KA660 CPU Module Technical Manual

Order Number: EK-KA660-TM-001

Digital Equipment Corporation

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

This document was prepared with VAX DOCUMENT, Version 1.2.

Preface

The *KA660-AA CPU Module Technical Manual* documents the functional, physical and environmental characteristics of both variations of the KA660 CPU module, and includes information on the MS650 memory expansion modules. The KA660-AA is for a multiuser environment. The KA660-BA is for a single user environment and does not support multiuser VMS or ULTRIX operating system licenses.

This manual is intended for a design engineer or applications programmer who is familiar with Digital's extended LSI-11 bus (Q22-bus) and the VAX instruction set. This manual should be used along with the *VAX Architecture Reference Manual* as a programmer's reference to the module.

The manual is divided into twelve chapters and eleven appendices.

Chapter 1, **Overview,** introduces the KA660 Subsystem including the KA660-AA CPU module, the MS650 memory module, and the H3602 console module.

Chapter 2, **Installation and Configuration,** describes procedures for installing and configuring the CPU, the memory, and the console modules in the Q22-bus backplanes and system enclosures.

Chapter 3, **Central Processor** provides information about the SOC central processor and basic VAX architecture. This chapter lists all the internal processor registers used in the KA660 processor design. Some information on error handling is given also.

Chapter 4, **KA660 Cache Memory** describes the organization of the KA660 cache. It shows the format of cache entries provides information on cache address translation.

Chapter 5, **KA660 Main Memory** provides information about the organization of the main memory including the registers associated with main memory error checking and status and includes information on cycle access times.

Chapter 6, **KA660 Console Serial Line** describes the serial line interface and its associated registers on the KA660.

Chapter 7, **KA660 Clock and Timer Registers** provides information about the VAX standard TOY clock and timers.

Chapter 8, **KA660 Boot and Diagnostic Facility** describes the KA660 initialization process and provides information about Boot and Diagnostic Registers and the EPROM memory resident firmware.

Chapter 9, **KA660 Q22-bus Interface** describes the Q22-bus interface which is implemented with the CQBIC. Information on Q22-bus address translation is provided along with descriptions of all Q22-bus interface registers. Some information on CQBIC error handling is provided.

Chapter 10, **KA660 Network Interface** provides information on the SGEC chip and the logic that supports the Ethernet network interface. This chapter provides an over of Ethernet principles with descriptions of packet format and programming instructions. The transmission and reception processes are described also.

Chapter 11, **KA660 Mass Storage Interface** describes how the Single Host Adapter Chip (SHAC) provides a DSSI mass storage interface for the KA660. An overview is provided as well as descriptions of all the registers associated with the DSSI interface.

Chapter 12, **KA660 Firmware** describes the functional firmware that is located in the EPROMS. The services provided by the firmware are described.

Appendix A, **Q22-bus Specification,** describes the low-end member of Digital's bus family. All of Digital's microcomputers, such as the MicroVAX I, MicroVAX II, MicroVAX 3500, MicroVAX 3600, MicroPDP–11, use the Q22-bus.

Appendix B, **Specifications,** describe the physical, electrical, and environmental characteristics of the KA660-AA CPU module.

Appendix C, **Address Assignments,** provides a map of VAX memory space.

Appendix D, **VAX Instruction Set,** is a list of the VAX instructions.

Appendix E, **Machine State on Power-up,** describes the state of the KA660 after a power-up halt.

Appendix F, **Maintenance Operation Protocol Support,** describes the Maintenance Operation Protocol (MOP)support features in the KA660 firmware.

Appendix G, **ROM Partitioning,** describes the ROM partitioning and subroutine entry points that are public and guaranteed to be compatible over future versions of the KA660 firmware.

Appendix H, **BBURAM Partitioning,** describes how the KA660 firmware partitions the 1KB of battery backed up RAM.

Appendix I, **Data Structures,** describes the global data structures that are used by the KA660 firmware.

Appendix J, **Error Messages** list the firmware error messages detected by the KA660.

Appendix K , **Glossary of Terms and Acronyms,** provides a list of the acronyms and terms used in this manual.

Conventions

The following table lists the conventions used in this manual.

Table 1: Conventions

Convention	Meaning
<x:y>	Represents a bit field, a set of lines, or signals, ranging from x through y. For example, R0 <7:4> Indicates bits 7 through 4 in a general purpose register R0.
[x:y]	Represents a range of bits, from y through x.
$\boxed{\text{Return}}$	A label enclosed in a box represents a key (usually a control or a special character key) on the keyboard (in this case, the carriage return key.)
Note	Contains general information.
Caution	Contains information to prevent damage to equipment.
n	Boldface small n indicates a variable.
{}	Represents a console command element
[ ]	Represents a console command element that is optional
...	Represents a list command elements

Related Documents

The following documents are related to the KA660 CPU.

- Microcomputer Interfaces Handbook (EB-20175-20)
- Microcomputers and Memories Handbook (EB-18451-20)
- VAX Architecture Handbook (EB-19580-20)
- VAX–11 Architecture Reference Manual (EK-VAXAR-RM)

You can order these documents from Digital Equipment Corporation, at the address given below.

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

Attention: Documentation Products

Contents

FIGURES

Chapter 1

OVERVIEW

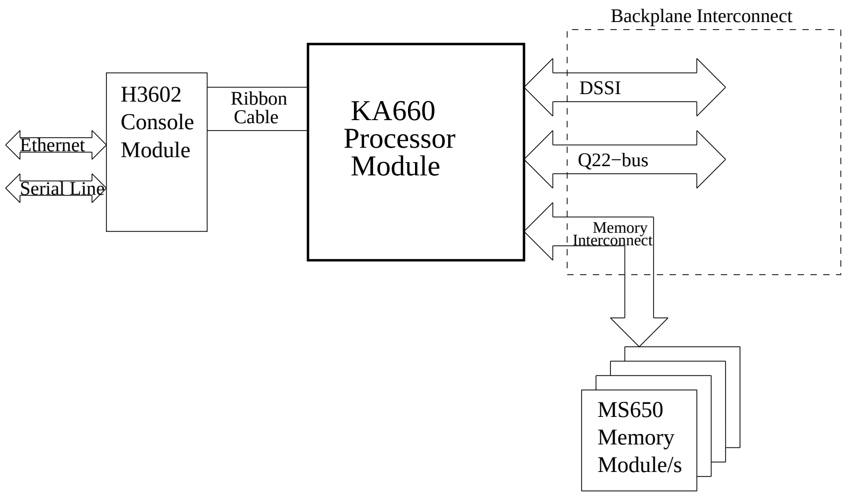
This chapter provides a brief description of the KA660 CPU/Memory Subsystem.

1.1 Introduction to The KA660 Subsystem

The KA660 processor module combines with the MS650 Memory module and the H3602 Console module to form the CPU/Memory subsystem for the VAX 4000-200 product. The subsystem is available in two enclosures, the BA430 or the BA215. It uses the DSSI bus to communicate with mass storage devices and the Q22-bus to communicate with I/O devices. A single KA660 CPU module can support up to four MS650 memory modules.

Figure 1–1 is a block diagram of the CPU/Memory subsystem's major functions.

Figure 1–1: KA660 Module in a System



The KA660 and the MS650 designs are implemented in standard quad-height sized modules. Both modules mount in standard Q22-bus backplane slots which implement the Q22-bus in the AB rows and the CD interconnect in the CD rows.

The KA660 Processor Module communicates with the memory modules across a memory interconnect routed through a 50-pin ribbon cable and the CD interconnect on the backplane. The DSSI connects through a 50-Pin ribbon cable located on top of the memory interconnect cable. The backplane connector also connects the subsystem with the Q22-bus. There are no jumpers or switches to configure on the processor module. The KA660 connects to the H3602 Console Module and the Ethernet Controller with a 40-pin ribbon cable. The Console Module contains configuration switches, Ethernet and DSSI connectors, fuses, and an LED display.

1.2 KA660 Processor Module

The KA660 processor can be configured *only* as an arbiter on the Q22-bus. An arbiter is the single entity responsible for controlling the Q22-bus. It must reside in the first backplane slot where it arbitrates bus mastership and fields bus interrupt requests and any on-board interrupt requests. This processor module is designed for use in high-speed, real-time applications and for multiuser, multitasking environments. There are two variants: the KA660-AA, runs multiuser software; and the KA660-BA, runs single-user software.

Figure 1–2 is a photograph of the KA660 Processor Module.

Figure 1–2: The KA660 Processor Module

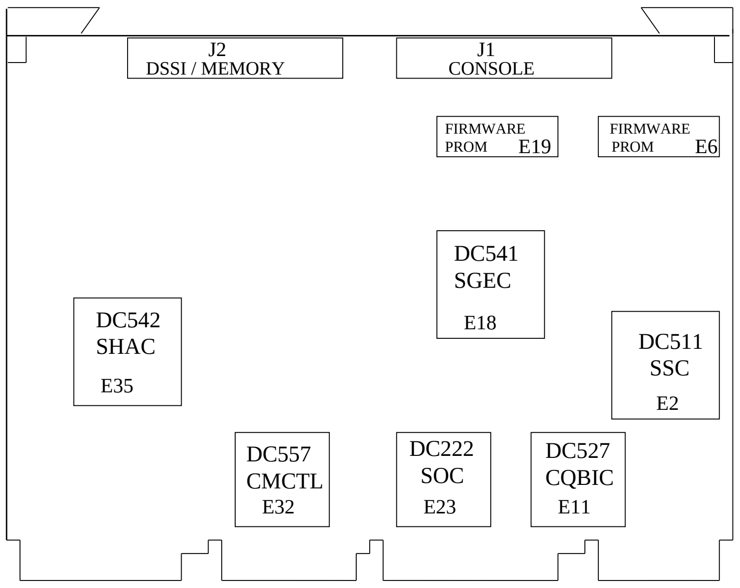
The major hardware components of the KA660 CPU module are listed below. The chip identification numbers are shown in Figure 1–3

• System on a Chip (SOC) CPU	DC222
• A Main Memory Controller (CMCTL)	DC557
• Q22-bus Interface (CQBIC)	DC527
• System Support Chip (SSC)	DC511
• Second Generation Ethernet Controller (SGEC)	DC541
• Single Host Adapter Chip to interface DSSI (SHAC)	DC542

- Two Firmware EPROMs
- A Boot and diagnostic facility
- Console Connection
- VAX compatible console port
- Backplane Connection

Figure 1–3 shows the positions of the major chips on the KA660.

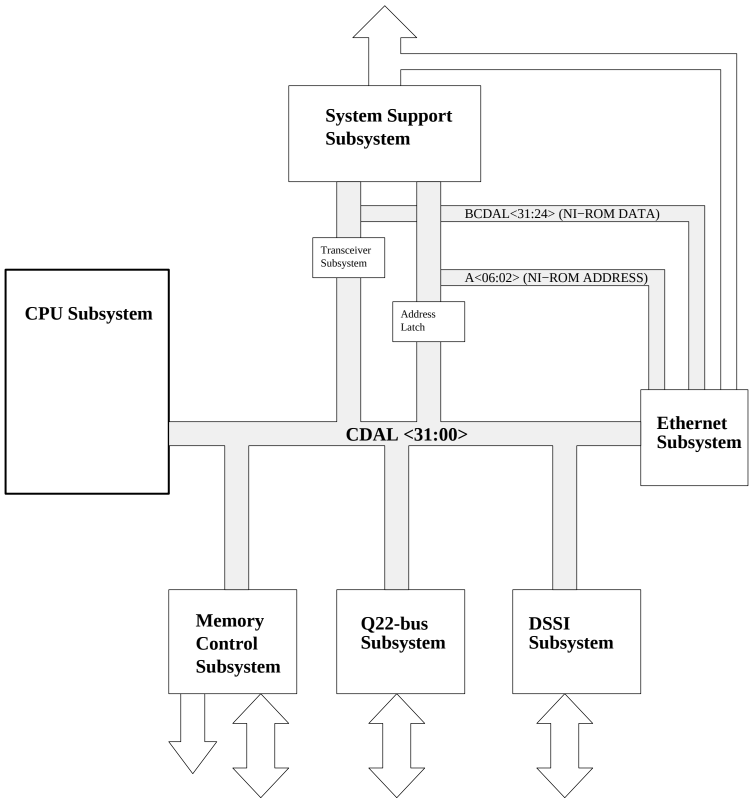
Figure 1–3: KA660 CPU Module Component Side



The KA660 Processor Module is divided into several major functional subsystems as listed below and shown in Figure 1–4.

- The Central Processing Subsystem
- The Memory Control Subsystem
- The Q22-bus Subsystem
- The DSSI Subsystem
- The System Support Subsystem
- The Ethernet Subsystem

Figure 1–4: KA660 Processor Module Major Functional Blocks



The rest of Section 1.1 describes the subsystems.

- **The Central Processing Subsystem**

The Central Processing Subsystem features the System On a Chip (SOC) CPU and its accompanying support logic. The SOC Chip is a unique design that contains several system components on a single substrate contained in a 132-pin surface mount, CERQUAD chip package. The SOC contains the Central Processing Unit (CPU), The Floating Point Accelerator Unit (FPA), and 8 KB of cache to optimize system performance.

The central processor in the SOC supports the following MicroVAX instruction set with the following string instructions:

- CMPC3 (Compare Character - 3 operand)
- CMPC5 (Compare Character - 5 operand)
- LOCC (Locate Character)
- SKPC (Skip Character)
- SPANC (Span Character)
- SCANC (Scan Character)

The following subset of the VAX data types are provided:

- Byte
- Word
- Longword
- Quadword
- Character string
- Variable-length bit field
- Absolute queues
- Self-relative queues
- F-floating
- G-floating
- D-floating

Support for the remaining VAX data types can be provided through macrocode emulation. The processor also supports full VAX memory management with demand paging and a 4GB virtual address space.

The Floating Point Accelerator Unit (FPU) in the SOC executes the VAX f_, d_, and g_ floating point instructions. It executes 61 floating point instructions and 2 longword-length integer multiply instructions in the VAX base instruction group. It supports the MicroVAX chip subset of the VAX floating point instruction set and data types.

- **The Memory Control Subsystem**

  The Memory Control Subsystem contains the Memory Controller Chip (CMCTL) and its associated termination logic. This subsystem provides an interface between the Data and Address Lines (CDAL) lines from the KA660 CDAL bus and the Data and Address lines on the MS650 MDATA bus.

  The CMCTL chip contains approximately 25,000 transistors in a 132-pin CERQUAD surface mount package. It supports up to 64 Kbytes of ECC memory, with a 450 ns cycle time for longword transfers and a 720 ns cycle time for quadword transfers,

  This memory resides on one to four MS650 memory modules, depending on the system configuration. The MS650 communicates with the KA660 through the memory interconnect which uses the CD interconnect and a 50-pin ribbon cable.

- **The Q22-bus Subsystem**

  The Q22-bus subsystem contains the Q22-bus interface and asociated termination logic. This subsystem provides an interface between the Q22-bus and the Central Processor's CDAL bus. The interface is implemented with the CQBIC. The CQBIC contains approximatley 40,870 transistors in a 132-pin CERQUAD surface mount package. The CQBIC is a 32-bit to 16 bit adapter which provides physical memory address translation for direct memory access (DMA) devices on the Q22-bus. It supports up to 16-word, block mode transfers between a Q22-bus DMA device and main memory, and up to 2-word, block mode transfers between the CPU and Q22-bus devices. The Q22-bus interface contains the following:

- A 16-entry map cache for the 8192-entry, main memory-resident scatter-gather-map, used for translating 22-bit Q22-bus addresses into 26-bit main memory addresses.
- Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7-BR4
- Q22-bus termination (240 Ω)

- **The DSSI Subsystem**

  This subsystem provides an interface between the DSSI bus and the KA660 CDAL bus. It contains the Single Host Adapter Chip (SHAC), the DSSI jumpers, 16MHZ Oscillator, and associated termination and control logic. The SHAC is in a 164 pin CERQUAD package. It facilitates scatter and gather mapping along with internal FIFO buffering.

  The DSSI interface allows the DSSI bus on the KA660 to transmit packets of data to, and receive packets from, up to seven other DSSI devices. These devices include the RF-series Integrated Storage Elements (ISEs), a KFQSA module, a second KA660 module, or a KA640 module.

  The DSSI bus improves system performance because it has a higher transfer rate than the Q22-bus and it relieves the Q22-bus of disk traffic. The DSSI bus has eight data lines, one parity line, and eight control lines. Controllers are built into the ISEs, enabling many functions to be handled without host or adapter intervention.

- **The System Support Subsystem**

  The system support subsystem handles the basic functions required to support the console in a system environment. This subsystem contains the System Support Chip (SSC), the Firmware ROMs, the Boot and Diagnostic Register, and the Station Address ROM.

  The SSC chip is implemented in an 84 pin CERQUAD surface mount package. It provides console and boot code support functions, operating system support functions, timers, and the following features:

  - Word-wide ROM unpacking
  - Halt-arbitration logic
  - Interval timer with 10ms interrupts
  - IORESET register
  - Two programmable timers
  - 1kB battery backed-up RAM
  - Console serial line
  - VAX-standard time-of-year clock with battery backup
  - Programmable CDAL bus timeout
  - Register controlling the diagnostic LEDs

Resident firmware Read Only Memory is located on two chips, each 128 KByte by 8-bit EPROMS. The firmware gains control when the CPU halts. This code contains programs that provide the following services:

- Board initialization
- Power-up self-testing of the KA660 and MS650 modules

- Emulation of a subset of the VAX standard console (auto or manual bootstrap, auto or manual restart, and a simple command language for examining or altering the state of the processor)
- Booting from supported Q22-bus devices
- Multilingual translation of key system messages

The Boot and Diagnostic Register (BDR) allows the firmware and the operating system to read KA660 configuration bits. The station address ROM contains the network address of the system. It is implemented in a 32-Byte by 8-bit ROM (6331).

- **The Ethernet Subsystem**

  The Ethernet subsystem handles communications between the CPU module and other nodes on the Ethernet. It is implemented with the Second Generation Ethernet Controller Chip (SGEC, DC541) on-board network interface. Used in connection with the H3602 console module, the SGEC allows the KA660 to connect to either a thinwire or standard Ethernet. It supports the Ethernet Data Link Layer and the CP Bus Parity Protection. The SGEC chip is in a 84 pin package. The chip facilitates scatter and gather mapping along with dual internal FIFO buffering.

1.3 MS650 Memory Module

The MS650 memory module for the KA660 CPU is available in two variations. The MS650-BA contains 16 Mbyte of memory and the MS650-BB contains 8 Mbytes. The memory is arranged in 39-bit wide arrays implemented with 1 Mbyte, 120 ns, dynamic RAMs in surface mount packages. Of the 39 bits, 32 bits are data and 7 bits are Error Checking and Correction (ECC) Bits. The MS650 modules are single, quad-height, Q22-bus modules as shown in <REFERENCE>(MS650-BA_PHOTO)and <REFERENCE>(MS650-BB_PHOTO).

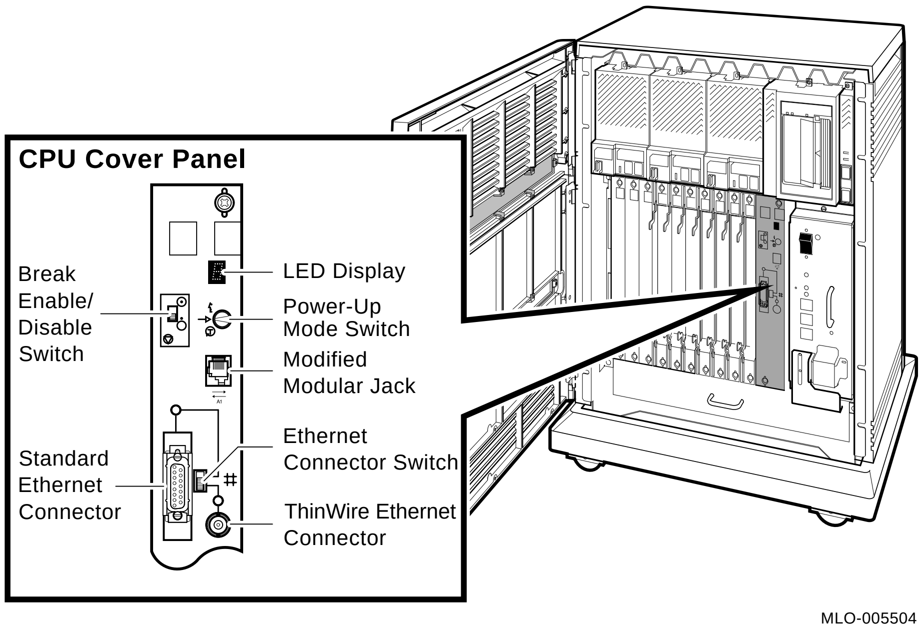
Figure 1–4: MS650 Memory Module (16 MB)

Figure 1–4: MS650 Memory Module (8 MB)

1.4 H3602 Console Module

The H3602 console module (Figure 1–7) is a unique I/O panel that is used in BA213 and BA215 enclosures, A one-piece ribbon cable on the H3602 plugs into J1 (system support connector) on the KA660. The H3602 fits over backplane slots 1 and 2, covering both the KA660 Processor module and the first of four possible MS650 memory modules. The H3602 allows the KA660 CPU module to interface to a serial line console device, a DSSI bus, and to the Ethernet. Adhesive tags are included for the user to name the modules in the respective slots.

Figure 1–7: H3602 Console Module



CPU Cover Panel
Break Enable/ Disable Switch
LED Display
Power-Up Mode Switch
Modified Modular Jack
Standard Ethernet Connector
Ethernet Connector Switch
ThinWire Ethernet Connector
MLO-005504

The outside H3602 console panel contains the following features:

- Modified modular jack (MMU) SLU connector
- Power-up mode switch
- Hexadecimal LED display
- Break enable switch
- Standard/Thin wire Ethernet connectors
- Standard/ThinWire Ethernet selector switch
- Indicator LEDs

The console panel also has the following features inside:

- Baud Rate rotary switch
- Battery backup unit (BBU) for TOY clock
- 40-pin cable connector
- List of baud rate switch settings

Chapter 2

Installation and Configuration

2.1 Introduction

This chapter describes how to install the KA660 in a system. It discusses the following topics.

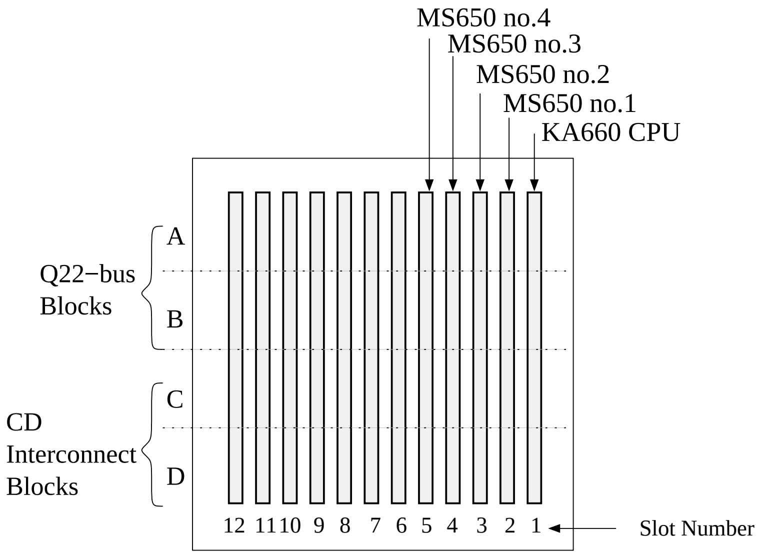
- Installing the KA660 and MS650 modules
- Configuring the KA660
- The KA660 connectors

2.2 Installing the KA660 and MS650 Memory Modules

The KA660 and MS650 (-BB or -BA only) modules must be installed in system enclosures having Q22/CD slots. These modules are not compatible with Q/Q backplane slots and therefore should only be installed in Q22/CD backplane slots.

The KA660 CPU module and the MS650 memory modules must be installed in the five rightmost backplane slots. The KA660 CPU module must be installed in slot 1 of the Q22/CD backplane. MS650 memory modules must be installed in slots immediately adjacent to the CPU module. Figure 2–1 shows the positions of the module slots in the backplane.

Figure 2–1: Backplane Slots

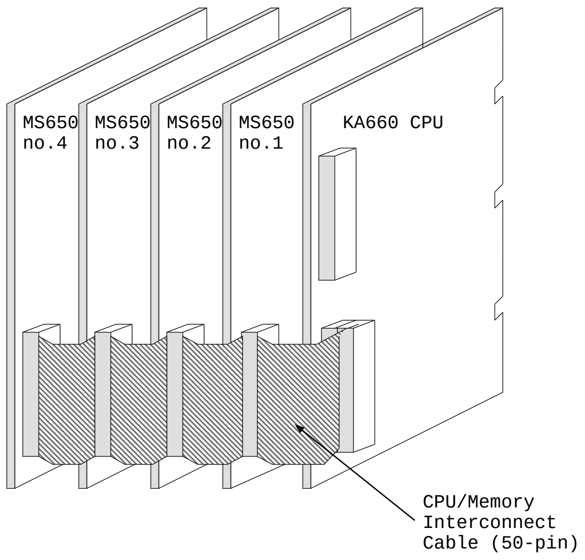


Up to four MS650 memory modules can be installed, occupying slots 2,3,4 and 5 respectively. A 50-pin ribbon cable is used to connect the KA660 processor module and the MS650 memory module(s) , as shown in Figure 2–2.

The KA660 module is installed in backplane slot 1 and the memory modules are installed in slots 2 through 5. Use the following procedure to install the KA660 and MS650 modules.

1. Install the KA660 CPU in slot 1 of the Q22-bus/CD backplane.
2. Install the MS650 memory module in slots 2, immediately adjacent to the KA660 CPU. When installing additional memory use slots 3 through 5. Do not leave a gap between memory modules.
3. Install a 50-pin ribbon cable between the KA660 CPU and the MS650 Memory Module/s. (see Figure 2–2)

Figure 2–2: Processor and Memory Module Connection



2.3 Module Configuration and Naming

Each module in a system must use a unique device address and interrupt vector. The device address is also known as the control and status register (CSR) address. Most modules have switches or jumpers for setting the CSR address and interrupt vector values. The value of a floating address depends on what other modules are housed in the system.

Set CSR addresses and interrupt vectors for a module as follows:

1. Determine the correct values for the module with the CONFIGURE command at the console I/O prompt (>>>). The CONFIG utility eliminates the need to boot the VMS operating system to determine CSRs and interrupt vectors. Enter the CONFIGURE command, then HELP for the list of supported devices:

```
>>> config
Enter device configuration, HELP, or EXIT
Device, Number? help
Devices:
```

LPV11	KXJ11	DLV11J	DZQ11	DZV11	DFA01
RLV21	TSV05	RXV21	DRV11W	DRV11B	DPV11
DMV11	DELQA	DEQNA	RQDX3	KDA50	RRD50
RQC25	KXXXX-DISK	TQK50	TQK70	TU81E	RV20
KXXXX-TAPE	KMV11	IEQ11	DHQ11	DHV11	CXA16
CXB16	CXY08	VCB02	QDSS	DRV11J	DRQ3B
VSV21	IBQ01	IDV11A	IDV11B	IDV11C	IDV11D
IAV11A	IAV11B	MIRA	ADQ32	DTC04	DESQA
IGQ11					

The LPV11–SA has two sets of CSR address and interrupt vectors. To determine the correct values for an LPV11–SA, enter LPV11,2 at the DEVICE prompt for one LPV11–SA, or enter LPV11,4 for two LPV11–SA modules.

2. See the *KA660 CPU System Maintenance Manual* for switch and CSR and interrupt vector jumper settings for supported options.

2.4 Mass Storage Configuration

In a BA213 enclosure there is space for four mass storage devices, three integrated storage element (ISE)s and one TK70 or else four ISEs. The ISEs, are part of the DIGITAL Storage System Interconnect (DSSI) bus.

The DSSI bus is part of the backplane. The ISEs are of the RF-series, and they plug into the backplane to become part of the bus. Each ISE must have its own unique DSSI node ID. The ISE receives its node ID from a plug on the operator control panel (OCP) on the front panel.

The VMS operating system creates DSSI disk device names according to the following scheme:

nodename $ DIA unit number. For example, *SUSAN$DIA3*

You can use the device name for booting, as follows:

```
>>> BOOT SUSAN$DIA3
```

You can access local programs in the RF-series ISE through the MicroVAX Diagnostic Monitor (MDM), or through the VMS operating system (version 5.0) and console I/O mode SET HOST /DUP command. This command creates a virtual terminal connection to the storage device and the designated local program using the Diagnostic and Utilities Protocol (DUP) standard dialog. Section 2.4.3 describes the procedure for accessing DUP through the VMS operating system.

2.4.1 Changing the Node Name

Each ISE has a node name that is maintained in EPROM on board the controller module. This node name is determined in manufacturing from an algorithm based on the drive serial number. You can change the node name of the DSSI device to something more meaningful by following the procedure in Example 2–1. In the example, the node name for the ISE at DSSI node address 1 is changed from R3YBNE to DATADISK.

Example 2–1: Changing a DSSI Node Name

```
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF71)

DSSI Node 1 (R3YBNE)       !The node name for this drive will be
-DIA1 (RF71)               !changed from R3YBNE to DATADISK.

DSSI Node 7 (*)
>>>
>>> set host/dup/dssi 1
Starting DUP server...
Copyright  1988  Digital Equipment Corporation
DRVEXR V1.0  D  5-NOV-1988 15:33:06
DRVTST V1.0  D  5-NOV-1988 15:33:06
HISTRY V1.0  D  5-NOV-1988 15:33:06
ERASE  V1.0  D  5-NOV-1988 15:33:06
PARAMS V1.0  D  5-NOV-1988 15:33:06
DIRECT V1.0  D  5-NOV-1988 15:33:06
End of directory
Task Name? params
Copyright  1988  Digital Equipment Corporation

PARAMS> sho nodename

Parameter     Current           Default        Type     Radix
--------- ----------------  ----------------  --------  -----
NODENAME          R3YBNE            RF71        String  Ascii  B


PARAMS> set nodename datadisk

PARAMS> write                  !This command writes the change
                               !to EPROM.
Changes require controller initialization, ok? [Y/(N)] y

Stopping DUP server...
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF71)

DSSI Node 1 (DATADISK)     !The node name has changed from
-DIA1 (RF71)               !R3YBNE to DATADISK.

DSSI Node 7 (*)
```

2.4.2 Changing the DSSI Unit Number

By default, the ISE drive assigns the disk's unit number to the same value as the DSSI node address for that drive.

Example 2–2 shows how to change the unit number of a DSSI device. This example changes the unit number for the RF71 drive at DSSI node address 2 from 1 to 50 (decimal). You must change two parameters: UNITNUM and FORCEUNI. Changing these parameters overrides the default, which assigns the unit number the same value as the node address.

Example 2–2: Changing a DSSI Unit Number

```
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF71)

DSSI Node 1 (R3QJNE)     !The unit number for this drive will be
-DIA1 (RF71)             !changed from 1 to 50 (DIA1 to DIA50).

DSSI Node 7 (*)
>>>
>>> set host/dup/dssi 1
Starting DUP server...
Copyright  1988  Digital Equipment Corporation
DRVEXR V1.0  D  5-NOV-1988 15:33:06
DRVTST V1.0  D  5-NOV-1988 15:33:06
HISTRY V1.0  D  5-NOV-1988 15:33:06
ERASE  V1.0  D  5-NOV-1988 15:33:06
PARAMS V1.0  D  5-NOV-1988 15:33:06
DIRECT V1.0  D  5-NOV-1988 15:33:06
End of directory


Task Name? params
Copyright 1988  Digital Equipment Corporation

PARAMS> sho unitnum

Parameter     Current           Default        Type     Radix
--------- ----------------  ----------------  --------  -----
UNITNUM               0                0      Word    Dec   U

PARAMS> sho forceuni

Parameter     Current           Default        Type     Radix
--------- ----------------  ----------------  --------  -----
FORCEUNI              1                1   Boolean    0/1   U

PARAMS> set unitnum 50

PARAMS> set forceuni 0

PARAMS> write       !This command writes the changes to EPROM.

PARAMS> ex
Exiting...

Task Name?

Stopping DUP server...
>>>
>>>sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF71)

DSSI Node 1 (R3QJNE)          !The unit number has changed
-DIA50 (RF71)                 !and the node ID remains at 1.

DSSI Node 7 (*)
```

2.4.3 Access to RF-series Firmware in VMS Through DUP

You can also access the RF-series ISE firmware utilities from the VMS operating system as well as through the console commands.

Access the ISE firmware through the VMS operating system to look up or to view parameter settings, but not to change them. To change ISE parameter settings, enter the ISE firmware through the console I/O mode SET HOST/DUP command.

Load the FYDRIVER using the following commands in SYSGEN:

```
$ MCR SYSGEN
SYSGEN> LOAD FYDRIVER/NOADAPTER
SYSGEN> CONNECT FYA0/NOADAPTER
SYSGEN> EXIT
$
```

You can then access the ISE firmware utilities using the following VMS command:

```
$ SET HOST/DUP/SERVER=MSCP$DUP/TASK=PARAMS nodename
```

2.4.3.1 Allocation Class

When a KA660 system containing ISEs is configured in a cluster, either as a boot node or a satellite node, you must assign the allocation class in VMS SYSGEN and for the ISE matching non-zero values. To change the allocation class of the ISE, use the following commands:

```
>>> SET HOST/DUP/DSSI <DSSI node number> PARAMS
Starting DUP server..

PARAMS> SET ALLCLASS <allocation class value>

PARAMS> WRITE
Changes require controller initialization, ok? [Y/N] Y

Stopping DUP server..
>>>
```

2.5 DSSI Cabling, Device Identity, and Bus Termination

The ISEs in one particular BA430 enclosure are connected to the system backplane, and communicate internally over the backplane. There are no internal DSSI cables. Externally, a 50-pin ribbon cable connects the DSSI bus to other devices, either hosts or expanders.

All DSSI devices on the same bus must have unique identifiers.

The ID plug provides an identity for the DSSI bus.

2.6 KA660 Connectors

The KA660 uses two connectors, J1 and J2. J1 (system support connector) is the connector for the 40-pin ribbon cable that goes to the console module. Users configure the KA660 through the H3602 console module. Figure 1–3 in Chapter 1, OVERVIEW shows the location of the connectors on the KA660 module. J2 is a dual connector. The upper half contains 50 pins for the DSSI connection and the lower half contains 50 pins for the memory module connection.

Chapter 3

Central Processor

This section provides summary information about the SOC/C (CPU) chip and the MicroVAX architecture. It is not intended as a complete reference, but rather as an overview of the user-visible features.

The central processor of the KA660 supports the MicroVAX Chip subset (plus six additional string instructions) of the VAX instruction set and data types and full VAX memory management. It is implemented as part of the SOC/C chip.

3.1 Processor State

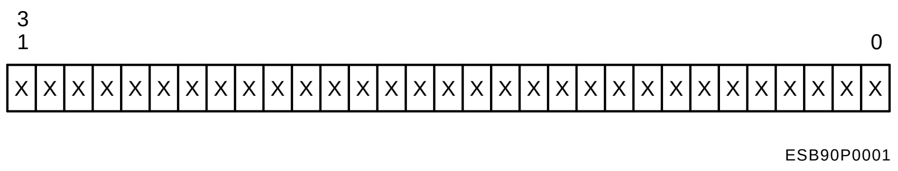
The *processor state* consists of that portion of the state of a process which is stored in processor registers rather than in memory. The processor state is composed of sixteen General Purpose Registers (GPR's), the Processor Status Longword (PSL), and the Internal Processor Registers (IPR's).

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 Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions which can be executed only while running in kernel mode.

3.2 General Purpose Registers (GPRs)

The KA660 implements 16 General Purpose Registers as implemented per the *VAX Architecture Reference Manual*. 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 right to left, <0> through <31>. Figure 3–1 shows the General Purpose Register format. Table 3–1 describes the registers.

Figure 3–1: GPR Format



31
0
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
ESB90P0001

Table 3–1 lists certain registers that have been assigned special meaning by the VAX–11 architecture standard.

Table 3–1: Special GPRs

Register	Register Name	Mnemonic	Description
R15	Program Counter	PC	The PC contains the address of the next instruction byte of the program.
R14	Stack Pointer	SP	The SP contains the address of the top of the processor defined stack.
R13	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	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.

Consult the *VAX Architecture Reference Manual* for more information on the operation and use of these registers.

3.3 Processor Status Longword (PSL)

The KA660 Processor Status Longword (PSL) is implemented per The *VAX Architecture Reference Manual* . 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_{16}. Figure 3–2 shows the Processor Status Longword format. Table 3–2 lists the bits and definitions.

Figure 3–2: PSL Format

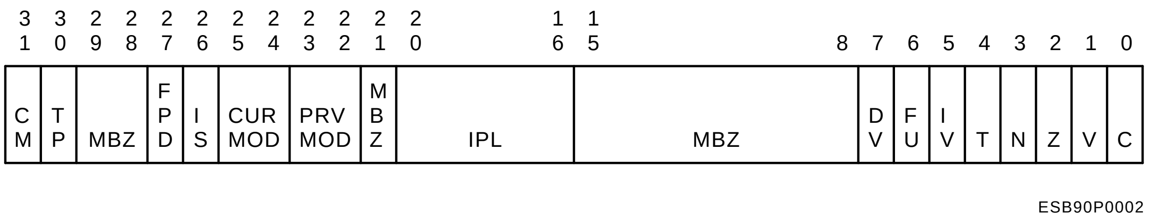


Table 3–2: Processor Status Longword Format

PSL Data Bit	Name	Definition
<31>	CM	Compatibility Mode. This bit always reads as ZERO, loading a ONE into this bit is a NOP.
<30>	TP	Trace Pending
<29:28>	MBZ	Must Be written as Zero
<27>	FPD	First Part Done
<26>	IS	Interrupt Stack
<25:24>	CUR	Current Mode
<23:22>	PRV	Previous Mode
<21>	MBZ	Must Be written as Zero
<20:16>	IPL	Interrupt Priority Level
<15:8>	MBZ	Must Be written as Zero
<7>	DV	Decimal Overflow Trap Enable This read/write bit has no effect on KA660 hardware; it can be used by macrocode which emulates VAX decimal instructions.
<6>	FU	Floating Underflow Fault Enable

Table 3–2 (Cont.): Processor Status Longword Format

PSL Data Bit	Name	Definition
<5>	IV	Integer Overflow Trap Enable
<4>	T	Trace Trap Enable
<3>	N	Negative Condition Code
<2>	Z	Zero Condition Code
<1>	V	Overflow Condition Code
<0>	C	Carry Condition Code

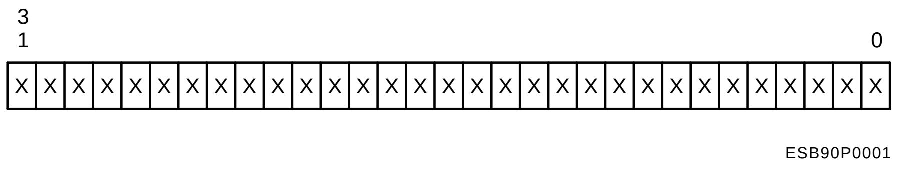
NOTE

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

3.4 Internal Processor Registers (IPRs)

The privileged internal processor register space provides access to many types of CPU control and status registers such as the memory management base registers, parts of the PSL, and the multiple stack pointers. These registers are explicitly accessible only by the Move to Processor Register (MTPR) and Move from Processor Register (MFPR) instructions which require kernel privileges. The addresses of the KA660 internal processor registers are given in Table B–2. Internal processor registers are longword size, as shown in Figure 3–3.

Figure 3–3: Internal Processor Register (IPR) Format


31
0
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
ESB90P0001

IPR Categories

Each IPR falls into one of the following categories:

(1) VAX standard IPRs implemented by KA660 in the *SOC/C* chip.
(2) VAX standard IPRs implemented by KA660 in the *SSC* chip.
(3) Unique KA660 IPRs implemented by all designs that use the *SOC/C* chip.
(4) Unique KA660 IPRs implemented by all designs that use the *SSC* chip.
(5) Not implemented, timed out by the CDAL Bus Timer (in the SSC chip) after 4µs. Read as zero, nop on write.
(6) Access not allowed; accesses result in a reserved operand fault.
(7) Accessible, but not fully implemented, accesses yield unpredictable results.

Table 3–3 explains each IPR.

Table 3–3: KA660 Internal Processor Registers

IPR Number						Implemented		
Decimal	**Hex**	**Register Name**	**Mnemonic**	**Type**	**Scope**	**Where**	**Init?**	**Category**
0	0	Kernel Stack Pointer	KSP	RW	PROC	SOC/C		1
1	1	Executive Stack Pointer	ESP	RW	PROC	SOC/C		1
2	2	Supervisor Stack Pointer	SSP	RW	PROC	SOC/C		1
3	3	User Stack Pointer	USP	RW	PROC	SOC/C		1
4	4	Interrupt Stack Pointer	ISP	RW	CPU	SOC/C		1
5-7	5-7	Reserved						5
8	8	P0 Base Register	P0BR	RW	PROC	SOC/C		1
9	9	P0 Length Register	P0LR	RW	PROC	SOC/C		1
10	A	P1 Base Register	P1BR	RW	PROC	SOC/C		1
11	B	P1 Length Register	P1LR	RW	PROC	SOC/C		1
12	C	System Base Register	SBR	RW	CPU	SOC/C		1
13	D	System Length Register	SLR	RW	CPU	SOC/C		1
14-15	E-F	Reserved						5
16	10	Process Control Block Base	PCBB	RW	PROC	SOC/C		1
17	11	System Control Block Base	SCBB	RW	CPU	SOC/C		1
18	12	Interrupt Priority Level	IPL	RW	CPU	SOC/C	Yes	1
19	13	AST Level	ASTLVL	RW	PROC	SOC/C	Yes	1
20	14	Software Interrupt Request Register	SIRR	W	CPU	SOC/C		1
21	15	Software Interrupt Summary Register	SISR	RW	CPU	SOC/C	Yes	1
22-23	16-17	Reserved						5
24	18	Interval Counter Control Status	ICCS	RW	CPU	SOC/C	Yes	3
25	19	Next Interval Count	NICR					5
26	1A	Interval Count	ICR					5
27	1B	Time of Year Register	TODR	RW	CPU	SOC/C		2

Table Headings Meaning

IPR Number (*Decimal*—Decimal number of the Processor Register, *Hex*—Hex Number of the Processor Register.)
Type (*R*—Read-Only Register, *W*—Write-Only Register, *RW*—Read/Write Register.)
Scope (*CPU*—CPU wide register, *PROC*—per processor register.)
Implemented Where? (*SOC/C* —Implemented in the SOC CPU Chip, *SSC* —Implemented in the MicroVAX System Support Chip)
Init?(Is this register initialized on Module RESET? (Power-up, Negation of DCOK) *yes* or *no*)
Category—(Processor Register Category as previously defined in <REFERENCE>(IPR_CAT).)

Table 3–3 (Cont.): KA660 Internal Processor Registers

IPR Number						Implemented		
Decimal	**Hex**	**Register Name**	**Mnemonic**	**Type**	**Scope**	**Where**	**Init?**	**Category**
28	1C	Console Storage Receiver Status	CSRS	RW	CPU	SOC/C	Yes	7
29	1D	Console Storage Receiver Data	CSRD	R	CPU	SOC/C	Yes	7
30	1E	Console Storage Transmitter Status	CSTS	RW	CPU	SOC/C	Yes	7
31	1F	Console Storage Transmitter Data	CSTD	W	CPU	SOC/C	Yes	7
32	20	Console Receiver Control /Status	RXCS	RW	CPU	SOC/C	Yes	4
33	21	Console Receiver Data Buffer	RXDB	R	CPU	SOC/C	Yes	4
34	22	Console Transr Control /Status	TXCS	RW	CPU	SOC/C	Yes	4
35	23	Console Transr Data Buffer	TXDB	W	CPU	SOC/C	Yes	4
36	24	Translation Buffer Disable	TBDR					5
37	25	Cache Control	CCR	RW		SOC/C	Yes	3
39	27	Memory System Error	MSER	RW	CPU	SOC/C	YES	3
40	28	Reserved						5
41	29	Reserved						5
42	2A	Console Saved PC	SAVPC	R	CPU	SOC/C		3
43	2B	Console Saved PSL	SAVPSL	R	CPU	SOC/C		3
44-54	2C-36	Reserved						5
55	37	I/O System Reset Register	IORESET	W	CPU	SOC/C		4
56	38	Memory Management Enable	MAPEN	RW	CPU	SOC/C	Yes	1
57	39	Translation Buffer Invalidate All	TBIA	W	CPU	SOC/C		1
58	3A	Translation Buffer Invalidate Single	TBIS	W	CPU	SOC/C		1
59-61	3B-3D	Reserved						5

Table Headings Meaning

IPR Number (*Decimal*—Decimal number of the Processor Register, *Hex*—Hex Number of the Processor Register.)
Type (*R*—Read-Only Register, *W*—Write-Only Register, *RW*—Read/Write Register.)
Scope (*CPU*—CPU wide register, *PROC*—per processor register.)
Implemented Where? (*SOC/C* —Implemented in the SOC CPU Chip, *SSC* —Implemented in the MicroVAX System Support Chip)
Init?(Is this register initialized on Module RESET? (Power-up, Negation of DCOK) *yes* or *no*)
Category—(Processor Register Category as previously defined in <REFERENCE>(IPR_CAT).)

Table 3–3 (Cont.): KA660 Internal Processor Registers

IPR Number						Implemented		
Decimal	**Hex**	**Register Name**	**Mnemonic**	**Type**	**Scope**	**Where**	**Init?**	**Category**
62	3E	System Identification	SID	R	CPU	SOC/C		1
63	3F	Translation Buffer Check	TBCHK	W	CPU	SOC/C		1
64-127	40-7F	Reserved						6

Table Headings Meaning

IPR Number (*Decimal*—Decimal number of the Processor Register, *Hex*—Hex Number of the Processor Register.)
Type (*R*—Read-Only Register, *W*—Write-Only Register, *RW*—Read/Write Register.)
Scope (*CPU*—CPU wide register, *PROC*—per processor register.)
Implemented Where? (*SOC/C* —Implemented in the SOC CPU Chip, *SSC* —Implemented in the MicroVAX System Support Chip)
Init?(Is this register initialized on Module RESET? (Power-up, Negation of DCOK) *yes* or *no*)
Category—(Processor Register Category as previously defined in <REFERENCE>(IPR_CAT).)

3.5 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 KA660 implements these structures as defined in the *VAX Architecture Reference Manual*, which should be referenced for a description of the PCB and the PCBB.

3.6 Data Types

The central processor provides the following subset of the VAX data types:

- Byte
- Word
- Longword
- Quadword
- Character string
- Variable-length bit field
- F-floating
- G-floating
- D-floating

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

3.7 Instruction Set

The KA660 CPU implements the following subset of the VAX instruction set types in microcode.

- Integer arithmetic and logical
- Address
- Variable length bit field
- Control
- Procedure call
- Miscellaneous

- Operating system support
- G_floating
- Queue
- F_floating
- D_floating
- Character string
  - MOVC3
  - MOVC5
  - CMPC3*
  - CMPC5*
  - LOCC*
  - SCANC*
  - SKPC*
  - SPANC*

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

The KA660 SOC CPU provides special microcode assistance to aid the macrocode emulation of the following instruction groups:

- Character string (except MOVC3, MOVC5, CMPC3*, CMPC5*, LOCC*, SCANC*, SKPC*, SPANC*)
- Decimal string
- CRC
- EDITPC

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

- Octaword
- Compatibility mode instructions

Appendix C lists the entire KA660 instruction set, indicating which instructions are implemented in the Floating Point Accelerator (FPA), and which instructions have microcode assists to speed up macrocode emulation.

3.8 Memory Management

The KA660 implements full VAX Memory Management as defined in the *VAX Architecture Reference Manual* . System space addresses are virtually mapped through single-level page tables, and process space addresses are virtually mapped through two-level page tables. See the *VAX Architecture Reference Manual* for descriptions of the virtual to physical address translation process, and the format for VAX Page Table Entries (PTE's).

3.8.1 Translation Buffer

To reduce the overhead associated with translating virtual addresses to physical addresses, the KA660 employs a 28-entry, fully associative, translation buffer for caching VAX PTE's in modified form. Each entry can store a modified PTE for translating virtual addresses in either the VAX Process Space, or VAX System Space. The translation buffer is flushed whenever the following actions are performed:

- Memory management is enabled or disabled (for example, by writes to IPR 56)
- Any page table base or length registers are modified (for example, by writes to IPR's 13:8)
- IPR 57 (TBIA) or IPR 58 (TBIS) are written to.

Each entry is divided into two parts: a 23-bit Tag Register and a 32-bit PTE Register. The Tag Register is used to store the Virtual Page Number (VPN) of the virtual page that the corresponding PTE Register maps. The PTE register stores the 21-bit Page Frame Number field, the PTE.V bit, the PTE.M bit and an 8-bit partially decoded representation of the 4-bit VAX PTE PROT field, from the corresponding VAX PTE, as well as a Translation Buffer Valid (TB.V) bit. The SOC CPU Design Spec can be referenced for details of the 8-bit PROT field.

During virtual-to-physical address translation, the contents of the 28 Tag Registers are compared with the Virtual Page Number Field (bits <31:9>) of the virtual address of the reference. If there is a match with one of the Tag Registers then a translation buffer "hit" has 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.

3.8.2 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)
IPR 63 (TBCHK)

Memory management can be enabled/disabled via IPR 56 (MAPEN). Writing 0 to this register with a MTPR instruction disables memory management and writing a 1 to this register with a MTPR instruction enables memory management. 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 0 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 IPRs are write only. The operation of a MFPR instruction from any of these registers is UNDEFINED.**

3.9 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 (for example, an interrupt from an external hardware device). An *exception* is caused by the execution of the current instruction and is typically handled by the current process (for example, an arithmetic overflow).

3.9.1 Interrupts

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

Non-maskable interrupts cause a halt via the hardware halt procedure. The hardware halt procedure does the following:

- Saves the PC, PSL, MAPEN<0> and a halt code in IPR's
- Raises the processor IPL to 1F
- 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 because this halt code is generated after a hardware reset).

Maskable interrupts cause the following:

- The PC and PSL is saved
- The processor IPL is raised to the priority level of the interrupt (except for Q22-bus, Mass Storage and Network Interface interrupts where the processor IPL is set to 17 independent of the level at which the interrupt was received)
- The interrupt is dispatched to the appropriate service routine through the SCB.

The various interrupt conditions for the KA660 are listed in Table 3–4 along with their associated priority levels and SCB offsets.

Table 3–4: Interrupt Priority Levels

Priority Level	Interrupt Condition	SCB Offset
Non-maskable	BDCOK and BPOK negated then asserted on Q22-bus (Power Up)	*
	BDCOK negated then asserted while BPOK asserted on Q22-bus (SCR<7> clear) .	*
	BDCOK negated then asserted while BPOK asserted on Q22-bus (SCR<7> set).	**
	BINIT asserted on Q22-bus when configured as an auxiliary	*
	BHALT asserted on Q22-bus	**
	BREAK generated by the console device	**
1F	Unused	
1E	BPOK negated on Q22-bus	0C
1D	CDAL Bus parity error	60
	Q22-bus NXM on a write	60
	Uncorrectable main memory errors	60
	CDAL Bus timeout during DMA	60
	Main memory NXM errors	60
1C:1B	Unused	
1A	Correctable main memory errors	54
19:18	Unused	
17	BR7 L asserted	Q22-bus Vector plus 200_{16}
16	Interval Timer Interrupt	C0
	BR6 L asserted	Q22-bus Vector plus 200_{16}
15	BR5 L asserted	Q22-bus Vector plus 200_{16}
14	Console Terminal	F8,F6
	Programmable Timers	78,7C
	SHAC Mass Storage Interface (DSSI port 1)(External)	104
	SGEC Network Interface	10C
	Interprocessor Doorbell	204

* —These conditions generate a hardware halt procedure with a halt code of 3 (hardware reset).
**—These conditions generate a hardware halt procedure with a halt code of 2 (external halt).

Table 3–4 (Cont.): Interrupt Priority Levels

Priority Level	Interrupt Condition	SCB Offset
	BR4 L asserted	Q22-bus Vector plus 200_{16}
13:10	Unused	
0F:01	Software interrupt requests	84-BC

* —These conditions generate a hardware halt procedure with a halt code of 3 (hardware reset).
**—These conditions generate a hardware halt procedure with a halt code of 2 (external halt).

NOTE

Because the Q22-bus does not allow differentiation between the four bus grant levels (for example, a level 7 device could respond to a level 4 bus grant), the KA660 CPU raises the IPL to 17 after responding to interrupts generated by the assertion of either BR7 L, BR6 L, BR5 L, or BR4 L. The KA660 maintains the IPL at the priority of the interrupt for all other interrupts.

The interrupt system is controlled by three IPR's:

- IPR 18, the Interrupt Priority Level Register (IPLR) (Figure 3–4), is used for loading the processor priority field in the PSL (bits<20:16>).
- IPR 20, the Software Interrupt Request Register (SIRR) (Figure 3–5), is used for creating software interrupt requests.
- IPR 21, the Software Interrupt Summary Register (SISR) (Figure 3–6), records pending software interrupt requests at levels 1 through 15.

The format of these registers is presented in Figure 3–4, Figure 3–5, and Figure 3–6. Refer to the *VAX Architecture Reference Manual* for more information on these registers.

Figure 3–4: Interrupt Priority Level Register (IPLR) - (IPR 18_{10} 12_{16})

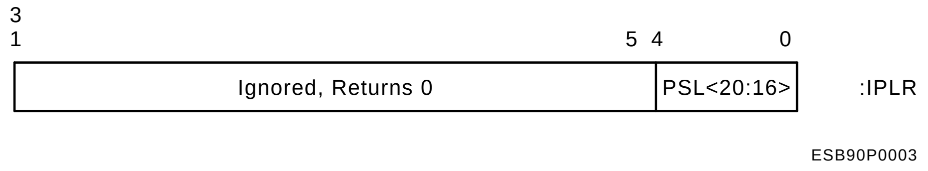


Figure 3–5: Software Interrupt Request Register (SIRR) - (IPL 20_{10} 14_{16})

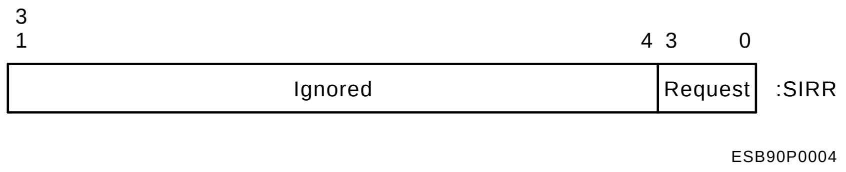
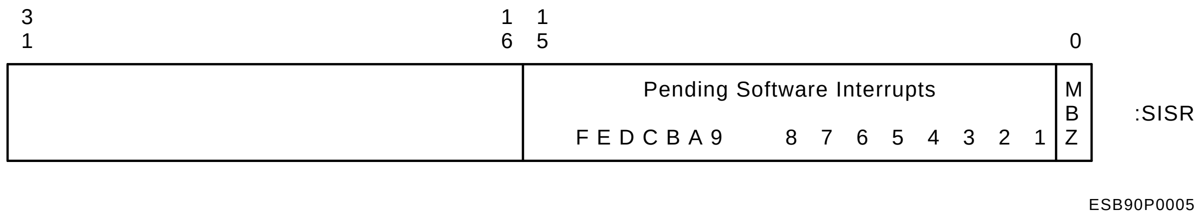


Figure 3–6: Software Interrupt Summary Register (SISR) - (IPL 21_{10} 15_{16})



3.9.2 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. It 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 that 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 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 be divided into six classes. A list of exceptions, grouped by class, is given in the table below. The exceptions listed in Table 3–5 (except machine check) are described in greater detail in the *VAX Architecture Reference Manual*. The machine check exception is described in greater detail in section Section 3.9.3.2.

Table 3–5: 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		18
Instruction Execution Exceptions		
Reserved/Privileged instruction	Fault	10
Emulated instruction	Fault	C8,CC
Change mode	Trap	40-4C
Breakpoint	Fault	2C
Tracing Exception		
Trace	Fault	28
System Failure Exceptions		
Interrupt stack not valid	abort	*
Kernel stack not valid	abort	08
Machine check including the following: • CDAL bus parity errors • Cache parity errors • Q22-bus NXM Errors • Q22-bus device parity Errors • Q22-bus NO GRANT Errors • CDAL Bus Timeout Errors • Main memory NXM Errors • Main Memory Uncorrectable Errors	abort	04

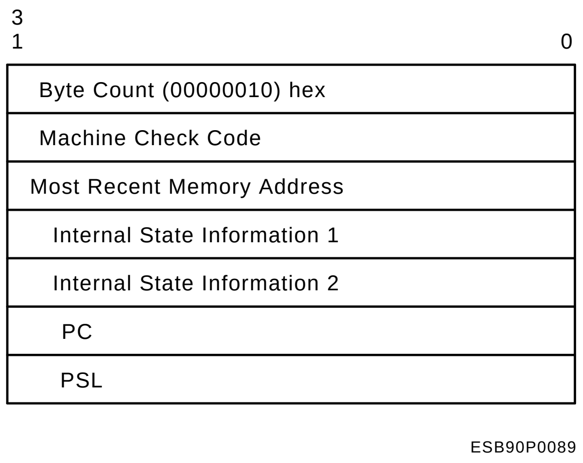
* —Dispatched by resident firmware rather than through the SCB

3.9.3 Information Saved On A Machine Check Exception

In response to a machine check exception, the following information is pushed onto the stack as shown in Figure 3–7:

- Contents of the Processor Status Longword
- Contents of the Program Counter
- Four parameters
- A Byte Count

Figure 3–7: The Processor Stack After a Machine Check Exception


31
0
Byte Count (00000010) hex
Machine Check Code
Most Recent Memory Address
Internal State Information 1
Internal State Information 2
PC
PSL
ESB90P0089

The diagram of the stack pointer is explained in the following paragraphs.

3.9.3.1 Byte Count

Byte Count<31:0> 00000010_{16}, 16_{10}. This value indicates the number of bytes of information that follow on the stack (not including the PC and PSL).

3.9.3.2 Machine Check Code Parameter

Machine Check Code<31:0> A code value that 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) detected an error while communicating with the CPU during the execution of a floating point instruction. The most likely cause of these types of machine checks is a problem internal to the SOC CPU chip. Machine checks due to floating point errors MAY BE RECOVERABLE, depending on the state of the VAX CAN'T RESTART flag (captured in Internal State Information 2 <15> and the FIRST PART DONE flag (captured in PSL <27>). If the FIRST PART DONE flag is set, the error is recoverable. If the FIRST PART DONE FLAG is cleared, then the VAX CAN'T RESTART flag must also be cleared for the error to be 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 3–6: Floating Point Error Machine Checks

Hex Code	Error Description
01	A protocol error was detected by the FPA while attempting to execute a floating point instruction.
02	A reserved instruction was detected by the FPA while attempting to execute a floating point instruction.
03	An illegal status code was returned by the FPA while attempting to execute a floating point instruction. ?CPSTA<1:0>=10
04	An illegal status code was returned by the FPA while attempting to execute a floating point instruction. ?CPSTA<1:0>=01

Memory Management Errors - These codes indicate that the microcode in the SOC CPU chip detected an impossible situation while performing functions associated with memory management. The most likely cause of this type of a machine check is a problem internal to the SOC chip. Machine checks due to memory management errors are NON-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 3–7: Memory Management Error Machine Checks

Hex Code	Error Description
05	The calculated virtual address for a process PTE was in the P0 space instead of the System Space when the CPU attempted to access a process PTE after a translation buffer "miss".
06	The calculated virtual address for a process PTE was in the P1 space instead of the System Space when the CPU attempted to access a process PTE after a translation buffer "miss".
07	The calculated virtual address for a process PTE was in the P0 space instead of the System Space when the CPU attempted to access a process PTE to change the PTE<M> bit before writing to a previously unmodified page.
08	The calculated virtual address for a process PTE was in the P1 space instead of the System Space when the CPU attempted to access a process PTE to change the PTE<M> bit before writing to a previously unmodified page.

Interrupt Errors - This code indicates that the interrupt controller in the SOC 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 SOC CPU chip. Machine checks due to unused IPL errors are NON-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 3–8: Interrupt Error Machine Checks

Hex Code	Error Description
09	A hardware interrupt was requested at an unused Interrupt Priority Level (IPL)

Microcode Errors - This code indicates that an impossible situation was detected by the microcode during instruction execution. Note that most erroneous branches in the SOC 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 SOC CPU chip. Machine checks due to microcode errors are NON-RECOVERABLE. Depending on the current mode, either the current process or the operating system should be terminated.

Table 3–9: Microcode Error Machine Checks

Hex Code	Error Description
0A	An impossible state was detected during a MOVC3 or MOVC5 instruction (not move forward, move backward, or fill).

Read Errors - These codes indicate that an error was detected while the SOC CPU was attempting to read from either the cache, main memory, or the Q22-bus. The most likely cause of this type of machine check must be determined from the state of the MSER, DSER, MEMCSR16, MEAR and SEAR. Machine checks due to read errors MAY BE RECOVERABLE, depending on the state of the VAX CAN'T RESTART flag (captured in Internal State Information 2 <15> and the FIRST PART DONE flag (captured in PSL <27>). If the FIRST PART DONE flag is set, the error is recoverable. If the FIRST PART DONE FLAG is cleared, then the VAX CAN'T RESTART FLAG must also be cleared for the error to be 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 3–10: Read Error Machine Checks

Hex Code	Error Description
80	An error occurred while reading an operand, a Process Page Table Entry during address translation, or on any read generated as part of an interlocked instruction.
81	An error occurred while reading a System Page Table Entry (SPTE), during address translation, a Process Control Block (PCB) entry during a context switch, or a System Control Block (SCB) entry while processing an interrupt.

Write Errors - These codes indicate that an error was detected while the SOC CPU was attempting to write to either the cache, main memory, or the q22-bus. The most likely cause of this type of machine check must be determined from the state of the MSER, DSER, MEMCSR16, MEAR, SEAR and CBTCR. Machine checks due to write errors are NON-RECOVERABLE because the CPU is capable of performing many read operations out of the 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 3–11: Write Error Machine Checks

Hex Code	Error Description
82	An error occurred while writing an operand, or a Process Page Table Entry to change the PTE<M> bit before writing a previously unmodified page.
83	An error occurred while writing a System Page Table Entry (SPTE) to change the PTE<M> bit before writing a previously unmodified page, or a Process Control Block (PCB) entry during a context switch or during the execution of instructions that modify any stack pointers stored in the PCB.

3.9.3.3 Most Recent Virtual Address Parameter

Most Recent Virtual Address <31:0> captures the contents of the Virtual Address Pointer Register at the time of the machine check. If a machine check other than a machine check 81 occurred on a read operation, this field represents the virtual address of the location that was being read when the error occurred, plus four. If a machine check 81 occurred, this field represents the physical address of the location that was being read when the error occurred, plus four. If a machine check other than a machine check 83 occurred on a write operation, this field represents the virtual address of a location that was being referenced either when the error occurred, or sometime after the error occurred, plus four. If a machine check 83 occurred, this field represents the physical address of the location that was being referenced either when the error occurred, or sometime after the error occurred, plus four. In other words, if the machine check occurred on a write operation, the contents of this field cannot be used for error recovery.

3.9.3.4 Internal State Information 1 Parameter

Internal State Information 1 is divided into five fields. The contents of these fields are described below.

Table 3–12: Internal State Information 1 Field Description

Bit Field	Description
<31:24>	This field captures the opcode of the instruction that was being read or executed at the time of the machine check.
<23:20>	This field is read as <1111>.
<19:16>	This field captures the current contents of HSIR<3:0>.
<15:8>	This field captures the state of CCR<7:0> at the time of the machine check. See section 5.2.5 for interpretation of the contents of this register.
<7:0>	This field captures the state of MSER<7:0> at the time of the machine check. See section 5.2.6 for interpretation of the contents of this register.

3.9.3.5 Internal State Information 2 Parameter

Internal State Information 2 is divided into seven fields. The contents of these fields are described below.

Table 3–13: Internal State Information 2 Field Description

Bit Field	Description
<31:24>	This field captures internal state of the SOC CPU chip at the time of the machine check. This field contains the SC register <7:0>.
<23:22>	This field is read as <11>.
<21:16>	This field contains the State Flags <5:0>.
<15>	This field captures the state of the VAX CAN'T RESTART flag at the time of the machine check.
<14:12>	This field is read as <111>.
<11:8>	These bits contain the Arithmetic Logic Unit (ALU) condition codes.
<7:0>	This field captures the offset between the virtual address of the start of the instruction being executed at the time of the machine check (saved PC) and the virtual address of the location being accessed (PC) at time of the machine check.

3.9.3.6 PC

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

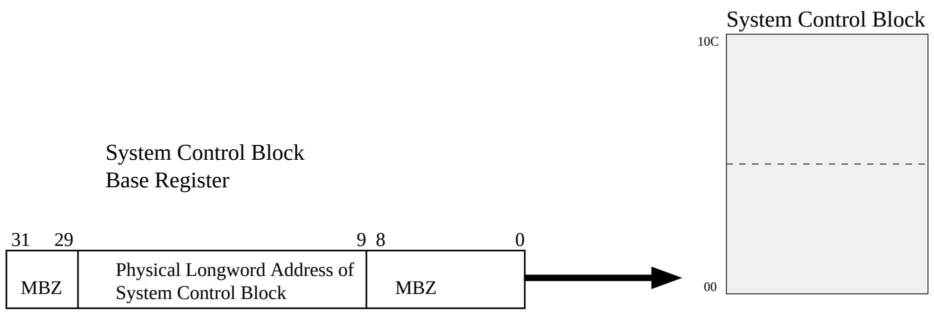
3.9.3.7 PSL

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

3.9.4 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). The format of the System Control Block Base Register is shown in Figure 3–8.

Figure 3–8: System Control Block Base Register (SCBB)



The description of the format is listed in Table 3–14.

Table 3–14: The System Control Block Format

SCB Offset	Interrupt/Exception Name	Type	# Params	Notes
00	Unused	-	-	IRQ passive release on other VAXes.
04	Machine Check	Abort	4	Parameters depend on error type.
08	Kernel Stack Not Valid	Abort	0	Must be serviced on interrupt stack
0C	Power Fail	Interrupt	0	IPL is raised to 1E
10	Reserved/Privileged Instruction	Fault	0	
14	Customer Reserved Instruction	Fault	0	XFC instruction
18	Reserved Operand	Fault/Abort	0	Not always recoverable
1C	Reserved Addressing Mode	Fault	0	

Table 3–14 (Cont.): The System Control Block Format

SCB Offset	Interrupt/Exception Name	Type	# Params	Notes
20	Access Control Violation	Fault	2	Parameters are virtual address, status code
24	Translation Not Valid	Fault		2 Parameters are virtual address, status code
28	Trace Pending (TP)	Fault	0	
2C	Breakpoint Instruction	Fault	0	
30	Unused	-	-	Compatibility mode in other VAXes
34	Arithmetic	Trap/Fault	1	Parameter is type code
38-3C	Unused	-	-	
40	CHMK	Trap	1	Parameter is sign-extended operand word
44	CHME	Trap	1	Parameter is sign-extended operand word
48	CHMS	Trap	1	Parameter is sign-extended operand word
4C	CHMU	Trap	1	Parameter is sign-extended operand word
50	Unused	-	-	
54	Correctable main memory errors	Interrupt	0	IPL is 1A (CRD_L)
58-5C	Unused	-	-	
60	Memory Error	Interrupt	0	IPL is 1D (MEMERR_L)
64-74	Unused	-	-	
78	Programmable Timer 0	Interrupt	0	IPL is 14
7C	Programmable Timer 1	Interrupt	0	IPL is 14
80	Unused	-	-	
84	Software Level 1	Interrupt	0	
88	Software Level 2	Interrupt	0	Ordinarily used for AST delivery
8C	Software Level 3	Interrupt	0	Ordinarily used for process scheduling
90-BC	Software Levels 4-15	Interrupt	0	
C0	Interval Timer	Interrupt	0	IPL is 16(INTTIM_L)
C4	Unused	-	-	
C8	Emulation Start	Fault	10	Same mode exception,FPD = 0; parameters are opcode, PC, specifiers

Table 3–14 (Cont.): The System Control Block Format

SCB Offset	Interrupt/Exception Name	Type	# Params	Notes
CC	Emulation Continue	Fault	0	Same mode exception,FPD = 1: no parameters
D0-DC	Unused	-	-	
E0-EC	Unused	-	-	
F0-F4	Unused	-	-	
F8	Console Receiver	Interrupt	0	IPL is 14
FC	Console Transmitter	Interrupt	0	IPL is 14
104	Mass Storage Interface (DSSI PORT)	Interrupt	0	IPL is 14
10C	Network Interface	Interrupt	0	IPL is 14

3.9.5 Hardware Detected Errors

The KA660 is capable of detecting eleven types of error conditions during program execution:

1. CDAL Bus parity errors indicated by MSER<6> (on a read) or MEMCSR16<7> (on a write) being set.
2. Cache Tag parity errors indicated by MSER<0> being set.
3. Cache Data parity errors indicated by MSER<1> being set.
4. Q22-bus NXM errors indicated by DSER<7> being set.
5. Q22-bus NO SACK errors (no indicator).
6. Q22-bus NO GRANT errors indicated by DSER<2> being set.
7. Q22-bus device parity errors indicated by DSER<5> being set.
8. CDAL-Bus timeout errors indicated by DSER<4>(only on DMA) being set.
9. Main Memory NXM errors indicated by DSER<0> (only on DMA) being set.
10. Main Memory correctable errors indicated by MEMCSR16<29> being set.
11. Main Memory uncorrectable errors indicated by MEMCSR16<31> and DSER<4> (only on DMA) being set.

These errors will cause either a Machine Check Exception, a Memory Error Interrupt, or a Corrected Read Data Interrupt depending on the severity of the error and the reference type that caused the error.

3.9.6 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). The formats for (SAVPC) and (SAVPSL) are shown in Figure 3–9 and Figure 3–10 respectively.

Figure 3–9: Console Saved PC (SAVPC) - (IPR 42_{10} $2A_{16}$)

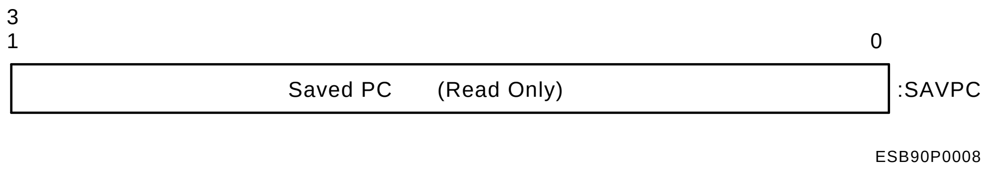
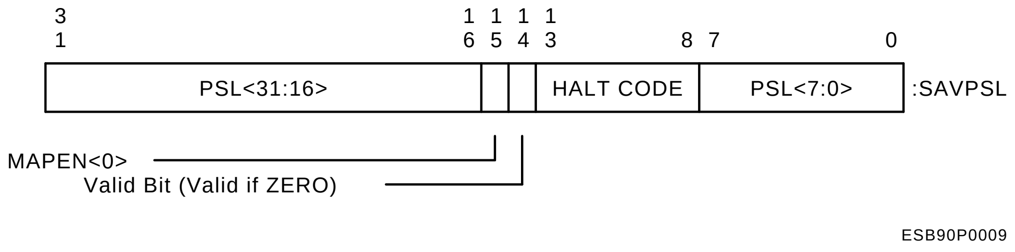


Figure 3–10: Console Saved PSL (SAVPSL) - (IPR 43_{10} $2B_{16}$)



The current stack pointer is saved in the appropriate internal register. The PSL is set to 041F 0000 $_{16}$ (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 $_{16}$. Table 3–15 shows the state of the CPU.

Table 3–15: 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 restart code
SP	Current Interrupt Stack (IPR 4)
PSL	041F 0000 $_{16}$
PC	2004 0000 $_{16}$
MAPEN	0
ICCS	0 (for a halt code of 3)
MSER	0 (for a halt code of 3)
CCR	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 hardware event indicators to dispatch the interrupt or exception that caused the halt, to the appropriate firmware routine (either console emulation, power-up, reboot, or restart). A list of halt codes and their event indicators is given in Table 3–16. Complete halt code descriptions are given in Table I–1.

Table 3–16: HALT Codes

Halt Code	Interrupt Condition	Event Indicator
HALT Codes for UnMaskable Interrupts		
2	**External Halt (SOC/C HALT_L pin asserted)**	
	BHALT asserted on the Q22-bus	DSER<15>
	Remote Boot serviced by SGEC	NICSR5<7>
	BDCOK negated and asserted on the Q22-bus while BPOK stays asserted (Q22-bus REBOOT/RESTART) and SCR<7> is set	DSER<14>
	BREAK generated by the console	RXDB<11>
3	**Hardware Reset (SOC/C RESET pin asserted)**	
	BDCOK and BPOK negated then asserted on the Q22-bus (Power-Up)	-

Table 3–16 (Cont.): HALT Codes

Halt Code	Interrupt Condition	Event Indicator
	BDCOK negated and asserted on the Q22-bus while BPOK stays asserted (Q22-bus REBOOT/RESTART) and SCR<7> is clear.	-
HALT Codes for Exceptions		
6	HALT instruction executed in Kernel Mode	
HALT Codes for Exceptions that occurred while Serving an Interrupt or Exception		
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	Access Control Violation (ACV) or Translation Not Valid (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	

3.10 System Identification

The firmware and operating system software references two registers to determine the processor on which they are running. The first, the System Identification register (SID), is an internal processor register. The second, the System Identification Extension register (SIE), is a firmware register located in the on board EPROM.

3.10.1 System Identification Register

The System Identification Register (SID), IPR 62, is a read-only register implemented in the SOC CPU chip . This 32-bit, Read-Only register is used to identify the processor type and its microcode revision level. The SID longword is read from IPR 62 using the MFPR instruction. This longword value is processor specific. The format is shown in Figure 3–11. Bit definitions are listed in Table 3–17.

Figure 3–11: System Identification Register (SID) - (IPR 62_{10} $3E_{16}$)

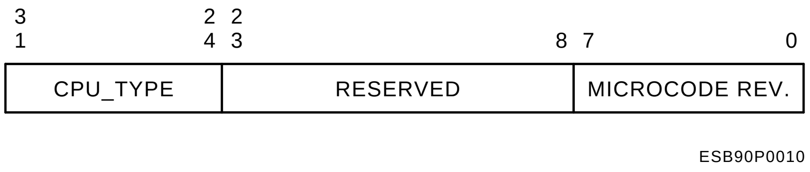


Table 3–17:

Field	Name	RW	Description
<31:24>	CPU_TYPE	ro	CPU type is the processor specific identification code. This field always reads as 20_{10} indicating the processor is implemented with an SOC CPU chip.
<23:8>	RESERVED	ro	Reserved for future use.
<7:0>	VERSION	ro	Version of the microcode.

3.10.2 System Identification Extension Register (SIE) (20040004)

The System Identification Extension register is an extension of the SID register and is used to further differentiate between hardware configurations. The SID register identifies which CPU and microcode is executing, and the SIE register identifies what module and firmware revision are present. Note, the fields in this register are dependent on SID<31:24>(CPU_TYPE).

By convention, all MicroVAX systems implement a longword at physical location 20040004 in the firmware EPROM for the SIE register. This 32-bit Read-Only register is implemented in the KA660 ROM. Figure 3–12 shows the format of this register. Table 3–18 lists the definitions of the register bits.

Figure 3–12: System Identification Extension Register (SIE)

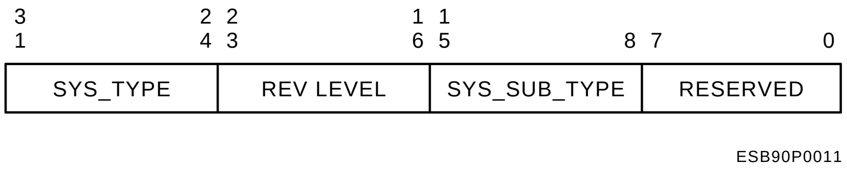


Table 3–18:

Field	**Name**	**RW**	**Description**
31:24	SYS_TYPE	ro	This field identifies the type of system for a specific processor. *01 : Q22-bus single processor system.*
23:16	VERSION	ro	This eight bit field contains two hexadecimal digits that reflect the version of the resident firmware (EPROM). For example, if the firmware version is V5.0, the banner will display V5.0 and this field will encode as a 50_{16}.
15:8	SYS_SUB_ TYPE	ro	This field reflects the particular system sub-type. *01 : KA650* *02 : KA640* *03 : KA655* *04 : KA670*
7:0	RESERVED		This field is reserved.

3.11 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
- Write references

3.11.1 Request Instruction-Stream Read References

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

3.11.2 Demand 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 the following references:

- Operand
- Page Table Entry (PTE)
- System Control Block (SCB)
- Process Control Block (PCB)

When interlocked instructions, such as Branch on Bit Set and Set Interlock (BBSSI) are executed, a Demand D-Stream Read-Lock reference is generated.

Because the CPU does not impose any restrictions on data alignment (other than the aligned operands of the ADAWI and interlocked queue instructions) and because memory can only be accessed one aligned longword at a time, all data read references are translated into an appropriate combination of masked and unmasked, aligned longword read references.

If the required data is a byte, a word within a longword or an aligned longword then a single, aligned longword Demand D-Stream read reference is generated. If the required data is a word that crosses a longword boundary, or an unaligned longword then two successive aligned longword Demand D-Stream read references are generated. Data larger than a longword is divided into a number of successive aligned longword Demand D-Stream reads, with no optimization.

3.11.3 Write References

Whenever data is stored or moved, a write reference is generated. Because the CPU does not impose any restrictions on data alignment (other than the aligned operands of the ADAWI and interlocked queue instructions) and because memory can only be accessed one aligned longword at a time, all data write references are translated into an appropriate combination of masked and unmasked aligned longword write references.

If the required data is a byte, a word within a longword , or an aligned longword then a single, aligned longword write reference is generated. If the required data is a word that crosses a longword boundary, or an unaligned longword then two successive aligned longword write references are generated. Data larger than a longword is divided into a number of successive aligned longword writes.

Chapter 4

KA660 Cache Memory

To maximize CPU performance, the SOC CPU provides an 6 Kilo Byte, 6-way associative write-through cache with an 8 byte block and fill size.

4.1 Cacheable References

Any reference that is stored by the cache is called a *cacheable reference*. The cache stores CPU read references to the VAX Memory Space (bit <29> of the physical address equals (0)) only. It does not store references to the VAX I/O space, or DMA references by the Q22-Bus Interface, the DSSI interface, or the Ethernet interface.

Both I-Stream and D-Stream references are cached in the enabled banks of the cache.

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

Whenever the CPU generates a cacheable reference that is stored in the cache, no reference is generated on the CDAL Bus.

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

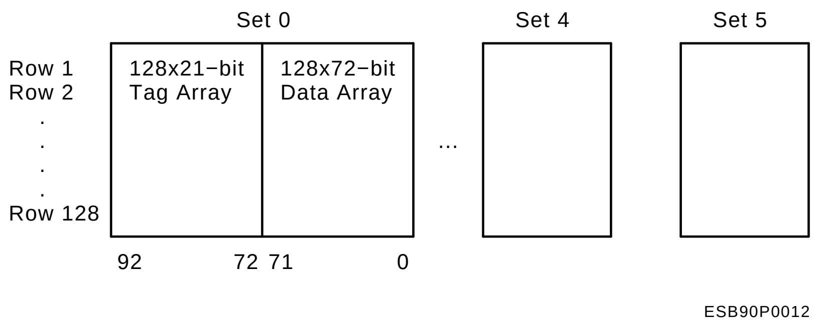
If the CPU is retried on the second (fill) longword then the first longword is delivered to the CPU but not cached. The request will not be retried if the need for the data is resolved (for example, it was a prefetch but a branch was taken).

4.2 Cache Organization

The cache is logically organized as 6, 1KB banks as shown below. It is a read-allocate, no-write-allocate, and write-through cache. A single parity bit is generated, stored and checked for each byte of data and each tag. The cache is enabled/disabled via a bit in the Cache Control Register (CCR).

Other bits in these two registers allow the data and tags to be addressed directly, to check the parity generating/checking logic and to provide the hit/miss status of each bank for the most recent D-stream read or write cycle.

Figure 4–1: Logical Organization of Cache



Each row within a set corresponds to a cache entry, and there are 128 entries in each set. Each entry contains a 21-bit Tag Block and a 72-bit (eight-byte) Data Block. A cache entry is logically organized as shown in Figure 4–2:

Figure 4–2: Cache Entry

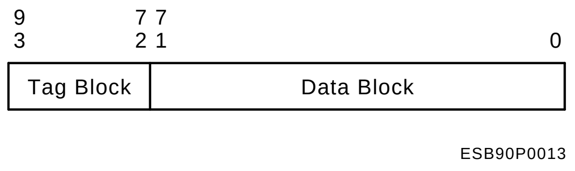
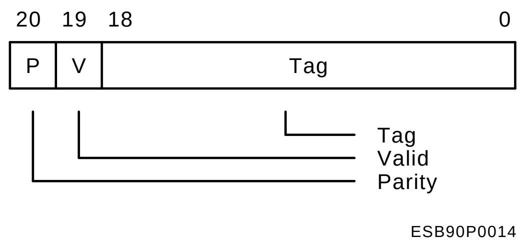


Figure 4–3 shows the format of a cache tag entry.

Figure 4–3: Cache Tag Entry



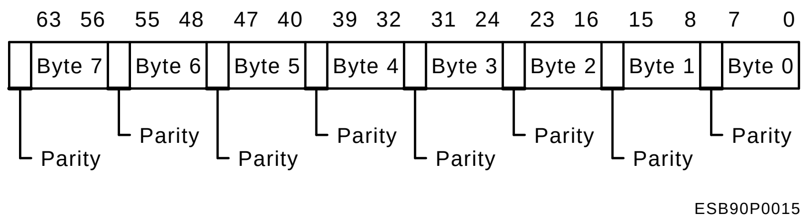
The Parity bit stores odd parity over the Tag field only. The Valid bit is not included in the parity calculation.

The Valid bit indicates whether or not the corresponding entry in the cache refers to a useable data/address pair.

The Tag consists of bits <28:10> of the physical address.

Figure 4–4 shows the format of a cache data entry.

Figure 4–4: Cache Data Entry



Each Data entry consists of eight bytes of data, each with an associated parity bit. Odd parity is generated for the odd data bytes and even parity is generated for the even data bytes.

4.2.1 Cache Address Translation

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

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

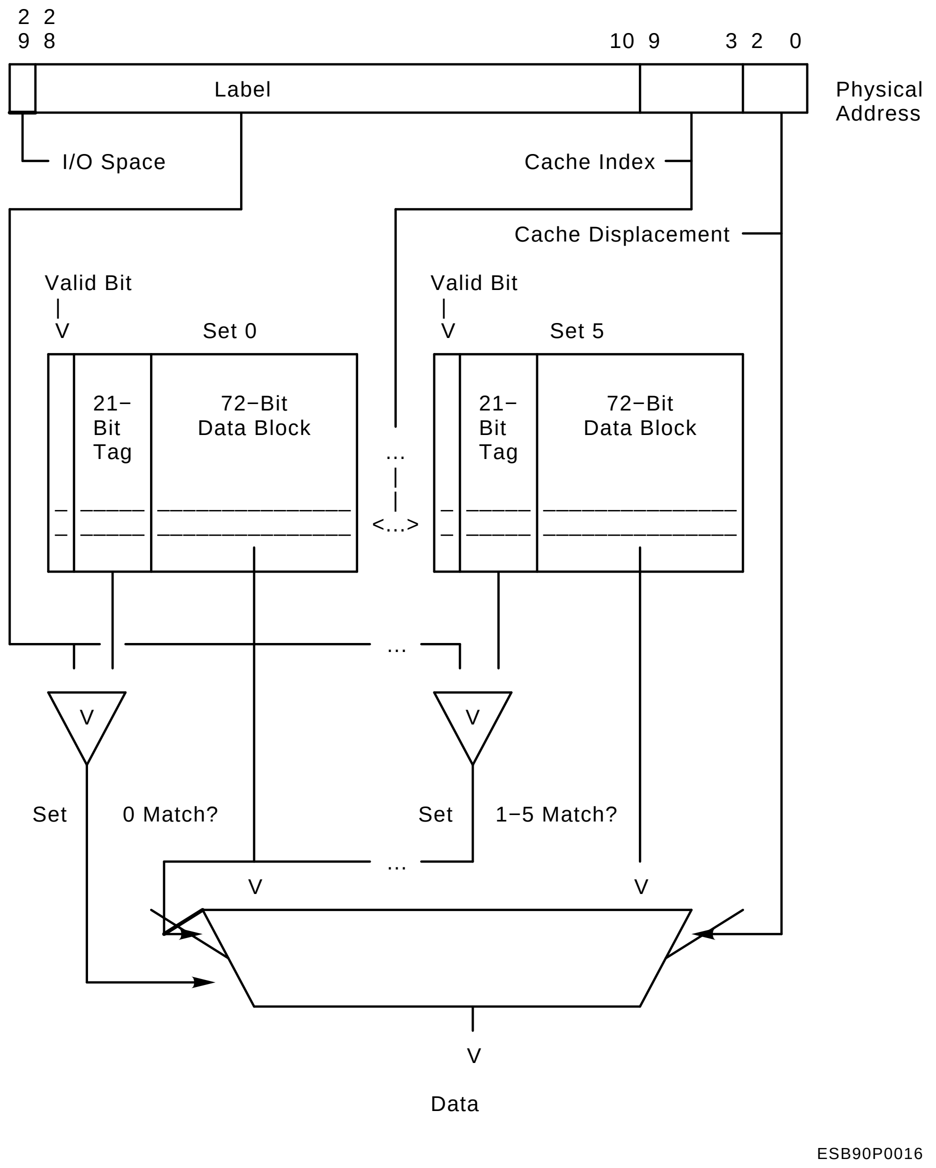
On cacheable references, the physical address must be translated to determine if the contents of the referenced location is resident in the cache. The Cache Index Field, bits <9:3> of the physical address, is used to select one of the 128 rows of the cache, with each row containing a single entry from each set. The Label Field, bits <28:10> of the physical address, is then compared to the Tag Block of the entry from all 6 sets in the selected row.

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

If no match occurs, then the contents of the referenced location is not contained in the cache and a cache "miss" occurs. In this case, the data must be obtained from the main memory controller, so a quadword transfer is initiated on the CDAL Bus.

Figure 4–5 shows how cache addresses are translated.

Figure 4–5: Cache Address Translation



4.2.2 Cache Data Block Allocation

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

Because the cache is six-way associative, there are six data blocks (one in each set) that can be allocated to a given quadword. The Cache Index field determines which row of 6 data blocks is to be used while set selection is determined by a Not Most Recently Used (NMRU) algorithm. The bank to be selected is pointed to by a three bit counter. The counter is set to 000 when the cache is disabled (CCR ENABLE_CACHE=0); it is advanced:

- Every time a block is allocated
- Every time a read or write hits in the bank to which the counter is currently pointing.

The counter counts modulo 6 with no missing counts regardless of which banks are enabled. The contents of the counter is driven out on pins TEST<2:0> (test mode is in Observe Miscellaneous State), which enables external logic to track which addresses are cached internally.

4.2.3 Cache Behavior on Writes

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

When a DMA write references hits the cache, the cache entry containing the copy of the referenced location is invalidated.

4.2.4 Cache Control Register (CCR, IPR 37)

The Cache Control Register (CCR), Internal Processor Register 37, controls the mode of operation of the cache, flushing the cache and is unique to CPU designs that use the SOC CPU chip. Figure 4–6 shows the register format and Table 4–1 describes the bits.

Figure 4–6: Cache Control Register

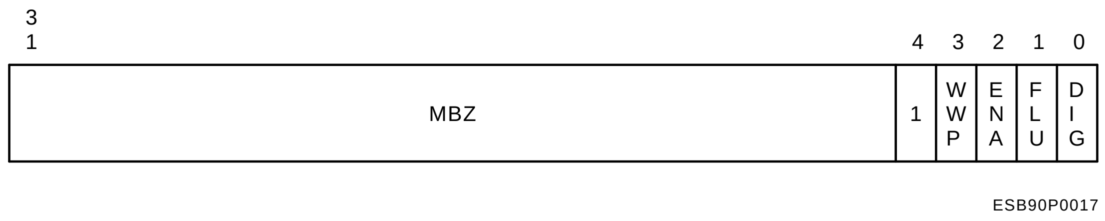


Table 4–1:

Data Bit	Name	Description	Type
<31:5>	Unused	These bits always read as zeros. Writes have no effect.	
<4>	Unused	This bit is always read as a one. Writes have no effect.	
<3>	WWP	Write Wrong Parity. For diagnostic use. When set, it inverts the data parity bits accessed from the cache. This will cause a parity error when they are compared to parity computed from the accessed data, and is used to test the parity generation/checking logic. When clear data parity bits are not affected. This bit does not affect tag parity bits. Cleared when RESET_L is asserted.	Read/Write

Table 4–1 (Cont.):

Data Bit	Name	Description	Type
<2>	ENA	Enable Cache. It enables or disables normal operation of the cache. When set, both I-stream and D-stream references are cached in the enabled banks of the cache, and cache tag and data parity errors are reported. When clear, all references (read and write) result in a miss and no cache parity is checked. When Enable Cache is set the cache should be flushed by writing a one to the Flush Cache bit. Enable Cache is cleared when RESET_L is asserted. **NOTE** The cache may be operated with both the Diagnostic bit set and the Enable Cache bit set.	Read/Write
<1>	FLU	Flush Cache. Always read as zero. Writing a one to this bit clears all valid bits in the cache tag array. Writing a zero has no effect.	Write Only
<0>	DIA	Diagnostic. When this bit is set, the cache entries and BEHR register may be accessed via a region in I/O space. When clear, references to the same region of I/O space result in bus cycles.	Read/Write

Table 4–2: Cache Diagnostic Mode Addresses

Information	Address Range
Cache Tag	20150000 - 201503FF
Cache Data	20150400 - 201507FF
BEHR	20150800 - 20150FFF

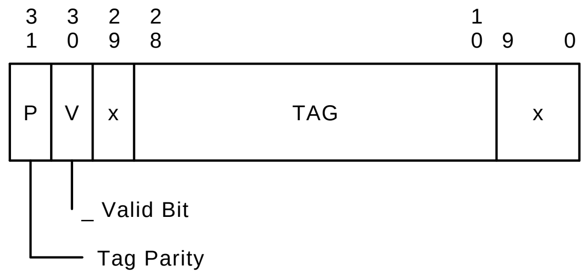
Note the BEHR register may be accessed at multiple addresses.

When the Diagnostic bit is set, writes to cache data addresses will write to the cache data longword indexed by bits <9:2> of the address. All banks which have the corresponding BEHR ENABLE_BANK bit set will be written, and correct data parity will also be written. Byte and word writes as well as longword are possible.

Reads from cache data addresses read the cache data longword addressed; if more than one ENABLE_BANK bit is set then the highest priority bank enabled will return the data (bank 0 is highest priority, bank 7 is lowest).

Writes to cache tag addresses write to the cache tag indexed by bits <9:3> of the address. All banks which have the corresponding BEHR ENABLE_BANK bit set will be written. The write data format is shown in Figure 4–7.

Figure 4–7: Tag Diagnostic Write Data Format

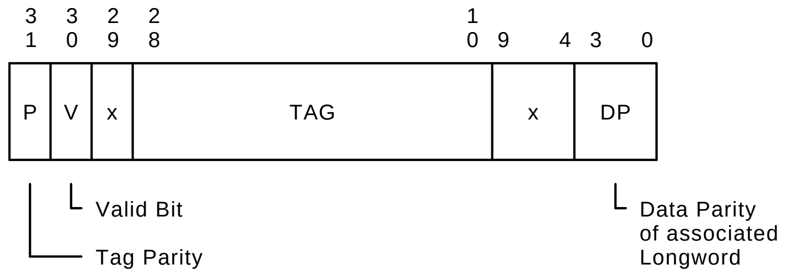


ESB90P0018

Reads from cache tag addresses read the cache tag addressed, plus the data parity from the data longword addressed by bits <9:2> of the address as shown below. If more than one ENABLE_BANK bit is set then the highest priority bank enabled will return the tag.

The read data format is shown in Figure 4–8.

Figure 4–8: Tag Diagnostic Read Data Format



ESB90P0019

4.2.5 Bank Enable/Hit Miss Register (BEHR)

The BEHR register allows individual sets of the cache to be enabled/disabled and also provides bits which indicate the hit/miss status for each set of the cache. The format is shown in Figure 4–9.

Figure 4–9: Bank Enable/Hit Miss Register

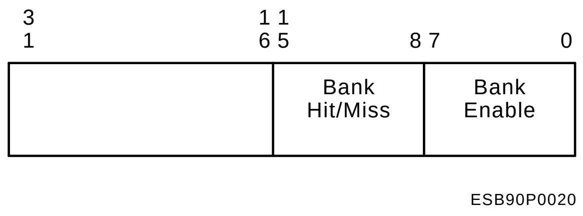


Table 4–3: Bank Enable/Hit Miss Register (BEHR)

Data Bit	Name	Description	Type
<15:8>	Bank Hit	These bits are provided for use in testing the cache. They represent the hit/miss status of each bank of the cache for the most recent D-stream read or write cycle. A '1' indicates that there was a hit in the corresponding bank, a '0' indicates a miss. Bank Hit<7:0> are cleared when RESET_L is asserted.	Read only

Table 4–3 (Cont.): Bank Enable/Hit Miss Register (BEHR)

Data Bit	Name	Description	Type
<7:0>	Enable Bank	These bits are written by the power-up diagnostics. The diagnostics are responsible for determining and enabling good cache banks. The diagnostics search for a "good" cache bank and enable it by If this filed contains a 1, this bank is enabled. When set, these bits enable banks <7:0> of the cache, respectively. Note that for any bank to be enabled, CCR Enable Cache or CCR Diagnostic must be set. When clear, the corresponding banks are disabled. Enable Bank <7:0> are cleared when RESET_L is asserted. Whenever a bank is either enabled or disabled, software should also flush the cache by writing a one to CCR Flush. **For any reference there should be a hit in only one bank. If, due to a hardware malfunction (corrupted tag bit), the reference hits in more than one bank, only the bank with the higher priority drives the data. The banks are numbered in decreasing order of priority. Bank 0 has the highest priority and bank 5 has the lowest priority. Normally, the corrupted tag would be reported via the MSER**	Read Only

4.2.6 Memory System Error Register (MSER, IPR 39)

The Memory System Error Register (MSER), Internal Processor Register 39, reports information about DAL bus and cache errors. The two basic classes of errors reported are:

- Errors that don't immediately affect operation of the SOC CPU and will therefore post an interrupt but not change instruction flow.
- Errors that do affect operation and will cause a machine check (trap through SCB vector 4.)

The format is shown in Figure 4–10.

Figure 4–10: Memory System Error Register (MSER, IPR 39)

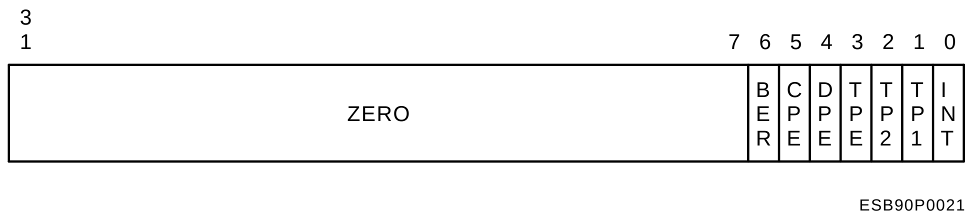


Table 4–4: Memory System Error Register (MSER, IPR 39)

Data Bit	Name	Description	Type
<31:7>		Always read as zero	
<6>	DPE	DAL Parity Error – This bit is set when a DAL parity error is detected on either a demand or request read cycle which receives a normal termination response (RDY_L asserted, ERR_L deasserted).	sticky
<5>	MCADPE	Machine Check Abort—DAL parity Error - This bit is set whenever a machine check is caused by a DAL parity Error. A DAL parity error will only cause a machine check on a demand read cycle.	sticky
<4>	MCACPE	Machine Check Abort Cache Parity Error - This bit is set whenever a machine check is caused by a cache parity error (tag or data). A cache parity error will only cause a machine check on a demand read cycle that hits the cache.	sticky
<3:2>		Always read as zero	
<1:0>		These bits are set independandtly to show the scope of a cache parity error on either a demand or request cycle, MSER<0> is set to indicate that the cache parity error was caused by a tag error; MSER<1> by a data error. Note that a simultaneuos cache tag and data parity error will only log the fact that a cache tag parity error occured.	sticky

NOTE: MSER bits <6:4, 1:0> are "sticky" in the sence that once set, they will remain set until MSER is explicitly cleared by writing the MSER (with a MTPR instruction) irrespective of the data, Parity errors occuring while an error condition is posted in MSER can only set an additional bit (For example MSER <6:4, 1:0> cannot be cleared through subsequent errors.)

4.2.7 Cache Error Detection

Both the tag and data arrays in the cache are protected by parity. Each 8-bit byte of data and the 19-bit tag is stored with an associated parity bit. The valid bit in the tag is not covered by parity. Odd data bytes are stored with odd parity and even data bytes are stored with even parity. The tag is stored with odd parity. The stored parity is valid only when the valid bit associated with the cache entry is set. Tag and data parity (on the entire longword) are checked on read references that hit the cache, while only tag parity is checked on CPU and DMA write references that hit the cache.

The action taken following the detection of a cache parity error depends on the reference type. The following list outlines the different reference types:

- During a demand D-stream read reference, the entire cache is flushed, the CCR is cleared (which disables the cache.) The cause of the error is logged in MSER<4,1:0> and a machine check abort is initiated.
- During a request I-stream read reference, the entire cache is flushed (unless CCR<0> is set), the cause of the error is logged in MSER<1:0>, the prefetch is halted, but no machine check abort occurs, and the cache remains enabled.
- During a masked or unmasked write reference, the entire cache is flushed (unless CCR<0> is set), the cause of the error is logged in MSER<0> (only tag parity is checked on CPU writes that hit the cache) there is no effect on CPU execution, and the cache remains enabled.
- During a DMA Write reference the cause of the error is logged in MSER<0> (only tag parity is checked on DMA writes that hit the cache) there is no effect on CPU execution, both caches remain enabled, and no invalidate operation occurs.

Chapter 5

KA660 Main Memory System

The KA660-AA includes a main memory controller implemented via a single VLSI chip called the CMCTL. The KA660 Main Memory Controller communicates with the MS650 memory boards over the MS650 Memory Interconnect, which uses the CD interconnect for the address and control lines and a 50-pin, ribbon cable for the data lines. It supports up to 4 (four) MS650 memory boards, for a maximum of 64MB of ECC memory.

NOTE

The KA660 supports only MS650_B* variations and does not support MS650-AA variations.

The memory controller supports synchronous longword read references, and masked or unmasked synchronous write references generated by the CPU as well as synchronous, quadword read references generated by cacheable CPU references that miss the cache.

5.0.1 KA660 Timing

The system clock for the VAX 4000-200 operates at 114.285 MHz creating a cycle time of 70 ns. Table 5–1 lists the reference times for CPU reads and writes, and Q22-bus interface reads and writes. This table also provides information about error handling times.

Table 5–1: KA660 Reference Timing

CPU Read Reference	
longword	280 ns
quadword	420 ns
first longword	280 ns
second longword	140 ns

Table 5–1 (Cont.): KA660 Reference Timing

aborted reference	280 ns
longword (locked)	630 ns min
aborted reference	280 ns
retry (locked)	350 ns
CPU Write Reference	
longword	140 ns
longword (masked)	350 ns

The controller also supports asynchronous longword and quadword DMA read references and masked and unmasked asynchronous longword, quadword, hexword, and octaword DMA write references from the Q22-bus Bus Interface.

Q22-bus Interface Read Reference	
longword	420 ns
quadword	560 ns
first longword	350 ns
second longword	210 ns
longword (locked)	420 ns
Q22-bus Interface Write Reference	
longword	280 ns
longword (masked)	420 ns
quadword	490 ns
first longword	2800 ns
second longword	210 ns
quadword (masked)	770 ns
first longword	280 ns
second longword	490 ns
hexword	700 ns
first longword	280 ns
second longword	210 ns
third longword	210 ns
hexword (masked)	980 ns
first longword	280 ns
second longword	210 ns
third longword	4900 ns
octaword	910 ns

Table 5–1 (Cont.): KA660 Reference Timing

first longword	280 ns
second longword	210 ns
third longword	210 ns
fourth longword	2100 ns
octaword (masked)	1190 ns
first longword	280 ns
second longword	210 ns
third longword	210 ns
fourth longword	490 ns

The above timing assumes no exception conditions are encountered during the reference. Exception conditions will add the following amount of time if they are encountered during a reference.

Error Handling	
correctable error	70 ns
uncorrectable error	140 ns-read
uncorrectable error	70 ns-write
CDAL parity error	70 ns-write
refresh collision	280 ns

5.0.2 Main Memory Organization

Main memory is logically and physically divided into four boards which correspond to the four possible MS650 memory expansion modules that can be attached to a KA660-AA. Each memory board can contain zero (no memory module present) or four (MS650-BA present), memory banks. Each bank contains 1,048,576 (1M) aligned longwords. Each aligned longword is divided into four data bytes and is stored with seven ECC check bits, resulting in a memory array width of 39 bits.

5.0.3 Main Memory Addressing

The KA660-AA Main Memory Controller is capable of controlling up to 16 banks of RAM, each bank containing 4MB of storage. Each bank of main memory has a programmable base address, determined by the state of bits <25:22> of the Main Memory Configuration Register associated with the bank.

A 4MB bank is accessed when bit <29> of the physical address is equal to zero, indicating a VAX Memory Space read/write reference, bits <28:26> of the physical address are equal to zero, indicating a reference within the range of the main memory controller, and the bank number of the bank matches bits <25:22> of the physical address. The remainder of the physical address (bits <21:2>) are used to determine the row and column of the desired longword within the bank. The Byte Mask lines are ignored on read operations, but are used to select the proper byte(s) within a longword during masked longword write references.

5.1 Main Memory Behavior on Writes

On unmasked CPU write references, the main memory controller operates in "dump and run" mode, terminating the CDAL Bus transaction after latching the data, but before checking CDAL Bus parity, calculating the ECC check bits, and transferring the data to main memory.

On unmasked DMA write references by the Q22-bus Bus Interface, the data is latched, CDAL Bus parity is NOT checked, the CDAL Bus transaction is terminated, the ECC check bits are calculated, and the data is transferred to main memory.

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

On multiple transfer masked DMA writes, each longword write is acknowledged then the CDAL transaction is terminated.

5.2 Main Memory Error Status Register (MEMCSR16)

The Main Memory Error Status Register, address 2008 0140_{16}, is used to capture main memory error data. This register is unique to CPU designs that use the CMCTL memory controller chip.

Data Bit	Name	Description	Type
<31>	RDS ERROR	When set, an uncorrectable ECC error occurred during a memory read or masked write reference. Cleared by writing a one to it. Writing a zero has no effect. Undefined if MEMCSR16<7> (CDAL BUS ERROR) is set. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write to clear
<30>	RDS HIGH ERROR RATE	When set, an uncorrectable ECC error occurred while the RDS ERROR LOG REQUEST bit was set, indicating multiple uncorrectable memory errors. Cleared by writing a one to it. Writing a zero has no effect. Undefined if MEMCSR16<7> (CDAL BUS ERROR) is set. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write to Clear.
<29>	CRD ERROR	When set, a correctable (single bit) error occurred during a memory read or masked write reference. Cleared by writing a one to it. Writing a zero has no effect. Undefined if MEMCSR16<7> (CDAL BUS ERROR) is set. Cleared by writing one, on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write to Clear.

Data Bit	Name	Description	Type
<28:9>	PAGE ADDRESS OF ERROR	This field identifies the page (512 byte block) containing the location that caused the memory error. In the event of multiple memory errors, the types of errors are prioritized and the page address of the error with the highest priority is captured. Errors with equal priority do not overwrite previous contents. Writes have no effect. Cleared on power-up and the negation of DCOK when SCR<7> is clear. The types of error conditions follow in order of priority: • CDAL Bus parity errors during a CPU write reference, as logged by the CDAL BUS ERROR bit. • Uncorrectable ECC errors during a CPU or DMA read or masked write reference, as logged by the RDS ERROR LOG bit. • Correctable ECC errors during a CPU or DMA read or masked write reference, as logged by CRD ERROR bit.	Read Only.
<8>	DMA ERROR	When set, an error occured during a DMA read or write reference. Cleared by writing a one to it. Writing a zero has no effect. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write to Clear.
<7>	CDAL BUS ERROR	When set, a CDAL Bus parity error occurred on a CPU write reference. Cleared by writing a one to it. Writing a zero has no effect. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write to Clear.
<6:0>	ERROR SYNDROME	This field stores the error syndrome. A non-zero syndrome indicates a detectable error has occured. A unique syndrome is generated for each possible single bit (correctable) error. A list of the these syndromes and their associated single bit errors appears on the next page. Any non-zero syndrome that is not contained on this list indicates a multiple bit (uncorrectable) error has occured. This field handles multiple errors in the same manner as MEMCSR16<28:9>. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read Only.

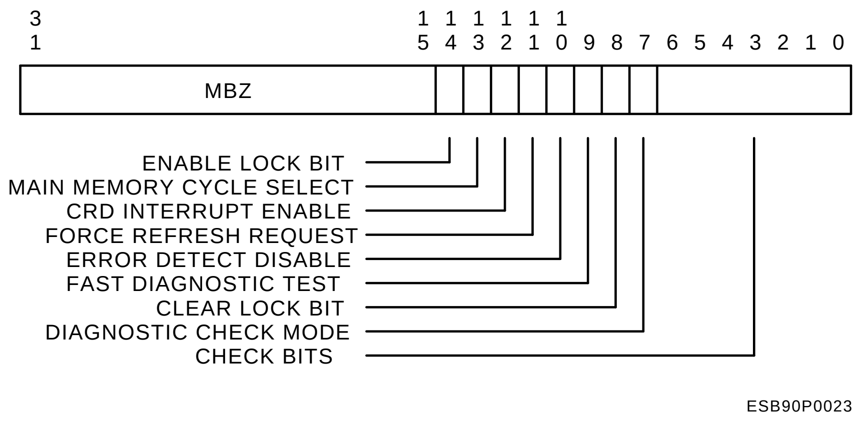
The following is a list of the syndromes and their associated single bit errors:

```
SYNDROME<6:0>                BIT POSITION IN ERROR
=============                =====================
   0000000                     no error detected

                                         Data Bits (0-31 decimal)
   1011000                              0    |
   0011100                              1    |
   0011010                              2    V
   1011110                              3
   0011111                              4
   1011011                              5
   1011101                              6
   0011001                              7
   1101000                              8
   0101100                              9
   0101010                             10
   1101110                             11
   0101111                             12
   1101011                             13
   1101101                             14
   0101001                             15
   1110000                             16
   0110100                             17
   0110010                             18
   1110110                             19
   0110111                             20
   1110011                             21
   1110101                             22
   0110001                             23
   0111000                             24
   1111100                             25
   1111010                             26
   0111110                             27
   1111111                             28
   0111011                             29
   0111101                             30
   1111001                             31
                                          Check Bits (32-38 decimal)
   0000001                             32      |
   0000010                             33      |
   0000100                             34      V
   0001000                             35
   0010000                             36
   0100000                             37
   1000000                             38

   0000111                         Result of incorrect check
                                   bits written on detection of
                                   a CDAL parity error.

   All others                      Multi-bit errors
```

5.3 Main Memory Control and Diagnostic Status Register (MEMCSR17)

The Main Memory Control and Diagnostic Status Register, address 2008 0144 $_{16}$, is used to control the operating mode of the main memory controller as well as to store diagnostic status information. This register is unique to CPU designs that use the CMCTL memory controller chip. Figure 5–1 shows the format of this register.

Figure 5–1: Main Memory Control and Diagnostic Status Register (MEMCSR17)



Data Bit	Name	Description	Type
<31:15>	Unused	This field reads as 0, must be written as 0.	
<14>	ENABLE LOCK BIT	When cleared, the main memory locking function (reflected by the LOCK BIT in MEMCSR15-0<6>) is disabled. When set, the main memory locking function (reflected by the LOCK BIT in MEMCSR15-0<6>) is enabled. Writing this bit has no effect on MEMCSR15-0<6>. This bit should always be clear, because the KA660-AA implements the main memory locking function in the Q22-bus Interface chip (CQBIC). Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write.

Data Bit	Name	Description	Type
<13>	MAIN MEMORY CYCLE SELECT	When set, longword reads and the first longword in quadword reads occur in four CPU cycles (320ns) and the second longword in a quadword read occurs in two CPU cycles (160ns). When cleared, longword reads and the first longword in quadword reads occur in five CPU cycles (450ns) and the second longword in a quadword read occurs in three CPU cycles (240ns). **NOTE** With the KA660-AA this bit must be cleared by the firmware memory configuration routine thus using the 5/3 timing. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write
<12>	CRD INTERRUPT ENABLE	When cleared, single-bit errors are corrected by the ECC logic, but no interrupt is generated. When set, single-bit errors are corrected by the ECC logic and they cause an interrupt to be generated at IPL 1A with a vector of 54_{16}, . This bit has no effect on the capturing of error information in MEMCSR16, or on the reporting of uncorrectable errors. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write.
<11>	FORCE REFRESH REQUEST	When cleared, the refresh control logic operates in normal mode (refresh every 9.1us for 80ns cycles and 1 wait state). When set, one memory refresh operation occurs immediately after the MEMCSR write reference that set this bit. Setting this bit provides a mechanism for speeding up the testing of the refresh logic during manufacturing test of the controller chip. This bit is cleared by the memory controller upon completion of the refresh operation. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write.
<10>	MEMORY ERROR DETECT DISABLE	When set, error detection and correction (ECC) is disabled, so all memory errors go undetected. When cleared, error detection, correction, state capture and reporting (via MEMCSR16) is enabled. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write.

Data Bit	Name	Description	Type
<9>	FAST DIAGNOSTIC TEST	This bit provides a mechanism for speeding up the diagnostic testing of main memory. When set, all main memory banks are read and written in parallel. When cleared, this bit has no effect on memory reads or writes. Cleared on power-up and the negation of DCOK when SCR<7> is clear. **NOTE** Due to excess power consumption do not use MOVC instructions in fast diagnostic test mode.	Read/Write.
<8>	CLEAR LOCK BIT	Writing a one to this bit clears MEMCSR15-0<6> and "unlocks" main memory. Always read as zero. This bit is used to unlock memory that was locked by an interlocked instruction that was aborted, due to an error condition, before it could unlock memory. This bit should never be needed, because the KA660-AA implements the main memory locking function in the Q22-bus Interface chip (CQBIC). Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Write Only.
<7>	DIAGNOSTIC CHECK MODE	When set, the contents of MEMCSR17<6:0> are written into the 7 ECC check bits of the location (even if a CDAL parity error is detected) during a memory write reference. When cleared, the 7 check bits calculated by the ECC generation logic are loaded into the 7 ECC check bits of the location during a write reference and a memory read reference will load the state of the 7 ECC check bits of the location that was read into MEMCSR17<6:0>. Cleared on power-up and the negation of DCOK when SCR<7> is clear. **NOTE** DIAGNOSTIC CHECK MODE is restricted to unmasked memory write references. No masked write references are allowed when DIAGNOSTIC CHECK MODE is enabled.	Read/Write.

Data Bit	Name	Description	Type
<6:0>	CHECK BITS	When the DIAGNOSTIC CHECK MODE bit is set, these bits are substituted for the check bits that are generated by the ECC generation logic during a write reference. When the DIAGNOSTIC CHECK MODE bit is cleared, memory read references load the state of the 7 ECC check bits of the location that was read into MEMCSR16<6:0>. Cleared on power-up and the negation of DCOK when SCR<7> is clear.	Read/Write.

5.4 Main Memory Error Detection and Correction

The KA660-AA Main Memory Controller generates CDAL Bus parity on CPU read references, and checks CDAL Bus parity on CPU write references.

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

For unmasked CPU write references, incorrect check bits are written to main memory (potentially masking an as yet undetected memory error) along with the data and an interrupt is generated at IPL 1D through vector 60_{16}, on the next cycle and MCSR16<7> is set. The incorrect check bits are determined by calculating the seven "correct" check bits, and complementing the three least significant bits.

For masked CPU write references, incorrect check bits are written to main memory (potentially masking an as yet undetected memory error) along with the data, unless an uncorrectable error is detected during the read portion, MEMCSR16<7> is set, and a machine check abort is initiated. If an uncorrectable error is detected on the read portion, no write operation takes place. The incorrect check bits are determined by calculating the seven "correct" check bits, and complementing the three least significant bits.

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

Upon detecting a correctable error on a read reference or the read portion of a masked write reference, the data is corrected (if it is in the data field), before placing it on the CDAL Bus, or back in main memory, an interrupt is generated at IPL 1A through vector 54_{16}, , bit <29> of MEMCSR16 is set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, and bits <6:0> are loaded with the error syndrome which indicates which bit was in error. If the error was detected on a DMA reference, MEMCSR16<8> is also set.

NOTE

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

Upon detecting an uncorrectable error, the action depends on the type of reference being performed. Table 5–2 lists the actions performed on uncorrectable errors.

Table 5–2: Uncorrectable Error Actions

Error	Action Performed
On a demand read reference	The affected row of the cache is invalidated. Bit <31> of MEMCSR16 is set. bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error, bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable. A machine check abort is initiated. If the read was a local-miss, global-hit read, or a read of the Q22-bus Bus map, MEMCSR16<8> and DSER<4> are also set, and DEAR<12:0> are loaded with the address of the page containing the location that caused the error.
On a request read reference	The prefetch or fill cycle is aborted, but no machine check occurs. Bit <31> of MEMCSR16 is set. Bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error. Bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable.
On the read portion of masked write reference	Bit <31> of MEMCSR16 is set. Bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error. Bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable. A machine check abort is initiated.

Table 5–2 (Cont.): Uncorrectable Error Actions

Error	Action Performed
On a DMA read reference	Bit <31> and bit <8> of MEMCSR16 are set, bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error. Bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable, DSER<4> is set, DEAR<12:0> are loaded with the address of the page containing the location that caused the error. BDAL<17:16> are asserted on the Q22-bus Bus along with the data to notify the receiving device (unless it was a map read by the Q22-bus bus interface during translation). An interrupt is generated at IPL 1D through vector 60_{16}.
On a DMA masked write reference	Bit<31> and bit <8> of MEMCSR16 are set. Bits <28:9> of MEMCSR16 are loaded with the address of the page containing the location that caused the error. Bits <6:0> are loaded with the error syndrome which indicates that the error was uncorrectable DSER<4> is set. DEAR<12:0> are loaded with the address of the page containing the location that caused the error. ICR<15> is set to notify the initiating device An interrupt is generated at IPL 1D through vector 60_{16}, .

Chapter 6

KA660 Console Serial Line

The console serial line provides the KA660 processor with a full duplex, RS-423 EIA, serial line interface, which is also RS-232C compatible. The only data format supported is 8-bit data with no parity and one stop bit. The four Internal Processor Registers (IPR's) that control the operation of the console serial line are a superset of the VAX Console Serial Line Registers described in the *VAX Architecture Reference manual* .

6.1 Console Registers

There are four registers associated with the Console Serial Line Unit. They are implemented in the SSC chip and are accessed as IPR's 32-35. Refer to Table 6–1.

Table 6–1: Console Registers

IPR Number		Register Name	Mnemonic
Dec	**Hex**		
32	20	Console Receiver Control/Status	RXCS
33	21	Console Receiver Data Buffer	RXDB
34	22	Console Transmit Control/Status	TXCS
35	23	Console Transmit Data Buffer	TXDB

6.1.1 Console Receiver Control/Status Register - (IPR 32)

The Console Receiver Control/Status Register (RXCS), Internal Processor Register 32, is used to control and report the status of incoming data on the console serial line. The format is shown in Figure 6–1. Table 6–2 lists the bit descriptions.

Figure 6–1: Console Receiver Control/Status Register - (IPR 32_{10} 20_{16})

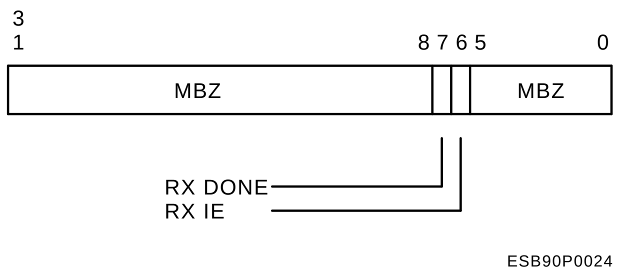


Table 6–2: Console Receiver Control/Status Register

Data Bit	Name	Description
<31:8>	MBZ	These bits Read as ZEROs, Writes have no effect.
<7>	RX DONE	Receiver Done. Read Only. Writes have no effect. This bit is set when an entire character has been received and is ready to be read from the RXDB Register. This bit is automatically cleared when the RXDB register is read. It is also cleared on power-up and the negation of DCOK.
<6>	RX IE	Receiver Interrupt Enable. Read/Write. When set, this bit causes an interrupt to be requested at IPL 14 with an SCB offset of F8 If RX DONE is set. When cleared, interrupts from the Console Receiver are disabled. This bit is cleared on power-up and the negation of DCOK when CSR<7> is clear .
<5:0>	Unused	These bits read as ZEROs. Writes have no effect.

6.1.2 Console Receiver Data Buffer - (IPR 33)

The Console Receiver Data Buffer (RXDB), internal processor register 33, is used to buffer incoming data on the serial line and capture error information. The format is shown in Figure 6–2. Bit descriptions are listed in Table 6–3.

Figure 6–2: Console Receiver Data Buffer - (IPR 33_{10} 21_{16})

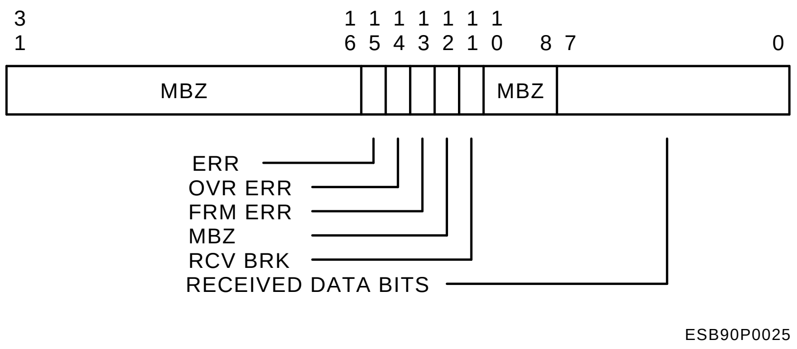


Table 6–3: Console Receiver Data Buffer

Data Bit	Name	Description
<31:16>	MBZ	These bits always read as ZERO. Writes have no effect.
<15>	ERR	Error. Read only. Writes have no effect. This bit is set if RBUF <14> or <13> is set. It is clear if these two bits are clear. This bit cannot generate a program interrupt. Cleared on power-up and the negation of DCOK.
<14>	OVR ERR	Overrun Error. Read only. Writes have no effect. This bit is set if a previously received character was not read before being overwritten by the present character. Cleared by reading the RXDB, on power-up and the negation of DCOK.

Table 6–3 (Cont.): Console Receiver Data Buffer

Data Bit	Name	Description
<13>	FRM ERR	Framing Error. Read only. Writes have no effect. This bit is set if the present character did not have a valid stop bit. Cleared by reading the RXDB, on power-up and the negation of DCOK. **Error conditions are updated when the character is received and it remains present until the character is read, at which point, the error bits are cleared.**
<12>	MBZ	This bit always reads as ZERO. Writes have no effect.
<11>	RCV BRK	Received Break. Read only. Writes have no effect. This bit is set at the end of a received character for which the serial data input remained in the SPACE condition for 20 bit times. Cleared by reading the RXDB, on power-up and the negation of DCOK.
<10:8>	MBZ	These bits always read a as ZERO. Writes have no effect.
<7:0>	Received Data Bits.	Read only. Writes have no effect. These bits contain the last received character.

6.1.3 Console Transmitter Control/Status Register - (IPR 34)

The Console Transmitter Control/Status Register (TXCS), Internal Processor Register 34, is used to control and report the status of outgoing data on the console serial line. The format is shown in Figure 6–3. Bit descriptions are listed in Table 6–4.

Figure 6–3: Console Transmitter Control/Status Register - (IPR 34_{10} 22_{16})

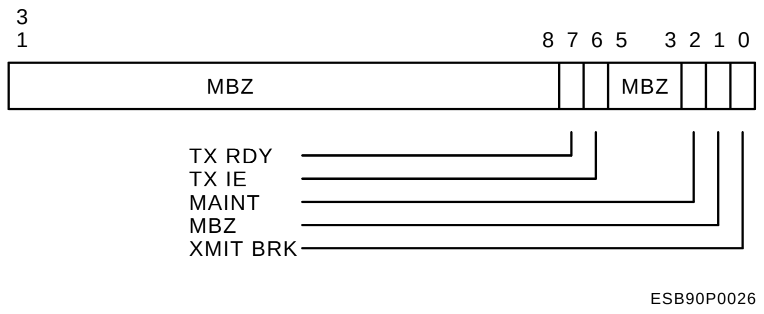


Table 6–4: Console Transmitter Control/Status Buffer

Data Bit	Name	Description
<31:8>	MBZ	These bits read as ZEROs. Writes have no effect.
<7>	TX RDY	Transmitter Ready. Read only. Writes have no effect. This bit is cleared when TXDB is loaded and set when TXDB can receive another character. This bit is set on power-up and the negation of DCOK.
<6>	TX IE	Transmitter Interrupt Enable. Read/Write. When set, this bit causes an interrupt to be requested at IPL 14 with an SCB offset of FC if TX RDY is set. When cleared, interrupts from the Console Receiver are disabled. This bit is cleared on power-up and the negation of DCOK.
<5:3>	MBZ	Read as zeros. Writes have no effect.
<2>	MAINT	Maintenance.Read/Write. This bit is used to facilitate a maintenance self-test. When MAINT is set, the external serial output is set to MARK and the serial output is used as the serial input. This bit is cleared on power-up and the negation of DCOK.
<1>	Unused	This bit read as ZERO. Writes have no effect.
<0>	XMIT BRK	Transmit Break. ead/Write. When this bit is set, the serial output is forced to the SPACE condition after the character in TXDB<7:0> is sent. While XMIT BRK is set, the transmitter will operate normally, but the output line will remain low. Thus, software can transmit dummy characters to time the break. This bit is cleared on power-up.

6.1.4 Console Transmitter Data Buffer - (IPR 35)

The Console Transmitter Data Buffer (TXDB), Internal Processor Register 35, is used to buffer outgoing data on the serial line. The format is shown in Figure 6–4. Table 6–5 lists the bit descriptions.

Figure 6–4: Console Transmitter Data Buffer - (IPR 35_{10} 23_{16})

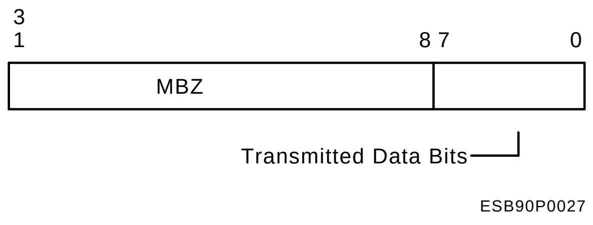


Table 6–5: Console Transmitter Data Buffer

Data Bit	Name	Description
<31:8>	MBZ	Read as 0. Writes have no effect.
<7:0>	Transmitted Data Bits	Write only. These bits are used to load the character to be transmitted on the console serial line.

6.2 Break Response

The console serial line unit recognizes a BREAK condition which consists of 20 consecutively received SPACE bits. If the console detects a valid break condition, the RCV BRK bit is set in the RXDB register. If the break was the result of 20 consecutively received SPACE bits, the FRM ERR bit is also set. If halts are enabled the KA660will halt and transfer program control to EPROM location 2004 0000_{16} when the RCV BRK bit is set. RCV BRK is cleared by reading RXDB. Another MARK followed by 20 consecutive SPACE bits must be received to set RCV BRK again.

6.3 Baud rate

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

The user selects the desired baud rate through the Baud Rate Select signals which are received from an external 8-position switch mounted on the console module (H3602) . The KA660firmware reads this code from Boot and Diagnostic Register bits <6:4> and compliments and then loads it into SSC Configuration Register bits <14:12>.

Table 6–6 shows the Baud Rate Selection, the corresponding code as read in the Boot and Diagnostic Register bits <6:4>, and the INVERTED code that should be loaded into SSC Configuration Register bits <14:12>:

Table 6–6: Baud Rate Selection

Baud Rate	BDR<6:4>	SSC<14:12>
300	111	000
600	110	001
1200	101	010
2400	100	011
4800	011	100
9600	010	101
19200	001	110
38400	000	111

6.4 Console Interrupt Specifications

The console serial line receiver and transmitter both generate interrupts at IPL 14. The receiver interrupts with a vector of $F8_{16}$, while the transmitter interrupts with a vector of FC_{16}.

Chapter 7

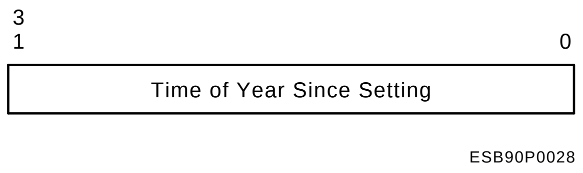
KA660 Clock and Timer Registers

The KA660 clocks include Time Of Year Clock (TOY), a subset Interval Clock (subset ICCS), as defined in the *VAX Architecture Reference Manual* and two additional programmable timers modeled after the VAX Standard Interval Clock.

7.1 Time-of-Year Clock (TOY) - EPR 27

The KA660 Time-of-Year clock (TOY) forms an unsigned 32-bit binary counter that is driven from a 100Hz oscillator, so that the least significant bit of the clock represents a resolution of 10 milliseconds, with less than .0025% error. The register counts only when it contains a non-zero value. This register is implemented in the SSC chip. The format is shown in Figure 7–1.

Figure 7–1: Time-of-Year Clock (TOY) - (EPR 27_{10} $1B_{16}$)



The Time Of Year Clock is maintained during power failure by battery backup circuitry which interfaces, via the external connector, to a set of batteries which are mounted on the CPU Console Module. The (TOY) will remain valid for greater than 162 hours when using the NiCad battery pack (3 batteries in series) mounted on the H3602 cover panel .

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

NOTE

After writing a non-zero value into the TOY, software should clear the BLO bit by writing a one to it.

7.2 Interval Timer (ICCS) - EPR 24

The KA660 Interval Timer (ICCS), Internal Processor Register 24, is implemented according to the *VAX Architecture Reference Manual*. The Interval Clock Control/Status Register (ICCS), is implemented as the standard subset of the Standard VAX ICCS in the CPU chip; while NICR and ICR are not implemented. Figure 7–2 shows the format and Table 7–1 describes the bits.

Figure 7–2: Interval Timer (ICCS) - (EPR 24_{10} 18_{16})

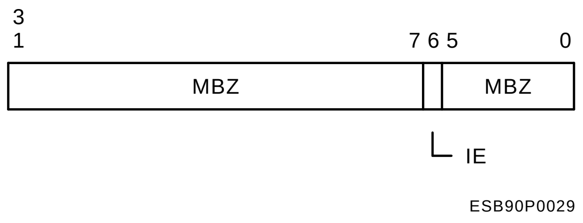


Table 7–1: Interval Timer Bit Descriptions

Data Bit	Name	Description
<31:7>	MBZ	Read as zeros, must be written as zeros.
<6>	IE	Interrupt Enable. Read/Write. This bit enables and disables the interval timer interrupts. When the bit is set, an interval timer interrupt is requested every 10 msec with an error of less than .01%. When the bit is clear, interval timer interrupts are disabled. This bit is cleared on power-up.
<5:0>	MBZ	Read as zeros, must be written as zeros.

Interval timer requests are posted at IPL 16 with a vector of C0: the interval timer is the highest priority device at this IPL.

7.3 Programmable Timers

The KA660 features two programmable timers. Although they are modeled after the VAX Standard Interval Clock, they are accessed as I/O space registers (rather than as Internal Processor Registers) and a control bit has been added which stops the timer upon overflow. If so enabled, the timers will interrupt at IPL 14 upon overflow. The interrupt vectors are programmable and are set to 78 and 7C by the firmware.

Each timer is composed of four registers:

a Timer **n** Control Register,
a Timer **n** Interval Register,
a Timer **n** Next Interval Register, and
a Timer **n** Interrupt Vector Register,

where **n** represents the timer number (0 or 1).

7.3.1 Timer Control Registers (TCR0 and TCR1)

The KA660 has two Timer Control Registers, one for controlling timer 0 (TCR0), and one for controlling timer 1 (TCR1). TCR0 is accessible at address 2014 0100_{16}, and TCR1 is accessible at 2014 0110_{16}. These registers are implemented in the SSC chip. Figure 7–3 shows the format. Table 7–2 lists the bit descriptions.

Figure 7–3: Timer Control Registers (TCR0 and TCR1)

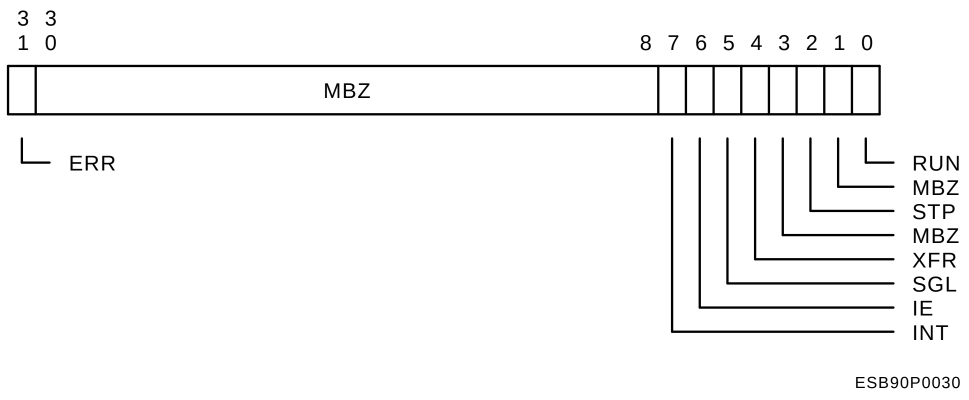


Table 7–2: Timer Control Register Bit Descriptions

Date Bit	Name	Description
<31>	ERR	Error. Read/Write to Clear. This bit is set whenever the Timer Interval Register overflows and the INT bit is already set. Thus, the ERR bit indicates a missed overflow. Writing a one to this bit clears it. Cleared on power-up.
<30:8>	MBZ	Read as zeros, must be written as zeros.
<7>	INT	Read/Write to Clear. This bit is set whenever the Timer Interval Register overflows. If IE is set when INT is set, an interrupt is posted at IPL 14. Writing a one to this bit clears it. Cleared on power-up.

Table 7–2 (Cont.): Timer Control Register Bit Descriptions

Date Bit	Name	Description
<6>	IE	Read/Write. When this bit is set, the timer will interrupt at IPL 14 when the INT bit is set. Cleared on power-up.
<5>	SGL	Read/Write. Setting this bit causes the Timer Interval Register to be incremented by one if the RUN bit is cleared. If the RUN bit is set, then writes to the SGL bit are ignored. This bit is always read as zero.Cleared on power-up.
<4>	XFR	Read/Write. Setting this bit causes the Timer Next Interval Register to be copied into the Timer Interval Register. This bit is always read as zero. Cleared on power-up.
<3>	MBZ	Read as zeros, must be written as zeros.
<2>	STP	Read/Write. This bit determines whether the timer stops after an overflow when the RUN bit is set. If the STP bit is set at overflow, the RUN bit is cleared by the hardware at overflow and counting stops. Cleared on power-up.
<1>	MBZ	Read as zeros, must be written as zeros.
<0>	RUN	Read/Write. When set, the Timer Interval Register is incremented once every microsecond. The INT bit is set when the timer overflows. If the STP bit is set at overflow, the RUN bit is cleared by the hardware at overflow and counting stops. When the RUN bit is clear, the Timer Interval Register is not incremented automatically. Cleared on power-up.

7.3.2 Timer Interval Registers (TIR0 and TIR1)

The KA660 has two Timer Interval Registers, one for timer 0 (TIR0), and one for timer 1 (TIR1). TIR0 is accessible at address 2014 0104_{16}, and TIR1 is accessible at 2014 0114_{16}.

The Timer Interval Register is a read only register containing the interval count. When the RUN bit is 0, writing a 1 increments the register. When the RUN bit is 1, the register is incremented once every microsecond. When the counter overflows, the INT bit is set, and an interrupt is posted at IPL14 if the IE bit is set. Then, if the RUN and STP bits are both set, the RUN bit is cleared and counting stops. Otherwise, the counter is reloaded. The maximum delay that can be specified is approximately 1.2 hours. This register is cleared on power-up. Figure 7–4 shows the format.

Figure 7–4: Timer Interval Registers (TIR0 and TIR1)

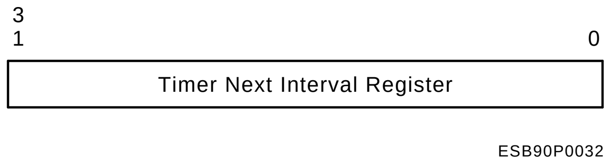
3
1
0
Timer Interval Register
ESB90P0031

7.3.3 Timer Next Interval Registers (TNIR0 and TNIR1)

The KA660 has two Timer Next Interval Registers, one for timer zero (TNIR0), and one for timer one (TNIR1). TNIR0 is accessible at address 2014 0108_{16}, and TNIR1 is accessible at 2014 0118_{16}. These registers are implemented in the SSC chip. The format is shown in Figure 7–5.

This Read/Write register contains the value which is written into the Timer Interval Register after overflow, or in response to a one written to the XFR bit. This register is cleared on power-up.

Figure 7–5: Timer Next Interval Registers (TNIR0 and TNIR1)

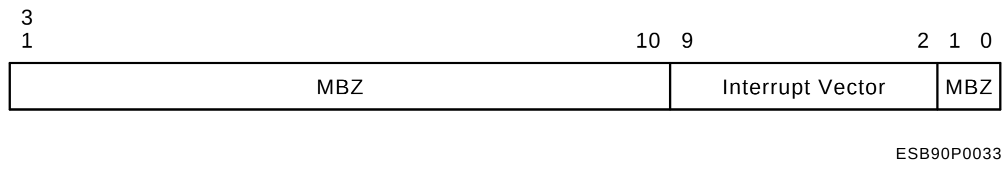


7.3.4 Timer Interrupt Vector Registers (TIVR0 and TIVR1)

The KA660 has two Timer Interrupt Vector Registers, one for timer zero (TIVR0), and one for timer one (TIVR1). TIVR0 is accessible at address 2014 $010C_{16}$, and TIVR1 is accessible at 2014 $011C_{16}$. These registers are implemented in the SSC chip and are set to 78_{16} and $7C_{16}$ respectively by the resident firmware. The format is shown in Figure 7–6.

This Read/Write register contains the timer's interrupt vector. Bits <31:10> and <1:0> are read as zero and must be written as zero. When TCRn<6> (IE) and TCRn<7> (INT) transition to one, an interrupt is posted at IPL 14. When a timer's interrupt is acknowledged, the content of the interrupt vector register is passed to the CPU, and the INT bit is cleared. Interrupt requests can also be cleared by clearing either the IE or INT bit. This register is cleared on power-up.

Figure 7–6: Timer Interrupt Vector Registers (TIVR0 and TIVR1)



NOTE

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

Chapter 8

KA660 Boot and Diagnostic Facility

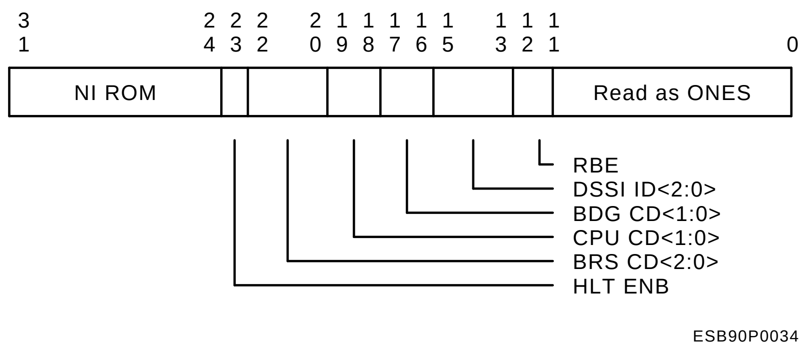
The KA660 Boot and Diagnostic Facility features two registers, 256K bytes of Erasable Programmable Read Only Memory (EPROM) and 1KB of battery backed up RAM. The EPROM and battery backed up RAM may be accessed with longword, word or byte references.

The 256K bytes of EPROM contain the resident firmware. If this EPROM is reprogrammed for special applications, the new code must initialize and configure the board, provide HALT and console emulation, as well as provide boot diagnostic functionality.

8.1 Boot and Diagnostic Register (BDR)

The Boot and Diagnostic Register (BDR) is a byte-wide register repeated in the VAX I/O page at physical addresses 2008 4004 - 2008 407C$_{16}$. It is implemented uniquely on the KA660. It can be accessed by KA660 software, but not by external Q22-Bus devices. The BDR allows the Boot and Diagnostic EPROM programs to read various KA660 configuration bits.

Figure 8–1: Boot and Diagnostic Register



Data Bit	Name	Description	Type
<31:24>	NI ROM	NI Station Address. This byte delivers the NI Station address in the next 32 longwords.	Read only
<23>	HLT ENB	Halt Enable. This bit reflects the state of pin 35 (ENBHALT L) of the 40-pin connector. The assertion of this signal enables the halting of the CPU upon detection of a console BREAK condition. On a power-up, the KA660 resident firmware reads the HLT ENB bit to decide whether to enter the console emulation program (HLT ENB set) or to boot the operating system (HLT ENB clear). On the execution of of a HALT instruction while in kernal mode, the KA660 resident firmware reads the HLT ENB bit to decide whether to enter the console emulation program (HLT ENB set) or to restart the operating system (HLT ENB clear).	Read Only.Writes have no effect.

Data Bit	Name	Description	Type
<22:20>	BRS CD	Baud Rate Select. Writes have no effect. These three bits originate from pins <20:28> (BRS<2:0>) of the 40-pin connector. They reflect the setting of the the baud rate select switch on the CPU cover panel.	Read only

BDR<6:4>	Baud Rate
000	300
001	600
010	1200
011	2400
100	4800
101	9600
110	19200
111	38400

Data Bit	Name	Description	Type
<19:18>	CPU CD	CPU Code. Writes have no effect. These two bits always read as ZERO's because the KA660 cannot be configured as an auxiliary CPU.	Read Only

CPU CD <1:0>	Configuration
00	KA660-AA Arbiter

Data Bit	Name	Description	Type
<17:16>	BDG CD	Boot and Diagnostic Code. This 2-bit code reflects the status of configuration and display connector pins <37:36> (BDG CD<1:0>). The KA660 EPROM programs use BDG CD <1:0> to determine the power up mode as follows:	Read only, writes have no effect.

BDR<1:0>	Power-up Mode
00	Run
01	Language Inquiry
10	Test
11	Manufacturing

Data Bit	Name	Description	Type
<15:13>	BDG DSSIID	DSSI ID. Writes have no effect. This 3-bit code reflects the setting of the DSSI ID jumpers. This code must be decoded and written to the SHAC PPR register.	read only
<12>	RBE	Remote Boot Enable.	
<11:0>	Unused	Read as one's	

8.2 Diagnostic LED Register (DLEDR)

The Diagnostic LED Register (DLEDR), address 2014 0020_{16}, is implemented in the SSC chip and contains four Read/Write bits that control the external LED display. A zero in a bit lights the corresponding LED; all four bits are cleared on power-up and the negation of DCOK when SCR<7> is clear to provide a power-up lamp test. Figure 8–2 shows the register format. Table 8–3 lists the bit descriptions.

Figure 8–2: Diagnostic LED Register (DLEDR)

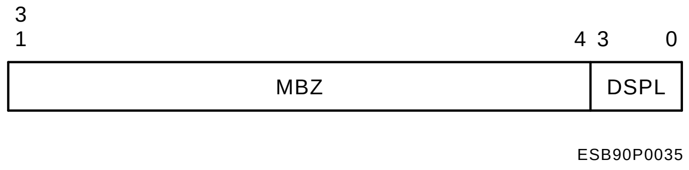


Table 8–3: Diagnostic LED Register Bit Descriptions

Data Bit	Name	Description
<31:4>	MBZ	Read as zeros, must be written as zeros.
<3:0>	DSPL	Display. Read/Write. These four bits update an external LED display. Writing a ZERO to a bit lights the corresponding LED. Writing a ONE to a bit turns its LED off. The display bits are cleared (all LED's are lit) on power-up and the negation of DCOK.

8.3 EPROM Memory

The KA660 has 256KB of EPROM memory for storing code for board initialization, VAX standard console emulation, board self tests, and boot code. The EPROM memory may be accessed with byte, word and longword references. EPROM read accesses take 250 ns. The EPROM is organized as a 128K x 8-bit array. CDAL Bus parity is neither checked nor generated on EPROM references. *It should be noted that the EPROM size must be set in the SSC Configuration Register before attempting to reference outside the first 8KB block of the Local EPROM Space. (2004 0000 - 2004 1FFF$_{16}$)*

8.3.1 EPROM Address Space

The entire 256KB Boot and Diagnostic EPROM can only be read in the 256KB Halt Protect EPROM Space (2004 0000 - 2007 FFFF$_{16}$).

Any I-Stream Read from the EPROM space places the KA660 in Halt Mode. The Q22-bus SRUN signal is deasserted causing the front panel RUN light to extinguish and the CPU is protected from further halts.

Writes and D-Stream Reads to any address space have no effect on Run Mode/Halt Mode status. *The KA660 logic that controls Halt Mode/Run Mode cannot detect I-stream read references that "hit" the Cache; therefore Halt Mode/Run Mode is unaffected by these cache hits.*

8.3.2 KA660 Resident Firmware Operation

The KA660 CPU Module's 256K bytes of EPROM contain the resident firmware, which can be entered by transferring program control to location 2004 0000$_{16}$.

Chapter 12 lists the various halt conditions which cause the KA660 to transfer program control to location 2004 0000$_{16}$.

When running, the resident firmware provides the services expected of a VAX–11 console system. In particular, the following services are available:

- Automatic restart or bootstrap following processor halts or initial power-up.
- An interactive command language allowing the user to examine and alter the state of the processor.
- Diagnostic tests executed on power-up that check out the CPU, the memory system, and the Q22-bus Map.
- Support of video or hardcopy terminals as the console terminal.

Refer to the KA660 Console Program Specification for a complete description of these features.

8.3.2.1 Power-Up Modes

The Boot and Diagnostic EPROM programs use Boot and Diagnostic Code <1:0> (Refer to Section 12.8 to determine the power-up modes shown in Table 8–4.

Table 8–4: Power-Up Modes

Code	Power-up Mode
00	Run (factory setting). If the console terminal supports the Multi-national Character Set (MCS), the user will be prompted for a language if the time-of-year clock battery backup has failed, or if the SSC RAM is corrupted or uninitialized (1st power-up). Full startup diagnostics are run.
01	Language Inquiry. If the console terminal supports MCS, the user will be prompted for language on every power up and restart. Full startup diagnostics are run.
10	Test. UVROM programs run wrap-around serial line unit (SLU) tests.
11	Manufacturing. To provide for rapid startup during certain manufacturing test procedures, the EPROM programs omit the power up memory diagnostics and set up the memory bit map on the assumption that all available memory is functional.

8.4 Battery Backed-up RAM

The KA660 contains 1KB of battery backed-up static RAM found in the SSC,for use as a console "scratchpad". The power for the RAM is provided on pins 38 (VBAT H) and 20, 18, 16-13, 10-8, 6-3 (GND) of the 40-pin connector. This RAM supports byte, word and longword references. Read operations take 700ns to complete while write operations require 600ns. The RAM is organized as a 256 X 32-bit (one longword) array. The array appears in a 1KB block of the VAX I/O page at addresses 2014 0400- 2014 07FF$_{16}$. This array is not protected by parity, and CDAL Bus parity is neither checked nor generated on reads or writes to this RAM.

8.5 KA660 Initialization

The VAX Architecture defines three kinds of hardware initialization:

- Power-up initialization
- I/O Bus initialization
- Processor initialization

8.5.1 Power-Up Initialization

Power-up initialization is the result of the restoration of power and includes a hardware reset, a processor initialization an I/O bus initialization, as well as the initialization of several registers defined in the *VAX Architecture Reference Manual.*

Hardware Reset

A KA660 hardware reset occurs on power-up or the negation of DCOK. A hardware reset causes the hardware halt procedure (see Section 3.9.6) to be initiated with a halt code of 03. It also initializes some IPR's and most I/O Page registers to a known state. Those IPR's affected by a hardware reset are noted in Section 3.4. The effect a hardware reset has on I/O Space registers is documented in the description of the register.

8.5.2 I/O Bus Initialization

An I/O Bus Initialization occurs on power-up, the negation of DCOK, or as the result of a MTPR to IPR 55 (IORESET) If the KA660 is an arbiter, an I/O Bus initialization clears the IPCR and DSER, and causes the Q22-bus interface to acquire both the CDAL and Q22-buses, then assert the Q22-bus BINIT signal.

I/O Bus Reset Register (IPR 55)

The I/O Bus Reset Register (IORESET), IPR 55_{10} is implemented in the SSC chip. If the KA660 is configured as an arbiter, an MTPR of any value to IORESET causes an I/O bus initialization.

8.5.3 Processor Initialization

A Processor Initialization occurs on power-up, the negation of DCOK, and after a halt caused by an error condition.

In addition to initializing those registers defined in the *VAX Architecture Reference Manual,* the KA660 firmware must also configure Main Memory, the Local I/O Page, and the Q22-bus Map during a processor initialization.

8.5.3.1 Configuring the Local I/O Page

The following registers control the configuration of the local I/O Page. They are unique to CPU designs that use the SSC system support chip and they must be configured by the firmware during a processor initialization:

- SSC Base Address Register
- BDR Address Decode Match Register
- BDR Address Decode Mask Register
- SSC Configuration Register
- CDAL Bus Timeout Register

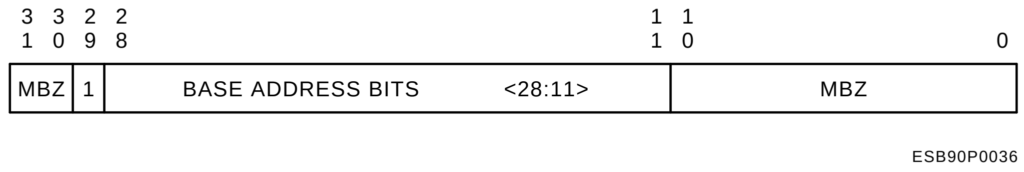
8.5.4 SSC Base Address Register (SSCBR)

The SSC Base Address Register, address 2014 0000_{16}, controls the base addresses of a 2KB block of the Local I/O Space which includes the the following:

- Battery backed-up RAM,
- The registers for the programmable timers,
- The BDR Address Decode Match and Mask Registers,
- The Diagnostic LED Register,
- The CDAL Bus Timeout Register
- A set of diagnostic registers that allow several EPR's to be accessed via I/O page addresses.

This read/write register is set to 2014 0000_{16} on power-up and the negation of DCOK when SCR <7> is clear. Bits SSCBR<31:30,10:0> are unused. They read as ZEROs, and must be written as ZEROs. SSCBR<29> is read as ONE and must be written as ONE. This register should also be set to 2014 0000_{16} by firmware during processor initialization. The SSCBR has the format shown in Figure 8–3.

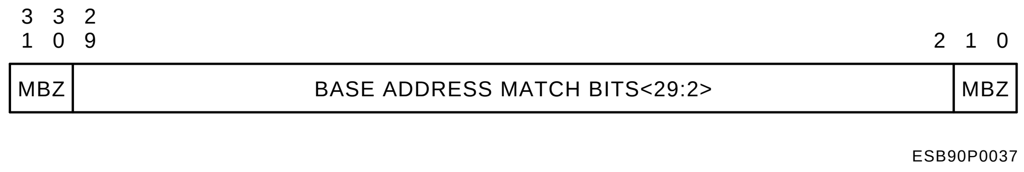
Figure 8–3: SSC Base Address Register (SSCBR)



8.5.5 BDR Address Decode Match Register (BDMTR)

The Local I/O Device Address Decode Match Register, address 2014 0130_{16}, controls the base address of the Boot and Diagnostic Register. This read/write register is cleared on power-up and the negation of DCOK. BDMTR<31:30,1:0> are unused. They read as ZEROs, and must be written as ZEROs. This register should be set to 2008 4000_{16} by firmware during processor initialization. The BDMTR has the format shown in Figure 8–4.

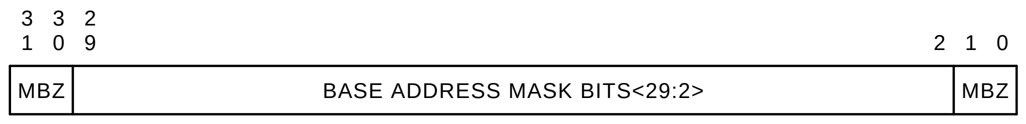
Figure 8–4: BDR Address Decode Match Register (BDMTR)



8.5.6 BDR Address Decode Mask Register (BDMKR)

The BDR Address Decode Mask Register, address 2014 0134_{16}, controls the range of addresses to which the BDR responds to. (An example is the number of copies of the BDR that appear in the physical address space). This read/write register is cleared on power-up and the negation of DCOK. Bits BDMKR<31:30,1:0> are unused. They read as ZEROs, and must be written as ZEROs. This register should should be set to 0000 007F $_{16}$ (32 copies of the BDR) by firmware during processor initialization . The BDMKR has the format shown in Figure 8–5.

Figure 8–5: BDR Address Decode Mask Register (BDMKR)



ESB90P0038

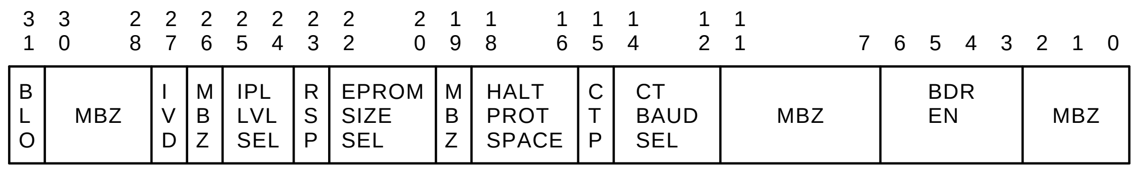
NOTE

The KA660 uses only one of the SSC's address strobes. The other strobe's control registers located at 2014 0130_{16} and 2014 0134_{16} are reserved and should not be accessed as they could cause unpredictable behavior.

8.5.7 SSC Configuration Register (SSCCR)

The SSC Configuration Register, address 2014 0010_{16}, controls the set-up parameters for the console serial line, programmable timers, EPROM, TOY Clock and BDR. The format is shown in Figure 8–6. Table 8–5 contains a list of the bit descriptions.

Figure 8–6: SSC Configuration Register (SSCCR)



ESB90P0039

Table 8–5: SSC Configuration Register Bit Descriptions

Data Bit	Name	Description
<31>	BLO	Battery Low. Read/Write. If the battery voltage goes below threshold while the module is powered down, this bit is set on power-up, after the assertion of DCOK after the assertion of POK. Once set, this bit can only be cleared by software writing it as ONE. If this bit is set, then the TOY Clock will be cleared by power-up and and by the negation of DCOK.
<30:28>	MBZ	Read as ZEROs, must be written as ZEROs.
<27>	IVD	Interrupt Vector Disable. Read/Write. When set, the console serial line and programmable timers will not respond to interrupt acknowledge cycles. Cleared on power-up and, by the negation of DCOK, and by a processor initialization.
<26>	MBZ	Read as ZEROs, must be written as ZEROs.
<25:24>	IPL_LVL_ SEL	IPL Level Select Read/Write. These bits are used to specify the IPL level of interrupt acknowledge cycle that the console serial line and programmable timers respond to. These bits must be cleared (programmed to 00 (binary)) for the console serial line and programmable timers to respond to interrupt acknowledge cycles that they generated (IPL 14). These bits are cleared on power-up, by the negation of DCOK and by a processor initialization.
<23>	RSP	ROM Speed. Read/Write. This bit is used to select the EPROM access time. This bit must be set for the KA660 EPROMs to run at maximum speed. This bit is cleared on power-up and by the negation of DCOK. It must be set to ONE by a processor initialization.
<22:20>	ROM_ SIZE_SEL	EPROM Address Space Size Select. Read/Write. These bits control the size of the range of addresses to which the EPROM responds. These bits must be set to 101 (binary) because the KA660 contains 256KB of EPROM, yielding an address range of 256KB (2004 0000 - 2007 $FFFF_{16}$). These bits are cleared on power-up and by the negation of DCOK, yielding an address range of 8KB (2004 0000 - 2004 $1FFF_{16}$). These bits must be set to the proper value by a processor initialization.
<18:16>	HALT PROT SPACE	EPROM Halt Protect Address Space Size Select. Read/Write. These bits control the size of the Halt Mode address range. These bits must be set to 101 (binary) because the KA660 contains 256KB of EPROM, yielding a Halt Mode address range of 256KB (2004 0000 - 2007 $FFFF_{16}$). These bits are cleared on power-up and by the negation of DCOK yielding a Halt Mode address range of 8KB (2004 0000 - 2004 $1FFF_{16}$). These bits must be set to the proper value by a processor initialization. Note, any instruction fetch from the EPROM put the KA660 in HALT Protect Mode.
<15>	CTP	Control P Enable. Read/Write. When this bit is set, a CTRL/P typed at the console causes the CPU to be halted, if halts are enabled (BDR<7> set). When this bit is cleared, a BREAK typed at the console causes the CPU to be halted, if halts are enabled (BDR<7> set). This bit is cleared on power-up and by the negation of DCOK.

Table 8–5 (Cont.): SSC Configuration Register Bit Descriptions

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

SSCCR<14:12>	Baud Rate
000	300
001	600
010	1200
011	2400
100	4800
101	9600
110	19200
111	38400

Data Bit	Name	Description
<11:7>	MBZ	Read as ZERO, must be written as ZERO.
<6:4>	BDR EN	Read/Write. These bits are used to enable the Boot and Diagnostic Register. They are cleared on power-up and by the negation of DCOK. These bits must be set to 111 (binary) by a processor initialization to enable the BDR.
<3:0>	MBZ	Read as ZERO, must be written as ZERO.

NOTE

The SSC baud clock runs about 1.7% fast which is within the SRM mandated accuracy. This is due to the accuracy of the crystal oscillator.

8.6 CDAL Bus Timeout Control Register (CBTCR)

The CDAL Bus Timeout Register, address 2014 0020 $_{16}$, controls the amount of time allowed to elapse before a CDAL Bus cycle is aborted. The effect of this timer is blocked by the KA660 logic on all Q22-Bus references, because the Q22-Bus interface has its own timers for Q22-Bus references. This timer prevents unanswered CDAL Bus cycles (other than those that go to the Q22-Bus interface) from hanging the system any longer than the timeout interval. **Even though the effect of the timer is blocked on all Q22-bus references, bits<31:30> will still be set on Q22-bus references that take longer than the programmmed value (4us), so these bits are not useful as error indicators.** Figure 8–7 shows the format. Table 8–6 lists the bit descriptions.

Figure 8–7: CP Bus Timeout Control Register (CBTCR)

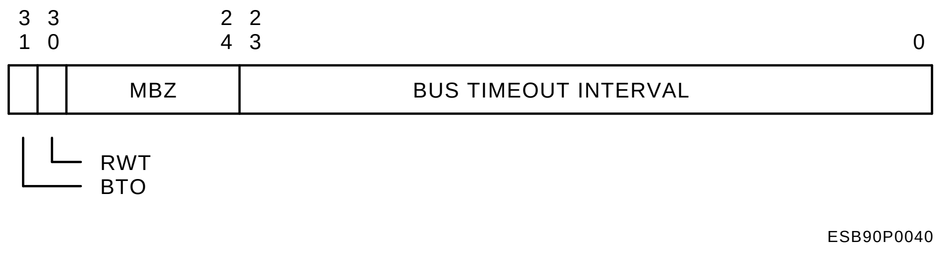


Table 8–6: CP Bus Timeout Control Register Bit Descriptions

Data Bit	Name	Description
<31>	BTO	CP Bus Timeout. Read/Write to Clear. This bit is set when the BUS TIMEOUT INTERVAL set in bits <23:0> has expired during any CP Bus cycle. This bit is cleared by writing a ONE, on power-up and the negation of DCOK.
<30>	RWT	CP Bus Read/Write Timeout. Read/Write to Clear. This bit is set when the BUS TIMEOUT INTERVAL set in bits <23:0> has expired during a CPU or DMA read or write cycle on the CP Bus. This bit is cleared by writing a one, on power-up and the negation of DCOK.
<29:22>	MBZ	Read as ZEROs, must be written as ZEROs.
<23:0>	BUS TIMEOUT INTERVAL	Read/Write. These bits are used to program the desired timeout period. The available range of 1 to $FFFFFF_{16}$ corresponds to a selectable timeout range of 1µs to 16.77 seconds in 1µs increments. Writing a zero to this field disables the bus timeout function. The BTO bit is used to signify that a bus timeout has occurred. This field is cleared on power-up and the negation of DCOK. This register should be loaded with $0000\ 4000_{16}$ on a processor initialization for a timeout value of 15 milliseconds.

Chapter 9

KA660 Q22-bus Interface

The KA660 includes a Q22-bus interface implemented via a single VLSI chip called the CQBIC. It contains a CDAL Bus to Q22-bus interface, that supports the following:

- A programmable mapping function (scatter-gather map) for translating 22-bit, Q22-bus addresses into 29-bit CDAL addresses that allows any page in the Q22-bus memory space to be mapped to any page in main memory.
- A direct mapping function for translating 29-bit CDAL addresses in the local Q22-bus address space and local Q22-bus I/O page into 22-bit, Q22-bus addresses.
- Masked and unmasked longword reads and writes from the CPU to the Q22-bus Memory and I/O Space and the Q22-bus interface registers. Longword reads and writes of the local Q22-bus memory space are buffered and translated into two-word, block mode, transfers on the Q22-bus. Longword reads and writes of the local Q22-bus I/O space are buffered and translated into two, single-word transfers on the Q22-bus.
- Up to sixteen-word, block mode, writes from the Q22-bus to main memory. These words are buffered then transferred to main memory using two asynchronous DMA octaword transfers. For block mode writes of less than sixteen words, the words are buffered and transferred to main memory using the most efficient combination of octaword, quadword, and longword asynchronous DMA transfers. The maximum write bandwidth for block mode references is 3.3 MB/sec. Block mode reads of main memory from the Q22-bus cause the Q22-bus interface to perform an asynchronous DMA quadword read of main memory and buffer all four words, so that on block mode reads, the next three words of the block mode read can be delivered without any additional CDAL cycles. The maximum read bandwidth for Q22-bus block mode references is 2.4 MB/sec. Q22-bus Burst Mode DMA transfers result in single-word reads and writes of Main Memory.
- Transfers from the CPU to the local Q22-bus memory space, that result in the Q22-bus Map translating the address back into main memory (Local-Miss, Global-Hit transactions).

The Q22-bus interface contains several registers for Q22-bus control and configuration, inter-processor communication, and error reporting.

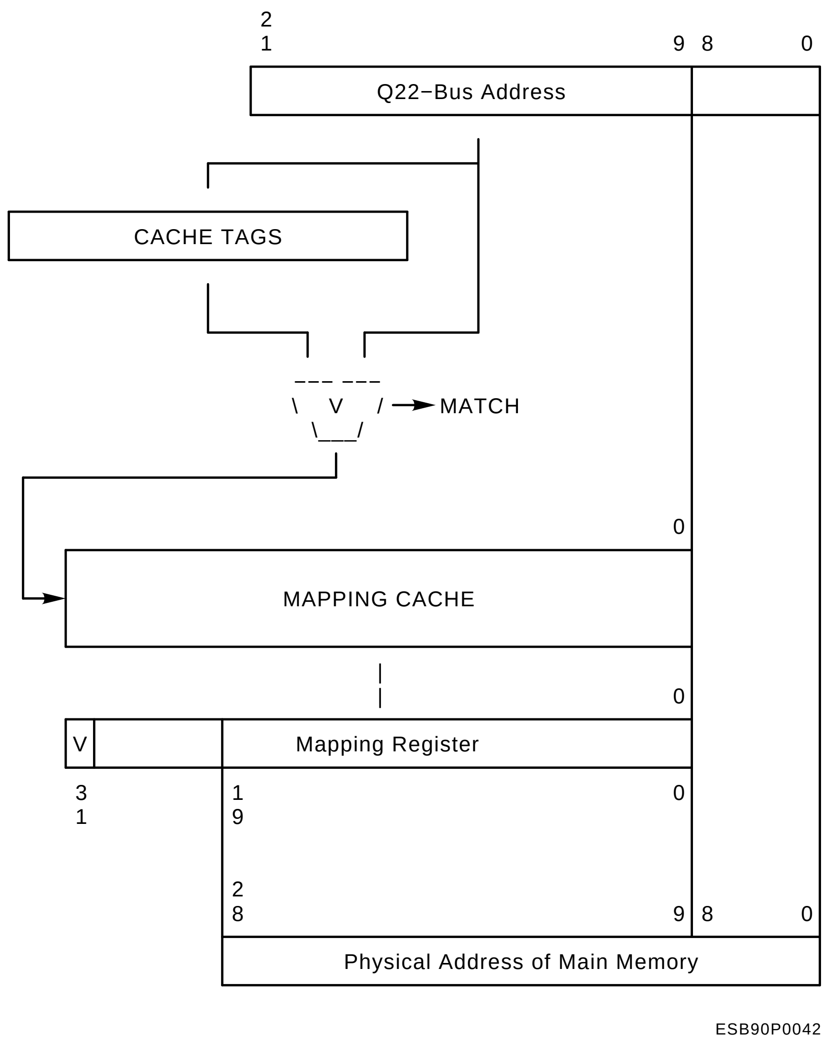
The interface also contains Q22-bus interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7-BR4 and translates them into CPU interrupts at levels 17-14.

The Q22-bus interface detects Q22-bus "NO SACK" timeouts, Q22-bus Interrupt Acknowledge timeouts, Q22-bus non-existent memory timeouts, main memory errors on DMA accesses from the Q22-bus and Q22-bus device parity errors.

9.1 Q22-bus to Main Memory Address Translation

On DMA references to main memory, the 22-bit, Q22-bus address must be translated into a 29-bit main memory address (Figure 9–1.) This translation process is performed by the Q22-bus interface by using the Q22-bus Map. This map contains 8192 mapping registers, (one for each page in the Q22-bus Memory Space), each of which can map a page (512 bytes) of the Q22-bus Memory address space into any of the 1024K pages in main memory. Since Local I/O Space addresses cannot be mapped to Q22-bus pages, the Local I/O Page is unaccessible to devices on the Q22-bus. Figure 9–1 shows how Q22-bus addresses are translated into main memory addresses.

Figure 9–1: Q22-bus Address Translation



At power up time, the Q22-bus Map Registers, including the valid bits, are undefined. External access to main memory is disabled so long as the Interprocessor Communication Register LM EAE bit is cleared. The Q22-bus Interface monitors each Q22-bus cycle and responds if the following three conditions are met:

1. The Interprocessor Communication Register LM EAE bit is set.
2. The Valid bit of the selected mapping register is set.

3. During read operations, the mapping register must map into existent main memory, or a Q22-bus timeout occurs. (During write operations, the Q22-bus Interface returns Q22-bus BRPLY before checking for existent local memory; the response depends only on conditions 1 and 2 above).

NOTE

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

If the map cache does not contain the needed Q22-bus Map Register, then the Q22-bus interface will perform an asynchronous DMA read of the Q22-bus Map register before proceeding with the Q22-bus Bus DMA transfer.

9.1.1 Q22-bus Map Registers (QMR)'s

The Q22-bus Map contains 8192 registers that control the mapping of Q22-bus addresses into main memory. Each register maps a page of the Q22-bus Memory Space into a page of main memory. These registers are implemented in a 32KB block of main memory, but are accessed through the CQBIC chip via a block of addresses in the I/O Page.

The Local I/O Space address of each register was chosen so that register address bits <14:2> are identical to Q22-bus address bits <21:9> of the Q22-bus page which the register maps. Table 9–1 lists the register addresses.

Table 9–1: Q22-bus Map Register Addresses

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

The Q22-bus Map Registers (QMR's) have the format shown in Figure 9–2.

Figure 9–2: Q22-bus Map Register Format

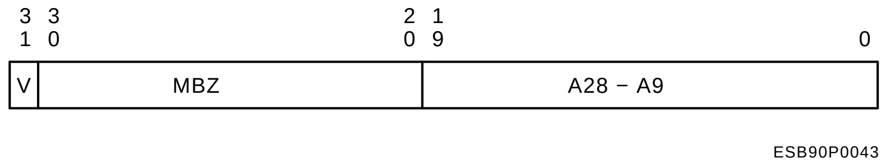


Table 9–2 describes the bits in the Q22-bus Map Register.

Table 9–2: Q22-bus Map Register Bit Description

Data Bit	Name	Description
<31>	V	Valid. Read/Write. When a Q22-bus Map Register is selected by bits <21:9> of the Q22-bus address, the Valid bit determines whether mapping is enabled for that Q22-bus page. If the Valid bit is set, the mapping is enabled, and Q22-bus addresses within the page controlled by the register are mapped into the main memory page determined by bits <28:9>. If the Valid bit is clear, the mapping register is disabled, and the Q22-bus interface does not respond to addresses within that page. This bit is UNDEFINED on power-up and the negation of DCOK.
<30:20>	Unused	These bits always read as zero and must be written as zero.
<19:0>	A28-A9	Address Bits <28:9> Read/Write. When a Q22-bus Map Register is selected by a Q22-bus address, and if that register's Valid bit is set, then these 20 bits are used as main memory address bits. Q22-bus address bits <8:0> are used as main memory address bits <8:0>. These bits are UNDEFINED on power-up and the negation of DCOK.

9.1.2 Accessing the Q22-bus Map Registers

Although the CPU accesses the Q22-bus Map Registers with aligned, longword references to the Local I/O Page (addresses 2008 8000 - 2008 FFFC $_{16}$), the map actually resides in a 32KB block of main memory. The starting address of this block is controlled by the contents of the Q22-bus Map Base Register. The Q22-bus interface also contains a 16-entry, fully associative, Q22-bus Map Cache to reduce the number of main memory accesses required for address translation.

NOTE

The system software must protect the pages of memory that contain the Q22-bus Map from direct accesses that will corrupt the map or cause the entries in the Q22-bus Map Cache to become stale. Either of these conditions will result in the incorrect operation of the mapping function.

When the CPU accesses the Q22-bus Map through the Local I/O Page addresses, the Q22-bus interface reads or writes the map in main memory. The Q22-bus Bus interface does not have to gain Q22-bus mastership when accessing the Q22-bus Map. Because these addresses are in the local I/O space, they are not accessible from the Q22-bus.

On a Q22-bus Map read by the CPU, the Q22-bus interface decodes the Local I/O Space address (2008 8000 - 2008 FFFC$_{16}$. If the register is in the Q22-bus Map Cache, the Q22-bus interface will internally resolve any conflicts between CPU and Q22-bus transactions (if both are attempting to access the Q22-bus Map Cache entries at the same time), then return the data. If the map register is not in the map cache, the Q22-bus interface will force the CPU to retry, acquire the CDAL bus and perform an asynchronous DMA read of the map register. On completion of the read, the CPU is provided with the data when its read operation is retried. A map read by the CPU does not cause the register that was read to be stored in the map cache.

On a Q22-bus Map write by the CPU, the Q22-bus interface latches the data, then on the completion of the CPU write, acquires the CDAL bus and performs an asynchronous DMA write to the map register. If the map register is in the Q22-bus Map Cache, then the CAMValid bit for that entry will be cleared to prevent the entry from becoming stale. A Q22-bus Map write by the CPU does not update any cached copies of the Q22-bus Map Register.

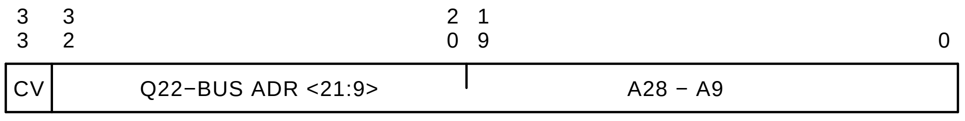
9.1.3 The Q22-bus Map Cache

To speed up the process of translating Q22-bus address to Main Memory addresses, the Q22-bus interface utilizes a fully associative, sixteen entry, Q22-bus Map Cache which is implemented in the CQBIC chip.

If a DMA transfer ends on a page boundary, the Q22-bus interface will prefetch the mapping register required to translate the next page and load it into the cache, before starting a new DMA transfer. This allows Q22-bus block mode DMA transfers that cross page boundaries to proceed without delay. The replacement algorithm for updating the Q22-bus Map Cache is FIFO.

The cached copy of the Q22-bus Map Register is used for the address translation process. If the required map entry for a Q22-bus address (as determined by bits <21:9> of the Q22-bus address), is not in the map cache, then the Q22-bus interface uses the contents of the Map Base Register to access main memory and retrieve the required entry. After obtaining the entry from main memory, the valid bit is checked. If it is set, the entry is stored in the cache and the Q22-bus cycle continues. Figure 9–3 shows the format. Table 9–3 contains a description of the Q22-bus Map Cache Entry bits.

Figure 9–3: Q22-bus Map Cache Entry Format



ESB90P0044

Table 9–3: Q22-bus Map Cache Entry Bit Description

Data Bit	Name	Description
<33>	CAMValid	When a mapping register is selected by a Q22-bus address, the CAMValid bit determines whether the cached copy of the mapping register for that address is valid. If the CAMValid bit is set, the mapping register is enabled, and addresses within that page can be mapped. If the CAMValid bit is clear, the Q22-bus interface must read the map in local memory to determine if the mapping register is enabled. This bit is cleared on power-up, the negation of DCOK, by setting the QMCIA (Q22-bus Map Cache Invalidate All) bit in the Interprocessor Communication Register, on writes to IPR 55 (IORESET), by a write to the Q22-bus Map Base Register, or by writing to the QMR that is being cached.
<32:20>	QBUS ADR	These bits contain the Q22-bus address bits <21:9> of the page that this entry maps. This is the content addressable field of the 16 entry cache for determining if the map register for a particular Q22-bus address is in the map cache. These bits are undefined on power-up.
<19:0>	Address bits A28-A9	When a mapping register is selected by a Q22-bus address, and if that register's CAMValid bit is set, then these 20 bits are used as main memory address bits 28 through 9. Q22-bus address bits 8 through 0 are used as local memory address bits 8 through 0. These bits are undefined on power-up.

9.2 CDAL to Q22-bus Address Translation

CDAL bus addresses within the CDAL Translation) Local Q22-bus I/O Space, addresses 2000 0000 - 2000 $1FFF_{16}$, are translated into Q22-bus I/O Space addresses by using bits <12:0> of the CDAL bus address as bits <12:0> of the Q22-bus address and asserting BBS7. Q22-bus address bits <21:13> are driven as zeros.

CDAL bus addresses within the Local Q22-bus Memory Space, addresses 3000 0000 - 303F FFFF $_{16}$, are translated into Q22-bus Memory Space addresses by using bits <21:0> of the CDAL bus address as bits <21:0> of the Q22-bus address.

9.3 Interprocessor Communications Facility

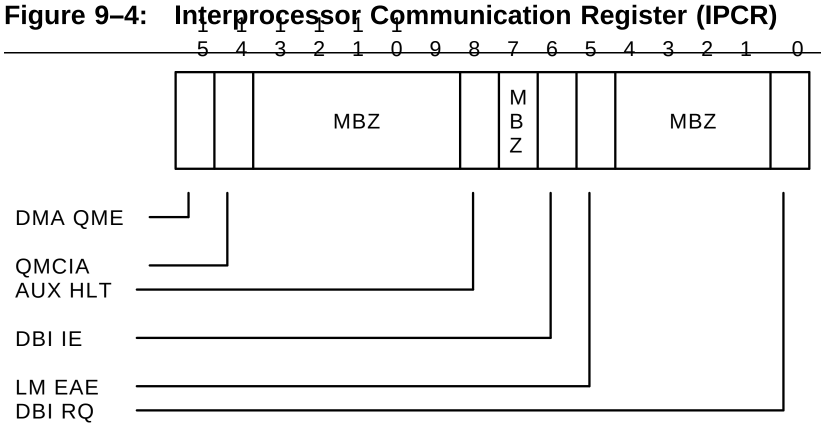
The KA660 can only be configured as a Q22-bus arbiter.

The KA660 Interprocessor Communication Facility allows other processors on the system to request program interrupts from the KA660 without using the Q22-bus interrupt request lines. It also controls external access to local memory (via the Q22-bus map).

9.3.1 Interprocessor Communication Register (IPCR)

The Interprocessor Communication Register is a 16-bit register which resides in the Q22-bus I/O page address space and can be accessed by any device which can become Q22-bus master (including the KA660 itself). The IPCR is implemented in the CQBIC chip and is byte accessible, meaning that a write byte instruction can write to either the low or high byte without affecting the other byte. The I/O page address of the IPCR is constant with the KA660 because it only supports arbiter mode and not auxiliary mode. The hex 32-bit address is 2000 1F40 and the octal 22-bit address is 17 777 500. Figure 9–4 shows the format. Table 9–4 describes the bits.

Figure 9–4: Interprocessor Communication Register (IPCR)



ESB90P0045

Table 9–4: Interprocessor Communication Register Bit Description

Data Bit	Name	Description
<15>	DMA QME	DMA Q22-bus Address Space Memory error. Read/Write to clear. This bit indicates that an error occurred when a Q22-bus device was attempting to read main memory. It is set if DMA System Error Register bit DSER<4> (MAIN MEMORY ERROR) is set, or the CDAL timer expires. The MAIN MEMORY ERROR bit indicates that an uncorrectable error occurred when an external device (or CPU) was accessing the KA660 local memory. The CDAL timer expiring indicates that the memory controller did not respond when the Q22-bus interface initiated a DMA transfer. This bit is cleared by writing a one to it,on power-up, by the negation of DCOK, by writes to IPR 55 (IORESET), and whenever DSER<4> is cleared.
<14>	QMCIA	Q22-bus Map Cache Invalidate All. Write Only. Writing a one to this bit clears the CAMValid bits in the cached copy of the MAP. This bit always reads as zero. Writing a zero has no effect.
<13:09>	Unused	Read as zeros. Must be written as zeros.
<8>	AUX HLT	Auxiliary Halt. Read Only. When this bit is set it has no effect on the operation of the on-board CPU. This bit is cleared on power-up, by the negation of DCOK, by writes to IPR 55 (IORESET). **Note: This bit should never be set because the processor does not support auxiliary mode**.
<7>	Unused	Read as zero. Must be written as zero.
<6>	DBI IE	Doorbell Interrupt Enable. Read/Write when the KA660 is Q22-bus master. Read Only when another device is Q22-bus master. When set, this bit enables interprocessor doorbell interrupt requests via IPCR<0>. Cleared on power-up, by the negation of DCOK, and writes to IPR 55 (IORESET).
<5>	LM EAE	Local Memory External Access Enable. Read/Write when the KA660 is Q22-bus master. Read Only when another device is Q22-bus master. When set, this bit enables external access to local memory (via the Q22-bus map). Cleared on power-up and by the negation of DCOK.
<4:1>	Unused	Read as zeros. Must be written as zeros.
<0>	DBI RQ	Doorbell Interrupt Request.Read/Write. If IPCR<6> (DBI IE) is set, setting this bit generates a doorbell interrupt request. If IPCR<6> is clear, setting this bit has no effect. Clearing this bit has no effect. DBI RQ is cleared when the CPU grants the doorbell interrupt request. DBI RQ is held clear whenever DBI IE is clear. This bit is cleared on power-up and the negation of DCOK.

9.3.2 Interprocessor Doorbell Interrupts

If the Interprocessor Communication Register DBI IE bit is set, any Q22-bus master can request an interprocessor doorbell interrupt by writing a one into IPCR bit <0>.

The interrupt vector is 204_{16} and the Interrupt priority is 14_{16}. This IPL is the same as BR4 on the Q22-bus. The interprocessor doorbell is the third highest priority IPL 14 device, directly after the console serial line unit and the programmable timers.

NOTE

Following an interprocessor doorbell interrupt, the KA660 CPU sets the IPL to 14. The IPL is set to 17 for external Q22-bus BR4 interrupts.

9.4 Q22-bus Interrupt Handling

The KA660 responds to interrupt requests BR7-4 with the standard Q22-bus interrupt acknowledge protocol (DIN followed by IAK). The console serial line unit, the programmable timers, and the interprocessor doorbell request interrupt at IPL 14 and have priority over all Q22-bus BR4 interrupt requests. After responding to any interrupt request BR7-4, the CPU sets the processor priority to IPL 17. All BR7-4 interrupt requests are disabled unless software lowers the interrupt priority level.

Interrupt requests from the KA660 interval timer are handled directly by the CPU. Interval timer interrupt requests have a higher priority than BR6 interrupt requests. After responding to an interval timer interrupt request, the CPU sets the processor priority to IPL 16. Thus, BR7 interrupt requests remain enabled.

9.5 Configuring the Q22-bus Map

The KA660 implements the Q22-bus Map in an 8K longword (32KB) block of main memory. This Map must be configured by the KA660 firmware during a processor initialization by writing the base address of the uppermost 32KB block of good main memory into the Q22-bus Map Base Register. The base of this map must be located on a 32KB boundary.

NOTE

This 32KB block of main memory must be protected by the system software. The only access to the map should be through Local I/O page addresses 2008 8000 - 2008 FFFC $_{16}$.

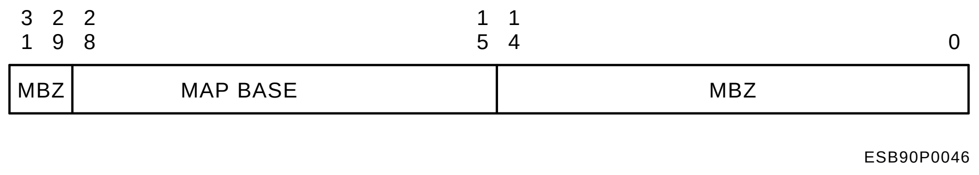
9.5.1 Q22-bus Map Base Address Register (QBMBR)

The Q22-bus Map Base Address Register, address 2008 0010 $_{16}$ controls the main memory location of the 32KB block of Q22-bus Map Registers. This Read/Write register is accessible by the CPU on a longword boundary only. Bits <31:29,14:0> are unused and should be written as zero and will return zero when read. Figure 9–5 shows the format.

A write to the Map Base Register will flush the Q22-bus Map Cache by clearing the CAMValid bits in all the entries.

The contents of this register are undefined on power-up and the negation of DCOK when SCR<7> is clear and is not affected by BINIT being asserted on the Q22-bus.

Figure 9–5: Q22-bus Map Base Address Register (QBMBR)



9.6 System Configuration Register (SCR)

The System Configuration Register, address 2008 0000 $_{16}$, contains the processor number which determines the address of the IPCR register, a BHALT enable bit, a power OK flag and an AUX flag. Figure 9–6 shows the format. Table 9–5 describes the bits in the System Configuration Register.

The System Configuration Register (SCR) is longword, word, and byte accessible. Programmable option fields are cleared on power-up and by the negation of DCOK when SCR<7> is clear.

Figure 9–6: System Configuration Register (SCR)

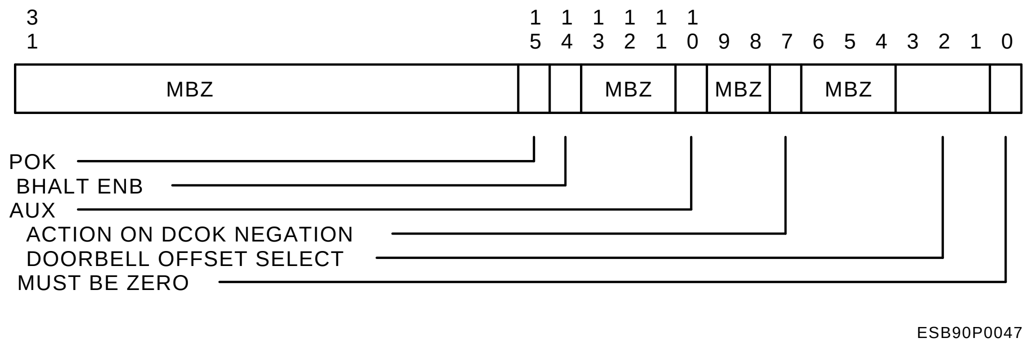


Table 9–5: System Configuration Register Bit Description

Data Bit	Name	Description
<31:16>	Unused	Read as zero. Must be written as zero.
<15>	POK	Power OK. Read Only. Writes have no effect. This bit is set if the Q22-bus BPOK signal is asserted and clear if it is negated. This bit is cleared on power-up and by the negation of DCOK.
<14>	BHALT EN	BHALT Enable. Read/Write. This bit controls the effect the Q22-bus BHALT signal has on the CPU. When set, asserting the Q22-bus BHALT signal will halt the CPU and assert DSER<15>. When cleared, The Q22-bus BHALT signal will have no effect. This bit is cleared on power-up and by the negation of DCOK.
<13:11>	Unused	Read as zero. Must be written as zero.
<10>	AUX	Auxiliary. Read Only. Writes have no effect. This bit defines Auxiliary and Arbiter mode of operation of the KA660 When read as a zero, Arbiter mode is selected, and when read as a one, Auxiliary mode is selected. Because the KA660 can only be configured as an arbiter this bit should always read as zero.
<9:8>	Unused	Read as zero. Must be written as zero.
<7>	ACTION ON DCOK NEGATION	Read/Write. When cleared, the Q22-bus interface asserts SYSRESET (causing a hardware reset of the board and control to be passed to the resident firmware via the hardware halt procedure with a halt code of 3) when DCOK is negated on the Q22-bus. When set, the Q22-bus interface asserts HALCYON (causing control to be passed to the resident firmware via the hardware halt procedure with a halt code of 2) when DCOK is negated on the Q22-bus. Cleared on power-up and the negation of DCOK.
<6:4>	Unused	Read as zero. Must be written as zero.

Table 9–5 (Cont.): System Configuration Register Bit Description

Data Bit	Name	Description
<3:1>	RESERVED	Reserved for use by Digital.
<0>	Unused	Read as zero. Must be written as zero.

9.7 Error Reporting Registers

There are three registers associated with Q22-bus interface error reporting:

- The DMA System Error Register (DSER)
- The Q22-bus Error Address Register (QBEAR)
- The DMA Error Address Register (DEAR)

These registers are located in the local VAX I/O address space and can only be accessed by the local processor. The **DSER** is implemented in the CQBIC chip and it logs main memory errors on DMA transfers, Q22-bus parity errors, Q22-bus non-existent memory errors, and Q22-bus no-Grant The **QBEAR** contains the address of the page in Q22-bus space which caused a parity error during an access by the local processor. The **DEAR** contains the address of the page in local memory which caused a memory error during an access by an external device or the processor during a local-miss global-hit transaction. An access by the local processor which the Q22-bus interface maps into main memory will provide error status to the processor when the processor does a RETRY for a READ local miss-global hit, or by an interrupt in the case of a local-miss global-hit write.

9.7.1 DMA System Error Register (DSER)

The DSER (address 2008 0004_{16})is a longword, word, or byte accessible Read/Write register available to the local processor. The bits in this register are cleared to zero on power-up, by the negation of DCOK when SCR <7> is clear, and by writes to IPR 55 (IORESET). All bits are set to one to record the occurrence of an event. They are cleared by writing a one, writing zeros has no effect.
The format of the DMA System Error Register is shown in Figure 9–7. Table 9–6 describes the bits in the System Error Register.

Figure 9–7: DMA System Error Register (DSER)

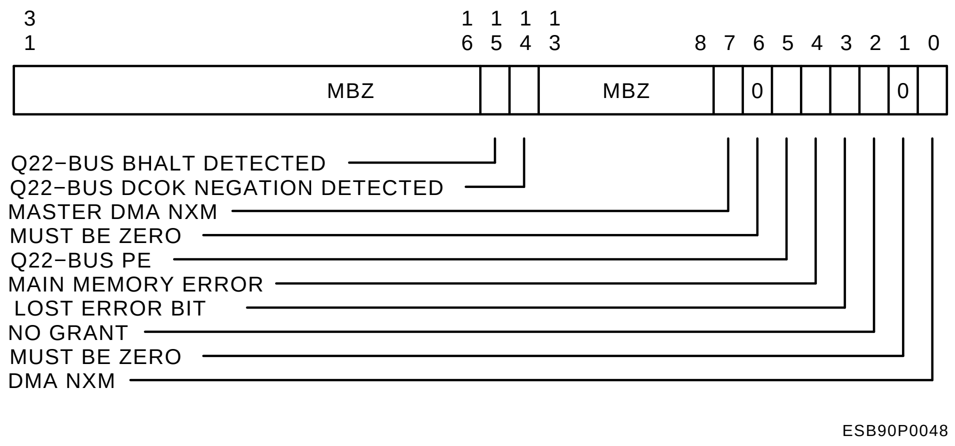


Table 9–6: DMA System Error Register Bit Description

Data Bit	Name	Description
<31:16>	Unused	Read as zero. Must be written as zero.
<15>	Q22-BUS BHALT DETECTED	Read/Write to clear. This bit is set when the Q22-bus interface detects that the Q22-bus BHALT line was asserted and SCR<14> (BHALT ENABLE) is set. Cleared by writing a one, writes to IPR 55 (IORESET), on power-up and the negation of DCOK.
<14>	Q22-BUS DCOK NEGATION DETECTED	Read/Write to clear. This bit is set when the Q22-bus interface detects the negation of DCOK on the Q22-bus and SCR<7> (ACTION ON DCOK NEGATION) is set. Cleared by writing a one, writes to IPR 55 (IORESET), on power-up and the negation of DCOK.
<13:8>	Unused	Read as zero. Must be written as zero.
<7>	MASTER DMA NXM	Read/Write to clear. This bit is set when the CPU performs a demand Q22-bus read cycle or write cycle that does not reply after 10us. During interrupt acknowledge cycles, or request read cycles, this bit is not set. Cleared by writing a one, on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).
<6>	Unused	Read as zero. Must be written as zero.
<5>	Q22-bus PARITY ERROR	Read/Write to clear. This bit is set when the CPU performs a Q22-bus Demand read cycle which returns a parity error. During interrupt acknowledge cycles, or request read cycles this bit is not set. Cleared by writing a one, on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).
<4>	MAIN MEMORY ERROR	Read/Write to clear. This bit is set if an external Q22-bus device or local miss global hit receives a memory error while reading local memory. The IPCR<15> reports the memory error to the external Q22-bus device. Cleared by writing a one, on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).

Table 9–6 (Cont.): DMA System Error Register Bit Description

Data Bit	Name	Description
<3>	LOST ERROR	Read/Write to clear. This bit indicates that an error address has been lost because of DSER<7,5,4,0> having been previously set and a subsequent error of either type occurs which would have normally captured an address and set either DSER<7,5,4,0> flag. Cleared by writing a one, on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).
<2>	NO GRANT TIMEOUT	Read/Write to clear. This bit is set if the Q22-bus does not return a bus grant within 10ms of the bus request from a CPU demand read cycle, or write cycle. During interrupt acknowledge or request read cycles this bit is not set. Cleared by writing a one, on power-up, by the negation of DCOK and by writes to IPR 55 (IORESET).
<1>	Unused	Read as zero. Must be written as zero.
<0>	DMA NXM	Read/Write to clear. This bit is set on a DMA transfer to a non-existent main memory location. This includes local-miss global-hit cycles and map accesses to non-existent memory. Cleared by writing a one, on power-up, by the negation of DCOK when SCR<7> is clear, and by writes to IPR 55 (IORESET).

9.7.2 Q22-bus Error Address Register (QBEAR)

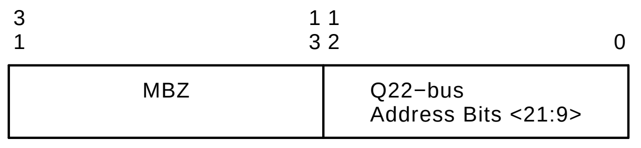
The Q22-bus Error Address Register, address 2008 0008 $_{16}$, is a Read Only, longword accessible, register which is implemented in the CQBIC chip. Its contents are valid only if DSER<5> (Q22-BUS PARITY ERROR) is set, or if DSER<7> (MASTER DMA NXM) is set.

Reading this register when DSER<5> and DSER<7> are clear will return undefined results. Additional Q22-bus parity errors that could have set DSER<5> or Q22-bus timeout errors that could have caused DSER<7> to set, will cause DSER<3> to set.

The QBEAR contains the address of the page in Q22-bus space which caused a parity error during an access by the on-board CPU which set DSER<5> or a master timeout which set DSER<7>.

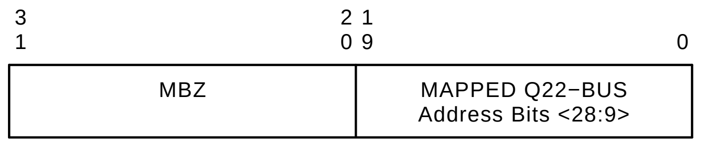
Q22-bus address bits <21:9> are loaded into QBEAR bits <12:0>. QBEAR bits <31:13> always read as zeros.

Figure 9–8: Q22-bus Error Address Register (QBEAR)



ESB90P0049

Figure 9–9: DMA Error Address Register (DBEAR)



ESB90P0050

NOTE

This is a Read Only register, if a write is attempted a hard error (IPL 1D) will be generated.

9.7.3 DMA Error Address Register (DEAR)

The DMA Error Address Register address 2008 000C $_{16}$ is a read only, longword accessible, register which is implemented in the CQBIC chip. It contains valid information only when DSER<4> (MAIN MEMORY ERROR) is set or when DSER<0> (DMA NXM) is set . Reading this register when DSER<4> and DSER <0> are clear will return UNDEFINED data. Figure 9–9 shows the format.

The DBEAR contains the map translated address of the page in local memory which caused a memory error or non existent memory error during an access by an external device or the Q22-bus interface for the CPU during a local-miss global-hit transaction or Q22-bus Map access.

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

Mapped Q22-bus address bits <28:9> are loaded into DBEAR bits <19:0>. DBEAR bits <31:20> always read as zeros.

NOTE

This is a Read Only register, if a write is attempted a hard error (IPL 1D) will be generated.

9.8 Q22-bus Interface Error Handling

The Q22-bus interface does not generate or check CDAL parity.

The Q22-bus interface checks all CPU references to Q22-bus Memory and I/O spaces to insure that nothing but masked and un-masked longword accesses are attempted. Any other type of reference will cause a machine check abort to be initiated.

The Q22-bus interface maintains several timers to prevent incomplete accesses from hanging the system indefinitely. They include: a 10µs non-existent memory timer for accesses to the Q22-bus Memory and I/O Spaces, a 10µs "NO SACK" timer for acknowledgment of Q22-bus DMA grants, and a 10ms "NO GRANT" timer for acquiring the Q22-bus.

If there is a non existent memory (NXM) error (10µs timeout) while accessing the Q22-bus on a demand read reference, associated row in the cache is invalidated , bit DSER<7> is set, the address of the Q22-bus page being accessed is captured in QBEAR<12:0>, and a machine check abort is initiated.

If there is a NXM error on a prefetch read, or an interrupt acknowledge vector read, then the prefetch or interrupt acknowledge reference is aborted but no information is captured and no machine check occurs.

If there is a NXM error on a masked write reference, then DSER<7> is set, the address of the Q22-bus page being accessed is captured in QBEAR<12:0>, and an interrupt is generated at IPL 1D through vector 60_{16}.

If the Q22-bus interface does not receive an acknowledgment within 10µs after it has granted the Q22-bus, the grant is withdrawn, no errors are reported, and the Q22-bus interface waits 500ns to clear the Q22-bus grant daisy chain before beginning arbitration again.

If the Q22-bus interface tries to obtain Q22-bus mastership on a CPU demand read reference, and does not obtain it within 10ms, associated row in the cache is invalidated DSER<2> is set, and a machine check abort is initiated.

The Q22-bus interface also monitors Q22-bus signals BDAL<17:16> while reading information over the Q22-bus so that parity errors detected by the device being read from are recognized.

If a parity error is detected by another Q22-bus device on a CPU demand read reference to Q22-bus Memory or I/O Space, then associated row in the cache is invalidated , DSER<5> is set, the address of the Q22-bus page being accessed is captured in QBEAR<12:0>, and a machine check abort is initiated.

If a parity error is detected by another Q22-bus device on a prefetch request read by the CPU, the prefetch is aborted, associated row in the cache is invalidated , DSER<5> is set, the address of the Q22-bus page being accessed is captured in QBEAR<12:0>, but no machine check is generated.

The Q22-bus interface also monitors the backplane BPOK signal to detect power failures. If BPOK is negated on the Q22-bus, a power fail trap is generated, and the CPU traps through vector $0C_{16}$. The state of the Q22-bus BPOK signal can be read from SCR<15>. The Q22- Bus interface continues to operate after generating the powerfail trap, until DCOK is negated.

Chapter 10

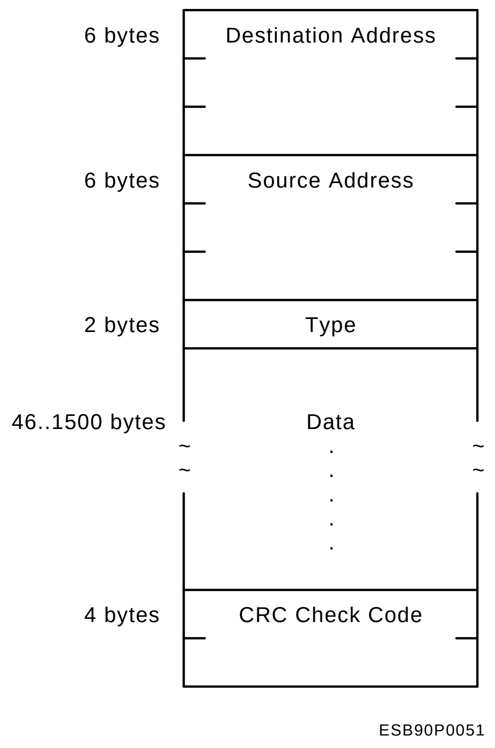
KA660 Network Interface

The KA660 includes a network interface that is implemented via the Second Generation Ethernet Controller chip (SGEC). When used in conjunction with the H3602 cover panel, this interface allows the KA660 to be connected to either a thinwire or standard Ethernet network. It supports the Ethernet Data Link Layer as specified in the *VAX Architecture Reference Manual*. The SGEC also supports CP Bus Parity Protection.

10.1 Ethernet Overview

Ethernet is a serial bus that can support up to 1,024 nodes with a maximum separation of 2.8 kilometers (1.7 miles). Data is passed over the Ethernet in Manchester encoded format at a rate of 10 million bits per second in variable-length packets. Each packet has the format shown in Figure 10–1.

Figure 10–1: 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.

All nodes on the Ethernet have equal priority. The technique used to control access to the bus is Carrier Sense, Multiple Access, with Collision Detection (CSMA/CD). To access the bus, devices must first wait for the bus to clear (no carrier sensed). Once the bus is clear, all devices that want to access the bus have equal priority (multi-access), so they all attempt to transmit. After starting transmission, devices must monitor the bus for collisions (collision detection). If no collision is detected, the device may continue with transmission. If a collision is detected, then the device waits for a random amount of time and repeats the access sequence.

Ethernet allows point to point communication between two devices, as well as simultaneous communication between multiple devices. To support these two modes of communication, there are two types of network addresses, physical and multicast. These two types of addresses are both 48 bits (6 bytes) long and are described below.

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 hardware address of the KA660 module is determined at the time of manufacture and is stored in the Network Interface Station Address ROM. Because every device that is intended to connect to an Ethernet network must have a unique physical address, the bit pattern blasted into the Network Interface Station Address ROM must be unique for each KA660 . The Multicast Addresses to which the KA660 will respond are determined by the Multicast Address Filter Mask in the Network Interface Initialization Block.

10.2 NI Station Address ROM (NISA ROM)

The Network Interface includes a byte-wide, 32-byte, socketted ROM called the Network Interface Station Address ROM. One byte of this ROM appears in the second byte of each of 32 consecutive longwords in the address range 2008 4000 - 2008 $407C_{16}$. Bytes one, three and four of each longword are defined in the Boot Diagnostic Register section 9.1. The second byte of the first six longwords contain the 48-bit Network Physical Address (NPA) of the KA660 . The low-order byte in the remaining 26 longwords are used for testing. This address range is Read Only. Writes to this address range will result in a CP Bus Timeout and a machine check.

10.3 Programming the SGEC

The operation of the SGEC is controlled by a program in host memory called the port driver. The SGEC and the port driver communicate through two data structures: *Network Interface Command and Status Registers (NICSRs)* located in the SGEC and mapped in the host I/O address space, and through *descriptor lists and data buffers*, , collectively called *Host Communication Area* , in host memory.

The NICSRs are used for initialization, global pointers, commands and global errors reporting, while the host memory resident structures handle the actions and statuses related to buffer management.

The SGEC can be viewed as two independent, concurrently executing processes: *Reception* and *Transmission*. After the SGEC completes its *Initialization* sequence, these two processes alternate between three states: *STOPPED, RUNNING* or *SUSPENDED*. State transitions occur as a result of port driver commands (writing to a NICSR) or various external events occurrences. Some of the port driver commands require the referenced process to be in a specific state.

A simple programming sequence of the chip may be summarized as:

1. After power on (or reset), verifying the self test completed successfully.
2. Writing NICSRs to set major parameters such as System Base Register, Interrupt Vector, Address Filtering mode and so on.

3. Creating the transmit and receive lists in memory and writing the NICSRs to identify them to the SGEC.
4. Placing a setup frame in the transmit list, to load the internal reception address filtering table.
5. Starting the Reception and Transmission processes placing them in the RUNNING state.
6. Waiting for SGEC interrupts. NICSR5 contains all the global interrupt status bits.
7. Remedying the suspension cause, if either of the Reception or Transmission processes enter the SUSPENDED state.
8. Issuing a *Tx Poll Demand* command, to return the Transmission process to the RUNNING state. In addition to remedy the Reception process suspension cause, a Rx Poll Demand could be issued to return the Reception process to the RUNNING state.

   If the Rx poll demand is not issued, the Reception process will return to the RUNNING state when the SGEC receives the next recognized incoming frame.

The following sections contain detailed programming and state transitions information.

10.3.1 Command and Status Registers

The SGEC contains 16 command and status registers which may be accessed by the host.

10.3.2 Host access to NICSRs

The SGEC's NICSRs are located in VAX I/O address space.

The NICSRs must be **longword** aligned and can only be accessed using **longword** instructions. The address of NICSRx is the base address plus 4x bytes. For example, if the base address is 2000 8000, then the address of NICSR2 is 2000 8008. In the following paragraphs, NICSRs bits are specified with several access modes. The different access modes for bits are as follows:

Table 10–1: Bit access modes

Bit marked	Meaning
0	Reserved for future expansion - Ignored on Write, Read as “0”
1	Reserved for future expansion - Ignored on Write, Read as “1”
R	Read only, ignored on Write
R/W	Read or Write
W	Write only, unpredictable on Read
R/W1	Read, or Clear by writing a “1”. Writing with a “0” has no effect.

In order to save chip real estate, yet not tie up the host bus for extended periods of time, the 16 NICSRs are subdivided into two groups:

1. Physical NICSRs - 0 through 7, 15.
2. Virtual NICSRs - 8 through 14.

The group the NICSR is part of, determines the way the host will access it.

10.3.2.1 Physical NICSRs

These registers are physically present in the chip. Host access to these NICSRs is by a single instruction (For example, MOVL). There is no host perceivable delay and the instruction completes immediately. Most commonly used SGEC features are contained in the physical NICSRs.

10.3.2.2 Virtual NICSRs

These registers are not physically present in the SGEC and are incarnated by the on-chip processor. Accesses to SGEC functions implied by these registers may take up to 20 µseconds. So as not to tie up the host bus, virtual NICSR access requires several steps by the host.

NICSR5<DN> is used to synchronize access to the virtual NICSRs: after the first virtual NICSR access, the SGEC de-asserts NICSR5<DN> until it will complete the action.

NOTE

Accessing the virtual NICSRs, without polling first on the NICSR5<DN> reassertion will cause unpredictable results.

10.3.2.2.1 NICSR write

To write to a virtual NICSR the host takes the following actions:

1. Issue a write NICSR instruction. Instruction completes immediately, but the data is not yet copied by the SGEC.
2. Wait for NICSR5<DN>. *No SGEC virtual NICSR may be accessed before NICSR5<DN> asserts.*

10.3.2.2.2 NICSR read

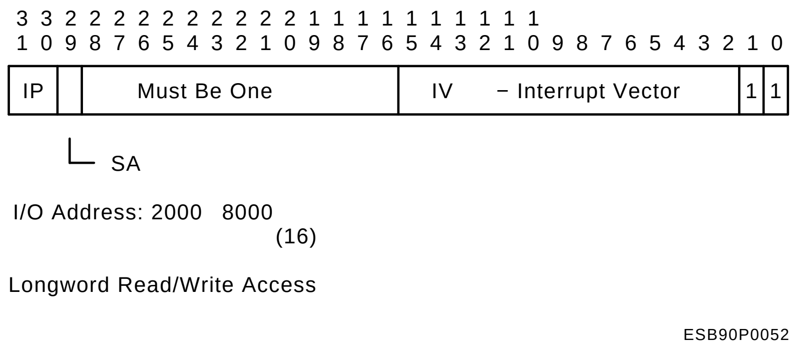
To read a virtual NICSR the host takes the following actions:

1. Issue a read NICSR instruction. Instruction completes immediately, but no valid data is sent to the host.
2. Wait for NICSR5<DN>. *No SGEC virtual NICSR may be accessed before NICSR5<DN> asserts.*
3. Reissue a read NICSR instruction, to the *same* NICSR as in step 1. The host receives valid data.

10.3.3 Vector Address, IPL, Sync/Asynch (NICSR0)

Because the SGEC may generate an interrupt, on parity errors, during host writes to NICSR's, this register must be the first one written by the host. The format is shown in Figure 10–2 and the bit description is given in Table 10–2.

Figure 10–2: Vector Address, IPL, Sync/Asynch (NICSR0)



NOTE

A parity error during NICSR0 host write may cause a host system crash due to an erroneous Interrupt Vector. To protect against such an eventuality, NICSR0 must be written as follows while the IPL - to which the SGEC is assigned - is disabled:

1. Write NICSR0.
2. Read NICSR0.
3. Compare value read to value written. If values mismatch, repeat from step 1.
4. Read NICSR5 and examine NICSR5<ME> for pending parity interrupt. Should an interrupt be pending, write NICSR5 to clear it.

Table 10–2: NICSR0 bits

Bit	Name	Access	Description
<31:30>	IP	R/W	Interrupt Priority - is the VAX interrupt priority level that the SGEC will respond to.

IP	IPL (hex)
00	14
01	15
10	16
11	17

Bit	Name	Access	Description
			Although the SGEC has only one interrupt request pin, that pin might be wired to any of the four IRQ pins on the host. The value in IP should be 14_{16} for the KA690.
<29>	SA	R/W	Sync/Asynch - This bit determines the SGEC operating mode when it is the bus master. When set, the SGEC will operate as a synchronous device and when clear, the SGEC will operate as an asynchronous device.
<15:00>	IV	R/W	Interrupt Vector - During an Interrupt Acknowledge cycle for an SGEC interrupt, this is the value that the SGEC will drive on the host bus CDAL<31:0> pins (CDAL pins <1:0> and <31:16> are set to “0”). Bits <1:0> are ignored when NICSR0 is written, and set to “1” when read.

Table 10–3: NICSR0 access

Value after **RESET**:	1FFF0003 hex
Read access rules:	None
Write access rules:	The IPL to which the SGEC is assigned - must be **DISABLED**

10.3.4 Transmit Polling Demand (NICSR1)

The Polling Demand NICSR (NICSR1) is used by the port driver to tell the SGEC that it put a packet on the transmit or receive list. The format is shown in Figure 10–3 and the bit description is in Table 10–4.

Figure 10–3: Polling Demand (NICSR1)

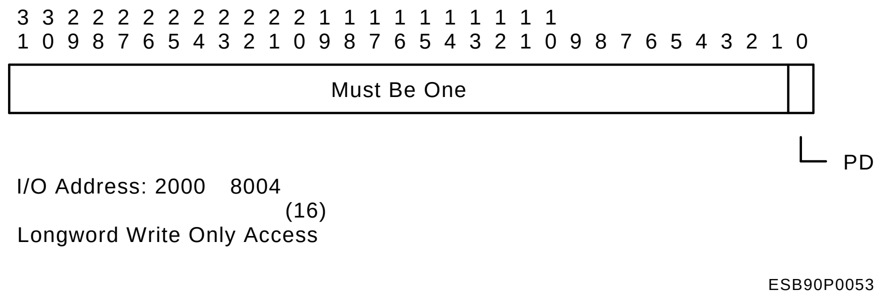


Table 10–4: NICSR1 bits

Bit	Name	Access	Description
<31:01>	MBZ		Must be one. This field is reserved for future expansion. Write as ONE.
<00>	PD	W	Tx Polling Demand - Checks the transmit list for frames to be transmitted. The PD value is meaningless.

Table 10–5: NICSR1 access

Value after **RESET**:	not applicable
Read access rules:	None
Write access rules:	Tx process SUSPENDED

10.3.5 Receive Polling Demand (NICSR2)

Figure 10–4: NICSR2 format

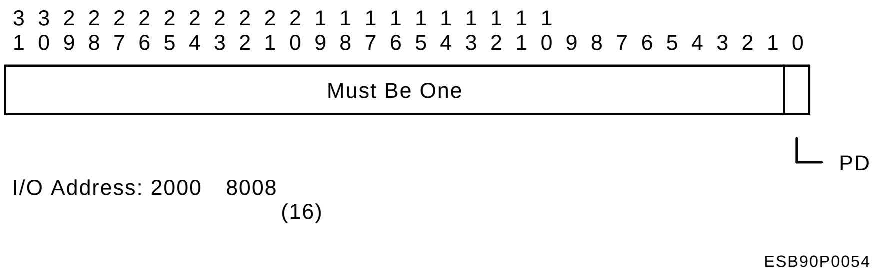


Table 10–6: NICSR2 bits

Bit	Name	Access	Description
<31:01>	MBO		Must be one. This field is reserved for future expansion. Write as ONE.
<00>	PD	W	Rx Polling Demand - Checks the receive list for receive descriptors to be acquired. The PD value is meaningless.

Table 10–7: NICSR2 access

Value after **RESET**:	not applicable
Read access rules:	None
Write access rules:	Rx process SUSPENDED

10.3.6 Descriptor List Addresses (NICSR3, NICSR4)

The two descriptor list address registers are identical in function, one being used for the transmit buffer descriptors and one being used for the receive buffer descriptors. In both cases, the registers are used to point the SGEC to the start of the appropriate buffer descriptor list.

The descriptor lists reside in VAX **physical** memory space and must be **longword** aligned.

NOTE

For best performance, it is recommended that the descriptor lists be OCTAWORD aligned.

TRANSMIT LIST

If the Transmit descriptor list is built as a ring (the chain descriptor points at the first descriptor of the list), the ring must contain, at least, TWO descriptors in addition to the chain descriptor.

Initially, these registers **must be written before the respective Start command is given** (see Section 10.3.8), else the respective process will remain in the STOPPED state. New list head addresses are only acceptable while the respective process is in the STOPPED or SUSPENDED states. Addresses written while the respective process is in the RUNNING state, are *ignored* and *discarded*.

If the host attempts to read any of these registers before ever writing to them, the SGEC responds with unpredictable values.

Figure 10–5: Descriptor list addresses format

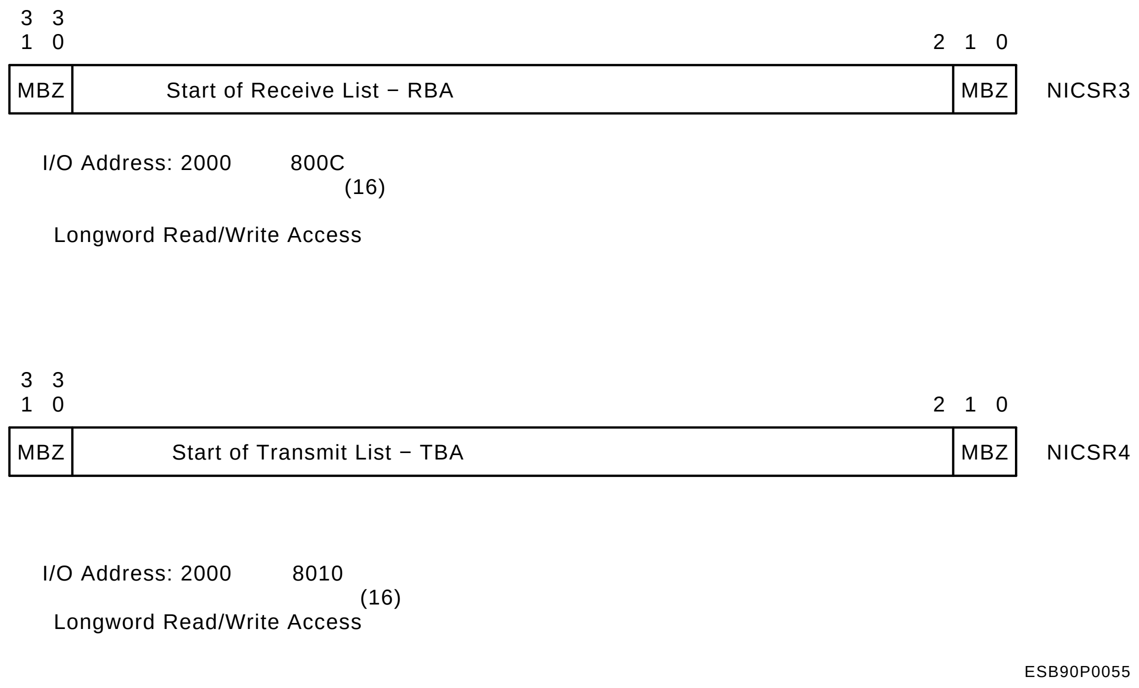

3 3
1 0
2 1 0
MBZ
Start of Receive List − RBA
MBZ
NICSR3
I/O Address: 2000 800C (16)
Longword Read/Write Access
3 3
1 0
2 1 0
MBZ
Start of Transmit List − TBA
MBZ
NICSR4
I/O Address: 2000 8010 (16)
Longword Read/Write Access
ESB90P0055

Table 10–8: Descriptor lists addresses bits

Bit	Name	Access	Description
<31:30>	MBZ		Must be ZERO. Ignored on writes, read as ZERO

Table 10–8 (Cont.): Descriptor lists addresses bits

Bit	Name	Access	Description
<29:00>	RBA or TBA	R/W	Address of the start of the receive list (NICSR3) or transmit list (NICSR4).This is a 30-bit VAX physical address.

NOTE

The descriptor lists must be longword aligned.

Table 10–9: NICSR3 access

Value after **RESET**:	unpredictable
Read access rules:	None
Write access rules:	Rx process STOPPED or SUSPENDED

Table 10–10: NICSR4 access

Value after **RESET**:	unpredictable
Read access rules:	None
Write access rules:	Tx process STOPPED or SUSPENDED

After either of NICSR3 or NICSR4 are written, the new address is readable from the written NICSR. However, if the SGEC status did not match the related write access rules, the new address does not take effect and the written information is *lost*, **EVEN if the SGEC matches later the right condition**.

10.3.7 Status Register (NICSR5)

This register contains all the status bits the SGEC reports to the host. Figure 10–6 shows the register format and Table 10–11 describes the register bits.

Figure 10–6: NICSR5 bits

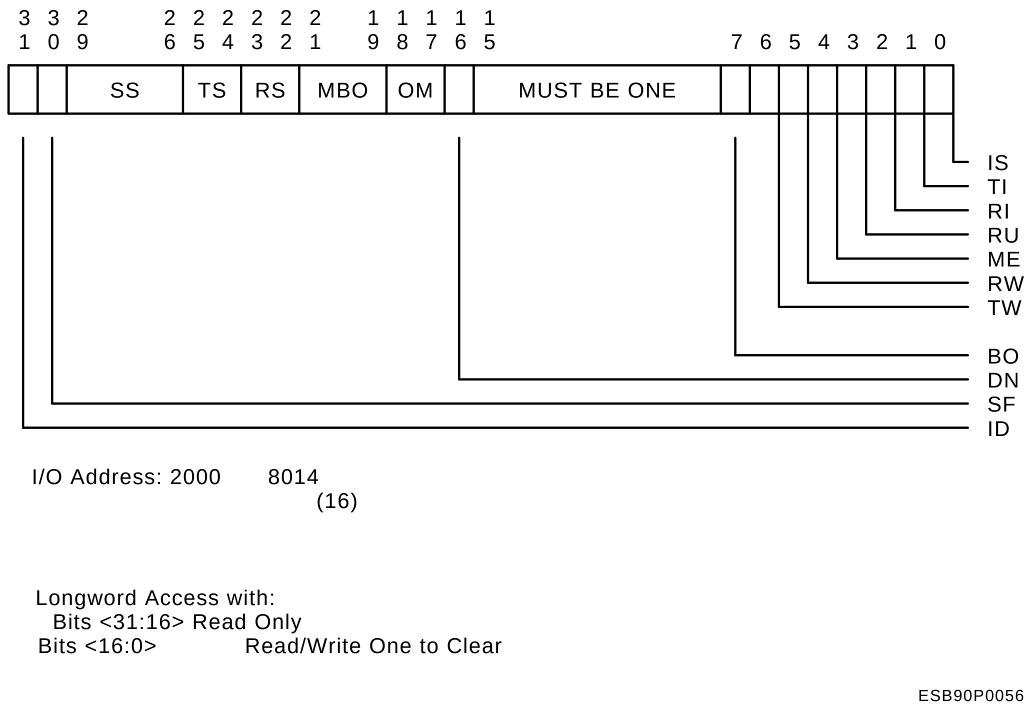


Table 10–11: NICSR5 bits

Bit	Name	Access	Description
<31>	ID	R	Initialization Done - When set, indicates the SGEC has completed the Initialization (reset and self test) sequences, and is ready for further commands. When clear, indicates the SGEC is performing the Initialization sequence and ignoring all commands. After the Initialization sequence completes, the Transmission and Reception processes are in the STOPPED state.
<30>	SF	R	Self test Failed - When set, indicates the SGEC self test has failed. The self test completion code bits indicate the failure type.

Table 10–11 (Cont.): NICSR5 bits

Bit	Name	Access	Description
<29:26>	SS	R	Self test Status - The self test completion code according to the following table. Only valid if SF is set.

Value	Meaning
0001	ROM error
0010	RAM error
0011	Address filter RAM error
0100	Transmit FIFO error
0101	Receive FIFO error
0110	Self_test loopback error

INFO

Self test takes 25ms to complete after Hardware or Software RESET.

<25:24>	TS	R	Transmission process state - Indicates the current state of the Transmission process, as follows:

Value	Meaning
00	STOPPED
01	RUNNING
10	SUSPENDED

Section 10.3.19.5 explains the Transmission process operation and state transitions.

<23:22>	RS	R	Reception process State - Indicates the current state of the Reception process, as follows:

Value	Meaning
00	STOPPED
01	RUNNING
10	SUSPENDED

Section 10.3.19.4 explains the Reception process operation and state transitions.

Table 10–11 (Cont.): NICSR5 bits

Bit	Name	Access	Description
<18:17>	OM	R	Operating Mode - These bits indicate the current SGEC operating mode as in the following table:
			Value — **Meaning**
			00 — Normal operating mode.
			01 — Internal Loopback - Indicates the SGEC is disengaged from the Ethernet wire. Frames from the transmit list are looped back to the receive list, subject to address filtering. Section 10.3.19.6 explains this mode of operation.
			10 — External Loopback - Indicates the SGEC is working in full duplex mode. Frames from the transmit list are transmitted on the Ethernet wire and also looped back to the receive list, subject to address filtering. Section 10.3.19.6 explains this mode of operation.
			11 — Reserved for Diagnostics
<16>	DN	R	Done - When set, indicates the SGEC has completed a requested virtual NICSR access. After a reset, this bit is set.
<15:8>	MBO		Must Be One . This field is reserved. Writes are ignored, read as ONE.
<7>	BO	R/W1	Boot_Message - When set, indicates that the SGEC has detected a boot_message on the serial line and has set the external pin BOOT_L.
<6>	TW	R/W1	Transmit Watchdog Timer interrupt - When set, indicates the transmit watchdog timer has timed out, indicating the SGEC transmitter was babbling. The Transmission process is *aborted* and placed in the STOPPED state. (Also reported into the Tx descriptor status TDES0<TO> flag).

Table 10–11 (Cont.): NICSR5 bits

Bit	Name	Access	Description
<5>	RW	R/W1	Receive Watchdog Timer interrupt - When set, indicates the Receive Watchdog Timer has timed out, indicating that some other node is babbling on the network. Current frame reception is aborted and RDES0<LE> and RDES0<LS> will be set. Bit NICSR5<RI> will also set. The Reception process remains in the RUNNING state.
<4>	ME	R/W1	Memory Error - Is set when any of the followings occur: • SGEC is the CP-BUS Master and the ERR_L pin is asserted by external logic (generally indicative of a memory problem). • Parity error detected on an host to SGEC NICSR write or SGEC read from memory. When a Memory Error is set, the Reception and Transmission processes are **aborted** and placed in the STOPPED state. **NOTE** At this point, it is mandatory that the port driver issue a Reset command and rewrite all NICSRs.
<3>	RU	R/W1	Receive buffer Unavailable - When set, indicates that the next descriptor on the receive list is owned by the host and could not be acquired by the SGEC. The Reception process is placed in the SUSPENDED state. Section 10.3.19.4 explains the Reception process state transitions. Once set by the SGEC, this bit will not be set again until the SGEC encounters a descriptor it can not acquire. To resume processing receive descriptors, the host must flip the ownership bit of the descriptor and can issue the Rx Poll Demand command. If no Rx poll demand is issued, the Reception process resumes when the next recognized incoming frame is received.
<2>	RI	R/W1	Receive Interrupt - When set, indicates that a frame has been placed on the receive list. Frame specific status information was posted in the descriptor. The Reception process remains in the RUNNING state.

Table 10–11 (Cont.): NICSR5 bits

Bit	Name	Access	Description
<1>	TI	R/W1	Transmit Interrupt - When set, indicates one of the following: • Either all the frames in the transmit list have been transmitted (next descriptor owned by the host), or a frame transmission was aborted due to a locally induced error. The port driver must scan down the list of descriptors to determine the exact cause. The Transmission process is placed in the SUSPENDED state. Section 10.3.19.5 explains the Transmission process state transitions. To resume processing transmit descriptors, the port driver must issue the Poll Demand command. • A frame transmission completed, and TDES1<IC> was set. The Transmission process remains in the RUNNING state, unless the next descriptor is owned by the host or the frame transmission aborted due to an error. In the latter cases, the Transmission process is placed in the SUSPENDED state.
<0>	IS	R/W1	Interrupt Summary - The logical “OR” of NICSR5 bits 1 through 6.

Table 10–13: NICSR5 access

Value after **RESET**:	0039FF00 hex
Read access rules:	None
Write access rules:	NICSR5<07:01> bits cleared by 1, others bits not write-able

10.3.7.1 NICSR5 status report

The Status register NICSR5 is split into two words:

- the high word which contains the global status of the SGEC, as the initialization status, the DMA and operation mode and the Receive and Transmit process states.

-the low word which contains the status related to the Receive and Transmit frames.

Any change of the NICSR5 bits <ID>, <SF>, <OM> or <DN> - which is always the result of a host command - is reported without an interrupt.

Any process state change **initiated by a host command** NICSR6<ST> or NICSR6<SR>, is reported without an interrupt.

In the above two cases, the driver must poll on NICSR5 to get the acknowledge of its command (For example, polling on <ID, SF> after Reset or polling on <TS> after a START_TX command).

Any process state change **initiated by the SGEC activity** is immediately followed by at least one of the NICSR5<6:1> interrupts and the interrupt_summary NICSR5<IS>.

The SGEC 16 bit internal processor updates the 32 bits NICSR5 register in two phases: the high word is modified first, then the low word is written, which generates an interrupt to the host. In this case, the driver must scan first the NICSR5 low word to get the interrupt status, then the NICSR high word to get the related process state. (For example, <TI> interrupt with <TS> = SUSPENDED reports an end of transmission due to a Tx descriptor unavailable.)

If the host polls on the process state change, it may detect a change without interrupt, due to the small time window separating the NICSR5 high word and low word updates.

Maximum time window is 4*Tcycles of the host clock.

10.3.8 Command and Mode Register (NICSR6)

This register is used to establish operating modes and for port driver commands.

Figure 10–7: NICSR6 format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE	IE	r	BL				Must be ONE				BE	SE	r	r	r	Must be ONE				ST	SR	OM		DC	FC	r	r	PB	AF		r

I/O Address: 2000 8018 (16)

Longword Read/Write Access

r = reserved

ESB90P0057

Table 10–14: NICSR6 bits

Bit	Name	Access	Description
<31>	RE	R/W	Reset command - Upon being set, the SGEC will abort all processes and start the reset sequence. After completing the reset and self test sequence, the SGEC will set bit NICSR5<ID>. Clearing this bit has no effect. **NOTE** The NICSR6<RE> value is unpredictable on read after HARDWARE reset.
<30>	IE	R/W	Interrupt Enable mode - When set, setting of NICSR5 bits 1 through 6 will cause an interrupt to be generated.
<29>	r		reserved
<28:25>	BL	R/W	Burst Limit mode - Specifies the maximum number of longwords to be transferred in a single DMA burst on the host bus. When NICSR6<SE> is cleared, permissible values are 1,2,4,8 . When SE is set, the only permissible values are 1 and 4: a value of 2 or 8 is respectively forced to 1 or 4. After Initialization, the burst limit is set to 1.
<24:21>	MBO		This field is reserved Writes are ignored, read as one.
<20>	BE	R/W	Boot_message Enable mode - When set, enables the boot_message recognition. When the SGEC recognizes an incoming boot message on the serial line, NICSR5<BO> is set and the external pin BOOT_L is asserted for a duration of 6*Tcycles (of the host clock).
<19>	SE	R/W	Single_cycle Enable mode - When set, the SGEC transfers only a single longword or an octaword in a single DMA burst on the host bus.
<18:12>	MBO		Must Be One. This field is reserved. Writes are ignored, read as ONE.
<11>	ST	R/W	Start/Stop Transmission command - When set, the Transmission process is placed the RUNNING state, the SGEC checks the transmit list at the *current* position for a frame to transmit - the address set by *NICSR4* or the position retained when the Tx process was previously stopped -. If it does not find a frame to transmit, the Transmission process enters the SUSPENDED state. The Start Transmission command is honored only when the Transmission process is in the STOPPED state. *The first time this command is issued, an additional requirement is that NICSR4 have already been written to, else the Transmission process will remain in the STOPPED state.* When cleared the Transmission process is placed in the STOPPED state after completing transmission of the current frame. The next descriptor position in the transmit list is saved, and becomes the current position after transmission is restarted. The Stop Transmission command is honored only when the Transmission process is in the RUNNING or SUSPENDED states. Refer to Section 10.3.19.5 for more information.

Table 10–14 (Cont.): NICSR6 bits

Bit	Name	Access	Description
<10>	SR	R/W	Start/Stop Reception command - When set, the Reception process is placed in the RUNNING state, the SGEC attempts to acquire a descriptor from the receive list and process incoming frames. Descriptor acquisition is attempted from the *current* position in the list - the address set by *NICSR3* or the position retained when the Rx process was previously stopped -. If no descriptor can be acquired, the Reception process enters the SUSPENDED state. The Start Reception command is honored only when the Reception process is in the STOPPED state. *The first time this command is issued, an additional requirement is that NICSR3 have already been written to, else the Reception process will remain in the STOPPED state.* When cleared, the Reception process is placed in the STOPPED state, after completing reception of the current frame. The next descriptor position in the receive list is saved, and becomes the *current* position after reception is restarted. The Stop Reception command is honored only when the Reception process is in the RUNNING or SUSPENDED states. Refer to Section 10.3.19.4 for more information.
<9:8>	OM	R/W	Operating Mode - These bits determine the SGEC main operating mode.
<7>	DC	R/W	Disable data Chaining mode - When set, no data chaining will occur in reception; frames, longer than the current receive buffer, will be truncated. RDES0<FS,LS> will always be set. The frame length returned in RDES0<FL> will be the *true* length of the non-truncated frame while RDES0<BO> will indicate that the frame has been truncated due to buffer overflow. When clear, frames too long for the current receive buffer, will be transferred to the next buffer(s) in the receive list.

Value	Meaning
00	Normal operating mode.
01	Internal Loopback - The SGEC will loopback buffers from the transmit list. The data will be passed from the transmit logic back to the receive logic. The receive logic will treat the looped frame as it would any other frame, and subject it to the address filtering and validity check process.
10	External Loopback - The SGEC transmits normally and in addition, will enable its receive logic to its own transmissions. The receive logic will treat the looped frame as it would any other frame, and subject it to the address filtering and validity check process.
11	Reserved for Diagnostic

Table 10–14 (Cont.): NICSR6 bits

Bit	Name	Access	Description
<6>	FC	R/W	Force Collision mode - This bit allows the collision logic to be tested. The chip must be in **internal loopback** mode for FC to be valid. If FC is set, a collision will be forced during the next transmission attempt. This will result in 16 transmission attempts with Excessive Collision reported in the transmit descriptor.
<5:4>	MBO		Must Be ONE. This field is reserved. Writes are ignored, read as ONE.
<3>	PB	R/W	Pass Bad Frames mode - When this bit is set, the SGEC will pass frames that have been damaged by collisions or are too short due to premature reception termination. Both events should have occurred within the collision window (64 bytes), else other errors will be reported. When clear, these frames will be discarded and never show up in the host receive buffers. **NOTE** Pass Bad Frames is subject the address filtering mode. For example, to monitor the network, this mode must be set together with the promiscuous address filtering mode.
<2:1>	AF	R/W	Address Filtering mode - These bits define the way incoming frames will be address filtered:
<0>	MBO		Must Be ONE. This field is reserved. Writes are ignored, read as ONE.

Value	Meaning
00	Normal - Incoming frames will be filtered according to the values of the <HP> and <IF> bits of the setup frame descriptor.
01	Promiscuous - All incoming frames will be passed to the host, regardless of the <HP> bit value.
10	All Multicast - All incoming frames with Multicast address destinations will be passed to the host. Incoming frames with physical address destinations will be filtered according to the <HP> bit value.
11	Unused - Reserved.

Table 10–15: NICSR6 access

Value after **RESET**:	83E0F000 hex or 03E0F000 hex
Read access rules:	None
Write access rules:	
* <RE, IE, BE>	Unconditional
* <BL, SE, OM>	Rx and Tx processes STOPPED
* <FC>	Rx and Tx processes STOPPED, Internal_Loopback mode
* <DC, PB, AF>	Rx STOPPED
* Start_Receive <SR>=1	Rx STOPPED and NICSR3 Initialized
* Start_Transmit <ST>=1	Tx STOPPED and NICSR4 Initialized
* Stop_Receive <SR>=0	Rx RUNNING or SUSPENDED
* Stop_Transmit <ST>=0	Tx RUNNING or SUSPENDED

After NICSR6 is written, the new value is readable from NICSR6. However, if the SGEC status does not match the related write access rules, the new mode setting and command do not take effect and the written information is *lost*, **EVEN if the SGEC matches later the right condition**.

10.3.9 System Base Register (NICSR7)

This NICSR contains the physical starting address of the VAX System Page Table. This register must be loaded by host software before any address translation occurs so that memory will not be corrupted.

Figure 10–8: NICSR7 format

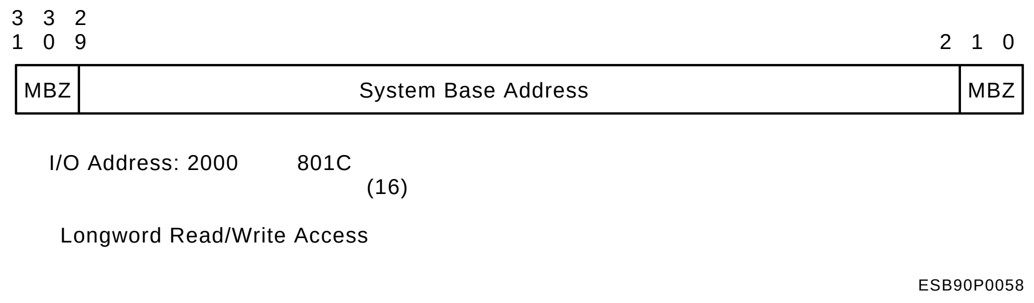


Table 10–16: NICSR7 bits

Bit	Name	Access	Description
<31:30>	MBZ		Must Be ZERO. Read as zero. Writes are ignored.
<29:00>	SB	R/W	System Base address - The physical starting address of the VAX System Page Table. Not used if VA (Virtual Addressing) is cleared in all descriptors. **This register should be loaded only once after a reset. Subsequent modifications of this register at any other time may cause unpredictable results.**

Table 10–17: NICSR7 access

Value after **RESET**:	unpredictable
Read access rules:	None
Write access rules:	Writing once after initialization

10.3.10 Reserved register (NICSR8)

This entire register is reserved.

10.3.11 Watchdog Timers (NICSR9)

The SGEC has two timers that restrict the length of time in which the chip can receive or transmit.

Figure 10–9: NICSR9 format

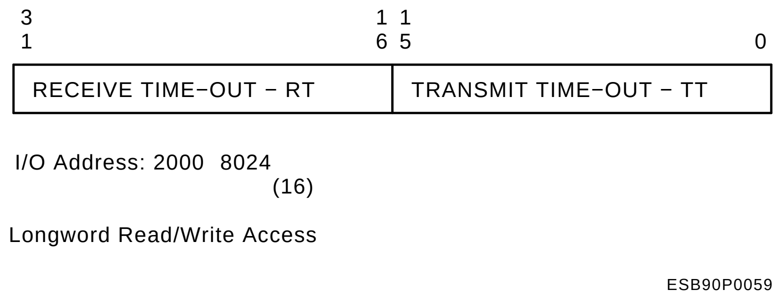


Table 10–18: NICSR9 bits

Bit	Name	Access	Description
<31:16>	RT	R/W	RECEIVE WATCHDOG TIME-OUT - The Receive Watchdog Timer protects the host CPU against babbling transmitters on the network. If the receiver stays on for $RT * 16$ cycles of the serial clock, the SGEC will cut off reception and set the NICSR5<RW> bit. If the timer is set to zero, it will never time-out. The value of RT is an unsigned integer. With a 10 Mhz serial clock, this provides a range of 72µs to 100ms. The default value is 1250 corresponding to 2ms. The Rx watchdog timer is programmed only while the Reception process is in the STOPPED state. **NOTE** A Rx watchdog value between 1 and 44 is forced to the minimum time_out value of 45 (72µs).

Table 10–18 (Cont.): NICSR9 bits

Bit	Name	Access	Description
<15:00>	TT	R/W	TRANSMIT WATCHDOG TIME-OUT - The Transmit Watchdog Timer protects the network against babbling SGEC transmissions, on top of any such circuitry present in tranceivers. If the transmitter stays on for $TT * 16$ cycles of the serial clock, the SGEC will cut off the transmitter and set the NICSR5<TW> bit. If the timer is set to zero, it will never time-out. The value of TT is an unsigned integer. With a 10 Mhz serial clock, this provides a range of 72µs to 100ms. The default value is 1250 corresponding to 2ms. The Tx watchdog timer is programmed only while the Transmission process is in the STOPPED state. **NOTE** A Tx watchdog value between 1 and 44 is forced to the minimum time_out value of 45 (72µs).

Table 10–19: NICSR9 access

Value after **RESET**:	00000000 hex
Read access rules:	None
Write access rules:	
* Rx Watchdog timer	Rx process STOPPED
* Tx Watchdog timer	Tx process STOPPED

These watchdog timers are enabled by default. These timers will assume the default values after hardware or software resets.

10.3.12 Revision Number and Missed Frame Count (NICSR10)

This register contains a missed frame counter and SGEC identification information.

Figure 10–10: Revision Number and Missed Frame Count (VIRTUAL NICSR10)

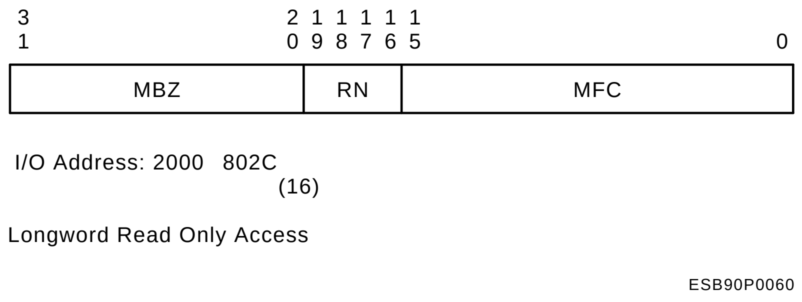


Table 10–20: NICSR10 bits

Bit	Name	Access	Description
<31:21>	MBZ		Must BE ZERO. Read as ZERO. Writes are ignored.
<20:16>	RN	R	Chip Revision Number - This stores the revision number for this particular SGEC.
<15:00>	MFC	R	Missed Frame Count - Counter for the number of frames that were discarded and lost because host receive buffers were unavailable. The counter is cleared when read by the host.

Table 10–21: NICSR10 access

Value after **RESET**:	00030000 hex
Read access rules:	Missed_frame counter cleared by read
Write access rules:	Not applicable

10.3.13 Boot Message (NICSR11, 12, 13)

These registers contain the Boot message VERIFICATION and PROCESSOR fields. The format is shown in Figure 10–11 and the bit descriptions are in Table 10–22.

Figure 10–11: Boot Message

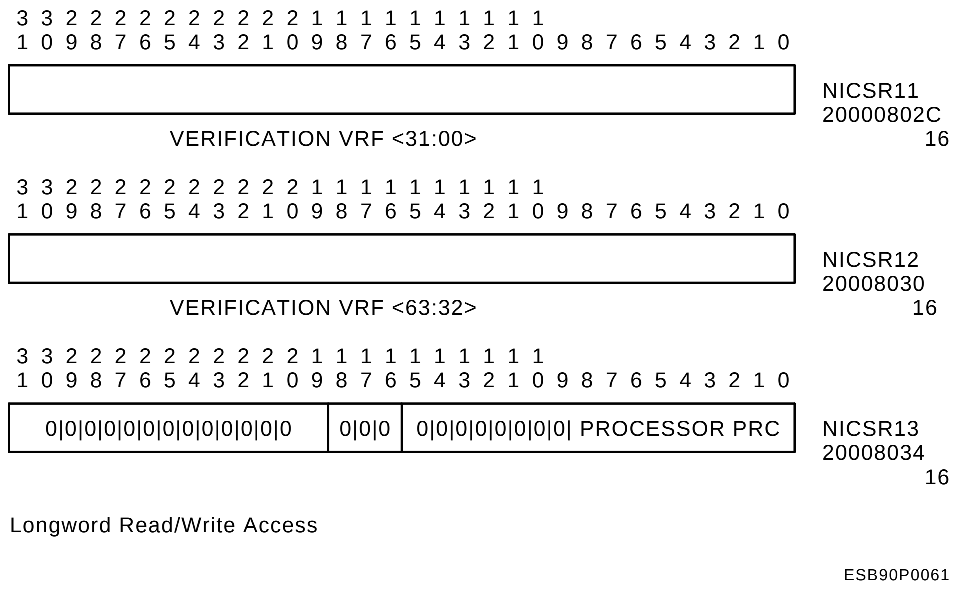


Table 10–22: NICSR11,12,13 bits

Bit	Name	Access	Description
NICSR11 <31:00>	VRF<31:00>	R/W	Boot message VERIFICATION field <31:00>
NICSR12 <31:00>	VRF<63:32>	R/W	Boot message VERIFICATION field <63:32>
NICSR13 <07:00>	PRC	R/W	Boot message PROCESSOR field -

NOTE

The least significant bit of the Verification Field (VRF<0>) corresponds to the first incoming bit of the verification field in the serial boot message.

Table 10–23: NICSR11,12,13 access

Value after **RESET**:	00000000 hex for each of NICSR11,NICSR12,NICSR13
Read access rules:	None
Write access rules:	Boot message DISABLED (<NICSR6<BE> = 0)

10.3.14 Diagnostic Registers (NICSR14, 15)

These registers are reserved for diagnostic features.

10.3.14.1 Diagnostic Breakpoint Address Register (NICSR14)

This register is virtual CSR. It contains the breakpoint address that will cause the internal CPU to jump to a patch address. Figure 10–12 shows the format of the register. Table 10–24 lists the bits and descriptions. This register can be loaded only in diagnostic mode (NICSR6 <OM>=<11>).

Figure 10–12: NICSR14 format

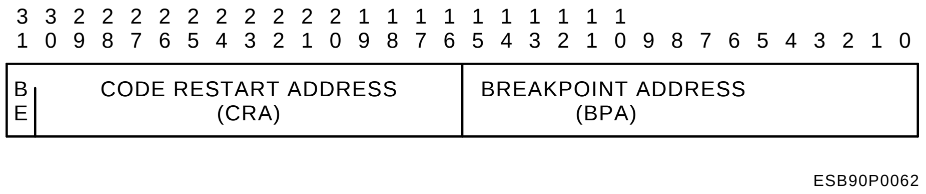


Table 10–24: NICSR14 bits

Bit	Name	Type	Description
<31>	BE	R/W	When set, Breakpoint enabled.
<30:16>	CRA	R/W	Code Restart Address - The first address in the internal RAM where the internal processor will jump to after a breakpoint occurred.
<15:0>	BPA	R/W	Breakpoint Address - The internal processor address at which the program will halt and jump to the RAM loaded code.

NOTE

This register in conjunction with the diagnostic descriptors allows software patches.

Table 10–25: NICSR14 access

Value after **RESET:**	00000000_{16}
Read access rules:	None
Write access rules:	DIAGNOSTIC mode
Violation:	Addressing NICSR14 while NICSR5<DN> is deasserted

10.3.14.2 Monitor Command Register (NICSR15)

This register is a physical CSR. It contains the bits which will select the internal test block operation mode. Figure 10–13 shows the format of the register. Table 10–26 lists the bits and descriptions.

Figure 10–13: NICSR15 format

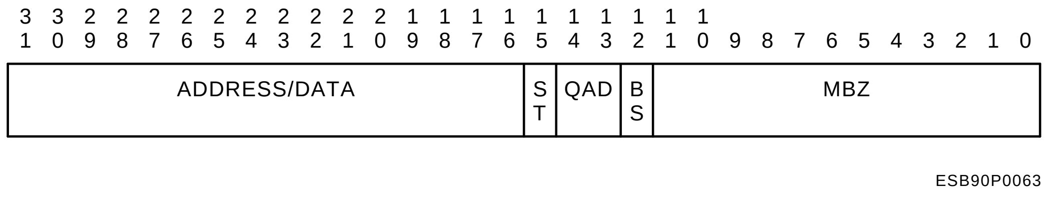


Table 10–26: NICSR15 bits

Bit	Name	Type	Description
<31:16>	ADDR/DATA	R/W	- Before the "Examine" cycle, it points to the location to be read. Three cycles after the assertion of <ST>, it contains the READ data.
<15>	ST	W	Start Read - When set, starts the "Examine" cycle: the data addressed by CSR<31:16> is fetched and stored into the same register field. Reset by hardware at the end of operation.
<14:13>	QAD	W	Quad selects bits- These bits define the specific four bits of the internal Data_bus or Address_ bus which are monitored on the external test pins BM_L/TEST<3:0>. Meaningful only in test mode (TSM=1). The 2 bit code is interpreted as follows.

QAD	data	address
00	<03:00>	<03:00>
01	<07:04>	<07:04>
10	<11:08>	<11:08>
11	<15:12>	0, IOP_WR_L,<13:12>

Table 10–26 (Cont.): NICSR15 bits

Bit	Name	Type	Description
<12>	BS	W	Bus select- When reset the internal Data_bus is monitored on the external test pins BM_L /TEST<3:0>. When set, the monitoring is applied on the internal Address_bus. Meaningful only in test mode (TSM=1).
<11:0>	MBZ	-	Must be zero.

Table 10–27: NICSR15 access

Value after **RESET**:	$00000FFF_{16}$
Read access rules:	None
Write access rules:	Reserved for DEBUGGING
Violation:	Setting <ST> with "random" SGEC internal address

10.3.15 Descriptors and buffers format

The SGEC transfers frame data to and from receive and transmit buffers in host memory. These buffers are pointed to by descriptors which are also resident in host memory.

There are two descriptor lists: one for receive and one for transmit. The starting address of each list is written into NICSRs 3 and 4 respectively. A descriptor list is a forward-linked (either implicitly or explicitly) list of descriptors, the last of which may point back to the first entry, thus creating a ring structure. Explicit chaining of descriptors, through setting xDES1<CA> is called *Descriptor Chaining*. The descriptor lists reside in VAX *physical* memory address space.

NOTE

The SGEC first reads the descriptors, ignoring all unused bits regardless of their state. The only word the SGEC writes back, is the first word (xDES0) of each descriptor. Unused bits in xDES0 will be written as “0”. Unused bits in xDES1 - xDES3 may be used by the port driver and the SGEC will never disturb them.

A data buffer can contain an entire frame or part of a frame, but it cannot contain more than a single frame. Buffers contain only data; buffer status is contained in the descriptor. The term *Data Chaining* is used to refer to frames spanning multiple data buffers. Data Chaining can be enabled or disabled, in reception, through NICSR6<DC>. Data buffers reside in VAX memory space, either physical or virtual.

NOTE

The virtual to physical address translation is based on the assumption that PTEs are locked in the host memory the time the SGEC owns the related buffer.

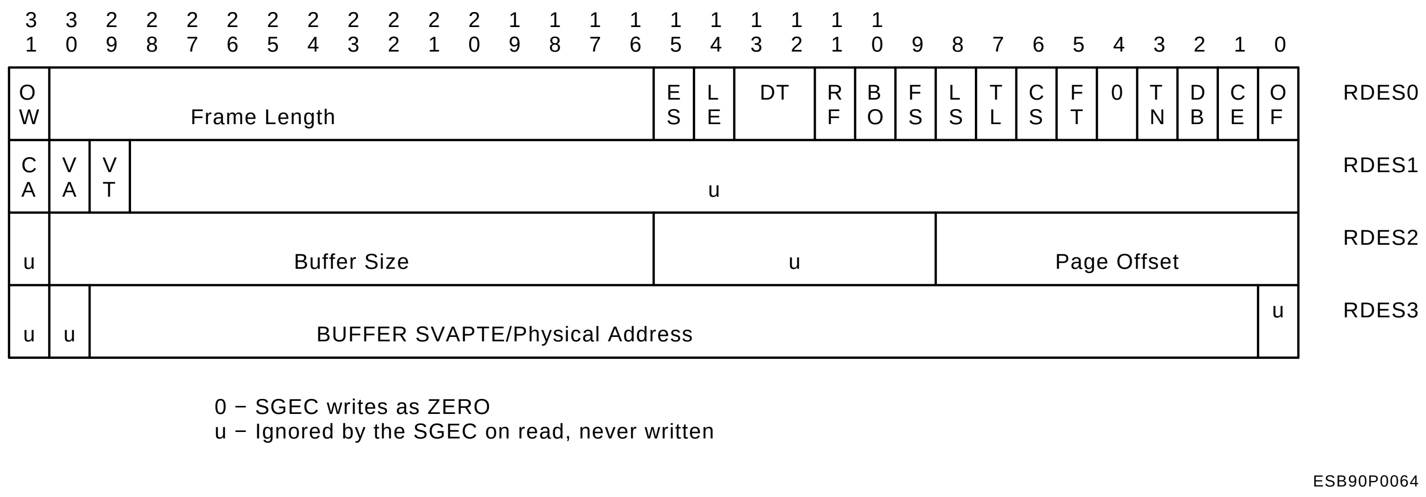
NOTE

For best performance in virtual addressing mode, PPTE vectors must not cross a page of the PPTE table.

10.3.16 Receive Descriptors

The receive descriptor format is shown in Figure 10–14, and described in the following paragraphs.

Figure 10–14: Receive descriptor format



10.3.16.1 RDES0 word

RDES0 word contains received frame status, length and descriptor ownership information.

Table 10–28: RDES0 bits

Bit	Name	Description
<31>	OW	Own bit - When set, indicates the descriptor is owned by the SGEC. When cleared, indicates the descriptor is owned by the host. The SGEC clears this bit upon completing processing of the descriptor and its associated buffer.
<30:16>	FL	Frame Length - The length in bytes of the received frame. Meaningless if RDES0<LE> is set.
<15>	ES	Error Summary - The logical "OR" of RDES0 bits OF,CE,TN,CS,TL,LE,RF.
<14>	LE	Length Error - When set, indicates a frame truncation caused by one of the following: • The frame segment does not fit within the current buffer and the SGEC does not own the next descriptor. The frame is truncated. • The Receive Watchdog timer expired. NICSR5<RW> is also set.
<13:12>	DT	Data Type - Indicates the type of frame the buffer contains, according to the following table:

Value	Meaning
00	Serial received frame
01	Internally looped back frame
10	Externally looped back frame , Serial received frame [2]

Bit	Name	Description
<11>	RF	Runt Frame - When set, indicates this frame was damaged by a collision or premature termination before the collision window had passed. Runt frames will only be passed on to the host if (NICSR6<PB>) is set. Meaningless if RDES0<OF> is set.
<10>	BO	Buffer overflow - When set, indicates that the frame has been truncated due to a buffer too small to fit the frame size. This bit may only be set if Data Chaining is disabled (NICSR6<DC> = 1).
<09>	FS	First Segment - When set, indicates this buffer contains the first segment of a frame.
<08>	LS	Last Segment - When set, indicates this buffer contains the last segment of a frame and status information is valid.
<07>	TL	Frame Too Long - When set, indicates the frame length exceeds the maximum Ethernet specified size of 1518 bytes. **NOTE** Frame Too Long is only a frame length indication and does not cause any frame truncation.
<06>	CS	Collision Seen - When set, indicates the frame was damaged by a collision that occurred after the 64 bytes following the SFD.

Table 10–28 (Cont.): RDES0 bits

Bit	Name	Description
<05>	FT	Frame Type - When set, indicates the frame is an Ethernet type frame (Frame Length_Field > 1500). When clear, indicates the frame is an IEEE 802.3 type frame. Meaningless for Runt frames < 14 bytes.
<04>	0	Zero. SGEC writes as ZERO.
<03>	TN	Translation Not Valid - When set, indicates that a translation error occurred when the SGEC was translating a VAX virtual buffer address. It will only set if RDES1<VA> was set. The Reception process remains in the RUNNING state and attempts to acquire the next descriptor.
<02>	DB	Dribbling Bits - When set, indicates the frame contained a non-integer multiple of eight bits. This error will be reported only if the number of dribbling bits in the last byte is greater than two. Meaningless if RDES0<CS> or RDES0<RF> are set. The CRC check is performed independent of this error, however, only whole bytes are run through the CRC logic. **Consequently, received frames with up to six dribbling bits will have this bit set, but if <CE> (or another error indicator) is not set, these frames should be considered valid:**
<01>	CE	CRC Error - When set, indicates that a CRC error has occurred on the received frame.
<00>	OF	Overflow - When set, indicates received data in this descriptor's buffer was corrupted due to internal FIFO overflow. This will generally occur if SGEC DMA requests are not granted before the internal receive FIFO fills up.

CE	DB	Error
0	0	None
0	1	None
1	0	CRC error
1	1	Alignment error

10.3.16.2 RDES1 word

Table 10–29: RDES1 bits

Bit	Name	Descriptor
<31>	CA	Chain Address - When set, RDES3 is interpreted as another descriptor's VAX physical address. This allows the SGEC to process multiple, non-contiguous descriptor lists and explicitly "chain" the lists. Note that contiguous descriptors are implicitly chained. In contrast to what is done for a Rx buffer descriptor, the SGEC clears neither the ownership bit RDES0<OW> nor one of the other bits of RDES0 of the chain descriptor after processing. To protect against infinite loop, a chain descriptor pointing back itself, is seen as **owned by the host**, regardless of the ownership bit state.
<30>	VA	Virtual Addressing - When set, RDES3 is interpreted as a virtual address. The type of virtual address translation is determined by the RDES1<VT> bit. The SGEC uses RDES3 and RDES2<Page Offset> to perform a VAX virtual address translation process to obtain the physical address of the buffer. When clear, RDES3 is interpreted as the actual physical address of the buffer:
<29>	VT	Virtual Type - In case of virtual addressing (RDES1<VA> = 1), indicates the type of virtual address translation. When set, the buffer address RDES3 is interpreted as a SVAPTE (System Virtual Address of the Page Table Entry). When clear, the buffer address is interpreted as a PAPTE (Physical Address of the Page Table Entry). Meaningful only if RDES1<VA> is set.
<28:0>	u	Unused. Ignored by the SGEC on reads. Never written.

VA	VT	Addressing mode
0	x	Physical
1	0	Virtual - SVAPTE
1	1	Virtual - PAPTE

10.3.16.3 RDES2 word

Table 10–30: RDES2 bits

Bit	Name	Descriptor
<31>	u	Unused. Ignored by the SGEC on reads. Never written.
<30:16>	BS	Buffer Size - The size, in bytes, of the data buffer.

NOTE

Receive buffers size must be an EVEN number of bytes.

Table 10–30 (Cont.): RDES2 bits

Bit	Name	Descriptor
<15:9>	u	Unused. Ignored by the SGEC on reads. Never written.
<08:00>	PO	Page Offset - The byte offset of the buffer within the page. Only meaningful if RDES1<VA> is set. **NOTE** Receive buffers must be word aligned.

10.3.16.4 RDES3 word

Table 10–31: RDES3 bits

Bit	Name	Descriptor
<31:00>	SV/PV/PA	SVAPTE/PAPTE/Physical Address - When RDES1<VA> is set, RDES3 is interpreted as the address of the Page Table Entry and used in the virtual address translation process. The type of the address System Virtual address (SVAPTE) or Physical Address (PAPTE) is determined by RDES1<VT>. When RDES1<VA> is clear, RDES3 is interpreted as the physical address of the buffer. When RDES1<CA> is set, RDES3 is interpreted as the VAX physical address of another descriptor. **NOTE** Receive buffers must be word aligned.

10.3.16.5 Receive descriptor status validity

The following table summarizes the validity of the Receive descriptor status bits regarding the Reception completion status:

Table 10–32: Receive descriptor status validity

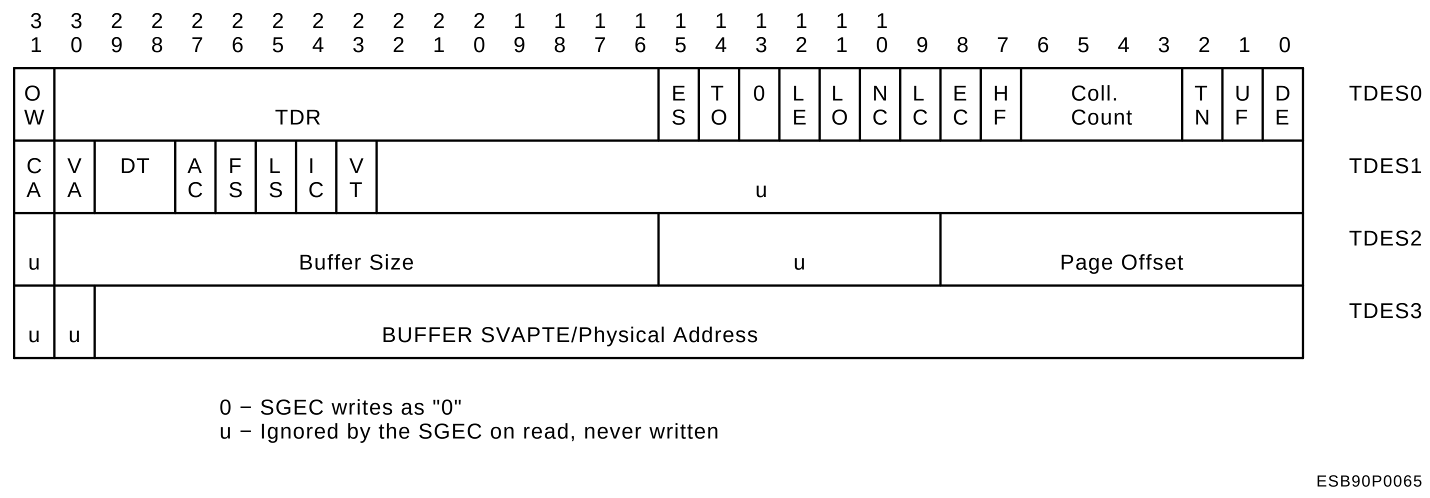
Reception	Rx Status report						
status	RF	TL	CS	FT	DB	CE	(ES,LE,BO,DT,FS,LS,FL,TN,OF)
Overflow	X	V	X	V	X	X	V
Collision after 512 bits	V	V	V	V	X	X	V
Runt frame	V	V	V	V	X	X	V
Runt frame < 14 bytes	V	V	V	X	X	X	V
Watchdog timeout	V	V	X	V	X	X	V

V - Valid
X - Meaningless

10.3.17 Transmit descriptors

The transmit descriptor format is shown in Figure 10–15, and described in the following paragraphs.

Figure 10–15: Transmit descriptor format



10.3.17.1 TDES0 word

TDES0 word contains transmitted frame status and descriptor ownership information.

Table 10–33: TDES0 bits

Bit	Name	Description
<31>	OW	Own bit - When set, indicates the descriptor is owned by the SGEC. When cleared, indicates the descriptor is owned by the host. The SGEC clears this bit upon completing processing of the descriptor and its associated buffer.
<29:16>	TDR	Time Domain Reflectometer - This is a count of bit time and is useful for locating a fault on the cable using the velocity of propagation on the cable. Only valid if TDES0<EC> is also set. Two Excessive Collisions in a row and with the same or similar (within 20) TDR values indicate a possible cable open.
<15>	ES	Error Summary - The logical “OR” of UF, TN, EC, LC, NC, LO, LE and TO.
<14>	TO	Transmit Watchdog Timeout - When set, indicates the transmit watchdog timer has timed out, indicating the SGEC transmitter was babbling. The interrupt NICSR5<TW> is set and the Transmission process is *aborted* and placed in the STOPPED state.
<13>	MBZ	Must be ZERO. SGEC writes as ZERO

Table 10–33 (Cont.): TDES0 bits

Bit	Name	Description
<12>	LE	Length Error - When set, indicates one of the following: • Descriptor unavailable (owned by the host) in the middle of data chained descriptors. • Zero length buffer in the middle of data chained descriptors. • Setup or Diagnostic descriptors (Data type TDES1<DT> <> 0) in the middle of data chained descriptors. • Incorrect order of first_segment TDES1<FS> and last_segment TDES1<LS> descriptors in the descriptor list. The Transmission process enters the SUSPENDED state and sets NICSR5<TI>.
<11>	LO	Loss of Carrier - When set, indicates loss of carrier during transmission (possible short circuit in the Ethernet cable). Meaningless in internal loopback mode (NICSR5<OM>=1).
<10>	NC	No Carrier - When set, indicates the carrier signal from the transceiver was not present during transmission (possible problem in the transceiver or transceiver cable). Meaningless in internal loopback mode (NICSR5<OM>=1).
<09>	LC	Late Collision - When set, indicates frame transmission was aborted due to a late collision. Meaningless if TDES0<UF>.
<08>	EC	Excessive Collisions - When set, indicates that the transmission was aborted because 16 successive collisions occurred while attempting to transmit the current frame.
<07>	HF	Heartbeat Fail - When set, indicates Heartbeat Collision Check failure (the transceiver failed to return a collision pulse as a check after the transmission. Some tranceivers do not generate heartbeat, and so will always have this bit set. If the transceiver does support it, it indicates transceiver failure.) Meaningless if TDES0<UF>.
<06:03>	CC	Collision Count - A four bit counter indicating the number of collisions that occurred before the transmission attempt succeeded or failed. Meaningless when TDES0<EC> is also set.
<02>	TN	Translation Not Valid - When set, indicates that a translation error occurred when the SGEC was translating a VAX virtual buffer address. It may only set if TDES1<VA> was set. The Transmission process enters the SUSPENDED state and sets NICSR5<TI>.
<01>	UF	Underflow Error - When set, indicates that the transmitter has truncated a message due to data late from memory. UF indicates that the SGEC encountered an empty transmit FIFO while in the midst of transmitting a frame. The Transmission process enters the SUSPENDED state and sets NICSR5<TI>.
<00>	DE	Deferred - When set, indicates that the SGEC had to defer while trying to transmit a frame. This condition occurs if the channel is busy when the SGEC is ready to transmit.

10.3.17.2 TDES1 word

Table 10–34: TDES1 bits

Bit	Name	Descriptor
<31>	CA	Chain Address - When set, TDES3 is interpreted as another descriptor's VAX physical address. This allows the SGEC to process multiple, non-contiguous descriptor lists and explicitly "chain" the lists. Note that contiguous descriptors are implicitly chained. In contrast to what is done for a Rx buffer descriptor, the SGEC clears neither the ownership bit TDES0<OW> or one of the other bits of TDES0 of the chain descriptor after processing. To protect against infinite loop, a chain descriptor pointing back to itself, is seen as **owned by the host**, regardless of the ownership bit state.
<30>	VA	Virtual Addressing - When set, TDES3 is interpreted as a virtual address. The type of virtual address translation is determined by the TDES1<VT> bit. The SGEC uses TDES3 and TDES2<Page Offset> to perform a VAX virtual address translation process to obtain the physical address of the buffer. When clear, TDES3 is interpreted as the actual physical address of the buffer:

VA	VT	Addressing mode
0	x	Physical
1	0	Virtual - SVAPTE
1	1	Virtual - PAPTE

Bit	Name	Descriptor
<29:28>	DT	Data Type - Indicates the type of data the buffer contains, according to the following table:

Value	Meaning
00	Normal transmit frame data
10	Setup frame - Explained in Section 10.3.18.
11	Diagnostic frame

Bit	Name	Descriptor
<27>	AC	Add CRC disable - When set, the SGEC will not append the CRC to the end of the transmitted frame. To take effect, this bit must be set in the descriptor where FS is set. **NOTE** If the transmitted frame is shorter than 64 bytes, the SGEC will add the padding field and the CRC regardless the <AC> flag.
<26>	FS	First Segment - When set, indicates the buffer contains the first segment of a frame.
<25>	LS	Last Segment - When set, indicates the buffer contains the last segment of a frame.

Table 10–34 (Cont.): TDES1 bits

Bit	Name	Descriptor
<24>	IC	Interrupt on Completion - When set, the SGEC will set NICSR5<TI> after this frame has been transmitted. To take effect, this bit must be set in the descriptor where LS is set.
<23>	VT	Virtual Type - In case of virtual addressing (TDES1<VA> = 1), indicates the type of virtual address translation. When set, the buffer address TDES3 is interpreted as a SVAPTE (System Virtual Address of the Page Table Entry). When clear, the buffer address is interpreted as a PAPTE (Physical Address of the Page Table Entry). Meaningful only if TDES1<VA> is set.
<22:0>	u	Unused. Ignored by the SGEC on reads. Never written.

10.3.17.3 TDES2 word

Table 10–35: TDES2 bits

Bit	Name	Descriptor
<31>	u	Unused. Ignored by the SGEC on reads. Never written.
<30:16>	BS	Buffer Size - The size, in bytes, of the data buffer. If this field is 0, the SGEC will skip over this buffer and ignore it. The frame size is the sum of all BS fields of the frame segments (between and including the descriptors having TDES1<FS> and TDES1<LS> set.) **NOTE** If the port driver wishes to suppress transmission of a frame, this field must be set to 0 in all descriptors comprising the frame and prior to the SGEC acquiring them. If this rule is not adhered to, corrupted frames might be transmitted.
<08:00>	PO	Page Offset - The byte offset of the buffer within the page. Only meaningful if TDES1<VA> is set. **NOTE** Transmit buffers may start on arbitrary byte boundaries.

10.3.17.4 TDES3 word

Table 10–36: TDES3 bits

Bit	Name	Descriptor
<31:00>	SV/PV/PA	SVAPTE/PAPTE/Physical Address - When TDES1<VA> is set, TDES3 is interpreted as the address of the Page Table Entry and used in the virtual address translation process. The type of the address System Virtual address (SVAPTE) or Physical Address (PAPTE) is determined by TDES1<VT>. When TDES1<VA> is clear, TDES3 is interpreted as the physical address of the buffer. When TDES1<CA> is set, TDES3 is interpreted as the VAX physical address of another descriptor. **NOTE** Transmit buffers may start on arbitrary byte boundaries.

10.3.17.5 Transmit descriptor status validity

Table 10–37 summarizes the validity of the Transmit descriptor status bits regarding the Transmission completion status:

Table 10–37: Transmit descriptor status validity

Transmission	Tx Status report						
status	LO	NC	LC	EC	HF	CC	(ES,TO,LE,TN,UF,DE)
Underflow	X	X	V	V	X	V	V
Excessive collisions	V	V	V	V	V	X	V
Watchdog timeout	X	V	X	X	X	V	V
Internal Loopback	X	X	V	V	X	V	V

V - Valid
X - Meaningless

10.3.18 Setup frame

A setup frame defines SGEC Ethernet destination addresses. These addresses will be used to filter all incoming frames. The setup frame is *never* transmitted over the Ethernet, nor looped back to the receive list. While the setup frame is being processed, the receiver logic will temporarily disengage from the Ethernet wire. The setup frame size is *always* 128 bytes and must be wholly contained in a single transmit buffer. There are two types of setup frames:

1. Perfect Filtering addresses (16) list

2. Imperfect Filtering hash bucket (512) heads + one physical address.

10.3.18.1 First setup frame

A setup frame must be *queued* (**placed in the transmit list with SGEC ownership**) to the SGEC before the Reception process is started, except for when the SGEC operates in promiscuous reception mode.

NOTE

The self test completes with the SGEC Address filtering table fully set to "0". A Reception process started without loading a setup frame will reject all the incoming frames except those with a destination physical address = 000000h .

10.3.18.2 Subsequent setup frame

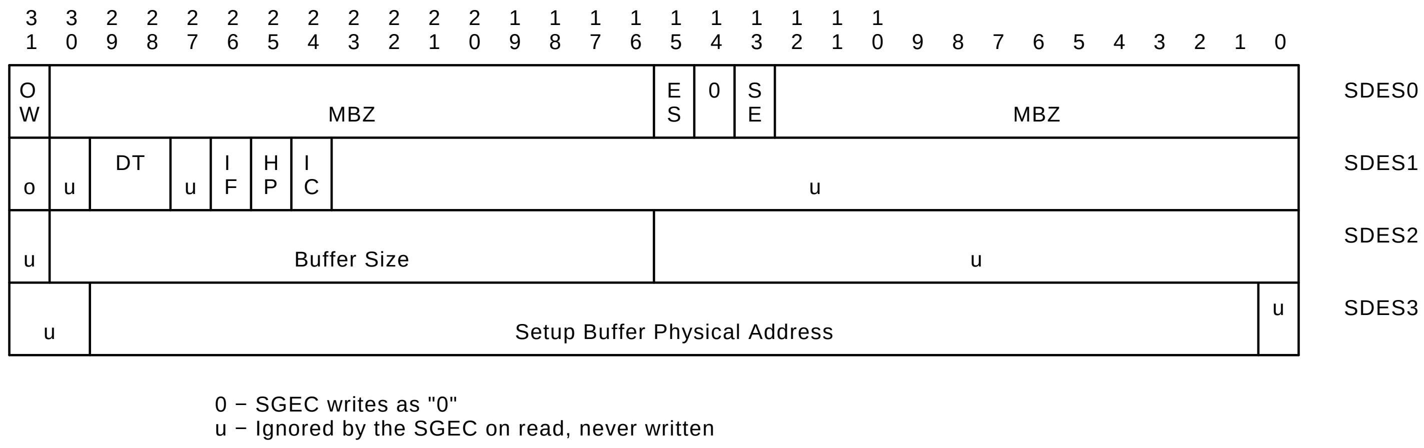
Subsequent setup frames may be queued to the SGEC regardless of the Reception process state. The only requirement for the setup frame to be processed, is that the Transmission process be in the RUNNING state. The setup frame will be processed after all preceding frames have been transmitted and after the current frame reception, if any, is completed.

The setup frame does not affect the Reception process state but during the setup frame processing, the SGEC is disengaged from the Ethernet wire.

10.3.18.3 Setup frame descriptor

The setup frame descriptor format is shown in Figure 10–16, and described in the following paragraphs.

Figure 10–16: Setup frame descriptor format



ESB90P0066

Table 10–38: Setup frame descriptor bits

Word	Bit	Name	Description
SDES0	<13>	SE	Setup Error - When set, indicates the setup frame buffer size in not 128 bytes.
	<15>	ES	Error Summary - Set when SE is set.
	<31>	OW	Own bit - When set, indicates the descriptor is owned by the SGEC. When cleared, indicates the descriptor is owned by the host. The SGEC clears this bit upon completing processing of the descriptor and its associated buffer.
SDES1	<24>	IC	Interrupt on Completion - When set, the SGEC will set NICSR5<TI> after this setup frame has been processed.

Table 10–38 (Cont.): Setup frame descriptor bits

Word	Bit	Name	Description
	<25>	HP	Hash/Perfect filtering mode - When set, the SGEC will interpret the setup frame as a hash table, and do an imperfect address filtering. The imperfect mode is useful when there are more than 16 Multicast addresses to listen to. When clear, the SGEC will do a perfect address filter of incoming frames according to the addresses specified in the setup frame.
	<26>	IF	Inverse filtering - When set, the SGEC will do an inverse filtering: the SGEC will receive the incoming frames with destination address not matching the perfect addresses and will reject the frames with destination address matching one of the perfect addresses. Meaningful only for Perfect_filtering (SDES1<HP>=0), while Promiscuous and All_Multicast modes are not selected (NICSR6<AF>=0).
	<29:28>	DT	Data Type - Must be 2 to indicate setup frame.
SDES2	<30:16>	BS	Buffer Size - Must be 128.
SDES3	<29:1>	PA	Physical Address - Physical address of setup buffer.

NOTE

Setup buffer must be word aligned.

10.3.18.4 Perfect Filtering setup frame buffer

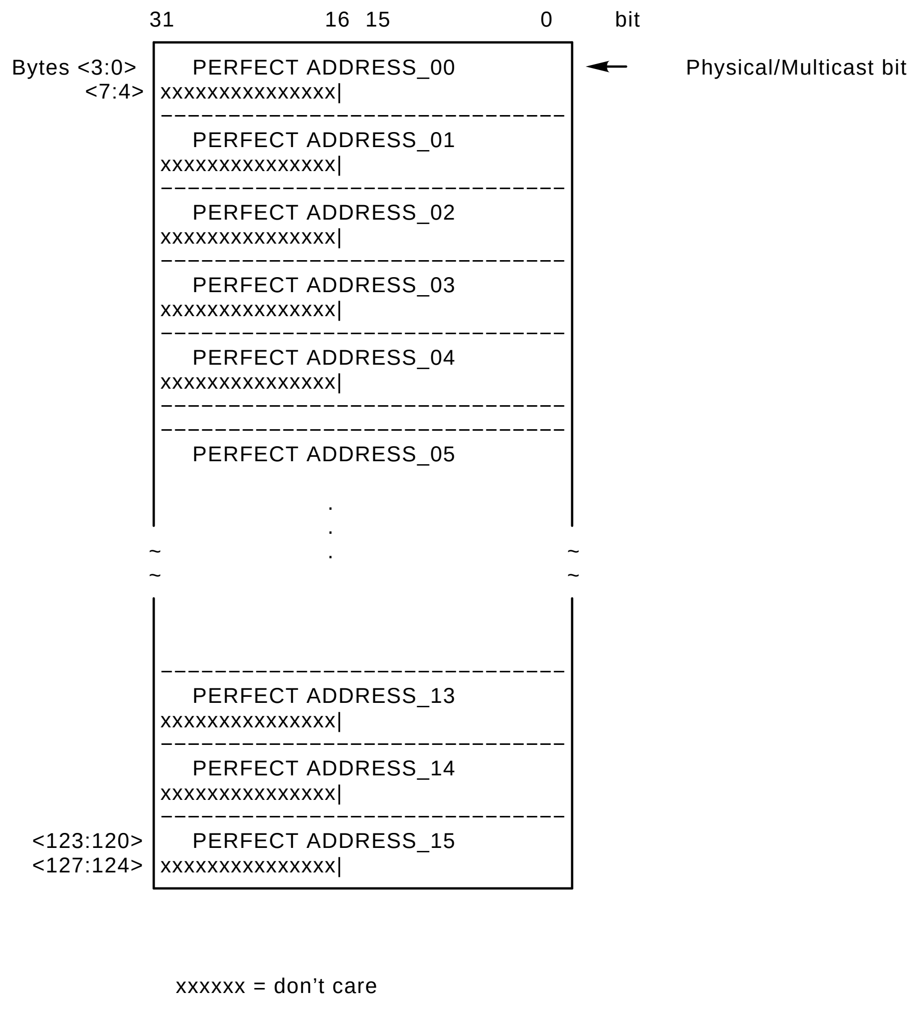
This section describes how the SGEC interprets a setup frame buffer when SDES1<HP> is clear.

The SGEC can store 16 - full 48 bits Ethernet - destination addresses. It will compare the addresses of any incoming frame to these, and regarding the status of Inverse_Filtering flag SDES1<IF>, will reject the following:

- Those which do not match, if SDES1<IF> = 0
- Those which match, if SDES1<IF> = 1

The setup frame *must always* supply all 16 addresses. Any mix of physical and Multicast addresses can be used. Unused addresses should be duplicates of one of the valid addresses. The addresses are formatted as shown in Figure 10–17.

Figure 10–17: Perfect Filtering setup frame buffer format



ESB90P0067

The low-order bit of the low-order bytes is the address's Multicast bit.

Example 10–1 illustrates a Perfect Filtering Setup buffer (fragment).

Example 10–1: Perfect filtering buffer

```
  Ethernet addresses to be filtered:
❶   A8-09-65-12-34-76
    09-BC-87-DE-03-15
      .
      .
      .

  Setup frame buffer fragment:
❷   126509A8
    00007634
    DE87BC09
    00001503
      .
      .
      .
```

❶ Two Ethernet addresses written according to the DEC STD 134 specification for address display.

❷ Those two addresses as they would appear in the buffer.

10.3.18.5 Imperfect Filtering setup frame buffer

This section describes how the SGEC interprets a setup frame buffer when SDES1<HP> is set.

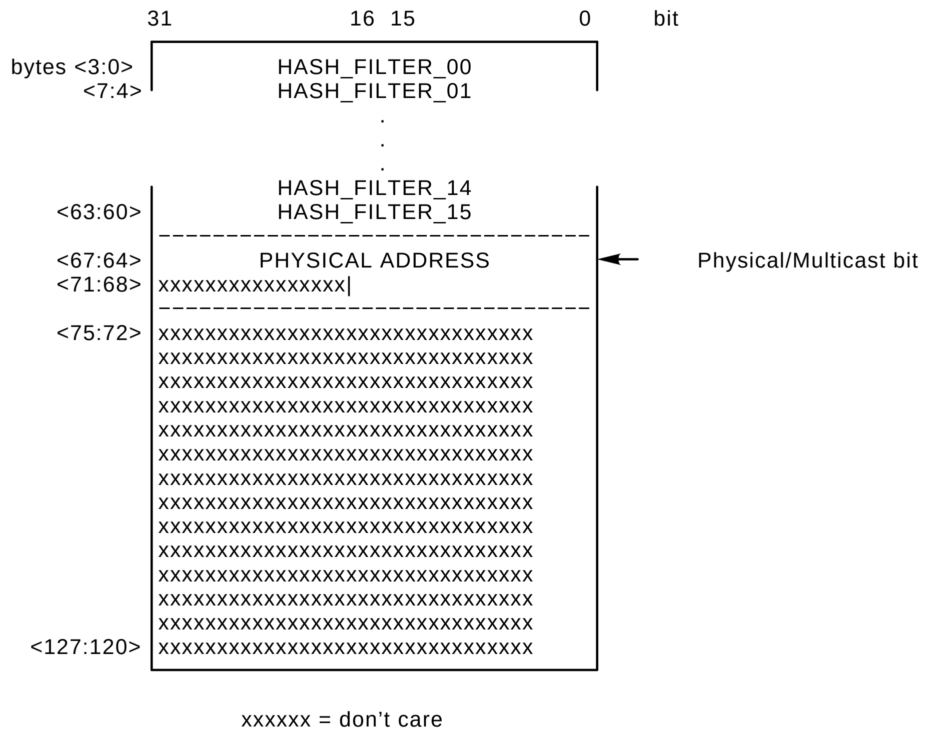
The SGEC can store 512 bits, serving as hash bucket heads, and one *physical* 48 bit Ethernet address. Incoming frames with Multicast destination addresses will be subjected to the imperfect filtering. Frames with physical destination addresses will be checked against the single physical address.

For any incoming frame with a **Multicast** destination address, the SGEC applies the standard Ethernet CRC function to the first six bytes containing the destination address, then uses the most significant nine bits of the result, as a bit index into the table. If the indexed bit is set, the frame is accepted. If it is cleared, the frame is rejected.

This filtering mode is called imperfect, because Multicast frames not addressed to this station may slip through, but it will still cut down the number of frames the host will be presented with.

The format for the hash table and the physical address is shown in Figure 10–18.

Figure 10–18: Imperfect Filtering setup frame format



ESB90P0068

Bits are sequentially numbered from right to left and down the table. For example, if CRC(destination address)<8:0> = 33, the SGEC will examine bit #1 in the second longword.

Example 10–2 illustrates an Imperfect Filtering Setup frame buffer.

Example 10–2: Imperfect filtering buffer

```
  Ethernet addresses to be filtered:
❶   25-00-25-00-27-00
    A3-C5-62-3F-25-87
    D9-C2-C0-99-0B-82
    7D-48-4D-FD-CC-0A
    E7-C1-96-36-89-DD
    61-CC-28-55-D3-C7
    6B-46-0A-55-2D-7E

❷   A8-12-34-35-76-08

  Setup frame buffer:
❸   00000000
    10000000
    00000000
    00000000
    00000000
    40000000
    00000080
    00100000
    00000000
    10000000
    00000000
    00000000
    00000000
    00010000
    00000000
    00400000
❹   353412A8
    00000876
```

❶ Ethernet Multicast addresses written according to the DEC STD 134 specification for address display.

❷ An Ethernet physical address.

❸ The first part of an Imperfect filter Setup frame buffer with set bits for the ❶ Multicast addresses.

❹ The second part of the buffer with the ❷ physical address.

Example 10–3 shows a C program to compute the hash bucket heads and create the resultant Setup frame buffer.

Example 10–3: Imperfect filtering Setup frame buffer creation C program

```
#include <stdio>

unsigned int imperfect_setup_frame[128/4],   /* The setup buffer - 128   */
      /* bytes      */
  address[2],
  crc[33];   /* CRC residue vector     */

main()
{
    int i, hash;
/*             */
/* This program accepts 48 bits Ethernet addresses and builds a Setup frame */
/* buffer for imperfect filtering.         */
/*             */
/* Addresses must be entered in hexadecimal. The multicast bit is the least */
/* significant bit of the least significant digit of the first 32 bits.     */
/* Non-multicast addresses are ignored.         */
/*              */
/* Input is terminated by keying CTRL/Z after which the program prints out  */
/* the buffer.            */
/*             */
main_loop:

/* Prompt user for the Ethernet address         */
    printf("\n\n Enter the first 32 bits (HEX) - ");
    if (scanf("%x", &address[0]) == EOF)
 {

     printf("\n\n Imperfect Setup buffer printout\n");
     for (i=0;  i < 128/4;  i++)
  printf("%08X\n", imperfect_setup_frame[i]);
     exit(1);
 }
    printf("\n Enter the remaining 16 bits (HEX) - ");
    scanf("%x",&address[1]);

/* Ignore non multicast addresses         */
     if ((address[0] & 1) == 0)
  goto main_loop;

/* Compute the hash function          */
    hash = address_crc(address[0],address[1]);

/* Set the appropriate bit in the Setup buffer        */
    imperfect_setup_frame[hash/32] =
 imperfect_setup_frame[hash/32] | 1 << hash%32;

    goto main_loop;
}
```

Example 10–3 (continued on next page)

Example 10–3 (Cont.): Imperfect filtering Setup frame buffer creation C program

```
int address_crc( unsigned int lsb32 , unsigned int msb16)
{
    int j,hash = 0;

/* Set CRC to all 1's           */

    for (j=0;  j < 33;  j++)
 crc[j] = 1;

/* Compute the address CRC by running the CRC 48 steps       */

    for (j=0;  j < 32;  j++)
 nextstate(lsb32 & 1<<j ? 1 : 0);
    for (j=0;  j < 16;  j++)
 nextstate(msb16 & 1<<j ? 1 : 0);

/* Extract 9 most significant bits from the CRC residue      */

    for (j=24;  j < 33;  j++)
 hash = hash<<1 | crc[j];

    return hash;
}


nextstate(dat)
int dat;
{
  int i,mean;
  mean = crc[32] ^ dat;
  for(i=32;i>=2;i--) crc[i]=crc[i-1];
  crc[27] = crc[27] ^ mean;
  crc[24] = crc[24] ^ mean;
  crc[23] = crc[23] ^ mean;
  crc[17] = crc[17] ^ mean;
  crc[13] = crc[13] ^ mean;
  crc[12] = crc[12] ^ mean;
  crc[11] = crc[11] ^ mean;
  crc[9] = crc[9] ^ mean;
  crc[8] = crc[8] ^ mean;
  crc[6] = crc[6] ^ mean;
  crc[5] = crc[5] ^ mean;
  crc[3] = crc[3] ^ mean;
  crc[2] = crc[2] ^ mean;
  crc[1] = mean;
```

10.3.19 SGEC operation

10.3.19.1 Hardware and Software Reset

The SGEC responds to two types of reset commands: a hardware reset through the RESET_L pin, and a software reset command triggered by setting NICSR6<RE>. In **both cases**, the SGEC **aborts** all ongoing processing and starts the Reset sequence. The SGEC restarts and reinitializes all internal states and registers. *No internal states are retained, no descriptors are owned and all the host visible registers are set to "0", except where otherwise noted.*

NOTE

The SGEC does not explicitly disown any owned descriptor; so descriptors OWNED bits might be left in a state indicating SGEC ownership.

Table 10–39 indicates the NICSR fields which are not set to “0” after reset:

Table 10–39:

Field	Value
NICSR3	Unpredictable
NICSR4	Unpredictable
NICSR5<DN>	1
NICSR6<BL>	1
NICSR6<RE>	Unpredictable after HARDWARE reset
	1 after SOFTWARE reset
NICSR7	Unpredictable
NICSR9	RT = TT = 1250

After the reset sequence completes, the SGEC executes the self test procedure to do basic checking.

If the self test completes successfully, the SGEC initializes the SGEC, and sets the Initialization Done flag NICSR5<ID>.

At the first failure detected in one of the basic tests executed in the self_test routine, the test is aborted and the self_test failure NICSR5<SF> is set together with the self_test error status NICSR5<SS>which indicates the failure reason.

INFO

The self test takes 25ms to complete after Hardware or Software RESET.

If the initialization completes successfully, the SGEC is ready to accept further host commands. Both the Reception and Transmission processes are placed in the STOPPED state.

Successive reset commands (either hardware or software) may be issued. The only restriction is that SGEC NICSRs should not be accessed during a 1µsecond period following the reset. Access during this period will result in a CP-BUS timeout error. Access to SGEC NICSRs during the self test are permitted; however, only NICSR5 reads should be performed.

10.3.19.2 Interrupts

Interrupts are generated as a result of various events. NICSR5 contains all the status bits which may cause an interrupt, provided NICSR6<IE> is set. The port driver must clear the interrupt bits (by writing a “1” to the bit position), to enable further interrupts from the same source.

Interrupts are *not queued*, and if the interrupting event reoccurs *before* the port driver has responded to it, no additional interrupts will be generated. For example, NICSR5<RI> indicates *one or more* frames were delivered to host memory. The port driver should scan *all* descriptors, from its last recorded position up to the first SGEC owned one.

An interrupt will only be generated *once* for simultaneous, multiple interrupting events. It is the port driver responsibility to scan NICSR5 for the interrupt cause(s). The interrupt will not be *regenerated*, unless a *new* interrupting event occurs *after* the host acknowledged the previous one, and provided the port driver *cleared* the appropriate NICSR5 bit(s). For example, NICSR5<TI> and NICSR5<RI> may both set, the host acknowledges the interrupt and the port driver begins executing by reading NICSR5. Now NICSR5<RU> sets. The port driver writes back its copy of NICSR5, clearing NICSR5<TI> and NICSR5<RI>. After the host IPL is lowered below the SGEC level, another interrupt will be delivered with the NICSR5<RU> bit set.

Should the port driver clear *all* NICSR5 set interrupt bits before the interrupt has been acknowledged, the interrupt will be suppressed.

10.3.19.3 Startup procedure

A sequence of checks and commands must be performed by the port driver to prepare the SGEC for operation.

1. Wait for the SGEC to complete its Initialization sequence by polling on NICSR5<ID> and NICSR5<SF> (refer to Section 10.3.7 for details).
2. Examine NICSR5<SF> to find out whether the SGEC passed its self test. If it did not, it should be replaced (refer to Section 10.3.7 for details).
3. Write NICSR0 to establish system configuration dependent parameters (refer to Section 10.3.3 for details).
4. If the port driver intends to use VAX virtual addresses, NICSR7 must be written to identify the System Page Table to the SGEC (refer to Section 10.3.9 for details).
5. If the port driver wishes to change the default settings of the watchdog timers, it must write to NICSR9 (refer to Section 10.3.11 for details).
6. The port driver must create the transmit and receive descriptor lists, then write to NICSR3 and NICSR4 to provide the SGEC with the starting address of each list. The first descriptor on the transmit list will usually contain a setup frame (refer to Section 10.3.6 for details).
7. Write NICSR6 to set global operating parameters and start the Transmission and Reception processes. The Reception and Transmission processes enter the RUNNING state and attempt to acquire descriptors from the respective descriptor lists and begin processing incoming and outgoing frames (refer to Section 10.3.8 for details). The Reception and Transmission processes are independent of each other and can be started and stopped separately.

CAUTION

If address filtering (either perfect or imperfect) is desired, the Reception process should only be started after the Setup frame has been processed.

8. The port driver now waits for any SGEC interrupts. If either the Reception or Transmission processes were SUSPENDED, the port driver must issue the Poll Demand command after it has rectified the suspension cause.

10.3.19.4 Reception process

While in the RUNNING state, the Reception process polls the receive descriptor list, attempting to acquire free descriptors. Incoming frames are processed and placed in acquired descriptors' data buffers, while status information is written to the descriptor RDES0 words. The SGEC always tries to acquire an extra descriptor in anticipation of incoming frames. Descriptor acquisition is attempted under the following conditions:

- Immediately after being placed in the RUNNING state through setting of NICSR6<SR>.
- The SGEC begins writing frame data to a data buffer pointed to by the current descriptor.
- The last acquired descriptor chained (RDES1<CA> = 1) to another descriptor.
- A virtual translation error was encountered RDES0<TN> while the SGEC was translating the buffer base address of the acquired descriptor .

As incoming frames arrive, the SGEC strips the preamble bits and stores the frame data in the receive FIFO. Concurrently, it performs address filtering according to NICSR6 fields AF, HP and its internal filtering table. If the frame fails the address filtering, it is ignored and purged from the FIFO. Frames which are shorter than 64 bytes, due to collision or premature termination are also ignored and purged from the FIFO, unless NICSR6<PB> is set.

After 64 bytes have been received, the SGEC begins transferring the frame data to the buffer pointed to by the current descriptor. If Data Chaining is enabled (NICSR6<DC> clear), the SGEC will write frame data overflowing the current data buffer into successive buffer(s). The SGEC sets the RDES0<FS> and RDES0<LS> in the first and last descriptors, respectively, to delimit the frame. Descriptors are released (RDES0<OW> bit cleared) as their data buffers fill up or the last segment of a frame has been transferred to a buffer.

The SGEC sets RDES0<LS> and the RDES0 status bits in the last descriptor it releases for a frame. After the last descriptor of a frame is released, the SGEC sets NICSR5<RI>.

This process is repeated until the SGEC encounters a descriptor flagged as owned by the host. After filling up all previously acquired buffers, the Reception sets NICSR5<RU> and enters the SUSPENDED state. The position in the receive list is retained.

Any incoming frames while in this state will cause the SGEC to fetch the current descriptor in the host memory. If the descriptor is now owned by the SGEC, the Reception re-enters the RUNNING state and starts the frame reception.

If the descriptor is still owned by the host, the SGEC increments the Missed Frames Counter (NICSR10<MFC>) and discards the frame.

Table 10–40 summarizes the Reception process state transitions and resulting actions:

Table 10–40: Reception process state transitions

From state	Event	To state	Action
STOPPED	Start Reception command	RUNNING	Receive polling begins from last list position or from the the list head if this is the first Start command issued, or if the receive descriptor list address (NICSR3) was modified by the port driver.
RUNNING	SGEC attempts acquisition of a descriptor owned by the host	SUSPENDED	NICSR5<RU> is set when the last acquired descriptor buffer is consumed. Position in list retained.
RUNNING	Stop Reception command	STOPPED	Reception process is STOPPED after the current frame, if any, is completely transferred to data buffer(s). Position in list retained.
RUNNING	Memory or host bus parity error encountered	STOPPED	Reception is cut off and NICSR5<ME> is set.
RUNNING	Reset command	STOPPED	Reception is cut off.
SUSPENDED	Rx Poll demand or Incoming frame and available descriptor	RUNNING	Receive polling resumes from last list position or from the list head if NICSR3 was modified by the port driver.
SUSPENDED	Stop Reception command	STOPPED	None.
SUSPENDED	Reset command	STOPPED	None.

10.3.19.5 Transmission process

While in the RUNNING state, the Transmission process polls the transmit descriptor list for any frames to transmit. Frames are built and transmitted on the Ethernet wire. Upon completing frame transmission (or giving up), status information is written to the TDES0 words. Once polling starts, it continues (in sequential or descriptor chained order) until the SGEC encounters a descriptor flagged as owned by the host, or an error condition. At this point, the Transmission process is placed in the SUSPENDED state and NICSR5<TI> is set.

NICSR5<TI> will also be set after completing transmission of a frame which has TDES1<IC> set in its last descriptor. In this case, the Transmission process remains in the RUNNING state.

Frames may be data chained and span several buffers. Frames must be delimited by TDES1<FS> and TDES1<LS> in the first and last descriptors, respectively, containing the frame. While in the RUNNING state, as the Transmission process starts, it first expects a descriptor with TDES1<FS> set. Frame data transfer from the host buffer to the internal FIFO is initiated. Concurrently, if the current frame had TDES1<LS> clear, the Transmission process attempts to acquire the next descriptor, expecting TDES1<FS> and TDES1<LS> to be clear indicating an intermediary buffer, or TDES1<LS> to be set, indicating the end of the frame. After the last buffer of the frame has been transmitted, the SGEC writes back final status information to the TDES0 word of the descriptor having TDES1<LS> set, optionally sets NICSR5<TI> if TDES1<IC> was set, and repeats the process with the next descriptor(s). Actual frame transmission begins *after at least 72 bytes* have been transferred to the internal FIFO, or *a full frame is contained* in the FIFO. Descriptors are released (TDES0<OW> bit cleared) as soon as the SGEC is through processing a descriptor.

Transmit polling suspends under the following conditions:

- the SGEC reaches a descriptor with TDES0<OW> clear. To resume, the port driver must give descriptor ownership to the SGEC and issue a Poll Demand command.
- The TDES1<FS> and TDES1<LS> are incorrectly paired or out of order. TDES0<LE> will be set.
- A frame transmission is given up due to a locally induced error. The appropriate TDES0 bit is set.

The Transmission process enters the SUSPENDED state and sets NICSR5<TI>. Status information is written to the TDES0 word of the descriptor causing the suspension. The position in the transmit list, in all of the above cases, is retained. The retained position is that of the *descriptor following the last descriptor closed (set to host ownership) by the SGEC*.

NOTE

The SGEC *does not* automatically poll the Tx descriptor list and the port driver *must* explicitly issue a Tx Poll Demand command after rectifying the suspension cause.

The following table summarizes the Transmission process state transitions:

Table 10–41: Transmission process state transitions

From state	Event	To state	Action
STOPPED	Start Transmission command	RUNNING	Transmit polling begins from the last list position or from the head of the list if this is the first Start command issued, or if the transmit descriptor list address (NICSR4) was modified by the port driver.
RUNNING	SGEC attempts acquisition of a descriptor owned by the host	SUSPENDED	NICSR5<TI> is set. Position in list retained.
RUNNING	Out of order delimiting flag (TDES0<FS> or TDES0<LS>) encountered.	SUSPENDED	TDES0<LE> and NICSR5<TI> are set. Position in list retained.

Table 10–41 (Cont.): Transmission process state transitions

From state	Event	To state	Action
RUNNING	Frame transmission aborts due to a locally induced error.	SUSPENDED	Appropriate TDES0 and NICSR5<TI> bits are set. Position in list retained.
RUNNING	Stop Transmission command	STOPPED	Transmission process is STOPPED after the current frame, if any, is transmitted. Position in list retained.
RUNNING	Transmit watchdog expires	STOPPED	Transmission is cut off and NICSR5<TW> , TDES0<TO> are set. Position in list retained.
RUNNING	Memory or host bus parity error encountered	STOPPED	Transmission is cut off and NICSR5<ME> is set.
RUNNING	Reset command	STOPPED	Transmission is cut off.
SUSPENDED	Tx Poll Demand command	RUNNING	Transmit polling resumes from last list position or from the list head if NICSR4 was modified by the port driver.
SUSPENDED	Stop Transmission command	STOPPED	None.
SUSPENDED	Reset command	STOPPED	None.

10.3.19.6 Loopback operations

The SGEC supports two loopback modes:

- Internal loopback

  This mode is generally used to verify correct operations of the SGEC internal logic. While in this mode, the SGEC will take frames from the transmit list and loop them back, internally, to the receive list. The SGEC is disengaged from the Ethernet wire while in this mode.
- External loopback

  This mode is generally used to verify correct operations up to the Ethernet cable. While in this mode, the SGEC will take frames from the transmit list and transmit them on the Ethernet wire. Concurrently, the SGEC listens to the line which carries its own transmissions and places incoming frames in the receive list.

NOTE

Caution should be exercised in this mode as transmitted frames are placed on the Ethernet wire. Furthermore, the SGEC does not check the origin of any incoming frames, consequently , frames not necessarily originating from the SGEC might make it to the receive buffers.

In either of these modes, all the address filtering and validity checking rules apply. The port driver needs to take the following actions:

1. Place the Reception and Transmission processes in the STOPPED state. The port driver must wait for any previously scheduled frame activity to cease. This is done by polling the TS and RS fields in NICSR5.
2. Prepare appropriate transmit and receive descriptor lists in host memory. These may follow the existing lists at the point of suspension, or may be new lists which will have to be identified to the SGEC by appropriately writing NICSR3 and NICSR4.
3. Write to NICSR6<OM> according to the desired loopback mode and place the Transmission and Reception processes in the RUNNING state through Start commands.
4. Respond and process any SGEC interrupts, as in normal processing.

To restore normal operations, the port driver must execute above step #1, then write the OM field in NICSR6 with “00”.

10.3.19.7 DNA CSMA/CD counters and events support

This section describes the SGEC features that support the port driver in implementing and reporting the specified counters and events [4].

Table 10–42: CSMA/CD counters

Counter	SGEC feature
Time since counter creation	Supported by the host driver.
Bytes received	Port driver must add up the RDES0<FL> fields of all successfully received frames.
Bytes sent	Port driver must add up the TDES2<BS> fields of all successfully transmitted buffers.
Frames received	Port driver must count the successfully received frames in the receive descriptor list.
Frames sent	Port driver must count the successfully transmitted frames in the transmit descriptor list.
Multicast bytes received	Port driver must add up the RDES0<FL> fields of all successfully received frames with Multicast address destinations.
Multicast frames received	Port driver must count the successfully received frames with Multicast address destinations.
Frames sent, initially deferred	Port driver must count the successfully transmitted frames with TDES0<DE> set.

[4] As specified in the DNA Maintenance Operations (MOP) Functional Specification, Version T.4.0.0, 28 January 1988.

Table 10–42 (Cont.): CSMA/CD counters

Counter	SGEC feature
Frames sent, single collision	Port driver must count the successfully transmitted frames with TDES0<CC> equal to 1.
Frames sent, multiple collisions	Port driver must count the successfully transmitted frames with TDES0<CC> greater than 1.
Send failure- Excessive collisions	Port driver must count the transmit descriptors having TDES0<EC> set.
Send failure- Carrier check failed	Port driver must count the transmit descriptors having TDES0<LC> set.
Send failure- Short circuit	Two successive transmit descriptors with the No_carrier flag TDES0<NC> set, indicates a short circuit.
Send failure- Open circuit	Two successive transmit descriptors with the Excessive_ collisions flag TDES0<EC> set with the same Time domain reflectometer value TDES0<TDR>, indicate an open circuit.
Send failure- Remote Failure to Defer	Flagged as a late collision TDES0<LC> in the transmit descriptors.
Receive failure- Block Check Error	Port driver must count the receive descriptors having RDES0<CE> set with RDES0<DB> cleared.
Receive failure- Framing Error	Port driver must count the receive descriptors having both RDES0<CE> and RDES0<DB> set.
Receive failure- Frame too long	Port driver must count the receive descriptors having RDES0<TL> set.
Unrecognized frame destination	Not applicable.
Data overrun	Port driver must count the receive descriptors having RDES0<OF> set.
System buffer unavailable	Reported in the Missed_frame counter NICSR10<MFC> (refer to Table 10–20).
User buffer unavailable	Not applicable.
Collision detect check failed	Port driver must count the transmit descriptors having TDES0<HF> set.

CSMA/CD specified events can be reported by the port driver based on the above table. The Initialization Failed event is reported through NICSR5<SF>.

Chapter 11

KA660 Mass Storage Interface

The KA660 contains a DSSI Bus Interface which is implemented with the Single Host Adapter Chip (SHAC). This interface allows the KA660 to transmit packets of data to, and receive packets of data from, up to seven other DSSI devices (typically RF type disk drives and TF type streaming tape drives). It should also be noted that the SHAC supports CP Bus Parity Protection.

11.1 SHAC Introduction

SHAC (Single Host Adapter Chip) is a single-chip, VLSI version of an SCA port that uses a DSSI bus as the physical interconnect. Another SCA realization, CI, has defined a port-driver /port interface which has been used to connect VAXs in clusters. DSSI has adopted the same interface, so the same VMS port driver will be able to drive either a CI-port or SHAC. The SHAC can be used to connect a host to any other device that can communicate through the CI-DSSI protocol. In particular, it provides a solution to the following:

- The problem of interfacing a group of mass-storage device controllers (MSDCs) to a VAX.
- The problem of interfacing several VAXs to a common group of MSDCs and, if higher level protocols support this option, to one another.

Where two or more VAXs connect to a group of MSDCs (or to one another) through DSSI, each has a SHAC or another DSSI port. When a group of MSDCs connect to the DSSI bus, the controllers provide both the bus interface and the intelligent control required to respond to the CI commands received over the DSSI.

On the 1-byte wide DSSI bus both the MSDCs and the several VAXs communicate at high-speed, with a 4 to 5 MB/s burst transfer rate. The SHAC handles the problem of providing effective, efficient and reliable interfacing between this DSSI bus and the SOC CPU , having direct host memory access (DMA) over the host's 32-bit wide, 16 MB/s CP bus. All communications between those connected to the DSSI will follow the CI protocol with the DSSI protocols providing handshaking in the transactions.

Structural parameters limit the number of possible combinations that can be realized with DSSI and SHAC.

- A single DSSI bus has room for 8 nodes which may be partitioned among host adapters (for example, SHACs) and MSDCs.

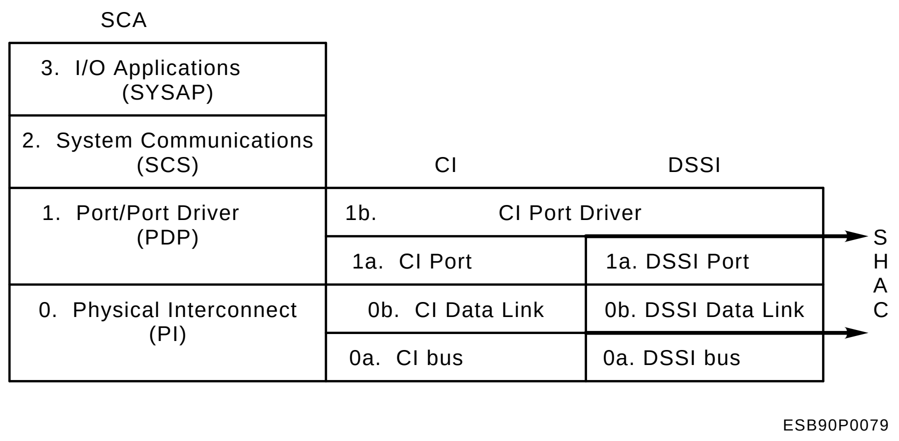
- Up to 4 SHACs can be installed on a single host bus.
- Because there must be a host, there can be up to 7 MSDCs on a single DSSI.

The SHAC provides a small amount of buffering (1.2KB) on chip to improve bus utilization on both sides, but the SHAC is designed to pass data through from one bus to the other as rapidly as the two busses permit. DMA services to and from the main memory reside in the SHAC, which responds to requests for transfers between the host and the remote nodes.

The SHAC is operated by an on-chip RISC that obtains its code and internal data from on-chip RAM and ROM. The RAM will be loaded from main memory both during initialization and as circumstances require during normal run time. With this capability, it can read in new code and data from the main memory and thus adapt its behavior to new circumstances. This will permit inexpensive upgrades of SHACs after they are installed in the field. Furthermore, it will allow the SHAC to store infrequently accessed code in main memory, providing more capability than could be included in on-chip ROM.

The overall communication architecture under which the SHAC works is Digital's Systems Communications Architecture (SCA). In this general architecture, four layers are defined, as shown in Figure 11–1. The architecture can be realized in a variety of ways. Two particularizations of the lowest two levels in the diagram are CI (Computer Interconnect) and DSSI (Digital Storage System Interconnect). They share the same lowest host layer (CI Port driver) but have distinctly different physical interconnects. The layers between the Port Driver and the DSSI bus itself can be realized at both board and chip level and products at both levels are in design within Digital. The SHAC is a chip-level product which connects the host-bus to the DSSI bus, controlled by the SOC CPU through a CI port driver and accepts and delivers CI-defined packets over the DSSI bus. Layers above the port driver are invisible to SHAC.

Figure 11–1: Relationship of the DSSI to SCA and CI



The port driver maintains a set of 7 queues in its system space. Four of these contain commands for the SHAC to execute. The priority of the command is determined by the queue it is on; order is determined by the position in the queue. Another queue contains all of the responses for the host (from the SHAC or the remote nodes). Finally, there are two queues of "empty envelopes" for the host and the SHAC to use to stuff with commands and responses and then to queue them on the other queues.

These "envelopes" are simply standard-sized "queuable" blocks of host memory. All commands and responses are copied into one of these standard-sized blocks. Included in the header on each block are a pair of queue pointers (for a doubly linked queue) and various standard identifiers which specify what is contained in the block and how much of the block represents the actual command or response. To be visible, a block must be on a queue, where pointers from other elements or the queue header show its presence. Once a block is removed from a queue, it is visible only to the entity which removed it.

The SHAC's principal task is in accepting and delivering "mail" to other nodes. Externally (For example, on DSSI) the SHAC deals only in standard CI formats. Internally, the SHAC deals with the envelopes just described and with blocks of data. Because DSSI deals with bytes and the CP bus deals in longwords, the SHAC must frequently do byte alignment tasks during transcription.

The SHAC deals with the port driver in the virtual-address mode, unloading from the SOC CPU the obligation to do virtual-to-physical address translation and to be aware of page crossings in virtually-contiguous blocks of information. The SHAC supports full virtual address translation including the use of global I/O pages (to a depth of 1).

The rest of this SHAC overview section describes a typical set of steps that the SHAC goes through in serving its role as the CI Port, with "mail" in both directions.

11.2 CI-DSSI Overview

At start-up, the host provides the SHAC with a number of pointers to internal host structures, one of which, the Port Queue Block (PQB), contains pointers and data on all of the queues that the host maintains for CI. The SHAC uses this data to carry on its normal business in the following way.

If traffic is not coming in on the DSSI bus, the SHAC goes to the highest command queue which has something enqueued. Choices are CMDQ0..CMDQ3, with 3 being most urgent. It dequeues an element from the queue and examines its header to see what it must do with the queue entry. It could be a command for the SHAC or an item to be delivered to one of the nodes on the DSSI. A command might be an order to deliver a block of data to a remote node. An item to be delivered would be either a datagram or a message.

A *datagram* is a "one-sided" communication—that is, one which will be sent without any assurance of either receipt or reply. An obvious application for such a communication is a request for the party at the other node to identify itself. If the host does not know if anything at all is out there, it must transmit its request without expectation. For this or any similar purpose, it employs a datagram. Datagrams are all of lengths guaranteed to fit in a datagram envelope.

A *message* is a "two-sided" communication used when a virtual circuit (an established formal relationship) between members of the bus exist. Once such a virtual circuit is established, the host(s) understand how to make requests of the other side. Such a request could be an order for a data transfer in either direction. The message itself (*move data*) is contained in a command (*deliver this message to ...*). Messages are all of lengths guaranteed to fit in a message envelope. Messages are always delivered sequentially to a given node—that is, in the order in which they were enqueued on a particular queue. While the SHAC supports retries if a message fails to get through, once the retry limit is reached without successful delivery, SHAC returns the command to the host, marking it as *undeliverable*, and then breaks the virtual circuit to that node.

A full transaction might go something like this:

1. The host queues a message for node 3 (for example, a disk controller) to copy a block of 16 KB from host memory, starting at location X and to be stored in location Y on disk. The queues are doubly-linked, so at the top of every envelope there is a forward link **FLINK** and a backward link **BLINK**. Enqueuing involves putting link values into the new element's FLINK and BLINK and making the previous last-element's FLINK and the queue header's BLINK point to the new element.
2. When this message gets to the head of the queue, the SHAC dequeues it [1], reads the header and finds that it should "dial up" node 3. To do this, the SHAC goes through the DSSI protocols, contending for the DSSI bus and then, if successful in getting bus, specifying node 3 as the target. These steps are called *arbitration* and *selection*.

[1] Note that the SHAC ends up holding the only pointer to the dequeued block of memory that constitutes the queue element. The port driver no longer "knows" where it is.

3. Node 3 responds by asking for the DSSI-command (*command-out phase*). In this phase, the SHAC tells node 3 how many bytes are coming and repeats the identification information to confirm a proper selection. Node 3 then tells the SHAC to switch to the *data-out phase*. The SHAC sends a pair of CI header bytes to identify what type of message this is, and then proceeds to transmit the actual message read from the message block in host memory. The step-by-step details of the transfer are handled by hardware in the SHAC which permits simultaneous, buffered reading and writing on the two busses to which the SHAC is connected. Upon proper completion of the transmission, node 3 responds with a 1-byte acknowledgment of success (parity and check-sum proper and no other errors).
4. The SHAC is still holding the only pointer to the message block in host memory. It returns this to the host in one of two ways. If the host has requested a "return receipt", the SHAC puts the block on the Response Queue **RSPQ** to indicate proper delivery. This is where the port-driver software in the host will look for responses.

   Alternatively, the SHAC simply puts it back on the MFREEQ which holds the standard envelopes for messages. At this point the single message has been delivered and the message envelope is back in circulation.
5. After whatever delay node 3 needed to process the message, it contends for the bus and upon winning it, selects the SHAC as its target. It then sends a standard CI message as above telling the SHAC to transmit the required data. In general, the SHAC does not do this immediately, because it is obliged to handle traffic according to position in the queue and according to queue priority. Instead, it takes an empty envelope from MFREEQ, writes into it the message it is receiving and puts it on the proper CMDQ as specified in the message it just received.
6. When that message gets to the head of its queue, the SHAC dequeues it once more, carries out its command (in transmissions of 4 KB whenever possible—a 4 KB transmission takes about 1 msec on the DSSI), possibly interleaving other transmissions of higher priority to this node or any priority to other nodes, until the last byte is sent. Once the SHAC has completed this operation, it returns the message block to the MFREEQ.
7. Node 3 has put its data on the disk and must report to the host the successful completion of the transaction. Again it contends for the bus and upon winning specifies the SHAC as its target. Then it sends a message to the port-driver through the SHAC confirming the successful transaction. The SHAC dequeues another free envelope and writes this message into that block. Then it queues it on the host's RSPQ. Except for higher level responses in the host, that concludes a whole transaction.

The enqueue/dequeue operations represent a considerable fraction of the effort in delivering a message or datagram. To minimize this effort, the SHAC caches a small number of the envelopes (that is, it hangs onto the pointers to the memory blocks) as they become free in its normal activity. It only fetches an envelope from the free queues when its own supply has disappeared, and it only returns them to the free queues when it has a full supply (4 of a type). By this and other attention to effort reduction and traffic conservation, the SHAC attempts to optimize its rate of doing useful work.

11.3 SHAC Registers

The CPU communicates directly with the SHAC chip through a set of device registers in each the SHAC. These registers occupy a one-page (512-byte) region in I/O address-space, aligned on a page boundary.

All of the registers are longword registers. They may be accessed only through longword operations.

In addition to the access restrictions listed for specific registers, no register other than SHAC Software Chip Reset (SSWCR) may be read or written while certain chip intialization functions are being executed. The results of such an access during the 100 milliseconds following a reset (power-up or a write to SSWCR), or during the 50 microseconds following a MIN-bit (PMCSR<0>) reset are UNPREDICTABLE.

The registers can be divided into two categories:

- The CI Port registers defined in the CI Port Architecture specification;
- The SHAC specific registers.

11.3.1 CI Port Registers

11.3.1.1 Port Queue Block Base Register (PQBBR)

SHAC I/O Address: 2000 4248$_{16}$

This Port Queue Block Base Register (PQBBR) contains the uppermost bits of the physical address of the base of the Port Queue Block (PQB). After a RESET the PQBBR is loaded by the SHAC with configuration information. This information remains in the PQBBR until the PQBBR is written with the address of the Port Queue Block. Figure 11–2 shows the format. Table 11–1 lists the bit descriptions.

PQBBR is writeable only when the port is in the disabled or disabled/maintenance state and readable anytime (except during chip intialization).

Figure 11–2: Port Queue Block Base Register (PQBBR)

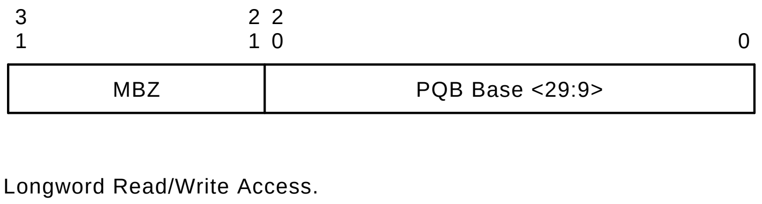


Table 11–1: Port Queue Block Base Address Register (PQBBR)

Data Bit	Name	Description
<31:21>	MBZ	Read as ZERO, must be Written as ZERO.
<20:0>	PQB Base <29:9>	This field contains the uppermost bits of the physical address of the base of the Port Queue Block (PQB). Note, the PQB must be page-aligned, so the remaining bits of the address are assumed to be ZERO.

Following chip reset, PQBBR contains the configuration shown in Figure 11–3. The bit descriptions are listed in Table 11–2.

Figure 11–3: Port Queue Block Base Register (PQBBR) After RESET

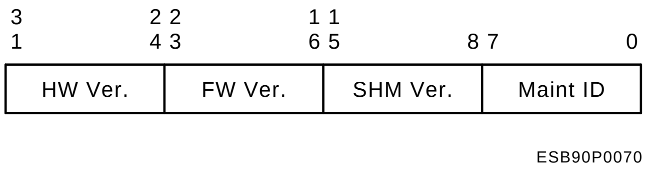


Table 11–2: Port Queue BLock Base Address Register Bits After RESET

Data Bit	Name	Description
<31:24>	HW Ver.	Hardware Version. The Hardware version of the SHAC which is greater than ZERO.
<23:16>	FW Ver.	Firmware Version. The Firmware version of the SHAC which is greater than ZERO.
<15:8>	SHW Ver.	Shared Host Memory Version. The Shared Host Memory version of the SHAC which is ZERO until the Shared Host Memory Data Area has been read in; thereafter, greater than ZERO.
<7:0>	Maint ID	CI Port Maintenance ID. The CI Port Maintenance ID which should always be 22_{16}.

11.3.1.2 Port Status Register (PSR)

SHAC I/O Address: 2000 424C$_{16}$

The Port Status Register (PSR) contains a status report. If interrupts are enabled, for example (PMCSR<2>) set, the port interrupts the SOC CPU each time that it writes to this register. Once an interrupt is requested by the port, the value of PSR is fixed and is not changed until the SOC CPU releases it by writing the Port Status Release Control Register (PSRCR). The Port Status Register format is shown in Figure 11–4 and the bit descriptions are in Table 11–3.

PSR is read only and may be read anytime by the port driver, except during chip initialization. Its value following a write to it is UNPREDICTABLE.

Figure 11–4: Port Status Register (PSR)

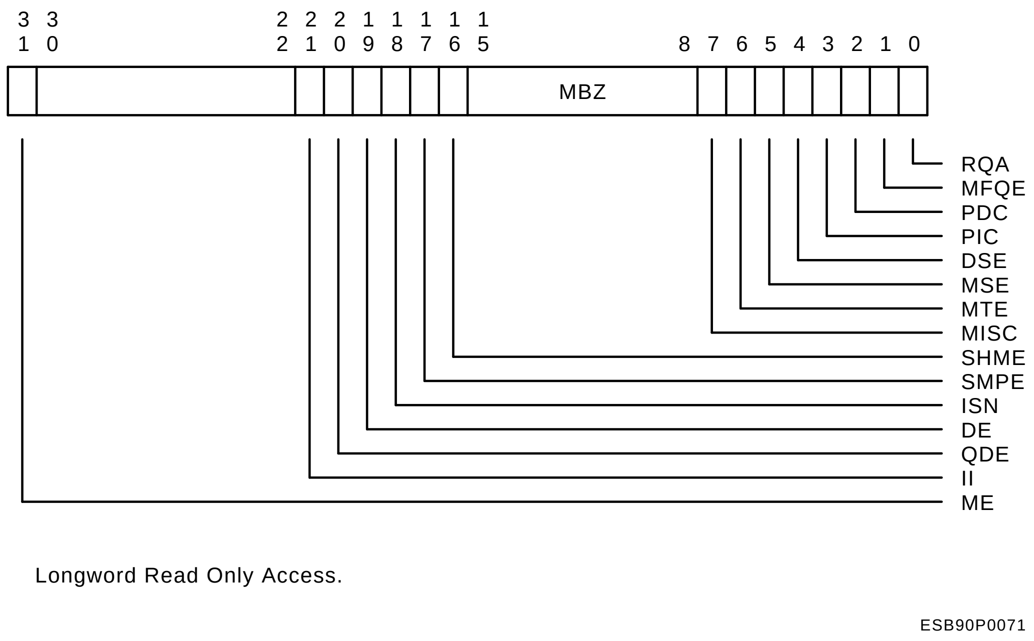


Table 11–3: Port Status Register Bit Descriptions

Data Bit	Name	Description
<31>	MTE	Maintenance Error. When set, the port has detected an implementation specific error (or hardware status condition). The source of the error may be more accurately determined from the other bits in the upper word of this register (PSR) and the contents of other registers. Also when set the port is in the *uninitialized* state (port is non-functional). Maintenance Errors normally indicate a severe SHAC hardware or software failure.
<30:22>	MBZ	Read as ZERO, writes have no effect.
<21>	II	Illegal Interrupt. When set, this bit indicates a SHAC internal error, detected when the SHAC's microprocessor received an interrupt from a invalid source. This causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<20>	QDE	QUIP Detected Error. When set this bit indicates a SHAC internal error detected when the SHAC's microprocessor (QUIP) was given an invalid instruction. This causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<19>	DE	Diagnostic Error. When set an error was detected while the SHAC was running its internal self-test. The causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<18>	ISN	Illegal Segment Number. When set this indicates a SHAC internal error in which it attempted to load a non-existent External Segment from the SHAC Shared Host Memory. The causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<17>	SMPE	Slave Mode Parity Error. This bit is set by the occurrence of a parity error during a SOC CPU access of a SHAC device register. The causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<16>	SHME	Share Host Memory Error. This bit is set by the occurrence of an error involving the SHAC shared Host Memory. The causes ME (PSR<31>) to set and the port to enter the *uninitialized* state (port is non-functional).
<15:8>	MBZ	Read as ZERO, writes have on effect.
<7>	MISC	Miscellaneous. When set this bit indicates that the port microcode has detected one of the miscellaneous errors and the port is about to enter the *disabled/maintenance* state. The actual error code is stored in the Port Error Status Register.
<6>	ME	Maintenance Timer Expiration. When set the maintenance timer has expired. The port is in the *uninitialized/maintenance* state.
<5>	MSE	Memory System Error. When set the port has encountered an uncorrectable data or non-existent memory error in referencing memory. Port is in the *disabled* or *disabled/maintenance* state. See the Port Failing Address Register (PFAR) for further information.

Table 11–3 (Cont.): Port Status Register Bit Descriptions

Data Bit	Name	Description
<4>	DSE	Data Structure Error. When set, the port has encountered an error in a port data structure (for example, queue entry, PQB, BDT or page table). Port is in the *disabled* or *disabled/maintenance* state. See the Port Error Status Register (PESR) and the Port Failing Address Register (PFAR) for further information. Note that errors in queue structures leave the queues locked.
<3>	PIC	Port Initialization Complete. When set, the port has completed internal initialization. The port is in the *disabled* or *disabled/maintenance* state.
<2>	PDC	Port Disable Complete. When set, the port is in the *disabled* or *disabled /maintenance* state.
<1>	MFQE	Message Free Queue Empty. When set, the port attempted to remove an entry from the Message Free Queue (MFREEQ) and found it empty. Port processing of commands continues and, thus, the MFREEQ may not be empty at the time the port driver gets control.
<0>	RQA	Response Queue Available. When set this bit indicates port has inserted an entry on an empty Response Queue.

11.3.1.3 Port Error Status Register (PESR)

SHAC I/O Address: 2000 4250_{16}

The Port Error Status Register (PESR) indicates the type of error which resulted in a DSE (PSR<4>) or an MISC (PSR<7>) error. Figure 11–5 shows the format. Table 11–4 lists the bit descriptions.

PESR is read only by the SOC CPU and valid only after either a DSE or MISC error, or after certain ME (PSR<31>) and DE (PSR<19>) errors. Its value at any other time, or following a write to it, is UNPREDICTABLE.

Figure 11–5: Port Error Status Register (PESR)

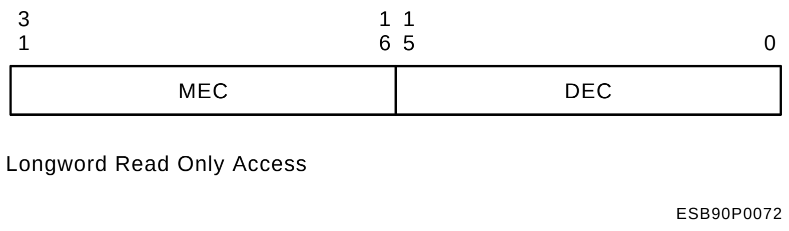


Table 11–4: Port Error Status Register Bit Definitions

Data Bit	Name	Description
<31:16>	MEC	Miscellaneous Error Code. This code comprises two fields: bits <31:24> define the the module within the SHAC code where the error occurred, and bits <23:16> contain the specific error that occurred. These codes are implementation specific.
<15:0>	DEC	Data Structure Error Code.

11.3.1.4 Port Failing Address Register (PFAR)

SHAC I/O Address: 2000 4254$_{16}$

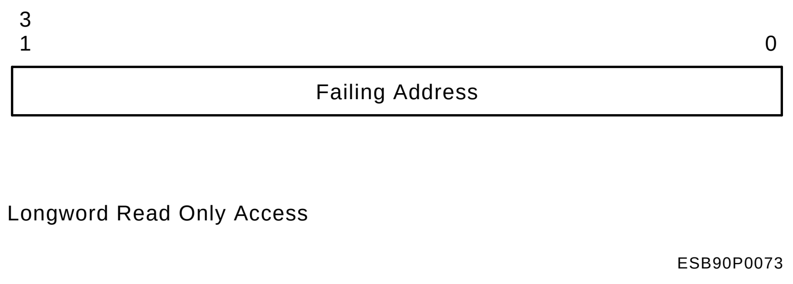
The format for the Port Failing Address Register is shown in Figure 11–6.

After an DSE, MSE, and ME or DE error (as indicated by PSR), or after a response with Buffer Memory System Error status, the Port Failing Address Register (PFAR) contains the memory address at which the failure occurred. The address may be the exact failing address, an address in the same page as the exact failing address or, in the case of DSE, an address in some part of the data structure. For DSE, PFAR contains a virtual address or offset, while for MSE interrupts and buffer memory system errors the PFAR contains a physical address. For ME, the interpretation of the address is error-dependent.

Because the port continues command execution and packet processing after Buffer Memory System Errors, the PFAR is overwritten if subsequent errors occur. For DSE, MSE, and ME errors the PFAR is effectively fixed because the port enters the *disabled*, *disabled /maintenance*, or *uninitialized* state.

PFAR is read only by the SOC CPU and readable after a DSE, MSE, or ME or DE errors or after a response with Buffer Memory System Error status. Its value at any other time, or following a write to it, is UNPREDICTABLE.

Figure 11–6: Port Failing Address Register (PFAR)



11.3.1.5 Port Parameter Register (PPR)

SHAC I/O Address: 2000 4258_{16}

The Port Parameter Register (PPR) contains port implementation parameters and the port number. The value of the PPR is set by the port during initialization and valid after a PIC (PSR <3>) interrupt. Its value at any other time, or following a write to it, is UNPREDICTABLE. PPR is read only by the SOC CPU . The Port Parameter Register format is shown in Figure 11–7. The bit descriptions are listed in Table 11–5.

Figure 11–7: Port Parameter Register (PPR)

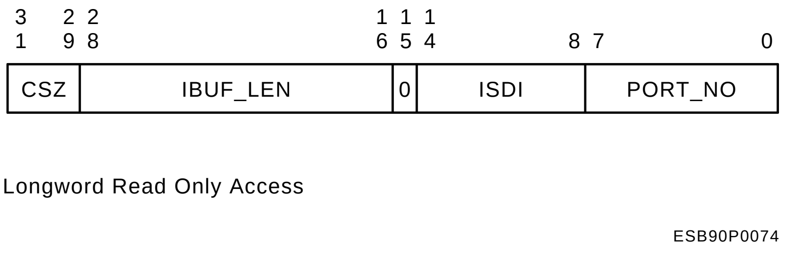


Table 11–5: Port Parameter Register Bit Descriptions (PPR)

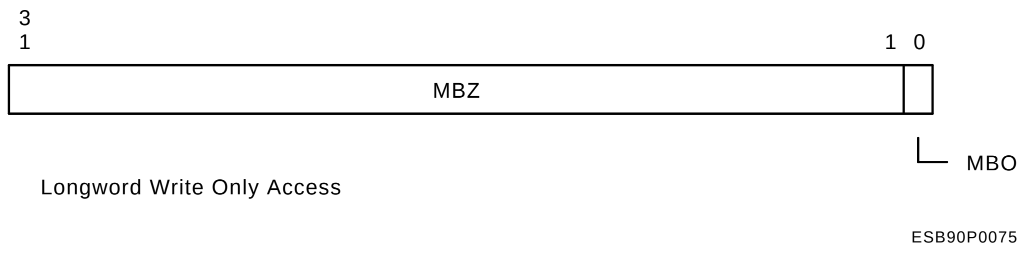
Data Bit	Name	Description
<31:29>	CSZ	Cluster Size. For SHAC, this value always is ZERO, indicating a maximum of 16 ports on the DSSI bus. (Note that the DSSI architecture only allows up to 8 ports on the bus, but 16 is the smallest size defined for the CSZ field.)
<28:16>	IBUF_LEN	Internal Buffer Length. This field indicates the size of internal buffers available for message and data transfers. Maximum data packet = IBUF_LEN - 16 bytes. Maximum message or datagram length = IBUF_LEN. For SHAC, the value is 4112 1010_{16}.
<15>	MBZ	Read as ZERO, writes have an UNPREDICTABLE effect.
<14:8>	ISDI	**Implementation Specific Diagnostic Information.** The bits in this field contain information about the local adapter's link layer configuration. For SHAC, the definitions of these bits are Read as Zero.
<7:0>	Port_NO	Port Number. This is the same as the SHAC's DSSI ID.

11.3.1.6 Port Control Registers

The port control registers are 32-bit registers which are write-only by the SOC CPU . To invoke the function provided by any of the control registers, the SOC CPU writes a ONE to the register.

The result of writing any other value to any of these registers is UNPREDICTABLE. The value read from any of them is also UNPREDICTABLE. The format for the Port Control Registers is shown in Figure 11–8.

Figure 11–8: Port Control Registers



11.3.1.6.1 Port Command Queue 0 Control Register (PCQ0CR)

SHAC I/O Address: 2000 4280$_{16}$

When the port driver inserts an entry in an empty CMDQ0, the port driver writes PCQ0CR to initiate port execution of the Command Queue. PCQ0CR can be written only when the port is in the *enabled* or *enabled/maintenance* state. Writing to PCQ0CR when the port is in any other state has no effect. The SHAC I/O Address is 2000 4280$_{16}$

11.3.1.6.2 Port Command Queue 1 Control Register (PCQ1CR)

SHAC I/O Address: 2000 4284$_{16}$
Same as PCQ0CR except refers to CMDQ1. The SHAC I/O Address is 2000 4284$_{16}$

11.3.1.6.3 Port Command Queue 2 Control Register (PCQ2CR)

SHAC I/O Address: 2000 4288_{16}
Same as PCQ0CR except refers to CMDQ2. The SHAC I/O Address is 2000 4288_{16}

11.3.1.6.4 Port Command Queue 3 Control Register (PCQ3CR)

SHAC I/O Address: 2000 $428C_{16}$
Same as PCQ0CR except refers to CMDQ3. The SHAC I/O Address is 2000 $428C_{16}$.

11.3.1.6.5 Port Datagram Free Queue Control Register (PDFQCR)

SHAC I/O Address: 2000 4290_{16}
When the port driver inserts an entry on the DFREEQ and the latter was previously empty, the port driver writes PDFQCR to indicate the availability of DFREEQ entries. PDFQCR can be written only if the port is in the *enabled* or *enabled/maintenance* State. Writing to PDFQCR when the port is in any other state has no effect. The SHAC I/O Address is 2000 4290_{16}

11.3.1.6.6 Port Message Free Queue Control Register (PMFQCR)

SHAC I/O Address: 2000 4294_{16}
Same as PDFQCR except refers to MFREEQ. The SHAC I/O Address is 2000 4294_{16}

11.3.1.6.7 Port Status Release Control Register (PSRCR)

SHAC I/O Address: 2000 4298_{16}
After the port driver has received an interrupt and read the PSR, it returns the PSR to the port by writing PSRCR. The SHAC I/O Address is 2000 4298_{16}

11.3.1.6.8 Port Enable Control Register (PECR)

SHAC I/O Address: 2000 $429C_{16}$
The port driver enables the port by writing PECR. PECR is ignored if the port is in the *uninitialized* , *uninitialized/maintenance* , *enabled* , or *enabled/maintenance* state. The SHAC I/O Address is 2000 $429C_{16}$

11.3.1.6.9 Port Disable Control Register (PDCR)

SHAC I/O Address: 2000 42A0$_{16}$

The port driver disables the port by writing PDCR. When the port is disabled, the port sets PDC (PSR <2>) and if interrupts are enabled requests an interrupt. PDCR is ignored if the port is in the *uninitialized*, *uninitialized/maintenance*, *disabled*, or *disabled/maintenance* state. The SHAC I/O Address is 2000 42A0$_{16}$

11.3.1.6.10 Port Initialize Control Register (PICR)

SHAC I/O Address: 2000 42A4$_{16}$

The port driver initializes the port by writing PICR. When the initialization is complete the port sets PDC (PSR <2>) and requests an interrupt if interrupts are enabled. As part of the initialization, the maintenance timer is set to expire in 100 seconds. The SHAC I/O Address is 2000 42A4$_{16}$

11.3.1.6.11 Port Maintenance Timer Control Register (PMTCR)

SHAC I/O Address: 2000 42A8$_{16}$

The port driver forces the maintenance timer to reset its expiration time by writing the PMTCR. If the PMTCR is not written again before the expiration time, the port will enter the *uninitialized/maintenance* state setting MTE (PSR <6>) and request an interrupt if interrupts are enabled. PMTCR is ignored if the maintenance timer is not running. The SHAC I/O Address is 2000 42A8$_{16}$

11.3.1.6.12 Port Maintenance Timer Expiration Control Register (PMTECR)

SHAC I/O Address: 2000 42AC$_{16}$

The port driver forces a Maintenance-Timer-Expiration Interrupt by writing the PMTECR. This register may be written only when the port is in the *enabled*, *enabled/maintenance*, *disabled*, and *disabled/maintenance* states and only while the Maintenance Timer is not disabled. The SHAC I/O Address is 2000 42AC$_{16}$

11.3.1.6.13 Port Maintenance Control and Status Register (PMCSR)

SHAC I/O Address: 2000 425C$_{16}$

The Port Maintenance Control and Status Register (PMCSR) is used for maintenance level control and status reporting. The CI Port specification defines all but the 2 least significant bits. The format is shown in Figure 11–9 and the bit descriptions are listed in Table 11–6

The bits can be divided into the following categories:

- Status bits - Which are set by the port to report various conditions. They are cleared by maintenance initialization or clearing the condition in another register. PMCSR does not include any status bits at this time.
- Function control bits are read/write by the port driver only. They are clear on a RESET.

  These bits are of two classes:

  1. Init: this type of bit invokes a function (for example, initialization) by setting it. It always reads as zero, except while the function is active.
  2. Enable/disable: this type of bit causes an activity or state to exist while the bit is set. Clearing the bit stops the activity or changes the state. The bit always reads the most recently written value. The bit is never changed by the port.

Figure 11–9: Port Maintenance Control And Status Register (PMCSR)

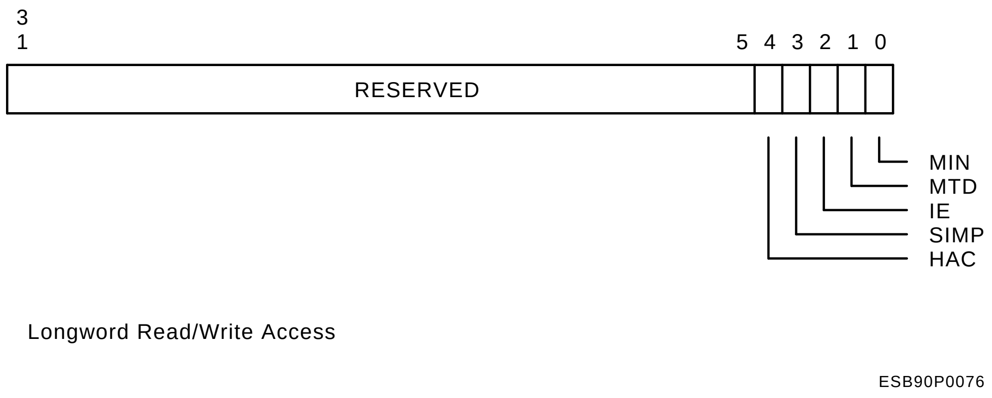


Table 11–6: Port Maintenance Control and Status Register (PMCSR) Bits

Data Bit	Name	Description
<31:5>	RESERVED	This bits are reserved. They should not be written; reads return UNPREDICTABLE results.
<4>	HAC	Host Access Feature. This bit Must Be ZERO, except for diagnostic purposes. This is an enable/disable class control bit.
<3>	SIMP	Simple SHAC Mode. Must Be ZERO, except for diagnostic purposes. This is an enable/disable class control bit.
<2>	IE	Interrupt Enable. When set, interrupts from the port to the SOC CPU are enabled. Power-up state is clear (interrupts disabled). This is an enable/disable class control bit.

Table 11–6 (Cont.): Port Maintenance Control and Status Register (PMCSR) Bits

Data Bit	Name	Description
<1>	MTD	Maintenance Timer Disable. Read/Write by SOC CPU . If set, the maintenance timer is turned off. Timer is set to the initial value and suspended. If clear, timer functions normally. Power-up state is clear (timer enabled). This is an enable/disable class control bit.
<0>	MIN	Maintenance Init. Writing a ONE to this bit resets the port. Upon completion, the port is in the *uninitialized* state and MIN is clear. Writing a ZERO to this bit has no effect. It always reads as ZERO, except while the reset function is active. Although Maintenance Init resets the port, it is not equivalent to a write to the SHAC Software Chip Reset register. In particular, the SHAC Shared Host Memory Address is not reset by Maintenance Init.

11.3.2 SHAC Specific Registers

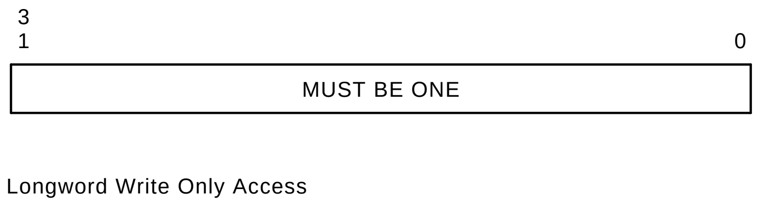
These registers, which are not defined in the CI Port Architecture, are used for additional maintenance level control.

11.3.2.1 SHAC Software Chip Reset Register (SSWCR)

SHAC I/O Address: 2000 4230$_{16}$
When the SOC CPU writes FFFF FFFF$_{16}$ to the SHAC Software Chip Reset register (SSWCR), a chip reset is performed. The result is equivalent to that of the hardware chip reset that occurs following system power-up. On completion, all device registers are reset to their power-up state, and the port is in the *uninitialized* state. The format is shown in Figure 11–10.

SSWCR is write only by the SOC CPU and may be written to at any time. Its value when read is UNPREDICTABLE. The result if other than FFFF FFFF$_{16}$ is written to SSWCR is UNDEFINED.

Figure 11–10: SHAC Software Chip Reset (SSWCR)



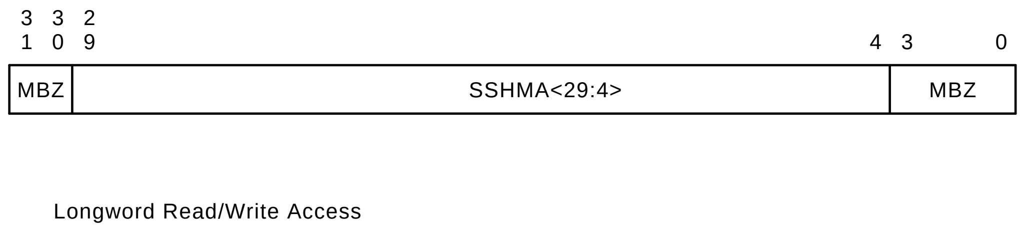
ESB90P0077

11.3.2.2 SHAC Shared Host Memory Address (SSHMA)

SHAC I/O Address: 2000 4244$_{16}$

The format for the SHAC Shared Host Memory Address is shown in Figure 11–11.

Figure 11–11: SHAC Shared Host Memory Address (SSHMA)



ESB90P0078

Following chip reset, the SOC CPU writes into the SHAC Shared Memory Address register (SSHMA) the physical address of the Shared Host Memory Header. The area must be octaword aligned and contiguous in physical memory.

SSHMA is read/write by the SOC CPU , but may be written only when the port is in the *uninitialized* state. Writing when the port is in any other state can produce unpredictable results.

Chapter 12

KA660 Firmware

This document describes the KA660 functional firmware. The firmware is VAX–11 code which resides in EPROM on the KA660 module. Typically KA660 firmware gains control whenever the onboard CPU "halts", or more precisely, performs a "processor restart" operation. However, portions of the firmware can also be invoked by applications through a public subroutine linkage.

When the KA660 firmware is running, it provides services expected of a standard VAX console subsystem. In particular, the following services are available:

- Automatic restart or bootstrap of customer application images at power-up, on reset, or conditionally after processor halts.
- Diagnostic tests executed both at power-up and by request, which verify the correct operation of the CPU and memory modules.
- Operator interface providing complete examination or modification of the processor state.

Throughout this document, "firmware" is a generic term describing all program code located in the KA660 EPROM. Sometimes it is referred to as either the "boot ROM", "diagnostics ROM", or "console ROM", depending on context. Each major element of the firmware is referred to by other terms, for instance, the boot program as "VMB" or "primary bootstrap", the ROM based diagnostic program as the "diagnostic" or "self-test", and the operator interface as the "console" or "console program".

Certain terminology and conventions are used throughout this document. With one exception, numbers unless otherwise indicated or implied are decimal. Eight digit numbers throughout this document are hexadecimal longwords, typically representing VAX 32 bit addresses or data. Where there is ambiguity, the radix is explicitly stated. For instance, 72 is assumed to be decimal and for clarity can be written as 72 (dec). However, alternate representations for 72 are 1001000 (bin) for binary, 110 (oct) for octal, or 48 (hex) for hexadecimal. On the other hand, 20040000 is the hexadecimal address or the base of the firmware EPROM.

Ranges of integers are expressed as a pair of numbers separated by a colon and are always inclusive. For example, 7:4 specifies the range of integers from 7 to 4, namely 7, 6, 5, and 4.

A bit field or position within a register or data structure follows the structure name and is enclosed in angle brackets. The associated field name (if defined) typically follows the field definition and appears in parenthesis. For instance, PSL<20:16> (IPL) represents the five bit field for the Interrupt Priority Level in the Processor Status Longword.

12.1 General Description

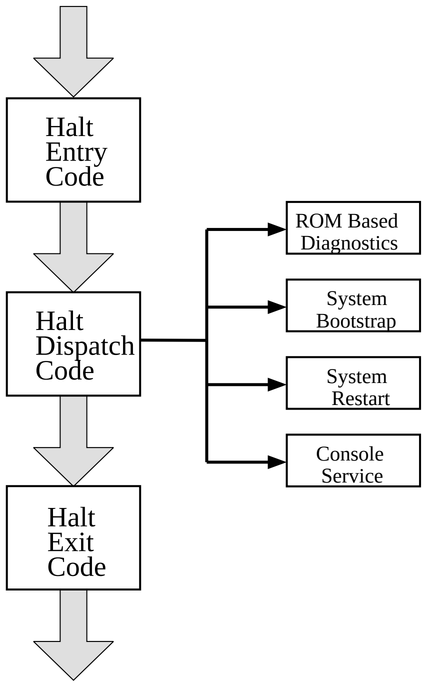
The KA660 firmware provides the following services:

- Diagnostics that test all components on the board and verify the module is working correctly.
- Automatic/manual bootstrap of an operating system following processor halts.
- Automatic/manual restart of an operating system following processor halts.
- An interactive command language that allows the user to examine and alter the state of the processor.
- Support of various terminals and devices as the system console.
- Multilanguage support for displaying critical system messages and handling LK201 country specific keyboards.

The remainder of this section describes in detail the functions and external characteristics of the KA660 firmware.

The KA660 firmware is comprised of several major functional blocks of code. The halt **entry** code is invoked following system halts, resets, or severe errors. This code is responsible for saving the machine state and transfering control to the halt dispatcher code. The halt **dispatcher** code determines the nature of the halt, then transfers control to the appropriate subcode. The halt **exit** code is invoked whenever a transition is desired from a halted state to the running state. This code performs a restoration of the saved context prior to the transition. Figure 12–1 illustrates this and these functions are discussed in detail in Section 12.2.

Figure 12–1: KA660 Firmware Structural Components



The *ROM based diagnostics* consist of an initial power-up test and a series of functional component diagnostics invoked by a diagnostic executive. These functions are described in Section 12.3 on power-up and in Section 12.8 on diagnostics.

Depending on the nature of the halt and the hardware context, the firmware attempts either an *operating system restart* (discussed in Section 12.5), *a bootstrap operation* (described in Section 12.4), or transitions to *console I/O mode* (covered in Section 12.6).

12.2 Halt Code

The main purpose of the halt code is to save the state of the machine on halt entry, invoke the dispatcher, and restore the state of the machine on exit to program I/O mode. It is comprised of Halt Entry, Halt Dispatch, and Halt Exit codes.

12.2.1 Halt Entry - Saving Processor State

The entry code, residing at physical address 20040000, is executed whenever the KA660 halts. The value that the program counter contained when the processor was halted is saved in IPR 42 (PR$_SAVPC). On a power-up, the PR$_SAVPC register value is undefined.

The processor will halt for a variety of reasons. The reason for the halt is stored in PR$_ SAVPSL<13:8>(RESTART_CODE), IPR 43. A complete list of the halt reasons and the associated console messages can be found in Table I–1 in Appendix I.

After a halt, the firmware first saves the current LED code then writes an "E" to the diagnostic LEDs. This action occurs within several instructions after the firmware has been invoked. The intent of saving the LED code is to let the user know that at least some instructions have been successfully executed.

The KA660 firmware unconditionally saves the contents of the following registers on any halt:

- R0 through R15, the general purpose registers.
- PR$_SAVPSL, the saved PSL register.
- PR$_SCBB, the system control block base register.
- DLEDR, the diagnostic LED register.

NOTE

The SSC programmable timer registers are not saved. In some cases, such as bootstrap, the timers are used by the firmware and previous "time" context is lost.

Several registers are unconditionally set to pre-determined values by the firmware on any halt, processor init, or bootstrap. This action insures that the firmware itself can run and protects the board from physical damage.

The following is a list of registers that fall into this category.

- The SSC configuration register (SSCCR)
- The SSC address match and mask registers (ADxMCH & ADxMSK)
- The CDAL bus timeout control register (CBTCR)
- The SSC timer interrupt vector registers (TIVRx)

Whenever the Halt Entry Code is invoked, the firmware sets the console serial line baud rate based on the value read from the BDR and extends the halt protection from 8KB to 256KB to include the all of the EPROM.

12.2.2 Halt Dispatch

The action taken by the firmware on a halt is dependent primarily on the following information:

- The state of BREAK enable switch, BDR<23>(HALT_ENABLE).
- The state of the console program mailbox, CPMBX<1:0>(HALT_ACTION).
- The user defined halt action (SET HALT).
- The halt code, PR$_SAVPSL<13:8>(RESTART_CODE).

In general, the BREAK enable switch governs whether or not a BREAK condition from the console serial line is recognized by the KA660. This swith also determines the default action taken on a power-up or other internal halt condition. By default, if BREAKs are enabled, the firmware invokes the console emulation code. If BREAKs are disabled, the firmware attempts a recovery operation.

However, the console program mailbox, CPMBX<1:0>(HALT_ACTION). (refer to Figure G–2) is used by operating systems to override the BREAK enable switch and instruct the firmware to invoke the console service, attempt to restart the operating system, or reboot the system following a halt, regardless of the setting of the BREAK enable switch.

The user defined halt action invoked by using the SET HALT console command (refer to the description of the SET command in Section 12.7, Console commands is an alternative way to specify a default halt action. This feature allows users to specify auto-booting on power-ups, even when BREAKs are enabled. For HALT instructions and error halt conditions, it is similar in function to the console program mailbox but has lower precedence and is only used when the console program mailbox is 0. This provides the user with a mechanism by which to specify what action should be taken, in the event that the operating system or user application does not set the console program mailbox.

The halt (or restart) code is automatically deposited in PR$_SAVPSL<13:8>(RESTART_CODE) on any halt condition. This field indicates the cause of the halt and for the purpose of dispatching collapses into three categories.

02: External halts.
03: Reset/power-up.
xx: (All other values) HALT instruction and all error halts.

Table 12–1 summarizes the action taken on all halt conditons, except external halts which are described in Section 12.2.2.1. The actual halt dispatch state machine is described in detail in Section H.0.1 of Appendix H.

Table 12–1: Halt Action Summary

Halt Code= 3	Break Enable Switch	User Defined Halt Action	Console Program Mailbox	Action(s)
T	1	0,1,3	x	diagnostics, console
T	1	2,4	x	diagnostics, if success boot, if either fail console
T	0	x	x	diagnostics, if success boot, if either fail console
F	1	0	0	console
F	0	0	0	restart, if this fails boot, if that fails console
F	x	1	0	restart, if it fails console
F	x	2	0	boot, if it fails console
F	x	3	0	console
F	x	4	0	restart, if this fails boot, if that fails console
F	x	x	1	restart, if it fails console
F	x	x	2	boot, if it fails console
F	x	x	3	console

"T" TRUE—indicates a Reset or Power-up condition.
"F" FALSE—indicates a HALT instruction or error halt condition.
"x" DON'T CARE—indicates that the condition is "don't care".

Because the KA660 does not support battery backed up main memory, an operating system restart operation is not attempted on a power-up.

12.2.2.1 External Halts

Several conditions can trigger an external halt and different actions are taken depending on the condition.

An external halt can be caused by one of the following conditions.

1. A BREAK condition on the system console serial line, if the BREAK enable switch is set to "enabled". In this case BDR<23>(HALT_ENABLE) = 1 and the console code is invoked. **Control-P may be established as the "BREAK" condition by using the SET CONTROLP ENABLE console command.**
2. The assertion of the BHALT line on the Q22-bus, causes an external halt if the SCR<14>(BHALT_ENABLE) bit in the CQBIC is set. As a result, the console code is invoked.
3. The negation of DCOK on the Q22-bus, if the SCR<7>(DCOK_ACTION) bit is set causes an external halt. (by default this bit is clear). As a result, the console code is invoked.

4. Recognition of a valid MOP BOOT message by an appropriately initialized SGEC, if the REMOTE_BOOT_ENABLE jumper is in place (BDR<12>(REMOTE_BOOT_ENABLE) = 1). As a result, a bootstrap is attempted and if that fails, the console is entered.

NOTE

The firmware does not initialize the SGEC for this operation. The operating system must set up the SGEC to support this feature.

NOTE

The switch labeled "RESTART" negates DCOK. The DCOK bit may also be negated by the DEQNA sanity timer, or any other Q22-bus module that chooses to implement the Q22-bus restart/reboot protocol. Because the SCR<7>(DCOK_ACTION) bit is cleared on power-up, the default consequence to deasserting DCOK is to generate a processor restart. Hence, pushing the "RESTART" button typically initiates a power-up sequence and destroys system state.

12.2.3 Halt Exit - Restoring Processor state

When the firmware exits, it uses the currently defined saved context. This context is initially determined by what was saved when the firmware code was invoked. However this context may be modified by console commands, or automatic operations such as an automatic bootstrap on power-up.

When restoring the context, the firmware will flush the CPU internal cache if enabled, and invalidate all translation buffer entries via the internal processor register PR$_TBIA, IPR 57.

In restoring the context, the console pushes the user's PSL and PC onto the user's interrupt stack, then executes a *return from exception or interrupt* instruction (REI) from that stack. This implies that the user's Interrupt Stack Pointer (ISP) is valid before the firmware can exit. This is done automatically on a bootstrap. However, it is suggested that the stack pointer (SP) be set to a valid memory location before issuing the START or CONTINUE command. Furthermore, the user should validate the System Control Block Base Register (SCBB or PR$_ SCBB) prior to executing a NEXT command, because the firmware uses the trace trap vector for this function. At power-up, the user ISP is set to 200 (hex) and the SYstem Control Block Base Register is undefined.