resvrd   7E MOVAO    A2 resvrd   C7 resvrd   EC resvrd
;  12 resvrd   37 resvrd   5C resvrd   7F PUSHAH   A3 resvrd   C8 resvrd   ED resvrd
;  13 resvrd   38 resvrd   5D resvrd   7F PUSHAO   A4 resvrd   C9 resvrd   EE resvrd
;  14 resvrd   39 resvrd   5E resvrd   80 resvrd   A5 resvrd   CA resvrd   EF resvrd
;  15 resvrd   3A resvrd   5F resvrd   81 resvrd   A6 resvrd   CB resvrd   F0 resvrd
;  16 resvrd   3B resvrd   60 ADDH2    82 resvrd   A7 resvrd   CC resvrd   F1 resvrd
;  17 resvrd   3C resvrd   61 ADDH3    83 resvrd   A8 resvrd   CD resvrd   F2 resvrd
;  18 resvrd   3D resvrd   62 SUBH2    84 resvrd   A9 resvrd   CE resvrd   F3 resvrd
;  19 resvrd   3E resvrd   63 SUBH3    85 resvrd   AA resvrd   CF resvrd   F4 resvrd
;  1A resvrd   3F resvrd   64 MULH2    86 resvrd   AB resvrd   D0 resvrd   F5 resvrd
;  1B resvrd   40 ADDG2    65 MULH3    87 resvrd   AC resvrd   D1 resvrd   F6 CVTHF
;  1C resvrd   41 ADDG3    66 DIVH2    88 resvrd   AD resvrd   D2 resvrd   F7 CVTHD
;  1D resvrd   42 SUBG2    67 DIVH3    89 resvrd   AE resvrd   D3 resvrd   F8 resvrd
;  1E resvrd   43 SUBG3    68 CVTHB    8A resvrd   AF resvrd   D4 resvrd   F9 resvrd
;  1F resvrd   44 MULG2    69 CVTHW    8B resvrd   B0 resvrd   D5 resvrd   FA resvrd
;  20 resvrd   45 MULG3    6A CVTHL    8C resvrd   B1 resvrd   D6 resvrd   FB resvrd
;  21 resvrd   46 DIVG2    6B CVTRHL   8D resvrd   B2 resvrd   D7 resvrd   FC resvrd
;  22 resvrd   47 DIVG3    6C CVTBH    8E resvrd   B3 resvrd   D8 resvrd   FD resvrd
;  23 resvrd   48 CVTGB    6D CVTWH    8F resvrd   B4 resvrd   D9 resvrd   FE resvrd
;  24 resvrd   49 CVTGW    6E CVTLH    90 resvrd   B5 resvrd   DA resvrd   FF resvrd
```

```
;999    .TOC    "       Tables and charts     : Compatibility mode opcode chart"
;1000   ; +---------------+----------+-----------------------------+ +---------------+----------+----------------------------+
;1001   ; |    16-Bit     | Legal    |          Faulting           | |    16-Bit     | Legal    |          Faulting          |
;1002   ; |    Opcode     | Instruc. |        Instructions         | |    Opcode     | Instruc. |        Instructions        |
;1003   ; +---------------+----------+-----------------------------+ +---------------+----------+----------------------------+
;1004   ; | 000000        |          | HALT    (Rsvrd   inst fault) | | 050000-057777 | BIS      |                            |
;1005   ; | 000001        |          | WAIT    (Rsvrd   inst fault) | | 060000-067777 | ADD      |                            |
;1006   ; | 000002        | RTI      |                              | | 070000-070777 | MUL      |                            |
;1007   ; | 000003        |          | BPT     (BPT     inst fault) | | 071000-071777 | DIV      |                            |
;1008   ; | 000004        |          | IOT     (IOT     inst fault) | | 072000-072777 | ASH      |                            |
;1009   ; | 000005        |          | RESET   (Rsvrd   inst fault) | | 073000-073777 | ASHC     |                            |
;1010   ; | 000006        | RTT      |                              | | 074000-074777 | XOR      |                            |
;1011   ; | 000007        |          | MFPT    (Rsvrd   inst fault) | | 075000-075007 |          | FADD    (Rsvrd inst fault) |
;1012   ; | 000010-000077 |          | Unused  (Rsvrd   inst fault) | | 075010-075017 |          | FSUB    (Rsvrd inst fault) |
;1013   ; | 000100-000107 |          | JMP     (Illegal inst fault) | | 075020-075027 |          | FMUL    (Rsvrd inst fault) |
;1014   ; | 000110-000177 | JMP      |                              | | 075030-075037 |          | FDIV    (Rsvrd inst fault) |
;1015   ; | 000200-000207 | RTS      |                              | | 075040-075777 |          | Unused  (Rsvrd inst fault) |
;1016   ; | 000210-000227 |          | Unused  (Rsvrd   inst fault) | | 076000-076777 |          | XTD INS (Rsvrd inst fault) |
;1017   ; | 000230-000237 |          | SPL     (Rsvrd   inst fault) | | 077000-077777 | SOB      |                            |
;1018   ; | 000240        | NOP      |                              | | 100000-100377 | BPL      |                            |
;1019   ; | 000241-000257 | CLR CC'S |                              | | 100400-100777 | BMI      |                            |
;1020   ; | 000260-000277 | SET CC'S |                              | | 101000-101377 | BHI      |                            |
;1021   ; | 000300-000377 | SWAB     |                              | | 101400-101777 | BLOS     |                            |
;1022   ; | 000400-000777 | BR       |                              | | 102000-102377 | BVC      |                            |
;1023   ; | 001000-001377 | BNE      |                              | | 102400-102777 | BVS      |                            |
;1024   ; | 001400-001777 | BEQ      |                              | | 103000-103377 | BCC,BHIS |                            |
;1025   ; | 002000-002377 | BGE      |                              | | 103400-103777 | BCS,BLO  |                            |
;1026   ; | 002400-002777 | BLT      |                              | | 104000-104377 |          | EMT     (BPT inst fault)   |
;1027   ; | 003000-003377 | BGT      |                              | | 104000-104777 |          | TRAP    (TRAP inst fault)  |
;1028   ; | 003400-003777 | BLE      |                              | | 105000-105077 | CLRB     |                            |
;1029   ; | 004x00-004x07 |          | JSR     (Illegal inst fault) | | 105100-105177 | COMB     |                            |
;1030   ; | 004x10-004x77 | JSR      |                              | | 105200-105277 | INCB     |                            |
;1031   ; | 005000-005077 | CLR      |                              | | 105300-105377 | DECB     |                            |
;1032   ; | 005100-005177 | COM      |                              | | 105400-105477 | NEGB     |                            |
;1033   ; | 005200-005277 | INC      |                              | | 105500-105577 | ADCB     |                            |
;1034   ; | 005300-005377 | DEC      |                              | | 105600-105677 | SBCB     |                            |
;1035   ; | 005400-005477 | NEG      |                              | | 105700-105777 | TSTB     |                            |
;1036   ; | 005500-005577 | ADC      |                              | | 106000-106077 | RORB     |                            |
;1037   ; | 005600-005677 | SBC      |                              | | 106100-106177 | ROLB     |                            |
;1038   ; | 005700-005777 | TST      |                              | | 106200-106277 | ASRB     |                            |
;1039   ; | 006000-006077 | ROR      |                              | | 106300-106377 | ASLB     |                            |
;1040   ; | 006100-006177 | ROL      |                              | | 106400-106477 |          | MTPS    (Rsvrd inst fault) |
;1041   ; | 006200-006277 | ASR      |                              | | 106500-106577 | MFPD     |                            |
;1042   ; | 006300-006377 | ASL      |                              | | 106600-106677 | MTPD     |                            |
;1043   ; | 006400-006477 |          | MARK    (Rsvrd inst fault)   | | 106700-106777 |          | MFPS    (Rsvrd inst fault) |
;1044   ; | 006500-006577 | MFPI     |                              | | 107000-107777 |          | Unused  (Rsvrd inst fault) |
;1045   ; | 006600-006677 | MTPI     |                              | | 110000-117777 | MOV3     |                            |
;1046   ; | 006700-006777 | SXT      |                              | | 120000-127777 | CMPB     |                            |
;1047   ; | 007000-007077 |          | CSM     (Rsvrd inst fault)   | | 130000-137777 | BITB     |                            |
;1048   ; | 007100-007777 |          | Unused  (Rsvrd inst fault)   | | 140000-147777 | BICB     |                            |
;1049   ; | 010000-017777 | MOV      |                              | | 150000-157777 | BISB     |                            |
;1050   ; | 020000-027777 | CMP      |                              | | 160000-167777 | SUB      |                            |
;1051   ; | 030000-037777 | BIT      |                              | | 170000-177777 |          | FLOAT   (Rsvrd inst fault) |
;1052   ; | 040000-047777 | BIC      |                              | |               |          |                            |
;1053   ; +---------------+----------+-----------------------------+ +---------------+----------+----------------------------+
```

```
;1054   .TOC "  Macro Level Charts              : Operand Specifier Addressing Modes"
;1055
;1056   ; General Register Addressing
;1057   ;
;1058   ;           7     4 3     0
;1059   ;          +-------+-------+
;1060   ;          ! mode  !  reg  !
;1061   ;          +-------+-------+
;1062   ;
;1063   ; Hex  Dec  Name                        Assembler    r m w a v   PC SP  Indexable?
;1064   ;
;1065   ; 0-3  0-3  literal                     S^#literal   y f f f f   -  -       f
;1066   ;  4    4   indexed                     i[Rx]        y y y y y   f  y       f
;1067   ;  5    5   register                    Rn           y y y f y   u  uq      f
;1068   ;  6    6   register deferred           (Rn)         y y y y y   u  y       y
;1069   ;  7    7   autodecrement               -(Rn)        y y y y y   u  y       ux
;1070   ;  8    8   autoincrement               (Rn)+        y y y y y   p  y       ux
;1071   ;  9    9   autoincrement deferred      @(R)+        y y y y y   p  y       ux
;1072   ;  A   10   byte displacement           B^D(Rn)      y y y y y   p  y       y
;1073   ;  B   11   byte displacement deferred  @B^D(Rn)     y y y y y   p  y       y
;1074   ;  C   12   word displacement           W^D(Rn)      y y y y y   p  y       y
;1075   ;  D   13   word displacement deferred  @W^D(Rn)     y y y y y   p  y       y
;1076   ;  E   14   longword displacement       L^D(Rn)      y y y y y   p  y       y
;1077   ;  F   15   longword displacement
;1078   ;           deferred                    @L^D(Rn)     y y y y y   p  y       y
;1079   ;
;1080   ; Program Counter Addressing (reg=15)
;1081   ;
;1082   ;           7     4 3 2 1 0
;1083   ;          +-------+-+-+-+-+
;1084   ;          ! mode  !1 1 1 1!
;1085   ;          +-------+-+-+-+-+
;1086   ;
;1087   ; Hex  Dec  Name                        Assembler    r m w a v   PC SP  Indexable?
;1088   ;
;1089   ;  8    8   immediate                   I^#constant  y u u y y   -  -       y
;1090   ;  9    9   absolute                    @#address    y y y y y   -  -       y
;1091   ;  A   10   byte relative               B^address    y y y y y   -  -       y
;1092   ;  B   11   byte relative deferred      @B^address   y y y y y   -  -       y
;1093   ;  C   12   word relative               W^address    y y y y y   -  -       y
;1094   ;  D   13   word relative deferred      @W^address   y y y y y   -  -       y
;1095   ;  E   14   long word relative          L^address    y y y y y   -  -       y
;1096   ;  F   15   long word relative
;1097   ;           deferred                    @L^address   y y y y y   -  -       y
;1098   ;
;1099   ;    D -  displacement                   r -  read access
;1100   ;    i -  any indexable addressing mode  m -  modify access
;1101   ;    - -  logically impossible           w -  write access
;1102   ;    f -  reserved addressing mode fault a -  address access
;1103   ;    p -  Program Counter addressing     v -  field access
;1104   ;    u -  UNPREDICTABLE
;1105   ;    uq - UNPREDICTABLE for quad and double (and field if position + size
;1106   ;         greater than 32)
;1107   ;    ux - UNPREDICTABLE for index register same as base register
```

```
;1108   .TOC "  Macro Level Charts                : Console Commands"
;1109
;1110   ;     E [<QUALIFIER-LIST>] [<SP> <ADDRESS>] <CR>                ; Examine
;1111   ;     D [<QUALIFIER-LIST>] <SP> <ADDRESS> <SP> <DATA> <CR>      ; Deposit
;1112   ;         Qualifiers :     /B         ; Size IS Byte
;1113   ;                          /W         ; Size IS Word
;1114   ;                          /L         ; Size IS Long
;1115   ;                          /V         ; Virtual Address
;1116   ;                          /P         ; Physical Address
;1117   ;                          /I         ; IPR
;1118   ;                          /G         ; GPR
;1119   ;         Address     : Hex Number
;1120   ;                            *        ; Last location
;1121   ;                            P        ; The PSL
;1122   ;
;1123   ;     E<CR>                                                     ; Examine Next Location
;1124   ;
;1125   ;     D<SP> + <SP> <DATA> <SP> <CR>                             ; Deposit Next Location
;1126   ;
;1127   ;     H <CR>                                                    ; HALT (No-op)
;1128   ;
;1129   ;     I <CR>                                                    ; Initialize
;1130   ;
;1131   ;     T <CR>                                                    ; Execute Micro Verify Sequence
;1132   ;
;1133   ;     C <CR>                                                    ; Continue the Processor
;1134   ;
;1135   ;     N <CR>                                                    ; Single Step
;1136   ;
;1137   ;     B [<QUALIFIER-LIST>] [<SP> ddcu ] <CR>                    ; Boot
;1138   ;         Qualifiers : /X             ; Inhibit Execution of Microverify Sequence
;1139   ;                    : /Hex Number ; Boot Control Flags (R5)
;1140   ;         dd                          ; Device
;1141   ;         c                           ; Adapter
;1142   ;         u                           ; Unit number
;1143   ;
;1144   ;     S <SP> <ADDRESS> <CR>                                     ; Start
;1145   ;                                   ; Initialize the Processor
;1146   ;                                   ; PC <- Address
;1147   ;                                   ; Start the processor
;1148   ;
;1149   ;     X <SP><ADDRESS><SP><0'COUNT><CR><CKSUM1><DATA><CKSUM2>   ; Binary Load
;1150   ;         Address     ; Start Address of Load
;1151   ;         Count       ; No. of Bytes (Unsigned 30 bit number)
;1152   ;         Cksum1      ; 2's Comp CheCksum of Command String
;1153   ;         Data        ; Count Bytes of Binary Data
;1154   ;         Cksum2      ; 2's Comp Checksum of Data
;1155   ;
;1156   ;     X <SP> <ADDRESS> <SP> <1'COUNT> <CR> <CKSUM>              ; Binary Read
;1157   ;         Address     ; Start Address of Load
;1158   ;         Count       ; No. of Bytes (Unsigned 30 bit number)
;1159   ;         Cksum       ; 2's Comp CheCksum of Command String
```

```
.TOC "  Macro Level Charts              : Console Error and Halt Codes"


;  +-----------------------------------------------------------------------------------------------+
;  |                                     Console Error Codes                                       |
;  +-----------------------------------------------------------------------------------------------+
;  |                                                                                               |
;  |   20         ACV, TNV, or Machine Check during a read or write.                               |
;  |                                                                                               |
;  |   11         Error in accessing IPR, or PSL.                                                  |
;  |                                                                                               |
;  |   30         Binary transfer checksum error.                                                  |
;  |                                                                                               |
;  |   33         Unrecognizable Boot device.                                                      |
;  |                                                                                               |
;  |   34         Controller not A, B, C, or D in BOOT command.                                    |
;  |                                                                                               |
;  +-----------------------------------------------------------------------------------------------+





;  +-----------------------------------------------------------------------------------------------+
;  |                                      Console Halt Codes                                       |
;  +-----------------------------------------------------------------------------------------------+
;  |                                                                                               |
;  |   1  Executed TEST console command                                                            |
;  |   2  Control P Halt or single macro instruction mode (ie >>>N<CR>)                            |
;  |   4  Interrupt Stack Not Valid                                                                |
;  |   5  Double Bus Write Error Halt                                                              |
;  |   6  Halt Instruction Executed                                                                |
;  |   7  Vector Bits <1:0>=3, Halt at Vector                                                      |
;  |   8  Vector Bits <1:0>=2, WCS disabled or not present                                         |
;  |   A  Change Mode Instruction executed on Interrupt Stack                                      |
;  |   B  Change Mode Instruction executed and vector <1:0>not=0                                   |
;  |   11 Power up and can't find RPB, FPS1 at RESTART/HALT                                        |
;  |   12 Power up and warm start flag false FPS1 at RESTART/HALT                                  |
;  |   13 Power up and can't find good 64K of memory                                               |
;  |   14 Power up and booting, but bad Boot ROM or no ROM                                         |
;  |   15 Power up and cold start flag set during boot subroutine                                  |
;  |   16 Power up halt FPS1 at HALT position                                                      |
;  |   FF Micro verify test failure                                                                |
;  +-----------------------------------------------------------------------------------------------+
```

```
;1204   .TOC "  Macro Level Charts                  : Micro Verify Error Codes"
;1205   ; +-----+-----+------------------------------+-----+-----+-----------------------------------------------------------+
;1206   ; |  C  |  P  |                              |  C  |  P  |                                                           |
;1207   ; |  O  |  C  |                              |  O  |  C  |                                                           |
;1208   ; |  D  |  +  |  TEST NAME/ERROR MESSAGE     |  D  |  +  |  TEST NAME/ERROR MESSAGE                                  |
;1209   ; |  E  |  2  |                              |  E  |  2  |                                                           |
;1210   ; +-----+-----+------------------------------+-----+-----+-----------------------------------------------------------+
;1211   ; | '@' |     | RBUS, WBUS TEST              | 'O' |     | XB/IR/OSR BIT TEST                                        |
;1212   ; |     | 000 | BAD BIT IN DREG OR SUPROT    |     | 0F1 | ERROR IN XB<31:0>                                         |
;1213   ; |     | 001 | BAD BIT IN RBUS OR WBUS      |     | 0F2 | ERROR IN XB<63:32>                                        |
;1214   ; |     |     |                              |     | 0F4 | ERROR IN IR                                               |
;1215   ; | 'C' |     | MBUS TEST                    |     | 0F7 | ERROR IN OSR                                              |
;1216   ; |     | 031 | BAD BIT IN QREG              |     |     |                                                           |
;1217   ; |     | 032 | BAD BIT IN MBUS              | 'Q' |     | SOURCE XB PC INCREMENT TEST                               |
;1218   ; |     |     |                              |     | 111 | ERROR SOURCING ONE BYTE FROM XB                           |
;1219   ; | 'E' |     | SCRATCH PAD BIT TEST         |     | 112 | ERROR SOURCING 2 BYTES FROM XB OR INCREMENTING PC BY 1    |
;1220   ; |     | 051 | ERROR CLEARING RTEMP         |     | 114 | ERROR SOURCING AN UNALIGNED LONGWORD  OR INC PC BY 2      |
;1221   ; |     | 052 | ERROR FILLING RTEMP WITH ONES |    | 117 | ERROR INCREMENTING PC BY 4                                |
;1222   ; |     | 054 | ERROR CLEARING GPR           |     |     |                                                           |
;1223   ; |     | 057 | ERROR FILLING GPR WITH ONES  | 'R' |     | RNUM/DSIZE TEST                                           |
;1224   ; |     | 058 | ERROR CLEARING IPR           |     | 121 | ERROR READING DSIZE ROM OPERAND 1                         |
;1225   ; |     | 05B | ERROR FILLING IPR WITH ONES  |     | 122 | ERROR LOADING/READING RNUM                                |
;1226   ; |     | 05D | ERROR CLEARING MTEMP         |     | 124 | ERROR READING DSIZE ROM OPERAND 2                         |
;1227   ; |     | 05E | ERROR FILLING MTEMP WITH ONES |    | 127 | ERROR LOADING/READING RNUM                                |
;1228   ; |     |     |                              |     | 128 | ERROR READING DSIZE ROM OPERAND 3                         |
;1229   ; | 'F' |     | MTEMP EXPLICIT ADDRESS TEST  |     | 12B | ERROR LOADING/READING RNUM                                |
;1230   ; |     | 061 | ERROR ADDRESSING MTEMP0      |     | 12D | ERROR READING DSIZE ROM OPERAND 4                         |
;1231   ; |     | 062 | ERROR ADDRESSING MTEMP1      |     | 12E | ERROR LOADING/READING RNUM                                |
;1232   ; |     | 064 | ERROR ADDRESSING MTEMP2      |     |     |                                                           |
;1233   ; |     | 067 | ERROR ADDRESSING MTEMP4      | 'T' |     | RNUM/DSIZE TEST CONTINUED                                 |
;1234   ; |     | 068 | ERROR ADDRESSING MTEMP8      |     | 141 | ERROR READING DSIZE ROM OPERAND 5                         |
;1235   ; |     |     |                              |     | 142 | ERROR LOADING/READING RNUM                                |
;1236   ; | 'I' |     | RTEMP EXPLICIT ADDRESS TEST  |     | 144 | ERROR READING DSIZE ROM OPERAND 6                         |
;1237   ; |     | 091 | ERROR ADDRESSING RTEMP0      |     |     |                                                           |
;1238   ; |     | 092 | ERROR ADDRESSING RTEMP1      | 'X' |     | CACHE PARITY ERROR TEST                                   |
;1239   ; |     | 094 | ERROR  ADDRESSING RTEMP2     |     | 181 | FAILED TO GET CACHE PARITY ERROR                          |
;1240   ; |     | 097 | ERROR ADDRESSING RTEMP4      |     | 182 | BAD MACHINE CHECK ERROR SUMMARY REGISTER                  |
;1241   ; |     | 098 | ERROR ADDRESSING RTEMP8      |     | 184 | BAD CACHE ERROR REGISTER                                  |
;1242   ; |     |     |                              |     |     |                                                           |
;1243   ; | 'J' |     | IPR EXPLICIT ADDRESS TEST    | '[' |     | TB PARITY ERROR TEST                                      |
;1244   ; |     | 0A1 | ERROR ADDRESSING IPR0        |     | 1B1 | FAILED TO GET GROUP 0 TB PARITY ERROR                     |
;1245   ; |     | 0A2 | ERROR ADDRESSING IPR1        |     | 1B2 | BAD TB GROUP PARITY ERROR REGISTER                        |
;1246   ; |     | 0A4 | ERROR ADDRESSING IPR2        |     | 1B4 | BAD MACHINE CHECK ERROR SUMMARY REGISTER                  |
;1247   ; |     | 0A7 | ERROR ADDRESSING IPR4        |     | 1B7 | FAILED TO GET GROUP 1 TB PARITY ERROR                     |
;1248   ; |     | 0A8 | ERROR ADDRESSING IPR8        |     | 1B8 | BAD TB GROUP PARITY ERROR REGISTER                        |
;1249   ; |     |     |                              |     | 1BB | BAD MACHINE CHECK ERROR SUMMARY REGISTER                  |
;1250   ; | 'L' |     | GPR EXPLICIT ADDRESS TEST    |     |     |                                                           |
;1251   ; |     | 0C1 | ERROR ADDRESSING R0          | ']' |     | CONTROL STORE PARITY ERROR TEST                           |
;1252   ; |     | 0C2 | ERROR ADDRESSING R1          |     | 1D1 | FAILED TO GET CONTROL STORE PARITY ERROR                  |
;1253   ; |     | 0C4 | ERROR ADDRESSING R2          |     | 1D2 | ERROR IN CONTROL STORE PARITY ERROR                       |
;1254   ; |     | 0C7 | ERROR ADDRESSING R4          |     |     |                                                           |
;1255   ; |     | 0C8 | ERROR ADDRESSING R8          | '^' |     | CACHE TEST                                                |
;1256   ; |     | 0CE | ERROR ADDRESSING DUAL PORT   |     | 1E1 | ERROR FILLING CACHE WITH ONES. LOCATION NOT INITIALLY = 0 |
;1257   ; |     |     |                              |     | 1E2 | ERROR FILLING CACHE WITH ONES. UNABLE TO WRITE ONES       |
;1258   ; +-----+-----+------------------------------+-----+-----+-----------------------------------------------------------+
```

```
.TOC "  Micro Level Charts             : Micro Code Label Prefixes"
;   Label ID    Micro Code Description
;       BO      Bootstrap
;       CH      Change Mode
;       CM      Compatability Mode
;       CN      Console Command Parser
;       CO      Control
;       CR      Cyclic Redundancy Check
;       CS      Character String
;       DS      Decimal String
;       ED      Edit
;       FI      FPA Interface
;       FP      Floating Point
;       FX      Floating Point Extension For G and H Data Types
;       GL      Global Code
;       IE      Interrupts and Exceptions
;       IL      Integer, Logical, & Address
;       IN      Initialize
;       LS      Load and Save Processor Context
;       MM      Memory Management
;       MP      Move To and From Processor Register
;       MS      Miscellaneous
;       MV      Micro Verify
;       OS      Native Mode Operand Specifier
;       PC      Procedure Call
;       PR      Probe
;       QU      Queue
;       VF      Variable Length Bit Field
```

```
.TOC "  Micro Level Charts              : Fixed Control Store Address"


;
;    +------+------------------------------------+---------------------------+
;    | Add  | Function of Vector                 | Method of Initiation      |
;    +------+------------------------------------+---------------------------+
;    | 0000 | Power Up                           |                           |
;    | 0011 | Arithmetic Trap                    | DO Service                |
;    | 0012 | FPA Integer Overflow Trap          | DO Service                |
;    | 0014 | Timer Service                      | DO Service                |
;    | 0015 | T-Bit trap                         | DO Service                |
;    | 0016 | Console ^P Trap                    | DO Service                |
;    |      |                                    |                           |
;    | 0020 | Control Store Parity Error         | Micro Trap                |
;    | 0021 | Read Unaligned Data                | Micro Trap                |
;    | 0022 | MSRC XB Miss                       | Micro Trap                |
;    | 0023 | MSRC XB ACV                        | Micro Trap                |
;    | 0024 | write Unlock Unaligned Data        | Micro Trap                |
;    | 0025 | Write Unaligned Data               | Micro Trap                |
;    | 0026 | Write Unlock Crossing Page Boundry | Micro Trap                |
;    | 0027 | write Crossing Page Boundry        | Micro Trap                |
;    | 0028 | Machine Check Exceptions(See Note) | Micro Trap                |
;    | 0029 | But XB Miss                        | Micro Trap                |
;    | 002A | Read TB Miss                       | Micro Trap                |
;    | 002B | Write TB Miss                      | Micro Trap                |
;    | 002C | FPA Reserved Operand               | Micro Trap                |
;    | 002D | But XB ACV                         | Micro Trap                |
;    | 002E | Read ACV                           | Micro Trap                |
;    | 002F | Write ACV                          | Micro Trap                |
;    |      |                                    |                           |
;    | 0038 | Soft Interrupt                     | DO Service, Execution Flows |
;    | 0039 | Console Interrupt                  | DO Service, Execution Flows |
;    | 003A | Unibus  Interrupt                  | DO Service, Execution Flows |
;    | 003B | Interval Timer Interrupt           | DO Service, Execution Flows |
;    | 003C | Corrected Memory Interrupt         | DO Service, Execution Flows |
;    | 003E | Write Bus Error Interrupt          | DO Service, Execution Flows |
;    | 003F | Power Fail                         | DO Service, Execution Flows |
;    +------+------------------------------------+---------------------------+
;
;     NOTE :  MSRC XB TB Error
;             MSRC XB Bus Error
;             Bus Error
;             Unaligned UNIBUS Data
;             TB Error
;             But XB TB Error
;             But XB Bus Error
```

```
.TOC "  Micro Level Charts             : BUT/UVCTR Chart"


; BUT/UVCTR is not valid During any micro cycle that follows an IRD1.
; Because of this, it cannot be used in any of the micro instructions that are pointed to by the IRD1 rom.

;                       +----------------------------+---------------------------+-------------------+-------------------+
;                       |                            |                           |                   |                   |
;                       |          UVCTR<3>          |          UVCTR<2>         |     UVCTR<1>      |     UVCTR<0>      |
;                       |                            |                           |                   |                   |
; +---------------------+----------------------------+---------------------------+-------------------+-------------------+
; |                     |                            |                           |                   |                   |
; | BUS/PRB.RD          |                            |                           |                   |                   |
; | BUS/PRB.RD.MODE     |                            |                           |                   |                   |
; | BUS/PRB.WR          | (PBOK.AND.V.AND.AC).OR.PA  | M.AND.((V.AND.AC).OR.PA)  |      V.OR.PA      | (AC.AND.V).OR.PA  |
; | BUS/PRB.WR.MODE     |                            |                           |                   |                   |
; |                     |                            |                           |                   |                   |
; +---------------------+----------------------------+---------------------------+-------------------+-------------------+
; |                     |                            |                           |                   |                   |
; | BUS/PRB.RD.PTE      |                            |                           |                   |                   |
; | BUS/PRB.RD.PTE.K    |             0              |      M.AND.V.AND.AC       |     V.AND.AC      |        AC         |
; | BUS/PRB.WR.PTE      |                            |                           |                   |                   |
; |                     |                            |                           |                   |                   |
; +---------------------+----------------------------+---------------------------+-------------------+-------------------+
; |                     |                            |                           |                                       |
; | WCTRL/UVCTR_CM.IS   |             1              |          PSL<IS>          |                PSL<CUR>               |
; |                     |                            |                           |                                       |
; +---------------------+----------------------------+---------------------------+---------------------------------------+
; |                     |                            |                           |                                       |
; |                     |                            |                           |   0 = REI Check Ok                    |
; | WCTRL/REICHK and    |             1              |             0             |   1 = REI Check Ok & AST Pending      |
; | Wbus = Saved Psl    |                            |                           |   2 = REI Check is not Ok             |
; |                     |                            |                           |   3 = REI Check is not Ok             |
; |                     |                            |                           |                                       |
; +---------------------+----------------------------+---------------------------+---------------------------------------+
; |                     |                            |                                                                   |
; | WCTRL is not        |                            |                     0 = Soft Interrupt                            |
; | UVCTR_CM.IS or      |                            |                     1 = Console Interrupt                         |
; | REICHK(see above)   |                            |                     2 = Unibus Interrupt                          |
; |                     |             1              |                     3 = Interval Timer Interrupt                  |
; | BUS is not one      |                            |                     4 = Corrected Memory Interrupt                |
; | the Probe Micro     |                            |                     6 = Write Bus Error Interrupt                 |
; | Orders(see above)   |                            |                     7 = Power Fail Interrupt                      |
; |                     |                            |                                                                   |
; +---------------------+----------------------------+-------------------------------------------------------------------+
;
; Legend : M    = PTE Modify Bit
;          V    = 1 if Valid PTE
;          AC   = 1 if Access Allowed
;          PBOK = 1 if Not Crossing a Page Boundry
;          PA   = 1 if Memory Mapping is not Enabled
```

```
.TOC "  Micro Level Charts             : Description of FPD "

;  When PSL<FPD> is set, there are 4 things that might require the instruction
;  flow to be interrupted and the instruction packed up.
;
;       1) Memory Fault (ACV or TNV),
;       2) Reserved Operand Fault,
;       3) Floating Point Fault (overflow, underflow), or
;       4) Interrupt.
;
;  In the first case the memory management flows will branch to the
;  pack routine at IE.FPD.PACK + FPDOFFSET, and initiate the fault.
;
;  In the second case the micro code will branch to IE.OPER.FAULT which
;  will branch to the pack routine and initiate the fault.
;
;  In the third case the micro code will branch to IE.FLxxx.FAULT which
;  will branch to the pack routine and initiate the fault.
;
;  In the fourth case when the micro code decides to allow an interrupt it will
;  branch to IE.SERV.IP.TS which will branch to the pack routine and initiate
;  the interrupt (interrupts may be taken by the memory management flows
;  if a TB miss occurs and the flag MM.NOINT is clear).
;
;  The micro code pack routines loaded at IE.FPD.PACK + FPDOFFSET observe
;  the following rules for packing:
;
;       1) When packing is complete the pack routine will branch
;          to IE.PACK.DONE.
;
;       2) For faults, DREG is modified by IANDE before going to the
;          pack routine and must not be modified by the pack routine.
;
;       3) For floating faults and reserved operand faults, FLAG2 is modified
;          by IANDE before going to the pack routine and must not be modified
;          by the pack routine.
;
;       4) Micro temp FPDOFFSET<16:8> is modified by IANDE before going to the
;          pack routine and must not be modified by the pack routine
;          (FPDOFFSET<7:0> will be preserved).
;
;       5) In some cases the pack routine may need to return to the instruction
;          flows before branching to IE.PACK.DONE, if so the address at the
;          top of the micro stack will have one of two possible values:
;               a) IF the pack routine was branched to because of a memory
;                     management fault or because of an interrupt taken from a
;                     tbmiss routine (MM.NOINT clear)
;                  THEN the address at the top of the micro stack is the
;                     address of the micro trapped instruction that caused the
;                     tb miss.
;               b) IF the pack routine was reached by any other path
;                     (ie. NEXT/IE.SERV.IP.TS or NEXT/IE.OPER.FAULT)
;                  THEN the address at the top of the micro stack is the address
;                     of the last unmatched PUSH in the instruction flows.
```

```
.TOC "  Micro Level Charts            : Compatability Mode Condition Codes"

; CCOPS are not valid During any micro cycle that follows an IRD1. Because of this, they cannot be used in any of the micro
; instructions that are pointed to by the IRD1 entries in the Compatibility mode rom.

; NOTE 1 : 'SIGN', 'WX', 'OV', 'CRY', ARE ALL FUNCTIONS OF DSIZE
; NOTE 2 : (SIGN.XOR.OV).OR.CRY

; +---------------+--------+--------------------+--------------+----------------+-------+
; | INSTRUCTION   |   N    |         Z          |      V       |       C        | CCOPx |
; +---------------+--------+--------------------+--------------+----------------+-------+
; | ADC(B)        | SIGN   | WX.EQ.0            | OV           | CRY            |   1   |
; | ADD           | SIGN   | WX.EQ.0            | OV           | CRY            |   1   |
; | ASH           | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; | ASHC          | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; |               | SIGN   | (WX.EQ.0).AND.Z    | 0            | C              |   2   |
; | ASL           | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31:16>.NE.0 |   2   |
; | ASLB          | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31:8>.NE.0  |   2   |
; | ASR(B)        | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31>         |   1   |
; | BI(T,S,C)(B)  | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; | CLR(B)        | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; | CMP(B)        | SIGN   | WX.EQ.0            | OV           | .NOT.CRY       |   2   |
; | COM(B)        | SIGN   | WX.EQ.0            | OV           | .NOT.CRY       |   2   |
; | DEC(B)        | SIGN   | WX.EQ.0            | OV           | C              |   1   |
; | DIV           | SIGN   | WX.EQ.0            | OV           | CRY            |   1   |
; | INC(B)        | SIGN   | WX.EQ.0            | OV           | C              |   1   |
; | MFP(I,D)      | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; | MTP(I,D)      | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; | MOV(B)        | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; | MUL           | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; |               | NOTE 2 | (WX.EQ.0).AND.Z    | 0            | WB<L>.NE.0     |   2   |
; | NEG(B)        | SIGN   | WX.EQ.0            | OV           | .NOT.CRY       |   2   |
; | ROL           | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31:16>.NE.0 |   2   |
; | ROLB          | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31:8>.NE.0  |   2   |
; | ROR(B)        | SIGN   | WX.EQ.0            | N.XOR.C(IN)  | WB<31>         |   1   |
; | SBC(B)        | SIGN   | WX.EQ.0            | OV           | .NOT.CRY       |   2   |
; | SUB           | SIGN   | WX.EQ.0            | OV           | .NOT.CRY       |   2   |
; | SWAB          | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; | SXT           | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; | TSI(B)        | SIGN   | WX.EQ.0            | 0            | 0              |   1   |
; | XOR           | SIGN   | WX.EQ.0            | 0            | C              |   1   |
; +---------------+--------+--------------------+--------------+----------------+-------+
```

```
.TOC "  Micro Level Charts            : Native Mode Condition Codes Part 1"


; CCOPS are not valid During any micro cycle that follows an IRD1. Because of this, they cannot be used in any of the micro
; instructions that are pointed to by the IRD1 rom.
; NOTE 1 : 'SIGN', 'WX', 'OV', 'CRY', ARE ALL FUNCTIONS OF DSIZE
; NOTE 2 : WB<15> + [(WX<15:0>.EQ.0).AND.CRY]

; +-------------------+-------------+------------------+------------+-----------+-------+
; | INSTRUCTION       |      N      |        Z         |      V     |     C     | CCOPx |
; +-------------------+-------------+------------------+------------+-----------+-------+
; | ACB(B,W)          | SIGN        | WX.EQ.0          | OV         | C         |   1   |
; | ACBL              | SIGN        | WX.EQ.0          | OV         | C         |   2   |
; | ACB(F,D)          | WB<15>      | WX.EQ.0          | 0          | C         |   2   |
; | ADAWI             | SIGN        | WX.EQ.0          | OV         | CRY       |   2   |
; | ADD(B,W,L)(2,3)   | SIGN        | WX.EQ.0          | OV         | CRY       |   1   |
; | ADD(F,D)(2,3)     | WB<15>      | WX.EQ.0          | 0          | 0         |   1   |
; | ADWC              | SIGN        | WX.EQ.0          | OV         | CRY       |   1   |
; | AOB(LEQ,LSS)      | SIGN        | WX.EQ.0          | OV         | C         |   2   |
; | ASHL              | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | ASHQ              | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; |                   | SIGN        | (WX.EQ.0).AND.Z  | 0          | C         |   2   |
; | BIC(B,W,L)(2,3)   | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | BIS(B,W,L)(2,3)   | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | BIT(B,W,L)        | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | CASE(B.W.L)       | SIGN.XOR.OV | WX.EQ.0          | 0          | .NOT.CRY  |   1   |
; | CLR(B,W,L)        | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | CLRD              | SIGN        | WX.EQ.0          | 0          | C         |   1   |
; | CLRF              | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | CLRQ              | SIGN        | WX.EQ.0          | OV         | C         |   1   |
; | CMP(B,W,L)        | SIGN.XOR.OV | WX.EQ.0          | 0          | .NOT.CRY  |   1   |
; | CMP(V,ZV)         | SIGN.XOR.OV | WX.EQ.0          | 0          | .NOT.CRY  |   1   |
; | CMPC(3,5)         | SIGN.XOR.OV | WX.EQ.0          | 0          | .NOT.CRY  |   1   |
; | CMPD              | SEE NOTE 2  | WX.EQ.0          | 0          | 0         |   2   |
; |                   | .NOT.CRY    | WX.EQ.0          | 0          | 0         |   1   |
; | CMPF              | WB<15>      | WX.EQ.0          | 0          | 0         |   1   |
; |                   | SEE NOTE 2  | WX.EQ.0          | 0          | 0         |   2   |
; | CRC               | SIGN        | WX.EQ.0          | 0          | C         |   2   |
; | CVT(BW,BL)        | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | CVT(FB,DB,FW,DW)  | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; |                   | N           | Z                | WX<L>.NE.0 | C         |   2   |
; | CVT(FD,DF,BF,BD,  | WB<15>      | WX.EQ.0          | 0          | 0         |   1   |
; |     WF,WD,LD,LF)  |             |                  |            |           |       |
; | CVT(FL,DL,        | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; |     RFL,RDL)      |             |                  |            |           |       |
; | CVT(LP,PL)        | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | CVT(WB,LB,LW)     | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; |                   | N           | Z                | WX<L>.NE.0 | C         |   2   |
; | CVTWL             | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | DEC(B,W,L)        | SIGN        | WX.EQ.0          | OV         | .NOT.CRY  |   1   |
; | DIV(B,W,L)(2,3)   | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | DIV(F,D)(2,3)     | WB<15>      | WX.EQ.0          | 0          | 0         |   1   |
; | EDIV              | SIGN        | WX.EQ.0          | 0          | 0         |   1   |
; | EMOD(F,D)         | WB<15>      | WX.EQ.0          | 0          | 0         |   1   |
; +-------------------+-------------+------------------+------------+-----------+-------+
```

```
.TOC "  Micro Level Charts                : Native Mode Condition Codes Part 2"

; 'SIGN', 'WX', 'OV', 'CRY', ARE ALL FUNCTIONS OF DSIZE

; +-----------------+-------------+-----------------+------------+----------+-------+
; | INSTRUCTION     |      N      |        Z        |      V     |    C     | CCOPx |
; +-----------------+-------------+-----------------+------------+----------+-------+
; | EMUL            | SIGN        | WX.EQ.0         | 0          | 0        |   1   |
; |                 | SIGN        | (WX.EQ.0).AND.Z | 0          | C        |   2   |
; | EXT(V,ZV)       | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | FF(S,C)         | 0           | WX.EQ.0         | 0          | 0        |   1   |
; | INC(B,W,L)      | SIGN        | WX.EQ.0         | OV         | CRY      |   1   |
; | INDEX           | SIGN        | WX.EQ.0         | 0          | 0        |   2   |
; | INSQUE          | SIGN.XOR.OV | WX.EQ.0         | 0          | .NOT.CRY |   1   |
; | INSV            | SIGN        | WX.EQ.0         | OV         | C        |   2   |
; | LOCC            | 0           | WX.EQ.0         | 0          | 0        |   1   |
; | M(T,F)PR        | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MATCH           | 0           | WX.EQ.0         | 0          | 0        |   1   |
; | MCOM(B,W,L)     | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MNEG(B,W,L)     | SIGN        | WX.EQ.0         | OV         | .NOT.CRY |   1   |
; | MNEG(F,D)       | WB<15>      | WX.EQ.0         | 0          | 0        |   1   |
; | MOV(B,W,L)      | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MOV(F,D)        | WB<15>      | WX.EQ.0         | 0          | C        |   2   |
; | MOVA(B,W,L)     | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MOVAQ           | SIGN        | WX.EQ.0         | 0          | C        |   1   |
; | MOVC(3,5)       | 0           | WX.EQ.0         | 0          | 0        |   2   |
; |                 | SIGN.XOR.OV | WX.EQ.0         | 0          | .NOT.CRY |   1   |
; | MOVQ            | SIGN        | WX.EQ.0         | OV         | C        |   1   |
; |                 | SIGN        | (WX.EQ.0).AND.Z | 0          | C        |   2   |
; | MOVTC           | SIGN.XOR.OV | WX.EQ.0         | 0          | .NOT.CRY |   1   |
; | MOVTUC          | SIGN.XOR.OV | WX.EQ.0         | 0          | .NOT.CRY |   1   |
; | MOVZ(BW,BL)     | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MOVZWL          | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | MUL(B,W,L)(2,3) | SIGN        | WX.EQ.0         | 0          | 0        |   1   |
; |                 | N           | Z               | WX<L>.NE.0 | C        |   2   |
; | MUL(F,D)(2,3)   | WB<15>      | WX.EQ.0         | 0          | 0        |   1   |
; | POLY(F,D)       | WB<15>      | WX.EQ.0         | 0          | 0        |   1   |
; | PROBE(R,W)      | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | PUSHA(B,W,L)    | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | PUSHAQ          | SIGN        | WX.EQ.0         | 0          | C        |   1   |
; | PUSHL           | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | REMQUE          | SIGN.XOR.OV | WX.EQ.0         | 0          | .NOT.CRY |   1   |
; |                 | N           | Z               | WX<L>.EQ.0 | C        |   2   |
; | ROTL            | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; | S(C,P)ANC       | 0           | WX.EQ.0         | 0          | 0        |   2   |
; | SBWC            | SIGN        | WX.EQ.0         | OV         | .NOT.CRY |   1   |
; | SKPC            | 0           | WX.EQ.0         | 0          | 0        |   1   |
; | SOB(GEQ,GTR)    | SIGN        | WX.EQ.0         | OV         | C        |   2   |
; | SUB(B,W,L)(2,3) | SIGN        | WX.EQ.0         | OV         | .NOT.CRY |   2   |
; | SUB(F,D)(2,3)   | WB<15>      | WX.EQ.0         | 0          | 0        |   1   |
; | TST(B,W,L)      | SIGN        | WX.EQ.0         | 0          | 0        |   1   |
; | TST(F,D)        | WB<15>      | WX.EQ.0         | 0          | 0        |   1   |
; | XOR(B,W,L)(2,3) | SIGN        | WX.EQ.0         | 0          | C        |   2   |
; +-----------------+-------------+-----------------+------------+----------+-------+
```

.TOC " Micro Level Charts : Native Mode Operand Specifier Chart"

	OS.RED	OS.MOD	OS.ADD	OS.VADD	OS.WRT1	OS.WRT2	OS.FRED	OS.QRED	OS.DMOD	OS.DRED
Rn	1	1	ERROR	1	1	1	1	2	2	2
(Rn)	2	2	1	1	1	1	2	3	4	3
(Rn)+	2	2	2	2	2	2	2	3	4	3
(PC)+	2	UNP	4	2	UNP	UNP	2	2	UNP	2
@(Rn)+	4	4	4	2	3	3	4	5	6	5
@(PC)+	2	2	2	1	1	2	2	3	4	3
-(Rn)	2	2	2	1	1	2	2	3	4	3
X^D(Rn)	2	2	2	1	1	2	2	3	4	3
X^D(PC)	2	2	3	1	1	2	2	3	4	3
@X^D(Rn)	4	4	2	2	3	3	4	5	6	5
@X^D(PC)	4	4	2	2	3	3	4	5	6	5
S^#lit	1	ERROR	ERROR	ERROR	ERROR	ERROR	1	2	ERROR	3
INDEX	4	4	4	4	4	4	4	5	6	5
PC INDEX	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR
MDR	OPERAND	OPERAND	OPERAND	OPER 1	UNP 3	UNP 3	OPERAND	MSB OPER	MSB OPER	MSB OPER
Q-REG	MDR OLD	MDR OLD	MDR OLD			MDR OLD	MDR OLD	MDR OLD		
TEMP0	UNP	UNP	UNP	UNP	UNP	UNP	UNP	UNP	UNP	UNP
TEMP1								LSB OPER	LSB OPER	LSB OPER
VA/RNUM	UNP 3	ADD 3	UNP	UNP 2	ADD 3	ADD 3	UNP 3	UNP 3	ADD 3	UNP 3
	RET+1	RET+1	RET+1	RET+3	RET+1	RET+1	RET+1	RET+1	RET+1	RET+1

; UNP : UNPREDICTABLE
; ADD : Address of the Operand is left in VA if the Operand is in memory and in RNUM if the Operand is in GPR
; NOTE :
; 1. In reg mode MDR <- GPR(RNUM), 2. In reg mode RNUM <- # of addressed reg, 3. In reg mode MDR & VA are not affected

CHARTS (CONT)

```
;1671   .TOC "  Micro Level Charts              :               Native Mode Addressing Branch"
;1672
;1673
;1674
;1675    ;    +---------+-------------+----------------------------------------+
;1676    ;    |         |  Operand    |                                        |
;1677    ;    | Branch  |  Specifier  |                                        |
;1678    ;    | Offset  +-------+-----+           Addressing Mode              |
;1679    ;    |         | Mode  | Reg |                                        |
;1680    ;    +---------+-------+-----+----------------------------------------+
;1681    ;    |         |       |     |                                        |
;1682    ;    |  0000   | 5     | 0-F | Rn         REGISTER MODE               |
;1683    ;    |         |       |     |                                        |
;1684    ;    |  0001   | 8     | 0-E | (Rn)+      AUTOINCREMENT MODE          |
;1685    ;    |         |       |     |                                        |
;1686    ;    |  0010   | 8     |  F  | I^#cons    IMMEDIATE MODE              |
;1687    ;    |         |       |     |                                        |
;1688    ;    |  0011   | 0-3   | --- | S^#cons    LITERAL MODE                |
;1689    ;    |         |       |     |                                        |
;1690    ;    |  0100   | 7     | 0-F | -(Rn)      AUTODECREMENT MODE          |
;1691    ;    |         |       |     |                                        |
;1692    ;    |  0101   | A,C,E |  F  | Addr       RELATIVE MODE               |
;1693    ;    |         |       |     |                                        |
;1694    ;    |  0110   | A,C,E | 0-E | D(Rn)      DISPLACEMENT MODE           |
;1695    ;    |         |       |     |                                        |
;1696    ;    |  0111   | 9     |  F  | @#Addr     ABSOLUTE MODE               |
;1697    ;    |         |       |     |                                        |
;1698    ;    |  1000   | 6     | 0-F | (Rn)       REGISTER DEFERRED MODE      |
;1699    ;    |         |       |     |                                        |
;1700    ;    |  1001   | B,D,F |  F  | @Addr      RELATIVE DEFERRED MODE      |
;1701    ;    |         |       |     |                                        |
;1702    ;    |  1010   | B,D,F | 0-E | @D(Rn)     DISPLACEMENT DEFERRED MODE  |
;1703    ;    |         |       |     |                                        |
;1704    ;    |  1011   | 9     | 0-E | @(Rn)+     AUTO-INCREMENT DEFERRED MODE |
;1705    ;    |         |       |     |                                        |
;1706    ;    |  1100   | 4     |  F  | (Rn)[PC]   INDEX MODE PC               |
;1707    ;    |         |       |     |                                        |
;1708    ;    |  1101   | 4     | 0-E | (Rn)[Rx]   INDEX MODE                  |
;1709    ;    |         |       |     |                                        |
;1710    ;    +---------+-------------+----------------------------------------+
```

```
.TOC "  Micro Level Charts                              Compatibility Mode Addressing Branch"


;     +--------+------------+-------------------------------------+
;     |        |  Operand   |                                     |
;     | Branch |  Specifier |                                     |
;     | Offset +--------+---+          Addressing Mode            |
;     |        |  Mode  |Reg|                                     |
;     +--------+--------+---+-------------------------------------+
;     |        |        |   |                                     |
;     |  0000  |   0    |0-6| Rn            REGISTER MODE         |
;     |        |        |   |                                     |
;     |  0001  |   0    | 7 | PC            REGISTER MODE PC      |
;     |        |        |   |                                     |
;     |  0010  |   1    |0-6| (Rn)          REGISTER DEFERRED MODE |
;     |        |        |   |                                     |
;     |  0011  |   1    | 7 | (PC)          REGISTER DEFERRED MODE PC |
;     |        |        |   |                                     |
;     |  0100  |   2    |0-5| (Rn)+         AUTOINCREMENT MODE    |
;     |        |        |   |                                     |
;     |  0101  |   2    | 6 | (SP)+         AUTOINCREMENT MODE SP |
;     |        |        |   |                                     |
;     |  0110  |   3    |0-6| @(Rn)+        AUTO-INC DEFERRED MODE |
;     |        |        |   |                                     |
;     |  0111  |   3    | 7 | @#Addr        ABSOLUTE MODE         |
;     |        |        |   |                                     |
;     |  1000  |   4    |0-5| -(Rn)         AUTO-DECREMENT MODE   |
;     |        |        |   |                                     |
;     |  1001  |   4    | 6 | -(SP)         AUTO-DECREMENT MODE SP |
;     |        |        |   |                                     |
;     |  1010  |   5    |0-7| @-(Rn)        AUTO-DEC DEFERRED MODE |
;     |        |        |   |                                     |
;     |  1011  |   4    | 7 | -(PC)         AUTO-DECREMENT MODE PC |
;     |        |        |   |                                     |
;     |  1100  |   6    |0-6| X(Rn)         INDEX MODE            |
;     |        |        |   |                                     |
;     |  1101  |   6    | 7 | Addr X(PC)    RELATIVE MODE         |
;     |        |        |   |                                     |
;     |  1110  |   7    |0-7| @ADDR @X(Rn)  INDEX DEFERRED MODE   |
;     |        |        |   |                                     |
;     |  1111  |   2    | 7 | #CONS         IMMEDIATE MODE        |
;     |        |        |   |                                     |
;     +--------+--------+---+-------------------------------------+
```

```
.TOC "  Micro Level Charts                : WBUS Drive Chart"

;                                                    +--------------------------------+
;                                                    |   FIELDS THAT DRIVE WBUS       |
;                                                    +-----------------+--------------+
;                                                    | ALPCTL          |              |
;                                                    | ALU             |              |
;                                                    | ALUOD           |              |
;                                                    | DQ1             |  ALU GROUP   |
;                                                    | DQ2             |              |
;                                                    | DQ3             |              |
;                                                    | LIT             |              |
;                                                    | MUX             |              |
;                                                    +-----------------+--------------+
;                                                    | WCTRL           |              |
;                                                    | CCMISC          | WCTRL GROUP  |
;                                                    | CCPSL           |              |
;                                                    +-----------------+--------------+
;                                                    | FPA             |              |
;                                                    | MSRC            | OTHERS       |
;                                                    +-----------------+--------------+
;
;  +------------------------------------+   +----------------------------------------------+   +------------------------------+
;  |                                    |   |                                              |   |                              |
;  | ALU Group Output Disable Chart     |   |          WCTRL Group Drive Micro Orders      |   | Others Drive Micro Orders    |
;  |  (Always enabled when LONLIT)      |   |                                              |   |                              |
;  +------+------------------+----------+   +-------+--------+--------+--------------------+   +-----+------+-----------------+
;  | ALU  |      MUX=09      | MUX=0D   |   | WCTRL | CCMISC | CCPSL  |                    |   | FPA | MSRC |                 |
;  +------+------------------+----------+   +-------+--------+--------+--------------------+   +-----+------+-----------------+
;  |      |                  |          |   |  03   |        |        | FPTCR              |   | 04  |      | WBUS_FPA        |
;  |  08  | Disable          |          |   |       |        |   04   | WB_PSL.CCBR_SIGND  |   | 05  |      | WBUS_FPA.CC     |
;  |      |                  |          |   |       |   07   |        | WB_ATCR.CCBR_SIGND |   |     |      |                 |
;  |  09  | Disable          | Disable  |   |  0C   |        |        | CM.TP.FPD.F5.STEPC |   |     |  16  | WBUS_RNUM       |
;  |      |                  |          |   |  0D   |        |        | TODCLK             |   |     |  1C  | READRBS         |
;  |  0A  | Disable IF DQ=0  | Disable  |   |  0E   |        |        | INIR               |   |     |  1E  | WB_RBSP         |
;  |      |                  |          |   |  0F   |        |        | TCSR.IICR          |   |     |      |                 |
;  |  0B  | Disable IF DQ=1  | Disable  |   |  11   |        |        | REVLEVEL           |   |     |      |                 |
;  |      |                  |          |   |  1A   |        |        | READCRAR           |   |     |      |                 |
;  |  0C  | Disable          |          |   |  1B   |        |        | READTRAR           |   |     |      |                 |
;  |      |                  |          |   |  1C   |        |        | CM.TP.FPD.FLAGS    |   |     |      |                 |
;  |  0D  | Disable          | Disable  |   |  1D   |        |        | ACLOSYNC           |   |     |      |                 |
;  |      |                  |          |   |  1E   |        |        | CONREAD            |   |     |      |                 |
;  |  0E  | Disable IF DQ=0  |          |   |  1F   |        |        | TU58READ           |   |     |      |                 |
;  |      |                  |          |   |  32   |        |        | MEMSCR             |   |     |      |                 |
;  |  0F  | Disable IF DQ=1  |          |   |  3A   |        |        | ASTLVL             |   |     |      |                 |
;  |      |                  |          |   |  3F   |        |        | IPR                |   |     |      |                 |
;  +------+------------------+----------+   +-------+--------+--------+--------------------+   +-----+------+-----------------+
```

```
.TOC "  Micro Level Charts          : WCTRL/CCPSL vs BUS Micro Order Conflict Chart"

; V000 : (04) WB_PSL.CCBR_SIGND
;        (21) RESERVED
;        (22) VA_VA+4
;        (24) PC_WB
;        (25) VA_WB
;        (2C) PC_PC+WB
;        (31) PREV_WB
;        (34) MEMSCAR_WB
;        (38) ASTLVL_WB
;        (3A) ASTLVL
;        (3C) SOFTIPR_WB
;        (3D) IPL_WB
;        (3F) IPR
;
; V001 : (00) PSL_WB.CCBR_ALUS
;        (20) VA_PC+I+W.PC_PC+I
;        (30) MEMSCR_WB
;        (32) MEMSCR
;        (35) PREV_CUR.ISCUR_WB
;
; V002 : (23) MDR_WB
;        (27) MDR_0
;        (2A) WDR_WB.UR
;        (2F) MDR_OSR.CCBR_BRATST
;        (2E) WDR_WB
;        (2B) MDR_IR
;
;
; V004 : (2D) CLRCH.VA_WB
;
; V005 : (37) REICHK
;
; V006 : (33) GRANT
;
; V007 : (26) MBUS_WDR
;
; V010 : (29) CLRTB.VA_WB
;
; V011 : (28) TB_WB
;
;

; ***  VALID IF BUT.NE.UVCTR
```

	V000	V001	V002	V004	V005	V006	V007	V010	V011
(00) READ.PHY		X	X	X	X	X	X	X	X
(01) PRINIT	X	X	X	X	X	X	X	X	X
(02) READ.NT		X	X	X	X	X	X	X	X
(03) IOINIT	X	X	X	X	X	X	X	X	X
(04) RESERVED		X	X	X	X	X	X	X	X
(05) RESERVED					X	X			
(06) READ.SEC		X	X	X	X	X	X	X	X
(07) NOP					X	X			
(08) WRITE.PHY		X		X	X	X	X	X	X
(09) REICHK	X	X	X	X		X	X	X	X
(0A) WRITE.SEC		X		X	X	X	X	X	X
(0B) WRITE.UL.SEC		X		X	X	X	X	X	X
(0C) WRITE.NT		X		X	X	X	X	X	X
(0D) RESERVED					X	X			
(0E) WRITE.NT.LNG		X		X	X	X	X	X	X
(0F) GRANT	X	X	X	X	X		X	X	X
(10) READ		X	X	X	X	X	X	X	X
(11) READ.LNG		X	X	X	X	X	X	X	X
(12) PRB.WR.PTE			X	X	X	X	X	X	X
(13) READ.MOD.LCK		X	X	X	X	X	X	X	X
(14) READ.MOD		X	X	X	X	X	X	X	X
(15) READ.LNG.MOD		X	X	X	X	X	X	X	X
(16) PRB.RD.PTE			X	X	X	X	X	***	X
(17) PRB.RD.PTE.K			X	X	X	X	X	X	X
(18) WRITE		X		X	X	X	X	X	X
(19) WRITE.LNG		X		X	X	X	X	X	X
(1A) WRITE.NOREG		X		X	X	X	X	X	X
(1B) WRITE.UL		X		X	X	X	X	X	X
(1C) PRB.WR.MODE				X	X	X		X	X
(1D) PRB.WR				X	X	X		X	X
(1E) PRB.RD.MODE				X	X	X		X	X
(1F) PRB.RD				X	X	X		X	***

```
.TOC "   Micro Level Charts                : BUS vs MSRC Micro Order Conflict Chart"

; V020 : (08) WRITE.PHY           +-------------------+----+----+----+----+----+
;        (0A) WRITE.SEC           |                   |    |    |    |    |    |
;        (0B) WRITE.UL.SEC        |                   | V  | V  | V  | V  | V  |
;        (0C) WRITE.NT            |                   | 0  | 0  | 0  | 0  | 0  |
;        (0E) WRITE.NT.LNG        |                   | 2  | 2  | 2  | 2  | 2  |
;        (18) WRITE               |                   | 0  | 1  | 2  | 3  | 4  |
;        (19) WRITE.LNG           |                   |    |    |    |    |    |
;        (1A) WRITE.NOREG         +-------------------+----+----+----+----+----+
;        (1B) WRITE.UL            | (00) TEMP0        |    |    |    |    |    |
;                                 | (01) TEMP1        |    |    |    |    |    |
; V021 : (01) PRINIT              | (02) TEMP2        |    |    |    |    |    |
;        (03) IOINIT              | (03) TEMP3        |    |    |    |    |    |
;        (0F) GRANT               | (04) TEMP4        |    |    |    | X  |    |
;                                 | (05) TEMP5        |    |    |    | X  |    |
; V022 : (00) READ.PHY            | (06) TEMP6        |    |    |    | X  |    |
;        (02) READ.NT             | (07) TEMP7        |    |    |    | X  |    |
;        (06) READ.SEC            | (08) TEMP8        |    |    |    |    |    |
;        (10) READ                | (09) TEMP9        |    |    |    |    |    |
;        (11) READ.LNG            | (0A) TEMP10       |    |    |    |    |    |
;        (13) READ.MOD.LCK        | (0B) TEMP11       |    |    |    |    |    |
;        (14) READ.MOD            | (0C) FPDOFFSET    |    |    |    | X  |    |
;        (15) READ.LNG.MOD        | (0D) MM.TEMP0     |    |    |    | X  |    |
;                                 | (0E) SCBB         |    |    |    | X  |    |
; V023 : (12) PRB.WR.PTE          | (0F) SISR         |    |    |    | X  |    |
;        (16) PRB.RD.PTE          | (10) TEMP.R       |    |    |    |    |    |
;        (17) PRB.RD.PTE.K        | (11) TEMP.R+1     |    |    |    |    |    |
;                                 | (12) MDR          |    | X  |    |    |    |
; V024 : (1C) PRB.WR.MODE         | (13) WDR          |    | X  | X  | X  |    |
;        (1D) PRB.WR              | (14) PSHSUB       |    |    |    | X  |    |
;        (1E) PRB.RD.MODE         | (15) PSHADD       |    |    |    | X  |    |
;        (1F) PRB.RD              | (16) WBUS_RNUM    |    |    |    | X  |    |
;                                 | (17) XB.PC_PC+I   |@@@ | X  |@@@ | X  |    |
;                                 | (18) MA           | X  | X  | X  | X  |*** |
;                                 | (19) PCBACK       | X  | X  | X  | X  |*** |
;                                 | (1A) PC           | X  | X  | X  | X  |*** |
;                                 | (1B) VA           | X  | X  | X  | X  |*** |
;                                 | (1C) READRBS      |    |    |    | X  |    |
;                                 | (1D) PNUM_WBUS    |    |    |    | X  |    |
;                                 | (1E) WB_RBSP      |    |    |    | X  |    |
;                                 | (1F) TB           | X  | X  | X  | X  |*** |
;                                 +-------------------+----+----+----+----+----+
;
; *** Valid if BUT.ne.UVCTR
; @@@ Valid if in INIT code because MM is not set
```

```
.TOC "  Micro Level Charts             : wCTRL/CCPSL vs MSRC Micro Order Conflict Chart"

; V030 : (00) PSL_WB.CCBR_ALUS        +----------------+---+---+---+---+---+---+---+---+---+
;        (04) WB_PSL.CCBR_SIGND       |                |   |   |   |   |   |   |   |   |   |
;        (22) VA_VA+4                 |                | V | V | V | V | V | V | V | V | V |
;        (24) PC_WB                   |                | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
;        (25) VA_WB                   |                | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 |
;        (2C) PC_PC+WB                |                | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 |
;        (31) PREV_WB                 |                |   |   |   |   |   |   |   |   |   |
;        (34) MEMSCAR_WB              +----------------+---+---+---+---+---+---+---+---+---+
;        (35) PREV_CUR.ISCUR_WB       | (00) TEMP0     |   | X |   |   | X |   |   | X |   |
;        (37) REICHK                  | (01) TEMP1     |   | X |   |   | X |   |   | X |   |
;        (38) ASTLVL_WB               | (02) TEMP2     |   | X |   |   | X |   |   | X |   |
;        (3A) ASTLVL                  | (03) TEMP3     |   | X |   |   | X |   |   | X |   |
;        (3C) SOFTIPR_WB              | (04) TEMP4     |   | X |   |   | X |   |   | X |   |
;        (3D) IPL_WB                  | (05) TEMP5     |   | X |   |   | X |   |   | X |   |
;        (3F) IPR                     | (06) TEMP6     |   | X |   |   | X |   |   | X |   |
;                                     | (07) TEMP7     |   | X |   |   | X |   |   | X |   |
; V031 : (20) VA_PC+I+W.PC_PC+I       | (08) TEMP8     |   | X |   |   | X |   |   | X |   |
;                                     | (09) TEMP9     |   | X |   |   | X |   |   | X |   |
; V032 : (21) RESERVED                | (0A) TEMP10    |   | X |   |   | X |   |   | X |   |
;        (30) MEMSCR_WB               | (0B) TEMP11    |   | X |   |   | X |   |   | X |   |
;        (32) MEMSCR                  | (0C) FPDOFFSET |   | X |   |   | X |   |   | X |   |
;                                     | (0D) MM.TEMP0  |   | X |   |   | X |   |   | X |   |
; V033 : (23) MDR_WB                  | (0E) SCBB      |   | X |   |   | X |   |   | X |   |
;        (2A) WDR_WB.UR               | (0F) SISR      |   | X |   |   | X |   |   | X |   |
;        (2F) MDR_OSR.CCBR_BRATST     | (10) TEMP.R    |   | X |   |   | X |   |   | X |   |
;        (2D) CLRCH.VA_WB             | (11) TEMP.R+1  |   | X |   |   | X |   |   | X |   |
;        (2E) WDR_WB                  | (12) MDR       |   | X |   |   | X |   | X | X | X |
;        (2B) MDR_IR                  | (13) WDR       | X | X | X | X | X | X | X |   | X |
;                                     | (14) PSHSUB    |   | X |   |   | X |   |   | X |   |
; V034 : (28) TB_WB                   | (15) PSHADD    |   | X |   |   | X |   |   | X |   |
;                                     | (16) WBUS_RNUM |   | X |   |   | X |   |   | X |   |
; V035 : (29) CLRTB.VA_WB             | (17) XB.PC_PC+I|   |   | X |   | X |   | X | X | X |
;                                     | (18) MA        |   | X |   |   | X |   |   | X | X |
; V036 : (33) GRANT                   | (19) PCBACK    |   | X |   |   | X | X |   | X | X |
;                                     | (1A) PC        |   | X |   |   | X | X |   | X | X |
; V037 : (26) MBUS_WDR                | (1B) VA        |   | X |   |   |   |   |   | X | X |
;                                     | (1C) READRBS   |   | X |   |   | X |   |   | X |   |
; V021 : (27) MDR_0                   | (1D) RNUM_WBUS |   | X |   |   | X |   |   | X |   |
;                                     | (1E) WB_RBSP   |   | X |   |   | X |   |   | X |   |
;                                     | (1F) TB        |   | X |   | X | X |   | X | X | X |
;                                     +----------------+---+---+---+---+---+---+---+---+---+
```

```
.TOC "  Micro Level Charts              : BUT/CCBR Chart"


;         +-------------------------+---------------------------------+-------------------+
;         |                         |                                 |   CCBR CONTROL    |
;         |       MICRO ORDER       |            OPERATION            +---------+---------+
;         |                         |                                 | CCBR<1> | CCBR<0> |
; +-------+-------------------------+---------------------------------+---------+---------+
; |       | NOP.CCBR<-SIGND         |                                 | LSS 0   | EQL 0   |
; |   C   | NOP.CCBR<-ALUS          |                                 | ALUS<1> | ALUS<0> |
; |   C   | CCOP1.CCBR<-SIGND       | CC OP 1                         | LSS 0   | EQL 0   |
; |       | CCOP2.CCBR<-SIGND       | CC OP 2                         | LSS 0   | EQL 0   |
; +-------+-------------------------+---------------------------------+---------+---------+
; |   C   | NOP.CCBR<-BRATST        |                                 | 0       | BRA TST |
; |   C   | NOP.CCBR<-CSIGNS        |                                 | ALUS<1> | LSS 0   |
; |   M   | WB<-ATCR.CCBR<-SIGND    | WB<3:0> <-ATCR                  | LSS 0   | EQL 0   |
; |   I   | ALUS<-DSDZ.CCBR<-ALUS   | ALUS<1:0> <- (BCD SIGN)'(BCD 0) | ALUS<1> | ALUS<0> |
; |   S   | ALUS<-SIGND.CCBR<-ALUS  | ALUS<1:0> <- (LSS 0)'(EQL 0)    | ALUS<1> | ALUS<0> |
; |   C   | ALUS<-UNSGN.CCBR<-ALUS  | ALUS<1:0> <- (LSSU 0)'(EQL 0)   | ALUS<1> | ALUS<0> |
; |       | SETV.CCBR<-SIGND        | PSL<V> <- 1                     | LSS 0   | EQL 0   |
; +-------+-------------------------+---------------------------------+---------+---------+
; |   C   | WB<-PSL.CCBR<-SIGND     | WB<31:0> <-PSL                  | LSS 0   | EQL 0   |
; |   C   | CC<-WB.CCBR<-ALUS       | CC<-WB<3:0>                     | ALUS<1> | ALUS<0> |
; |   P   | PSL<-WB.CCBR<-ALUS      | PSL<-WB<31:0>   ***             | ALUS<1> | ALUS<0> |
; |   S   | PSW<-WB.CCBR<-ALUS      | PSW<-WB<15:0>   ***             | ALUS<1> | ALUS<0> |
; |   L   | MDR_OSR.CCBR_BRATST     | MDR <- ZEXT OSR                 | 0       | BRA TST |
; +-------+-------------------------+---------------------------------+---------+---------+
;
;   WHENEVER THE OPERATION ALTERS ALUS, CCBR <- ALUS OLD
;
;   UNLESS OTHERWISE NOTED ALL VALUES ARE SIZE DEPENDENT
;
;       LSS 0    : WBUS<SIGN>.XOR.ALU 'OVERFLOW' (THIS WILL PRODUCE 'LSS 0' FOR A-B OR A+(-B))
;
;       OVERFLOW : XOR OF CIN AND COUT OF MSB(SIZE) ALU
;
;       EQL 0    : WMUX = 0
;
;       LSSU 0   : .NOT.(ALU CARRY) (THIS WILL PRODUCE 'LSSU 0' FOR A-B OR A+(-B))
;
;       BCD 0    : (WMUX<31:8>.EQ.0).AND.(WBUS<7:4>.EQ.0)
;
;       BCD SIGN : IF WBUS<3:0> > 9 THEN, ALUS<1> IS SET IF SIGN IS NEGATIVE
;
;       BRA TST  : FOR BRANCH INSTRUCTIONS CCBR<0> IS SET IF THE BRANCH CONDITION IS TRUE

;       *** IF V GETS SET NO TRAP WILL OCCUR
```

```
.TOC "  Micro Level Charts              : BUT/SPASTA Chart"

; +-----------------------------------------------------------------------+-----------------------------------------------------------------------+
; |                                                                       |                                                                       |
; |                      IPR DECODE & RBS STATUS                          |      COMPATIBILITY REG DECODE & NATIVE REG=SP TEST                    |
; |                                                                       |                                                                       |
; +-----------------------------------------------------------------------+-----------------------------------------------------------------------+
; |                                                                       |                                                                       |
; | RSRC does not Specify GPR[0,15]                                       | RSRC Specifies GPR[0,15]                                              |
; |                                                                       |                                                                       |
; |  RSRC : TEMP0              KSP                TEMP.R                  |  RSRC : R0                  GPR.R                                     |
; |         TEMP1              ESP                TEMP.R+1                |         R1                  GPR.R+1                                   |
; |         TEMP2              SSP                TEMP.ROR1               |         R2                  GPR.ROR1                                  |
; |         TEMP3              USP                IPR.R                   |         R3                  DST.R                                     |
; |         TEMP4              ISP                IPR.R+1                 |         R4                  DST.R+1                                   |
; |         TEMP5              PCBB               IPR.ROR1                |         R5                  DST.ROR1                                  |
; |         TEMP6              MM.TEMP2           LONLIT                  |         R6                                                            |
; |         TEMP7              MM.TEMP3           ZERO                    |         R7                                                            |
; |         TEMP8              P0BR               ZERO.CLRRBSP            |         R8                                                            |
; |         TEMP9              P0LR                                       |         R9                                                            |
; |         TEMP10             P1BR                                       |         R10                                                           |
; |         TEMP11             P1LR                                       |         R11                                                           |
; |         TEMP12             SBR                                        |         R12                                                           |
; |         TEMP13             SLR                                        |         R13                                                           |
; |         MM.TEMP5           SPNICR.SPICR                               |         SP                                                            |
; |         MM.TEMP1           MM.TEMP4                                   |         RTMPGPR                                                       |
; |                                                                       |                                                                       |
; +------------+-----------------------------------+----------------------+------------+-----------------------------------+----------------------+
; |            |                                   |                      |            |                                   |                      |
; | MSRC       |                                   | SPASTA<1:0>          | MSRC       |                                   | SPASTA<1:0>          |
; |            |                                   |                      |            |                                   |                      |
; +------------+-----------------------------------+----------------------+------------+-----------------------------------+----------------------+
; |            |                                   |                      |            |                                   |                      |
; | RNUM_WBUS  | WBUS.EQ.[8,9,10,11,12,13]         |     00               | RNUM_WBUS  |                                   | UNDEFINED            |
; |            |                                   |                      |            |                                   |                      |
; | RNUM_WBUS  | WBUS.EQ.[5,6,7,14,15]             |     10               | READRBS    |                                   | UNDEFINED            |
; |            |                                   |                      |            |                                   |                      |
; | RNUM_WBUS  | WBUS.EQ.[0,1,2,3,4]               |     11               | WB_RBSP    |                                   | UNDEFINED            |
; |            |                                   |                      |            |                                   |                      |
; | READRBS    | RBS<6>.EQ.1(PSHADD)               |     10               | ALL OTHERS | RNUM.NE.[6,7,14]                  |     00               |
; |            |                                   |                      |            |                                   |                      |
; | READRBS    | RBS<6>.EQ.0(PSHSUB)               |     00               | ALL OTHERS | RNUM.EQ.14                        |     01               |
; |            |                                   |                      |            |                                   |                      |
; | WB_RBSP    | RBSP.EQ.0                         |     01               | ALL OTHERS | RNUM.EQ.7                         |     10               |
; |            |                                   |                      |            |                                   |                      |
; | WB_RBSP    | RBSP.NE.0                         |     00               | ALL OTHERS | RNUM.EQ.6                         |     11               |
; |            |                                   |                      |            |                                   |                      |
; | ALL OTHERS |                                   |     00               |            |                                   |                      |
; |            |                                   |                      |            |                                   |                      |
; +------------+-----------------------------------+----------------------+------------+-----------------------------------+----------------------+
```

```
;2032   .TOC "  Micro Level Charts            : TABLE 1 - ALU & Q Rotate and Shift Functions Part 1"
;2033
;2034
;2035   ; ALU  Q  |                ROTATE                     |                 SHIFT                      |
;2036   ;         |                                           |                                            |
;2037   ;         |      +-----------+      +-----------+     |      +-----------+      +-----------+      |
;2038   ;  L    L |  ---|  <-ALU    |<-----|   <-Q     |<--   |  <--|  <-ALU    |<-----|   <-Q     |<--0   |
;2039   ;         | |    +-----------+      +-----------+   | |      +-----------+      +-----------+      |
;2040   ;         | |                                       | |                                            |
;2041   ;         |  ---------------------------------------  |                                            |
;2042   ;         |                                           |                                            |
;2043   ;         |                                           |                                            |
;2044   ;         |                                           |                                            |
;2045   ;         |                                           |                                            |
;2046   ;         |                                           |                                            |
;2047   ;         |                                           |                                            |
;2048   ;         |           +-----------+                   |           +-----------+                    |
;2049   ;         |        ---|  <-ALU    |<--                |        <--|  <-ALU    |<--                 |
;2050   ;         |       |   +-----------+   |               |           +-----------+   |                |
;2051   ;  L    R |       |                   |               |                           |                |
;2052   ;         |       |   +-----------+   |               |           +-----------+   |                |
;2053   ;         |        -->|    Q->    |---                |       0-->|    Q->    |---                 |
;2054   ;         |           +-----------+                   |           +-----------+                    |
;2055   ;         |                                           |                                            |
;2056   ;         |                                           |                                            |
;2057   ;         |                                           |                                            |
;2058   ;         |                                           |                                            |
;2059   ;         |                                           |                                            |
;2060   ;         |                                           |                                            |
;2061   ;         |                                           |                                            |
;2062   ;         |           +-----------+                   |           +-----------+                    |
;2063   ;         |        -->|   ALU->   |---                |        -->|   ALU->   |-->                 |
;2064   ;         |       |   +-----------+   |               |       |   +-----------+                    |
;2065   ;  R    L |       |                   |               |       |                                    |
;2066   ;         |       |   +-----------+   |               |       |   +-----------+                    |
;2067   ;         |        ---|   <-Q     |<--                |        ---|   <-Q     |<--0                |
;2068   ;         |           +-----------+                   |           +-----------+                    |
;2069   ;         |                                           |                                            |
;2070   ;         |                                           |                                            |
;2071   ;         |                                           |                                            |
;2072   ;         |                                           |                                            |
;2073   ;         |                                           |                                            |
;2074   ;         |                                           |                                            |
;2075   ;         |                                           |                                            |
;2076   ;         |      +-----------+      +-----------+     |      +-----------+      +-----------+      |
;2077   ;  R    R |  -->|   ALU->   |----->|    Q->    |---   |  0-->|   ALU->   |----->|    Q->    |-->    |
;2078   ;         | |    +-----------+      +-----------+   | |      +-----------+      +-----------+      |
;2079   ;         | |                                       | |                                            |
;2080   ;         |  ---------------------------------------  |                                            |
```

```
;2081   .TOC "  Micro Level Charts          : TABLE 2 - ALU & Q Rotate and Shift Functions Part 2"
;2082
;2083
;2084   ; ALU  Q  |                   ROTATE                     |                  SHIFT                     |
;2085   ;         |                                              |                                            |
;2086   ;         |     +-----------+     +-----------+          |     +-----------+     +-----------+        |
;2087   ;  N   L  |     |    ALU    |  ---|    <-Q    |<--       |     |    ALU    |  <--|    <-Q    |<--     |
;2088   ;         |     +-----------+  |  +-----------+  |       |     +-----------+     +-----------+  |     |
;2089   ;         |                    |                 |       |                                      |     |
;2090   ;         |                    ------------------        |                         WBUS<31>---        |
;2091   ;         |                                              |                                            |
;2092   ;         |                                              |                                            |
;2093   ;         |                                              |                                            |
;2094   ;         |                                              |                                            |
;2095   ;         |                                              |                                            |
;2096   ;         |                                              |                                            |
;2097   ;         |     +-----------+     +-----------+          |     +-----------+     +-----------+        |
;2098   ;  N   R  |     |    ALU    |  -->|    Q->    |---       |     |    ALU    |  -->|    Q->    |-->     |
;2099   ;         |     +-----------+  |  +-----------+  |       |     +-----------+  |  +-----------+        |
;2100   ;         |                    |                 |       |                    |                       |
;2101   ;         |                    ------------------        |                    ---WBUS<31>             |
;2102   ;         |                                              |                                            |
;2103   ;         |                                              |                                            |
;2104   ;         |                                              |                                            |
;2105   ;         |                                              |                                            |
;2106   ;         |                                              |                                            |
;2107   ;         |                                              |                                            |
;2108   ;         |     +-----------+     +-----------+          |     +-----------+     +-----------+        |
;2109   ;  L   N  |  ---|   <-ALU   |<--  |     Q     |          |  <--|   <-ALU   |<--  |     Q     |        |
;2110   ;         |  |  +-----------+  |  +-----------+          |     +-----------+  |  +-----------+        |
;2111   ;         |  |                 |                         |                    |                       |
;2112   ;         |  ------------------                          |  Q<31>(SEE NOTE)>---                       |
;2113   ;         |                                              |                                            |
;2114   ;         |                                              |                                            |
;2115   ;         |                                              |                                            |
;2116   ;         |                                              |                                            |
;2117   ;         |                                              |                                            |
;2118   ;         |                                              |                                            |
;2119   ;         |     +-----------+     +-----------+          |     +-----------+     +-----------+        |
;2120   ;  R   N  |  -->|   ALU->   |---  |     Q     |          |  -->|   ALU->   |-->  |     Q     |        |
;2121   ;         |  |  +-----------+  |  +-----------+          |  |  +-----------+     +-----------+        |
;2122   ;         |  |                 |                         |  |                                         |
;2123   ;         |  ------------------                          |  ---Q<31>(SEE NOTE)                        |
;2124   ;
;2125   ;       NOTE: Q<31> IS UNDEFINED FOR ANY LOAD Q FUNCTION
```

```
.TOC "  Micro Level Charts              : TABLE 2 - Console Interface Control"


;       CRAR    : Console 'Register Address' Register
;
;       XBUF    : Xmitter data BUFfer
;       RBUF    : Reciever data BUFfer
;       XCSR    : Xmitter  Command and Status Register
;       RCSR    : Reciever Command and Status Register
;       CSR     : Command and Status Register
;
;
;
;                  +----------------------+----------------------+----------------------+----------------------+
;                  |                      |                      |                      |                      |
;                  |        CRAR=0        |        CRAR=1        |        CRAR=2        |        CRAR=3        |
;                  |        (DATA)        |    (XMIT STATUS)     |    (RECV STATUS)     | (COMMAND & STATUS)   |
;                  |                      |                      |                      |                      |
;  +---------------+----------------------+----------------------+----------------------+----------------------+
;  |               |                                                                                           |
;  | WCTRL/LOADCRAR|                                     CRAR <- WBUS<23:22>                                   |
;  |               |                                                                                           |
;  +---------------+----------------------+----------------------+----------------------+----------------------+
;  |               |                      |                      |                      | CSR <- WBUS<23:20>   |
;  | WCTRL/CONWRITE| XBUF <- WBUS<23:16>  | XCSR<6> <- WBUS<22>  | RCSR<6> <- WBUS<22>  |  23=HALT PENDING     |
;  |               |   (CLEARS READY)     |  INT ENABLE          |  INT ENABLE          |  22=CLEAR HALT       |
;  |               |                      |                      |                      |  21=CLR XMIT INT REQ |
;  |               |                      |                      |                      |  20=SET HALT         |
;  +---------------+----------------------+----------------------+----------------------+----------------------+
;  |               |                      |                      |                      |                      |
;  | WCTRL/CONREAD | WBUS<23:16> <- RBUF  | WBUS<23:22> <- XCSR  | WBUS<23:22> <- RCSR  | WBUS<23:21> <- CSR   |
;  |               | (CLEARS DONE AND     |  23=READY            |  23=DONE             |  23=HALT PENDING     |
;  |               |  RECEIVER INT REQ)   |  22=INT ENABLE       |  22=INT ENABLE       |  22=HALT             |
;  |               |                      |                      |                      |  21=0                |
;  +---------------+----------------------+----------------------+----------------------+----------------------+
;  |               |                                                                                           |
;  | WCTRL/READCRAR|                                     WBUS<25:24> <- CRAR                                   |
;  |               |                                                                                           |
;  +---------------+-------------------------------------------------------------------------------------------+
```

```
	.TOC "	Micro Level Charts		: TABLE 3 - TU58 Interface Control"


;	TRAR	: TU58 'Register Address' Register
;
;	XBUF	: Xmitter data BUFfer
;	RBUF	: Reciever data BUFfer
;	XCSR	: Xmitter  Command and Status Register
;	RCSR	: Reciever Command and Status Register
;	CSR	: Command and Status Register
;
;
;
;				+--------------------+--------------------+--------------------+--------------------+
;				|                    |                    |                    |                    |
;				|       TRAR=0       |       TRAR=1       |       TRAR=2       |       TRAR=3       |
;				|       (DATA)       |    (XMIT STATUS)   |    (RECV STATUS)   | (COMMAND & STATUS) |
;				|                    |                    |                    |                    |
;		+-----------------+--------------------+--------------------+--------------------+--------------------+
;		|                 |                                                                                   |
;		| WCTRL/LOADTRAR  |                             TRAR <- WBUS<23:22>                                   |
;		|                 |                                                                                   |
;		+-----------------+--------------------+--------------------+--------------------+--------------------+
;		|                 |                    |                    |                    |                    |
;		| WCTRL/TU58WRITE | XBUF <- WBUS<23:16> | XCSR<6> <- WBUS<22> | RCSR<6> <- WBUS<22> | CSR <- WBUS<23:20> |
;		|                 |   (CLEARS READY)   |  INT ENABLE        |  INT ENABLE        | 23=COLD START BIT  |
;		|                 |                    |                    |                    | 22=CLEAR START BREAK|
;		|                 |                    |                    |                    | 21=CLR XMIT INT REQ |
;		|                 |                    |                    |                    | 20=SET START BREAK |
;		+-----------------+--------------------+--------------------+--------------------+--------------------+
;		|                 |                    |                    |                    |                    |
;		| WCTRL/TU58READ  | WBUS<23:16> <- RBUF | WBUS<23:22> <- XCSR | WBUS<23:22> <- RCSR | WBUS<23> <-        |
;		|                 | (CLEARS DONE AND   |  23=READY          |  23=DONE           |     COLD START BIT |
;		|                 |  RECEIVER INT REQ) |  22=INT ENABLE     |  22=INT ENABLE     | WBUS<21> <-        |
;		|                 |                    |                    |                    |     XMIT INT REQ   |
;		+-----------------+--------------------+--------------------+--------------------+--------------------+
;		|                 |                                                                                   |
;		| WCTRL/READTRAR  |                             WBUS<25:24> <- TRAR                                   |
;		|                 |                                                                                   |
;		+-----------------+-----------------------------------------------------------------------------------+
```

```
;2205   .TOC "  Micro Level Charts            : ROM layout map for control store"
;2206   ;                                                     +-------+-------+---+
;2207   ;                                                     |       |       |   |
;2208   ;                                                     |  MUX  |  ALU  |DQ1|
;2209   ;                                                     |       |       |   |
;2210   ;         +-+-+---+-------+---+-------+-------+-------+-------+-------+---+
;2211   ;         | | |   |       |   |       |       |       |               |       | D |
;2212   ;         |P|P|   |       |   |       |       |       |               |       | T |
;2213   ;         |A|A|LIT|  MISC |SPW| MSRC  |  ROT  |       ALPCTL          |  BUT  | Y |
;2214   ;         |R|R|   |       |   |       |       |                       |       | P |
;2215   ;         |2|1|   |       |   |       |       |                       |       | E |
;2216   ;         +-+-+---+-------+---+-------+-------+-------+-------+-------+-------+---+
;2217   ;         |7 7 7 7|7 7 7 7|7 7 6 6|6 6 6 6|6 6 6 6|5 5 5 5|5 5 5 5|5 5 4 4|4 4 4 4|4 4 4 4|
;2218   ;         |9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|
;2219   ; +---------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
;2220   ; | 0000-03FF |  146  |  140  |  134  |  128  |  122  |  116  |  110  |  104  |   98  |   92  |
;2221   ; |           |       |       |       |       |       |       |       |       |       |       |
;2222   ; | 0400-07FF |  145  |  139  |  133  |  127  |  121  |  115  |  109  |  103  |   97  |   91  |
;2223   ; |           |       |       |       |       |       |       |       |       |       |       |
;2224   ; | 0800-0BFF |  144  |  138  |  132  |  126  |  120  |  114  |  108  |  102  |   96  |   90  |
;2225   ; |           |       |       |       |       |       |       |       |       |       |       |
;2226   ; | 0C00-0FFF |  143  |  137  |  131  |  125  |  119  |  113  |  107  |  101  |   95  |   89  |
;2227   ; |           |       |       |       |       |       |       |       |       |       |       |
;2228   ; | 1000-13FF |  142  |  136  |  130  |  124  |  118  |  112  |  106  |  100  |   94  |   88  |
;2229   ; |           |       |       |       |       |       |       |       |       |       |       |
;2230   ; | 1400-17FF |  141  |  135  |  129  |  123  |  117  |  111  |  105  |   99  |   93  |   87  |
;2231   ; +---------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
;2232   ;
;2233   ;         +---------------+
;2234   ;         |               |      * note :
;2235   ;         |     LITRL     |      listing is inverted with respect to acutal prom pattern
;2236   ;         |               |
;2237   ;         +-------+-+-+---+-----------+-------+-------+-+-+-----------------------+
;2238   ;         |       |I|     |           |       |       |C|J|                       |
;2239   ;         |       |S|     |           |       |       |L|S|                       |
;2240   ;         |  RSRC |T| C   |   WCTRL   |  BUS  |  FPA  |K|R|         NEXT          |
;2241   ;         |       |R| C   |           |       |       |X| |<- inverted ->|*       |
;2242   ;         |       |M|     |           |       |       | | |                       |
;2243   ;         +-------+-+-+---+-----------+-------+-------+-+-+-----------------------+
;2244   ;         |3 3 3 3|3 3 3 3|3 3 2 2|2 2 2 2|2 2 2 2|1 1 1 1|1 1 1 1|1 1 0 0|0 0 0 0|0 0 0 0|
;2245   ;         |9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|9 8 7 6|5 4 3 2|1 0 9 8|7 6 5 4|3 2 1 0|
;2246   ; +---------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
;2247   ; | 0000-03FF |   86  |   80  |   74  |   68  |   62  |   56  |   50  |   44  |   38  |   32  |
;2248   ; |           |       |       |       |       |       |       |       |       |       |       |
;2249   ; | 0400-07FF |   85  |   79  |   73  |   67  |   61  |   55  |   49  |   43  |   37  |   31  |
;2250   ; |           |       |       |       |       |       |       |       |       |       |       |
;2251   ; | 0800-0BFF |   84  |   78  |   72  |   66  |   60  |   54  |   48  |   42  |   36  |   30  |
;2252   ; |           |       |       |       |       |       |       |       |       |       |       |
;2253   ; | 0C00-0FFF |   83  |   77  |   71  |   65  |   59  |   53  |   47  |   41  |   35  |   29  |
;2254   ; |           |       |       |       |       |       |       |       |       |       |       |
;2255   ; | 1000-13FF |   82  |   76  |   70  |   64  |   58  |   52  |   46  |   40  |   34  |   28  |
;2256   ; |           |       |       |       |       |       |       |       |       |       |       |
;2257   ; | 1400-17FF |   81  |   75  |   69  |   63  |   57  |   51  |   45  |   39  |   33  |   27  |
;2258   ; +---------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
                          ;2259    .BIN
```

```
;     CMT094.MCR              MICRO2  1M(01)      1-SEP-82  13:33:09     CLOKX Rev  9.01, Clock rate = 160ns
;      MACRO.MIC              MACRO.MIC

                              .TOC "MACRO.MIC"
                              .TOC "REVISION 58.0"
                              ;     Gerard Koeckhoven, Brian Allison, C. E. MCDOWELL,  P. R. GUILBAULT

.NOBIN
.TOC "  Revision History"

; 58    Add Macro for interval timer
; 57    Add Macros to support CMT053 (GH)
; 56    Add Macro to support CMT052
; 55    ADD MACROS TO SUPPORT CMT049
; 54    Initial release.
```

```
.TOC "  Basic Macros"

CCOP1                          "CC/CCOP1.CCBR_SIGND"
CCOP2                          "CC/CCOP2.CCBR_SIGND"
CLEAR ADD1(FLAG0)              "MISC/CLR.FLAG0"
CLEAR ADD2(FLAG1)              "MISC/CLR.FLAG1"
CLEAR ARITH TRAPS              "CCMISC/WB_ATCR.CCBR_SIGND"
CLEAR BOOT(FLAG MMNOINT)       "MISC/CLR.MMNOINT"
CLEAR FLAG0                    "MISC/CLR.FLAG0"
CLEAR FLAG1                    "MISC/CLR.FLAG1"
CLEAR FLAG2                    "MISC/CLR.FLAG2"
CLEAR FLAG3                    "MISC/CLR.FLAG3"
CLEAR FLAG4                    "MISC/CLR.MMNOINT"
CLEAR FP TRAPS                 "WCTRL/FPTCR"
CLEAR FPA(FLAG0)               "MISC/CLR.FLAG0"
CLEAR FPD                      "MISC/CLR.FPD"
CLEAR GFLOAT(FLAG4)            "MISC/CLR.MMNOINT"
CLEAR MM.NOINT                 "MISC/CLR.MMNOINT"
CLEAR MOPZERO(FLAG1)           "MISC/CLR.FLAG1"
CLEAR MUL1(FLAG2)              "MISC/CLR.FLAG2"
CLEAR MUL2(FLAG3)              "MISC/CLR.FLAG3"
CLEAR OPZERO(FLAG3)            "MISC/CLR.FLAG3"
CLEAR OVER(FLAG2)              "MISC/CLR.FLAG2"
CLEAR POP1C(FLAG4)             "MISC/CLR.MMNOINT"
CLEAR READ(FLAG1)              "MISC/CLR.FLAG1"
CLEAR REGINT(FLAG1)            "MISC/CLR.FLAG1"
CLEAR SAMESIGN(FLAG4)          "MISC/CLR.MMNOINT"
CLEAR STACK FLAG               "MISC/CLR.STACKFLG"
CLEAR SUB(FLAG1)               "MISC/CLR.FLAG1"
CLEAR TP                       "MISC/CLR.TP"
CLEAR WRITE(FLAG1)             "MISC/CLR.FLAG1"
CLOBBER MTEMPO                 "MSRC/TEMPO,SPW/MLONG"
CLOBBER MTEMPO DEF             "SPW/MLONG"

DEC STEPC                      "MISC/DEC.SC"
DIVDA SOR IN R[]               "ALPCTL/DIVDA,RSRC/@1,ROT/0"
DIVDS SOR IN R[]               "ALPCTL/DIVDS,RSRC/@1,ROT/0"
DIVFAST+ SOR IN R[]            "ALPCTL/DIVFAST+,RSRC/@1,ROT/0"
DIVFAST- SOR IN R[]            "ALPCTL/DIVFAST-,RSRC/@1,ROT/0"

FLUSH XB                       "WCTRL/PC_WB,WB_M[PC]"
FPAWAIT                        "LIT/FPAWAIT"
FORCE 32 BITS OF VA            "BUS/PRB.RD,VSIZE/1"
FORCE CACHE PARITY             "MISC/FORCE.CACHE,VSIZE/1"

IO RESET                       "BUS/IOINIT"
IRD1                           "BUT/IRD1,NEXT/3F9"      ; 3F9 = IE.IRD1.ERROR
IRD1TEST                       "BUT/IRD1TST"
IRDX []                        "BUT/IRDX,NEXT/@1"
ISIZE[]                        "ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/@1"

MULFAST+ CAND IN R[]           "ALPCTL/MULFAST+,RSRC/@1,ROT/0"
MULFAST- CAND IN R[]           "ALPCTL/MULFAST-,RSRC/@1,ROT/0"

NOP                            "ALPCTL/NOP"
```

```
;3682
;3683   PUSH                                "JSR/PUSH"
;3684   PUSH RBS+                           "MSRC/PSHADD"
;3685   PUSH RBS-                           "MSRC/PSHSUB"
;3686
;3687   PROCESS INIT                        "BUS/PRINIT"
;3688
;3689   RESTORE POWER FAIL INT              "WCTRL/NOP"
;3690   RETURN []                           "BUT/RETURN,NEXT/@1"
;3691   RETURN AND INHIBIT DESTINATIONS     "BUT/RET.DINH"
;3692   RETURN AND SUPPRESS BUS CYCLE       "BUT/RETURN,NEXT/0,MISC/RSBC"
;3693
;3694   SET ADD1(FLAG0)                     "MISC/SET.FLAG0"
;3695   SET ADD2(FLAG1)                     "MISC/SET.FLAG1"
;3696   SET BOOT(FLAG MMNOINT)              "MISC/SET.MMNOINT"
;3697   SET FLAG0                           "MISC/SET.FLAG0"
;3698   SET FLAG1                           "MISC/SET.FLAG1"
;3699   SET FLAG2                           "MISC/SET.FLAG2"
;3700   SET FLAG3                           "MISC/SET.FLAG3"
;3701   SET FLAG4                           "MISC/SET.MMNOINT"
;3702   SET FPA(FLAG0)                      "MISC/SET.FLAG0"
;3703   SET FPD                             "MISC/SET.FPD"
;3704   SET GFLOAT(FLAG4)                   "MISC/SET.MMNOINT"
;3705   SET MM.NOINT                        "MISC/SET.MMNOINT"
;3706   SET MOPZERO(FLAG1)                  "MISC/SET.FLAG1"
;3707   SET MUL1(FLAG2)                     "MISC/SET.FLAG2"
;3708   SET MUL2(FLAG3)                     "MISC/SET.FLAG3"
;3709   SET OPZERO(FLAG3)                   "MISC/SET.FLAG3"
;3710   SET OVER(FLAG2)                     "MISC/SET.FLAG2"
;3711   SET POP1C(FLAG4)                    "MISC/SET.MMNOINT"
;3712   SET READ(FLAG1)                     "MISC/SET.FLAG1"
;3713   SET REGINT(FLAG1)                   "MISC/SET.FLAG1"
;3714   SET SAMESIGN(FLAG4)                 "MISC/SET.MMNOINT"
;3715   SET SIGN(FLAG0)                     "MISC/SET.FLAG0"
;3716   SET STACK FLAG                      "MISC/SET.STACKFLG"
;3717   SET SUB(FLAG1)                      "MISC/SET.FLAG1"
;3718   SET UNDER(FLAG3)                    "MISC/SET.FLAG3"
;3719   SET V                               "CCMISC/SETV.CCBR_SIGND"
;3720   SET WRITE(FLAG1)                    "MISC/SET.FLAG1"
;3721   SIZE[]                              "VSIZE/1,DTYPE/@1"
```

```
.TOC "  Bus Function Macros"

COMPLETE CPU BUS CYCLES          "BUS/PRB.RD.PTE.K"

READ                             "BUS/READ"
READ.LONG                        "BUS/READ.LNG"
READ.LONG.MOD                    "BUS/READ.LNG.MOD"
READ.MOD                         "BUS/READ.MOD"
READ.MOD.LOCK                    "BUS/READ.MOD.LCK"
READ.NOTRAP                      "BUS/READ.NT"
READ.PHY                         "BUS/READ.PHY"
READ.SECOND                      "BUS/READ.SEC"

WRITE                            "BUS/WRITE,WCTRL/WDR_WB"
WRITE (M[] R[]).RR.4             "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/@2,ALPCTL/WX_S,ROT/RR.MR.4"
WRITE -M[]                       "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/ZERO,ALU/B-A-CI,ALUCI/ZERO,MUX/M.R1"
WRITE -Q                         "BUS/WRITE,WCTRL/WDR_WB,MUX/R.Q,RSRC/ZERO,ALU/A-B-CI,ALUCI/ZERO"
WRITE CVTNP(M[])                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,ALU/A+B+CI.BCD,MUX/R.S,RSRC/ZERO,ROT/CVTNP"
WRITE CVTPN(M[])                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/TEMPO,ALPCTL/WX_S,ROT/CVTPN"
WRITE D+R[]+ALKC                 "BUS/WRITE,WCTRL/WDR_WB,RSRC/@1,MUX/D.R1,ALU/A+B+CI,ALUCI/ALKC"
WRITE D.OR.ZLIT28[]              "BUS/WRITE,WCTRL/WDR_WB,MUX/D.S,ROT/ZLIT28,LIT/LITRL,LITRL/@1,ALU/OR"
WRITE M[]                        "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,ALU/OR,MUX/M.S,ROT/ZERO"
WRITE M[]+PSLC                   "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A+B+CI,ALUCI/PSLC"
WRITE M[]+Q                      "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/A+B+CI"
WRITE M[]+Q+PSLC                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/A+B+CI,ALUCI/PSLC"
WRITE M[]-PSLC                   "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A-B-CI,ALUCI/PSLC"
WRITE M[]-Q                      "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/A-B-CI"
WRITE M[]-Q-PSLC                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/A-B-CI,ALUCI/PSLC"
WRITE M[].AND.ZLITO[]            "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/AND"
WRITE M[].ANDNOT.Q               "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/ANDNOT"
WRITE M[].ANDNOT.R[]             "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/@2,ALU/ANDNOT,MUX/M.R1"
WRITE M[].ANDNOT.ZLIT8[]         "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/M.S,ALU/ANDNOT"
WRITE M[].OR.Q                   "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/OR"
WRITE M[].OR.R[]                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,RSRC/@2,ALU/OR,MUX/M.R1"
WRITE M[].OR.ZLITO[]             "BUS/WRITE,WCTRL/WDR_WB,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
WRITE M[].OR.ZLIT28[]            "BUS/WRITE,WCTRL/WDR_WB,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT28,LIT/LITRL,LITRL/@2"
WRITE M[].RR.P                   "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,ROT/RR.MM.P,ALPCTL/WX_S"
WRITE M[].SL.1                   "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,ROT/ZERO,MUX/M.S,ALU/A+B+CI.SL"
WRITE M[].XOR.Q                  "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/XOR"
WRITE M[].XZ                     "BUS/WRITE,WCTRL/WDR_WB,MSRC/@1,ALPCTL/WX_S,ROT/XZ.MM"
WRITE NOTREG                     "BUS/WRITE.NOREG,WCTRL/WDR_WB"
WRITE Q                          "BUS/WRITE,WCTRL/WDR_WB,RSRC/ZERO,MUX/R.Q,ALU/OR"
WRITE Q.NOT                      "BUS/WRITE,WCTRL/WDR_WB,RSRC/ZERO,MUX/R.Q,ALU/A-B-CI,ALUCI/ONE"
WRITE Q_(Q.SL.1).OR.1            "BUS/WRITE,WCTRL/WDR_WB,DQ1/Q_WX,ALU/A+B+CI.SL,MUX/R.Q,RSRC/ZERO,ALUSHF/ONE"
WRITE R[]                        "BUS/WRITE,WCTRL/WDR_WB,RSRC/@1,ALU/OR,MUX/R.S,ROT/ZERO"
WRITE R[]+CONX(4)                "BUS/WRITE,WCTRL/WDR_WB,RSRC/@1,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
WRITE R[]-D-ALKC                 "BUS/WRITE,WCTRL/WDR_WB,RSRC/@1,MUX/D.R1,ALU/B-A-CI,ALUCI/ALKC"
WRITE R[]-M[]                    "BUS/WRITE,WCTRL/WDR_WB,MSRC/@2,RSRC/@1,ALU/B-A-CI,MUX/M.R1"
WRITE R[]-M[]-1                  "BUS/WRITE,WCTRL/WDR_WB,MSRC/@2,RSRC/@1,ALU/B-A-CI,MUX/M.R1,ALUCI/ONE"
WRITE XB PC_PC+1                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/XB.PC_PC+I,ROT/ZERO,ALU/OR,MUX/M.S,
                                         ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
WRITE XB PC_PC+4                 "BUS/WRITE,WCTRL/WDR_WB,MSRC/XB.PC_PC+I,ROT/ZERO,ALU/OR,MUX/M.S,
                                         ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/LONG"
WRITE ZLITO[]                    "BUS/WRITE,WCTRL/WDR_WB,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
WRITE.LONG                       "BUS/WRITE.LNG,WCTRL/WDR_WB.UR"
```

MACROS (CONT)

```
;3777   WRITE.LONG D                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/ZERO,MUX/D.R1,ALU/OR"
;3778   WRITE.LONG M[].ANDNOT.Q      "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,MSRC/@1,MUX/M.Q1,ALU/ANDNOT"
;3779   WRITE.LONG M[].OR.Q          "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,MSRC/@1,MUX/M.Q1,ALU/OR"
;3780   WRITE.LONG.NOTRAP            "BUS/WRITE.NT.LNG"
;3781   WRITE.NOTRAP                 "BUS/WRITE.NT"
;3782   WRITE.PHY                    "BUS/WRITE.PHY"
;3783   WRITE.PHY M[]                "BUS/WRITE.PHY,WCTRL/WDR_WB,MSRC/@1,ALU/OR,MUX/M.S,ROT/ZERO"
;3784   WRITE.PHY R[]                "BUS/WRITE.PHY,WCTRL/WDR_WB,RSRC/@1,ALU/OR,MUX/R.S,ROT/ZERO"
;3785   WRITE.SECOND                 "BUS/WRITE.SEC"
;3786   WRITE.SECOND.UL              "BUS/WRITE.UL.SEC"
;3787   WRITE.UL M[]                 "BUS/WRITE.UL,WCTRL/WDR_WB,MSRC/@1,ALU/OR,MUX/M.S,ROT/ZERO"
;3788   WRITE.UL M[]+Q               "BUS/WRITE.UL,WCTRL/WDR_WB,MSRC/@1,MUX/M.Q1,ALU/A+B+CI"
;3789   WRITE.UL M[].ANDNOT.Q        "BUS/WRITE.UL,WCTRL/WDR_WB.UR,MSRC/@1,MUX/M.Q1,ALU/ANDNOT"
;3790   WRITE.UL M[].OR.Q            "BUS/WRITE.UL,WCTRL/WDR_WB.UR,MSRC/@1,MUX/M.Q1,ALU/OR"
;3791   WRITE.UL M[].OR.ZLITO[]      "BUS/WRITE.UL,WCTRL/WDR_WB,MSRC/@1,ALU/OR,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;3792   WRITE.UL M[].ANDNOT.ZLITO[]  "BUS/WRITE.UL,WCTRL/WDR_WB,MSRC/@1,ALU/ANDNOT,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
```

MACROS (CONT)

```
.TOC "  Register Transfer Macros"
ALUS_BCD SIGN.ZERO              "CCMISC/ALUS_DSDZ.CCBR_ALUS"
ALUS_BCD SIGN.ZERO(M[])         "CCMISC/ALUS_DSDZ.CCBR_ALUS,MSRC/@1,RSRC/ZERO,ALU/OR,MUX/M.R1"
ALUS_SIGND                      "CCMISC/ALUS_SIGND.CCBR_ALUS"
ALUS_UNSGN                      "CCMISC/ALUS_UNSGN.CCBR_ALUS"
ASTLVL_M[].RL.24                "WCTRL/ASTLVL_WB,ALPCTL/WX_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE"
ASTLVL_R[]_M[]                  "WCTRL/ASTLVL_WB,SPW/RLONG,RSRC/@1,ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@2"
ASTLVL_[]                       "WCTRL/ASTLVL_WB,LITRL/@1,LIT/LITRL,ROT/ZLIT24,ALPCTL/WX_S"
BUS GRANT M[]_IPL               "BUS/GRANT,WCTRL/GRANT,SPW/MLONG,MSRC/@1"
CC_FPA                          "CCPSL/CC_WB.CCBR_ALUS,FPA/WBUS_FPA.CC"
CC_M[]                          "CCPSL/CC_WB.CCBR_ALUS,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZERO"
CC_M[].NOTAND.R[]               "CCPSL/CC_WB.CCBR_ALUS,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/NOTAND"
CC_M[].OR.R[]                   "CCPSL/CC_WB.CCBR_ALUS,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/OR"
CC_M[].OR.ZLITO[]               "CCPSL/CC_WB.CCBR_ALUS,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/OR"
CC_M[].XOR.ZLITO[]              "CCPSL/CC_WB.CCBR_ALUS,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/XOR"
CC_M[]_MB.AND.ZLITO[]           "CCPSL/CC_WB.CCBR_ALUS,MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
CC_M[]_MB.ANDNOT.CONX(1)        "CCPSL/CC_WB.CCBR_ALUS,ALU/ANDNOT,MUX/M.S,SPW/MLONG,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
CC_M[]_MB.ANDNOT.CONX(4)        "CCPSL/CC_WB.CCBR_ALUS,ALU/ANDNOT,MUX/M.S,SPW/MLONG,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
CC_M[]_MB.OR.CONX(1)            "CCPSL/CC_WB.CCBR_ALUS,ALU/OR,MUX/M.S,SPW/MLONG,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
CC_M[]_ZLITO[]                  "CCPSL/CC_WB.CCBR_ALUS,SPW/MLONG,MSRC/@1,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
CC_R[]                          "CCPSL/CC_WB.CCBR_ALUS,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
CC_ZLITO[]                      "CCPSL/CC_WB.CCBR_ALUS,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
CC_[]                           "CCPSL/CC_WB.CCBR_ALUS,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
CONREGS_D_M[]_R[]               "WCTRL/CONWRITE,MSRC/@1,SPW/MLONG,ALU/OR,MUX/R.S,ROT/ZERO,RSRC/@2,DQ1/D_WX"
CONREGS_M[]                     "WCTRL/CONWRITE,WB_M[@1]"
CONREGS_M[].OR.ZLIT16[]         "WCTRL/CONWRITE,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
CONREGS_M[].RR.16               "WCTRL/CONWRITE,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,MSRC/@1,ALPCTL/WX_S"
CONREGS_M[]_R[]                 "WCTRL/CONWRITE,MSRC/@1,SPW/MLONG,RSRC/@2,MUX/R.S,ALU/OR,ROT/ZERO"
CONREGS_R[]                     "WCTRL/CONWRITE,RSRC/@1,MUX/R.S,ALU/OR,ROT/ZERO"
CONREGS_ZLIT16[]                "WCTRL/CONWRITE,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
CRAR_ZLIT16[]                   "WCTRL/LOADCRAR,LITRL/@1,LIT/LITRL,ROT/ZLIT16,ALPCTL/WX_S"
D(OD)_ZLITO[]                   "DQ1/D_WX,ROT/ZLITO,LIT/LITRL,LITRL/@1,ALUOD/OR.OD,MUX/Z.S"
D_(M[] R[]).RR.9                "DQ1/D_WX,ALPCTL/WX_D_S,MSRC/@1,RSRC/@2,ROT/RR.MR.9"
D_(M[] R[]).RR.P                "ALPCTL/WX_D_S,MSRC/@1,RSRC/@2,ROT/RR.MR.P"
D_(M[]+CONX(2)).SR.1            "DQ1/D_WX,ALU/A+B+CI.SR,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
D_(M[].RR.P).AND.R[]            "DQ1/D_WX,MSRC/@1,RSRC/@2,ROT/RR.MM.P,ALU/AND,MUX/R.S"
D_(R[] M[]).RL.P                "ALPCTL/WX_D_S,MSRC/@2,RSRC/@1,ROT/RL.RM.P"
D_(R[]+CONX(2)).SR.1            "DQ1/D_WX,ALU/A+B+CI.SR,MUX/R.S,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
D_-1                            "ALPCTL/WX_D_S,ROT/MINUS1"
D_CONX.SIZ                      "ALPCTL/WX_D_S,ROT/CONX.SIZ"
D_D+OLIT24[]                    "DQ1/D_WX,MUX/D.S,ALU/A+B+CI,ROT/OLIT24,LIT/LITRL,LITRL/@1"
D_D+R[]                         "DQ1/D_WX,RSRC/@1,MUX/D.R1,ALU/A+B+CI"
D_D+R[]+ALKC                    "DQ1/D_WX,RSRC/@1,MUX/D.R1,ALU/A+B+CI,ALUCI/ALKC"
D_D+ZLITO[]                     "DQ1/D_WX,MUX/D.S,ALU/A+B+CI,ROT/ZLITO,LIT/LITRL,LITRL/@1"
D_D-1                           "DQ1/D_WX,ALU/A+B+CI,MUX/D.S,ROT/MINUS1"
D_D-CONX(2)                     "DQ1/D_WX,ALU/A-B-CI,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
D_D-CONX(4)                     "DQ1/D_WX,ALU/A-B-CI,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
D_D-R[]                         "DQ1/D_WX,RSRC/@1,MUX/D.R1,ALU/A-B-CI"
D_D-ZLITO[]                     "DQ1/D_WX,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/A-B-CI"
```

MACROS (CONT)

```
;3848   D_D.AND.OLIT24[]              "DQ1/D_WX,ALU/AND,MUX/D.S,ROT/OLIT24,LIT/LITRL,LITRL/@1"
;3849   D_D.AND.ZLIT0[]               "DQ1/D_WX,ALU/AND,MUX/D.S,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3850   D_D.AND.ZLIT28[]              "DQ1/D_WX,ALU/AND,MUX/D.S,ROT/ZLIT28,LIT/LITRL,LITRL/@1"
;3851   D_D.OR.R[]                    "DQ1/D_WX,ALU/OR,MUX/D.R1,RSRC/@1"
;3852   D_D.XOR.Q                     "DQ1/D_WX,ALU/XOR,MUX/D.Q1"
;3853   D_D.XOR.ZLIT12[]              "DQ1/D_WX,MUX/D.S,ALU/XOR,ROT/ZLIT12,LIT/LITRL,LITRL/@1"
;3854   D_M[]                         "DQ1/D_WX,MSRC/@1,RSRC/ZERO,ALU/OR,MUX/M.R1"
;3855   D_M[] SQR                     "DQ2/SQR.D_WX,MSRC/@1,RSRC/ZERO,ALU/OR,MUX/M.R2"
;3856   D_M[]+R[]                     "DQ1/D_WX,MSRC/@1,RSRC/@2,ALU/A+B+CI,MUX/M.R1"
;3857   D_M[]+ZLIT0[]                 "DQ1/D_WX,ALU/A+B+CI,MUX/M.S,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3858   D_M[]-R[]                     "DQ1/D_WX,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/A-B-CI"
;3859   D_M[]-ZLIT0[]                 "DQ1/D_WX,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3860   D_M[].AND.R[]                 "DQ1/D_WX,ALU/AND,MUX/M.R1,MSRC/@1,RSRC/@2"
;3861   D_M[].AND.ZLIT16[]            "DQ1/D_WX,ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;3862   D_M[].OR.(R[].RL.24)          "DQ1/D_WX,ALU/OR,MUX/M.S,MSRC/@1,ROT/RR.RR.SIZ,RSRC/@2,DTYPE/BYTE,VSIZE/1"
;3863   D_M[].OR.R[]                  "DQ1/D_WX,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/OR"
;3864   D_M[].RR.16                   "ALPCTL/WX_D_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
;3865   D_M[].RR.16 Q_R[]             "ALPCTL/WX_D_S.Q_R,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,RSRC/@2"
;3866   D_M[].SR.1                    "DQ1/D_WX,ALU/A+B+CI.SR,MUX/M.S,MSRC/@1,ROT/ZERO"
;3867   D_Q Q_D                       "ALPCTL/WX_D_Q.Q_D"
;3868   D_Q_-1                        "ALPCTL/WX_D_Q_S,ROT/MINUS1"
;3869   D_Q_0                         "ALPCTL/WX_D_Q_S,ROT/ZERO"
;3870   D_Q_D.XOR.R[]                 "DQ1/Q_D_WX,ALU/XOR,MUX/D.R1,RSRC/@1"
;3871   D_Q_M[]                       "DQ1/Q_D_WX,MSRC/@1,ROT/ZERO,MUX/M.S,ALU/OR"
;3872   D_Q_M[].XOR.Q                 "DQ1/Q_D_WX,ALU/XOR,MUX/M.Q1,MSRC/@1"
;3873   D_RNUM_D+ZLIT0[]              "DQ1/D_WX,MSRC/RNUM_WBUS,ALU/A+B+CI,MUX/D.S,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3874   D_RNUM_D-ZLIT0[]              "DQ1/D_WX,MSRC/RNUM_WBUS,ALU/A-B-CI,MUX/D.S,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3875   D_RNUM_ZLIT0[]                "ALPCTL/WX_D_S,MSRC/RNUM_WBUS,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3876   D_R[]                         "DQ1/D_WX,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
;3877   D_R[]+CONX.SIZ                "DQ1/D_WX,RSRC/@1,ROT/CONX.SIZ,MUX/R.S,ALU/A+B+CI"
;3878   D_R[]-1                       "DQ1/D_WX,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/MINUS1"
;3879   D_R[]-D-ALKC                  "DQ1/D_WX,RSRC/@1,MUX/D.R1,ALU/B-A-CI,ALUCI/ALKC"
;3880   D_R[]-M[]                     "DQ1/D_WX,RSRC/@1,MSRC/@2,MUX/M.R1,ALU/B-A-CI"
;3881   D_R[].OR.Q                    "DQ1/D_WX,RSRC/@1,MUX/R.Q,ALU/OR"
;3882   D_R[].SR.1                    "DQ1/D_WX,ALU/A+B+CI.SR,MUX/R.S,RSRC/@1,ROT/ZERO"
;3883   D_R[]_ZEXT(XB) PC_PC+1        "DQ1/D_WX,RSRC/@1,SPW/RLONG,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/XM.S,ALU/OR,
;3884                                         ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;3885   D_SEXT(XB) PC_PC+1            "DQ1/D_WX,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR,VSIZE/1,DTYPE/BYTE,
;3886                                         ISTRM/ISIZE_DSIZE"
;3887   D_SEXT(XB) PC_PC+2            "DQ1/D_WX,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR,VSIZE/1,DTYPE/WORD,
;3888                                         ISTRM/ISIZE_DSIZE"
;3889   D_ZEXT(M[])                   "DQ1/D_WX,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR"
;3890   D_ZLIT0[]                     "ALPCTL/WX_D_S,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3891   D_ZLIT0[]-D                   "DQ1/D_WX,MUX/D.S,ALU/B-A-CI,ROT/ZLIT0,LIT/LITRL,LITRL/@1"
;3892   D_ZLIT12[]                    "ALPCTL/WX_D_S,ROT/ZLIT12,LIT/LITRL,LITRL/@1"
;3893   D_ZLIT16[]                    "ALPCTL/WX_D_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
;3894   D_ZLIT24[]                    "ALPCTL/WX_D_S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;3895   D_ZLIT24[] M[]_HARD.REV       "DQ1/D_WX,ALUOD/OR.OD,MUX/Z.S,ROT/ZLIT24,LIT/LITRL,LITRL/@1,MSRC/@2,SPW/MLONG,
;3896                                         WCTRL/REVLEVEL"
;3897   D_ZLIT4[]                     "ALPCTL/WX_D_S,ROT/ZLIT4,LIT/LITRL,LITRL/@1"
;3898   D_ZLIT8[]                     "ALPCTL/WX_D_S,ROT/ZLIT8,LIT/LITRL,LITRL/@1"
;3899
;3900   FLAGS_D_R[]                   "WCTRL/FLAGS_WB,RSRC/@1,ROT/ZERO,ALU/OR,MUX/R.S,DQ1/D_WX"
;3901   FLAGS_M[].AND.ZLIT0[]         "WCTRL/FLAGS_WB,MSRC/@1,ALU/AND,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3902   FLAGS_R[]                     "WCTRL/FLAGS_WB,RSRC/@1,ROT/ZERO,ALU/OR,MUX/R.S"
```

```
;3903   FLAGS_ZLITO[]              "WCTRL/FLAGS_WB,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;3904
;3905   FPA.ENABLE_M[].RR.P        "WCTRL/FPA.ENABLE_WB5,ALPCTL/WX_S,ROT/RR.MM.P,MSRC/@1"
;3906   FPA_MB M[]_R[]             "FPA/FPA_DATA.MBUS,MSRC/@1,SPW/MLONG,RSRC/@2,MUX/R.S,ROT/ZERO,ALU/OR"
;3907   FPA_M[]                    "FPA/FPA_DATA.MBUS,MSRC/@1"
;3908   FPA_M[] FPA_WB_R[]-0       "FPA/FPA_MBUS.FPA_WBUS,MSRC/@1,RSRC/@2,MUX/R.S,ALU/A-B-CI,ROT/ZERO"
;3909   FPA_M[] MDR_R[]            "FPA/FPA_DATA.MBUS,MSRC/@1,WCTRL/MDR_WB,ALU/OR,MUX/R.S,ROT/ZERO,RSRC/@2"
;3910   FPA_Q_MDR MTEMPO_R[]       "FPA/FPA_DATA.MBUS,SPW/MLONG,MSRC/MDR,RSRC/@1,ALPCTL/WX_R.Q_M"
;3911   FPA_Q_M[]                  "FPA/FPA_DATA.MBUS,MSRC/@1,MUX/M.S,ALU/OR,ROT/ZERO,DQ1/Q_WX"
;3912   FPA_Q_M[] MDR_Q            "FPA/FPA_DATA.MBUS,WCTRL/MDR_WB,MSRC/@1,ALPCTL/WX_Q.Q_M"
;3913   FPA_Q_M[] MDR_R[]          "FPA/FPA_DATA.MBUS,ALPCTL/WX_R.Q_M,WCTRL/MDR_WB,MSRC/@1,RSRC/@2"
;3914   FPA_Q_M[].LITNXT           "FPA/FPA_MBUS.LITNXT,MSRC/@1,MUX/M.S,ALU/OR,ROT/ZERO,DQ1/Q_WX"
;3915   FPA_Q_M[] VA_R[]           "FPA/FPA_DATA.MBUS,ALPCTL/WX_R.Q_M,MSRC/@1,RSRC/@2,WCTRL/VA_WB"
;3916   FPA_R[].SIZ_M[]            "FPA/FPA_DATA.MBUS,RSRC/@1,SPW/RSIZE,ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@2"
;3917   FPA_R[]_M[]                "FPA/FPA_DATA.MBUS,RSRC/@1,MSRC/@2,ROT/ZERO,SPW/RLONG,MUX/M.S,ALU/OR"
;3918   FPA_WB_R[]-0               "FPA/FPA_DATA.WBUS,RSRC/@1,MUX/R.S,ALU/A-B-CI,ROT/ZERO"
;3919
;3920   INIR_M[]_Q                 "WCTRL/INIR_WB,MSRC/@1,SPW/MLONG,ALU/OR,MUX/R.Q,RSRC/ZERO"
;3921   IPL_M[].RL.16              "WCTRL/IPL_WB,ALPCTL/WX_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
;3922   IPL_[]                     "WCTRL/IPL_WB,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
;3923
;3924   LONLIT_[]                  "LIT/LONLIT,LONLIT/<.NOT[<LONLIT/@1>]>"
;3925
;3926   MB_M[]                     "MSRC/@1"
;3927   MDR_(M[] R[]).RR.9         "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,ROT/RR.MR.9,ALPCTL/WX_S"
;3928   MDR_(M[] R[]).RR.P         "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,ROT/RR.MR.P,ALPCTL/WX_S"
;3929   MDR_-1                     "WCTRL/MDR_WB,ROT/MINUS1,ALPCTL/WX_S"
;3930   MDR_-M[]                   "WCTRL/MDR_WB,MSRC/@1,ALU/B-A-CI,ALUCI/ZERO,RSRC/ZERO,MUX/M.R1"
;3931   MDR_0                      "WCTRL/MDR_0"
;3932   MDR_M[]                    "WCTRL/MDR_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR"
;3933   MDR_M[]+ALKC               "WCTRL/MDR_WB,MSRC/@1,ALU/A+B+CI,ALUCI/ALKC,RSRC/ZERO,MUX/M.R1"
;3934   MDR_M[]+CONX(1)            "WCTRL/MDR_WB,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,ALU/A+B+CI,MUX/M.S"
;3935   MDR_M[]+R[]+ALKC           "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/ALKC"
;3936   MDR_M[]-CONX.SIZ           "WCTRL/MDR_WB,MSRC/@1,ALU/A-B-CI,ROT/CONX.SIZ,MUX/M.S"
;3937   MDR_M[].AND.OLIT8[]        "WCTRL/MDR_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/OLIT8,MUX/M.S,ALU/AND"
;3938   MDR_M[].AND.ZLITO[]        "WCTRL/MDR_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/AND"
;3939   MDR_M[].ANDNOT.R[]         "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/ANDNOT"
;3940   MDR_M[].ANDNOT.ZLITO[]     "WCTRL/MDR_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/ANDNOT"
;3941   MDR_M[].ASR.P              "WCTRL/MDR_WB,MSRC/@1,ROT/ASR.M.P,ALPCTL/WX_S"
;3942   MDR_M[].FPLIT              "WCTRL/MDR_WB,MSRC/@1,ROT/FPLIT,ALPCTL/WX_S"
;3943   MDR_M[].OR.(R[].RR.24)     "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG,MUX/M.S,ALU/OR"
;3944   MDR_M[].OR.CVTNP(R[])      "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,ROT/CVTNP,ALU/OR,MUX/M.S"
;3945   MDR_M[].OR.R[]             "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/OR"
;3946   MDR_M[].OR.ZLIT24[]        "WCTRL/MDR_WB,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;3947   MDR_M[].RL.16              "WCTRL/MDR_WB,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;3948   MDR_M[].RL.24              "WCTRL/MDR_WB,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;3949   MDR_M[].RL.8               "WCTRL/MDR_WB,MSRC/@1,VSIZE/1,DTYPE/LONG,ROT/RR.MM.SIZ,ALPCTL/WX_S"
;3950   MDR_M[].RL.9               "WCTRL/MDR_WB,ALPCTL/WX_S,ROT/RL.MM.PTE,MSRC/@1"
;3951   MDR_M[].RR.16              "WCTRL/MDR_WB,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;3952   MDR_M[].XOR.R[]            "WCTRL/MDR_WB,MSRC/@1,RSRC/@2,ALU/XOR,MUX/M.R1"
;3953   MDR_M[].XOR.ZLIT12[]       "WCTRL/MDR_WB,MSRC/@1,ROT/ZLIT12,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/XOR"
;3954   MDR_M[]_R[]                "WCTRL/MDR_WB,MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR"
;3955   MDR_M[]_R[].RR.16          "WCTRL/MDR_WB,MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;3956   MDR_M[]_ZLITO[]            "WCTRL/MDR_WB,MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLITO,ALPCTL/WX_S"
;3957   MDR_Q                      "WCTRL/MDR_WB,RSRC/ZERO,MUX/R.Q,ALU/OR"
```

MACROS (CONT)

```
;3958   MDR_Q Q_M[]                     "WCTRL/MDR_WB,MSRC/@1,ALPCTL/WX_Q.Q_M"
;3959   MDR_Q_M[]                       "WCTRL/MDR_WB,DQ1/Q_WX,MSRC/@1,ROT/ZERO,MUX/M.S,ALU/OR"
;3960   MDR_Q_M[]-ZLIT8[]               "WCTRL/MDR_WB,DQ1/Q_WX,MSRC/@1,ALU/A-B-CI,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;3961   MDR_R[]                         "WCTRL/MDR_WB,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR"
;3962   MDR_R[]-CONX(1)                 "WCTRL/MDR_WB,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,ALU/A-B-CI,MUX/R.S"
;3963   MDR_R[]-M[]-ALKC                "WCTRL/MDR_WB,MSRC/@2,RSRC/@1,ALU/B-A-CI,ALUCI/ALKC,MUX/M.R1"
;3964   MDR_R[].RR.24                   "WCTRL/MDR_WB,RSRC/@1,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG,ALPCTL/WX_S"
;3965   MDR_R[]_M[]                     "WCTRL/MDR_WB,RSRC/@1,SPW/RLONG,MSRC/@2,ALU/OR,MUX/M.S,ROT/ZERO"
;3966   MDR_R[]_RB-CONX.SIZ             "WCTRL/MDR_WB,RSRC/@1,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,SPW/RLONG,VSIZE/1,DTYPE/IDEP"
;3967   MDR_SEXT(M[])                   "WCTRL/MDR_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR"
;3968   MDR_SEXT(XB)+R[] PC_PC+I        "WCTRL/MDR_WB,RSRC/@1,MSRC/XB.PC_PC+I,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI"
;3969   MDR_XB PC_PC+1                  "WCTRL/MDR_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,ISTRM/ISIZE_DSIZE,
;3970                                                        VSIZE/1,DTYPE/BYTE"
;3971   MDR_XB PC_PC+2                  "WCTRL/MDR_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,ISTRM/ISIZE_DSIZE,
;3972                                                        VSIZE/1,DTYPE/WORD"
;3973   MDR_XB PC_PC+4                  "WCTRL/MDR_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,ISTRM/ISIZE_DSIZE,
;3974                                                        VSIZE/1,DTYPE/LONG"
;3975   MDR_XB PC_PC+I                  "WCTRL/MDR_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,ISTRM/ISIZE_DSIZE,
;3976                                                        VSIZE/1,DTYPE/IDEP"
;3977   MDR_ZEXT(IR)                    "WCTRL/MDR_IR"
;3978   MDR_ZEXT(M[])                   "WCTRL/MDR_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR"
;3979   MDR_ZEXT(OSR)                   "CCPSL/MDR_OSR.CCBR_BRATST"
;3980   MDR_ZLIT0[]                     "WCTRL/MDR_WB,LIT/LITRL,LITRL/@1,ROT/ZLIT0,ALPCTL/WX_S"
;3981   MDR_ZLIT16[]                    "WCTRL/MDR_WB,LIT/LITRL,LITRL/@1,ROT/ZLIT16,ALPCTL/WX_S"
;3982   MDR_ZLIT24[]                    "WCTRL/MDR_WB,LIT/LITRL,LITRL/@1,ROT/ZLIT24,ALPCTL/WX_S"
;3983
;3984   MEMSCAR_0 M[]_ZLIT0[]           "WCTRL/MEMSCAR_WB,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3985   MEMSCAR_M[]-ZLIT0[]             "WCTRL/MEMSCAR_WB,MSRC/@1,ALU/A-B-CI,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3986   MEMSCAR_M[]_0                   "WCTRL/MEMSCAR_WB,MSRC/@1,SPW/MLONG,ROT/ZERO,ALPCTL/WX_S"
;3987   MEMSCAR_M[]_MB-ZLIT0[]          "WCTRL/MEMSCAR_WB,MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3988   MEMSCAR_Q_R[]                   "WCTRL/MEMSCAR_WB,DQ1/Q_WX,RSRC/@1,ROT/ZERO,ALU/OR,MUX/R.S"
;3989   MEMSCAR_R[]_0                   "WCTRL/MEMSCAR_WB,RSRC/@1,SPW/RLONG,ROT/ZERO,ALPCTL/WX_S"
;3990   MEMSCAR_ZLIT24[]                "WCTRL/MEMSCAR_WB,ALPCTL/WX_S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;3991
;3992   MEMSCR_-1                       "WCTRL/MEMSCR_WB,ROT/MINUS1,ALPCTL/WX_S"
;3993   MEMSCR_0                        "WCTRL/MEMSCR_WB,ROT/ZERO,ALPCTL/WX_S"
;3994   MEMSCR_0 M[]_ZLIT0[]            "WCTRL/MEMSCR_WB,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;3995   MEMSCR_M[].ANDNOT.(R[].RL.24)   "WCTRL/MEMSCR_WB,ALU/ANDNOT,MUX/M.S,MSRC/@1,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/BYTE"
;3996   MEMSCR_M[].RL.24                "WCTRL/MEMSCR_WB,ALPCTL/WX_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE"
;3997   MEMSCR_M[]_ZEXT(MB)             "WCTRL/MEMSCR_WB,MSRC/@1,SPW/MLONG,ROT/ZERO,MUX/XM.S,ALUXM/ZERO,ALU/OR"
;3998   MEMSCR_RTEMP7_M[]-ZLIT0[]       "WCTRL/MEMSCR_WB,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A-B-CI,SPW/RLONG"
;3999   MEMSCR_R[]                      "WCTRL/MEMSCR_WB,RSRC/@1,MUX/R.S,ROT/ZERO,ALU/OR"
;4000   MEMSCR_R[]_M[].RL.24            "WCTRL/MEMSCR_WB,RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;4001   MEMSCR_ZLIT24[]                 "WCTRL/MEMSCR_WB,ALPCTL/WX_S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;4002
;4003   MTEMPO_(MDR.RR.P).AND.R[]       "SPW/MLONG,MSRC/MDR,RSRC/@1,ROT/RR.MM.P,ALU/AND,MUX/R.S"
;4004   MTEMPO_D_0 Q_SEXT(MDR)          "SPW/MLONG,MSRC/MDR,RSRC/ZERO,ALPCTL/WX_D_R.Q_XM,ALUXM/SIGN"
;4005   MTEMPO_D_MDR-ZLIT0[]            "SPW/MLONG,MSRC/MDR,ROT/ZLIT0,LIT/LITRL,LITRL/@1,MUX/M.S,ALU/A-B-CI,DQ1/D_WX"
;4006   MTEMPO_MDR+1                    "SPW/MLONG,MSRC/MDR,ROT/MINUS1,MUX/M.S,ALU/A-B-CI"
;4007   MTEMPO_MDR+Q                    "SPW/MLONG,MSRC/MDR,MUX/M.Q1,ALU/A+B+CI"
;4008   MTEMPO_MDR+R[]                  "SPW/MLONG,MSRC/MDR,RSRC/@1,MUX/M.R1,ALU/A+B+CI"
;4009   MTEMPO_MDR+R[]+ALKC             "SPW/MLONG,MSRC/MDR,RSRC/@1,MUX/M.R1,ALU/A+B+CI,ALUCI/ALKC"
;4010   MTEMPO_MDR-ZLIT0[]              "SPW/MLONG,MSRC/MDR,ROT/ZLIT0,LIT/LITRL,LITRL/@1,MUX/M.S,ALU/A-B-CI"
;4011   MTEMPO_MDR.AND.OLIT8[]          "SPW/MLONG,MSRC/MDR,LIT/LITRL,LITRL/@1,ROT/OLIT8,MUX/M.S,ALU/AND"
;4012   MTEMPO_MDR.AND.R[]              "SPW/MLONG,MSRC/MDR,RSRC/@1,ALU/AND,MUX/M.R1"
```

MACROS (CONT)

```
;4013   MTEMPO_MDR.AND.ZLITO[]            "SPW/MLONG,MSRC/MDR,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/M.S,ALU/AND"
;4014   MTEMPO_MDR.NOTAND.R[]             "SPW/MLONG,MSRC/MDR,RSRC/@1,MUX/M.R1,ALU/NOTAND"
;4015   MTEMPO_MDR.OR.R[]                 "SPW/MLONG,MSRC/MDR,RSRC/@1,MUX/M.R1,ALU/OR"
;4016   MTEMPO_MDR.OR.ZLITO[]             "SPW/MLONG,MSRC/MDR,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/M.S,ALU/OR"
;4017   MTEMPO_MDR.OR.ZLIT16[]            "SPW/MLONG,MSRC/MDR,LIT/LITRL,LITRL/@1,ROT/ZLIT16,MUX/M.S,ALU/OR"
;4018   MTEMPO_MDR.OR.ZLIT8[]             "SPW/MLONG,MSRC/MDR,LIT/LITRL,LITRL/@1,ROT/ZLIT8,MUX/M.S,ALU/OR"
;4019   MTEMPO_MDR_(MDR+ZLITO[]).SL.1     "SPW/MLONG,MSRC/MDR,WCTRL/MDR_WB,ALU/A+B+CI.SL,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4020   MTEMPO_PC-R[]                     "SPW/MLONG,MSRC/PC,RSRC/@1,MUX/M.R1,ALU/A-B-CI"
;4021   MTEMPO_PC.ANDNOT.R[]              "SPW/MLONG,MSRC/PC,RSRC/@1,MUX/M.R1,ALU/ANDNOT"
;4022   MTEMPO_PC.OR.R[]                  "SPW/MLONG,MSRC/PC,RSRC/@1,MUX/M.R1,ALU/OR"
;4023   MTEMPO_R[] Q_MDR                  "SPW/MLONG,MSRC/MDR,RSRC/@1,ALPCTL/WX_R.Q_M"
;4024   MTEMPO_R[]-PCBACK                 "SPW/MLONG,MSRC/PCBACK,ALU/B-A-CI,MUX/M.R1,RSRC/@1"
;4025   MTEMPO_R[].RR.24 Q_PCBACK         "SPW/MLONG,RSRC/@1,ROT/RR.RR.SIZ,MSRC/PCBACK,ALPCTL/WX_S.Q_XM,VSIZE/1,DTYPE/LONG"
;4026   MTEMPO_RBSP                       "MSRC/WB_RBSP,SPW/MLONG"
;4027   MTEMPO_SEXT(MDR).XOR.Q            "SPW/MLONG,MSRC/MDR,MUX/XM.Q,ALUXM/SIGN,ALU/XOR"
;4028   MTEMPO_SEXT(XB)+R[] PC_PC+2       "SPW/MLONG,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI,
;4029                                               ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/WORD"
;4030   MTEMPO_SEXT(XB)+R[] PC_PC+4       "SPW/MLONG,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI,
;4031                                               ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/LONG"
;4032   MTEMPO_SEXT(XB)+R[] PC_PC+I       "SPW/MLONG,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI"
;4033   MTEMPO_UNPACK(MDR R[])            "SPW/MLONG,MSRC/MDR,ROT/GETFPF,RSRC/@1,ALPCTL/WX_S"
;4034   MTEMPO_VA+ZLITO[]                 "SPW/MLONG,MSRC/VA,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/M.S,ALU/A+B+CI"
;4035   MTEMPO_VA-R[]                     "SPW/MLONG,ALU/A-B-CI,MUX/M.R1,MSRC/VA,RSRC/@1"
;4036   MTEMPO_XB PC_PC+4                 "SPW/MLONG,ALU/OR,MUX/M.S,MSRC/XB.PC_PC+I,ROT/ZERO,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/LONG"
;4037   MTEMPO_ZEXT(M[])                  "MSRC/@1,SPW/MLONG,ROT/ZERO,MUX/XM.S,ALU/OR,ALUXM/ZERO"
;4038   MTEMPO_ZEXT(XB) PC_PC+1           "SPW/MLONG,ALU/OR,MUX/XM.S,MSRC/XB.PC_PC+I,ROT/ZERO,ISTRM/ISIZE_DSIZE,
;4039                                                    VSIZE/1,DTYPE/BYTE,ALUXM/ZERO"
;4040
;4041   M[]_(MB Q).RL.1                   "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALU/A+B+CI.SL,MUX/M.R2,ALUSHF/ROT,ALUCI/ZERO,DQ2/SQL"
;4042   M[]_(MB Q).SL.1                   "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALU/A+B+CI.SL,MUX/M.R2,ALUSHF/SHF,ALUCI/ZERO,DQ2/SQL"
;4043   M[]_(MB R[]).RR.4                 "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.4,ALPCTL/WX_S"
;4044   M[]_(MB R[]).RR.9                 "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.9,ALPCTL/WX_S"
;4045   M[]_(MB R[]).RR.P                 "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.P,ALPCTL/WX_S"
;4046   M[]_(MB R[]).RR.PS                "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.PS,ALPCTL/WX_S"
;4047   M[]_(MB R[]).RR.S                 "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.S,ALPCTL/WX_S"
;4048   M[]_(MB R[]).XZ                   "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/XZ.MR,ALPCTL/WX_S"
;4049   M[]_(MB+ALKC).BCD                 "MSRC/@1,SPW/MLONG,ALU/A+B+CI.BCD,MUX/M.R1,RSRC/ZERO,ALUCI/ALKC,VSIZE/1,DTYPE/LONG"
;4050   M[]_(MB+R[]).BCD                  "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI.BCD,MUX/M.R1,VSIZE/1,DTYPE/LONG"
;4051   M[]_(MB+R[]).RL.1                 "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SL,MUX/M.R1,ALUSHF/ROT,RSRC/@2"
;4052   M[]_(MB+R[]).RR.1                 "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SR,MUX/M.R1,ALUSHF/ROT,RSRC/@2"
;4053   M[]_(MB+R[]).SL.1                 "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI.SL,MUX/M.R1,ALUSHF/ZERO,ALUCI/ZERO"
;4054   M[]_(MB+R[]).SR.1                 "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI.SR,MUX/M.R1,ALUSHF/ZERO,ALUCI/ZERO"
;4055   M[]_(MB+R[]+ALKC).BCD             "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI.BCD,MUX/M.R1,ALUCI/ALKC,VSIZE/1,DTYPE/LONG"
;4056   M[]_(MB+R[]+ALKC).SL.1 ALU<0>_0   "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SL,MUX/M.R1,ALUSHF/ZERO,RSRC/@2,ALUCI/ALKC"
;4057   M[]_(MB+R[]+ALKC).SL.1 ALU<0>_1   "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SL,MUX/M.R1,ALUSHF/ONE,RSRC/@2,ALUCI/ALKC"
;4058   M[]_(MB+ZLIT24[]).BCD             "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT24,ALU/A+B+CI.BCD,MUX/M.S,VSIZE/1,DTYPE/LONG"
;4059   M[]_(MB-R[]).BCD                  "MSRC/@1,RSRC/@2,MUX/M.R1,VSIZE/1,DTYPE/LONG,ALU/A-B-CI.BCD,SPW/MLONG"
;4060   M[]_(MB-R[]).SL.1                 "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A-B-CI.SL,MUX/M.R1,ALUSHF/ZERO,ALUCI/ZERO"
;4061   M[]_(MB-R[]-ALKC).BCD             "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,VSIZE/1,DTYPE/LONG,ALU/A-B-CI.BCD,ALUCI/ALKC"
;4062   M[]_(MB-R[]-ALKC).SL.1 ALU<0>_0   "MSRC/@1,SPW/MLONG,ALU/A-B-CI.SL,MUX/M.R1,ALUSHF/ZERO,RSRC/@2,ALUCI/ALKC"
;4063   M[]_(MB-R[]-ALKC).SL.1 ALU<0>_1   "MSRC/@1,SPW/MLONG,ALU/A-B-CI.SL,MUX/M.R1,ALUSHF/ONE,RSRC/@2,ALUCI/ALKC"
;4064   M[]_(MB.RR.4).NOT                 "MSRC/@1,SPW/MLONG,ALPCTL/WX_.NOT.S,ROT/RR.MR.4,RSRC/@1"
;4065   M[]_(R[] MB).RL.4                 "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RL.RM.4,RSRC/@2"
;4066   M[]_(R[] MB).RL.P                 "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RL.RM.P,RSRC/@2"
;4067   M[]_(R[] MB).RL.PS                "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RL.RM.PS,ALPCTL/WX_S"
```

MACROS (CONT)

```
;4068   M[]_(R[]+PL).SR.1          "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/PL,ALU/A+B+CI.SR,MUX/R.S"
;4069   M[]_(R[].ASL.1)-MB         "MSRC/@1,SPW/MLONG,ALU/B-A-CI,MUX/M.S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/WORD,RSRC/@2"
;4070   M[]_(R[].ASL.SIZE)         "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ASL.R.SIZ,ALPCTL/WX_S"
;4071   M[]_-1                     "MSRC/@1,SPW/MLONG,ROT/MINUS1,ALPCTL/WX_S"
;4072   M[]_-MB                    "MSRC/@1,SPW/MLONG,ALUCI/ZERO,ALU/B-A-CI,RSRC/ZERO,MUX/M.R1"
;4073   M[]_-R[]                   "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/B-A-CI"
;4074   M[]_0                      "MSRC/@1,SPW/MLONG,ROT/ZERO,ALPCTL/WX_S"
;4075   M[]_0+ZLITO[]              "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/Z.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4076   M[]_ASTLVL                 "MSRC/@1,SPW/MLONG,WCTRL/ASTLVL"
;4077   M[]_ATCR                   "MSRC/@1,SPW/MLONG,CCMISC/WB_ATCR.CCBR_SIGND"
;4078   M[]_CC_ZLITO[]             "MSRC/@1,SPW/MLONG,CCPSL/CC_WB.CCBR_ALUS,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4079   M[]_CONREGS                "MSRC/@1,SPW/MLONG,WCTRL/CONREAD"
;4080   M[]_CONX(1)                "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4081   M[]_CONX(2)                "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4082   M[]_CONX(4)                "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4083   M[]_CONX.SIZ               "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CONX.SIZ"
;4084   M[]_CVTNP(MB)              "MSRC/@1,SPW/MLONG,ROT/CVTNP,ALU/A+B+CI.BCD,MUX/R.S,RSRC/ZERO"
;4085   M[]_D                      "MSRC/@1,SPW/MLONG,MUX/D.S,ROT/ZERO,ALU/OR"
;4086   M[]_D+(R[].RR.16)          "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/D.S,ROT/RR.RR.SIZ,DTYPE/WORD,VSIZE/1,RSRC/@2"
;4087   M[]_D+ALKC                 "MSRC/@1,SPW/MLONG,MUX/D.R1,ALU/A+B+CI,RSRC/ZERO,ALUCI/ALKC"
;4088   M[]_D+R[]                  "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/D.R1,ALU/A+B+CI"
;4089   M[]_D+R[]+ALKC             "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI,MUX/D.R1,ALUCI/ALKC"
;4090   M[]_D+ZLIT12[]             "MSRC/@1,SPW/MLONG,MUX/D.S,ROT/ZLIT12,LIT/LITRL,LITRL/@2,ALU/A+B+CI"
;4091   M[]_D+ZLIT8[]              "MSRC/@1,SPW/MLONG,MUX/D.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2,ALU/A+B+CI"
;4092   M[]_D-ZLITO[]              "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/D.S,ALU/A-B-CI"
;4093   M[]_D-ZLIT12[]             "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT12,MUX/D.S,ALU/A-B-CI"
;4094   M[]_D-ZLIT8[]              "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/D.S,ALU/A-B-CI"
;4095   M[]_D.AND.ZLITO[]          "MSRC/@1,SPW/MLONG,ALU/AND,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4096   M[]_D.AND.ZLIT8[]          "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/D.S,ALU/AND"
;4097   M[]_D.ANDNOT.ZLITO[]       "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4098   M[]_D.OR.(MB.RR.16)        "MSRC/@1,SPW/MLONG,ALU/OR,MUX/D.S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
;4099   M[]_D.OR.PACK(MB R[])      "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/FPACK,ALU/OR,MUX/D.S"
;4100   M[]_D.OR.ZLIT8[]           "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/D.S,ALU/OR"
;4101   M[]_D_(MB R[]).RR.P        "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.MR.P,ALPCTL/WX_D_S"
;4102   M[]_D_(MB+Q).SL.1 Q<0>_1   "MSRC/@1,SPW/MLONG,DQ2/SQL.D_WX,ALU/A+B+CI.SL,MUX/M.Q2,ALUSHF/ALU0.Q1"
;4103   M[]_D_D.ANDNOT.ZLITO[]     "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/D.S,ALU/ANDNOT,DQ1/D_WX"
;4104   M[]_D_SEXT(MB)             "MSRC/@1,SPW/MLONG,DQ1/D_WX,MUX/XM.R,ALU/OR,RSRC/ZERO,ALUXM/SIGN"
;4105   M[]_D_Q                    "MSRC/@1,SPW/MLONG,RSRC/ZERO,MUX/R.Q,ALU/OR,DQ1/D_WX"
;4106   M[]_D_Q Q_MB               "MSRC/@1,SPW/MLONG,ALPCTL/WX_D_Q.Q_M"
;4107   M[]_D_Q_0                  "MSRC/@1,SPW/MLONG,ALPCTL/WX_D_Q_S,ROT/ZERO"
;4108   M[]_D_Q_D+Q+1              "MSRC/@1,SPW/MLONG,DQ1/Q_D_WX,ALU/A+B+CI,MUX/D.Q1,ALUCI/ONE"
;4109   M[]_D_REM                  "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALPCTL/REM,ROT/0"
;4110   M[]_D_R[]                  "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX"
;4111   M[]_D_ZLITO[]              "MSRC/@1,SPW/MLONG,ALPCTL/WX_D_S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4112   M[]_D_ZLIT12[]             "MSRC/@1,SPW/MLONG,ALPCTL/WX_D_S,ROT/ZLIT12,LIT/LITRL,LITRL/@2"
;4113   M[]_D_ZLIT16[]             "MSRC/@1,SPW/MLONG,DQ1/D_WX,ROT/ZLIT16,LIT/LITRL,LITRL/@2,MUX/Z.S,ALU/A+B+CI"
;4114   M[]_FLAGS                  "MSRC/@1,SPW/MLONG,WCTRL/CM.TP.FPD.FLAGS"
;4115   M[]_FPTCR                  "MSRC/@1,SPW/MLONG,WCTRL/FPTCR"
;4116   M[]_MB+(R[].ASL.1)         "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/WORD,RSRC/@2"
;4117   M[]_MB+(R[].ASL.2)         "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/LONG,ALU/A+B+CI,MUX/M.S"
;4118   M[]_MB+(R[].RR.24)         "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/LONG"
;4119   M[]_MB+(R[].SL.2)          "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/LONG,RSRC/@2"
;4120   M[]_MB+1                   "MSRC/@1,SPW/MLONG,ALU/A+B+CI,ALUCI/ONE,MUX/M.R1,MSRC/@1,RSRC/ZERO"
;4121   M[]_MB+PL                  "MSRC/@1,SPW/MLONG,MUX/M.S,ALU/A+B+CI,ROT/PL"
;4122   M[]_MB+Q                   "MSRC/@1,SPW/MLONG,MUX/M.Q1,ALU/A+B+CI,ALUCI/ZERO"
```

```
;4123   M[]_MB+R[]               "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO"
;4124   M[]_MB+R[]+ALKC          "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/ALKC"
;4125   M[]_MB+ZLIT0[]           "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/A+B+CI"
;4126   M[]_MB+ZLIT12[]          "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT12,MUX/M.S,ALU/A+B+CI"
;4127   M[]_MB+ZLIT16[]          "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4128   M[]_MB+ZLIT24[]          "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4129   M[]_MB+ZLIT8[]           "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4130   M[]_MB-1                 "MSRC/@1,SPW/MLONG,ALU/A-B-CI,ALUCI/ONE,MUX/M.R1,RSRC/ZERO"
;4131   M[]_MB-CONX(1)           "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4132   M[]_MB-CONX(2)           "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4133   M[]_MB-R[]               "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/A-B-CI"
;4134   M[]_MB-R[]-ALKC          "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/A-B-CI,ALUCI/ALKC"
;4135   M[]_MB-ZLIT0[]           "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4136   M[]_MB-ZLIT12[]          "MSRC/@1,SPW/MLONG,MUX/M.S,ALU/A-B-CI,ROT/ZLIT12,LIT/LITRL,LITRL/@2"
;4137   M[]_MB-ZLIT16[]          "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4138   M[]_MB-ZLIT8[]           "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4139   M[]_MB.AND.CONX(1)       "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4140   M[]_MB.AND.OLIT0[]       "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/OLIT0,MUX/M.S,ALU/AND"
;4141   M[]_MB.AND.OLIT16[]      "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/OLIT16,LIT/LITRL,LITRL/@2"
;4142   M[]_MB.AND.OLIT24[]      "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/OLIT24,LIT/LITRL,LITRL/@2"
;4143   M[]_MB.AND.R[]           "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.R1,RSRC/@2"
;4144   M[]_MB.AND.ZLIT0[]       "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4145   M[]_MB.AND.ZLIT16[]      "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4146   M[]_MB.AND.ZLIT24[]      "MSRC/@1,SPW/MLONG,ALU/AND,MUX/M.S,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4147   M[]_MB.AND.ZLIT8[]       "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/M.S,ALU/AND"
;4148   M[]_MB.ANDNOT.CONX(1)    "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/CONX.SIZ,DTYPE/BYTE,VSIZE/1"
;4149   M[]_MB.ANDNOT.CONX(2)    "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1"
;4150   M[]_MB.ANDNOT.CONX(4)    "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/CONX.SIZ,DTYPE/LONG,VSIZE/1"
;4151   M[]_MB.ANDNOT.O1IT16[]   "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/OLIT16,LIT/LITRL,LITRL/@2"
;4152   M[]_MB.ANDNOT.R[]        "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/ANDNOT"
;4153   M[]_MB.ANDNOT.ZLIT0[]    "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4154   M[]_MB.ANDNOT.ZLIT12[]   "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT12,MUX/M.S,ALU/ANDNOT"
;4155   M[]_MB.ANDNOT.ZLIT16[]   "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4156   M[]_MB.ANDNOT.ZLIT24[]   "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4157   M[]_MB.ANDNOT.ZLIT28[]   "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLIT28,LIT/LITRL,LITRL/@2"
;4158   M[]_MB.ANDNOT.ZLIT8[]    "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4159   M[]_MB.ANDNOT.ZLITPL[]   "MSRC/@1,SPW/MLONG,ALU/ANDNOT,MUX/M.S,ROT/ZLITPL,LIT/LITRL,LITRL/@2"
;4160   M[]_MB.ASR.P             "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ASR.M.P"
;4161   M[]_MB.BCDSWP            "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/BCDSWP"
;4162   M[]_MB.CLR1B             "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CLR1BM"
;4163   M[]_MB.CLR2B             "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CLR2BM"
;4164   M[]_MB.OR.(MB.RL.16)     "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,MUX/M.S,ALU/OR"
;4165   M[]_MB.OR.(MB.RL.8)      "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG,MUX/M.S,ALU/OR"
;4166   M[]_MB.OR.(R[].RR.16)    "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD"
;4167   M[]_MB.OR.(R[].RR.24)    "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/LONG"
;4168   M[]_MB.OR.(R[].RR.8)     "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,RSRC/@2,DTYPE/BYTE,VSIZE/1"
;4169   M[]_MB.OR.(R[].RL.8)     "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/LONG"
;4170   M[]_MB.OR.CONX(1)        "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/CONX.SIZ,DTYPE/BYTE,VSIZE/1"
;4171   M[]_MB.OR.CONX(2)        "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1"
;4172   M[]_MB.OR.CONX(4)        "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/CONX.SIZ,DTYPE/LONG,VSIZE/1"
;4173   M[]_MB.OR.R[]            "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/OR"
;4174   M[]_MB.OR.ZLIT0[]        "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4175   M[]_MB.OR.ZLIT12[]       "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLIT12,LIT/LITRL,LITRL/@2"
;4176   M[]_MB.OR.ZLIT16[]       "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4177   M[]_MB.OR.ZLIT24[]       "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
```

```
;4178   M[]_MB.OR.ZLIT28[]          "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT28,MUX/M.S,ALU/OR"
;4179   M[]_MB.OR.ZLIT8[]           "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4180   M[]_MB.OR.ZLITPL[]          "MSRC/@1,SPW/MLONG,ALU/OR,MUX/M.S,ROT/ZLITPL,LIT/LITRL,LITRL/@2"
;4181   M[]_MB.RL.1                 "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALU/A+B+CI.SL,MUX/M.R1,ALUCI/ZERO,ALUSHF/ROT"
;4182   M[]_MB.RL.24                "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE"
;4183   M[]_MB.RL.4                 "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RL.RM.4,RSRC/@1"
;4184   M[]_MB.RL.8                 "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,ALPCTL/WX_S,VSIZE/1,DTYPE/LONG"
;4185   M[]_MB.RR.16                "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;4186   M[]_MB.RR.24                "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,ALPCTL/WX_S,VSIZE/1,DTYPE/LONG"
;4187   M[]_MB.RR.4                 "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RR.MR.4,RSRC/@1"
;4188   M[]_MB.RR.8                 "MSRC/@1,SPW/MLONG,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;4189   M[]_MB.RR.P                 "MSRC/@1,SPW/MLONG,ROT/RR.MM.P,ALPCTL/WX_S"
;4190   M[]_MB.SR.1                 "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALU/A+B+CI.SR,MUX/M.R1,ALUCI/ZERO,ALUSHF/ZERO"
;4191   M[]_MB.XOR.ZLIT12[]         "MSRC/@1,SPW/MLONG,ROT/ZLIT12,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/XOR"
;4192   M[]_MEMSCR                  "MSRC/@1,SPW/MLONG,WCTRL/MEMSCR"
;4193   M[]_OLIT0[]                 "MSRC/@1,SPW/MLONG,ROT/OLIT0,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4194   M[]_OLIT16[]                "MSRC/@1,SPW/MLONG,ROT/OLIT16,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4195   M[]_OLIT24[]                "MSRC/@1,SPW/MLONG,ROT/OLIT24,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4196   M[]_OLIT8[]                 "MSRC/@1,SPW/MLONG,ROT/OLIT8,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4197   M[]_PACK(MB R[])            "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/FPACK,ALPCTL/WX_S"
;4198   M[]_PL                      "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/PL"
;4199   M[]_PL_Q                    "MSRC/@1,SPW/MLONG,ROT/SL.PL_WB,RSRC/ZERO,MUX/R.Q,ALU/OR"
;4200   M[]_PSL                     "MSRC/@1,SPW/MLONG,CCPSL/WB_PSL.CCBR_SIGND"
;4201   M[]_Q                       "MSRC/@1,SPW/MLONG,RSRC/ZERO,MUX/R.Q,ALU/OR"
;4202   M[]_Q Q_D                   "MSRC/@1,SPW/MLONG,ALPCTL/WX_Q.Q_D"
;4203   M[]_Q Q_MB PL_[]            "MSRC/@1,SPW/MLONG,ALPCTL/WX_Q.Q_M,ROT/OLIT0.PL_LIT,LIT/LITRL,LITRL/@2"
;4204   M[]_Q_D+R[]+ALKC            "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/A+B+CI,MUX/D.R1,ALUCI/ALKC,DQ1/Q_WX"
;4205   M[]_Q_D_R[]                 "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/Q_D_WX"
;4206   M[]_Q_FLAGS_R[]             "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/Q_WX,WCTRL/FLAGS_WB"
;4207   M[]_Q_MB.ASR.P              "MSRC/@1,SPW/MLONG,ALPCTL/WX_Q_S,ROT/ASR.M.P"
;4208   M[]_Q_MB.XOR.R[]            "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,DQ1/Q_WX,ALU/XOR"
;4209   M[]_Q_R[]                   "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/R.S,ROT/ZERO,ALU/OR,DQ1/Q_WX"
;4210   M[]_Q_UNPACK(MB R[])        "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/GETFPF,ALPCTL/WX_Q_S"
;4211   M[]_Q_ZLIT0[]               "MSRC/@1,SPW/MLONG,ALPCTL/WX_Q_S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4212   M[]_R[]                     "MSRC/@1,SPW/MLONG,ALU/OR,MUX/R.S,ROT/ZERO,RSRC/@2"
;4213   M[]_R[] Q_MB                "MSRC/@1,SPW/MLONG,RSRC/@2,ALPCTL/WX_R.Q_M"
;4214   M[]_R[] Q_ZEXT(MB)          "MSRC/@1,SPW/MLONG,ALPCTL/WX_R.Q_XM,RSRC/@2,ALUXM/ZERO"
;4215   M[]_R[]+0                   "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/R.S,RSRC/@2,ROT/ZERO"
;4216   M[]_R[]+1                   "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/MINUS1,MUX/R.S,ALU/A-B-CI"
;4217   M[]_R[]+Q                   "MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/R.Q,RSRC/@2"
;4218   M[]_R[]-0                   "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/R.S,RSRC/@2,ROT/ZERO"
;4219   M[]_R[]-1                   "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/MINUS1,MUX/R.S,ALU/A+B+CI"
;4220   M[]_R[]-D-ALKC              "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/D.R1,ALUCI/ALKC,ALU/B-A-CI"
;4221   M[]_R[]-MB                  "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/B-A-CI,ALUCI/ZERO"
;4222   M[]_R[]-MB-ALKC             "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/B-A-CI,ALUCI/ALKC"
;4223   M[]_R[]-Q                   "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/R.Q,RSRC/@2"
;4224   M[]_R[].ASL.1               "MSRC/@1,SPW/MLONG,RSRC/@2,ALPCTL/WX_S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/WORD"
;4225   M[]_R[].ASL.P               "MSRC/@1,SPW/MLONG,RSRC/@2,ALPCTL/WX_S,ROT/ASL.R.P"
;4226   M[]_R[].NOT                 "MSRC/@1,SPW/MLONG,ALU/NOTAND,MUX/R.S,ROT/MINUS1,RSRC/@2"
;4227   M[]_R[].OR.((MB RB).RR.4)   "MSRC/@1,SPW/MLONG,ROT/RR.MR.4,RSRC/@2,MUX/R.S,ALU/OR"
;4228   M[]_R[].OR.((RB MB).RL.4)   "MSRC/@1,SPW/MLONG,ROT/RL.RM.4,RSRC/@2,MUX/R.S,ALU/OR"
;4229   M[]_R[].OR.D                "MSRC/@1,SPW/MLONG,RSRC/@2,ALU/OR,MUX/D.R1"
;4230   M[]_R[].RR.16               "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD,RSRC/@2"
;4231   M[]_R[].RR.24               "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG,ALPCTL/WX_S"
;4232   M[]_R[].RR.8                "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/BYTE"
```

MACROS (CONT)

```
;4233   M[]_R[].RR.PS             "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.RR.PS,ALPCTL/WX_S"
;4234   M[]_R[].RR.SIZ            "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/RR.RR.SIZ,ALPCTL/WX_S"
;4235   M[]_R[].SL.1              "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SL,MUX/R.S,RSRC/@2,ROT/ZERO"
;4236   M[]_R[].SR.1              "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SR,MUX/R.S,RSRC/@2,ROT/ZERO"
;4237   M[]_R[].XZ                "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/XZ.RR,ALPCTL/WX_S"
;4238   M[]_SEXT(MB)              "MSRC/@1,SPW/MLONG,RSRC/ZERO,ALUXM/SIGN,MUX/XM.R,ALU/OR"
;4239   M[]_SEXT(MB).XOR.Q        "MSRC/@1,SPW/MLONG,MUX/XM.Q,ALUXM/SIGN,ALU/XOR"
;4240   M[]_STEPC                 "MSRC/@1,SPW/MLONG,WCTRL/CM.TP.FPD.F5.STEPC"
;4241   M[]_TCSR.IICR             "MSRC/@1,SPW/MLONG,WCTRL/TCSR.IICR"
;4242   M[]_TOYOS.TOYCN           "MSRC/@1,SPW/MLONG,WCTRL/TODCLK"
;4243   M[]_TRAR_ZLIT16[]         "MSRC/@1,SPW/MLONG,WCTRL/LOADTRAR,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4244   M[]_TU58REGS              "MSRC/@1,SPW/MLONG,WCTRL/TU58READ"
;4245   M[]_UNPACK(MB R[])        "MSRC/@1,SPW/MLONG,ROT/GETFPF,RSRC/@2,ALPCTL/WX_S"
;4246   M[]_VA_R[]-1              "MSRC/@1,SPW/MLONG,RSRC/@2,ROT/MINUS1,MUX/R.S,ALU/A+B+CI,WCTRL/VA_WB"
;4247   M[]_WB                    "SPW/MLONG,MSRC/@1"
;4248   M[]_ZEXT(MB)              "MSRC/@1,SPW/MLONG,ROT/ZERO,ALU/OR,MUX/XM.S,ALUXM/ZERO"
;4249   M[]_ZEXT(MB)+R[]          "MSRC/@1,SPW/MLONG,RSRC/@2,MUX/XM.R,ALU/A+B+CI,ALUXM/ZERO"
;4250   M[]_ZEXT(MB)-R[]          "MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/XM.R,RSRC/@2,ALUXM/ZERO"
;4251   M[]_ZEXT(MB).SR.1         "MSRC/@1,SPW/MLONG,ALU/A+B+CI.SR,MUX/XM.R,RSRC/ZERO,ALUSHF/ZERO"
;4252   M[]_ZLITO[]               "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4253   M[]_ZLITO[] Q_0           "MSRC/@1,SPW/MLONG,ALPCTL/WX_S.Q_0,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4254   M[]_ZLITO[]-MB            "MSRC/@1,SPW/MLONG,ALU/B-A-CI,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4255   M[]_ZLIT8[]-MB            "MSRC/@1,SPW/MLONG,ALU/B-A-CI,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4256   M[]_ZLIT12[]              "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLIT12,LIT/LITRL,LITRL/@2"
;4257   M[]_ZLIT16[]              "MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4258   M[]_ZLIT24[]              "MSRC/@1,SPW/MLONG,ROT/ZLIT24,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4259   M[]_ZLIT28[]              "MSRC/@1,SPW/MLONG,ROT/ZLIT28,LIT/LITRL,LITRL/@2,ALPCTL/WX_S"
;4260   M[]_ZLIT8[]               "MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT8,ALPCTL/WX_S"
;4261
;4262   PC_-M[]                   "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/B-A-CI"
;4263   PC_D_D+OLITO[]            "WCTRL/PC_WB,LIT/LITRL,LITRL/@1,ROT/OLITO,MUX/D.S,ALU/A+B+CI,DQ1/D_WX"
;4264   PC_D_D-1                  "WCTRL/PC_WB,MUX/D.R1,RSRC/ZERO,ALU/A-B-CI,ALUCI/ONE,DQ1/D_WX"
;4265   PC_D_D-ZLITO[]            "WCTRL/PC_WB,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/A-B-CI,DQ1/D_WX"
;4266   PC_D_M[]                  "WCTRL/PC_WB,MSRC/@1,ROT/ZERO,MUX/M.S,ALU/OR,DQ1/D_WX"
;4267   PC_D_M[]-1                "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A-B-CI,ALUCI/ONE,DQ1/D_WX"
;4268   PC_D_M[]-ZLITO[]          "WCTRL/PC_WB,MSRC/@1,LITRL/@2,LIT/LITRL,ROT/ZLITO,MUX/M.S,ALU/A-B-CI,DQ1/D_WX"
;4269   PC_M[]                    "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR"
;4270   PC_M[]+(R[].RR.16)        "WCTRL/PC_WB,ALU/A+B+CI,MUX/M.S,MSRC/@1,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/WORD"
;4271   PC_M[]+PSLC               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A+B+CI,ALUCI/PSLC"
;4272   PC_M[]+Q                  "WCTRL/PC_WB,MSRC/@1,MUX/M.Q1,ALU/A+B+CI"
;4273   PC_M[]+R[]                "WCTRL/PC_WB,MSRC/@1,RSRC/@2,ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1"
;4274   PC_M[]+ZLITO[]            "WCTRL/PC_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/A+B+CI"
;4275   PC_M[]+ZLIT4[]            "WCTRL/PC_WB,MSRC/@1,ROT/ZLIT4,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A+B+CI"
;4276   PC_M[]+ZLIT8[]            "WCTRL/PC_WB,MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A+B+CI"
;4277   PC_M[]-CONX(2)            "WCTRL/PC_WB,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4278   PC_M[]-PSLC               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A-B-CI,ALUCI/PSLC"
;4279   PC_M[]-R[]                "WCTRL/PC_WB,MSRC/@1,RSRC/@2,ALU/A-B-CI,ALUCI/ZERO,MUX/M.R1"
;4280   PC_M[]-ZLITO[]            "WCTRL/PC_WB,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4281   PC_M[].ANDNOT.ZLITO[]     "WCTRL/PC_WB,ALU/ANDNOT,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4282   PC_M[].NOTAND.R[]         "WCTRL/PC_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/NOTAND"
;4283   PC_M[].OR.R[]             "WCTRL/PC_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/OR"
;4284   PC_M[].OR.ZLITO[]         "WCTRL/PC_WB,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4285   PC_M[].RR.P               "WCTRL/PC_WB,MSRC/@1,ROT/RR.MM.P,ALPCTL/WX_S"
;4286   PC_M[].SL.1               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A+B+CI.SL"
;4287   PC_M[].SR.1               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A+B+CI.SR"
```

MACROS (CONT)

```
;4288  PC_M[].XOR.Q               "WCTRL/PC_WB,MSRC/@1,MUX/M.Q1,ALU/XOR"
;4289  PC_M[]_D                   "WCTRL/PC_WB,MSRC/@1,SPW/MLONG,ALU/OR,MUX/D.S,ROT/ZERO"
;4290  PC_M[]_R[]                 "WCTRL/PC_WB,MSRC/@1,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR,SPW/MLONG"
;4291  PC_PC+((D+ZLITO[]).SL.1)   "WCTRL/PC_PC+WB,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/A+B+CI.SL"
;4292  PC_PC+(M[].SL.1)           "WCTRL/PC_PC+WB,MSRC/@1,ROT/MINUS1,MUX/M.S,ALU/AND.SL"
;4293  PC_PC+(SEXT(M[]).SL.1)     "WCTRL/PC_PC+WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI.SL"
;4294  PC_PC+1 MB_XB              "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;4295  PC_PC+2 MB_XB              "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/WORD"
;4296  PC_PC+D                    "WCTRL/PC_PC+WB,MUX/D.R1,RSRC/ZERO,ALU/OR"
;4297  PC_PC+I MB_XB              "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/IDEP"
;4298  PC_PC+M[]                  "WCTRL/PC_PC+WB,MUX/M.R1,MSRC/@1,RSRC/ZERO,ALU/OR"
;4299  PC_PC+Q                    "WCTRL/PC_PC+WB,MUX/R.Q,RSRC/ZERO,ALU/OR"
;4300  PC_PC+SEXT(M[])            "WCTRL/PC_PC+WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR"
;4301  PC_PC+SEXT(XB)+1           "WCTRL/PC_PC+WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI,ALUCI/ONE,
;4302                                     ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;4303  PC_Q                       "WCTRL/PC_WB,RSRC/ZERO,MUX/R.Q,ALU/OR"
;4304  PC_Q-M[]                   "WCTRL/PC_WB,MSRC/@1,MUX/M.Q1,ALU/B-A-CI"
;4305  PC_Q-M[]-1                 "WCTRL/PC_WB,MSRC/@1,MUX/M.Q1,ALU/B-A-CI,ALUCI/ONE"
;4306  PC_Q_M[].RR.SIZ            "WCTRL/PC_WB,MSRC/@1,ROT/RR.MM.SIZ,ALPCTL/WX_Q_S"
;4307  PC_R[]                     "WCTRL/PC_WB,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
;4308  PC_R[]+CONX(2)             "WCTRL/PC_WB,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1"
;4309  PC_R[]_M[]                 "WCTRL/PC_WB,SPW/RLONG,RSRC/@1,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/OR"
;4310  PC_R[]_M[]-1               "WCTRL/PC_WB,MSRC/@2,RSRC/@1,ROT/MINUS1,MUX/M.S,ALU/A+B+CI,SPW/RLONG"
;4311  PC_R[]_M[]-CONX(2)         "WCTRL/PC_WB,RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD,MUX/M.S,ALU/A-B-CI"
;4312  PC_R[]_Q Q_D               "WCTRL/PC_WB,SPW/RLONG,RSRC/@1,ALPCTL/WX_Q.Q_D"
;4313  PC_R[]_RB+1                "WCTRL/PC_WB,RSRC/@1,ROT/MINUS1,MUX/R.S,ALU/A-B-CI,SPW/RLONG"
;4314  PC_R[]_RB-1                "WCTRL/PC_WB,RSRC/@1,ROT/MINUS1,MUX/R.S,ALU/A+B+CI,SPW/RLONG"
;4315  PC_SEXT(M[])               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR"
;4316  PC_ZEXT(M[])               "WCTRL/PC_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR"
;4317  PC_ZEXT(M[])+Q             "WCTRL/PC_WB,ALU/A+B+CI,MUX/XM.Q,MSRC/@1,ALUXM/ZERO"
;4318  PC_ZEXT(M[])+R[]           "WCTRL/PC_WB,ALU/A+B+CI,MUX/XM.R,MSRC/@1,RSRC/@2,ALUXM/ZERO"
;4319  PC_ZEXT(XB<7-0>)+R[]       "WCTRL/PC_WB,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/ZERO,ALU/A+B+CI,
;4320                                   ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;4321  PC_ZLITO[]                 "WCTRL/PC_WB,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4322
;4323  PL<4-3>_WB                 "ROT/OLITO.PL43_WB"
;4324  PL_-D                      "ROT/SL.PL_WB,RSRC/ZERO,MUX/D.R1,ALU/B-A-CI"
;4325  PL_-M[]                    "ROT/SL.PL_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/B-A-CI"
;4326  PL_D_M[]-R[]               "ROT/SL.PL_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/A-B-CI,DQ1/D_WX"
;4327  PL_D_Q_0                   "ROT/SL.PL_WB,DQ1/Q_D_WX,ALU/AND,MUX/R.S,RSRC/ZERO"
;4328  PL_MSS M[]                 "ROT/PL_MSS,MSRC/@1,RSRC/ZERO,ALU/OR,MUX/M.R1"
;4329  PL_M[]                     "ROT/SL.PL_WB,ALU/OR,MUX/M.R1,MSRC/@1,RSRC/ZERO"
;4330  PL_M[]_0                   "ROT/SL.PL_WB,MSRC/@1,SPW/MLONG,ALU/AND,MUX/R.S,RSRC/ZERO"
;4331  PL_M[]_D_Q_0               "ROT/SL.PL_WB,MSRC/@1,SPW/MLONG,DQ1/Q_D_WX,ALU/AND,MUX/R.S,RSRC/ZERO"
;4332  PL_M[]_MB+Q                "ROT/SL.PL_WB,MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.Q1"
;4333  PL_Q                       "ROT/SL.PL_WB,MUX/R.Q,RSRC/ZERO,ALU/OR"
;4334  PL_Q-M[]                   "ROT/SL.PL_WB,MSRC/@1,MUX/M.Q1,ALU/B-A-CI"
;4335  PL_Q_M[]-R[]               "ROT/SL.PL_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/A-B-CI,DQ1/Q_WX"
;4336  PL_Q_R[]-M[]               "ROT/SL.PL_WB,MSRC/@2,RSRC/@1,MUX/M.R1,ALU/B-A-CI,DQ1/Q_WX"
;4337  PL_R[]-M[]                 "ROT/SL.PL_WB,RSRC/@1,MSRC/@2,MUX/M.R1,ALU/B-A-CI"
;4338  PL_R[]_RB+Q                "ROT/SL.PL_WB,RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.Q"
;4339  PL_[]                      "ROT/OLITO.PL_LIT,LIT/LITRL,LITRL/@1"
;4340
;4341  PME_0                      "ROT/ZERO,ALPCTL/WX_S,WCTRL/PME"
;4342  PME_0 FPDOFFSET_3          "WCTRL/PME,SPW/MLONG,MSRC/FPDOFFSET,ROT/ZLITO,LIT/LITRL,LITRL/3,ALPCTL/WX_S"
```

```
;4343   PME_Q_M[].AND.ZLIT24[]        "WCTRL/PME,DQ1/Q_WX,ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4344   PME_Q.RR.1                    "WCTRL/PME,ALU/A+B+CI.SR,MUX/R.O,RSRC/ZERO,ALUSHF/ROT"
;4345
;4346   PSL(PREV_CURM ISCURM_R[])     "WCTRL/PREV_CUR.ISCUR_WB,RSRC/@1,MUX/R.S,ROT/ZERO,ALU/OR"
;4347   PSL(PREV_CURM ISCURM_[])      "WCTRL/PREV_CUR.ISCUR_WB,ALPCTL/WX_S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;4348   PSL<CURM>_M[].RR.8            "WCTRL/PREV_CUR.ISCUR_WB,ALPCTL/WX_S,ROT/RR.MM.SIZ,MSRC/@1,VSIZE/1,DTYPE/BYTE"
;4349   PSL<NZVC>_0                   "CCPSL/CC_WB.CCBR_ALUS,ALPCTL/WX_S,ROT/ZERO"
;4350   PSL_M[]                       "CCPSL/PSL_WB.CCBR_ALUS,ALU/OR,MUX/M.R1,RSRC/ZERO,MSRC/@1"
;4351   PSL_M[].ANDNOT.ZLIT0[]        "CCPSL/PSL_WB.CCBR_ALUS,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/ANDNOT"
;4352   PSL_M[].ANDNOT.ZLIT24[]       "CCPSL/PSL_WB.CCBR_ALUS,ALU/ANDNOT,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4353   PSL_M[].OR.ZLIT0[]            "CCPSL/PSL_WB.CCBR_ALUS,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/OR"
;4354   PSL_M[].OR.ZLIT24[]           "CCPSL/PSL_WB.CCBR_ALUS,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4355   PSL_M[]_MB<31-16>.0           "CCPSL/PSL_WB.CCBR_ALUS,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/CLR2BM"
;4356   PSL_M[]_ZLIT0[]               "CCPSL/PSL_WB.CCBR_ALUS,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLIT0,LIT/LITRL,LITRL/@2"
;4357   PSL_R[]                       "CCPSL/PSL_WB.CCBR_ALUS,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/XOR"
;4358   PSL_ZLIT16[]                  "CCPSL/PSL_WB.CCBR_ALUS,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
;4359
;4360   PSW_M[]                       "CCPSL/PSW_WB.CCBR_ALUS,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR"
;4361   PSW_M[].AND.ZLIT0[]           "CCPSL/PSW_WB.CCBR_ALUS,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/AND"
;4362   PSW_M[].NOTAND.R[]            "CCPSL/PSW_WB.CCBR_ALUS,ALU/NOTAND,MUX/M.R1,MSRC/@1,RSRC/@2"
;4363   PSW_M[].OR.R[]                "CCPSL/PSW_WB.CCBR_ALUS,ALU/OR,MUX/M.R1,MSRC/@1,RSRC/@2"
;4364
;4365   Q_(M[] R[]).RR.P              "ALPCTL/WX_Q_S,MSRC/@1,RSRC/@2,ROT/RR.MR.P"
;4366   Q_(R[] M[]).RL.4              "ALPCTL/WX_Q_S,ROT/RL.RM.4,RSRC/@1,MSRC/@2"
;4367   Q_(R[] M[]).RL.P              "ALPCTL/WX_Q_S,RSRC/@1,MSRC/@2,ROT/RL.RM.P"
;4368   Q_(R[]+Q).RR.1                "DQ1/Q_WX,ALU/A+B+CI.SR,MUX/R.Q,RSRC/@1,ALUSHF/ROT"
;4369   Q_(R[]+Q).SL.1                "DQ1/Q_WX,ALU/A+B+CI.SL,MUX/R.Q,RSRC/@1"
;4370   Q_-1                          "ALPCTL/WX_Q_S,ROT/MINUS1"
;4371   Q_-SEXT(M[])                  "DQ1/Q_WX,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALU/B-A-CI,ALUXM/SIGN"
;4372   Q_0                           "ALPCTL/WX_Q_S,ROT/ZERO"
;4373   Q_D+Q                         "DQ1/Q_WX,MUX/D.Q1,ALU/A+B+CI,ALUCI/ZERO"
;4374   Q_D.XOR.(M[].RL.P)            "DQ1/Q_WX,ALU/XOR,MUX/D.S,ROT/RL.MM.P,MSRC/@1"
;4375   Q_MDR VA_MTEMPO_R[]           "ALPCTL/WX_R.Q_M,RSRC/@1,MSRC/MDR,SPW/MLONG,WCTRL/VA_WB"
;4376   Q_M[]                         "DQ1/Q_WX,MSRC/@1,ROT/ZERO,MUX/M.S,ALU/OR"
;4377   Q_M[] D_QOLD                  "ALPCTL/WX_D_Q.Q_M,MSRC/@1"
;4378   Q_M[] D_R[]                   "ALPCTL/WX_D_R.Q_M,MSRC/@1,RSRC/@2"
;4379   Q_M[] MDR_R[]                 "ALPCTL/WX_R.Q_M,WCTRL/MDR_WB,MSRC/@1,RSRC/@2"
;4380   Q_M[] VA_ZLIT0[]              "ALPCTL/WX_S.Q_XM,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2,WCTRL/VA_WB,VSIZE/1,DTYPE/LONG"
;4381   Q_M[] MDR_ZLIT0[]             "ALPCTL/WX_S.Q_XM,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2,WCTRL/MDR_WB,VSIZE/1,DTYPE/LONG"
;4382   Q_M[] VA_R[]                  "ALPCTL/WX_R.Q_M,WCTRL/VA_WB,MSRC/@1,RSRC/@2"
;4383   Q_M[] WB_R[]                  "ALPCTL/WX_R.Q_M,MSRC/@1,RSRC/@2"
;4384   Q_M[]+R[]                     "DQ1/Q_WX,ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2"
;4385   Q_M[]+R[]+ALKC                "DQ1/Q_WX,ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2,ALUCI/ALKC"
;4386   Q_M[]-Q                       "DQ1/Q_WX,MSRC/@1,MUX/M.Q1,ALU/A-B-CI"
;4387   Q_M[]-R[]                     "DQ1/Q_WX,MSRC/@1,RSRC/@2,ALU/A-B-CI,MUX/M.R1"
;4388   Q_M[]-ZLIT0[]                 "DQ1/Q_WX,MSRC/@1,ROT/ZLIT0,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A-B-CI"
;4389   Q_M[]-ZLIT8[]                 "DQ1/Q_WX,MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A-B-CI"
;4390   Q_M[].AND.((MB R[]).RR.4)     "DQ1/Q_WX,ALU/AND,MUX/M.S,MSRC/@1,ROT/RR.MR.4,RSRC/@2"
;4391   Q_M[].AND.(R[].RR.24)         "DQ1/Q_WX,ALU/AND,MUX/M.S,MSRC/@1,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/LONG"
;4392   Q_M[].AND.R[]                 "DQ1/Q_WX,ALU/AND,MUX/M.R1,MSRC/@1,RSRC/@2"
;4393   Q_M[].AND.ZLIT0[]             "DQ1/Q_WX,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/AND"
;4394   Q_M[].ANDNOT.ZLIT0[]          "DQ1/Q_WX,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT0,MUX/M.S,ALU/ANDNOT"
;4395   Q_M[].ASL.P                   "ALPCTL/WX_Q_S,MSRC/@1,ROT/ASL.M.P"
;4396   Q_M[].RL.P                    "ALPCTL/WX_Q_S,MSRC/@1,ROT/RL.MM.P"
;4397   Q_M[].RR.16                   "ALPCTL/WX_Q_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
```

MACROS (CONT)

```
Q_M[].SL.1                  "DQ1/Q_WX,ALU/AND.SL,MUX/M.S,MSRC/@1,ROT/MINUS1"
Q_M[].XZ.MM                 "ALPCTL/WX_Q_S,MSRC/@1,ROT/XZ.MM"
Q_M[]_R[]                   "DQ1/Q_WX,MSRC/@1,SPW/MLONG,ALU/OR,MUX/R.S,ROT/ZERO,RSRC/@2"
Q_OLITO[]                   "ROT/OLITO,LIT/LITRL,LITRL/@1,ALPCTL/WX_Q_S"
Q_Q+M[]                     "DQ1/Q_WX,MSRC/@1,MUX/M.Q1,ALU/A+B+CI"
Q_Q+R[]                 "DQ1/Q_WX,RSRC/@1,MUX/R.Q,ALU/B-A-CI"
Q_Q-M[]                     "DQ1/Q_WX,MSRC/@1,MUX/M.Q1,ALU/B-A-CI"
Q_Q-R[]                     "DQ1/Q_WX,RSRC/@1,MUX/R.Q,ALU/B-A-CI"
Q_R[]                       "DQ1/Q_WX,RSRC/@1,ROT/ZERO,ALU/OR,MUX/R.S"
Q_R[] M[]_0                 "ALPCTL/WX_S.Q_R,SPW/MLONG,MSRC/@2,RSRC/@1,ROT/ZERO"
Q_R[]+0                     "DQ1/Q_WX,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/A+B+CI"
Q_R[]-Q                     "DQ1/Q_WX,RSRC/@1,MUX/R.Q,ALU/A-B-CI"
Q_R[].SR.1                  "DQ1/Q_WX,ALU/A+B+CI.SR,MUX/R.S,RSRC/@1,ROT/ZERO"
Q_R[]_D+RB+ALKC             "DQ1/Q_WX,RSRC/@1,SPW/RLONG,MUX/D.R1,ALU/A+B+CI,ALUCI/ALKC"
Q_SEXT(M[])                 "DQ1/Q_WX,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALU/OR,ALUXM/SIGN"
Q_SEXT(XB) PC_PC+1          "DQ1/Q_WX,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALU/OR,ALUXM/SIGN,DTYPE/BYTE,
                                 ISTRM/ISIZE_DSIZE,VSIZE/1"
Q_SEXT(XB)-R[] PC_PC+1      "DQ1/Q_WX,MSRC/XB.PC_PC+I,RSRC/@1,ALU/A-B-CI,MUX/XM.R,ALUXM/SIGN,VSIZE/1,
                                 DTYPE/BYTE,ISTRM/ISIZE_DSIZE"
Q_SEXT.M[] STEPC_[]         "ALPCTL/WX_S.Q_XM,ALUXM/SIGN,ROT/ZLITO,LIT/LITRL,LITRL/@2,MSRC/@1,WCTRL/STEPC_WB"
Q_UNPACK(M[] R[])           "ALPCTL/WX_Q_S,MSRC/@1,RSRC/@2,ROT/GETFPF"
Q_XB PC_PC+I                "DQ1/Q_WX,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,ISTRM/ISIZE_DSIZE,DTYPE/IDEP,VSIZE/1"
Q_XB-R[] PC_PC+1            "DQ1/Q_WX,ALU/A-B-CI,MUX/M.R1,MSRC/XB.PC_PC+I,RSRC/@1,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
Q_ZEXT(M[])                 "DQ1/Q_WX,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALU/OR,ALUXM/ZERO"
Q_ZEXT(M[])-R[]             "DQ1/Q_WX,MSRC/@1,RSRC/@2,ALU/A-B-CI,MUX/XM.R,ALUXM/ZERO"
Q_ZEXT(M[]).SL.1            "DQ1/Q_WX,MSRC/@1,ROT/ZERO,MUX/XM.S,ALUXM/ZERO,ALU/A+B+CI.SL"
Q_ZEXT(XB)-R[] PC_PC+1      "DQ1/Q_WX,ALU/A-B-CI,MUX/XM.R,MSRC/XB.PC_PC+I,RSRC/@1,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
Q_ZLITO[]                   "ALPCTL/WX_Q_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
Q_ZLITO[]-M[]               "DQ1/Q_WX,ROT/ZLITO,LIT/LITRL,LITRL/@1,MSRC/@2,MUX/M.S,ALU/B-A-CI"
Q_ZLIT4[]-M[]               "DQ1/Q_WX,ROT/ZLIT4,LIT/LITRL,LITRL/@1,MSRC/@2,MUX/M.S,ALU/B-A-CI"
Q_ZLITPL[]                  "ALPCTL/WX_Q_S,ROT/ZLITPL,LIT/LITRL,LITRL/@1"

RBSP_0                      "RSRC/ZERO.CLRRBSP"

RNUM_0                      "MSRC/RNUM_WBUS,ROT/ZERO,ALPCTL/WX_S"
RNUM_1                      "MSRC/RNUM_WBUS,ROT/MINUS1,MUX/Z.S,ALU/A-B-CI"
RNUM_D_D+1                  "MSRC/RNUM_WBUS,DQ1/D_WX,ALU/A+B+CI,ALUCI/ONE,MUX/D.R1,RSRC/ZERO"
RNUM_D_D-1                  "MSRC/RNUM_WBUS,DQ1/D_WX,ALU/A-B-CI,ALUCI/ONE,MUX/D.R1,RSRC/ZERO"
RNUM_D_ZLITO[]              "MSRC/RNUM_WBUS,DQ1/D_WX,ALU/A+B+CI,MUX/Z.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
RNUM_POPRBS MTEMPO_POPSIZ   "MSRC/READRBS,SPW/MLONG"
RNUM_Q_D_Q+1                "MSRC/RNUM_WBUS,DQ1/Q_D_WX,ALU/A+B+CI,MUX/R.Q,RSRC/ZERO,ALUCI/ONE"
RNUM_Q_Q+1                  "MSRC/RNUM_WBUS,DQ1/Q_WX,ALU/A+B+CI,MUX/R.Q,RSRC/ZERO,ALUCI/ONE"
RNUM_Q_Q-1                  "MSRC/RNUM_WBUS,DQ1/Q_WX,ALU/B-A-CI,MUX/R.Q,RSRC/ZERO,ALUCI/ONE"
RNUM_Q_ZLITO[]              "MSRC/RNUM_WBUS,ALPCTL/WX_Q_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
RNUM_R[]                    "MSRC/RNUM_WBUS,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR"
RNUM_R[]+CONX(2)            "MSRC/RNUM_WBUS,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
RNUM_R[]_0                  "MSRC/RNUM_WBUS,SPW/RLONG,RSRC/@1,ROT/ZERO,ALPCTL/WX_S"
RNUM_R[]_RB+1               "MSRC/RNUM_WBUS,SPW/RLONG,RSRC/@1,ALU/A-B-CI,MUX/R.S,ROT/MINUS1"
RNUM_R[]_RB+CONX(2)         "MSRC/RNUM_WBUS,SPW/RLONG,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD,MUX/R.S,ALU/A+b+CI"
RNUM_R[]_RB-1               "MSRC/RNUM_WBUS,SPW/RLONG,RSRC/@1,ALU/A+B+CI,MUX/R.S,ROT/MINUS1"
RNUM_[]                     "MSRC/RNUM_WBUS,ALPCTL/WX_S,ROT/ZLITO,LITRL/@1,LIT/LITRL"

RTEMPO_G_-Q                 "SPW/RLONG,RSRC/ZERO,DQ1/Q_WX,MUX/R.Q,ALU/A-B-CI"
RTEMPO_RNUM_PL_0            "MSRC/RNUM_WBUS,ROT/SL.PL_WB,ALU/AND,MUX/R.S,RSRC/ZERO,SPW/RLONG"
```

```
;4453   RTEMP7_M[] PL<4-0>_31 PL<5>_1   "MSRC/@1,SPW/RLONG,ROT/OLITO.PL_LIT,LIT/LITRL,LITRL/1FF,MUX/M.S,ALU/AND"
;4454   RTEMP7_M[]-ZLITO[]              "MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A-B-CI,SPW/RLONG"
;4455   RTEMP7_M[].AND.ZLIT8[]          "MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/AND,SPW/RLONG"
;4456   RTEMP7_RNUM PL<4-0>_31 PL<5>_1  "ALPCTL/NOP,MSRC/wBUS_RNUM,SPW/RLONG,ROT/OLITO.PL_LIT,LIT/LITRL,LITRL/1FF"
;4457   RTEMP7_RNUM_ZLITO[]             "SPW/RLONG,MSRC/RNUM_WBUS,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4458
;4459   R[]                             "RSRC/@1"
;4460   R[].SIZ_(RB+M[].ASR.3).SR.1     "RSRC/@1,SPW/RSIZE,ALU/A+B+CI.SR,MUX/R.S,ROT/ASR.M.3,MSRC/@2,DTYPE/BYTE,VSIZE/1"
;4461   R[].SIZ_-1                      "RSRC/@1,SPW/RSIZE,ROT/MINUS1,ALPCTL/WX_S"
;4462   R[].SIZ_-D                      "RSRC/@1,SPW/RSIZE,ROT/ZERO,MUX/D.S,ALU/B-A-CI"
;4463   R[].SIZ_-M[]                    "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/B-A-CI"
;4464   R[].SIZ_-RB                     "RSRC/@1,SPW/RSIZE,ROT/ZERO,MUX/R.S,ALU/B-A-CI"
;4465   R[].SIZ_0                       "RSRC/@1,SPW/RSIZE,ROT/ZERO,ALPCTL/WX_S"
;4466   R[].SIZ_D                       "RSRC/@1,SPW/RSIZE,ALU/OR,MUX/D.S,ROT/ZERO"
;4467   R[].SIZ_FLAGS                   "RSRC/@1,SPW/RSIZE,WCTRL/CM.TP.FPD.FLAGS"
;4468   R[].SIZ_FPA                     "RSRC/@1,SPW/RSIZE,FPA/WBUS_FPA"
;4469   R[].SIZ_M[]                     "RSRC/@1,SPW/RSIZE,ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@2"
;4470   R[].SIZ_M[]+1                   "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.S,ROT/MINUS1,ALU/A-B-CI"
;4471   R[].SIZ_M[]+Q                   "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/A+B+CI"
;4472   R[].SIZ_M[]+Q+PSLC              "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/A+B+CI,ALUCI/PSLC"
;4473   R[].SIZ_M[]+RB                  "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/A+B+CI"
;4474   R[].SIZ_M[]+RB+PSLC             "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/PSLC"
;4475   R[].SIZ_M[]-1                   "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.S,ROT/MINUS1,ALU/A+B+CI"
;4476   R[].SIZ_M[]-Q                   "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/A-B-CI"
;4477   R[].SIZ_M[].AND.RB              "RSRC/@1,SPW/RSIZE,ALU/AND,MUX/M.R1,MSRC/@2"
;4478   R[].SIZ_M[].ANDNOT.Q            "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/ANDNOT"
;4479   R[].SIZ_M[].ASR.-P              "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/ASR.M.-P,ALPCTL/WX_S"
;4480   R[].SIZ_M[].BCDSWP              "RSRC/@1,SPW/RSIZE,MSRC/@2,ALPCTL/WX_S,ROT/BCDSWP"
;4481   R[].SIZ_M[].NOT                 "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/MINUS1,MUX/M.S,ALU/NOTAND"
;4482   R[].SIZ_M[].NOTAND.RB           "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/NOTAND"
;4483   R[].SIZ_M[].OR.Q                "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/OR"
;4484   R[].SIZ_M[].OR.RB               "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/OR"
;4485   R[].SIZ_M[].RR.8                "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/RR.MM.SIZ,ALPCTL/WX_S,DTYPE/BYTE,VSIZE/1"
;4486   R[].SIZ_M[].RR.P                "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/RR.MM.P,ALPCTL/WX_S"
;4487   R[].SIZ_M[].RR.SIZ              "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/RR.MM.SIZ,ALPCTL/WX_S"
;4488   R[].SIZ_M[].SL.1                "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/A+B+CI.SL"
;4489   R[].SIZ_M[].XOR.Q               "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.Q1,ALU/XOR"
;4490   R[].SIZ_M[].XOR.RB              "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/XOR"
;4491   R[].SIZ_Q-RB-1                  "RSRC/@1,SPW/RSIZE,MUX/R.Q,ALU/B-A-CI,ALUCI/ONE"
;4492   R[].SIZ_Q Q_D                   "RSRC/@1,SPW/RSIZE,ALPCTL/WX_Q.Q_D"
;4493   R[].SIZ_Q_D_M[].ASL.P           "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/ASL.M.P,ALPCTL/WX_D_Q_S"
;4494   R[].SIZ_Q_M[]                   "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/OR,DQ1/Q_WX"
;4495   R[].SIZ_Q_M[].RR.SIZ            "RSRC/@1,SPW/RSIZE,MSRC/@2,ROT/RR.MM.SIZ,ALPCTL/WX_Q_S"
;4496   R[].SIZ_RB+1                    "RSRC/@1,SPW/RSIZE,ALU/A-B-CI,MUX/R.S,ROT/MINUS1"
;4497   R[].SIZ_RB+CONX(2)              "RSRC/@1,SPW/RSIZE,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1,MUX/R.S,ALU/A+B+CI"
;4498   R[].SIZ_RB-CONX(2)              "RSRC/@1,SPW/RSIZE,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1,MUX/R.S,ALU/A-B-CI"
;4499   R[].SIZ_RB+D                    "RSRC/@1,SPW/RSIZE,MUX/D.R1,ALU/A+B+CI"
;4500   R[].SIZ_RB-D                    "RSRC/@1,SPW/RSIZE,MUX/D.R1,ALU/B-A-CI"
;4501   R[].SIZ_RB-1                    "RSRC/@1,SPW/RSIZE,ROT/MINUS1,MUX/R.S,ALU/A+B+CI"
;4502   R[].SIZ_RB-M[]                  "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/B-A-CI"
;4503   R[].SIZ_RB-M[]-PSLC             "RSRC/@1,SPW/RSIZE,MSRC/@2,MUX/M.R1,ALU/B-A-CI,ALUCI/PSLC"
;4504   R[].SIZ_RB.SL.1                 "RSRC/@1,SPW/RSIZE,ROT/ZERO,MUX/R.S,ALU/A+B+CI.SL"
;4505
;4506   R[]_((M[] RB).RR.4.AND.MB).SL.1 "RSRC/@1,SPW/RLONG,ALU/AND.SL,MUX/M.S,ROT/RR.MR.4,MSRC/@2"
;4507   R[]_(M[] RB).RR.4               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MR.4,ALPCTL/WX_S"
```

MACROS (CONT)

```
;4508  R[]_(M[] RB).RR.9              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MR.9,ALPCTL/WX_S"
;4509  R[]_(M[] RB).RR.P              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MR.P,ALPCTL/WX_S"
;4510  R[]_(M[] RB).RR.PS             "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MR.PS,ALPCTL/WX_S"
;4511  R[]_(M[] RB).RR.S              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MR.S,ALPCTL/WX_S"
;4512  R[]_(M[] RB).XZ                "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/XZ.MR,ALPCTL/WX_S"
;4513  R[]_(M[]+CONX(2)).SR.1         "RSRC/@1,SPW/RLONG,ALU/A+B+CI.SR,MUX/M.S,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4514  R[]_(M[]+RB).BCD               "RSRC/@1,SPW/RLONG,ALU/A+B+CI.BCD,MUX/M.R1,VSIZE/1,DTYPE/LONG,MSRC/@2"
;4515  R[]_(M[]+RB+ALKC).BCD          "RSRC/@1,SPW/RLONG,ALU/A+B+CI.BCD,ALUCI/ALKC,MUX/M.R1,VSIZE/1,DTYPE/LONG,MSRC/@2"
;4516  R[]_(M[]-RB).BCD               "RSRC/@1,SPW/RLONG,ALU/A-B-CI.BCD,MUX/M.R1,MSRC/@2,VSIZE/1,DTYPE/LONG"
;4517  R[]_(M[]-RB-ALKC).BCD          "RSRC/@1,SPW/RLONG,ALU/A-B-CI.BCD,ALUCI/ALKC,MUX/M.R1,MSRC/@2,VSIZE/1,DTYPE/LONG"
;4518  R[]_(M[].AND.RB).RL.1          "RSRC/@1,SPW/RLONG,ALU/AND.SL,MUX/M.R1,MSRC/@2,ALUSHF/ROT"
;4519  R[]_(RB M[]).RL.4              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RL.RM.4,ALPCTL/WX_S"
;4520  R[]_(RB M[]).RL.P              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RL.RM.P,ALPCTL/WX_S"
;4521  R[]_(RB+Q).SL.1                "RSRC/@1,SPW/RLONG,ALU/A+B+CI.SL,MUX/R.Q,ALUSHF/ZERO"
;4522  R[]_(RB-M[]<3-0>)              "RSRC/@1,SPW/RLONG,ALU/A-B-CI,MUX/R.S,MSRC/@2,ROT/GETNIB"
;4523  R[]_(RB.OR.M[]<3-0>)           "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,MSRC/@2,ROT/GETNIB"
;4524  R[]_-1                         "RSRC/@1,SPW/RLONG,ROT/MINUS1,ALPCTL/WX_S"
;4525  R[]_-D                         "RSRC/@1,SPW/RLONG,ROT/ZERO,MUX/D.S,ALU/B-A-CI"
;4526  R[]_-M[]                       "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/B-A-CI"
;4527  R[]_-RB                        "RSRC/@1,SPW/RLONG,ALU/B-A-CI,MUX/R.S,ROT/ZERO"
;4528  R[]_-ZEXT(M[])                 "RSRC/@1,SPW/RLONG,MUX/XM.S,ROT/ZERO,ALU/B-A-CI,ALUXM/ZERO,MSRC/@2"
;4529  R[]_0                          "RSRC/@1,SPW/RLONG,ROT/ZERO,ALPCTL/WX_S"
;4530  R[]_8                          "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,ALU/A+B+CI.SL,MUX/Z.S"
;4531  R[]_CONX(1)                    "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;4532  R[]_CONX(2)                    "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;4533  R[]_D                          "RSRC/@1,SPW/RLONG,ROT/ZERO,MUX/D.S,ALU/OR"
;4534  R[]_D+CONX(1)                  "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4535  R[]_D+CONX(2)                  "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4536  R[]_D+CONX(4)                  "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4537  R[]_D+RB                       "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/D.R1"
;4538  R[]_D+RB+ALKC                  "RSRC/@1,SPW/RLONG,MUX/D.R1,ALU/A+B+CI,ALUCI/ALKC"
;4539  R[]_D-0                        "RSRC/@1,SPW/RLONG,ROT/ZERO,MUX/D.S,ALU/A-B-CI"
;4540  R[]_D.NOTAND.RB                "RSRC/@1,SPW/RLONG,ALU/NOTAND,MUX/D.R1"
;4541  R[]_D.OR.(M[].RL.8)            "RSRC/@1,SPW/RLONG,ALU/OR,MUX/D.S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG,MSRC/@2"
;4542  R[]_D.OR.RB                    "RSRC/@1,SPW/RLONG,ALU/OR,MUX/D.R1"
;4543  R[]_D_(D+RB).SL.1 Q<0>_1       "RSRC/@1,SPW/RLONG,DQ3/SQL.D_WX,ALU/A+B+CI.SL,MUX/D.R2,ALUSHF/ALU0.Q1"
;4544  R[]_D_-D                       "RSRC/@1,SPW/RLONG,ROT/ZERO,MUX/D.S,ALU/B-A-CI,DQ1/D_WX"
;4545  R[]_D_-M[]                     "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/B-A-CI,DQ1/D_WX"
;4546  R[]_D_0                        "RSRC/@1,SPW/RLONG,ROT/ZERO,ALPCTL/WX_D_S"
;4547  R[]_D_D+CONX(4)                "RSRC/@1,SPW/RLONG,DQ1/D_WX,ALU/A+B+CI,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/D.S"
;4548  R[]_D_D+Q+1                    "RSRC/@1,SPW/RLONG,DQ1/D_WX,ALU/A+B+CI,MUX/D.Q1,ALUCI/ONE"
;4549  R[]_D_D-1                      "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/D.S,ALU/A+B+CI,DQ1/D_WX"
;4550  R[]_D_D-CONX(4)                "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/D.S,ALU/A-B-CI,DQ1/D_WX"
;4551  R[]_D_M[]                      "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@2,DQ1/D_WX"
;4552  R[]_D_M[]-0                    "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/A-B-CI,DQ1/D_WX"
;4553  R[]_D_M[]-1                    "RSRC/@1,SPW/RLONG,DQ1/D_WX,MSRC/@2,ROT/MINUS1,MUX/M.S,ALU/A+B+CI"
;4554  R[]_D_Q Q_D                    "RSRC/@1,SPW/RLONG,ALPCTL/WX_D_Q.Q_D"
;4555  R[]_D_Q_-1                     "RSRC/@1,SPW/RLONG,ALPCTL/WX_D_Q_S,ROT/MINUS1"
;4556  R[]_D_Q_RNUM_-1                "RSRC/@1,SPW/RLONG,ALPCTL/WX_D_Q_S,MSRC/RNUM_WBUS,ROT/MINUS1"
;4557  R[]_D_RB+1                     "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A-B-CI,DQ1/D_WX"
;4558  R[]_D_RB+CONX(4)               "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,DQ1/D_WX"
;4559  R[]_D_RB-1                     "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A+B+CI,DQ1/D_WX"
;4560  R[]_D_RB-CONX(4)               "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/R.S,ALU/A-B-CI,DQ1/D_WX"
;4561  R[]_D_ZEXT(M[])                "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/XM.S,ALU/OR,DQ1/D_WX"
;4562  R[]_EXP(M[])                   "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/GETEXP,ALPCTL/WX_S"
```

MACROS (CONT)

```
;4563   R[]_FLAGS                   "RSRC/@1,SPW/RLONG,WCTRL/CM.TP.FPD.FLAGS"
;4564   R[]_FLAGS_0                 "RSRC/@1,SPW/RLONG,WCTRL/FLAGS_WB,ROT/ZERO,MUX/R.S,ALU/AND"
;4565   R[]_FPA                     "RSRC/@1,SPW/RLONG,FPA/WBUS_FPA"
;4566   R[]_MEMSCR                  "RSRC/@1,SPW/RLONG,WCTRL/MEMSCR"
;4567   R[]_M[]                     "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@2"
;4568   R[]_M[]+(RB.ASL.SIZ)        "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASL.R.SIZ,MUX/M.S,ALU/A+B+CI"
;4569   R[]_M[]+1                   "RSRC/@1,SPW/RLONG,ALU/A-B-CI,MUX/M.S,MSRC/@2,ROT/MINUS1"
;4570   R[]_M[]+CONX(1)             "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,MUX/M.S,ALU/A+B+CI"
;4571   R[]_M[]+CONX(2)             "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD,MUX/M.S,ALU/A+B+CI"
;4572   R[]_M[]+CONX(4)             "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/M.S,ALU/A+B+CI"
;4573   R[]_M[]+PL                  "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/PL,MUX/M.S,ALU/A+B+CI"
;4574   R[]_M[]+Q                   "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.Q1,ALU/A+B+CI"
;4575   R[]_M[]+Q+1                 "RSRC/@1,SPW/RLONG,ALU/A+B+CI,ALUCI/ONE,MUX/M.Q1,MSRC/@2"
;4576   R[]_M[]+RB                  "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/M.R1,MSRC/@2"
;4577   R[]_M[]+RB+ALKC             "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/ALKC"
;4578   R[]_M[]+RB+PSLC             "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/PSLC"
;4579   R[]_M[]+SL                  "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/SL,MUX/M.S,ALU/A+B+CI"
;4580   R[]_M[]-0                   "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/A-B-CI"
;4581   R[]_M[]-1                   "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/M.S,MSRC/@2,ROT/MINUS1"
;4582   R[]_M[]-CONX(2)             "RSRC/@1,SPW/RLONG,MSRC/@2,VSIZE/1,DTYPE/WORD,ROT/CONX.SIZ,MUX/M.S,ALU/A-B-CI"
;4583   R[]_M[]-CONX(4)             "RSRC/@1,SPW/RLONG,MSRC/@2,VSIZE/1,DTYPE/LONG,ROT/CONX.SIZ,MUX/M.S,ALU/A-B-CI"
;4584   R[]_M[]-Q                   "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.Q1,ALU/A-B-CI"
;4585   R[]_M[]-RB                  "RSRC/@1,SPW/RLONG,ALU/A-B-CI,MUX/M.R1,MSRC/@2"
;4586   R[]_M[].AND.RB              "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/AND"
;4587   R[]_M[].ANDNOT.Q            "RSRC/@1,SPW/RLONG,MSRC/@2,ALU/ANDNOT,MUX/M.Q1"
;4588   R[]_M[].ASL.P               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASL.M.P,ALPCTL/WX_S"
;4589   R[]_M[].ASR.-P              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASR.M.-P,ALPCTL/WX_S"
;4590   R[]_M[].ASR.3               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASR.M.3,ALPCTL/WX_S"
;4591   R[]_M[].ASR.P               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASR.M.P,ALPCTL/WX_S"
;4592   R[]_M[].BCDSWP              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/BCDSWP,ALPCTL/WX_S"
;4593   R[]_M[].CLR1B               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CLR1BM,ALPCTL/WX_S"
;4594   R[]_M[].CLR2B               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CLR2BM,ALPCTL/WX_S"
;4595   R[]_M[].CLR3B               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CLR3BM,ALPCTL/WX_S"
;4596   R[]_M[].FPLIT               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/FPLIT,ALPCTL/WX_S"
;4597   R[]_M[].NIBBLE              "RSRC/@1,SPW/RLONG,ROT/GETNIB,MSRC/@2,ALPCTL/WX_S"
;4598   R[]_M[].OR.(RB.RR.16)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,MSRC/@2,VSIZE/1,DTYPE/WORD"
;4599   R[]_M[].OR.(RB.RL.16)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,ROT/RR.RR.SIZ,MSRC/@2,VSIZE/1,DTYPE/WORD"
;4600   R[]_M[].OR.(RB.RR.SIZ)      "RSRC/@1,SPW/RLONG,ROT/RR.RR.SIZ,MSRC/@2,ALU/OR,MUX/M.S"
;4601   R[]_M[].OR.Q                "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.Q1,MSRC/@2"
;4602   R[]_M[].OR.RB               "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/OR"
;4603   R[]_M[].RL.24               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;4604   R[]_M[].RL.9                "RSRC/@1,SPW/RLONG,MSRC/@2,ALPCTL/WX_S,ROT/RL.MM.PTE"
;4605   R[]_M[].RL.P                "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RL.MM.P,ALPCTL/WX_S"
;4606   R[]_M[].RR.16               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;4607   R[]_M[].RR.24               "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG,MSRC/@2"
;4608   R[]_M[].RR.8                "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,MSRC/@2"
;4609   R[]_M[].RR.P                "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MM.P,ALPCTL/WX_S"
;4610   R[]_M[].RR.PS               "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/RR.MM.PS,ALPCTL/WX_S"
;4611   R[]_M[].SR.1                "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.S,ROT/ZERO,ALU/A+B+CI.SR"
;4612   R[]_M[].XOR.RB              "RSRC/@1,SPW/RLONG,ALU/XOR,MUX/M.R1,MSRC/@2"
;4613   R[]_M[].XZ                  "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/XZ.MM,MSRC/@2"
;4614   R[]_M[]<3-0>+RB             "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,MSRC/@2,ROT/GETNIB"
;4615   R[]_NOT(M[]).AND.RB         "RSRC/@1,SPW/RLONG,ALU/NOTAND,MUX/M.R1,MSRC/@2"
;4616   R[]_NOT(M[].ASR.-P)         "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASR.M.-P,ALPCTL/WX_.NOT.S"
;4617   R[]_PL_M[]-RB               "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/A-B-CI,ROT/SL.PL_WB"
```

```
;4618   R[]_PL_Q Q_M[]              "RSRC/@1,SPW/RLONG,ROT/SL.PL_WB,MSRC/@2,ALPCTL/WX_Q.Q_M"
;4619   R[]_PL_Q-M[]                "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.Q1,ALU/B-A-CI,ROT/SL.PL_WB"
;4620   R[]_PSL                     "RSRC/@1,SPW/RLONG,CCPSL/WB_PSL.CCBR_SIGND"
;4621   R[]_Q Q_D                   "RSRC/@1,SPW/RLONG,ALPCTL/WX_Q.Q_D"
;4622   R[]_Q Q_M[]                 "RSRC/@1,SPW/RLONG,MSRC/@2,ALPCTL/WX_Q.Q_M"
;4623   R[]_Q-M[]                   "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.Q1,ALU/B-A-CI"
;4624   R[]_Q_((M[]+Q).SL.1).OR.1   "RSRC/@1,SPW/RLONG,DQ1/Q_WX,ALU/A+B+CI.SL,MUX/M.Q1,MSRC/@2,ALUSHF/ONE"
;4625   R[]_Q_0                     "RSRC/@1,SPW/RLONG,ROT/ZERO,ALPCTL/WX_Q_S"
;4626   R[]_Q_CONX(1)               "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_Q_S"
;4627   R[]_Q_CONX(4)               "RSRC/@1,SPW/RLONG;ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,ALPCTL/WX_Q_S"
;4628   R[]_Q_D                     "RSRC/@1,SPW/RLONG,ROT/ZERO,MUX/D.S,ALU/OR,DQ1/Q_WX"
;4629   R[]_Q_D+RB+ALKC             "RSRC/@1,SPW/RLONG,DQ1/Q_WX,MUX/D.R1,ALU/A+B+CI,ALUCI/ALKC"
;4630   R[]_Q_D_M[]                 "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/OR,DQ1/Q_D_WX"
;4631   R[]_Q_EXP(M[])              "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/GETEXP,ALU/A+B+CI,MUX/Z.S,DQ1/Q_WX"
;4632   R[]_Q_M[]                   "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/OR,DQ1/Q_WX"
;4633   R[]_Q_M[].XZ.MM             "RSRC/@1,SPW/RLONG,ALPCTL/WX_Q_S,MSRC/@2,ROT/XZ.MM"
;4634   R[]_Q_Q-M[]                 "RSRC/@1,SPW/RLONG,DQ1/Q_WX,ALU/B-A-CI,MUX/M.Q1,MSRC/@2"
;4635   R[]_Q_RB+Q                  "RSRC/@1,SPW/RLONG,DQ1/Q_WX,MUX/R.Q,ALU/A+B+CI"
;4636   R[]_Q_ZEXT(M[])             "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/XM.S,ALU/OR,DQ1/Q_WX"
;4637   R[]_RB+1                    "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A-B-CI"
;4638   R[]_RB+CONX(1)              "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4639   R[]_RB+CONX(2)              "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A+B+CI,VSIZE/1,DTYPE/WORD"
;4640   R[]_RB+CONX(4)              "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4641   R[]_RB+CONX.SIZ             "RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/IDEP"
;4642   R[]_RB+Q                    "RSRC/@1,SPW/RLONG,MUX/R.Q,ALU/A+B+CI"
;4643   R[]_RB-1                    "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A+B+CI"
;4644   R[]_RB-CONX(2)              "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,VSIZE/1,DTYPE/WORD"
;4645   R[]_RB-CONX(4)              "RSRC/@1,SPW/RLONG,ALU/A-B-CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4646   R[]_RB-CONX.SIZ             "RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI"
;4647   R[]_RB-D-ALKC               "RSRC/@1,SPW/RLONG,MUX/D.R1,ALU/B-A-CI,ALUCI/ALKC"
;4648   R[]_RB-M[]                  "RSRC/@1,SPW/RLONG,ALU/B-A-CI,MUX/M.R1,MSRC/@2"
;4649   R[]_RB-M[]-PSLC             "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/B-A-CI,ALUCI/PSLC"
;4650   R[]_RB-Q                    "RSRC/@1,SPW/RLONG,MUX/R.Q,ALU/A-B-CI"
;4651   R[]_RB.AND.(M[].RR.16)      "RSRC/@1,SPW/RLONG,ALU/AND,MUX/R.S,ROT/RR.MM.SIZ,MSRC/@2,VSIZE/1,DTYPE/WORD"
;4652   R[]_RB.AND.(M[].RR.8)       "RSRC/@1,SPW/RLONG,ALU/AND,MUX/R.S,ROT/RR.MM.SIZ,MSRC/@2,VSIZE/1,DTYPE/BYTE"
;4653   R[]_RB.ANDNOT.Q             "RSRC/@1,SPW/RLONG,MUX/R.Q,ALU/ANDNOT"
;4654   R[]_RB.ASL.1                "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/WORD"
;4655   R[]_RB.ASL.2                "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/ASL.R.SIZ,VSIZE/1,DTYPE/LONG"
;4656   R[]_RB.ASL.7                "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/ASL.R.7"
;4657   R[]_RB.ASL.SIZ              "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/ASL.R.SIZ"
;4658   R[]_RB.OR.(M[].ASL.P)       "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ASL.M.P,MUX/R.S,ALU/OR"
;4659   R[]_RB.OR.(M[].RL.16)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,ROT/RR.MM.SIZ,MSRC/@2,VSIZE/1,DTYPE/WORD"
;4660   R[]_RB.OR.(M[].RL.24)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,ROT/RR.MM.SIZ,MSRC/@2,VSIZE/1,DTYPE/BYTE"
;4661   R[]_RB.OR.(M[].RL.8)        "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,ROT/RR.MM.SIZ,MSRC/@2,VSIZE/1,DTYPE/LONG"
;4662   R[]_RB.OR.(M[].RL.9)        "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/R.S,ALU/OR,ROT/RL.MM.PTE"
;4663   R[]_RB.OR.(M[].RR.16)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
;4664   R[]_RB.OR.(M[].RR.24)       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG"
;4665   R[]_RB.OR.(M[].RR.8)        "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,MSRC/@2,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE"
;4666   R[]_RB.OR.(M[].RR.SIZ)      "RSRC/@1,SPW/RLONG,ROT/RR.MM.SIZ,MUX/R.S,ALU/OR,MSRC/@2"
;4667   R[]_RB.OR.(M[].XZ)          "RSRC/@1,SPW/RLONG,ROT/XZ.MM,MUX/R.S,ALU/OR,MSRC/@2"
;4668   R[]_RB.OR.CONX(1)           "RSRC/@1,SPW/RLONG,ALU/OR,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4669   R[]_RB.OR.Q                 "RSRC/@1,SPW/RLONG,MUX/R.Q,ALU/OR"
;4670   R[]_RB.OR.ZEXT(M[])         "RSRC/@1,SPW/RLONG,MSRC/@2,MUX/XM.R,ALUXM/ZERO,ALU/OR"
;4671   R[]_RB.RL.16                "RSRC/@1,SPW/RLONG,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;4672   R[]_RB.RL.8                 "RSRC/@1,SPW/RLONG,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG,ALPCTL/WX_S"
```

MACROS (CONT)

```
;4673   R[]_RB.RL.P                          "RSRC/@1,SPW/RLONG,ROT/RL.RR.P,ALPCTL/WX_S"
;4674   R[]_RB.RR.24                         "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG"
;4675   R[]_RB.RR.8                          "RSRC/@1,SPW/RLONG,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/BYTE,ALPCTL/WX_S"
;4676   R[]_RB.RR.P                          "RSRC/@1,SPW/RLONG,ROT/RR.RR.P,ALPCTL/WX_S"
;4677   R[]_RB.XOR.MINUS1                    "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/XOR"
;4678   R[]_RB.XZ                            "RSRC/@1,SPW/RLONG,ALPCTL/WX_S,ROT/XZ.RR"
;4679   R[]_RBSP                             "RSRC/@1,SPW/RLONG,MSRC/WB_RBSP,ALPCTL/NOP"
;4680   R[]_RNUM                             "RSRC/@1,SPW/RLONG,ALPCTL/NOP,MSRC/WBUS_RNUM"
;4681   R[]_RNUM_MTEMPO.XZ                   "RSRC/@1,SPW/RLONG,MSRC/RNUM_WBUS,ALPCTL/WX_S,ROT/XZ.MM"
;4682   R[]_SEXT(M[])                        "RSRC/@1,SPW/RLONG,MSRC/@2,ALUXM/SIGN,MUX/XM.S,ROT/MINUS1,ALU/AND"
;4683   R[]_TB                               "RSRC/@1,SPW/RLONG,MSRC/TB,BUS/PRB.RD,VSIZE/1,ALU/OR,MUX/M.S,ROT/ZERO"
;4684   R[]_VA_M[]                           "RSRC/@1,SPW/RLONG,MSRC/@2,ROT/ZERO,MUX/M.S,ALU/OR,WCTRL/VA_WB"
;4685   R[]_VA_RB+1                          "RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A-B-CI,WCTRL/VA_WB"
;4686   R[]_VA_RB-Q                          "RSRC/@1,SPW/RLONG,MUX/R.Q,ALU/A-B-CI,WCTRL/VA_WB"
;4687   R[]_WDR                              "RSRC/@1,SPW/RLONG,WCTRL/MBUS_WDR,MSRC/WDR,ALU/OR,MUX/M.S,ROT/ZERO"
;4688   R[]_XB PC_PC+1                       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,MSRC/XB.PC_PC+I,ROT/ZERO,ISTRM/ISIZE_DSIZE,
;4689                                                          VSIZE/1,DTYPE/BYTE"
;4690   R[]_XB PC_PC+4                       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,MSRC/XB.PC_PC+I,ROT/ZERO,ISTRM/ISIZE_DSIZE,
;4691                                                          VSIZE/1,DTYPE/LONG"
;4692   R[]_XB PC_PC+I                       "RSRC/@1,SPW/RLONG,ALU/OR,MUX/M.S,MSRC/XB.PC_PC+I,ROT/ZERO,ISTRM/ISIZE_DSIZE,
;4693                                                          VSIZE/1,DTYPE/IDEP"
;4694   R[]_ZEXT(M[])                        "RSRC/@1,SPW/RLONG,ALU/OR,MUX/XM.S,MSRC/@2,ROT/ZERO"
;4695   R[]_ZEXT(M[]).SL.1                   "RSRC/@1,SPW/RLONG,ALU/A+B+CI.SL,MUX/XM.S,ALUXM/ZERO,MSRC/@2,ROT/ZERO"
;4696   R[]_ZEXT(XB) PC_PC+1                 "RSRC/@1,SPW/RLONG,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/XM.S,ALU/OR,ISTRM/ISIZE_DSIZE,
;4697                                                          VSIZE/1,DTYPE/BYTE"
;4698   R[]_ZEXT(XB) PC_PC+2                 "RSRC/@1,SPW/RLONG,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/XM.S,ALU/OR,ISTRM/ISIZE_DSIZE,
;4699                                                          VSIZE/1,DTYPE/WORD"
;4700
;4701   SL_[]                                "ROT/OLITO.SL_LIT,LIT/LITRL,LITRL/@1"
;4702   SOFTIPR_0                            "WCTRL/SOFTIPR_WB,ROT/ZERO,ALPCTL/WX_S"
;4703   SOFTIPR_M[].RR.16                    "WCTRL/SOFTIPR_WB,ALPCTL/WX_S,MSRC/@1,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD"
;4704   SPICR_SPNICR                         "RSRC/SPNICR.SPICR,SPW/RSIZE,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD,ALPCTL/WX_S"
;4705
;4706   STEPC_(M[]+2).SR.1                   "WCTRL/STEPC_WB,ALU/A+B+CI.SR,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/2"
;4707   STEPC_(M[]+ZLITO[]).SR.1             "WCTRL/STEPC_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,ALU/A+B+CI.SR,MUX/M.S"
;4708   STEPC_(SEXT(M[])+2).SR.1             "WCTRL/STEPC_WB,ALU/A+B+CI.SR,MUX/XM.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/2"
;4709   STEPC_(ZEXT(M[])+CONX(2)).SR.1       "WCTRL/STEPC_WB,ALU/A+B+CI.SR,MUX/XM.S,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD,ALUXM/ZERO"
;4710   STEPC_14.                            "MISC/SC_14"
;4711   STEPC_2                              "MISC/SC_2"
;4712   STEPC_30.                            "MISC/SC_30"
;4713   STEPC_6                              "MISC/SC_6"
;4714   STEPC_D_(M[]+CONX(2)).SR.1           "WCTRL/STEPC_WB,DQ1/D_WX,ALU/A+B+CI.SR,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4715   STEPC_D_M[]-ZLITO[]                  "WCTRL/STEPC_WB,DQ1/D_WX,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/A-B-CI"
;4716   STEPC_D_M[].SR.1                     "WCTRL/STEPC_WB,DQ1/D_WX,ALU/A+B+CI.SR,MUX/M.R1,MSRC/@1,RSRC/ZERO"
;4717   STEPC_D_RNUM_ZLITO[]                 "WCTRL/STEPC_WB,ALPCTL/WX_D_S,MSRC/RNUM_WBUS,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4718   STEPC_D_SEXT(M[]).SR.1               "WCTRL/STEPC_WB,DQ1/D_WX,ALU/A+B+CI.SR,MUX/XM.R,ALUXM/SIGN,MSRC/@1,RSRC/ZERO"
;4719   STEPC_M[]+ZLITO[]                    "WCTRL/STEPC_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,ALU/A+B+CI,MUX/M.S"
;4720   STEPC_M[]-R[]                        "WCTRL/STEPC_WB,MSRC/@1,RSRC/@2,ALU/A-B-CI,MUX/M.R1"
;4721   STEPC_M[]-ZLITO[]                    "WCTRL/STEPC_WB,MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLITO,ALU/A-B-CI,MUX/M.S"
;4722   STEPC_M[].SR.1                       "WCTRL/STEPC_WB,ALU/A+B+CI.SR,MUX/M.S,MSRC/@1,ROT/ZERO"
;4723   STEPC_M[]<3-0>.                      "WCTRL/STEPC_WB,MSRC/@1,ROT/GETNIB,ALPCTL/WX_S"
;4724   STEPC_M[]_R[].RR.P                   "WCTRL/STEPC_WB,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,RSRC/@2,ROT/RR.RR.P"
;4725   STEPC_M[]_ZLITO[]                    "WCTRL/STEPC_WB,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4726   STEPC_M[]_ZLIT28[]                   "WCTRL/STEPC_WB,MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLIT28,ALPCTL/WX_S"
;4727   STEPC_PL                             "WCTRL/STEPC_WB,ROT/PL,ALPCTL/WX_S"
```

MACROS (CONT)

```
STEPC_Q_M[].SR.1                   "WCTRL/STEPC_WB,DQ1/Q_WX,ALU/AND.SR,MUX/M.S,MSRC/@1,ROT/MINUS1"
STEPC_Q_R[].SR.1                   "WCTRL/STEPC_WB,DQ1/Q_WX,ALU/AND.SR,MUX/R.S,RSRC/@1,ROT/MINUS1"
STEPC_R[].RR.P                     "WCTRL/STEPC_WB,RSRC/@1,ROT/RR.RR.P,ALPCTL/WX_S"
STEPC_R[].SR.1                     "WCTRL/STEPC_WB,ALU/A+B+CI.SR,MUX/R.S,RSRC/@1,ROT/ZERO"
STEPC_ZLITO[]                      "WCTRL/STEPC_WB,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"

TB_BAD PARITY                      "WCTRL/TB_WB,MISC/FORCE.TB,MSRC/VA,ALPCTL/WX_S,ROT/MINUS1"
TB_D+ZLIT8[]                       "WCTRL/TB_WB,ALU/A+B+CI,MUX/D.S,ROT/ZLIT8,LIT/LITRL,LITRL/@1,MSRC/VA"
TB_R[]                             "WCTRL/TB_WB,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO,MSRC/VA"

TCSR_0                             "WCTRL/TCSR_WB,ROT/ZERO,ALPCTL/WX_S"
TCSR_M[].AND.ZLIT16[]              "WCTRL/TCSR_WB,ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
TCSR_M[].OR.R[]                    "WCTRL/TCSR_WB,ALU/OR,MUX/M.R1,MSRC/@1,RSRC/@2"

TOYCR_D.XOR.ZLIT16[]               "WCTRL/TODCLK_WB,ALU/XOR,MUX/D.S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
TOYCR_D_(M[].XZ).OR.R[]            "WCTRL/TODCLK_WB,DQ1/D_WX,ALU/OR,MUX/R.S,RSRC/@2,ROT/XZ.MM,MSRC/@1"
TOYCR_M[].OR.R[]                   "WCTRL/TODCLK_WB,ALU/OR,MUX/M.R1,MSRC/@1,RSRC/@2"
TOYCR_M[].OR.ZLIT16[]              "WCTRL/TODCLK_WB,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
TOYCR_M[]_MB-ZLIT16[]              "WCTRL/TODCLK_WB,MSRC/@1,SPW/MLONG,ALU/A-B-CI,MUX/M.S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
TOYCR_M[]_ZLIT16[]                 "WCTRL/TODCLK_WB,SPW/MLONG,MSRC/@1,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
TRAR_ZLIT16[]                      "WCTRL/LOADTRAR,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"

TU58REGS_M[].AND.OLIT16[]          "WCTRL/TU58WRITE,MSRC/@1,ROT/OLIT16,LIT/LITRL,LITRL/@2,ALU/AND,MUX/M.S"
TU58REGS_M[].OR.ZLIT16[]           "WCTRL/TU58WRITE,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2,ALU/OR,MUX/M.S"
TU58REGS_M[].RR.16                 "WCTRL/TU58WRITE,ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,MSRC/@1"
TU58REGS_M[]_MB.RL.8               "WCTRL/TU58WRITE,MSRC/@1,SPW/MLONG,ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG"
TU58REGS_R[]_0                     "WCTRL/TU58WRITE,SPW/RLONG,RSRC/@1,ALPCTL/WX_S,ROT/ZERO"
TU58REGS_ZLIT16[]                  "WCTRL/TU58WRITE,ALPCTL/WX_S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"

VA_D+R[]                           "WCTRL/VA_WB,ALU/A+B+CI,MUX/D.R1,RSRC/@1"
VA_D+R[]+1                         "WCTRL/VA_WB,ALU/A+B+CI,MUX/D.R1,RSRC/@1,ALUCI/ONE"
VA_D+ZLITO[]                       "WCTRL/VA_WB,ALU/A+B+CI,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
VA_D-ZLITO[]                       "WCTRL/VA_WB,LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/A-B-CI"
VA_D.XOR.ZLIT16[]                  "WCTRL/VA_WB,ALU/XOR,MUX/D.S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
VA_D_D+ZLIT8[] INVALIDATE TB       "WCTRL/CLRTB.VA_WB,DQ1/D_WX,ALU/A+B+CI,MUX/D.S,ROT/ZLIT8,LIT/LITRL,LITRL/@1,BUS/PRB.RD.PTE"
VA_D_M[]+R[]                       "WCTRL/VA_WB,DQ1/D_WX,ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2"
VA_D_M[]-ZLITO[]                   "WCTRL/VA_WB,DQ1/D_WX,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
VA_D_M[]_R[]                       "WCTRL/VA_WB,DQ1/D_WX,MSRC/@1,SPW/MLONG,RSRC/@2,MUX/R.S,ROT/ZERO,ALU/OR"
VA_D_ZLITO[]                       "WCTRL/VA_WB,ALPCTL/WX_D_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
VA_D_ZLIT24[]                      "WCTRL/VA_WB,ALPCTL/WX_D_S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
VA_MTEMPO_D_VA+ZLIT8[]             "WCTRL/VA_WB,SPW/MLONG,DQ1/D_WX,MSRC/VA,ROT/ZLIT8,LIT/LITRL,LITRL/@1,MUX/M.S,ALU/A+B+CI"
VA_MTEMPO_PC                       "WCTRL/VA_WB,MSRC/PC,SPW/MLONG,RSRC/ZERO,MUX/M.R1,ALU/OR"
VA_MTEMPO_PC+SEXT(XB)+2 PC_PC+2    "WCTRL/VA_PC+I+W.PC_PC+I,MSRC/XB.PC_PC+I,SPW/MLONG,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR,
                                                           VSIZE/1,DTYPE/WORD,ISTRM/ISIZE_DSIZE"
VA_MTEMPO_VA.AND.OLITO[]           "WCTRL/VA_WB,MSRC/VA,ROT/OLITO,LIT/LITRL,LITRL/@1,MUX/M.S,ALU/AND,SPW/MLONG"
VA_MTEMPO_SEXT(XB)+R[] PC_PC+2     "WCTRL/VA_WB,MSRC/XB.PC_PC+I,SPW/MLONG,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI,
                                                    VSIZE/1,DTYPE/WORD,ISTRM/ISIZE_DSIZE"
VA_MTEMPO_ZEXT(MDR)                "WCTRL/VA_WB,MSRC/MDR,SPW/MLONG,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR"
VA_MTEMPO_ZEXT(XB) PC_PC+2         "WCTRL/VA_WB,MSRC/XB.PC_PC+I,SPW/MLONG,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR,
                                                    ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/WORD"

VA_M[]                             "WCTRL/VA_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR"
VA_M[]+(R[].ASL.SIZE)              "WCTRL/VA_WB,MSRC/@1,RSRC/@2,ROT/ASL.R.SIZ,MUX/M.S,ALU/A+B+CI"
VA_M[]-(R[].RR.8)                  "WCTRL/VA_WB,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/RR.RR.SIZ,RSRC/@2,VSIZE/1,DTYPE/BYTE"
VA_M[]+CONX(1)                     "WCTRL/VA_WB,MSRC/@1,VSIZE/1,DTYPE/BYTE,ROT/CONX.SIZ,MUX/M.S,ALU/A+B+CI"
VA_M[]+CONX(2)                     "WCTRL/VA_WB,MSRC/@1,VSIZE/1,DTYPE/WORD,ROT/CONX.SIZ,MUX/M.S,ALU/A+B+CI"
```

```
;4783   VA_M[]+CONX(4)                   "WCTRL/VA_WB,MSRC/@1,VSIZE/1,DTYPE/LONG,ROT/CONX.SIZ,MUX/M.S,ALU/A+B+CI"
;4784   VA_M[]+Q+1                       "WCTRL/VA_WB,ALU/A+B+CI,ALUCI/ONE,MUX/M.Q1,MSRC/@1"
;4785   VA_M[]+R[]                       "WCTRL/VA_WB,ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2"
;4786   VA_M[]+R[]+1                     "WCTRL/VA_WB,ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2,ALUCI/ONE"
;4787   VA_M[]+ZLITO[]                   "WCTRL/VA_WB,ALU/A+B+CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4788   VA_M[]+ZLIT8[]                   "WCTRL/VA_WB,MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2,MUX/M.S,ALU/A+B+CI"
;4789   VA_M[]-(R[].ASL.SIZE)            "WCTRL/VA_WB,MSRC/@1,RSRC/@2,ROT/ASL.R.SIZ,MUX/M.S,ALU/A-B-CI"
;4790   VA_M[]-R[]                       "WCTRL/VA_WB,ALU/A-B-CI,MUX/M.R1,MSRC/@1,RSRC/@2"
;4791   VA_M[]-ZLITO[]                   "WCTRL/VA_WB,ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4792   VA_M[].AND.ZLITO[]               "WCTRL/VA_WB,ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4793   VA_M[].ANDNOT.ZLITO[]            "WCTRL/VA_WB,MSRC/@1,ROT/ZLITO,MUX/M.S,ALU/ANDNOT,LITRL/@2,LIT/LITRL"
;4794   VA_M[].OR.Q                      "WCTRL/VA_WB,ALU/OR,MUX/M.Q1,MSRC/@1"
;4795   VA_M[].XOR.R[]                   "WCTRL/VA_WB,ALU/XOR,MUX/M.R1,MSRC/@1,RSRC/@2"
;4796   VA_M[].XOR.ZLITO[]               "WCTRL/VA_WB,ALU/XOR,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4797   VA_M[].XOR.ZLIT24[]              "WCTRL/VA_WB,ALU/XOR,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4798   VA_M[]_MB+ZLITO[]                "WCTRL/VA_WB,MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4799   VA_M[]_MB+ZLIT8[] CLEAR CACHE    "WCTRL/CLRCH.VA_WB,MSRC/@1,SPW/MLONG,ALU/A+B+CI,MUX/M.S,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4800   VA_M[]_MB-1                      "WCTRL/VA_WB,MSRC/@1,SPW/MLONG,MUX/M.R1,ALU/A-B-CI,ALUCI/ONE,RSRC/ZERO"
;4801   VA_M[]_MB-ZLITO[]                "WCTRL/VA_WB,MSRC/@1,SPW/MLONG,LIT/LITRL,LITRL/@2,ROT/ZLITO,MUX/M.S,ALU/A-B-CI"
;4802   VA_M[]_MB.AND.R[]                "WCTRL/VA_WB,MSRC/@1,SPW/MLONG,RSRC/@2,MUX/M.R1,ALU/AND"
;4803   VA_M[]_Q                         "WCTRL/VA_WB,MSRC/@1,SPW/MLONG,RSRC/ZERO,MUX/R.Q,ALU/OR"
;4804   VA_M[]_R[]                       "WCTRL/VA_WB,MSRC/@1,RSRC/@2,ROT/ZERO,MUX/R.S,ALU/OR,SPW/MLONG"
;4805   VA_M[]_R[]+CONX(4)               "WCTRL/VA_WB,MSRC/@1,RSRC/@2,VSIZE/1,DTYPE/LONG,ROT/CONX.SIZ,MUX/R.S,ALU/A+B+CI,SPW/MLONG"
;4806   VA_M[]_R[]-1                     "WCTRL/VA_WB,MSRC/@1,RSRC/@2,SPW/MLONG,ALU/A+B+CI,MUX/R.S,ROT/MINUS1"
;4807   VA_M[]_R[].OR.Q                  "WCTRL/VA_WB,SPW/MLONG,MSRC/@1,ALU/OR,MUX/R.Q,RSRC/@2"
;4808   VA_PC+SEXT(XB)+2 PC_PC+2         "WCTRL/VA_PC+I+W.PC_PC+I,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR,
;4809                                              VSIZE/1,DTYPE/WORD,ISTRM/ISIZE_DSIZE"
;4810   VA_PC+SEXT(XB)+I PC_PC+I         "WCTRL/VA_PC+I+W.PC_PC+I,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR"
;4811   VA_Q                             "WCTRL/VA_WB,ALU/OR,MUX/R.Q,RSRC/ZERO"
;4812   VA_Q Q_D                         "WCTRL/VA_WB,ALPCTL/WX_Q.Q_D"
;4813   VA_Q_D_D+ZLIT8[]                 "WCTRL/VA_WB,DQ1/Q_D_WX,ROT/ZLIT8,LIT/LITRL,LITRL/@1,MUX/D.S,ALU/A+B+CI"
;4814   VA_Q_D_R[]                       "WCTRL/VA_WB,DQ1/Q_D_WX,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR"
;4815   VA_Q_M[]+R[]                     "WCTRL/VA_WB,MSRC/@1,RSRC/@2,MUX/M.R1,ALU/A+B+CI,DQ1/Q_WX"
;4816   VA_RNUM_R[]                      "WCTRL/VA_WB,RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR,MSRC/RNUM_WBUS"
;4817   VA_RNUM_R[] CLOBBER Q            "WCTRL/VA_WB,RSRC/@1,ALPCTL/WX_R.Q_M,MSRC/RNUM_WBUS"
;4818   VA_R[]                           "WCTRL/VA_WB,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
;4819   VA_R[] Q_D                       "WCTRL/VA_WB,RSRC/@1,ALPCTL/WX_R.Q_D"
;4820   VA_R[] Q_M[]                     "WCTRL/VA_WB,RSRC/@1,ALPCTL/WX_R.Q_M,MSRC/@2"
;4821   VA_R[]+(M[].ASL.P)               "WCTRL/VA_WB,RSRC/@1,MSRC/@2,ROT/ASL.M.P,MUX/R.S,ALU/A+B+CI"
;4822   VA_R[]+1                         "WCTRL/VA_WB,ALU/A-B-CI,MUX/R.S,RSRC/@1,ROT/MINUS1"
;4823   VA_R[]+CONX(4)                   "WCTRL/VA_WB,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4824   VA_R[]+CONX.SIZ                  "WCTRL/VA_WB,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/IDEP"
;4825   VA_R[]+M[].PTX                   "WCTRL/VA_WB,ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/XZ.PTX,MSRC/@2"
;4826   VA_R[]-CONX(1)                   "WCTRL/VA_WB,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE,MUX/R.S,ALU/A-B-CI"
;4827   VA_R[]-CONX(2)                   "WCTRL/VA_WB,RSRC/@1,ALU/A-B-CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4828   VA_R[]-CONX(4)                   "WCTRL/VA_WB,RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/R.S,ALU/A-B-CI"
;4829   VA_R[]-CONX.SIZ                  "WCTRL/VA_WB,RSRC/@1,ALU/A-B-CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/IDEP"
;4830   VA_R[]-M[]                       "WCTRL/VA_WB,ALU/B-A-CI,MUX/M.R1,MSRC/@2,RSRC/@1"
;4831   VA_R[].ASL.SIZE                  "WCTRL/VA_WB,ALPCTL/WX_S,RSRC/@1,ROT/ASL.R.SIZ"
;4832   VA_R[].SIZ_RB-CONX(2)            "WCTRL/VA_WB,RSRC/@1,SPW/RSIZE,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1,MUX/R.S,ALU/A-B-CI"
;4833   VA_R[].SIZ_RB-D                  "WCTRL/VA_WB,RSRC/@1,SPW/RSIZE,MUX/D.R1,ALU/B-A-CI"
;4834   VA_R[].XOR.Q                     "WCTRL/VA_WB,ALU/XOR,MUX/R.Q,RSRC/@1"
;4835   VA_R[]_D-CONX(4)                 "WCTRL/VA_WB,RSRC/@1,VSIZE/1,DTYPE/LONG,ROT/CONX.SIZ,MUX/D.S,ALU/A-B-CI,SPW/RLONG"
;4836   VA_R[]_M[]                       "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALU/OR,MUX/M.S,MSRC/@2,ROT/ZERO"
;4837   VA_R[]_M[]+CONX(1)               "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALU/A+B+CI,MUX/M.S,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
```

```
;4838   VA_R[]_M[]+CONX(2)              "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALU/A+B+CI,MUX/M.S,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/WORD"
;4839   VA_R[]_M[]+CONX(4)              "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALU/A+B+CI,MUX/M.S,MSRC/@2,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG"
;4840   VA_R[]_M[]+RB                   "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,MSRC/@2,MUX/M.R1,ALU/A+B+CI"
;4841   VA_R[]_M[]-CONX(4)              "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,DTYPE/LONG,MUX/M.S,ALU/A-B-CI,VSIZE/1"
;4842   VA_R[]_M[]-CONX.SIZ             "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,MSRC/@2,ROT/CONX.SIZ,MUX/M.S,ALU/A-B-CI,VSIZE/1,DTYPE/IDEP"
;4843   VA_R[]_Q Q_M[]                  "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALPCTL/WX_Q.Q_M,MSRC/@2"
;4844   VA_R[]_RB+1                     "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,ROT/MINUS1,MUX/R.S,ALU/A-B-CI"
;4845   VA_R[]_RB+CONX(1)               "WCTRL/VA_WB,SPW/RLONG,RSRC/@1,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4846   VA_R[]_RB+CONX(4)               "WCTRL/VA_WB,RSRC/@1,ROT/CONX.SIZ,MUX/R.S,ALU/A+B+CI,SPW/RLONG,ALUCI/ZERO,VSIZE/1,
;4847                                           DTYPE/LONG"
;4848   VA_R[]_RB+CONX.SIZ              "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,ALU/A+B+CI,MUX/R.S,ROT/CONX.SIZ"
;4849   VA_R[]_RB-CONX(1)               "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,ALU/A-B-CI,MUX/R.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4850   VA_R[]_RB-CONX(2)               "WCTRL/VA_WB,RSRC/@1,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,VSIZE/1,DTYPE/WORD"
;4851   VA_R[]_RB-CONX(4)               "WCTRL/VA_WB,RSRC/@1,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,SPW/RLONG,ALUCI/ZERO,VSIZE/1,
;4852                                           DTYPE/LONG"
;4853   VA_R[]_RB-CONX.SIZ              "WCTRL/VA_WB,RSRC/@1,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,SPW/RLONG,ALUCI/ZERO,VSIZE/1,
;4854                                           DTYPE/IDEP"
;4855   VA_SEXT(XB)+R[] PC_PC+2         "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI,
;4856                                        VSIZE/1,DTYPE/WORD,ISTRM/ISIZE_DSIZE"
;4857   VA_SEXT(XB)+R[] PC_PC+I         "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/SIGN,ALU/A+B+CI"
;4858   VA_VA+4                         "WCTRL/VA_VA+4"
;4859   VA_VA+4 CLEAR CACHE             "WCTRL/CLRCH.VA_WB,LITRL/4,ROT/ZLITO,LIT/LITRL,MSRC/VA,MUX/M.S,ALU/A+B+CI"
;4860   VA_XB PC_PC+4                   "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/M.R1,ALU/OR,VSIZE/1,DTYPE/LONG,ISTRM/ISIZE_DSIZE"
;4861   VA_XB+R[] PC_PC+4               "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/@1,MUX/M.R1,ALU/A+B+CI,DTYPE/LONG,
;4862                                           ISTRM/ISIZE_DSIZE,VSIZE/1"
;4863   VA_XB.XOR.R[] PC_PC+4           "WCTRL/VA_WB,ALU/XOR,MUX/XM.R,ALUXM/ZERO,MSRC/XB.PC_PC+I,RSRC/@1,ISTRM/ISIZE_DSIZE,
;4864                                           VSIZE/1,DTYPE/LONG"
;4865   VA_ZEXT(M[])                    "WCTRL/VA_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR"
;4866   VA_ZEXT(M[])+R[]                "WCTRL/VA_WB,ALU/A+B+CI,MUX/XM.R,MSRC/@1,RSRC/@2,ALUXM/ZERO"
;4867   VA_ZEXT(M[])-R[]                "WCTRL/VA_WB,ALU/A-B-CI,MUX/XM.R,MSRC/@1,RSRC/@2,ALUXM/ZERO"
;4868   VA_ZEXT(XB) PC_PC+2             "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/ZERO,MUX/XM.R,ALUXM/ZERO,ALU/OR,
;4869                                        ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/WORD"
;4870   VA_ZEXT(XB).XOR.(R[].RR.8) PC_PC+1      "WCTRL/VA_WB,ALU/XOR,MUX/XM.S,MSRC/XB.PC_PC+I,ROT/RR.RR.SIZ,RSRC/@1,
;4871                                           ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;4872   VA_ZEXT(XB)+R[] PC_PC+1         "WCTRL/VA_WB,MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.R,ALUXM/ZERO,ALU/A+B+CI,
;4873                                        ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/BYTE"
;4874   VA_ZEXT(XB)-R[] PC_PC+1         "WCTRL/VA_WB,ALU/A-B-CI,MUX/XM.R,MSRC/XB.PC_PC+I,RSRC/@1,ALUXM/ZERO,ISTRM/ISIZE_DSIZE,
;4875                                           VSIZE/1,DTYPE/BYTE"
;4876   VA_ZEXT(XB)-R[] PC_PC+2         "WCTRL/VA_WB,ALU/A-B-CI,MUX/XM.R,MSRC/XB.PC_PC+I,RSRC/@1,ALUXM/ZERO,ISTRM/ISIZE_DSIZE,
;4877                                           VSIZE/1,DTYPE/WORD"
;4878   VA_ZLITO[]                      "WCTRL/VA_WB,ALPCTL/WX_S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4879
;4880   WB_(D+CONX(1)).SR.1             "ALU/A+B+CI.SR,MUX/D.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4881   WB_(M[] R[]).RR.4               "ALPCTL/WX_S,MSRC/@1,RSRC/@2,ROT/RR.MR.4"
;4882   WB_(M[]+ALKC).BCD               "ALU/A+B+CI.BCD,ALUCI/ALKC,VSIZE/1,DTYPE/LONG,MUX/M.R1,MSRC/@1,RSRC/ZERO"
;4883   WB_(M[]+R[]).BCD                "MSRC/@1,RSRC/@2,ALU/A+B+CI.BCD,MUX/M.R1,VSIZE/1,DTYPE/LONG"
;4884   WB_(M[]-R[]).BCD                "VSIZE/1,DTYPE/LONG,RSRC/@2,MSRC/@1,MUX/M.R1,ALU/A-B-CI.BCD"
;4885   WB_(M[]-R[]-ALKC).BCD           "VSIZE/1,DTYPE/LONG,RSRC/@2,MSRC/@1,MUX/M.R1,ALU/A-B-CI.BCD,ALUCI/ALKC"
;4886   WB_(M[].AND.ZLITO[]).SR         "MSRC/@1,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@2,ALU/AND.SR"
;4887   WB_(R[] M[]).RL.P               "RSRC/@1,MSRC/@2,ROT/RL.RM.P,ALPCTL/WX_S"
;4888   WB_(R[] M[]).RL.PS              "RSRC/@1,MSRC/@2,ROT/RL.RM.PS,ALPCTL/WX_S"
;4889   WB_(R[]+Q).SL.1                 "MUX/R.Q,RSRC/@1,ALU/A+B+CI.SL,ALUSHF/ZERO"
;4890   WB_.NOT.(M[].RR.16)             "ALPCTL/WX_.NOT.S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,MSRC/@1"
;4891   WB_.NOT.(M[].RR.P)              "ALPCTL/WX_.NOT.S,ROT/RR.MM.P,MSRC/@1"
;4892   WB_.NOT.Q                       "ALU/A-B-CI,MUX/R.Q,RSRC/ZERO,ALUCI/ONE"
```

MACROS (CONT)

```
;4893   WB_O-CONX(1)              "ALU/A-B-CI,MUX/Z.S,ROT/CONX.SIZ,VSIZE/1,DTYPE/BYTE"
;4894   WB_ATCR                   "CCMISC/WB_ATCR.CCBR_SIGND"
;4895   WB_D                      "MUX/D.R1,RSRC/ZERO,ALU/OR"
;4896   WB_D+ALKC                 "MUX/D.R1,RSRC/ZERO,ALU/A+B+CI,ALUCI/ALKC"
;4897   WB_D+ZLITO[]              "ALU/A+B+CI,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4898   WB_D+ZLIT16[]             "ALU/A+B+CI,MUX/D.S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
;4899   WB_D-OLIT24[]             "MUX/D.S,ROT/OLIT24,LIT/LITRL,LITRL/@1,ALU/A-B-CI"
;4900   WB_D-PL                   "MUX/D.S,ROT/PL,ALU/A-B-CI"
;4901   WB_D-Q                    "ALU/A-B-CI,ALUCI/ZERO,MUX/D.Q1"
;4902   WB_D-R[]                  "RSRC/@1,MUX/D.R1,ALU/A-B-CI"
;4903   WB_D-ZLITO[]              "LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/A-B-CI"
;4904   WB_D.AND.ZLITO[]          "ALU/AND,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4905   WB_D.AND.ZLIT16[]         "LIT/LITRL,LITRL/@1,ROT/ZLIT16,MUX/D.S,ALU/AND"
;4906   WB_D.AND.ZLIT24[]         "ALU/AND,MUX/D.S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;4907   WB_D.ANDNOT.ZLITO[]       "LIT/LITRL,LITRL/@1,ROT/ZLITO,MUX/D.S,ALU/ANDNOT"
;4908   WB_D.XOR.R[]              "ALU/XOR,MUX/D.R1,RSRC/@1"
;4909   WB_D.XOR.ZLITO[]          "ALU/XOR,MUX/D.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"
;4910   WB_D.XOR.ZLIT16[]         "ALU/XOR,MUX/D.S,ROT/ZLIT16,LIT/LITRL,LITRL/@1"
;4911   WB_D.XOR.ZLIT24[]         "ALU/XOR,MUX/D.S,ROT/ZLIT24,LIT/LITRL,LITRL/@1"
;4912   WB_EXP(M[])               "MSRC/@1,ROT/GETEXP,ALPCTL/WX_S"
;4913   WB_EXP(M[]) Q_ZEXT(MB)    "MSRC/@1,ROT/GETEXP,ALPCTL/WX_S.Q_XM,ALUXM/ZERO"
;4914   WB_FPA                    "FPA/WBUS_FPA"
;4915   WB_M[]                    "ALU/OR,MUX/M.S,ROT/ZERO,MSRC/@1"
;4916   WB_M[] PL_MB.MSS          "MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR,ROT/PL_MSS"
;4917   WB_M[]+CONX(1)            "ALU/A+B+CI,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,DTYPE/BYTE,VSIZE/1"
;4918   WB_M[]+PSLC               "MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A+B+CI,ALUCI/PSLC"
;4919   WB_M[]+Q                  "MUX/M.Q1,MSRC/@1,ALU/A+B+CI"
;4920   WB_M[]+Q+PSLC             "MSRC/@1,MUX/M.Q1,ALU/A+B+CI,ALUCI/PSLC"
;4921   WB_M[]+R[]+ALKC           "ALU/A+B+CI,MUX/M.R1,MSRC/@1,RSRC/@2,ALUCI/ALKC"
;4922   WB_M[]+R[]+PSLC           "ALU/A+B+CI,ALUCI/PSLC,MUX/M.R1,MSRC/@1,RSRC/@2"
;4923   WB_M[]+ZLITO[]            "ALU/A+B+CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4924   WB_M[]-(MB.CLR2B)         "ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/CLR2BM"
;4925   WB_M[]-(R[].XZ)           "ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/XZ.RR,RSRC/@2"
;4926   WB_M[]-1                  "MSRC/@1,ROT/MINUS1,MUX/M.S,ALU/A+B+CI"
;4927   WB_M[]-PL                 "ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/PL"
;4928   WB_M[]-PSLC               "MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/A-B-CI,ALUCI/PSLC"
;4929   WB_M[]-Q                  "MUX/M.Q1,MSRC/@1,ALU/A-B-CI"
;4930   WB_M[]-Q-PSLC             "MSRC/@1,MUX/M.Q1,ALU/A-B-CI,ALUCI/PSLC"
;4931   WB_M[]-R[]                "MUX/M.R1,MSRC/@1,RSRC/@2,ALU/A-B-CI"
;4932   WB_M[]-R[]-ALKC           "MUX/M.R1,MSRC/@1,RSRC/@2,ALU/A-B-CI,ALUCI/ALKC"
;4933   WB_M[]-ZLITO[]            "ALU/A-B-CI,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4934   WB_M[]-ZLIT8[]            "MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT8,MUX/M.S,ALU/A-B-CI"
;4935   WB_M[].AND.CONX(1)        "ALU/AND,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,DTYPE/BYTE,VSIZE/1"
;4936   WB_M[].AND.CONX(2)        "ALU/AND,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,DTYPE/WORD,VSIZE/1"
;4937   WB_M[].AND.CONX(4)        "ALU/AND,MUX/M.S,MSRC/@1,ROT/CONX.SIZ,DTYPE/LONG,VSIZE/1"
;4938   WB_M[].AND.OLITO[]        "MSRC/@1,LIT/LITRL,LITRL/@2,ROT/OLITO,MUX/M.S,ALU/AND"
;4939   WB_M[].AND.Q              "MUX/M.Q1,MSRC/@1,ALU/AND"
;4940   WB_M[].AND.R[]            "ALU/AND,MUX/M.R1,MSRC/@1,RSRC/@2"
;4941   WB_M[].AND.ZLITO[]        "MSRC/@1,LITRL/@2,MUX/M.S,ALU/AND,ROT/ZLITO,LIT/LITRL"
;4942   WB_M[].AND.ZLIT12[]       "MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT12,MUX/M.S,ALU/AND"
;4943   WB_M[].AND.ZLIT16[]       "ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT16,LIT/LITRL,LITRL/@2"
;4944   WB_M[].AND.ZLIT20[]       "ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT20,LIT/LITRL,LITRL/@2"
;4945   WB_M[].AND.ZLIT24[]       "ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4946   WB_M[].AND.ZLIT4[]        "MSRC/@1,LIT/LITRL,LITRL/@2,ROT/ZLIT4,MUX/M.S,ALU/AND"
;4947   WB_M[].AND.ZLIT8[]        "ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
```

MACROS (CONT)

```
;4948   WB_M[].ANDNOT.ZLITO[]          "ALU/ANDNOT,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4949   WB_M[].ANDNOT.ZLIT24[]         "ALU/ANDNOT,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
;4950   WB_M[].ASR.-P                  "MSRC/@1,ROT/ASR.M.-P,ALPCTL/WX_S"
;4951   WB_M[].ASR.P                   "ALPCTL/WX_S,ROT/ASR.M.P,MSRC/@1"
;4952   WB_M[].BCDSWP                  "MSRC/@1,ROT/BCDSWP,MUX/R.S,RSRC/ZERO,ALU/OR"
;4953   WB_M[].CLR2B                   "ALPCTL/WX_S,ROT/CLR2BM,MSRC/@1"
;4954   WB_M[].OR.R[]                  "MSRC/@1,MUX/M.R1,ALU/OR,RSRC/@2"
;4955   WB_M[].RR.16                   "ALPCTL/WX_S,ROT/RR.MM.SIZ,MSRC/@1,VSIZE/1,DTYPE/WORD"
;4956   WB_M[].RR.24                   "ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/LONG,MSRC/@1"
;4957   WB_M[].RR.4                    "ALPCTL/WX_S,ROT/RR.MR.4,MSRC/@1,RSRC/@1"
;4958   WB_M[].RR.8                    "ALPCTL/WX_S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/BYTE,MSRC/@1"
;4959   WB_M[].RR.P                    "ALPCTL/WX_S,ROT/RR.MM.P,MSRC/@1"
;4960   WB_M[].VPN-R[]                 "ALU/B-A-CI,MUX/R.S,RSRC/@2,ROT/XZ.VPN,MSRC/@1"
;4961   WB_M[].XOR.Q                   "MSRC/@1,MUX/M.Q1,ALU/XOR"
;4962   WB_M[].XOR.Q Q_(Q.SL.1).OR.1   "ALU/XOR,MUX/M.Q2,MSRC/@1,DQ2/SQL,ALUSHF/ONE"
;4963   WB_M[].XOR.Q Q_Q.RL.1          "ALU/XOR,MUX/M.Q2,MSRC/@1,DQ2/SQL,ALUSHF/ROT"
;4964   WB_M[].XOR.R[]                 "ALU/XOR,MUX/M.R1,MSRC/@1,RSRC/@2"
;4965   WB_M[].XOR.ZLITO[]             "ALU/XOR,MUX/M.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;4966   WB_M[].XOR.ZLIT8[]             "ALU/XOR,MUX/M.S,MSRC/@1,ROT/ZLIT8,LIT/LITRL,LITRL/@2"
;4967   WB_M[].XZ.MM                   "MSRC/@1,ALPCTL/WX_S,ROT/XZ.MM"
;4968   WB_NOT(M[].ASR.-P)             "MSRC/@1,ROT/ASR.M.-P,ALPCTL/WX_.NOT.S"
;4969   WB_PL*2                        "ROT/PL,RSRC/ZERO,MUX/R.S,ALU/A+B+CI.SL"
;4970   WB_PSL                         "CCPSL/WB_PSL.CCBR_SIGND"
;4971   WB_Q                           "ALU/OR,MUX/R.Q,RSRC/ZERO"
;4972   WB_Q Q_M[]                     "MSRC/@1,ALPCTL/WX_Q.Q_M"
;4973   WB_Q-D                         "MUX/D.Q1,ALU/B-A-CI,ALUCI/ZERO"
;4974   WB_Q-M[]                       "MUX/M.Q1,MSRC/@1,ALU/B-A-CI"
;4975   WB_Q-R[]                       "RSRC/@1,MUX/R.Q,ALU/B-A-CI"
;4976   WB_Q_M[]                       "MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR,DQ1/Q_WX"
;4977   WB_RNUM                        "MSRC/WBUS_RNUM"
;4978   WB_R[]                         "RSRC/@1,MUX/R.S,ROT/ZERO,ALU/OR"
;4979   WB_R[]+0                       "ALU/A+B+CI,MUX/R.S,RSRC/@1,ROT/ZERO"
;4980   WB_R[]+1                       "ALU/A-B-CI,MUX/R.S,RSRC/@1,ROT/MINUS1"
;4981   WB_R[]-0                       "RSRC/@1,ROT/ZERO,MUX/R.S,ALU/A-B-CI"
;4982   WB_R[]-CONX(4)                 "RSRC/@1,ROT/CONX.SIZ,VSIZE/1,DTYPE/LONG,MUX/R.S,ALU/A-B-CI"
;4983   WB_R[]-D                       "RSRC/@1,MUX/D.R1,ALU/B-A-CI"
;4984   WB_R[]-M[]                     "RSRC/@1,MSRC/@2,MUX/M.R1,ALU/B-A-CI,ALUCI/ZERO"
;4985   WB_R[]-Q                       "RSRC/@1,MUX/R.Q,ALU/A-B-CI"
;4986   WB_R[]-XB PC_PC+4              "MSRC/XB.PC_PC+I,RSRC/@1,MUX/M.R1,ALU/B-A-CI,VSIZE/1,
;4987                                          DTYPE/LONG,ISTRM/ISIZE_DSIZE"
;4988   WB_R[]-ZEXT(M[])               "RSRC/@1,MSRC/@2,MUX/XM.R,ALUXM/ZERO,ALU/B-A-CI"
;4989   WB_R[]-ZEXT(XB) PC_PC+1        "MSRC/XB.PC_PC+I,RSRC/@1,ALUXM/ZERO,MUX/XM.R,ALU/B-A-CI,VSIZE/1,
;4990                                          DTYPE/BYTE,ISTRM/ISIZE_DSIZE"
;4991   WB_R[].AND.Q                   "RSRC/@1,MUX/R.Q,ALU/AND"
;4992   WB_R[].NOTAND.Q                "ALU/NOTAND,MUX/R.Q,RSRC/@1"
;4993   WB_R[].OR.(M[].ASR.-P)         "RSRC/@1,MUX/R.S,ALU/OR,MSRC/@2,ROT/ASR.M.-P"
;4994   WB_R[].RR.16                   "ALPCTL/WX_S,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/WORD,RSRC/@1"
;4995   WB_R[].RR.24                   "ALPCTL/WX_S,RSRC/@1,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/LONG"
;4996   WB_R[].RR.8                    "ALPCTL/WX_S,ROT/RR.RR.SIZ,VSIZE/1,DTYPE/BYTE,RSRC/@1"
;4997   WB_R[].RR.SIZ                  "RSRC/@1,ROT/RR.RR.SIZ,ALPCTL/WX_S"
;4998   WB_R[].SR.5(IF MDR IS 0)       "RSRC/@1,MSRC/MDR,ROT/RR.MR.4,MUX/M.S,ALU/A+B+CI.SR"
;4999   WB_R[].XOR.Q                   "ALU/XOR,MUX/R.Q,RSRC/@1"
;5000   WB_SEXT(M[])-Q                 "MSRC/@1,ALUXM/SIGN,MUX/XM.Q,ALU/A-B-CI"
;5001   WB_SEXT(M[]).ANDNOT.ZLITO[]    "ALU/ANDNOT,MUX/XM.S,MSRC/@1,ROT/ZLITO,LIT/LITRL,LITRL/@2"
;5002   WB_SEXT(XB) PC_PC+2            "MSRC/XB.PC_PC+I,RSRC/ZERO,ALUXM/SIGN,ALU/OR,ISTRM/ISIZE_DSIZE,VSIZE/1,DTYPE/WORD"
```

MACROS (CONT)

```
WB_SEXT(XB)-R[] PC_PC+1           "MSRC/XB.PC_PC+I,RSRC/@1,ALU/A-B-CI,MUX/XM.R,ALUXM/SIGN,VSIZE/1,
                                                          DTYPE/BYTE,ISTRM/ISIZE_DSIZE"
WB_SEXT(XB).XOR.MINUS1 PC_PC+1   "MSRC/XB.PC_PC+I,ROT/MINUS1,MUX/XM.S,ALUXM/SIGN,ALU/XOR,DTYPE/BYTE,VSIZE/1,
                                                          ISTRM/ISIZE_DSIZE"
WB_SEXT(XB).XOR.MINUS1 PC_PC+2   "MSRC/XB.PC_PC+I,ROT/MINUS1,MUX/XM.S,ALUXM/SIGN,ALU/XOR,DTYPE/WORD,VSIZE/1,
                                                          ISTRM/ISIZE_DSIZE"
WB_XB-R[] PC_PC+4                "MSRC/XB.PC_PC+I,RSRC/@1,MUX/M.R1,ALU/A-B-CI,VSIZE/1,
                                                          DTYPE/LONG,ISTRM/ISIZE_DSIZE"
WB_ZEXT(M[])                     "MSRC/@1,MUX/XM.S,ROT/ZERO,ALU/OR,ALUXM/ZERO"
WB_ZEXT(M[])-R[]                 "MSRC/@1,RSRC/@2,MUX/XM.R,ALUXM/ZERO,ALU/A-B-CI"
WB_ZEXT(M[]).AND.R[]             "MSRC/@1,RSRC/@2,MUX/XM.R,ALUXM/ZERO,ALU/AND"
WB_ZEXT(XB) PC_PC+1              "ALU/OR,ALUXM/ZERO,MUX/XM.R,MSRC/XB.PC_PC+I,RSRC/ZERO,VSIZE/1,DTYPE/BYTE,ISTRM/ISIZE_DSIZE"
WB_ZEXT(XB)-(R[].RR.16) PC_PC+2  "MSRC/XB.PC_PC+I,RSRC/@1,MUX/XM.S,ROT/RR.RR.SIZ,ALU/A-B-CI,VSIZE/1,
                                                          DTYPE/WORD,ISTRM/ISIZE_DSIZE"
WB_ZEXT(XB)-R[] PC_PC+1          "MSRC/XB.PC_PC+I,RSRC/@1,ALUXM/ZERO,MUX/XM.R,ALU/A-B-CI,VSIZE/1,
                                                          DTYPE/BYTE,ISTRM/ISIZE_DSIZE"
WB_ZEXT(XB)-R[] PC_PC+2          "MSRC/XB.PC_PC+I,RSRC/@1,ALUXM/ZERO,MUX/XM.R,ALU/A-B-CI,VSIZE/1,
                                                          DTYPE/WORD,ISTRM/ISIZE_DSIZE"
WB_ZLITO[]-M[]                   "MSRC/@2,ALU/B-A-CI,MUX/M.S,ROT/ZLITO,LIT/LITRL,LITRL/@1"

WDR_UNROT(M[].OR.ZLIT24[])       "WCTRL/WDR_WB.UR,ALU/OR,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/@2"
WDR_UNROT(R[])                   "WCTRL/WDR_WB.UR,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
WX_CONX.SIZ                      "ALUOD/OR.OD,MUX/Z.S,ROT/CONX.SIZ"
```

```
.TOC "  Branching Macros"

(M[TEMP3]-SL)BYTE RANGE CHECK?  "BUT/SRKSTA,MSRC/@1,MUX/M.S,ALU/A-B-CI,ROT/SL"
(PL+SL).GT.32?                  "BUT/SRKSTA,ROTSRK/VIELD.000"

ABSVAL M[]<7-0>?                "BUT/SRKSTA,ROT/MINUS1,MSRC/@1,MUX/M.S,ALU/AND"
ACLO FPLOCK?                    "BUT/FPS3"
ADD1(FLAG0)?                    "BUT/FLAG0"
ADD2(FLAG1) ADD1(FLAG0)?        "BUT/FLAG1TO0"
ALKC?                           "BUT/WBUS31TO30,ALPCTL/WB_ALUF"
ALLOW INT?                      "BUT/CCBR1.INT-TS"
ALUS?                           "BUT/CCBR,CC/NOP.CCBR_ALUS"
ALUS_UNSGN OLDALUS?             "BUT/CCBR,CCMISC/ALUS_UNSGN.CCBR_ALUS"
ASCII SIGN(M[])?                "BUT/SRKSTA,MSRC/@1,RSRC/ZERO,ALU/OR,MUX/M.R1,ROTSRK/ASCIISIGN.073"

BCD CHECK?                      "BUT/BCDCHK"
BCD CHECK M[]?                  "BUT/BCDCHK,MSRC/@1,ALU/A+B+CI.BCD,MUX/R.S,RSRC/ZERO,ROT/CVTNP"
BCD SIGN M[]?                   "BUT/SRKSTA,ROT/BCDSWP,MSRC/@1"
BCD SIGN.ZERO?                  "BUT/CCBR,CC/NOP.CCBR_ALUS"
BCD SIGN.ZERO(DEF)?             "BUT/CCBR"

BINARY LOAD?                    "BUT/CCBR,CC/NOP.CCBR_ALUS"
BOOT(FLAG MMNOINT)?             "BUT/MM.NOINT"
BRA ON ADD?                     "BUT/BRA.ON.ADD"

CCOP1 SIGND?                    "BUT/CCBR,CC/CCOP1.CCBR_SIGND"
CCOP2 SIGND CMP .NOT.IR0?       "BUT/CCBR1.CCBR0.IR0,CC/CCOP2.CCBR_SIGND"
CC_ZLITO[] ALUS?                "BUT/CCBR,CCPSL/CC_WB.CCBR_ALUS,LIT/LITRL,LITRL/@1,ROT/ZLITO,ALPCTL/WX_S"
CHECK INTERRUPTS?               "BUT/CCBR1.INT-TS,VSIZE/1,DTYPE/LONG"
CMP SIGNS?                      "BUT/CCBR,CCMISC/NOP.CCBR_CSIGNS"
COUNT OR INT TIMER?             "BUT/CCBR1.INT-TS"

DIVIDEND SIGN?                  "MSRC/TEMP1,BUT/FRO.FLTZ"
DBZ STEPC?                      "BUT/DBZ.SC"
DSIZE?                          "BUT/DSIZE"

EMODH(FLAG4)?                   "BUT/MM.NOINT"
EXPONENT RANGE?                 "BUT/SRKSTA"

FLAG0?                          "BUT/FLAG0"
FLAG1 (FLAG2.XOR.FLAG3)?        "BUT/F1.XOR23"
FLAG1?                          "BUT/FLAG1"
FLAG2?                          "BUT/FLAG2"
FLAG3?                          "BUT/FLAG3"
FLAG4?                          "BUT/MM.NOINT"
FLAG<1-0>?                      "BUT/FLAG1TO0"
FLAG<2-0>?                      "BUT/FLAG2TO0"
FPA PRESENT?                    "BUT/NO.FPA"
FPA(FLAG0)?                     "BUT/FLAG0"
FPD?                            "BUT/FPD"
FPS1?                           "BUT/FPS1"
FPS2?                           "BUT/FPS2"
FPS3?                           "BUT/FPS3"
FRO.FLTZ?                       "BUT/FRO.FLTZ"
```

MACROS (CONT)

```
GFLOAT(FLAG4)?                  "BUT/MM.NOINT"

INTPEND OR TIMER?               "BUT/INT-TIMSERV"
IP.TS(SIGND CMP)?               "CC/NOP.CCBR_SIGND,BUT/CCBR1.INT-TS"
IP.TS?                          "BUT/CCBR1.INT-TS,CCMISC/NOP.CCBR_BRATST"
IR<2>?                          "BUT/IR2"
IR<5>?                          "BUT/IR5"
IR<2-0>?                        "BUT/IR.2TO0"

LOD BRA?                        "BUT/LOD.BRA"
LOD INC BRA?                    "BUT/LOD.INC.BRA"

MDR_GPR.R RNUM.EQ.7?            "BUT/SPASTA,WCTRL/MDR_WB,RSRC/GPR.R,ROT/ZERO,MUX/R.S,ALU/OR"
MDR_ZEXT(OSR) BRATST?           "BUT/CCBR,CCPSL/MDR_OSR.CCBR_BRATST"

MICRO VECTOR?                   "BUT/UVCTR,CLKX/XTND"

MM.ALLOW.INT?                   "BUT/MM.ALLOW.INT"
MM.NOINT?                       "BUT/MM.NOINT"

MOPZERO (MUL1 XOR MUL2)?        "BUT/F1.XOR23"
MOPZERO(FLAG1)?                 "BUT/FLAG1"

MUL1(FLAG2) XOR MUL2(FLAG3)?    "BUT/F1.XOR23"
MUL2(FLAG3)?                    "BUT/FLAG3"

M[] BIT24?                      "BUT/WX.NE.0,ALU/AND,MUX/M.S,MSRC/@1,ROT/ZLIT24,LIT/LITRL,LITRL/1"
M[].EQ.-1?                      "BUT/CCBR0.SRKSTA0,CC/NOP.CCBR_SIGND,MSRC/@1,MUX/M.S,ALU/NOTAND,ROT/MINUS1"
M[]<31-16>.EQ.FFFF?             "BUT/CCBR,ALPCTL/WX_.NOT.S,ROT/RR.MM.SIZ,VSIZE/1,DTYPE/WORD,MSRC/@1,CC/NOP.CCBR_SIGND"

ODD ADDRESS?                    "BUT/CM.ODD.ADD"
OTHER UTRAPS?                   "BUT/BUTXB"
OPZERO(FLAG3)?                  "BUT/FLAG3"
OVER(FLAG2)?                    "BUT/FLAG2"

PL<4-0>.EQ.0?                   "BUT/SRKSTA"
PL<4-0>.EQ.0? PL<5>?            "BUT/SRKSTA,ROTSRK/PL.EQ.0.SIGN.120"
POP1C(FLAG4)?                   "BUT/MM.NOINT"
POS>31? (PL+SL)>32?             "BUT/CCBR0.SRKSTA0,CC/NOP.CCBR_ALUS,ROTSRK/VIELD.012"

NOT.Q<31>?                      "BUT/WBUS31TO30,RSRC/ZERO,MUX/R.Q,ALU/A-B-CI.SR,ALUCI/ONE,ALUSHF/ZERO"

PROBE READ MODE ON WB?          "BUT/UVCTR,CLKX/XTND,BUS/PRB.RD.MODE"
PROBE READ?                     "BUT/UVCTR,CLKX/XTND,BUS/PRB.RD"
PROBE WRITE MODE ON WB?         "BUT/UVCTR,CLKX/XTND,BUS/PRB.WR.MODE"
PROBE WRITE?                    "BUT/UVCTR,CLKX/XTND,BUS/PRB.WR"

PSL<C>?                         "BUT/PSLC"
PSL<IS.CURM>?                   "BUT/UVCTR,CLKX/XTND,WCTRL/UVCTR_CM.IS"
PSL<TP>?                        "BUT/PSLTP"

PTE CHECK READ KERNAL?          "BUT/UVCTR,CLKX/XTND,BUS/PRB.RD.PTE.K"
PTE CHECK READ?                 "BUT/UVCTR,CLKX/XTND,BUS/PRB.RD.PTE"
PTE CHECK WRITE?                "BUT/UVCTR,CLKX/XTND,BUS/PRB.WR.PTE"
```

MACROS (CONT)

```
;5136   Q IS NEG?                       "BUT/CCBR,RSRC/ZERO,MUX/R.Q,ALU/OR,CC/NOP.CCBR_SIGND"
;5137   Q IS ZERO?                      "BUT/WX.EQ.0,MUX/R.Q,RSRC/ZERO,ALU/OR,DQ1/Q_WX"
;5138   Q.NOT IS POS?                   "BUT/WBUS31TO30,RSRC/ZERO,MUX/R.Q,ALU/A-B-CI,ALUCI/ONE"
;5139
;5140   R6_RB+CONX(2) RNUM.EQ.7?        "BUT/SPASTA,RSRC/R6,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A+B+CI,VSIZE/1,DTYPE/WORD"
;5141   R7_M[] RNUM.EQ.7?               "BUT/SPASTA,RSRC/R7,SPW/RLONG,MSRC/@1,ALU/OR,MUX/M.S,ROT/ZERO"
;5142   RBS + OR - ?                    "BUT/SPASTA"
;5143   RBSP.EQ.0?                      "BUT/SPASTA"
;5144   READ(FLAG1) FPA(FLAG0)?         "BUT/FLAG1TO0"
;5145   READ(FLAG1)?                    "BUT/FLAG1"
;5146   REG MODE?                       "BUT/REGMODE"
;5147   REGINT(FLAG1) ADD1(FLAG0)?      "BUT/FLAG1TO0"
;5148   REGINT(FLAG1)?                  "BUT/FLAG1"
;5149   REI CHECK?                      "BUT/UVCTR,CLKX/XTND,WCTRL/REICHK,BUS/REICHK"
;5150   RNUM.EQ.7?                      "BUT/SPASTA,RSRC/GPR.R"
;5151   RNUM_R[] IPR CHECK?             "BUT/SPASTA,MSRC/RNUM_WBUS,ALU/OR,MUX/R.S,RSRC/@1,ROT/ZERO"
;5152
;5153   S<3-0>.NE.0?                    "BUT/SRKSTA,ROTSRK/BCDSIGN.063"
;5154   SAMESIGN(FLAG4)?                "BUT/MM.NOINT"
;5155   SET V SIGND CMP?                "BUT/CCBR,CCMISC/SETV.CCBR_SIGND"
;5156   SHIFT SIZE?                     "BUT/SRKSTA,ROTSRK/PL.EQ.0.SIGN.121"
;5157   SIGN(FLAG0)?                    "BUT/FLAG0"
;5158   SIGND CMP .NOT.IR0?             "BUT/CCBR1.CCBR0.IR0,CC/NOP.CCBR_SIGND"
;5159   SIGND CMP DEF?                  "BUT/CCBR"
;5160   SIGND CMP?                      "BUT/CCBR,CC/NOP.CCBR_SIGND"
;5161   SL_D_SEXT(M[]) WBRANGE?         "BUT/SRKSTA,ROT/PL.SL_WB,MSRC/@1,RSRC/ZERO,MUX/XM.R,ALUXM/SIGN,ALU/OR,DQ1/D_WX"
;5162   SL_M[] WBRANGE?                 "BUT/SRKSTA,ROT/PL.SL_WB,MSRC/@1,RSRC/ZERO,MUX/M.R1,ALU/OR"
;5163   STACK FLAG?                     "BUT/STACKFLG"
;5164   STEPC.GE.4? DECBY4              "BUT/BR.SC-4.INT-TS"
;5165   SUB(FLAG1)?                     "BUT/FLAG1"
;5166
;5167   TIMER?                          "BUT/INT-TIMSERV"
;5168
;5169   VA<0>?                          "BUT/WBUS0,MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR"
;5170   VA_R6_RB-CONX(2) RNUM.EQ.7?     "BUT/SPASTA,WCTRL/VA_WB,RSRC/R6,SPW/RLONG,ROT/CONX.SIZ,MUX/R.S,ALU/A-B-CI,VSIZE/1,
;5171                                                   DTYPE/WORD"
;5172
;5173   WB<0>?                          "BUT/WBUS0"
;5174   WB<1-0>?                        "BUT/WBUS1TO0"
;5175   WB<1-0>.NE.0?                   "BUT/WBUS1TO0.NE.0"
;5176   WX<15-0>.NE.0?                  "BUT/SRKSTA"
;5177   WX<31-16>.NE.0?                 "BUT/SRKSTA"
;5178   WB<31-30>?                      "BUT/WBUS31TO30"
;5179   WB<5-0>?                        "BUT/WBUS5TO0"
;5180   WB_M[]+63 PL_31 WB<7>EQ0?       "BUT/SRKSTA,MSRC/@1,MUX/M.S,ALU/A+B+CI,ROT/OLITO.PL_LIT,LIT/LITRL,LITRL/3F,VSIZE/1,
;5181                                                   DTYPE/BYTE"
;5182   WCS DISABLED?                   "BUT/WCSENA"
;5183   WRITE(FLAG1)?                   "BUT/FLAG1"
;5184   WX(SIZ).EQ.0?                   "BUT/CCBR0.SRKSTA0,CC/NOP.CCBR_SIGND"
;5185   WX.EQ.0?                        "BUT/WX.EQ.0"
;5186   WX.NE.0?                        "BUT/WX.NE.0"
;5187
;5188   ZEXT(M[]).EQ.0?                 "BUT/WX.EQ.0,MSRC/@1,MUX/XM.R,ALUXM/ZERO,ALU/OR,RSRC/ZERO"
```

MACROS (CONT)

```
;5189   .TOC    "       Ird and Dsize Rom Macros"
;5190
;5191   .NOCREF                 ;SET UP FOR NEVER CREF
;5192   .NOCREF
;5193
;5194   .ICODE
;5195   FPD [][] [][]           "FOP/@1,FPD/<.SHIFT[<.AND[<NEXT/@2>,3F8]>,-3]>,FFOP/@3,FPD.FPA/<.SHIFT[<.AND[<NEXT/@4>,3F8]>,-3]>,
;5196                            VFPD/<.EQL[0,  <.AND[<NEXT/@2>,3C07]>,  <.AND[<NEXT/@4>,3C07]>  ]>"
;5197
;5198   IRD1[][] [][]           "IOP/@1,IRD1/<.SHIFT[<.AND[<NEXT/@2>,3F8]>,-3]>,IFOP/@3,IRD1.FPA/<.SHIFT[<.AND[<NEXT/@4>,3F8]>,-3]>,
;5199                            VIRD1/<.EQL[0,  <.AND[<NEXT/@2>,3C07]>,  <.AND[<NEXT/@4>,3C07]>  ]>"
;5200
;5201   .OCODE
;5202   CNT0[][][] [][][]       "0OP/ @1,CNT0.REG/    <.AND[<NEXT/@2>,7FF]>,CNT0.MEM/    <.AND[<NEXT/@3>,7FF]>,
;5203                            0FOP/@4,CNT0.FPA.REG/<.AND[<NEXT/@5>,7FF]>,CNT0.FPA.MEM/<.AND[<NEXT/@6>,7FF]>,
;5204                            VCNT0/<.EQL[ 0, <.AND[<NEXT/@2>,3800]> , <.AND[<NEXT/@5>,3800]> ]>"
;5205
;5206   CNT1[][][] [][][]       "1OP/ @1,CNT1.REG/    <.AND[<NEXT/@2>,7FF]>,CNT1.MEM/    <.AND[<NEXT/@3>,7FF]>,
;5207                            1FOP/@4,CNT1.FPA.REG/<.AND[<NEXT/@5>,7FF]>,CNT1.FPA.MEM/<.AND[<NEXT/@6>,7FF]>,
;5208                            VCNT1/<.EQL[ 0, <.AND[<NEXT/@2>,3800]> , <.AND[<NEXT/@5>,3800]> ]>"
;5209
;5210   .CCODE
;5211   REG [] [] []            "IRD1.REG/<.AND[<NEXT/@1>,7FF]>,CNT0.REG/<.AND[<NEXT/@2>,7FF]>,CNT1.REG/<.AND[<NEXT/@3>,7FF]>,
;5212                            VREG/<.EQL[ 0, <.AND[<NEXT/@1>,3800]> , <.AND[<NEXT/@2>,3800]> , <.AND[<NEXT/@3>,3800]> ]>"
;5213
;5214   MEM [] [] []            "IRD1.MEM/<.AND[<NEXT/@1>,7FF]>,CNT0.MEM/<.AND[<NEXT/@2>,7FF]>,CNT1.MEM/<.AND[<NEXT/@3>,7FF]>,
;5215                            VMEM/<.EQL[ 0, <.AND[<NEXT/@1>,3800]> , <.AND[<NEXT/@2>,3800]> , <.AND[<NEXT/@3>,3800]> ]>"
;5216
;5217   .DCODE
;5218   SIZE [] [] [] [] [] []  "OS1/@1,OS2/@2,OS3/@3,OS4/@4,OS5/@5,OS6/@6"
;5219
;5220   .CREF
;5221   .CREF                   ;END OF NEVER CREF THE FOLLOWING WILL ALWAYS CREF
;5222   .UCODE
                                        ;5223    .BIN
```

NOTES

CLOCKS:

M CLOCK = 320 nS
B CLOCK = 160 nS

PCS L0008 - YA
- YB 1K WCS
- YC 2K WCS

RDM = L0006 (CNTRL D TO INVOKE)
L0006-YA = MTM

LEVEL 5 DIAGNOSTICS NEED THE MTM OPTION.

OPTION NAMES:

KA750	-	CPU
RH750	-	MBA
KC750	-	MTM
DW750	-	UBI
FP750	-	FPA
MS750	-	CMC
PCS750	-	PCS
KU750	-	WCS

TU-58 LED IS SERIAL TO MOTOR!! NO LED NO GO!

$$\left.\begin{array}{lll} \text{CCS} & - & \text{0 - 17FF} \\ \text{DCS} & - & \text{1800 - 183F} \\ \text{WCS} & - & \text{2000 - 23FF} \end{array}\right\} \text{80 bits wide}$$

NOTES

BOOT ROM ADDRESSES A - 0F20400
B - F20500
C - F20600
D - F20700

BOOTING WITH CACHE DISABLED:

SEE 750 DIAGNOSTIC SYSTEM OVERVIEW MANUAL - PAGE 5-2

NOTES

NOTES

NOTES

NOTES

VAX MAINTENANCE HANDBOOK: VAX-11/750

Reader's Comments

VWO

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc? Is it easy to use? ____

What features are most useful? ____

What faults or errors have you found in the manual? ____

Does this manual satisfy the need you think it was intended to satisfy? ____

Does it satisfy *your* needs? ____ Why? ____

Please send me the current copy of the *Documentation Products Directory*, which contains information on the remainder of DIGITAL's technical documentation.

Name ____ Street ____
Title ____ City ____
Company ____ State/Country ____
Department ____ Zip ____

Additional copies of this document are available from:

Digital Equipment Corporation
Accessories and Supplies Group
P.O. Box CS2008
Nashua, New Hampshire 03061

Attention: Documentation Products
Telephone: 1-800-258-1710

Order No. EK-VAXV3-HB

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730

Digital Equipment Corporation • Bedford, MA 01730