VAX 6000 Series
Vector Processor Owner's Manual

Order Number EK–60VAA–OM–001

This manual is for the system manager or system operator of a VAX 6000 system with a vector processor. The manual expands upon information found in the *VAX 6000–400 Owner's Manual* and the *Mini-Reference*.

digital equipment corporation
maynard, massachusetts

First Printing, May 1990

Contents

Appendix C Vector Module Registers

Index

Examples

Figures

Tables

Preface

Intended Audience

This manual is for the system manager or system operator of a VAX 6000 system with a vector processor. The day-to-day operations of the system are detailed in the *Owner's Manual* that ships with the system; this manual focuses on information pertaining to systems with vector processors.

Document Structure

The manuals in the VAX 6000 series documentation set are designed using structured documentation theory. Each topic has a boldface indented abstract, to help you use the manual as a reference tool. Other typical components of a topic include an illustration or example, a chart or list, and descriptive text.

This manual has two chapters and three appendixes:

- **Chapter 1, VAX Vector Processing System,** gives an overview of VAX 6000 systems with vector processors.
- **Chapter 2, Vector Console Commands,** describes the console commands used with vector processors.
- **Appendix A, Self-Test,** describes how to interpret the console display for self-test and the LEDs on processor modules.
- **Appendix B, Vector Processor Error Messages,** lists error messages associated with the vector module.
- **Appendix C, Vector Module Registers,** gives the registers associated with the vector module.

<REFERENCE>(VAX_XXXX) Documents

Documents in the <REFERENCE>(VAX_XXXX) documentation set include:

Title	Order Number
<REFERENCE>(VAX_XXXX) Installation Guide	EK–640EA–IN
<REFERENCE>(VAX_XXXX) Owner's Manual	EK–640EA–OM
<REFERENCE>(VAX_XXXX) Mini-Reference	EK–640EA–HR
<REFERENCE>(VAX_XXXX) System Technical User's Guide	EK–640EB–TM
<REFERENCE>(VAX_XXXX) Options and Maintenance	EK–640EB–MG
VAX 6000 Series Upgrade Manual	EK–600EB–UP
VAX 6000 Series Vector Processor Owner's Manual	EK–60VAA–OM
VAX 6000 Series Vector Processor Programmer's Guide	EK–60VAA–PG

Associated Documents

Other documents that you may find useful include:

Title	Order Number
CIBCA User Guide	EK–CIBCA–UG
DEBNI Installation Guide	EK–DEBNI–IN
Guide to Maintaining a VMS System	AA–LA34A–TE
Guide to Setting Up a VMS System	AA–LA25A–TE
HSC Installation Manual	EK–HSCMN–IN
H4000 DIGITAL Ethernet Transceiver Installation Manual	EK–H4000–IN
H7231 Battery Backup Unit User's Guide	EK–H7231–UG
Installing and Using the VT320 Video Terminal	EK–VT320–UG
Introduction to VMS System Management	AA–LA24A–TE
KDB50 Disk Controller User's Guide	EK–KDB50–UG
RA90 Disk Drive User Guide	EK–ORA90–UG

Title	Order Number
RV20 Optical Disk Owner's Manual	EK–ORV20–OM
SC008 Star Coupler User's Guide	EK–SC008–UG
TK70 Streaming Tape Drive Owner's Manual	EK–OTK70–OM
TU81/TA81 and TU81 PLUS Subsystem User's Guide	EK–TUA81–UG
ULTRIX–32 Guide to System Exercisers	AA–KS95B–TE
VAX Architecture Reference Manual	EY–3459E–DP
VAX Systems Hardware Handbook — VAXBI Systems	EB–31692–46
VAX Vector Processing Handbook	EC–H0419–46
VAXBI Expander Cabinet Installation Guide	EK–VBIEA–IN
VAXBI Options Handbook	EB–32255–46
VMS Installation and Operations: VAX 6000 Series	AA–LB36B–TE
VMS Networking Manual	AA–LA48A–TE
VMS System Manager's Manual	AA–LA00A–TE
VMS VAXcluster Manual	AA–LA27A–TE
VMS Version 5.4 New and Changed Features Manual	AA–MG29C–TE

Chapter 1
VAX Vector Processing System

This chapter describes the architecture for the VAX 6000 Model 400 systems that support attached vector processors. Earlier models of the 6000 series do not support vector processing.

1.1 VAX 6000 System Architecture

The VAX 6000 computer systems use the high-speed system bus, the XMI, to interconnect processors and memory modules. The VAX 6000 Model 400 series supports multiprocessing with up to six scalar processors or one or two scalar/vector pairs. In Figure 1–1 the DWMBA adapter serves as the interface to I/O devices on the VAXBI bus.

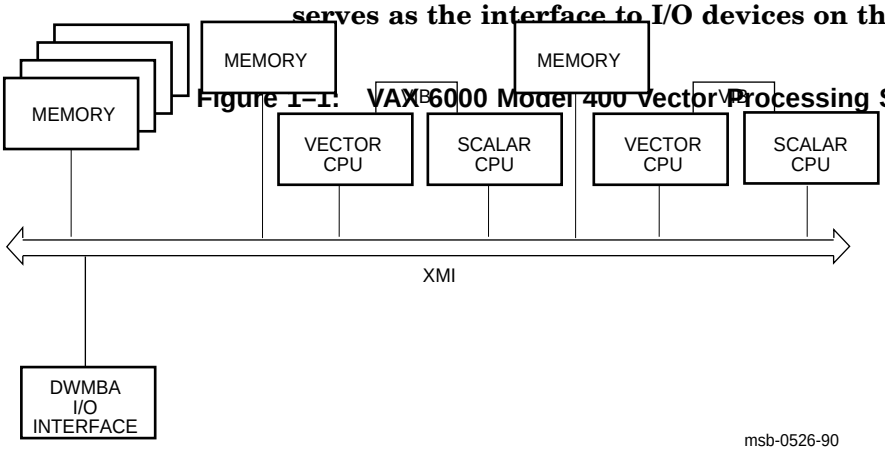


Figure 1–1: VAX 6000 Model 400 Vector Processing System

VAX 6000 Model 400 systems support vector processing. The FV64A vector processor is an integrated vector processor; that is, the vector processor module performs as a coprocessor that is tightly coupled with a host scalar processor. To an executing program, the scalar/vector pair of modules appear as one processor.

The two processor modules are physically connected by an intermodule cable, the VIB. The scalar processor is specifically designed to support its vector coprocessor, and the VAX vector instruction set is implemented as part of the host native instruction set. Both the scalar and vector processors are on the XMI bus, and they both access a common memory.

A VAX 6000 Model 400 system can have one or two scalar/vector pairs. If the system has only one pair, it can also have up to three additional scalar processors. Table 1–1 lists the maximum number of scalar and vector processor modules allowed.

Table 1–1: Processor Module Combinations

Maximum CPUs	Maximum Vectors	Configuration (Slot 1 at Right)
6	0	P P P P P P
4	1	M V P P P P
2	2	M V P M V P

For performance reasons, the scalar processor of a scalar/vector pair should not be made the primary processor when other scalar processors are in the system. For optimal performance, two memory modules are required for one scalar/vector pair, and four memory modules are required for two scalar/vector pairs.

NOTE: *Installation of an <REFERENCE>(xrv) vector processor requires that the* ***attached*** *<REFERENCE>(xrp) module (T2015) be at a minimum revision of K. In addition, the ROMs on any additional <REFERENCE>(xrp) modules must be at a minimum revision of V2.0 (ROM 0 and ROM 1).*

Chapter 2
Vector Console Commands

This chapter describes the console commands that allow communication with a vector processor module.

Individual sections include:

- Console commands
- DEPOSIT command
- EXAMINE command
- SET CPU command
- Sample console session

A sample console session (see Section 2.5) shows the system response to the SHOW CPU and SHOW CONFIGURATION console commands.

2.1 Console Commands

Using the console program, you can examine and modify the system memory and registers, boot or restart an operating system, designate a primary processor, disable a vector processor, and return to program mode.

Section 2.2 through Section 2.4 give details on the console commands that are used with a vector processor; these are the DEPOSIT, EXAMINE, and SET CPU commands. For details on all console commands in Table 2–1, see your system *Owner's Manual*.

Table 2–1: Console Commands and Qualifiers

Command and Qualifiers	Function
BOOT /R3:n /R5:n /XMI:n /BI:m /NODE:n	Initializes the system, causing a self-test, and begins the boot program.
CLEAR EXCEPTION	Cleans up error state in XBER and RCSR registers.
CONTINUE	Begins processing at the address where processing was interrupted by a CTRL/P console command.
DEPOSIT /B /G /I /L /M /N /P /Q /V /VE /W	Stores data in a specified address.
EXAMINE /B /G /I /L /M /N /P /Q /V /VE /W	Displays the contents of a specified address.
FIND /MEMORY /RPB	Searches main memory for a page-aligned 256-Kbyte block of good memory or for a restart parameter block.
HALT	Null command; no action is taken since the processor has already halted in order to enter console mode.
HELP	Prints explanation of console commands.

Table 2–1 (Cont.): Console Commands and Qualifiers

Command and Qualifiers	Function
INITIALIZE [n] /BI:n	Performs a system reset, including self-test.
REPEAT	Executes the command passed as its argument.
RESTORE EEPROM	Copies the TK tape's EEPROM contents to the EEPROM of the processor executing the command.
SAVE EEPROM	Copies to the TK tape the contents of the EEPROM of the processor executing the command.
SET BOOT	Stores a boot command by a nickname.
SET CPU [n] /ENABLED /ALL /NOENABLED /NEXT_PRIMARY /PRIMARY /ALL /NOPRIMARY	Specifies eligibility of processors to become the boot processor.
/VECTOR_ENABLED /NOVECTOR_ENABLED	Specifies whether the vector processor is to be included in the system configuration.
SET LANGUAGE ENGLISH INTERNATIONAL	Changes the output of the console error messages between numeric code only (international mode) and code plus explanation (English mode).
SET MEMORY /CONSOLE_LIMIT:n /INTERLEAVE:(n+n...) /INTERLEAVE:DEFAULT /INTERLEAVE:NONE	Designates the method of interleaving the memory modules; supersedes the console program's default interleaving.
SET TERMINAL /BREAK /NOBREAK /HARDCOPY /NOHARDCOPY /SCOPE /NOSCOPE /SPEED:n	Sets console terminal characteristics.
SHOW ALL	Displays the current value of parameters set.
SHOW BOOT	Displays all boot commands and nicknames that have been saved using SET BOOT.

Table 2–1 (Cont.): Console Commands and Qualifiers

Command and Qualifiers	Function
SHOW CONFIGURATION	Displays the hardware device type and revision level for each XMI and VAXBI node and indicates self-test status.
SHOW CPU	Identifies the primary processor and the status of other processors.
SHOW ETHERNET	Locates all Ethernet adapters on the system and displays their addresses.
SHOW LANGUAGE	Displays the mode currently set for console error messages, international or English.
SHOW MEMORY	Displays the memory lines from the system self-test, showing interleave and memory size.
SHOW TERMINAL	Displays the baud rate and terminal characteristics functioning on the console terminal.
START	Begins execution of an instruction at the address specified in the command string.
STOP /BI:n	Halts the specified node.
TEST /RBD	Passes control to the self-test diagnostics.
UPDATE	Copies contents of the EEPROM on the processor executing the command to the EEPROM of another processor.
Z /BI:n	Logically connects the console terminal to another processor on the XMI bus or to a VAXBI node.
!	Introduces a comment.

2.2 DEPOSIT Command

The DEPOSIT command stores data in a specified address. Various qualifiers provide access to the vector data registers (/VE), IPRs (/I), and vector indirect registers (/M). No qualifier is needed to deposit to VMR, VCR, and VLR.

2.2.1 Syntax and Qualifiers

Table 2–2: DEPOSIT Command Qualifiers

Qualifier	Meaning
/B	Defines data size as a byte.
/G	Defines the address space as the general register set, R0 through R15.
/I	Defines the address space as the internal processor registers, accessed through MTPR and MFPR instructions.
/L	Defines data size as a longword; initial default.
/M	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/N:<count>	Defines the address space as the first of a range.[1] <count> is a required value with /N.
/P	Defines the address space as physical memory; initial default.
/Q	Defines data size as a quadword; initial default for vector registers (except for VCR and VLR).
/V	Defines the address space as virtual memory. All access and protection checking occur. Use when your operating system has been running prior to system halt.[2]
/VE	Defines the address space as the vector register set.
/W	Defines data size as a word.

[1]The console deposits to the first address, then to the specified number of succeeding addresses. Even if the address is '–', the succeeding addresses are at higher addresses (that is, the symbol specifies only the starting address, not the direction).

[2]If memory management has not been enabled, virtual addresses are equal to physical addresses. If access is not allowed to a program running with the current processor status longword (PSL), the console issues an error message. Virtual space deposits cause the PTE<M> bit to be set in the mapping PTE and force the processor write buffer to be flushed.

The DEPOSIT command syntax is:

```
D[EPOSIT] [/qualifier] <address>  <data>
```

where /qualifier is a value from Table 2–2, and the variable <data> is a hexadecimal value to be stored. The value must fit in the data size to be deposited. The variable <address> is a 1- to 8-digit hexadecimal value or one of the following:

- PSL, the processor status longword. You cannot use any address space qualifier with PSL.
- PC, the program counter. The address space is set to /G.
- SP, the stack pointer. The address space is set to /G.
- Rn, the general purpose register *n*. The register number is in decimal. The address space is set to /G.
- VCR, 7-bit Vector Count Register. No address qualifier is permitted.
- VLR, 7-bit Vector Length Register. No address qualifier is permitted.
- VMR, 64-bit Vector Mask Register. No address qualifier is permitted.
- V0–V15, vector registers. Elements of a vector register are specified V*n:mm*, where *n* is a decimal number 0–15 specifying the vector register, and *mm* is a hex number 0–3F specifying the element within the vector register. The address qualifier must be set to /VE.
- +, the location immediately following the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location plus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address plus one.
- –, the location immediately preceding the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location minus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address minus one.
- *, the last location you referenced in an EXAMINE or DEPOSIT command.
- @, the location addressed by the last location you referenced in an EXAMINE or DEPOSIT command.

If no qualifiers are given with subsequent commands, the system uses the qualifiers from the preceding command as the defaults. With the /M qualifier, the address is a 3-digit hex number (400 or above).

2.2.2 Examples

Examples

```
1.  >>> DEPOSIT/VE V12 0        !  Deposits zero into all 64 elements
                                !  of vector register V12.


2.  >>> DEPOSIT V6:2C/n:2  0    !  Deposits zero into V6 beginning at
                                !  element 2C (hex) and also in the next
                                !  two elements.


3.  >>> DEPOSIT VLR 1           !  Deposits one in the Vector Length
                                !  Register.


4.  >>> DEPOSIT/Q/P 200 FFFFFFFF45370201
                                !  Deposits FFFFFFFF45370201, a quadword
                                !  of data into physical memory at address
                                !  200.


5.  >>> DEPOSIT/M  440 0        !  Deposits zeros to vector indirect
                                !  register with address 440 (hex).
```

The DEPOSIT command directs data into the specified address. If you do not specify any address space or data size qualifiers, the defaults are the last address space or data size specified in a DEPOSIT or EXAMINE command. After processor initialization, the default address space is physical memory, the default data size is longword, and the default address is zero.

If the specified value is too large to fit in the data size, the console program ignores the command and issues an error message. If the specified value is smaller than the data size to be deposited, the console program fills the high order data positions with zeros. If you specify conflicting data sizes or address spaces, the console program ignores the command and issues an error message.

2.3 EXAMINE

The EXAMINE command displays the contents of a specified address. Various qualifiers provide access to the vector data registers (/VE), IPRs (/I), and vector indirect registers (/M). No qualifier is needed to examine VMR, VCR, and VLR.

2.3.1 Syntax and Qualifiers

Table 2–3: EXAMINE Command Qualifiers

Qualifier	Meaning
/B	Defines data size as a byte.
/G	Defines the address space as the general register set, R0 through R15.
/I	Defines the address space as the internal processor registers, accessed through MTPR and MFPR instructions.
/L	Defines data size as a longword; initial default.
/M	Defines the address space as a vector indirect register; accesses addresses 400 and higher.
/N:<count>	Defines the address space as the first of a range.[1] <count> is a required value with /N.
/P	Defines the address space as physical memory; initial default.
/Q	Defines data size as a quadword; initial default for vector registers (except for VCR and VLR).
/V	Defines the address space as virtual memory. All access and protection checking occur.[2]
/VE	Defines the address space as the vector register set.
/W	Defines data size as a word.

[1]The console examines the first address, then the specified number of succeeding addresses. Even if the address is '–', the succeeding addresses are at higher addresses; that is, the symbol specifies only the starting address, not the direction.

[2]If memory management has not been enabled, virtual addresses are equal to physical addresses. If access is not allowed to a program running with the current processor status longword (PSL), the console issues an error message. Virtual space deposits cause the PTE<M> bit to be set in the mapping PTE and force the processor write buffer to be flushed.

The EXAMINE command syntax is:

```
E[XAMINE] [/qualifier] [<address>]
```

where /qualifier is a value from Table 2–3, and <address> is a 1- to 8-digit hexadecimal value or one of the following:

- PSL, the processor status longword. You cannot use any address space qualifier with PSL.
- PC, the program counter. The address space is set to /G.
- SP, the stack pointer. The address space is set to /G.
- Rn, the general purpose register *n*. The register number is in decimal. The address space is set to /G.
- VCR, 7-bit Vector Count Register. No address qualifier is permitted.
- VLR, 7-bit Vector Length Register. No address qualifier is permitted.
- VMR, 64-bit Vector Mask Register. No address qualifier is permitted.
- V0–V15, vector registers. Elements of a vector register are specified V*n:mm*, where *n* is a decimal number 0–15 specifying the vector register, and *mm* is a hex number 0–3F specifying the element within the vector register. The address qualifier must be set to /VE.
- +, the location immediately following the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location plus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address plus one.
- –, the location immediately preceding the last location you referenced in an EXAMINE or DEPOSIT command. For physical and virtual memory, the referenced location is the last location minus the size of the reference (1 for byte, 2 for word, 4 for longword). For other address spaces, the address is the last referenced address minus one.
- *, the last location you referenced in an EXAMINE or DEPOSIT command.
- @, the location addressed by the last location you referenced in an EXAMINE or DEPOSIT command.

If no qualifiers are given with subsequent commands, the system uses the qualifiers from the preceding command as the defaults. With the /M qualifier, the address is a 3-digit hex number (400 or above).

2.3.2 Examples

Examples

1.
```
>>> EXAMINE VLR              !  Examines the Vector Length
                             !  Register.
    M 00000001 0E
```

2.
```
>>> EXAMINE/VE V0         !  Examines vector register V0; system
                          !  displays all 64 elements of register V0.

    VE V00:00  00000000 00000002      VE V00:01  00000000 00000002
    VE V00:02  00000000 00000002      VE V00:03  00000000 00000002
    VE V00:04  00000000 00000002      VE V00:05  00000000 00000002
    VE V00:06  00000000 00000002      VE V00:07  00000000 00000002
    VE V00:08  00000000 00000002      VE V00:09  00000000 00000002
    VE V00:0A  00000000 00000002      VE V00:0B  00000000 00000002
    VE V00:0C  00000000 00000002      VE V00:0D  00000000 00000002
    VE V00:0E  00000000 00000002      VE V00:0F  00000000 00000002
    VE V00:10  00000000 00000002      VE V00:11  00000000 00000002
    VE V00:12  00000000 00000002      VE V00:13  00000000 00000002
    VE V00:14  00000000 00000002      VE V00:15  00000000 00000002
    VE V00:16  00000000 00000002      VE V00:17  00000000 00000002
    VE V00:18  00000000 00000002      VE V00:19  00000000 00000002
    VE V00:1A  00000000 00000002      VE V00:1B  00000000 00000002
    VE V00:1C  00000000 00000002      VE V00:1D  00000000 00000002
    VE V00:1E  00000000 00000002      VE V00:1F  00000000 00000002
    VE V00:20  00000000 00000002      VE V00:21  00000000 00000002
    VE V00:22  00000000 00000002      VE V00:23  00000000 00000002
    VE V00:24  00000000 00000002      VE V00:25  00000000 00000002
    VE V00:26  00000000 00000002      VE V00:27  00000000 00000002
    VE V00:28  00000000 00000002      VE V00:29  00000000 00000002
    VE V00:2A  00000000 00000002      VE V00:2B  00000000 00000002
    VE V00:2C  00000000 00000002      VE V00:2D  00000000 00000002
    VE V00:2E  00000000 00000002      VE V00:2F  00000000 00000002
    VE V00:30  00000000 00000002      VE V00:31  00000000 00000002
    VE V00:32  00000000 00000002      VE V00:33  00000000 00000002
    VE V00:34  00000000 00000002      VE V00:35  00000000 00000002
    VE V00:36  00000000 00000002      VE V00:37  00000000 00000002
    VE V00:38  00000000 00000002      VE V00:39  00000000 00000002
    VE V00:3A  00000000 00000002      VE V00:3B  00000000 00000002
    VE V00:3C  00000000 00000002      VE V00:3D  00000000 00000002
    VE V00:3E  00000000 00000002      VE V00:3F  00000000 00000002
```

3.
```
>>> EXAMINE/Q/P 200           !  Examines the quadword in
                              !  physical memory at address 200.
```

```
4.  >>> EXAMINE/VE V12:2E          !  Examines element 2E (hex)
                                   !  (which is 41 decimal) of vector
                                   !  data register V12.


5.  >>> EXAMINE/M  440             !  Examines the vector indirect
                                   !  register at hex address 440.
      M 440 FFFFFFFF 00000000      !  /M is used to access vector
                                   !  indirect registers.
```

The system response to the EXAMINE command is in hexadecimal notation:

```
<address space identifier>  <address>  <data>
```

where <address space identifier> can be one of these values:

- P — Physical memory. When virtual memory is examined, the <address space identifier> is P and <address> is the translated physical address.
- G — General register.
- I — Internal processor register.
- M — Vector indirect register. This identifier is also returned when the PSL is examined.
- VE — Vector data register.

2.4 SET CPU Command

The SET CPU command allows you to specify a particular processor as the primary processor or designate its eligibility to become the primary processor. You can also disable a vector processor module.

2.4.1 Syntax and Qualifiers

Table 2–4: SET CPU Command Qualifiers

Qualifier	Meaning
/E[NABLED] /ALL	Processor is included in the system configuration and is eligible to become the boot processor. With the /ALL qualifier all processors are eligible to become the boot processor; initial default.
/NOE[NABLED]	Processor is immediately excluded from the system configuration; START, BOOT, and CONTINUE commands are ignored.
/NEX[T_PRIMARY]	Processor will be the primary (boot) processor at the next system reset.
/P[RIMARY] /ALL	Processor will be eligible to be selected as the primary (boot) processor at the next system reset. With the /ALL qualifier all processors are eligible to become the boot processor; initial default.
/NOP[RIMARY]	Processor will not be eligible to be selected as the primary (boot) processor at the next system reset.
/V[ECTOR_ENABLED]	Vector processor attached to the specified scalar processor is included in the system configuration and can be sent vector instructions; initial default.
/NOV[ECTOR_ENABLED]	Vector processor attached to the specified scalar processor is excluded from the system configuration.
None	Processor immediately becomes the new primary processor; the next system prompt comes from the new primary processor.

The SET CPU command syntax is:

```
SE[T] C[PU] [/qualifier] [<XMI-node>]
```

where <XMI-node> is the <REFERENCE>(XMI) node number of the processor to be affected. If you omit <XMI-node>, the system uses the current processor.

If you omit all qualifiers, the SET CPU command immediately causes the specified processor to become the primary processor. The console terminal is then connected to the new primary processor, and the next console prompt is generated by the designated processor.

If you use qualifiers, the SET CPU command changes the processor parameters that take effect at the next system reset. These qualifiers modify the EEPROM (if the lower key switch is set to Update) and take effect immediately:

- /ENABLE
- /NOENABLED
- /VECTOR_ENABLED
- /NOVECTOR_ENABLED

The /NEXT_PRIMARY qualifier acts the same as if you had issued a SET CPU/NOPRIMARY command for all other nodes. To undo /NEXT_ PRIMARY, you can issue the SET CPU/PRIMARY/ALL command.

The /NOVECTOR_ENABLED qualifier removes the vector processor from the system configuration. The scalar processor is not affected. The /VECTOR_ENABLED qualifier restores the vector processor to the configuration.

The effect of the SET CPU command qualifiers is shown on the BPD lines of the system self-test display (see Section 2.5).

NOTE: *For performance reasons, the scalar processor of a scalar/vector pair should not be made the primary processor when other scalar processors are in the system.*

2.4.2 Examples

Examples

```
1.  >>> SET CPU/NOVECTOR_ENABLED 4  ! The vector processor attached
                                    ! to the scalar processor at node 4
                                    ! is disabled.


2.  >>> SET CPU/VECTOR_ENABLED 4    ! The vector processor attached
                                    ! to the scalar processor at node 4
                                    ! is included in the system configur-
                                    ! ation.
```

Table 2–5: SET CPU Command Qualifiers' Effect After a System Reset

Qualifier	BPD Value at Next Reset[1]
/NEX[T_PRIMARY]	B for boot processor; must be chosen the boot processor at the next system reset. All other CPUs show as D.
/NOE[NABLED]	D for disable; processor is not included in the configuration.
/NOP[RIMARY]	D for disable; can be only a secondary processor.
/P[RIMARY]	B if selected as the boot processor; E if it is a secondary processor.
/NOV[ECTOR_ENABLED]	D for disable; vector processor is not included in the configuration.
None	B for boot processor.

[1]The key switch must be at Update when the SET CPU command is issued.

2.5 Sample Console Session

```
#123456789 0123456789 0123456789 01234567#   ❶

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #
                                            ❷
    A   A   .   .   M   M   M   .   .   M   V- -P   P   P       TYP
    o   o   .   .   +   +   +   .   .   +   +   +   +   +       STF
    .   .   .   .   .   .   .   .   .   .   E   E   E   B       BPD
    .   .   .   .   .   .   .   .   .   .   +   +   +   +       ETF
    .   .   .   .   .   .   .   .   .   .   E   E   E   B       BPD

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -
.   .   .   .   .   .   .   .   .   +   .   +   .   +   +   .   XBI E +

    .   .   .   .  A4   A3  A2  .   .   A1  .   .   .   .       ILV
    .   .   .   .  32   32  32  .   .   32  .   .   .   .       128Mb

ROM0 = V2.00  ROM1 = V2.00  EEPROM = 2.00/2.00  SN = SG01234567

>>> SHOW CPU  ❸
  Current Primary: 1
  /NOENABLED-
  /NOVECTOR_ENABLED-
  /NOPRIMARY-

>>> SHOW CONFIGURATION  ❹

     Type           Rev
 1+  KA64A   (8082) 0007
 2+  KA64A   (8082) 0007
 3+  KA64A   (8082) 0007
 4+  FV64A   (0000) 0001  ❺
 5+  MS62A   (4001) 0002
 8+  MS62A   (4001) 0002
 9+  MS62A   (4001) 0002
 A+  MS62A   (4001) 0002
 D-  DWMBA/A (2001) 0002
 E+  DWMBA/A (2001) 0002

 XBI D
 XBI E
 1+  DWMBA/B (2107) 0007
 3+  DRB32   (0101) 0001
 4+  KDB50   (010E) 0F1C
 6+  TBK70   (410B) 0307

>>> SET CPU 3  ❻

>>> EX/M 440   ❼
   M 440 FFFFFFFF 00000000
```

```
>>> SET CPU/NOVECTOR  ❽
>>> SHOW CPU          ❾
  Current Primary: 3
  /NOENABLED-
  /NOVECTOR_ENABLED-3
  /NOPRIMARY-
>>> SET CPU/VEC  ❿
>>> SET CPU 1  ⓫
```

Sections of the sample console session flagged by the numbered callouts are explained below.

❶ At power-up, the system performs self-test and displays the results. Note that the number of tests displayed in the progress trace differs if a vector module is attached to a CPU in node 1. See Appendix A for a detailed explanation of self-test.

❷ The TYP line in the sample self-test display indicates that a vector processor is at node 4, and the dashes show that it is attached to the scalar processor at node 3.

❸ Enter a SHOW CPU command. Information is given about the current primary processor and any attached vector processor. If a vector processor were attached to the CPU at node 1, the response to the SHOW CPU command would tell if the vector processor were enabled or disabled (from the SET CPU command).

❹ Enter a SHOW CONFIGURATION command to show the hardware configuration. The system response indicates device node numbers, self-test status, device types, and contents of the revision register of the devices.

❺ A vector processor, FV64A, is at node 4. A null device type appears in the parentheses. The FV64A is an XMI module, but it has no device type, since it functions as a coprocessor.

❻ Make the scalar processor with the attached vector processor the primary processor by issuing the SET CPU 3 command.

❼ The EXAMINE/M console command provides access to vector indirect registers. The register being read is that of the primary processor.

❽ The SET CPU command can be used to disable a vector processor.

❾ The vector processor attached to the CPU at node 3 has been disabled.

❿ Issue SET CPU/VECTOR to return the vector processor to the configuration.

⓫ Issue another SET CPU command to make the processor at node 1 the boot processor.

Appendix A
Self-Test

Self-test results are displayed on the console terminal and are reported by module LEDs. Example A–1 is a sample self-test display for a VAX 6000 Model 400 system without a vector processor; the example deliberately includes some failures to illustrate the type of information reported. Example A–2 shows a sample self-test for a Model 400 system with two vector processors.

Figure A–1 shows the <REFERENCE>(xrp) LEDs after self-test. If the <REFERENCE>(xrp) has an attached vector module, the red LEDs on the <REFERENCE>(xrp) are also used to find the failing test number for the vector module. The vector module has a yellow self-test LED that lights when that module passes self-test.

For a more detailed description of self-test, see your system *Owner's Manual* Chapter 6.

Example A–1: Sample Self-Test Results, Scalar Processors Only

```
#123456789 0123456789 0123456789 01234567#  ❶

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE # ❷

    A   A   .   .   M   M   M   M   .   .   P   P   P   P       TYP    ❸
    o   o   .   .   +   +   +   +   .   .   +   +   -   +       STF    ❹
    .   .   .   .   .   .   .   .   .   .   E   B   E   D       BPD    ❺
    .   .   .   .   .   .   .   .   .   .   +   -   -   +       ETF    ❻
    .   .   .   .   .   .   .   .   .   .   B   E   E   D       BPD    ❼

.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   XBI D -❽
.   .   .   .   .   .   .   .   .   +   .   +   .   -   +   .   XBI E +

    .   .   .   .  B2   B1  A2  A1  .   .   .   .   .   .       ILV    ❾
    .   .   .   .  32   32  32  32  .   .   .   .   .   .      128Mb   ❿

ROM0 = V2.00  ROM1 = V2.00 ⓫ EEPROM = 2.00/2.01 ⓬ SN = SG01234567 ⓭
>>>
```

❶ The progress trace. This line appears when slot 1 holds a <REFERENCE>(xrp) module. The <REFERENCE>(xrp) in slot 1 passed all 37 tests in self-test. (Note that the progress trace differs in a system when a vector processor is attached to the CPU in slot 1; see Example A–2).

❷ Identifies the node number (NODE #).

Lines 3 through 7 refer to XMI node numbers; the XBI lines refer to VAXBI node numbers.

❸ Identifies the module type (TYP).

P = processor
M = memory
A = adapter

❹ Gives self-test failure results (STF).

\+ = passed
– = failed
o = not tested as part of the initial power-up test

❺ Shows boot processor designation (BPD).

E = eligible to be boot processor
D = ineligible to be boot processor

B = designated as boot processor

❻ Gives extended CPU/memory tests failure results (ETF). Same interpretation as STF.

❼ Shows the second boot processor designation, which may be different from that on the first BPD line.

❽ Shows DWMBA test results, node number, and self-test results of the VAXBI nodes (XBI). The + or – at the right means that the DWMBA passed or failed when tested by the boot processor. If the DWMBA passed, a + or – corresponding to each VAXBI node indicates whether that node passed or failed its own self-test.

❾ Displays the memory array membership in interleave sets (ILV). Each letter denotes a different interleave set.

❿ Gives each memory array size and the total working memory size (Mb).

⓫ Shows the version number of the boot processor's ROMs (ROM0 and ROM1).

⓬ Gives the version number and revision number of the boot processor's EEPROM. The first number is the base revision of the EEPROM, which rarely changes. The second number is the revision of console and diagnostic patches applied to the EEPROM. This number increments with every patch operation.

⓭ Lists the serial number of the system (SN).

The self-test display in Example A–2 shows a system with two vector processors.

Example A–2: Sample Self-Test Results with Vector Processors

```
#123456789 0123456789 0123456789 0123456789 0123456789 #     ❶

F   E   D   C   B   A   9   8   7   6   5   4   3   2   1   0   NODE #

    A   A   .   .   M   M   .   .   M   V- -P   M   V- -P       TYP ❷
    o   o   .   .   +   +   .   .   +   +   +   +   +   +       STF
    .   .   .   .   .   .   .   .   .   E   E   .   E   B       BPD ❸
    .   .   .   .   .   .   .   .   .   +   +   .   +   +       ETF ❹
    .   .   .   .   .   .   .   .   .   E   E   .   E   B       BPD ❸

.   .   .   .   .   .   .   .   .   +   +   +   +   .   +   .   XBI D +
.   .   .   .   .   .   .   .   .   +   .   +   .   +   +   .   XBI E +

    .   .   .   .  A4   A3   .   .   A2   .   .   A1   .   .       ILV
    .   .   .   .  32   32   .   .   32   .   .   32   .   .       128Mb

ROM0 = V2.00  ROM1 = V2.00  EEPROM = 2.00/2.00  SN = SG01234567 ❺
>>>
```

❶ The progress trace indicates that the processor in slot 1 passed all 49 tests that comprise self-test for CPUs with vector processors. This progress trace differs from that shown in Example A–1. In a system where the CPU in slot 1 has no attached vector processor, self-test for that CPU consists of 37 tests.

❷ Vector processors (V) are in slots 2 and 5. The dashed lines indicate that they are attached to the scalar processors to their right.

❸ The boot processor is determined and is indicated by B. The E for the other scalar processor indicates that it is eligible to be boot processor.

The E for the vector processor means that it is enabled. A vector processor can be disabled with the SET CPU n /NOVECTOR_ ENABLED console command. If this command were issued, a D would be on the BPD lines to indicate that the specified vector processor has been disabled.

❹ All processors pass the extended test.

❺ Version 2 (or greater) of the ROMs and EEPROM are required for vector processing support.

Figure A–1: <REFERENCE>(XRP) LEDs After Self-Test

NOTE: *Interpretation of small red LEDs: ON is a zero, and OFF is a one.*

Appendix B
Vector Processor Error Messages

This appendix lists the error messages associated with the vector module. See the *VAX 6000–400 Owner's Manual* Appendix B for a listing of other error messages.

Table B–1: Vector Error Messages

Error Message	Meaning
?78 Vector module configuration error at node *n*	The console detected a vector module configuration error. Problem can be that the vector node number is not one greater than the scalar CPU or that the module to the left of a vector processor is not a memory module.
?79 Vector synchronization error.	The console could not synchronize with the vector processor on a console entry. The Busy bit in the Vector Processor Status Register remained set after a timeout, or a vector processor error occurred.
?7A No vector module associated with CPU at specified node.	No vector module is in the slot to the left of the specified CPU, or the VIB cable either is not attached or is bad.
?7B An error occurred while accessing the vector module.	Attempt to access VCR, VLR, or VMR registers failed.
?7D Vector module is disabled—check KA64A revision at XMI node *n*	The vector module is attached to a <REFERENCE>(xrp) module that is not at the revision level required.

Appendix C
Vector Module Registers

The vector module registers consist of the following:

- Internal processor registers (IPRs) (see Table C–1)
- Vector indirect registers (see Table C–2)
- Vector Length, Vector Count, and Vector Mask control registers

This appendix explains how to access the registers and then shows the registers. See your *System Technical User's Guide* for complete descriptions of the registers.

C.1 Console Commands to Access Registers

From the console, the EXAMINE and DEPOSIT commands are used to read and write the IPRs and the vector indirect registers. The vector data registers can also be accessed from the console. The qualifiers differ:

- /I — to read and write the IPRs
- /M — to read and write the vector indirect registers, except for the 16 vector data registers
- /VE — to read and write the vector data registers

From the console, the Vector Length, Vector Count, and Vector Mask control registers can be specified as VLR, VCR, and VMR after DEPOSIT and EXAMINE commands with no qualifiers. VLR and VCR are 7-bit registers (Figure C–1), and VMR is a 64-bit register (Figure C–2).

Figure C–1: Vector Length (VLR) and Vector Count (VCR) Registers

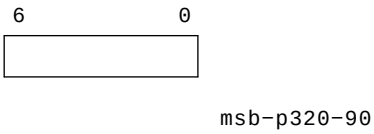


Figure C–2: Vector Mask Register (VMR)

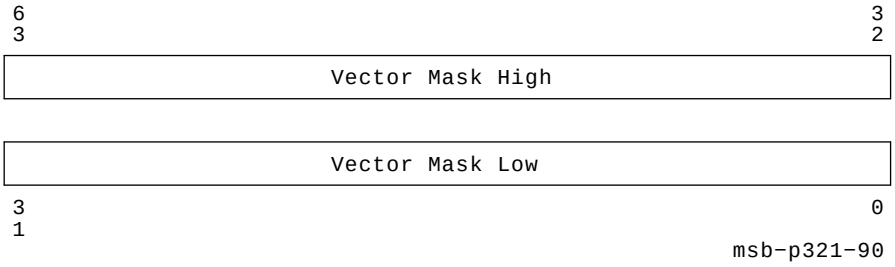


Table C–1: Internal Processor Registers

Register	Mnemonic	Address decimal (hex)	Type	Class
Vector Copy—P0 Base	P0BR	8 (8)	WO	1
Vector Copy—P0 Length	P0LR	9 (9)	WO	1
Vector Copy—P1 Base	P1BR	10 (A)	WO	1
Vector Copy—P1 Length	P1LR	11 (B)	WO	1
Vector Copy—System Base	SBR	12 (C)	WO	1
Vector Copy—System Length	SLR	13 (D)	WO	1
Accelerator Control and Status	ACCS	40 (28)	R/W	2 I
Vector Copy—Memory Management Enable	MAPEN	56 (38)	WO	1
Vector Copy—Translation Buffer Invalidate All	TBIA	57 (39)	WO	1
Vector Copy—Translation Buffer Invalidate Single	TBIS	58 (3A)	WO	1
Vector Interface Error Status	VINTSR	123 (7B)	R/W	2
Vector Processor Status	VPSR	144 (90)	R/W	3
Vector Arithmetic Exception	VAER	145 (91)	RO	3
Vector Memory Activity Check	VMAC	146 (92)	RO	3
Vector Translation Buffer Invalidate All	VTBIA	147 (93)	WO	3
Vector Indirect Register Address	VIADR	157 (9D)	R/W	3
Vector Indirect Data Low	VIDLO	158 (9E)	R/W	3

Key to Types:

RO–Read only, WO–Write only, R/W–Read/write

Key to Classes:

1–Implemented by <REFERENCE>(XRP) CPU with a copy in the <REFERENCE>(xrv) vector module.
2–Implemented by <REFERENCE>(XRP) CPU module.
3–Implemented by <REFERENCE>(XRV) vector module.
I–Initialized on <REFERENCE>(XRP) reset (power-up, system reset, and node reset).

Table C–1 (Cont.): Internal Processor Registers

Register	Mnemonic	Address decimal (hex)	Type	Class
Vector Indirect Data High	VIDHI	159 (9F)	R/W	3

Table C–2: FV64A Registers—Vector Indirect Registers

Register	Mnemonic	Register Field Address (hex)	Type
Vector Register 0	VREG0	000–03F	R/W
Vector Register 1	VREG1	040–07F	R/W
Vector Register 2	VREG2	080–0BF	R/W
Vector Register 3	VREG3	0C0–0FF	R/W
Vector Register 4	VREG4	100–13F	R/W
Vector Register 5	VREG5	140–17F	R/W
Vector Register 6	VREG6	180–1BF	R/W
Vector Register 7	VREG7	1C0–1FF	R/W
Vector Register 8	VREG8	200–23F	R/W
Vector Register 9	VREG9	240–27F	R/W
Vector Register 10	VREG10	280–2BF	R/W
Vector Register 11	VREG11	2C0–2FF	R/W
Vector Register 12	VREG12	300–33F	R/W
Vector Register 13	VREG13	340–37F	R/W
Vector Register 14	VREG14	380–3BF	R/W
Vector Register 15	VREG15	3C0–3FF	R/W
Arithmetic Instruction	ALU_OP	440*	R/BW
Scalar Operand Low	ALU_SCOP_LO	448	R/BW
Scalar Operand High	ALU_SCOP_HI	44C	R/BW
Vector Mask Low	ALU_MASK_LO	450	BR/BW
Vector Mask High	ALU_MASK_HI	451	BR/BW
Exception Summary	ALU_EXC	454	R/BW
Diagnostic Control	ALU_DIAG_CTL	45C	R/BW
Current ALU Instruction	VCTL_CALU	480	R/W
Deferred ALU Instruction	VCTL_DALU	481	R/W

*Addresses from 400–45F in this column specify the address of Verse chip 0; addresses for Verse chips 1, 2, and 3 are found by adding 1, 2, and 3 to the address given. A read must specify each Verse chip by its own address; a write to the address given in the table (for Verse chip 0) is broadcast to all Verse chips.

Table C–2 (Cont.): FV64A Registers—Vector Indirect Registers

Register	Mnemonic	Register Field Address (hex)	Type
Current ALU Operand Low	VCTL_COP_LO	482	R/W
Current ALU Operand High	VCTL_COP_HI	483	R/W
Deferred ALU Operand Low	VCTL_DOP_LO	484	R/W
Deferred ALU Operand High	VCTL_DOP_HI	485	R/W
Load/Store Instruction	VCTL_LDST	486	R/W
Load/Store Stride	VCTL_STRIDE	487	R/W
Illegal Instruction	VCTL_ILL	488	R/W
Vector Controller Status	VCTL_CSR	489	R/W
Module Revision	MOD_REV	48A	R
Vector Copy—P0 Base	LSX_P0BR	500	WO
Vector Copy—P0 Length	LSX_P0LR	501	WO
Vector Copy—P1 Base	LSX_P1BR	502	WO
Vector Copy—P1 Length	LSX_P1LR	503	WO
Vector Copy—System Base	LSX_SBR	504	WO
Vector Copy—System Length	LSX_SLR	505	R/W
Load/Store Exception	LSX_EXC	508	RO
Translation Buffer Control	LSX_TBCSR	509	WO
Vector Copy—Memory Management Enable	LSX_MAPEN	50A	WO
Vector Copy—Translation Buffer Invalidate All	LSX_TBIA	50B	WO
Vector Copy—Translation Buffer Invalidate Single	LSX_TBIS	50C	WO
Vector Mask Low	LSX_MASKLO	510	WO
Vector Mask High	LSX_MASKHI	511	WO
Load/Store Stride	LSX_STRIDE	512	WO
Load/Store Instruction	LSX_INST	513	WO
Cache Control	LSX_CCSR	520	R/W

Table C–2 (Cont.): FV64A Registers—Vector Indirect Registers

Register	Mnemonic	Register Field Address (hex)	Type
Translation Buffer Tag	LSX_TBTAG	530	R/W
Translation Buffer PTE	LSX_PTE	531	R/W

C.2 <REFERENCE>(XRP) IPRs Related to the Vector Module

Figure C–3: Vector Interface Error Status Register (VINTSR) IPR123 (7B hex)

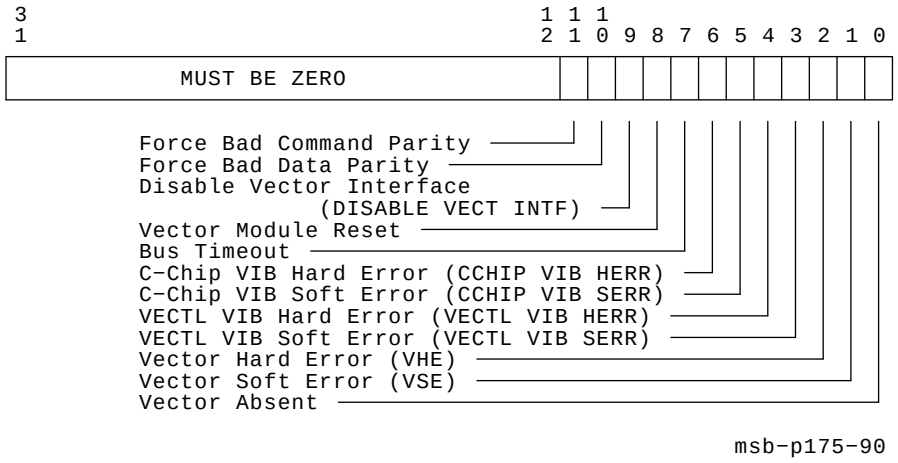
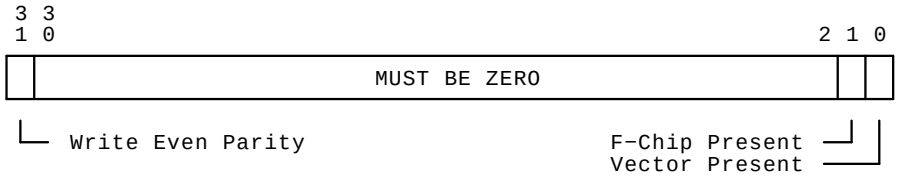


Figure C–4: Accelerator Control and Status Register (ACCS) IPR40 (28 hex)



C.3 <REFERENCE>(xrv) Internal Processor Registers

Figure C–5: Vector Processor Status Register (VPSR) IPR144 (90 hex)

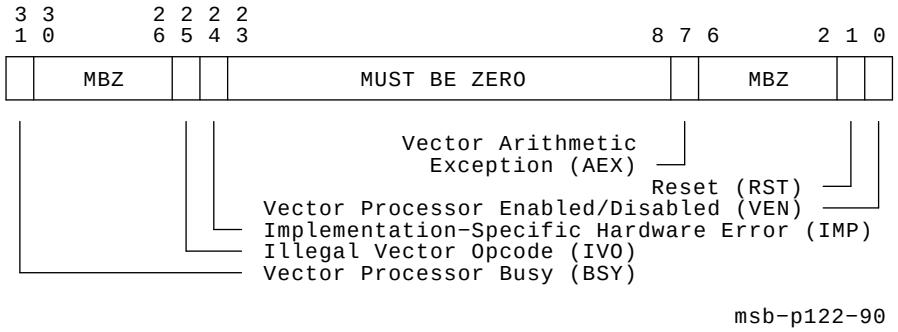


Figure C–6: Vector Arithmetic Exception Register (VAER) IPR145 (91 hex)

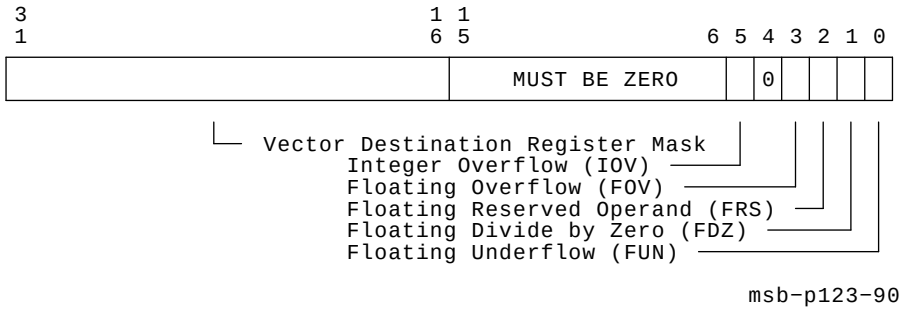
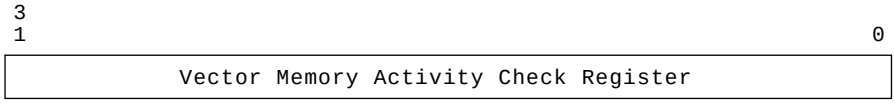
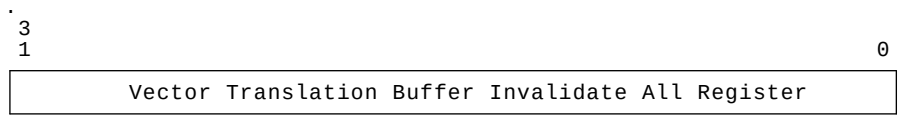


Figure C–7: Vector Memory Activity Check Register (VMAC) IPR146 (92 hex)



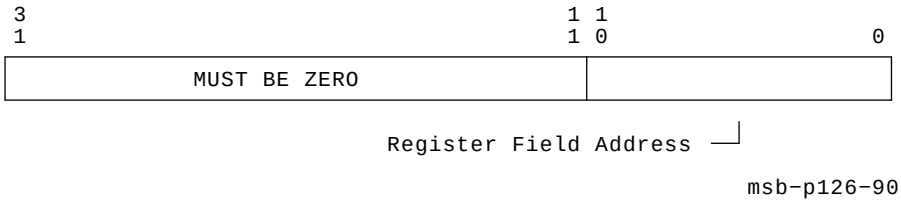
msb–p124–90

Figure C–8: Vector Translation Buffer Invalidate All Register (VTBIA) IPR147 (93 hex)



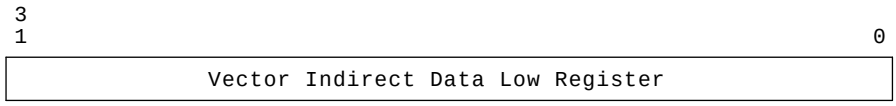
msb–p125–90

Figure C–9: Vector Indirect Address Register (VIADR) IPR157 (9D hex)



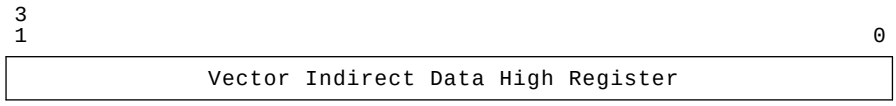
msb–p126–90

Figure C–10: Vector Indirect Data Low Register (VIDLO) IPR158 (9E hex)



msb–p127–90

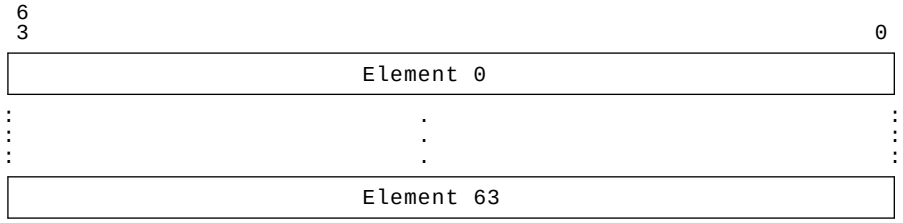
Figure C–11: Vector Indirect Data High Register (VIDHI) IPR159 (9F hex)



msb–p128–90

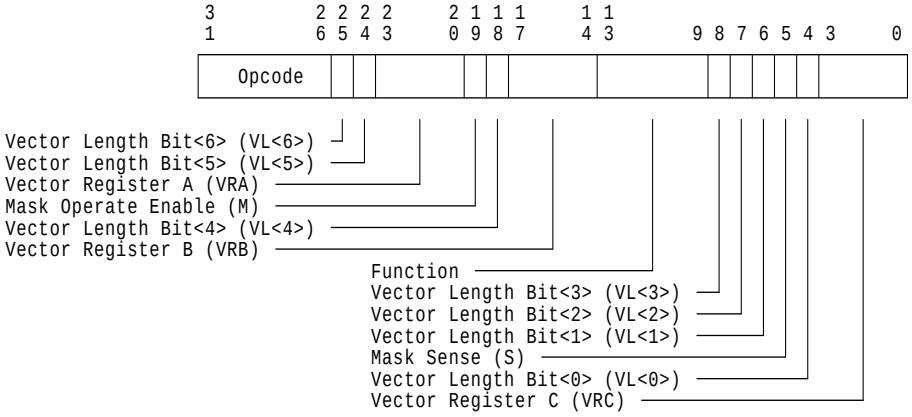
C.4 <REFERENCE>(xrv) Registers — Vector Indirect Registers

Figure C–12: Vector Register *n* (VREG*n*) 000—3FF, 16 registers



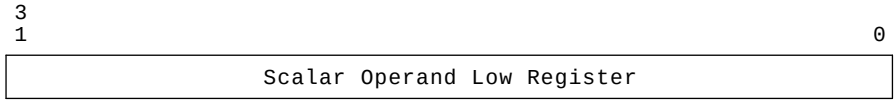
msb–p138–90

Figure C–13: Arithmetic Exception Register (ALU_OP) 440 hex



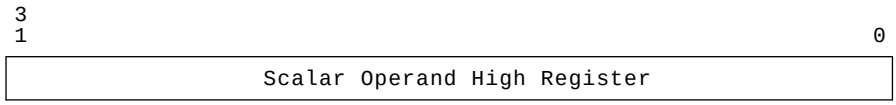
msb–p139r–90

Figure C–14: Scalar Operand Low Register (ALU_SCOP_LO) 448 hex



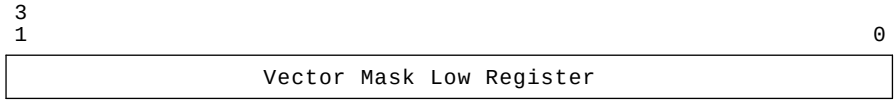
msb–p140–90

Figure C–15: Scalar Operand High Register (ALU_SCOP_HI) 44C hex



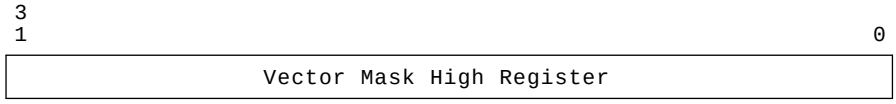
msb–p141–90

Figure C–16: Vector Mask Low Register (ALU_MASK_LO) 450 hex



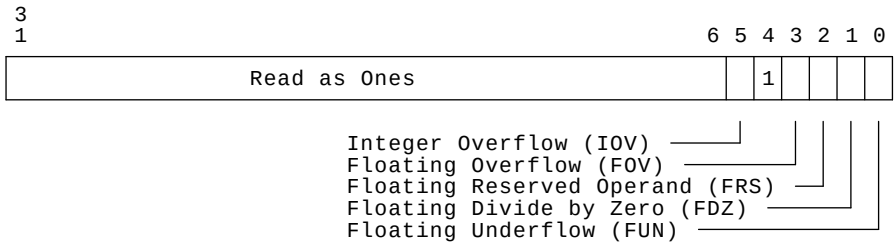
msb–p142–90

Figure C–17: Vector Mask High Register (ALU_MASK_HI) 451 hex



msb–p143–90

Figure C–18: Exception Summary Register (ALU_EXC) 454 hex



msb–p144–90

Figure C–19: Diagnostic Control Register (ALU_DIAG_CTL) 45C hex

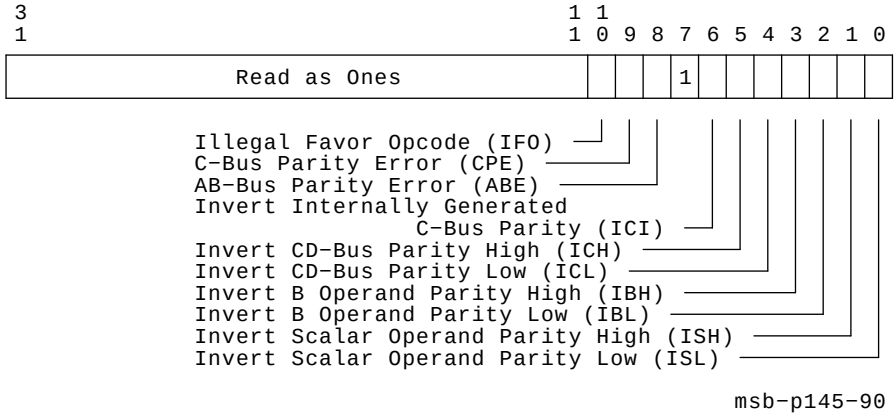


Figure C–20: Current ALU Instruction Register (VCTL_CALU) 480 hex

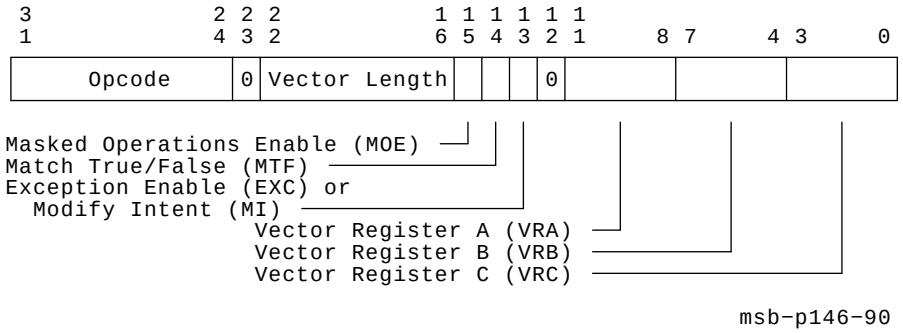
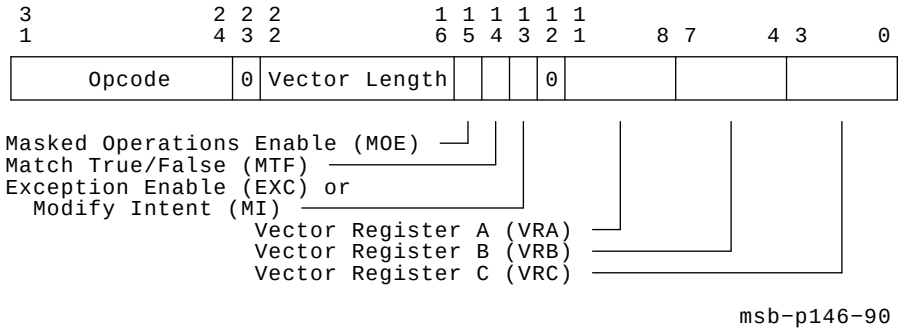
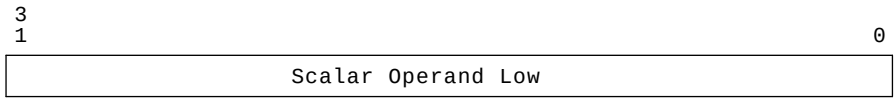


Figure C–21: Deferred ALU Instruction Register (VCTL_DALU) 481 hex



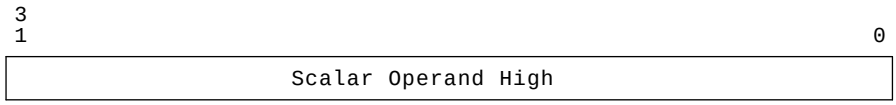
msb–p146–90

Figure C–22: Current ALU Operand Low Register (VCTL_COP_LO) 482 hex



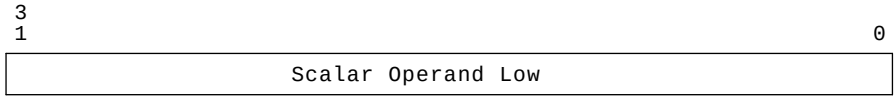
msb–p147–90

Figure C–23: Current ALU Operand High Register (VCTL_COP_HI) 483 hex



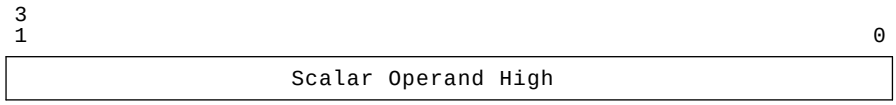
msb–p148–90

Figure C–24: Deferred ALU Operand Low Register (VCTL_DOP_LO) 484 hex



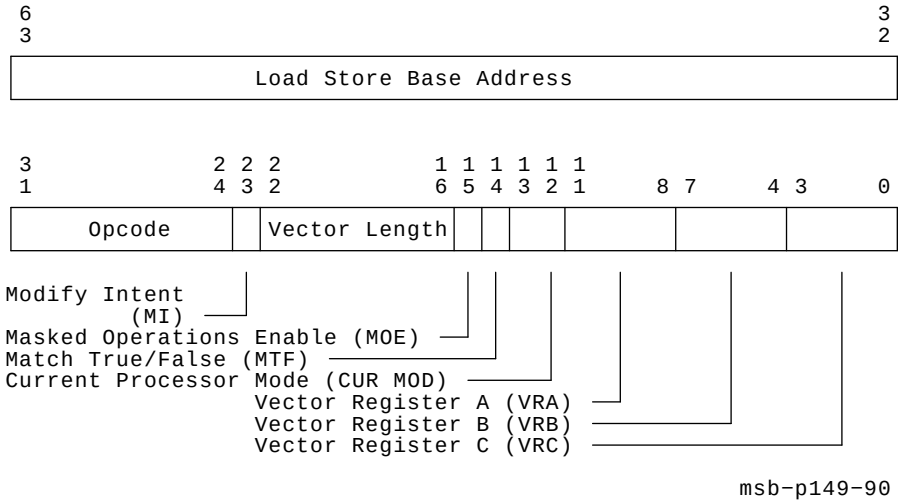
msb–p147–90

Figure C–25: Deferred ALU Operand High Register (VCTL_DOP_HI) 485 hex



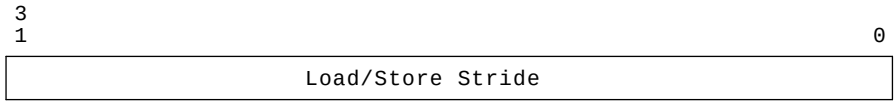
msb–p148–90

Figure C–26: Load/Store Instruction Register (VCTL_LDST) 486 hex



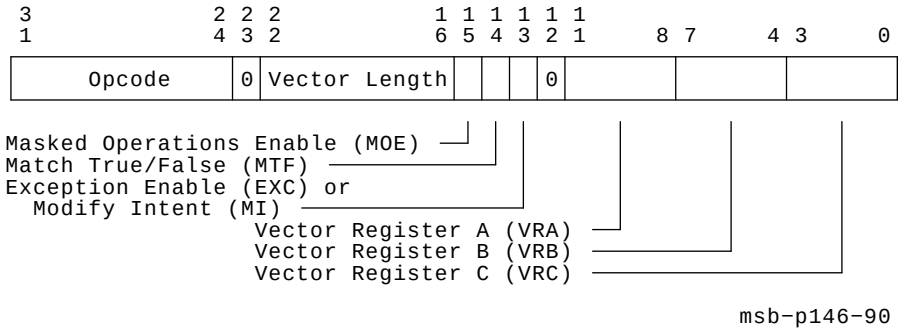
msb–p149–90

Figure C–27: Load/Store Stride Register (VCTL_STRIDE) 487 hex



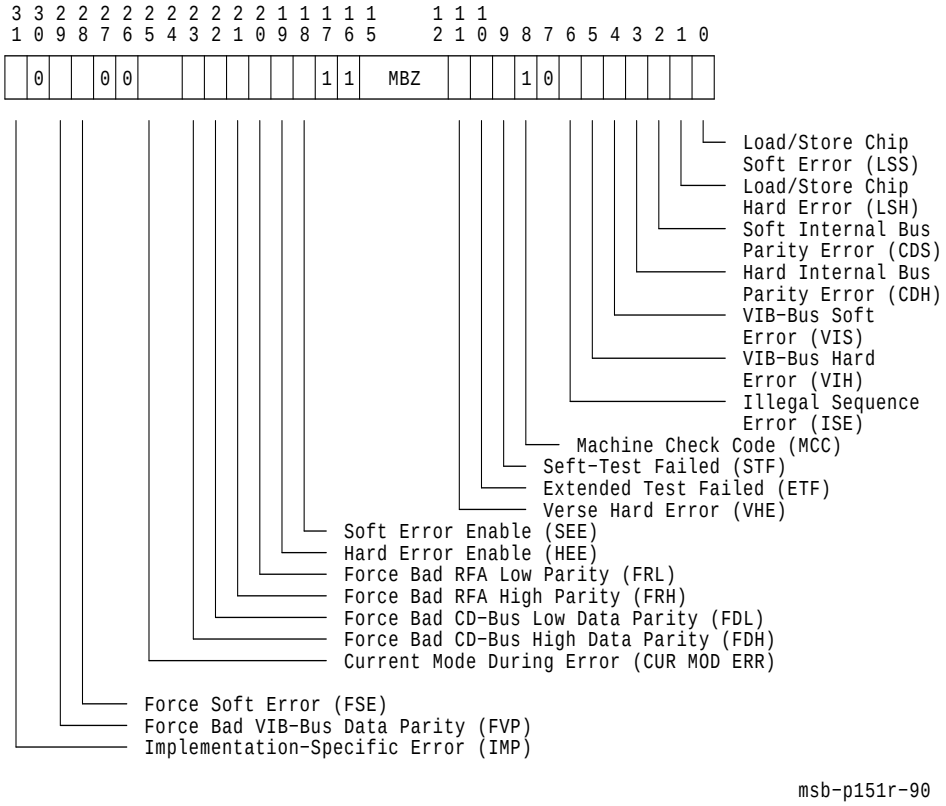
msb–p150–90

Figure C–28: Illegal Instruction (VCTL_ILL) 488 hex



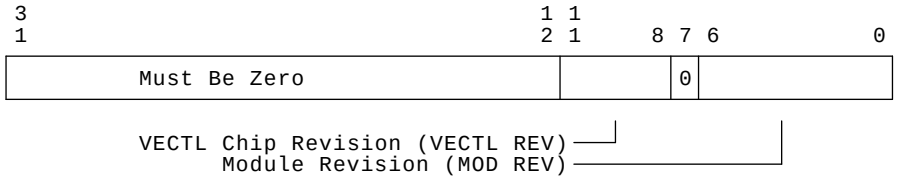
msb–p146–90

Figure C–29: Vector Controller Status (VCTL_CSR) 489 hex



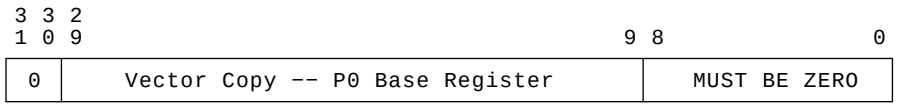
msb–p151r–90

Figure C–30: Module Revision (MOD_REV) 48A hex



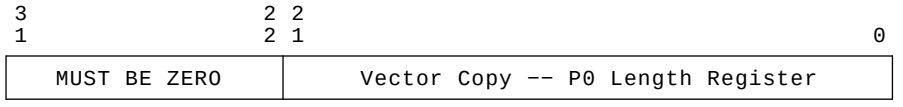
msb–p174–90

Figure C–31: P0 Base Register (LSX_P0BR) 500 hex



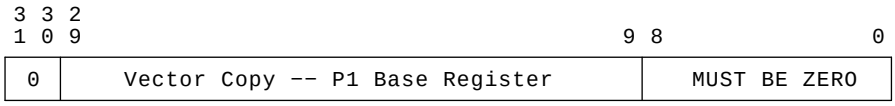
msb–p129–90

Figure C–32: P0 Length Register (LSX_P0LR) 501 hex



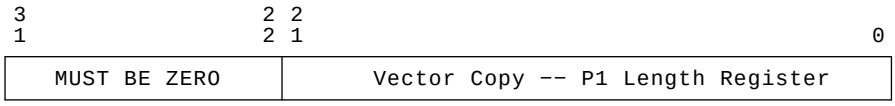
msb–p130–90

Figure C–33: P1 Base Register (LSX_P1BR) 502 hex



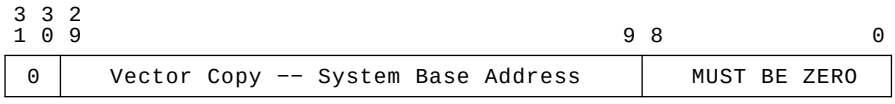
msb–p131–90

Figure C–34: P1 Length Register (LSX_P1LR) 503 hex



msb–p132–90

Figure C–35: System Base Register (LSX_SBR) 504 hex



msb–p133–90

Figure C–36: System Length Register (LSX_SLR) 505 hex

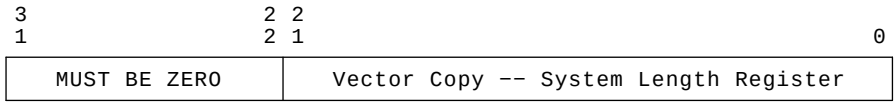


Figure C–37: Load/Store Exception Register (LSX_EXC) 508 hex

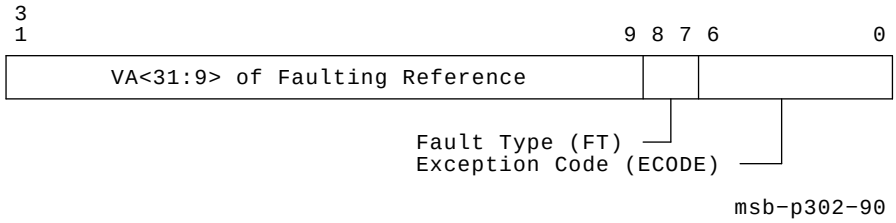


Figure C–38: Translation Buffer Control Register (LSX_TBCSR) 509 hex

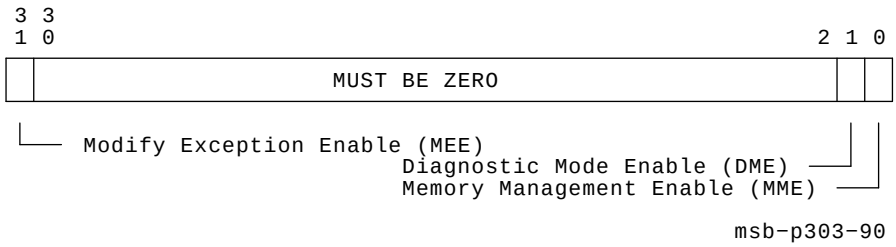
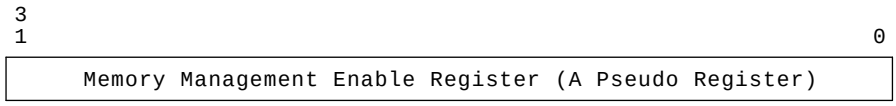
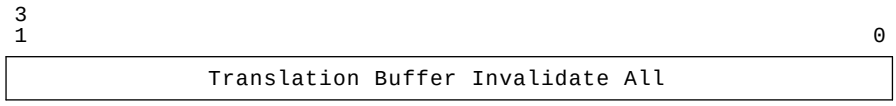


Figure C–39: Memory Management Enable (LSX_MAPEN) 50A hex



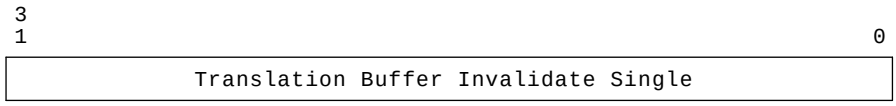
msb–p135–90

Figure C–40: Translation Buffer Invalidate All Register (LSX_TBIA) 50B hex



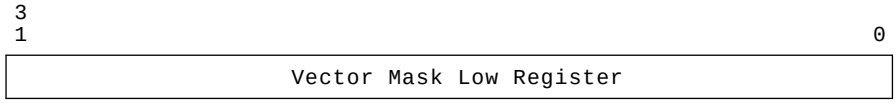
msb–p136–90

Figure C–41: Translation Buffer Invalidate Single Register (LSX_TBIS) 5C hex



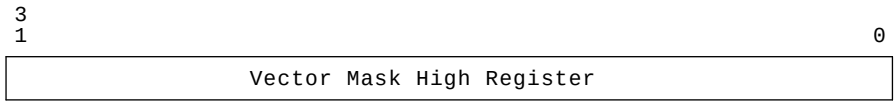
msb–p137–90

Figure C–42: Vector Mask Low Register (LSX_MASKLO) 510 hex



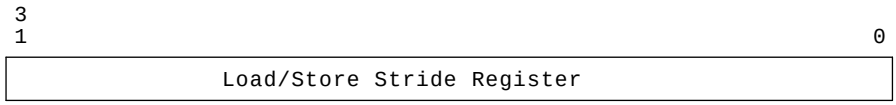
msb–p304–90

Figure C–43: Vector Mask High Register (LSX_MASKHI) 511 hex



msb–p305–90

Figure C–44: Load/Store Stride Register (LSX_STRIDE) 512 hex



msb–p306–90

Figure C–45: Load/Store Instruction Register (LSX_INST) 513 hex

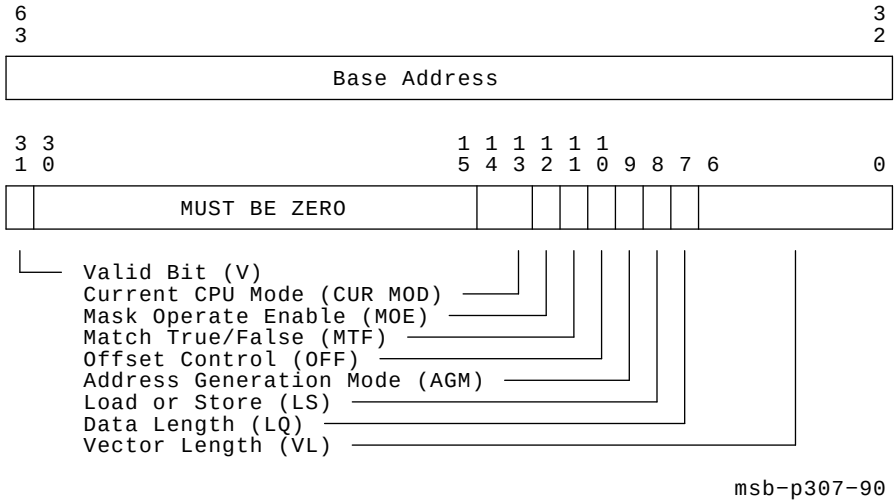


Figure C–46: Cache Control Register (LSX_CCSR) 520 hex

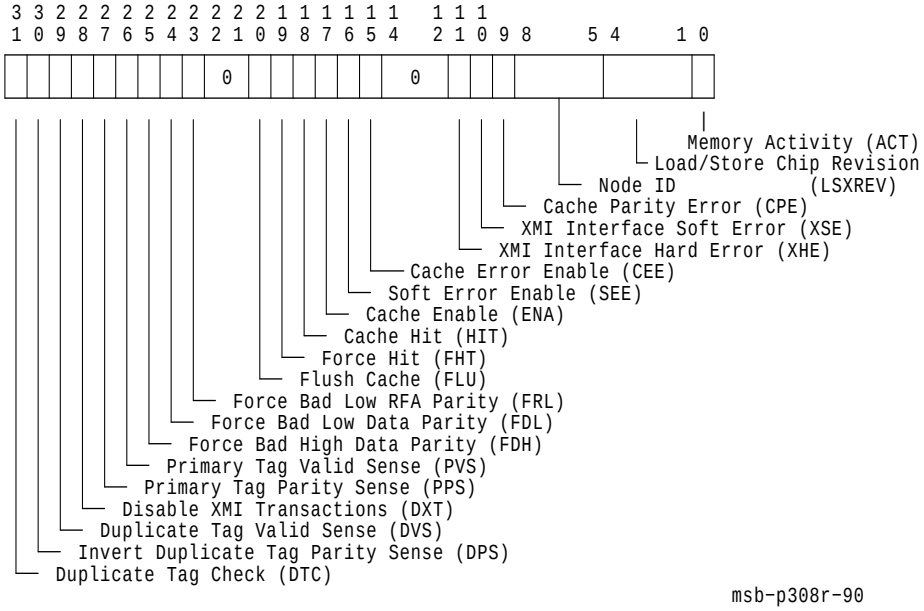
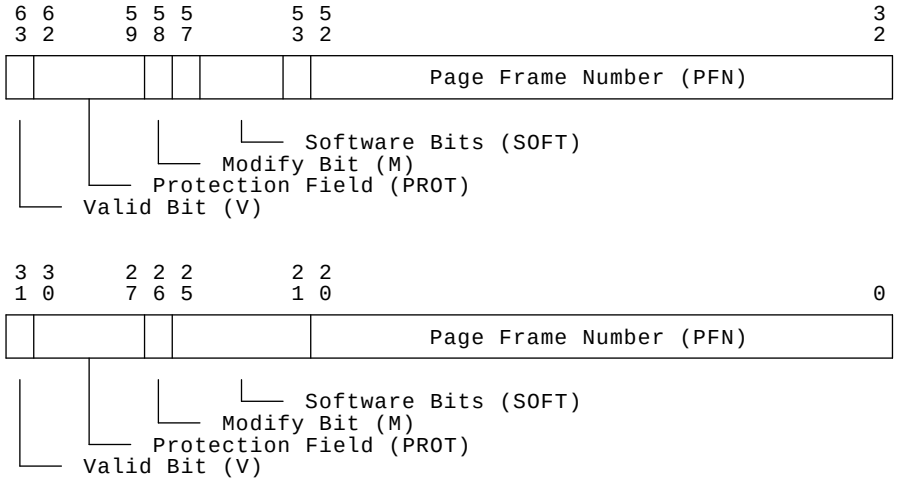


Figure C–47: Translation Buffer Tag Register (LSX_TBTAG) 530 hex

Figure C–48: Translation Buffer PTE Register (LSX_PTE) 531 hex



msb–p310–90

Index

M

P

R

S

T

V