Preliminary Engineering Specification for the KA650-AA PROCESSOR MODULE

specification V1.00
18 April 1986

Gary Lidington
Digital Equipment Corporation
MLO5-5/E71
DTN 223-3771

CONTENTS

REVISION HISTORY

REV	DATE	AUTHOR	REASON
0.00	20-DEC-1985	Charlie Devane	First draft document for review
0.01	11-MAR-1986	Gary Lidington	Preliminary draft for first design review.
1.00	18-APR-1986	Gary Lidington	First draft for general review.

1 INTRODUCTION

1.1 Scope Of Document

This specification documents the functional, physical and environmental characteristics of the KA650-AA Processor Module. It should be used along with the VAX Architecture Standard (DEC STD 032) to provide a complete programmer's reference to the module.

Some information on the MS650 memory expansion modules is also provided. These modules are documented more completely in the MS650 Memory Module Specification.

1.2 Applicable Documents

The following reference material contains detailed information regarding the VAX-11 family, the KA650-AA module, and the required environment.

- o KA650-AA Console Program Specification
- o MS650 Memory Module Specification
- o CVAX CPU Chip Engineering Specification
- o CVAX Clock Chip Engineering Specification
- o CVAX FPU Chip Engineering Specification
- o CVAX Memory Controller Chip Engineering Specification
- o CVAX Q22-Bus Interface Chip Engineering Specification
- o MicroVAX System Support Chip Functional Specification
- o VAX Architecture Handbook
- o VAX Software Handbook
- o VAX Hardware Handbook
- o DEC STD 032 VAX Architecture Standard
- o DEC STD 102 Environmental Specification
- o DEC STD 160 Q-Bus Specification

2 GENERAL DESCRIPTION

The KA650 CPU module and MS650 memory modules combine to form a VAX CPU/Memory subsystem which uses the Q22-Bus to communicate with mass storage and I/O devices. The KA650 and MS650 modules mount in standard Q22-Bus backplane slots which implement the Q22-Bus in the AB rows and the CD interconnect in the CD rows. These modules communicate via the MS650 Memory Interconnect, which utilizes the CD interconnect and a 50-pin ribbon cable. A single KA650 module can support up to four MS650 modules, provided that sufficient Q22/CD slots are available.

The KA650 may be configured as either an arbiter or auxilliary CPU. Every system must contain an arbiter, which always resides in the first backplane slot. This arbiter CPU arbitrates Q22-Bus mastership and fields Q22-Bus interrupt requests plus any on-board interrupt requests. Systems with a sufficient number of Q22/CD slots can also support up to three auxiliary KA650 modules. An auxilliary CPU module can only field its own on-board interrupts. To access the Q22-Bus, an auxilliary CPU must request bus mastership from the arbiter.

The KA650 communicates with the console device via the CPU rear I/O distribution insert, which also contains configuration switches and a LED display.

Estimated compute performance for the CPU/Memory subsystem is 2.5 times a VAX 11/780.

2.1 KA650-AA Module Summary

The KA650-AA consists of a single, quad height, Q22-Bus module (M7620-AA) that uses six new in-house VLSI chips implement the following functionality:

- o A VAX central processor that supports the MicroVAX chip subset of the VAX instruction set and data types, as well as full VAX memory management with demand paging and 4GB of virtual memory.
- o A floating point accelerator with the MicroVAX chip subset of the VAX floating point instruction set and data types.
- o A two-level cache consisting of a 1KB, 100ns, first-level cache and a 64KB, 200ns, second-level cache. Both caches provide parity protection on the tag and data stores.
- o A main memory controller that supports up to 64MB of 400ns, ECC memory, that resides on one to four MS650 memory modules, depending on the system configuration.
- o A VAX compatible console port whose baud rate can be set via an external switch located on the CPU rear I/O distribution insert.

- A set of processor clock registers that support:
  - A VAX standard Time Of Year (TOY) clock with support for battery back-up (Batteries are located on the CPU rear I/O distribution insert.)
  - An interval timer with 10Ms interrupts.
  - Two programmable timers, similar in function to the VAX standard interval timer.

- A boot and diagnostic facility with four on-board LED's and support for an external 4-bit display and configuration switches located on the CPU rear I/O distribution insert, and 64KB of byte-wide ROM containing programs for:
  - Board initialization.
  - Emulation of a subset of the VAX standard console.
  - Power-up self-testing of the KA650 and MS650 modules.
  - Booting from supported Q22-Bus devices.

- A Q22-Bus interface that supports up to 16-word, block mode transfers between a Q22-Bus DMA device and main memory, and up to 2-word, block mode transfers between the CPU and Q22-Bus devices. This interface contains:
  - A 16-entry map cache for the 8,192 entry, main memory resident, "scatter-gather" map, which is used for translating 22-bit, Q22-Bus addresses into 26-bit, main memory addresses.
  - Interrupt arbitration logic that recognizes Q22-Bus interrupt requests BR7-BR4.
  - An interprocessor communication facility that supports communication between the Q22-Bus arbiter and up to three auxiliary processors via "doorbell" interrupts.
  - 240 ohm termination

2.2 MS650 Module Summary

MS650 memory options come in two variations. Each option consists of a single, quad height, Q22-Bus module. The difference between options is the array size, which is dependent on the number, density, and packaging style of the RAM chips used. The two variations are listed below:

- MS650-AA (M7621-A) An 8MB, 400ns, 39-bit wide array (32-bit data and 7-bit ECC) implemented with 256Kb dynamic RAMS in zig-zag in-line packages (ZIPS).
- MS650-BA (M7622-A) A 16MB, 400ns, 39-bit wide array (32-bit data and 7-bit ECC) implemented with 1Mb dynamic RAMS in dual in-line packages (DIPS).

3 KA650-AA CENTRAL PROCESSOR

The central processor of the KA650-AA supports the MicroVAX Chip subset of the VAX instruction set and data types and full VAX memory management. It is implemented via a single VLSI chip called the CVAX.

3.1 Processor State

The processor state consists of that portion of a process's state which is stored in processor registers rather than in memory. The processor state is composed of sixteen General Purpose Registers (GPR's), the Processor Status Longword (PSL), and the Internal Processor Registers (IPR's).

Non-privileged software can access the GPR's and the Processor Status Word (bits 15:00 of the PSL). The IPR's and bits 31:16 of the PSL can only be accessed by privileged software. The IPR's are explicitly accessible only by the the Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions which require kernal mode privileges.

3.1.1 General Purpose Registers

The KA650-AA implements 16 General Purpose Registers per DEC STD 032. These registers are used for temporary storage, accumulators, and base and index registers for addressing. These registers are denoted R0 - R15. The bits of a register are numbered from the right <0> through <31>.

```
  3
  1                                              0
+------------------------------------------------+
|                                                |
+------------------------------------------------+
```

Certain of these registers have been assigned special meaning by the VAX-11 architecture:

- o R15 is the program counter (PC). The PC contains the address of the next instruction byte of the program.
- o R14 is the stack pointer (SP). The SP contains the address of the top of the processor defined stack.
- o R13 is the current frame pointer (FP). The VAX-11 procedure call convention builds a data structure on the stack called a stack frame. The FP contains the address of the base of this data structure.
- o R12 is the argument pointer (AP). The VAX-11 procedure call convention uses a data structure termed an argument list. The AP contains the address of the base of this data structure.

DEC STD 032 should be consulted for more information on the operation and use of these registers.

3.1.2 Processor Status Longword

The KA650-AA Processor Status Longword (PSL) is implemented per the DEC STD 032, which should be consulted for a detailed description of the operation of this register. The PSL is saved on the stack when an exception or interrupt occurs and is saved in the Process Control Block (PCB) on a process context switch. Bits 15:00 may be accessed by non-privileged software, while bits 31:16 may only be accessed by privileged software. Processor initialization sets the PSL to 041F 0000 (hex).

```
 3 3 2 2 2 2 2 2 2 2 2 2       1 1
 1 0 9 8 7 6 5 4 3 2 1 0       6 5               8 7 6 5 4 3 2 1 0
+-+-+---+-+-+---+---+-+---------+---------------+-+-+-+-+-+-+-+-+
| | |   |F| |   |   |M|         |               | | | | | | | | |
|C|T|   |P|I|CUR|PRV|B|         |               |D|F|I| | | | | |
|M|P|MBZ|D|S|MOD|MOD|Z|   IPL   |      MBZ      |V|U|V|T|N|Z|V|C|
+-+-+---+-+-+---+---+-+---------+---------------+-+-+-+-+-+-+-+-+
```

PSL<31> (CM) Compatibility Mode. This bit always reads as zero, loading a "1" into this bit is a NOP.

NOTE

VAX Compatibility Mode Instructions can be emulated by macrocode, but the emulation software runs in native mode, so the CM bit is never set.

PSL<30> (TP) Trace Pending.

PSL<29:28> Unused, must be written as zero.

PSL<27> (FPD) First Part Done.

PSL<26> (IS) Interrupt Stack.

PSL<25:24> (CUR) Current Mode.

PSL<23:22> (PRV) Previous Mode.

PSL<21> Unused, must be written as zero.

PSL<20:16> (IPL) Interrupt Priority Level.

PSL<15:8> Unused, must be written as zero.

PSL<7> (DV) Decimal Overflow Trap Enable. This read/write bit has no effect on KA650-AA hardware; it can be used by macrocode which emulates VAX decimal instructions.

PSL<6> (FU) Floating Underflow Fault Enable.

PSL<5> (IV) Integer Overflow Trap Enable.

PSL<4> (T) Trace Trap Enable.

PSL<3> (N) Negative Condition Code.

PSL<2> (Z) Zero Condition Code.

PSL<1> (V) Overflow Condition Code.

PSL<0> (C) Carry Condition Code.

3.1.3 Internal Processor Registers

The KA650-AA Internal Processor Registers (IPR's) can be accessed by using the MFPR and MTPR privileged instructions. Each IPR falls into one of the following five categories:

1= Implemented by KA650-AA as specified in the VAX Architecture Standard (DEC STD 032)

2= Implemented by KA650-AA uniquely.

3= Not implemented, read as zero, nop on write.

4= Access not allowed; accesses result in a reserved operand fault.

5= Accessible, but not fully implemented, accesses yield UNPREDICTABLE results.

A table listing each of the KA650-AA IPR's with it's Mnemonic, its access type (read or write) and it's category number appears below. A list of Category One IPR's and the section where they are referenced is given in section 3.1.3.1. A list of Category Two registers and the section where they are described in full is given in section 3.1.3.2.

An "I" following the category number in the table below indicates that the register is initialized on power-up and by the negation of DCOK.

Number	Register Name	Mnemonic	Type	Category
0	Kernel Stack Pointer	KSP	r/w	1
1	Executive Stack Pointer	ESP	r/w	1
2	Supervisor Stack Pointer	SSP	r/w	1
3	User Stack Pointer	USP	r/w	1
4	Interrupt Stack Pointer	ISP	r/w	1
<7:5>	reserved			3
8	P0 Base Register	P0BR	r/w	1
9	P0 Length Register	P0LR	r/w	1
10	P1 Base Register	P1BR	r/w	1
11	P1 Length Register	P1LR	r/w	1
12	System Base Register	SBR	r/w	1
13	System Length Register	SLR	r/w	1
<14:15>	reserved			3
16	Process Control Block Base	PCBB	r/w	1
17	System Control Block Base	SCBB	r/w	1
18	Interrupt Priority Level	IPL	r/w	1I
19	AST Level	ASTLVL	r/w	1I
20	Software Interrupt Request	SIRR	w	1
21	Software Interrupt Summary	SISR	r/w	1I
<23:22>	reserved			3
24	Interval Clock Control/Status	ICCS	r/w	2I
25	Next Interval Count	NICR	w	3
26	Interval Count	ICR	r	3
27	Time Of Year	TODR	r/w	1
28	Console Storage Receiver Status	CSRS	r/w	5I
29	Console Storage Receiver Data	CSRD	r	5I
30	Console Storage Transmit Status	CSTS	r/w	5I
31	Console Storage Transmit Data	CSTD	w	5I
32	Console Receiver Control/Status	RXCS	r/w	2I
33	Console Receiver Data Buffer	RXDB	r	2I
34	Console Transmit Control/Status	TXCS	r/w	2I
35	Console Transmit Data Buffer	TXDB	w	2I
36	Translation Buffer Disable	TBDR	r/w	3
37	Cache Disable	CADR	r/w	2I
38	Machine Check Error Summary	MCESR	r/w	3
39	Memory System Error	MSER	r/w	2I
<41:40>	reserved			3
42	Console Saved PC	SAVPC	r	2
43	Console Saved PSL	SAVPSL	r	2
<47:44>	reserved			3
48	SBI Fault/Status	SBIFS	r/w	3
39	SBI Silo	SBIS	r	3
50	SBI Silo Comparator	SBISC	r/w	3
51	SBI Maintenance	SBIMT	r/w	3
52	SBI Error Register	SBIER	r/w	3

53	SBI Timeout Address Register	SBITA	r	3
54	SBI Quadword Clear	SBIQC	w	3
55	IO Bus Reset	IORESET	w	2
56	Memory Management Enable	MAPEN	r/w	1
57	TB Invalidate All	TBIA	w	1
58	TB Invalidate Single	TBIS	w	1
59	TB Data	TBDATA	r/w	3
60	Microprogram Break	MBRK	r/w	3
61	Performance Monitor Enable	PMR	r/w	3
62	System Identification	SID	r	1
63	Translation Buffer Check	TBCHK	w	1
64:127	and all those not listed - reserved			4

3.1.3.1 KA650-AA VAX Standard Internal Processor Registers

Internal Processor Registers (IPR's) which are implemented per DEC STD 032 are classified as Category One IPR's. DEC STD 032 should be consulted for details on the operation and use of these registers.

The following category one registers are also referenced in other sections of this specification:

Number	Register Name	Mnemonic	Section
12	System Base Register	SBR	3.5.2
13	System Length Register	SLR	3.5.2
16	Process Control Block Base	PCBB	3.2
17	System Control Block Base	SCBB	3.6.4
18	Interrupt Priority Level	IPL	3.6.1
20	Software Interrupt Request	SIRR	3.6.1
21	Software Interrupt Summary	SISR	3.6.1
27	Time Of Year Clock	TODR	8.1
56	Memory Management Enable	MAPEN	3.5.2
57	Trans. Buffer Invalidate All	TBIA	3.5.2
58	Trans. Buffer Invalidate Sing.	TBIS	3.5.2
62	System Identification	SID	3.9
63	Translation Buffer Check	TBCHK	3.5.2

3.1.3.2 KA650-AA Unique Internal Processor Registers

Internal Processor Registers (IPR's) which are implemented uniquely on the KA650-AA (i.e. they are not contained in, or do not fully conform to DEC STD 032) are classified as Category Two IPR's and are described in detail in this specification. Refer to the following sections for a description of these registers:

Number	Register Name	Mnemonic	Section
24	Interval Clock Control/Status	ICCS	8.2
32	Console Receiver Control/Status	RXCS	7.1.1
33	Console Receiver Data Buffer	RXDB	7.1.2
34	Console Transmit Control/Status	TXCS	7.1.3
35	Console Transmit Data Buffer	TXDB	7.1.4
37	Cache Disable	CADR	5.1.5
39	Memory System Error	MSER	5.1.6
42	Console Saved PC	SAVPC	3.7
43	Console Saved PSL	SAVPSL	3.7
55	IO Bus Reset	IORESET	9.5.1

3.2 Process Structure

A process is a single thread of execution. The context of the current process is contained in the Process Control Block (PCB) which is pointed to by the Process Control Block Base Register (PCBB). The KA650-AA implements these structures as defined in DEC STD 032, which should be referenced for a description of the PCB and the PCBB.

3.3 Data Types

The KA650-AA CPU supports the following subset of the VAX data types:

1. byte

2. word

3. longword

4. quadword

5. character string

6. variable length bit field

Support for the remaining VAX data types can be provided via macrocode emulation.

3.4 Instruction Set

The KA650-AA CPU implements the following subset of the VAX instruction set types in microcode.

1. Integer arithmetic and logical

2. Address
3. Variable length bit field
4. Control
5. Procedure call
6. Miscellaneous
7. Queue
8. Character string moves (MOVC3 and MOVC5)
9. Operating system support
10. F_floating
11. G_floating
12. D_floating

The KA650-AA CVAX chip provides special microcode assistance to aid the macrocode emulation of the following instruction groups:

1. Character string (except MOVC3 and MOVC5)
2. Decimal string
3. CRC
4. EDITPC

The following instruction groups are not implemented, but may be emulated by macrocode:

1. Octaword
2. Compatibility Mode Instructions

Appendix E lists the entire KA650-AA instruction set, indicating which instructions are implemented in the Floating Point Accelerator FPA, and which instructions have microcode assists to speed up macrocode emulation.

3.5 Memory Management

The KA650-AA implements full VAX Memory Management as defined in DEC STD 032. System Space addresses are virtually mapped through single-level page tables, and process space addresses are virtually mapped through two-level page tables. See DEC STD 032 for descriptions of the virtual to physical address translation process, and the format for VAX Page Table Entries (PTE's).

3.5.1 Translation Buffer

To reduce the overhead associated with translating virtual addresses to physical addresses, the KA650-AA employs a 32-entry, fully associative, translation buffer for caching modified VAX PTE's. Each entry can store a modified PTE for translating virtual addresses in either the VAX Process Space, or VAX System Space.

Each entry is divided into two parts: a 23-bit Tag Register and a 31-bit PTE Register. The Tag Register is used to store the Virtual Page Number (VPN) of the virtual page that the corresponding PTE Register maps. The PTE register stores the 21-bit PFN field, the PTE.V bit, the PTE.M bit and an 8-bit partially decoded representation of the 4-bit VAX PTE PROT field, from the corresponding VAX PTE, as well as a Translation Buffer Valid (TB.V) bit. The CVAX CPU Design Spec can be referenced for details of the 8-bit PROT field.

During virtual to physical address translation, the contents of the 32 Tag Registers are compared with the Virtual Page Number Field (bits <31:9>) of the virtual address of the reference. If there is a match with one of the Tag Registers, then a translation buffer "hit" has occured, and the contents of the corresponding PTE register is used for the translation.

If there is no match, the translation buffer does not contain the necessary VAX PTE information to translate the address of the reference, and the PTE must be fetched from memory. Upon fetching the PTE, the translation buffer is updated by replacing the entry that is selected by the replacement pointer. Since this pointer is moved to the next sequential translation buffer entry whenever it is pointing to an entry that is accessed, the replacement algorithm is Not Last Used (NLU).

3.5.2 Memory Management Control Registers

There are four IPR's that control the Memory Management Unit (MMU): IPR 56 (MAPEN), IPR 57 (TBIA), IPR 58 (TBIS) and IPR 63 (TBCHK).

Memory management can be enabled/disabled via IPR 56 (MAPEN). Writing "0" to this register with a MTPR instruction disables memory management and writing a "1" to this register with a MTPR instruction enables memory management. To determine whether or not memory management is enabled, IPR 56 is read using the MFPR instruction.

NOTE

> Whenever memory management is disabled, the contents of translation buffer are UNPREDICTABLE. Therefore, before enabling memory management during processor initialization, or any other time, the entire translation buffer must be cleared by software.

Translation buffer entries that map a particular virtual address can be invalidated by writing the virtual address to IPR 58 (TBIS) using the MTPR instruction.

NOTE

> Whenever software changes a valid Page Table Entry for the system or current process region, or a System Page Table Entry that maps any part of the current process page table, all process pages so mapped must be invalidated in the translation buffer.

The entire translation buffer can be invalidated by writing a "0" to IPR 57 (TBIA) using the MTPR instruction.

NOTE

> Whenever the location or size of the system map is changed by changing the SBR or SLR, the entire translation buffer must be cleared.

The translation buffer can be checked to see if it contains a valid translation for a particular virtual page by writing a virtual address within that page to IPR 63 (TBCHK) using the MTPR instruction. If the translation buffer contains a valid translation for the page, the condition code V bit (bit<1> of the PSL) is set.

NOTE

> The TBIS, TBIA, and TBCHK IPR's are write only. The operation of a MFPR from any of these register is UNDEFINED.

3.6 Exceptions And Interrupts

Both exceptions and interrupts divert execution from the normal flow of control. An exception is caused by the execution of the current instruction and is typically handled by the current process (e.g. an arithmetic overflow), while an interrupt is caused by some activity outside the current process and typically transfers control outside the process (e.g. an interrupt from an external hardware device).

3.6.1 Interrupts

The VAX architecture specifies 31 interrupt levels which are used by the KA650-AA as follows:

Interrupt Levels	Interrupt Condition
non-maskable	BHALT asserted on Q-22 Bus, BREAK generated by console
1F	unused
1E 1D	BPOK negated on Q-22 Bus, CDAL Bus parity error, Q-22 Bus NXM on a write, CDAL Bus timeout during DMA, uncorrectable main memory errors
1B - 1C	unused
1A	2nd-level cache tag parity errors, correctable main memory errors
18 - 19	unused
17	BR7 L asserted
16	Interval Timer Interrupt, BR6 L asserted
15	BR5 asserted
14	Console Terminal Interrupts, Programmable Timer Interrupts, Interprocessor Doorbell Interrupts, BR4 L asserted
10 - 13	unused
01 - 0F	software interrupt request

NOTE

Because the Q22-Bus does not allow differentiation between the four bus grant levels (i.e a BR7 device could grab a level 4 bus grant), the KA650-AA CPU must set the IPL to 17 after responding to any interrupt request BR7-4. The IPL is set to 14 after a console terminal, programmable timer interrupt or interprocessor doorbell, and it is set to 16 after an interval timer interrupt.

NOTE

When the KA650-AA is configured as an auxiliary CPU it ignores Q22-Bus BR7-4 interrupt requests, but does respond to IPL 14 requests from its own console serial line unit, programmable timers, and interprocessor doorbell (in that order of priority). It also responds to interrupt requests from its own interval timer at IPL 16.

The interrupt system is controlled by three IPR's: IPR 18, the Interrupt Priority Level Register (IPL), IPR 20, the Software Interrupt Request Register (SIRR), and IPR 21, the Software Interrupt Summary Register (SISR). The IPL is used for loading the processor priority field in the PSL (bits<20:16>. The SIRR is used for creating software interrupt requests. The SISR records pending software interrupt requests at levels 1 through 15. The format of these regiters is presented below, refer to DEC STD 032 for more information on these registers.

```
 3
 1                                             5 4       0
+---------------------------------------------+----------+
|                ignored, returns 0           |PSL<20:16>| :IPL
+---------------------------------------------+----------+

 3
 1                                                4 3    0
+------------------------------------------------+-------+
|                   ignored                      |request| :SIRR
+------------------------------------------------+-------+


 3                         1 1
 1                         6 5                         0
+-------------------------+-------------------------+--+
|                         | Pending Software Interrupts |M|
|                         |                          |B| :SISR
|                         |F E D C B A 9 8 7 6 5 4 3 2 1|Z|
+-------------------------+-------------------------+--+
```

3.6.2 Exceptions

The VAX architecture recognizes six classes of exceptions.

Exception Class	Instances
arithmetic traps/faults	integer overflow trap integer divide by zero trap subscript range trap floating overflow fault floating divide by zero fault floating underflow fault
memory management exceptions	access control violation fault translation not valid fault
operand reference exceptions	reserved addressing mode fault reserved operand fault or abort
instruction execution exceptions	reserved/privileged instr. fault emulated instruction faults extended function fault breakpoint fault
tracing exception	trace fault
system failure exceptions	machine check abort (including CDAL Bus parity errors, cache parity errors, Q-22 Bus timeout errors, Q-22 Bus device parity errors, CDAL Bus timeout errors and main memory uncorrectable errors) kernel stack not valid abort interrupt stack not valid abort

3.6.3 Machine Check Parameters

In response to a machine check, the following parameters are pushed on the stack:

```
+---------------------------------------------------+
|              byte count (00000010 hex)             |  :SP
+---------------------------------------------------+
|              machine check code                    |
+---------------------------------------------------+
|              most recent virtual address           |
+---------------------------------------------------+
|              internal state information  1         |
+---------------------------------------------------+
|              internal state information  2         |
+---------------------------------------------------+
|                     PC                             |
+---------------------------------------------------+
|                     PSL                            |
+---------------------------------------------------+
```

The parameters are:

machine check code (hex):

1	=	undefined MMGT.STATUS
2	=	floating point protocol error
3	=	floating point reserved instruction
4	=	undefined MOVCx state
5	=	process PTE in P0 space during TB miss flows
6	=	process PTE in P1 space during TB miss flows
7	=	process PTE in P0 space during M = 0 flows
8	=	process PTE in P1 space during M = 0 flows
9	=	undefined INT.ID value
80	=	memory read error
81	=	SCB, PCB, or SPTE read error
82	=	memory write error
83	=	SCB, PCB, or SPTE write error

most recent virtual address:

<31:0>	=	current contents of VAP register

internal state information 1:

<31:24>	=	opcode
<23:16>	=	1111, highest priority software interrupt <3:0>
<15:8>	=	CADR<7:0>
<7:0>	=	MSER<7:0>

internal state information 2:

<31:24>	=	most recent contents of SC register <7:0>

```
        <23:16> =       11, state flags <5:0>
        <15:8>  =       restart flag, 111, ALU CC flags <3:0>
        <7:0>   =       offset from saved PC to PC at time of
                        machine check

     PC:    <31:0>  =     PC at the start of the current instruction

     PSL:   <31:0>  =     current contents of PSL
```

3.6.4 System Control Block (SCB)

The System Control Block (SCB) is a page containing the vectors for servicing interrupts and exceptions. The SCB is pointed to by IPR 17, the System Control Block Base Register (SCBB).

```
      3 3 2
      1 0 9                                   9 8                 0
     +---+------------------------------------+-----------------+
     |MBZ|   physical longword address of PCB |       MBZ       | :SCBB
     +---+------------------------------------------------------+
```

The System Control Block format:

vector	name	type	param	notes
00	unused	-	-	IRQ passive release on other VAXes
04	machine check	abort	3	parameters depend on error type
08	kernel stack not valid	abort	0	must be serviced on interrupt stack
0C	power fail	interrupt	0	IPL is raised to 1E
10	reserved/privileged instruction	fault	0	
14	customer reserved instruction	fault	0	XFC instruction
18	reserved operand	fault/ abort	0	not always recoverable
1C	reserved addressing mode	fault	0	
20	access control violation	fault	2	parameters are virtual address, status code
24	translation not valid	fault	2	parameters are virtual address, status code
28	trace pending (TP)	fault	0	
2C	breakpoint instruction	fault	0	
30	unused	-	-	compatibility mode in other VAXes
34	arithmetic	trap/ fault	1	parameter is type code
38-3C	unused	-	-	-
40	CHMK	trap	1	parameter is sign-extended operand word
44	CHME	trap	1	parameter is sign-extended operand word
48	CHMS	trap	1	parameter is sign-extended operand word

4C	CHMU	trap	1	parameter is sign-extended operand word
50	unused	-	-	-
54	corrected read data	interrupt	0	IPL is 1A (CRD L)
58-5C	unused	-	-	-
60	memory error	interrupt	0	IPL is 1D (MEMERR L)
64-80	unused	-	-	-
84	software level 1	interrupt	0	
88	software level 2	interrupt	0	ordinarily used for AST delivery
8C	software level 3	interrupt	0	ordinarily used for process scheduling
90-BC	software levels 4-15	interrupt	0	
C0	interval timer	interrupt	0	IPL is 16 (INTTIM L)
C4	unused	-	-	-
C8	emulation start	fault	10	same mode exception, FPD = 0; parameters are opcode, PC, specifiers
CC	emulation continue	fault	0	same mode exception, FPD = 1: no parameters
D0-F4	unused	-	-	-
F8	Console Receiver	interrupt	0	IPL is 14
FC	Console Transmitter	interrupt	0	IPL is 14
100-1FC	adapter vectors	interrupt	0	Not implemented by the KA650-AA
200-3FC	device vectors	interrupt	0	Correspond to Q22-Bus Vectors 000 - 1FC; KA650-AA appends the assertion of bit <9,0>
400-FFFC	unused vectors	interrupt	0	

3.6.5 Hardware Detected Errors

The KA650 detects certain error conditions during program execution. These conditions, and the resultant actions, are summarized in Appendix F. Error information is also included in the section describing each major function.

3.7 The Hardware Restart Process

The KA650-AA processor enters the hardware restart process upon the occurance of any of several events:

- o On power-up and the negation of BDCOK on the Q-22 Bus.
- o When BINIT is asserted on the Q22-bus, or IPR<8> (Auxiliary Halt) is set and the KA650-AA is configured as an auxiliary.
- o When HALTs are enabled and a BREAK is generated by the console, or BHALT is asserted on the Q-22 Bus.
- o When the hardware or kernel software environment becomes severely corrupted and the CPU cannot continue normal processing.

In these instances, the KA650-AA executes a restart process and passes control to recovery code beginning at physical address 2004 0000 (hex). The current value of the PC is stored in IPR 42 (SAVPC). The PSL, MAPEN<0>, and the restart code are saved in IPR 43 (SAVPSL). The current stack pointer is saved in the appropriate internal register. The PSL is set to 041F0000 (hex) and the current stack pointer is loaded from the interrupt stack pointer. The restart process sets the state of the KA650-AA CPU as follows:

Register		New Contents
SAVPC	=	saved PC
SAVPSL<31:16,7:0>	=	saved PSL<31:16,7:0>
SAVPSL<15>	=	saved MAPEN<0>
SAVPSL<14>	=	valid PSL flag (unknown on power-up, negation of DCOK, BINIT asserted*)
SAVPSL<13:8>	=	saved restart code
SP	=	current interrupt stack
PSL	=	041F 0000 (hex)
PC	=	2004 0000 (hex)
MAPEN	=	0
ICCS	=	0 (on power-up, negation of DCOK, BINIT asserted*)
MSER	=	0 (on power-up, negation of DCOK, BINIT asserted*)
CADR	=	0 (on power-up, negation of DCOK, BINIT asserted or IPR<8> set*) first-level cache is also flushed)
SISR	=	0 (on power-up ¯negation of DCOK, BINIT asserted*)

ASTLVL	=	0 (on power-up ¯negation of DCOK, BINIT asserted*)
all else	=	undefined

*if KA650-AA is configured as an auxilary processor

The restart codes indicating the reason for the restart are listed below:

Code	Condition
2	external halt asserted
3	power-up, DCOK negated, BINIT asserted*
4	interrupt stack not valid during exception
5	machine check during normal exception
6	HALT instruction executed kernel mode
7	SCB vector bits<1:0> = 11
8	SCB vector bits<1:0> = 10
A	CHMx executed while on interrupt stack
B	CHMx executed to the interrupt stack
10	ACV or TNV during machine check exception
11	ACV or TNV during kernel stack not valid exception
12	machine check during machine check exception
13	machine check during kernel stack not valid exception
19	PSL<26:24> = 101 during interrupt or exception
1A	PSL<26:24> = 110 during interrupt or exception
1B	PSL<26:24> = 111 during interrupt or exception
1D	PSL<26:24> = 101 during REI
1E	PSL<26:24> = 110 during REI
1F	PSL<26:24> = 111 during REI

*if KA650-AA is configured as an auxilary processor

3.8 Latency Specifications

This section will discuss interrupt and DMA latency times.

3.9 System Identification

The System Identification Register (SID), IPR 62, is a read-only register implemented per the DEC STD 032 in the CVAX chip. This 32-bit, read-only register is used to identify the processor type and it's microcode revision level.

```
 3            2 2                     8 7              0
 1            4 3                                         
+-------------+-------------------------+----------------+
|    TYPE     |        RESERVED         | MICROCODE REV. |
+-------------+-------------------------+----------------+
```

SID<31:24> (TYPE) Processor type. This field always reads as 10 (decimal) indicating the processor is implemented using the CVAX chip.

SID<23:8> Reserved for future use.

SID<7:0> (MICROCODE REV.) Microcode Revision. This field reflects the microcode revision level of the CVAX chip.

In order to distinguish between different CPU implementations that use the same CPU chip, the KA650-AA, as must all VAX processors which use the CVAX chip, implements a MicroVAX System Type Register (SYS_TYPE) at physical address 2004 0004. This 32-bit read-only register is implemented in the KA650-AA ROM. The format of this register appears below:

```
 3            2 2            1 1                        0
 1            4 3            6 5                        0
+-------------+---------------+-------------------------+
|  SYS_TYPE   |   REV LEVEL   |        RESERVED         |
+-------------+---------------+-------------------------+
```

SYS_TYPE<31:24> (SYS_TYPE) System Code. This field reads as <TBD> for the KA650-AA.

SYS_TYPE<23:16> (REV LEVEL) Revision Level. This field reflects revision level of the KA650-AA firmware.

SYS_TYPE<15:00> Reserved for future use.

3.10 CPU References

All references by the CPU can be classified into one of three groups: Request Instruction-Stream Read references, Demand Data-Stream Read references or Write references.

3.10.1 Instruction-Stream Read References

The CPU has an instruction prefetcher with a 12-byte (3 longword) Instruction Prefetch Queue (IPQ) for prefetching program instructions from either cache or main memory. Whenever there is an empty longword in the IPQ, and the prefetcher is not halted due to an error, the instruction prefetcher will generate an aligned longword, Request Instruction-Stream (I-Stream) read reference.

3.10.2 Data-Stream Read References

Whenever data is immediately needed by the CPU to continue processing, a Demand Data-Stream (D-Stream) read reference is generated. More specifically, Demand D-Stream references are generated on operand, Page Table Entry (PTE), System Control Block (SCB), and Process Control Block (PCB) references. When interlocked instructions, such as Branch on Bit Set and Set Interlock (BBSSI) are executed, a Demand

D-Stream Read-Lock reference is generated. Since the CPU does not impose any restrictions on data alignment (other than the aligned operands of the ADAWI and interlocked queue instructions) and since memory can only be accessed one aligned longword at a time, all data read references are translated into an appropriate combination of masked and unmasked, aligned longword read references. If the required data is a byte, a word within a longword, or an aligned longword, then a single, aligned longword, Demand D-Stream read reference is generated. If the required data is a word that crosses a longword boundry, or an unaligned longword, then two successive aligned longword Demand D-Stream read references are generated. Data larger than a longword is divided into a number of successive aligned longword Demand D-Stream reads, with no optimization.

3.10.3 Write References

Whenever data is stored or moved a write reference is generated. Since the CPU does not impose any restrictions on data alignment (other than the aligned operands of the ADAWI and interlocked queue instructions) and since memory can only be accessed one aligned longword at a time, all data write references are translated into an appropriate combination of masked and unmasked aligned longword write references. If the required data is a byte, a word within a longword, or an aligned longword, then a single, aligned longword, write reference is generated. If the required data is a word that crosses a longword boundry, or an unaligned longword, then two successive aligned longword write references are generated. Data larger than a longword is divided into a number of successive aligned longword writes.

4 KA650AA FLOATING POINT ACCELERATOR

The KA650-AA Floating Point Accelerator is implemented via a single VLSI chip called the CFPA.

4.0.1 Floating Point Accelerator Instructions

The KA650-AA Floating Point Accelerator processes f_floating, d_floating and g_floating format instructions (except CLRx, MOVx, and TSTx which are processed by the CPU) and accelerates the execution of MULL, DIVL, and EMUL integer instructions.

4.0.2 Floating Point Accelerator Data Types

The KA650-AA Floating Point Accelerator supports byte, word, longword, quadword, f_floating, d_floating and g_floating data types. The h_floating data type is not supported, but may be implemented by macrocode emulation.

5 KA650-AA CACHE MEMORY

To maximize CPU performance, the KA650-AA incorporates a two-level cache. The first-level cache is implemented within the CVAX chip. The second-level cache is implemented using 16K x 4bit static RAMS.

5.1 Cacheable References

Any reference that is stored by the First-Level Cache is called a "cacheable reference". The First-Level Cache stores CPU read references to the VAX Memory Space (bit <29> of the physical address equals 0) only. It does not store references to the VAX I/O space, or DMA references by the Q22-Bus Interface. The the type(s) of CPU references that can be stored, (either Request Instruction Stream (I-Stream) Read References, or Demand Data Stream (D-Stream) Read References other than Read-Lock References), is determined by the the state of Cache Disable Register (CADR) bits <5:4>. The normal operating mode is for both I-Stream and D-stream references to be stored.

Whenever the CPU generates a non-cacheable reference, a single longword reference of the same type is generated on the CDAL Bus.

Whenever the CPU generates a cacheable reference that is stored in the First-Level Cache, no reference is generated on the CDAL Bus.

Whenever the CPU generates a cacheable reference that is not stored in the First-Level Cache, a quadword transfer is generated on the CDAL Bus. If the CPU reference was a Request I-Stream Read, then the quadword transfer consists of two indivisible longword transfers, the first being a Request I-Stream Read (prefetch) and the second being a Request I-Stream Read (fill). If the CPU reference was a Demand D-Stream Read, then the quadword transfer consists of two indivisible longword transfers, the first being a Demand D-Stream Read and the second being a Request D-Stream Read (fill).

The Second-Level Cache only stores references on the CDAL Bus that are part of a quadword transfer. Since quadword transfers on the CDAL Bus can only be generated on cacheable references, the Second-Level Cache is automatically configured to store the same type(s) of references as the First-Level Cache.

5.2 First-Level Cache

The KA650 includes a 1KB, two-way associative, write through, first-level cache with a 100ns cycle time. CPU read references access one longword at a time, while CPU writes can access one byte at a time. A single parity bit is generated, stored and checked for each byte of data and each tag.

5.2.1 First-Level Cache Organization

The First-Level Cache is divided into two independent storage arrays called Set 1 and Set 2. Each set contains a 64 Row x 22-bit Tag Array and a 64 Row x 72-Bit Data Array. The two sets are organized as shown below:

```
                       KA650 First-Level Cache Organization
                       ------------------------------------

                          Set 1                             Set 2
           +------+-----------------+  +------+-----------------+
          /|      |                 |  |      |                 |
         | |64x22-|    64x72-Bit    |  |64x22-|    64x72-Bit    |
         | |Bit   |    Data Array   |  |Bit   |    Data Array   |
 64 Rows<  |Tag   |                 |  |Tag   |                 |
         | |Array |                 |  |Array |                 |
         | |      |                 |  |______|_________________|
         | |      |                 |  |______|_________________|<-+
          \|      |                 |  |      |                 |  |
           +------+-----------------+  +------+-----------------+  |
          93    72 71               0  93   72 71               0  |
                                                                   |
                                                  Cache Entry ---+
```

A row within a set corresponds to a cache entry, so there are 64 entries in each set and a total of 128 entries in the entire cache. Each entry contains a 22-bit Tag Block and a 72-bit (eight-byte) Data Block. A cache entry is organized as shown below:

Cache Entry:

```
9            7 7
3            2 1                                       0
+-------------+----------------------------------------+
| Tag Block   |              Data Block                |
+-------------+----------------------------------------+
```

A Tag Block consists of a parity bit, a valid bit, and a 20-bit tag. A Tag Block is organized as shown below:

Tag Block:

```
         1
         9                                             0
+---+---+----------------------------------------------+
| P | V |               Tag                            |
+---+---+----------------------------------------------+
  |   |
  |   +--- Valid Bit
  +------- Parity Bit
```

A data block consists of eight bytes of data, each with an associated parity bit. The total data capacity of the cache is 128 eight-byte blocks, or 1024 bytes. A data block is organized as shown below:

Data Block:

```
   6  5   5  4   4  4   3  3   3  2   2  1   1
   3  6   5  8   7  0   9  2   1  4   3  6   5  8   7  0
+-+----+-+----+-+----+-+----+-+----+-+----+-+----+-+----+
|P| B7 |P| B6 |P| B5 |P| B4 |P| B3 |P| B2 |P| B1 |P| B0 |
+-+----+-+----+-+----+-+----+-+----+-+----+-+----+-+----+
                                                  ^   ^
                                                  |   |
                                      Parity Bit -+   +-- Data Byte 0
```

5.2.2 First-Level Cache Address Translation

Whenever the CPU requires an instruction or data, the contents of the first-level cache is checked to determine if the referenced location is stored there. The cache contents is checked by translating the physical address as follows:

On non-cacheable references, the reference is never stored in the cache, so a first-level cache "miss" occurs and a single longword reference is generated on the CDAL Bus.

On cacheable references, the physical address must be translated to determine if the contents of the referenced location is resident in the cache. The Cache Index Field, bits <8:3> of the physical address, is used to select one of the 64 rows of the cache, with each row containing a single entry from each set. The Cache Tag Field, bits <28:9> of the physical address, is then compared to the Tag Block of the entry from both sets in the selected row.

If a match occurs with the Tag Block of one of the set entries, and the valid bit within the entry is set, the contents of the referenced location is contained in the cache and a cache "hit" occurs. On a cache hit, the Set Match Signals generated by the compare operation select the data block from the appropriate set. The Cache Displacement Field, bits <2:0> of the physical address, is used to select the byte(s) within the block. No CDAL Bus transfers are initiated on CPU references that "hit" the first-level cache.

If no match occurs, then the contents of the referenced location is not contained in the cache and a cache "miss" occurs. In this case, the data must be obtained from either the second-level cache, or the main memory controller. If the reference is cacheable, than a quadword transfer is initiated on the CDAL Bus. If the reference is not cacheable, than a single longword transfer is initiated on the CDAL Bus.

KA650 First-level Cache Address Translation

```
       2 2
       9 8                                            9 8     3 2  0
      +-+----------------------------------------------+-------+----+
      | |            Cache Tag                         |       |    |
      +-+----------------------------------------------+-------+----+
       |                |                                  |      |
       +-- I/O Space    |                     Cache Index -+      |
                        |                                  |      |
      +-----------------+           +----------------------+      |
      |                             |         Cache Displacement -+
      |                             |                             |
      | Valid Bit                   | Valid Bit                   |
      |   |                         |   |                         |
      |   V          Set 1          |   V       Set 2             |
      |  +-+-----+---------------+  |  +-+-----+---------------+  |
      |  | |     |               |  |  | |     |               |  |
      |  | | 20- |    64-Bit     |  |  | | 20- |    64-Bit     |  |
      |  | | Bit |  Data Block   |  |  | | Bit |  Data Block   |  |
      |  | | Tag |               |  |  | | Tag |               |  |
      |  | |     |               |  |  | |     |               |  |
      |  |_|_____|_______________|  |  |_|_____|_______________|  |
      |  |_|_____|_______________|<-+->|_|_____|_______________|  |
      |  | |     |       |       |     | |     |       |       |  |
      |  +-+-----+-------|-------+     +-+-----+-------|-------+  |
      |       |          |                  |          |          |
      |       |          |                  |          |          |
      +-------|--+-------|------------------|--+       |          |
              |  |       |                  |  |       |          |
             -- --       |                 -- --       |          |
             \ V /       |                 \ V /       |          |
              \_/        |                  \_/        |          |
               |         |                   |         |          |
         Set 1 | Match ? |             Set 2 | Match ? |          |
               |         |                   |         |          |
               |  +------|-------------------+         |          |
               |  |      V                             V          |
               |  |   -------------------------------------       |
               |  +-->\                                   /<------+
               +------>\                                 /
                        ---------------------------------
                                         |
                                         V
                                       Data
```

5.2.3 First-Level Cache Data Block Allocation

Cacheable references that "miss" the first-level cache, cause a quadword read to be initiated on the CDAL bus. When the requested quadword is supplied by either the second-level cache or the main memory controller, the requested longword is passed on to the CPU, and a data block is allocated in the cache to store the entire quadword.

Due to the fact that the cache is two-way associative, there are only two data blocks (one in each set) that can be allocated to a given quadword. These two data blocks are determined by the Cache Index Field of the address of the quadword, which selects a unique row within the cache. Selection of a data block within the row (i.e. set selection) for storing the new entry is random.

Since the KA650 supports 64MB (8M quadwords) of physical memory, up to 128K quadwords "share" each row (two data blocks) of the cache. Contiguous programs larger than 512B or any non-contiguous programs separated by 512B have a 50% chance of over-writing themselves when cache data blocks are allocated for the first time for data separated by 512B (one page). After six allocations, there is a 97% probability both sets will be filled.

5.2.4 First-Level Cache Behavior On Writes

On CPU generated write references, the first-level cache is "write through". All CPU write references that "hit" the first-level cache cause the contents of the referenced location in main memory to be updated as well as the copy in the cache.

On DMA write references that "hit" the first-level cache, the cache entry containing the copy of the referenced location is invalidated. If the first-level cache is configured to store only I-Stream references, then the entire first-level cache is also flushed whenever an REI instruction is executed. (DEC STD 032 requires that an REI instruction be executed before running code out of a page of memory that has been updated.)

5.2.5 Cache Disable Register (IPR 37)

The Cache Disable Register (CADR), Internal Processor Register 37, controls the first-level cache, and is unique to CPU designs that use the CVAX chip.

```
 3
 1                                                   8 7 6 5 4 3 2 1 0
+---------------------------------------------------+---+---+-+-+-+-+
|                                                   |   |   | | |W|D|
|                         0                         |SEN|CEN|1|1|W|I|
|                                                   |   |   | | | |A|
+---------------------------------------------------+---+---+-+-+-+-+
```

CADR<31:8> Unused. Always read as 0's.

CADR<7:6> (SEN) Set Enable. Read/Write. These bits are used to selectively enable or disable each set within the cache. Cleared on power-up and the negation of DCOK.

CADR<7> When set, Set 2 of the cache is enabled. When cleared, Set 1 of the cache is disabled.

CADR<6> When set, Set 1 of the cache is enabled. When cleared, Set 0 of the cache is disabled.

CADR<5:4> (CEN) Cache Enable. Read/Write. These bits are used to selectively enable or disable the storing I-Stream and D-Stream references in the cache. Cleared on power-up and the negation of DCOK.

CADR<5> When set, I-Stream, memory space references are stored in cache. When cleared, I-Stream memory references are not stored in the cache.

CADR<4> When set, D-Stream, memory space references are stored in cache. When cleared, D-Stream memory references are not stored in the cache.

NOTE

The first-level cache can be disabled by either disabling both Set 1 and Set 2, or by not storing both I-Stream and D-Stream references.

NOTE

For maximum performance, the cache should be configured to store both I and D-Stream references. I-Stream only mode suffers from a degradation in performance from what would normally be expected relative to I and D-Stream mode and D-Stream only mode, due to the fact that invalidation of cache entries due to writes to memory by a DMA device are handled less efficiently. In I-Stream only mode, the entire first-level cache is flushed whenever an REI instruction is executed (Dec Standard 032 states that an REI instruction must be executed before running code out of a page of memory that has been updated.) whereas in the other two modes of operation, cache entries are invalidated on an individual basis, only if a DMA write operation results in a cache "hit".

CADR<3:2> Unused. Always read as 1's.

CADR<1> (WW) Write Wrong Parity. Read/write. When set, incorrect parity is stored in the first-level cache whenever it is written.

When cleared, correct parity is stored in the cache whenever the cache is written. Cleared on power-up and the negation of DCOK.

CADR<0> (DIA) Diagnostic Mode. Read/write. When cleared, writes to the CADR will cause the first-level cache to be flushed, (All valid bits set to the invalid state.) and the cache is configured for "normal" write-through operation. When set, writes to the CADR will not cause the first-level cache to be flushed, and all CPU write references write the data into the cache as well as main memory, irrespective of whether or not a cache hit occured. In addition, errors are ignored (they do not cause a machine check trap to be generated, or prevent data from being stored in the cache), Diagnostic Mode does not affect read references. Cleared on power-up and the negation of DCOK.

NOTE

Diagnostic Mode should only be selected when one and only one of the two sets are enabled. Operation of this mode with both or neither set enabled yields UNPREDICTABLE results.

5.2.6 Memory System Error Register (IPR 39)

The Memory System Error Register (MSER), Internal Processor Register 39, records the occurance of first-level cache hits, as well as parity errors on the CDAL Bus and in the first and second-level caches. This register is unique to CPU designs that use the CVAX chip. MSER<6:0> are sticky in the sense that they remain set until explicitly cleared. Additional errors occuring after a bit has been set can only set another bit. The MSER is explicitly cleared via the MTPR MSER instruction irrespective of the write data.

```
 3
 1                                        8 7 6 5 4 3 2 1 0
+----------------------------------------+-+-+-+-+-+-+-+-+-+
|                                        |H|D|M|M|S|S|D|T|
|                 0                      |M|A|C|C|T|T|A|A|
|                                        | |L|D|C|2|1|T|G|
+----------------------------------------+-+-+-+-+-+-+-+-+-+
```

MSER<31:8> Unused. Always read as 0's.

MSER<7> (HM) Hit/Miss. Read Only. Set on all cacheable references that hit the first-level cache. Cleared on all cacheable references that miss the first-level cache. Cleared on power-up and the negation of DCOK.

MSER<6> (DAL) DAL Parity Error. Read/Cleared on Write. This bit is set whenever a CDAL Bus or second-level cache data store parity error is detected. Cleared on power-up and the negation of DCOK.

MSER<5> (MCD) Machine Check - DAL Parity Error. Read/Cleared on Write. This bit is set whenever a machine check is caused by a CDAL Bus or second-level cache parity error. These errors will only generate machine checks on Demand D-Stream read references. Cleared on power-up and the negation of DCOK.

MSER<4> (MCC) Machine Check - First-Level Cache Parity Error. Read/Cleared on Write. This bit is set whenever a machine check is caused by a first-level cache parity error in the tag or data store. These errors will only generate machine checks on Demand D-Stream read references. Cleared on power-up and the negation of DCOK.

MSER<3> (ST2) Set 2 Parity Error Read/Cleared on Write. This bit is set when a parity error is detected in Set 2 of the first-level cache. Cleared on power-up and the negation of DCOK.

MSER<2> (ST1) Set 1 Parity Error Read/Cleared on Write. This bit is set when a parity error is detected in Set 1 of the first-level cache. Cleared on power-up and the negation of DCOK.

MSER<1> (DAT) Data Parity Error. Read/Cleared on Write. This bit is set when a parity error is detected in the data store of the first-level cache. Cleared on power-up and the negation of DCOK.

MSER<0> (TAG) Tag Parity Error. Read/Cleared on Write. This bit is set when a parity error is detected in the tag store of the first-level cache. Cleared on power-up and the negation of DCOK.

5.2.7 First-Level Cache Error Detection

Both the tag and data arrays in the first-level cache are protected by parity. Each 8-bit byte of cache data and the 20-bit tag is stored with an associated parity bit. The Valid Bit in the tag is not covered by parity. Odd data bytes are stored with odd parity and even data bytes are stored with even parity. The tag is stored with odd parity. The stored parity is valid only when the valid bit associated with the cache entry is set. Tag and data parity (on the entire longword) are checked on read references that hit the cache, while only tag parity is checked on CPU and DMA write references that hit the cache.

The action taken following the detection of a cache parity error depends on the reference type:

During a demand D-stream reference, the entire cache is flushed, the cache is disabled (CADR is cleared), the cause of the error is logged in MSER<5,3:0> and a machine check abort is initiated.

During a request D-stream reference, the entire cache is flushed, the cause of the error is logged in MSER<3:0>, but no abort occurs and the cache remains enabled.

During a request I-stream reference, the entire cache is flushed, the cause of the error is logged in MSER<3:0>, the prefetch is halted, but no machine check abort occurs, and the cache remains enabled.

5.3 Second-Level Cache

The KA650-AA also includes a 64KB, direct mapped, write through, second-level cache with a 200ns cycle time for longword transfers and 300ns cycle time for quadword transfers. CPU read references access one longword at a time. Cacheable references that miss the first-level cache can access up to one quadword at a time, while CPU writes can access a single byte at a time. A single parity bit is generated, stored and checked for each tag. The cache does not generate or check parity on each data byte. The parity bits stored with each data byte are taken from the CDAL parity lines when a data block is written or allocated. On second-level cache hits, these parity bits are placed back on the CDAL parity lines, so that parity checking on the data bytes is performed by the CVAX chip. This makes second-level cache data parity errors appear as CDAL Bus parity errors.

5.3.1 Second-Level Cache Organization

The second-level cache, being direct mapped, consists of a single storage array called Set 1. This array contains a 8K Row x 12-bit Tag Array and a 8K Row x 72-Bit Data Array.

KA650 Second-Level Cache Organization

```
                    Set 1
          +-------+------------------+
         /|       |                  |
        | |8Kx12- |    8Kx72-Bit     |
        | |Bit    |    Data Array    |
8K Rows<  |Tag    |                  |
        | |Array  |                  |
        | |       |                  |
        | |       |                  |
        | |       |                  |
        | |_______|__________________|
        | |_______|__________________|<-- Cache Entry
        | |       |                  |
        | |       |                  |
         \|       |                  |
          +-------+------------------+
         83     72 71                0
```

A row within the set corresponds to a single cache entry, so there are 8K entries in the entire cache. Each entry contains a 12-bit Tag Block and a 72-bit (eight-byte) Data Block. A cache entry is organized as shown below:

```
Cache Entry:

8             7 7
3             2 1                                        0
+-------------+------------------------------------------+
|  Tag Block  |               Data Block                 |
+-------------+------------------------------------------+
```

A Tag Block consists of a parity bit, a valid bit, and a 10-bit tag. A Tag Block is organized as shown below:

Tag Block:

```
         9                                           0
+---+---+-------------------------------------------+
| P | V |                   Tag                     |
+---+---+-------------------------------------------+
  |   |
  |   +--- Valid Bit
  +------- Parity Bit
```

A data block consists of eight bytes of data, each with an associated parity bit. The total data capacity of the cache is 8K eight-byte blocks, or 64K bytes. A data block is organized as shown below:

Data Block:

```
   6  5    5  4    4  4    3  3    3  2    2  1    1
   3  6    5  8    7  0    9  2    1  4    3  6    5  8    7  0
+-+----+-+----+-+----+-+----+-+----+-+----+-+----+-+----+
|P| B7 |P| B6 |P| B5 |P| B4 |P| B3 |P| B2 |P| B1 |P| B0 |
+-+----+-+----+-+----+-+----+-+----+-+----+-+----+-+----+
                                                  ^   ^
                                                  |   |
                                     Parity Bit -+   +-- Data Byte 0
```

5.3.2 Second-Level Cache Address Translation

Whenever a CPU reference that can be stored in the first-level cache causes a "miss" of the first-level cache, a quadword transfer is initiated on the CDAL Bus and the second-level cache is checked to determine if the contents of the location(s) being addressed are stored there. The cache is checked by translating the address as follows:

On non-cacheable references, the reference is never stored in the cache, so a second-level cache "miss" occurs, the main memory cycle is allowed to complete and the data is provided by the main memory controller.

On cacheable references, the physical address must be translated to determine if the contents of the referenced location(s) are resident in the cache. In this case, the Cache Index Field, bits <15:3> of the physical address, is used to select one of the 8K entries (rows) in the set. The Cache Tag Field, bits <28:16> of the physical address, is then compared to the Tag Block of the selected entry. Bits <28:26>

of this field are ignored since the second-level cache is designed to support a maximum of 64MB of main memory.

If a match occurs with the Tag Block of the entry, and the valid bit within the entry is set, then the contents of the location is contained in the cache and a second-level cache "hit" occurs. The Cache Displacement Field, bits <2:0> of the physical address, is used to select the longword within the block. Bits <1:0> of this field are ignored since the Byte Mask signals are used to select the desired byte(s) within a longword. Main memory cycles are initiated on all CDAL Bus cycles, but they are aborted before completion on second-level cache hits.

If there is no match, then the contents of the location is not contained in the second-level cache, a cache "miss" occurs, the main memory cycle is allowed to complete, and the data is provided by the main memory controller.

KA650 Second-level Cache Address Translation

```
       2 2                                     1 1
       9 8                                     6 5            3 2  0
      +-+---------------------------------------+--------------+----+
      | |              Cache Tag                |              |    |
      +-+---------------------------------------+--------------+----+
       |                |                                  |         |
       +-- I/O Space    |                     Cache Index -+         |
                        |                                  |         |
                +-------+                     +------------+         |
                |                             |                      |
                |                             |Cache Displacement   -+
                |  Valid Bit                  |                      |
                |    |                        |  +-------------------+
                |    V          Set 1         |  |
                |  +-+-----+----------------+ |  |
                |  | |     |                | |  |
                |  | | 10- |     64-Bit     | |  |
                |  | | Bit |   Data Block   | |  |
                |  | | Tag |                | |  |
                |  | |     |                | |  |
                |  | |     |                | |  |
                |  | |     |                | |  |
                |  |_|_____|________________| |  |
                |  |_|_____|________________|<-+  |
                |  | |     |       |        |    |
                |  | |     |       |        |    |
                |  | |     |       |        |    |
                |  +-+-----+-------|--------+    |
                |       |          |             |
                +-------|--+       |             |
                        |  |       |             |
                       -- --       |             |
                       \ V /       |             |
                        \_/        |             |
                         |         |             |
                   Set 1 | Match ? |             |
                         |         V             |
                         |      -------          |
                         |      \     /<---------+
                         +------>\   /
                                  ---
                                   |
                                   V
                                  Data
```

5.3.3 Second-Level Cache Data Block Allocation

On cacheable references that miss the first-level cache, a quadword read is initiated on the CDAL Bus. If the requested quadword cannot be found in the second-level cache, it is provided by the main memory controller, both caches allocate a data block for storing the entire quadword, and the requested longword is passed on to the CPU.

Due to the fact that the second-level cache is direct mapped, there is one and only one data block in the cache that can be allocated to a given quadword. This data block is determined by the Cache Index Field of the physical address of the quadword, which selects a unique row (data block) within the cache.

Since the KA650 supports 64MB (8M quadwords) of physical memory, up to 1K quadwords "share" each data block (row) of the cache. Contiguous programs larger than 64KB, or non-contiguous programs separated by 64KB will over-write themselves in the cache when cache data blocks are allocated for memory references separated by 64KB.

5.3.4 Second-Level Cache Behavior On Writes

On CPU generated write references, the second-level cache is "write through". All CPU write references that "hit" the second-level cache cause the contents of the referenced location in main memory to be updated as well as the copy in the cache.

On DMA write references that "hit" the cache, the cache entry containing the copy of the referenced location is invalidated.

5.3.5 Cache Control Register (CACR)

The Cache Control Register, address 2008 4000 (hex) controls the second-level cache and is unique to the KA650-AA. Only the low byte of this register should be written.

```
 3           2 2
 1           4 3                               8 7 6 5 4 3 2 1 0
+------------+--------------------------------+---+-+-+-+-+-+-+
|            |                                |   |C|C| | |W|D|
|      1     |     CACHE DIAGNOSTIC FIELD     |CSP|P|E|1|1|W|I|
|            |                                |   |E|N| | | |A|
+------------+--------------------------------+---+-+-+-+-+-+-+
```

CACR<31:24> Unused. Read as 1's.

CACR<23:8> (CACHE DIAGNOSTIC FIELD) Read only. See section 5.3.8 for details on this field.

CACR<7:6> (CSP) CVAX Cycle Speed. Read only. These bits are used to indicate the speed of the CVAX chip being used. They are encoded as follows:

<7:6>	Speed
00	reserved for future use
01	60ns
10	80ns
11	100ns

CACR<5> (CPE) Cache Parity Error. Read only. This bit is set whenever a cache tag parity error is detected. Cleared on power-up and the negation of DCOK.

CACR<4> (CEN) Cache Enable. Read/Write. When cleared, all references miss the cache except those to the Cache Diagnostic Space and the allocation of cache blocks is prevented. When set, the configuration of the first-level cache determines which types of references are stored. Cleared on power-up and the negation of DCOK.

NOTE

Whenever the second-level cache is disabled, it should be flushed before re-enabling to insure that data that may have become stale while the cache was disabled is not utilized.

CACR<3:2> Unused. Always read as one.

CACR<1> (WW) Write Wrong Parity. Read/write. When set, the parity bit stored in the second-level cache Tag Block is forced to a "1" whenever the cache is written. When cleared, correct parity is stored in the second-level cache Tag Block whenever the cache is written. Cleared on power-up and the negation of DCOK.

CACR<0> (DIA) Diagnostic Mode. Read/write. When set, the second-level cache is disabled, and writes to the cache diagnostic space set the valid bit for the entry that is written. When cleared, CACR<4> determines if the cache is enabled, and writes to the cache diagnostic space clear the valid bit for the entry that is written. Cleared on power-up and the negation of DCOK.

5.3.6 Second-Level Cache Error Detection

Both the tag and data arrays in the second-level cache are protected by parity. Each 8-bit byte of cache data and the 10-bit tag is stored with an associated parity bit. Odd data bytes are stored with odd parity and even data bytes are stored with even parity. The tag is stored with odd parity. The stored parity is valid only when the valid bit associated with the cache entry is set.

Tag parity is checked by the second-level cache logic on CPU read, CPU write and DMA write references that hit the cache. Tag parity is generated by the second-level cache logic on CPU write references that hit the cache and during the alocation of a cache block.

Data parity is checked on a byte basis by the CVAX chip for CPU read references that hit the cache. Data parity is taken directly from the CDAL Bus parity lines on CPU write operations that hit the cache and during the allocation of a cache block.

Upon detecting second-level cache tag parity errors the entire second-level cache is disabled, CACR<5> (second-level cache parity error) is set and an interrupt at IPL 1E through vector 0C (hex) is generated on each cycle until CACR<5> is cleared.

The action taken following the detection of a second-level cache data parity error is identical to that for CDAL Bus parity errors and depends on the reference type:

During a demand D-stream reference, the first-level cache entry is invalidated, the cause of the error is logged in MSER<6,4> and a machine check abort is initiated.

During a request D-stream and I-Stream references, The first-level cache entry is invalidated, the cause of the error is logged in MSER<4>, but no machine check is generated.

5.3.7 Second-Level Cache As Fast Memory

The second-level cache can be accessed as part of main memory for diagnostic purposes as well as for fast execution of bootstrap or self-test code. 1024 copies of the second-level cache data array appear starting at the first address .in the upper half of VAX Memory Space (physical addresses 1000 0000 - 13FF FFFF hex). This area is called the Cache Diagnostic Space. Read or write references to this address range will access the second-level cache as high speed (200ns) RAM. Read references will not affect the existing tag block for the accessed cache entry. When the Diagnostic Mode Bit CACR<0>, is cleared, write references will invalidate any cache entry that is accessed via the Cache Diagnostic Space. This prevents stale data from accumulating when the cache is used as high speed RAM. When the diagnostic mode bit is set, write references will set the valid bit in the Tag Block and write the Tag Field of the physical address into the tag of any entry that is accessed via the Cache Diagnostic Space. This allows any of the 1024 possible cache tag bit-patterns to be written into the Tag Block of any cache entry by writing to one of the 1024 copies of the cache entry.

Parity errors that occur while using the second-level cache as high speed RAM have the same effect as parity errors encountered during the normal operation of the cache.

NOTE

> To flush the second-level cache, each cache entry must be written via the Cache Diagnostic Space with the Diagnostic Mode Bit cleared.

5.3.8 Second-Level Cache Fault Isolation

The state of the second-level cache Tag can be read directly in the Cache Diagnostic Field of the CACR via a range of addresses in the VAX I/O Space. This information can be used by diagnostic programs to improve the level of isolation of second-level cache faults. To access this information, 777,216 (16M) copies of the CACR are created in a 64MB block of VAX I/O Space starting at 3800 0000 (hex). This is done by writing 3800 0000 (hex) to the CACR Address Match Register (address 2014 0130 hex), and 03FF FFFC (hex) to the CACR Address Mask Register (address 2014 0134).

NOTE

> The CACR Address Match and CACR Address Mask Registers must be reset to their original values at the completion of testing.

Read references to this address range will then return the state of the stored parity bit, valid bit and ten-bit tag from the Tag Block of the second-level cache entry selected by the Cache Index Field of the physical address. The parity tree output which is used to check parity on the Tag Block, the predictive parity tree output which calculates parity on bits <25:16> of the Cache Tag Field of the physical address, and the outputs of the comparators that determine if a match has occured between the Cache Tag Field of the physical address and the Tag Block of the entry are also provided.

The 64MB block of adresses is divided into 1024 (1K), 64KB blocks. The contents of the entire Cache Tag Array can be read by referencing all even longword addresses in any of the 64KB (16K longword) blocks. (Since there are two physical longword addresses associated with the same Cache Tag Block, references to successive addresses with the same Cache Index Field will return the same information.)

The state of bits <25:16> of the Cache Tag Field of the physical address is the same for all addresses within each 64KB block. One address must be read from each of the 1024 blocks to generate all the possible states of the Cache Tag Field of the physical address, which is the input to both the predictive parity tree and the comparators.

Each copy of the CACR appears as below:

```
 3           2 2 2 2 2 1 1 1
 1           4 3 2 1 0 9 8 7                   8 7 6 5 4 3 2 1 0
+-------------+-+-+-+-+-+-+-------------------+---+-+-+-+-+-+-+
|             |P|P|M|M| | |                   |   |C|C| | |W|D|
|      1      |T|P|O|O|P|V|    10-Bit Tag     |CSP|P|E|1|1|W|I|
|             |O|O|2|1| | |                   |   |E|N| | | |A|
+-------------+-+-+-+-+-+-+-------------------+---+-+-+-+-+-+-+
```

CACR<31:24> Unused. Read as 1's.

CACR<23> (PTO) Parity Tree Output. Read Only.

CACR<22> (PPO) Predictive Parity Tree Output. Read Only.

CACR<21> (MO2) Match Output 2. Read Only.

CACR<20> (MO1) Match Output 1. Read Only.

CACR<19> (P) Tag Block Parity Bit. Read Only.

CACR<18> (V) Tag Block Valid Bits. Read Only.

CACR<17:8> 10-Bit Tag.

CACR<7:0> As defined in section

6 KA650-AA PROCESSOR MAIN MEMORY SYSTEM

The KA650-AA includes a main memory controller implemented via a single VLSI chip called the CMCTL. The KA650-AA Main Memory Controller communicates with the MS650 memory boards over the MS650 Memory Interconnect, which utilizes the CD interconnect for the address and control lines and a 50-pin, ribbon cable for the data lines. It supports up to four MS650 memory boards, for a maximum of 64MB of ECC memory.

The controller supports either masked or unmasked, synchronous, longword read and write references generated by the CPU and synchronous, quadword transfers generated by cacheable CPU references that miss the first-level cache. Single longword read references and the first longword read reference in a quadword transfer require 400ns to complete. The second longword read reference in a quadword transfer requires 200ns to complete. Single, unmasked, longword write references by the CPU require 300ns to complete. Single, masked, longword write references by the CPU require 500ns to complete. A read reference that was aborted by a second-level cache hit requires 300ns to complete.

The controller also supports asynchronous DMA reads and writes from the Q-22 Bus interface.

The main memory controller contains eighteen registers. Sixteen registers are used to configure each of the sixteen possible banks in main memory. One register is used to control the operating mode of all memory banks and one register captures state on main memory errors.

6.1 Main Memory Organization

Main memory is logically and physically divided into four boards which correspond to the four possible MS650 memory expansion modules that can be attached to a KA650-AA. Each board can contain zero (no memory module present), two (MS650-AA present), or four (MS650-BA present), memory banks. Each bank contains 1,048,576 (1M) aligned longwords. Each aligned longword is divided into four data bytes and is stored with seven ECC check bits, resulting in a memory array width of 39 bits.

6.2 Main Memory Addressing

The KA650-AA Main Memory Controller is capable of controlling up to 16 banks of RAM, each bank consisting 4MB of data. Each bank of main memory has a programmable base address, determined by the state of bits <25:22> of the Main Memory Configuration Register associated with the bank.

A 4MB bank is accessed when bit <29> of the physical address is equal to 0, indicating a VAX Memory Space read/write reference, bits <28:26>

of the physical address are equal to zero, indicating a reference within the range of the main memory controller, and the bank number of the bank matches bits <25:22> of the physical address. The remainder of the physical address (bits <21:2>) are used to determine the row and column of the desired longword within the bank. The Byte Mask lines are ignored on read operations, but are used to select the proper byte(s) within a longword during masked longword write references.

The main memory controller accesses main memory on read/write references in parallel with the address translation process in the second-level cache. On CPU read references that "hit" the second-level cache the memory controller reads the longword from main memory, but the operation is aborted before the data gets placed on the CDAL Bus.

6.3 Main Memory Behavior On Writes

On unmasked CPU write references, the main memory controller operates in "dump and run" mode, terminating the CDAL Bus transaction after latching the data, but before checking CDAL Bus parity, calculating the ECC check bits, and transfering the data to main memory. This allows Main Memory to keep up with the second-level cache without impacting the CPU write performance.

On unmasked DMA write references by the Q-22 Bus Interface, CDAL Bus parity is not checked, but the ECC check bits are calculated, and the data is transferred to main memory before the CDAL Bus transaction is terminated.

On masked CPU or DMA write references , CDAL Bus parity is checked (for CPU writes only), the referenced longword is read from main memory, the ECC code checked, the check bits recalculated to acount for the new data byte(s), and the longword is rewritten before the CDAL transaction is terminated.

6.4 Configuring Main Memory

To configure main memory, a SIGNATURE READ REQUEST is issued to each memory board by setting bit <5> of the the first MEMCSR associated with each memory board (MEMCSR0 for board one, MEMCSR4 for board two, MEMCSR8 for board three, and MEMCSR12 for board four). This will return the characteristics of the banks on each memory board in bits <4:0> of all four MEMCSR registers associated with each board. The number of banks implemented on each board is then obtained by reading bits <1:0> (BANK USAGE) of MEMCSR0, MEMCSR4, MEMCSR8 and MEMCSR12. This information is then used to enable and assign bank numbers (0-16 decimal) to each bank being used, starting with the first bank of the first memory module. Unused banks should be left disabled with a bank number of 0.

6.5 Main Memory Configuration Registers (MEMCSR0 - MEMCSR15)

There are sixteen Main Memory Configuration Registers which are used to configure the sixteen possible banks of main memory. These registers are unique to CPU designs that use the CMCTL memory controller chip. All sixteen registers have the same format. MEMCSR0 - MEMCSR3 configure banks one through four on the first memory module. MEMCSR4 - MEMCSR7 configure banks one through four on the second memory module. MEMCSR8 - MEMCSR11 configure banks one through four on the third memory module. MEMCSR12 - MEMCSR15 configure banks one through four on the fourth memory module.

NOTE

> All four banks may not be implemented on each memory module and all four memory modules may not be present in the system. Also, bits <4:0>, which indicate the characteristics of the bank, are common to all banks within a memory board. All four registers associated with a board are loaded with this information after a single SIGNATURE READ REQUEST to any one of the four registers.

The addresses of these sixteen registers are listed below:

MEMCSRn	Address (Hex)
0	2008 0100
1	2008 0104
2	2008 0108
3	2008 010C
4	2008 0110
5	2008 0114
6	2008 0118
7	2008 011C
8	2008 0120
9	2008 0124
10	2008 0128
11	2008 012C
12	2008 0130
13	2008 0134
14	2008 0138
15	2008 013C

Format for MEMCSR0 - MEMCSR15:

```
 3 3       2 2     2 2
 1 0       6 5     2 1                         6 5 4 3 2 1 0
+-+---------+-------+---------------------------+-+---+-+---+
| |   MBZ   |       |           MBZ             | |   | |   |
+-+---------+-------+---------------------------+-+---+-+---+
 |              |                                |  |  |  |
 |              +---- BANK NUMBER                |  |  |  |
 +------------------- BANK ENABLE                |  |  |  |
                                                 |  |  |  |
              SIGNATURE READ REQUEST ------------+  |  |  |
                     BANK ERROR MODE ---------------+  |  |
                           BANK SIZE ------------------+  |
                          BANK USAGE ---------------------+
```

MEMCSRn<31> - BANK ENABLE: Read/Write. When set, this bit indicates that the bank number (MEMCSRn<25:22>) is valid, and addressing of the bank is enabled. When cleared, this bit indicates the base address is invalid and addressing of the bank is disabled. This bit is cleared on power-up and the negation of DCOK. This bit should be set to enable for all banks in use when the bank number is written by the firmware memory configuration routine. In addition, this bit may be used by diagnostics to selectively disable banks during testing.

MEMCSRn<30:29> - NOT USED: Always read as 0, must be written as 0.

MEMCSRn<25:22> - BANK NUMBER: Read/Write. This field determines the base address of the associated bank of main memory. This field is cleared on power-up and the negation of DCOK. This field should be written with the proper bank number (0-16 decimal) for all banks in use by the firmware memory configuration routine.

MEMCSRn<19:6> - NOT USED: Always read as 0's, must be written as 0.

MEMCSRn<5> - SIGNATURE READ REQUEST: Read/Write. This bit is used by the memory configuration routine to cause the memory controller to read the memory board characteristics. When set, the memory controller reads the signature register of the memory module containing the associated bank and loads the information into MEMCSRn<4:0> of all the registers associated with four possible banks on this board. This bit is cleared by the main memory controller upon completion of the signature read operation. This bit is also cleared on power-up and the negation of DCOK.

MEMCSRn<4:3> - BANK ERROR MODE: Read only. After a signature read is performed, this field indicates the type of error detection/correction used for the associated bank. (The same for all banks within a board.) This field should read as 10 (binary) for both the MS650-AA and MS650-BA indicating 7-bit ECC is implemented. This field is cleared on power-up and the negation of DCOK.

MEMCSRn<2> - BANK SIZE: Read only. After a signature read is performed, this bit indicates the size of the associated memory bank. (The same for all banks within a board, and not valid if the register corresponds to a bank that is "not in use".) When set, the bank size is 4MB. When cleared, the bank size is 1MB. This bit should be set for both the This information is used by the memory initialization routine to configure the base address of each bank in main memory.

MEMCSRn<1:0> - BANK USAGE: Read only. After a signature read is performed, this field indicates the number of banks present on the memory board containing the associated bank. (The same for all banks within a board.) This field is cleared on power-up and the negation of DCOK.

The field is encoded as follows:

BANK USE<1:0>	NUMBER OF POPULATED BANKS
00	0 (no banks in use, no module present)
01	1 (not used, illegal state)
10	2 (first two banks in use, MS650-AA module present)
11	4 (all four banks in use, MS650-BA module present)

6.6 Main Memory Error Status Register (MEMCSR16)

The Main Memory Status Register, address 2008 0140, is used to capture main memory error data.

Format for MEMCSR16:

```
 3 3 2 2
 1 0 9 8                                  9 8 7 6           0
+-+-+-+----------------------------------+-+-+-------------+
| | | |      PAGE ADDRESS OF ERROR       | | |  SYNDROME   |
+-+-+-+----------------------------------+-+-+-------------+
 | | |                                    | |         |
 | | |              DMA ERROR LOG ----------+ |         |
 | | |         CDAL BUS ERROR LOG ------------+         |
 | | |         ECC ERROR SYNDROME ------------------------+
 | | |
 | | +- CRD ERROR LOG REQUEST
 | +--- RDS HIGH ERROR RATE
 +----- RDS ERROR LOG REQUEST
```

MEMCSR16<31> - RDS ERROR LOG REQUEST: Read/Write to clear. When set, an uncorrectable ECC error occurred during a memory read or masked write reference. Cleared by writing a 1 to it. Cleared on power-up and the negation of DCOK.

MEMCSR16<30> - RDS HIGH ERROR RATE: Read/Write to clear. When set, an uncorrectable ECC error occurred while the RDS ERROR LOG REQUEST bit was set, indicating multiple uncorrectable memory errors. Cleared by writing a 1 to it. Cleared on power-up and the negation of DCOK.

MEMCSR16<29> - CRD ERROR LOG REQUEST: Read/Write to clear. When set, a correctable (single bit) error occurred during a memory read or masked write reference. Cleared by writing a 1 to it. Cleared on power-up and the negation of DCOK.

MEMCSR16<28:9> - PAGE ADDRESS OF ERROR: Read only. This field identifies the page (512 byte block) containing the location that caused the memory error. In the event of multiple memory errors, the types of errors are prioritized and the page address of the error with the highest priorty is captured. Cleared on power-up and the negation of DCOK.

The types of error conditions follow in order of priority:

1. CDAL Bus parity errors during a CPU write references, as logged by the CDAL BUS ERROR LOG bit.

2. Uncorrectable ECC errors during a CPU or DMA read or masked write references, as logged by the RDS ERROR LOG bit.

3. Correctable ECC errors during a CPU or DMA read or masked write references, as logged by CRD ERROR LOG bit.

MEMCSR16<8> - DMA ERROR LOG: Read/Write to clear. When set, an error occured during a DMA read or write references. Cleared by writing a 1 to it. Cleared on power-up and the negation of DCOK.

MEMCSR16<7> - CDAL BUS ERROR LOG: Read/Write to clear. When set, a CDAL Bus parity error occurred on a CPU write reference. Cleared by writing a 1 to it. Cleared on power-up and the negation of DCOK.

MEMCSR16<6:0> - ERROR SYNDROME: Read only. This field stores the error syndrome which, on correctable errors, indicates the bit position in the page that caused the memory error. Cleared on power-up and the negation of DCOK.

6.7 Main Memory Control And Diagnostic Status Register (MEMCSR17)

The Main Memory Control and Diagnostic Status Register, address 2008 0144, is used to control the operating mode of the main memory controller as well as to store diagnostic status information.

Format for MEMCSR17:

```
 3                           1 1 1 1 1 1
 1                           5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-----------------------------+-+-+-+-+-+-+-+-+-+-------------+
|              MBZ            | | | | | | | | | |             |
+-----------------------------+-+-+-+-+-+-+-+-+-+-------------+
                                | | | | | | | |         |
              DMA BUS MODE  ------+ | | | | | | |         |
 MAIN MEMORY CYCLE SELECT  --------+ | | | | | |         |
    ENABLE CRD INTERRUPT   ----------+ | | | | |         |
   FORCE REFRESH REQUEST   ------------+ | | | |         |
    DISABLE ERROR DETECT   --------------+ | | |         |
    FAST DIAGNOSTIC TEST   ----------------+ | |         |
             BOARD ERROR   ------------------+ |         |
   DIAGNOSTIC CHECK MODE   --------------------+         |
              CHECK BITS   ------------------------------+
```

MEMCSR17<31:15> - Unused. This field reads as 0, must be written as 0.

MEMCSR17<14> - DMA BUS MODE: Read/Write. When cleared, DMA transfers are handled asynchronously by the memory controller. When set, DMA transfers are handled synchronously by the memory controller. This bit should always be cleared by the firmware memory configuration routine for proper operation of the Q-22 Bus Interface. Cleared on power-up and the negation of DCOK.

MEMCSR17<13> - MAIN MEMORY CYCLE SELECT: Read/Write. When cleared, longword reads and the first longword in quadword reads occur in four CPU cycles (400ns) and the second longword in a quadword read occurs in two CPU cycles (200ns). When set, longword reads and the first longword in quadword reads occur in five CPU cycles (500ns) and the second longword in a quadword read occurs in three CPU cycles (300ns). This bit should be set by the firmware memory configuration routine if CACR<7:6> equals 11 (binary) and cleared otherwise. Cleared on power-up and the negation of DCOK.

MEMCSR17<12> - ENABLE CRD INTERRUPT: Read/write. When cleared, single-bit errors are corrected by the ECC logic, but no interrupt is generated. When set, single-bit errors are corrected by the ECC logic and they cause an interrupt to be generated at IPL 1A with a vector of 54 (hex). This bit has no effect on the capturing of error information in MEMCSR16, or on the reporting of uncorrectable errors. Cleared on power-up and the negation of DCOK.

MEMCSR17<11> - FORCE REFRESH REQUEST: Read/write. When cleared, the refresh control logic operates in normal mode. When set, one memory refresh operation occurs immediately after the MEMCSR write reference that set this bit. Setting this bit provides a mechanism for speeding up the testing of the refresh logic during manufacturing test of the controller chip. This bit is cleared by the memory controller upon completion of the signature read operation. Cleared on power-up and the negation of DCOK.

MEMCSR17<10> - DISABLE MEMORY ERROR DETECT: Read/write. When set, error detection and correction (ECC) is disabled, so all memory errors go undetected. When cleared error detection, correction, state capture and reporting is enabled. Cleared on power-up and the negation of DCOK.

MEMCSR17<9> - FAST DIAGNOSTIC TEST: Read/Write. This bit provides a mechanism for speeding up the diagnostic testing of main memory. When set, all main memory banks are read and written in parallel and MEMCSR17<08> is enabled as a error log bit for indicating memory board errors during read references. When cleared, this bit has no effect on memory reads or writes. Cleared on power-up and the negation of DCOK.

NOTE

Due to the fact that FAST DIAGNOSTIC TEST mode draws large amounts of power, memory read and write referneces executed in this mode should have a main memory duty cycle of no more than TBD %.

MEMCSR17<8> - BOARD ERROR: Read/Write to clear. Set during a FAST DIAGNOSTIC TEST read reference when the contents of all memory banks is not identical. Cleared by writing a one to it. Cleared on power-up and the negation of DCOK.

MEMCSR17<7> - DIAGNOSTIC CHECK MODE: Read/write. When set, the contents of MEMCSR17<6:0> are written into the 7 ECC check bits of the location during a memory write reference. When cleared, a memory read reference will load the state of the 7 ECC check bits of the location that was read into MEMCSR17<6:0>. Cleared on power-up and the negation of DCOK.

NOTE

DIAGNOSTIC CHECK MODE is restricted to unmasked memory write referneces. No masked write references are allowed when DIAGNOSTIC CHECK MODE is enabled.

MEMCSR17<6:0> - CHECK BITS: Read/write. When the DIAGNOSTIC CHECK MODE bit is set, the contents of MEMCSR17<6:0> are substituted for the check bits that are generated by the ECC generation logic during a write reference. When the DIAGNOSTIC TEST MODE bit is cleared, memory read reference load the state of the 7 ECC check bits of the location that was read into MEMCSR<6:0>. Cleared on power-up and the negation of DCOK.

6.8 Main Memory Error Detection And Correction

The KA650-AA Main Memory Controller generates CDAL Bus parity on CPU read references, and checks CDAL Bus parity on CPU write references.

The actions taken following the detection of a CDAL Bus parity error depend on the type of write reference.

For unmasked CPU write references, incorrect check bits are written to main memory along with the data and an interrupt is generated at IPL 1D through vector 60 (hex) on the next cycle and MCSR16<17> is set. The incorrect check bits are determined by calculating the seven "correct" check bits, and complementing the three least significant bits.

For masked CPU write references, incorrect check bits are written to main memory along with the data, unless an uncorrectable error is detected during the read portion, and an interrupt is generated at IPL1D through vector 60 (hex) on the same cycle and MEMCSR16<17> is set. The incorrect check bits are determined by calculating the seven "correct" check bits, and complementing the three least significant bits.

The memory controller protects main memory by using a 32-bit Modified Hamming code to "encode" the 32-bit data longword into seven Check Bits. This allows the controller to detect and correct single-bit errors in the data field and detect single bit errors in the check bit field and double-bit errors in the data field. The most likely causes of these errors are failures in either the memory array or the 50-pin cable.

Upon detecting a correctable error on a read reference or the read portion of a masked write reference, the data is corrected (if it is in the data field), before placing it on the CDAL Bus, or back in main memory, an interrupt is generated at IPL1A through vector 54 (hex), bit <29> of MEMCSR16 is set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, and bits <6:0> are loaded with the error syndrome which indicates which bit was in error.

NOTE

> The corrected data is not rewritten to main memory, so the single bit error will remain there until rewritten by software.

Upon detecting an uncorrectable error, the action depends on the type of reference being performed.

On a demand read reference, the cycle is aborted, a machine check occurs, bit <31> of MEMCSR16 is set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, and bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable.

On a request read reference, the prefetch or fill cycle is aborted, but no machine check occurs, bit <31> of MEMCSR16 is set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, and bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable.

On the read portion of masked write reference, the cycle is aborted, a machine check occurs, bit <31> of MEMCSR16 is set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, and bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable.

7 KA650-AA CONSOLE SERIAL LINE

The console serial line provides the KA650-AA processor with a full duplex, RS-423 EIA, serial line interface, which is also RS-232C compatible. The only data format supported is 8-bit data with no parity and one stop bit. The four Internal Processor Registers (IPR's) that control the operation of the console serial line are a super-set of the VAX Console Serial Line Registers described in DEC STD 032.

7.1 Console Registers

There are four registers associated with the Console Serial Line Unit. They are accessed as IPR's 32-35 (decimal):

IPR	Register Name	Mnemonic
32	Console Receiver Control/Status	RXCS
33	Console Receiver Data Buffer	RXDB
34	Console Transmit Control/Status	TXCS
35	Console Transmit Data Buffer	TXDB

7.1.1 Console Receiver Control/Status Register (IPR 32)

The Console Receiver Control/Status Register (RXCS), Internal Processor Register 32, is used to control and report the status of incoming data on the console serial line.

```
 3
 1                                        8 7 6 5            0
+------------------------------------------+-+-+-------------+
|           unused, returns 0              | | | 0 0 0 0 0 0 |
+------------------------------------------+-+-+-------------+
                                            | |
                                            | |
                      RX DONE --------------+ |
                      RX IE   ----------------+
```

RXCS<31:8> Unused. Read as zeros.

RXCS<7> (RX DONE) Receiver Done. Read-only. This bit is set when an entire character has been received and is ready to be read from the RBUF Register. This bit is automatically cleared when RBUF is read. It is also cleared on power-up and the negation of DCOK.

RXCS<6> (RX IE) Receiver Interrupt Enable. Read/Write. If RX DONE and RX IE are both set, a program interrupt is requested. This bit is cleared on power-up and the negation of DCOK.

RXCS<5:0> Unused. Read as zeros.

7.1.2 Console Receiver Data Buffer (IPR 33)

The Console Reciever Data Buffer (RXDB), Internal Processor Register 33, is used to buffer incoming data on the serial line and capture error information.

```
 3                           1 1 1 1 1 1 1
 1                           6 5 4 3 2 1 0   8 7                  0
+----------------------------+-+-+-+-+-+-----+--------------------+
|             0              | | | |0| |0 0 0|                    |
+----------------------------+-+-+-+-+-+-----+--------------------+
                               | | |   |                   |
                  ERR ---------+ | |   |                   |
                  OVR ERR -------+ |   |                   |
                  FRM ERR ---------+   |                   |
                  RCV BRK -------------+                   |
                  RECEIVED DATA BITS ----------------------+
```

RXDB<31:16> Unused. Always read as zero.

RXDB<15> (ERR) Error. Read only. This bit is set if RBUF <14> or <13> is set. It is clear if these two bits are clear. This bit cannot generate a program interrupt. Cleared on power-up and the negation of DCOK.

RXDB<14> (OVR ERR) Overrun Error. Read only. This bit is set if a previously received character was not read before being overwritten by the present character. Cleared on power-up and the negation of DCOK.

RXDB<13> (FRM ERR) Framing Error. Read only. This bit is set if the present character did not have a valid stop bit. Cleared on power-up and the negation of DCOK.

NOTE

> Error conditions remain present until the next character is received, at which point, the error bits are updated.

RXDB<12> Unused. This bit always reads as "0".

RXDB<11> (RCV BRK) Received Break. Read only. This bit is set at the end of a received character for which the serial data input remained in the SPACE condition for 20 bit times. RCV BRK then remains set until the register is read. Cleared on power-up and the negation of DCOK.

RXDB<10:8> Unused. These bits always read as "0".

RXDB<7:0> Received Data Bits. Read only. These bits contain the last received character.

7.1.3 Console Transmitter Control/Status Register (IPR 34)

The Console Transmitter Control/Status Register (TXCS), Internal Processor Register 34, is used to control and report the status of outgoing data on the console serial line.

```
 3
 1                                          8 7 6 5   3 2 1 0
+------------------------------------------+-+-+-----+-+-+-+
|             unused, returns 0            | | |0 0 0| |0| |
+------------------------------------------+-+-+-----+-+-+-+
                                            | |       |   |
                  TX RDY ------------------+ |       |   |
                  TX IE  --------------------+       |   |
                  MAINT  ----------------------------+   |
                  XMIT BRK ------------------------------+
```

TXCS<31:8> Unused. Read as zeros.

TXCS<7> (TX RDY) Transmitter Ready. Read only. This bit is cleared when XBUF is loaded and set when XBUF can receive another character. This bit is set on power-up and the negation of DCOK.

TXCS<6> (TX IE) Transmitter Interrupt Enable. Read/Write. If both TX RDY and TX IE are set, a program interrupt is requested. This bit is cleared on power-up and the negation of DCOK.

TXCS<5:3> Unused. Read as zeros.

TXCS<2> (MAINT) Maintenance. Read/Write. This bit is used to facilitate a maintenance self-test. When MAINT is set, the external serial input is set to MARK and the serial output is used as the serial input. This bit is cleared on power-up and the negation of DCOK.

TXCS<1> Unused. Read as zero.

TXCS<0> (XMIT BRK) Transmit Break. Read/Write. When this bit is set, the serial output is forced to the SPACE condition. While XMIT BRK is set, the transmitter will operate normally, but the output line will remain low. Thus, software can transmit dummy characters to time the break. This bit is cleared on power-up and the negation of DCOK.

7.1.4 Console Transmitter Data Buffer (IPR 35)

The Console Transmitter Data Buffer (TXDB), Internal Processor Register 35, is used to buffer outgoing data on the serial line.

TXDB<31:8> Unused.

TXDB<7:0> Transmitted Data Bits. Write only. These bits are used to load the character to be transmitted on the console serial line.

7.2 Break Response

The console serial line unit recognizes a BREAK condition which consists of 20 consecutively received SPACE bits. If the console detects a valid break condition, the RBR bit is set in the RXDB register. If the break was the result of 20 consecutively received SPACE bits, the FRE bit is also set. If halts are enabled (HLT ENB asserted on the 20-pin connector), the KA650-AA will halt and transfer program control to ROM location 2004 0000 when the RBR bit is set. RBR is cleared by reading RXDB. Another MARK followed by 20 consecutive SPACE bits must be received to set RBR again.

7.3 Baud Rate

The receive and transmit baud rates are always identical and are controlled by the SSC Configuration Register bits <14:12>.

The user selects the desired baud rate through the Baud Rate Select signals (BRS <2:0> L) which are received from an external 8-position switch via the 20-pin connector mounted at the top of the module. The KA650-AA firmware reads this code from Boot and Diagnostic Register bits <6:4> and loads it into SSC Configuration Register bits <14:12>.

The table below shows the Baud Rate Select signal voltage levels (H or L), the corresponding INVERTED code as read in the Boot and Diagnostic Register bits <6:4>, and the code that should be loaded into SSC Configuration Register bits <14:12>:

Baud Rate	BRS <2:0>	BDR <6:4>	SSC <14:12>
300	HHH	000	000
600	HHL	001	001
1200	HLH	010	010
2400	HLL	011	011
4800	LHH	100	100
9600	LHL	101	101
19200	LLH	110	110
38400	LLL	111	111

7.4 Console Interrupt Specifications

The console serial line receiver and transmitter both generate interrupts at IPL 14. The reciever interrupts with a vector of F8 (hex), while the transmitter interrupts with a vector of FC (hex).

8 KA650-AA TOY CLOCK AND TIMERS

The KA650-AA clocks include Time Of Year Clock (TODR) as defined in DEC STD 032, a subset Interval Clock (subset ICCS), as defined in DEC STD 032, and two additional programmable timers modeled after the VAX Standard Interval Clock.

8.1 Time-of-Year Clock (IPR 27)

The KA650-AA Time-of-Year clock (TODR), Internal Processor Register 27, forms an unsigned 32-bit binary counter that is driven from a 100Hz oscillator, so that the least significant bit of the clock represents a resolution of 10 milliseconds. The register counts only when it contains a non-zero value.

```
 3
 1                                                           0
+------------------------------------------------------------+
|               time of year since setting                   |
+------------------------------------------------------------+
```

Time of Year Clock (TODR)

The Time Of Year Clock is maintained during power failure by battery backup circuitry which interfaces, via the external connector, to a set of batteries which are mounted on the CPU distribution Insert. The (TODR) will remain valid for greater than <TBD> hours when using three Ni Cad batteries in series.

The SSC Configuration Register contains a Battery Low bit which, if set after initialization, the TODR is cleared, and will remain at zero until software writes a non-zero value into it.

8.2 Interval Timer

The KA650-AA Interval Timer is implemented according to DEC STD 032 for subset processors. The Interval Clock Control/Status Register (ICCS), Privileged Register 24 is implemented as the standard subset of the Standard VAX ICCS; while NICR and ICR are not implemented.

```
 3
 1                                         7 6 5           0
+------------------------------------------+-+-------------+
|                0                         |I|      0      |
|                                          |E|             |
+------------------------------------------+-+-------------+
```

ICCS<31:7> Unused. Read as 0's.

ICCS<6> (IE) Interrupt Enable. Read/Write. This bit enables and disables the interval timer interrupts. When the bit is set, an interval timer interrupt is requested every 10 msec. When the bit is clear, interval timer interrupts are disabled. This bit is cleared on power-up and the negation of DCOK.

Interval timer requests are posted at IPL 16 with a vector of CO: the interval timer is the highest priority device at this IPL.

8.3 Programmable Timers

The KA650-AA features two programmable timers. Although they are modeled after the VAX Standard Interval Clock, they are accessed as I/O space registers (rather than as Internal Processor Registers) and a control bit has been added which stops the timer upon overflow. If so enabled, the timers will interrupt at IPL 14 upon overflow. The interrupt vector is programmable.

Each timer is composed of four registers: a Timer n Control Register, a Timer n Interval Register, a Timer n Next Interval Register, and a Timer n Interrupt Vector Register, where n represents the timer number (0 or 1).

8.3.1 Timer Control Registers (TCR0-TCR1)

The KA650-AA has two Timer Control Registers, one for controlling timer 0 (TCR0), and one for controlling timer 1 (TCR1). TCR0 is accessible at address 2014 0100 (hex), and TCR1 is accessible at 2014 0110 (hex).

```
 3 3
 1 0                                        8 7 6 5 4   2   0
+-+-----------------------------------------+-+-+-+-+-+-+-+-+
|E|                                         |I|I|S|X|M|S|M|R|
|R|                 MBZ                     |N|E|G|F|B|T|B|U|
|R|                                         |T| |L|R|Z|P|Z|N|
+-+-----------------------------------------+-+-+-+-+-+-+-+-+
```

TCRn<31> (ERR) Error. Read/Write to clear. This bit is set whenever the Timer Interval Register overflows and INT is already set. Thus, ERR indicates a missed overflow. Writing a 1 to this bit clears it. Cleared on power-up and the negation of DCOK.

TCRn<30:8> Unused. Read as 0's, must be written as 0's.

TCRn<7> (INT) Read/Write to clear. This bit is set whenever the Timer Interval Register overflows. If IE is set when INT is set, an interrupt is posted at IPL 14. Writing a 1 to this bit clears it. Cleared on power-up and the negation of DCOK.

TCRn<6> (IE) Read/Write. When this bit is set, the timer will interrupt at IPL 14 when INT is set. Cleared on power-up and the negation of DCOK.

TCRn<5> (SGL) Read/Write. Setting this bit causes the Timer Interval

Register to be incremented by 1 if the RUN bit is cleared. If RUN is set, then writes to SGL are ignored. This bit always reads as 0. Cleared on power-up and the negation of DCOK.

TCRn<4> (XFR) Read/Write. Setting this bit causes the Timer Next Interval Register to be copied into the Timer Interval Register. This bit is always read as 0. Cleared on power-up and the negation of DCOK.

TCRn<3> Unused. Read as 0's, must be written as 0's.

TCRn<2> (STP) Read/Write. This bit determines whether the timer stops after an overflow. If the STP bit is set at overflow, RUN is cleared by the hardware at overflow and counting stops. Cleared on power-up and the negation of DCOK.

TCRn<1> Unused. Read as 0's, must be written as 0's.

TCRn<0> (RUN) Read/Write. When set, the Timer Interval Register is incremented once every microsecond. INT is set when the timer overflows. If the STP bit is set at overflow, RUN is cleared by the hardware at overflow and counting stops. When RUN is clear, the timer Interval Register is not incremented automatically. Cleared on power-up and the negation of DCOK.

8.3.2 Timer Interval Registers (TIR0-TIR1)

The KA650-AA has two Timer Interval Registers, one for timer 0 (TIR0), and one for timer 1 (TIR1). TIR0 is accessible at address 2014 0104 (hex), and TIR1 is accessible at 2014 0114 (hex).

The Timer Interval Register is a read only register containing the interval count. When the run bit is 0, writing a 1 increments the register. When the RUN bit is 1, the register is incremented once every microsecond. When the counter overflows, the INT bit is set, and an interrupt is posted at IPL14 if IE is set. Then, if STP is 0, the RUN bit is cleared and counting stops. The maximum delay that can be specified is approximately 1.2 hours. This register is cleared on power-up and the negation of DCOK.

```
 3
 1                                                             0
+---------------------------------------------------------------+
|                     Timer Interval Register                   |
+---------------------------------------------------------------+
```

8.3.3 Timer Next Interval Registers (TNIR0-TNIR1)

The KA650-AA has two Timer Next Interval Registers, one for timer 0 (TNIR0), and one for timer 1 (TNIR1). TNIR0 is accessible at address 2014 0108 (hex), and TNIR1 is accessible at 2014 0118 (hex).

This read/write register contains the value which is written into the Timer Interval Register after overflow, or in response to a 1 written to XFR. This register is cleared on power-up and the negation of DCOK.

```
 3
 1                                                             0
+--------------------------------------------------------------+
|                  Timer Next Interval Register                |
+--------------------------------------------------------------+
```

8.3.4 Timer Interrupt Vector Register (TIVR0-TIVR1)

The KA650-AA has two Timer Interrupt Vector Registers, one for timer 0 (TIVR0), and one for timer 1 (TIVR1). TIVR0 is accessible at address 2014 010C (hex), and TIVR1 is accessible at 2014 011C (hex).

This read/write register contains the timer's interrupt vector. Bits <31:10> and <1:0> are read as 0 and must be written as 0. When {IE AND INT} transitions to 1, an interrupt is posted at IPL 14. When a timer's interrupt is acknowledged, the content of the interrupt vector register is passed to the CPU, and the INT bit is cleared. Interrupt requests can also be cleared by clearing IE or INT. This register is cleared on power-up and the negation of DCOK.

```
 3
 1                                      10 9                2 1 0
+-----------------------------------------+----------------+---+
|                   MBZ                   |Interrupt vector|MBZ|
+-----------------------------------------+----------------+---+
```

NOTE

Note that both timers interrupt at the same IPL (IPL 14) as the console serial line unit. When multiple interrupts are pending, the console serial line has priority over the timers, and timer 0 has priority over timer 1.

9 KA650-AA PROCESSOR BOOT AND DIAGNOSTIC FACILITY

The KA650-AA Boot and Diagnostic Facility features two registers, two 28-pin ROM Sockets for up to 128K bytes of read-only memory, and 1KB of battery backed up RAM The ROM memory may be accessed via longword, word or byte references.

The KA650-AA CPU Module populates the ROM sockets with 64K bytes of 16-bit ROM (or EPROM). This ROM contains the KA650-AA Console Program. If this ROM is replaced for special applications, the new ROM must contain some version of the Console Program.

9.1 Boot And Diagnostic Register (BDR)

The Boot and Diagnostic Register is a byte-wide register located in the VAX I/O page at physical address 2008 4004 (hex) It can be accessed by KA650-AA software, but not by external Q22-Bus devices. byte operations on the low byte. The BDR allows the Boot and Diagnostic ROM programs to read various KA650-AA configuration bits. Only the low byte of the BDR should be accessed, bits <31:8> are undefined.

```
 3
 1                                         8 7 6 5 4 3 2 1 0
+-------------------------------------------+-+-+-+-+-+-+-+-+
|       Undefined                           | | | | | | | | |
+-------------------------------------------+-+-+-+-+-+-+-+-+
                                             | | | | | | | |
                 HLT ENB --------------------+ | | | | | | |
                                               | | | | | | |
                 BRS CD2 ----------------------+ | | | | | |
                 BRS CD1 ------------------------+ | | | | |
                 BRS CD0 --------------------------+ | | | |
                                                     | | | |
                 CPU CD1 ----------------------------+ | | |
                 CPU CD0 ------------------------------+ | |
                                                         | |
                 BDG CD1 --------------------------------+ |
                 BDG CD0 ----------------------------------+
```

BDR<31:08> Undefined. Should not be read or written.

BDR<7> (HLT ENB) Halt Enable. Read only. This bit reflects the status pin 15 (HLT ENB L) of the 20-pin connector. The assertion of this signal enables the halting of the CPU upon detection of a console BREAK condition. Also, following the execution of a HALT instruction in kernel mode, the KA650-AA ROM Program reads the HLT ENB bit to decide whether to enter the Console program (HLT ENB set) or to restart the operating system (HLT ENB clear). Cleared on power-up and the negation of DCOK.

BDR<4:6> (BRS CD) Baud Rate Select <2:0>. Read only. These three bits originate from pins <19:17> (BRS<2:0>). of the 20-pin connector

They indicate the setting of the the baud rate select switch on the CPU distribution insert. Cleared on power-up and the negation of DCOK.

BDR<3:2> (CPU CD) CPU Code <1:0>. Read only. These two bits originate from connector pins <5:4> (CPU CD<1:0>. Cleared on power-up and the negation of DCOK. They indicate whether the KA650-AA is configured as the arbiter or as one of the three auxiliaries:

CPU CD <1:0>	Configuration
00	KA650-AA Arbiter
01	KA650-AA Auxiliary 1
10	KA650-AA Auxiliary 2
11	KA650-AA Auxiliary 3

BDR<1:0> (BDG CD) Boot and Diagnostic Code <1:0>. Read only. This 2-bit code reflects the status of configuration and display connector pins <14:13> (BDG CD<1:0>) Cleared on power-up and the negation of DCOK. The KA650-AA ROM programs use BDG CD <1:0> to determine the power up mode as follows:

BDG CD <1:0>	Power Up Mode
00	Run
01	Language Inquiry
10	Test
11	Manufacturing

9.2 Diagnostic LED Register

The Diagnostic LED Register (DLEDR) contains four read/write bits that control the external LED display. A "0" in a bit lights the corresponding LED; all four bits are cleared on on power-up and the negation of DCOK to provide a power-up lamp test.

```
 3
 1                                                   4 3 2 1 0
+---------------------------------------------------+-+-+-+-+-+
|                        MBZ                        | | | | | |
+---------------------------------------------------+-+-+-+-+-+
                                                       | | | |
                                                       | | | |
          DSPL 03 ------------------------------------+ | | |
          DSPL 02 --------------------------------------+ | |
          DSPL 01 ----------------------------------------+ |
          DSPL 00 ------------------------------------------+
```

DLEDR<31:4> Unused. Read as 0's, must be written as 0's.

DLEDR<3:0> (DSPL) Display <3:0>. Read/Write. These four bits update an external LED display. Writing a "0" to a bit lights the corresponding LED. Writing a "1" to a bit turns its LED off. The display bits are cleared (all LED's are lit) on power-up and the negation of DCOK.

9.3 ROM Memory

The KA650-AA supports up to 128KB of ROM memory for storing code for board initialization, VAX standard console emulation, board self-tests, and boot code. ROM memory may be accessed via byte, word and longword references. ROM accesses take 1800ns if one ROM (byte-wide) is used, and 1200ns if two ROMS (word-wide) are used. ROM is organized as a 64K x 8-bit array for one 64KB ROM, as a 32K by 16-bit array for two 32KB ROMs, and as a 64K by 16-bit array for two 64KB ROMs. CDAL Bus parity is niether checked or generated on ROM references.

NOTE

The ROM size must be set in the SSC Configuration Register before attempting to references outside the first 8KB block of the Local ROM Space. (2004 0000 - 2004 1FFF)

9.3.1 ROM Socket

The KA650-AA provides two ROM sockets which can be configured to accept one 64K by 8, two 32K by 8, or two 64K by 8 ROMs or EPROMs.

9.3.2 ROM Address Space

The entire 64KB Boot and Diagnostic ROM may be read from either the 64KB Halt Mode ROM Space (Hex Addresses: 2004 0000 - 2004 FFFF), or the 64KB Run Mode ROM space (Hex Addresses: 2005 0000 - 2005 FFFF).

Note that the Run Mode ROM space reads exactly the same ROM code as the Halt Mode ROM space.

Writes to either of these address spaces will result in a machine check.

Any I-Stream Read from the Halt Mode ROM space places the KA650-AA in Halt Mode. The front panel RUN light is off and the CPU is protected from further halts.

Any I-Stream Read which does not access the Halt Mode ROM space, including reads from the Run Mode ROM space, places the KA650-AA in Run Mode. The front panel RUN light is lit and the CPU can be halted if BDR<7> (Halt Enable) is set.

Writes and D-Stream Reads to any address space have no effect on Run Mode/Halt Mode status.

NOTE

The KA650-AA logic that controls Halt Mode/Run Mode cannot detect I-stream read references that "hit" the first-level cache; therefore Halt Mode/Run Mode is unaffected by these cache hits.

9.3.3 KA650-AA Console Program Operation

The KA650-AA CPU Module populates the ROM socket with 64K bytes of 16-bit ROM (or EPROM). This ROM contains the KA650-AA Console Program which can be entered by transferring program control to location 2004 0000.

Section 3.7 lists the various halt conditions which cause the CVAX CPU to transfer program control to location 2004 0000. These conditions include the kernel mode halt instruction, assertion of the external halt input to the chip and certain fatal machine checks. When DCOK has been negated, either at power up time or by reboot, the combined assertion of DCOK and POK initiates program execution at location 2004 0000.

When running, the KA650-AA Console Program provides the services expected of a VAX-11 console system. In particular, the following services are available:

1. Automatic restart or bootstrap following processor halts or initial power up.

2. An interactive command language allowing the user to examine and alter the state of the processor.

3. Diagnostic tests executed on power up that check out the CPU, the memory system and the Q22-Bus Map.

4. Support of video or hardcopy terminals as the console terminal as well as support of QVSS based bit-mapped terminals.

Refer to the KA650-AA Console Program Specification for a complete description of these features.

9.3.3.1 Power Up Modes - The Boot and Diagnostic ROM programs use Boot and Diagnostic Code <1:0> (section 9.1) to determine the power up modes as follows:

Code	Mode
00	Run (factory setting). If the console terminal supports the Multi-national Character Set (MCS), the user will be prompted for language only if the time-of-year clock battery backup has failed. Full startup diagnostics are run.
01	Language Inquiry. If the console terminal supports MCS, the user will be prompted for language on every power up and restart. Full startup diagnostics are run.
02	Test. ROM programs run wrap-around serial line unit (SLU) tests.
03	Manufacturing. To provide for rapid startup during certain manufacturing test procedures, the ROM programs omit the power up memory diagnostics and set up the memory bit map on the assumption that all available memory is functional.

9.4 Battery Backed-up RAM

The KA650 contains 1KB of battery backed-up static RAM, for use as a console "scratchpad". The power for the RAM is provided via pins 10 (BTRY VCC) and 12 (GND) of the 20-pin connector.

This RAM supports byte, word and longword references. Read operations take 700ns to complete while write operations require 600ns.

The RAM is organized as a 256 X 32-bit (one longword) array. The array appears in a 1KB block of the VAX I/O page at addresses 2014 0400- 2014 7FFF (hex).

This array is not protected by parity, and CDAL Bus parity is neither checked or generated on reads or writes to this RAM.

9.5 KA650-AA Initialization

The VAX Architecture defines three kinds of hardware initialization: power-up initialization, processor initialization, and I/O Bus initialization.

9.5.1 Power-Up Initialization

Power-up initialization is the result of the restoration of power and includes a hardware reset, a processor initialization an I/O bus initialization, as well as the initialization of several registers defined in DEC STD 032.

9.5.2 Hardware Reset

A KA650-AA hardware reset occurs on power-up, the negation of DCOK, and if the KA650-AA is configured as an arbiter, on the assertion of BINIT on the Q-22 Bus. A hardware reset initializes some IPR's and most I/O Page registers to a known state. Those IPR's that are affected by a module reset are noted in section 3.1.3. The affect a module reset has on I/O Space registers is documented in the description of the register.

9.5.3 I/O Bus Initialization

An I/O Bus Initialization occurs on power-up, the negation of DCOK, or as the result of a MTPR to IPR 55 (IORESET) or console UNJAM command if the KA650-AA is configured as an arbiter. If the KA650-AA is an arbiter, an I/O Bus Initialization clears the ICR and DSER, and causes the Q-22 Bus Interface to acquire both the CDAL and Q-22 Buses, then assert the Q-22 Bus BINIT signal.

The assertion of BINIT on the Q-22 Bus has no effect on a KA650-AA that is configured as an arbiter, and causes a processor initialization if the KA650-AA is configured as an auxiliary.

9.5.3.1 I/O Bus Reset Register (IPR 55) - The I/O Bus Reset Register (IORESET), Internal Processor Register 55, is implemented in the SSC chip. If the KA650-AA is configured as an arbiter, a MTPR IORESET causes an I/O Bus initialization. If the KA650-AA is configured as an auxiliary, a MTPR to this register is a NOP.

9.5.4 Processor Initialization

A Processor Initialization occurs on power-up, the negation of DCOK, the assertion of BINIT (if the KA650-AA is configured as an auxiliary), as the result of a console INITIALIZE command, and after a halt caused by an error condition.

In addition to inializing those registers defined in DEC STD 032, the KA650-AA firmware must also configure Main Memory (see section 6.4), the Local I/O Page, and the Q-22 Bus Map during a processor initialization.

9.5.4.1 Configuring The Local I/O Page - There are several registers that control the configuration of the KA650-AA local I/O Page. These Registers are unique to CPU designs that use the SSC system support chip. These registers must be configured by the KA650-AA firmware during a Processor Initialization, and include: the SSC Base Address Register, the CACR Address Decode Match Register, the CACR Address Decode Mask Register, the BDR Address Decode Match Register, the BDR Address Decode Mask Register, the SSC Configuration Register, and the CDAL Bus Timout Register.

9.5.5 SSC Base Address Register (SSCBR)

The SSC Base Address Register, address 2014 0000 (hex), controls the base addresses of a 2KB block of the Local I/O Space which includes the battery backed-up RAM, the registers for the programmable timers, the CACR and BDR Address Decode Match and Mask Registers, the Diagnostic LED Register, the CDAL Bus Timeout Register, and a set of diagnostic registers that allow several IPR's to be accessed via I/O page addresses. This read/write register is set to 2014 0000 (hex) on power-up and the negation of DCOK. SSCBR<31:30,10:0> are unused. They read as 0's, and must be written as 0's. SSCBR<29> is read as 1 and must be written as 1. This register should should also be set to 2014 0000 (hex) by firmware during processor initialization. The SSCBR has the following format:

```
 3 3 2 2                                  1 1
 1 0 9 8                                  1 0                    0
+---+-+-----------------------------------+---------------------+
|MBZ|1|    BASE ADDRESS BITS<28:11>        |         MBZ         |
+---+-+-----------------------------------+---------------------+
```

9.5.6 CACR Address Decode Match Register (CDMTR)

The CACR Address Decode Match Register, address 2014 0130 (hex), controls the base address of the CACR. This read/write register is cleared on power-up and the negation of DCOK. CDMTR<31:30,1:0> are unused. They read as 0's, and must be written as 0's. This register should should be set to 2008 4000 (hex) by firmware during processor initialization. The CDMTR has the following format:

```
 3 3 2
 1 0 9                                                       2 1 0
+---+-------------------------------------------------------+---+
|MBZ|              BASE ADDRESS MATCH BITS<29:2>            |MBZ|
+---+-------------------------------------------------------+---+
```

9.5.7 CACR Address Decode Mask Register (CDMKR)

The CACR Address Decode Mask Register, address 2014 0134 (hex), controls the range of addresses that the CACR responds to (i.e. the number of copies of the CACR that appear in the physical address space). This read/write register is cleared on power-up and the negation of DCOK. CDMKR<31:30,1:0> are unused. They read as 0's, and must be written as 0's. This register should should be set to 3FFF FFFC (hex) (1 copy of the CACR) by firmware during processor initialization. The CDMKR has the following format:

```
 3 3 2
 1 0 9                                                       2 1 0
+---+-------------------------------------------------------+---+
|MBZ|               BASE ADDRESS MASK BITS<29:2>            |MBZ|
+---+-------------------------------------------------------+---+
```

9.5.8 BDR Address Decode Match Register (BDMTR)

The BDR Address Decode Match Register, address 2014 0140 (hex), controls the base address of the BDR. This read/write register is cleared on power-up and the negation of DCOK. BDMTR<31:30,1:0> are unused. They read as 0's, and must be written as 0's. This register should be set to 2008 4004 (hex) by firmware during processor initialization. The BDMTR has the following format:

```
 3 3 2
 1 0 9                                                       2 1 0
+---+-------------------------------------------------------+---+
|MBZ|              BASE ADDRESS MATCH BITS<29:2>            |MBZ|
+---+-------------------------------------------------------+---+
```

9.5.9 BDR Address Decode Mask Register (BDMKR)

The BDR Address Decode Mask Register, address 2014 0134 (hex), controls the range of addresses that the BDR responds to (i.e. the number of copies of the CACR that appear in the physical address space). This read/write register is cleared on power-up and the negation of DCOK. BDMKR<31:30,1:0> are unused. They read as 0's, and must be written as 0's. This register should should be set to 3FFF FFFC (hex) (1 copy of the BDR) by firmware during processor initialization. The BDMKR has the following format:

```
 3 3 2
 1 0 9                                                    2 1 0
+---+----------------------------------------------------+---+
|MBZ|            BASE ADDRESS MASK BITS<29:2>            |MBZ|
+---+----------------------------------------------------+---+
```

9.5.10 SSC Configuration Register (SSCCR)

The SSC Configuration Register, address 2014 0010 (hex), controls the set-up parameters for the console serial line, programmable timers, ROM, TOY Clock, BDR and CACR.

```
 3 3   2 2 2 2 2 2 2   2 1 1   1 1 1   1 1 1

 1 0   8 7 6 5 4 3 2   0 9 8   6 5 4   2 1 0   8 7 6 5 4 3 2 1 0
+-+-----+-+-+---+-+-----+-+-----+-+-----+-------+-+-----+-+------+
|B|     |I|M|IPL|R|ROM  |M|HALT |C|CT   |M|AUX  |M|CACR |M|BDR   |
|L| MBZ |V|B|LVL|S|SIZE |B|PROT |T|BAUD |B|BAUD |B|EN   |B|EN    |
|O|     |D|Z|SEL|P|SEL  |Z|SPACE|P|SEL  |Z|SEL  |Z|     |Z|      |
+-+-----+-+-+---+-+-----+-+-----+-+-----+-------+-+-----+-+------+
```

SSCCR<31> (BLO) Battery Low. Read/Write. If the battery voltage goes below threshold while the module is powered down, this bit is set on power up, after the assertion of DCOK, but before the assertion of POK. Once set, this bit can only be cleared by software writing it as "0". If this bit is set, then the TOY Clock will be cleared by power-up and and by the negation of DCOK.

SSCCR<30:28> Unused. Read as 0's, must be written as 0's.

SSCCR<27> (IVD) Interrupt Vector Disable. Read/Write. When set, the console serial line and programmable timers will not respond to interrupt acknowledge cycles. Cleared on power-up and,by the negation of DCOK, and by a processor initialization.

SSCCR<26> Unused. Read as 0's, must be written as 0's.

SSCCR<25:24> (IPL LVL SEL) IPL Level Select. Read/Write. These bits are used to specify the IPL level of interrupt acknowledge cycle that the console serial line and programmable timers respond to. These bits must be cleared for the console serial line and programmable timers to respond to interrupt acknowledge cycles that they generated (IPL 14). These bits are cleared on power-up, by the negation of DCOK, and by a processor initialization.

SSCCR<23> (RSP) ROM Speed. Read/Write. This bit is used to select the ROM access time. This bit must be set for the KA650-AA ROMs to run at maximum speed. This bit is cleared on power-up and by the negation of DCOK. It must be set by processor initialization.

SSCCR<22:20> (ROM SIZE SEL) ROM Address Space Size Select. Read/Write. These bits control the size of the range of addresses to which the ROM responds. These bits must be set to 100 (binary) if the KA650-AA contains 64KB of ROM, yielding an address range of 128KB (2004 0000 - 2005 FFF hex), and 101 (binary) if the KA650-AA has 128KB of ROM, yielding an address range of 256KB (2004 0000 - 2007 FFFF hex). These bits are cleared on power-up and by the negation of DCOK, yielding an address range of 8KB (2004 1FFF).

SSCCR<18:16> (HALT PROT SPACE) ROM Halt Protect Address Space Size Select. Read/Write. These bits control the size of the Halt Mode address range.. These bits must be set to 011 (binary) if the KA650-AA contains 64KB of ROM, yielding an Halt Mode address range of 64KB (2004 0000 - 2004 FFF hex), and 100 (binary) if the KA650-AA has 128KB of ROM, yielding an address range of 128KB (2004 0000 - 2005 FFFF hex). These bits are cleared on power-up and by the negation of DCOK, yielding a Halt Mode address range of 8KB (2004 1FFF). These bits must be set to the proper value by a processor initialization.

SSCCR<15> (CTP) Control P Enable. Read/Write. When this bit is set, a CTRL/P typed at the console causes the CPU to be halted, if halts are enabled (BDR<7> set). When this bit is cleared, a BREAK typed at the console causes the CPU to be halted, if halts are enabled (BDR<7> set). This bit is cleared on power-up and by the negation of DCOK.

SSCCR<14:12> (CT BAUD SELECT) Console Terminal Baud Rate Select. Read/Write. These bits are used to select the baud rate of the Console Terminal Serial Line. They are cleared on power-up and by the negation of DCOK. They should be loaded from BDR<6:4> by a processor initialization. The bit encodings correspond to selected baud rates as shown below:

SSCCR<14:12>	Baud Rate
000	300
001	600
010	1200
011	2400
100	4800
101	9600
110	19.2K
111	38.4K

NOTE

The SSC baud clock runs about 1.7% fast.

SSCCR<11> Unused. Read as 0, must be written as 0.

SSCCR<10:8> Unused. Read as Written. Cleared on power-up and by the negation of DCOK.

SSCCR<7> Unused. Read as 0, must be written as 0.

SSCCR<6:4> (CACR EN) Read/Write. These bits are used to enable the CACR. They are cleared on power-up and by the negation of DCOK. These bits must be set to 111 (binary) by a processor initialization to enable the CACR.

SSCCR<3> Unused. Read as 0, must be written as 0.

SSCCR<2:0> (BDR EN) Read/Write. These bits are used to enable the BDR. They are cleared on power-up and by the negation of DCOK. These bits must be set to 111 (binary) by a processor initialization to enable the BDR.

9.6 CDAL Bus Timeout Control Register (CBTCR)

The CDAL Bus Timeout Register, address 2014 0020, controls the amount of time allowed to elapse before a CDAL Bus cycle is aborted. This prevents unanswered CPU read or write accesses, or interrupt acknowledge cycles from hanging the system any longer than the timeout interval.

```
 3 3          2 2
 1 0          4 3                                          0
+-+-----------+-------------------------------------------+
|B|           |                                           |
|T|    MBZ    |          BUS TIMEOUT INTERVAL             |
|O|           |                                           |
+-+-----------+-------------------------------------------+
```

CBTCR<31> (BTO) CDAL Bus Timeout. Read/Write to Clear. This bit is set when the BUS TIMOUT INTERVAL set in bits <23:0> has expired during a CPU read, write, or interrupt acknowledge cycle. This bit is cleared on power-up and the negation of DCOK.

CBTCR<30:22> Unused. Read as 0's, must be written as 0's.

CBTCR<23:0> (BUS TIMEOUT INTERVAL) Read/Write. These bits are used to program the desired timeout period. The available range of 1 to FFFFFF (hex) corresponds to a selectable timeout range of 1us to 16.77 seconds in 1us increments. Writing a zero to this field disables the bus timeout function. The BTO bit is used to signify that a bus timeout has occurred. This field is cleared on power-up and the negation of DCOK. This field should be set to <TBS> by a processor initialization.

10 KA650-AA Q22-BUS INTERFACE

The KA650 includes a Q-22 Bus interface implemented via a single VLSI chip called the CQBIC. It contains a CDAL Bus to Q-22 Bus interface, that supports the following:

- o A programmable mapping function (scatter-gather map) for translating 22-bit, Q-22 Bus addresses into 29-bit CDAL Bus addresses that allows any page in the Q-22 Bus memory space to be mapped to any page in main memory.

- o A direct mapping function for translating 29-bit CDAL addresses in the local Q-22 Bus address space and local Q-22 Bus I/O page into 22-bit, Q-22 Bus addresses.

- o Masked and unmasked longword reads and writes from the CPU to the Q-22 Bus Memory and I/O Space and the Q-22 Bus interface registers. Longword reads and writes of the local Q-22 Bus memory space are buffered and translated into two-word, block mode, transfers on the Q-22 Bus. Longword reads and writes of the local Q-22 Bus I/O space are buffered and translated into two, single-word transfers on the Q-22 Bus.

- o Up to sixteen-word, block mode, writes from the Q-22 Bus to main memory. These words are buffered then transferred to main memory using two asynchronous DMA octaword transfers. For block mode writes of less than sixteen words, the words are buffered and transferred to main memory using the most efficient combination of octaword, quadword, and longword asynchronous DMA transfers. Reads of main memory from the Q-22 Bus cause the Q-22 Bus interface to perform an asynchronous DMA quadword read of main memory and buffer all four words, so that on block mode reads, the next three words of the block mode read can be delivered without any additional CDAL Bus cycles. Q-22 Bus Burst Mode DMA transfers result in single-word reads and writes of Main Memory.

- o Transfers from the CPU to the local Q-22 Bus memory space, that result in the Q-22 Bus Map translating the address back into main memory (Local-Miss, Global-Hit transactions).

The Q-22 Bus interface contains several registers for Q-22 Bus control and configuration, interprocessor comunication, and error reporting.

The interface also contains Q-22 Bus interrupt arbitration logic that recognizes Q-22 Bus interrupt requests BR7-BR4 and translates them into CPU interrupts at levels 17-14.

The Q-22 Bus interface detects Q-22 Bus "No Sack" timeouts, Q-22 Bus Interrupt Acknowledge timeouts, Q-22 Bus non-existent memory timeouts, main memory errors on DMA accesses from the Q-22 Bus and Q-22 Bus parity errors.

10.1 Q-22 Bus To Main Memory Address Translation

On DMA references to main memory, the 22-bit, Q-22 Bus address must be translated into a 29-bit main memory address. This translation process is performed by the Q-22 Bus interface by using the Q-22 Bus Map. This map contains 8192 mapping registers, (one for each page in the Q-22 Bus Memory Space), each of which can map a page (512 bytes) of the Q-22 Bus Memory address space into any of the 128K pages in main memory. Since Local I/O Space addresses cannot be mapped to Q-22 Bus pages, the Local I/O Page is unaccessible to devices on the Q-22 Bus.

Q-22 Bus addresses are translated to main memory addresses as follows:

```
                       2
                       1                            9 8      0
                      +------------------------------+--------+
                      |       Q-22 bus address       |        |
                      +------------------------------+--------+
                                      |              |        |
                  extract to select   |              |        |
                  map register        |              |        |
                                      |              |        |
      +-------------------------------+              |        |
      |                                              |        |
      |                                              |        |
      |                                              |        |
      |                                              |        |
      |                                             0|        |
      |  +-+--------+--------------------------------+        |
      +->|V|        |   Mapping Register             |        |
         +-+--------+--------------------------------+        |
          3         |1                             0|        |
          1         |9                              |        |
                    |                               |        |
                    |                               |        |
                    |2                              |        |
                    |8                             9|8      0|
                    +--------------------------------+--------+
                    |      Physical Address of Main Memory    |
                    +--------------------------------+--------+
```

At power up time, the Q-22 Bus Map Registers, including the valid bits, are undefined. External access to main memory is disabled so long as the Interprocessor Communication Register LM EAE bit is cleared. The Q-22 Bus Interface monitors each Q-22 Bus cycle and responds if the following three conditions are met:

1. The Interprocessor Communication Register LM EAE bit is set.

2. The Valid bit of the selected mapping register is set.

3. During read operations, the mapping register must map into existent main memory, or a Q-22 Bus timeout occurs. (During write operations, the Q-22 Bus Interface returns Q-22 Bus BRPLY before checking for existent local memory; the response depends only on conditions 1 and 2 above).

NOTE

In the case of local-miss, global-hit, the state of the LM EAE bit is ignored.

If the map cache does not contain the needed Q-22 Bus Map Register, then the Q-22 Bus interface will perform an asychronous DMA read of the Q-22 Bus Map register before proceding with the Q-22 Bus DMA transfer.

10.1.1 Q-22 Bus Map Registers (QMR's)

The Q-22 Bus Map contains 8192 registers that control the mapping of Q-22 Bus addresses into main memory. Each register maps a page of the Q-22 Bus Memory Space into a page of main memory.

The Local I/O Space address of each register was chosen so that register address bits <14:2> are identical to Q-22 Bus address bits <21:9> of the Q-22 Bus page which the register maps.

Register Address	Q22-Bus Addresses Mapped (Hex)	Q22-Bus Addresses Mapped (Octal)
2008 8000	00 0000 - 00 01FF	00 000 000 - 00 000 777
2008 8004	00 0200 - 00 03FF	00 001 000 - 00 001 777
2008 8008	00 0400 - 00 05FF	00 002 000 - 00 002 777
2008 800C	00 0600 - 00 07FF	00 003 000 - 00 003 777
2008 8010	00 0800 - 00 09FF	00 004 000 - 00 004 777
2008 8014	00 0A00 - 00 0BFF	00 005 000 - 00 005 777
2008 8018	00 0C00 - 00 0DFF	00 006 000 - 00 006 777
2008 801C	00 0E00 - 00 0FFF	00 007 000 - 00 007 777
.	.	.
.	.	.
.	.	.
2008 FFF0	3F F800 - 3F F9FF	17 774 000 - 17 774 777
2008 FFF4	3F FA00 - 3F FBFF	17 775 000 - 17 775 777
2008 FFF8	3F FC00 - 3F FDFF	17 776 000 - 17 776 777
2008 FFFC	3F FE00 - 3F FFFF	17 776 000 - 17 777 777

The Q-22 Bus Map Registers (QMR's) have the following format:

```
 3 3                        2 1
 1 0                        0 9                              0
+-+--------------------------+--------------------------------+
|V|          MBZ             |           A28 - A9             |
+-+--------------------------+--------------------------------+
```

QMR<31> (V) Valid. Read/Write. When a Q-22 Bus Map Register is selected by bits <21:9> of the Q-22 Bus address, the Valid bit determines whether mapping is enabled for that Q-22 Bus page. If the Valid bit is set, the mapping is enabled, and Q-22 Bus addresses within the page controlled by the register are mapped into the main memory page determined by bits <28:9>. If the Valid bit is clear, the mapping register is disabled, and the Q-22 Bus interface does not respond to addresses within that page. This bit is UNDEFINED on power-up.

QMR<30:20> Unused. These bits always read as zero and must be written as 0.

QMR<19:0> (A28-A9) Address Bits <28:9>. Read/Write. When a Q-22 Bus Map Register is selected by a Q-22 Bus address, and if that register's Valid bit is set, then these 20 bits are used as main memory address bits <28:9>. Q-22 Bus address bits <8:0> are used as main memory address bits <8:0>. These bits are UNDEFINED on power-up.

10.1.2 Accessing The Q-22 Bus Map Registers

Although the CPU accesses the Q-22 Bus Map Registers via aligned, masked longword references to the Local I/O Page (adresses 2008 8000 - 20008 FFFC hex), the map actually resides in a 32KB block of main memory. The starting address of this block is controlled by the contents of the Q-22 Bus Map Base Register. The Q-22 Bus interface also contains a 16-entry, fully associative, Q-22 Bus Map Cache to reduce the number of main memory accesses require dfor address translation.

NOTE

> The system software must protect the pages of memory that contain the Q-22 Bus Map from direct accesses that will corrupt the map or cause the entries in the Q-22 Bus Map Cache to become stale. Either of these conditions will result in the incorrect operation of the mapping function.

When the CPU accesses the Q-22 Bus Map through the Local I/O Page addresses, the Q-22 Bus interface reads or writes the map in main memory. The Q-22 Bus interface does not have to gain Q-22 Bus mastership when accessing the Q-22 Bus Map. Since these addresses are in the local I/O space, they are not accessible from the Q-22 Bus.

On a Q-22 Bus Map read by the CPU, the Q-22 Bus interface decodes the Local I/O Space address (2008 8000 - 2008 FFFC). If the register is in the Q-22 Bus Map Cache, the Q-22 Bus interface will internally resolve any conflicts between CPU and Q-22 Bus transactions (if both are attempting to access the Q-22 Bus Map Cache entries at the same time), then return the data. If the map register is not in the map cache, the Q-22 Bus Interface will force the CPU to retry, acquire the CDAL Bus, perform an asynchronous DMA read of the map register. On completion of the read, the CPU is provided with the data when its read operation is retried. A map read by the CPU does not cause the register that was read to be stored in the map cache.

On a Q-22 Bus Map write by the CPU, the Q-22 Bus interface latches the data, then on the completion of the CPU write, acquires the CDAL Bus and performs an asynchronous DMA write to the map register. If the map register is in the Q-22 Bus Map Cache, then the CamValid bit for that entry will be cleared to prevent the entry from becoming stale. A Q-22 Bus Map write by the CPU does not update any cached copies of the Q-22 Bus Map Register.

10.1.3 The Q-22 Bus Map Cache

To speed up the process of translating Q-22 Bus address to Main Memory addresses, the Q-22 Bus Interface utilizes a fully associative, sixteen entry, Q-22 Bus Map Cache.

If a DMA transfer ends on a page boundry, the Q-22 Bus interface will prefetch the mapping register required to translate the next page and load it into the cache, before starting a new DMA transfer. This allows Q-22 Bus block mode DMA transfers that cross page boundries to proceed without delay. The replacement algorithm for updating the Q-22 Bus Map Cache is FIFO.

The cached copy of the Q-22 Bus Map Register is used for the address translation process. If the required map entry for a Q-22 Bus address (as determined by bits <21:9> of the Q-22 Bus address), is not in the map cache, then the Q-22 Bus interface uses the contents of the Map Base Register to access main memory and retrieve the required entry. After obtaining the entry from main memory, the valid bit is checked. If it is set, the entry is stored in the cache and the Q-22 Bus cycle continues.

The format of a Q-22 Bus Map Cache entry appears below:

```
 3  3                         2 1                                 0
 3  2                         0 9                                 0
+--+--------------------------+-----------------------------------+
|CV|   QBUS ADR<21:9>         |          A28 - A9                 |
+--+--------------------------------------------------------------+
```

CQMR<33> (CamValid) When a mapping register is selected by a Q22-Bus address, the CamValid bit determines whether the cached copy of the mapping register for that address is valid. If the CamValid bit is set, the mapping register is enabled, and addresses within that page can be mapped. If the CamValid bit is clear, the Q-22 Bus interface must read the map in local memory to determine if the maping register is enabled. This bit is cleared on power-up, by setting the TBIA (Translation Buffer Invalidate All) bit in the Interprocessor Communication Register, on writes to IPR 55 (IORESET), or by a write to the Q-22 Bus Map Base Register

CQMR<32:20> (QBUS ADR) These bits contain the Q-22 Bus address bits <21:9> of the page that this entry maps. This is the content addressable field of the 16 entry cache for determining if the map register for a particular Q-22 Bus address is in the map cache. These bits are UNDEFINED on power-up.

CQMR<19:0> (Address bits A28-A9) When a mapping register is selected by a Q22-Bus address, and if that register's CamValid bit is set, then these 20 bits are used as main memory address bits 28 thru 9. Q-22 Bus address bits 8 thru 0 are used as local memory address bits 8 thru 0. These bits are UNDEFINED on power-up.

10.2 CDAL Bus To Q-22 Bus Address Translation

CDAL Bus addresses within the Local Q-22 Bus I/O Space, addresses 2000 0000 - 2000 1FFF)hex), are translated into Q-22 Bus I/O Space addresses by using bits <12:0> of the CDAL address as bits <12:0> of the Q-22 Bus address and asserting BBS7. Q-22 Bus address bits <21:13> are driven as 0's.

CDAL Bus addresses within the Local Q-22 Bus Memory Space, addresses 3000 0000 - 303F FFFF (hex), are translated into Q-22 Bus Memory Space addresses by using bits <21:0> of the CDAL address as bits <21:0> of the Q-22 Bus address.

10.3 Interprocessor Communications Facility

The KA650-AA can be configured as a Q-22 Bus Arbiter, or as one of up to three Q-22 Bus auxiliary processors.

The KA650-AA Interprocessor Communication Facility allows other processors on the system to request program interrupts from the KA650-AA without using the Q-22 Bus interrupt request lines. It also controls external access to local memory (via the Q-22 Bus Map) and allows other processors to "halt" an auxiliary KA650-AA.

10.3.1 Interprocessor Communication Register (ICR)

The Interprocessor Communication Register is a 16-bit register which resides in the Q-22 Bus I/O page address space and can be accessed by any device which can become Q-22 Bus master (including the KA650-AA itself). The ICR is byte accessible, meaning that a write byte instruction can write to either the low or high byte without affecting the other byte.

The I/O Page address of the ICR varies with the four configurations of arbiter and auxiliary KA650-AA:

Hex 32-Bit Address	Octal 22-Bit Address	Register
2000 1F40	17 777 500	ICR (KA650-AA Arbiter CPU)
2000 1F42	17 777 502	ICR (KA650-AA Auxiliary #1)
2000 1F44	17 777 504	ICR (KA650-AA Auxiliary #2)
2000 1F46	17 777 506	ICR (KA650-AA Auxiliary #3)

```
                   1  1  1  1  1  1
                   5  4  3  2  1  0  9  8  7  6  5  4  3  2  1  0
                 +--+--+----------------+--+--+--+--+-----------+--+
                 |  |  |       0        |  | 0|  |  |     0     |  |
                 +--+--+----------------+--+--+--+--+-----------+--+
                   |  |                   |       |  |              |
     DMA QME    ---+  |                   |       |  |              |
     TBIA       ------+                   |       |  |              |
     AUX HLT   ---------------------------+       |  |              |
                                                  |  |              |
     DBI IE    -----------------------------------+  |              |
     LM EAE    --------------------------------------+              |
                                                                    |
     DBI RQ    -----------------------------------------------------+
```

ICR<15> (DMA QME) DMA Q22-Bus Address Space Memory Error. Read only. This bit indicates that an error occured when a Q-22 Bus device was attempting to read main memory. It is set if DMA System Error Register bit DSER<4> (DMA QME) or DSER<0> (SLAVE DMA NXM) is set. The DMA QME bit indicates that an uncorrectable error occurred when an external device (or CPU) was accessing the KA650-AA local memory. The SLAVE DMA NXM bit indicates that a DMA transfer to non-existent memory was attempted. This bit is cleared on power-up, by the negation of DCOK, by writes to IPR 55 (IORESET), and whenever DSER<4> is cleared.

ICR<14> (TBIA) Translation Buffer Invalidate All. Writing a "1" to this bit clears the CamVALID bits in the cached copy of the MAP. This bit always reads as zero. Writing a "0" has no effect.

ICR<13:09> (Unused) Read as zeros.

ICR<8> (AUX HLT) Auxiliary Halt. Read/Write on an auxiliary KA650-AA. Read only on an arbiter KA650-AA. When this bit is set on an auxiliary KA650-AA, typically by the arbiter processor, the on-board CPU transfers program control to the Halt Mode ROM Code. When this bit is set on an arbiter KA650-AA, it has no effect on the operation of the on-board CPU. This bit is cleared on power-up, by the negation of DCOK, by writes to IPR 55 (IORESET).

ICR<7> Unused. Read as zero.

ICR<6> (DBI IE) Doorbell Interrupt Enable. Read/Write when the KA650-AA is Q-22 Bus master. Read only when another device is Q-22 Bus master. When set, this bit enables interprocessor doorbell interrupt requests via ICR<0>. Cleared on power-up, by the negation of DCOK, and writes to IPR 55 (IORESET).

ICR<5> (LM EAE) Local Memory External Access Enable. Read/Write when the KA650-AA is Q-22 Bus master. Read only when another device is Q-22 Bus master. When set, this bit enables external access to local memory (via the Q22-Bus map). Cleared on power-up, by the negation of DCOK, and writes to IPR 55 (IORESET) if the KA650-AA is configured as an auxiliary.

ICR<4:1> Unused. Read as zeros.

ICR<0> (DBI RQ) Doorbell Interrupt Request. Read/Write. If ICR<6> (DBI IE) is set, setting this bit generates a doorbell interrupt request. If ICR<6> is clear, setting this bit has no effect. Clearing this bit has no effect. DBI RQ is cleared when the CPU grants the doorbell interrupt request. DBI RQ is held clear whenever DBI IE is clear. This bit is cleared on power-up and the negation of DCOK.

10.3.2 Interprocessor Doorbell Interrupts

If the Interprocessor Communication Register DBI IE bit is set, any Q22-Bus Master can request an interprocessor doorbell interrupt by writing a "1" into ICR bit <0>.

Interrupt Vector:	204 (hex)
Interrupt Priority:	IPL 14 (hex); same as BR4 on Q-Bus

(the interprocessor doorbell is the third highest priority IPL 14 device, directly after the console serial line unit and the programmable timers).

NOTE

Following an interprocessor doorbell interrupt, the KA650-AA CPU sets the IPL to 14. The IPL is set to 17 for external Q-22 Bus BR4 interrupts.

10.4 Q-22 Bus Interrupt Handling

When the KA650-AA is configured as the arbiter CPU, it responds to interrupt requests BR7-4 with the standard Q22-Bus interrupt acknowledge protocol (DIN followed by IAK). The console serial line unit, the programmable timers, and the interprocessor doorbell request interrupts at IPL 14 and have priority over all Q22-Bus BR4 interrupt requests. After responding to any interrupt request BR7-4, the CPU sets the processor priority to IPL 17. All BR7-4 interrupt requests are disabled unless software lowers the processor priority.

When the KA650-AA is configured as an auxiliary, it does not respond to interrupt requests from the Q22-Bus. However, it does respond to the BR4 level interrupt requests from its console serial line unit, interprocessor doorbell and programmable timers.

Interrupt requests from the KA650-AA interval timer are handled directly by the CPU. Interval timer interrupt requests have a higher priority than BR6 interrupt requests. After responding to an interval timer interrupt request, the CPU sets the processor priority to IPL 16. Thus, BR7 interrupt requests remain enabled.

10.5 Configuring The Q-22 Bus Map

The KA650-AA implements the Q-22 Bus Map in an 8K longword (32KB) block of main memory. This Map must be configured by the KA650-AA firmware during a processor initialization by writing the base address of the uppermost 32KB block of good main memory into the Q-22 Bus Map Base Register. This base of this map must be located on a 32KB boundry.

NOTE

This 32KB block of main memory must be protected by the system software. The only access to the map should be through Local I/O page addresses 2008 8000 - 2008 FFFC (hex).

10.5.1 Q-22 Bus Map Base Address Register (QBMBR)

The Q-22 Bus Map Base Address Register, address 2008 0010 (hex) controls the main memory location in of the 32KB block of Q-22 Bus Map Registers. This read/write register is accessable by the CPU on a longword boundary only. Bits <31:29,14:0> are unused and should be written as 0 and will return zero when read.

A write to the Map Base Register will flush the Q-22 Bus Map Cache by clearing the CamValid bits in all the entries.

This register is undefined on power up and is not affected by BINIT being asserted on the Q-22 Bus even if the KA650-AA is configured as an auxilliary.

```
 3 2 2                      1 1
 1 9 8                      5 4                              0
+---+-------------------------+------------------------------+
| 0 |      MAP BASE           |               0              |
+---+-------------------------+------------------------------+
```

10.6 System Configuration Register (SCR)

The System Configuration Register, address 2008 0000 (hex), contains the processor number which determines the address of the ICR register, a BHALT enable bit, a power OK flag and an AUX flag.

The System Configuration Register (SCR) is longword, word, and byte accessible. Programmable option fields are cleared on power-up but and by the negation of DCOK.

The SCR register format follows:

```
        3                               1 1 1 1 1 1
        1                               5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
       +-------------------------------+-+-+-----+-+-----------+-----+-+
       |              0                | | |  0  | |     0     |     |0|
       +-------------------------------+-+-+-----+-+-----------+-----+-+
                                        | |       |                 |
                                        | |       |                 |
 POK ----------------------------------+ |       |                 |
 BHALT ENB ------------------------------+       |                 |
 AUX --------------------------------------------+                 |
 DOORBELL OFFSET SELECT -------------------------------------------+
```

SCR<0> Unused. Read as 0.

SCR<3:1> (DOORBELL OFFSET SELECT) These Read/Write bits determine the ICR address if the KA650 is configured as an auxiliary processor. If the KA650 is configured as an arbiter processor, these bits have no effect. If the KA650 is configured as an auxiliary processor, programming all zeros will disable access of the ICR from the Q22-Bus, this includes local processor accesses. A doorbell address is selected by programming a non-zero offset into SCR bits <3:1>. A CPU write to the SCR arbitrates for the Q22-Bus mastership before allowing the write data to be updated. These bits are cleared on power-up and by the negation of DCOK.

SCR<4:9> Unused. Read as 0.

SCR<10> (AUX) This read only bit defines Auxiliary and Arbiter mode of operation of the KA650. When read as a zero, Arbiter mode is selected, and when read as a one, Auxiliary mode is selected. This bit is cleared on power-up and by the negation of DCOK.

SCR<14> (BHALT EN) Enable bit. This read/write bit controls the effect the Q22-Bus BHALT signal has on the CPU. When set, Q-22 Bus BHALT signal will halt the CPU. When cleared, The Q-22 Bus BHALT signal will have no effect. This bit is cleared on power-up and by the negation of DCOK.

SCR<15> (POK) This read-only bit is set if the Q-22 Bus BPOK signal is asserted and clear if it is negated. This bit is cleared on power-up and by the negation of DCOK.

10.7 DMA System Error Register (DSER)

The DMA System Error Register addresse 2008 0004 (hex) is one of three registers associated with Q-22 Bus Interface error reporting. These registers are located in the local VAX I/O address space and can only be accessed by the local processor. The DMA System Error Register logs main memory errors on DMA transfers, Q-22 Bus parity errors, Q-22 Bus non-existent memory errors, and Q22-Bus no-Grant errors. The DMA Master Error Address Register contains the address of the page in Q22-Bus space which caused a parity error during an access by the local procesor. The Slave Error Address Register contains the address of the page in local memory which caused a memory error during an access by an external device or the processor during a local miss global hit transaction. An access by the local processor which the Q-22 Bus Interface maps into main memory will provide error status to the processor when the processor does a RETRY for a READ local miss-global hit, or by an interrupt in the case of a local-miss global-hit write.

The DSER is a longword, word, or byte accessable read/write register available to the local processor. The bits in this register are cleared to zero on power-up, by the negation of DCOK, and by writes to IPR 55 (IORESET). All bits are set to one to record the occurance of an event. They are cleared by writing a "1", writing zeros has no effect.

```
        3                               1 1 1 1 1 1
        1                               5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
       +-------------------------------+---------------+-+-+-+-+-+-+-+-+
       |           0                   |       0       | |0| | | | |0| |
       +-------------------------------+---------------+-+-+-+-+-+-+-+-+
                                                        |   | | | |   |
Q-22 BUS NXM   -----------------------------------------+   | | | |   |
                                                            | | | |   |
Q-22 BUS PE  -----------------------------------------------+ | | |   |
MAIN MEMORY ERROR  -------------------------------------------+ | |   |
LOST ERROR BIT  ------------------------------------------------+ |   |
NO GRANT ---------------------------------------------------------+   |
DMA NXM --------------------------------------------------------------+
```

DSER<0> (DMA NXM) is a read/write bit and is set on an DMA transfer to a non-existent main memory location. This includes local miss global hit cycles and map accesses to non-existent memory. It is asserted as a 1 and cleared by writing a 1. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

DSER<2> (NO GRANT TIMEOUT) is a read/write bit and is set to one if the Q22-Bus does not return a bus grant within 10ms of the bus request from a CPU demand read cycle, or write cycle. It is asserted as a 1 and cleared by writing a 1. During interrupt acknowledge or request read cycles this bit is not set. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

DSER<3> (LOST ERROR) is a read/write bit and indicates that an error address has been lost because of DSER<7,5,4,0> having been previously set and a subsequent error of either type occurs which would have normally captured an address and set either DSER<7,5,4,0> flag. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

DSER<4> (MAIN MEMORY ERROR) is a read/write bit and is set if an external Q22-Bus device or local miss global hit receives a memory error while reading local memory. It is asserted as a 1 and cleared by writing a 1. The doorbell error bit 15 reports the memory error to the external Q22-Bus device. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

DSER<5> (Q-22 BUS PARITY ERROR) is a read/write bit and is set when the CPU performs a Q22-Bus Demand read cycle which returns a parity error. It is asserted as a one and cleared by writing a 1. During interrupt acknowledge cycles, or request read cycles this bit is not set. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

DSER<7> (MASTER DMA NXM) is a read/write bit and is set when the CPU performs a demand Q22-Bus read cycle or write cycle that does not reply after 10us. It is asserted as a one and cleared by writing a 1. During interrupt acknowledge cycles, or request read cycles, this bit is not set. It is cleared on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

10.8 Master Error Address Register (MEAR)

The Master Error Address Register, address: 2008 0008 (hex), is a read only, longword accessable register whose contents are valid only if DSER <5> is set (Q-22 Bus Parity Error) or if DSER<7> is set (Q-22 Bus timeout).

Reading this register when DSER<5> and DSER<7> are clear will return undefined results . Additional Q-22 Bus parity errors that could have set DSER<5> or Q-22 Bus timeout errors that could have caused DSER<7> to set, will cause DSER<3> to set.

The MEAR contains the address of the page in Q-22 Bus space which caused a parity error during an access by the on-board CPU which set DSER<5> or a master timeout which set DSER<7>.

Q-22 Bus address bits <21:9> are loaded into MEAR bits <12:0>. MEAR bits <31:13> always read as zeros.

```
 3                              1 1
 1                              3 2
+------------------------------------------------------------------
|        Unused, Returns 0        |      Q-22 Bus
|                                 |      Address Bits <21:9>
+------------------------------------------------------------------
```

NOTE

This is a read only register, if a write is attempted a machine check will be generated.

10.9 Slave Error Address Register (SEAR)

The Slave Error Address Register address 2008 000C (hex) is a read only, longword accessable, register which contains valid information only when DSER<4> (main memory error) is set or when DSER<0> (DMA NXM) is set. Reading this register when DSER<4> and DSER<0> are clear will return UNDEFINED data.

The SEAR contains the map translated address of the page in local memory which caused a memory error or non existent memory error during an access by an external device or the Q-22 Bus interface for the CPU during a local miss - global hit transaction or Q-22 Bus Map access.

The contents of this register are latched when DSER<4> or DSER<0> is set. Additional main memory errors or non-existent memory errors have no effect on the SEAR until software clears DSER<4> and DSER<0>.

Mapped QBUS address bits <28:9> are loaded into SEAR bits <19:0>. SEAR bits <31:20> always read as zeros.

```
 3                             2 1                            
 1                             0 9                            0
+---------------------------------------------------------------
|        unused, returns 0      |          MAPPED QBUS         
|                               |      Address Bits <28:9>     
+---------------------------------------------------------------
```

NOTE

This is a read only register, if a write is attempted a machine check will be generated.

10.10 Error Handling

The classes of errors that the Q-22 Bus interface detects and reports are: non-existent Q-22 Bus Memory Space and Q-22 Bus I/O Space references, non existent local memory , no-grant timeout, no-sack abort, main memory errors on DMA accesses, Q-22 Bus parity errors and local-miss, global-hit non-existent memory and main memory errors.

Q-22 Bus Errors

If there is a non existent memory error (10us timeout) on a DEMAND READ then the cycle is aborted, a machine check occurs and a NXM flag is set in the DSER and the address is captured in the MEAR. If there is a NXM on a prefetch read, or an interrupt acknowledge vector read, then the prefetch is aborted without a machine check and no information is captured. Q-22 Bus timeout errors on write references, are reported by setting DSER<7> and posting an interrupt request at IPL 1D with a vector of 60 (hex)

If an uncorrectable main memory error or CDAL bus timeout occurs during a read-lock reference from the CPU, then the cycle is aborted, a machine check occurs, the lock is released and Q-22 Bus mastership is given up.

If the CPU attempts to do a multiple longword transfer to the Q-22 Bus I/O Space or Q-22 Bus Memory Space, the Q-22 Bus interface will cause the cycle to be aborted and a machine check generated, since these addresses are in VAX I/O Space and only longword transfers with byte masks are legal.

If the local processor tries to obtain Q-22 Bus mastership on a demand read reference, and does not obtain it within 10ms, then the cycle is aborted, a machine check is generated, and the NO GRANT bit will be set in the DSER.

If a Q-22 Bus parity error is detected on a CPU demand read reference to the Q22-Bus the cycle is aborted, a machine check is generated, a flag is set in the DSER, and the error address is captured in the MEAR. If a Q-22 Bus Parity error is detected on a prefetch request read by the CPU, the prefetch is aborted, but no machine check is generated.

All parity or memory error flags and error addresses are latch first held, subsequent parity or memory errors cause a lost error bit to be set unless the error flag has been cleared by the processor.

A MAP write reference to non existent memory or that results in an uncorrectable error will generate an interrupt at IPL 1D with vector 60, set DSER<4 or 0> and capture the address in the SEAR.

A MAP read reference to non existant memory or that results in an uncorrectable memory error will cause the cycle to be aborted, and a machine check generated.

Errors On DMA Transfers to Main Memory

The Q-22 Bus interface does not generate or check CDAL Bus Parity.

Attempted transfers to non existent memory are reported by setting DSER<0> and generating an interrupt at IPL 1D with vector 60, and capturing the address in the SEAR.

A DMA read or write reference which results in an uncorrectable main memory error or CDAL Bus timeout will capture the translated error address in the SEAR, set a flag in the doorbell register as well as set a flag in the DSER. A DMA read reference that results in an uncorrectable main memory error or CDAL Bus timeout will report a parity error to the QBUS with BDALL<17:16>. A DMA write reference that results in an uncorrectable main memory error or CDAL Bus timeout will generate an interrupt at IPL 1D with vector 60, and not inform the Q-22 Bus master at the time of the error, that ther error occured.

When an uncorrectable memory error or CDAL Bus timeout error occurs while reading the Q-22 Bus Map to complete a DMA cycle, an interrupt is generated at IPL 1D with vector of 60.

Miscellaneous Errors

After the KA650-AA has granted the Q-22 Bus, if it does not receive BSACK within 10us, then the grant is withdrawn, no errors are reported, and the arbiter waits 500 nano seconds to clear the BDMGOL<0> daisy chain before beginning arbitration again.

If an error occurs during a Local-miss, global-hit demand read reference, the cycle is aborted, a machine check generated, the address captured in the SEAR, and DSER<4> is set for an uncorrectable memory error or DSER<0> for a non existent memory reference.

If an error occurs during a local-miss, global-hit write reference, an interrupt is generated at IPL 1D with vector 60, the address is captured in the SEAR, and DSER<4> is set for an uncorrectable memory error or DSER<0> for a non existent memory reference.

If the BPOK is negated on the Q-22 Bus, a power fail trap is generated, and the CPU traps through the power-fail SCB vector. The state of the Q-22 Bus BPOK signal can be read from SCR register bit<15>. The Q-22 Bus interface continues to operate after generating the powerfail trap, until DCOK is negated.

11 KA650-AA MULTI-PROCESSOR CONSIDERATIONS

11.1 Auxiliary/Arbiter Differences

When the KA650-AA is configured as an auxiliary, its operation differs from operation as an arbiter KA650-AA in several important areas:

1. The arbiter KA650-AA arbitrates bus mastership per the Q22-Bus DMA protocol; the arbitration logic is disabled on an auxiliary KA650-AA.

2. Both the arbiter and auxiliary KA650-AA request bus mastership via the Q22-Bus DMA Request protocol.

   a. They both assert BDMR on the Q22-Bus.

   b. The arbiter KA650-AA receives DMGI from its arbitration logic; the auxiliary receives DMGI from its Q22-Bus BDMGI pin.

   c. Only the auxiliary KA650-AA actually asserts BSACK on the Q22-Bus.

3. The arbiter KA650-AA asserts the Q22-Bus BINIT signal when DCOK is negated and when its CPU software writes to its Bus Initialize Register; the auxiliary KA650-AA never asserts Q22-Bus BINIT, but receives BINIT and uses it to initialize the CVAX chip and to initialize all internal registers which are initialized on reset (see section <TBD>).

4. The physical address of the Interprocessor Communication Register is different for each of the eight KA650-AA arbiter/auxiliary configurations (per section <TBD>).

5. An auxiliary KA650-AA can be halted by setting bit <08> (AUX HLT) of its Interprocessor Communication Register. On an arbiter KA650-AA, this feature is disabled and AUX HLT is a read-only bit which always reads as zero. (Refer to section <TBD>).

6. The CPU halts are controlled by the external connector HLT ENB input. However, the external halts which are affected differ somewhat for the arbiter and auxiliary KA650-AA modules. (Refer to section <TBD>).

7. Each arbiter or auxiliary KA650-AA module can field interrupt requests from its interval timer, console device, programmable timer, and interprocessor doorbell. Only the arbiter KA650-AA can field interrupts from Q22-Bus interrupt request lines BR7-4.

8. The arbiter asserts BIAKO to the Q22-Bus when it responds to a Q22-Bus interrupt request; the auxiliary asserts BIAKO to the Q22-Bus when it receives the assertion of BIAKI from the Q22-Bus.

9. Although both the arbiter and auxiliary KA650-AA modules contain the same time of year clock and battery back-up circuitry, it is assumed that the auxiliary will be configured without batteries and that its clock will never actually be enabled.

11.2 Multi-Processor Features

The following features have been added to the KA650-AA to allow its use in multi-processor systems:

1. A 3-bit code, received at the external connector, allows the KA650-AA module to be configured as the arbiter or as one of seven auxiliaries (section <TBD>). Section <TBD> presents a list of arbiter/auxiliary differences.

2. The Interprocessor Communication Register (section <TBD>) provides a mechanism for interprocessor interrupts, for enabling and disabling external access to local memory and for flagging local memory parity errors caused by external references. On auxiliary KA650-AA modules, it also provides a mechanism for "halting" the CPU.

11.3 KA650-AA Based Multi-Processor Systems

The KA650-AA multi-processor features were designed for use in a message passing environment similar to the System Communications Architecture (SCA) which is currently layered on the CI Port Architecture.

Each KA650-AA processor in a system fetches instructions and data primarily from its own local memory. The various processors communicate via message queues stored in local memory which has been mapped to the Q22-Bus address space. Typically, the processors use the interprocessor doorbell feature to interrupt each other after placing a message in an empty queue.

In most systems all Q22-Bus devices would be under the direct control of the arbiter processor which fields all interrupts. When a disk controller is under the direct control of the arbiter CPU, then the arbiter must set up the transfer of program and data information between the corresponding disks and the auxiliary processors. The auxiliary processor would be responsible for setting up its own Q22-Bus Map to point to the local memory space which is a target of that transfer.

Following a power up or system restart, the auxiliary CPU runs its self-test diagnostics, clears the valid bits in its Q22-Bus Map mapping registers, enters Halt Mode ROM space and then sets its own Interprocessor Communication Register bits <08> (AUX HLT) and <06:05> (DBI IE and LM EAE). The arbiter CPU waits for the auxiliary's LM EAE bit to set, "boots" the auxiliary CPU by loading the appropriate programs and data into the arbiter's own local memory which are mapped (via the Q22-Bus map) to an assigned Q22-Bus address space. The arbiter then clears the auxiliary's AUX HLT bit. The auxiliary CPU, still in Halt Mode ROM space, waits for its AUX HLT bit to clear and then begins auxiliary execution at a specified location in the Q22-Bus address space (referencing local memory in the arbiter).

11.4 PDP-11 Based Multi-Processor Systems

Up to seven auxiliary KA650-AA modules may be added to a KDF11-B or KDJ11-B based Q22-Bus system. Operation of a PDP-11 based system is similar to that of a KA650-AA based system. However, the following issues must be addressed:

1. When a PDP-11 processor is arbiter, its "local" memory is actually Q22-Bus memory. This appears to present no special problems. Obviously, a portion of the Q22-Bus memory address space must be reserved for mapping the auxiliary KA650-AA modules' local memory.

2. Since the PDP-11 does not contain an interprocessor communication register, an external device must be added which allows the auxiliary KA650-AA modules to interrupt the PDP-11.

   Since the KA650-AA console program does not interrupt the arbiter CPU, they do not require modification if this external device is not compatible with the KA650-AA interprocessor communication register (one could use either a DLV11 or a DRV11).

APPENDIX A

KA650-AA PHYSICAL SPECIFICATIONS (PINOUTS/CONNECTORS)

Specs of fingers - A/B (including Q-bus), C/D, connectors, jumpers.

A.1 DIMENSIONS

The KA650-AA, MS650-AA, and MS650-BA are quad height modules with the following dimensions:

Height	-	10.457 +.015/ -.020 inches
Length	-	8.430 +.010/ -.010 inches
Width	-	.375 inches maximum (non-conductive) .343 inches maximum (conductive)

NOTE

Width, as defined for Digital Equipment modules, is the height of components above the surface of the module.

A.2 KA650-AA CONNECTORS

The KA650-AA has five connector interfaces: two fingers that plug into rows A and B of the backplane (the A/B row fingers), two fingers that plug into rows C and D of the backplane (the C/D row fingers), a 50-pin connector that connects the KA650-AA and MS650-A modules (the KA650 memory connector), a 20-pin connector that connects the KA650-AA with the LED's and switches on the CPU distribution insert (the KA650-AA configuration and display connector), and a 10-pin connector that connects the KA650-AA to the console terminal port on the CPU distribution insert (the KA650-AA console SLU connector).

A.2.1 KA650-AA A/B Row Fingers

ıhe KA650-AA A/B row fingers are compatible with the Q22-Bus specification (DEC standard 160). The SRUN L signal appears on pin AF1.

A.2.2 KA650-AA C/D Row

The pinout of the KA650-AA C/D row fingers is <TBD>.

A.2.3 KA650-AA Memory Connector

The placement and pinout of the KA650-AA memory connector are <TBD>.

A.2.4 KA650-AA Configuration And Display Connector

The placement and pinout of the configuration and display connector are <TBD>.

A.2.5 KA650-AA Console SLU Connector

The placement and pinout of the configuration and display connector are <TBD>.

..3 MS650-AA CONNECTORS

The MS650-AA has three connector interfaces: two fingers that plug into rows A and B of the backplane (the A/B row fingers), two fingers that plug into rows C and D of the backplane (the C/D row fingers), and a 50-pin connector that connects the KA650-AA and MS650-A modules (the MS650 memory connector).

A.3.1 MS650 A/B Row

The MS650-AA and MS650-BA memory modules are quad height modules, each of which can mount in the next successive slot after either a KA650-AA module or another MS650 module.

The MS650 A/B row fingers connect with +5 volts (pins AA2, BA2 and BV1) and ground (pins AC2, AJ1, AM1 AT1, BC2, BJ1, BM1 and BT1) only. The MS650 also connects pin AM2 to pin AN2 (passing BIAK) and pin AR2 to AS2 (passing BDMG).

A.3.2 MS650 C/D Row

The pinout of the MS650 C/D row is <TBD>.

A.3.3 MS650 Memory Connector

.he placement and pinout of the MS650 memory connector are <TBD>.

APPENDIX B

KA650-AA ELECTRICAL SPECIFICATIONS

B.1 DC POWER CONSUMPTION

The KA650-AA CPU module power requirements are:

	+5V (+/- 5%)	+12V (+/- 5%)
KA650-AA	<TBD>	<TBD> Amps maximum

Typical currents are 10% less than the specified maximum.

The MS650 memory module power requirements are determined for both the active read/write condition and the inactive (non-read/non-write) condition. Under normal operating conditions only one memory module will be active during a given bus cycle. Thus the system power requirement is calculated ›y adding together the requirements for the KA650-AA CPU module, one MS650 memory in active mode, and all the remaining MS650s in the inactive mode. Note that during a special <TBD> memory test mode all MS650 memory modules may be active simultaneously.

The MS650 memory module power requirements are:

	+5V (+/- 5%)		
	read/write	refresh only	
MS650-AA	<TBD>	<TBD>	Amps maximum
MS650-BA	<TBD>	<TBD>	Amps maximum

B.2 BUS LOADS

The KA650-AA CPU Bus Loads are:

<TBD> AC Loads
<TBD> DC Loads

The MS650 modules do not place any DC or AC loads on the Q-Bus.

B.3 DCOK SIGNAL

.4 POK SIGNAL

B.5 BATTERY BACKUP SPECIFICATIONS

APPENDIX C

KA650-AA ENVIRONMENTAL AND RELIABILITY SPECIFICATIONS

C.1 STORAGE CONDITIONS

The KA650-AA module has an ambient storage temperature range of -40'C to +65'C (-40'F to 149'F). Storage relative humidity is 10% to 90%, non-condensing, altitudes to 9.1 km (50,000 ft).

C.2 OPERATING CONDITIONS

The KA650-AA module will meet or exceeds the requirements for operation within a system placed in a DEC Standard 102 Class C Environment.

The operating temperature for a KA650-AA module mounted in a box within a cabinet is 5'C (41'F) to 50'C (122'F) ambient at the module. The maximum ›utlet temperature rise allowed is 5'C (9'F) above 40'C (104'F). Derate maximum temperature by 1'C for each 1000 meters (1'F for each 1000 ft) of altitude. Operating relative humidity is 10% to 90%, non-condensing.

The airflow required to meet these specifications is <TBD> lfm.

C.3 MEAN TIME BETWEEN FAILURES (MTBF) ESTIMATE

The estimated hard error rates for the KA650-AA and MS650 modules are as follows:

	Class B Environment	Class C Environment
KA650-AA	25.0 Khrs.	18.5 Khrs.
MS650-AA	37.0 Khrs.	
MS650-BA	74.0 Khrs.	

The estimated soft error rates for the KA650-AA and MS650 modules are as follows:

Memory	Soft Error	Memory	Soft Error

Size	MTBF	Size	MTBF
<TBD>	<TBD>	<TBD>	<TBD>

APPENDIX D

KA650-AA ADDRESS ASSIGNMENTS

D.1 KA650-AA GENERAL LOCAL ADDRESS SPACE MAP

VAX Memory Space

Address Range	Contents
0000 0000 - 03FF FFFF	Local Memory Space (64MB)
0400 0000 - 07FF FFFF	Reserved Memory Space (64MB)
0800 0000 - 0BFF FFFF	Reserved Memory Space (64MB)
0C00 0000 - 0FFF FFFF	Reserved Memory Space (64MB)
1000 0000 - 13FF FFFF	Cache Diagnostic Space (64MB)
1400 0000 - 17FF FFFF	Reserved Cache Diagnostic Space (64MB)
1800 0000 - 1BFF FFFF	Reserved Cache Diagnostic Space (64MB)
1C00 0000 - 1FFF FFFF	Reserved Cache Diagnostic Space (64MB)

VAX I/O Space

Address Range	Contents
2000 0000 - 2000 1FFF	Local Q22-Bus I/O Space (8KB)
2000 2000 - 2003 FFFF	Reserved Local I/O Space (248KB)
2004 0000 - 2004 FFFF	Local ROM Space - Halt Mode (64KB)
2005 0000 - 2005 FFFF	Local ROM Space - Run Mode (64KB)
2006 0000 - 2006 FFFF	Reserved Local ROM Space (64KB)
2007 0000 - 2007 FFFF	Reserved Local ROM Space (64KB)
2008 0000 - 201F FFFF	Local Register I/O Space (1.5MB)
2020 0000 - 23FF FFFF	Reserved Local I/O Space (62MB)
2400 0000 - 27FF FFFF	Reserved Local I/O Space (64MB)
2008 0000 - 2BFF FFFF	Reserved Local I/O Space (64MB)
2C08 0000 - 2FFF FFFF	Reserved Local I/O Space (64MB)

3000 0000	-	303F FFFF	Local Q22-Bus Memory Space (4MB)
3040 0000	-	33FF FFFF	Reserved Local I/O Space (60MB)
3400 0000	-	37FF FFFF	Reserved Local I/O Space (64MB)
3800 0000	-	3BFF FFFF	Cache Tag Diagnostic Space (64MB)*
3C00 0000	-	3FFF FFFF	Reserved Cache Tag Diag Space (64MB)

* Not visible during normal operation.

D.2 KA650-AA DETAILED LOCAL ADDRESS SPACE MAP

```
VAX Memory Space
________

Local Memory Space (up to 64MB)           0000 0000 - 03FF FFFF
    Q-22 Bus Map - top 32KB of Main Memory
Reserved Memory Space                     0400 0000 - 0FFF FFFF

Cache Diagnostic Space                    1000 0000 - 13FF FFFF
Reserved Cache Diagnostic Space           1800 0000 - 1FFF FFFF


VAX I/O Space
-------------

Local Q-22 Bus I/O Space                  2000 0000 - 2000 1FFF
    Reserved Q-22 Bus I/O Space           2000 0000 - 2000 0007
    Q-22 Bus Floating Address Space       2000 0008 - 2000 07FF
    User Reserved Q-22 Bus I/O Space      2000 0800 - 2000 0FFF
    Reserved Q-22 Bus I/O Space           2000 1000 - 2000 1F3F
    Interprocessor Comm Reg (If Arbiter)  2000 1F40
    Interprocessor Comm Reg (If Aux  1)   2000 1F42
    Interprocessor Comm Reg (If Aux  2)   2000 1F44
    Interprocessor Comm Reg (If Aux  3)   2000 1F46
    Reserved Q-22 Bus I/O Space           2000 1F48 - 2000 1FFF

Reserved Local I/O Space                  2000 2000 - 2003 FFFF

Local ROM Space                           2004 0000 - 2007 FFFF
    Local ROM - Halt Mode                 2004 0000 - 2004 FFFF
    uVAX System Type Register (In ROM)    2004 0004
    Local ROM - Run Mode                  2005 0000 - 2005 FFFF
    Reserved Local ROM Space              2006 0000 - 2007 FFFF

Local Register I/O Space                  2008 0000 - 201F FFFF
    DMA System Configuration Register     2008 0000
    DMA System Error Register             2008 0004
    DMA Master Error Address Register     2008 0008
    DMA Slave Error Address Register      2008 000C
    Q-22 Bus Map Base Register            2008 0010
    Reserved Local Register I/O Space     2008 0014 - 2008 00FF
```

KA650-AA DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

Local Register I/O Space (Cont.)

Main Memory Configuration Reg 00	2008 0100
Main Memory Configuration Reg 01	2008 0104
Main Memory Configuration Reg 02	2008 0108
Main Memory Configuration Reg 03	2008 010C
Main Memory Configuration Reg 04	2008 0110
Main Memory Configuration Reg 05	2008 0114
Main Memory Configuration Reg 06	2008 0118
Main Memory Configuration Reg 07	2008 011C
Main Memory Configuration Reg 08	2008 0120
Main Memory Configuration Reg 09	2008 0124
Main Memory Configuration Reg 10	2008 0128
Main Memory Configuration Reg 11	2008 012C
Main Memory Configuration Reg 12	2008 0130
Main Memory Configuration Reg 13	2008 0134
Main Memory Configuration Reg 14	2008 0138
Main Memory Configuration Reg 15	2008 013C
Main Memory Error Status Register	2008 0140
Main Memory Control/Diag Status Reg	2008 0144
Reserved Local Register I/O Space	2008 0018 - 2008 3FFF
Cache Control Register	2008 4000
Boot and Diagnostic Register	2008 4004
Reserved Local Register I/O Space	2008 4008 - 2007 FFFF
Q-22 Bus Map Registers	2008 8000 - 2008 FFFF
Reserved Local Register I/O Space	2009 0000 - 2013 FFFF
SSC Base Address Register	2014 0000
SSC Configuration Register	2014 0010
CDAL Bus Timeout Control Register	2014 0020
Diagnostic LED Register	2014 0030
Reserved Local Register I/O Space	2014 0034 - 2014 0068

```
***********************************************************************
The following addresses allow those KA650-AA Internal Processor
Registers that are implemented in the SSC chip (External, Internal
Processor Registers) to be accessed via the local I/O page. These
addresses are documented for diagnostic purposes only and should
not be used by non-diagnostic programs.

    Time Of Year Register                 2014 006C

    Console Storage Receiver Status       2014 0070*
    Console Storage Receiver Data         2014 0074*
    Console Storage Transmitter Status    2014 0078*
    Console Storage Transmitter Data      2014 007C*
    Console Receiver Control/Status       2014 0080
    Console Receiver Data Buffer          2014 0084
    Console Transmitter Control/Status    2014 0088
    Console Transmitter Data Buffer       2014 008C
    Reserved Local Register I/O Space     2014 0090 - 2014 00DB

    I/O Bus Reset Register                2014 00DC
    Reserved Local Register I/O Space     2014 00E0

    Rom Data Register                     2014 00F0**
    Bus Timeout Counter                   2014 00F4**
    Interval Timer                        2014 00F8**
    Reserved Local Register I/O Space     2014 00FC - 2014 00FF

 * These registers are not fully implemented, accesses yield
   UNPREDICTABLE results.

** These registers are internal SSC registers used for SSC chip
   test purposes only. They should not be accessed by the CPU.

***********************************************************************
```

KA650-AA DETAILED LOCAL ADDRESS SPACE MAP (Cont.)

Local Register I/O Space (Cont.)	
Timer 0 Control Register	2014 0100
Timer 0 Interval Register	2014 0104
Timer 0 Next Interval Register	2014 0108
Timer 0 Interrupt Vector	2014 010C
Timer 1 Control Register	2014 0110
Timer 1 Interval Register	2014 0114
Timer 1 Next Interval Register	2014 0118
Timer 1 Interrupt Vector	2014 011C
Reserved Local Register I/O Space	2014 0120 - 2014 01FF
CACR Address Decode Match Register	2014 0130
CACR Decode Mask Register	2014 0134
Reserved Local Register I/O Space	2014 0138 - 2014 013F
BDR Address Decode Match Register	2014 0140
BDR Decode Mask Register	2014 0144
Reserved Local Register I/O Space	2014 0148 - 2014 03FF
Battery Backed-Up RAM	2014 0400 - 2014 07FF
Reserved Local Register I/O Space	2014 0800 - 201F FFFF
Reserved Local I/O Space	2020 0000 - 2FFF FFFF
Local Q-22 Bus Memory Space	3000 0000 - 303F FFFF
Reserved Local Register I/O Space	3040 0000 - 37FF FFFF
Cache Tag Diagnostic Space	3800 0000 - 1BFF FFFF*
Reserved Cache Tag Diag Space	3C00 0000 - 3FFF FFFF

* Not visible during normal operation

D.3 EXTERNAL, INTERNAL PROCESSOR REGISTERS

Several of the Internal Processor Registers (IPR's) on the KA650-AA are implemented in the SSC chip rather than the CVAX chip. These registers are referred to as External, Internal Processor Registers and are listed below.

IPR #	Register Name	Abbrev.
27	Time of Year Register	TOY
28	Console Storage Receiver Status	CSRS*
29	Console Storage Receiver Data	CSRD*
30	Console Storage Transmitter Status	CSTS*
31	Console Storage Transmitter Data	CSDB*
32	Console Receiver Control/Status	RXCS
33	Console Receiver Data Buffer	RXDB
34	Console Transmitter Control/Status	TXCS
35	Console Transmitter Data Buffer	TXDB
55	I/O System Reset Register	IORESET

* These registers are not fully implemented, accesses yield UNPREDICTABLE results.

D.4 GLOBAL Q-22 BUS ADDRESS SPACE MAP

```
Q-22 Bus Memory Space
---------------------

Q-22 Bus Memory Space  (Octal)          0000 0000 - 17777 7777


Q-22 Bus I/O Space  (BBS7 Asserted)
-----------------------------------

Q-22 Bus I/O Space (Octal)              1776 0000 - 1777 7777
    Reserved Q-22 Bus I/O Space         1776 0000 - 1776 0007
    Q-22 Bus Floating Address Space     1776 0010 - 1776 3777
    User Reserved Q-22 Bus I/O Space    1776 4000 - 1776 7777
    Reserved Q-22 Bus I/O Space         1776 8000 - 1777 7477
    Interprocessor Comm Reg (If Arbiter) 1777 7500
    Interprocessor Comm Reg (If Aux #1) 1777 7502
    Interprocessor Comm Reg (If Aux #2) 1777 7504
    Interprocessor Comm Reg (If Aux #3) 1777 7506
    Reserved Q-22 Bus I/O Space         1777 7508 - 1777 7777
```

APPENDIX E

KA650-AA INSTRUCTION SET

The information in this appendix is excerpted directly from the CVAX CPU engineering specification, and is supplied for reference only.

The standard notation for operand specifiers is:

    <name>.<access type><data type>

where:

1. Name is a suggestive name for the operand in the context of the instruction. It is the capitalized name of a register or block for implied operands.

2. Access type is a letter denoting the operand specifier access type.

a	=	address operand
b	=	branch displacement
m	=	modified operand (both read and written)
r	=	read only operand
v	=	if not "Rn", same as a, otherwise R[n+1]'R[n]
w	=	write only operand

3. Data type is a letter denoting the data type of the operand.

b	=	byte
d	=	dfloating
f	=	ffloating
g	=	gfloating
l	=	longword
q	=	quadword
v	=	field (used only in implied operands)
w	=	word
*	=	multiple longwords (used only in implied operands)

4. Implied operands, that is, locations that are accessed by the instruction, but not specified in an operand, are denoted by curly braces {}.

The abbreviations for condition codes are:

*	=	conditionally set/cleared
-	=	not affected
0	=	cleared
1	=	set

The abbreviations for exceptions are:

rsv	=	reserved operand fault
iov	=	integer overflow trap
idvz	=	integer divide by zero trap
fov	=	floating overflow fault
fuv	=	floating underflow fault
fdvz	=	floating divide by zero fault
dov	=	decimal overflow trap
ddvz	=	decimal divide by zero trap
sub	=	subscript range trap
prv	=	privileged instruction fault

ınteger Arithmetic And Logical Instructions

Opcode	Instruction	N	Z	V	C	Exceptions
58	ADAWI add.rw, sum.mw	*	*	*	*	iov
80	ADDB2 add.rb, sum.mb	*	*	*	*	iov
C0	ADDL2 add.rl, sum.ml	*	*	*	*	iov
A0	ADDW2 add.rw, sum.mw	*	*	*	*	iov
81	ADDB3 add1.rb, add2.rb, sum.wb	*	*	*	*	iov
C1	ADDL3 add1.rl, add2.rl, sum.wl	*	*	*	*	iov
A1	ADDW3 add1.rw, add2.rw, sum.ww	*	*	*	*	iov
D8	ADWC add.rl, sum.ml	*	*	*	*	iov
78	ASHL cnt.rb, src.rl, dst.wl	*	*	*	0	iov
79	ASHQ cnt.rb, src.rq, dst.wq	*	*	*	0	iov
8A	BICB2 mask.rb, dst.mb	*	*	0	-	
CA	BICL2 mask.rl, dst.ml	*	*	0	-	
AA	BICW2 mask.rw, dst.mw	*	*	0	-	
8B	BICB3 mask.rb, src.rb, dst.wb	*	*	0	-	
CB	BICL3 mask.rl, src.rl, dst.wl	*	*	0	-	
AB	BICW3 mask.rw, src.rw, dst.ww	*	*	0	-	
88	BISB2 mask.rb, dst.mb	*	*	0	-	
C8	BISL2 mask.rl, dst.ml	*	*	0	-	
A8	BISW2 mask.rw, dst.mw	*	*	0	-	

```
89          BISB3 mask.rb, src.rb, dst.wb                                     * *
0 -
C9          BISL3 mask.rl, src.rl, dst.wl                                     * *
0 -
A9          BISW3 mask.rw, src.rw, dst.ww                                     * *
0 -

93          BITB mask.rb, src.rb                                              * *
0 -
D3          BITL mask.rl, src.rl                                              * *
0 -
B3          BITW mask.rw, src.rw                                              * *
0 -

94          CLRB dst.wb                                                       0 1
0 -
D4          CLRL{=F} dst.wl                                                   0 1
0 -
7C          CLRQ{=D=G} dst.wq                                                 0 1
0 -
B4          CLRW dst.ww                                                       0 1
0 -

91          CMPB src1.rb, src2.rb                                             * *
0 *
D1          CMPL src1.rl, src2.rl                                             * *
0 *
B1          CMPW src1.rw, src2.rw                                             * *
0 *

98          CVTBL src.rb, dst.wl                                              * *
0 0
99          CVTBW src.rb, dst.wl                                              * *
0 0
F6          CVTLB src.rl, dst.wb                                              * *
* 0            iov
F7          CVTLW src.rl, dst.ww                                              * *
* 0            iov
33          CVTWB src.rw, dst.wb                                              * *
* 0            iov
32          CVTWL src.rw, dst.wl                                              * *
0 0

97          DECB dif.mb                                                       * *
* *            iov
D7          DECL dif.ml                                                       * *
* *            iov
B7          DECW dif.mw                                                       * *
* *            iov

86          DIVB2 divr.rb, quo.mb                                             * *
* 0            iov, idvz
C6          DIVL2 divr.rl, quo.ml                                             * *
* 0            iov, idvz
```

```
A6       DIVW2 divr.rw, quo.mw                                         * *
* 0         iov, idvz

87       DIVB3 divr.rb, divd.rb, quo.wb                                * *
* 0         iov, idvz
C7       DIVL3 divr.rl, divd.rl, quo.wl                                * *
* 0         iov, idvz
A7       DIVW3 divr.rw, divd.rw, quo.ww                                * *
* 0         iov, idvz

7B       EDIV divr.rl, divd.rq, quo.wl, rem.wl                         * *
* 0         iov, idvz

7A       EMUL mulr.rl, muld.rl, add.rl, prod.wq                        * *
0 0

96       INCB sum.mb                                                   * *
* *         iov
D6       INCL sum.ml                                                   * *
* *         iov
B6       INCW sum.mw                                                   * *
* *         iov

92       MCOMB src.rb, dst.wb                                          * *
0 -
D2       MCOML src.rl, dst.wl                                          * *
0 -
B2       MCOMW src.rw, dst.ww                                          * *
0 -

8E       MNEGB src.rb, dst.wb                                          * *
* *         iov
CE       MNEGL src.rl, dst.wl                                          * *
* *         iov
AE       MNEGW src.rw, dst.ww                                          * *
* *         iov

90       MOVB src.rb, dst.wb                                           * *
0 -
D0       MOVL src.rl, dst.wl                                           * *
0 -
7D       MOVQ src.rq, dst.wq                                           * *
0 -
B0       MOVW src.rw, dst.ww                                           * *
0 -

9A       MOVZBW src.rb, dst.wb                                         0 *
0 -
9B       MOVZBL src.rb, dst.wl                                         0 *
0 -
3C       MOVZWL src.rw, dst.ww                                         0 *
0 -

84       MULB2 mulr.rb, prod.mb                                        * *
```

```
* 0         iov
C4        MULL2 mulr.rl, prod.ml                                              * *
* 0         iov
A4        MULW2 mulr.rw, prod.mw                                              * *
* 0         iov

85        MULB3 mulr.rb, muld.rb, prod.wb                                     * *
* 0         iov
C5        MULL3 mulr.rl, muld.rl, prod.wl                                     * *
* 0         iov
A5        MULW3 mulr.rw, muld.rw, prod.ww                                     * *
* 0         iov

DD        PUSHL src.rl, {-(SP).wl}                                            * *
0 -

9C        ROTL cnt.rb, src.rl, dst.wl                                         * *
0 -

D9        SBWC sub.rl, dif.ml                                                 * *
* *         iov

82        SUBB2 sub.rb, dif.mb                                                * *
* *         iov
C2        SUBL2 sub.rl, dif.ml                                                * *
* *         iov
A2        SUBW2 sub.rw, dif.mw                                                * *
* *         iov

83        SUBB3 sub.rb, min.rb, dif.wb                                        * *
* *         iov
C3        SUBL3 sub.rl, min.rl, dif.wl                                        * *
* *         iov
A3        SUBW3 sub.rw, min.rw, dif.ww                                        * *
* *         iov

95        TSTB src.rb                                                         * *
0 0
D5        TSTL src.rl                                                         * *
0 0
B5        TSTW src.rw                                                         * *
0 0

8C        XORB2 mask.rb, dst.mb                                               * *
0 -
CC        XORL2 mask.rl, dst.ml                                               * *
0 -
AC        XORW2 mask.rw, dst.mw                                               * *
0 -

8D        XORB3 mask.rb, src.rb, dst.wb                                       * *
0 -
CD        XORL3 mask.rl, src.rl, dst.wl                                       * *
0 -
```

Opcode	Instruction	Exceptions	N	Z	V	C
AD	XORW3 mask.rw, src.rw, dst.ww		*	*	0	-

Address Instructions

Opcode	Instruction	Exceptions	N	Z	V	C
9E	MOVAB src.ab, dst.wl		*	*	0	-
DE	MOVAL{=F} src.al, dst.wl		*	*	0	-
7E	MOVAQ{=D=G} src.aq, dst.wl		*	*	0	-
3E	MOVAW src.aw, dst.wl		*	*	0	-
9F	PUSHAB src.ab, {-(SP).wl}		*	*	0	-
DF	PUSHAL{=F} src.al, {-(SP).wl}		*	*	0	-
7F	PUSHAQ{=D=G} src.aq, {-(SP).wl}		*	*		-
3F	PUSHAW src.aw, {-(SP).wl}		*	*	0	-

Variable Length Bit Field Instructions

Opcode	Instruction	Exceptions	N	Z	V	C
EC	CMPV pos.rl, size.rb, base.vb, {field.rv}, src.rl	rsv	*	*	0	*
ED	CMPZV pos.rl, size.rb, base.vb, {field.rv}, src.rl	rsv	*	*	0	*
EE	EXTV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	rsv	*	*	0	-
EF	EXTZV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	rsv	*	*	0	-
F0	INSV src.rl, pos.rl, size.rb, base.vb, {field.wv}	rsv	-	-	-	-

```
EB          FFC startpos.rl, size.rb, base.vb, {field.rv}, findpos.wl        0 *
0 0            rsv
EA          FFS startpos.rl, size.rb, base.vb, {field.rv}, findpos.wl        0 *
0 0            rsv


Control Instructions


Opcode      Instruction                                                       N Z
V C            Exceptions
------      ------------
-------            ----------

9D          ACBB limit.rb, add.rb, index.mb, displ.bw                         * *
* -            iov
F1          ACBL limit.rl, add.rl, index.ml, displ.bw                         * *
* -            iov
3D          ACBW limit.rw, add.rw, index.mw, displ.bw                         * *
* -            iov

F3          AOBLEQ limit.rl, index.ml, displ.bb                               * *
* -            iov

F2          AOBLSS limit.rl, index.ml, displ.bb                               * *
  -            iov

1E          BCC{=BGEQU} displ.bb                                              - -
- -
1F          BCS{=BLSSU} displ.bb                                              - -
- -
13          BEQL{=BEQLU} displ.bb                                             - -
- -
18          BGEQ displ.bb                                                     - -
- -
14          BGTR displ.bb                                                     - -
- -
1A          BGTRU displ.bb                                                    - -
- -
15          BLEQ displ.bb                                                     - -
- -
1B          BLEQU displ.bb                                                    - -
- -
19          BLSS displ.bb                                                     - -
- -
12          BNEQ{=BNEQU} displ.bb                                             - -
- -
1C          BVC displ.bb                                                      - -
- -
1D          BVS displ.bb                                                      - -
- -

E1          BBC pos.rl, base.vb, displ.bb, {field.rv}                         - -
```

```
- -         rsv
E0       BBS pos.rl, base.vb, displ.bb, {field.rv}                        - -
- -         rsv

E5       BBCC pos.rl, base.vb, displ.bb, {field.mv}                       - -
- -         rsv
E3       BBCS pos.rl, base.vb, displ.bb, {field.mv}                       - -
- -         rsv
E4       BBSC pos.rl, base.vb, displ.bb, {field.mv}                       - -
- -         rsv
E2       BBSS pos.rl, base.vb, displ.bb, {field.mv}                       - -
- -         rsv

E7       BBCCI pos.rl, base.vb, displ.bb, {field.mv}                      - -
- -         rsv
E6       BBSSI pos.rl, base.vb, displ.bb, {field.mv}                      - -
- -         rsv

E9       BLBC src.rl, displ.bb                                            - -
- -
E8       BLBS src.rl, displ.bb                                            - -
- -

11       BRB displ.bb                                                     - -
- -
31       BRW displ.bw                                                     - -
- -

10       BSBB displ.bb, {-(SP).wl}                                        - -
- -
30       BSBW displ.bw, {-(SP).wl}                                        - -
- -

8F       CASEB selector.rb, base.rb, limit.rb, displ.bw-list              * *
0 *
CF       CASEL selector.rl, base.rl, limit.rl, displ.bw-list              * *
0 *
AF       CASEW selector.rw, base.rw, limit.rw, displ.bw-list              * *
0 *

17       JMP dst.ab                                                       - -
- -

16       JSB dst.ab, {-(SP).wl}                                           - -
- -

05       RSB {(SP)+.rl}                                                   - -
- -

F4       SOBGEQ index.ml, displ.bb                                        * *
* -         iov

F5       SOBGTR index.ml, displ.bb                                        * *
* -         iov
```

Procedure Call Instructions

Opcode	Instruction	N	Z	V	C	Exceptions
FA	CALLG arglist.ab, dst.ab, {-(SP).w*}	0	0	0	0	rsv
FB	CALLS numarg.rl, dst.ab, {-(SP).w*}	0	0	0	0	rsv
04	RET {(SP)+.r*}	*	*	*	*	rsv

Miscellaneous Instructions

Opcode	Instruction	N	Z	V	C	Exceptions
B9	BICPSW mask.rw	*	*	*	*	rsv
B8	BISPSW mask.rw	*	*	*	*	rsv
03	BPT {-(KSP).w*}	0	0	0	0	
00	HALT {-(KSP).w*}	-	-	-	-	prv
0A	INDEX subscript.rl, low.rl, high.rl, size.rl, indexin.rl, indexout.wl	*	*	0	0	sub
DC	MOVPSL dst.wl	-	-	-	-	
01	NOP	-	-	-	-	
BA	POPR mask.rw, {(SP)+.r*}	-	-	-	-	

BB	PUSHR mask.rw, {-(SP).w*}		-	-	-	-
FC	XFC {unspecified operands}		0	0	0	0

Queue Instructions

Opcode	Instruction	Exceptions	N	Z	V	C
5C	INSQHI entry.ab, header.aq	rsv	0	*	0	*
5D	INSQTI entry.ab, header.aq	rsv	0	*	0	*
0E	INSQUE entry.ab, pred.ab		*	*	0	*
5E	REMQHI header.aq, addr.wl	rsv	0	*	*	*
5F	REMQTI header.aq, addr.wl	rsv	0	*	*	*
0F	REMQUE entry.ab, addr.wl		*	*	*	*

Character String Instructions

Opcode	Instruction	Exceptions	N	Z	V	C
28	MOVC3 len.rw, srcaddr.ab, dstaddr.ab, {R0-5.wl}		0	1	0	0
2C	MOVC5 srclen.rw, srcaddr.ab, fill.rb, dstlen.rw, dstaddr.ab, {R0-5.wl}		*	*	0	*

Operating System Support Instructions

Opcode	Instruction	Exceptions	N	Z	V	C
BD	CHME param.rw, {-(ySP).w*}		0	0	0	0
BC	CHMK param.rw, {-(ySP).w*}		0	0	0	0
BE	CHMS param.rw, {-(ySP).w*}		0	0	0	0
BF	CHMU param.rw, {-(ySP).w*}		0	0	0	0
	Where y=MINU(x, PSL<currentmode>)					
06	LDPCTX {PCB.r*, -(KSP).w*}	rsv, prv	-	-	-	-
DB	MFPR procreg.rl, dst.wl	rsv, prv	*	*	0	-
DA	MTPR src.rl, procreg.rl	rsv, prv	*	*	0	-
0C	PROBER mode.rb, len.rw, base.ab		0	*	0	-
0D	PROBEW mode.rb, len.rw, base.ab		0	*	0	-
02	REI {(SP)+.r*}	rsv	*	*	*	*
07	SVPCTX {(SP)+.r*, PCB.w*}	prv	-	-	-	-

Floating Point Instructions

These instructions are implemented by the KA650-AA floating point accelerator.

Opcode	Instruction	Exceptions	N	Z	V	C
6F	ACBD limit.rd, add.rd, index.md,displ.bw	rsv, fov, fuv	*	*	0	-
4F	ACBF limit.rf, add.rf, index.mf,displ.bw	rsv, fov, fuv	*	*	0	-
4FFD	ACBG limit.rg, add.rg, index.mg,displ.bw	rsv, fov, fuv	*	*	0	-
60	ADDD2 add.rd, sum.md		*	*		

```
  0         rsv, fov, fuv
40       ADDF2 add.rf, sum.mf                                              * *
0 0         rsv, fov, fuv
40FD     ADDG2 add.rg, sum.mg                                              * *
0 0         rsv, fov, fuv

61       ADDD3 add1.rd, add2.rd, sum.wd                                    * *
0 0         rsv, fov, fuv
41       ADDF3 add1.rf, add2.rf, sum.wf                                    * *
0 0         rsv, fov, fuv
41FD     ADDG3 add1.rg, add2.rg, sum.wg                                    * *
0 0         rsv, fov, fuv

71       CMPD src1.rd, src2.rd                                             * *
0 0         rsv
51       CMPF src1.rf, src2.rf                                             * *
0 0         rsv
51FD     CMPG src1.rg, src2.rg                                             * *
0 0         rsv

6C       CVTBD src.rb, dst.wd                                              * *
0 0
4C       CVTBF src.rb, dst.wf                                              * *
0 0
4CFD     CVTBG src.rb, dst.wg                                              * *
0 0
 3       CVTDB src.rd, dst.wb                                              * *
* 0         rsv, iov
76       CVTDF src.rd, dst.wf                                              * *
0 0         rsv, fov
6A       CVTDL src.rd, dst.wl                                              * *
* 0         rsv, iov
69       CVTDW src.rd, dst.ww                                              * *
* 0         rsv, iov
48       CVTFB src.rf, dst.wb                                              * *
* 0         rsv, iov
56       CVTFD src.rf, dst.wd                                              * *
0 0         rsv
99FD     CVTFG src.rf, dst.wg                                              * *
0 0         rsv
4A       CVTFL src.rf, dst.wl                                              * *
* 0         rsv, iov
49       CVTFW src.rf, dst.ww                                              * *
* 0         rsv, iov
48FD     CVTGB src.rg, dst.wb                                              * *
* 0         rsv, iov
33FD     CVTGF src.rg, dst.wf                                              * *
0 0         rsv, fov, fuv
4AFD     CVTGL src.rg, dst.wl                                              * *
* 0         rsv, iov
49FD     CVTGW src.rg, dst.ww                                              * *
* 0         rsv, iov
6E       CVTLD src.rl, dst.wd                                              * *
0 0
```

```
4E      CVTLF src.rl, dst.wf                                       * *
0 0
4EFD    CVTLG src.rl, dst.wg                                       * *
0 0
6D      CVTWD src.rw, dst.wd                                       * *
0 0
4D      CVTWF src.rw, dst.wf                                       * *
0 0
4DFD    CVTWG src.rw, dst.wg                                       * *
0 0

6B      CVTRDL src.rd, dst.wl                                      * *
* 0        rsv, iov
4B      CVTRFL src.rf, dst.wl                                      * *
* 0        rsv, iov
4BFD    CVTRGL src.rg, dst.wl                                      * *
* 0        rsv, iov

66      DIVD2 divr.rd, quo.md                                      * *
0 0        rsv, fov, fuv, fdvz
46      DIVF2 divr.rf, quo.mf                                      * *
0 0        rsv, fov, fuv, fdvz
46FD    DIVG2 divr.rg, quo.mg                                      * *
0 0        rsv, fov, fuv, fdvz

67      DIVD3 divr.rd, divd.rd, quo.wd                             * *
0 0        rsv, fov, fuv, fdvz
47      DIVF3 divr.rf, divd.rf, quo.wf                             * *
0 0        rsv, fov, fuv, fdvz
47FD    DIVG3 divr.rg, divd.rg, quo.wg                             * *
0 0        rsv, fov, fuv, fdvz

74      EMODD mulr.rd, mulrx.rb, muld.rd, int.wl, fract.wd         * *
* 0        rsv, fov, fuv, iov
54      EMODF mulr.rf, mulrx.rb, muld.rf, int.wl, fract.wf         * *
* 0        rsv, fov, fuv, iov
54FD    EMODG mulr.rg, mulrx.rw, muld.rg, int.wl, fract.wg         * *
* 0        rsv, fov, fuv, iov

72      MNEGD src.rd, dst.wd                                       * *
0 0        rsv
52      MNEGF src.rf, dst.wf                                       * *
0 0        rsv
52FD    MNEGG src.rg, dst.wg                                       * *
0 0        rsv

70      MOVD src.rd, dst.wd                                        * *
0 -        rsv
50      MOVF src.rf, dst.wf                                        * *
0 -        rsv
50FD    MOVG src.rg, dst.wg                                        * *
0 -        rsv

64      MULD2 mulr.rd, prod.md                                     * *
```

```
0 0        rsv, fov, fuv
44         MULF2 mulr.rf, prod.mf                                           * *
0 0        rsv, fov, fuv
44FD       MULG2 mulr.rg, prod.mg                                           * *
0 0        rsv, fov, fuv

65         MULD3 mulr.rd, muld.rd, prod.wd                                  * *
0 0        rsv, fov, fuv
45         MULF3 mulr.rf, muld.rf, prod.wf                                  * *
0 0        rsv, fov, fuv
45FD       MULG3 mulr.rg, muld.rg, prod.wg                                  * *
0 0        rsv, fov, fuv

75         POLYD arg.rd, degree.rw, table.ab                                * *
0 0        rsv, fov, fuv
55         POLYF arg.rf, degree.rw, table.ab                                * *
0 0        rsv, fov, fuv
55FD       POLYG arg.rf, degree.rw, table.ab                                * *
0 0        rsv, fov, fuv

62         SUBD2 sub.rd, dif.md                                             * *
0 0        rsv, fov, fuv
42         SUBF2 sub.rf, dif.mf                                             * *
0 0        rsv, fov, fuv
42FD       SUBG2 sub.rg, dif.mg                                             * *
0 0        rsv, fov, fuv

63         SUBD3 sub.rd, min.rd, dif.wd                                     * *
0 0        rsv, fov, fuv
43         SUBF3 sub.rf, min.rf, dif.wf                                     * *
0 0        rsv, fov, fuv
43FD       SUBG3 sub.rg, min.rg, dif.wg                                     * *
0 0        rsv, fov, fuv

73         TSTD src.rd                                                      * *
0 0        rsv
53         TSTF src.rf                                                      * *
0 0        rsv
53FD       TSTG src.rg                                                      * *
0 0        rsv
```

Microcode-Assisted Emulated Instructions

The KA550-AA CPU provides microcode assistance for the macrocode emulation of these instructions. The CPU processes the operand specifiers,
creates a standard argument list, and invokes an emulation routine to perform emulation.

```
Opcode   Instruction                                                        N Z
V C         Exceptions
------   ------------
-------        ----------
```

```
20        ADDP4 addlen.rw, addaddr.ab, sumlen.rw, sumaddr.ab                   * *
* 0          rsv, dov

21        ADDP6 add1len.rw, add1addr.ab, add2len.rw, add2addr.ab,              * *
* 0          rsv, dov
                sumlen.rw, sumaddr.ab

F8        ASHP cnt.rb, srclen.rw, srcaddr.ab, round.rb,                        * *
* 0          rsv, dov
               dstlen.rw, dstaddr.ab

29        CMPC3 len.rw, src1addr.ab, src2addr.ab                               * *
0 *

2D        CMPC5 src1len.rw, src1addr.ab, fill.rb,                              * *
0 *
                src2len.rw, src2addr.ab

35        CMPP3 len.rw, src1addr.ab, src2addr.ab                               * *
0 0

37        CMPP4 src1len.rw, src1addr.ab, src2len.rw, src2addr.ab               * *
0 0

0B        CRC tbl.ab, inicrc.rl, strlen.rw, stream.ab                          * *
0 0

F9        CVTLP src.rl, dstlen.rw, dstaddr.ab                                  * *
* 0          rsv, dov
36        CVTPL srclen.rw, srcaddr.ab, dst.wl                                  * *
* 0          rsv, iov

08        CVTPS srclen.rw, srcaddr.ab, dstlen.rw, dstaddr.ab                   * *
* 0          rsv, dov
09        CVTSP srclen.rw, srcaddr.ab, dstlen.rw, dstaddr.ab                   * *
* 0          rsv, dov

24        CVTPT srclen.rw, srcaddr.ab, tbladdr.ab,                             * *
* 0          rsv, dov
                dstlen.rw, dstaddr.ab
26        CVTTP srclen.rw, srcaddr.ab, tbladdr.ab,                             * *
* 0          rsv, dov
                dstlen.rw, dstaddr.ab

27        DIVP divrlen.rw, divraddr.ab, divdlen.rw, divdaddr.ab,               * *
* 0          rsv, dov, ddvz
               quolen.rw, quoaddr.ab

38        EDITPC srclen.rw, srcaddr.ab, pattern.ab, dstaddr.ab                 * *
* *          rsv, dov

3A        LOCC char.rb, len.rw, addr.ab                                        0 *
0 0
```

```
39        MATCHC objlen.rw, objaddr.ab, srclen.rw, srcaddr.ab            0 *
0 0

34        MOVP len.rw, srcaddr.ab, dstaddr.ab                            * *
0 0

2E        MOVTC srclen.rw, srcaddr.ab, fill.rb, tbladdr.ab,              * *
0 *
                dstlen.rw, dstaddr.ab

2F        MOVTUC srclen.rw, srcaddr.ab, esc.rb, tbladdr.ab,              * *
* *
                 dstlen.rw, dstaddr.ab

25        MULP mulrlen.rw, mulraddr.ab, muldlen.rw, muldaddr.ab,         * *
* 0          rsv, dov
               prodlen.rw, prodaddr.ab

2A        SCANC len.rw, addr.ab, tbladdr.ab, mask.rb                     0 *
0 0

3B        SKPC char.rb, len.rw, addr.ab                                  0 *
0 0

2B        SPANC len.rw, addr.ab, tbladdr.ab, mask.rb                     0 *
0 0

22        SUBP4 sublen.rw, subaddr.ab, diflen.rw, difaddr.ab             * *
* 0          rsv, dov

23        SUBP6 sublen.rw, subaddr.ab, minlen.rw, minaddr.ab,            * *
* 0          rsv, dov
                diflen.rw, difaddr.ab
```

APPENDIX F

KA650-AA ERROR MATRICES

```
                                        Mayfair Error Response Matrix - CDAL Bus
Parity Errors
                                        ----------------------------------------
-------------

Reference      Effect On       Effect On      Effect On        Effect On
Effect On       State Captured
Type           CPU Execution   Prefetcher     1st-Level Cache  2nd-Level Cache
Main Memory     On Error        Notes
----------     -------------   ----------     ---------------  ---------------
------------    --------------   ------------------
Demand         Abort Cycle,                   Invalidate       Store
                MSER<5> Set      Machine Check
D-Stream       Machine Check      -           Row              Bad Data**
     -          MSER<6> Set      Routine Must Flush
Read           80 or 81
                                 2nd-Level Cache


Request                                       Invalidate       Store
                MSER<6> Set
D-Stream            -              -          Row, Abort       Bad Data**
     -
Read (Fill)                                   Fill


Request                       Abort           Invalidate       Store
                MSER<6> Set
I-Stream            -         Prefetch        Row              Bad Data**
     -
Read (Prefetch)


Request                                       Invalidate       Store
                MSER<6> Set
I-Stream            -              -          Row, Abort       Bad Data**
     -
Read (fill)                                   Fill


Read-Lock      Abort Cycle,        -          Invalidate             -
     -          MSER<6> Set      Next CPU Write
               Machine Check                  Row
                                 Releases Lock &
               80 or 81
                                 Relinquishes Q-22

                                 Bus Mastership


Masked         Interrupt @                                     Store
Store Data      MCSR16<17> Set   MEMERR Interrupt
Write          IPL 1D (MEMERR)    -                -           Bad Data***
With Bad ECC                     Routine Must Flush
               Vector of 60
                                 2nd-Level Cache &
               (Same Cycle)
                                 Main Memory


UnMasked       Interrupt @                                     Store
Store Data      MCSR16<17> Set   MEMERR Interrupt
Write          IPL 1D (MEMERR)    -                -           Bad Data***
With Bad ECC                     Routine Must Flush
               Vector of 60
```

						(Next Cycle)	2nd-Level Cache & Main Memory
'MA Read	-	-	-	-	-	-	Parity Not Checked
DMA Write	-	-	-	-	-	-	Parity Not Checked

* CDAL parity is NOT checked on External Processor Register Read Cycles, or
ansfers between the CVAX and the CFPA, CSSC,
or CQBIC.
** On cacheable miss references only (i.e. when 2nd-level cache allocates a
ck)
*** On 2nd-level cache hits only

Mayfair Error Response Matrix - First-Level Cache Parity Errors*

Reference Type	Effect On CPU Execution	Effect On Prefetcher	Effect On 1st-Level Cache	Effect On 2nd-Level Cache	Effect On Main Memory	State Captured On Error	Notes
Demand D-Stream Read	Abort Cycle, Machine Check 80 or 81	-	Flush Cache,** Disable Cache	-	-	MSER<4> Set MSER<3> Set - if Data Error In Set 2 MSER<2> Set - if Data Error In Set 1 MSER<1> Set - if Data Error MSER<0> Set - if Tag Error	
Request D-Stream Read (Fill)	-	-	-	-	-	-	Reference Type Never "Seen" By 1st-Level Cache
Request I-Stream Read (Prefetch)	-	Abort Prefetch	Flush Cache**	-	-	MSER<3> Set - if Data Error In Set 2 MSER<2> Set - if Data Error In Set 1 MSER<1> Set - if Data Error MSER<0> Set - if Tag Error	
Request I-Stream Read (Fill)	-	-	-	-	-	-	Reference Type Never "Seen" By 1st-Level Cache
Read-Lock	-	-	-	-	-	-	Parity Not Checked
Masked Write	-	-	Flush Cache**	-	-	MSER<0> Set	Only Tag Parity Is Checked On Writes That Hit 1st-Level

							Cache
UnMasked Write	-	-	Flush Cache**	-	-	MSER<0> Set	Only Tag Parity Is Checked On Writes That Hit 1st-Level Cache
DMA Read	-	-	-	-	-	-	Parity Not Checked
DMA Write	-	-	No Invalidate Performed, Bad Tag Unaltered	-	-	-	Only Tag Parity is Checked on writes That Hit 1st-Level Cache

* 1st-Level Cache Parity Errors can be detected only on references that hit
e cache
** The 1st-Level Cache is flushed only if CADR<0> (Diagnostic Mode) is cleare

```
                                    Mayfair Error Response Matrix - Second-Level Cach
e Data Parity Errors*
                                    -------------------------------------------------
---------------------


Reference        Effect On        Effect On       Effect On        Effect On
Effect On         State Captured
Type             CPU Execution    Prefetcher      1st-Level Cache  2nd-Level Cache
Main Memory       On Error          Notes
---------        -------------    ----------      ---------------  ---------------
------------      --------------    ------------------
Demand           Abort Cycle,                     Invalidate       Bad Data
                  MSER<5> Set       Machine Check
D-Stream         Machine Check        -           Row              Unaltered
      -           MSER<6> Set       Routine Must Flush
Read             80 or 81
                                    2nd-Level Cache


Request                                           Invalidate       Bad Data
                  MSER<6> Set
D-Stream              -                 -         Row, Abort       Unaltered
      -
Read (Fill)                                       Fill


Request                            Abort           Invalidate       Bad Data
                  MSER<6> Set
I-Stream              -            Prefetch        Row              Unaltered
      -
Read (Prefetch)


Request                                           Invalidate       Bad Data
                  MSER<6> Set
I-Stream              -                 -         Row, Abort       Unaltered
      -
Read (fill)                                       Fill


Read-Lock             -                 -              -                  -
      -               -             Parity Not Checked


Masked                -                 -              -                  -
      -               -             Parity Not Checked,
Write
                                    Entry Updated On

                                    All CPU Writes

                                    That Hit Cache


UnMasked              -                 -              -                  -
      -               -             Parity Not Checked,
Write
                                    Entry Updated On
```

All CPU Writes

That Hit Cache

MA Read - - - -
- - Parity Not Checked

DMA Write - - - -
- - Parity Not Checked,

Entry Invalidated

On All DMA Writes

That Hit Cache

* 2nd-Level Cache Parity Errors can be detected only on references that hit t
e cache

```
                          Mayfair Error Response Matrix - Second-Level Cach
e Tag Parity Errors*
                          -------------------------------------------------
---------------------


Reference         Effect On         Effect On        Effect On          Effect On
Effect On          State Captured
Type              CPU Execution     Prefetcher       1st-Level Cache    2nd-Level Cache
Main Memory        On Error          Notes
----------        -------------     ----------       ---------------    ---------------
------------       -------------      ------------------
Demand            Interrupt @                                           Force Cache Miss,
                   CACR<5> Set        CRD Interrupt
D-Stream          IPL 1A (CRD)          -                  -            Disable Cache,
      -                               Routine Must Flush
Read              Vector of 54                                          Bad Data Unaltere
d                                     2nd-Level Cache


Request
D-Stream                                    Same Behavior As Demand D-Stream Read
Read (Fill)


Request
:-Stream                                    Same Behavior As Demand D-Stream Read
Read (Prefetch)


Request
I-Stream                                    Same Behavior As Demand D-Stream Read
Read (fill)


Read-Lock                 -                 -               -                  -
      -                   -          Parity Not Checked


Masked                                      Same Behavior As Demand D-Stream Read

Write



UnMasked                                    Same Behavior As Demand D-Stream Read

Write


DMA Read                  -                 -               -                  -
      -                   -          Parity Not Checked


DMA Write                                   Same Behavior As Demand D-Stream Read
```

* 2nd-Level Cache Tag Parity Errors can be detected on all I & D Stream refer
nces to the VAX Memory Space, except Read-Lock
and DMA Read references.

```
                                          Mayfair Error Response Matrix - Q-22 Bus
imeout Errors
                                          ------------------------------------------
-------------


Reference       Effect On        Effect On      Effect On        Effect On
Effect On        State Captured
Type            CPU Execution    Prefetcher     1st-Level Cache  2nd-Level Cache
Main Memory      On Error         Notes
---------       -------------    ----------     ---------------  ---------------
-----------      --------------    -------------------
Demand          Abort Cycle,                    Invalidate
                 DSER<7> Set       10us Timer - NXM
D-Stream        Machine Check      -            Row                     -
    -            MEAR<12:0> - Q-22 Bus Page Address
Read            80 or 81



Request
                                   Quadword Transfers
I or D-Stream         -              -           -                      -
    -                 -            To Q-22 Bus Space
Read (Fill)
                                   Should Never Occur,

                                   Machine Check 80

                                   or 81 if they do

Request                          Abort          Invalidate
                                   10us Timer - NXM
I-Stream             -           Prefetch       Row                     -
    -                 -
Read (Prefetch)


Read-Lock       Abort Cycle,       -            Invalidate              -
    -            DSER<2> Set       10ms Timer
(Including      Machine Check                   Row
                                   No-Grant Timeout
 LMGH*)         80 or 81
                                   While Aquiring

                                   Q-22 Bus For Lock


Masked          Interrupt @
                 DSER<7>Set        10us Timer - NXM
Write           IPL 1D (MEMERR)    -             -                      -
    -            MEAR<12:0> - Q-22 Bus Page Address
                Vector of 60

                (Same Cycle)


UnMasked        Interrupt @
                 DSER<7>Set        10us Timer - NXM
```

```
Write           IPL 1D (MEMERR)      -               -                 -
     -           MEAR<12:0> - Q-22 Bus Page Address
                Vector of 60

                (Same Cycle)


DMA Read             -               -               -                 -
     -                -         No Such Reference


DMA Write            -               -               -                 -
     -                -         No Such Reference


Interrupt       Abort Cycle          -               -                 -
     -                -         10us Timer
Acknowledge


DMA Grant       Abort Cycle          -               -                 -
     -           DSER<2> Set    10ms Timer

* Local-Miss, Global-Hit Reference
```

Mayfair Error Response Matrix - Q-22 Bus Device Parity Errors

Reference Type	Effect On CPU Execution	Effect On Prefetcher	Effect On 1st-Level Cache	Effect On 2nd-Level Cache	Effect On Main Memory	State Captured On Error	Notes
Demand D-Stream Read	Abort Cycle, Machine Check 80 or 81	-	Invalidate Row	-	-	DSER<5> Set MEAR<12:0> - Q-22 Bus Page Address	
Request I or D-Stream Read (Fill)	-	-	-	-	-	-	Quadword Transfers To Q-22 Bus Space Should Never Occur, Machine Check 80 or 81 if they do
Request I-Stream Read (Prefetch)	-	Abort Prefetch	Invalidate Row	-	-	DSER<5> Set MEAR<12:0> - Q-22 Bus Page Address	
Read-Lock	Abort Cycle, Machine Check 80 or 81	-	Invalidate Row	-	-	DSER<5> Set MEAR<12:0> - Q-22 Bus Page Address	Release Lock, Relinquish Q-22 Bus
Masked or Un-Masked Write	-	-	-	-	-	-	Parity Not Checked
DMA Read	-	-	-	-	-	-	No Such Reference By CPU

DMA Write	-	-	-	-
-	-	No Such Reference By CPU		
Interrupt Acknowledge	-	-	-	-
-	-	Parity Not Checked		

```
                                             Mayfair Error Response Matrix - CDAL Bus T
imeout Errors                                ------------------------------------------
-------------


Reference         Effect On         Effect On        Effect On         Effect On
Effect On          State Captured
Type              CPU Execution     Prefetcher       1st-Level Cache   2nd-Level Cache
Main Memory        On Error           Notes
---------         -------------     ----------       ---------------   ---------------
------------       --------------     -------------------
Demand             Abort Cycle,                      Invalidate
                   BTCR<31> Set       <TBS> Timer - NXM
D-Stream           Machine Check        -            Row                        -
       -
Read               80 or 81

Request                                              Invalidate
                   BTCR<31> Set       <TBS> Timer - NXM
I or D-Stream            -              -            Row, Abort                 -
       -
Read (Fill)                                          Fill

(Including LMGH*)


Request                             Abort            Invalidate
                   BTCR<31> Set       <TBS> Timer - NXM
I-Stream                 -          Prefetch         Row                        -
       -
Read (Prefetch)
(Including LMGH*)

Read-Lock         Abort Cycle,          -            Invalidate                 -
       -           BTCR<31> Set       <TBS> Timer - NXM
                  Machine Check                      Row
                                      CQBIC releases lock
                  80 or 81
                                      Relinquishes Q-22

                                      Bus Mastership

Masked or         Abort Cycle,
                   BTCR<31> Set       <TBS> Timer - NXM
Unmasked          Machine Check         -                                       -
       -
Write             80 or 81


DMA Read          Interrupt @
                   BTCR<31> Set       <TBS> Timer - NXM
                  IPL 1D (MEMERR)       -                -                      -
       -           DSER<0> Set
                  Vector of 60
                   SEAR<12:0> -Main Memory Page Address
                  (Same Cycle)
                   BDAL<17:16> -Asserted on QBus W/Data

DMA Write         Interrupt @
```

```
                BTCR<31> Set     <TBS> Timer - NXM
               IPL 1D (MEMERR)     -                ?                 ?
      -         DSER<0> Set
               Vector of 60
                SEAR<12:0> -Main Memory Page Address
               (Same Cycle)
                ICR<15> Set

Interrupt      Abort Cycle         -                -                 -
      -         BTCR<31> Set     <TBS> Timer
Acknowledge

DMA Grant      Hangs               -                -                 -
      -                -          No Timer

Map Read By    Abort Cycle,                      Invalidate
                BTCR<31> Set     <TBS> Timer - NXM
CPU or Demand  Machine Check       -             Row                  -
      -         DSER<0> Set
LMGH* Read     80 or 81
                SEAR<12:0> -Main Memory Page Address

Map Write or  Interrupt @
                BTCR<31> Set     <TBS> Timer - NXM
Map Read      IPL 1D (MEMERR)      -                ?                 ?
      -         DSER<0> Set
During        Vector of 60
                SEAR<12:0> -Main Memory Page Address
Translation   (Same Cycle)
```

* Local-Miss, Global-Hit Reference

Mayfair Error Response Matrix - Main Memo:

orrectable Errors

```
Reference         Effect On        Effect On       Effect On          Effect On
Effect On          State Captured
Type              CPU Execution    Prefetcher      1st-Level Cache    2nd-Level Cac
Main Memory        On Error          Notes
---------         -------------    ----------      ---------------    --------------
-----------        --------------    ------------------

Demand            Interrupt @
Read & Correct     MEMCSR16<29> Set
D-Stream          IPL 1A (CRD)         -                 -                  -
Data, Bad Data     MEMCSR<28:9>-Main Memory Page Add.
Read              Vector of 54
In Memory          MEMCSR<6:0> -Identify Bit Position

Unaltered

                                    CRD Interrupt

                                    Routine Must Flush

                                    Main Memory Page


Request
D-Stream                                      Same Behavior As Demand D-Stream Re
Read (Fill)


Request
I-Stream                                      Same Behavior As Demand D-Stream Re
Read (Prefetch)


Request
I-Stream                                      Same Behavior As Demand D-Stream Re
Read (fill)


Read-Lock                                     Same Behavior As Demand D-Stream Re


Masked                                        Same Behavior As Demand D-Stream Re

Write



UnMasked               -               -               -                 -
         -             -            ECC Not Checked
Write


DMA Read                                      Same Behavior As Demand D-Stream Re
```

DMA Masked 'rite		Same Behavior As Demand D-Stream Rea		
DMA UnMasked - Write	- -	- ECC Not Checked	-	-

```
                                    Mayfair Error Response Matrix - Main Memory
correctable Errors
                                    ---------------------------------------------
------------------


Reference       Effect On       Effect On      Effect On        Effect On
Effect On        State Captured
Type            CPU Execution   Prefetcher     1st-Level Cache  2nd-Level Cache
Main Memory      On Error         Notes
---------       -------------   ----------     ---------------  ---------------
-----------      --------------   ------------------
Demand           Abort Cycle,                  Invalidate
                 MEMCSR16<31> Set
D-Stream         Machine Check      -          Row                            -
     -           MEMCSR16<28:9>-Main Memory Page Add.
Read             80 or 81
                 MEMCSR16<6:0> -Identify Bit Position

Request                                        Invalidate
                 MEMCSR16<31> Set
I or D-Stream        -              -          Row, Abort                     -
     -           MEMCSR16<28:9>-Main Memory Page Add.
Read (Fill)                                    Fill
                 MEMCSR16<6:0> -Identify Bit Position
(Including LMGH*)


Request                         Abort          Invalidate
                 MEMCSR16<31> Set
I-Stream             -          Prefetch       Row                            -
     -           MEMCSR16<28:9>-Main Memory Page Add.
Read (Prefetch)
                 MEMCSR16<6:0> -Identify Bit Position
(Including LMGH*)

Read-Lock       Abort Cycle,        -          Invalidate                     -
     -           MEMCSR16<31> Set
                Machine Check                  Row
                 MEMCSR16<28:9>-Main Memory Page Add.
                80 or 81
                 MEMCSR16<6:0> -Identify Bit Position


                                  CQBIC releases lock

                                  Relinquishes Q-22

                                  Bus Mastership


Masked          Abort Cycle,
                 MEMCSR16<31> Set
Write           Machine Check       -              -                          -
     -           MEMCSR16<28:9>-Main Memory Page Add.
                80 or 81
                 MEMCSR16<6:0> -Identify Bit Position

UnMasked             -              -              -                          -
     -               -            ECC Not Checked
```

Write				
DMA Read -	Interrupt @ MEMCSR16<31> Set IPL 1D (MEMERR) MEMCSR16<28:9>-Main Memory Page Add. Vector of 60 MEMCSR16<6:0> -Identify Bit Position (Same Cycle) DSER<4> Set SEAR<12:0> -Main Memory Page Address BDAL<17:16> -Asserted on QBus W/Data	-	-	-
DMA Masked Write -	Interrupt @ MEMCSR16<31> Set IPL 1D (MEMERR) MEMCSR16<28:9>-Main Memory Page Add. Vector of 60 MEMCSR16<6:0> -Identify Bit Position (Same Cycle) DSER<4> Set SEAR<12:0> -Main Memory Page Address ICR<15> Set	-	?	?
DMA UnMasked - Write (Including Map Writes)	- -	- ECC Not Checked	-	-

Local-Miss, Global-Hit References

Continued

Mayfair Error Response Matrix - Main Memory Ur

correctable Errors

Reference Type	Effect On CPU Execution	Effect On Prefetcher	Effect On 1st-Level Cache	Effect On 2nd-Level Cache	Effect On Main Memory	State Captured On Error	Notes
Map Read By CPU or Demand LMGH* Read	Abort Cycle, Machine Check 80 or 81	-	Invalidate Row	-	-	MEMCSR16<31> Set MEMCSR16<28:9>-Main Memory Page Add. MEMCSR16<6:0> -Identify Bit Position DSER<4> Set SEAR<12:0> -Main Memory Page Address	
Map Read During Translation	Interrupt @ IPL 1D (MEMERR) Vector of 60 (Same Cycle)	-	?	?	-	MEMCSR16<31> Set MEMCSR16<28:9>-Main Memory Page Add. MEMCSR16<6:0> -Identify Bit Position DSER<4> Set SEAR<12:0> -Main Memory Page Address	

* Local-Miss, Global-Hit Reference