Engineering Specification for the KA43 Processor Module

Revision 1.0

1–May–1989

Digital Equipment Corporation

CONTENTS

FIGURES

TABLES

Preface

Change History

Rev	Date	Author	Summary of Changes
1.0	1–May–1989	Ken Curewitz	First draft for general review

CHAPTER 1

INTRODUCTION

1.1 Scope of Document

This engineering specification defines the functiona characteristics of the KA43 desktop workstation. KA43 has a processor with 8 times the performance of a VAX 11/780, up to 32 megabytes of memory, an integral disk, network, serial and monochrome bitmap graphics interfaces. The system board has a connector for a Scanproc module for high performance color graphics. This specification describes the workstation subsystems and describes the software interface of VLSI devices on the module

1.2 General Description

The KA43 System Board forms the basis for one system:

The RigelMAX single-user engineering workstation, which includes 19-inch VR262-AA (1024x864x60Hz) bit-mapped display, an LK201 keyboard, and a VSXXX-AA mouse or VSXXX-AB tablet. A Scanproc video controller and a VR295-AA (1280x1024x66Hz) 19-inch can be added for color display. A SCSI controller on the system board provides connection to internal and external mass storage. Each RigelMAX is housed in a desktop enclosure which contains a KA43 system board and an H7821 power supply. The KA43 system board is the central logic component of a RigelMAX system. It contains the following standard components:

- DC520 Rigel processor (25 ns)
- DC521 Rigel clock chip
- DC523 Rigel floating point unit
- 128 Kbyte second level write-back cache memory
- 8-32 Megabyte RAM memory with parity checking
- 256 Kbyte ROM memory (socketed)
- 32-byte network address ROM (socketed)
- time-of-year clock
- four asynchronous serial ports for:

  keyboard
  mouse or tablet
  printer
  communications
- 256 Kbyte video RAM

- Dual Asynchronous SCSI ports
- controller for 1024 x 864 monochrome display
- controller for ThinWire or Transceiver Ethernet network

Also within the system unit enclosure there are connectors and mounting a Scanproc video option, a mono video SIMM and from 2 to 8 MS43-AA memory SIMMs.

The video subsystem comprises one of the following options: Scanproc color video controller, or VSF01 mono frame buffer (chapter15).

1.3 Applicable Documents

The following documents describe the principal LSI chips which are used in RigelMAX:

- DC520 Rigel processor chip spec
- DC521 RIGEL Clock Chip spec
- DC523 Rigel floating point chip spec
- A-PS-2122769-0-0 DC367B DZART chip spec
- A-PS-2124941-0-0 DC503 Programmable Sprite Cursor chip spec
- A-PS-2128651-0-0 DC7098 CVMTC video/memory
- A-PS-2121672-0-0 LANCE Ethernet Controller chip spec
- A-PS-2118795-0-0 MC146818 Watch chip spec
- A-PS-23365A1-0-0 Ethernet Network Address ROM

CHAPTER 2

KA43 ROM MEMORY

The system board ROM contains processor restart, diagnostic and console code and the primary bootstrap program. (The contents of this ROM is detailed in the ??? specification.) Another small ROM is uniquely programmed for each system with its network address. The VS43G-AA also contains ROM memory.

2.1 System Board ROM

The system board contains two 40-pin sockets for 64K by 16 EPROM chips which collectively hold 256 Kbytes of data. ROM data appears at physical addresses 2004.0000 through 2007.FFFF. The data path to this ROM is 32 bits wide.

Certain physical addresses in the ROM have fixed uses. These are:

Table 1: ROM Address Map

Location	Use
2004.0000	Processor restart address. The processor begins execution at this address in non-mapped mode when a processor restart occurs. (See section 17)
2004.0004	System type register SYS_TYPE. The contents of this longword supplement the internal processor SID register to identify the processor and system type. (See section 3.0.7.2.)
2004.0020	Interrupt vector numbers. Eight consecutive longwords starting at this address are automatically referenced by the hardware to supply the interrupt vector numbers for the eight interrupt sources connected to the interrupt controller. (See section 9.6.)

2.2 Network Address ROM

A 32-byte ROM on the system board contains a unique network address for each system. Data from this ROM is read in the low-order bytes of 32 consecutive longwords at physical addresses 2009.0000 through 2009.007C. The network address occupies the first six bytes (addresses 2009.0000 through 2009.0014). The byte at 2009.0000 is the first byte to be transmitted or received in an address field of an Ethernet packet; its low-order bit (bit 0) is transmitted or received first in the serial bit stream. Digital purchase specification A-PS-23365A1-0-0 describes this ROM in detail and discusses the checksum bytes which follow the six address bytes.

This ROM is installed in a socket so it can be moved in the event that a system's system board is replaced.

2.3 Option Board ROM

The video option has ROM which identifies the option type and which contains firmware initialization and diagnostic code. Standard address ranges are defined for these ROM memories, each spanning 256 Kbytes. The firmware specification specification describes the signature data required in the first bytes of the range to identify the option and its ROM configuration. The system firmware and any operating system software which explores to determine what options are installed in a particular system should examine the signature area of each of these four ROM address ranges to see whether a valid option ROM is present and, if so, what is the type of option.

The address ranges are listed below, together with the option boards whose ROM uses the range.

Table 2: Option ROM Addresses

Address	Use
2014.0000-2017.FFFF	Color video or serial adapter
2018.0000-201B.FFFF	(future options)
201C.0000-201F.FFFF	(future options)

Each board is required to have at least one ROM chip, which must be connected to the low-order byte (data lines 7:0) of the data bus and whose first byte is at the starting address of the address range. Its data is read in the low-order byte of each longword address. If there is only one ROM on the option board, then bits 31:8 of each longword are UNPREDICTABLE. If two chips are used, they must be connected to data lines 15:0 and bits 31:16 of each longword are UNPREDICTABLE. Four chips allow full use of the data bus and direct execution of code in the ROMs. Three-chip configurations are forbidden. If the size of the ROM is less than 256 Kbytes (i.e. each chip stores less than 64 Kbytes), the ROM image may repeat in the address range. The configuration of the ROM chips is indicated by the standard signature data at the beginning of the low-order ROM chip.

CHAPTER 3

DC520 RIGEL CENTRAL PROCESSOR

This section provides summary information about the RIGEL VAX CPU chip and the MicroVAX architecture. It is not intended as a complete reference, but rather to give an overview of the user-visible features. For a complete description, consult the "RIGEL CPU Chip Engineering Specification" and the "VAX Architecture Reference Manual".

The central processor of the KA43 supports the MicroVAX Chip subset (plus six additional string instructions) of the VAX instruction set and datatypes and full VAX memory management. It is implemented via a single VLSI chip called the RIGEL CPU (REX520).

3.0.1 Processor State

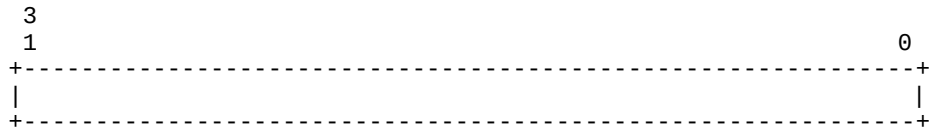
The processor state consists of that portion of the state of a process which is stored in processor registers rather than in memory. The processor state is composed of sixteen General Purpose Registers (GPR's), the Processor Status Longword (PSL), and the Internal Processor Registers (IPR's).

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 can be executed only while running in kernel mode.

3.0.1.1 General Purpose Registers

The KA43 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>.

Figure 1: General Purpose Register



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

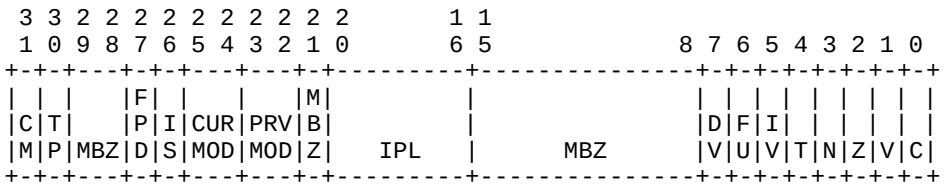
- R15 is the Program Counter (PC). The PC contains the address of the next instruction byte of the program.
- R14 is the Stack Pointer (SP). The SP contains the address of the top of the processor defined stack.
- R13 is the Frame Pointer (FP). The VAX-11 procedure call convention builds a data structure on the stack called a stack frame. The FP contains the address of the base of this data structure.
- 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.

3.0.1.2 Processor Status Longword

The KA43 Processor Status Longword (PSL) is implemented per 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).

Figure 2: Processor Status Longword



PSL<31> (CM) Compatibility Mode. This bit always reads as ZERO, loading a ONE 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> (MBZ) 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> (MBZ) Must Be written as Zero.

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

PSL<15:8> (MBZ) Must Be written as Zero.

PSL<7> (DV) Decimal Overflow Trap Enable. This read/write bit has no effect on KA43 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.

3.0.1.3 Internal Processor Registers

The KA43 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 KA43 as specified in the VAX Architecture Standard (DEC STD 032).

2. RIGEL-specific implementation which is unique or different from the VAX Architecture Standard (DEC STD 032).
3. Not implemented by KA43, read as ZERO, NOP on writes.
4. Not implemented by KA43, access causes RESERVED OPERAND FAULT.
5. Not fully implemented by KA43, access causes UNPREDICTABLE results.

A table listing each of the KA43 IPR's with its mnemonic, its access type (read or write), its scope (CPU-wide or per-process), which chip it is implemented in, whether it is initialized by hardware reset and its 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 IPR's and the section where they are described in full is given in Section 3.1.3.2.

Table 3: KA43 Internal Processor Registers

Register Name	Mnemonic	Number Decimal	Number Hex	Type	Scope	Impl	Init?	Category
Kernel Stack Pointer	KSP	0	0	RW	PROC	REX520		1
Executive Stack Pointer	ESP	1	1	RW	PROC	REX520		1
Supervisor Stack Pointer	SSP	2	2	RW	PROC	REX520		1
User Stack Pointer	USP	3	3	RW	PROC	REX520		1
Interrupt Stack Pointer	ISP	4	4	RW	CPU	REX520		1
Reserved		5-7	5-7					3
P0 Base Register	P0BR	8	8	RW	PROC	REX520		1
P0 Length Register	P0LR	9	9	RW	PROC	REX520		1
P1 Base Register	P1BR	10	A	RW	PROC	REX520		1
P1 Length Register	P1LR	11	B	RW	PROC	REX520		1

Table Headings Meaning

Decimal—Decimal number of the Processor Register
Hex —Hex Number of the Processor Register
Type —Read or Write Access

R —Read-Only Register
W —Write-Only Register
RW—Read-Write Register

Scope —Processor Register's Scope

CPU —CPU-Wide Register
PROC—Per-Process Register

Impl —Chip in which the Processor Register is Implemented

REX520 —Implemented in the REX520 chip
SSC —Implemented in the System Support Chip
C-chip —Implemented in the C-chip

Init? —Initialized on Module RESET (Power-up, Negation of DCOK)
Category—Processor Register Category as previously defined

Table 3 (Cont.): KA43 Internal Processor Registers

Register Name	Mnemonic	Number Decimal	Number Hex	Type	Scope	Impl	Init?	Category
System Base Register	SBR	12	C	RW	CPU	REX520		1
System Length Register	SLR	13	D	RW	CPU	REX520		1
Reserved		14-15	E-F					3
Process Control Block Base	PCBB	16	10	RW	PROC	REX520		1
System Control Block Base	SCBB	17	11	RW	CPU	REX520		1
Interrupt Priority Level	IPL	18	12	RW	CPU	REX520	Yes	1
AST Level	ASTLVL	19	13	RW	PROC	REX520	Yes	1
Software Interrupt Request Register	SIRR	20	14	W	CPU	REX520		1
Software Interrupt Summary Register	SISR	21	15	RW	CPU	REX520	Yes	1
Reserved		22-23	16-17					3
Interval Counter Control Status	ICCS	24	18	RW	CPU	REX520		2
Reserved		25-26	19-1A					3
Time of Year Register	TODR	27	1B	RW	CPU	SSC		1
Console Storage Receiver Status	CSRS	28	1C	RW	CPU	SSC	Yes	5
Console Storage Receiver Data	CSRD	29	1D	R	CPU	SSC	Yes	5
Console Storage Transmitter Status	CSTS	30	1E	RW	CPU	SSC	Yes	5
Console Storage Transmitter Data	CSTD	31	1F	W	CPU	SSC	Yes	5
Console Receiver Control/Status	RXCS	32	20	RW	CPU	SSC	Yes	2
Console Receiver Data Buffer	RXDB	33	21	R	CPU	SSC	Yes	2
Console Transr Control/Status	TXCS	34	22	RW	CPU	SSC	Yes	2
Console Transr Data Buffer	TXDB	35	23	W	CPU	SSC	Yes	2
Reserved		36-37	24-25					3

Table Headings Meaning

Decimal—Decimal number of the Processor Register
Hex —Hex Number of the Processor Register
Type —Read or Write Access

R —Read-Only Register
W —Write-Only Register
RW—Read-Write Register

Scope —Processor Register's Scope

CPU —CPU-Wide Register
PROC—Per-Process Register

Impl —Chip in which the Processor Register is Implemented

REX520 —Implemented in the REX520 chip
SSC —Implemented in the System Support Chip
C-chip —Implemented in the C-chip

Init? —Initialized on Module RESET (Power-up, Negation of DCOK)
Category—Processor Register Category as previously defined

Table 3 (Cont.): KA43 Internal Processor Registers

Register Name	Mnemonic	Number Decimal	Number Hex	Type	Scope	Impl	Init?	Category
Machine Check Error Register	MCESR	38	26	W	CPU	REX520		2
Reserved		39	27					3
Accelerator Control and Status Register	ACCS	40	28	RW	CPU	REX520	Yes	2
Reserved		41	29					3
Console Saved PC	SAVPC	42	2A	R	CPU	REX520		2
Console Saved PSL	SAVPSL	43	2B	R	CPU	REX520		2
Reserved		44-46	2C-2E					3
Translation Buffer Tag	TBTAG	47	2F	W	CPU	REX520		2
Reserved		48-54	30-36					3
I/O System Reset Register	IORESET	55	37	W	CPU	SSC		2
Memory Management Enable	MAPEN	56	38	RW	CPU	REX520	Yes	1
Translation Buffer Invalidate All	TBIA	57	39	W	CPU	REX520		1
Translation Buffer Invalidate Single	TBIS	58	3A	W	CPU	REX520		1
Translation Buffer Data	TBDATA	59	3B	W	CPU	REX520		2
Reserved		60-61	3C-3D					3
System Identification	SID	62	3E	R	CPU	REX520		1
Translation Buffer Check	TBCHK	63	3F	W	CPU	REX520		1
Reserved		64-122	40-7A					3
Vector Interface Error Status Register	VINTSR	123	7B	RW	CPU	C-Chip		2
Primary Cache Tag Store	PCTAG	124	7C	RW	CPU	REX520		2
Primary Cache Index Register	PCIDX	125	7D	RW	CPU	REX520		2
Primary Cache Error Address Register	PCERR	126	7E	RW	CPU	REX520		2

Table Headings Meaning

Decimal—Decimal number of the Processor Register
Hex —Hex Number of the Processor Register
Type —Read or Write Access

R —Read-Only Register
W —Write-Only Register
RW—Read-Write Register

Scope —Processor Register's Scope

CPU —CPU-Wide Register
PROC—Per-Process Register

Impl —Chip in which the Processor Register is Implemented

REX520 —Implemented in the REX520 chip
SSC —Implemented in the System Support Chip
C-chip —Implemented in the C-chip

Init? —Initialized on Module RESET (Power-up, Negation of DCOK)
Category—Processor Register Category as previously defined

Table 3 (Cont.): KA43 Internal Processor Registers

Register Name	Mnemonic	Number Decimal	Number Hex	Type	Scope	Impl	Init?	Category
Primary Cache Status Register	PCSTS	127	7F	RW	CPU	REX520	Yes	2
Reserved		128-255	80-FF					3
Reserved		>255	>FF					4

Table Headings Meaning

Decimal—Decimal number of the Processor Register
Hex —Hex Number of the Processor Register
Type —Read or Write Access

R —Read-Only Register
W —Write-Only Register
RW—Read-Write Register

Scope —Processor Register's Scope

CPU —CPU-Wide Register
PROC—Per-Process Register

Impl —Chip in which the Processor Register is Implemented

REX520 —Implemented in the REX520 chip
SSC —Implemented in the System Support Chip
C-chip —Implemented in the C-chip

Init? —Initialized on Module RESET (Power-up, Negation of DCOK)
Category—Processor Register Category as previously defined

ACCESS TO CATEGORY 3 REGISTERS

Category 3 processor registers in the list above are passed to the RDAL by the Rigel CPU. As these registers are not implemented by the KA43 module, the SSC terminates the EPR read or write transaction after the period specified by the SSC Bus Timeout Control Register. During this time, no other instructions are executed by the CPU, and no other DAL transactions are possible. Therefore, Category 3 processor registers should not be referenced during normal operation of the system, as this may cause device or CPU timeouts to occur.

3.0.1.3.1 KA43 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:

Table 4: Category One Internal Processor Registers

Number				
Decimal	Hex	Register Name	Mnemonic	Section
18	12	Interrupt Priority Level	IPL	3.6.1
20	14	Software Interrupt Request	SIRR	3.6.1
21	15	Software Interrupt Summary	SISR	3.6.1
27	1B	Time Of Year Clock	TODR	8.1
56	38	Memory Management Enable	MAPEN	3.5.2
57	39	Translation Buffer Invalidate All	TBIA	3.5.2
58	3A	Translation Buffer Invalidate Single	TBIS	3.5.2
62	3E	System Identification	SID	3.7.1
63	3F	Translation Buffer Check	TBCHK	3.5.2

3.0.1.3.2 KA43 Unique Internal Processor Registers

Internal Processor Registers (IPR's) which are implemented uniquely on the KA43 (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:

Table 5: Category Two Internal Processor Registers

Number				
Decimal	Hex	Register Name	Mnemonic	Section
24	18	Interval Clock Control/Status	ICCS	3.6.3.6
32	20	Console Receiver Control/Status	RXCS	7.1.1
33	21	Console Receiver Data Buffer	RXDB	7.1.2
34	22	Console Transmit Control/Status	TXCS	7.1.3
35	23	Console Transmit Data Buffer	TXDB	7.1.4
38	26	Machine Check Error Register	MCESR	3.6.4
40	28	Accelerator Control and Status	ACCS	3.8
42	2A	Console Saved PC	SAVPC	3.6.7
43	2B	Console Saved PSL	SAVPSL	3.6.7
47	2F	Translation Buffer Tag	TBTAG	3.5.2
55	37	I/O System Reset Register	IORESET	9.5.3.1
59	3B	Translation Buffer Data	TBDATA	3.5.2
123	7B	Vector Interface Error Status Reg	VINTSR	5.3.6.4
124	7C	Primary Cache Tag Store	PCTAG	5.2.5.4
125	7D	Primary Cache Index Register	PCIDX	5.2.5.3
126	7E	Primary Cache Error Address Register	PCERR	5.2.5.2
127	7F	Primary Cache Status Register	PCSTS	5.2.5.1

3.0.2 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 KA43 implements these structures as defined in DEC STD 032, which should be referenced for a description of the PCB and the PCBB.

3.0.3 Data Types

The KA43 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
7. Absolute queues
8. Self-relative queues
9. Character string
10. F-floating
11. G-floating
12. D-floating

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

3.0.4 Instruction Set

The KA43 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 (MOVC3, MOVC5, CMPC3*, CMPC5*, LOCC*, SCANC*, SKPC*, SPANC*)
9. Operating system support
10. F_floating
11. G_floating

12. D_floating

*These instructions were in the microcode assisted category on the KA630-A (MicroVAX II) and therefore had to be emulated.

The RIGEL CPU chip (REX520) provides special microcode assistance to aid the macrocode emulation of the following instruction groups:

1. Character string (except MOVC3, MOVC5, CMPC3*, CMPC5*, LOCC*, SCANC*, SKPC*, SPANC*)
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 KA43 instruction set, indicating which instructions are implemented in the Floating Point Accelerator (FPA), and which instructions have microcode assists to speed up macrocode emulation.

3.0.5 Memory Management

The KA43 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).

3.0.5.1 Translation Buffer

To reduce the overhead associated with translating virtual addresses to physical addresses, the Rigel CPU employs a 64-entry, fully associative, translation buffer for caching VAX PTE's. Each entry can store a PTE for translating virtual addresses in either the VAX Process Space, or VAX System Space. The translation buffer is flushed whenever memory management is enabled or disabled (i.e., by writes to IPR 56), any page table base or length registers are modified (i.e. by writes to IPR's 13:8) and by writing to IPR 57 (TBIA) or IPR 58 (TBIS).

Each entry is divided into two parts: a 24-bit Tag Register and a 27-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, and a valid bit (TB.V) that indicates that the tag contains a valid VPN. The PTE register stores the 21-bit PFN field, the PTE.V bit, the PTE.M bit, and the 4-bit PROT field from the corresponding VAX PTE.

During virtual-to-physical address translation, the contents of the 64 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 and the TB.V bit indicates the entry is valid, then a translation buffer "hit" has occurred, and the contents of the corresponding PTE register are 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). This pointer is called the NLU pointer.

3.0.5.2 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 ZERO to this register with a MTPR instruction disables memory management and writing a ONE to this register with a MTPR instruction enables memory management. Writes to this register flush the translation buffer. To determine whether or not memory management is enabled, IPR 56 is read using the MFPR instruction. 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 mapped by the Page Table Entry must be invalidated in the translation buffer.

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

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.

There are three pairs of base and length registers which specify the base and length of P0, P1, and S0 space: IPR 8 (P0BR), IPR 9 (P0LR), IPR 10 (P1BR), IPR 11 (P1LR), IPR 12 (SBR), and IPR 13 (SLR).

The base and length of the P0, P1, and S0 page tables may be changed by writing the appropriate address or length to IPR 8 (P0BR), IPR 9 (P0LR), IPR 10 (P1BR), IPR 11 (P1LR), IPR 12 (SBR), and IPR 13 (SLR).

NOTE

Whenever the location or size of the system map is changed by changing the SBR (IPR 12) or SLR (IPR 13), the entire translation buffer must be cleared. The Rigel CPU accomplishes this by flushing the TB on any change to SBR, SLR, or to P0BR, P1BR, P0LR, and P1LR.

When a process context is loaded with LDPCTX, all TB entries that map process-space pages are automatically cleared. System-space mappings are preserved.

There are two IPRs which are used by diagnostic software to test the translation buffer: IPR 47 (TBTAG) and IPR 59 (TBDATA).

Figure 3: Translation Buffer Tag (TBTAG) - (IPR 47(DEC) 2F(HEX))

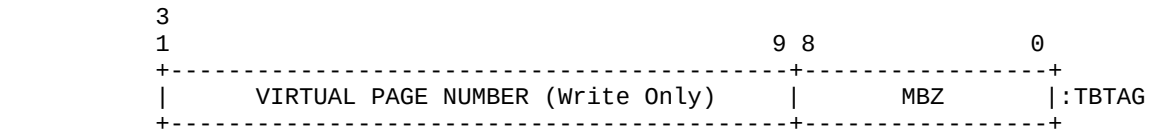
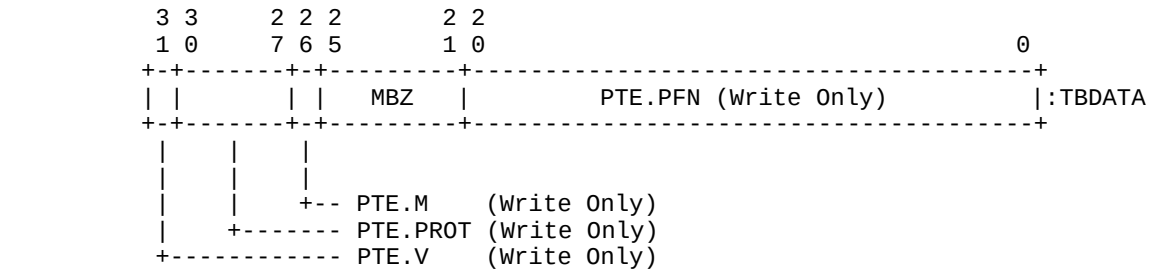


Figure 4: Translation Buffer Data (TBDATA) - (IPR 59(DEC) 3B(HEX))



Diagnostic software may use IPR 47 (TBTAG) and IPR 59 (TBDATA) to test the operation of the TB. A write to TBTAG writes bits <31:9> of the source data into the VPN field of the current tag location and clears the TB.V bit. A subsequent write to TBDATA interprets the source data as a PTE and writes PTE.V, PTE.M, PTE.PROT, and PTE.PFN into the current PTE location, sets the TB.V bit, and increments the NLU pointer.

These registers are provided for diagnostic purposes only and should not be written during normal operation. Writes to these registers must be done under very controlled conditions to achieve the desired results. Specifically, the following restrictions apply:

- The NLU pointer must be in a known state. A TBIA will initialize it to the first location in the array.
- Memory management must be enabled during the use of TBTAG and TBDATA, because writing to MAPEN implicitly does a TBIA and resets the NLU pointer.
- Data- and instruction-stream references during the use of TBTAG and TBDATA must not be allowed to change the NLU pointer.

NOTE

The TBIS, TBIA, TBCHK, TBTAG, and TBDATA IPRs are write only. An MFPR used to read any of these registers will cause the RIGEL CPU chip (REX520) to initiate a reserved operand fault.

3.0.6 Interrupts And Exceptions

Both interrupts and exceptions divert execution from the normal flow of control. 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), while an exception is caused by the execution of the current instruction and is typically handled by the current process (e.g., an arithmetic overflow),

3.0.6.1 Interrupts

Interrupts can be divided into two classes: non-maskable and maskable.

Non-maskable interrupts cause a halt via the hardware halt procedure which saves the PC, PSL, MAPEN<0> and a halt code in IPR's, raises the processor IPL to 1F and then passes control to the resident firmware. The firmware dispatches the interrupt to the appropriate service routine based on the halt code and hardware event indicators. Non-maskable interrupts cannot be blocked by raising the processor IPL, but can be blocked by running out of the Halt Protected Address Space (except those non-maskable interrupts that generate a halt code of 3). Non-maskable interrupts with a halt code of 3 cannot be blocked since this halt code is generated after a hardware reset).

Maskable interrupts cause the PC and PSL to be saved, the processor IPL to be raised to the priority level of the interrupt (except for Mass Storage and Network Interface interrupts where the processor IPL is set to 17 independent of the level at which the interrupt was received) and the interrupt to be dispatched to the appropriate service routine through the SCB.

The various interrupt conditions for the KA43 are listed below along with their associated priority levels and SCB offsets.

Table 6: Interrupt Priority Levels

Priority Level	Interrupt Condition	SCB Offset
	BREAK Generated by console device	*
1F:1E	Unused	
1D	Second Level Cache Tag Parity Error or Inconsistency Error	60
1C:17	Unused	
16	Interval Timer Interrupt	C0
14	Network	280
	Disk	284
	Programmable Cursor Chip (PCC)	288
	Serial Line (DZ)	28C

The interrupt system is controlled by three IPR's: IPR 18, the Interrupt Priority Level Register (IPLR), IPR 20, the Software Interrupt Request Register (SIRR), and IPR 21, the Software Interrupt Summary Register (SISR). The IPLR 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 registers is presented below, refer to DEC STD 032 for more information on these registers.

Figure 5: Interrupt Priority Level Register (IPLR) - (IPR 18(DEC) 12(HEX))

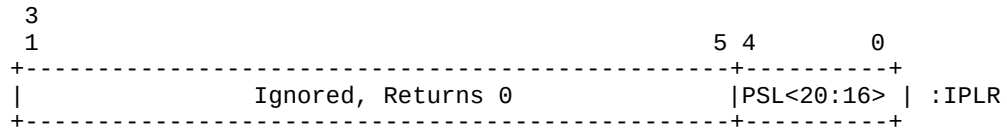


Figure 6: Software Interrupt Request Register (SIRR) - (IPL 20(DEC) 14(HEX))

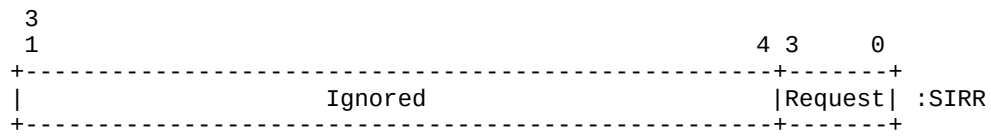
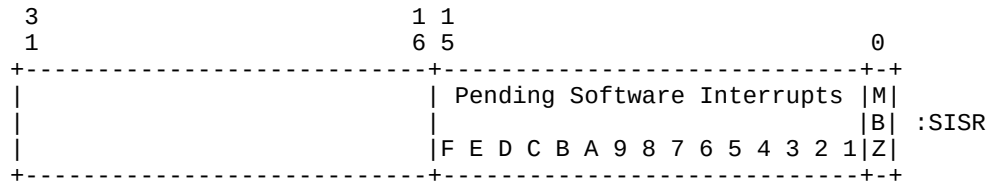


Figure 7: Software Interrupt Summary Register (SISR) - (IPL 21(DEC) 15(HEX))



3.0.6.2 Exceptions

Exceptions can be divided into 3 types: trap, fault and abort.

A trap is an exception that occurs at the end of the instruction that caused the exception. After an instruction traps, the PC saved on the stack is the address of the next instruction that would have normally been executed and the instruction can be restarted.

A fault is an exception that occurs during an instruction, and that leaves the registers and memory in a consistent state such that the elimination of the fault condition and restarting the instruction will give correct results. After an instruction faults, the PC saved on the stack points to the instruction that faulted.

An abort is an exception the occurs during an instruction, leaving the value of registers and memory UNPREDICTABLE, such that the instruction cannot necessarily be correctly restarted, completed, simulated or undone. After an instruction aborts, the the PC saved on the stack points to the instruction that was aborted, (which may or may not be the instruction that caused the abort) and the instruction may or may not be restarted depending on the class of the exception and the contents of the parameters that were saved.

Exceptions can also be divided into six classes. A list of exceptions, grouped by class, is given in the table below. Exceptions save the PC and PSL and in some cases, one or more parameters, on the stack. Most exceptions do not change the IPL of the processor (except the exceptions in serious system failures class, which set the processor IPL to 1F) and cause the exception to be dispatched to the appropriate service routine through the SCB (except for the

Interrupt stack not valid exception, and exceptions that occur while an interrupt or another exception are being serviced, which cause the exception to be dispatched to the appropriate service routine by the resident firmware). The exceptions listed below (except machine check) are described in greater detail in DEC STD 032. The machine check exception is described on greater detail in section 3.6.3. Exceptions that can occur while servicing an interrupt or another exception are listed in a table in section 3.6.7.

Table 7: Exception Classes

Exception Class	Type	SCB Offset
Arithmetic Exceptions		
Integer overflow	trap	34
Integer divide by zero	trap	34
Subscript range	trap	34
Floating overflow	fault	34
Floating divide by zero	fault	34
Floating underflow	fault	34
Memory Management Exceptions		
Access control violation	fault	20
Translation not valid	fault	24
Operand Reference Exceptions		
Reserved addressing mode	fault	1C
Reserved operand fault or abort		18
Instruction Execution Exceptions		
Reserved/Privileged instr.	fault	10
Emulated instruction	fault	C8,CC
Change mode	trap	40-4C
Breakpoint	fault	2C
Tracing Exception		
Trace	fault	28
System Failure Exceptions		
Console Error Halt	abort	*
Interrupt stack not valid	abort	*
Kernel stack not valid	abort	08

* —Dispatched by resident firmware rather than through the SCB

Table 7 (Cont.): Exception Classes

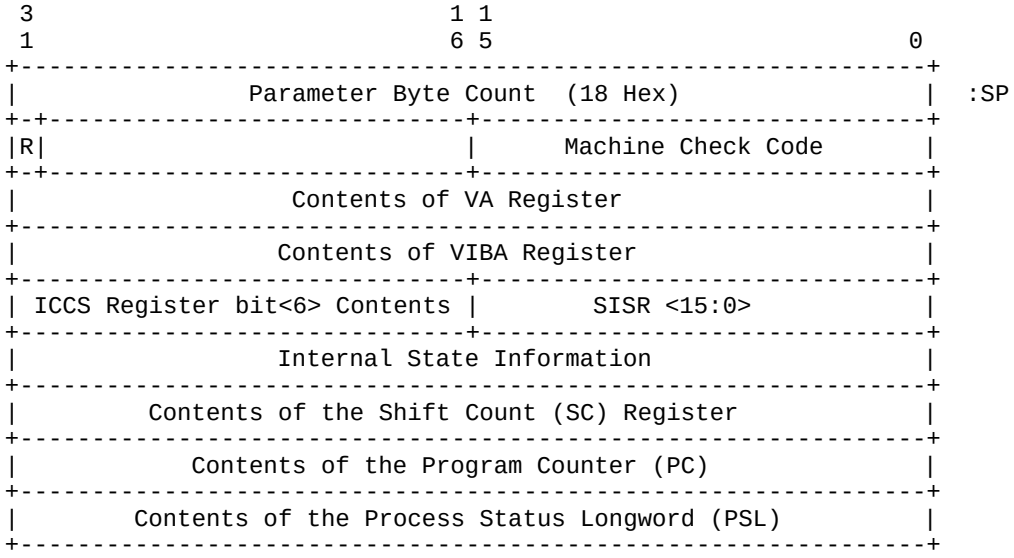
Exception Class	Type	SCB Offset
Machine check including: • RDAL Read or Write Error • CP_Bus Parity Errors • Primary Cache Read Error • CP_Bus Timeout Errors • Main memory NXM Errors • Main Memory Uncorrectable Errors • FPU Chip Status Error • Translation Buffer Status Error • Internally Detected Inconsistency	abort	04

* —Dispatched by resident firmware rather than through the SCB

3.0.6.3 Information Saved On A Machine Check Exception

In response to a machine check exception the PSL, PC, eight parameters, and a Byte Count are pushed on the stack as shown in the diagram below:

Figure 8: Information Saved On A Machine Check Exception



The meaning of this information and how it effects the recovery procedure is described in the following sections.

3.0.6.3.1 Byte Count

Byte Count<31:0> 00000018 (Hex), 24 (Decimal). This value indicates the number of bytes of information that follow on the stack (not including the PC and PSL).

3.0.6.3.2 "R" - Vax Restart Bit

Bit <31> of the Longword at (SP)+4 after a Machine Check is the Vax Restart Bit (R). If R is ONE , no state has been changed by the instruction which was executing when the error was detected. If R is ZERO, state has been changed by the instruction.

3.0.6.3.3 Machine Check Code Parameter

Bits <15:0> of the Longword at (SP)+4 after a Machine Check contents the Machine Check Parameter Code. This code value indicates the type of Machine check that occurred. A list of the possible machine check codes (in hex) and their associated causes follows:

Floating Point Errors - These codes indicate that the Floating Point Accelerator (FPA or CPU) chip detected an error while communicating with the RIGEL CPU/FPA chip during the execution of a floating point instruction. The most likely cause(s) of these types of machine checks are: a problem internal to the RIGEL CPU chip, a problem internal to the FPA or a problem with the interconnect between the two chips. Machine checks due to floating point errors MAY BE RECOVERABLE, depending on the state of the VAX RESTART bit (captured in bit tag(31) of the Longword at (SP)+4, the FIRST PART DONE flag (captured in PSL <27>). The error is recoverable only if the VAX RESTART bit is set and the FIRST PART DONE flag is cleared. Otherwise, the error is unrecoverable and depending on the current mode, either the current process or the operating system should be terminated or disable the FPA. The information pushed on the stack by this type of machine check is from the instruction that caused the machine check.

Table 8: Floating Point Error Machine Checks

Hex Code	Error Description
01	A protocol error was detected by the FPA chip during a F-Chip operand/result transfer.
02	An illegal opcode was detected by the FPA chip.
03	The FPA chip detected an operand parity error.
04	Unknown status was returned by the FPA chip.
05	The returned FPA chip result had an parity error.

Memory Management Errors - These codes indicate that the microcode in the RIGEL CPU chip detected an impossible situation (as described below) while performing functions associated with memory management. The most likely cause of this type of a machine check is a problem internal to the RIGEL chip. Machine checks due to memory management errors MAY BE RECOVERABLE. Depending on the current mode, either the current process or the operating system should be terminated. The state of the P0BR, P0LR, P1BR, P1LR, SBR and SLR should be logged.

Table 9: Memory Management Error Machine Checks

Hex Code	Error Description
08	Memory Management ERROR occurred while the RIGEL CPU was handling an Access Control Violation/Translation Not-Valid fault. The READ/WRITE that caused the error MISSED the translation buffer. This error is RECOVERABLE if the VAX RESTART bit or the FIRST PART DONE flag is set.
09	Memory Management ERROR occurred while the RIGEL CPU was handling an Access Control Violation/Translation Not-Valid fault. The READ/WRITE that caused the error HIT the translation buffer. This error is RECOVERABLE if the VAX RESTART bit or the FIRST PART DONE flag is set. The fact that the errant reference HIT the the Translation Buffer means that the RIGEL CPU is the most likely cause of the error.

Interrupt Errors - This code indicates that the interrupt controller in the RIGEL CPU requested a hardware interrupt at an unused hardware IPL. The most likely cause of this type of a machine check is a problem internal to the RIGEL CPU chip. Machine checks due to unused IPL errors may be RECOVERABLE. A non-vectored interrupt generated by a serious error condition (memory error, power fail or processor halt) has probably been lost. The operating system should be terminated.

Table 10: Interrupt Error Machine Checks

Hex Code	Error Description
0A	A hardware interrupt was requested at an unused Interrupt Priority Level (IPL). This error is RECOVERABLE if the VAX RESTART bit or the FIRST PART DONE flag is set.

Microcode Errors - This code indicates that an impossible situation was detected by the microcode during instruction execution. Note that most erroneous branches in the RIGEL CPU microcode will cause random microinstructions to be executed. The most likely cause of this type of machine check is a problem internal to the RIGEL CPU chip. Machine checks due to microcode errors may be RECOVERABLE. Depending on the current mode, either the current process or the operating system should be terminated.

Table 11: Microcode Error Machine Checks

Hex Code	Error Description
0B	An impossible state (ie. an undefined state bit combination in the microsequencer) was detected during a MOVC3 or MOVC5 instruction (not move forward, move backward, or fill). This error is RECOVERABLE if the FIRST PART DONE flag is set.
0C	An undefined TRAP code was produced by the I-BOX in the RIGEL CPU. This error is RECOVERABLE if the VAX RESTART bit is set, FIRST PART DONE is cleared.
0D	An undefined Control Store address was reached by the microsequencer. This error is RECOVERABLE if either the VAX RESTART bit or FIRST PART DONE is set.

Read Errors - These codes indicate that an error was detected while the RIGEL CPU was attempting to read from either the primary cache, the backup cache, or main memory. The most likely cause of this type of machine check must be determined from the state of the PCSTS, PCERR, BCERR DSER, MEMCSR32, MEMCSR33 and MEMCSR34. Machine checks due to read errors MAY BE RECOVERABLE, depending on the state of the VAX RESTART

flag, the FIRST PART DONE flag and the PCSTS<TRAP2> double error bit. If either the FIRST PART DONE flag or VAX RESTART is set, and PCSTS<TRAP2> are cleared then the error is recoverable. Otherwise, the error is unrecoverable and depending on the current mode, either the current process or the operating system should be terminated. The information pushed on the stack by this type of machine check is from the instruction that caused the machine check.

Table 12: Read Error Machine Checks

Hex Code	Error Description
10	A Primary Cache tag or data parity error occurred during a read.
11	A RDAL bus or data parity error occurred during a read.

Write Errors - These codes indicate that an error was detected while the RIGEL CPU was attempting to write to either the primary cache, the backup cache, or main memory. The most likely cause of this type of machine check must be determined from the state of the PCSTS, PCERR, BCERR DSER, MEMCSR32, MEMCSR33, MEMCSR34 and CBTCR. Machine checks due to write errors are most likely NON-RECOVERABLE because the CPU is capable of performing many read operations out of the primary cache before a write operation completes. For this reason, the information that is pushed onto the stack by this type of machine check cannot be guaranteed to be from the instruction that caused the machine check.

Table 13: Write Error Machine Checks

Hex Code	Error Description
12	An RDAL bus error (ie. ERR L terminated cycle) occurred on a write or clear write buffer.

RDAL Bus Errors - This codes indicate that the RIGEL CPU detected that the RDAL bus was in an UNDEFINED state. This Machine check is NON-RECOVERABLE.

Table 14: RDAL Bus Error Machine Check

Hex Code	Error Description
13	An Undefined RDAL bus state was detected by the RIGEL CPU.

3.0.6.3.4 Contents of the RIGEL CPU's internal VA Register

After a Machine Check the location at (SP)+8 captures the contents of the RIGEL CPU's VA Register at the time of the machine check. After a machine check of 10 or 11 this field represents the virtual address of the memory location that was being read when the error occurred. On machine checks of 12 this field represents the virtual address of a location that was being referenced either when the error occurred, or sometime after the error occurred. In other words, if the machine check occurred on a write operation, the contents of this field cannot be used for error recovery.

3.0.6.3.5 Contents of the RIGEL CPU's internal VIBA Register

After a Machine Check the location at (SP)+12 captures the contents of the RIGEL CPU's VIBA Register at the time of the machine check. After a machine check this field represents the virtual address of the last I-Stream fetch plus four.

3.0.6.3.6 ICCS Register Bit<6> Contents

After a Machine Check the location at (SP)+16 Bit<22> captures the contents of the RIGEL CPU's ICCS (Interval Clock Control and Status) Register Bit<6> the Interrupt Enable (IE) at the time of the machine check.

3.0.6.3.7 SISR Register Bits<15:0> Contents

After a Machine Check the location at (SP)+16 Bits<15:0> captures the contents of the RIGEL CPU's SISR (Software Interrupt Summary Register) Bits<15:0> at the time of the machine check.

3.0.6.3.8 Internal State Information

The Internal State Information Field is divided into five sub-fields. The contents of these sub-fields are described below:

Table 15: Internal State Information Field Description

Bits	Description
<31:24>	Delta PC (PC - backup PC)
<20:18>	AT - The Access Type at Machine Check time where: <000> —Read Access <001> —Write Access <010> —Modify Access <101> —Address Access <110> —Variable Bit Access <111> —Branch Access
<17:16>	DL - The Data Length at Machine Check time where: <00> —Byte Long <01> —Word Long <10> —Longword Long <11> —Quadword Long
<15:8>	Opcode - This field captures the opcode of the instruction that was being read or executed at the time of the machine check.
<3:0>	RN - This field captures the Register Number of the register that was the destination of the instruction that was being executed at the time of the machine check.

3.0.6.3.9 Contents of the Shift Count (SC) Register

After a Machine Check the location at (SP)+24 captures the contents of the RIGEL CPU's Shift Count (SC) Register at the time of the machine check. The RIGEL CPU uses this register is different ways dependent the instruction being executed. For a more complete description of the SC register please refer to the RIGEL CPU chip Specification.

3.0.6.3.10 Contents of the Program Counter (PC)

PC<31:0> Captures the virtual address of the start of the instruction being executed at the time of the machine check.

3.0.6.3.11 Contents of the Process Status Longword (PSL)

PSL<31:0> Captures the contents of the PSL at the time of the machine check.

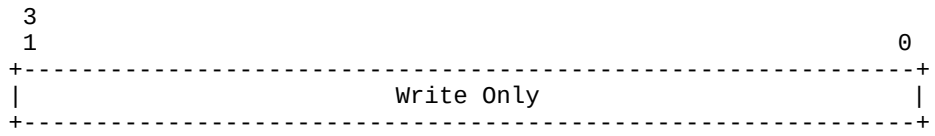
NOTE

Machine Checks must be acknowledged by the software by writing a ZERO to the MCESR (IPR 38)

3.0.6.4 Machine Check Error Register (MCESR) IPR 38

The Machine Check Error Register (IPR 38, MCESR) provides the mechanism by which software acknowledges receipt of a machine check. MCESR is a write-only register, and has the following format:

Figure 9: Machine Check Error Register (MCESR) - (IPR 38(DEC) 26(HEX))



When the RIGEL CPU microcode invokes the software machine check handler, it sets a "machine check in progress" flag. If a machine check or memory management exception occurs while this flag is set, the microcode initiates a console double error halt.

Software should clear the "machine check in progress" flag as soon as possible in the machine check handler by writing a ZERO to IPR MCESR. Doing so re-enables normal machine check and memory management exception reporting.

3.0.6.5 System Control Block (SCB)

The System Control Block (SCB) consists of two pages in main memory that contain the vectors by which interrupts and exceptions are dispatched to the appropriate service routines. The SCB is pointed to by IPR 17, the System Control Block Base Register (SCBB).

Figure 10: System Control Block Base Register (SCBB) - (IPL 17(DEC) 11(HEX))

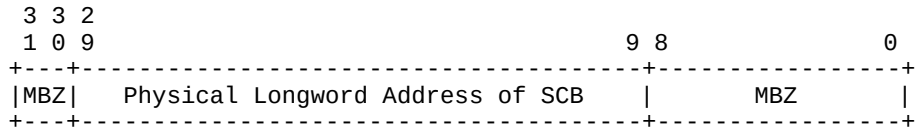


Table 16: The System Control Block format

SCB Offset	Interrupt/Exception Name	Type	# Params	Notes
00	Passive Release	Interrupt	0	IPL is raised to request IPL
04	Machine Check	Abort	6	Parameters reflect machine state
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/Vector Alignment Fault	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	Memory Soft Error Notification (Unused)			
58-5C	Unused	-	-	
60	Memory Hard Error Notification	Interrupt	0	IPL is 1D
64:80	Unused	-	-	
84	Software Level 1	Interrupt	0	

Table 16 (Cont.): The System Control Block format

SCB Offset	Interrupt/Exception Name	Type	# Params	Notes
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
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:FC	Unused			

The SCBB vector index is determined from bits <15:2> of the value supplied by external hardware. The new PSL priority level is determined by either the external interrupt request level that caused the interrupt or by bit <0> of the value supplied by external hardware. If bit<0> is 0, the new IPL level is determined by the interrupt request level being serviced. IRQ<3> sets the IPL to 17 (hex); IRQ<2>, 16 (hex); IRQ<1>, 15 (hex); and IRQ<0>, 14 (hex). If bit<0> of the value supplied by external hardware is 1, then the new IPL is forced to 17 (hex). If an IRQ<1> is being serviced, there is no guarantee that a higher priority device will not intercept the grant. Software must determine the level of the device that was serviced and set the IPL to the correct value.

Only device vectors in the range of 100 to FFFC (hex) should be used, except by devices emulating console storage and terminal hardware.

3.0.6.6 The Hardware Halt Procedure

The Hardware Halt Procedure is the mechanism by which the hardware assists the firmware in emulating a processor halt. The hardware Halt Procedure saves the current value of the PC in IPR 42 (SAVPC), and the current value of the PSL, MAPEN<0>, a halt code and VALID bit in IPR 43 (SAVPSL). SAVPSL<14> (VALID BIT) is set to ZERO if the PSL is valid and to ONE if it is not. The VALID BIT is undefined after a halt due to a system reset).

Figure 11: Console Saved PC (SAVPC) - (IPR 42(DEC) 2A(HEX))

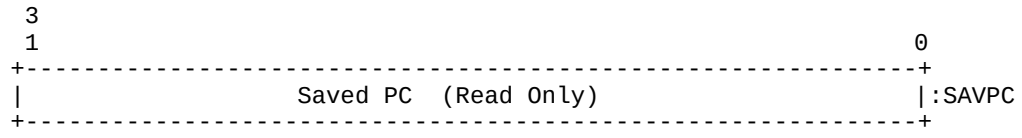
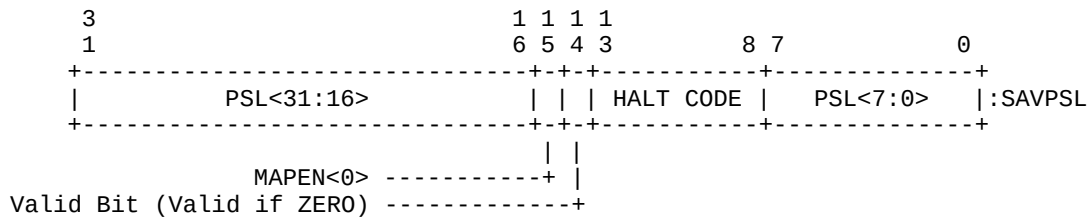


Figure 12: Console Saved PSL (SAVPSL) - (IPR 43(DEC) 2B(HEX))



The current stack pointer is saved in the appropriate internal register. The PSL is set to 041F 0000 (hex) (IPL=1F, kernel mode, using the interrupt stack) and the current stack pointer is loaded from the interrupt stack pointer. Control is then passed to the resident firmware at physical address 2004 0000 (hex) with the state of the CPU as follows:

Table 17: CPU State After a HALT

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 for halt code of 3)
SAVPSL<13:8>	Saved HALT code
SP	Current Interrupt Stack (IPR 4)
PSL	041F 0000 (hex)
PC	2004 0000 (hex)
MAPEN	0
ICCS	0 (for a halt code of 3)
SISR	0 (for a halt code of 3)
ASTLVL	0 (for a halt code of 3)
All else	Undefined

The firmware uses the halt code in combination with any hardware event indicators to dispatch the exception or interrupt that caused the halt to the appropriate firmware routine (either console emulation, power-up, reboot, or restart). A list of the interrupts and exceptions that can cause a halt along with their corresponding halt codes and event indicators follows:

Table 18: HALT Codes for UnMaskable Interrupts

Halt Code	Interrupt Condition	Event Indicator
2	External Halt (RIGEL CPU HALT_L pin asserted)	Halt Button Pressed
3	Hardware Reset (RIGEL CPU RESET pin negated)	Power-up only

Table 19: HALT Codes for Exceptions

Halt Code	Interrupt Condition
6	HALT instruction executed in Kernel Mode

Table 20: HALT Codes for Exceptions that occurred while Servicing an Interrupt or Exception

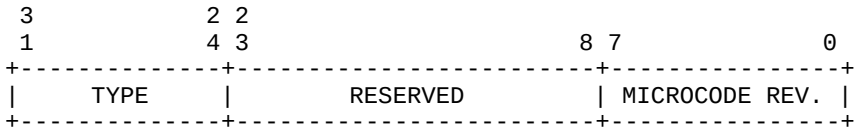
Halt Code	Interrupt Condition
4	Interrupt stack not valid during exception
5	Machine check during normal exception
7	SCB vector bits<1:0>= 11
8	SCB vector bits<1:0>= 10
A	CHMx executed while on interrupt stack
10	ACV or TNV during machine check exception
11	ACV or TNV during kernel stack not valid exception
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
3F	Powerup selftest failed in the RIGEL CPU (Microcoded)

3.0.7 System Identification

3.0.7.1 System Identification Register (SID) IPR 62

The System Identification Register (SID), IPR 62, is a Read Only register implemented per the DEC STD 032 in the RIGEL CPU chip. This 32-bit, Read-Only register is used to identify the processor type and its microcode revision level.

Figure 13: System Identification Register (SID) -(IPR 62(DEC) 3E(HEX))



SID<31:24> (TYPE) Processor type. This field always reads as 11 (decimal) indicating the processor is implemented using the RIGEL CPU chip.

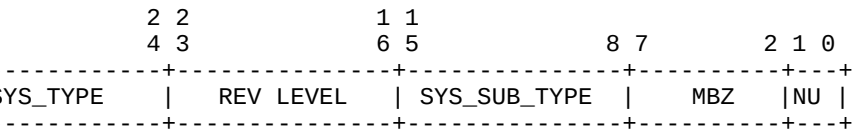
SID<23:8> Reserved for future use.

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

3.0.7.2 System Type Register (SYS_TYPE)

In order to distinguish between different CPU implementations that use the same CPU chip, the KA43 as must all VAX processors which use the RIGEL CPU chip, implements a MicroVAX System Type Register (SYS_TYPE) at physical address 2004 0004 (hex). This 32-bit Read-Only register is implemented in the KA43 ROM. The format of this register appears below:

Figure 14: System Type Register (SYS_TYPE)



SYS_TYPE<31:24> (SYS_TYPE) System Type Code. 4

SYS_TYPE<23:16> (REV LEVEL) Revision Level. This field reflects the revision level of the KA43 firmware. The upper nibble (bits <15-12>) is the MAJOR revision level and the lower nibble (bits <12-8>) is the MINOR revision level.

SYS_TYPE<15:8> (SYS_SUB_TYPE) System Sub-Type Code. This field reads as ??? (hex) for the KA43.

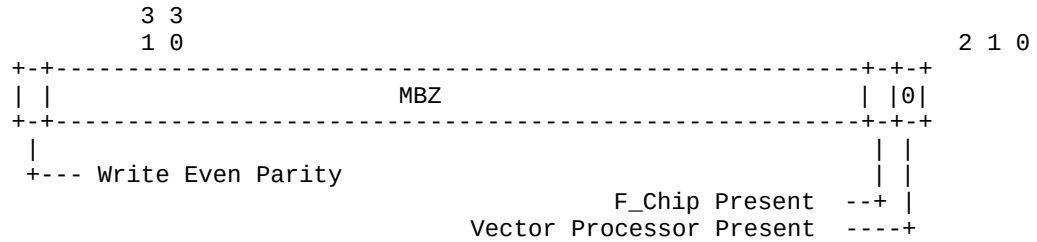
SYS_TYPE<7:2> Reserved. Must be read as zero.

SYS_TYPE<1:0> (NU) Number of Users. ???

3.0.8 Accelerator Control and Status Register (ACCS) IPR 40

The Accelerator Control and Status Register (IPR 40, ACCS) provides control for the F-chip, an optional vector unit, and the ability to generate bad data parity on write operations. The format of ACCS is as follows:

Figure 15: Accelerator Control and Status Register (ACCS) - (IPR 40(DEC) 28(HEX))



ACCS<31> Write Even Parity. Write Only. This bit enables the generation of bad data parity for write operations. If Write Even Parity is set to a ONE, all subsequent cache or memory writes and F-chip operand transfers will be done with bad data parity on all bits. This bit is used for diagnostics only, and should never be set during normal operations. Write Even Parity is automatically cleared if ACCS is read with an MFPR instruction. Also, the write-even-parity state is cleared at the start of any interrupt, exception, or console halt to prevent an exception stack frame from being written with bad data parity. The Write Even Parity is cleared during reset.

NOTE

The M bit should be set in any PTE that maps pages that are to be written while WRITE_EVEN_PARITY is enabled. Failure to do so may result in a PTE being written with bad parity during an M bit update.

ACCS<30:2> Reserved. Must be read as ZERO.

ACCS<1> F_Chip Present. Read/Write.

This bit enables use of the F-chip. If F_Chip Present is a ONE, floating point and longword-length integer multiply instructions are passed to the F-chip for execution. If F_Chip Present is a ZERO, the execution of a floating point instruction results in a reserved instruction fault. As an F-chip is included on every KA43 module, F_CHIP_PRESENT should be set during normal operation. If an F-chip error is detected, the F-chip may be disabled if the operating system emulation software is loaded. This bit is cleared during reset.

ACCS<0> Vector Processor Present. Read/Write.

If this bit is a ONE, vector instructions are decoded and passed to a vector unit for execution. If VECTOR_PRESENT is a ZERO, the execution of a vector instruction results in a reserved instruction fault. At Powerup, the console should set this bit to ZERO as a Vector Processor is NOT included on the KA43 module. This bit is cleared during reset.

CHAPTER 4

DC520 CPU REFERENCES

4.0.1 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.

4.0.1.1 Instruction-Stream Read references

The CPU has an instruction prefetcher with a 16-byte (4 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 quadword, Request Instruction-Stream (I-Stream) read reference.

4.0.1.2 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 quadword at a time, all data read references are translated into an appropriate combination of masked and unmasked, aligned quadword read references. If the required data is a byte, a word within a quadword, or an aligned quadword, then a single, aligned quadword, Demand D-Stream read reference is generated. If the required data is a word that crosses a quadword boundary, or an unaligned quadword, then two successive aligned quadword Demand D-Stream read references are generated. Data larger than a quadword is divided into a number of successive aligned quadword Demand D-Stream reads, with no optimization.

4.0.1.3 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 quadword at a time, all data write references are translated into an appropriate combination of masked and unmasked aligned quadword write references. If the required data is a byte, a word within a quadword, or an aligned quadword, then a single, aligned quadword, write reference is generated. If the required data is a word that crosses a quadword boundary, or an unaligned quadword, then two successive aligned quadword write references are

generated. Data larger than a quadword is divided into a number of successive aligned quadword writes.

CHAPTER 5

DC523 RIGEL FPU

The KA43 module includes a floating point accelerator to enhance the performance of floating point and certain integer calculations. These functions are implemented by the F-chip (KIWI Chip).

5.0.1 Floating Point Accelerator Instructions

The KA43 KIWI F-chip processes the following VAX instructions:

- F-floating add, subtract, multiply, divide, convert, move, compare, negate, and test instructions. ACBF, EMODF and POLYF are not processed by the F-chip (These are EMULATED).
- D-floating add, subtract, multiply, divide, convert, move, compare, negate, and test instructions. ACBD, EMODD and POLYD are not processed by the F-chip (These are EMULATED).
- G-floating add, subtract, multiply, divide, convert, move, compare, negate, and test instructions. ACBG, EMODG and POLYG are not processed by the F-chip (These are EMULATED).
- Longword length integer multiply instructions.

If the F-chip is not present or is disabled, the execution of a floating point instruction results in a reserved instruction exception. The execution of a longword length integer multiply is done by the RIGEL CPU microcode.

5.0.2 Floating Point Accelerator Data Types

The KA43 floating point accelerator supports the following data types:

- F-floating.
- D-floating.
- G-floating.
- Byte (conversion to and from floating formats).
- Word (conversion to and from floating formats).
- Longword (conversion to and from floating formats and multiply).

5.0.3 Operand and Result Transfer

The execution of the instructions that are accelerated by the F-chip is done through a cooperative effort between the RIGEL CPU and the F-chip. The RIGEL CPU parses the opcode and instruction specifiers and sends opcode and operands to the F-chip.

Operands from the GPRs, the instruction stream, and the primary cache are explicitly transferred from the RIGEL CPU to the F-chip. Floating point short literals are transferred in unexpanded form; it is the responsibility of the F-chip to expand them to the correct format.

Operands from the backup cache or from memory are returned to both the RIGEL CPU and the F-chip simultaneously—they are not received by the RIGEL CPU and then re-broadcast to the F-chip.

When the F-chip receives the last operand for an instruction, it begins the computation of the result. In parallel, the RIGEL CPU completes any instruction setup (e.g., parsing a destination specifier). The RIGEL CPU then requests the result from the F-chip and stalls until the result is returned. Finally, the RIGEL CPU stores the result in GPR or memory and sets the PSL condition codes.

The F-chip tests for exception conditions and reports them to the RIGEL CPU in response to the request for result. Detected exceptions include reserved operands, floating divide by zero, floating overflow, floating underflow, and data parity errors.

5.0.4 Powerup State

At powerup, the RIGEL CPU microcode disables the F-chip as part of the chip initialization process. The execution of any floating point instruction results in a reserved instruction exception until the F-chip is enabled. The console should enable the F-chip by setting bit <1> of the ACCS processor register and test the operation of the F-chip. If the F-chip fails these tests, the console should clear ACCS<1> again.

CONSOLE PROGRAMMER'S NOTE

The F-chip will not accept memory operands in I/O space. Because the F-chip executes longword-length integer multiply instructions as well as the floating point instructions, it may not produce correct results, or report operand parity errors if it is enabled during the execution of the console code from the boot ROM.

Therefore, the F-chip should be disabled on any console entry by writing a 0 to bit 1 of the ACCS processor register. Doing so will cause the RIGEL CPU to execute the integer instructions in microcode, and invoke a reserved instruction fault for the floating point instructions.

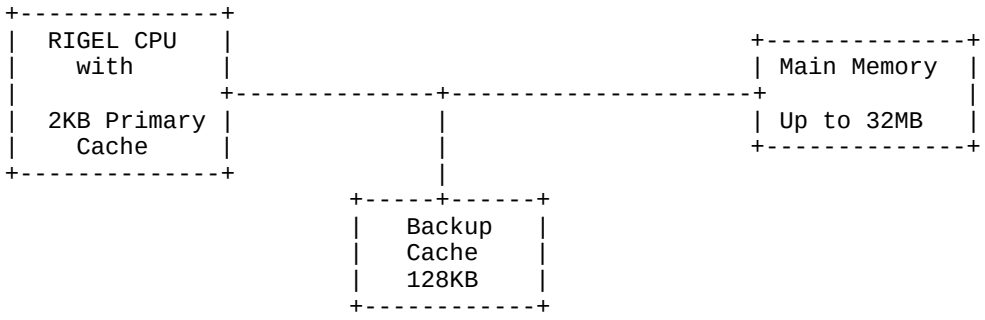
It is expected that the F-chip will be tested during powerup self-test. In this case, it is the responsibility of the console programmer to understand the restrictions involved and perform the test in a controlled manner.

CHAPTER 6

CACHE_MEMORY

To maximize CPU performance, the KA43 incorporates a two-level cache hierarchy. The first-level cache consists of a 2-kilobyte primary cache, which is contained entirely in the Rigel CPU. The backup cache is a 128-kilobyte cache which is used in combination with the RIGEL CPU to provide a performance boost for the system. The following diagram illustrates the memory hierarchy contained in the system.

Figure 16: KA43 Memory Hierarchy



6.1 Cacheable References

Any reference that is stored by the Primary or Backup Cache is called a "cacheable reference". The Primary and Backup 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. The types of CPU references that are stored are Request Instruction-Stream Read References, and Demand Data-Stream Read References other than Read-Lock References.

Whenever the CPU generates a non-cacheable reference, or a cacheable reference not stored in the Primary Cache, a single quadword reference of the same type is generated on the RDAL Bus.

Whenever the CPU generates a cacheable reference that is stored in the Primary Cache, no reference is generated on the RDAL Bus.

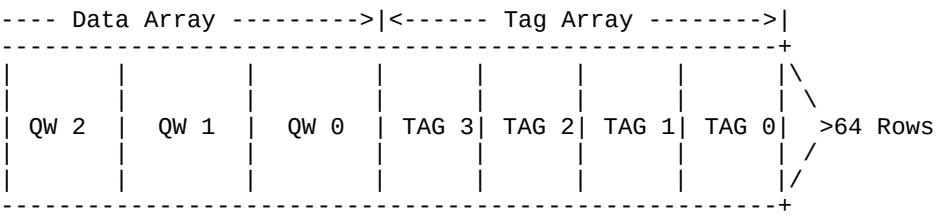
6.2 Primary Cache Overview

The primary cache is a 2-kilobyte cache, direct mapped, with a quadword fill and allocate (block) size. The cache is read-allocate, no-write-allocate, and write-through. The primary cache tag store contains one tag and one valid bit for each primary cache block. There are 256 tags mapping 256 quadword data blocks. Each tag entry includes an 18-bit tag, one valid bit, and one parity bit. Each data block contains 8 data bytes and 8 parity bits, one for each data byte.

6.2.1 Primary Cache Organization

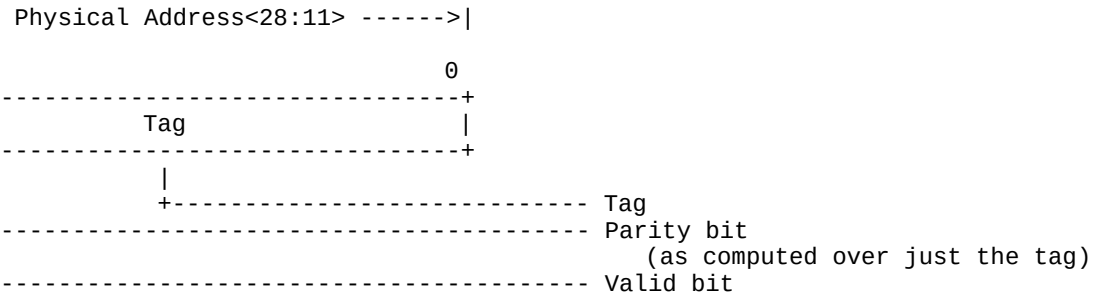
The primary cache is a 2KB, direct-mapped, write-through cache. The primary cache is arranged in 64 rows with 4 quadwords and four tag entries per row. This arrangement is shown in the following diagram:

Figure 17: Primary Cache Data and Tag Layout



Each tag entry of the memory is organized as follows:

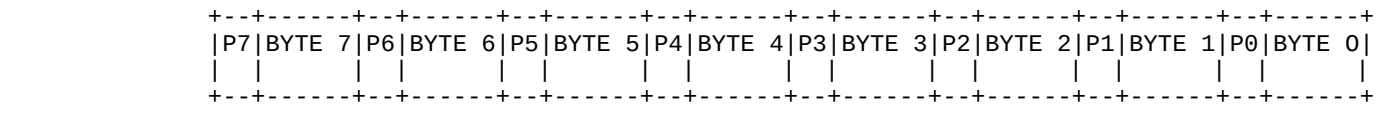
Figure 18: Primary Cache Tag Entry



The tag consists of bits <28:11> of the physical address (PA). The tag parity bit is the odd parity computed over 18 address bits, PA<28:11>. It is computed by the primary cache. The valid bit is used to indicate whether or not the corresponding entry in the primary cache is valid. The valid bit is not included in the tag parity calculation.

The data array has each quadword logically arranged as follows:

Figure 19: Primary Cache Data Entry



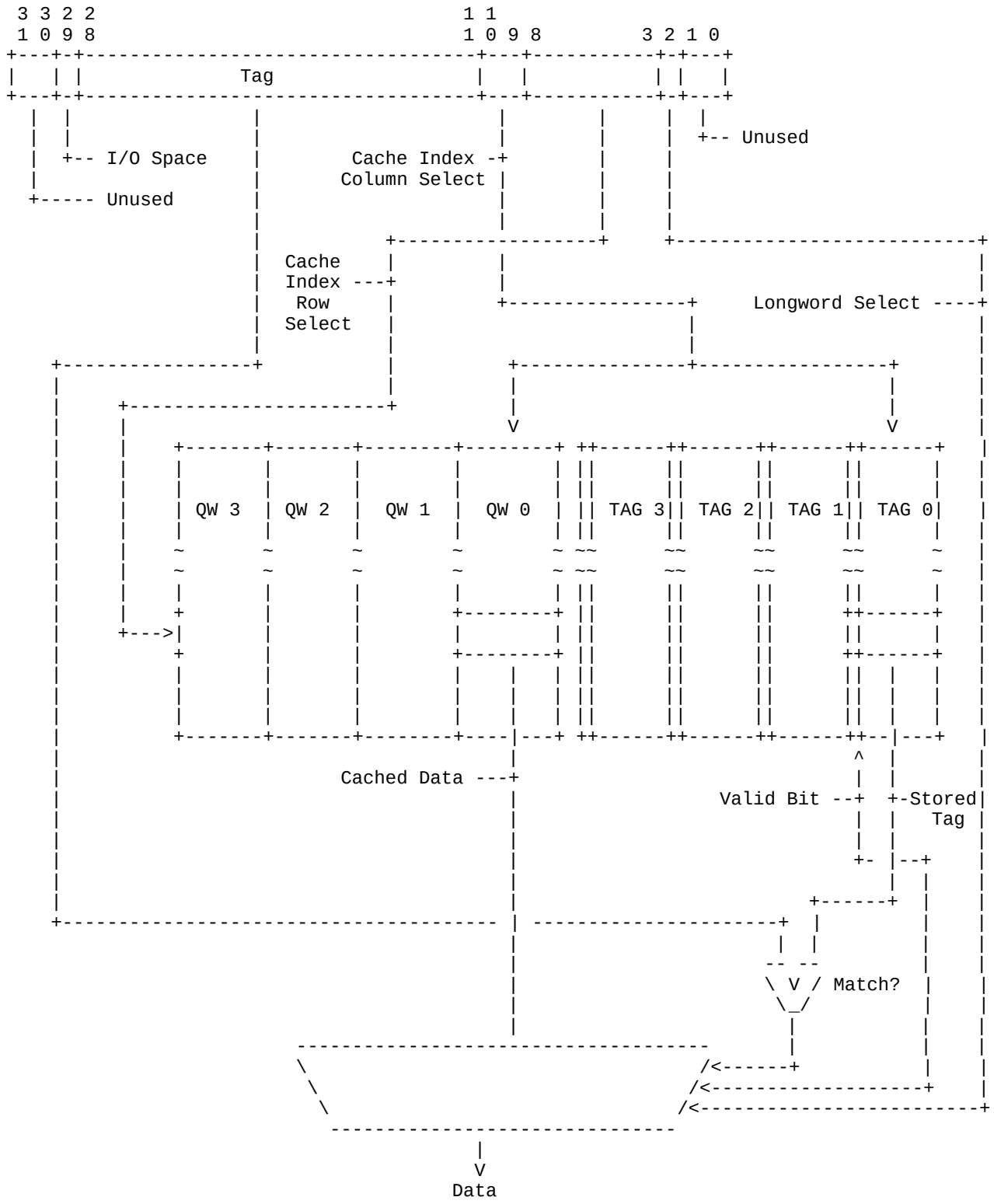
Each primary cache entry consists of one quadword. Odd parity information is maintained separately for each byte. The primary cache neither generates nor checks parity on data; it only stores it. The Rigel CPU Bus Interface Unit (BIU) takes the responsibility of checking parity for the data. If a parity error is detected on the data coming from the primary cache or data written into the primary cache, the primary cache may be flushed or switched off by the resulting microtrap routine.

6.2.2 Primary Cache Address Translation

The physical addresses supplied to the primary cache consist of 28 bits (address<29:2>). Bit <2> of the physical address is used to select a longword out of the quadwords of the primary cache. Bits <8:3> are used to select one of the rows of the primary cache memory. Since there are 4 tag entries per row, two bits of the address, bits <10:9>, are used to select one of the four columns. Bits <28:11> are stored as tags in the primary cache. Bit <29> of the address specifies I/O space. I/O space addresses are not cached.

Whenever the CPU requires an instruction or data, the contents of the Primary cache is checked to determine if the referenced location is stored there. The cache contents is checked by translating the physical address as shown in figure Figure 20 on the following page.

Figure 20: Primary Cache Physical Address Translation



On non-cacheable references, the reference is never stored in the cache, so a Primary cache "miss" occurs and a single quadword reference is generated on the RDAL Bus.

6.2.3 Primary Cache Data Block Allocation

Cacheable references that "miss" the primary cache, cause a quadword read to be initiated on the RDAL bus. When the requested quadword is supplied by either the backup cache or the main memory controller, the requested quadword is passed on to the CPU, and a data block is allocated in the cache to store it.

Since the KA43 supports 32MB (8M quadwords) of physical memory, up to 1 Mega quadwords "share" each row (four data blocks) of the cache. Contiguous programs larger than 2 KB or any non-contiguous programs separated by 2 KB will overwrite themselves when cache data blocks are allocated.

6.2.4 Primary Cache Behavior on Writes

On CPU generated write references, the primary cache is "write through". All CPU write references that "hit" the primary 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 primary cache, the cache entry containing the copy of the referenced location is invalidated.

6.2.5 Primary Cache Internal Processor Registers

The primary cache includes four registers that may be accessed using IPR reads and writes (MFPR and MTPR). These four registers are used to control the primary cache operation, store status, and for diagnostics and error recovery. The four registers are listed in the following table:

Table 21: Primary Cache Internal Processor Registers

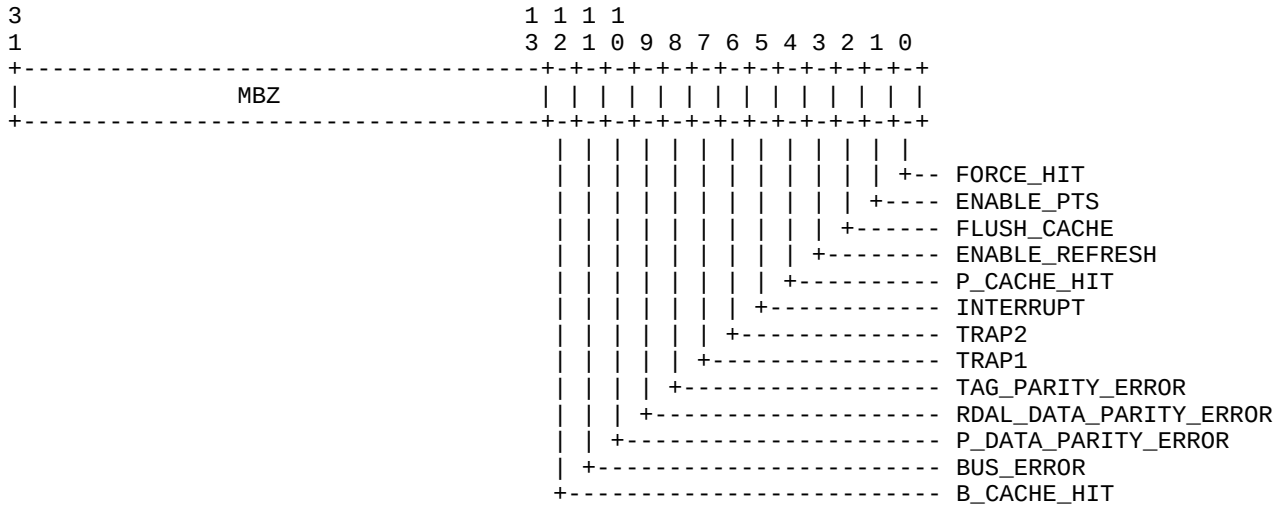
		IPR Number		
Register Name	Mnemonic	Hex	Dec	Type
Primary Cache Tag Array	PCTAG	7C	124	READ/WRITE
Primary Cache Index Register	PCIDX	7D	125	READ/WRITE
Primary Cache Error Address Register	PCERR	7E	126	READ/WRITE
Primary Cache Status Register	PCSTS	7F	127	READ/WRITE

6.2.5.1 Primary Cache Status Register (PCSTS) - IPR 127

The Primary Cache Status Register (PCSTS) is used for controlling the mode of operation of the primary cache, flushing the cache and maintaining information about errors. The PCSTS Register is considered LOCKED to errors that result in an interrupt if INTERRUPT (PCSTS<5>) or TRAP1 (PCSTS<7>) is set. For errors that result in a TRAP this register is considered LOCKED only if TRAP1 is set.

The format of the Status Register is shown below:

Figure 21: Primary Cache Status Register (PCSTS) - (IPL 127(DEC) 7F(HEX))



PCSTS<31:13> (MBZ) Must Be Zero Always read as Zeros. Write has no effect.

PCSTS<12> (B_CACHE_HIT) Backup Cache Hit Read Only. This bit indicates that the error condition that caused the Primary Cache Status Register to LOCK was a reference that hit in the backup cache. For RDAL parity errors, this bit can be used to determine the who was driving the the RDAL bus. (If B_CACHE_HIT is set the source was the backup cache and if B_CACHE_HIT is clear the memory subsystem was the source.) This bit is updated for any reference that has an error associated with it and if the Primary Cache Status Register is not already LOCKED. B_CACHE_HIT is cleared on reset.

PCSTS<11> (BUS_ERROR) Bus Error Read Only. This bit is set when a RDAL read, write or clear-write-buffer command results in an error. If the RDAL command was an I-STREAM read, INTERRUPT (PCSTS<5> is also set, and the error is reported as an IPL 1A (Hex) soft error interrupt. If the RDAL command was a D-STREAM read, write or clear-write-buffer, TRAP1 (PCSTS<7>) (or TRAP2 (PCSTS<6>)) is also set, and the error results in a MACHINE CHECK. This bit is updated for any reference that has an error associated with it and if the Primary Cache Status Register is not already LOCKED. BUS_ERROR is cleared on reset.

PCSTS<10> (P_DATA_PARITY) Primary Cache Data Error Read Only. This bit is set when a read hits in the primary cache and the requested data has a parity error. If the RDAL command was an I-STREAM read, INTERRUPT (PCSTS<5> is also set, and the error is reported as an IPL 1A (Hex) soft error interrupt. If the RDAL command was a D-STREAM read, write or clear-write-buffer, TRAP1 (PCSTS<7>) (or TRAP2 (PCSTS<6>)) is also set, and the error results in a MACHINE CHECK. This bit is updated for any reference that has an error associated with it and if the Primary Cache Status Register is not already LOCKED. P_DATA_ERROR is cleared on reset.

PCSTS<9> (RDAL_DATA_PARITY) RDAL Data Parity Error Read Only. This bit is set when the data returned in response to a non-I/O space RDAL read has a parity error. If the error is detected on the non-requested longword of a D-stream read, or on a either longword of an I-stream read, INTERRUPT (PCSTS<5> is also set, and the error is reported as an IPL 1A (Hex) soft error interrupt. If the RDAL Data parity error is detected on the requested longword of a D-stream read, TRAP1 (PCSTS<7>) (or TRAP2 (PCSTS<6>)) is also set, and the error is results in a MACHINE CHECK. This bit is updated for any reference

that has an error associated with it and if the Primary Cache Status Register is not already LOCKED. RDAL_DATA_PARITY is cleared on reset.

PCSTS<8> (TAG_PARITY_ERROR) Primary Cache TAG Parity Error Read Only. This bit is set if a primary cache tag parity error is detected during a read, write or invalidate reference and if the PCSTS register has not been not LOCKED by a previous error. If TAG_PARITY_ERROR is set INTERRUPT (PCSTS<5>) is also set, and the error is reported as an IPL 1A (Hex) soft error interrupt. If the reference was a D-STREAM read that hit, TRAP1 (PCSTS<7>) (or TRAP2 (PCSTS<6>)) is also set, and the error results in a MACHINE CHECK. This bit is updated for any reference that has an error associated with it and if the Primary Cache Status Register is not already LOCKED. TAG_PARITY_ERROR is cleared on reset.

PCSTS<7> (TRAP1) Write ONE to Clear. This bit is set when an error detected by the primary cache results in a MACHINE CHECK. PCSTS<12:8> and Primary Cache Error Address Register (PCERR IPR 126) are latched until TRAP1 is cleared. If this bit is set the primary cache is NOT automatically FLUSHED, however it is automatically disabled (although ENABLE_PTS (PCSTS<1>) is not changed). TRAP1 is cleared on reset and by writing ONE to it with a MTPR instruction.

PCSTS<6> (TRAP2) Write ONE to Clear. This bit is set when an error detected by the primary cache results in a MACHINE CHECK and TRAP1 (PCSTS<7> is already set. When TRAP2 is set, this indicates that a nested error occurred and that PCSTS<12:8> and Primary Cache Error Address Register (PCERR IPR 126) contain information about the first error that set TRAP1. This should consider a FATAL ERROR condition. If this bit is set the primary cache is NOT automatically FLUSHED, however it is automatically disabled (although ENABLE_PTS (PCSTS<1>) is not changed). TRAP2 is cleared on reset and by writing ONE to it with a MTPR instruction.

PCSTS<5> (INTERRUPT) Write ONE to Clear. This bit is set when an error detected by the primary cache results in an interrupt at IPL 1A (Hex). PCSTS<12:8> are latched unless INTERRUPT or TRAP1 (PCSTS<7>) was previously set and remain latched until INTERRUPT is cleared or another error sets TRAP1. If this bit is set the primary cache is automatically disabled (although ENABLE_PTS (PCSTS<1>) is not changed). INTERRUPT is cleared on reset and by writing ONE to it with a MTPR instruction.

PCSTS<4> (P_CACHE_HIT)Primary Cache Hit Read Only. This bit is the latched output value of the tag comparator. This bit is updated for all D-stream reads, writes or invalidate cycles and may used to test the primary cache hit logic. P_CACHE_HIT is cleared on reset and should be used for diagnostic purposes only.

PCSTS<3> (ENABLE_REFRESH)Enable Refresh Read/Write. When this bit is set the automatic refresh of the primary cache take place and the refresh counter increments. When this bit is a cleared, refresh is disabled, the refresh counter does not increment, and the refresh timer logic is disabled. This bit should be set during normal primary cache operations. ENABLE_REFRESH is cleared on reset.

PCSTS<2> (FLUSH_CACHE)Flush Primary Cache Write Only. This bit is used to clear all valid bits in the primary cache tag array. If this bit is written with a ONE, the primary cache is flushed. The hardware then clears this bit in the next cycle so that it is always read as a ZERO.

NOTE

The state of the primary cache is unpredictable if ENABLE_PTS=0 (PCSTS<1>). Therefore, the primary cache should be flushed before it is enabled. This may be done as a separate IPR write of FLUSH_CACHE before ENABLE_PTS is set in the Primary Cache Status Register. It may also be done by setting FLUSH_CACHE and ENABLE_PTS in the same IPR write to the Primary Cache Status Register.

PCSTS<1> (ENABLE_PTS)Enable Primary Cache Read/Write. This bit enables or disables normal operation of the primary cache. If the bit is set, both I-stream and D-stream references are cached, and primary cache tag and data parity errors are reported. I/O references are never cached. If the bit is cleared, all references (read, write, and invalidate) result in a miss. ENABLE_PTS is cleared on reset.

PCSTS<0> (FORCE_HIT)Force a Primary Cache Hit Read/Write. When this bit is set, the primary cache forces a hit for all memory references. Memory write requests still go to the external memory. I/O references are not affected (they are not cached).

When this bit is set, primary cache tag parity error reporting associated with D-stream reads, writes, and invalidates, and primary cache data parity errors associated with D-stream reads are disabled. RDAL errors (parity error associated with the data present on the RDAL or bus errors) are not affected by it. FORCE_HIT should not be used to satisfy I-stream reads, as primary cache tag or data parity errors detected during I-stream reads may cause a loop. The FORCE_HIT bit may be used to initialize the primary cache data array. FORCE_HIT is cleared on a reset. This bit is for diagnostics only and should be cleared during normal operation.

NOTE

When the primary cache is off (ENABLE_PTS = 0) and FORCE_HIT is set, the operation of the primary cache is unpredictable.

6.2.5.2 Error Address Register (PCERR) - IPR 126

For read commands, the Primary Cache Error Address Register (PCERR) latches and holds the physical address of an error which causes TRAP1 (PCSTS<7>) to set. As write errors are asynchronous to the instruction pipeline, the address latched for write commands is not the address of the error. For write commands, no address is available. The PCERR Register remains locked until TRAP1 is cleared in the Primary Cache Status Register (PCSTS).

The Primary Cache Error Address Register (PCERR) is also used to provide visibility into the Refresh Counter and Refresh Timer. An IPR write (MTPR) to the PCERR Register is used to update the the Refresh Counter and Timer. An IPR write to the PCERR Register loads Refresh Counter with <8:3> of the data, and Refresh Timer with <15:9> of the data.

An IPR read (MFPR) of the Primary Cache Error Address Register (PCERR) reads the error address out if TRAP1 (PCSTS<7>) is set. If TRAP1 is not set an IPR read is used to read the Refresh Timer in bits <15:9> and the value of the Refresh Counter in bits <8:3>.

Access to the Refresh Counter and Refresh Timer is provided for diagnostics only.

The format for the Error Address Register is as follows:

Figure 22: P-cache Err Addr Reg (PCERR) - (IPL 126(DEC) 7E(HEX)) (Read format if TRAP1 is set)

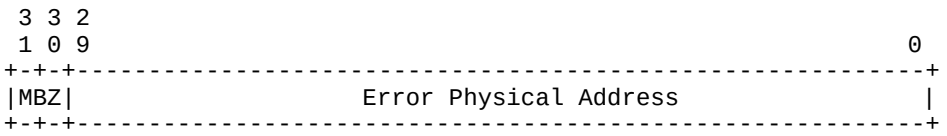
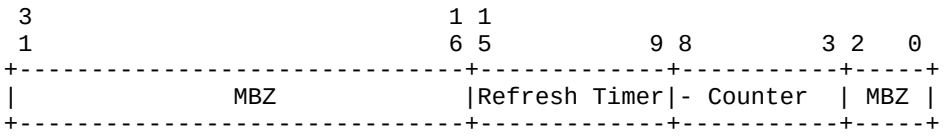


Figure 23: P-cache Err Addr Reg (PCERR) - (IPL 126(DEC) 7E(HEX)) (Read format if TRAP1 is NOT set; Write format irrespective of TRAP1)



NOTE

When ENABLE_REFRESH (PCSTS<3>) is set, a read of the Timer/Counter (through the PCERR Register) following an IPR write to it will result in a value different from the value written. This is due to the fact that when the ENABLE_REFRESH (PCSTS<3>) bit is set, the Refresh Counter will get incremented for every NOP operation, and the Refresh Timer will be incremented for every cycle during which the operation is not a NOP or an IPR write to the PCERR Register. Because of the internal latency involved in the execution of MxPRs, the count values of the Refresh Counter and Timer may change. Therefore, if the count values of the Refresh Counter and Timer are to remain unchanged, the ENABLE_REFRESH (PCSTS<3>) bit should be cleared.

6.2.5.3 Primary Cache Index Register (PCIDX) - IPR 125

The Primary Cache Index Register (PCIDX) provides the mechanism for reading and writing the tag array of the primary cache. During IPR (MTPR) writes to the Primary Cache Tag Array Register (PCTAG) (discussed later), the contents of the PCIDX Register are used to index the desired tag entry in the array. Therefore, the PCIDX Register must be written with the desired index before an IPR write (MTPR) to the PCTAG Register is performed.

The format of this register is as follows:

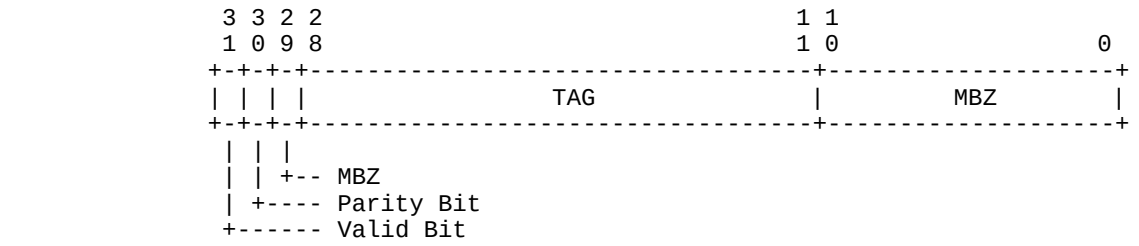
Figure 24: Primary Cache Index Register (PCIDX) - (IPR 125(DEC) 7D(HEX))

6.2.5.4 Primary Cache Tag Array Register (PCTAG) - IPR 124

The Primary Cache Tag Array Register (PCTAG) is a 32-bit logical register that provides the mechanism for reading and writing the tag array of the primary cache.

The format for this register is as follows:

Figure 25: Primary Cache Tag Array Register (PCTAG) - (IPR 124(DEC) 7C(HEX))



6.2.6 Writing and Read the Primary Cache Tag Array

As mentioned earlier, during IPR read/write of the Primary Cache Tag Array Register (PC-TAG), the index for the tag entry to be accessed, is supplied by the Primary Cache Index Register (PCIDX). In order to write a tag entry in the primary cache, first the index of the tag entry is written in the PCIDX Register by issuing an MTPR. Then an MTPR is issued for the Primary Cache Tag Array Register (PCTAG) with the desired value of the valid bit, parity bit and tag address<28:11> in data bits<31:30,28:11>.

In order to read a tag entry in the primary cache, the index of the tag entry is written in the Primary Cache Index Register (PCIDX) by issuing an MTPR. Then an MFPR is issued for the Primary Cache Tag Array Register (PCTAG).

6.2.7 Primary Cache Error recovery

When an error is detected in the primary cache, the primary cache latches error information in the PCSTS and PCERR which describes the error it then is automatically disabled. The information in the PCSTS Register, along with the way that the error was reported (machine check or interrupt), indicates the exact type of error.

If the error was a tag parity error, the entire tag store must be written with invalid tag with good parity. The PCERR contains the address if the tag in error only if the tag parity error is report as a machine check.

For all other errors, the P-cache should simply be flushed by writing ONE the the FLUSH_ CACHE bit in the P-cache Status Register (PCSTS<2>).

The following is the recommended sequence to bring the P-cache back to normal operation:

1. Save the Primary Cache Status Register.
2. Save the Primary Cache Error Address Register.
3. If the error was tag parity error that write all tags in the P-cache by:
    1. Write the Primary Cache Index Register with the next index.
    2. Write the Primary Cache Tag Array Register with TAG=0 (arbitrarily chosen), parity=0 (odd parity for chosen tag value), and valid=0
4. OR the FLUSH_CACHE bit (<2>) into the saved value of the Primary Cache Status Register and write this value back into the Primary Cache Status Register. This will clear those error bits that were set, flush the cache, and enable if it was enabled before.

6.2.8 Primary Cache Initialization

At powerup, the primary cache must be initialized. The console macrocode should load the Primary Cache Status Register (PCSTS) with the desired values for the FORCE_HIT, ENABLE_PTS, and ENABLE_REFRESH bits. The microcode should clear the INTERRUPT, TRAP1 and TRAP2 bit sin the PCSTS register. It should also invalidate the entire P-cache by issuing and IPR write (MTPR) to PCSTS, writing ONE in bit <2> (FLUSH_CACHE). Then each tag store entry should be loaded with an invalid tag with good parity. Each entry may be written via a write to PCIDX, followed by a write to PCTAG.

6.2.9 Primary Cache Diagnostics

The primary cache may be tested by reading and writing tags via PCIDX and PCTAG. Error detection may be tested by constructing an error and then reading the state from PCSTS and PCERR. The primary cache refresh counter and timer may be tested by reading and writing the Primary Cache Error Register (PCERR).

6.2.10 Error Handling by the Primary Cache

The primary cache is responsible for latching error signals (if any) of the following errors occurs: primary cache tag parity error, primary cache data parity error, RDAL data parity error, RDAL bus error, or an F-chip result parity error. The latter four errors are detected by the RIGEL CPU. Errors are reported in one of two ways: as a soft error interrupt at IPL 1A (hex), or as a machine check.

When an error is detected, the primary cache sets TRAP1, TRAP2, or INTERRUPT in the Primary Cache Status Register (PCSTS) and conditionally latches other bits to indicate the type of error. When TRAP1, TRAP2, or INTERRUPT are set in the PCSTS Register, the primary cache is automatically disabled.

Primary cache tag parity errors are reported if a tag parity error is detected during a read, write, or invalidate reference and the if Primary Cache Status Register has not been already LOCKED by a previous error (ie.: ENABLE_PTS=1, TRAP1=0, TRAP2=0, INTERRUPT=0, and FORCE_HIT=0). Tag parity errors are always reported as an interrupt. If the reference was a D-stream read that hit, the error is also reported as a machine check.

Primary cache data parity errors are reported if a data parity error is detected during a read reference that hit in the primary cache (unless FORCE_HIT (PCSTS<0>=1). Primary cache data parity errors are reported as a machine check if the reference was a D-stream read. If the reference was an I-stream read, primary cache data parity errors are reported an interrupt.

RDAL data parity errors are reported if a data parity error is detected during a non-I/O space read reference that missed in the primary cache. RDAL data parity errors detected on the requested longword of a D-stream read are reported as a machine check. RDAL parity errors detected on the non-requested longword of a D-stream read or on either longword of an I-stream read are reported as an interrupt.

RDAL bus errors are reported if a read, write, or clear write buffer command is terminated with an error (RDAL bus signal ERR_L asserted). However, this should never occur in this system. Bus errors detected during D-stream read, write, or clear write buffer commands are reported as a machine check. Bus errors detected during an I-stream read are reported as an interrupt.

NOTE

RDAL bus errors may also be reported for an EPR read or read interrupt vector command that is terminated with the RDAL signal ERR_L. In these cases, however, the Primary Cache Status Register (PCSTS) does not lock and the RIGEL CPU processes the error entirely by microcode, with no error reported to the software.

F-chip result parity errors are reported if a data parity error is detected during a result transfer from the F-chip. Result parity errors are always reported as a machine check.

Errors reported as interrupts set INTERRUPT (PCSTS<5>) and update bits PCSTS<12:8> with information describing the error. However, if either INTERRUPT or TRAP1 are already set when the error is detected, bits PCSTS<12:8> are not updated, bits PCSTS<12:8> reflect the first error detected.

Errors reported as a machine check set TRAP1 in the Primary Cache Status Register (PCSTS), update bits PCSTS<12:8>, and load the Primary Cache Error Address Register (PCERR). However, if TRAP1 is already set when the error is detected, TRAP2 is set and neither bits PCSTS<12:8> nor the PCERR Register are updated. This causes bits PCSTS<12:8> and the PCERR Register to reflect the first error detected. Note that the state corresponding to the machine check will overwrite any information latched due to a previous interrupt. It is assumed that errors reported as a machine check are more important than those reported as an interrupt.

In the RIGEL CPU, primary cache data parity errors are reported only if the read reference hits in the cache. On the other hand, primary cache tag parity errors are reported whenever the error is detected. These are following two reasons for this inconsistency:

1. Primary Cache Tag entries can be directly written without any side effects, using MTPR macro instructions. There is no direct and easy way of writing the Primary Cache data array.

2. If we reported primary cache tag parity error only under hits, there is a possibility that a stuck-at fault in the tag array might not get detected for a long time. Meanwhile the system will run at degraded performance; this is undesirable.

For each error type, the following table shows the resulting status register values:

Figure 26: Primary Cache Detectable Single Errors

<---------- Error Conditions ---------->				<- Resulting Primary Cache Status Register Values -->												
			P-cache	12	11	10	09	08	07	06	05	04	03	02	01	00
Error	Command	LW	Hit	BCH[5]	BER	PDP	DDP	PTP	TR1	TR2	INT	HIT	RFR[1]	FLS	PON[2]	FHT
P-cache tag par err	D-Read	x	Yes	0	0	0	0	1	1	0	1[3]	x	1	0	1	0
	D-Read	x	No	0	0	0	0	1	0	0	1	x	1	0	1	0
	I-Read	x	Yes	0	0	0	0	1	0	0	1	x	1	0	1	0
	I-Read	x	No	0	0	0	0	1	0	0	1	x	1	0	1	0
	Write	x	x	0	0	0	0	1	0	0	1	x	1	0	1	0
	Inval	x	x	0	0	0	0	1	0	0	1	x	1	0	1	0
	Other[6]	x	x	0	0	0	0	0	0	0	0	x	1	0	x	0
P-cache data par err	D-Read	x	Yes	0	0	1	0	0	1	0	0	x	1	0	1	0
	I-Read	x	Yes	0	0	1	0	0	0	0	1	x	1	0	1	0
	Other[6]	x	x	0	0	0	0	0	0	0	0	x	1	0	x	0
RDAL data par err	D-Read	1	No	bch	0	0	1	0	1	0	0	x	1	0	x	0
	D-Read	2	No	bch	0	0	1	0	0	0	1	x	1	0	x	0
	I-Read	x	No	bch	0	0	1	0	0	0	1	x	1	0	x	0
	Other[6]	x	x	0	0	0	0	0	0	0	0	x	1	0	x	0
RDAL bus err	D-Read	x	No	bch	1	0	0[4]	0	1	0	0	x	1	0	x	0
	I-Read	x	No	bch	1	0	0[4]	0	0	0	1	x	1	0	x	0
	Write	x	x	bch	1	0	0	0	1	0	0	x	1	0	x	0
	Cl Wr Buf	x	x	bch	1	0	0	0	1	0	0	x	1	0	x	0
	Other[6]	x	x	0	0	0	0	0	0	0	0	x	1	0	x	0
F-chip Rslt par err	Rd Rslt	x	x	0	0	0	0	0	1	0	0	x	1	0	x	0

Notes:

1. In all of these cases, it is assumed that ENABLE_REFRESH (PCSTS<3>) and FORCE_HIT (PCSTS<0>) bits are set to ONE and ZERO respectively. This is the normal state of the cache, and other states may change the way errors are reported.

2. The P-cache must be enabled to get a P-cache tag or data parity error. It may or may not be enabled when a DAL data parity error,RDAL bus error, or an F-chip result parity error are detected.

3. P-cache tag parity errors always cause an interrupt request. If the error was the result of a D-stream read that hit, a microtrap is also started.

4. If a read transaction is terminated by ERR_L, data parity is ignored. Therefore, the RDAL data parity error bit in the status register is never set for a read terminated in ERR_L.

Figure 26 Cont'd. on next page

Figure 26 (Cont.): Primary Cache Detectable Single Errors

5. B_CACHE_HIT is always loaded when an error is detected. However, P-cache tag and data parity errors are detected as part of a P-cache reference, so the bit will always be a 0 for those errors.

6. TRAP1, TRAP2, and INTERRUPT are not set in the Primary Cache Status Register as the result of commands for which error detection is inhibited. For example, tag parity errors are inhibited for commands that do not access the tag store. Similarly, RDAL bus errors are not reported for EPR read, EPR write, or read interrupt vector transactions terminated by ERR_L, as those commands are handled specially by the microcode.

The resulting values shown in the table above assume that TRAP1, TRAP2 and INTERRUPT are all ZERO when the error is detected. In that case, bits PCSTS<12:8> are updated as shown. If TRAP1, TRAP2, or INTERRUPT are ONE when the error is detected, the update of the bits is as shown in the following table.

Figure 27: Primary Cache Detectable Double Errors

State Before Error				State After Error						
7 TR1	6 TR2	5 INT	Type	7 TR1	6 TR2	5 INT	Load <12:8>	Lock Err Addr Reg	Disable P-cache	Notes
0	0	0	Int	0	0	1	Yes	No	Yes	Single interrupt
0	0	1	Int	0	0	1	No	No	Yes	Multiple interrupt
0	1	0	Int	0	1	1	Yes	No	Yes	Possible but not likely
0	1	1	Int	0	1	1	No	No	Yes	Possible but not likely
1	0	0	Int	1	0	1	No	No	Yes	Interrupt after trap
1	0	1	Int	1	0	1	No	No	Yes	Multiple interrupt after trap
1	1	0	Int	1	1	1	No	No	Yes	Interrupt after multiple trap
1	1	1	Int	1	1	1	No	No	Yes	Multiple interrupt after multiple trap
0	0	0	Trap	1	0	0	Yes	Yes	Yes	Single trap
0	0	1	Trap	1	0	1	Yes	Yes	Yes	Trap after interrupt
0	1	0	Trap	1	1	0	Yes	Yes	Yes	Possible but not likely
0	1	1	Trap	1	1	1	Yes	Yes	Yes	Possible but not likely
1	0	0	Trap	1	1	0	No	No	Yes	Double trap
1	0	1	Trap	1	1	1	No	No	Yes	Double trap after interrupt
1	1	0	Trap	1	1	0	No	No	Yes	Multiple trap
1	1	1	Trap	1	1	1	No	No	Yes	Multiple trap after interrupt

6.2.10.1 Primary Cache Error Recovery

When an error is detected, the primary cache latches state which describes the error in the Primary Cache Status Register (PCSTS) and the Primary Cache Error Address Register (PCERR). The primary cache is then automatically disabled. Error recovery consists of reading the error information, taking the appropriate action to correct the error, and clearing the lock bits in the PCSTS Register.

The first step in error recovery is to read both the Primary Cache Status Register (PCSTS) and Primary Cache Error Address (PCERR) Register. The information in the PCSTS Register, along with the way that the error was reported (machine check or interrupt), indicates the exact type of error.

If the error was a tag parity error, the entire tag store must be written with invalid tags with good parity. The Primary Cache Error Address Register (PCERR) contains the address of the tag in error only if the tag parity error is reported as a machine check.

For all other errors, the primary cache should simply be flushed by writing a ONE to the FLUSH_CACHE bit (PCSTS<2>.

To complete error recovery, the error bits should be cleared in the Primary Cache Status Register (PCSTS), and the primary cache should be enabled. If the error rate is such that the primary cache is to remain disabled.

The following is the recommended sequence to bring the primary cache back to normal operation:

1. Save the Primary Cache Status Register (PCSTS).
2. Save the Primary Cache Error Address Register (PCERR).
3. If the error was a tag parity error then write all tags in the primary cache:

   Write the Primary Cache Index Register (PCIDX) with the next index.
   Write the Primary Cache Tag Array Register (PCTAG) with TAG=0 (arbitrarily chosen), PARITY=1 (odd parity for tag chosen), and VALID=0.
4. OR the FLUSH_CACHE bit (PCSTS<2> into the saved value of the Primary Cache Status Register (PCSTS) and write this value back into the PCSTS Register. This will clear those error bits that were set, flush the cache, and enable it if it was enabled before.

6.3 Backup Cache Overview

The 128 KB 2nd level write-back cache has quadword allocate and fill. Lines are allocated for both read-miss and write-miss. For diagnostic purposes, the 128 KB of cache data RAM is directly addressable in the high 256MB of memory space. Also the tags (16Kx11) are directly addressable in an area of I/O space.

Parity errors in the cache are detected by the Rigel P-chip or F-chip. In general, a parity error in the cache will result in a machine check. As is the case with the Rigel XRP module, particularly gross hardware errors (hard failure of a cache RAM chip or a soft RAM chip failure combined with bad luck) may result in the P-chip being unable to execute the machine check handler. In this case, the processor halts with a double error.

Tag parity errors are detected by hardware on the system board. Unfortunately, detection does not take place in the same cycle in which the error occurs. In addition, since this is a write-back cache with no logic in the memory system for marking entries "checked out", tag errors result in irrecoverable loss of state. Upon detection of the error, hardware shuts off the 2nd level cache (to prevent a potential "sunset loop") and signals the P-chip via a

hard-error interrupt (level 1D). Upon fielding a level 1D interrupt, software should attempt to log the error and then crash the machine.

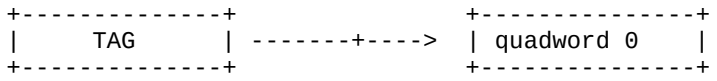
There are eight bits of parity, one bit corresponding to each data byte. Fourteen bits of address are needed to access the cache; bits <16:3> of the VAX physical address are used as the cache index.

When the backup cache returns data to the primary cache, it returns a quadword, which is the fill size of the primary cache.

6.3.1 Backup Cache Organization

The backup cache tag store is organized such that one tag corresponds to one quadword block of the cache. When a cache tag miss occurs on a read or a write, a block is allocated, and the block is filled.

Figure 28: Tag Bits they Correspond To Backup Cache Data



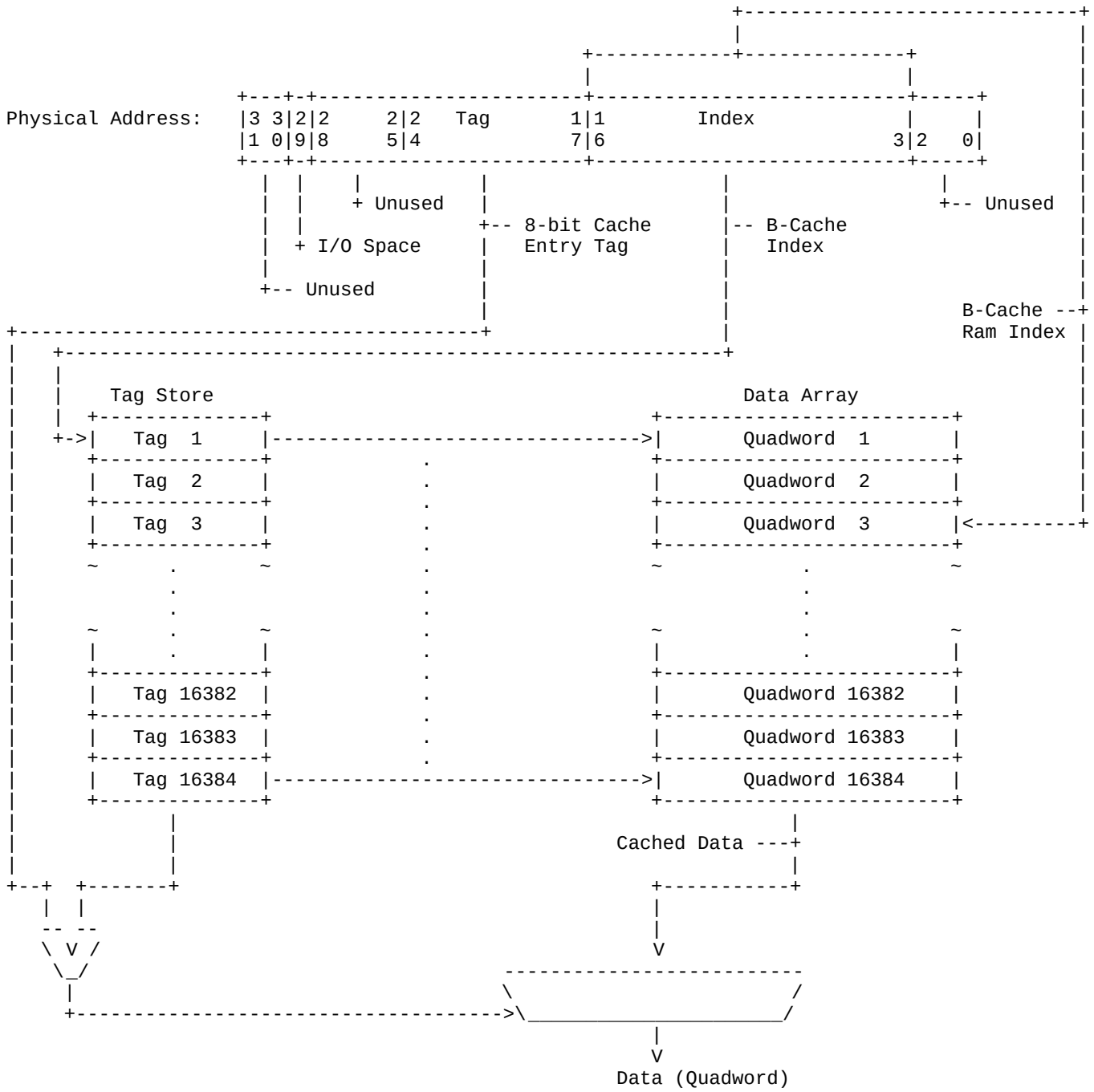
6.3.2 Backup Cache Address Translation

The physical addresses supplied to the backup cache consists of 27 bits (address<29:3>). There are 16K Quadwords of data, fourteen bits of the address, bits <16:3>, are used to select one. Fourteen bits of the address, bits <16:3>, are used to select one tag. Bits <24:17> are stored as tags in the backup cache. Bit <29> of the address specifies I/O space and is not used since I/O space addresses are not cached.

On non-cacheable references, the reference is never stored in the cache, so a backup cache "miss" occurs and an octaword reference is generated on the RDAL Bus.

Whenever the Rigel CPU requires an instruction or data, that was not found in the primary cache the contents of the backup cache is checked to determine if the referenced location is stored there. The cache contents are checked by translating the physical address as follows:

Figure 29: Backup Cache Physical Address Translation



6.3.3 Backup Cache Data Block Allocation

On cacheable references that miss the primary cache, a quadword read is initiated on the RDAL Bus. If the requested quadword cannot be found in the backup cache, an octaword is provided by the main memory controller, both caches allocate a data block for storing the data, (the primary allocates and fills a quadword; the backup cache allocates and fills a quadword as well) and the requested quadword is passed on to the CPU.

Since the KA43 supports 32MB (4M quadwords) of physical memory, up to 256 quadwords "share" each data block (quadword) of the cache. Contiguous programs larger than 128KB, or non-contiguous programs separated by 128KB will over-write themselves in the backup cache when cache data blocks are allocated.

6.3.4 Backup Cache Behavior on Writes

The backup cache is "write back". All CPU write references that "hit" the backup cache update the copy in the cache but do NOT cause the contents of the referenced location in main memory to be updated.

6.3.5 Maintaining Primary Cache Consistency

The following is a matrix showing the proper sequence of events for re-enabling a tag store which has been disabled. The matrix assumes that each tag store has been properly initialized. It also assumes that STATUS_LOCK (BCSTS<0>) is not set. If STATUS_LOCK is set, the sequence given in the error recovery section should be followed.

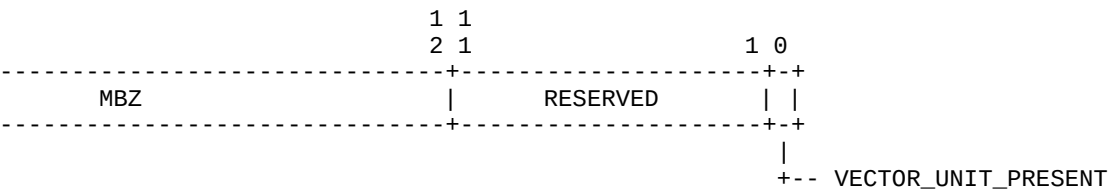
Figure 30: How to Re-enable a Tag Store which has been Turned Off

	RIGEL CPU P-CACHE OFF	RIGEL CPU P-CACHE ON
^ENABLE_BTS, ^ENABLE_PTS	Everything is off: Flush Backup T/S. Flush Primary T/S. Write ENABLE_BTS, ENABLE_PTS. Flush and turn on P-cache.	ILLEGAL if the I_BUS is being used Primary T/S must be on if the P-cache is on. If the I_BUS is not being used, take the actions in the box below.
^ENABLE_BTS, ENABLE_PTS	Backup T/S and P-cache are off: Flush Backup T/S. Flush Primary T/S. Write ENABLE_BTS. Flush and turn on P-cache.	Backup T/S is off: Flush the Backup T/S. Write ENABLE_BTS.
ENABLE_BTS, ENABLE_PTS	Primary T/S and P-cache are off: Flush Primary T/S. Write ENABLE_PTS. Flush and turn on P-cache.	ILLEGAL if the I_BUS is being used Primary T/S must be on if the P-cache is on. If the I_BUS is not being used, take the actions in the box above.
ENABLE_BTS, ENABLE_PTS	Primary T/S is on and P-cache is off: Flush Primary T/S. Flush and turn on P-cache.	NORMAL STATE.

6.3.5.1 Vector Interface Error Status Register (VINTSR) - EPR 123

The Vector Interface Error Status Register (VINTSR) contains error, status and control information relevant to the vector interface and vector unit. Since a vector processor is NOT present on the KA43 must of information contained in the VINTSR in meaningless. The VECTOR_UNIT_PRESENT (VINTSR<0>) bit is the only bit that will be discussed in this specification.

Figure 31: Vector Interface Error Status Register (VINTSR) - (EPR 123(DEC) 7B(HEX))



VINTSR<31:12> (MBZ) Read as ZERO.

VINTSR<11:1> (RESERVED) These bits contains error, status and control information that is only relevant if a vector processor were present. On the KA43 these are reserved.

VINTSR<0> VECTOR_UNIT_PRESENT Read Only. This bit should always read as ZERO, indicating that NO vector processor is present.

6.3.5.2 Cache Initialization

After ROM diagnostics are run, the backup cache tag store and primary tag store must be initialized. This can be done by:

1. Initializing the Second Level Cache Tag Storage (2100.0000-2101.FFFF)
2. Write 128 consecutive KB of data to high memory
3. Read 128 consecutive KB of low memory

6.3.5.3 Diagnostics

The tag stores and the backup cache data RAMs may be tested by reading and writing cache tags at Memory Addresses 2100.0000-2101.FFFF.

CHAPTER 7

KA43 SYSTEM ERROR STATUS REGISTER

This chapter describes the System Error Status Register (SESR). The SESR resides at address 2110.0000-2110.0007(hex).

Figure 32: System Error Status Register(SESR)

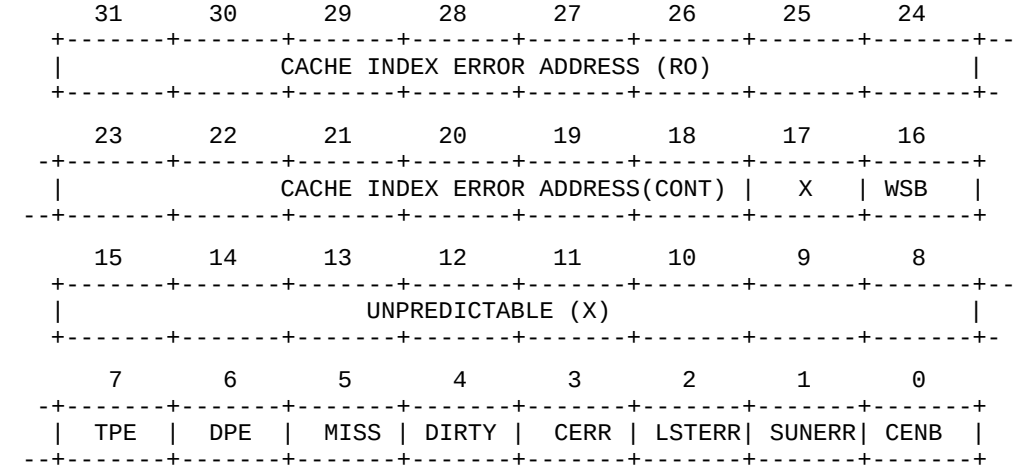


Table 22: System Error Status Register

Bit	Name	Meaning	Access
31:18	CIEA	Cache Index Error Address	Read Only
17	Unused	Unpredictable (X)	
16	WSB	Write Status Bit	Read Only
15:8	Unused	Unpredictable (X)	
7	TPE	Tag or Parity Error	Read Only
6	DPE	Dirty or Parity Error	Read Only
5	MISS	Miss	Read Only
4	DIRTY	Dirty	Read Only
3	CERR	Cache Error	Read/Write (Write 1 to Clear)
2	LSTERR	Lost Error	Read/Write (Write 1 to Clear)
1	SUNERR	Sunset Error	Read/Write (Write 1 to Clear)

Table 22 (Cont.): System Error Status Register

Bit	Name	Meaning	Access
0	CENB	Cache Enable	Read/Write

CHAPTER 8

KA43 MISCELLANEOUS I/O REGISTERS

This chapter describes several miscellaneous I/O registers. Their names and physical addresses are listed in the table below.

Table 23: Miscellaneous I/O Registers

Address	Name	Description
2002.0000	IORESET	I/O Reset register. Used to generate a reset signal to certain I/O controllers.
2002.0000	CFGTST	Configuration and Test register. Indicates which options are present in a system.
2008.0000	HLTCOD	Halt Code register. Used as a temporary storage location by system firmware during a processor restart.
2008.0010	DIAGDISP	Diagnostic Display register. Controls diagnostic display LEDs on the system board.
2008.0014	PAR_CTL	Parity control register. Controls enabling/ disabling of parity checking by the RIGEL cpu chip and by the Network Controller during a DMA cycle, and controls whether LANCE DMA cycles occur to Low or High Memory.
2008.001E	DIAGTIME	Diagnostic Timer register. A one millisecond time counter.

8.1 IO Reset Register (IORESET)

The IO reset register (IORESET) is a write-only byte register at physical address 2002.0000. Any write access to this register (the data value is ignored) generates a reset signal to the following:

1. network controller (chapter Chapter 13);
2. video option connector (chapter Chapter 15).

The indicated chapters should be consulted for details of the effects of writing to IORESET. Note that the processor, FPA, interrupt controller, and serial line controller are *not* affected by IORESET.

The minimum duration of the reset signal is 700 nsec.

8.2 Configuration and Test Register (CFGTST)

The configuration and test register (CFGTST) is a 32-bit read-only register at physical address 2002.0000.

Figure 33: Configuration and Test Register(CFGTST)

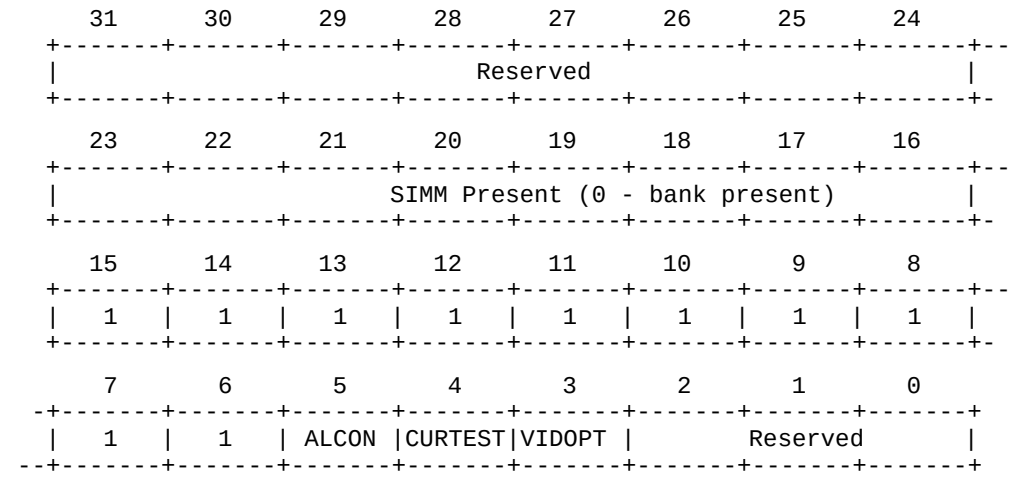


Table 24: Configuration and Test Register

Bits	Meaning
<31:24>	Reserved.
<23:16>	SIMM Present Field. Bit is 0 if SIMM is installed, 1 otherwise.
<15:6>	Reserved. Read as one.
MULTU	Video Present (bit 7). This bit is set by jumper W12 on the system board to indicate whether the system board has the monochrome video present (populated). A zero indicates the video controller is there. A one indicates that it has been de-populated.
CACHPR	Cache Present (bit 6). This bit is a zero if the cache is not present on the system board. (Depopulated). It is a one if the cache is present. It is set by jumper W11 on the system board.
ALCON	Alternate Console (bit 5). When a special jumper is installed in J15 on the KA43 System Board, allows line 3, the printer port, to be used as the diagnostic console and this bit is reported as a one. When the jumper is not installed, this bit is reported as a zero.
CURTEST	Cursor Test (bit 4). This bit is the complement of the TEST pin output from the monochrome video cursor chip.
VIDOPT	Video Option Present (bit 3). This bit is one when a Scanproc color video board is present in the video option board connector.
<2:0>	Reserved.

WARNING

The CFGTST register shares its physical address with the IORESET register (section 8.1, above). Programs must take care not to attempt to write to the CFGTST register, since this will generate an IO reset signal.

8.3 Halt Code Register (HLTCOD)

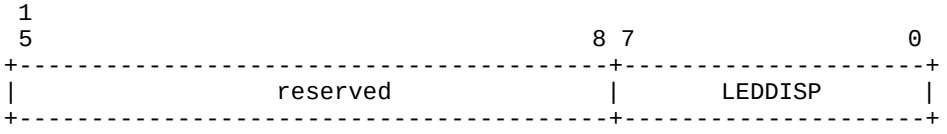
The halt code register (HLTCOD) is a read/write longword register at physical address 2008.0000. It is intended for use by the ROM-resident program which handles a processor restart.

Figure 34: Halt Code Register(HLTCOD)

8.4 Diagnostic Display Register (DIAGDISP)

The diagnostic display register is a 16-bit write-only register at physical address 2008.0010. Its low-order 8 bits control the eight diagnostic display LEDs on the system board. This display is used by the system firmware for diagnostic displays during system test and bootstrap. Upon power-on this register is cleared to zero, which illuminates all the LEDs. The register is not affected by an I/O reset.

Figure 35: Diagnostic Display Register(DIAGDISP)



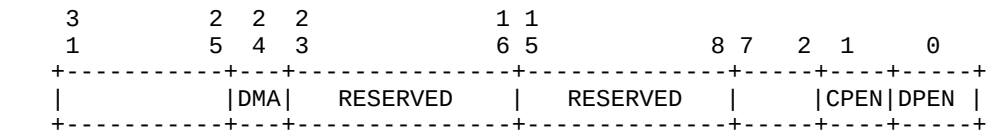
<15:8> Reserved. Must be written as zeros.

LEDDISP Led Display (bits 7:0). Each bit controls one LED: zero lights an LED and one extinguishes it. Bit 0 corresponds to the rightmost of the eight LEDs when viewed, with the cpu module mounted in the system enclosure, from the right side.

8.5 Parity Control Register (PAR_CTL)

The parity control register PAR_CTL is an 32-bit read-write register at physical address 2008.0014. It controls whether the CVAX cpu does main memory parity checking, controls whether the network controller checks memory parity during DMA read cycles, and it controls to which part of memory (High or Low) LANCE DMA cycles occur. The register is cleared to zero upon power-on.

Figure 36: System Parity Register (PAR_CTL)



<7:2> Reserved. Return UNPREDICTABLE data upon reading. Must be written with zeros.

DMA LANCE DMA Control (bit 24).

CPEN CPU Parity enable (bit 1). When this bit is one, the CVAX cpu chip checks the parity of main memory and second level cache data; when the bit is zero, parity is not checked. Cleared to zero upon power-on.

DPEN DMA Parity enable (bit 0). When this bit is one, the DC7098 checks the parity of data read from main memory during DMA cycles; when the bit is zero, parity is not checked. Cleared to zero upon power-on. (The effect of parity errors encountered while DPEN is one is discussed below in section 13.5.)

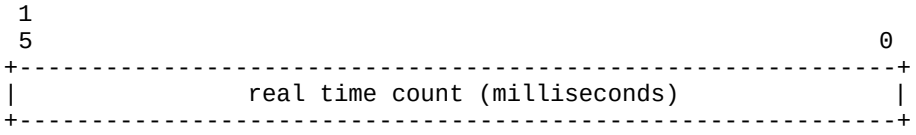
8.6 Diagnostic Timer Register (DIAGTIME)

The diagnostic timer register is a 16-bit register at physical address 2008.001E which can be used to measure real time intervals. Its contents are a 16-bit unsigned binary integer whose value is incremented by one every 10.01 millisecond. When it has reached its maximum value of 662 seconds it wraps to 0.000 seconds at the next increment. The contents of this register can be read by any word access instruction at any time.

Any write operation to the DIAGTIME register clears its contents to zero (the data supplied by the write instruction is ignored). Since the incrementing of this timer is synchronous with the system clock, the time between clearing it with a write instruction and its first increment is uncertain; it can vary between zero and a count of one.

The DIAGTIME register is cleared to zero upon power-on. It is not affected by an I/O reset.

Figure 37: Diagnostic Timer Register(DIAGTIME)



CHAPTER 9

KA43 INTERRUPT CONTROLLER

The interrupt controller receives eight interrupt request signals from the system's I/O devices, latches them, and presents a single interrupt request to the CPU at interrupt priority level 14 (hex). The controller contains three 8-bit registers:

Table 25: Interrupt Controller Registers

Name	Use
INT_REQ	holds the latched interrupt requests received from I/O devices (read-only).
INT_MSK	contains a mask which determines which interrupt requests will generate a processor interrupt (read/write).
INT_CLR	enables a program to selectively reset interrupt request bits in the INT_REQ register (write-only).

The bits in each of these registers are uniformly associated with interrupt sources according to the numbering given in section 9.1. For example, bit 0 in each register is associated with the disk controller interrupt.

The eight latches which comprise the INT_REQ register are edge-triggered: each latch records an interrupt request from an I/O device when the device's interrupt request signal changes from false to true. The contents of the INT_REQ register are ANDed with those of the INT_MSK register and if the result is non-zero, an interrupt request is presented to the CPU at IPL 14 (hex).

When the CPU acknowledges an interrupt request, the interrupt controller sends to it the interrupt vector associated with the highest-numbered bit which is set in both the INT_REQ and INT_MSK registers, and clears that bit in the INT_REQ register (the INT_MSK bit is not affected). The interrupt vector values are listed in section 9.6.

A bit in the INT_REQ register can also be cleared by a writing a byte to the INT_CLR register which has a one in the corresponding bit position. (This is a transient operation; the data written is not stored and does not prevent the INT_REQ bit from being set in the future.) This enables a program to handle a device in non-interrupt mode by clearing the device's INT_MSK bit, polling its requests by reading the INT_REQ register, and clearing its requests by writing to the INT_CLR register.

Since the bits of the INT_REQ register are edge sensitive rather than level sensitive, a program which responds to a request from a device (either in an interrupt service routine or by polling the INT_REQ register) and clears its bit in INT_REQ must do whatever is necessary to return that device's interrupt request signal to its false state in order that a subsequent request from the device will set its bit in INT_REQ again.

Upon power-on, all bits in the INT_REQ and INT_MSK registers are cleared.

9.1 Interrupt Sources and Ranking

The eight interrupt sources are listed in the following table. The interrupt numbers 7:0 indicate their bit positions in the INT_xxx registers and their relative priority when more than one request is pending; 7 is the highest priority. The edge column indicates which transition of the interrupt signal sets the device's bit in the INT_REQ register (the opposite transition has no effect).

Interrupts 5, 6 and 7 are dedicated to devices on the system board. Interrupts 0, 1, 2, and 4 come from devices attached to option board connectors. Interupt 3 comes from either the system board or from the video option connector, according to the setting of the VDC_SEL register.

Table 26: Interrupt Signal Sources

Number	Name	Edge	Source
7	SR	Up	Serial line receiver done or silo full
6	ST	Up	Serial line transmitter done
5	NI	Down	Network controller
4	NS	Down	Video option connector
3	VF	Down	Monochrome video end-of-frame or video option connector, according to the VDCSEL register
2	VS	Down	Video option connector
1	SC	Up	SCSI controller

Chapter 15 describes how the SC, DC, VF, VS and NS interrupts are used by various system options.

9.2 Video Interrupt Select Register (VDC_SEL)

The source of the video end-of-frame interrupt signal (number 3 in the table above) is determined by the VDC_SEL register, which is a one-byte read/write register at physical address 2008.000E.

Figure 38: Video Interrupt Select Register(VDC_SEL)

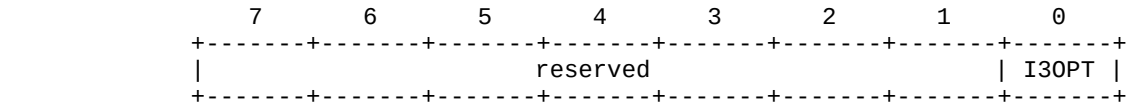


Table 27: Video Interrupt Select Register Bit Assignments

Name	Meaning
<7:1>	Reserved (bits 7:1). Return UNPREDICTABLE data when read; must be written as zeros.
I3OPT	Interrupt 3 source (bit 0). If this bit is zero, interrupt 3 comes from the system board monochrome controller (described in chapter 12). If bit 0 is one, the interrupt comes from the controller in the video option position. Upon power-on, this bit is cleared to zero.

9.3 Interrupt Request Register (INT_REQ)

The interrupt request register is an 8-bit read-only register at physical address 2008.000F each of whose bits reflects the state of the interrupt request latch for one interrupt source. Bits 7:0 correspond to interrupt numbers 7:0 as listed section 9.1.

Figure 39: Interrupt Request Register(INT_REQ)

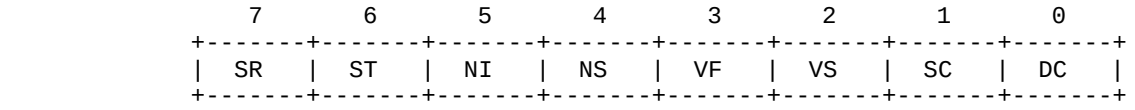


Table 28: Interrupt Request Register Bit Assignments

Name	Meaning
SR	Serial line receiver or silo full
ST	Serial line transmitter done
NI	Network controller
NS	Network controller secondary
VF	Video end of frame
VS	Video option secondary
SC	SCSI controller

A bit in the INT_REQ register is set only by an active transition on the corresponding device's interrupt request line (the electrical polarity of the active transition for each source is given in section 9.1). The bit will be set by an active transition regardless of the state of the corresponding bit in the interrupt mask register INT_MSK. However, an interrupt request is sent to the CPU only when the corresponding bits in both INT_REQ and INT_MSK are set.

A bit in the INT_REQ register is cleared either by a program which writes to the INT_CLR register with a one in the corresponding bit position, or by a CPU interrupt acknowledge cycle during which the bit is the highest-numbered bit in INT_REQ which is set and whose corresponding bit in the interrupt mask register INT_MSK is also set.

INT_REQ may be read at any time; reading it does not alter the state of the system in any way.

Upon power-on, the interrupt request register is cleared to zero.

9.4 Interrupt Clear Register (INT_CLR)

The interrupt clear register is an 8-bit write-only register at physical address 2008.000F which is used to selectively clear bits in the interrupt request register INT_REQ. Bits 7:0 correspond to interrupt numbers 7:0 as listed in section 9.1.

Figure 40: Interrupt Clear Register(INT_CLR)

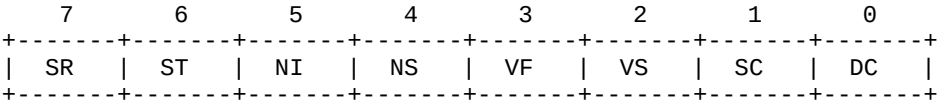


Table 29: Interrupt Clear Register Bit Assignments

Name	Meaning
SR	Serial line receiver or silo full
ST	Serial line transmitter done
NI	Network controller
NS	Network controller secondary
VF	Video end of frame
VS	Video option secondary
SC	SCSI controller

For each bit of INT_CLR which is a one, the corresponding bit of INT_REQ is cleared. For each bit of INT_CLR which is a zero, the corresponding bit of INT_REQ is not changed. The effect of writing to INT_CLR is transient; its contents are not stored and writing to it does not prevent any INTREQ bits from being set in the future.

WARNING

Use of a Read/Modify/Write instruction when accessing the INT_CLR register, could cause loss of interrupts.

9.5 Interrupt Mask Register (INT_MSK)

The interrupt mask register is an 8-bit read/write register at physical address 2008.000C each of whose bits is a mask for one interrupt source. Bits 7:0 correspond to interrupt numbers 7:0 as listed in section 9.1. Each mask bit is ANDed with the corresponding bit of the INT_REQ register before input to the priority encoder and the CPU interrupt request logic. If a mask bit is zero, the corresponding device's latched request is not presented to the CPU.

Figure 41: Interrupt Mask Register(INT_MSK)

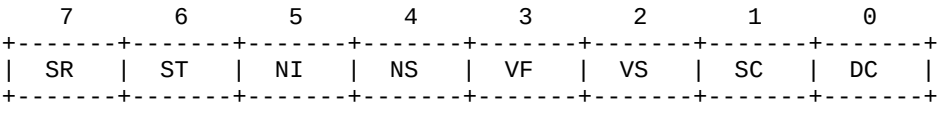


Table 30: Interrupt Mask Register Bit Assignments

Name	Meaning
SR	Serial line receiver or silo full
ST	Serial line transmitter done
NI	Network controller
NS	Network controller secondary
VF	Video end of frame
VS	Video option secondary
SC	SCSI controller

Note that a zero in a mask register bit does not prevent the corresponding device from setting its interrupt request register bit. If a request bit is set whose the corresponding mask bit is zero, a CPU interrupt is not requested until the mask bit is subsequently set to one (assuming that the request bit has not meanwhile been cleared by writing to INT_CLR). A program which is changing from polled to interrupt servicing of a device should be sure to clear the device's bit in INT_REQ prior to setting its bit in INT_MSK in order to avoid a possible false interrupt signal to the CPU.

Upon power-on, the interrupt mask register is cleared to zero.

9.6 Interrupt Vector Generation

When the CPU acknowledges an interrupt from the interrupt controller, the interrupt controller places a vector number on the bus which corresponds to the highest priority pending interrupt whose mask bit is not zero. It obtains this vector number from one of eight longwords in the system board ROM. The longword physical address is calculated from the interrupt number (which is in the range 0 to 7 as shown in section 9.1) by the following formula:

ROM address = 2004.0020 + (interrupt number * 4)

The format of an interrupt vector longword in the ROM is:

Figure 42: Interrupt Vector Longword

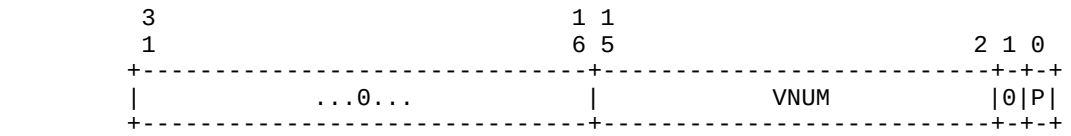


Table 31: Interrupt Vector Longword Bit Assignments

Name	Meaning
<31:16>	Ignored; should be zero.
VNUM	Interrupt vector number which is multiplied by 4 to form an offset to a vector position in the current SCB. Since only vectors in the range 200h through 3FCh should be used for I/O devices, bits 15:10 should always be zero and bit 9 should always be one.
<1>	Must be zero.
P	Priority level flag which selects the IPL to which the processor is raised when it acknowledges the interrupt. If this bit is zero, the IPL will be 14 (hex) ; if it is one, the IPL will be 17 (hex).

The conventional vector values established by the system ROM firmware for the eight devices are as follows (the value in the Vect column represents bits 15:0 of the longword; the value in the P column is then placed in bit 0 of the longword):

Table 32: Interrupt Vectors

Number	Name	Vector	P	Source
7	SR	02C0	0	Serial line controller receiver done or silo full
6	ST	02C4	0	Serial line controller transmitter done
5	NI	0250	0	Network controller
4	NS	0254	0	network controller secondary
3	VF	0244	0	Video end-of-frame (system board or video option connector, according to the VDCSEL register)
2	VS	0248	0	Video controller secondary (video option connector)
1	SC	03F8	0	SCSI controller

CHAPTER 10

KA43 TIME OF YEAR CLOCK

The time of year clock consists of an MC146818 CMOS watch chip which keeps the date and time of day and contains 50 bytes of general purpose RAM storage, and a 32.768 KHz time base oscillator. A rechargable battery powers the chip and oscillator while system power is off.

It is expected that data from the watch chip will be used by an operating system during its startup to determine the date and time, which the operating system will thereafter maintain by using interrupts from the interval timer. Therefore the watch chip's alarm and periodic interrupt features are not used in this system and the watch chip cannot generate a processor interrupt.

10.1 Battery Backup

A nickel-cadmium battery in the system box supplies power to the watch chip and its time base oscillator while system power is off. When starting from a fully charged condition, the battery will maintain valid time and RAM data in the watch chip for a minimum of 100 hours. The battery is automatically recharged while system power is on.

10.2 Watch Chip Registers

The watch chip contains 64 8-bit registers. Ten of these contain date and time data, 4 are control and status registers, and the remaining 50 provide general purpose RAM storage. The registers occupy 64 consecutive longwords of address space as shown in the table below.

Each register is accessed as bits 9:2 of a longword (bits 31:10 and 1:0 are ignored on writing and undefined on reading).

WARNING

Because each register spans two bytes on the system bus, only *word* or *longword* access instructions may be used to manipulate these registers. The effects of using byte access instructions are undefined. In particular, instructions for modifying bits such as BBSS, BBSC, BBCC and BBCS cannot be used–they generate byte-access read-modify-write cycles which will corrupt the portion of the register which is not in the byte being accessed.

Table 33: Watch Chip Register Addresses

Address	Name	Description
200B.0000	WAT_SEC	Time seconds, 0..59
200B.0004	WAT_ALMS	Alarm seconds (not used)
200B.0008	WAT_MIN	Time minutes, 0..59
200B.000C	WAT_ALMM	Alarm minutes (not used)
200B.0010	WAT_HOUR	Time hours, 0..23
200B.0014	WAT_ALMH	Alarm hours (not used)
200B.0018	WAT_DOW	Day of week, 1..7
200B.001C	WAT_DAY	Day of month, 1..31
200B.0020	WAT_MON	Month of year, 1..12
200B.0024	WAT_YEAR	Year of century, 0..99
200B.0028	WAT_CSRA	Time base divisor
200B.002C	WAT_CSRB	Date mode and format
200B.0030	WAT_CSRC	Interrupt flags (not used)
200B.0034	WAT_CSRD	Valid RAM and time flag
200B.0038		First byte of RAM data
		.....
200B.00FC		Last byte of RAM data

10.3 Control and Status Registers

Figure 43: Watch Time Base Divisor(WAT_CSRA)

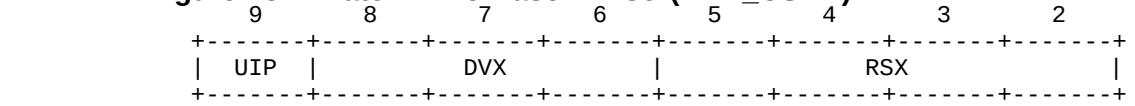


Table 34: Watch Time Register Bit Assignments

Bit	Meaning
UIP	Update in progress (bit 9). This read-only bit indicates when the date and time registers are being updated and are hence unstable. It is set to one 244 microseconds before the beginning of an update cycle and remains one until the cycle is complete.
DVX	Time base divisor (bits 8:6). These read/write bits set the amount by which the time base oscillator input to the watch chip is divided. These bits must be set to "010" to accomodate the 32.768 KHz time base in this system.
RSX	Rate select (bits 5:2). These read/write bits select the rate at which the watch chip generates periodic interrupts. Since this feature is not used, these bits must be set to "0000" to disable it.

Figure 44: Watch Date Mode and Format(WAT_CSRB)

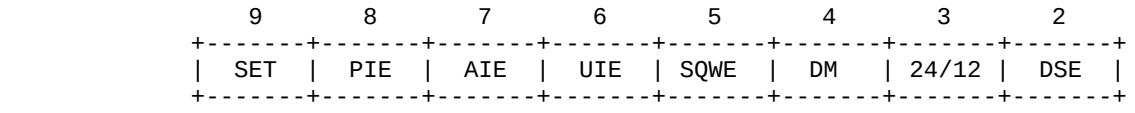


Table 35: Watch Date Mode Bit Definitions

Bit	Meaning
SET	Set Time (bit 9).
	When this read/write bit is zero, the time and date registers are updated once per second. When this bit is one, any update cycle in progress is aborted and updates are inhibited so that a program can set new date and time values.
PIE	Periodic Interrupt Enable (bit 8).
	Not used; must be set to 0.
AIE	Alarm Interrupt Enable (bit 7).
	Not used; must be set to 0.
UIE	Update Interrupt Enable (bit 6).
	Not used; must be set to 0.
SQWE	Square-wave Enable (bit 5).
	Not used; must be set to 0.
DM	Data Mode (bit 4).
	This read/write bit selects the numeric representation in the time and date registers. If DM is one, the data format is binary; if DM is zero, the data format is two 4-bit decimal digits (BCD).
24/12	Hours Format (bit 3).
	This read/write bit selects the format of the WAT_HOUR AND WAT_ALMH registers. A value of one selects 24-hour mode; a value of zero selects 12-hour AM/PM mode. In the latter case, bit 7 of the hours registers is zero for AM and one for PM.
DSE	Daylight Saving Enable (bit 2).
	This read/write bit is zero for normal operation. If set to one, two special time updates occur: on the last Sunday in April the time increments from 01:59:59 AM to 03:00:00 AM, and on the last Sunday in October when the time first reaches 01:59:59 AM it changes to 01:00:00 AM. (This feature is obsolete, since it does not conform to current United States Law.)

Figure 45: Watch Valid RAM and Time Flag(WATCSRD)

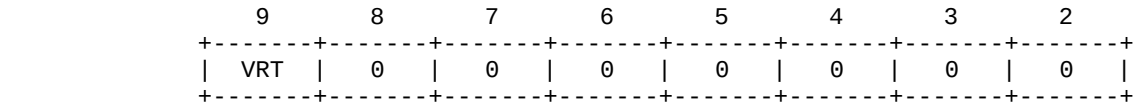


Table 36: Watch Valid RAM and Time Flag Bit Assignments

Bit	Meaning
VRT	Valid RAM and Time (bit 9). This bit indicates whether the contents of the time and RAM registers may have been corrupted by loss of power. This bit is set to zero whenever system power is off and the backup battery voltage drops below the value required for the watch chip to function properly. This bit is set to one after any read of this register (the register may not be written). Programming note: Since ANY read of this register to test the VRT bit will reset that bit to one, a program which finds VRT equal to zero must be prepared to either load valid data into the time and RAM registers or take other action to indicate that the contents of the watch chip are invalid.
<8:2>	Not used. Always read as zeros.

10.4 Time of Year Data

The time of year is kept in six registers: WAT_SEC, WAT_MIN, WAT_HOUR, WAT_DAY, WAT_MON, and WAT_YEAR. A seventh register, WAT_DOW, indicates the day of the week (days are numbered from 1 (Sunday) through 7). The contents of each register may be in either binary form or BCD (two 4-bit decimal digits) as selected by register WAT_CSRB bit DM.

The time value is incremented once each second. Such an update requires 1948 microseconds, during which time the date and time register contents are unstable and should not be read by a program. Register WAT_CSRA bit UIP indicates when an update is in progress. This bit is one from 244 microseconds before the beginning of an update cycle until the cycle is complete. Therefore a program should read WAT_CSRA until it finds bit UIP zero, at which time it has at least 244 microseonds to read the date and time registers. The program should inhibit interrupts while reading the registers to ensure that an interrupt does not prolong its reading beyond the 244 microsecond window.

10.5 Non-volatile RAM Storage

The 50 bytes of RAM storage are used by system firmware. The use of these bytes is defined in the firmware specification specification.

10.6 Initialization

When a program finds the VRT bit equal to zero, it must assume that the contents of all other registers in the chip are invalid. To initialize the chip, a program should:

Table 37: Non Volitile Ram Initialization

Step	Procedure
1	Load register WAT_CSRB with bit SET equal to one to inhibit time updates and bits PIE, AIE, UIE and SQWE equal to zero to disable unused features. Bits TM, 24/12 and DSE should be set for the desired date format.
2	Load the seven time registers with the current date and time.
3	Load register WAT_CSRA to set the proper time base divisor. The DVX bits should be set to "010" and the RSX bits to "0000".
4	Load register WAT_CSRB with the same value used in step 1 except that bit SET should now be zero to enable normal time updating.

As long as the backup battery voltage is sufficient, the contents and operation of the watch chip are not affected by system power-on and power-off events.

CHAPTER 11

KA43 SERIAL LINE CONTROLLER

The main board serial line controller handles four asynchronous serial lines. The heart of the controller is a DC367B gate array. Input characters from all four lines are buffered in a common 64-position silo.

Note: This controller is similar to a DZQ11 serial controller board, since both use the same gate array and silo arrangement. The operation of the receivers and transmitters is virtually identical, but the arrangement of modem signals, interrupt controls, and the register addresses are NOT the same.

11.1 Line Usage

The four serial lines are numbered 0 through 3, and each has a particular primary use, as follows:

Table 38: Serial Line Usage

Line	Device	Comments
0	Keyboard	Connected to the video monitor cable and a 4-pin Modular Jack mounted on the system board. Data leads only. Supports LK201 keyboard.
1	Pointer	Connected to the video monitor cable and a 7-pin DIN Connector mounted on the System Board. Data leads only. Supports VSXXX-AA mouse or VSXXX-AB Tablet.
2	Communications	Connected to a 6-pin Modified Modular Jack mounted on the system board, DEC423 compatible. Data leads plus ready-out (Data Terminal Ready) and ready-in (Data Set Ready).
3	Printer	Connected to a 6-pin MMJ mounted on the system board. DEC423 data leads only.

11.2 Diagnostic Terminal Connection

Line 3 is normally connected to a printer thru a BC16E cable. If S3 is in the up position on the KA42 System Board, a received break condition on this line will assert the cpu halt signal which will cause a processor restart with a restart code of 02h.

11.3 Interrupts

The controller generates two types of interrupt requests, each with a separate vector and bit in the INT_REQ and INT_MSK registers. These are receiver done or silo alarm, and transmitter done. Section 9.6 lists the vector values. In order for these interrupts to be signalled to the CPU, the appropriate bits in the interrupt mask register INT_MSK must be set (see section 9.5). Note that, unlike a DZQ11 board, there are no interrupt enable bits in the control and status register SER_CSR.

11.4 Register Summary

The controller contains six addressable registers as follows:

Table 39: Serial Line Controller Register Addresses

Address	Name	Access
200A.0000	SERCSR	read/write
200A.0004	SERRBUF	read
200A.0004	SERLPR	write
200A.0008	SERTCR	read/write
200A.000C	SERMSR	read
200A.000C	SERTDR	write

11.4.1 Control and Status Register (SER_CSR)

The Control and Status register is a 16-bit register at address 200A.0000. It must be read on a word basis but can be written on either a word or byte basis. All bits in SER_CSR are cleared to zero by power-on or by setting the master clear bit SER_CSR<CLR>.

Figure 46: Serial Line Control and Status Register(SER_CSR)

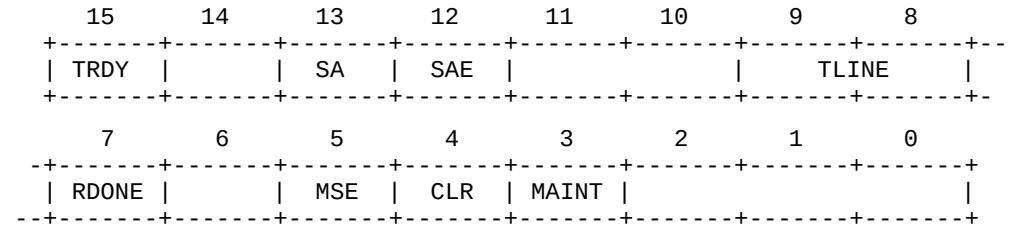


Table 40: Serial Line Control and Status Register Bit Meanings

TRDY	Transmitter Ready (bit 15). This read-only bit is set by the hardware when the transmitter scanner stops on a line whose transmitter buffer is ready to be loaded with another character and whose related transmitter control register bit SER_TCR<TXEN_x>is set. While the <TRDY> bit is one, and at no other time, the transmitter line number bits SER_CSR<TLINE> are valid. When <TRDY> changes from zero to one, the interrupt request register bit INT_REQ<ST> is set to one. If interrupt mask register bit INT_MSK<ST> is also one, a transmitter interrupt request to the CPU will be generated; otherwise SER_CSR<TRDY> can be polled by the host program. However, the interrupt request bit INT_REQ<ST> is not automatically cleared while interrupts are masked, so when changing from polled to interrupt operation there may be a spurious interrupt request to the CPU unless the host program clears INT_REQ<ST> by writing a one to INT_CLR<ST>. The <TRDY> bit is cleared when data is loaded into the transmitter for the line number indicated in SER_CSR<TLINE> by writing to register SER_TDR. If additional transmitter lines need service, <TRDY> is set again within 1.4 microseconds of the completion of the transmitter data load operation. The <TRDY> bit is also cleared when the master scan enable bit SER_CSR<MSE> is cleared, or when the related transmitter control register bit SER_TCR<TXEN_x> is cleared.
<14>	Not used.
SA	Silo Alarm (bit 13). This read-only bit is set by the hardware when 16 characters have been entered into the FIFO silo buffer. While the silo alarm enable bit SER_CSR<SAE> is one, the transition of <SA> from zero to one sets interrupt request register bit INT_REQ<SR> to one. If interrupt mask register bit INT_MSK<SR> is also one, an interrupt is signalled to the CPU; otherwise the <SA> bit may be polled. However, the interrupt request register bit INT_REQ<SR> is not automatically cleared while that interrupt is masked, so when changing from polled to interrupt operation, there may be a spurious interrupt request to the CPU unless the host program clears INT_REQ<SR> by writing a one to INT_CLR<SR>. The <SA> bit is cleared by reading the receiver buffer register SER_RBUF. When responding to a silo alarm, the host program must read characters from the silo until it is empty (i.e. until SER_RBUF<DVAL> is zero), since the silo alarm bit will not be set again until 16 additional characters have been stored in the silo. The <SA> bit is always zero while the silo alarm enable bit SER_CSR<SAE> is zero.
SAE	Silo Alarm Enable (bit 12). This read/write bit selects the source of the receive interrupt request signal which is sent to bit INT_REQ<SR> in the interrupt controller. If <SAE> is one, the silo alarm bit <SA> discussed above is used as the signal; if <SAE> is zero, the receiver done bit <RDONE> discussed below is used instead. Note that while <SAE> is zero, <SA> is also forced to be zero.
<11:10>	Not used.
TLINE	Transmitter Line Number (bits 9:8). These read-only bits indicate the number of the line whose transmitter buffer needs servicing (bit 8 is the least significant bit). These bits are valid only while the transmitter ready bit SER_CSR<TRDY> is one. These bits are cleared when the master scan enable bit SER_CSR<MSE> is cleared.
RDONE	Receiver Done (bit 7).

Table 40 (Cont.): Serial Line Control and Status Register Bit Meanings

	This read-only bit is set by the hardware when an incoming character appears at the output of the silo buffer. While the silo alarm enable bit SER_CSR<SAE> is zero, the transition of <RDONE> from zero to one sets interrupt request register bit INT_REQ<SR> to one. If interrupt mask register bit INT_MSK<SR> is also one, an interrupt is signalled to the CPU; otherwise the <RDONE> bit may be polled. However, the interrupt request register bit INT_REQ<SR> is not automatically cleared while that interrupt is masked, so when changing from polled to interrupt operation, there may be a spurious interrupt request to the CPU unless the host program clears INT_REQ<SR> by writing a one to INT_CLR<SR>. <RDONE> is cleared when the receiver buffer register SER_RBUF is read. If another character is available in the silo, <RDONE> will be set again after a delay of between 0.1 and 1.0 microsecond. This bit is also cleared when the master scan enable bit SER_CSR<MSE> is cleared.
<6>	Not used.
MSE	Master Scan Enable (bit 5). This read/write bit must be set to one to permit the receiver and transmitter control sections to scan the lines to service them. When this bit is zero, the transmitter ready bit SER_CSR<TRDY> is cleared and the receiver silo is cleared.
CLR	Master Clear (bit 4). When this bit is set by a program, the hardware performs an internal initialization process. At the conclusion of this process the hardware clears this bit. If a program reads this bit as one, then the internal process is not complete. This initialization clears all registers, the silo, and all UARTs with the following exceptions: 1. In the receiver buffer register only bit SER_RBUF<DVAL> is cleared; the remaining bits are not. 2. Bits <15:8> of the transmitter control register SER_TCR (the modem control outputs) are not cleared. 3. The modem status register SER_MSR is not cleared.
MAINT	Maintenance (bit 3). This read/write bit, when set, loops the serial output connections of the transmitters to the corresponding serial input connections of the receivers. This feature is intended for hardware diagnostic use.
<2:0>	Not used.

NOTE

After setting the master clear bit SER_CSR<CLR>, a program must repeatedly read SER_CSR until it finds SER_CSR<CLR> equal to zero before attempting any other operations with the serial line controller. Neither of the interrupt controller registers INT_REQ and INT_MSK are altered when <CLR> is set. The host program must clear bits <SR> and <ST> of INT_MSK to zero and must clear those bits in INT_REQ by writing ones to the corresponding bits of INT_CLR to complete the initialization process.

11.4.2 Receiver Buffer Register (SER_RBUF)

The Receiver Buffer register is a 16-bit read-only register at address 200A.0004 which must be read as a word. It contains the received character at the bottom of the silo buffer (that is, the oldest character in the silo). Reading this register removes the character from the silo buffer, and all the other characters in the silo then shift down to the lowest unoccupied location. When this register is read (or when the master clear bit SER_CSR<CLR> is set or

after a power-on reset), the data valid bit SER_RBUF<DVAL> is cleared and the remaining bits of the register (although not cleared) are invalid.

Figure 47: Serial Line Receiver Buffer Register(SER_RBUF)

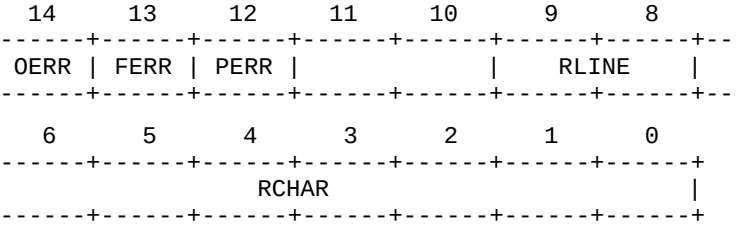


Table 41: Serial Line Receiver Buffer Register Bit Assignments

Name	Meaning
DVAL	Data Valid (bit 15). This bit, when one, indicates that the data in bits <14:0> of the register is valid. This permits an interrupt handling program to read the receiver buffer register repeatedly and store each character until this bit is read as zero, which indicates that the silo is empty.
OERR	Overrun Error (bit 14). This bit is one when a received character is overwritten in a UART buffer by a following character before the first character was transferred to the silo. This condition indicates that the program is not emptying the silo fast enough.
FERR	Framing Error (bit 13). This bit is one if the received character did not have a stop bit present at the correct time. The combination of <FERR> set and <RCHAR> entirely zero is usually interpreted as indicating that a BREAK has been received. The receipt of a framing error on line 3 (the printer port) is a special case. If the hardware detects a framing error on line 3 and the accompanying character contains all zeros (i.e. a BREAK has been received), the line controller hardware asserts a signal whose effect is described under Diagnostic Terminal Connection (section 11.2).
PERR	Parity Error (bit 12). This bit is one if the sense of the parity of the accompanying character does not agree with the parity which was defined for the line when its line parameter register SER_LPR was last loaded.
<11:10>	Not used.
RLINE	Receiver Line Number (bits 9:8). These bits indicate the number of the line from which the character was received (bit 8 is the least significant bit).
RCHAR	Received Character (bits 7:0). Characters with a width of fewer than 8 bits (as defined when the line's line parameter register was last loaded) are right justified with the unused bit positions cleared. The parity bit is not included in the received character.

11.4.3 Line Parameter Register (SER_LPR)

The Line Parameter register is a 16-bit register at address 200A.0004 which controls the operating parameters of each line. This register is write-only and must be written as a 16-bit word. The parameters for each line must be reloaded after each power-on reset or setting of the master clear bit SER_CSR<CLR>. The operating parameters should not be modified for a line while data transmission or reception is in progress on that line.

Figure 48: Serial Line Parameter Register(SER_LPR)

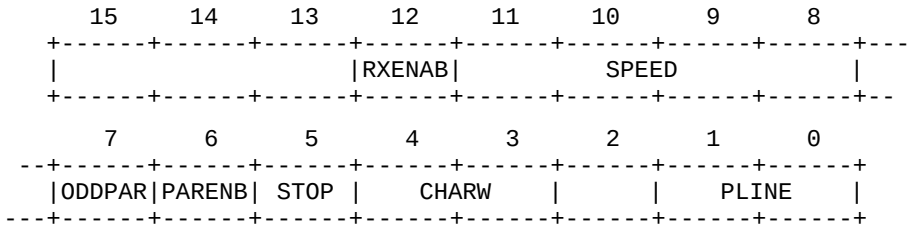


Table 42: Serial Line Parameter Register Bit Meanings

Bit	Meaning
<15:13>	Not used.
RXENAB	Receiver Enable (bit 12). This bit must be set in order for the UART for this line to receive bits and assemble them into characters.
SPEED	Speed Code (bits 11:8). These bits select the data bit rate for the receiver and transmitter for the line. The bits are described in the table which follows this one.

<LINEART> 11 10 9 8 Data rate (bits/second) ————————- 0 0 0 0 50 0 0 0 1 75 0 0 1 0 110 0 0 1 1 134.5 0 1 0 0 150 0 1 0 1 300 0 1 1 0 600 0 1 1 1 1200 1 0 0 0 1800 1 0 0 1 2000 1 0 1 0 2400 1 0 1 1 3600 1 1 0 0 4800 1 1 0 1 7200 1 1 1 0 9600 1 1 1 1 19200 (nominal; actual rate is 19800)

ODDPAR	Odd Parity (bit 7). If this bit is set and the parity enable bit SER_LPR<PARENB> is also set, then characters with odd parity are transmitted to the line and characters received from the line are expected to have odd parity. If this bit is clear and the parity enable bit SER_LPR<PARENB> is set, then characters with even parity are transmitted to the line and characters received from the line are expected to have even parity. If the parity enable bit SER_LPR<PARENB> is clear, then the setting of this bit is immaterial.
PARENB	Parity Enable (bit 6). If this bit is set, characters transmitted to the line have a parity bit appended and characters received from the line have their parity checked. The sense of the parity is according to the setting of the odd parity bit SER_LPR<ODDPAR>.
STOP	Stop Code (bit 5). If this bit is clear, the stop code following the last transmitted bit is one bit time long. If this bit is set, the stop code lasts 1.5 bit times for characters whose width is 5 bits and 2 bit times for characters whose width is 6, 7 or 8 bits.
CHARW	Character Width (bits 4:3). These bits control the number of data bits (exclusive of any parity bit) in the characters transmitted and expected in the characters received. The encoding is:

```
 4   3  Character width (bits)
--  --  ---------------
 0   0     5
 0   1     6
 1   0     7
 1   1     8
```

<2>	Not used.
PLINE	Parameter Line Number (bits 1:0). These bits specify the number of the line to which the parameters in the rest of the register apply. Bit 0 is the least significant bit.

11.4.4 Transmitter Control Register (SER_TCR)

The Transmitter Control register is a 16-bit register at address 200A.0008 which must be read on a word basis and can be written on either a word or byte basis.

Figure 49: Serial Line Transmitter Control Register(SER_TCR)

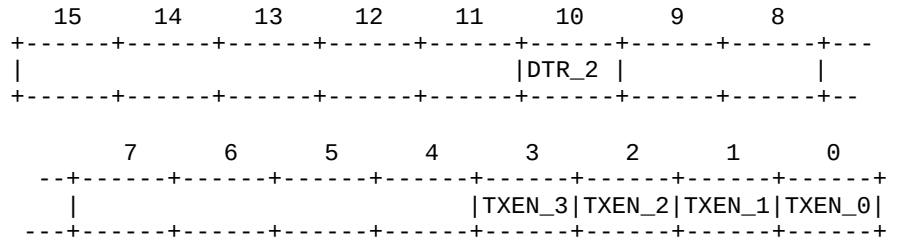


Table 43: Serial Line Transmitter Control Register Bit Assignments

Bit	Meaning
<15:11>	Not used.
DTR_2	Data Terminal Ready (bit 10). This read/write bit controls the state of the Data Terminal Ready modem control signal (CCITT circuit 108/2) for line 2. Setting the bit asserts the ON state of the DTR signal. This bit is cleared by a power-on reset; it is NOT cleared when the master clear bit SER_CSR<CLR> is set.
<9:4>	Not used.
TXEN_x	Transmitter Line Enable (bits 3:0). These read/write bits enable the transmitter logic for lines 3, 2, 1, and 0 respectively. Setting each of these bits causes the transmitter scanner to stop and assert the transmitter ready bit SER_CSR<TRDY> if the UART for that line has a transmitter buffer empty condition. The transmitter scanner resumes scanning when either the transmitter data register for the line at which the scanner stopped is loaded with another character, or when that line's transmitter line enable bit is cleared. A transmitter line enable bit should only be cleared while the scanner is not running (i.e. when the transmitter ready bit SER_CSR<TRDY> is set or the master scan enable bit SER_CSR<MSE> is clear). The transmitter line enable bits are cleared by a power-on reset and whenever the master clear bit SER_CSR<CLR> is set.

11.4.5 Modem Status Register (SER_MSR)

The Modem Status register is a 16-bit read-only register at address 200A.000C which contains the status of modem input signals for line 2. The ON condition of a modem signal is presented as the set state of the corresponding bit.

Figure 50: Serial Line Modem Status Register(SER_MSR)

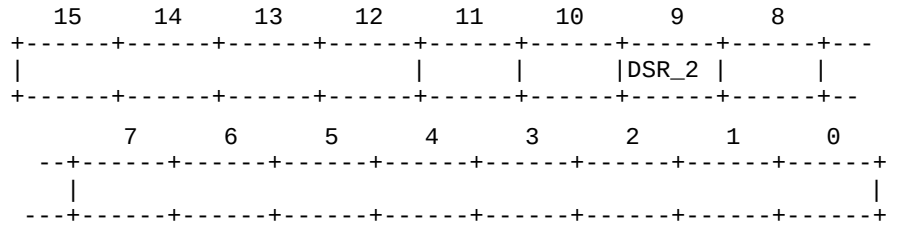


Table 44: Serial Line Modem Status Register Bit Assignments

Bit	Meaning
<15:10>	Not used; read values undefined.
DSR_2	Data Set Ready (bit 9). This bit reflects the state of the Data Set Ready signal from an external modem (CCITT circuit 107) on line 2. The set state corresponds to the ON state of the signal.
<8:0>	Not used; read values undefined.

11.4.6 Transmitter Data Register (SER_TDR)

The Transmitter Data register is a 16-bit write-only register at address 200A.000C. It can be written on either a word or byte basis.

Figure 51: Serial Line Transmitter Data Register(SER_TDR)

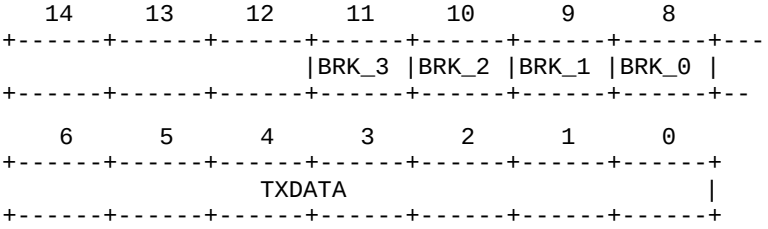


Table 45: Transmitter Data Register Bit Assignments

Bit	Meaning
<15:12>	Not used.
BRK_x	Break Control (bits 11:8). These write-only bits control the assertion of a BREAK condition on lines 3, 2, 1, and 0, respectively. Setting a bit immediately forces the transmitter output for the corresponding line to the SPACE condition. This condition will persist until the break control bit is cleared. These bits are cleared by a power-on reset and when the master clear bit SER_CSR<CLR> is set.
TXDATA	Transmitter Buffer (bits 7:0). Data to be transmitted by a line's UART is loaded into these 8 bits. If the character width is less than 8, the unused bits are at the high-order (bit 7) end of the byte. This register may be written to only while the transmitter ready bit SER_CSR<TRDY> is set. The line to which the character is sent is indicated by the transmitter line number bits SER_CSR<TLINE>.

CHAPTER 12

KA43 MONOCHROME DISPLAY CONTROLLER

The main board video display controller generates a monochrome image which is 1024 pixels wide by 864 pixels high. The controller consists of one bit-mapped display data plane and hardware which can superimpose a cursor at any position on the display independently of the contents of the data plane.

12.1 Video Timing

All video timing is derived from the pixel clock crystal whose frequency is 69.1968 MHz, which yields a pixel time of approximately 14.5 nanoseconds. The timing of all the synchronization and blanking signals is fixed; it cannot be changed by a program.

Table 46: Monochrome Video Timing

	Pixel frequency:	69.1968 MHz	
	Horizontal frequency:	54.06 KHz	
	Vertical frequency:	60.0 Hz	
	Horizontal Timing:	Microseconds	Pixels
	Entire line	18.50	1280
	Visible raster	14.80	1024
	Blanking	3.70	256
|	Sync front porch	0.11	8
	Sync pulse	1.85	128
|	Sync back porch	1.74	120
	Vertical timing:	Milliseconds	Lines
	Entire frame	16.667	901
	Visible raster	15.982	864
	Blanking	0.684	37
	Sync front porch	0.000	0
	Sync pulse	0.055	3
	Sync back porch	0.629	34

12.2 End-of-frame Interrupt

An interrupt request is generated at the trailing edge of each vertical sync pulse, which is three horizontal scan times after the beginning of each vertical blanking interval. (The interrupt vector is listed in section 9.6.) The time between this interrupt and the end of the vertical blanking interval is approximately 620 microseconds (34 line times). Interrupts occur at the frame rate of 60 Hz. Interrupts may be masked by clearing bit <VF> of the interrupt mask register INT_MSK to zero (see section 9.5). Upon power-up, this mask bit

is cleared to zero. In order for this end-of-frame signal to be recognized as an interrupt, the VDC_SEL register (section 9.2) must be set to select this source rather than the video option board.

12.3 Data Plane Storage

The display data plane is stored in a 256 Kb block of dual port RAM which the processor can manipulate just like any other RAM storage. It occupies the physical address range 3000.0000 through 3003.FFFF. A program may use any mode of access–byte, word, or longword–to this storage.

One displayed line of 1024 pixels is represented by 32 consecutive longwords, beginning at an address whose low-order 7 bits are all zero (i.e. a multiple of 128 decimal). Each longword appears as 32 consecutive pixels on a display line. Bit 0 of a longword (least significant) is displayed as the leftmost pixel and bit 31 (most significant) is displayed as the rightmost pixel of the 32-pixel group. Longword addresses increase from left to right across a displayed line and exactly 32 longwords are required for each line. The 256 Kb data plane storage holds 2048 line images, 864 of which are visible on the display at any one time.

12.4 Display Origin Register (VDC_ORG)

The address in the data plane storage which corresponds to the top line of the display raster is determined by the 8-bit read/write register VDC_ORG, whose address is 2008.000D. This register supplies bits 17:10 of the address of the top line. Thus, the address of the first longword in the topmost displayed line is:

address = 3000.0000 + (VDC_ORG * 1024)

The visible display can begin on any 8-line boundary and wraps from the last line in the data plane storage (beginning at 3003.FF80) to the first line (beginning at 3000.0000). The contents of VDC_ORG are used at the beginning of the vertical blanking interval to reset the video controller address counter. Programs may write the VDC_ORG register at any time. The contents of VDC_ORG are cleared to zero upon power-up.

Changing VDC_ORG does not affect the displayed position of the cursor sprite on the screen. The sprite's position registers operate relative to the first line displayed, regardless of what memory address it comes from.

12.5 Cursor Coordinate Offsets

The visible raster is 1024 pixels wide in the X direction and 864 lines high in the Y direction. The nominal range of cursor coordinates is 0 through 1023 (left to right) and 0 through 863 (top to bottom). Because the hardware resets the X and Y position counters in the cursor chip at some time prior to the beginning of the visible display, an offset must be added to nominal raster coordinate values before loading them into the cursor position and region limits registers. These offset values are:

X offset: 216 pixels
Y offset: 33 lines

For example, to display a sprite cursor with its upper left corner in pixel 100, line 300, a program must load CUR_XPOS with (100 + 216) and CUR_YPOS with (300 + 33).

12.6 Cursor Generation

The cursor can take two forms: a 16 by 16 bit pattern (sprite) or a crosshair whose lines may extend to the edges of the visible raster or may be clipped to a programmed region. The cursor hardware uses a DC503 programmable sprite cursor chip which generates two display planes called the A and B planes. Bits from these planes are combined with bits from the data plane according to the following equation:

Display = (Data .and. (.not. B)) .xor. A

where a ONE displays a white pixel and a ZERO displays a black pixel. In tabular form, this is:

Data	B plane	A plane	Displayed	Cursor appearance
0	0	0	Black	Invisible
0	0	1	White	Inverted data
0	1	0	Black	Black
0	1	1	White	White
1	0	0	White	Invisible
1	0	1	Black	Inverted data
1	1	0	Black	Black
1	1	1	White	White

12.7 Cursor Control Registers

The cursor chip contains the following programmable elements:

- Two 16-word arrays to store a 16 by 16 bit sprite pattern for each cursor plane.
- X and Y position registers to control where the cursor pattern is displayed in the raster.
- Two region detectors, each of which defines a rectangle in the raster which can be used to clip the display of a crosshair cursor.
- A control register which determines how the cursor is generated.

To a program the cursor chip appears as 12 write-only registers, each one word (16 bits) wide. These registers should always be written with word-access instructions; they cannot be read (hence read-modify-write instructions such as BIS cannot be used). Their contents after power-up are indeterminate. The addresses and names of the registers are:

Table 47: Monochrome Cursor Register Addresses

Address	Name	Note	Function
200F.0000	CURCMD		Cursor command register

Table 47 (Cont.): Monochrome Cursor Register Addresses

Address	Name	Note	Function
200F.0004	CURXPOS	D	Cursor X position
200F.0008	CURYPOS	D	Cursor Y position
200F.000C	CURXMIN1	D	Region 1 left edge
200F.0010	CURXMAX1	D	Region 1 right edge
200F.0014	CURYMIN1	D	Region 1 top edge
200F.0018	CURYMAX1	D	Region 1 bottom edge
200F.002C	CURXMIN2	D	Region 2 left edge
200F.0030	CURXMAX2	D	Region 2 right edge
200F.0034	CURYMIN2	D	Region 2 top edge
200F.0038	CURYMAX2	D	Region 2 bottom edge
200F.003C	CURLOAD		Cursor sprite pattern load

In order to prevent unsightly effects on the display, the registers marked "D" in the Note column are buffered, as are some of the bits in the cursor command register. The processor may write into such a register or bit at any time (except within three horizontal scan times following the beginning of vertical blanking), but the new value will take effect only at the beginning of the next vertical blanking interval. Since the processor receives its end-of-frame interrupt signal three line times after vertical blanking begins, a program may ensure that it has ample time to perform a multi-register update by waiting for the end-of-frame interrupt before starting to load new values. From the time of the interrupt, it has nearly an entire frame time (16.612 milliseconds) to load the registers.

12.8 Cursor Command Register (CUR_CMD)

The cursor command register is a 16-bit write-only register at address 200F.0000. As in the preceding list of cursor registers, the bits marked with "D" in figure 52 are buffered and do not take effect until the beginning of the next vertical blanking interval.

Figure 52: Cursor Command Register(CUR_CMD)

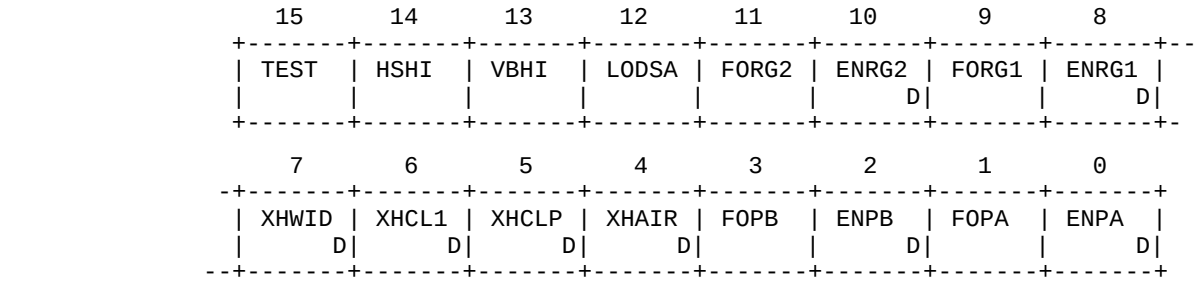


Table 48: Cursor Command Register Bit Assignments

Bit	Meaning
TEST	Diagnostic test (bit 15). This bit is used to manipulate the cursor test flipflop (described in section 12.15). When this bit is set to one, the test flipflop (as seen in the CURTEST bit of the CFGTEST register) is forced to "one". When this bit is cleared to zero, the test flipflop can be cleared to zero by any of the various test conditions. The value in TEST affects only the test flipflop; it may have either value for normal display operation.
HSHI	Horizontal sync polarity (bit 14). This bit must be one to indicate to the chip that the horizontal sync input from the video controller is active high.
VBHI	Vertical blanking polarity (bit 13). This bit must be zero to indicate to the chip that the vertical blanking input from the video controller is active low.
LODSA	Load/display sprite array (bit 12). When this bit is zero, the cursor sprite is displayed normally from the contents of the sprite arrays. When this bit is one, display of the sprite is inhibited and the sprite arrays can be loaded by successive writes to the CUR_LOAD register. Upon the transition of LODSA from one to zero, the internal array address counter is reset so that the next write to CUR_LOAD will load the top row of sprite plane A.
FORG2	Force region detector 2 output to one (bit 11). When this bit is one, the output of region detector 2 is forced to one (true). When this bit is zero, the detector operates normally.
ENRG2	Enable region detector 2 (bit 10). When this bit is zero, the output of region detector 2 is inhibited; it will be zero (false) unless the FORG2 bit is also set, which takes precedence and forces the output to one (true). When ENRG2 is one, the detector operates normally.
FORG1	Force region detector 1 output to one (bit 09). When this bit is one, the output of region detector 1 is forced to one (true). When this bit is zero, the detector operates normally.
ENRG1	Enable region detector 1 (bit 08). When this bit is zero, the output of region detector 1 is inhibited; it will be zero (false) unless the FORG1 bit is also set, which takes precedence and forces the output to one (true). When ENRG1 is one, the detector operates normally.
XHWID	Crosshair cursor line width (bit 07).

Table 48 (Cont.): Cursor Command Register Bit Assignments

Bit	Meaning
	When this bit is zero, the crosshair cursor lines are one pixel wide. When this bit is one, the lines are two pixels wide. The extra pixels are added to the right of and below the pixels which lie on the lines corresponding to the cursor X and Y positions.
XHCL1	Select crosshair clipping region (bit 06). If this bit is one, region detector 1 is used to clip the crosshair cursor; if it is zero, region detector 2 is used. This bit is effective only if the crosshair cursor is selected (bit XHAIR is one) and crosshair clipping is selected (bit XHCLP is one).
XHCLP	Clip crosshair inside region (bit 05). If this bit is one, the crosshair cursor is clipped so that it is displayed only within the region selected by the XHCL1 bit. If this bit is zero, the crosshairs extend to the edges of the displayed raster. This bit is effective only if the crosshair cursor is selected (bit XHAIR is one).
XHAIR	Crosshair/sprite cursor select (bit 04). If this bit is one, the cursor chip generates a crosshair whose lines intersect at the cursor X, Y position. If this bit is zero, the cursor chip generates the sprite pattern with its upper left corner at the cursor X, Y position.
FOPB	Force cursor plane B output to one (bit 03). When this bit is one, the output from cursor plane B is forced to one throughout the display, regardless of the settings of bits ENPB, XHAIR, XHCLP, XHCL1, XHWID, and of the contents of the sprite plane B array. When this bit is zero, the cursor is displayed normally.
ENPB	Enable cursor plane B (02). When this bit is zero, the output from cursor plane B is inhibited; it will be zero throughout the display. When this bit is one, the output from cursor plane B is displayed normally.
FOPA	Force cursor plane A output to one (bit 01). When this bit is one, the output from cursor plane A is forced to one throughout the display, regardless of the settings of bits ENPA, XHAIR, XHCLP, XHCL1, XHWID, and of the contents of the sprite plane A array. When this bit is zero, the cursor is displayed normally.
ENPA	Enable cursor plane A (bit 00). When this bit is zero, the output from cursor plane A is inhibited; it will be zero throughout the display. When this bit is one, the output from cursor plane A is displayed normally.

12.9 Loading the Cursor Sprite Pattern

The cursor sprite pattern is stored in two arrays, each of sixteen 16-bit words. Each word of an array is displayed as 16 pixels on a scan line with bit 0 (least significant) in the leftmost display position. All thirty-two words are loaded by writing to the CUR_LOAD register. An internal address counter in the chip is incremented after each write to point to the next word in the array to be loaded.

Cursor command register bit LODSA controls access to the sprite arrays. When this bit is zero, the arrays are read during normal raster scanning to display the sprite pattern. When LODSA is one, normal display of the sprite is inhibited and data can be written into the arrays. The act of changing LODSA from one to zero resets the internal array address counter. The next write to CUR_LOAD loads the top line of the A plane array; the next fifteen writes load its remaining lines. The 16th through 32nd writes load the B plane array from top to bottom. When loading is completed, cursor command register bit LODSA must be reset to zero to resume normal sprite display.

Loading the sprite arrays should be synchronized by waiting for the end-of-frame interrupt so that it is done during the vertical blanking interval. (Note: Any of the other registers of the cursor chip may be written to while the sprite arrays are being loaded; only writes to CUR_LOAD advance the address counter.)

12.10 Cursor Region Detectors

There are two region detectors, 1 and 2, each of which defines a rectangular area of the raster which can be used to clip the display of a crosshair cursor. (The detectors also emit a hardware signal which is not used by this system.) Each region detector is programmed by setting four registers: CUR_XMIN, CUR_XMAX, CUR_YMIN, and CUR_YMAX. The horizontal boundaries of a region are controlled by the CUR_X... registers and can be specified only to a four-pixel boundary: the least significant two bits of their contents are ignored and the system behaves as if those two bits were always zero. The vertical boundaries are controlled by the CUR_Y... registers and can be specified to any line boundary. The offsets described above under Cursor Coordinate Offsets must be applied to the values loaded into these registers.

The contents of the ...MIN registers determine the leftmost pixel or topmost line in a region. The contents of the ...MAX registers determine the first subsequent pixel or line which is no longer in the region. In other words, a ...MAX register should be loaded with the sum of the ...MIN value and the width or height of the region. The contents of a ...MAX register must always be greater than those of its corresponding ...MIN register; otherwise the behavior of the detector is unspecified.

12.11 Displaying a Sprite Cursor

A 16-by-16 pixel sprite cursor is displayed when cursor command register bit XHAIR is cleared to zero. The displayed position of the upper left corner of the sprite is controlled by the contents of the CUR_XPOS and CUR_YPOS registers. The values loaded into these registers must include an offset as described above under Cursor Coordinate Offsets. The cursor may be positioned at any pixel in both axes and may be positioned so that part of it falls outside the visible raster.

12.12 Displaying a Crosshair Cursor

A crosshair cursor is displayed when cursor command register bit XHAIR is set to one. This cursor consists of a vertical line and a horizontal line which cross at the point determined by the contents of the CUR_XPOS and CUR_YPOS registers. The values loaded into these registers must include an offset as described above under Cursor Coordinate Offsets. The cursor may be positioned at any pixel in both axes.

Cursor command register bit XHWID controls the width of the lines: if it is zero the lines are one pixel wide; if it is one the lines are doubled in width by adding another line one pixel to the right of the vertical line and below the horizontal line.

The length of the lines is controlled by cursor command register bit XHCLP. If it is zero, the lines extend the full width and height of the raster. If XHCLP is one, the lines are clipped by the region detector selected by cursor command register bit XHCL1: one selects region 1; zero selects region 2.

12.13 Controlling Cursor Plane Outputs

For each cursor plane (A and B) there are two bits in the cursor command register which control its output: the enable bit and the force bit.

The enable bits are ENPA for plane A and ENPB for plane B. If one of these is one, normal cursor data (sprite or crosshair) is generated for the corresponding plane. If one of these is zero, the corresponding plane output is always zero. Setting both of these bits to zero suppresses the cursor display so that the screen shows only the contents of the data plane. These bits are buffered so that they take effect only at the start of a vertical blanking interval.

The force bits are FOPA for plane A and FOPB for plane B. If one of these is one, the output of the corresponding plane is always one throughout the entire display raster and regardless of the state of the plane's enable bit. The force bits are not buffered; they take effect immediately upon loading. These bits must be zero for normal display operation.

12.14 Blanking the Display

The screen may be blanked without disturbing the display data plane or cursor by using the cursor plane control bits to force the output of the B plane to one (set cursor command register bit FOPB) and the A plane to zero (clear cursor command register bits FOPA and ENPA).

12.15 Cursor Chip Test

The cursor chip has a test flipflop which can be used to verify that the chip is functioning correctly. The state of this flipflop appears in bit CURTEST of the configuration and test register CFGTST (see section 8.2). (The value of the CURTEST bit is the complement of the hardware flipflop output; this document describes flipflop values as reflected by the CURTEST bit rather than the hardware values described in the cursor chip hardware specification.)

The test flipflop is forced to one whenever the TEST bit in the cursor command register is one. Whenever the TEST bit of the command register is zero, the test flipflop will be set to zero by the logical OR of the outputs from any of cursor plane A, cursor plane B, region detector 1, and region detector 2, as qualified by the "enable" and "force" bits for each plane and detector in the command register.

Note that a test requires several frame times to execute. A test procedure should wait for an end-of-frame interrupt, set up the test conditions, wait for another end-of-frame interrupt, use the cursor command register to reset the test flipflop, wait for the next end-of-frame interrupt, and then look at the test flipflop value.

12.16 Power-on Initialization

Upon power-on the following are true:

- end-of-frame interrupt is masked off;
- display origin register VDC_ORG is 00h;
- cursor chip register contents are indeterminate;

- data plane storage contents are indeterminate.

The video timing signals are generated at fixed rates by the hardware; no programmed initialization is required for them. The cursor chip requires two vertical blanking cycles to perform internal initialization before its registers can be loaded.

To provide a clean appearance on the monitor, the startup code should wait for at least 50 milliseconds (for cursor chip internal initialization) and then set cursor command register bits HSHI and FOPB to one and clear the others to zero. This sets the proper sync signal polarity and blanks the screen by forcing the B plane output to one and the A plane output to zero.

Note that during normal operation cursor command register bits HSHI, ENPA, and ENPB must be one, and bit VBHI must be zero.

CHAPTER 13

KA43 NETWORK CONTROLLER

The network controller enables the connection of a RigelMAX system to an Ethernet network via a ThinWire connection using RG-58 coax cable or to a transceiver cable. The controller is on the system board and consists of a Lance Ethernet controller chip, a serial interface adapter, an Ethernet transceiver chip, a BNC connector for the RG-58 cable to the Ethernet and a 15 pin Dsub connector for the transceiver cable.

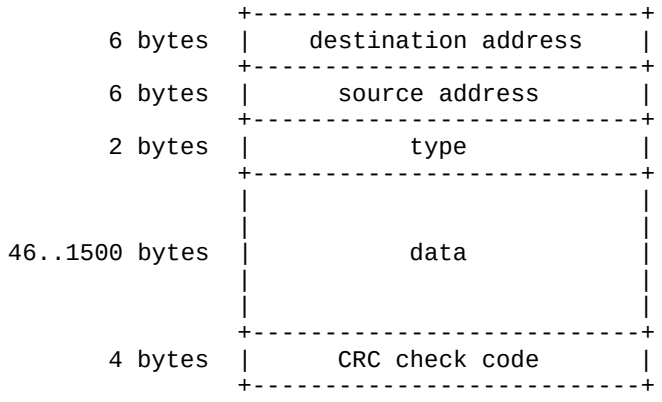
13.1 Ethernet Implementation

This option supports the Ethernet Data Link Layer which is specified in DEC Standard 134: *The Digital Ethernet Specification*.

13.1.1 Packet Format

Data is passed over the Ethernet at a serial data rate of 10 million bits per second in variable-length packets. Each packet has the following format:

Figure 53: Ethernet Packet Format



The minimum size of a packet is 64 bytes, which implies a minimum data length of 46 bytes. Packets shorter than this are called "runt packets" and are treated as erroneous when received by the network controller.

13.1.2 Network Addresses

Network addresses are 48 bits (6 bytes) long and are of two types:

Physical address: The unique address associated with a particular station on the Ethernet, which should be distinct from the physical address of any other station on any other Ethernet.

Multicast address: A multi-destination address associated with one or more stations on a given Ethernet (sometimes called a logical address). There are two kinds of multicast addresses:

Multicast-group address: An address associated by higher-level convention with a group of logically related stations.

Broadcast address: A predefined multicast address which denotes the set of *all* the stations on the Ethernet.

Bit 0 (the least significant bit of the first byte) of an address denotes the type: it is 0 for physical addresses and 1 for multicast addresses. In either case the remaining 47 bits form the address value. A value of 48 ones is always treated as the broadcast address.

The physical address of each RigelMAX system is determined at the time of manufacture and is stored in the Ethernet Address ROM on the system board (see section 2.2).

13.2 Lance Chip Overview

The Lance chip is a microprogrammed controller which can conduct extensive operations independently of the central processor. There are four control and status registers (CSR's) within the Lance chip which are programmed by the central processor (i.e. the CVAX CPU chip) to initialize the Lance chip and start its independent operation. Once started, the Lance uses its builtin DMA controller to directly access RAM memory to get additional operating parameters and to manage the buffers it uses to transfer packets to and from the Ethernet. The Lance uses three structures in memory:

Table 49: Lance Memory Structures

Structure	Purpose
1)	Initialization Block—24 bytes of contiguous memory starting on a word boundary. The initialization block is set up by the central processor and is read by the Lance when the processor starts the Lance's initialization process. The initialization block contains the system's network address and pointers to the receive and transmit descriptor rings; it is described in section 13.6 below.
2)	Descriptor Rings—two logically circular rings of buffer descriptors, one ring used by the chip receiver for incoming data and one ring used by the chip transmitter for outgoing data. Each buffer descriptor in a ring is 8 bytes long and starts on a quadword boundary. It points to a data buffer elsewhere in memory, contains the size of that buffer, and holds various status information about the buffer's contents. Buffer descriptors are described in section 13.7 below.
3)	Data Buffers—contiguous portions of memory to buffer incoming or outgoing packets. Data buffers must be at least 64 bytes long (100 bytes for the first buffer of a packet to be transmitted) and may begin on any byte boundary. They are discussed in section 13.7 below.

When the system is ready to begin network operation, the central processor sets up the initialization block, the receive descriptor ring, the transmit descriptor ring, and their data buffers in memory, and then starts the Lance by writing to its CSR's. The Lance performs its initialization process and then enters its polling loop. In this loop, it listens to the network for packets whose destination addresses are of interest and it scans the transmit descriptor ring for descriptors which have been marked by the central processor to indicate that they contain outgoing data packets. When it detects a network packet of interest, it receives and stores that packet in one or more receive buffers and marks their descriptors accordingly. When it finds a packet to be transmitted, it transmits it to the network and marks its descriptor when transmission is complete. Whenever it completes a reception or transmission (or encounters an error condition), the Lance chip sets flags in its control and status register 0 to signal the central processor (usually by an interrupt) that it has done something of interest.

13.3 Program Control of the Lance

Program control of the Lance chip is via two 16-bit read/write ports, each of which appears as the low-order word of a longword address. These ports are:

Address	Name	Description
200E.0000	NI_RDP	Register data port
200E.0004	NI_RAP	Register address port

These ports provide access to four 16-bit control and status registers which are named NI_CSR0 through NI_CSR3. A CSR is accessed by first writing its number into the register address port NI_RAP after which the contents of the CSR are read or written by accesses to the register data port NI_RDP. Note that registers other than NI_CSR0 may be accessed only while the STOP bit of NI_CSR0 is set.

13.3.1 Register Address Port(NI_RAP)

The register address port is a 16-bit read/write port at physical address 200E.0004. It selects which of the four CSR's is accessed via the register data port.

Figure 54: Lance Register Address Port(NI_RAP)

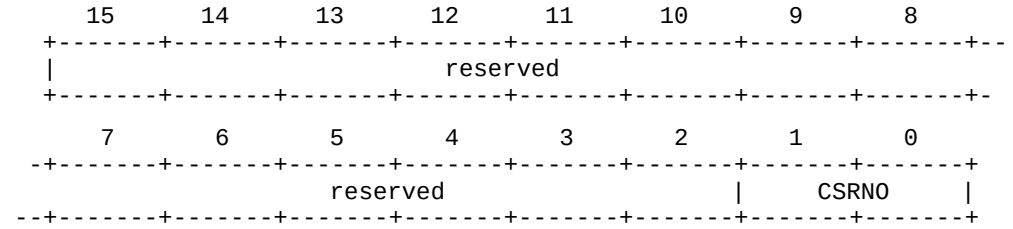


Table 50: Lance Register Address Port Bit Assignments

Bit	Meaning
<15:2>	Reserved. Ignored on write; read as zeros.
CSRNO	CSR select (bits 1:0). These read/write bits select which of the four CSR's is accessible via the register data port. They are cleared to zero upon power-on. Values are: Bits 1:0 — Register 0 0 — NI_CSR0 0 1 — NI_CSR1 1 0 — NI_CSR2 1 1 — NI_CSR3

13.3.2 Register Data Port(NI_RDP)

The register data port at physical address 200E.0000 is a 16-bit window through which the CPU can read and write the CSR designated by the register address port NI_RAP.

Note that registers NI_CSR1, NI_CSR2, and NI_CSR3 are accessible only while the STOP bit in NI_CSR0 is set. If that STOP bit is clear (i.e. the Lance chip is active), attempts to read from those CSR's will return UNDEFINED data and attempts to write to them will be ignored. Accesses to a CSR via NI_RDP do not alter the register address pointer NI_RAP. In normal operation only NI_CSR0 can be accessed, so NI_RAP should be set to point to NI_CSR0 and left that way.

13.3.3 Control and Status Register 0 (NI_CSR0)

This register is used by the controlling program to start and stop the operation of the Lance chip and to monitor its status. It is accessible to the processor via port NI_RDP when bits 1:0 of NI_RAP are set to 00. All of its bits can be read at any time and none of its bits is affected by reading the register. The effects of a write operation are described individually for each bit.

When power is applied to the system, all the bits in this register are cleared except the STOP bit which is set.

Figure 55: Lance Control and Status Register 0(NI_CSR0)

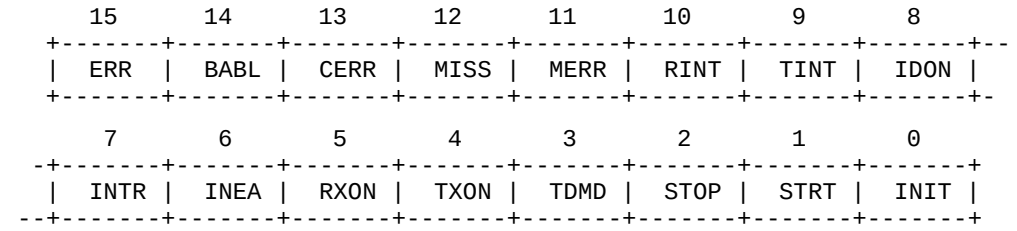


Table 51: Lance Control and Status Register Bit Assignments

Bit	Meaning
ERR	Error summary (bit 15). This read-only bit is one whenever any of the bits BABL, CERR, MISS, or MERR in this register are ones. Writing to this bit has no effect. It is cleared when all of the bits which set it are zero or when the STOP bit is set.
BABL	Transmitter timeout error (bit 14). This bit is set when the transmitter has been on the channel longer than the time required to send the maximum length packet. It will be set after 1519 data bytes have been transmitted (the chip will continue to transmit until the whole packet is transmitted or until there is a failure before the whole packet is transmitted). This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the ERR and INTR bits are also ones.
CERR	Collision error (bit 13). This bit is set when the collision input to the chip failed to activate within 2 microsec after a chip-initiated transmission is completed. This collision-after-transmission is a transceiver test feature. This function is also known as heartbeat or SQE (signal quality error) test. This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the ERR bit is also one.
MISS	Missed packet (bit 12). This bit is set when the receiver loses a packet because it does not own a receive buffer. The MISS bit is not valid in internal loopback mode. This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the ERR and INTR bits are also ones.
MERR	Memory error (bit 11). This bit is set when the chip attempts a DMA transfer and does not receive a ready response from the memory within 25.6 microsec after beginning the memory cycle. This condition occurs when a parity error occurred on an immediately preceding DMA bus read cycle which asserted the ERR signal, as described in section <REFERENCE>(MSER_REG\VALUE). When MERR is set, the receiver and transmitter are turned off (bits RXON and TXON of this register are cleared to zero). This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the ERR and INTR bits are also ones.
RINT	Receive interrupt (bit 10). This bit is set when the chip updates an entry in the receive descriptor ring for the last buffer received or when reception is stopped due to a failure. This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the INTR bit is also one.
TINT	Transmitter interrupt (bit 9).

Table 51 (Cont.): Lance Control and Status Register Bit Assignments

Bit	Meaning
	This bit is set when the chip updates an entry in the transmit descriptor ring for the last buffer sent or when transmission is stopped due to a failure. This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the INTR bit is also one.
IDON	Initialization done (bit 8). This bit is set when the chip completes the initialization process which was started by setting the INIT bit in this register. When IDON is set, the chip has read the initialization block from memory and stored the new parameters. This bit is cleared when a one is written to it (writing a zero has no effect) or when the STOP bit is set. When this bit is one, the INTR bit is also one.
INTR	Interrupt request (bit 7). This read-only bit is one whenever any of the bits BABL, MISS, MERR, RINT, TINT, or IDON in this register are ones. Writing to this bit has no effect. It is cleared when all of the bits which set it are zero or when the STOP bit is set. When both the INTR and INEA bits in this register are set, an interrupt request is sent to the system interrupt controller (see section 13.4).
INEA	Interrupt enable (bit 6). This read/write bit controls whether the setting of the INTR bit generates an interrupt request. When both the INTR and INEA bits in this register are set, an interrupt request is sent to the system interrupt controller (see section 13.4). This bit is set when a one is written to it. It is cleared when a zero is written to it or when the STOP bit is set.
RXON	Receiver on (bit 5). This read-only bit indicates (when it is one) that the receiver is enabled. RXON is set when initialization is completed (i.e. when IDON is set, unless the DRX bit of the initialization block MODE register was one) and then the STRT bit in this register is set. Writing to this bit has no effect. RXON is cleared when either the MERR or STOP bits of this register are set.
TXON	Transmitter on (bit 4). This read-only bit indicates (when it is one) that the transmitter is enabled. TXON is set when initialization is completed (i.e. when IDON is set, unless the DTX bit of the initialization block MODE register was one) and then the STRT bit in this register is set. Writing to this bit has no effect. TXON is cleared when either the MERR or STOP bits of this register are set or when any of bits UFLO, BUFF, or RTRY in a Transmit Buffer Descriptor (see section 66) are set.
TDMD	Transmit demand (bit 3). Setting this bit signals the chip to access the transmit descriptor ring without waiting for the polltime interval to elapse. This bit need not be set to transmit a packet; setting it merely hastens the chip's response to the insertion of a transmit descriptor ring entry by the host program. This bit is set by writing a one to it (writing a zero has no effect) and is cleared by the chip when it recognizes the bit (the bit may read as one for a short time after it is set, depending upon the level of activity in the chip). TDMD is also cleared when the STOP bit is set.
STOP	Stop external activity (bit 2). Setting this bit stops all external activity and clears the internal logic of the chip; this has the same effect as the electrical reset signalled upon power-on. The chip remains inactive and STOP remains set until the STRT or INIT bits in this register are set. This bit is set by writing a one to it (writing a zero has no effect) or upon power-on. It is cleared when either INIT or STRT is set. If the processor writes ones to STOP, INIT, and STRT at the same time, STOP takes precedence and neither STRT nor INIT is set. Setting STOP clears all the other bits in this register. After STOP has been set, the other three CSR's (NI_CSR1, NI_CSR2, and NI_CSR3) must be reloaded before setting INIT or STRT (note that those three registers may be accessed *only* while STOP is set).
STRT	Start operation (bit 1).

Table 51 (Cont.): Lance Control and Status Register Bit Assignments

Bit	Meaning
	Setting this bit enables the chip to send and receive packets, perform DMA and do buffer management. The STOP bit must be set prior to setting the STRT bit (setting STRT then clears STOP). STRT is set by writing a one to it (writing a zero has no effect). It is cleared when the STOP bit is set.
INIT	Initialize (bit 0).
	Setting this bit causes the chip to perform its initialization process, which reads the initialization block from the memory addressed by the contents of NI_CSR1 and NI_CSR2 using DMA accesses. The STOP bit must be set prior to setting the INIT bit (setting INIT then clears STOP). INIT is set by writing a one to it (writing a zero has no effect). It is cleared when the STOP bit is set.

NOTE

The INIT and STRT bits must not be set at the same time. The proper initialization procedure is as follows:

Table 52: Lance Initialization

Action
set STOP in NI_CSR0;
set up the initialization block in memory;
load NI_CSR1 and NI_CSR2 with the starting address of the initialization block;
set INIT in NI_CSR0;
wait for IDON in NI_CSR0 to become set;
set STRT in NI_CSR0 to begin operation.

13.3.4 Control and Status Register 1(NI_CSR1)

This read/write register is used in conjunction with NI_CSR2 to supply the 24-bit physical memory address of the initialization block which the chip reads when it performs its initialization process. The register is accessible to the processor via NI_RDP when bits 1:0 of NI_RAP are 01 and the STOP bit of NI_CSR0 is set. Its contents upon power-on are UNPREDICTABLE.

Figure 56: Lance Control and Status Register 1(NI_CSR1)

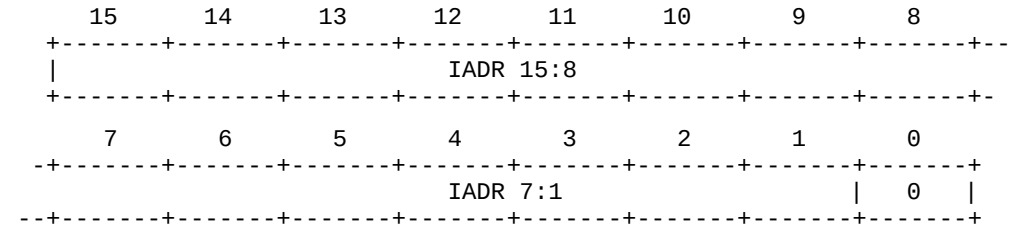


Table 53: Lance Control and Status Register 1 Bit Assignments

Bit	Meaning
IADR	Initialization block address (bits 15:0). These are the low-order sixteen bits of the (24-bit physical) byte address of the first byte of the initialization block. Note that since the block must be word-aligned, bit 0 must be zero.

13.3.5 Control and Status Register 2 (NI_CSR2)

This read/write register is used in conjunction with NI_CSR1 to supply the 24-bit physical memory address of the initialization block which the chip reads when it performs its initialization process. The register is accessible to the processor via NI_RDP when bits 1:0 of NI_RAP are 10 and the STOP bit of NI_CSR0 is set. Its contents upon power-on are UNPREDICTABLE.

Figure 57: Lance Control and Status Register 2(NI_CSR2)

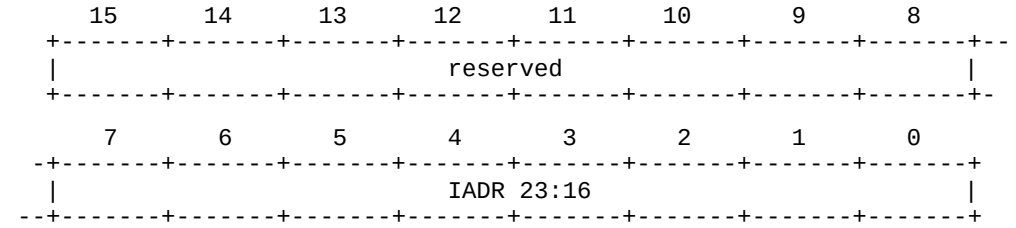


Table 54: Lance Control and Status Register 2 Bit Assignments

Bit	Meaning
<15:8>	Reserved. Write with zeros.
IADR	Initialization block address (bits 7:0). These are the high-order eight bits of the (24-bit physical) byte address of the first byte of the initialization block.

13.3.6 Control and Status Register 3(NI_CSR3)

This read/write register controls certain aspects of the electrical interface between the Lance chip and the system. It *must* be set as indicated for each bit. The register is accessible to the processor via NI_RDP when bits 1:0 of NI_RAP are 11 and the STOP bit of NI_CSR0 is set. Its contents upon power-on are entirely zeros.

Figure 58: Lance Control and Status Register 3(NI_CSR3)

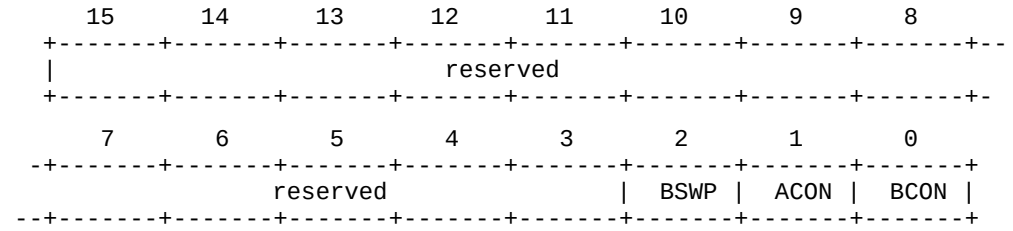


Table 55: Lance Control and Status Register 3 Bit Assignments

Bit	Meaning
<15:3>	Reserved. Ignored on write; read as zeros.
BSWP	Byte swap (bit 2).
	When this bit is set, the chip will swap the high and low bytes for DMA data transfers between the silo and bus memory in order to accomodate processors which consider bus bits 15:08 to be the least significant byte of data. This bit is read/write; it is cleared when the STOP bit in NI_CSR0 is set. For this system, this bit *must be ZERO*.
ACON	ALE control (bit 1).
	This bit controls the polarity of the signal emitted on the chip's ALE/AS pin during DMA operation. This bit is read/write; it is cleared when the STOP bit in NI_CSR0 is set. For this system, this bit *must be ZERO*.
BCON	Byte control (bit 0).
	This bit controls the configuration of the byte mask and hold signals on the chip's pins during DMA operation. This bit is read/write; it is cleared when the STOP bit in NI_CSR0 is set. For this system, this bit *must be ZERO*.

13.4 Interrupts

The Lance chip asserts an interrupt request signal whenever the INTR and INEA bits bit in its control and status register 0 (NI_CSR0) are both ones. This signal is presented to the system interrupt controller as interrupt number 5, the "network controller primary" source, whose vector number is 250h. The change of the interrupt signal from false to true will set bit NP in the interrupt request register INT_REQ (which will generate a CPU interrupt when the corresponding bit in the interrupt mask register INT_MSK is also set). Note that since the input to INT_REG is transition sensitive rather than level sensitive, a program which services an interrupt request from the Lance must either service all the conditions which contributed to the setting of the INTR bit in NI_CSR0 so that INTR will become zero, or must generate another transition of the interrupt signal by setting the INEA bit of NI_CSR0 to zero and then back to one again. (The interrupt controller is described in chapter 9. Interrupt number 4, the "network controller secondary" source, is not used by this option.)

13.5 DMA Operation

PROGRAMMER'S NOTE

Because the cache memory structure does not support DMA, DMA operations should transfer data to a pre-allocated buffer which is addressed in RigelMAX Memory Diagnostic Space (2800.0000-29FF.FFFF). Addresses in this range DO NOT use cache memory but DO reference main memory.

The Lance chip contains a built-in DMA controller which can transfer data directly between the chip and main memory in the address range 2800.0000 through 29FF.FFFF. (See note above.) The chip contains a 48-byte FIFO buffer to allow for DMA service latency and to minimize the number of request-grant arbitration cycles. When transferring large amounts of data in burst mode, the chip transfers 16 bytes per DMA request.

This DMA controller is used to read the initialization block, to read and write the descriptor rings, and to read and write data buffers. Note that all the memory addresses handled by the chip are physical addresses (see note above). Programs which operate with CPU memory management enabled must translate their addresses from virtual to physical form before presenting them to the Lance chip.

If the DPEN bit of the PAR_CTL register is set, then parity is checked during DMA read cycles. When a parity error is detected, the MERR bit of the NI_CSR0 register will be set (with the consequences described in section 13.3.3) but no CPU Machine Check will occur.

13.6 Initialization Block

When the Lance chip is initialized (by setting the INIT bit in NI_CSR0), it reads a 24-byte block of data called the initialization block from main memory using DMA accesses. The physical address of the initialization block (IADR) is taken from NI_CSR1 and NI_CSR2. Since the data must be word-aligned, the low-order bit of the address must be zero. The initialization block comprises 12 16-bit words arranged as follows:

Figure 59: Lance Initialization Block Layout

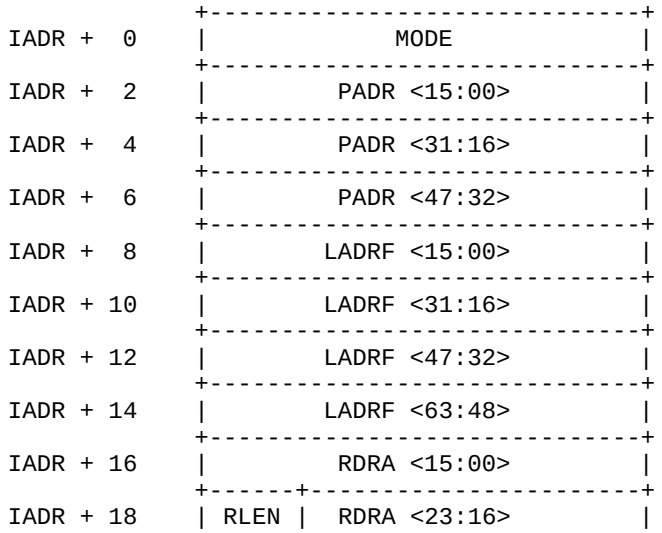
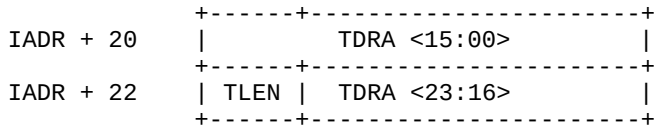


Figure 59 Cont'd. on next page

Figure 59 (Cont.): Lance Initialization Block Layout



13.6.1 Initialization Block MODE Word(NIB_MODE)

The MODE word of the initialization block allows alteration of the Lance chip's normal operation for testing and special applications. For normal operation the MODE word is entirely zero.

Figure 60: Initialization Block MODE Word(NIB_MODE)

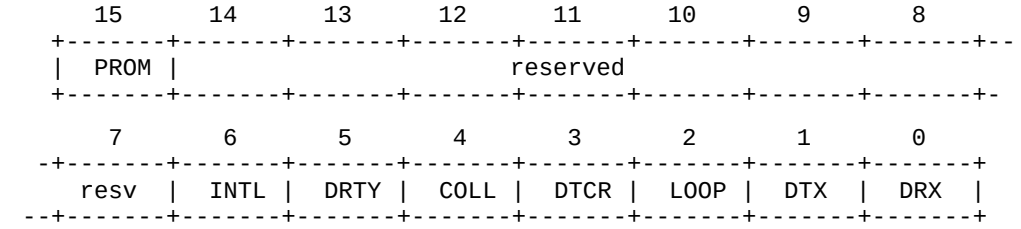


Table 56: Initialization Block MODE Word Bit Assignments

Bit	Meaning
PROM	Promiscuous mode (bit 15). When this bit is set, all incoming packets are accepted regardless of their destination addresses.
<14:7>	Reserved. Should be written with zeros.
INTL	Internal loopback (bit 6). This bit is used in conjunction with the LOOP bit in this word to control loopback operation. See the description of the LOOP bit, below.
DRTY	Disable retry (bit 5). When this bit is set, the chip will attempt only one transmission of a packet. If there is a collision on the first transmission attempt, a retry error (RTRY) will be reported in the transmit buffer descriptor (section 66).
COLL	Force collision (bit 4). Setting this bit allows the collision logic to be tested. The chip must be in internal loopback mode for COLL to be used. When COLL is one a collision will be forced during the subsequent transmission attempt. This will result in 16 total transmission attempts with a retry error reported in NI_TMD3.
DTCR	Disable transmit CRC (bit 3).

Table 56 (Cont.): Initialization Block MODE Word Bit Assignments

Bit	Meaning
	When DTCR is zero the transmitter will generate and append a 4-byte CRC to each transmitted packet (normal operation). When DTCR is one the CRC logic is allocated instead to the receiver and no CRC is sent with a transmitted packet. During loopback, setting DTCR to zero will cause a CRC to be generated and sent with the transmitted packet, but no CRC check can be done by the receiver since the CRC logic is shared and cannot both generate and check a CRC at the same time. The CRC transmitted with the packet will be received and written into memory following the data where it can be checked by software. If DTCR is set to one during loopback, the driving software must compute and append a CRC value to the data to be transmitted. The receiver will check this CRC upon reception and report any error.

LOOP	Loopback control (bit 2).
	Loopback allows the Lance chip to operate in full duplex mode for test purposes. The maximum packet size is limited to 32 data bytes (in addition to which 4 CRC bytes may be appended). During loopback, the runt packet filter is disabled because the maximum packet is forced to be smaller than the minimum size Ethernet packet (64 bytes). Setting LOOP to one allows simultaneous transmission and reception for a packet constrained to fit within the silo. The chip waits until the entire packet is in the silo before beginning serial transmission. The incoming data stream fills the silo from behind as it is being emptied. Moving the received packet out of the silo into memory does not begin until reception has ceased. In loopback mode, transmit data chaining is not possible. Receive data chaining is allowed regardless of the receive buffer length. (In normal operation, the receive buffers must be 64 bytes long, to allow time for buffer lookahead.) Valid loopback bit settings are:
	`LOOP  INTL  Operation`
	`----  ----  ---------`
	`  0     x   Normal on-line operation`
	`  1     0   External loopback`
	`  1     1   Internal loopback`

Internal loopback allows the chip to receive its own transmitted packet without disturbing the network. The chip will not receive any packets from the network while it is in internal loopback mode.

External loopback allows the chip to transmit a packet through the transceiver out to the network cable to check the operability of all circuits and connections between the Lance chip and the network cable. Multicast addressing in external loopback is valid only when DTCR is one (user needs to append the 4 CRC bytes). In external loopback, the chip also receives packets from other nodes.

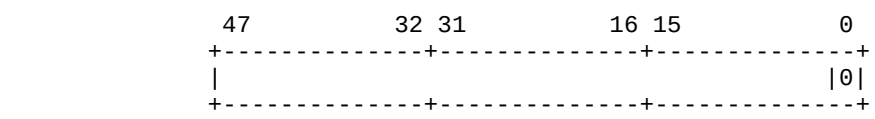
DTX Disable transmitter (bit 1). If this bit is set, the chip will not set the TXON bit in NI_CSR0 at the completion of initialization. This will prevent the Lance chip from attempting to access the transmit descriptor ring, hence no transmissions will be attempted.

DRX Disable receiver (bit 0). If this bit is set, the chip will not set the RXON bit in NI_CSR0 at the completion of initialization. This will cause the chip to reject all incoming packets and to not attempt to access the receive descriptor ring.

13.6.2 Network Physical Address (NIB_PADR)

The 48-bit physical Ethernet network node address is contained in bytes 2:7 of the initialization block. (This is a network address; it has no relationship to any memory address.)

Figure 61: Network Physical Address(NIB_PADR)

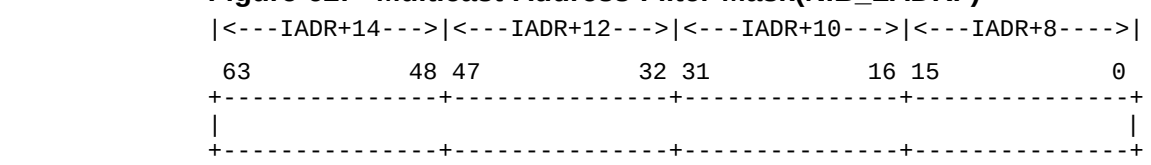


The contents of NIB_PADR identify this station to the network and must be unique within the domain of the network. Its value is normally taken from the Network Address ROM (see section 2.2). The low-order bit (bit 0) of this address must be zero since it is a physical address.

13.6.3 Multicast Address Filter Mask (NIB_LADRF)

Bytes 8:15 of the initialization block contain the 64-bit multicast address filter mask. The multicast address filter is a partial filter which assists the network controller driver program to selectively receive packets which contain multicast network addresses.

Figure 62: Multicast Address Filter Mask(NIB_LADRF)



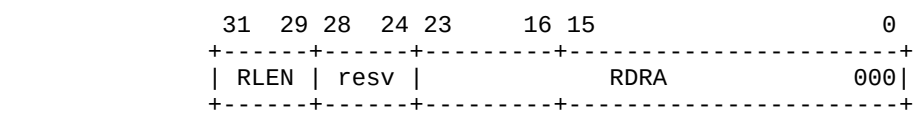
Multicast Ethernet addresses are distinguished from physical network addresses by the presence of a one in bit 0 of the 48-bit address field. If an incoming packet contains a physical destination address (bit 0 is zero), then its entire 48 bits are compared with the contents of NIB_PADR and the packet is ignored if they are not equal. If the packet contains a multicast destination address which is all ones (the broadcast address), it is always accepted and stored regardless of the contents of the multicast address filter mask.

All other multicast addresses are processed through the multicast address filter to determine whether the incoming packet will be stored in a receive buffer. This filtering is performed by passing the multicast address field through the CRC generator. The high-order 6 bits of the resulting 32-bit CRC are used to select one of the 64 bits of NIB_LADRF. (These high-order six bits represent in binary the number of the bit in NIB_LADRF, according to the labelling in figure 62.) If the bit selected from NIB_LADRF is one, the packet is stored in a receive buffer; otherwise it is ignored. This mechanism effectively splits the entire domain of 2**47 multicast addresses into 64 parts, and multicast addresses falling into each part will be accepted or ignored according to the value of the corresponding bit in NIB_LADRF. The driver program must examine the addresses of the packets accepted by this partial filtering to complete the filtering task.

13.6.4 Receive Descriptor Ring Pointer (NIB_RDRP)

Bytes 16:19 of the initialization block describe the starting address and extent of the receive descriptor ring.

Figure 63: Receive Descriptor Ring Pointer(NIB_RDRP)



RLEN Receive ring length (bits 31:29). This field gives the number of entries in the receive descriptor ring, expressed as a power of 2:

RLEN	*Entries*
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

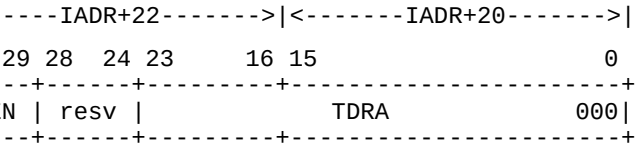
<28:24> Reserved; should be zeros.

RDRA Receive descriptor ring address (bits 23:0). This is the physical address in system memory of the first element in the ring. Since each 8-byte element must be aligned on a quadword boundary, bits 2:0 of this address must be zero.

13.6.5 Transmit Descriptor Ring Pointer (NIB_TDRP)

Bytes 20:23 of the initialization block describe the starting address and extent of the Transmit descriptor ring.

Figure 64: Transmit Descriptor Ring Pointer(NIB_TDRP)



TLEN Transmit ring length (bits 31:29). This field gives the number of entries in the transmit descriptor ring, expressed as a power of 2:

TLEN	*Entries*
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

<28:24> Reserved; should be zeros.

TDRA Transmit descriptor ring address (bits 23:0). This is the physical address in system memory of the first element in the ring. Since each 8-byte element must be aligned on a quadword boundary, bits 2:0 of this address must be zero.

13.7 Buffer Management

The Lance chip manages its data buffers by using two rings of buffer descriptors which are stored in memory: the receive descriptor ring and the transmit descriptor ring. Each buffer descriptor points to a data buffer elsewhere in memory, contains the size of that buffer, and contains status information about that buffer's contents.

The starting location in memory of each ring and the number of descriptors in it are given to the Lance chip via the initialization block (see sections 63 and 64) during the chip initialization process. Each descriptor is 8 bytes long and must be aligned on a quadword boundary (the three low-order bits of its address must be zero). The descriptors in a ring are physically contiguous in memory and the number of descriptors must be a power of 2. The Lance keeps an internal index to its current position in each ring which it increments modulo the number of descriptors in the ring as it advances around each ring.

Once started, the Lance polls each ring to find descriptors for buffers in which to receive incoming packets and from which to transmit outgoing packets, and revises the status information in buffer descriptors as it processes their associated buffers. When polling, the Lance is limited to looking only one ahead of the descriptor with which it is currently working. The high speed of the data stream requires that each buffer be at least 64 bytes long to allow time to chain buffers for packets which are larger than one buffer. (The first buffer of a packet to be transmitted should be at least 100 bytes to avoid problems in case a late collision is detected.)

Each descriptor in a ring is "owned" either by the Lance chip or by the host processor; this status is indicated by the OWN bit in each descriptor. Mutual exclusion is accomplished by the rule that each device can only relinquish ownership of a descriptor to the other device, it can never take ownership; and that each device cannot change any field in a descriptor

or its associated buffer after it has relinquished ownership. When the host processor sets up the rings of descriptors before starting the Lance, it sets the OWN bits such that the Lance will own all the descriptors in the receive descriptor ring (to be used by the Lance to receive packets from the network) and the host will own all the descriptors in the transmit descriptor ring (to be used by the host to set up packets to be transmitted to the network).

13.7.1 Receive Buffer Descriptor

A receive buffer descriptor comprises 4 words aligned in memory on a quadword address boundary.

Figure 65: Receive Buffer Descriptor

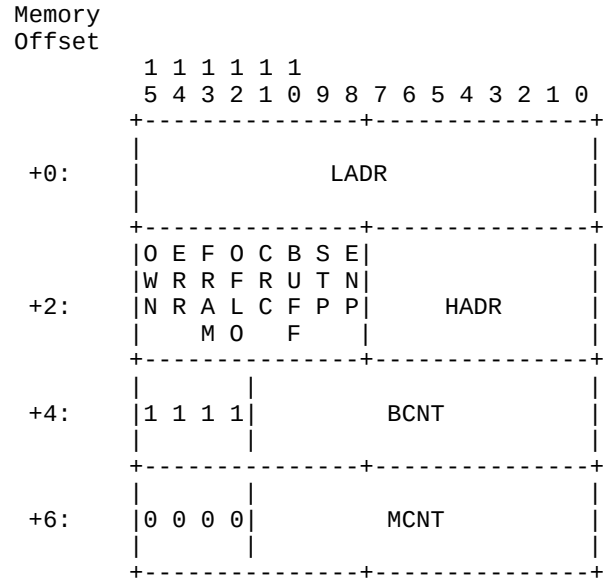


Table 57: Receive Buffer Descriptor Bit Assignments

Bit	Meaning
LADR	Low-order buffer address (offset 0, bits 15:0).
	These are the low-order 16 bits of the 24-bit physical memory address of the start of the buffer associated with this descriptor. Written by the host; unchanged by the Lance.
HADR	High-order buffer address (offset 2, bits 7:0).
	These are the high-order 8 bits of the 24-bit physical memory address of the start of the buffer associated with this descriptor. Written by the host; unchanged by the Lance.
OWN	Owned flag (offset 2, bit 15).
	This bit indicates whether the descriptor is owned by the host (OWN = 0) or by the Lance (OWN = 1). The Lance clears OWN after filling the buffer associated with the descriptor with an incoming packet. The host sets OWN after emptying the buffer. In each case, this must be the last bit changed by the current owner, since changing OWN passes ownership to the other party and the relinquishing party must not thereafter alter anything in the descriptor or its buffer.

Table 57 (Cont.): Receive Buffer Descriptor Bit Assignments

Bit	Meaning
ERR	Error summary (offset 2, bit 14). This is the logical OR of the FRAM, OFLO, CRC and BUFF bits in this word. Set by the Lance and cleared by the host.
FRAM	Framing error (offset 2, bit 13). This bit is set by the Lance to indicate that the incoming packet stored in the buffer had both a non-integral multiple of eight bits and a CRC error. It is cleared by the host.
OFLO	Overflow error (offset 2, bit 12). This bit is set by the Lance to indicate that the receiver has lost part or all of an incoming packet because it could not store it in the buffer before the chip's silo overflowed. Cleared by the host.
CRC	Checksum error (offset 2, bit 11). This bit is set by the Lance to indicate that the received packet has an invalid CRC checksum. Cleared by the host.
BUFF	Buffer error (offset 2, bit 10). This bit is set by the Lance when it has used all its owned receive descriptors or when it could not get the next descriptor in time while attempting to chain to a new buffer in the midst of a packet. When a buffer error occurs, an overflow error (bit OFLO) also occurs because the Lance continues to attempt to get the next buffer until its silo overflows. BUFF is cleared by the host.
STP	Start of packet (offset 2, bit 9). This bit is set by the Lance to indicate that this is the first buffer used for this packet. Cleared by the host.
ENP	End of packet (offset 2, bit 8). This bit is set by the Lance to indicate that this is the last buffer used for this packet. When both STP and ENP are set in a descriptor, its buffer contains an entire packet; otherwise two or more buffers have been chained together to hold the packet. ENP is cleared by the host.
1111	Offset 4, bits 15:12 must be set by the host to ones. Unchanged by the Lance.
BCNT	Buffer size (offset 4, bits 11:0). This is the number of bytes in the buffer (whose starting address is in HADR and LADR) in two's complement form. Note that the minimum buffer size is 64 bytes and that the maximum required for a legal packet is 1518 bytes. Written by the host; unchanged by the Lance.
0000	Offset 6, bits 15:12 are reserved; they should be set to zeros by the host when it constructs the descriptor.
MCNT	Byte count (offset 6, bits 11:0). This is the length in bytes of the received packet for which this is the last or only descriptor. MCNT is valid only in a descriptor in which ENP is set (last buffer) and ERR is clear (no error). Set by the Lance and cleared by the host.

13.7.2 Transmit Buffer Descriptor

A transmit buffer descriptor comprises 4 words aligned in memory on a quadword address boundary.

Figure 66: Transmit Buffer Descriptor

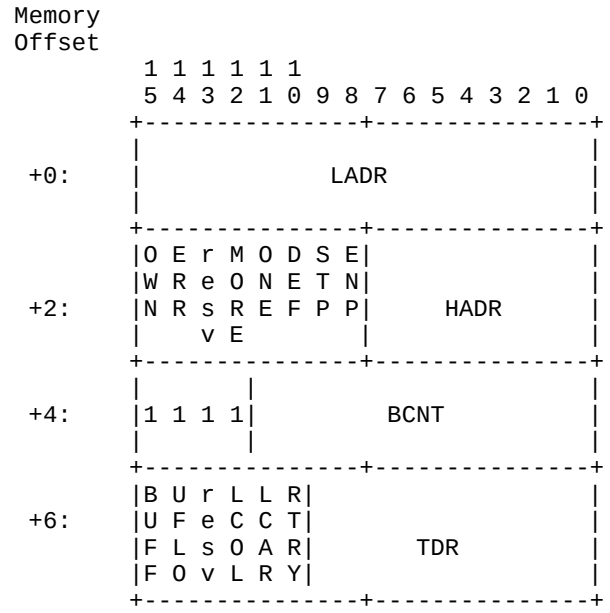


Table 58: Transmit Buffer Descriptor Bit Assignments

Bit	Meaning
LADR	Low-order buffer address (offset 0, bits 15:0).
	These are the low-order 16 bits of the 24-bit physical memory address of the start of the buffer associated with this descriptor. Written by the host; unchanged by the Lance.
HADR	High-order buffer address (offset 2, bits 7:0).
	These are the high-order 8 bits of the 24-bit physical memory address of the start of the buffer associated with this descriptor. Written by the host; unchanged by the Lance.
OWN	Owned flag (offset 2, bit 15).
	This bit indicates whether the descriptor is owned by the host (OWN = 0) or by the Lance (OWN = 1). The host sets OWN after filling the buffer with a packet to be transmitted. The Lance clears OWN after transmitting the contents of the buffer. In each case, this must be the last bit changed by the current owner, since changing OWN passes ownership to the other party and the relinquishing party must not thereafter alter anything in the descriptor or its buffer.
ERR	Error summary (offset 2, bit 14).
	This is the logical OR of the LCOL, LCAR, UFLO and RTRY bits in this descriptor. Set by the Lance and cleared by the host.
resv	Offset 2, bit 13 is reserved.
	The Lance will write a zero in this bit.
MORE	More retries (offset 2, bit 12).
	The Lance sets this bit when more than one retry was required to transmit the packet. Cleared by the host.
ONE	One retry (offset 2, bit 11).
	The Lance sets this bit when exactly one retry was required to transmit the packet. Cleared by the host.
DEF	Deferred (offset 2, bit 10).

Table 58 (Cont.): Transmit Buffer Descriptor Bit Assignments

Bit	Meaning
	The Lance sets this bit when it had to defer while trying to transmit the packet. This occurs when the network is busy when the Lance is ready to transmit. Cleared by the host.
STP	Start of packet (offset 2, bit 9).
	This bit is set by the host to indicate that this is the first buffer used for this packet. STP is not changed by the Lance.
ENP	End of packet (offset 2, bit 8). This bit is set by the host to indicate that this is the last buffer used for this packet. When both STP and ENP are set in a descriptor, its buffer contains an entire packet; otherwise two or more buffers have been chained together to hold the packet. ENP is not changed by the Lance.
1111	Offset 4, bits 15:12 must be set by the host to ones. Unchanged by the Lance.
BCNT	Byte count (offset 4, bits 11:0). This is the number of bytes, in two's complement form, which the Lance will transmit from this buffer. Note that for any buffer which is not the last of a packet, at least 64 bytes (100 bytes if it is the start of the packet) must be transmitted to allow adequate time for the Lance to acquire the next buffer. Written by the host; unchanged by the Lance.
	NOTE: The remaining fields of the descriptor (which make up its entire fourth word) are valid only when the ERR bit in the second word has been set by the Lance.
BUFF	Buffer error (offset 6, bit 15).
	This bit is set by the Lance during transmission when it does not find the ENP bit set in the current descriptor and it does not own the next descriptor. When BUFF is set, the UFLO bit (below) is also set because the Lance continues to transmit until its silo becomes empty. BUFF is cleared by the host.
UFLO	Underflow (offset 6, bit 14). This bit is set by the Lance when it truncates a packet being transmitted because it has drained its silo before it was able to obtain additional data from a buffer in memory. UFLO is cleared by the host.
resv	Offset 6, bit 13 is reserved.
	The Lance will write a zero in this bit.
LCOL	Late collision (offset 6, bit 12). This bit is set by the Lance to indicate that a collision has occured after the slot time of the network channel has elapsed. The Lance does not retry after a late collision. LCOL is cleared by the host.
LCAR	Loss of carrier (offset 6, bit 11).
	This bit is set by the Lance when the carrier-present input to the chip becomes false during a transmission initiated by the Lance. The Lance does not retry after such a failure. LCAR is cleared by the host.
RTRY	Retries exhausted (offset 6, bit 10).
	This bit is set by the Lance after 16 attempts to transmit a packet have failed due to repeated collisions on the network. (If the DRTY bit of the initialization block MODE word (section 60) is set, RTRY will instead be set after only only one failed transmission attempt.) RTRY is cleared by the host.
TDR	Time domain reflectometer (offset 6, bits 9:0).
	These bits are the value of an internal counter which is set by the Lance to count system clocks from the start of a transmission to the occurrence of a collision. This value is useful in determing the approximate distance to a cable fault; it is valid only when the RTRY bit in this word is set.

13.8 Lance Operation

The Lance chip operates independently of the host under control of its own internal microprogram. This section is a simplified description of the operation of the Lance in terms of its principal microcode routines (these should not be confused with device driver programming in the host, which is not a part of this specification). These microcode routines make use of numerous temporary storage cells within the Lance chip; most of these are not accessible from outside the chip but they are mentioned here when necessary to clarify the operation of the microcode.

Two such (conceptual) internal variables are of central importance: the pointers to the "current" entry in the receive descriptor ring and in the transmit descriptor ring, which are referred to below as TXP and RXP. Each of these designates the descriptor which the Lance will use for the next operation of that type. If the descriptor designated by one of these pointers is not owned by the Lance (the OWN bit is 0), then the Lance can neither perform activity of that type nor advance the pointer. For the transmit ring, the Lance will do nothing until the host sets up a packet in the buffer and sets the OWN bit in the descriptor designated by the Lance's TXP. (The host must keep track of the position of the TXP, since setting up a packet in some other descriptor will not be detected by the Lance). For the receive ring, if the Lance does not own the descriptor designated by RXP, it cannot receive a packet. In both rings, when the Lance finishes with a descriptor and relinquishes it to the host by clearing OWN, it then advances the ring pointer (modulo the number of entries in the ring).

When the Lance begins activity using the current descriptor (i.e. begins receiving or transmitting a packet), it may look ahead at the next descriptor and attempt to read its first three words in advance so it can chain to the next buffer in mid-packet without losing data. However, it does not actually advance its RXP or TXP until it has cleared the OWN bit in the current descriptor.

The Lance is a very complex chip and this system specification does not attempt to cover all the details of its operation. The chip purchase specification cited in section 1.3 and the chip vendor's literature should also be consulted.

13.8.1 Switch Routine

Upon power on, the STOP bit is set and the INIT and STRT bits are cleared in NI_CSR0. The Lance microprogram begins execution in the switch routine, which tests the INIT, STRT, and STOP bits. When the host sets either INIT or STRT, STOP is cleared. While STOP is set, if the host writes to NI_CSR1 and NI_CSR2, that data is stored for use by the initialization routine.

When the microprogram sees STOP cleared, it tests first the INIT bit and then the STRT bit. If INIT is set, it performs the initialization routine (section 13.8.2). Then if STRT is set, it begins active chip operation by jumping to the look-for-work routine (section 13.8.3). Control returns to the switch routine whenever the host again sets the STOP bit (which also clears the INIT and STRT bits). Note that the ring pointers RXP and TXP are not altered by the setting of either STOP or START; they are reset to the start of their rings only when INIT is set.

13.8.2 Initialization Routine

The initialization routine is called from the switch routine when the latter finds the INIT bit set. It reads the initialization block (section 13.6) from the memory addressed by NI_CSR1 and NI_CSR2 and stores its data within the Lance chip. This routine also sets the ring pointers RXP and TXP to the start of their rings (i.e. to point to the descriptor at the lowest memory address in the ring).

13.8.3 Look-for-work Routine

The look-for-work routine is executed while the Lance is active and looking for work. It is entered from the switch routine when the STRT bit is set, and is returned to from the receive and transmit routines after they have received or transmitted a packet.

This routine begins by testing whether the receiver is enabled (bit RXON of NI_CSR0 is set). If so, it tries to have a receive buffer available for immediate use when a packet addressed to this system arrives. It tests its internal registers to see whether it has already found a receive descriptor owned by the Lance and, if not, calls the receive poll routine (section 13.8.4) to attempt to get a receive buffer.

Next the routine tests whether the transmitter is enabled (bit TXON of NI_CSR0 is set). If so, it calls the transmit poll routine (section 13.8.7) to see whether there is a packet to be transmitted and to transmit it if so.

If there was no transmission and the TDMD bit of NI_CSR0 is not set, the microprogram delays 1.6 milliseconds and then goes to check the receive descriptor status again. If a packet was transmitted or the host has set TDMD, the delay is omitted so that multiple packets will be transmitted as quickly as possible.

If at any point in this routine the receiver detects an incoming packet whose destination address matches the station's physical address, is the broadcast address, or passes the multicast address filter (or if the PROM bit of NIB_MODE is set), the receive routine (section 13.8.5) is called.

13.8.4 Receive Poll Routine

The receive poll routine is called whenever the receiver is enabled and the Lance needs a free buffer from the receive descriptor ring. The routine reads the second word of the descriptor designated by RXP and, if the OWN bit in it is set, reads the first and third words also.

13.8.5 Receive Routine

The receive routine is called when the receiver is enabled and an incoming packet's destination address field matches one of the criteria described in 13.8.3 above. The routine has three sections: initialization, lookahead, and descriptor update.

In initialization, the routine checks whether a receive ring descriptor has already been acquired by the receive poll routine. If not, it makes one attempt to get the descriptor designated by RXP (if OWN is not set in it, MISS and ERR are set in NI_CSR0 and the packet is lost). The buffer thus acquired is used by the receive DMA routine to empty the silo.

In lookahead, the routine reads the second word of the next descriptor in the receive ring and, if the OWN bit is set, reads the rest of the descriptor and holds it in readiness for possible data chaining.

The descriptor update section is performed when either the current buffer is filled or the packet ends. If the packet ends but its total length is less than 64 bytes, it is an erroneous "runt packet" and is ignored: no status is posted in the descriptor, RXP is not moved, and the buffer will be reused for the next incoming packet (this is why a receive buffer must be at least 64 bytes long; otherwise the runt might be detected after advancing RXP).

If the packet ends (with or without error), the routine writes the packet length into MCNT, sets ENP and other appropriate status bits and clears OWN in the current descriptor, and sets RINT in NI_CSR0 to signal the host that a complete packet has been received. Then it advances RXP and returns to the look-for-work routine.

If the buffer is full and the packet has not ended, chaining is required. The routine releases the current buffer by writing status bits into its descriptor (clearing OWN and ENP, in particular), makes current the next descriptor data acquired in the lookahead section, advances RXP, and goes to the lookahead section to prepare for possible additional chaining. Note that RINT is not set in NI_CSR0, although the host would find OWN cleared if it looked at the descriptor, and it could begin work on that section of the packet, since the mutual exclusion rule prevents the Lance from going back and altering it.

13.8.6 Receive DMA Routine

The receive DMA routine is invoked asynchronously by the chip hardware during execution of the receive routine whenever the silo contains 16 or more bytes of incoming data or when the packet ends and the silo is not empty. It executes DMA cycles to drain data from the silo into the buffer designated by the current descriptor.

13.8.7 Transmit Poll Routine

The transmit poll routine is called by the look-for-work routine (section 13.8.3) to see whether a packet is ready for transmission. It reads the second word of the descriptor designated by TXP and tests the OWN bit. If OWN is zero, the Lance does not own the buffer and this routine returns to its caller. If OWN is set, the routine tests the STP bit, which should be set to indicate the start of a packet. If STP is clear, this is an invalid packet; the Lance sets its OWN bit to return it to the host, sets TINT in NI_CSR0 to notify the host, and advances TXP to the next transmit descriptor. If both OWN and STP are set, this is the beginning of a packet, so the transmit poll routine reads the rest of the descriptor and then calls the transmit routine (section 13.8.8) to transmit the packet. During this time the chip is still watching for incoming packets from the network and it will abort the transmit operation if one arrives.

13.8.8 Transmit Routine

The transmit routine is called from the transmit poll routine (section 13.8.7) when the latter finds the start of a packet to be transmitted. This routine has three sections: initialization, lookahead, and descriptor update.

In initialization, the routine sets the chip's internal buffer address and byte count from the transmit descriptor, enables the transmit DMA engine, and starts transmission of the packet preamble. It then waits until the transmitter is actually sending the bit stream (including possible backoff-and-retry actions in case of collisions).

In lookahead, the transmit routine test the current descriptor to see whether it is the last in the packet (the ENP bit is set). If so, no additional buffer is required so the routine waits until all the bytes from the current packet have been transmitted. If not, the routine attempts to get the next descriptor and hold it in readiness for data chaining, and then waits until all the bytes from the current buffer have been transmitted.

Descriptor update is entered when all the bytes from a buffer have been transmitted or an error has occured. If there is no error and the buffer was not the last of the packet, the pre-fetched descriptor for the next buffer is made current for use by the transmit DMA routine. The routine writes the appropriate status bits and clears the OWN bits in the current descriptor and advances TXP. If this was the last buffer in the packet, the routine sets the TINT bit in NI_CSR0 to notify the host and returns to the look-for-work routine (section 13.8.3); otherwise it goes back to the lookahead section in this routine.

13.8.9 Transmit DMA Routine

The transmit DMA routine is invoked asynchronously by the chip hardware during execution of the transmit routine whenever the silo has 16 or more empty bytes. It executes DMA cycles to fill the silo with data from the buffer designated by the current descriptor.

13.8.10 Collision Detect Routine

This routine is invoked asynchronously by the chip hardware during execution of the transmit routine when a collision is detected on the network. It ensures that the "jam" sequence is transmitted, then backs up the chip's internal buffer address and byte count registers, waits for a pseudo-random backoff time, and then attempts the transmission again. If 15 retransmission attempts fail (a total of 16 attempts), it sends the microcode to the descriptor update routine to report an error in the current transmit descriptor (bits RTRY and ERR are set).

13.9 Lance Programming Notes

1. The interrupt signal is simply the OR of the interrupt-causing conditions. If another such condition occurs while the interrupt signal is already asserted, there will not be another active transition of the interrupt signal and the interrupt request bit in INT_REQ will not be set again. An interrupt service routine should use logic similar to the following to avoid losing interrupts: Read NI_CSR0 and save the results in a register, say R0. Clear the interrupt enable bit INEA in the saved data in R0. Write NI_CSR0 with the saved data in R0. This will make the interrupt signal false because INEA is clear and will clear all the write-one-to-reset bits such as RINT, TINT and the error bits; it will not alter the STRT, INIT or STOP bits nor any interrupt-cause bits which came true after NI_CSR0 was read. Write NI_CSR0 with only INEA to enable interrupts again. Service all the interrupt and error conditions indicated by the flags in the data in R0. Exit from the interrupt service routine. Be sure to access NI_CSR0 only with instructions which do a single access, such as MOVE. Instructions such as BIS which do a read-modify-write operation can have unintended side effects.

2. An interrupt is signalled to the host only when the *last* buffer of a multibuffer (chained) packet is received or transmitted. However, the OWN bit in each descriptor is cleared as soon as the Lance has finished with that portion of the packet, and the mutual exclusion rule makes it safe for the host to process such a descriptor and its buffer.

3. When a transmitter underflow occurs (UFLO is set in a transmit descriptor because the silo is not filled fast enough), the Lance will turn off its transmitter and the Lance must be restarted to turn the transmitter back on again. This can be done by setting STOP in NI_CSR0 and then setting STRT in NI_CSR0 (DTX will still be clear in the chip's internal copy of NIB_MODE). It is necessary to set INIT to reread the initialization block.

   Note that setting STOP will immediately terminate any reception which is in progress. If the status of a receive descriptor has been updated and its OWN bit is now clear, then the contents of its buffer are valid. If the incoming packet was chained into more than one buffer, however, the packet is only valid if its last buffer has been completed (the one with the ENP bit set).

4. The network controller hardware requires up to five seconds after power on to become stable. Self-test routines must delay at least this time before attempting to use the controller for either internal or external testing.

5. The LCAR flag (loss of carrier) may be set in the transmit descriptor when a packet is sent in internal loopback mode. When the Lance is operating in internal loopback mode and a transmission is attempted with a non-matching address, the Lance will correctly reject that packet. If the next operation is an internal loopback transmission without first resetting the Lance, the packet will not be sent and LCAR will be set in the transmit descriptor for that packet. The receive descriptor will still be owned by the Lance. To avoid this problem, the Lance should be reinitialized after each internal loopback packet.

6. The ONE flag is occasionally set in a transmit descriptor after a late collision. The Lance does not attempt a retransmission even though ONE may be set. The host should disregard ONE if the LCOL flag is also set.

7. The chip's internal copy of NI_CSR1 may become invalid when the chip is stopped. The NI_CSR1 and NI_CSR2 registers should always be loaded prior to setting INIT to initialize the Lance chip.

8. Attempting an external loopback test on a busy network can cause a silo pointer misalignment if a transmit abort occurs while the chip was preparing to transmit the loopback packet. The resulting retransmission may cause the transmitter enable circuit to hang, and the resulting illegal length transmission must be terminated by the jabber timer in the transceiver. It is unlikely that there will be a corrupted receive buffer because the reception that caused the transmit abort will usually not pass address recognition.

   Since external loopback is a controlled situation it is possible to implement a software procedure to detect a silo pointer misalignment problem and prevent continuous transmissions. Since the test is being done in loopback the exact length and contents of the receive packet are known; thus the software can determine whether the data in the receive buffer has been corrupted.

On transmission the diagnostic software should allow up to 32 retries before a hard error is flagged. This is not to say that 32 errors are allowed for each condition; the sum of all errors encountered in the test should not exceed 32. The diagnostic software should expect to get a transmit done interrupt with 1 millisecond of passing the transmit packet to the Lance. If this does not occur, it should reset the Lance and retry the test. This prevents a continuous transmission (babble) longer than the longest legal packet in case the Lance has become hung.

9. When the chip is in internal loopback mode and a CRC error is forced, a framing error will also be indicated along with the CRC error. In external loopback, when a CRC error is forced only that error is indicated; a framing error is indicated only if the Lance actually receives extra bits.

10. When transmit data chaining, a BUFF error will be set in the current transmit descriptor if a late collision or retry error occurred while the Lance was still transmitting data from the previous buffer. The BUFF error in this case is an invalid error indication and should be ignored. BUFF is valid only when UFLO is also set.

11. When the host program sets up a packet for transmission in chained buffers, it should set the OWN bits in all the transmit buffers *except the first one* (i.e. the one containing the STP bit), and then as its last act set the OWN bit in the first descriptor. Once that bit is set, the Lance will start packet transmission and may encounter an underflow error if the subsequent descriptors for the packet are not available.

12. Do not set INIT and STRT in NI_CSR0 at the same time. After stopping the chip, first set INIT and wait for IDON, then set STRT. If both are set at once, corrupt transmit or receive packets can be generated if RENA becomes true during the initialization process.

13. When a missed packet error occurs, the Lance chip must be halted using the stop bit and the re-loaded with it's initialization block. The stop bit must be set within 75 microseconds of the missed packet error or it may result in a silo pointer misalignment, which can cause transmission of the next packet to be bad.

CHAPTER 14

KA43 SCSI CONTROLLER

This chapter describes the SCSI controller portion of the KA43

The SCSI controller is provided to attach a TZ30 tape controller and/or RZxx disk drives to to a RigelMAX system. This attachment uses the ANSI Small Computer System Interface (SCSI) specification.

14.1 SCSI Overview

The SCSI electrical and logical interface and operation is described in detail in the ANSI draft standard issued by ANSI task group X3T9.2, and the particular subset of that standard used by the tape controller is described in the TZ30 specification. The programmer must use both of those documents in conjunction with this specification as his guide. This section reviews a few important features of the ANSI document to set the context for the following discussion of the RigelMAX implementation of the SCSI interface.

The SCSI interface is a bi-directional 8-bit-wide bus to which up to eight devices can be attached. The RigelMAX itself is one of those devices, so up to seven additional devices can be attached. Devices may play one of two roles: initiator or target. An initiator originates an operation by sending a command to a specific target. A target performs an operation which was requested by an initiator. In this specification it is assumed that the RigelMAX is always an initiator and that all other SCSI devices attached to it are targets. (There is, however, no *hardware* feature of the RigelMAX SCSI interface which prevents its sharing the bus with a second initiator or assuming the role of a target.)

Each device attached to the SCSI bus is identified by a unique device ID number in the range 0 through 7. During the arbitration, selection, and reselection bus phases in which an initiator and a target establish a connection, the device ID's of the initiator and target are both placed on the data bus by asserting the data bits corresponding to the device ID numbers. By convention, the ID number of the RigelMAX system is zero (this is controlled by the program which drives the SCSI interface; it is not fixed in RigelMAX hardware).

The electrical interface consists of 18 signal lines on a 50-pin connector. Some of these lines are driven only by initiators, some only by targets, and some by either. These lines are summarized below, using the signal names by which they are called in this specification and in the ANSI specification.

In this specification and in all the registers of the SCSI interface chip, the "true" or "asserted" value of a signal appears as a "one", and the "false" or "negated" value of a signal appears as a "zero". The bus electrical signals are all low true and are driven by open-collector drivers.

The SCSI bus is always terminated at each end. The bus is permanently terminated at the controller (near end). Far end termination can take place in one of several locations:

- At the 68 pin connector on the rear of the system enclosure.

- At the second 68 pin connector on a storage expansion unit.
- Within (internal) a non-DIGITAL SCSI drive.

NOTE

Note that the TZ30 tape drive and RZxx disk drives do not provide SCSI bus termination.

- DB7..0 and DBP comprise an 8-bit parallel data bus with an associated odd parity bit. The use of the parity bit is optional but strongly encouraged. These lines may be driven by either an initiator or a terminator, depending upon the direction of data transfer.
- RST signals all devices on the SCSI bus to reset to their initial power-on states. The system firmware will assert this signal at least once during power-on self-test. Thereafter, it should be asserted only as a last resort during error recovery since it indiscriminately affects all devices on the bus. An RST signal generated by some other device on the bus causes an internal reset of the 5380 chip and sets the interrupt request bit (INTREQ in register SCS_STATUS).
- BSY and SEL are used by initators and targets during the arbitration, selection, and reselection bus phases to establish or resume a logical connection between an initiator and a target. Once the connection is established, the target asserts BSY and the SEL signal is not driven by anyone.
- C/D, I/O and MSG collectively indicate one of six possible information transfer phases, according to the following table. These signals are always driven by the target device.

Table 59: SCSI Information Transfer Phases

MSG	C/D	I/O	Phase name	Transfer direction
0	0	0	Data out	to Target
0	0	1	Data in	to Initiator
0	1	0	Command	to Target
0	1	1	Status	to Initiator
1	0	0	(reserved)	
1	0	1	(reserved)	
1	1	0	Message out	to Target
1	1	1	Message in	to Initiator

- ATN is used by an initiator to signal a target that it has a message ready. The target can receive the message by entering the "message out" phase. ATN is always driven by an initiator.
- REQ and ACK are used to synchronize information transfers over the data bus during any of the six information transfer phases. REQ is always driven by the sender of the information after it has placed data on the DB7..0 and DBP lines. ACK is driven by the receiver of the information after it has captured it from the data lines. The RigelMAX system supports only "asynchronous" data transfer in which a sender may assert REQ only once before receiving ACK from the receiver. The "synchronous" option is not supported.

14.2 Controller Adapter Overview

The heart of the RigelMAX tape controller adapter is an NCR 5380 SCSI controller chip through which a host program can examine and manipulate all the SCSI signals. Associated with the chip is DMA logic which can transfer data between the SCSI bus and the disk data buffer. (See section <REFERENCE>(DISK_BUFFER\VALUE) for a description of the disk data buffer and its **compatibility** and **expanded** modes.) The normal method of operation is for the host program to do programmed data transfers for the command, status, and message phases which handle only a few bytes at a time, and to set up DMA transfers for the data phases.

The 5380 chip can be used by both initiator and target devices. In this specification, only its use as an initiator is described.

Section 14.3 below describes the registers which make up the SCSI chip. Section 14.4) describes the registers of the DMA logic which supports the SCSI chip. Section 14.5) describes the conditions under which the chip will signal an interrupt to the system processor.

14.3 SCSI Chip Registers

The SCSI chip appears to the system as a group of thirteen 8-bit registers which are addressed on longword boundaries. Nine of these registers contain data bits which can be read and/or written by a host program. The remaining four have no data bits but are "action" registers: when the host program reads or writes one of them (as specified), the SCSI chip is signalled to take some action, but the data bits are ignored.

Table 60: SCSI Chip Register Addresses

Address	Name	Access	Description
200C.0088	SCS_MODE	r/w	Mode register
200C.0084	SCS_INI_CMD	r/w	Initiator command register
200C.008C	SCS_TAR_CMD	r/w	Target command register
200C.0094	SCS_STATUS	ro	Bus and status register
200C.0090	SCS_CUR_STAT	ro	Current bus status register
200C.0090	SCS_SEL_ENA	wo	Select enable register
200C.0080	SCS_OUT_DATA	wo	Output data register
200C.0080	SCS_CUR_DATA	ro	Current data register
200C.0098	SCS_IN_DATA	ro	Input data register
200C.0094	SCS_DMA_SEND	wo	Start DMA send action
200C.009C	SCS_DMA_IRCV	wo	Start DMA initiator receive action
200C.0098	SCS_DMA_TRCV	wo	Start DMA target receive action
200C.009C	SCS_RESET	ro	Reset interrupt/error action

14.3.1 Mode Register (SCS_MODE)

The mode register is an 8-bit read/write register at physical address 200C.0088 which controls the operation of the chip. This register determines whether the system operates as an initiator or target, whether DMA transfers are being used, whether parity is checked for SCSI bus data, and whether interrupts are signalled for various conditions.

Figure 67: SCSI Mode Register(SCS_MODE)

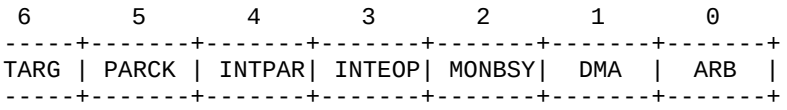


Table 61: SCSI Mode Register(SCS_MODE) Bit Assignments

Bit	Meaning
BLOCK	DMA block mode (bit 7).
	This bit controls the characteristics of the DRQ-DACK handshake between the 5380 chip and the DMA controller during DMA data transfers. For the RigelMAX system, this bit must always be zero.
TARG	Target role (bit 6).
	This bit determines whether the system performs the role of an initiator (TARG is zero) or target (TARG is one) on the SCSI bus. The RigelMAX system normally acts as an initiator, so this bit should normally be zero.
PARCK	Parity check enable (bit 5).
	This bit determines whether SCSI bus data parity errors will be ignored (PARCK is zero) or will set the PARERR bit in the SCS_STATUS register (PARCK is one).
INTPAR	Interrupt on parity error (bit 4).
	This bit determines whether parity errors detected on the SCSI bus will signal an interrupt. If INTPAR and PARCK are both one, then the occurence of a parity error will signal an interrupt.
INTEOP	Interrupt on end of DMA (bit 3).
	This bit determines whether an interrupt is generated at the end of a DMA transfer. If INTEOP is one, an interrupt is signalled when the DMA count register SCD_CNT (section 14.4.3) reaches zero.
MONBSY	Monitor BSY (bit 2).
	While this bit is one, the chip will signal an interrupt upon a loss of the bus BSY signal. When such an interrupt is generated, bits 5..0 of the SCS_INI_CMD register (bits DIFF, ACK, BSY, SEL, ATN, and ENOUT) are cleared to zero, which removes all signals generated by the system from the SCSI bus.
DMA	Enable DMA transfer (bit 1).

Table 61 (Cont.): SCSI Mode Register(SCS_MODE) Bit Assignments

Bit	Meaning
	This bit, when set to one, enables DMA transfers between the SCSI chip and the disk buffer. This bit must be set to zero for programmed data transfers. (Setting this bit to zero also clears the DMAEND bit in the SCS_STATUS register.) This bit must be set as part of the DMA initialization process which also includes initializing the DMA controller registers SCD_ADR, SCD_CNT, and SCD_DIR, and appropriately setting the ENOUT bit of the SCS_INI_CMD register. After this initialization, the host program must write the appropriate action register (SCS_DMA_SEND or SCS_DMA_IRCV) to begin actual data transfers. The DMA bit is not cleared when the DMA count register SCD_CNT reaches zero; it must be cleared by the host program. Once this bit is cleared, no further DMA data transfers will occur. NOTE: The host cannot reliably stop an ongoing DMA transfer by clearing this bit because the chip's data path may be in use for a DMA transfer, which thus interferes with host access to the chip's registers.
ARB	Start arbitration (bit 0). The host program sets this bit to one to start the bus arbitration process. Prior to setting this bit the program should load the system's device ID (conventionally bit 0) in the output data register SCS_OUT_DATA. The chip will wait for a bus-free condition before entering the arbitration phase. The results of the arbitration may be determined by reading bits AIP and LA of the initiator command register SCS_INI_CMD.

14.3.2 Initiator Command Register (SCS_INI_CMD)

The initiator command register is an 8-bit read/write register at physical address 200C.0084 which is used when the system is acting as an initiator (its normal role) to assert certain SCSI bus control signals, to monitor those signals, and to monitor the progress of bus arbitration. Bits 5 and 6 of this register have different definitions according to whether the register is read from or written to. Therefore, programs cannot use read-modify-write instructions such as BISB and BICB to access this register.

Figure 68: SCSI Initiator Command Register(SCS_INI_CMD)

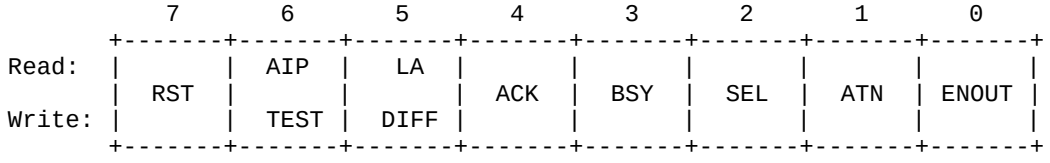


Table 62: SCSI Initiator Command Register Bit Assignments

Bit	Meaning
RST	Assert RST (bit 7, read/write). When this bit is changed from zero to one (by writing to the SCS_INI_CMD register), the RST signal is asserted, the chip interrupt request signal is asserted, and all the chip's internal logic and control registers are cleared (except for this bit and the interrupt request latch). While this bit is one, the RST signal is asserted on the SCSI bus. Writing a zero to this bit negates the RST signal. Reading this bit reflects only its value in the SCS_INI_CMD register and not necessarily the actual bus signal state.
AIP	Arbitration in progress (bit 6, read-only).

Table 62 (Cont.): SCSI Initiator Command Register Bit Assignments

Bit	Meaning
	This bit indicates, when one, that bus arbitration is in progress. In order for this bit to be set, the ARB bit in SCS_MODE must also be set. A one in the AIP bit indicates that a bus free condition has been detected and that the the chip has asserted BSY and the contents of the SCS_OUT_DATA register onto the SCSI bus. AIP will remain set until the ARB bit in SCS_MODE is cleared.
TEST	Test mode (bit 6, write-only).
	Setting this bit to one disables all the 5380 chip output drivers, effectively removing the chip from both the SCSI bus and the internal system bus. Note that setting this bit may generate spurious DMA requests or interrupts to the processor; the use of this bit is not recommended. This bit must be zero for normal operation.
LA	Lost arbitration (bit 5, read-only).
	This bit, when one, indicates that the chip detected a bus free condition, arbitrated for use of the bus by asserting BSY and the system's ID (in SCS_OUT_DATA) on the bus, but lost the arbitration because SEL was asserted by some other device on the bus. The LA bit can only be asserted while the ARB bit of SCS_MODE is set.
DIFF	Differential enable (bit 5, write-only).
	Must always be zero in this system.
ACK	Assert ACK (bit 4, read/write).
	While this bit is one, the ACK signal is asserted on the SCSI bus. This bit is effective only while the TARG bit of SCS_MODE is zero (i.e. when the system is acting as an initiator). Writing a zero to the ACK bit negates the ACK signal. Reading this bit reflects only its value in the SCS_INI_CMD register and not necessarily the actual bus signal state.
BSY	Assert BSY (bit 3, read/write).
	While this bit is one, the BSY signal is asserted on the SCSI bus. Asserting BSY indicates a successful selection or reselection. Writing a zero into the BSY bit negates the BSY signal, which indicates a bus disconnect condition. Reading this bit reflects only its value in the SCS_INI_CMD register and not necessarily the actual bus signal state.
SEL	Assert SEL (bit 2, read/write).
	While this bit is one, the SEL signal is asserted on the SCSI bus. SEL is normally asserted after arbitration has been successfully completed. Writing a zero into the SEL bit negates the SEL signal. Reading this bit reflects only its value in the SCS_INI_CMD register and not necessarily the actual bus signal state.
ATN	Assert ATN (bit 1, read/write).
	While this bit is one, the ATN signal is asserted on the SCSI bus. This bit is effective only while the TARG bit of SCS_MODE is zero (i.e. when the system is acting as an initiator). Writing a zero to the ATN bit negates the ATN signal. Reading this bit reflects only its value in the SCS_INI_CMD register and not necessarily the actual bus signal state.
ENOUT	Enable output (bit 0, read/write).
	This bit, when set to one, allows the contents of the SCS_OUT_DATA register to be sent out on the SCSI data bus. When operated as an initiator (i.e. the TARG bit of SCS_MODE is zero), the outputs are only enabled while the bus I/O signal is false and the three bus phase signals C/D, I/O and MSG match the contents of the corresponding bits in the SCS_TAR_CMD register. The ENOUT bit must be set to one during DMA operations which send data out to the SCSI data bus.

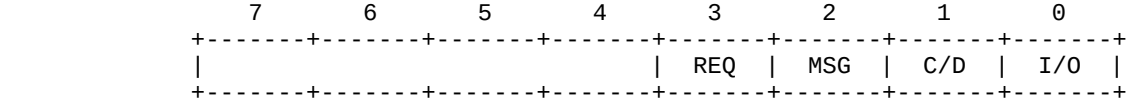
14.3.3 Target Command Register (SCS_TAR_CMD)

The target command register is an 8-bit read/write register at physical address 200C.008C. When the system is acting as an initiator (its normal role), this register is used during DMA data transfers (i.e. when bit DMA of the SCS_MODE register is one) to monitor the bus phase. When a target asserts REQ to request a data transfer, if the state of the bus

MSG, C/D and I/O signals does not match the values of those bits in this register, a "phase mismatch" interrupt is generated. This enables the host program to be notified when a DMA data transfer is ended by the target (this may occur prior to the DMA counter's reaching zero, since the target controls the length of data transfers). In initiator mode, the REQ bit in this register is ignored.

When the system is used as a target device (the TARG bit in SCS_MODE is one), this register allows a program to assert the REQ, MSG, C/D and I/O signals on the SCSI bus.

Figure 69: SCSI Target Command Register(SCS_TAR_CMD)



14.3.4 Bus and Status Register (SCS_STATUS)

The bus and status register is an 8-bit read-only register at physical address 200C.0094 which contains six chip status flags and monitors two of the SCSI bus control signals, ACK and ATN. (The other seven bus control signals are visible in the current bus status register SCS_CUR_STAT.)

Figure 70: SCSI Bus and Status Register(SCS_STATUS)

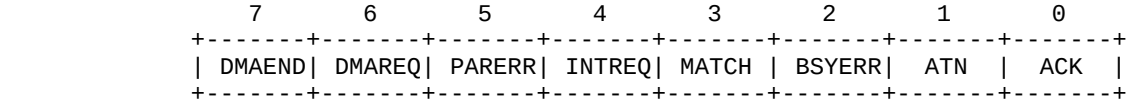


Table 63: SCSI Bus and Status Register(SCS_STATUS) Bit Assignments

Bit	Meaning
DMAEND	DMA end (bit 7). This bit is set when the DMA count register SCD_CNT becomes zero during a data transfer. After this bit is set, the chip will perform no additional DMA cycles. The DMAEND bit is cleared when the DMA bit in the SCS_MODE register is cleared.
DMAREQ	DMA request (bit 6). This pin reflects the status of the internal DMA request signal from the 5380 chip to the DMA controller. This bit becomes one when the chip requests the transfer of a byte to or from the disk buffer, and returns to zero when the DMA controller has performed the transfer.
PARERR	Parity error (bit 5). This bit is set upon receipt of a byte with incorrect parity from the SCSI data bus during a data transfer to the system or during device selection. PARERR will be set only if the PARCK bit of the SCS_MODE register has been set to one to enable parity checking; PARERR will not be set while PARCK is zero. The PARERR bit is cleared when the reset interrupt/error register SCS_RESET is read.
INTREQ	Interrupt request (bit 4).

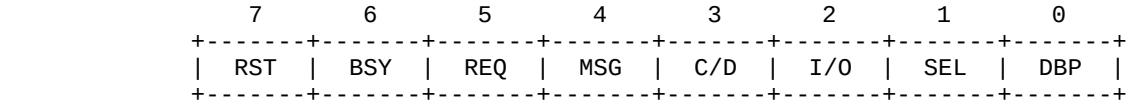
Table 63 (Cont.): SCSI Bus and Status Register(SCS_STATUS) Bit Assignments

Bit	Meaning
	This bit is set when any of the interrupt conditions described in section 14.5 occurs. It is cleared when the reset interrupt/error register SCS_RESET is read.
MATCH	Phase match (bit 3).
	This bit is one whenever the three SCSI bus phase signals MSG, C/D, and I/O match the values in the corresponding three bits of the target command register SCS_TAR_CMD. The MATCH bit is continuously updated and is only significant when the system is operating as an initiator (its usual mode). MATCH must be one for data transfers to occur on the SCSI bus.
BSYERR	Busy error (bit 2).
	This bit is set whenever the MONBSY bit of the mode register SCS_MODE is one and the SCSI bus BSY signal is false. This feature is used to monitor the bus for an unexpected loss of the logical connection between the system (as initiator) and a target device. When BSYERR is set, the DMA bit in the mode register SCS_MODE is cleared to stop any DMA data transfers, and the DIFF, ACK, BSY, SEL, ATN and ENOUT bits of the SCS_INI_CMD register are cleared to remove all signals generated by the system from the SCSI bus.
ATN	ATN signal (bit 1).
	This bit reflects the current state of the ATN signal on the SCSI bus.
ACK	ACK signal (bit 0).
	This bit reflects the current state of the ACK signal on the SCSI bus.

14.3.5 Current Bus Status Register (SCS_CUR_STAT)

The current bus status register is an 8-bit read-only register at physical address 200C.0090 which is used to monitor seven of the SCSI bus control signals plus the data bus parity bit. (The other two bus control signals, ACK and ATN, are in the Bus and Status register SCS_STATUS). The host program uses this register to determine the current bus phase and to poll REQ during programmed data transfers from the system to a target device. This register is also used to help determine why a particular interrupt occurred.

Figure 71: SCSI Current Bus Status Register(SCS_CUR_STAT)

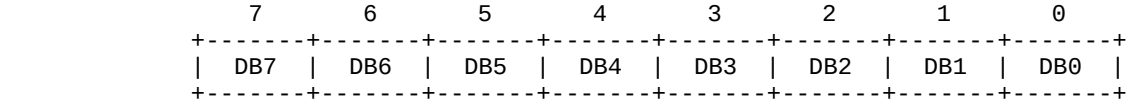


14.3.6 Select Enable Register (SCS_SEL_ENA)

The select enable register is an 8-bit write-only register at physical address 200C.0090 which contains the device ID which the device should recognize as its own during a selection or reselection attempt. For a RigelMAX system whose device ID is normally zero this register should contain a one in bit zero and zeros elsewhere.

The simultaneous occurrence of the correct ID bit on the data bus, BSY false, and SEL true (i.e. a selection or reselection phase) will generate an interrupt signal. Such interrupts can be disabled by writing all zeros into this register. If parity checking is enabled (the PARCK bit in the mode register SCS_MODE is set), the parity of the data on the data bus is checked during selection or reselection.

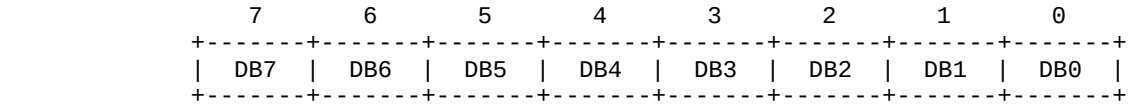
Figure 72: SCSI Select Enable Register(SCS_SEL_ENA)



14.3.7 Output Data Register (SCS_OUT_DATA)

The output data register is an 8-bit write-only register at physical address 200C.0080 which is used to send outgoing data to the SCSI bus. It is used during programmed I/O to write outgoing data bytes and to assert the proper ID bits on the SCSI bus during arbitration and selection phases. (This register is also implicitly used by the hardware during DMA transfers to the SCSI bus.)

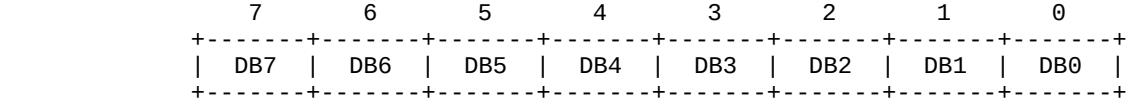
Figure 73: SCSI Output Data Register(SCS_OUT_DATA)



14.3.8 Current Data Register (SCS_CUR_DATA)

The current data register is an 8-bit read-only register at physical address 200C.0080. Its contents reflect the data currently on the data lines of the SCSI bus. It is used during programmed (rather than DMA) I/O to read incoming data bytes and during arbitration to check for higher priority arbitrating devices.

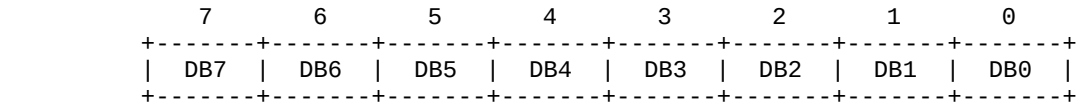
Figure 74: SCSI Current Data Register(SCS_CUR_DATA)



14.3.9 Input Data Register (SCS_IN_DATA)

The input data register is an 8-bit read-only register at physical address 200C.0098 which is used to read latched data from the SCSI bus during DMA operation. (During programmed I/O operation no data is latched in this register. The SCS_CUR_DATA register should be used instead for programmed I/O.) The input data register is implicitly used by the hardware during DMA transfers from the SCSI bus. When the system is acting as an initiator (its normal role), data is latched when the bus REQ signal is asserted by the target device. When the system is acting as a target, data is latched when the bus ACK signal is asserted.

Figure 75: SCSI Input Data Register(SCS_IN_DATA)



14.3.10 Start DMA Send Action (SCS_DMA_SEND)

The start DMA send action register is an 8-bit write-only register at physical address 200C.0094. The act of writing to this register (the data written is ignored) begins DMA transfers from the RigelMAX system disk buffer to a target device. Prior to writing to this register, the DMA controller registers SCD_ADR and SCD_CNT must be loaded, the DIR bit of the SCD_DIR register must be set to zero, the ENOUT bit of the SCSI_INI_CMD register must be set to one, and the DMA bit of the SCS_MODE register must be set to one.

14.3.11 Start DMA Initiator Receive Action (SCS_DMA_IRCV)

The start DMA initiator receive action register is an 8-bit write-only register at physical address 200C.009C. The act of writing to this register (the data written is ignored) begins DMA transfers from a target on the bus to the RigelMAX system disk buffer, when the RigelMAX system is acting as an initiator device (its normal role). Prior to writing to this register, the DMA controller registers SCD_ADR and SCD_CNT must be loaded, the DIR bit of the SCD_DIR register must be set to one, the ENOUT bit of the SCSI_INI_CMD register must be set to zero, and the DMA bit of the SCS_MODE register must be set to one.

14.3.12 Start DMA Target Receive Action (SCS_DMA_TRCV)

The start DMA target receive action register is an 8-bit write-only register at physical address 200C.0098. The act of writing to this register (the data written is ignored) begins DMA transfers from an initiator on the bus to the RigelMAX system disk buffer, when the RigelMAX system is acting as a target device (not its normal role). Prior to writing to this register, the DMA controller registers SCD_ADR and SCD_CNT must be loaded, the DIR bit of the SCD_DIR register must be set to one, the ENOUT bit of the SCSI_INI_CMD register must be set to zero, and the DMA bit of the SCS_MODE register must be set to one.

14.3.13 Reset Interrupt/Error Action (SCS_RESET)

The reset interrupt/error action register is an 8-bit read-only register at physical address 200C.009C. The act of reading this register clears bits PARERR (parity error), INTREQ (interrupt request), and BSYERR (busy error) in the bus and status register SCS_STATUS. No useful data is returned from reading this register.

14.4 DMA Operation

14.4.1 _Compatibility mode.

To accomodate software written for VS410 systems, the disk data buffer, DMA address register, and DMA count registers can be operated in **compatibility mode**. In this mode the apparent size of the buffer is reduced to 16 Kb to all parties (disk controller, SCSI controller, and CPU) and it is accessed at physical addresses 200D.0000 through 200D.3FFF (this is an alias for the normal address range of 202D.0000 through 202D.3FFF; that range

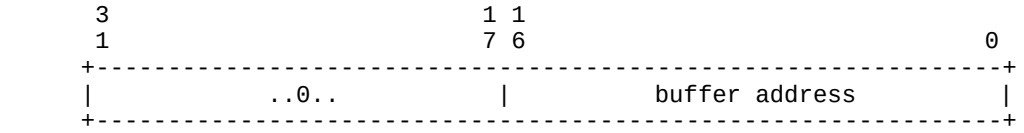
is also usable in compatibility mode). In compatibility mode, DMA addresses generated by the SCD_ADR register wrap at the 16 Kb boundary, from 003FFF to 000000.

Compatibility mode is selected by the STC_MODE register, which is described in section <REFERENCE>(STC_MODE_REG\VALUE).

14.4.2 DMA Address Register (SCD_ADR)

The DMA address register is a 32-bit read/write register at physical address 200C.00A0. It is is used to set the starting address in the disk buffer for the next DMA transfer.

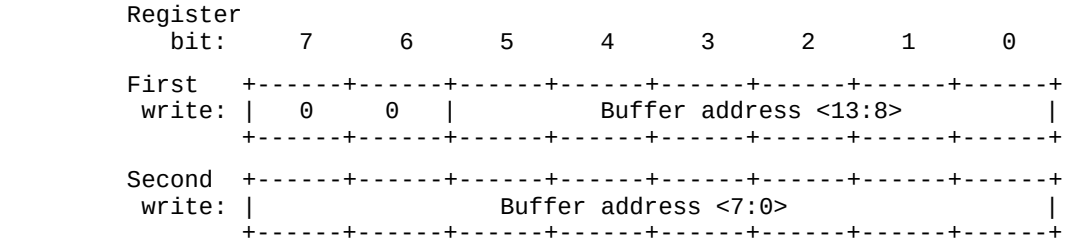
Figure 76: SCSI DMA Address Register(SCD_ADR)



Bits 16:0 address the disk data buffer, which is at physical addresses 202D.0000 through 202E.FFFF. Bits 31:17 return zeros upon reading and are ignored upon writing. Only long-word accesses should be used. This register must not be accessed while a DMA operation is either pending or in progress for the SCSI controller (reading of this register is harmless).

In *compatibility mode*, the buffer is limited to 16 Kbytes which requires only 14 address bits. The required 14-bit starting address must be loaded by two consecutive byte writes to SCD_ADR. The first write sets bits <13:8> of the starting address from bits <5:0> of the data byte; the second write sets address bits <7:0> from bits <7:0> of the second data byte. Bits <16:14> of SCDADD are cleared to zero. This loading pattern is illustrated below:

Figure 77: SCD_ADR Compatibility Loading



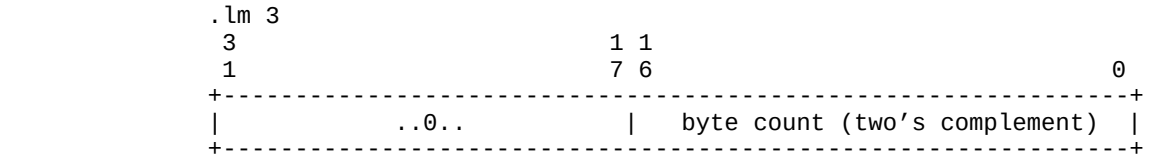
14.4.3 DMA Count Register (SCD_CNT)

The DMA count register is a 32-bit read/write register at physical address 200C.00C0. Its 17-bit counter counts the number of bytes transferred during a DMA operation and signals the SCSI controller chip when the specified number of bytes have been transferred. This register should be loaded with the 17-bit two's complement of the maximum number of bytes to be transferred by the next DMA transfer between the SCSI chip and the disk data buffer. (This counter is not used for and is not affected by DMA operations between the disk controller (described in chapter <REFERENCE>(DISK_CHAPTER\VALUE)) and the disk buffer.)

Only longword write accesses should be used when the controller is in expanded mode. When read, bits 31:17 return zeros. When written, bits 31:17 are ignored.

In *compatibility mode* bit 17 is forced to one on any write, and word write accesses may be used.

Figure 78: SCSI DMA Count Register(SCD_CNT)



As each byte is transferred, SCD_CNT is incremented by one. When SCD_CNT changes from minus one to zero, a counter overflow bit is set and the SCSI chip is signalled via its EOP pin to terminate DMA operation at the completion of the current byte transfer (i.e. the transfer during which the counter became zero). This sets the DMAEND bit in the SCS_STATUS register.

While the counter overflow bit is set, the DMA controller will not perform data transfers. The counter overflow bit is cleared whenever the count register is loaded by writing to SCD_CNT. If a transfer request is pending (i.e. the DMAREQ bit of the SCS_STATUS register is true) when the counter is loaded, a transfer will occur at once. Therefore, when restarting a DMA transfer, the host program must first load SCD_ADR and SCD_DIR and then load SCD_CNT with a single word write.

After a DMA operation ends (either because SCD_CNT reached zero or because the SCSI chip sensed a bus phase change), the host program may read SCD_CNT to get the number (in two's complement form) of bytes not transferred. Adding this to the true form of the count originally loaded into SCD_CNT will give the number of bytes actually transferred. This register must not be read or written while an SCSI DMA operation is either pending or in progress.

Upon power-on SCD_CNT and its overflow bit are cleared to zero.

14.4.4 DMA Direction Register (SCD_DIR)

The DMA direction register is an 8-bit read/write register at physical address 200C.00C4. It controls the direction in which data is transferred during DMA cycles requested by the SCSI chip.

Figure 79: SCSI DMA Direction Register (SCD_DIR)

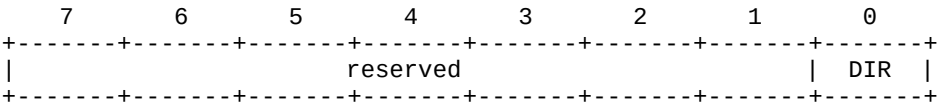


Table 64: SCSI DMA Direction Register (SCD_DIR)

Bit	Meaning
<7:1>	Reserved. Read as zeros. Must always be written as zeros.
DIR	Transfer direction (bit 0). When this bit is one, DMA cycles transfer data from the SCSI bus into the disk buffer (a READ operation). When this bit is zero, DMA cycles transfer data from the disk buffer to the SCSI bus (a WRITE operation). Upon power-on, DIR is cleared to zero.

14.5 Interrupts

The 5380 chip has one interrupt request signal which is sent to the CPU via the system interrupt controller (described in chapter 9). The state of this signal is visible in the INTREQ bit of the SCS_STATUS register. An interrupt request can be signalled by any of the following six events:

1. the controller is selected or reselected by another device on the SCSI bus;
2. the DMA count register reaches zero;
3. a parity error is detected during data transfer;
4. a bus phase mismatch occurs;
5. an SCSI bus disconnect occurs;
6. the RST signal is asserted on the SCSI bus.

When the host program responds to the interrupt, it must use the contents of the SCS_STATUS and SCS_CUR_STAT registers to determine what condition(s) caused the interrupt, and then either clear the interrupt causing condition or mask off that interrupt condition. Each of the six interrupt causes listed above, except receipt of the RST signal, can be individually masked by appropriate settings of the 5380 registers, as described in the paragraphs below.

Once it has serviced the interrupt, the host program must read the SCS_RESET register to reset the INTREQ bit. Note that if there is another unmasked interrupt causing condition, or if the original condition persists and has not been masked off, then reading SCS_RESET will generate another transition on the interrupt request line to the interrupt controller.

In order for the 5380 interrupt signal to cause a CPU interrupt, the SC bit of the interrupt mask register INT_MSK must be set (see section 9.5). Section 9.6 lists the value of the SCSI controller's interrupt vector.

14.5.1 Selection or Reselection

An interrupt can be signalled when another device on the SCSI bus attempts to select or reselect the RigelMAX system. (In system's normal role of an initiator, it may be reselected by a device to which it has previously issued a command. Selection is appropriate only if the system is acting as a target.) Such an interrupt occurs when the following conditions are simultaneously met:

- the SEL signal is true;
- the BSY signal is false for at least a bus settle delay (400 nanosec);
- the bitwise AND of the bits DB7..0 on the SCSI data bus with the corresponding bits of the select enable register SCS_SEL_ENA is non-zero.

The interrupt service routine can identify this case by noting that BSY is zero and SEL is one in the SCS_CUR_STAT register. If the I/O bit of SCS_CUR_STAT is zero, this is a select attempt; otherwise it is a reselect. The SCS_SEL_ENA register should contain a one only in the bit corresponding to the system's SCSI device ID (normally bit 0) and zeros in the other seven bits. Only two bits should be asserted on the SCSI data bus during selection or reselection: the ID of the initiator and the ID of the target. The host program should check this by examining the data bus via the SCS_CUR_DATA register. In addition, if bus parity is enabled (bit PARCK of SCS_MODE is true), then the parity error bit PARERR in SCS_STATUS should be tested as well.

Selection and reselection interrupts can be prevented by setting all the bits of SCS_SEL_ ENA to zero.

14.5.2 DMA Count Reaches Zero

An interrupt can be signalled when the DMA count register SCD_CNT reaches zero during a DMA transfer. Such an interrupt occurs when the following conditions are simultaneously met:

- the DMA bit in the SCS_MODE register is set;
- a DMA transfer to or from the disk buffer occurs, during which the count register SCD_ CNT reaches zero;
- the INTEOP bit in the SCS_MODE register is set.

If the first two conditions are satisfied, the DMAEND bit in the SCS_STATUS register is set. If all three conditions are satisfied, INTREQ is also set in SCS_STATUS.

Note that when DMAEND is set, the system DMA controller performs no additional transfers, but the target's block transfer is not necessarily complete. When the system is operating as an initiator, it must test the MATCH bit in SCS_STATUS to determine when the target has completed its data block. In addition, when sending data to the target, it must monitor REQ in SCS_CUR_STAT and ACK in SCS_STATUS until both are false to be sure that the last byte has been transferred.

14.5.3 Bus Parity Error

The chip can signal an interrupt when it detects invalid parity in data received from the SCSI bus. Such an interrupt is generated when:

- the PARCK bit in the SCS_MODE register is set;
- invalid parity is detected during a DMA transfer from the SCSI bus to the disk buffer, or during a processor read of the SCS_CUR_DATA register;
- the INTPAR bit in the SCS_MODE register is set.

If the first two conditions are satisfied, the PARERR bit in the SCS_STATUS register is set. If all three conditions are satisfied, INTREQ is also set in SCS_STATUS.

14.5.4 Phase Mismatch

The chip can signal an interrupt whenever the three bus phase signals MSG, C/D, and I/O do not match the corresponding bits in the SCS_TAR_CMD register during a DMA data transfer. The match state is continuously reflected in the MATCH bit of the SCS_STATUS register. The interrupt is signalled when:

- the MATCH bit in SCS_STATUS is zero;
- the DMA bit in SCS_MODE is one;
- the bus REQ signal is asserted to request a data transfer.

The identify status for such an interrupt are that DMA is one in SCS_MODE and that DMAEND and MATCH are both zero in SCS_STATUS.

14.5.5 Bus Disconnect

The chip can generate an interrupt when the SCSI bus BSY signal becomes false. Such an interrupt is generated when:

- the MONBSY bit in SCS_MODE is one;
- the bus BSY signal goes false for at least 400 nanosec.

This condition sets the BSYERR bit in the SCS_STATUS register.

14.5.6 SCSI Bus Reset

The chip generates an interrupt whenever the RST signal on the SCSI bus is asserted, either by another device on the bus or when the host program sets the RST bit in the SCS_INI_CMD register. When this occurs, the chip releases all bus signals within 800 nanosec. This interrupt cannot be disabled.

Note that the RST signal is not latched in the SCS_CUR_STAT register, so the RST bit may not still be set when the host responds to an interrupt which was caused by RST from another device on the bus.

14.6 Reset Conditions

There are three possible reset conditions for the 5380 chip.

14.6.1 System Hardware Reset

Upon system power on or when the system I/O reset signal is generated (see section 8.1), the 5380 chip is reinitialized and all internal logic and control registers are cleared. All signals are removed from the SCSI bus. This does NOT assert the RST signal on the SCSI bus.

14.6.2 RST Received from SCSI Bus

When the RST signal is asserted on the bus by some other SCSI device, all the 5380 internal logic and registers are cleared, except that the interrupt request signal is asserted (bit INTREQ of SCS_STATUS) and the RST bit of the SCS_INI_CMD register is not altered.

14.6.3 RST Issued to SCSI Bus

When the host program asserts RST on the SCSI bus by setting the RST bit in the SCS_ INI_CMD register, all the 5380 internal logic and registers are cleared, except that the interrupt request signal is asserted (bit INTREQ of SCS_STATUS) and the RST bit of the SCS_INI_CMD register remains asserted until cleared by the host program or by a system hardware reset.

14.7 SCSI Programming Notes

This section contains assorted hints and kinks that programmers should heed when writing drivers for the SCSI controller.

14.7.1 Device ID Values

The following SCSI device ID numbers are assigned by convention to the RigelMAX CPU and an attached TZK50 tape controller. For the CPU this is done in its device driver code. The tape controller requires that jumpers be set on its board, as described in the TZK50 specification.

Table 65: SCSI Device ID Values

Device ID number ID bits		
CPU	0	01h
TZK50	1	02h

In addition, both the CPU and TZK50 should assert and test parity on the SCSI bus. This is done for the CPU by asserting the PARCK bit in the SCS_MODE register (section 14.3.1). The tape controller requires that a jumper be set on its board, as described in the TZK50 specification.

CHAPTER 15

OTHER OPTIONS

This chapter discusses the configuration rules for options which are to be connected to a RigelMAX system via the video option connector. It also summarizes three specific option boards which are covered by separate specifications.

15.1 General Rules for Options

In order to avoid hardware conflicts and to accommodate operating system configuration and device driver support constraints, each option must conform to a set of rules which specify:

- its firmware ROM address range;
- its control and status register address range;
- which interrupts it uses.

As discussed in section 2.3, every option board must have a ROM to contain its device type identifier and diagnostic program. The device type ID values are specified in the RigelMAX Power On Initialization Specification. Whenever a new option is designed, a new device type ID value must be assigned and added to that specification.

An option's control and status registers (CSR's) must begin at the beginning of the address range specified by the rule; they may extend as far as necessary within that range and may occupy discontiguous subranges.

Three of the eight interrupts discussed in chapter 9 may be used by options connected to the video option connector: NS, VF, and VS. VF is also used by the monochrome video controller; an option which uses that interrupt requires that its device driver set the VDC_SEL register to switch the interrupt signal to the video option connector. Conflicts over the use of these interrupts may constrain which options can coexist in a particular system.

15.1.1 Video Option Rules

A video option board occupies a 10.4 by 7.6 inch board which is plugged into the video option board connectors. Such an option must fit one of the following three sets of rules:

Firmware ROM CSR range Interrupts

2014.0000 - 2017.FFFF 3800.0000 - 3FFF.FFFF}
201C.0000 - 201F.FFFF 2400.0000 - 25FF.FFFF} see below
2018.0000 - 201B.FFFF 2200.0000 - 23FF.FFFF}

A video option board may use any one of the following combinations of interrupts:

- VF
- VS
- VF and VS
- NS 8.2

Note that when interrupt VF is used by an option, that option's device driver must set bit I3OPT of the VDC_SEL register to one, which will disable the end-of-frame interrupt from the monochrome video controller.

15.2 Specific Options

These specific options have been defined for the RigelMAX:

- VS43G-AA eight-plane color video controller

This option is described in separate specifications; they are briefly summarized below.

For the convenience of operating system automatic configuration programs, the following table summarizes the attributes of the presently defined RigelMAX options:

Table 66: RigelMAX Options

Option	ROM base	Dev ID	CSR base	Rupts	VDC-SEL
Storage	2010.0000	0090h	200C.0000	DC,SC	
8-plane color	2014.0000	00D0h	3C00.0000	VF,VS	yes

15.2.1 Eight-plane Color Video Option

The VS43G-AA color video option is an eight-plane system using the Scanproc chip set. It has no external connector; its red, green, and blue video signals appear on the main board video connector.

The color video option uses the following physical addresses:

Table 67: Color Video Option Addresses

2014.0000-2015.FFFF	Firmware ROM (one 32 Kbyte chip)
3C00.0000-3C00.007F	ADDER chip registers
3C00.0200-3C00.02FF	FIFO compression chip registers
3C00.0300-3C00.037F	Video DAC registers
3C00.0400-3C00.041F	Cursor chip registers
3C00.0500-3C00.0501	Video readback register
3C00.8000-3C00.FFFF	FIFO/template RAM

The color video option uses the following interrupts and requires that the interrupt source register bit VDC_SEL<I3OPT> be one (which preempts the monochrome video end-of-frame interrupt):

- VF Adder chip interrupt
- VS FIFO compression chip interrupt

Installing this board asserts the VIDOPT bit in the CFGTST register.

APPENDIX A

KA43 PHYSICAL ADDRESS MAP

A.1 System Board Addresses

The addresses used by hardware on the KA43 system board and the MS43-AA memory option board are as follows:

Table 68: System Hardware Address Map

0000.0000-01FF.FFFF		System board RAM
1000.0000-1FFF.FFFF		Second level cache data storage
2002.0000	CFGTST	Config & test register (r/o)
2002.0000	IORESET	IO reset register (w/o)
2004.0000-2007.FFFF		System board ROM (256 Kb)
2004.0004	SYS_TYPE	System ID extension register
2004.0020-2004.003F		Interrupt vector numbers
2008.0000	HLTCOD	Halt code register
2008.0004		reserved
2008.0008		reserved
2008.000C	INT_MSK	Interrupt mask register
2008.000D	VDC_ORG	Monochrome display origin
2008.000E	VDC_SEL	Video interrupt select
2008.000F	INT_REQ	Interrupt request register (r/o)
2008.000F	INT_CLR	Interrupt request clear (w/o)
2008.0010	DIAGDISP	Diagnostic display (w/o)
2008.0014	PAR_CTL	Parity control
2008.001E	DIAGTIME	Diagnostic timer
2009.0000-2009.007F		Network address ROM
200A.0000-200A.000F		SER_xxx Serial line controller
200B.0000-200B.00FF		WAT_xxx Time-of-year clock and NV RAM
200C.0000-200C.00FF	Storage controller CSR's	
200D.0000-200D.FFFF	Storage controller buffer (compatibility range)	
202D.0000-202E.FFFF	Storage controller buffer	
200E.0000-200E.0007		Lance chip registers
200F.0000-200F.003F	CUR_xxx	Monochrome video cursor chip

Table 68 (Cont.): System Hardware Address Map

2100.0000-2101.FFFF		Second level cache tag storage
2110.0000-2110.0007	SESR	System Error Status Register
2800.0000-29FF.FFFF		Memory diagnostic space
3000.0000-3003.FFFF		Monochrome video RAM (256 Kb)

A.2 Option Board Address Ranges

The following address ranges are defined for use by option boards connected to the storage controller option and video option connectors. This section lists the nominal ranges; subsequent sections show the actual ranges used by each option type.

Table 69: Option Board Address Map

2200.0000-23FF.FFFF	Future option CSR's
2400.0000-25FF.FFFF	Future option CSR's
2010.0000-2013.FFFF	Storage controller option ROM
2014.0000-2017.FFFF	Video option ROM
2018.0000-201B.FFFF	Additional option 1 ROM
201C.0000-201F.FFFF	Additional option 2 ROM
3800.0000-3BFF.FFFF	Video option (32-bit path)
3C00.0000-3C00.FFFF	Video option (16-bit path)

A.2.1 Storage Controller Option Addresses

The following addresses are used by the SCSI storage controller described in a separate specification.

Table 70: Storage Controller Address Map

200C.0000-200C.0007	DKC_xxx Disk controller chip ports
200C.0080-200C.009F	SCS_xxx SCSI controller chip
200C.00A0	SCD_ADR SCSI DMA address register
200C.00C0	SCD_CNT SCSI DMA byte count register
200C.00C4	SCD_DIR SCSI DMA transfer direction
200C.00E0	STC_MODE Storage controller mode
200D.0000-200D.3FFF	Data buffer (16 Kb compatibility range)
202D.0000-202E.FFFF	Data buffer (128 Kb full range)
2010.0000-2011.FFFF	Firmware ROM (one 32 Kbyte chip)

A.2.2 Color Video Option Addresses

The following addresses are used by the Scanproc 8-plane color video option.

Table 71: Color Video Address Map

2014.0000-2015.FFFF	Firmware ROM (one 32 Kbyte chip)
3C00.0000-3C00.007F	ADDER chip registers
3C00.0200-3C00.02FF	FIFO compression chip registers
3C00.0300-3C00.037F	Video DAC registers
3C00.0400-3C00.041F	Cursor chip registers
3C00.0500-3C00.0501	Video readback register
3C00.8000-3C00.FFFF	FIFO/template RAM

APPENDIX B

VAX INSTRUCTION SET

B.1 VAX Instruction Set

The information in this appendix is excerpted directly from the RIGEL 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  =    d_floating
f  =    f_floating
g  =    g_floating
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
```

Integer 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 -	
AD	XORW3 mask.rw, src.rw, dst.ww	* * 0 -	

Address Instructions

Opcode	Instruction	N Z V C	Exceptions
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}	* * 0 -	
3F	PUSHAW src.aw, {-(SP).wl}	* * 0 -	

Variable Length Bit Field Instructions

Opcode	Instruction	N Z V C	Exceptions
EC	CMPV pos.rl, size.rb, base.vb, {field.rv}, src.rl	* * 0 *	rsv
ED	CMPZV pos.rl, size.rb, base.vb, {field.rv}, src.rl	* * 0 *	rsv
EE	EXTV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	* * 0 -	rsv
EF	EXTZV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	* * 0 -	rsv
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	N Z V C	Exceptions
5C	INSQHI entry.ab, header.aq	0 * 0 *	rsv
5D	INSQTI entry.ab, header.aq	0 * 0 *	rsv
0E	INSQUE entry.ab, pred.ab	* * 0 *	
5E	REMQHI header.aq, addr.wl	0 * * *	rsv
5F	REMQTI header.aq, addr.wl	0 * * *	rsv
0F	REMQUE entry.ab, addr.wl	* * * *	

Operating System Support Instructions

Opcode	Instruction	N Z V C	Exceptions
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*} Where y=MINU(x, PSL<CURRENT_MODE>)	0 0 0 0	
06	LDPCTX {PCB.r*, -(KSP).w*}	- - - -	rsv, prv
DB	MFPR procreg.rl, dst.wl	* * 0 -	rsv, prv
DA	MTPR src.rl, procreg.rl	* * 0 -	rsv, prv
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 KA670 floating point accelerator.

Opcode	Instruction	N Z V C	Exceptions
60	ADDD2 add.rd, sum.md	* * 0 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	
68	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
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
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 KA670 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, sumlen.rw, sumaddr.ab	* * * 0	rsv, dov
F8	ASHP cnt.rb, srclen.rw, srcaddr.ab, round.rb, dstlen.rw, dstaddr.ab	* * * 0	rsv, dov
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, dstlen.rw, dstaddr.ab	* * * 0	rsv, dov
26	CVTTP srclen.rw, srcaddr.ab, tbladdr.ab, dstlen.rw, dstaddr.ab	* * * 0	rsv, dov
27	DIVP divrlen.rw, divraddr.ab, divdlen.rw, divdaddr.ab, quolen.rw, quoaddr.ab	* * * 0	rsv,dov,ddvz
38	EDITPC srclen.rw, srcaddr.ab, pattern.ab, dstaddr.ab	* * * *	rsv, dov
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, dstlen.rw, dstaddr.ab	* * 0 *	
2F	MOVTUC srclen.rw, srcaddr.ab, esc.rb, tbladdr.ab, dstlen.rw, dstaddr.ab	* * * *	
25	MULP mulrlen.rw, mulraddr.ab, muldlen.rw, muldaddr.ab, prodlen.rw, prodaddr.ab	* * * 0	rsv, dov
22	SUBP4 sublen.rw, subaddr.ab, diflen.rw, difaddr.ab	* * * 0	rsv, dov
23	SUBP6 sublen.rw, subaddr.ab, minlen.rw, minaddr.ab, diflen.rw, difaddr.ab	* * * 0	rsv, dov