This document discusses the features, capabilities, and configuration requirements of Tsi148. It is intended for software engineers who are designing system interconnect applications with Tsi148 and require programming information about the device.

Information in this document is preliminary.
**Trademarks**

TUNDRA is a registered trademark of Tundra Semiconductor Corporation (Canada, U.S., and U.K.). TUNDRA, the Tundra logo, Tsi148, and Silicon Behind the Network, are trademarks of Tundra Semiconductor Corporation. All other registered and unregistered marks (including trademarks, service marks and logos) are the property of their respective owners. The absence of a mark identifier is not a representation that a particular product name is not a mark.

**Copyright**

Copyright © May 2004 Tundra Semiconductor Corporation. All rights reserved.
Published in Canada

This document contains information which is proprietary to Tundra and may be used for non-commercial purposes within your organization in support of Tundra products. No other use or transmission of all or any part of this document is permitted without written permission from Tundra, and must include all copyright and other proprietary notices. Use or transmission of all or any part of this document in violation of any applicable Canadian or other legislation is hereby expressly prohibited.

User obtains no rights in the information or in any product, process, technology or trademark which it includes or describes, and is expressly prohibited from modifying the information or creating derivative works without the express written consent of Tundra.

**Disclaimer**

Tundra assumes no responsibility for the accuracy or completeness of the information presented which is subject to change without notice. In no event will Tundra be liable for any direct, indirect, special, incidental or consequential damages, including lost profits, lost business or lost data, resulting from the use of or reliance upon the information, whether or not Tundra has been advised of the possibility of such damages.

Mention of non-Tundra products or services is for information purposes only and constitutes neither an endorsement nor a recommendation.
Corporate Profile

About Tundra

Tundra Semiconductor Corporation (Tundra)(TSX:TUN) designs, develops, and markets System Interconnect for use by the world's leading communications infrastructure equipment and storage companies. Tundra System Interconnect is a vital communications technology that enables customers to connect critical system components while compressing development cycles and maximizing performance. Tundra products offer value to a range of applications, including wireless infrastructure, storage, networking access, military applications, and industrial automation. Tundra headquarters are located in Ottawa, Ontario, Canada. The Company also has a design center in South Portland, Maine, United States and a sales office in Maidenhead, U.K. Tundra sells its products worldwide through a network of direct sales personnel, independent distributors and manufacturing representatives. Tundra employs about 200 employees worldwide.

Greater Demand, Greater Opportunity

The ever-increasing demand for bandwidth, scalability, and reliability, driven by the growth of the Internet, has created unprecedented performance challenges for interconnect technologies. Regardless of whether it is a high-speed router, a next-generation wireless base station, or a storage sub system, Tundra provides the System Interconnect to meet the bandwidth demands for greater performance throughout the communications and data pipeline.

Tundra System Interconnect

Tundra uses the term System Interconnect to refer to the technology used to connect all the components and sub-systems in almost any embedded system. This concept applies to the interfacing of functional elements (CPU, memory, I/O complexes) within a single-board system, and the interconnection of multiple boards in a larger system.

Advanced communications networks need advanced System Interconnect. It is a vital enabling technology for the networked world. Tundra System Interconnect provides the latest interface and throughput features, which enable communications infrastructure vendors to design and build more powerful, faster equipment in shorter timeframes.
Partnerships
Fundamental to the success of Tundra is its partnerships with leading technology companies, including IBM, Intel and Motorola. As a result of these alliances, Tundra devices complement our partners’ products, and greatly influence the design of customers’ architecture. Customers are changing their designs to incorporate Tundra products. This is evidence of Tundra’s commitment to be a significant part of its customers’ success.

Customers
The world's leading communications infrastructure and storage vendors use Tundra Semiconductor products. These vendors include Cisco Systems, Motorola, Siemens, Nortel Networks, Lucent Technologies, Nokia, Ericsson, Alcatel, and Hewlett-Packard.

The Tundra design philosophy is one in which a number of strategic customers are invited to participate in the definition, design, test, and early silicon phases of product development. Close working relationships with customers and clear product roadmaps ensure that Tundra can anticipate and meet the future directions and needs of communications systems designers and manufacturers.

Support
Tundra is respected throughout the industry for its outstanding commitment to customer support. Tundra ensures that its customers can take immediate advantage of the company’s products through telephone access to its in-house Applications Engineering Group, unmatched Design Support Tools, and full documentation accessible though the Web.

Tundra System Interconnect ... Silicon Behind the Network™
Contact Information

Tundra is dedicated to providing Tsi148 Early Access Program (TEAP) participants with superior technical documentation and support.

Tsi148 collateral and support are available on the TEAP Secure web site. Access to this web site is restricted to TEAP participants.

TEAP participants may also contact Tundra through the following means:

**Email**


Documentation Feedback Use docfeedback@tundra.com to provide feedback on the Tsi148 PCI/X-to-VME Bus Programming Manual.

**Fax**

613-592-1320

**Mailing Address**

Tundra Semiconductor Corporation
603 March Road
Ottawa, ON
K2K 2M5
# Contents

**Corporate Profile** ................................................................. 3  
**Contact Information** ............................................................... 5  
**About this Document** ............................................................... 23  
  - Revision History ................................................................. 23  
  - Document Conventions .......................................................... 24  
  - Related Information .............................................................. 27  
**1. Functional Overview** ......................................................... 29  
  - 1.1 Overview of Tsi148 .......................................................... 30  
    - 1.1.1 VME Renaissance ....................................................... 31  
    - 1.1.2 Tsi148 Features ........................................................ 32  
    - 1.1.3 Tsi148 Benefits ........................................................ 32  
    - 1.1.4 Typical Applications ................................................ 33  
  - 1.2 VMEbus Interface ............................................................. 35  
    - 1.2.1 2eVME Protocol ....................................................... 35  
    - 1.2.2 2eSST Protocol ........................................................ 35  
    - 1.2.3 VME Slave ............................................................... 36  
    - 1.2.4 VME Master ............................................................. 37  
    - 1.2.5 Tsi148 as a VMEbus System Controller .......................... 38  
  - 1.3 PCI/X Interface ............................................................... 40  
    - 1.3.1 PCI/X Target ............................................................ 40  
    - 1.3.2 PCI/X Master ............................................................ 41  
  - 1.4 Linkage Module .............................................................. 42  
  - 1.5 Register Overview .......................................................... 43  
    - 1.5.1 Control and Status Registers ....................................... 44  
  - 1.6 DMA Controllers ............................................................. 45  
    - 1.6.1 Data Movement ........................................................ 46
## 2. VME Interface

2.1 Overview of the VME Interface .................................................. 50
2.2 VME Slave ........................................................................ 50
  2.2.1 VME Slave Buffers ......................................................... 50
2.3 VME Master ........................................................................ 61
  2.3.1 Addressing Capabilities ................................................... 61
  2.3.2 VME Master Buffers ......................................................... 62
  2.3.3 VME Master Read-Modify Write (RMW) Cycles ..................... 62
  2.3.4 VME Master Bandwidth Control ......................................... 64
  2.3.5 VMEbus Exception Handling .............................................. 66
  2.3.6 Utility Functions ............................................................. 66
  2.3.7 Tsi148 as a VMEbus System Controller ................................ 71

## 3. PCI/X Interface

3.1 Overview of the PCI/X Interface ..................................................... 74
3.2 PCI Mode ........................................................................... 74
  3.2.1 PCI Target .................................................................. 74
  3.2.2 PCI Master .................................................................. 85
  3.2.3 PCI Bus Exception Handling ............................................. 86
3.3 PCI-X Mode ......................................................................... 88
  3.3.1 PCI-X Target ................................................................. 88
  3.3.2 PCI-X Master ................................................................. 98
  3.3.3 PCI-X Bus Exception Handling ......................................... 99

## 4. DMA Interface

4.1 Overview DMA Controller .......................................................... 102
4.2 Architecture ........................................................................ 102
4.3 DMA Buffers ....................................................................... 102
4.4 Operating Modes .................................................................. 103
  4.4.1 Linked-List Descriptors .................................................. 104
4.5 Direction of Data Movement ............................................................. 105
  4.5.1 PCI/X-to-VME ................................................................. 105
  4.5.2 VME-to-PCI/X ................................................................. 108
  4.5.3 PCI/X-to-PCI/X ................................................................. 111
  4.5.4 VME-to-VME ................................................................. 114
  4.5.5 Data Patterns ................................................................. 117
  4.5.6 DMA Transaction Termination .............................................. 119
  4.5.7 DMA Interrupts ............................................................... 120
  4.5.8 Transfer Throttling ........................................................... 120

5. Resets, Clocks, and Power-up Options ............................................ 121
  5.1 Overview of Resets, Clocks, and Power-up Options .......................... 122
  5.2 Resets ................................................................. 122
    5.2.1 Reset Inputs and Outputs ............................................. 123
    5.2.2 Reset Timing ............................................................. 125
  5.3 Clocks ................................................................. 126
  5.4 Power-up Options ............................................................ 127
    5.4.1 PCI/X Power-up Options .............................................. 127
    5.4.2 VMEbus Power-up Options ........................................... 129
    5.4.3 System Controller (SCON) ............................................. 133

6. Interrupt Controller ................................................................. 135
  6.1 Overview of the Interrupt Controller ......................................... 136
  6.2 VMEbus Interrupter ............................................................. 136
  6.3 Local Interrupter ............................................................... 136
  6.4 VMEbus Interrupt Handler ..................................................... 137

7. JTAG Module ................................................................. 139
  7.1 Overview of JTAG ............................................................. 140
  7.2 Instructions ................................................................. 140

8. Registers ................................................................. 143
  8.1 Overview of Registers ........................................................ 144
8.2 Register Groupings .................................................. 144
  8.2.1 Combined Register Group (CRG) ................................. 144
  8.2.2 PCI/X Configuration Space Registers (PCFS) ...................... 145
  8.2.3 Local Control and Status Registers (LCSR) ...................... 145
  8.2.4 Global Control and Status Registers (GCSR) ..................... 145
  8.2.5 Control and Status Registers (CSR) ............................. 145
  8.2.6 CR/CSR Register Access ........................................ 146
8.3 Register Endian Mapping ............................................. 147
8.4 Register Map ........................................................ 149
  8.4.1 Conventions .................................................... 149
  8.4.2 PCFS Register Group Overview .................................. 150
  8.4.3 LCSR Register Group Overview .................................. 152
  8.4.4 GCSR Register Group Overview .................................. 163
  8.4.5 CR/CSR Register Group Overview ............................... 164
  8.4.6 PCFS Register Group Description ............................... 165
  8.4.7 Vendor ID/ Device ID Registers ................................ 165
  8.4.8 Command/Status Registers ..................................... 166
  8.4.9 Revision ID / Class Code Registers ............................. 169
  8.4.10 Cache Line Size / Master Latency Timer / Header Type Registers .............. 170
  8.4.11 Memory Base Address Lower Register .......................... 172
  8.4.12 Memory Base Address Upper Register ........................... 173
  8.4.13 Subsystem Vendor ID/ Subsystem ID Registers .................. 174
  8.4.14 Capabilities Pointer Register ................................. 175
  8.4.15 Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Registers ...... 176
  8.4.16 PCI-X Capabilities Register ................................... 179
  8.4.17 PCI-X Status Register .......................................... 181
  8.4.18 LCSR Register Group Description ............................... 184
  8.4.19 Outbound Translation Starting Address Upper (0-7) Registers ............. 184
  8.4.20 Outbound Translation Starting Address Lower (0-7) Registers ............ 185
  8.4.21 Outbound Translation Ending Address Upper (0-7) Registers ............. 186
  8.4.22 Outbound Translation Ending Address Lower (0-7) Registers ............. 187
  8.4.23 Outbound Translation Offset Upper (0-7) Registers ............... 188
  8.4.24 Outbound Translation Offset Lower (0-7) Registers ............... 189
  8.4.25 Outbound Translation 2eSST Broadcast Select (0-7) Registers .............. 190
  8.4.26 Outbound Translation Attribute (0-7) Registers ................ 191
  8.4.27 VMEbus IACK (1-7) Registers .................................. 195
  8.4.28 VMEbus Read-Modify-Write (RMW) Address Upper Register .............. 196
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.29</td>
<td>VMEbus RMW Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.30</td>
<td>VMEbus RMW Enable Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.31</td>
<td>VMEbus RMW Compare Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.32</td>
<td>VMEbus RMW Swap Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.33</td>
<td>VME Master Control Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.34</td>
<td>VMEbus Control Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.35</td>
<td>VMEbus Status Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.36</td>
<td>PCI/X Control / Status Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.37</td>
<td>VMEbus Filter Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.38</td>
<td>VMEbus Exception Address Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.39</td>
<td>VMEbus Exception Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.40</td>
<td>VMEbus Exception Attributes Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.41</td>
<td>Error Diagnostic PCI/X Address Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.42</td>
<td>Error Diagnostic PCI/X Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.43</td>
<td>Error Diagnostic PCI-X Attribute Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.44</td>
<td>Error Diagnostic PCI-X Split Completion Message Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.45</td>
<td>Error Diagnostic PCI/X Attributes Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.46</td>
<td>Inbound Translation Starting Address Upper (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.47</td>
<td>Inbound Translation Starting Address Lower (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.48</td>
<td>Inbound Translation Ending Address Upper (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.49</td>
<td>Inbound Translation Ending Address Lower (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.50</td>
<td>Inbound Translation Offset Upper (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.51</td>
<td>Inbound Translation Offset Lower (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.52</td>
<td>Inbound Translation Attribute (0-7) Registers</td>
<td>197</td>
</tr>
<tr>
<td>8.4.53</td>
<td>GCSR Base Address Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.54</td>
<td>GCSR Base Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.55</td>
<td>GCSR Attribute Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.56</td>
<td>CRG Base Address Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.57</td>
<td>CRG Base Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.58</td>
<td>CRG Attribute Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.59</td>
<td>CR/CSR Offset Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.60</td>
<td>CR/CSR Offset Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.61</td>
<td>CR/CSR Attribute Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.62</td>
<td>Location Monitor Base Address Upper Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.63</td>
<td>Location Monitor Base Address Lower Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.64</td>
<td>Location Monitor Attribute Register</td>
<td>197</td>
</tr>
<tr>
<td>8.4.65</td>
<td>64-bit Counter Upper</td>
<td>197</td>
</tr>
</tbody>
</table>
8.4.66 64-bit Counter Lower ................................................. 255
8.4.67 Broadcast Pulse Generator Timer Register ......................... 256
8.4.68 Broadcast Programmable Clock Timer Register ................. 257
8.4.69 VMEbus Interrupt Control Register ............................... 258
8.4.70 Interrupt Enable Register ........................................... 261
8.4.71 Interrupt Enable Out Register ..................................... 266
8.4.72 Interrupt Status Register ............................................ 269
8.4.73 Interrupt Clear Register ............................................. 272
8.4.74 Interrupt Map 1 Register ............................................. 275
8.4.75 Interrupt Map 2 Register ............................................. 277
8.4.76 DMA Control (0-1) Registers ........................................ 279
8.4.77 DMA Status (0-1) Registers .......................................... 284
8.4.78 DMA Current Source Address Upper (0-1) Registers ........... 287
8.4.79 DMA Current Source Address Lower (0-1) Registers .......... 288
8.4.80 DMA Current Destination Address Upper (0-1) Registers .... 289
8.4.81 DMA Current Destination Address Lower (0-1) Registers .... 290
8.4.82 DMA Current Link Address Upper (0-1) Registers .......... 291
8.4.83 DMA Current Link Address Lower (0-1) Registers .......... 292
8.4.84 DMA Source Address Upper (0-1) Registers ..................... 293
8.4.85 DMA Source Address Lower (0-1) Registers ..................... 294
8.4.86 DMA Destination Address Upper (0-1) Registers .......... 295
8.4.87 DMA Destination Address Lower (0-1) Registers .......... 296
8.4.88 DMA Source Attribute (0-1) Registers ............................. 297
8.4.89 DMA Destination Attribute (0-1) Registers ..................... 301
8.4.90 DMA Next Link Address Upper (0-1) Registers ............... 305
8.4.91 DMA Next Link Address Lower (0-1) Registers ............... 306
8.4.92 DMA Count (0-1) Registers .......................................... 307
8.4.93 DMA Destination Broadcast Select (0-1) Registers .......... 308
8.4.94 GCSR Register Group ................................................ 309
8.4.95 Vendor ID / Device ID Registers ................................... 309
8.4.96 Control and Status Register ......................................... 310
8.4.97 Semaphore Registers (0-3) .......................................... 313
8.4.98 Semaphore Registers (4-7) .......................................... 314
8.4.99 Mail Box Registers (0-3) .............................................. 315
8.4.100 CR/CSR Register Group Description ............................. 316
8.4.101 CR/CSR Bit Clear Register .......................................... 316
8.4.102 CR/CSR Bit Set Register .............................................. 318
List of Figures

Figure 1: Tsi148 Block Diagram ................................................................. 31
Figure 2: Typical Application — Tsi148 In Single Board Computer Application .................. 34
Figure 3: Divisions of the CRG Register Space ............................................. 43
Figure 4: CR/CSR Register Space ............................................................... 44
Figure 5: Slave Image Programmable Address Offset ...................................... 53
Figure 6: VMEbus to PCI/X Read Request .................................................... 55
Figure 7: VMEbus to PCI/X Read Completion .............................................. 57
Figure 8: VMEbus to PCI/X Write ............................................................... 59
Figure 9: Steps Used to Perform RMW Cycles on the VMEbus ......................... 64
Figure 10: PCI Target Image Programmable Address Offset ............................. 76
Figure 11: PCI-VME Delayed Read Request .................................................. 79
Figure 12: PCI-to-VME Delayed Read Completion ........................................ 81
Figure 13: PCI-to-VME Posted Write .......................................................... 83
Figure 14: Target Image Programmable Address Offset ................................... 90
Figure 15: PCI-X-to-VME Delayed Read Request .......................................... 92
Figure 16: PCI-X-to-VME Delayed Read Completion ..................................... 94
Figure 17: PCI-X-to-VME Posted Write ....................................................... 96
Figure 18: Direct Mode ............................................................................... 103
Figure 19: Linked-list Mode ......................................................................... 103
Figure 20: DMA Transaction: PCI/X-to-VME Request .................................... 106
Figure 21: DMA Transaction: PCI/X-to-VME Completion ............................... 107
Figure 22: DMA Transaction: VME-to-PCI/X Request .................................... 109
Figure 23: DMA Transaction: VME-to-PCI/X Completion .............................. 110
Figure 24: DMA Transaction: PCI/X-to-PCI/X Request .................................. 112
Figure 25: DMA Transaction: PCI/X-to-PCI/X Completion ............................ 113
List of Figures

Figure 26: DMA Transaction: VME-to-VME Request ................................. 115
Figure 27: DMA Transaction: VME-to-VME Completion ......................... 116
Figure 28: 8-bit Pattern Writes ............................................................... 117
Figure 29: 32-bit Pattern Writes ............................................................... 118
Figure 30: Tsi148 Reset Structure ............................................................ 122
Figure 31: Timing for Power-up Reset ....................................................... 125
Figure 32: JTAG Functional Diagram ....................................................... 140
Figure 33: Combined Register Group (CRG) ........................................... 144
Figure 34: CR/CSR Address Space .......................................................... 146
Figure 35: Big to Little Endian Data Swap .............................................. 147
Figure 36: 456-Pin PBGA Package Diagram — Bottom and Side Views ........ 322
Figure 37: 456-Pin PBGA Package Diagram — Top View ....................... 323
List of Tables

Table 1: VMEbus Address Mode Codes ............................................................... 61
Table 2: Location Monitor Interrupt Addresses .............................................. 68
Table 3: PCI Read Data Size ........................................................................... 78
Table 4: DMA Controller Linked-List Descriptors .......................................... 104
Table 5: PCI Bus Configuration ..................................................................... 126
Table 6: PCI/X Bus Configuration .................................................................. 128
Table 7: VMEbus Power-up Options ................................................................. 129
Table 8: ASIDEN and GSIDEN Definition ....................................................... 130
Table 9: CR/CSR Base Address Configuration ................................................. 130
Table 10: VMEbus System Controller Configuration ...................................... 134
Table 11: Endian Register Views .................................................................... 148
Table 12: PCFS Register Group ...................................................................... 150
Table 13: LCSR Register Group ..................................................................... 152
Table 14: GCSR Register Group .................................................................... 163
Table 15: CR/CSR Register Group .................................................................. 164
Table 16: Vendor ID/ Device ID Registers ...................................................... 165
Table 17: Command/Status Register ................................................................. 166
Table 18: Revision ID / Class Code Register .................................................. 169
Table 19: Cache Line Size / Master Latency Timer / Header Type Register ...... 170
Table 20: Memory Base Address Lower Register ........................................... 172
Table 21: Memory Base Address Upper Register ........................................... 173
Table 22: Subsystem Vendor ID/ Subsystem ID Register ................................. 174
Table 23: Capabilities Pointer Register ............................................................ 175
Table 24: Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Register . 176
Table 25: CRG Space Type ............................................................................. 177
List of Tables

Table 26: INTP INTx Encoding ................................................................. 177
Table 27: PCI-X Capabilities Register .................................................. 179
Table 28: MOST Encoding ................................................................. 180
Table 29: MMRBC Encoding ............................................................... 180
Table 30: PCI-X Status Register ......................................................... 181
Table 31: DMCRS Encoding ............................................................... 181
Table 32: Outbound Translation Starting Address Upper (0-7) Register .... 184
Table 33: Outbound Translation Starting Address Lower (0-7) Register ..... 185
Table 34: Outbound Translation Ending Address Upper (0-7) Register .... 186
Table 35: Outbound Translation Ending Address Lower (0-7) Register .... 187
Table 36: Outbound Translation Offset Upper (0-7) Register ............... 188
Table 37: Outbound Translation Offset Lower (0-7) Register ............... 189
Table 38: Outbound Translation 2eSST Broadcast Select (0-7) Register .... 190
Table 39: Outbound Translation Attribute (0-7) Register ..................... 191
Table 40: Prefetch Size ................................................................. 192
Table 41: 2eSST Transfer Rate ......................................................... 193
Table 42: VMEbus Transfer Mode ...................................................... 193
Table 43: VMEbus Data Bus Width .................................................... 193
Table 44: VMEbus Address Mode ...................................................... 194
Table 45: VMEbus IACK (1-7) Register .............................................. 195
Table 46: VMEbus RMW Address Upper Register ................................ 196
Table 47: VMEbus RMW Address Lower Register ................................ 197
Table 48: VMEbus RMW Enable Register .......................................... 198
Table 49: VMEbus RMW Compare Register ....................................... 199
Table 50: VMEbus RMW Swap Register ............................................ 200
Table 51: VME Master Control Register ............................................ 201
Table 52: VME Master Time Off ...................................................... 203
Table 53: VME Master Time On ....................................................... 203
Table 54: VME Master Release Mode ............................................... 204
Table 55: VMEbus Control Register .................................................. 205
Table 56: Deadlock Timer ............................................................... 206
Table 57: VMEbus Global Time-out ................................................... 207
Table 58: VMEbus Status Register .................................................... 209
Table 59: PCI/X Control / Status Register .......................... 211
Table 60: PCI-X Split Read Time-out .............................. 212
Table 61: PCI/X Control / Status Register .......................... 215
Table 62: Acknowledge Delay Time ................................. 216
Table 63: VMEbus Exception Address Upper Register .......... 217
Table 64: VMEbus Exception Address Lower Register .......... 218
Table 65: VMEbus Exception Attributes Register ............... 219
Table 66: Error Diagnostic PCI/X Address Upper Register ...... 222
Table 67: Error Diagnostic PCI/X Address Lower Register ...... 223
Table 68: Error Diagnostic PCI-X Attribute Register .......... 224
Table 69: Error Diagnostic PCI-X Split Completion Message Register 225
Table 70: Error Diagnostic PCI/X Attributes Register .......... 226
Table 71: Inbound Translation Starting Address Upper (0-7) Register 229
Table 72: Inbound Translation Starting Address Upper (0-7) Register 230
Table 73: Inbound Translation Ending Address Upper (0-7) Register 231
Table 74: Inbound Translation Ending Address Lower (0-7) Register 232
Table 75: Inbound Translation Offset Upper (0-7) Register .... 233
Table 76: Inbound Translation Offset Lower (0-7) Register .... 234
Table 77: Inbound Translation Attribute (0-7) Register ......... 235
Table 78: Virtual FIFO Size ........................................ 236
Table 79: 2eSST Mode ............................................. 237
Table 80: VMEbus Address Space .................................. 237
Table 81: GCSR Base Address Upper Register ..................... 239
Table 82: GCSR Base Address Lower (0-7) Register ............. 240
Table 83: GCSR Attribute Register ................................ 241
Table 84: VMEbus Address Space .................................. 242
Table 85: CRG Base Address Upper Register ....................... 243
Table 86: CRG Base Address Lower Register ....................... 244
Table 87: CRG Attribute Register .................................. 245
Table 88: VMEbus Address Space .................................. 246
Table 89: CR/CSR Offset Upper Register .......................... 247
Table 90: CR/CSR Offset Lower Register .......................... 248
Table 91: CRG Attribute Register .................................. 249
Table 92: Location Monitor Base Address Upper Register .................................................. 250
Table 93: Location Monitor Base Address Lower Register ................................................ 251
Table 94: Location Monitor Register ................................................................................. 252
Table 95: VMEbus Address Space ....................................................................................... 253
Table 96: 64-bit Counter Upper Register ............................................................................ 254
Table 97: 64-bit Counter Lower Register ............................................................................ 255
Table 98: Broadcast Pulse Generator Timer Register ......................................................... 256
Table 99: Broadcast Programmable Clock Timer Register .................................................. 257
Table 100: VMEbus Interrupt Control Register ................................................................. 258
Table 101: Counter Source ................................................................................................. 259
Table 102: Edge Interrupt Source ....................................................................................... 259
Table 103: VMEbus IRQ[1]O Function ................................................................................. 259
Table 104: VMEbus IRQ[2]O Function ................................................................................. 260
Table 105: Interrupt Enable Register .................................................................................. 261
Table 106: Interrupt Enable Out Register .......................................................................... 266
Table 107: Interrupt Status Register ................................................................................... 269
Table 108: Interrupt Clear Register ..................................................................................... 272
Table 109: Interrupt Map 1 Register .................................................................................... 275
Table 110: Interrupt Map 2 Register .................................................................................... 277
Table 111: DMA Control (0-1) Register ............................................................................. 279
Table 112: DCTL BKS Encoding ......................................................................................... 281
Table 113: DCTL VBOT Encoding ....................................................................................... 282
Table 114: DCTL PBKS Encoding ....................................................................................... 282
Table 115: DCTL PBOT Encoding ....................................................................................... 283
Table 116: DMA Status (0-1) Register ................................................................................ 284
Table 117: DSTA ERT Encoding ......................................................................................... 286
Table 118: DMA Current Source Address Upper (0-1) Register ......................................... 287
Table 119: DMA Current Source Address Lower (0-1) Register ......................................... 287
Table 120: DMA Current Destination Address Upper (0-1) Register ................................. 289
Table 121: DMA Current Destination Address Lower (0-1) Register ................................. 289
Table 122: DMA Current Link Address Upper (0-1) Register ............................................ 290
Table 123: DMA Current Link Address Lower (0-1) Register ........................................... 292
Table 124: DMA Source Address Upper (0-1) Register ..................................................... 293
List of Tables

Table 125: DMA Source Address Lower (0-1) Register .................................................. 294
Table 126: DMA Destination Address Upper (0-1) Register ......................................... 295
Table 127: DMA Destination Address Lower (0-1) Register ........................................... 296
Table 128: DMA Source Attribute (0-1) Register ............................................................ 297
Table 129: DSAT TYP Encoding ......................................................................................... 298
Table 130: 2eSST Transfer Rate ......................................................................................... 299
Table 131: VMEbus Transfer Mode .................................................................................... 299
Table 132: VMEbus Data Bus Width ................................................................................... 299
Table 133: VMEbus Address Mode .................................................................................... 300
Table 134: DMA Destination Attribute (0-1) Register ..................................................... 301
Table 135: DDAT TYP Encoding ......................................................................................... 302
Table 136: 2eSST Transfer Rate ......................................................................................... 302
Table 137: VMEbus Transfer Mode .................................................................................... 303
Table 138: VMEbus Data Bus Width ................................................................................... 303
Table 139: VMEbus Address Mode .................................................................................... 304
Table 140: DMA Next Link Address Upper (0-1) Register ................................................ 305
Table 141: DMA Next Link Address Lower (0-1) Register ............................................... 306
Table 142: DMA Count (0-1) Register ............................................................................... 307
Table 143: DMA Destination Broadcast Select (0-1) Register ......................................... 308
Table 144: Vendor ID / Device ID Register ....................................................................... 309
Table 145: Control and Status Register ............................................................................. 310
Table 146: Semaphore Register (0-3) ............................................................................... 313
Table 147: Semaphore Registers (0-4) ............................................................................. 314
Table 148: Mail Box Registers (0-3) ................................................................................ 315
Table 149: CR/CSR Bit Clear Register .............................................................................. 316
Table 150: CR/CSR Bit Set Register .................................................................................. 318
Table 151: CR/CSR Base Address Register ...................................................................... 320
Table 152: Package Characteristics .................................................................................. 321
List of Tables
About this Document

This chapter discusses general document information about the Tsi148 PCI/X-to-VME Bus Programming Manual. The following topics are described:

- “Revision History” on page 23
- “Document Conventions” on page 24
- “Related Information” on page 27

Revision History

80A3020_MA002_01, Programming Manual, May 2004

This revision of the Tsi148 PCI/X-to-VME Bus Programming Manual has changes throughout the document.

80A3020_FD001_02, Programming Manual, June 2003

This revision of the Tsi148 PCI/X-to-VME Bus Bridge Advance User Manual has changes in the following section:

- “Typical Applications” on page 369

80A3020_FD001_01, Programming Manual, May 2003

This is the first version of the Tsi148 PCI/X-to-VME Bus Bridge Advance User Manual.
Document Conventions

This section explains the document conventions used in this manual.

Non-differential Signal Notation

Non-differential signals, such as those used by the PCI/X standard, are either active high or active low. Active low signals are defined as true (asserted) when they are at a logic low. Similarly, active high signals are defined as true at a logic high. Non-differential signals are considered asserted when active and negated when inactive, irrespective of voltage levels. For voltage levels, the use of 0 indicates a low voltage while a 1 indicates a high voltage.

Non-differential signals that assume a logic low state when asserted are followed by an underscore sign as the last non-numerical character, “_”. For example, SIGNAL_[0] is asserted low to indicate an active low signal. Non-differential signals not followed by an underscore are asserted when they assume the logic high state. For example, SIGNAL[0] is asserted high to indicate an active high signal.

Bit Ordering Notation

When referring to PCI/X transactions, this document assumes the most significant bit is the largest number (also known as little-endian bit ordering). For example, the PCI address/data bus consists of AD[31:0], where AD[31] is the most significant bit and AD[0] is the least-significant bit of the field.

Both bits and bytes have an ordering convention. The bit ordering convention for the PCI bus interface is little-endian bit ordering in which bit 0 is the least significant bit. The byte ordering convention of the PCI bus is little-endian. Byte 0 represents the least significant data bits of the word. This corresponds to the bit and byte ordering convention of the PCI bus. PCI is consistent in the bit and byte ordering.

The bit ordering convention for the VMEbus interface is little-endian bit ordering in which bit 0 is the least significant bit. The byte ordering convention is big-endian. Byte 0 represents the most significant bits of the word. This corresponds to the bit and byte ordering convention of the VMEbus. The VMEbus is not consistent in the bit and byte ordering.
Object Size Notation

The following object size conventions are used for PCI/X transactions:

- A byte is an 8-bit object.
- A word is a 16-bit (2-byte) object.
- A doubleword (dword) is a 32-bit (4-byte) object.
- A quadword is a 64-bit (8-byte) object.

Numeric Notation

The following numeric conventions are used:

- Hexadecimal numbers are denoted by the prefix 0x. For example, 0x04.
- Binary numbers are denoted by the suffix b. For example, 10b.

Typographic Notation

The following italic typographic conventions are used in this manual:

- Book titles: For example, PCI Local Bus Specification (Revision 2.2).
- Important terms: For example, when a device is granted access to the PCI bus it is called the bus master.
- Undefined values: For example, the device supports four channels depending on the setting of the PCI_Dx register.

Terminology

The following terms are used in this manual:

- PCI/X: Refers to both the PCI and PCI-X bus. The PCI/X interface can be configured for either PCI or PCI-X operation.
Symbols Used

The following symbols are used in this manual:

This symbol indicates important configuration information or suggestions.

This symbol indicates procedures or operating levels that may result in misuse or damage to the device.

This symbol indicates a basic design concept or information considered helpful.
Related Information

The following information is useful for reference purposes when using this manual:

**American National Standard for VME64**
This specification defines the VME64 hardware system including the protocol, electrical, mechanical and configuration specification for the VMEbus components and expansion boards.

**American National Standard for VME64 Extensions (ANSI/VITA 6.1 1996 (R2003))**
This specification defines extensions to the VME64 standard including the protocol, electrical, mechanical and configuration specification for the VMEbus components and expansion boards.

**Source Synchronous Transfer (2eSST) Standard (VITA 1.5 2003)**
This specification defines the 2eSST including the protocol, electrical, and configuration specifications.

**PCI Local Bus Specification (Revision 2.2)**
This specification defines the PCI hardware system including the protocol, electrical, mechanical and configuration specification for the PCI local bus components and expansion boards. For more information, see www.pcisig.com.

**PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b)**
This specification addresses the need for increased bandwidth of PCI Devices. PCI-X enables the design of systems and devices that can operate at speeds significantly higher than today's specification allows. For more information, see www.pcisig.com.
1. Functional Overview

This chapter describes the main features and functions of the Tsi148™. The following topics are discussed:

- “Overview of Tsi148” on page 30
- “Tsi148 Features” on page 32
- “Tsi148 Benefits” on page 32
- “Typical Applications” on page 33
- “VMEbus Interface” on page 35
- “PCI/X Interface” on page 40
- “Linkage Module” on page 42
- “Register Overview” on page 43
- “DMA Controllers” on page 45
- “Interrupter and Interrupt Handler” on page 46
- “JTAG” on page 47
1. Functional Overview

1.1 Overview of Tsi148

The Tundra Tsi148 device is the next generation component in our industry leading, high performance VMEbus system interconnect product family. Tsi148 is fully compliant with the 2eSST and VME64 Extension standards. This enables you to take advantage of the higher performance VME protocols, while preserving your existing investment in VME boards that implement legacy protocols.

Tsi148 increases a system’s usable bus bandwidth because its local bus interface is designed for the next generation PCI/X processors and peripherals that support either a 66 MHz PCI bus or a 133 MHz PCI-X bus interface.

Tsi148 eases design constraints of VME Single Board Computers (SBCs) by requiring less board real estate and power than the previous generation of VME-to-PCI/X bridge components.

These capabilities make Tsi148 a key building block of the VME Renaissance and the development of next generation VME single board computers.
1.1.1 VME Renaissance

*VME Renaissance* is a term defined by Motorola™ that describes an intense period of intellectual activity and technology infusion focused on the VMEbus. The renaissance is a period of innovation and performance improvement which maintains backwards compatibility to legacy VMEbus standards. This compatibility requirement protects existing customer investments.

The VME Renaissance gives VME a faster parallel backplane interconnect, a switched serial interconnect on the backplane coincident with the traditional parallel interconnect, point-to-point mezzanines on the cards and many other significant innovations.
1.1.2 **Tsi148 Features**
Tsi148 has the following key features:

**VMEbus Interface**
- Standards supported:
  - Legacy protocols to protect existing VME investment
  - VME64 Extensions
  - 2eVME and 2eSST protocols to bring support for higher bandwidth
- Full VMEbus system controller functionality

**PCI/X Interface**
- Fully compliant, programmable PCI or PCI-X bus interface
- Multiple modes of bus operation
  - Interface can be configured as PCI-X or PCI
  - PCI-X interface operates from 50-to-133 MHz
  - PCI interface operates from 33-to-66 MHz
- 32-bit or 64-bit addressing and data in PCI and PCI-X modes

**Other Features**
- Two, programmable DMA controllers with Direct mode and Linked-List mode support
- Interrupt and interrupt handling capability
- Flexible register set; programmable from both PCI/X and VMEbus
- IEEE 1149.1 Interface
- 456 PBGA package, 1.0 mm ball pitch

1.1.3 **Tsi148 Benefits**
Tsi148 offers the following benefits to designers:

- Increased bandwidth
  - 8x increase in usable system bus bandwidth over current solutions
- Less power required than existing devices due to reduced voltages
  - 3.3V I/O supply
  - 1.8V Core supply
1. Functional Overview

- Small device footprint
  - 40% less space required than existing products
- Reliable customer support with experience supporting the VME community for the past decade.

1.1.4 Typical Applications

Tsi148 is intended for VME Single Board Computers and VME I/O peripheral cards that serve the following markets:

- Telecommunications
- Industrial automation
- Medical
- Military
- Aerospace
1.1.4.1 **Typical Application — Single Board Computers**

The Tsi148 can be used on VME-based Single Board Computers (SBC) that employ PCI/X as their local bus and VME as the backplane bus, as shown in the accompanying diagram. These SBC cards support a variety of applications including telecommunications, datacommunications, medical, industrial automation, and military equipment.

The Tsi148 high performance architecture seamlessly bridges the PCI/X and VME busses, and is the VME industry's standard for single board computer interconnect device.

**Figure 2: Typical Application — Tsi148 In Single Board Computer Application**
1.2 VMEbus Interface

The Tsi148 VMEbus Interface is compliant with the following standards:

- *American National Standard for VME64 (ANSI/VITA 1.0 - 1994 (R2002))*
- *Source Synchronous Transfer (2eSST) Standard*

For more information on the VME Interface refer to Section 2. on page 49.

1.2.1 2eVME Protocol

The 2eVME protocol doubles the VME64 peak block data rate to 160 Mbytes/s by utilizing both edges of the DS* signal and the DTACK signal to validate data. The addressing phase of the transaction also differs from VME64 transactions because the address broadcast is split into three phases. The three phase address broadcast transmits extended AM codes (programmable limit of 256 codes), VME master information, and the transaction beat count.

The 2eVME protocol doubles peak block data rate and has flexibility in transaction terminations. The following terminations of transactions are allowed in 2eVME:

- Master termination: Before the beat count expires
- Slave terminated transactions: Using the RETRY* and BERR* signals
- Slave suspended terminations: Using the RETRY* and DTACK* signals

Refer to the *American National Standard for VME64 Extensions* for more information on the 2eVME protocol.

1.2.2 2eSST Protocol

The 2eSST protocol further increases VME transaction bandwidth with programmable transfer rates of 160, 267, and 320 Mbytes/s.

Although the 2eSST protocol is similar to the 2eVME protocol there are a number of differences and specific requirements for 2eSST protocol. Transactions are source synchronous in 2eSST; there is no acknowledgement from receiver of the data. This lack of acknowledgement enables transactions to happen at a faster rate; there are no delays caused by multiple acknowledgments as in the original VME standard.

Performance enhancements delivered by 2eSST require careful management of system-wide skew. 2eSST protocol implementation is possible on standard VME64x five row backplanes with Texas Instrument’s high performance bus transceivers.
1. Functional Overview

Refer to the *Source Synchronous Transfer (2eSST) Standard* for more information on the 2eSST protocol.

### 1.2.3 VME Slave

The Tsi148 VME Slave accepts most of the addressing and data transfer modes documented in the *VME64 Specification*, the *VME64x Specification*, and *Source Synchronous Transfer (2eSST) Standard* specification. The supported transactions include:

- Address: A16, A24, A32, and A64
- Data: D8, D16, and D32 Single Cycle Transaction (SCT)
- Data: D8, D16, D32 Block Transaction (BLT)
- Data: D64 Multiple Block Transaction (MBLT)
- Data: D64 2eVME
- Data: D64 2eSST

Incoming write transactions from the VMEbus are posted. With posted write transactions, data is written to a VME Slave write buffer. The VME Slave write buffer is a 4 Kbyte buffer. When the Tsi148 VME Slave accepts a write request, the initiating VMEbus master receives a data acknowledgment from Tsi148. Write data is transferred from the VME Slave write buffer, through the internal Linkage Module, to the PCI/X Master write buffer without involving the initiating VMEbus master. Refer to Section 2.2.1 on page 50 for a detailed description of transaction flow and buffer usage in Tsi148.

The VME Slave read operations depend on whether the transfer is a SCT or BLT transfer. If the transfer is a SCT transfer, the VME Slave requests a single beat transfer from the Linkage Module (see Section 1.4 on page 42). A PCI/X prefetched read is initiated when a VMEbus master initiates a block read (BLT, MBLT, 2eVME, or 2eSST) transaction on the VMEbus. When the Tsi148 PCI/X Master receives a read request (after the VME Slave sends the read request requirements through the Linkage Module), the PCI/X Master fills its read buffer by issuing burst requests to the PCI/X bus target.

The VME Slave read buffer is a 2 Kbyte read buffer with a programmable size and refill threshold. The design enables the initiating VMEbus master to acquire its block read data from the VME Slave (after the PCI/X Master has transferred the data through the Linkage Module to the VME Slave) instead of directly from the PCI/X resources. Refer to Section 2.5 on page 79 for a detailed description of transaction flow and buffer usage in Tsi148.
1.2.3.1 Features Not Supported
The following features are not supported by the Tsi148 VME Slave:

- A40 address modes
- D32 MBLT transfers
- VMEbus Lock commands
- RMW cycles are not guaranteed indivisible on the PCI bus

1.2.4 VME Master
The Tsi148 is VME Master when the VME Master is internally requested by the Linkage Module to service the PCI/X Target, DMA, or Interrupts. The internal Linkage Module arbitrates requests for each interface. Refer to Section 1.4 on page 42 for more information on the Linkage Module.

The Tsi148’s VME Master can generate the following addressing and data transfer modes:

- Address: A16, A24, A32, and A64
- Data: D8, D16, and D32 Single Cycle Transaction (SCT)
- Data: D16, D32 Block Transaction (BLT)
- Data: D64 Multiple Block Transaction (MBLT)
- Data: D64 2eVME
- Data: D64 2eSST

As VME Master, Tsi148 supports Read-Modify-Write (RMW) generation, and RETRY* as a termination from the external VMEbus slave.

Refer to the American National Standard for VME64 Extensions for more information on the RETRY* signal.

The VME Master has two 4 Kbyte posted write buffers and two 4 Kbyte prefetch read buffers. These buffers enable the VME Master to buffer two read or write transactions simultaneously.

Tsi148 provides several mechanisms to control VMEbus usage, including: time-on timer, time-off timer, and additional release mode control (see Section 2. on page 49).

1.2.4.1 Features Not Supported
The following features are not supported by the Tsi148 VME Master:

- A40 address modes
- D32 MBLT transfers
1. Functional Overview

- VMEbus lock commands

1.2.5 **Tsi148 as a VMEbus System Controller**

The Tsi148 supports the following VMEbus system controller functions:

- VMEbus Arbiter with three modes of programmable arbitration:
  - Priority (PRI)
  - Round-Robin-Select (RRS)
  - Single Level (SGL)
- IACK Daisy-Chain Driver
- SYSRESET Driver: Provides a global system reset
- Global VMEbus Timer: Monitors the VMEbus and generates a BERR_ when there is no VMEbus activity for the programmed value
- System Clock Driver: Generates a 16 MHz system clock

1.2.5.1 **Arbiter**

The Tsi148 VMEbus arbiter is programmable. All three of the following arbitration modes defined by the VMEbus standard are supported:

- Priority (PRI)
- Round-Robin-Select (RRS)
- Single Level (SGL)

A 16 us arbitration timer is included in the Tsi148 to prevent a bus lock-up from occurring when no requester assumes mastership of the bus after the arbiter has issued a grant. This timer can be enabled or disabled in the VMEbus Control and Status Register (see Section 8.4.34 on page 205).

1.2.5.2 **IACK Daisy-Chain Driver**

An IACK Daisy-Chain driver is included in the Tsi148 as part of the system controller functionality. This feature ensures that the timing requirements for starting the IACK Daisy-Chain are satisfied.
1. Functional Overview

1.2.5.3 SYSRESET Driver

A SYSRESET driver is included in the Tsi148 to provide a global system reset. The SRSTO signal is asserted in the following cases: the LSRSTI_ pin is asserted, the SRESET bit is asserted in the VMEbus Control Status Register, or the PURSTI_ pin is asserted. The SRSTO signal is always asserted for at least 200 ms. SRSTO is normally connected to the VMEbus SYSRESET_ signal through an inverting open collector buffer.

1.2.5.4 Global VMEbus Timer

The Tsi148 has a VMEbus global timer that monitors VMEbus cycles and generates a BERR signal when there is no VMEbus slave response for the programmed time period. The global timer only monitors VMEbus cycles when the system controller function is enabled. The global timer is compatible with SCT, BLT, MBLT, 2eVME, and 2eSST transfers. The global time-out period can be programmed for 8, 16, 32, 64, 128, 256, 512 µs. This timer can be enabled or disabled in the VMEbus Control and Status Register (see Section 8.4.34 on page 205).

1.2.5.5 System Clock Driver

Tsi148 generates the system clock (SYSCLK) signal when it is configured as the system controller. The SYSCLK signal is in spec for the following PCI/X clock frequencies: 33.3, 66.6, 100, or 133 MHz. The SYSCLK pin is connected through an external driver to the VMEbus. SYSCLK operates at 16 MHz. The external driver is enabled through the SCON pin (see Section 8.3.2 on page 159).

1.2.5.6 Configuration

The system controller functions can be configured at power-up. The system controller functionality can be enabled or disabled, or the auto system controller (SCON) function can be used. The auto SCON function automatically enables the system controller functions when the board is installed in slot 1. Table 10 on page 134 shows the different signal combinations that enable or disable the SCON functionality.
1.3 **PCI/X Interface**

The Tsi148 PCI/X Interface can operate either in PCI mode or PCI-X mode. The PCI interface is compliant with the *PCI Local Bus Specification (Revision 2.2)*, while the PCI-X interface is compliant with the *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b)*

The term *PCI/X* refers to functionality that applies to both PCI and PCI-X operating modes.

The PCI mode can operate at 33 to 66 MHz and has 32-bit/64-bit addressing and data capability. The PCI-X mode can operate at 50 to 133 MHz and has 32-bit/64-bit addressing and data capability.

For more information on the PCI/X Interface refer to Section 3. on page 73.

1.3.1 **PCI/X Target**

The PCI and PCI-X targets are described separately in the following sections because they respond differently to read requests and use different buffering techniques for read transactions.

1.3.1.1 **PCI Target**

Read transactions from the PCI bus are always processed as delayed transactions. The PCI Target has a 4 Kbyte read buffer, however, in conventional PCI mode a maximum of 512 bytes are used for storing prefetched data. When processing a read request the requesting PCI bus master is issued a retry from the Tsi148 PCI Target. The read request is then forwarded to the Linkage Module and then to the Tsi148 VME Master to be serviced. One delayed read is supported by the PCI Target.

During write transactions, the PCI Target posts write data in its write buffer. The write buffer consists of a 40 entry command queue and a 4 Kbyte data queue. Tsi148 issues the initiating PCI bus master immediate acknowledgement upon the write completing. Once the posted write completes on PCI, Tsi148 obtains the VMEbus and writes the data to the VMEbus resource independent of the initiating PCI master.

For more information on buffer structure and data flow in Tsi148 refer to Section 3. on page 73.

1.3.1.2 **Features Not Supported**

The following features are not supported by the Tsi148 PCI Target:

- No response to PCI I/O transfers
- PCI/X LOCK_ signal
1.3.1.3 **PCI-X Target**

Read transactions from the PCI-X bus are always processed as split transactions. The PCI-X Target has a 4 Kbyte read buffer used for storing prefetched data. The requesting external PCI-X master is issued a split response from the Tsi148 PCI-X Target. The PCI-X Target supports up to six split read transactions.

Prefetching is based on the byte count received by the Tsi148 PCI-X Target.

When the read data has been retrieved from the VMEbus and sent to the PCI-X Target’s read buffer, Tsi148 issues a split completion on the PCI-X bus and transfers the data from the PCI-X Target’s read data buffer to the original master.

During write transactions, the PCI-X Target posts write data in its write buffer. The write buffer consists of a 40 entry command queue and a 4 Kbyte data queue. Tsi148 issues the initiating PCI bus master immediate acknowledgement upon the write completing. Once the posted write completes on PCI-X, Tsi148 obtains the VMEbus and writes the data to the VMEbus resource independent of the initiating PCI-X master.

1.3.1.4 **Features Not Supported**

The following features are not supported by the Tsi148 PCI-X Target:

- No response to PCI-X I/O transfers
- PCI/X LOCK_ signal
- Message signalled interrupts

1.3.2 **PCI/X Master**

Tsi148 requests PCI/X ownership when the PCI/X Master is internally requested by Linkage Module to service the VME Slave or the DMA controllers.

The PCI/X Master has a 4 Kbyte read buffer and 4 Kbyte write buffer.

The size of the read buffer is dependent on what PCI/X mode (PCI or PCI-X) is used in the system (see Section 1.3.1.1 on page 40).

1.3.2.1 **Features Not Supported**

The following features are not supported in Tsi148:

- PCI/X LOCK_ signal
1.4 **Linkage Module**

The Tsi148 Linkage Module interconnects all the different modules that comprise Tsi148. The following modules are directly-connected to, and serviced by, the Linkage Module:

- VMEbus: Master and Slave
- PCI/X: Master and Target
- DMA Controllers
- Registers

The Linkage Module is used to arbitrate access to each interface. It controls the flow of data and data requests through the device. Every transaction processed through Tsi148 passes through the Linkage Module.
1.5 Register Overview

Tsi148’s 4 Kbyte register space is called the Combined Register Group (CRG). The CRG is divided into the following groups:

- PCI Configuration Space registers (PFCS)
- Local control and status registers (LCSR)
- Global control and status registers (GCSR)
- Control and Status Registers (CSR)

For more information on Tsi148’s registers, refer to Section 8.2 on page 144.

Tsi148’s registers can only be accessed through the Linkage Module. The interfaces that can access registers are the PCI/X Interface and the VMEbus Interface.

Figure 3: Divisions of the CRG Register Space

<table>
<thead>
<tr>
<th>4 Kbyte CRG</th>
<th>1024 bytes</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1504 bytes</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>32 bytes</td>
<td>GCSR</td>
</tr>
<tr>
<td></td>
<td>1280 bytes</td>
<td>LCSR</td>
</tr>
<tr>
<td></td>
<td>256 bytes</td>
<td>PCFS</td>
</tr>
</tbody>
</table>
1.5.1 Control and Status Registers

The 512 Kbyte CR/CSR space, shown in Figure 4, can be accessed from the VMEbus using the special A24 CR/CSR AM code (see Section 2.3.1 on page 61).

The Base Address is defined by either Geographical Address Implementation or Auto Slot ID. Tsi148’s VME Slave can be configured at power-up to use one of the two methods (see Section 5.4 on page 127). The CR/CSR offset registers consist of an enable and a translation offset (located at offsets 0x418 – 0x420).

The address space is separated into the following areas:

- The upper 4 Kbytes defines the Tsi148 CRG
- The remaining 508 Kbytes maps to the PCI/X bus.
  - When an access is initiated on the VMEbus using A24 CR/CSR AM code, Tsi148 initiates an access on the PCI/X bus when the CR/CSR offset register is enabled.

---

**Figure 4: CR/CSR Register Space**

<table>
<thead>
<tr>
<th>512 Kbyte CR/CSR</th>
<th>4 Kbyte</th>
<th>Tempe CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>508 Kbyte</td>
<td>Maps to PCI Bus</td>
<td></td>
</tr>
</tbody>
</table>

512 Kbyte CR/CSR area is defined in the VME64 Extensions Standard

0x00000 to 0x7FFFF
1.6 DMA Controllers

Tsi148 has two internal, independent, single channel DMA controllers for high performance data transfers. DMA operations between the source and destination bus are managed as separate transactions through the Linkage Module. Transactions are buffered in each DMA controller’s 64-bit by1024 (8 Kbyte) entry buffer. The Tsi148 DMA Controllers support both Direct mode and Linked-list mode operation.

There are no restrictions on addressing alignment or transfer sizes (transfer sizes can range anywhere from 1 byte to 4 Gbytes). There is support for transfer throttling through programmable transaction block sizes. There is also a back-off timer, which enables DMA transfers to occur in certain (programmable) periods of time. Parameters for DMA transfers are configured by software, or linked-list, activity.

The principal mechanism for DMA transfers is the same for operations in either direction (PCI-to-VMEbus, or VMEbus-to-PCI), only the identity of the source and destination bus changes. In a DMA transfer, the Tsi148 gains control of the source bus and reads data into the read buffer of the source master, then passes the data through the Linkage Module and into the DMA data buffer. The DMA controller then requests a transaction through the linkage and passes the data through the linkage and into the destination write buffer. The destination master then acquires the destination bus and empties its write buffer.

The DMA controller can be programmed to perform multiple blocks of transfers using Linked-list mode. The DMA works through the transfers in the linked-list following pointers at the end of each linked-list entry. Linked-list operation is initiated through a pointer in an internal Tsi148 register, but the linked-list itself resides in PCI/X memory.

For more information on Tsi148’s DMA Controller refer to Section 4. on page 101.
1.6.1 **Data Movement**

The DMA controllers support the following data movement scenarios:

- **PCI/X-to-VME**: Data is read from PCI/X and written to VME. The DMA buffer is emptied while being filled.
- **VME-to-PCI/X**: Data is read from VME and written to PCI/X. The DMA buffer is emptied while being filled.
- **PCI/X-to-PCI/X**: Data is read from PCI and written back later.
- **VME-to-VME**: Data is read from VME and written back later.
- **Data Pattern-to-VME**: Data pattern written into DMA buffer, then written to VMEbus. The DMA buffer is emptied while being filled.

Data patterns can be used for system debugging or clearing registers.

- **Data Pattern-to-PCI/X**: Data pattern written into DMA buffer, then written to PCI/X bus. The DMA buffer is emptied while being filled.

1.7 **Interrupter and Interrupt Handler**

Tsi148 can be programmed to act as interrupter and an interrupt handler in a VME system. As an interrupter, Tsi148 is capable of asserting interrupts on IRQ[7:1]O.

As an interrupt handler, Tsi148 has several VMEbus Interrupt Acknowledge registers which, when read, generate an IACK cycle on the VMEbus (see Section 8.4.70 on page 261).
1.8 **JTAG**

Tsi148 has a dedicated user-accessible test logic that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture*; also referred to as JTAG (Joint Test Action Group).

For more information on Tsi148’s JTAG capability refer to Section 7. on page 139.
2. VME Interface

This chapter describes the main features and functions of the Tsi148 VME Interface. The following topics are discussed:

- “Overview of the VME Interface” on page 50
- “VME Slave” on page 50
- “VME Master” on page 61
2. VME Interface

2.1 Overview of the VME Interface

The Tsi148 VMEbus Interface is compliant with the following standards:

- American National Standard for VME64
- American National Standard for VME64 Extensions
- Source Synchronous Transfer (2eSST) Standard

The interface is separated into the VME Slave and VME Master modules. Each module is discussed in the following sections.

2.2 VME Slave

The VME Slave is responsible for tracking and maintaining coherency to the VMEbus protocols. The VME Slave supports A16, A24, A32, and A64 address spaces and D8, D16, D32, and D64 data transfer sizes. The VME Slave supports SCT, BLT, MBLT, 2eVME, and 2eSST protocols.

During a read transaction, the VME Slave does not assert the DTACK* signal to acknowledge the data until after the data has been received from the PCI/X bus. During write transactions, the VME Slave posts the data into the write buffer. The VMEbus considers the write complete, and Tsi148 manages the completion of the write posted transaction on the PCI/X bus.

All transactions are completed on the PCI/X bus in the same order that they are completed on the VMEbus. A read transaction forces all previously issued posted write transactions to be flushed from the write buffers. All posted write transfers are completed before a read is begun to make sure that all transfers are completed in the order issued.

2.2.1 VME Slave Buffers

The VME Slave has a single read buffer that stores command information when servicing a transaction from the VMEbus, and receives the read data from the Linkage Module after the PCI/X Master has retrieved the data from the PCI/X bus. The read buffer is segmented into two parts: a data queue and a command queue. The command queue stores address and command information for a single VMEbus transaction. The amount of data in read buffer depends on the type of transaction requested. The data queue can store up to 2 Kbyte of data.

The single write buffer receives data and commands from the VMEbus. The write buffer is segmented into two parts: data queue and command queue. The data queue designed for large burst transfers and supports up to 4 Kbyte of data. The command queue stores address and command information and can accept six entries. The write buffer is considered full when either the command or data queue is full.
2.2.1.1 **Transaction Mapping**

The VMEbus is capable of many different transaction types, including one to four byte single beat transactions and burst transactions. These transactions must be mapped to corresponding transactions on the PCI/X bus. The Tsi148 supports all the different modes and protocols supported by the PCI/X bus and has numerous programmable options. Because of this flexibility there are many possible types of transactions between VME and PCI/X. The following rules can be applied to transactions:

1. A one, two, three, or four byte read or write on the VMEbus always maps to a corresponding read or write on the destination bus. VMEbus block reads can cause data to be prefetched from the PCI/X bus. Any locations with read sensitive bits should be accessed using a Single cycle Transaction (SCT) read that matches the width of the location. There is a one-to-one correspondence between the bytes written on the VMEbus and bytes written on the PCI/X bus.

2. The VME Slave does not merge, combine, or gather transactions. A transaction that completes in a single bus tenure on the VMEbus may not complete in a single bus tenure on the destination bus.

3. The VME Master does not generate the two and three byte unaligned transactions defined in the *American National Standard for VME64*.

**VMEbus-to-PCI Address Mapping**

The VME Slave interface maps a VMEbus address to the PCI/X bus address space using eight programmable slave images (see Section 8.4.46 on page 229). These slave images provide windows into the PCI/X bus from the VMEbus. The VMEbus address is compared with the address range of each slave image, and if the address falls within the specified range, an offset is added to the incoming address to form the PCI/X bus address.

The incoming address is within the slave images window if the incoming address is greater than or equal to the starting address and less than or equal to the ending address.

All programmable slave images should decode unique address ranges. However, if the slave images overlap, slave image zero has the highest priority and slave image seven has the lowest priority.
The address space of the current VME transaction determines how address comparisons are performed. The following list gives example programming and comparisons for the address mapping:

- If the VMEbus address is 64-bits, then bits 31 to 0 of the starting address in the Inbound Translation Starting Address Upper (ITSAUx) register (see Section 8.4.46 on page 229) and bits 31 to 16 of the starting address in the Inbound Translation Starting Address Lower (ITSALx) register (see Section 8.4.51 on page 234) and bits 31 to 0 of the ending address in the Inbound Translation Ending Address Upper (ITEAUx) register (see Section 8.4.48 on page 231) and bits 31 to 16 of the ending address in the Translation Ending Address Lower (ITEALx) register (see Section 8.4.49 on page 232), are compared against VMEbus address bits 63 to 16.

- If the VMEbus address is 32-bits, then bits 31 to 16 of the starting address in the ITSALx register and bits 31 to 16 of the ending address in the ITEALx register are compared against VMEbus address bits 31 to 16. The granularity is 64 Kbytes.

- If the VMEbus address is 24-bits, then bits 23 to 12 of the starting address in the ITSALx register and bits 23 to 12 of the ending address in the ITEALx register are compared against VMEbus address bits 23 to 12. The granularity is 4 Kbytes.

- If the VMEbus address is 16-bits, then bits 15 to 4 of the starting address in the ITSALx register and the bits 15 to 4 of the ending address in the ITEALx register are compared against VMEbus address bits 15 to 4. The granularity is 16 bytes.

There are no limits imposed on how large an address space a slave image can represent.

Each slave image has a set of attributes that are used to enable the image and define the VMEbus transfer characteristics. Each image has an attribute register (see Section 8.4.52 on page 235) with the following fields:

- Image enable
- Programmable threshold for read-ahead prefetching
- Programmable virtual FIFO size for inbound prefetch reads
- 2eSST slave response rate control: 160, 267, or 320 MB/s
- Slave response control: SCT, BLT, MBLT, 2eVME, 2eSST, 2eSST broadcast
- Slave address space response control: A16, A24, A32, or A64
- Slave response control: VMEbus non-privileged, supervisory, program and data access cycles
Each slave image also includes a programmable address offset. The offset is added to the VMEbus address, and the result is used as the PCI/X bus address. Figure 5 shows the programmable starting address, ending address, and translation offset.

**Figure 5: Slave Image Programmable Address Offset**

In Figure 5 the width of the starting address, the ending address, and the translation offset depends on the VME address bus size. In Figure 5 this dependency is represented by A?.

The following lists illustrates the address translation process for various VMEbus address spaces:

- If the VMEbus address is 64-bits, then bits 31 to 0 of the offset in the Inbound Translation Offset Upper (ITOFUx) register (see Section 8.4.50 on page 233) and bits 31 to 16 of the offset in the Inbound Translation Offset Lower (ITOFLx) register (see Section 8.4.51 on page 234) are added to VMEbus address bits 63 to 16.

- If the VMEbus address is 32-bits, then the incoming VMEbus address bits 63 to 32 are forced to zero and then bits 31 to 0 of the offset in the ITOFUx register and bits 31 to 16 of the offset in the ITOFLx register are added to VMEbus address bits 32 to 16.

- If the VMEbus address is 24-bits, then the incoming VMEbus address bits 63 to 24 are forced to zero and then bits 31 to 0 of the offset in the ITOFUx register and bits 31 to 12 of the offset in the ITOFLx register are added to VMEbus address bits 24 to 12.

- If the VMEbus address is 16-bits, then the incoming VMEbus address bits 63 to 16 are forced to zero and then bits 31 to 0 of the offset in the ITOFUx register and bits 31 to 4 of the offset in the ITOFLx register are added to VMEbus address bits 16 to 4.
2. VME Interface

2.2.1.2 VME Slave Transactions
The Tsi148 VMEbus Interface supports different transaction types, including one to four byte single beat transactions, and burst transactions. These transactions must be mapped to corresponding transactions on the destination bus. For more information on transaction mapping, refer to Section 2.2.1.1 on page 51.

VME Slave Read Transaction
VME Slave read operation depends on whether the transfer is a block or single cycle. If the transfer is a SCT, the VME Slave requests a single beat transfer from the Linkage Module. The VMEbus acknowledgement is held until the data is received from PCI/X.

If the read operation is a block transfer, the VME Slave requests a block of data from the Linkage Module. The VME Slave read buffer is used to store the data received from the Linkage Module. The data is stored in the buffer until it is needed to complete a VMEbus transaction.

The VME Slave read buffer has a programmable virtual buffer size and refill threshold. This flexibility enables the buffer to be optimized for various block sizes. The virtual buffer size can be set to 64, 128, 256 or 512 bytes. The virtual buffer size and refill threshold are programmable in the Slave Image registers (see Section 8.4.46 on page 229).

When the VME Slave receives a BLT or MBLT read command, the VME Slave prefetches data (through the Linkage Module to get the data) based on the virtual buffer size.

Prefetching is not used during Single Cycle Transfers.

As data is removed from the VME Slave read buffer, it is refilled based on the refill threshold. The refill threshold can be set to half-full or empty. When the refill threshold is set to half-full, the VME Slave read buffer is refilled when it is less than half-full. This functionality enables the VMEbus master to read data from PCI/X without interruption. In applications where the packet size is small, the data from the initial read can be all that is required and reading additional data would waste PCI/X bus bandwidth. In this case, the refill threshold can be set to empty to conserve bandwidth on the PCI/X bus. If the buffer is drained and additional data is required by the VME Master, the buffer is refilled based on the buffer size and address.

When the VME Slave receives a 2eVME or 2eSST read command, the prefetch size is determined by the byte count received from the VMEbus master. The entire byte count is read on the PCI/X bus. Tsi148 supports the maximum 2eVME/2eSST byte-count of 2 Kbytes.
Example VME Slave Read Transaction

In this example VME-to-PCI-X read, the transaction is separated into Request and Completion phases. The following list, and Figure 6, show the steps taken in the first part of the transaction (Request) and in the second part of the list, and Figure 7, shows the next part of the transaction (Completion).

1. A VMEbus master initiates a SCT, BLT, MBLT, 2eVME, or 2eSST read request to a PCI/X peripheral.

Figure 6: VMEbus to PCI/X Read Request

2. Tsi148 stores the command and address information, as well as byte count information (if it is a a 2eVME or 2eSST request) in the VME Slave’s read buffer command queue
   — Tsi148 supports one read request at a time
3. The VME Slave makes a read request to the Linkage Module. The initial amount of data requested is determined by the block transfer type.
   — If the transaction is a SCT, the VME Slave requests a single beat transfer from the Linkage Module.
   — If the transaction is a block transfer the VME Slave uses the virtual size buffer to determine how many bytes to request from the Linkage Module.
   — Since 2eVME and 2eSST transfers include a byte count, the VME Slave requests the entire byte counts.
4. After arbitration, the Linkage Module command and address information is passed to the PCI/X Master’s read buffer command queue. The PCI/X Master’s command queue is six entries deep.
5. The PCI/X Master issues the read request to the PCI/X target.
6. The PCI/X target satisfies the read request and the data is stored in the PCI/X Master’s read buffer data queue.

7. The PCI/X Master makes a request to the Linkage Module.

8. After arbitration, the read data is passed through the Linkage Module to the VME Slave’s read buffer data queue. The 2 Kbyte VME Slave’s read buffer data queue is used to store data received from the Linkage Module.

9. Once the VME Slave’s read buffer data queue is full (based on the virtual size programmed or byte count received), the read data is passed to the initiating VMEbus master.
2. VME Interface

— If the AS signal is asserted and the refill threshold has been reached in the VME Slave’s read buffer data queue, the VME Slave requests the Linkage Module to return to the PCI/X bus for more data.

**VME Slave Write Transaction**

During write transactions, the external master posts write data into the write buffer. All writes are posted and the write buffer stores the data necessary to complete the transfer and immediately acknowledges the transaction on the VMEbus. Tsi148 manages the completion of the posted write transaction.
Example VME Slave Write Transaction

In this example VME-to-PCI/X write transaction, the data passes through Tsi148 through the VME Slave, to the Linkage Module, and ends at the PCI/X Master. The following list, and Figure 8, show the steps taken in the write transaction.

1. A VMEbus master initiates a write to a PCI/X target.

Figure 8: VMEbus to PCI/X Write

2. The VME Slave queues the address and command information within its write buffer command queue. The command queue is six entries deep.
   - All write transactions are posted within the VME Slave’s write buffer data queue.
   - The VME Slave’s write buffer data queue is 4 Kbytes.
3. Once the transaction completes on the VME bus (that is, all the data is placed within the VME Slave’s write buffer data queue) the VME Slave sends a request to the Linkage Module.

4. After arbitration by the Linkage Module, the command and address information, as well as the write data, is passed to the PCI/X Master’s write command and data queue.
   — The PCI/X Master’s write buffer data queue is 4 Kbytes and the command queue (which is used to store commands from Linkage Module) is six entries deep.

5. The PCI/X Master completes the write transaction to the PCI/X target.

### 2.2.1.3 VME Slave Read-Modify Write (RMW) Cycles

The Tsi148 VME Slave responds to RMW cycles. The VME Slave does not complete VMEbus RMW cycles as indivisible cycles on the PCI/X bus. The PCI bus LOCK_ signal is not supported by the Tsi148 PCI/X Master and therefore the read and write cycles are indivisible on the PCI bus.

For information on how Tsi148 generates RMW cycles as a VME Master, refer to Section 2.3.3 on page 62.

### 2.2.1.4 Terminations

The VME Slave can terminate a SCT, BLT, or MBLT cycle with a DTACK_ signal or a RETRY_ signal. The VME Slave never terminates a SCT, BLT, or MBLT cycle with a BERR_ signal.

All 2eVME and 2eSST cycles are terminated with a normal termination or retry signal. The VME Slave never terminates a 2eVME or 2eSST cycle with a slave termination or error termination.
2. VME Interface

2.3 VME Master

The VME Master provides the interface from Linkage Module to the VMEbus. The VME Master can generate A16, A24, A32, and A64 VMEbus address cycles and D8 even, D8 odd, D16, D32, and D64 data transfers. The VME Master generates transfers using the SCT, BLT, MBLT, 2eVME, and 2eSST protocols. The VME Master supports the VMEbus RETRY_ signal.

2.3.1 Addressing Capabilities

The Tsi148's VMEbus addressing mode is controlled by programming the Outbound Translation Attribute registers (see Section 8.4.26 on page 191). The Tsi148 is capable of generating A16, A24, A32, A64, and CR/CSR address phases. The address mode and type (supervisor and program) are also programmed through the Outbound Translation Attribute registers.

The address and Address Modifier (AM) codes that are generated by the Tsi148 are functions of the mapping of the PCI/X memory space as defined above or through DMA programming (see Section 8.4.88 on page 297 and Section 8.4.89 on page 301). Table 1 shows the AM codes used for the VMEbus.

Table 1: VMEbus Address Mode Codes

<table>
<thead>
<tr>
<th>AMODE</th>
<th>Address Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>A16</td>
</tr>
<tr>
<td>0x0001</td>
<td>A24</td>
</tr>
<tr>
<td>0x0010</td>
<td>A32</td>
</tr>
<tr>
<td>0x0011</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0100</td>
<td>A64</td>
</tr>
<tr>
<td>0x0101</td>
<td>CR/CSR</td>
</tr>
<tr>
<td>0x0110</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0111</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x1000</td>
<td>User1 (AM 0x0100xx)</td>
</tr>
<tr>
<td>0x1001</td>
<td>User2 (AM 0x0101xx)</td>
</tr>
<tr>
<td>0x1010</td>
<td>User3 (AM 0x0110xx)</td>
</tr>
<tr>
<td>0x1011</td>
<td>User4 (AM 0x0111xx)</td>
</tr>
<tr>
<td>0x1100</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
2. VME Interface

There are four user defined AM codes. When the user defined AM codes are used, the AM[1] bit is defined by the VMEbus Supervisory Mode (SUP) bit and the AM[0] bit is defined by the VMEbus Program Mode (PGM) bit in the Outbound Translation Attribute register (see Section 8.4.26 on page 191).

### 2.3.2 VME Master Buffers

The VME Master interfaces to the Linkage Module through separate read and write buffers. The VME Master has two write buffers and two read buffers.

The read buffers are each segmented into two parts: the data queue and the command queue. The read buffers are used to store data received from the VMEbus. The data queue can accept up to 4 Kbytes of data. The command queue stores a single entry. The two read buffers allows the Tsi148 to perform back-to-back reads from the VMEbus.

The write buffers are each segmented into two parts: the data queue and the command queue. The data queue can have up to 4 Kbytes of data. The command queue can accept one entry. The write buffers are used to receive writes from the Linkage Module. The two write buffers allow the VME Master to accept two Linkage Module commands. The two write buffers allows the Tsi148 to perform back-to-back writes from the VMEbus.

### 2.3.3 VME Master Read-Modify Write (RMW) Cycles

A RMW cycle allows the VME Master to read from a VMEbus slave and then write to the same resource without relinquishing bus tenure between the two operations. RMW cycles can be generated by the Tsi148 VME Master. The VME Master generates RMW cycles on 8, 16, and 32-bit aligned transfers. For more information on the VME RMW registers, refer to Section 8.4.29 on page 197.

The following registers are used when the RMW functionality is enabled:

- The VMEbus RMW Address Upper (RMW AU) and VMEbus RMW Address Lower (RMW AL) registers: These registers specify the PCI/X address, both the upper bits (63:32) and lower bits (31:2), for the RMW cycle.
- **VMEbus RMW Enable (RMWEN):** This register defines the bits that are involved in the compare and swap operations of the RMW cycle.

- **VMEbus RMW Compare (RMWC):** This register defines the bits which are compared with the data read from the VMEbus.

- **VMEbus RMW Swap (RMWS):** This register defines the bits written to the VMEbus when the compare is successful.

The following steps are used to perform RMW cycles on the VMEbus (see Figure 9).

1. A PCI/X bus read access address matches the Target Address
   - The Target Address must be mapped to the VMEbus by one of the PCI/X bus-to-VME Slave images.
2. The VME Master reads the data at the Target Address.
3. The VME Master completes the read on the VMEbus.
4. The data read from the Target Address is compared with the data in the Compare register.
   - The bits in the RMW Enable register determine which bits are compared.
5. When the enable register is set and the compare is true, the enabled bits which compare are replaced with the data in the swap register and are written to the VMEbus. The bits which do not compare are written to the VMEbus without modification.
6. The data read from the VMEbus is returned to the PCI/X Master.
2. VME Interface

Figure 9: Steps Used to Perform RMW Cycles on the VMEbus

For information on how Tsi148 responds to RMW cycles as a VME Slave, refer to Section 2.2.1.3 on page 60.

2.3.4 VME Master Bandwidth Control

The VME Master has features to control VMEbus usage which can all be programmed in the VME Master Control register (see Section 8.4.33 on page 201). The features include the following:

- Time-on timer
  - The time-on timer specifies the length of time the VME Master can use the VMEbus.
• Time-off timer
  — The time-off timer specifies the length of time the VME Master must wait before
  re-requesting the VMEbus.

• Release mode control
  — The release mode control bits define when the VME Master releases the VMEbus.

The VME Master requests the VMEbus when it receives a command from the Linkage
Module or if a previously received command is not completed and the time-off timer has
expired. Once the VME Master has acquired the VMEbus, it maintains bus ownership until
one of the release conditions is met (see Section 2.3.4.1 on page 65).

2.3.4.1 VME Master Release Conditions
Tsi148 releases control of the VMEbus after it has been granted control as the VME Master
when one of the following VMEbus release conditions are met:

• The VMEbus is released when the time-on timer has reached its terminal count or the
  master is done

  The term done means the VME Master has completed the transfer and has no
  other requests in the queue.

• The VMEbus is released when the time-on-timer has reached its terminal count and there
  is a VMEbus request or the VME Master is done. This mode enables the master to
  continue using the VMEbus, if no other master is requesting the bus, even though the
time has expired.

• The VMEbus is released when the time-on timer has reached its terminal count and the
  VMEbus BCLR_ signal is asserted or the VME Master is done. This mode enables the
  master to continue using the VMEbus, if no other master is requesting the bus at a higher
  priority, even though the time has expired. The BCLR_ signal is asserted by the arbiter
  when a higher priority request is received (see Section 8.3.2 on page 159).

• The VMEbus is released when the time-on timer has reached its terminal count or the
  VME Master is done and there is a VMEbus request. This enables the VME Master to
  maintain VMEbus ownership even when there is no transfer in progress. Bus mastery is
  maintained until another master requests the bus.
2.3.5 **VMEbus Exception Handling**

When a VMEbus transfer initiated by the VME Master does not complete successfully, the status is saved in the VMEbus exception registers. The exception registers are updated when a transaction is terminated with a bus error, or a 2eVME or 2eSST transfer is terminated with a slave termination.

The VMEbus exception registers include:

- VMEbus Exception Address Upper (VEAU)
- VMEbus Exception Address Lower (VEAL)
- VMEbus Exception Attributes (VEAT)

For more information on the VMEbus exception registers, refer to Section 8.4.38 on page 217).

When the VME Master encounters one of these conditions, any write data in the buffers is removed (flushed). If the transaction was a VMEbus read, the VME Master completes the Linkage Module command by filling the buffer with a data pattern of all ones.

If a second exception occurs before the software has acknowledged the first exception, the status registers are not updated, however, the overflow bit is set to indicate that more than one exception occurred.

![Tip]
The interrupt controller can be programmed to generate an interrupt when the exception registers are updated.

2.3.6 **Utility Functions**

Tsi148 provides the following VMEbus utility functions:

- VMEbus Location Monitor which allows one VMEbus board to broadcast an interrupt to multiple boards. The processor sends an interrupt by reading, or writing to, one of the VMEbus monitored addresses. Other boards in the system monitor this address and interrupt their processors when an access is detected. The monitored VMEbus addresses are programmable and works in A16, A24, A32, and A64 VMEbus address space (see Section 2.3.6.1 on page 67).
  
  Three registers are provided for this function: Location Monitor Base Address Upper register (LMBAU), Location Monitor Base Address Lower register (LMBAL), and Location Monitor Attribute register (LMAT) (see Section 8.4.62 on page 250).
  
- Eight Semaphore registers for resource sharing (see Section 2.3.6.2 on page 68).
  
- Four Mailbox registers used to provide a communication path between the VMEbus and PCI/X Logic (see Section 2.3.6.3 on page 69).
2. VME Interface

- Logic is provided to generate VMEbus system fail and board fail signals (see Section 5.4.2.4 on page 132).
- Power-up options are provided for CR/CSR Base Address Configuration. Tsi148's VME Slave can be configured at power-up to use 1 of 2 methods: Geographical Address Implementation or Auto Slot ID (see Section 5.4.2 on page 129).
- Supports the following two non-standard VMEbus features: Broadcast Interrupt, Clock and 64-bit Counter (see Section 2.3.6.4 on page 69).

### 2.3.6.1 VMEbus Location Monitor

Location monitor functionality allows one VMEbus board to broadcast an interrupt to multiple boards. All boards which are participating in the broadcast are programmed to monitor a set of VMEbus addresses.

The Tsi148 location monitor is enabled in the Location Monitor (LMAT) Register by setting the Enable (EN) bit (see Section 8.4.64 on page 252). The monitored VMEbus addresses are programmable in the Location Monitor Base Address Upper (LMBAU) register (see Section 8.4.62 on page 250) and Location Monitor Base Address Lower (LMBAL) register (see Section 8.4.63 on page 251).

The location monitor can monitor addresses in VMEbus A16, A24, A32 or A64 space. If the VMEbus address is 64-bits, then bits 31 to 0 of the base address in the LMBAU register and bits 31 to 5 of the base address in the LMBAL register are compared against VMEbus address bits 63 to 5. If the VMEbus address is 32-bit, then bits 31 to 5 of the base address in the LMBAL register are compared against VMEbus address bits 31 to 5. If the VMEbus address is 24-bits, then bits 23 to 5 of the base address in the LMBAL register are compared against VMEbus address bits 23 to 5. If the VMEbus address is 16-bits, then bits 15 to 5 of the base address in the LMBAL register are compared against VMEbus address bits 15 to 5.
The processor sends an interrupt by reading or writing one to the VMEbus monitored address. The other boards in the system monitor this address and interrupt their processors when an access is detected. There are four locations which are monitored and each location is eight bytes. VMEbus address bits 3 and 4 are used to define the specific location. Table 2 shows the relationship between the VMEbus address and the location monitor interrupt.

Table 2: Location Monitor Interrupt Addresses

<table>
<thead>
<tr>
<th>VMEbus Address</th>
<th>Location Monitor Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMBA + (0-7)</td>
<td>LM0</td>
</tr>
<tr>
<td>LMBA + (8-F)</td>
<td>LM1</td>
</tr>
<tr>
<td>LMBA + (10-17)</td>
<td>LM2</td>
</tr>
<tr>
<td>LMBA + (18-1F)</td>
<td>LM3</td>
</tr>
</tbody>
</table>

When the location monitor detects an access to one of the locations being monitored, an interrupt is sent to the interrupter. If the interrupt is enabled, then the selected INTx signal is asserted. The status of the interrupt is available in the Global Control and Status (GCSR) registers (see Section 8.2.4 on page 145) and Local Control and Status (LCSR) registers (see Section 8.2.3 on page 145).

No data is transferred during a Location Monitor access. The slave boards monitoring the location do not respond. The board generating the location monitor cycle is responsible for terminating the VMEbus cycle with a DTACK* signal. The board generating a location monitor cycle must have its location monitor enabled and programmed to monitor the location monitor address.

### 2.3.6.2 Semaphore Registers

The GCSR registers include eight semaphore registers. These semaphore registers can be used to allow processes running on the local processor and processes running on processors on other VMEbus boards to share resources. Each semaphore register is 8-bits and there are four semaphore registers in a 32-bit register. The most significant bit (bit 7) is the semaphore bit and the remaining seven bits (bits 6 to 0) are the tag field.

To gain ownership of the semaphore, a process writes to the semaphore with bit 7 set and a unique code in the tag field. The process has gained ownership if a subsequent read returns the unique code. The process releases the semaphore by setting the semaphore register to 0.

A semaphore register is only updated when bit 7 in the register is zero and a one is written to bit 7 of the register, or when a zero is written to bit 7.
2.3.6.3 **Mailbox Registers**

The GCSR includes four mailbox registers which can be used to provide a communication path between the VMEbus and the PCI/X bus. These registers support read and write access from the PCI/X bus and the VMEbus. When the least significant byte of a mailbox register is written, an interrupt is sent to the interrupter. If the interrupt is enabled, an INTx signal is generated.

RMW access to a mailbox register from the VMEbus is not guaranteed to be indivisible. The semaphore registers should be used to control access if the RMW feature is required.

2.3.6.4 **Broadcast Interrupt and 64-bit Counter**

There are two Tsi148 VMEbus features which use the IRQ[1] or IRQ[2] signal lines in a device specific way: the Broadcast Interrupt and 64-bit Counter.

When the IRQ[1] or IRQ[2] signal lines are used for the Broadcast Interrupt or 64-bit Counter features, they must not be used for VMEbus interrupt signals by any other boards. These features are not defined in the VMEbus standards. The features can be programmed in the VMEbus Interrupt Control registers (see Section 8.4.70 on page 261).

The IRQ[1] and IRQ[2] signal lines received from the VMEbus can be routed to several internal modules. They are always sent to the local bus interrupter as standard interrupts. They may be sent to the local bus interrupter as an edge sensitive interrupt or they can be sent to a 64-bit counter.

The following functions can be assigned to the IRQ[1] or IRQ[2] signal lines:

- **VMEbus Interrupter**: The VMEbus interrupter allows VMEbus interrupts to be generated as defined in the VMEbus standard. For more information see “Interrupt Controller” on page 135.
- **Programmable Pulse Generator**: This generator allows a pulse on the VMEbus IRQ[1]O or IRQ[2]O signal line to be generated. The width of the pulse generated is programmable from 120 ns to 1.97 ms in approximately 30ns increments.
- **Programmable Clock Generator**: Enables a free running clock to be generated on IRQ[1] or IRQ[2]. The period of the clock generator is programmable from 2.04us to 17.11sec in approximately 1.02us increments. This provides frequencies from 0.49MHz to 0.06Hz.
- **Fixed 0.98MHz clock**: Generated on IRQ[1] or IRQ[2] signal line.
2. VME Interface

**Broadcast Interrupt**

Although the Tsi148 IRQ[1] and IRQ[2] signals can be used as VMEbus interrupts (as defined by the *American National Standard for VME64 Extensions*), the Tsi148 can also use one of the IRQ[1] or IRQ[2] signals as a broadcast interrupt. The broadcast interrupt allows a board to send an interrupt to multiple boards. Since all the boards receive the interrupt at the same time, the interrupt can be used as a synchronizing event.

In this mode, the transmitting board transmits a pulse on the IRQ[1] or IRQ[2] signal line. The receiving boards are programmed to treat the IRQ[1] or IRQ[2] signal as an edge sensitive interrupt. There is no VMEbus interrupt acknowledge cycle for a broadcast interrupt. The interrupt is treated as a local interrupt on the receiving boards.

The transmitting board can also be a receiving board.

An interrupt can be broadcast in multiple ways. Either the pulse generator can be programmed to generate a single broadcast interrupt or the programmable clock generator can be used to generate periodic broadcast interrupts. When the pulse generator is enabled and the BIP bit is set in the VMEbus Interrupt Control register (see Section 8.4.69 on page 258), a single interrupt is broadcast.

The Broadcast Pulse Generator Timer register (see Section 8.4.67 on page 256) is used to program the pulse width. A new pulse should not be generated when the BIPs bit is set in the VMEbus Interrupt Control register. When the programmable clock generator is enabled, periodic interrupts are broadcast. The Broadcast Programmable Clock Timer register (see Section 8.4.68 on page 257) is used to program the interrupt rate.

**64-bit Counter**

There is a 64-bit counter which can be incremented by a signal on the IRQ[1] or IRQ[2] signal line. In this mode, one board transmits a clock on either the IRQ[1] or IRQ[2] signal lines and the receiving boards use this clock signal to increment their 64-bit counter. This feature provides a reference counter that is synchronized on all the boards.

The transmitting board can also be a receiving board. The clock can be derived from the programmable clock generator or the 0.98 MHz clock. When the 0.98 MHz clock is used, the 64-bit counter can provide a unique time stamp every 1.02 µs.

2.3.6.5 **SYSFAIL Operation**

For more information on SYSFAIL functionality refer to Section 5.4.2 on page 129.

2.3.6.6 **VMEbus Configuration**

For more information on VMEbus configuration refer to Section 5.4.2 on page 129.
2.3.7 **Tsi148 as a VMEbus System Controller**

The Tsi148 supports the following system controller functions:

- **VMEbus Arbiter with three modes of programmable arbitration:**
  - Priority (PRI)
  - Round-Robin-Select (RRS)
  - Single Level (SGL)

- **IACK Daisy-Chain Driver**

- **SYSRESET Driver:** Provides a global system reset

- **Global VMEbus Timer:** Monitors the VMEbus and generates a BERR_ when there is no VMEbus activity for the programmed value

- **System Clock Driver:** Generates a 16 MHz system clock

2.3.7.1 **Arbiter**

The Tsi148 VMEbus arbiter is programmable. All three of the following arbitration modes defined by the VMEbus standard are supported:

- Priority (PRI)
- Round-Robin-Select (RRS)
- Single Level (SGL)

A 16 us arbitration timer is included in the Tsi148 to prevent a bus lock-up from occurring when no requester assumes mastership of the bus after the arbiter has issued a grant. This timer can be enabled or disabled in the VMEbus Control and Status Register (see Section 8.4.34 on page 205).

2.3.7.2 **IACK Daisy-Chain Driver**

An IACK Daisy-Chain driver is included in the Tsi148 as part of the system controller functionality. This feature ensures that the timing requirements for starting the IACK Daisy-Chain are satisfied.

2.3.7.3 **SYSRESET Driver**

A SYSRESET driver is included in the Tsi148 to provide a global system reset. The SRSTO signal is asserted in the following cases: the LSRSTI_ pin is asserted, the SRESET bit is asserted in the VMEbus Control Status Register, or the PURSTI_ pin is asserted. The SRSTO signal is always asserted for at least 200 ms. SRSTO is normally connected to the VMEbus SYSRESET_ signal through an inverting open collector buffer.
2. VME Interface

2.3.7.4 Global VMEbus Timer

The Tsi148 has a VMEbus global timer that monitors VMEbus cycles and generates a BERR signal when there is no VMEbus slave response for the programmed time period. The global timer only monitors VMEbus cycles when the system controller function is enabled. The global timer is compatible with SCT, BLT, MBLT, 2eVME, and 2eSST transfers. The global time-out period can be programmed for 8, 16, 32, 64, 128, 256, 512 μs. This timer can be enabled or disabled in the VMEbus Control and Status Register (see Section 8.4.34 on page 205).

2.3.7.5 System Clock Driver

Tsi148 generates the system clock (SYSCLK) signal when it is configured as the system controller. The SYSCLK signal is in spec for the following PCI/X clock frequencies: 33.3, 66.6, 100, or 133 MHz. The SYSCLK pin is connected through an external driver to the VMEbus. SYSCLK operates at 16 MHz. The external driver is enabled through the SCON pin (see Section 8.3.2 on page 159).

2.3.7.6 Configuration

The system controller functions can be configured at power-up. The system controller functionality can be enabled or disabled, or the auto system controller (SCON) function can be used. The auto SCON function automatically enables the system controller functions when the board is installed in slot 1. Table 10 on page 134 shows the different signal combinations that enable or disable the SCON functionality.
3. PCI/X Interface

This chapter describes the main features and functions of the Tsi148™. The following topics are discussed:

- “Overview of the PCI/X Interface” on page 74
- “PCI Mode” on page 74
- “PCI-X Mode” on page 88
3. PCI/X Interface

3.1 Overview of the PCI/X Interface

The PCI/X interface can be configured to operate in PCI mode or PCI-X mode. PCI-X mode is described in Section 3.3 on page 88.

3.2 PCI Mode

Tsi148 is compliant with the *PCI Local Bus Specification (Revision 2.2)*.

3.2.1 PCI Target

The PCI Target supports the PCI protocol, 32-bit and 64-bit data transfers, and 32-bit and 64-bit addresses.

The PCI Target supports configuration cycles to PCI configuration registers and memory space accesses. The Linkage Module provides access to the Combined Register Group (CRG) and the VMEbus (see Section 8.1 on page 144 for more register information). The VME Master provides the interface between the Linkage Module and the VMEbus.

The PCI Target does not respond to PCI I/O transfers.

3.2.1.1 PCI Target Buffers

The PCI Target shares buffers between the PCI and PCI-X protocols. When the PCI/X bus is configured for PCI mode, only 512 bytes of the 4 Kbyte read buffer can be used. The read buffer is segmented into two parts: a data queue and a command queue. The command queue stores address and command information from the PCI bus and can accept one delayed transaction. The data queue stores up to 512 bytes of data.

The PCI Target stores the address and command information in the command queue when servicing a read request from the PCI bus master. The amount of data pre-fetched and stored in the read buffer is determined by the read command (see Table 3 on page 78).

The write buffer receives data and commands from the PCI bus. The write buffer is segmented into two parts: data queue and command queue. The 4 Kbyte data queue is designed for large, burst transfers. The command queue stores address and command information and can accept up to 40 entries. The write buffer is full when either the command or data queue is full.
3.2.1.2 Transaction Mapping

The PCI bus is capable of many different transaction types, including: single beat transactions, burst transactions, each with flexible byte enable patterns. These transactions must be mapped to corresponding transactions on the VMEbus. There are many different modes and protocols supported by the VMEbus and the numerous programmable options. The following rules can be applied to transactions:

- **Writes**
  - During a PCI bus write, the selected bytes on the PCI bus maps directly to the destination bus. The chip does not write to bytes on the destination bus that are not selected on the PCI bus.

- **Reads**
  - Single byte reads on PCI maps to a single byte read on the destination bus. If the PCI Master inserts initial wait states during a read transaction (IRDY_ is not asserted one clock after FRAME_), the transaction is a burst and the PCI Target prefetches data from the VMEbus based on the programming in the Outbound Translation Attribute register (see Section 8.4.26 on page 191).
  - Read line and read multiple commands from a PCI Master causes data to be prefetched from the VMEbus based on the programming in the Outbound Translation Attribute Register.

  Any locations with read sensitive bits should be accessed using a byte read or a read that matches the width of the location. There is a one-to-one correspondence between the bytes written on the PCI bus and bytes written on the destination bus. PCI bus writes with byte holes do not result in writes to the non-selected bytes.

  - The PCI Target does not merge, combine, or gather transactions. Because of the different bus widths, a single beat transaction on the PCI bus may map to a multi beat transaction on the destination bus. A transaction that completes in a single bus tenure on the PCI bus may not complete in a single bus tenure on the destination bus.

**PCI-to-VME Address Mapping**

The PCI Target has eight programmable PCI bus target images which map PCI transactions to VME address space.

The PCI Target maps a PCI address to the destination address space using eight programmable target images. These target images provide windows into the VMEbus from the PCI bus. The PCI address is compared with the address range of each target image, and if the address falls within the specified range, an offset is added to the incoming address to form the destination address.
The incoming address is within the target images window if the incoming address is greater than or equal to the starting address and less than or equal to the ending address.

All programmable target images should decode unique address ranges. However, if the target images overlap, slave image zero has the highest priority and slave image seven has the lowest priority.

Figure 10 shows the programmable starting address, ending address, and translation offset.

### 3.2.1.3 PCI Transactions

All transactions through the PCI Target are completed on the VMEbus in the same order that they are completed on the PCI bus. A read transaction forces all previously issued posted write transactions to be flushed from the buffers. All posted write transfers are completed before a read is begun to make sure that all transfers are completed in the order issued. For more information on transaction mapping, refer to Section 3.2.1.2 on page 75.

**Commands**

The PCI Target responds to the following PCI bus commands:

- Memory read
- Memory write
- Configuration read
- Configuration write
- Memory read multiple
- Dual address cycle
  - 64-bit address transactions
- Memory read line
- Memory write and invalidate
**PCI Read Transaction**

During a read, the PCI Target uses delayed transactions. Delayed read transactions are used in order to free the PCI bus from waiting for the potentially long VMEbus arbitration and transfer. The PCI Target supports one delayed read transaction. Tsi148 manages the completion of the read transaction on the VMEbus.

When the PCI Target receives a read request, the PCI Target saves the information required to complete the transfer and then retries the PCI bus master. This allows the PCI bus to be used by other PCI bus masters while Tsi148 completes the transfer. The PCI Target continues to retry the PCI bus master until the VMEbus transfer has been completed. If any other PCI bus masters try to use the PCI Target, they are retried. If the read transfer completes on the VMEbus and the PCI master does not return within 2^{15} PCI bus clocks, the read data is discarded (flushed) and the transfer is terminated. The PCI Target uses its 512 byte data queue for storing prefetched read data. A prefetch read does not extend past the ending address defined by the PCI Target Image (see Section 8.4.20 on page 185).

The PCI bus command and PCI FRAME_ signal are used to define how much data to read from the VMEbus. If FRAME_ is asserted for a single clock, the transfer is considered to be a single beat transfer (regardless of the PCI command). In this case, a single beat read command is passed to the Linkage Module. If FRAME_ is asserted for more than one clock, the transfer is considered a burst transfer and the data size depends on the PCI bus command, and the programming of the Memory Read Prefetch Disable (MRPFD) bit and the Prefetch Size (PFS) field of the Outbound Translation Attribute (OTAT\_x) registers (see Section 8.4.26 on page 191).

The size of a single beat read command depends on the size of the PCI bus. If the PCI bus is 32-bit the single beat read command transfers 4 bytes, on a 64-bit bus the command transfers 8 bytes.

If the PCI bus request is a memory read burst transfer, and the MRPFD bit is clear, the read command passed to the Linkage Module requests 32 bytes (see Table 3). If the MRPFD bit is set, a single beat read command is passed to the Linkage Module.
When a PCI bus memory read line burst transfer is received, the read command passed to the Linkage Module requests 32 bytes. When a PCI bus memory read multiple command is received, the data size depends on the PFS bits. The read sizes are 64, 128, 256, or 512 bytes. The PCI read operations are summarized in Table 3.

### Table 3: PCI Read Data Size

<table>
<thead>
<tr>
<th>PCI Transfer</th>
<th>PCI Command</th>
<th>MRPFD Bit</th>
<th>PFS Bits</th>
<th>Linkage Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Beat</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Single Beat</td>
</tr>
<tr>
<td>Burst Read</td>
<td>Read</td>
<td>1</td>
<td>X</td>
<td>Single Beat</td>
</tr>
<tr>
<td>Burst Read</td>
<td>Read</td>
<td>0</td>
<td>X</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Burst Read Line</td>
<td>Read Line</td>
<td>X</td>
<td>X</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Burst Read Multiple</td>
<td>Read Multiple</td>
<td>X</td>
<td>0</td>
<td>64 bytes</td>
</tr>
<tr>
<td>Burst Read Multiple</td>
<td>Read Multiple</td>
<td>X</td>
<td>1</td>
<td>128 bytes</td>
</tr>
<tr>
<td>Burst Read Multiple</td>
<td>Read Multiple</td>
<td>X</td>
<td>2</td>
<td>256 bytes</td>
</tr>
<tr>
<td>Burst Read Multiple</td>
<td>Read Multiple</td>
<td>X</td>
<td>3</td>
<td>512 bytes</td>
</tr>
</tbody>
</table>

The PCI bus master is retried until all the requested data is available in the PCI Target read buffer. The read then completes on the PCI bus.

Care must be used when setting the value in the PFS field because the VMEbus read is completed before data is transferred on the PCI bus. If the value is too large, time is wasted reading data that is not used. If the value is too small, additional PCI bus commands are required. The optimum setting depends on the PCI bus masters and the requirements of the application. In many cases, the only read transfers from the PCI bus to the VMEbus are single beat processor load operations and prefetching is not required.
**Example PCI Read Transaction**

In this example read, the transaction is separated into Request and Completion phases. The following list, and Figure 11 and Figure 12, show the steps taken in the first part of the transaction, Request, and in the second part of the transaction, Completion.

1. A PCI bus master initiates a read request to a VME peripheral

**Figure 11: PCI-VME Delayed Read Request**

2. The Tsi148 PCI Target decodes the request and issues a retry to the PCI bus master

3. The PCI Target stores the command and address information in the PCI Target’s read buffer command queue
   
   — Tsi148 supports one delayed read request

4. The PCI Target makes a read request to the Linkage Module
3. PCI/X Interface

Table 3 describes the PCI bus read commands and the parameters which define the command that is passed to the Linkage Module.

The Linkage Module provides a common interface for all the modules and has the following ports: VMEbus, PCI bus, DMA 0, DMA1, and registers. The Linkage Module uses a round-robin arbitration scheme to fairly arbitrate between the ports.

5. After arbitration, the Linkage Module command and address information is passed to one of the VME Master’s read buffer command queues.

6. The VME Master issues the read request to the VMEbus slave.

7. The VMEbus slave satisfies the read request and the data is stored in one of the two 4 Kbyte VME Master’s read buffer data queues.

Having two buffers to store data allows the Tsi148 VME Master to do back-to-back reads on the VMEbus.

8. When the read request is satisfied and the data is queued in the VME Master’s data buffer, the VME Master makes a return request to the Linkage Module.
9. After Linkage Module arbitration, the read data is passed through the Linkage Module to the PCI Target’s read buffer data queue.

10. Once the entire read request is queued in the PCI Target’s read buffer data queue the initial read request can be satisfied on PCI.

   — If the initiating PCI bus master makes a request for the data before the full request is satisfied in the read buffer data queue, Tsi148 retries the PCI bus master.
3. PCI/X Interface

**Write Transaction**

During write transactions, the external master writes data into the PCI Target write buffer. All writes are posted. The buffer stores the data necessary to complete a PCI write transfer and immediately acknowledges the transaction on the PCI bus. Acknowledging the transaction frees the PCI bus from waiting for the potentially long VMEbus arbitration and transfer. This allows the PCI bus to be used by other PCI bus masters while Tsi148 completes the posted write transaction on the VMEbus. If the posted write buffer is full, the PCI Target retries the PCI bus master until there is space available in the write buffer.

The PCI Target write buffer includes a 40 deep command queue and a 4 Kbyte data queue. The PCI Target write buffer stores the commands and data in any combination of single and burst transactions. When a transfer is completed on the PCI bus, the data is transferred to the Linkage Module.
**Example PCI Write Transaction**

In this example posted write transaction, the transaction completes in one phase through the device. The following list, and Figure 13, show the steps taken in the transaction.

1. An external PCI bus master initiates a write to a VMEbus slave.

**Figure 13: PCI-to-VME Posted Write**

2. The PCI Target puts the address and command information in its command queue
   - All write transactions are posted within the PCI Target’s write buffer data queue. The PCI Target’s write buffer data queue is 4 Kbytes.

3. The PCI Target puts the corresponding data into its data queue

4. The PCI Target accepts write data until the write buffer fills or the transaction ends.

5. The PCI Target then sends a transaction request to the Linkage Module.
6. After arbitration, the Linkage Module passes the command information, address information, and the write data to one of the VME Master’s write buffers.

   Having two write buffers allows the VME Master to accept two write transactions from the Linkage Module.

7. The VME Master completes the write transaction to the addressed VMEbus slave.

**Transaction Terminations**

The PCI Target can terminate a transaction with a retry or a disconnect. The PCI Target terminates the transaction with a retry in the following cases:

- The transaction is a memory write and the PCI Target write buffer is full
- The transaction is the first transaction of a memory read
- The transaction is a memory read that does not match a pending memory read
- The transaction is a memory read that matches a pending memory read but the data is not available

The PCI Target terminates the transaction with a disconnect in the following cases:

- A write with byte holes is received and the Stop on Byte Holes (SBH) bit in the PCI Control Register is set (see Section 8.4.36 on page 211)
- A transfer reaches the end of a PCI Target image
- The burst ordering is non-linear
- A burst read requires more data than was prefetched
- A write burst fills the PCI Target write buffer

The Tsi148 PCI Target never terminates a transaction with a Target-abort.
3.2.2 PCI Master

The PCI Master provides the interface between Linkage Module and the PCI bus. The PCI Master supports a 32-bit and 64-bit data bus and 32-bit and 64-bit addresses.

3.2.2.1 PCI Master Commands

The PCI Master can generate the following PCI bus commands:

- Memory read
- Memory write
- Memory read multiple
- Dual address cycle
- Memory read line

3.2.2.2 PCI Master Buffers

The PCI Master has one read buffer and one write buffer. The buffers are segmented into two parts: a data queue and a command queue. Both the read and write buffer command queues are six entries deep. The read buffer data queue is 512 byte while the write data buffer is 4 Kbyte.

The read buffer stores Linkage Module commands when servicing a read request from the VMEbus to the PCI bus. The PCI Master requests the PCI bus when it receives a read command from the Linkage Module. After the read transaction has been satisfied on the PCI bus, and the PCI read buffer data queue has the requested data, the PCI Master transfers the data through the Linkage Module to the VMEbus.

The write buffer stores Linkage Module commands and data. The PCI Master requests the PCI bus when it receives write data from the Linkage Module. The write buffer is considered full when either the command queue or data queue is full.

3.2.2.3 PCI Master Bandwidth Control

The PCI bus latency timer can be used to control the PCI bus bandwidth used by Tsi148. The PCI Master requests the PCI bus when it has a transaction to complete (for example, when the PCI Master receives a command from the Linkage Module or when the master needs to complete a previously received command). The PCI Master maintains mastership of the PCI bus until the Linkage Module command is completed or until the PCI bus grant is removed and the latency timer has expired.
3.2.3 PCI Bus Exception Handling

Tsi148 includes error diagnostic registers which capture information when an error occurs. The information captured includes the PCI bus address and command (see Section 8.4.41 on page 222). The error diagnostic registers are updated when the first error occurs. If another error occurs before software has examined the registers, the information is not captured and the overflow bit is set.

The following list details the error diagnostic registers in Tsi148:

- Error Diagnostic PCI Address Upper (EDPAU)
- Error Diagnostic PCI Address Lower (EDPAL)
- Error Diagnostic PCI Attributes (EDPAT)

3.2.3.1 PCI Master Exception Handling

The error diagnostic registers are updated when Tsi148 is PCI Master and one of the following errors occurs: the master retry count is exceeded (programmed in the PCI Control / Status Register, see Section 8.4.36 on page 211), a Master-abort or Target-abort is received.

When the PCI Master receives a Master-abort, Target-abort, or the maximum count is exceeded the following steps are taken:

- Returns all FFs on VMEbus with the DTACK signal
- Log status information and update PCI bus error diagnostic registers
- Optional step: generate interrupt

When the PCI Master detects a data parity error the following steps are taken:

- Generate PERR (if enabled)
- Log status information and update PCI bus error diagnostic registers
- Optional step: generate interrupt

3.2.3.2 PCI Target Exception Handling

The error diagnostic registers are updated when the PCI Target detects an address parity error, a data parity error, or a delayed transaction time-out occurred.
When the PCI Target detects a address parity error the following steps are taken:

- Generate SERR (if enabled)
- Log status information

When the PCI Target detects a data parity error the following steps are taken:

- Generate PERR (if enabled)
- Log status information and update PCI bus exception registers
- Optional step: generate interrupt

When the PCI Target detects a delayed transaction time-out the following steps are taken:

- Discard Data
- Log status information and update PCI bus exception registers
- Optional step: generate interrupt

If the PCI Target detects the assertion of the SERR_ signal, no action is taken.
3.3 **PCI-X Mode**

Tsi148 is compliant with the *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b)*.

### 3.3.1 PCI-X Target

The PCI-X Target supports 32-bit and 64-bit data transfers and 32-bit and 64-bit addresses.

The PCI-X Target supports configuration cycles to PCI-X configuration registers and memory space accesses. The Linkage Module provides access to the combined register group and the VMEbus. The VME Master provides the interface between the Linkage Module and the VMEbus.

The PCI-X Target does not respond to PCI-X I/O transfers.

### 3.3.1.1 PCI-X Target Buffers

The PCI-X Target shares buffers between the PCI and PCI-X protocols. When the PCI-X bus is configured for PCI-X mode, the entire 4 Kbyte PCI-X Target read buffer can be used. The read buffer is segmented into two parts: a data queue and a command queue. The command queue stores address and attributes from the PCI-X bus and can accept up to six split transactions. The data queue stores up to 4 Kbyte of data.

The PCI-X Target read buffer stores the address and attributes of the transaction in the command queue when servicing a read request from the PCI-X bus master. The requested data comes from the VMEbus, through Linkage Module, to the PCI-X Target read buffer data queue. The amount of data in read buffer depends on the requested byte-count in the attribute phase of the PCI-X transaction.

The write buffer receives data and commands from the PCI-X bus. The write buffer segmented into two parts: data queue and command queue. The 4 Kbyte data queue is designed for large, burst transfers. The command queue stores address and attributes from PCI-X transactions and can accept up to 40 entries. The write buffer is full when either the command or data queue is full.
3.3.1.2 **Transaction Mapping**

The PCI-X bus is capable of many different transaction types, including: single beat transactions, burst transactions, each with flexible byte enable patterns. These transactions must be mapped to corresponding transactions on the VMEbus. There are many different modes and protocols supported by the VMEbus and the numerous programmable options. The following rules can be applied to transactions:

- **Writes**
  - During a PCI-X bus write, the selected bytes on the PCI-X bus map directly to the destination bus. The Tsi148 does not write to bytes on the destination bus that are not selected on the PCI-X bus.
  - During a PCI-X bus memory write block, the number of bytes in the byte count, along with the starting address map directly to the destination bus.

- **Reads**
  - The PCI-X bus protocol includes a byte count. The number of bytes requested from the destination bus generally matches the byte count requested by the PCI-X bus master.
  - The PCI-X Target does not merge, combine, or gather transactions. Because of the different bus widths, a single beat transaction on the PCI-X bus may map to a multi beat transaction on the destination bus. A transaction that completes in a single bus tenure on the PCI-X bus may not complete in a single bus tenure on the destination bus.

Any locations with read sensitive bits should be accessed using a byte read or a read that matches the width of the location (preferably the memory read DWORD command). There is a one-to-one correspondence between the bytes written on the PCI-X bus and bytes written on the destination bus.

**PCI-X-to-VME Address Mapping**

The PCI-X Target has eight programmable PCI-X bus target images which map PCI-X transactions to VME address space.

The PCI-X Target maps a PCI-X address to the destination address space using eight programmable target images. These target images provide windows into the VMEbus from the PCI-X bus. The PCI-X address is compared with the address range of each target image, and if the address falls within the specified range, the offset is added to the incoming address to form the destination address.
3. PCI/X Interface

The incoming address is within the target images window if the incoming address is greater than or equal to the starting address and less than or equal to the ending address.

All programmable target images should decode unique address ranges. However, if the target images overlap, slave image zero has the highest priority and slave image seven has the lowest priority.

Figure 14 shows the programmable starting address, ending address, and translation offset.

Figure 14: Target Image Programmable Address Offset

3.3.1.3 PCI-X Transactions

All transactions through the PCI-X Target are completed on the VMEbus in the same order that they are completed on the PCI-X bus. A read transaction forces all previously issued posted write transactions to be flushed from the buffers. All posted write transfers are completed before a read is begun to make sure that all transfers are completed in the order they are issued. For more information on transaction mapping, refer to Section 3.3.1.2 on page 89.

Commands

The PCI-X Target responds to the following PCI-X bus commands:

- Memory read DWORD
- Memory write
- Configuration read
- Configuration write
- Split completion
- Dual address cycle
- Memory read block
- Memory write block
### PCI-X Read Transaction

The PCI-X Target uses split read transactions for all reads, which frees the PCI/X bus from waiting for the potentially long VMEbus arbitration and transfer. Tsi148 supports up to six split reads.

> For more information on the PCI-X implementation of split reads, refer to the *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b).*

When the PCI-X Target receives a read request, the PCI-X Target saves the information required to complete the transfer and then issues a Split Response termination to the PCI-X bus master. This allows the PCI-X bus to be used by other PCI-X bus masters while Tsi148 completes the transfer. If the PCI-X Target receives a read request from a PCI-X bus master and the PCI-X Target read buffer command queue is full, the PCI-X Target retries the PCI-X bus master until there is space available in the read buffer.

> A Split Response means PCI-X Target does not have to issue retries as the read is being completed on the VMEbus while waiting for the requested data.

After the PCI-X Target has issued the Split Response to the PCI-X bus master, the PCI-X Target then issues a read command to the Linkage Module for the requested byte count. As defined in the *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b)*, byte counts up to 4 Kbyte are supported.

When the data is returned from the VME Master through the Linkage Module to the PCI-X Target read buffer, the Tsi148 PCI-X Master initiates a Split Completion and transfers the data from the PCI-X Target read buffer to the requesting PCI-X bus master. If the requested read extends past the ending address defined by the Target Image (see Section 8.4.20 on page 185), the PCI-X Master provides data up to the end of the image and then terminates the transaction with a Split Completion Error Message to the initiating PCI-X bus master (see Section 3.3.3.1 on page 99).
Example PCI-X Read Transaction

In this example PCI-X-to-VME read, the transaction is separated into Request and Completion phases. The following list, and Figure 15, show the steps taken in the first part of the transaction (Request). The second part of the list, and Figure 16, shows the next part of the transaction (Completion).

1. A PCI-X master initiates a read request to a VMEbus slave

**Figure 15: PCI-X-to-VME Delayed Read Request**

2. The Tsi148 PCI-X Target decodes the request and issues a Split Response termination to the initiating PCI-X bus master

3. The PCI-X Target stores the command, address, and attribute information in the PCI-X Target’s read buffer command queue
   — Tsi148 supports up to six split read transactions
4. The PCI-X Target sends a read request to the Linkage Module with the VME address information and required byte count.
   - Tsi148 supports byte counts of up to 4 Kbytes

The Linkage Module provides a common interface for all the modules. The Linkage Module interface has the following ports: VMEbus, PCI bus, DMA 0, DMA1, and registers. The Linkage Module uses a round-robin arbitration scheme to fairly arbitrate between the ports.

5. After arbitration, the Linkage Module command and address information is passed to a VME Master read buffer command queue.

6. The VME Master issues the read request to the VME bus slave.

7. The VME Slave satisfies the read request and the data is stored in one of the two, 4 Kbyte VME Master read buffer data queues.

   Having two buffers to store data allows the Tsi148, through the VME Master, to do back-to-back reads on the VME bus.

8. Once the full byte count of the read request is satisfied and the data is queued in a VME Master’s buffer, the VME Master makes a return request to the Linkage Module.
9. After Linkage Module arbitration, the read data is passed through the linkage to the PCI-X Target read buffer data queue.

10. Once the entire read request is queued in the PCI-X Target’s read buffer data queue, Tsi148 issues a Split Completion through the PCI-X Master onto the PCI-X bus to the original, initiating PCI-X bus master.

11. The PCI-X Master transfers the data from the PCI-X Target’s read buffer data queue to the PCI-X bus master.
3.3.1.4 PCI-X Write Transaction

During write transactions, the external master writes data into the PCI-X Target write buffer. All writes are posted and the buffer stores the data necessary to complete a PCI-X write transfer and immediately acknowledges the transaction on the PCI-X bus. Acknowledging the transaction frees the PCI-X bus from waiting for the potentially long VMEbus arbitration and transfer. This allows the PCI-X bus to be used by other PCI-X bus masters while Tsi148 completes the posted write transaction on the VMEbus. If the posted write buffer is full, the PCI-X Target retries the PCI-X bus master until there is space available in the write buffer.

The PCI-X Target write buffer has a 40 entry command queue and a 4 Kbyte data queue (see Section 3.3.1.1 on page 88). The PCI-X Target buffer stores the commands and data in any combination of single and burst transactions. When a transfer is completed on the PCI-X bus, the data is transferred to the Linkage Module. If the PCI-X Target receives a write command that extends past the space programmed in the Target Image, the PCI-X Target accepts the data up to the end of the Target Image and then issues a Disconnect.
Example PCI-X Write Transaction

In this example PCI-X-to-VME write transaction, the data passes through Tsi148 through the PCI-X Target, to the Linkage Module, and ends at the VME Master. The following list, and Figure 17, show the steps taken in the write transaction.

1. A PCI-X master initiates a write to a VMEbus slave.

Figure 17: PCI-X-to-VME Posted Write

- The PCI-X Target puts the address and command information in its write buffer command queue.
  - All write transactions are posted within the PCI-X Target write buffer data queue. The PCI-X Target write buffer data queue is 4 Kbytes.

2. The PCI-X Target puts the corresponding data into its data queue.
4. The PCI-X Target accepts write data until the write buffer fills or the transaction ends.

5. The PCI-X Target then sends a transaction request to the Linkage Module.

6. After arbitration, the Linkage Module passes the command information, address information, and the write data to a VME Master write buffer.

   Having two sets of write buffers allows the VME Master to accept two write commands and data from the Linkage Module.

7. The VME Master completes the write transaction to the addressed VMEbus slave.

### 3.3.1.5 Transaction Termination

The PCI-X Target can terminate a transaction with many of the terminations defined in the *PCI-X Specification*. For read requests, PCI-X Target uses split read terminations.

The PCI-X Target terminates a transaction with a retry in the following cases:

- The transaction is a memory write and the write buffer is full.
- The transaction is a read and the read buffer command queue is full

The PCI-X Target terminates a transaction with a disconnect on an *address data boundary* (ADB) in the following cases:

- A transfer reaches the end of a target image
- A burst write fills the write buffer
3.3.2 PCI-X Master

The PCI-X Master can generate the following PCI-X bus commands:

- Split completion
- Dual address cycle (A dual address cycle is generated when the PCI address is greater than 32 bits)
- Memory read block
- Memory write block

3.3.2.1 PCI-X Master Buffers

The PCI-X Master has one read buffer and one write buffer. The buffers are segmented into two parts: a data queue and a command queue. Both the read and write buffer command queues are six entries deep. The read and write data buffers are 4 Kbyte.

The read buffer stores Linkage Module commands when servicing a read request from the VMEbus to the PCI-X bus. The PCI-X Master requests the PCI-X bus when it receives a read command from the Linkage Module. After the read transaction has been satisfied on the PCI-X bus, and the read buffer data queue has the requested data, the PCI-X Master transfers the data through the Linkage Module to the VMEbus.

The write buffer stores Linkage Module commands and data. The PCI-X Master requests the PCI-X bus when it receives a command and write data from the Linkage Module. The write buffer is considered full when either the command or data queue is full.

3.3.2.2 PCI-X Master Bandwidth Control

The PCI-X bus latency timer can be used to control the PCI-X bus bandwidth used by Tsi148. The PCI-X Master requests the PCI-X bus when it has a transaction to complete (for example, when the PCI-X Master receives a command from the Linkage Module or when it needs to complete a previously received command). The PCI-X Master maintains mastership of the PCI-X bus until the linkage command is completed or until the PCI-X bus grant is removed and the latency timer has expired.
3.3.3 **PCI-X Bus Exception Handling**

Tsi148 includes error diagnostic registers which capture information when an error occurs. The information captured includes the PCI-X bus address, attribute, and command (see Section 8.4.43 on page 224). The error diagnostic registers are updated when the first error occurs. If another error occurs before software has examined the registers, the information is not captured and the overflow bit is set.

The following list details the error diagnostic registers in Tsi148:

- Error Diagnostic PCI Address Upper (EDPAU)
- Error Diagnostic PCI Address Lower (EDPAL)
- Error Diagnostic PCI-X Attribute (EDPXA)
- Error Diagnostic PCI-X Split Completion Message (EDPXS)
- Error Diagnostic PCI Attributes (EDPAT)

For more information on Tsi148 error diagnostic registers, refer to Section 8.4.43 on page 224.

3.3.3.1 **PCI-X Master Exception Handling**

The error diagnostic registers are updated when Tsi148 is PCI-X Master and one of the following errors occurs: the master retry count is exceeded, a split response time-out occurs, split completion error asserted, or a Master-abort or Target-abort is received.

> **Tip**

The Tsi148 interrupt controller can be programmed to generate an interrupt, when the exception registers are updated.

When the PCI-X Master receives a Master-abort, Target-abort, or the maximum retry count is exceeded the following steps are taken:

- Return FF's on VMEbus with the DTACK signal
- Log status information and update PCI-X bus exception registers (see Section 8.4.43 on page 224)
- Optional step: generate interrupt

When the PCI-X Master detects a data parity error the following steps are taken:

- Generate PERR (if enabled)
- Log status information and update PCI-X bus exception registers (see Section 8.4.43 on page 224)
- Optional step: generate interrupt
3. PCI/X Interface

3.3.3.2 PCI-X Target Exception Handling

The error diagnostic registers are updated when the PCI-X Target detects an address parity error, a data parity error has occurred, or a unexpected split completion is received.

When the PCI-X Target detects a address parity error the following steps are taken:

- Generate SERR (if enabled)
- Log status information

When the PCI-X Target detects a data parity error the following steps are taken:

- Generate PERR (if enabled)
- Log status information and update PCI-X bus exception registers (see Section 8.4.43 on page 224)
- Optional: step generate interrupt

When the PCI-X Target receives an unexpected split completion the following steps are taken:

- The split completion is discarded and the Split Completion Discarded (SCD) bit is set in the Error Diagnostic PCI Attribute register (see Section 8.4.43 on page 224).

If the PCI-X Target detects the assertion of the SERR_ signal, no action is taken.
4. DMA Interface

Direct memory access (DMA) allows a transaction to occur between two devices without involving the host processor (for example, a read transaction between a peripheral device and host processor memory). Because less time is required to complete transactions, applications that contain one or more DMA channels support faster read and write transfers than applications that support only host-assisted transactions.

This chapter discusses the following topics about the Tsi148 DMA:

- “Overview DMA Controller” on page 102
- “Architecture” on page 102
- “DMA Buffers” on page 102
- “Operating Modes” on page 103
- “Direction of Data Movement” on page 105
4. DMA Interface

4.1 Overview DMA Controller

The Tsi148 has two independent, single channel DMA controllers that enable the transfer of large blocks of data without processor intervention. Each DMA controller is programmed by a set of registers that reside within the LCSR group (see Section 8.2.3 on page 145).

The Combined Register Group (CRG) map decoder can be programmed to allow access to the control registers from the VMEbus.

Each DMA controller supports 64-bit addressing on the VMEbus and the PCI/X bus. The amount of data moved during a command is only limited by the 32-bit byte counter, allowing transfer counts to range from 1 byte to 4 Gbytes.

4.2 Architecture

Each DMA controller connects to the Linkage Module and uses the PCI/X Master and VME Master to transfer data. The core of the DMA controller is the DMA buffer - an 8 Kbyte buffer. The buffer is used for all transactions regardless of the direction.

The DMA controllers have been optimized to transfer data over the PCI/X bus in multiple cache-line bursts. All interactions with the VMEbus are handled by the VME Master. The controllers transfer data using 32-bit or 64-bit burst transfers on the PCI/X bus and 16-bit, 32-bit, or 64-bit transfers on the VMEbus.

4.3 DMA Buffers

Each DMA controller has an 8 Kbyte buffer that is used to hold data transferred between the source and destination bus. For example, if the transfer is from the PCI/X bus to the VMEbus, the DMA controller requests data from the PCI/X Master and then sends it to the VME Master.

The data moves from the PCI/X bus into the PCI/X Master’s read buffer data queue and then through the Linkage Module to the buffer in the DMA controller. The data then moves from the DMA buffer through the Linkage Module to the VME Master’s write buffer data queue. The data is then transferred to the VMEbus.
4.4 Operating Modes

There are two operating modes for the DMA Controller: Direct mode and Linked-list mode. In Direct mode, the DMA control registers are programmed by the processor. Once the command has completed, the status of the completed command is given within the DMA status registers and an optional interrupt is asserted on the INTx signal lines (see Section 8.3.2 on page 159).

In Linked-list mode, the DMA controller executes a list of commands which are stored in system memory. The DMA fetches these commands from the PCI/X bus. Once all the commands have been fetched and executed, the status of the completed commands is given within the DMA status registers and, optionally, an interrupt is asserted on the INTx signal lines.

Figure 18 shows the direct mode of the DMA controller.

Figure 18: Direct Mode

Figure 19 shows the linked-list mode of the DMA controller.

Figure 19: Linked-list Mode
4.4.1 **Linked-List Descriptors**

The PCI/X Master is responsible for fetching descriptors from local memory when using Linked-List Mode. Each descriptor consumes 32 bytes and must be aligned on 64-bit boundaries. This structure helps minimize the PCI/X bus bandwidth used when fetching descriptors.

**Table 4** shows the format of a descriptor.

**Table 4: DMA Controller Linked-List Descriptors**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DSAU</td>
</tr>
<tr>
<td>0x08</td>
<td>DDAU</td>
</tr>
<tr>
<td>0x10</td>
<td>DSAT</td>
</tr>
<tr>
<td>0x18</td>
<td>DNLAU</td>
</tr>
<tr>
<td>0x20</td>
<td>DCNT</td>
</tr>
</tbody>
</table>

Each field within the descriptor corresponds to a DMA control register. When a descriptor is loaded by the DMA controller, each field is placed into its corresponding DMA control register (see Section 8.4.76 on page 279).

The descriptors are linked together by the DNLA register (that is, the DNLA field within a descriptor). This field contains the address within PCI address space where the next descriptor may be found. The Last Link-descriptor Address field (LLA) within the DNLA indicates that this is the last descriptor.

Descriptors are not prefetched by the PCI/X Master. A linked-list mode command is started by the PCI/X Master reading one descriptor. The DMA controller then performs the transfer associated with that descriptor. If there are more descriptors to be executed, the fetching of the next descriptor does not occur until the current transfer has completed.
4.5 Direction of Data Movement

There are four possible directions for data movement within a transfer.

- **PCI/X-to-VME**: Data is read from PCI/X and written to the VMEbus. The PCI/X Master fills the DMA buffer at the same time the VME Master empties the DMA buffer.

- **VME-to-PCI/X**: Data is read from the VMEbus and written to the PCI/X bus. The VME Master fills the DMA buffer at the same time that the PCI/X Master empties the DMA buffer.

- **PCI/X-to-PCI/X**: Data is read from the PCI/X bus and written back sometime later to the PCI/X bus. The PCI/X Master fills the DMA buffer to a certain point, after which the PCI/X Master empties the DMA buffer.

- **VME-to-VME**: Data is read from the VMEbus and written back sometime later to the VMEbus. The VME Master fills the DMA buffer to a certain point, after which the VMEbus Master empties the DMA buffer.

- **Data Pattern to VME**: A data pattern is written into the DMA buffer and then written to the VMEbus. The pattern generator fills the DMA buffer at the same time that the VME Master empties the DMA buffer. The data pattern can either be a fixed pattern or an incrementing pattern. For data pattern programming information refer to Section 8.4.89 on page 301.

- **Data Pattern to PCI/X**: A data pattern is written into the DMA buffer and then written to the PCI/X bus. The pattern generator fills the DMA buffer at the same time that the PCI/X Master empties the DMA buffer. The data pattern can either be a fixed pattern or an incrementing pattern. For data pattern programming information refer to Section 8.4.89 on page 301.

4.5.1 PCI/X-to-VME

The Tsi148 DMA controllers support PCI/X-to-VME DMA transactions.

*Example DMA PCI/X-to-VME Transaction*

In this example, there is a DMA transaction between the PCI/X bus and VMEbus. The following list, and Figure 20 and Figure 21, show the steps taken in the DMA transaction.

1. Program the registers in the LCSR group.
   
   — The DMA registers set-up the following information:
     
     - Source and destination buses, and starting address
     - Mode of operation
4. DMA Interface

- Attributes
- Bus width
- Transfer throttling
- DMA transfer count

Transfer counts can be between 1 byte and 4 Gbytes.

**Figure 20: DMA Transaction: PCI/X-to-VME Request**

2. Once these registers have been programmed, writing to the DGO bit in the DMA control register to initiates the DMA transfer.

3. The DMA controller issues a read request to the Linkage Module.
4. After arbitration the Linkage Module passes the command, address information and transfer size to the PCI/X Master read buffer command queue.

5. The PCI/X Master issues a read request to the PCI/X target.

**Figure 21: DMA Transaction: PCI/X-to-VME Completion**

6. Once the read request is satisfied or the PCI/X Master’s read buffer data queue becomes full, the PCI/X Master makes a request to the Linkage Module.
   
   — The Block Size is programmed in the PBKS field in the DMA Control register when the PCI/X bus is the source bus (see **Section 8.4.76 on page 279**).

7. After arbitration, the read data is passed through the Linkage Module to the DMA controller’s data buffer. The data buffer is used to hold data that is transferred between the source and destination bus.
8. The DMA controller issues a write request to the Linkage Module.

9. After arbitration, the Linkage Module passes the command information, address information, and write data to a VME Master’s write buffer.

10. The VME Master completes the write transaction to the VMEbus slave. For large transfers the PCI/X Master attempts to fill the DMA buffer while the VME Master transfers data from the DMA buffer.

4.5.2 VME-to-PCI/X

The Tsi148 DMA controllers support VME-to-PCI/X DMA transactions.

4.5.2.1 Example DMA VME-to-PCI/X Transaction

In this example, there is a DMA transaction between the VMEbus and PCI/X bus. The following list, and Figure 22 and Figure 23, show the steps taken in the DMA transaction.

1. Program the registers in the LCSR group.
   — The DMA registers set-up the following information:
     – Source and destination buses, and starting address
     – Mode of operation
     – Attributes
     – Bus width
     – Transfer throttling
     – DMA transfer count

   Transfer counts can be between 1 byte and 4 Gbytes.
2. Once these registers have been programmed, writing the DGO bit in the DMA control register initiates the DMA transfer.

3. The DMA controller issues a read request to the Linkage Module.

4. After arbitration, the Linkage Module passes the command information and address information to a VME Master read buffer.

5. The VME Master issues a read request to the VMEbus slave.
6. Once the read request is satisfied, or the programmed VMEbus block size value is satisfied, the VME Master makes a request to the Linkage Module.

   — The Block Size is programmed in the VBKS field in the DMA Control Register when VME is the source bus (see Section 8.4.76 on page 279).

7. After arbitration, the read data is passed through the Linkage Module to the DMA controller’s data buffer. The data buffer is used to hold data that is transferred between the source and destination bus.

8. The DMA controller then issues a write request to the Linkage Module.
9. After arbitration, the Linkage Module passes command information, address information, and the write data to the PCI/X Master write buffer’s command and data queues.

10. The PCI/X Master initiates the write transaction to the PCI/X target. For large transfers the VME Master fills the DMA buffer while the PCI-X Master attempts to transfer data from the DMA buffer.

4.5.3 **PCI/X-to-PCI/X**

The Tsi148 DMA controllers support PCI/X-to-PCI/X DMA transactions.

4.5.3.1 **Example DMA PCI/X-to-PCI/X Transaction**

In this example, there is a DMA transaction between the PCI/X bus and PCI/X bus. The following list, and Figure 24 and Figure 25, show the steps taken in the DMA transaction.

1. Program the registers in the LCSR group.
   - The DMA registers set-up the following information:
     - Source and destination buses, and starting address
     - Mode of operation
     - Attributes
     - Bus width
     - Transfer throttling
     - DMA transfer count

   Transfer counts can be between 1 byte and 4 Gbytes.
2. Once these registers have been programmed writing the DGO bit in the DMA control register initiates the DMA transfer.

3. The DMA controller issues a read request to the Linkage Module.

4. After arbitration, the Linkage Module passes the command, address information, and transfer size are passed to the PCI/X Master read buffer command queue.

5. The PCI/X Master issues a read request to the PCI/X target.
6. The PCI/X target satisfies the read request and the data is stored in the PCI/X Master’s read buffer data queue.

7. Once the read request is satisfied, or the PCI/X Master’s read buffer data queue becomes full, the PCI/X Master makes a request to the Linkage Module.

8. After arbitration the read data is passed through the Linkage to the DMA Controllers data buffer. The data buffer is used to hold data that is transferred between the source and destination bus. In this example between the PCI/X bus and PCI/X bus.

9. The DMA controller then issues a write request to the Linkage Module.

10. Upon arbitration the command, address information as well as the write data is passed to the PCI/X Master write buffer command and data queues.
11. The PCI/X Master initiates the write transaction to the PCI/X peripheral. The PCI/X Master fills the DMA buffer to a certain point, after which the PCI/X Master empties the DMA buffer.

4.5.4 VME-to-VME

The Tsi148 DMA controllers support VME-to-VME DMA transactions.

4.5.4.1 Example DMA VME-to-VME Transaction

In this example, there is a DMA transaction between the VMEbus and VMEbus. The following list, and Figure 26 and Figure 27, show the steps taken in the DMA transaction.

1. Program the registers in the LCSR group.

   — The DMA registers set-up the following information:

       - Source and destination buses
       - Mode of operation
       - Attributes
       - Bus width
       - Transfer throttling
       - DMA transfer count

   Transfer counts can be between 1 byte and 4 Gbytes.
2. Once these registers have been programmed writing the DGO bit in the DMA control register initiates the DMA transfer.

3. The DMA controller issues a read request to the Linkage Module.

4. After arbitration, the Linkage Module passes command information, address information, and transfer size to one of the VME Master’s two read buffer command queues.

5. The VME Master issues a read request to the VMEbus slave.
6. Once the read request is satisfied, or the programmed VMEbus block size value is satisfied, the VME Master makes a request to the Linkage Module.

   — The Block Size is programmed in the VBKS field in the DMA Control Register when VME is the source bus (see Section 8.4.76 on page 279).

7. After arbitration, the Linkage Module passes the read data to the DMA controllers data buffer. The data buffer is used to hold data that is transferred between the source and destination bus.

8. The DMA controller issues a write request to the Linkage Module.

9. After arbitration, the command, information, address information, and write data is passed to a VME Master write buffer command and data queue.
10. The VME Master initiates the write transaction to the VMEbus slave. The VME Master fills the DMA buffer to a certain point, after which the VME Master empties the DMA buffer.

### 4.5.5 Data Patterns

The Tsi148’s DMA Controller can write data patterns to either VME or PCI/X space. The data patterns can be any size transfer, and there are no restrictions on the starting address.

The is a starting data pattern is supplied by software. Software can also specify whether the pattern should be static or incrementing. The DMA Controller can be programmed to work in terms of 8-bit patterns (see Figure 28) or 32-bit patterns (see Figure 29).

**Figure 28: 8-bit Pattern Writes**

<table>
<thead>
<tr>
<th>DSAD</th>
<th>Start Pattern = 0x20</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDAD</td>
<td>Destination Address = 0x..02</td>
</tr>
<tr>
<td>DCTL</td>
<td>Transfer Count = 27</td>
</tr>
</tbody>
</table>

**DMA Control Registers**

- **Static Pattern to VME Space...**
  - Start Pattern = 0x20
  - Destination Address = 0x..02
  - Transfer Count = 27
  - 8-bit Pattern Writes
    - 8-bit Pattern Writes
      - Start Pattern = 0x20
      - Destination Address = 0x..02
      - Transfer Count = 27
  - Incrementing Pattern to VME Space...
    - Start Pattern = 0x20
    - Destination Address = 0x..02
    - Transfer Count = 27
    - 8-bit Pattern Writes
      - 8-bit Pattern Writes
        - Start Pattern = 0x20
        - Destination Address = 0x..02
        - Transfer Count = 27
  - Static Pattern to PCI Space...
    - Start Pattern = 0x20
    - Destination Address = 0x..02
    - Transfer Count = 27
    - 8-bit Pattern Writes
      - 8-bit Pattern Writes
        - Start Pattern = 0x20
        - Destination Address = 0x..02
        - Transfer Count = 27
  - Incrementing Pattern to PCI Space...
    - Start Pattern = 0x20
    - Destination Address = 0x..02
    - Transfer Count = 27
    - 8-bit Pattern Writes
      - 8-bit Pattern Writes
        - Start Pattern = 0x20
        - Destination Address = 0x..02
        - Transfer Count = 27
4.5.5.1 Data Patterns and Endianness

When writing 32-bit patterns to PCI/X space, the pattern is not Endian byte swapped. Also, when writing 332-bit patterns, a transfer count that is not an even multiple of four is rounded off of the last data pattern written to either VME or PCI/X space. The rounding off occurs on the pattern according to the address space being written to. For example, a pattern written to PCI/X space is rounded off starting from the left side (Most Significant Bit) of the pattern, while a pattern written to VME space is rounded off starting from the right (or Least Significant Bit) side of the pattern.
4.5.6 DMA Transaction Termination

Tsi148 DMA activity can be terminated through either a transfer completion, commanded stop, commanded abort, or a detected error abort.

4.5.6.1 Transfer Completion

In most cases, a Direct mode transfer or a Linked-list mode transaction finishes without intervention or error. In Direct mode operation, the end of the transfer is considered completion. In Linked-List mode operation, the end of the last transfer of a command is considered completion. When the transaction is complete, the DMA controller returns a done status to the DMA Status (DSTA) register (see Section 8.4.77 on page 284) and, when enabled, interrupts the processor.

4.5.6.2 Commanded Stop

The commanded stop termination can be used during Linked-list transactions. Software is used to set the Commanded Stop bit (PAU) in the DMA Control register (see Section 8.4.76 on page 279). This bit can be set at any time during a DMA transaction.

When the DMA controller reaches a transfer boundary (that is, ready to fetch the next descriptor), it stops all DMA activity. If there are more Linked-list commands to be performed, the DMA controller returns a paused status to the DSTA register and, optionally, interrupts the processor. If the last command has completed, then the DMA controller returns a done status to the DSTA register (see Section 8.4.77 on page 284).

Once the transaction has been stopped, the linked list transaction can be started again at any time. The DMA controller starts the transaction where it left off. The first descriptor fetch occurs from the address that was placed within the DMA Next Link Address (DNLA) register during the previously completed transfer (see Section 8.4.90 on page 305).

4.5.6.3 Commanded Abort

The commanded abort termination can occur on either Direct mode or Linked-list mode. Software is used to set the Commanded Abort bit in the DMA Control (CTL) register (see Section 8.4.76 on page 279). This bit can be set at any time during a transaction.

When the Commanded Abort bit is set, the DMA controller aborts all DMA activity. This is considered a non-recoverable termination, and it takes affect immediately after the bit has been set. If the commanded abort took affect before all commands were completed, then the DMA controller returns an abort status to the DSTA register and, optionally, interrupts the processor. If all commands completed before the commanded abort took affect, then the controller returns a done status to the DMA Status (DSTA) register (see Section 8.4.77 on page 284).
4. DMA Interface

4.5.6.4 Detected Error Abort
If any of following system errors are encountered, the DMA controller aborts all DMA activity:

- PCI/X Master received a master abort
- PCI/X Master received a target abort
- PCI/X Master exceeds the maximum retry count
- VME Master received a bus error
- VME Master received slave termination

This is considered a non-recoverable termination, and takes affect immediately after the condition has been detected. Once all DMA activity has ceased, the DMA controller returns the appropriate error status to the DSTA register and, if enabled, interrupts the processor.

4.5.7 DMA Interrupts
The DMA Controller sends an interrupt to the interrupt controller when it returns to the idle state. If the DMA interrupt in the interrupt controller is enabled, an INT_x signal line is asserted to signal the interrupt.

The DSTA register can be read at any time to obtain the operating status of the controller.

4.5.8 Transfer Throttling
The Tsi148 has the ability to throttle DMA transfers. This features is for situations where the VMEbus or PCI/X bus bandwidth that is consumed by the DMA Controller could swamp the system with DMA activity. There are several methods available to control the bandwidth consumed by the DMA controller. The PCI/X bus latency timer and the VMEbus time-on timer can be used to control the VMEbus and PCI/X bus time allocated to the Tsi148. In addition the DMA controller has a programmable block size and back-off timer.

The block size can be set from 32 to 4096 bytes. The back-off value can be set from 0 to 64 us. The block size and back-off time are independently programmable for each bus. The DMA controller requests the selected block size and when that request is satisfied, it waits for the time set by the back-off timer before requesting a new block.

Larger DMA block sizes are more efficient but increase latency. Smaller DMA block sizes reduce latency but are less efficient.
5. Resets, Clocks, and Power-up Options

Reset options include how a device or components of a device are reset, and how the device responds to a reset event. Clock characteristics include how a device’s operating frequency is set, and if required, how it should be synchronized with other devices in a system. Power-up options include device-specific capabilities that are configured upon the completion of a power-up reset sequence. These include functions such as bus mode (PCI versus PCI/X) and data width size (32-bit versus 64-bit).

This chapter discusses the following topics about Tsi148 Resets, Clocks, and Power-up Options:

- “Overview of Resets, Clocks, and Power-up Options” on page 122
- “Resets” on page 122
- “Clocks” on page 126
- “Power-up Options” on page 127
5.1 Overview of Resets, Clocks, and Power-up Options

This section describes the reset capabilities, clocking requirements, and power-up options for the Tsi148 device.

5.2 Resets

Tsi148 can be reset from both the VMEbus and the PCI/X bus. The device responds to both hardware and software reset events. Figure 30 shows the logical representation of the Tsi148 reset structure.

Figure 30: Tsi148 Reset Structure
5.2.1 **Reset Inputs and Outputs**

Tsi148 has the following reset inputs and reset outputs:

- **Reset Inputs**
  - Power-up Reset (PURSTI\_): This signal resets all of the Tsi148 logic. When it is asserted both the PCI/X and VMEbus can be reset through the Tsi148 reset outputs LRSTO\_ and SRSTO.
  
  - VMEbus System Reset In (SRSTI\_): This signal resets all of the Tsi148 logic which is sensitive to SYSRESET. Typically, the backplane SYSRESET\_ is connected to this signal through a transceiver. When SRSTI\_ is asserted the PCI/X bus can be reset through the Tsi148 reset output LRSTO\_.
  
  - PLL Reset (PLL\_RSTI\_): This signal resets the Tsi148 PLL. The PLL\_RSTI\_ pin has to be asserted until the clock and power are stable.
  
  - JTAG Test Reset (TRST\_): Provides asynchronous initialization of the TAP controller in the Tsi148. This signal must be tied to ground if JTAG is not used in the system.
  
  - Local Bus (PCI/X) Reset In (LRSTI\_): Assertion of this signal resets all Tsi148’s internal logic except the logic required for VME services and clock service (see Figure 30). This signal should be connected to the board’s local bus (PCI/X) reset.
  
  - Local System Reset (LSRSTI\_): This signal is used to reset the VMEbus from the PCI/X bus. When this signal is asserted the Tsi148 output SRSTO is asserted. This signal allows on board logic to generate a VMEbus system reset.

- **Reset Outputs**
  
  - VMEbus System Reset Out (SRSTO): This signal is used to reset the VMEbus. Typically, this signal is connected to the backplane SYSRESET\_ signal through an inverting open collector buffer. When SRSTO is asserted, the VMEbus SYSRESET\_ signal is asserted.

  SRSTO can be asserted either through hardware or software events. The hardware reset events are detailed in Figure 30. The signal can be asserted through software by setting the SRESET bit in the VMEbus Control (VCTRL) register (see Section 8.4.34 on page 205).

  The SRESET bit is self-clearing.

  The SRSTO signal is asserted for a minimum of 200ms.
Local Bus (PCI/X) Reset Out (LRSTO_): This signal resets local (PCI/X) resources.
LRSTO_ can be combined with other board sources to generate a local (PCI/X) reset
signal.

LRSTO_ can be asserted either through hardware or software events. The hardware
reset events are detailed in Figure 30. This signal can be asserted through software by
the following methods:

- Setting the LRESET bit in the VCTRL register (see Section 8.4.34 on page 205).
The LRESET bit is self clearing. When the LRESET bit is set the LRSTO_
signal remains asserted for a minimum of 15us. Because this bit only resets the
board and not the entire system, setting this bit can have side effects. For
example, if there are VMEbus transfers in progress, local resources required to
complete the transfers are reset and unavailable. This may cause aborted
VMEbus cycles, VMEbus time-outs, or a VMEbus lockup. To avoid these side
effects, the following rules must be used when setting the LRESET bit:
1. The LRESET bit must only be used in exceptional cases and not during
   normal system operation.
2. The software must set VMEbus Stop (VS) bit and wait for the VMEbus Stop
   Acknowledge bit (VSA) to be set (see Section 8.4.33 on page 201). When the
   VS bit is set, Tsi148 acquires VMEbus ownership. This prevents any other
   VMEbus masters from acquiring the VMEBus. Setting the VS bit also prevents
   Tsi148 from starting any VMEbus cycles. This ensures that the VMEbus is in an
   idle state when the LRSTO_ signal is asserted. The LRESET bit can then be set.

- Setting the Local Reset (LRST) bit in the GCTRL register (see Section 8.4.96 on
  page 310). The LRSTO_ reset remains asserted as long as the LRST bit is set.

- Setting the Local Reset Set (LRSTS) bit in the CR/CSR Bit Set (CSRBSR)
  register (see Section 8.4.102 on page 318). When the bit is set the board is held
  in reset until a 1 is written to the LRSTC bit in the CSRBCR register.
5.2.2 Reset Timing

Figure 31 shows the power-up reset timing of Tsi148. The numbers in the figure correspond to the following values:

- 1 = PLL_RST_ hold time (0ns)
  - PLL_RST_ can be released once the PCLK and power are stable
- 2 = PURST_ hold time (150us)
  - PURSTI_ must be held after negation of PLL_RSTI_ to make sure the PLL is locked to the PCLK frequency
- 3 = Assertion of SRSTO (200ms)
  - Minimum assertion of SRSTO output (as required by the American National Standard for VME64)

Figure 31: Timing for Power-up Reset
5.3 **Clocks**

Tsi148 clocks are derived from the PCI/X bus clock. The PCI/X bus clock frequency can be 33, 66, 100, or 133 MHz.

- PCLK operation below 33 MHz is not recommended.

The PCI/X clock frequency and bus mode is configured on the rising edge of LRSTI_. (see Table 5)

### Table 5: PCI Bus Configuration

<table>
<thead>
<tr>
<th>PCI Bus Signal</th>
<th>PCI Bus Mode</th>
<th>PCI Clock Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAME__</td>
<td>IRDY_</td>
<td>DEVSEL_</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The **PCI Local Bus Specification (Revision 2.2)** does not require the PCI bus configuration signals to be valid until 10 clocks before the negation of PCI reset, however Tsi148 has tighter requirements. Tsi148 expects the PCI bus configuration signals to be valid when the PCI clock starts and remain valid until the LRSTI_ signal is negated. This allows the internal PLL to lock to the PCI/X bus clock.

The configuration signals are only latched on the first rising edge of LRSTI_. If LRSTI_ is asserted at a later time, the configuration signals are not latched again. However, if both PURSTI_ and LRSTI_ are reasserted, then the configuration signal latches are opened and the configuration signals are latched on the rising edge of LRSTI_.

The configuration signals are only latched once to make sure the PLL clock remains stable through a PCI/X bus reset. This stability enables a subset of the VMEbus logic to function while the PCI/X bus is in reset, including: the VMEbus SYSCLK, VMEbus arbiter, VMEbus daisy chain signals, VMEbus General Control and Status register access, and VMEbus Control and Control and Status register accesses.
The PCI/X bus clock input provides the reference clock for the internal PLL. The PLL is used to derive the 16 MHz VMEbus SYSCLK.

If the PCI/X clock input is below the maximum frequency defined for a specific configuration, the PLL frequency is scaled accordingly. When the PLL frequency is scaled down the VMEbus timing parameters are not violated, but the VMEbus performance and timer accuracy is affected. When this situation occurs the VMEbus SYSCLK output should not be used.

5.4 Power-up Options

Tsi148 samples various VMEbus and PCI/X bus signals during reset to enable or disable certain functions.

5.4.1 PCI/X Power-up Options

The PCI/X Interface has power-up options that control how the interface is configured for use in a system.

5.4.1.1 Bus Width

The PCI/X Interface supports 32-bit or 64-bit PCI/X bus widths. The PCI/X bus width is configured during a PCI/X bus reset. If REQ64_ is high during the rising edge of LRSTI_, then the chip is configured for 32-bit PCI/X. If REQ64_ is low during the rising edge of LRSTI_, then the chip is configured for 64-bit PCI/X. When the chip is used on a 32-bit PCI/X bus, REQ64_ should be pulled high with a weak pull-up resistor.

When the Tsi148 is used on a 32-bit PCI/X bus, it drives CBE[7:4]_, AD[63:32], PAR64 and ACK64_ at all times. These signals may be left unconnected when the chip is used on a 32-bit PCI/X bus. When the chip is used on a 64-bit PCI/X bus, it must not be configured for 32-bit operation. Other PCI/X devices may drive their 64-bit extension signals and this could cause excessive currents in the output drivers.
Table 6 on page 128 shows Tsi148’s PCI/X bus width configurations.

**Table 6: PCI/X Bus Configuration**

<table>
<thead>
<tr>
<th>Function</th>
<th>Register</th>
<th>Reset</th>
<th>Sample Signal(s)</th>
<th>Sample State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI/X Bus Data Width</td>
<td>PCI/X Configuration Status Register</td>
<td>LRSTI_</td>
<td>REQ64_</td>
<td>0</td>
<td>64-bit PCI/X bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>32-bit PCI/X bus</td>
</tr>
<tr>
<td>PCI/X Mode</td>
<td></td>
<td>LRSTI_</td>
<td>M66EN, FRAME_, IRDY_, TRDY_, STOP_, DEVSEL_</td>
<td></td>
<td>Refer to Table 5 on page 126.</td>
</tr>
</tbody>
</table>

### 5.4.1.2 Frequency

The mode and frequency of the PCI/X bus is determined the first time LRSTI_ is negated. The M66EN, FRAME_, IRDY_, TRDY_, STOP_, and DEVSEL_ signals are sampled on the rising edge of LRSTI_ and the PCI/X bus is configured (as defined in the *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b)*). For more information, refer to Section 5.3 on page 126.
5.4.2 VMEbus Power-up Options

The Tsi148 VMEbus Interface supports a number of power-up options. Power-up options are latched during the assertion of PURSTI_. During power-up reset Tsi148 negates the External Transceiver Enable (DBOE_) signal, which puts the VD[31:0], VA[31:1], LWORD transceivers into a high impedance state. External pull-ups or pull-downs placed between Tsi148 and the external transceivers bring these power-up option signals to their proper state while DBOE_ is negated.

Table 7 shows the data signal and the functionality it enables through power-up configuration.

<table>
<thead>
<tr>
<th>Description</th>
<th>Power-up Option</th>
<th>VMEbus Data Signal</th>
<th>Control Register</th>
<th>Detailed Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFAILEN Control Bit Reset Value</td>
<td>SFAILEN_RV</td>
<td>VD[0]</td>
<td>Control and Status Register&lt;br&gt;• SFAILEN bit</td>
<td>Section 5.4.2.4 on page 132</td>
</tr>
<tr>
<td>SFAILAI Control Bit Auto Clear</td>
<td>SFAILAI_AC</td>
<td>VD[1]</td>
<td>VMEbus Control Register&lt;br&gt;• SFAILAI bit</td>
<td>Section 5.4.2.2 on page 131</td>
</tr>
<tr>
<td>Auto Slot ID Enable</td>
<td>ASIDEN</td>
<td>VD[2]</td>
<td>None - power-up option only</td>
<td>Table 8 and Section 5.4.2.2 on page 131</td>
</tr>
<tr>
<td>Geographical Slot ID Enable</td>
<td>GSIDEN</td>
<td>VD[3]</td>
<td>None - power-up option only</td>
<td>Table 8 and Section 5.4.2.3 on page 132</td>
</tr>
</tbody>
</table>
5.4.2.1 ASIDEN and GSIDEN Power-up Options Assigning the CR/CSR Base Address

The data signals and the functionality they enable through power-up configuration are described individually, however the functions are not independent. The ASIDEN and GSIDEN functions define the method for assigning the CR/CSR base address. The interaction of these two functions is shown in Table 8.

Table 8: ASIDEN and GSIDEN Definition

<table>
<thead>
<tr>
<th>ASIDEN</th>
<th>GSIDEN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>CR/CSR Disabled</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Geographical Address</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Auto Slot ID</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Geographical Address defaults to Auto Slot ID if GA[4:0] pins are all high</td>
</tr>
</tbody>
</table>

Table 9 defines all combinations of the four VMEbus data bits.

Table 9: CR/CSR Base Address Configuration

<table>
<thead>
<tr>
<th>VD[3:0]</th>
<th>GA (All High)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00X0</td>
<td>X</td>
<td>CR/CSR disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRAT register, EN cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VCTRL register, SFAILAI cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCTRL register, SFAILEN cleared by S reset</td>
</tr>
<tr>
<td>00X1</td>
<td>X</td>
<td>CR/CSR disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRAT.EN cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VCTRL.SFAILAI cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCTRL.SFAILEN cleared by S reset</td>
</tr>
<tr>
<td>0100</td>
<td>X</td>
<td>Auto Slot ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRAT register, EN cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VCTRL register, SFAILAI set by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCTRL register, SFAILEN cleared by S reset</td>
</tr>
<tr>
<td>01X1</td>
<td>X</td>
<td>Illegal Configuration</td>
</tr>
<tr>
<td>0110</td>
<td>X</td>
<td>Auto Slot ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRAT register, EN cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VCTRL register, SFAILAI set by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCTRL register, SFAILEN cleared by S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cleared 1 ms after S reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCTRL register, SFAILEN cleared by S reset</td>
</tr>
</tbody>
</table>
5.4.2.2 Auto Slot ID Operation

Tsi148 has Auto Slot ID functionality which is described in the American National Standard for VME64.

When the Auto Slot ID functionality is enabled in a system, after system reset each board in the system generates an interrupt on level IRQ2_. A level two interrupt handler module, called the Monarch, performs interrupt acknowledge cycles in response to each interrupt request. Before the Monarch can respond with its interrupt acknowledge cycle all boards in the system must have SYSFAIL_ negated. Once SYSFAIL_ is negated, the Monarch performs the interrupt service routine. Each VMEbus slave responds with an initial CR/CSR address space of zero. The Monarch then configures the CR/CSR base address of each board through its CR/CSR base address register.

<table>
<thead>
<tr>
<th>VD[3:0]</th>
<th>GA (All High)</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10X0    | X             | Geographical Addressing  
|         |               | CRAT register, EN set by S reset  
|         |               | VCTRL register, SFAILAI cleared by S reset  
|         |               | GCTRL register, SFAILEN cleared by S reset |
| 10X1    | X             | Geographical Addressing  
|         |               | CRAT register, EN set by S reset  
|         |               | VCTRL register, SFAILAI cleared by S reset  
|         |               | GCTRL register, SFAILEN set by S reset |
| 11X0    | 0             | Geographical Addressing  
|         |               | CRAT register, EN set by S reset  
|         |               | VCTRL register, SFAILAI cleared by S reset  
|         |               | GCTRL register, SFAILEN cleared by S reset |
| 11X1    | X             | Illegal Configuration |
| 1100    | 1             | Default to Auto Slot ID  
|         |               | CRAT register, EN cleared by S reset  
|         |               | VCTRL register, SFAILAI set by S reset  
|         |               | GCTRL register, SFAILEN cleared by S reset |
| 1110    | 1             | Default to Auto Slot ID  
|         |               | CRAT register, EN cleared by S reset  
|         |               | VCTRL register, SFAILAI set by S reset, cleared 1 ms after S reset  
|         |               | GCTRL register, SFAILEN cleared by S reset |
Auto Slot ID Enable
The Auto Slot ID Enable (ASIDEN) feature is controlled through a power-up option. The ASIDEN feature allows the CR/CSR base address to be configured using the Auto Slot ID protocol. ASIDEN can be enabled through a power-up option (shown in Table 7 on page 129). The power-up option is sampled at the rising edge of the PURSTI signal.

System Failure Auto Slot ID (SFAILAI) Configuration
The System Failure Auto Slot ID (SFAILAI) bit is used when the Auto Slot ID protocol is enabled in the system to assign the CR/CSR base address. The initial value of the SFAILAI bit can be configured at power-up reset through the SFAILAI AC power-up option or a value can be programmed by software in the SFAILAI bit in the VMEbus Control register (VCTRL) (see Section 8.4.34 on page 205).

When Auto Slot ID is used to assign the CR/CSR base address, the SFAILAI bit is set by the assertion of the SRSTI signal. The SFAILAI bit must be cleared in order for Tsi148's System Fail Output (SFAILO) signal to be negated. SFAILO is automatically negated if the SFAILAI AC power-up option is selected, otherwise SFAILO is negated when software clears the SFAILAI bit in the VCTRL register.

This feature can be enabled through the SFAILAI AC power-up option as shown in Table 7 on page 129. The power-up option is sampled at the rising edge of the PURSTI signal.

Geographic Slot ID Enable
The Geographic Slot ID Enable function initializes the CR/CSR base address register using the VMEbus GA signals. The Geographic Slot ID Enable feature allows a board to come out of reset with the CR/CSR registers visible from the VMEbus and the base address of the CR/CSR is determined by the VMEbus GA signals.

The initial value of the CR/CSR Enable bit in the CR/CSR Attribute (CRAT) register and CBAR bits in the CR/CSR Base Address (CBAR) register can be configured at power-up reset using the Geographic Slot ID Enable function (see Table 7 on page 129). If the VD[3] signal is zero at the rising edge of the PURSTI signal, the CR/CSR enable bit and CBAR bits are cleared. If the VD[3] signal is one at the rising edge of the PURSTI signal, the CR/CSR enable bit is set and the CBAR bits 7 to 3 are set to the inverted value of the VMEbus geographic address signals. When the SRSTI signal is asserted, the CR/CSR EN bit and the CBAR bits are loaded with the power-up option reset values.

System Fail Enable (SFAILEN) Configuration
The Tsi148 System Failure Enable (SFAILEN) bit controls the assertion of the Tsi148 System Fail Output (SFAIL0) signal. The initial value of the SFAILEN bit can be configured at power-up reset through the SFAILEN RV power-up option. Additionally, a value can be programmed by software in the Control and Status register.
The Board Fail (BDFAIL_) signal, along with the SFAILEN bit, determine if Tsi148 generates the SFAILO signal. The Board Fail signal (BDFAIL_) can be generated either through software, by writing to the Board Fail bit (BRDFL) in the VMEbus Status register (see Section 8.4.35 on page 209), or by external logic on the board.

The SFAILO signal is controlled through the following registers:

- **GCSR Control and Status register**
  - The SFAILO signal can be enabled or disabled through the SFAILEN bit

- **CR/CSR Bit Clear register**
  - The SFAILO signal can be disabled through the SFAILC bit

- **CR/CSR Bit Set register**
  - The SFAILO signal can be enabled through the SFAILS bit

This feature can be enabled through the SFAILEN_RV power-up option as shown in Table 7 on page 129. The power-up option is sampled at the rising edge of the PURSTI_ signal.

The SFAILEN_RV power-up option must be cleared when using the Auto Slot ID (ASIDEN) power-up option to configure the CR/CSR base address register. Software must not set the SFAILEN bit until the Auto Slot ID process is complete.

### 5.4.3 System Controller (SCON)

Tsi148 has VMEbus System Controller (SCON) functionality. The SCONEN_ and SCONDIS_ signals are used to control the SCON function. If the SCONEN_ signal is low and the SCONDIS_ signal is high at the rising edge of PURSTI_, the SCON function is enabled. If the SCONEN_ signal is high and the SCONDIS_ signal is low at the rising edge of PURSTI_, the SCON function is disabled. If the SCONEN_ signal and the SCONDIS_ signal are both high at the rising edge of PURSTI_, the Auto System Controller feature is used.

The American National Standard for VME64 Extensions defines the Auto System Controller feature.
The Auto System Controller feature uses the BG3IN_ signal to enable a board to determine if it is in VMEbus slot 1. If the board is in VMEbus slot 1, the BG3IN_ signal is low and the SCON function is enabled. If the board is not in VMEbus slot 1, the BG3IN_ signal is high and the SCON function is disabled.

<table>
<thead>
<tr>
<th>Function</th>
<th>Register</th>
<th>Reset</th>
<th>Sample Signal(s)</th>
<th>Sample State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCON</td>
<td>VMEbus Status Register</td>
<td>PURSTI_</td>
<td>SCONEN_</td>
<td>0</td>
<td>SCON Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONDIS_</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONEN_</td>
<td>1</td>
<td>SCON Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONDIS_</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VBG3IN_</td>
<td>1</td>
<td>Auto System Controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONEN_</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONDIS_</td>
<td>1</td>
<td>SCON Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VBG3IN_</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONEN_</td>
<td>1</td>
<td>Auto System Controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCONDIS_</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VBG3IN_</td>
<td>1</td>
<td>SCON DISABLED</td>
</tr>
</tbody>
</table>
6. Interrupt Controller

An interrupt is a process by which a program is informed that an event has occurred in the system (for example, an interrupt signal is asserted to indicate an error). When a program receives an interrupt, it temporarily suspends normal processing and diverts the execution of instructions to a sub-routine handled by an interrupt controller. The controller communicates with the host processor and the device that initiated the interrupt to determine how to handle the interrupt.

Interrupt events originate from a variety of sources; however, they can be classified as one of two types: hardware and software interrupts. Interrupts generated by devices (for example, a printer) indicate an event has occurred and are called hardware interrupts. Interrupt events generated by software programs are called software interrupts.

This chapter discusses the following topics about the Tsi148 interrupt features:

- “Overview of the Interrupt Controller” on page 136
- “VMEbus Interrupter” on page 136
- “Local Interrupter” on page 136
- “VMEbus Interrupt Handler” on page 137
6.1 Overview of the Interrupt Controller

Tsi148 can be programmed to act as interrupter and an interrupt handler in a VME system. As an interrupter, Tsi148 is capable of asserting interrupts on IRQ[7:1].

As an interrupt handler, Tsi148 has seven VMEbus Interrupt Acknowledge registers which, when read, generate an IACK cycle on the VMEbus (see Section 8.4.69 on page 258).

6.2 VMEbus Interrupter

Tsi148 has a VMEbus interrupter which enables software to generate VMEbus interrupts. The interrupter operates in Release-on-Acknowledge (ROAK) mode. An 8-bit status/ID is provided upon receiving the IACK cycle.

The following steps illustrate how a VMEbus interrupt is generated:

1. The STATUS/ID and IRQL fields must be set in the VMEbus Interrupt Control (VICR) register. The IRQL field defines the level of VMEbus interrupt output signals (IRQ[7:1]O). A VMEbus interrupt is generated when the IRQL field is written. The interrupter asserts the requested interrupt onto the VMEbus and sets the VMEbus IRQ Status (IRQS) bit in the VMEbus Interrupt Control (VICR) register (see Section 8.4.69 on page 258).

   Only one interrupt at a time can be generated.

2. Once the interrupt is acknowledged the IRQS bit is cleared and the interrupt can be sent to the local bus interrupter (if enabled).

6.3 Local Interrupter

Tsi148’s local interrupter provides a mechanism to control the interrupts generated by internal and external sources. The local interrupter receives interrupts from internal and external sources and routes them to one of four interrupt output lines (INTA_, INTB_, INTC_, INTD_).

There are the following internal and external sources of interrupts:

- VMEbus IRQ[7:1]I_
- ACFAILI_
- SFAILI_
- VMEbus Error
6. Interrupt Controller

- DMA controllers
- VMEbus Interrupter Acknowledged
- VMEbus Edge (Broadcast interrupt, Clock and 64-bit Counter)
- PCI Error
- Mailbox[3:0]
- Location Monitor [3:0]

Each interrupt source has an enable bit, status bit, interrupt out enable bit, and two map bits. The edge sensitive interrupts also have a clear bit. These bits can be programmed in the Tsi148 Interrupt registers (see Section 8.4.69 on page 258).

The Tsi148 expects the interrupt handling intelligence to exist on the local (PCI/X) bus. The Tsi148 does not have the ability to route local interrupt outputs (INTA_, INTB_, INTC_, INTD_) to VMEbus interrupt outputs (IRQ[7:1])

6.4 VMEbus Interrupt Handler

Tsi148 has seven VMEbus Interrupt Acknowledge registers which generate an IACK cycle on the VMEbus when they are read. There is one IACK register for each of the VMEbus IRQ[7:1] signals. These features can be programmed in the VMEbus Interrupt Control registers (see Section 8.4.69 on page 258).

The interrupt handler has the following features:

- Supports 8, 16, and 32-bit IACK cycles
  - A word read of the IACK registers causes a 32-bit IACK cycle on the VMEbus
  - A half-word read causes a 16-bit IACK cycle on the VMEbus
  - A byte read causes an 8-bit IACK cycle on the VMEbus
- Once the IACK cycle is generated the interrupter supplies its status/ID.

---

80A3020_MA002_01
7. JTAG Module

The Joint Test Action Group (JTAG) created the boundary-scan testing standard (documented in the *IEEE 1149.1 Standard*) for testing printed circuit boards (PCBs). The boundary-scan approach involves designing boundary-scan circuitry into the integrated circuit. PCBs populated with 1149.1 compliant devices can be tested for connectivity, correct device orientation, correct device location, and device identification.

All the pins on compliant devices can be controlled and observed using (typically) five pins that are routed to the board edge connector. Board designers can develop a standard test for all 1149.1 compliant devices regardless of device manufacturer, package type, technology, or device speed.

This chapter discusses the following topics about Tsi148’s JTAG features:

- “Overview of JTAG” on page 140
- “Instructions” on page 140
7. JTAG Module

### 7.1 Overview of JTAG

Tsi148 has a dedicated user-accessible JTAG (Joint Test Action Group) module that is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture*.

The JTAG logic includes a Test Access Port (TAP) consisting of five dedicated signals (TCK, TRST_, TMS, TDI, TDO), a TAP controller, instruction register, bypass register, other test data registers (for example, device identity register, etc.), and boundary-scan register (see Figure 32).

#### Figure 32: JTAG Functional Diagram

![JTAG Functional Diagram](image)

### 7.2 Instructions

Tsi148’s IEEE 1149.1 implementation includes the following instructions:

- **EXTEST**: This instruction drives the data loaded into the boundary scan register through the output pin to drive another chip with the value loaded in the boundary scan cell by the SAMPLE/PRELOAD instruction. At the same time, this instruction also captures the data at the inputs. This process is useful for board interconnect testing.
- SAMPLE/PRELOAD: This instruction loads the boundary scan chain with proper values before driving it to another chip using the EXTEST or INTEST instruction.

- IDCODE: This instruction configures a 32-bit identification register between the TDI and TDO pins. The instruction selects the ID register and shifts out the identity of the manufacturer, the version, and device identification number. The value of the identification register for revision 1 is 0x{xxxxx}.

- BYPASS: This instruction places a one-bit register between the TDI and TDO pins. This provides a short path through the device for shifting data from one chip to another without going through the boundary scan chain.

- HIGHZ: This instruction is the same as BYPASS except that all the bidirect and 3-state outputs are 3-stated when this instruction is active. The boundary scan cell cannot be updated with a new value during this instruction.
7. JTAG Module
8. Registers

This appendix describes the Tsi148’s registers. The following topics are discussed:

- “Overview of Registers” on page 144
- “Register Groupings” on page 144
- “Register Endian Mapping” on page 147
- “Register Map” on page 149
8. Registers

8.1 Overview of Registers

This chapter provides a detailed description of the Tsi148’s internal registers. These registers are separated into four groups: the PCI/X Configuration Space registers (PCFS), the Local Control and Status Registers (LCSR), the VMEbus Global Control and Status Registers (GCSR) and the VMEbus Configuration ROM / Control and Status Registers (CR/CSR).

Registers can be accessed by the Tsi148 PCI/X Target or the VME Slave through the internal Linkage Module.

8.2 Register Groupings

Tsi148 register space is separated into different groups within Tsi148. Figure 33 shows the complete register map and individual groups.

Figure 33: Combined Register Group (CRG)

| 4 Kbyte CRG | 1024 bytes | CSR |
| 1504 bytes | Reserved |
| 32 bytes | GCSR |
| 1280 bytes | LCSR |
| 256 bytes | PCFS |

8.2.1 Combined Register Group (CRG)

The CRG requires 4 Kbytes of address space. The address space can be mapped into PCI/X address space using the standard PCI/X BAR (located at offsets 0x10h and 0x14h). All CRG accesses through the PCI/X BAR pass through the PCI/X Target Interface.
The CRG can also be mapped into A16, A24, A32 and A64 VME address space through the CRG image (located at offsets 0x40Ch – 0x414h). The CRG can be accessed using D8, D16 and D32 SCT transactions. All accesses pass through the VME Slave Interface.

Alternatively, the CRG can be accessed as part of the 512 Kbyte CR/CSR area defined in the American National Standard for VME64 by using the special A24 CR/CSR AM code.

8.2.2 PCI/X Configuration Space Registers (PCFS)

This register area is the standard PCI/X configuration space and is accessible from the PCI/X bus using PCI/X configuration cycles.

The PCFS area includes a standard 64-bit Base Address Register (see MBARL and MBARU registers in Section 8.4.2 on page 150) which enables the CRG to be mapped into PCI/X memory space.

The PCFS can also be accessed from the VMEbus as part of the CRG group.

8.2.3 Local Control and Status Registers (LCSR)

The LCSR register group contains the inbound and outbound map decoder registers, DMA, interrupt control registers, and other miscellaneous registers. It can accessed from either the PCI/X bus or VMEbus as part of the CRG.

8.2.4 Global Control and Status Registers (GCSR)

The GCSR register group contains control bits, semaphore, and mailbox registers which allow information to be passed between processors on other VMEbus boards and the local processor.

It can accessed from either the PCI/X bus or VMEbus as part of the CRG. Alternatively, the GCSR group can be independently accessed from the VMEbus by using the GCSR image (located at offsets 0x418h – 0x420h).

The GCSR can be mapped to the VMEbus A16, A24, A32 or A64 address spaces and accepts D8, D16 and D32 SCT transactions.

8.2.5 Control and Status Registers (CSR)

The CSR register group is a sub-set of the CR/CSR section of the CR/CSR registers defined in the American National Standard for VME64 Extensions. Tsi148 implementation of these standard registers include: the CR/CSR Bit Clear, CR/CSR Bit Set, and CR/CSR Base Address Registers.
8. Registers

8.2.6 CR/CSR Register Access

The 512 Kbyte CR/CSR space, shown in Figure 34, can be accessed from the VMEbus using the special A24 CR/CSR AM code.

The Base Address is defined by either Geographical Address Implementation or Auto Slot ID. Tsi148’s VME Slave can be configured at power-up to use one of the two methods (see Section 5.4 on page 127). When an access is initiated on the VMEbus using the A24 CR/CSR AM code, the Tsi148 initiates an access on the PCI/X bus when the enable bit in the CR/CSR Attribute Register is set (located at offset 0x420). The address generated on the PCI/X bus is determined by the values in the CR/CSR Offset registers (located at offsets 0x418 and 0x41C). These values are added to the internal VMEbus address to create the PCI/X bus address.

The address space is separated into the following areas:

- The upper 4 Kbytes defines the Tsi148 CRG
- The remaining 508 Kbytes maps to the PCI/X bus.
  
  — When an access is initiated on the VMEbus using A24 CR/CSR AM code, Tsi148 initiates an access on the PCI/X bus when the CR/CSR offset register is enabled.

Figure 34: CR/CSR Address Space
8.3 Register Endian Mapping

The VMEbus uses Big-Endian byte ordering and the PCI/X bus uses Little-Endian byte ordering. The byte ordering differences are accommodated by swapping the data in the PCI/X Master and PCI/X Target before it is passed to the Linkage Module. Data transferred between the PCI/X bus and the Linkage Module is swapped as shown in Figure 35. This method of handling the endian problem is called address invariance. If data is accessed using byte operations, little endian and big endian processors view the same data. If data is accessed using 32-bit accesses, little endian and big endian processors see different views of the same data.

Figure 35: Big to Little Endian Data Swap

When viewed from the VMEbus, the LCSR, GCSR and CR/CSR registers appear as presented in the programming section. When viewed from the VMEbus, the PCFS registers appear swapped.
When viewed from the PCI/X bus, the LCSR, GCSR and CR/CSR registers appear swapped. When viewed from the PCI/X bus, the PCFS registers appear as presented in the programming section.

Table 11 summarizes the register views. This table assumes the processor is operating in big endian mode and that the bridge between the processor bus and PCI/X bus swaps the data. This table assumes the 32-bit value ABCD is stored in a register and that the data is accessed using a 32-bit read.

### Table 11: Endian Register Views

<table>
<thead>
<tr>
<th>Register Group</th>
<th>Value in Register</th>
<th>Value on PCI/X bus</th>
<th>Value on Processor bus</th>
<th>Value on VMEbus</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFS</td>
<td>ABCD</td>
<td>ABCD</td>
<td>DCBA</td>
<td>DCBA</td>
</tr>
<tr>
<td>LCSR</td>
<td>ABCD</td>
<td>DCBA</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
<tr>
<td>GCSR</td>
<td>ABCD</td>
<td>DCBA</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
<tr>
<td>CR/CSR</td>
<td>ABCD</td>
<td>DCBA</td>
<td>ABCD</td>
<td>ABCD</td>
</tr>
</tbody>
</table>

Data transferred between the VMEbus and the PCI/X bus is swapped. When a processor operating in big endian mode transfers data from the VMEbus, the data is swapped in the VME bridge and swapped again in the host bridge. The two swaps effectively cancel each other and processor sees the VMEbus data in the correct order. When a processor operating in little endian mode is used, the VMEbus data appears swapped.
8.4 Register Map

The register map shows all the Tsi148 register groupings in the Combined Register Group (CRG). The CRG requires 4 Kbytes of address space. The address space can be mapped into PCI/X address space or VMEbus address space. Refer to Section 8.2.1 on page 144 for more information on the CRG and all the group that comprise the Tsi148 registers.

8.4.1 Conventions

The following conventions are used to describe the operation of a register bit and are found in the “Type” column of the register description table:

- R: Read Only field
- R/W: Read/Write field.
- S: Writing a 1 to this field sets this field.
- C: Writing a 1 to this field clears an associated field.
- R/S: Writing a 1 to this field sets an associated field. Reading this field returns the current value of the associated field.
- R/C: Writing a 1 to this field clears an associated field. Reading this field returns the current value of the associated field.

The following conventions are used to describe the effect of the reset signals on a register bit and are found in the “Reset By” heading of the register description table:

- L: The field is affected by PCI/X local bus reset.
- S: The field is affected by VMEbus system reset.
- P: The field is affected by power up reset.
- x: The reset value depends on configuration options.

Bits that are reset by multiple signals show the signals with a slash (/) separating them. For example, if a bit is reset by all the reset signals, the register table shows the following value: L/S/P/X.
8.4.2 **PCFS Register Group Overview**

This register area is the standard PCI/X configuration space and is accessible from the PCI/X bus using PCI/X configuration cycles. Refer to [Section 8.2.2 on page 145](#) for more information on the PCFS registers.

**Table 12: PCFS Register Group**

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI/X Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header</td>
<td>Device ID (DEVI)</td>
<td>0x00/0x000</td>
</tr>
<tr>
<td></td>
<td>Vendor ID (VENI)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATUS (STAT)</td>
<td>0x04/0x004</td>
</tr>
<tr>
<td></td>
<td>Command (CMMD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Class Code (CLAS)</td>
<td>0x08/0x008</td>
</tr>
<tr>
<td>Reserved</td>
<td>Header Type (HEAD)</td>
<td>0x0C/0x00C</td>
</tr>
<tr>
<td></td>
<td>Master Latency Timer (MLAT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cache Line Size (CLSZ)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memory Base Address Lower (MBARL)</td>
<td>0x10/0x010</td>
</tr>
<tr>
<td></td>
<td>Memory Base Address Upper (MBARU)</td>
<td>0x14/0x014</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>0x18/0x018</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>0x1C/0x01C</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>0x20/0x020</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>0x24/0x024</td>
</tr>
</tbody>
</table>
### Table 12: PCFS Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31 24 23 16 15  8  7 0</td>
<td>PCFS/CRG</td>
</tr>
<tr>
<td>PCI/X Configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header</td>
<td>Reserved</td>
<td>0x28/0x028</td>
</tr>
<tr>
<td></td>
<td>Subsystem ID (SUBI)</td>
<td>0x2C/0x02C</td>
</tr>
<tr>
<td></td>
<td>Subsystem Vendor ID (SUBV)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x30/0x030</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x34/0x034</td>
</tr>
<tr>
<td></td>
<td>Capabilities Pointer (CAPP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x38/0x038</td>
</tr>
<tr>
<td></td>
<td>Maximum Latency (MXLA)</td>
<td>0x3C/0x03C</td>
</tr>
<tr>
<td></td>
<td>Minimum Grant (MNGN)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupt Pin (INTP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupt Line (INTL)</td>
<td></td>
</tr>
<tr>
<td>PCI-X Capabilities</td>
<td>PCI-X Capabilities (PCIXCAP)</td>
<td>0x40/0x040</td>
</tr>
<tr>
<td></td>
<td>PCI-X Status (PCIXSTAT)</td>
<td>0x44/0x044</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>0x48/0x048</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0xFF/0x0FF</td>
</tr>
</tbody>
</table>
8.4.3 **LCSR Register Group Overview**

The LCSR register group contains the inbound and outbound slave image registers, DMA, interrupt control registers, and other miscellaneous registers. It can accessed from either the PCI/X bus or VMEbus as part of the CRG.

**Table 13: LCSR Register Group**

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Functions</td>
<td>Outbound Translation 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Starting Address Upper 0 (OTSAU0)</td>
<td>0x100</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Starting Address Lower 0 (OTSAL0)</td>
<td>0x104</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Ending Address Upper 0 (OTEAU0)</td>
<td>0x108</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Ending Address Lower 0 (OTEAL0)</td>
<td>0x10C</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Offset Upper 0 (OTOFU0)</td>
<td>0x110</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Offset Lower 0 (OTOFLO)</td>
<td>0x114</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation 2eSST Broadcast Select 0 (OTBS0)</td>
<td>0x118</td>
</tr>
<tr>
<td></td>
<td>Outbound Translation Attribute 0 (OTAT0)</td>
<td>0x11C</td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outbound Translation 1</td>
<td>OTSAU1</td>
<td>0x120</td>
</tr>
<tr>
<td></td>
<td>OTSAL1</td>
<td>0x124</td>
</tr>
<tr>
<td></td>
<td>OTEAU1</td>
<td>0x128</td>
</tr>
<tr>
<td></td>
<td>OTEAL1</td>
<td>0x12C</td>
</tr>
<tr>
<td></td>
<td>OTOFU1</td>
<td>0x130</td>
</tr>
<tr>
<td></td>
<td>OTOFL1</td>
<td>0x134</td>
</tr>
<tr>
<td></td>
<td>OTBS1</td>
<td>0x138</td>
</tr>
<tr>
<td></td>
<td>OTAT1</td>
<td>0x13C</td>
</tr>
<tr>
<td>Outbound Translation 2</td>
<td>OTSAU2</td>
<td>0x140</td>
</tr>
<tr>
<td></td>
<td>OTSAL2</td>
<td>0x144</td>
</tr>
<tr>
<td></td>
<td>OTEAU2</td>
<td>0x148</td>
</tr>
<tr>
<td></td>
<td>OTEAL2</td>
<td>0x14C</td>
</tr>
<tr>
<td></td>
<td>OTOFU2</td>
<td>0x150</td>
</tr>
<tr>
<td></td>
<td>OTOFL2</td>
<td>0x154</td>
</tr>
<tr>
<td></td>
<td>OTBS2</td>
<td>0x158</td>
</tr>
<tr>
<td></td>
<td>OTAT2</td>
<td>0x15C</td>
</tr>
<tr>
<td>Outbound Translation 3</td>
<td>OTSAU3</td>
<td>0x160</td>
</tr>
<tr>
<td></td>
<td>OTSAL3</td>
<td>0x164</td>
</tr>
<tr>
<td></td>
<td>OTEAU3</td>
<td>0x168</td>
</tr>
<tr>
<td></td>
<td>OTEAL3</td>
<td>0x16C</td>
</tr>
<tr>
<td></td>
<td>OTOFU3</td>
<td>0x170</td>
</tr>
<tr>
<td></td>
<td>OTOFL3</td>
<td>0x174</td>
</tr>
<tr>
<td></td>
<td>OTBS3</td>
<td>0x178</td>
</tr>
<tr>
<td></td>
<td>OTAT3</td>
<td>0x17C</td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Functions</td>
<td>OTSAU4</td>
<td>0x180</td>
</tr>
<tr>
<td>Translation 4</td>
<td>OTSAL4</td>
<td>0x184</td>
</tr>
<tr>
<td></td>
<td>OTEAU4</td>
<td>0x188</td>
</tr>
<tr>
<td></td>
<td>OTEAL4</td>
<td>0x18C</td>
</tr>
<tr>
<td></td>
<td>OTOFU4</td>
<td>0x190</td>
</tr>
<tr>
<td></td>
<td>OTOFL4</td>
<td>0x194</td>
</tr>
<tr>
<td></td>
<td>OTBS4</td>
<td>0x198</td>
</tr>
<tr>
<td></td>
<td>OTAT4</td>
<td>0x19C</td>
</tr>
<tr>
<td>Outbound Translation 5</td>
<td>OTSAU5</td>
<td>0x1A0</td>
</tr>
<tr>
<td></td>
<td>OTSAL5</td>
<td>0x1A4</td>
</tr>
<tr>
<td></td>
<td>OTEAU5</td>
<td>0x1A8</td>
</tr>
<tr>
<td></td>
<td>OTEAL5</td>
<td>0x1AC</td>
</tr>
<tr>
<td></td>
<td>OTOFU5</td>
<td>0x1B0</td>
</tr>
<tr>
<td></td>
<td>OTOFL5</td>
<td>0x1B4</td>
</tr>
<tr>
<td></td>
<td>OTBS5</td>
<td>0x1B8</td>
</tr>
<tr>
<td></td>
<td>OTAT5</td>
<td>0x1BC</td>
</tr>
<tr>
<td>Outbound Translation 6</td>
<td>OTSAU6</td>
<td>0x1C0</td>
</tr>
<tr>
<td></td>
<td>OTSAL6</td>
<td>0x1C4</td>
</tr>
<tr>
<td></td>
<td>OTEAU6</td>
<td>0x1C8</td>
</tr>
<tr>
<td></td>
<td>OTEAL6</td>
<td>0x1CC</td>
</tr>
<tr>
<td></td>
<td>OTOFU6</td>
<td>0x1D0</td>
</tr>
<tr>
<td></td>
<td>OTOFL6</td>
<td>0x1D4</td>
</tr>
<tr>
<td></td>
<td>OTBS6</td>
<td>0x1D8</td>
</tr>
<tr>
<td></td>
<td>OTAT6</td>
<td>0x1DC</td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outbound Functions</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation 7</td>
<td>OTSAU7</td>
<td>0x1E0</td>
</tr>
<tr>
<td></td>
<td>OTSAL7</td>
<td>0x1E4</td>
</tr>
<tr>
<td></td>
<td>OTEAU7</td>
<td>0x1E8</td>
</tr>
<tr>
<td></td>
<td>OTEAL7</td>
<td>0x1EC</td>
</tr>
<tr>
<td></td>
<td>OTOFU7</td>
<td>0x1F0</td>
</tr>
<tr>
<td></td>
<td>OTOFL7</td>
<td>0x1F4</td>
</tr>
<tr>
<td></td>
<td>OTBS7</td>
<td>0x1F8</td>
</tr>
<tr>
<td></td>
<td>OTAT7</td>
<td>0x1FC</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td><strong>VMEbus Interrupt</strong></td>
<td>VMEbus IACK 1 (VIACK1)</td>
<td>0x204</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>VIACK2</td>
<td>0x208</td>
</tr>
<tr>
<td></td>
<td>VIACK3</td>
<td>0x20C</td>
</tr>
<tr>
<td></td>
<td>VIACK4</td>
<td>0x210</td>
</tr>
<tr>
<td></td>
<td>VIACK5</td>
<td>0x214</td>
</tr>
<tr>
<td></td>
<td>VIACK6</td>
<td>0x218</td>
</tr>
<tr>
<td></td>
<td>VIACK7</td>
<td>0x21C</td>
</tr>
<tr>
<td></td>
<td>VMEbus RMW Address Upper (RMWAU)</td>
<td>0x220</td>
</tr>
<tr>
<td></td>
<td>VMEbus RMW Address Lower (RMWAL)</td>
<td>0x224</td>
</tr>
<tr>
<td></td>
<td>VMEbus RMW Enable (RMWEN)</td>
<td>0x228</td>
</tr>
<tr>
<td></td>
<td>VMEbus RMW Compare (RMWC)</td>
<td>0x22C</td>
</tr>
<tr>
<td></td>
<td>VMEbus RMW Swap (RMWS)</td>
<td>0x230</td>
</tr>
<tr>
<td><strong>VMEbus Control</strong></td>
<td>VME Master Control (VMCTRL)</td>
<td>0x234</td>
</tr>
<tr>
<td></td>
<td>VMEbus Control (VCTRL)</td>
<td>0x238</td>
</tr>
<tr>
<td></td>
<td>VMEbus Status (VSTAT)</td>
<td>0x23C</td>
</tr>
<tr>
<td><strong>PCI/X Status</strong></td>
<td>PCI/X Control / Status (PCSR)</td>
<td>0x240</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>VME Filters</td>
<td>VMEbus Filter (VMEFL)</td>
<td>0x250</td>
</tr>
<tr>
<td>VME Exception</td>
<td>VMEbus Exception Address Upper (VEAU)</td>
<td>0x260</td>
</tr>
<tr>
<td>VME Exception</td>
<td>VMEbus Exception Address Lower (VEAL)</td>
<td>0x264</td>
</tr>
<tr>
<td>VME Exception</td>
<td>VMEbus Exception Attributes (VEAT)</td>
<td>0x268</td>
</tr>
<tr>
<td>PCI/X Error</td>
<td>Error Diagnostic PCI Address Upper (EDPAU)</td>
<td>0x270</td>
</tr>
<tr>
<td>PCI/X Error</td>
<td>Error Diagnostic PCI Address Lower (EDPAL)</td>
<td>0x274</td>
</tr>
<tr>
<td>PCI/X Error</td>
<td>Error Diagnostic PCI-X Attribute (EDPXA)</td>
<td>0x278</td>
</tr>
<tr>
<td>PCI/X Error</td>
<td>Error Diagnostic PCI-X Split Completion Message (EDPXS)</td>
<td>0x27c</td>
</tr>
<tr>
<td>PCI/X Error</td>
<td>Error Diagnostic PCI Attributes (EDPAT)</td>
<td>0x280</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Starting Address Upper 0 (ITSAU0)</td>
<td>0x300</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Starting Address Lower 0 (ITSAL0)</td>
<td>0x304</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Ending Address Upper 0 (ITEAU0)</td>
<td>0x308</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Ending Address Lower (ITEAL0)</td>
<td>0x30C</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Offset Upper 0 (ITOFU0)</td>
<td>0x310</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Offset Lower 0 (ITOFL0)</td>
<td>0x314</td>
</tr>
<tr>
<td>Inbound Functions</td>
<td>Inbound Translation Attribute 0 (ITAT0)</td>
<td>0x318</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x31C</td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inbound Translation 1</td>
<td>ITSAU1 0x320</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITSAL1 0x324</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAU1 0x328</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAL1 0x32C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFU1 0x330</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFL1 0x334</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITAT0 0x338</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved 0x33C</td>
<td></td>
</tr>
<tr>
<td>Inbound Translation 2</td>
<td>ITSAU2 0x340</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITSAL2 0x344</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAU2 0x348</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAL2 0x34C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFU2 0x350</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFL2 0x354</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITAT2 0x358</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved 0x35C</td>
<td></td>
</tr>
<tr>
<td>Inbound Translation 3</td>
<td>ITSAU3 0x360</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITSAL3 0x364</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAU3 0x368</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITEAL3 0x36C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFU3 0x370</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITOFL3 0x374</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITAT3 0x378</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved 0x37C</td>
<td></td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inbound Translation 4</td>
<td>ITSAU4</td>
<td>0x380</td>
</tr>
<tr>
<td></td>
<td>ITSAU5</td>
<td>0x3A0</td>
</tr>
<tr>
<td></td>
<td>ITSAU6</td>
<td>0x3C0</td>
</tr>
<tr>
<td></td>
<td>ITSAL4</td>
<td>0x384</td>
</tr>
<tr>
<td></td>
<td>ITSAL5</td>
<td>0x3A4</td>
</tr>
<tr>
<td></td>
<td>ITSAL6</td>
<td>0x3C4</td>
</tr>
<tr>
<td></td>
<td>ITEAU4</td>
<td>0x388</td>
</tr>
<tr>
<td></td>
<td>ITEAU5</td>
<td>0x3A8</td>
</tr>
<tr>
<td></td>
<td>ITEAU6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL4</td>
<td>0x38C</td>
</tr>
<tr>
<td></td>
<td>ITEAL5</td>
<td>0x3A8</td>
</tr>
<tr>
<td></td>
<td>ITEAL6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITOFU4</td>
<td>0x390</td>
</tr>
<tr>
<td></td>
<td>ITOFU5</td>
<td>0x3B0</td>
</tr>
<tr>
<td></td>
<td>ITOFU6</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFL4</td>
<td>0x394</td>
</tr>
<tr>
<td></td>
<td>ITOFL5</td>
<td>0x3B4</td>
</tr>
<tr>
<td></td>
<td>ITAT4</td>
<td>0x398</td>
</tr>
<tr>
<td></td>
<td>ITAT5</td>
<td>0x3B8</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x39C</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x3BC</td>
</tr>
<tr>
<td>Inbound Translation 5</td>
<td>ITSAU5</td>
<td>0x3A0</td>
</tr>
<tr>
<td></td>
<td>ITSAU6</td>
<td>0x3C0</td>
</tr>
<tr>
<td></td>
<td>ITSAU7</td>
<td>0x3C2</td>
</tr>
<tr>
<td></td>
<td>ITSAL5</td>
<td>0x3A4</td>
</tr>
<tr>
<td></td>
<td>ITSAL6</td>
<td>0x3C4</td>
</tr>
<tr>
<td></td>
<td>ITSAL7</td>
<td>0x3C6</td>
</tr>
<tr>
<td></td>
<td>ITEAU5</td>
<td>0x3A8</td>
</tr>
<tr>
<td></td>
<td>ITEAU6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAU7</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL5</td>
<td>0x3A8</td>
</tr>
<tr>
<td></td>
<td>ITEAL6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL7</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITOFU5</td>
<td>0x3B0</td>
</tr>
<tr>
<td></td>
<td>ITOFU6</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFU7</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFL5</td>
<td>0x3B4</td>
</tr>
<tr>
<td></td>
<td>ITOFL6</td>
<td>0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITOFL7</td>
<td>0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITAT5</td>
<td>0x3B8</td>
</tr>
<tr>
<td></td>
<td>ITAT6</td>
<td>0x3D8</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x3DC</td>
</tr>
<tr>
<td>Inbound Translation 6</td>
<td>ITSAU6</td>
<td>0x3C0</td>
</tr>
<tr>
<td></td>
<td>ITSAU7</td>
<td>0x3C2</td>
</tr>
<tr>
<td></td>
<td>ITSAU8</td>
<td>0x3C4</td>
</tr>
<tr>
<td></td>
<td>ITSAL6</td>
<td>0x3C4</td>
</tr>
<tr>
<td></td>
<td>ITSAL7</td>
<td>0x3C6</td>
</tr>
<tr>
<td></td>
<td>ITSAL8</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAU6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAU7</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAU8</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL6</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL7</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITEAL8</td>
<td>0x3C8</td>
</tr>
<tr>
<td></td>
<td>ITOFU6</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFU7</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFU8</td>
<td>0x3D0</td>
</tr>
<tr>
<td></td>
<td>ITOFL6</td>
<td>0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITOFL7</td>
<td>0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITOFL8</td>
<td>0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITAT6</td>
<td>0x3D8</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x3DC</td>
</tr>
</tbody>
</table>
# Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interrupt</strong></td>
<td><strong>Inbound</strong></td>
<td><strong>Translation</strong></td>
</tr>
<tr>
<td><strong>7</strong></td>
<td><strong>7</strong></td>
<td>** Reserved**</td>
</tr>
<tr>
<td></td>
<td>ITSAU7</td>
<td>0x3E0</td>
</tr>
<tr>
<td></td>
<td>ITSAU7</td>
<td>0x3E4</td>
</tr>
<tr>
<td></td>
<td>ITSAU7</td>
<td>0x3E8</td>
</tr>
<tr>
<td></td>
<td>ITEAL7</td>
<td>0x3EC</td>
</tr>
<tr>
<td></td>
<td>ITOF7</td>
<td>0x3F0</td>
</tr>
<tr>
<td></td>
<td>ITOFL7</td>
<td>0x3F4</td>
</tr>
<tr>
<td></td>
<td>ITAT7</td>
<td>0x3F8</td>
</tr>
<tr>
<td></td>
<td><strong>Reserved</strong></td>
<td>0x3FC</td>
</tr>
<tr>
<td><strong>Inbound</strong></td>
<td><strong>Translation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GCSR</strong></td>
<td><strong>GCSR Base Address Upper (GBAU)</strong></td>
<td>0x400</td>
</tr>
<tr>
<td></td>
<td><strong>GCSR Base Address Lower (GBAL)</strong></td>
<td>0x404</td>
</tr>
<tr>
<td></td>
<td><strong>GCSR Attribute (GCSRAT)</strong></td>
<td>0x408</td>
</tr>
<tr>
<td><strong>Inbound</strong></td>
<td><strong>Translation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CRG</strong></td>
<td><strong>CRG Base Address Upper (CBAU)</strong></td>
<td>0x40C</td>
</tr>
<tr>
<td></td>
<td><strong>CRG Base Address Lower (CBAL)</strong></td>
<td>0x410</td>
</tr>
<tr>
<td></td>
<td><strong>CRG Attribute (CRGAT)</strong></td>
<td>0x414</td>
</tr>
<tr>
<td><strong>Inbound</strong></td>
<td><strong>Translation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CR/CSR</strong></td>
<td><strong>CR/CSR Offset Upper (CROU)</strong></td>
<td>0x418</td>
</tr>
<tr>
<td></td>
<td><strong>CR/CSR Offset Lower (CROL)</strong></td>
<td>0x41C</td>
</tr>
<tr>
<td></td>
<td><strong>CR/CSR Attribute (CRAT)</strong></td>
<td>0x420</td>
</tr>
</tbody>
</table>
Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>31</strong>  <strong>24</strong>  <strong>23</strong>  <strong>16</strong>  <strong>15</strong>  <strong>8</strong>  <strong>7</strong>  <strong>0</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Interrupt Functions</strong></td>
<td>Inbound Translation Location Monitor</td>
<td>Location Monitor Base Address Upper (LMBAU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Location Monitor Base Address Lower (LMBAL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Location Monitor Attribute (LMAT)</td>
</tr>
<tr>
<td></td>
<td>VMEbus Interrupt Control</td>
<td>64-bit Counter Upper (64BCU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64-bit Counter Lower (64BCL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Broadcast Pulse Generator Timer (BPGTR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Broadcast Programmable Clock Timer (BPCTR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMEbus Interrupt Control (VICR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td><strong>Local Bus Interrupt Control</strong></td>
<td></td>
<td>Interrupt Enable (INTEN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt Enable Out (INTEO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt Status (INTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt Clear (INTC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt Map 1 (INTM1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interrupt Map 2 (INTM2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA Controller</td>
<td>DMA Control (DCTL0)</td>
<td>0x500</td>
</tr>
<tr>
<td>0</td>
<td>DMA Status (DSTA0)</td>
<td>0x504</td>
</tr>
<tr>
<td></td>
<td>DMA Current Source Address Upper (DCSAU0)</td>
<td>0x508</td>
</tr>
<tr>
<td></td>
<td>DMA Current Source Address Lower (DCSAL0)</td>
<td>0x50C</td>
</tr>
<tr>
<td></td>
<td>DMA Current Destination Address Upper (DCDAU0)</td>
<td>0x510</td>
</tr>
<tr>
<td></td>
<td>DMA Current Destination Address Lower (DCDAL0)</td>
<td>0x514</td>
</tr>
<tr>
<td></td>
<td>DMA Current Link Address Upper (DCLAU0)</td>
<td>0x518</td>
</tr>
<tr>
<td></td>
<td>DMA Current Link Address Lower (DCLAL0)</td>
<td>0x51C</td>
</tr>
<tr>
<td></td>
<td>DMA Source Address Upper (DSAU0)</td>
<td>0x520</td>
</tr>
<tr>
<td></td>
<td>DMA Source Address Lower (DSAL0)</td>
<td>0x524</td>
</tr>
<tr>
<td></td>
<td>DMA Destination Address Upper (DDAU0)</td>
<td>0x528</td>
</tr>
<tr>
<td></td>
<td>DMA Destination Address Lower (DDAL0)</td>
<td>0x52C</td>
</tr>
<tr>
<td></td>
<td>DMA Source Attribute (DSAT0)</td>
<td>0x530</td>
</tr>
<tr>
<td></td>
<td>DMA Destination Attribute (DDAT0)</td>
<td>0x534</td>
</tr>
<tr>
<td></td>
<td>DMA Next Link Address Upper (DNLAU0)</td>
<td>0x538</td>
</tr>
<tr>
<td></td>
<td>DMA Next Link Address Lower (DNLAL0)</td>
<td>0x53C</td>
</tr>
<tr>
<td></td>
<td>DMA Count (DCNT0)</td>
<td>0x540</td>
</tr>
<tr>
<td></td>
<td>DMA Destination Broadcast Select (DDBS0)</td>
<td>0x544</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 13: LCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA Controller</td>
<td>DCTL1</td>
<td>0x580</td>
</tr>
<tr>
<td>DMA Controller</td>
<td>DSTA1</td>
<td>0x584</td>
</tr>
<tr>
<td></td>
<td>DCSAU1</td>
<td>0x588</td>
</tr>
<tr>
<td></td>
<td>DCSAL1</td>
<td>0x58C</td>
</tr>
<tr>
<td></td>
<td>DCDAU1</td>
<td>0x590</td>
</tr>
<tr>
<td></td>
<td>DCDAL1</td>
<td>0x594</td>
</tr>
<tr>
<td></td>
<td>DCLAU1</td>
<td>0x598</td>
</tr>
<tr>
<td></td>
<td>DCLAL1</td>
<td>0x59C</td>
</tr>
<tr>
<td></td>
<td>DSAU1</td>
<td>0x5A0</td>
</tr>
<tr>
<td></td>
<td>DSAL1</td>
<td>0x5A4</td>
</tr>
<tr>
<td></td>
<td>DDAU1</td>
<td>0x5A8</td>
</tr>
<tr>
<td></td>
<td>DDAL1</td>
<td>0x5AC</td>
</tr>
<tr>
<td></td>
<td>DSAT1</td>
<td>0x5B0</td>
</tr>
<tr>
<td></td>
<td>DDAT1</td>
<td>0x5B4</td>
</tr>
<tr>
<td></td>
<td>DNLAU1</td>
<td>0x5B8</td>
</tr>
<tr>
<td></td>
<td>DNLAL1</td>
<td>0x5BC</td>
</tr>
<tr>
<td></td>
<td>DCNT1</td>
<td>0x5C0</td>
</tr>
<tr>
<td></td>
<td>DDBS1</td>
<td>0x5C4</td>
</tr>
</tbody>
</table>
### 8.4.4 GCSR Register Group Overview

Table 14: GCSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset GCSR/CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCSR</td>
<td>Device ID (DEVI)</td>
<td>0x00/0x600</td>
</tr>
<tr>
<td>Control</td>
<td>Control and Status (GCTRL)</td>
<td>0x04/0x604</td>
</tr>
<tr>
<td>Semaphore</td>
<td>SEMAPHORE0 SEMAPHORE1 SEMAPHORE2 SEMAPHORE3</td>
<td>0x08/0x608</td>
</tr>
<tr>
<td>Mail Box</td>
<td>MBOX0</td>
<td>0x10/0x610</td>
</tr>
<tr>
<td></td>
<td>MBOX1</td>
<td>0x14/0x614</td>
</tr>
<tr>
<td></td>
<td>MBOX2</td>
<td>0x18/0x618</td>
</tr>
<tr>
<td></td>
<td>MBOX3</td>
<td>0x1C/0x61C</td>
</tr>
</tbody>
</table>
# 8.4.5 CR/CSR Register Group Overview

## Table 15: CR/CSR Register Group

<table>
<thead>
<tr>
<th>Function</th>
<th>Bits</th>
<th>Offset CR/CSR/CRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR/CSR</td>
<td>CR/CSR Bit Clear (CSRBCR)</td>
<td>0xFFFF4/0xFF4</td>
</tr>
<tr>
<td></td>
<td>CR/CSR Bit Set (CSRBSR)</td>
<td>0xFFFF8/0xFF8</td>
</tr>
<tr>
<td></td>
<td>CR/CSR Base Address (CBAR)</td>
<td>0xFFFFFC/0xFFC</td>
</tr>
</tbody>
</table>
8. Registers

8.4.6  PCFS Register Group Description
This register group represents the PCI/X Configuration Space. This register group can be viewed from PCI/X configuration space and from the CRG.

In many cases a register represented within the PCFS Register Group has different read/write characteristics than the same register represented within the CRG. Generally, the read/write characteristics of the registers within the PCFS Register Group are strictly limited to the abilities defined by the *PCI Local Bus Specification (Revision 2.2)* and *PCI-X Addendum to PCI Local Bus Specification (Revision 1.0b).*

8.4.7  Vendor ID/ Device ID Registers

Table 16: Vendor ID/ Device ID Registers

<table>
<thead>
<tr>
<th>Register Name: DEVI/VENI</th>
<th>Register Offset: PCFS + 0x00 - CRG + 0x000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEVI</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEVI</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VENI</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VENI</td>
<td></td>
</tr>
</tbody>
</table>

Vendor ID/ Device ID Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>DEVI</td>
<td>Device ID</td>
<td>R</td>
<td>N/A</td>
<td>0x0148</td>
</tr>
<tr>
<td>15:0</td>
<td>VENI</td>
<td>Vendor ID</td>
<td>R</td>
<td>N/A</td>
<td>0x10E3</td>
</tr>
</tbody>
</table>

**Device ID Register (DEVI):** This is a read-only register that uniquely identifies this particular device. This Tsi148 always returns a value of 0x0148.

**Vendor ID Register (VENI):** This is a read-only register that identifies the manufacturer of the device. This identifier is allocated by the PCI/X Special Interest Group to ensure uniqueness. 0x10E3 has been assigned to Tundra and is hard wired as a read-only value.
8.4.8 Command/Status Registers

The Status functionality in this register is used to record information for PCI/X bus related events while the command functionality in this register provides course control over the chips ability to generate and respond to PCI/X cycles.

Table 17: Command/Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>PCI Reset Value</th>
<th>PCI-X Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>DPE</td>
<td>Detected Parity Error</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>SIGSE</td>
<td>Signaled System Error</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>RCVMA</td>
<td>Received Master Abort</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>RCVTA</td>
<td>Received Target Abort</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>SIGTA</td>
<td>Signalled Target Abort</td>
<td>R</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>SELTIM1</td>
<td>DEVSEL Timing</td>
<td>R</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>SELTIM0</td>
<td>DEVSEL Timing</td>
<td>R</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>DPED</td>
<td>Data Parity Error Detected</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>FAST</td>
<td>Fast Back-to-Back Capable</td>
<td>R</td>
<td>N/A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>P66M</td>
<td>PCI 66 MHz</td>
<td>R</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>CAPL</td>
<td>Capabilities List</td>
<td>R</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19:9</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
DPE (Data Parity Error): This bit is set whenever a parity error is detected, even if the parity error response is disabled (see bit PERR in the Section 8.4.8 on page 166). It is cleared by writing it to 1 - writing a 0 has no effect.

SIGSE (Signaled System Error): This bit is set whenever the Tsi148 asserts SERR_. The register is cleared by writing it to 1 while writing a 0 has no effect.

RCVMA (Received Master Abort): This bit is set when a master transaction (except for Special Cycles) is terminated by a master-abort. It is cleared by writing it to 1; writing a 0 has no effect.

RCVTA (Received Target Abort): This bit is set when a master transaction is terminated by a target-abort. The register is cleared by writing it to 1 while writing a 0 has no effect.

SIGTA (Signalled Target Abort): The Tsi148 does not generate a target abort, therefore this bit is hard-wired to a logic 0.

SELTIM (DEVSEL Timing): This field indicates that Tsi148 always asserts DEVSEL_ as a medium responder.

DPED (Data Parity Error Detected): This bit is set when three conditions are met:

1. The Tsi148 asserted PERR_ itself or observed PERR_ asserted
2. The Tsi148 was the PCI/X Master for the transfer in which the error occurred
3. The PERR bit is set. This bit is cleared by writing it to 1; writing a 0 has no effect.

FAST (Fast Back-to-Back Capable): This bit indicates that the Tsi148 is capable of accepting fast back-to-back transactions with different targets.
**P66M (PCI 66 MHz):** This bit indicates the Tsi148 is capable of supporting a 66.67 MHz PCI/X bus.

**CAPL (Capabilities List):** This bit indicates that the address at offset 0x34 is a pointer for a New Capabilities linked list.

**SERR (System Error Enable):** This bit enables the SERR output pin. If cleared, the Tsi148 never drives SERR. If set, the Tsi148 drives SERR active when a system error is detected.

**PERR (Parity Error Response):** This bit enables the PERR output pin. If cleared, the Tsi148 never drives PERR. If set, the Tsi148 drives PERR active when a data parity error is detected.

**MSTR (Bus Master Enable):** If set, the Tsi148 may act as a master on PCI/X. If cleared, the Tsi148 may not act as a master.

**MEMSP (Memory Space Enable):** If set, the Tsi148 does respond to PCI/X memory space accesses when appropriate. If cleared, the Tsi148 does not respond to PCI/X memory space accesses.

**IOSP (I/O Space Enable):** This bit is hard wired to zero. The Tsi148 does not respond to PCI/X I/O space accesses.
8.4.9 Revision ID / Class Code Registers

Table 18: Revision ID / Class Code Register

<table>
<thead>
<tr>
<th>Register Name: CLAS/REVI</th>
<th>Register Offset: PCFS + 0x08 - CRG + 0x008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLAS</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCLAS</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PIC</td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>REVI</td>
</tr>
</tbody>
</table>

Revision ID / Class Code Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>BCLAS</td>
<td>Base Class Code Register</td>
<td>R</td>
<td>N/A</td>
<td>0x06</td>
</tr>
<tr>
<td>23:16</td>
<td>SCLAS</td>
<td>Sub Class Code Register</td>
<td>R</td>
<td>N/A</td>
<td>0x80</td>
</tr>
<tr>
<td>15:8</td>
<td>PIC</td>
<td>Program Interface Code Register</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>REVI</td>
<td>Revision ID Register</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
</tbody>
</table>

BCLAS (Base Class Code Register): This is a read-only register that identifies the base class code of the Tsi148. The Tsi148 always returns a value of 0x06.

SCLAS (Sub Class Code Register): This is a read-only register that identifies the sub class code of the Tsi148. The Tsi148 always returns a value of 0x80.

PIC (Program Interface Code Register): This is a read-only register that identifies the program interface code of the Tsi148. The Tsi148 always returns a value of 0x00.

REVI (Revision ID Register): This is a read-only register that identifies the Tsi148 revision level.
8.4.10  Cache Line Size / Master Latency Timer / Header Type Registers

Table 19: Cache Line Size / Master Latency Timer / Header Type Register

<table>
<thead>
<tr>
<th>Register Name: HEAD/MLAT/CLSZ</th>
<th>Register Offset: PCFS + 0x0C - CRG + 0x00C</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI Reset Value: 0x</td>
<td></td>
</tr>
<tr>
<td>PCI-X Reset Value: 0x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HEAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MLAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLSZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cache Line Size / Master Latency Timer / Header Type Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>PCI Reset Value</th>
<th>PCI-X Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>23:16</td>
<td>HEAD</td>
<td>Header Type</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>15:8</td>
<td>MLAT</td>
<td>Master Latency Timer</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td>0x40</td>
</tr>
<tr>
<td>7:0</td>
<td>CLSZ</td>
<td>Cache Line Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**CLSZ (Cache Line):** These bits represent the number of 32-bit words that define a cache-line. A cache line is defined as 32-bytes, which is eight 32-bit words. If a value of 0x08 is written to this register, the value is retained. If any other value is written to this register, a value of 0x00 is retained.

The **PCI Local Bus Specification (Revision 2.2)** states that this register must power up to all zeros. The Tsi148 does not generate memory write and invalidate command. This register is only used to inform other PCI/X masters of the supported cache-line size for read, read line, and read multiple commands.

**MLAT (Master Latency Timer):** These bits represent the value used for the Master Latency Timer. The Master Latency Timer specifies the amount of PCI/X clock periods that Tsi148 can remain on the PCI/X bus during burst cycles after GNT_ is taken away. The MLAT bits provides a minimum granularity of the 8 PCI/X clock periods.
The *PCI Local Bus Specification (Revision 2.2)* states that this register must power up to all zeros in PCI mode. Severe performance degradation may result if this register is not adjusted from the reset value. This register is initialized to 0x40 in PCI-X mode.

**HEAD (Header Type):** This is a read-only register that identifies this Tsi148 as a Single Function device.
8.4.11 Memory Base Address Lower Register

The MBARL register controls access to the Combined Register Group (CRG).

Table 20: Memory Base Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>BASEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>BASEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>BASEL</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>PRE</td>
<td>MTYP1</td>
<td>MTYP0</td>
<td>IO/MEM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory Base Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>BASEL</td>
<td>Base Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11:4</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>PRE</td>
<td>Prefetch</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>MTYP1</td>
<td>Memory Type</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>1</td>
<td>MTYP0</td>
<td>Memory Type</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>IO/MEM</td>
<td>I/O Space Indicator</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

BASEL (Base Address Lower): These bits define the memory space base address of the (CRG).

PRE (Prefetch): This is a read-only register that reflects the ability of the function to support prefetching. The CRG does not support prefetching.

MTYPx (Memory Type): These bits are hard-wired to 10b to indicate that the CRG can be located anywhere in the 64-bit address space.

IO/MEM (I/O Space Indicator): This bit is set to a zero indicating this resource is a memory space resource. The CRG can only be mapped to memory space.
8.4.12 Memory Base Address Upper Register

The MBARU register controls access to the Combined Register Group (CRG).

Table 21: Memory Base Address Upper Register

<table>
<thead>
<tr>
<th>Register Name: MBARU</th>
<th>Register Offset: PCFS + 0x14 - CRG + 0x014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BASEU</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BASEU</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BASEU</td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BASEU</td>
</tr>
</tbody>
</table>

**BASEU (Base Address Upper):** These bits define the memory space base address of the (CRG).
8.4.13 Subsystem Vendor ID/ Subsystem ID Registers

Table 22: Subsystem Vendor ID/ Subsystem ID Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBI</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBI</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBV</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subsystem Vendor ID/ Subsystem ID Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>CRG Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>SUBI</td>
<td>Subsystem ID</td>
<td>R</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x0000</td>
</tr>
<tr>
<td>15:0</td>
<td>SUBV</td>
<td>Subsystem Vendor ID</td>
<td>R</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x10E3</td>
</tr>
</tbody>
</table>

**SUBI (Subsystem ID):** This is a read-only register from within the PCI/X configuration space (PCFS), and may be written at any time from within the Combined Register Group (CRG). The SUBI register provides a second level of identification for this particular device. This register defaults to 0x0000 upon the release of reset.

**SUBV (Subsystem Vendor ID):** This is a read-only register from within the PCI/X configuration space, and may be written at any time from within the Combined Register Group. The SUBV register provides a second level of identification for the manufacturer of this particular device. This identifier is allocated by the PCI/X Special Interest Group to ensure uniqueness. This register is configured to the Tundra value of 0x10E3 upon release of reset.
8.4.14 Capabilities Pointer Register

This register contains the offset to the first entry in the capabilities list.

Table 23: Capabilities Pointer Register

<table>
<thead>
<tr>
<th>Register Name: CAPP</th>
<th>Reset Value: 0x</th>
<th>Register Offset: PCFS + 0x3C - CRG + 0x03C</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Capabilities Pointer Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>15:0</td>
<td>CAPP</td>
<td>Capabilities Pointer</td>
<td>R</td>
<td>N/A</td>
<td>0x40</td>
</tr>
</tbody>
</table>
8.4.15 Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Registers

Table 24: Interrupt Line/Interrupt Pin/Minimum Grant/Maximum Latency Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>CRG Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>MXLA</td>
<td>Maximum Latency</td>
<td>R</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23:16</td>
<td>MNGN</td>
<td>Minimum Grant</td>
<td>R</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:8</td>
<td>INTP</td>
<td>Interrupt Pin</td>
<td>R</td>
<td>see Table 25</td>
<td>P/S/L</td>
<td>0x01</td>
</tr>
<tr>
<td>7:0</td>
<td>INTL</td>
<td>Interrupt Line</td>
<td>R/W</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**MXLA (Maximum Latency):** This is a read-only register from the PCI/X configuration space, and may be written at any time from within the Combined Register Group. The MXLA register specifies how often access to the PCI/X bus is required. The value is presented in units of 0.25 us. This register defaults 0x00 following the release of reset which indicates that there are no particular latency requirements.

**MNGN (Minimum Grant):** This is a read-only register from the PCI/X configuration space, and may be written at any time from within the Combined Register Group. The MNGN register specifies how long of a burst period is required. The value is presented in units of 0.25 us. This register defaults to 0x00 following the release of reset which indicates that there are no particular grant requirements.
INTP (Interrupt Pin): This register contains information pertaining to the PCI/X interrupt pin being driven. This register is read-only from the PCI/X configuration space, and may be written at any time from within the Combined Register Group. Table 25 shows which bits in the INTP field are read only from within the CRG register group and which bits are both read and write.

### Table 25: CRG Space Type

<table>
<thead>
<tr>
<th>Register Bit</th>
<th>INTP Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>R</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>R</td>
</tr>
<tr>
<td>12</td>
<td>R</td>
</tr>
<tr>
<td>11</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>R/W</td>
</tr>
<tr>
<td>9</td>
<td>R/W</td>
</tr>
<tr>
<td>8</td>
<td>R/W</td>
</tr>
</tbody>
</table>

This Tsi148 is a single function device and is limited by the *PCI Local Bus Specification (Revision 2.2)* to only driving INTA_. In special cases, this Tsi148 can be programmed to drive any one of the four PCI/X interrupts.

This register may be modified to show which of the four interrupt lines is being driven. The recommended encoding of this field is shown in Table 26:

### Table 26: INTP INTx Encoding

<table>
<thead>
<tr>
<th>INTP</th>
<th>PCI/X Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>Undefined</td>
</tr>
<tr>
<td>0x001</td>
<td>INTA_</td>
</tr>
<tr>
<td>0x010</td>
<td>INTB_</td>
</tr>
<tr>
<td>0x011</td>
<td>INTC_</td>
</tr>
<tr>
<td>0x100</td>
<td>INTD_</td>
</tr>
<tr>
<td>0x101 - 0x111</td>
<td>Undefined</td>
</tr>
</tbody>
</table>
Note that the selection of a particular INTx line is handled by the interrupt map registers. The \textbf{INTP} register is for reference only and does not control any hardware.

\textbf{INTL (Interrupt Line)}: This register contains interrupt routing information. This Tsi148 does not have any hardware associated with this register, and is not affected by the contents of this register. Initialization software can write interrupt routing information into this register during system configuration.
### 8.4.16 PCI-X Capabilities Register

**Table 27: PCI-X Capabilities Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td>MOST</td>
<td>MMRBC</td>
<td>ERO</td>
<td>DPERE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>NCAPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>CAPID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:23</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>22:20</td>
<td>MOST</td>
<td>Maximum Outstanding Split Trans</td>
<td>R/W</td>
<td>P/S/L</td>
<td>010b</td>
</tr>
<tr>
<td>19:18</td>
<td>MMRBC</td>
<td>Maximum Memory Read Byte Count</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>ERO</td>
<td>Enable Relaxed Ordering</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>DPERE</td>
<td>Data Parity Recovery Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:8</td>
<td>NCAPP</td>
<td>Next Capabilities Pointer</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>CAPID</td>
<td>Capabilities ID</td>
<td>R</td>
<td>N/A</td>
<td>0x07</td>
</tr>
</tbody>
</table>
MOST (Maximum Outstanding Split Transactions): Three outstanding split transactions are supported. Changing the value of this field decreases the maximum number of outstanding split transactions:

Table 28: MOST Encoding

<table>
<thead>
<tr>
<th>MOST</th>
<th>Maximum Outstanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>1</td>
</tr>
<tr>
<td>001b</td>
<td>2</td>
</tr>
<tr>
<td>010b</td>
<td>3</td>
</tr>
<tr>
<td>011b - 111b</td>
<td>3</td>
</tr>
</tbody>
</table>

MMRBC (Maximum Memory Read Byte Count): This field sets the maximum byte count the device uses when initiating a read sequence with one of the burst memory commands:

Table 29: MMRBC Encoding

<table>
<thead>
<tr>
<th>MMRBC</th>
<th>Byte Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>512</td>
</tr>
<tr>
<td>01b</td>
<td>1024</td>
</tr>
<tr>
<td>10b</td>
<td>2048</td>
</tr>
<tr>
<td>11b</td>
<td>4096</td>
</tr>
</tbody>
</table>

ERO (Enable Relaxed Ordering): The Tsi148 does not support relaxed ordering. When this field is read, the value is always zero.

DPERE (Data Parity Recovery Enable): When this bit is set and the device is in PCI-X mode, the Tsi148 does not assert SERR_ when the master data parity error bit is set. When this bit is clear and the device is in PCI-X mode, the Tsi148 asserts SERR_ when the master data parity error bit is set.

NCAPP (Next Capabilities Pointer): This field points to the next item in the Capabilities List. A zero indicates that this is the final item in the list.

CAPID (Capabilities ID): This field defines this item in the capabilities list as a PCI-X register set. When this field is read, the value is always 0x07.
8.4.17 PCI-X Status Register

Table 30: PCI-X Status Register

<table>
<thead>
<tr>
<th>Register Name: PCIXSTAT</th>
<th>Reset Value: 0x</th>
<th>Register Offset: PCFS + 0x44 - CRG + 0x044</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>RSCEM</td>
</tr>
<tr>
<td>23:16</td>
<td>DMOST</td>
<td>DMMRC</td>
</tr>
<tr>
<td>15:8</td>
<td>BN</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>DN</td>
<td></td>
</tr>
</tbody>
</table>

**PCI-X Status Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>29</td>
<td>RSCEM</td>
<td>Received Split Completion Error Message</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>28:26</td>
<td>DMCRS</td>
<td>Designed Maximum Cumulative Read Size</td>
<td>R/C</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>25:23</td>
<td>DMOST</td>
<td>Designed Maximum Outstanding Split Transactions</td>
<td>R</td>
<td>N/A</td>
<td>0x02</td>
</tr>
<tr>
<td>22:21</td>
<td>DMMRC</td>
<td>Designed Maximum Memory Read Byte Count</td>
<td>R</td>
<td>N/A</td>
<td>0x03</td>
</tr>
<tr>
<td>20</td>
<td>DC</td>
<td>Device Complexity</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>19</td>
<td>USC</td>
<td>Unexpected Split Completion</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>SCD</td>
<td>Split Completion Discarded</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>133C</td>
<td>133 MHz Capable</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>16</td>
<td>64D</td>
<td>64-bit Device</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>15:8</td>
<td>BN</td>
<td>Bus Number</td>
<td>R</td>
<td>N/A</td>
<td>0xFF</td>
</tr>
<tr>
<td>7:3</td>
<td>DN</td>
<td>Device Number</td>
<td>R</td>
<td>N/A</td>
<td>0x1F</td>
</tr>
<tr>
<td>2:0</td>
<td>FN</td>
<td>Function Number</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**RSCEM (Received Split Completion Error Message):** This bit is set if a Split Completion Message is received with the Split Completion Error attribute set. This bit is cleared by
writing a one to it.

**DMCRS (Designed Maximum Cumulative Read Size):** These bits depend on the value of the MMRBC field (see Table 8.4.16 on page 179) as shown in the following table:

**Table 31: DMCRS Encoding**

<table>
<thead>
<tr>
<th>MMRBC</th>
<th>DMRCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>001b</td>
</tr>
<tr>
<td>01b</td>
<td>010b</td>
</tr>
<tr>
<td>10b</td>
<td>011b</td>
</tr>
<tr>
<td>11b</td>
<td>100b</td>
</tr>
</tbody>
</table>

**DMOST (Designed Maximum Outstanding Split Transactions):** These bits always return a value of two. The Tsi148 can have up to three outstanding read transactions.

**DMMRC (Designed Maximum Memory Read Byte Count):** These bits always return a value of three indicating that the Tsi148 has a maximum memory read byte count of 4096 bytes.

**DC (Device Complexity):** This bit always returns a value of one indicating that the Tsi148 is a bridge device.

**USC (Unexpected Split Completion):** This bit is set if an unexpected Split Completion is received. This bit is cleared by writing a one to it.

**SCD (Split Completion Discarded):** This bit is set if a Split Completion is discarded because the requestor would not accept it. This bit is cleared by writing a one to it.

**133C (133 MHz Capable):** This bit always returns a value of one indicating that the Tsi148 is capable of operating at 133 MHz.

**64D (64-bit Device):** This bit always returns a value of one indicating that the Tsi148 has a 64-bit AD interface.

**BN (Bus Number):** This field indicates the number of the bus segment the Tsi148 is attached to. During the attribute phase of a configuration write, the bus number is latched from AD[7:0]. This number is used as part of the Requester ID and Completer ID.

**DN (Device Number):** This field indicates the chip's device number. During the address phase of a configuration write, the device number is latched from AD[15:11]. This number is used as part of the Requester ID and Completer ID.
**FN (Function Number):** This field indicates the number of this function and always returns a value of zero. This number is used as part of the Requester ID and Completer ID.
8. Registers

8.4.18 LCSR Register Group Description
This section defines the Local Control and Status Registers.

8.4.19 Outbound Translation Starting Address Upper (0-7) Registers
The Outbound Translation Starting Address Upper Registers (OTSAU0-OTSAU7) contain address information associated with the mapping of PCI/X Memory space to VMEbus space. The outbound PCI/X address is decoded when the PCI/X address is greater than or equal to the start address and less than or equal to the end address.

Table 32: Outbound Translation Starting Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTSAUx</th>
<th>Register Offset: OTSAU0: CRG + 0x100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>OTSAU1: CRG + 0x120</td>
</tr>
<tr>
<td></td>
<td>OTSAU2: CRG + 0x140</td>
</tr>
<tr>
<td></td>
<td>OTSAU3: CRG + 0x160</td>
</tr>
<tr>
<td></td>
<td>OTSAU4: CRG + 0x180</td>
</tr>
<tr>
<td></td>
<td>OTSAU5: CRG + 0x1A0</td>
</tr>
<tr>
<td></td>
<td>OTSAU6: CRG + 0x1C0</td>
</tr>
<tr>
<td></td>
<td>OTSAU7: CRG + 0x1E0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>STAU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outbound Translation Starting Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>STAU</td>
<td>Start Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**STAU (Start Address Upper):** This field determines the start address of a particular memory area on the PCI/X bus which is used to access VMEbus resources. The value of this field is compared with A63-A32 of the PCI/X bus address.
8.4.20 Outbound Translation Starting Address Lower (0-7) Registers

The Outbound Translation Starting Address Lower Registers (OTSAL0-OTSAL7) contain address information associated with the mapping of PCI/X Memory space to VMEbus space. The outbound PCI/X address is decoded when the PCI/X address is greater than or equal to the start address and less than or equal to the end address.

Table 33: Outbound Translation Starting Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTSALx</th>
<th>Register Offset:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTSAL0: CRG + 0x104</td>
<td>OTSAL1: CRG + 0x124</td>
</tr>
<tr>
<td>OTSAL2: CRG + 0x144</td>
<td>OTSAL3: CRG + 0x164</td>
</tr>
<tr>
<td>OTSAL4: CRG + 0x184</td>
<td>OTSAL5: CRG + 0x1A4</td>
</tr>
<tr>
<td>OTSAL6: CRG + 0x1C4</td>
<td>OTSAL7: CRG + 0x1E4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STAL (Start Address Lower)**: This field determines the start address of a particular memory area on the PCI/X bus which is used to access VMEbus resources. The value of this field is compared with A31-A16 of the PCI/X bus address.
8.4.21 Outbound Translation Ending Address Upper (0-7) Registers

The Outbound Translation Ending Address Upper Registers (OTEAU0-OTEAU7) contain address information associated with the mapping of PCI/X Memory space to VMEbus space. The outbound PCI/X address is decoded when the PCI/X address is greater than or equal to the start address and less than or equal to the end address.

Table 34: Outbound Translation Ending Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ENDU</td>
<td>End Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**ENDU (End Address Upper):** This field determines the end address of a particular memory area on the PCI/X bus which is used to access VMEbus resources. The value of this field is compared with A63-A32 of the PCI/X bus address.
8.4.22 Outbound Translation Ending Address Lower (0-7) Registers

The Outbound Translation Ending Address Lower Registers (OTEAL0-OTEAL7) contain address information associated with the mapping of PCI/X Memory space to VMEbus space. The outbound PCI/X address is decoded when the PCI/X address is greater than or equal to the start address and less than or equal to the end address.

### Table 35: Outbound Translation Ending Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTEALx</th>
<th>Register Offset: OTEAL0: CRG + 0x10C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>OTEAL1: CRG + 0x12C</td>
</tr>
<tr>
<td></td>
<td>OTEAL2: CRG + 0x14C</td>
</tr>
<tr>
<td></td>
<td>OTEAL3: CRG + 0x16C</td>
</tr>
<tr>
<td></td>
<td>OTEAL4: CRG + 0x18C</td>
</tr>
<tr>
<td></td>
<td>OTEAL5: CRG + 0x1AC</td>
</tr>
<tr>
<td></td>
<td>OTEAL6: CRG + 0x1CC</td>
</tr>
<tr>
<td></td>
<td>OTEAL7: CRG + 0x1EC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ENDL (End Address Lower):** This field determines the end address of a particular memory area on the PCI/X bus which is used to access VMEbus resources. The value of this field is compared with A31-A16 of the PCI/X bus address.
8.4.23 Outbound Translation Offset Upper (0-7) Registers

The Outbound Translation Offset Upper Registers (OTOFU0-OTOFU7) contain information associated with the mapping of PCI/X Memory space to VMEbus space.

Table 36: Outbound Translation Offset Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTOFUx</th>
<th>Register Offset:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OTOFU0: CRG + 0x110</td>
</tr>
<tr>
<td></td>
<td>OTOFU1: CRG + 0x130</td>
</tr>
<tr>
<td></td>
<td>OTOFU2: CRG + 0x150</td>
</tr>
<tr>
<td></td>
<td>OTOFU3: CRG + 0x170</td>
</tr>
<tr>
<td></td>
<td>OTOFU4: CRG + 0x190</td>
</tr>
<tr>
<td></td>
<td>OTOFU5: CRG + 0x1B0</td>
</tr>
<tr>
<td></td>
<td>OTOFU6: CRG + 0x1D0</td>
</tr>
<tr>
<td></td>
<td>OTOFU7: CRG + 0x1F0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>OFFU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OFFU (Offset Upper):** This field contains the offset that is added to PCI/X address lines A63-A32 to create the VMEbus address.
8.4.24 Outbound Translation Offset Lower (0-7) Registers

The Outbound Translation Offset Lower Registers (OTOFL0-OTOFL7) contain address information associated with the mapping of PCI/X Memory space to VMEbus space.

Table 37: Outbound Translation Offset Lower (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTOFLx</th>
<th>Register Offset: OTOFL0: CRG + 0x114</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OTOFL1: CRG + 0x134</td>
</tr>
<tr>
<td></td>
<td>OTOFL2: CRG + 0x154</td>
</tr>
<tr>
<td></td>
<td>OTOFL3: CRG + 0x174</td>
</tr>
<tr>
<td></td>
<td>OTOFL4: CRG + 0x194</td>
</tr>
<tr>
<td></td>
<td>OTOFL5: CRG + 0x1B4</td>
</tr>
<tr>
<td></td>
<td>OTOFL6: CRG + 0x1D4</td>
</tr>
<tr>
<td></td>
<td>OTOFL7: CRG + 0x1F4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outbound Translation Starting Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>OFFL</td>
<td>Offset Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**OFFL (Offset Lower):** This field contains the offset that is added to PCI/X address lines A31-A16 to create the VMEbus address.
8.4.25 Outbound Translation 2eSST Broadcast Select (0-7) Registers

The Outbound Translation 2eSST Broadcast Select Registers (OTBS0-OTBS7) contain information associated with the mapping of PCI/X Memory space to VMEbus space.

The 2eSST protocol supports broadcast transfers which allow a master to write the same data to multiple slaves with a single transfer. When this functionality is used, this register determines which VMEbus slaves participates and receives the broadcast data.

Table 38: Outbound Translation 2eSST Broadcast Select (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>20:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2eBS</td>
</tr>
</tbody>
</table>

Outbound Translation 2eSST Broadcast Select (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>20:0</td>
<td>2eBS</td>
<td>2eSST Broadcast Select</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

2eBS (2eSST Broadcast Select): This register contains the 2eSST broadcast select bits. Each bit corresponds to one of the 21 possible slaves. The 2eSST master broadcasts this field during address phase three. Register bit 20 corresponds to VMEbus address line A21 and register bit 0 corresponds to VMEbus address line A1.
8.4.26 Outbound Translation Attribute (0-7) Registers

The Outbound Translation Attribute Registers (OTAT0-OTAT7) contain information associated with the mapping of PCI/X Memory space to VMEbus space.

Table 39: Outbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: OTATx</th>
<th>Register Offset: OTAT0: CRG + 0x11C OTAT1: CRG + 0x13C OTAT2: CRG + 0x15C OTAT3: CRG + 0x17C OTAT4: CRG + 0x19C OTAT5: CRG + 0x1BC OTAT6: CRG + 0x1DC OTAT7: CRG + 0x1FC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>EN</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td>MRPFD</td>
<td>PFS1</td>
<td>PFS0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>2eSSTM2</td>
<td>2eSSTM1</td>
<td>2eSSTM0</td>
<td>TM2</td>
<td>TM1</td>
<td>TM0</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>DBW1</td>
<td>DBW0</td>
<td>SUP</td>
<td>PGM</td>
<td>ADMODE3</td>
<td>ADMODE2</td>
<td>ADMODE1</td>
<td>ADMODE0</td>
</tr>
</tbody>
</table>

Outbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>30:19</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MRPFD</td>
<td>Memory Read Prefetch Disable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>PFS1</td>
<td>Prefetch Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>PFS0</td>
<td>Prefetch Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>13</td>
<td>2eSSTM2</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>2eSSTM1</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>2eSSTM0</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>TM2</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

Outbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>TM1</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>TM0</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>DBW1</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>DBW0</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>SUP</td>
<td>VMEbus Supervisory Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>PGM</td>
<td>VMEbus Program Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>ADMODE3</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>ADMODE2</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>ADMODE1</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>ADMODE0</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**EN (Enable):** If set, the corresponding outbound translation function is enabled.

**MRPFD (Memory Read Prefetch Disable):** If set, prefetching is disabled for all memory read commands. If cleared, a cache line is prefetched when a PCI/X bus memory read burst is received.

**PFS (Prefetch Size):** This field sets the data read prefetch size for PCI/X bus read multiple commands.

**Table 40: Prefetch Size**

<table>
<thead>
<tr>
<th>PFS</th>
<th>Prefetch Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>2 Cache Lines</td>
</tr>
<tr>
<td>01b</td>
<td>4 Cache Lines</td>
</tr>
<tr>
<td>10b</td>
<td>8 Cache Lines</td>
</tr>
<tr>
<td>11b</td>
<td>16 Cache Lines</td>
</tr>
</tbody>
</table>
8. Registers

2eSSTM (2eSST Mode): This field defines the 2eSST Transfer Rate.

Table 41: 2eSST Transfer Rate

<table>
<thead>
<tr>
<th>2eSSTM</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>160 MB/s</td>
</tr>
<tr>
<td>001b</td>
<td>267 MB/s</td>
</tr>
<tr>
<td>010b</td>
<td>320 MB/s</td>
</tr>
<tr>
<td>011b-111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

TM (Transfer Mode): This field defines the VMEbus transfer mode.

Table 42: VMEbus Transfer Mode

<table>
<thead>
<tr>
<th>TM</th>
<th>Transfer Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>SCT</td>
</tr>
<tr>
<td>001b</td>
<td>BLT</td>
</tr>
<tr>
<td>010b</td>
<td>MBLT</td>
</tr>
<tr>
<td>011b</td>
<td>2eVME</td>
</tr>
<tr>
<td>100b</td>
<td>2eSST</td>
</tr>
<tr>
<td>101b</td>
<td>2eSST Broadcast</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

DBW (VMEbus Data Bus Width): These bits define the maximum data bus width for VMEbus transfers initiated by the corresponding outbound translation function. These bits apply to SCT and BLT transfers. MBLT, 2eVME and 2eSST transfers are always 64-bit.

Table 43: VMEbus Data Bus Width

<table>
<thead>
<tr>
<th>DBW</th>
<th>Data Bus Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>16 bit</td>
</tr>
<tr>
<td>01b</td>
<td>32 bit</td>
</tr>
<tr>
<td>10b</td>
<td>Reserved</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
SUP (VMEbus Supervisory Mode): When this bit is set the AM code indicates Supervisory Access, when required. When this bit is cleared the AM code indicates Non-Privileged Access.

PGM (VMEbus Program Mode): When this bit is set the AM code indicates Program Access. When this bit is cleared the AM code indicates Data Access.

AMODE (Address Mode): This field defines the VMEbus Address mode.

Table 44: VMEbus Address Mode

<table>
<thead>
<tr>
<th>AMODE</th>
<th>Address Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000b</td>
<td>A16</td>
</tr>
<tr>
<td>0001b</td>
<td>A24</td>
</tr>
<tr>
<td>0010b</td>
<td>A32</td>
</tr>
<tr>
<td>0011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100b</td>
<td>A64</td>
</tr>
<tr>
<td>0101b</td>
<td>CR/CSR</td>
</tr>
<tr>
<td>0110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0111b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1000b</td>
<td>User1 (AM 0100xxb)</td>
</tr>
<tr>
<td>1001b</td>
<td>User2 (AM 0101xxb)</td>
</tr>
<tr>
<td>1010b</td>
<td>User3 (AM 0110xxb)</td>
</tr>
<tr>
<td>1011b</td>
<td>User4 (AM 0111xxb)</td>
</tr>
<tr>
<td>1100b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

When the User1-User4 modes are used, the AM[1] bit is defined by the SUP bit and the AM[0] bit is defined by the PGM bit.
8. Registers

8.4.27 VMEbus IACK (1-7) Registers

Reading these registers causes an interrupt acknowledge cycle on the VMEbus. A 32-bit read of these registers causes a 32-bit IACK cycle on the VMEbus. A 16-bit read of these registers causes a 16-bit IACK cycle on the VMEbus. An 8-bit read of these registers causes an 8-bit IACK cycle on the VMEbus. Since most VMEbus interrupters support 8-bit IACK cycles, byte reads from offset 3 (0xEF00047, 0xEF0004B, 0xEF0004F, 0xEF00053, 0xEF00057, 0xEF0005B or 0xEF0005F) should be used to retrieve the interrupt vector. Writes to this register are ignored.

Table 45: VMEbus IACK (1-7) Register

<table>
<thead>
<tr>
<th>Register Name: VIACKx</th>
<th>Reset Value: 0xxxxxxxx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Offset:</td>
<td></td>
</tr>
<tr>
<td>VIACK1: CRG + 0x204</td>
<td></td>
</tr>
<tr>
<td>VIACK2: CRG + 0x208</td>
<td></td>
</tr>
<tr>
<td>VIACK3: CRG + 0x20C</td>
<td></td>
</tr>
<tr>
<td>VIACK4: CRG + 0x210</td>
<td></td>
</tr>
<tr>
<td>VIACK5: CRG + 0x214</td>
<td></td>
</tr>
<tr>
<td>VIACK6: CRG + 0x218</td>
<td></td>
</tr>
<tr>
<td>VIACK7: CRG + 0x21C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VIACK</td>
</tr>
</tbody>
</table>

VMEbus IACK (1-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>VIACK</td>
<td>VMEbus IACK</td>
<td>R</td>
<td>N/A</td>
<td>0xxx</td>
</tr>
</tbody>
</table>
8. Registers

8.4.28 VMEbus Read-Modify-Write (RMW) Address Upper Register

This register defines the upper bits (63:32) of the PCI/X bus address for the RMW cycle. Refer to Section 2.5 on page 79 for more information on RMW cycles.

Table 46: VMEbus RMW Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RMWAU</td>
</tr>
</tbody>
</table>

VMEbus RMW Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RMWAU</td>
<td>VMEbus RMW Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.29 VMEbus RMW Address Lower Register

This register defines the lower bits (31:2) of the PCI/X bus address for the RMW cycle.

Table 47: VMEbus RMW Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RMWAL</td>
</tr>
</tbody>
</table>

VMEbus RMW Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RMWAL</td>
<td>VMEbus RMW Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.30 VMEbus RMW Enable Register

This register defines the bits which are involved in the compare and swap operation of the RMW cycle.

Table 48: VMEbus RMW Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VMEbus RMW Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RMWEN</td>
<td>VMEbus RMW Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.31 VMEbus RMW Compare Register

This register defines the bits which are compared with the data read from the VMEbus.

Table 49: VMEbus RMW Compare Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RMWC</td>
</tr>
</tbody>
</table>

VMEbus RMW Compare Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RMWC</td>
<td>VMEbus RMW Compare</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.32 VMEbus RMW Swap Register

This register defines the bits which are written to the VMEbus when the compare is successful.

Table 50: VMEbus RMW Swap Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VMEbus RMW Swap Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RMWS</td>
<td>VMEbus RMW Swap</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.33 VME Master Control Register

The VME Master Control Registers gives the user various control mechanisms on how Tsi148 behaves on the VMEbus as a master. The various data throttling methods are used by the Tsi148 VME Master in all cases when Tsi148 is master on the VMEbus including DMA accesses.

Table 51: VME Master Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>27</td>
<td>VSA</td>
<td>VMEbus Stop Acknowledge</td>
<td>R</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>26</td>
<td>VS</td>
<td>VMEbus Stop</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>25</td>
<td>DHB</td>
<td>Device Has Bus</td>
<td>R</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>24</td>
<td>DWB</td>
<td>Device Wants Bus</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>RMWEN</td>
<td>RMW Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19:17</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>A64DS</td>
<td>A64 Data Strobes</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>14:12</td>
<td>VTOFF</td>
<td>VME Master Time Off</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
VME Master Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:8</td>
<td>VTON</td>
<td>VME Master Time On</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7:5</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>4:3</td>
<td>VREL</td>
<td>VME Master Release Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>VFAIR</td>
<td>VME Master Fair Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1:0</td>
<td>VREQL</td>
<td>VME Master Request Level</td>
<td>R/W</td>
<td>P/S/L</td>
<td>11b</td>
</tr>
</tbody>
</table>

VSA (VMEbus Stop Acknowledge): When this bit is set, the VME Master has obtained mastership of the VMEbus in response to the VS request. This bit is not set if the VME Master has obtained VMEbus ownership for any other reason.

VS (VMEbus Stop): When this bit is set, the Tsi148 requests the VMEbus. When VMEbus ownership has been obtained, the VSA bit is set. VMEbus ownership is maintained until the VS bit is cleared. While the VS bit is set, the PCI/X to VMEbus channel and DMA controllers are prevented from accessing the VMEbus. This bit is used to ensure that the VMEbus is idle before the LRESET bit is set. This bit is cleared and the VMEbus released when the LRSTI_ signal is received.

DHB (Device Has Bus): When this bit is set, the VME Master has obtained mastership of the VMEbus in response to the DWB request. This bit is not set if the VME Master has obtained VMEbus ownership for any other reason.

DWB (Device Wants Bus): When this bit is set, the VME Master requests the VMEbus. When VMEbus ownership has been obtained, the DHB bit is set. VMEbus ownership is maintained until the DWB bit is cleared. While the DWB bit is set, the PCI/X to VMEbus channel and DMA controllers may access the VMEbus.

RMWEN (RMW Enable): If set, the VME Master RMW function is enabled. If cleared, the VME Master RMW function is disabled.

A64DS (A64 Data Strobes): If set, the VME Master asserts both the DS0_ and DS1_ signals during an A64 address phase. If cleared, the VME Master asserts the data strobes based on the following data phase.
VTOFF (VME Master Time Off): These bits define the time the VME Master must wait before re-requesting the VMEbus.

Table 52: VME Master Time Off

<table>
<thead>
<tr>
<th>VTOFF</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>0 μs</td>
</tr>
<tr>
<td>001b</td>
<td>1 μs</td>
</tr>
<tr>
<td>010b</td>
<td>2μs</td>
</tr>
<tr>
<td>011b</td>
<td>4 μs</td>
</tr>
<tr>
<td>100b</td>
<td>8 μs</td>
</tr>
<tr>
<td>101b</td>
<td>16 μs</td>
</tr>
<tr>
<td>110b</td>
<td>32 μs</td>
</tr>
<tr>
<td>111b</td>
<td>64 μs</td>
</tr>
</tbody>
</table>

VTON (VME Master Time On): These bits define the time the VME Master is allowed to spend on the VMEbus. The time on timer is defined in microseconds for the SCT, BLT and MBLT protocols. The time on timer is defined in bytes for the 2eVME and 2eSST protocols. Once the Tsi148 VME Master satisfies VTON it can then access the VMEbus again based on the value programmed for VTOFF.

Table 53: VME Master Time On

<table>
<thead>
<tr>
<th>VTON</th>
<th>Time</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>4 μs</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>001b</td>
<td>8 μs</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>010b</td>
<td>16 μs</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>011b</td>
<td>32 μs</td>
<td>256 Bytes</td>
</tr>
<tr>
<td>100b</td>
<td>64 μs</td>
<td>512 Bytes</td>
</tr>
<tr>
<td>101b</td>
<td>128 μs</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>110b</td>
<td>256 μs</td>
<td>2048 Bytes</td>
</tr>
<tr>
<td>111b</td>
<td>512 μs</td>
<td>4096 Bytes</td>
</tr>
</tbody>
</table>
**VREL** (VME Master **Release Mode**): These bits define the VMEbus release modes for the VMEbus interface.

**Table 54: VME Master Release Mode**

<table>
<thead>
<tr>
<th>VREL</th>
<th>MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>TIME ON or DONE</td>
</tr>
<tr>
<td>01b</td>
<td>(TIME ON and REQ) or DONE</td>
</tr>
<tr>
<td>10b</td>
<td>(TIME ON and BCLR) or DONE</td>
</tr>
<tr>
<td>11b</td>
<td>(TIME ON or DONE) and REQ</td>
</tr>
</tbody>
</table>

**VFAIR** (VME Master **Fair Mode**): If set, the VMEbus requester operates in fair mode. If cleared, the VMEbus requester operates in normal mode.

**VREQL** (VME Master **Request Level**): These bits define the VMEbus request level for the VME Master.
## 8. Registers

### 8.4.34 VMEbus Control Register

#### Table 55: VMEbus Control Register

<table>
<thead>
<tr>
<th>Register Name: VCTRL</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x238</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>7</strong></td>
<td><strong>6</strong></td>
</tr>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td>NELBB</td>
</tr>
<tr>
<td>15:8</td>
<td>SFAILAI</td>
<td>Reserved</td>
</tr>
<tr>
<td>7:0</td>
<td>ATOEN</td>
<td>ROBIN</td>
</tr>
</tbody>
</table>

#### VMEbus Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>N/A</td>
<td>R/W</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>30:28</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>27:24</td>
<td>DLT</td>
<td>Deadlock Timer</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23:21</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>NELBB</td>
<td>No Early Release of Bus Busy</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>19:18</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>SRESET</td>
<td>System Reset</td>
<td>S</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>LRESET</td>
<td>Local Reset</td>
<td>S</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>SFAILAI</td>
<td>System Fail Auto Slot ID</td>
<td>R/W</td>
<td>P/S</td>
<td>0xxx</td>
</tr>
<tr>
<td>14:13</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>12:8</td>
<td>BID</td>
<td>Broadcast ID</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>ATOEN</td>
<td>Arbiter Time-out Enable</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>ROBIN</td>
<td>Round Robin Mode</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>5:4</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>3:0</td>
<td>GTO</td>
<td>VME Master Release Mode</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>
**DLT (Deadlock Timer):** These bits define the time the VME Slave waits after detecting a potential deadlock before asserting the RETRYO signal.

The timer checks only for a potential deadlock. The deadlock condition it monitors is when the Tsi148’s PCI/X to VME write buffers are completely full and a VMEbus initiator attempts a read to a PCI/X target through the Tsi148. This is the only potential deadlock condition this timer monitors.

**Table 56: Deadlock Timer**

<table>
<thead>
<tr>
<th>DLT</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000b</td>
<td>Deadlock Retry Disabled</td>
</tr>
<tr>
<td>0001b</td>
<td>16 VCLKs(^a)</td>
</tr>
<tr>
<td>0010b</td>
<td>32 VCLKs</td>
</tr>
<tr>
<td>0011b</td>
<td>64 VCLKs</td>
</tr>
<tr>
<td>0100b</td>
<td>128 VCLKs</td>
</tr>
<tr>
<td>0101b</td>
<td>256 VCLKs</td>
</tr>
<tr>
<td>0110b</td>
<td>512 VCLKs</td>
</tr>
<tr>
<td>0111b</td>
<td>1024 VCLKs</td>
</tr>
<tr>
<td>1000b</td>
<td>2048 VCLKs</td>
</tr>
<tr>
<td>1001b</td>
<td>4096 VCLKs</td>
</tr>
<tr>
<td>1010b</td>
<td>8192 VCLKs</td>
</tr>
<tr>
<td>1011b</td>
<td>16384 VCLKs</td>
</tr>
<tr>
<td>1100b</td>
<td>32768 VCLKs</td>
</tr>
<tr>
<td>1101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

\(^a\) A VCLK is a Tsi148 internal clock which is a 133 MHz clock.

**NELBB (No Early Release of Bus Busy):** When this bit is set, the Tsi148 asserts BBSYO whenever ASI is asserted. This disables the early release of bus busy function for all VMEbus masters. This can sometimes help debug systems when noise is causing arbitration problems.
8. Registers

SRESET (System Reset): When this bit is set, the SRSTO signal is asserted. This bit is automatically cleared.

LRESET (Local Reset): When this bit is set, the LRSTO signal is asserted. This bit is automatically cleared. Before this bit is set, the software should set the VS bit and wait for the VSA bit to be set.

SFAILAI (SYSFAIL Auto Slot ID): When this bit is set, the SFAILO signal is asserted. When this bit is cleared, the SFAILO signal is negated. When the Auto Slot ID function is enabled, this bit is set by SRSTI_. It is cleared automatically when the auto clear mode is selected. Otherwise it is cleared by software.

BID (Broadcast ID): This field defines the broadcast ID which is used to receive 2eSST broadcast transfers. This field is compared with the broadcast slave select bits which are transmitted during address phase three of a 2eSST transfer. A value of one corresponds to address bit one set, a value of two corresponds to address bit two set and so on through a value of 21. If this field is zero, the VME Slave does not respond to a 2eSST broadcast transfer.

ATOEN (Arbiter Time-out Enable): When this bit is set, the VMEbus arbiter time-out function is enabled. When the time-out function is enabled, the arbiter asserts BBSY* if a bus grant out signal remains asserted for 16 microseconds. This causes the arbiter to re-arbitrate.

ROBIN (Round Robin): When this bit is set, the VMEbus arbiter operates in round robin mode. When this bit is cleared, the VMEbus arbiter operates in priority mode. This bit may be set or cleared at any time.

GTO (VMEbus Global Time-out): These bits define the VMEbus Global Time-out period.

**Table 57: VMEbus Global Time-out**

<table>
<thead>
<tr>
<th>GTO</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000b</td>
<td>8 µs</td>
</tr>
<tr>
<td>0001b</td>
<td>16 µs</td>
</tr>
<tr>
<td>0010b</td>
<td>32 µs</td>
</tr>
<tr>
<td>0011b</td>
<td>64 µs</td>
</tr>
<tr>
<td>0100b</td>
<td>128 µs</td>
</tr>
<tr>
<td>0101b</td>
<td>256 µs</td>
</tr>
<tr>
<td>0110b</td>
<td>512 µs</td>
</tr>
<tr>
<td>0111b</td>
<td>1024 µs</td>
</tr>
<tr>
<td>1000b</td>
<td>2048 µs</td>
</tr>
</tbody>
</table>
### Table 57: VMEbus Global Time-out

<table>
<thead>
<tr>
<th>GTO</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1010b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1100b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111b</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
8. Registers

8.4.35 VMEbus Status Register

Table 58: VMEbus Status Register

<table>
<thead>
<tr>
<th>Register Name: VSTAT</th>
<th>Reset Value: 0x</th>
<th>Register Offset: CRG + 0x23C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>CPURST</td>
<td>BDFAIL</td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>GAP</td>
</tr>
</tbody>
</table>

VME Master Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>CPURST</td>
<td>Clear Power Up Reset</td>
<td>C</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>14</td>
<td>BDFAIL</td>
<td>Board Fail</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x01</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>PURSTS</td>
<td>Power Up Reset Status</td>
<td>R</td>
<td>P</td>
<td>0x01</td>
</tr>
<tr>
<td>11</td>
<td>BDFAILS</td>
<td>Board Fail Status</td>
<td>R</td>
<td>N/A</td>
<td>0x01</td>
</tr>
<tr>
<td>10</td>
<td>SYSFLS</td>
<td>System Fail Status</td>
<td>R</td>
<td>N/A</td>
<td>0xx</td>
</tr>
<tr>
<td>9</td>
<td>ACFAILS</td>
<td>AC Fail Status</td>
<td>R</td>
<td>N/A</td>
<td>0xx</td>
</tr>
<tr>
<td>8</td>
<td>SCONS</td>
<td>System Controller Status</td>
<td>R</td>
<td>P</td>
<td>0xx</td>
</tr>
<tr>
<td>7:6</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>GAP</td>
<td>Geographic Address Parity</td>
<td>R</td>
<td></td>
<td>0xx</td>
</tr>
<tr>
<td>4:0</td>
<td>GA</td>
<td>Geographic Address</td>
<td>R</td>
<td>N/A</td>
<td>0xx</td>
</tr>
</tbody>
</table>

**CPURST (Clear Power Up Reset):** When this bit is set, the PURSTS bit is cleared. This bit always returns zero when read.
**BDFAIL (Board Fail):** This is the board fail control bit. When this bit is high, the BDFAIL_ signal is asserted by the Tsi148. When this bit is low, the BDFAIL_ signal is not asserted by the Tsi148. Board fail is set by a local bus reset and is cleared by software when the board is ready.

**PURSTS (Power Up Reset Status):** This bit is set when the PURSTI_ signal is asserted. It can be cleared by setting the CPURST bit.

**BDFAILS (Board Fail Status):** This is the board fail status bit. When this bit is high, the BDFAIL_ signal is asserted. When this bit is low, the BDFAIL_ signal is negated.

**SYSFLS (System Fail Status):** This bit indicates the current state of the SFAILI_ signal. When this bit is high, the SFAILI_ signal is asserted. When this bit is low, the SFAILI_ signal is negated.

**ACFAILS (AC Fail Status):** This bit indicates the current state of the ACFAILI_ signal. When this bit is high, the ACFAILI_ signal is asserted. When this bit is low, the ACFAILI_ signal is negated.

**SCONS (System Controller Status):** When this bit is high, the VMEbus system controller is enabled. When this bit is low, the VMEbus system controller is not enabled.

**GAP (Geographic Address Parity):** This bit is the parity bit for the Geographic Address. This bit is inverted from the VMEbus GAP_ signal.

**GA (Geographic Address):** These bits represent the Geographic Address of the board. These bits are inverted from the VMEbus GA[4:0]_ signals.
8. Registers

8.4.36 PCI/X Control / Status Register

The PCI/X Control Status Register (PCSR) contains the PCI/X bus configuration information captured from the PCI/X bus on the rising edge of the LRSTI_ signal. This information is used to define the mode, clock frequency and bus width.

Table 59: PCI/X Control / Status Register

<table>
<thead>
<tr>
<th>Register Name: PCSR</th>
<th>Register Offset: CRG + 0x240</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PCI/X Control / Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>26:24</td>
<td>SRTTO</td>
<td>PCI-X Split Response Time-out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>111b</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>22</td>
<td>SRTTS</td>
<td>Split Response Time-out Test</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>CCTM</td>
<td>Configuration Cycle Test Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>DRQ</td>
<td>Disregard REQ64_ Qualification</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>DTTT</td>
<td>Delayed Transaction Time-out Test</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MRCT</td>
<td>Maximum Retry Count Test</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>MRC</td>
<td>Maximum Retry Count</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>SBH</td>
<td>Stop on Byte Holes</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:11</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>SRTE</td>
<td>Split Response Time-out Error</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9</td>
<td>DTTE</td>
<td>Delayed Transaction Time-out Error</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
PCI/X Control / Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>MRCE</td>
<td>Maximum Retry Count Error</td>
<td>R/C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>REQ64S</td>
<td>REQ64 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>5</td>
<td>M66ENS</td>
<td>66 MHz enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>4</td>
<td>FRAMES</td>
<td>FRAME Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>3</td>
<td>IRDYS</td>
<td>IRDY Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>2</td>
<td>DEVSELS</td>
<td>DEVSEL Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>1</td>
<td>STOPS</td>
<td>STOP Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
<tr>
<td>0</td>
<td>TRSDYS</td>
<td>TRDY Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0xxx</td>
</tr>
</tbody>
</table>

**SRTO (PCI-X Split Response Time-out):** These bits define the PCI-X Split Response Time-out period. The Split Response Time-out should be set to a time that is longer than the VMEbus global time-out time. The VMEbus Global time-out timer is a VMEbus system controller function may be controlled by another device.

**Table 60: PCI-X Split Read Time-out**

<table>
<thead>
<tr>
<th>SRTO</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>16 µs</td>
</tr>
<tr>
<td>001b</td>
<td>32 µs</td>
</tr>
<tr>
<td>010b</td>
<td>64 µs</td>
</tr>
<tr>
<td>011b</td>
<td>128 µs</td>
</tr>
<tr>
<td>100b</td>
<td>256 µs</td>
</tr>
<tr>
<td>101b</td>
<td>512 µs</td>
</tr>
<tr>
<td>110b</td>
<td>1024 µs</td>
</tr>
<tr>
<td>111b</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
8. Registers

**SRTT (Split Response Time-out Test):** When this bit is set, the split response time-out time is reduced for test purposes. Only single beat transfers are supported. When this bit is cleared, the split response time-out time is controlled by the SRTO field. This bit is provided for test purposes.

**CCTM (Configuration Cycle Test Mode):** When this bit is set, any VME to PCI/X cycle that uses inbound map decoder number 7 generates PCI/X configuration read and write cycles. Only single beat transfers are supported. When this bit is cleared, inbound map decoder 7 behaves normally. This bit is provided for test purposes.

**DRQ (Disregard REQ64 Qualification):** This bit must be cleared to comply with the *PCI Local Bus Specification (Revision 2.2)*.

**DTTT (Delayed Transaction Time-out Test):** When this bit is set, a delayed transaction times-out after 160 PCI/X bus clocks. When this bit is cleared, a delayed transaction times-out after $2^{15}$ clocks. This bit reduces the time-out count for test purposes.

**MRCT (Maximum Retry Count Test):** When this bit is set and the MRC bit is set, the PCI/X Master retries 16 times before indicating an error. When this bit is cleared and the MRC bit is set, the PCI/X Master retries $2^{24}$ times before indicating an error. This bit reduces the retry count for test purposes.

**MRC (Maximum Retry Count):** When this bit is set, the PCI/X Master counts the number of sequential cycles that are retried. If the count is exceeded, the PCI/X Master aborts the transfer. When this bit is cleared, there is no limit to the number of retry attempts.

**SBH (Stop on Byte Holes):** When this bit is set and the PCI/X bus is configured for conventional mode, the PCI Target issues a stop command when a transfer has non contiguous byte enables. When this bit is clear, the PCI Target issues multiple linkage commands to handle transfers with non contiguous byte enables. This bit is provided for diagnostic purposes.

**SRTE (Split Response Time-out Error):** This bit is set when a split response time-out error occurs. This bit is cleared by writing a one to this bit.

**DTTE (Delayed Transaction Time-out Error):** This bit is set when a delayed transaction time-out error occurs. This bit is cleared by writing a one to this bit.

**MRCE (Maximum Retry Count Error):** This bit is set when the MRC bit is set and the maximum number of retries is exceeded. This bit is cleared by writing a one to this bit.
8. Registers

**REQ64S (REQ64 Status):** When this bit is set, the REQ64_ signal was sampled high at the rising edge of LRSTI_. and the PCI/X A/D bus is configured for 32-bit. When this bit is clear, the REQ64_ signal was sampled low at the rising edge of reset and the PCI/X A/D bus is configured for 64-bit operation.

**M66ENS (66 MHz Enable Status):** When this bit is set, the M66EN signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the M66EN signal was sampled low at the rising edge of LRSTI_.

**FRAMES (FRAME Status):** When this bit is set, the FRAME_ signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the FRAME_ signal was sampled low at the rising edge of LRSTI_.

**IRDYS (IRDY Status):** When this bit is set, the IRDY_ signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the IRDY_ signal was sampled low at the rising edge of LRSTI_.

**DEVSELS (DEVSEL Status):** When this bit is set, the DEVSEL_ signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the DEVSEL_ signal was sampled low at the rising edge of LRSTI_.

**STOPS (STOP Status):** When this bit is set, the STOP_ signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the STOP_ signal was sampled low at the rising edge of LRSTI_.

**TRDYS (TRDY Status):** When this bit is set, the TRDY_ signal was sampled high at the rising edge of LRSTI_. When this bit is clear, the TRDY_ signal was sampled low at the rising edge of LRSTI_.

8.4.37 VMEbus Filter Register.

Table 61: PCI/X Control / Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>25:24</td>
<td>ACKD</td>
<td>Acknowledge Delay</td>
<td>R/W</td>
<td>P</td>
<td>10b</td>
</tr>
<tr>
<td>23:12</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>BGFC</td>
<td>Bus Grant Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x01</td>
</tr>
<tr>
<td>10</td>
<td>BRFC</td>
<td>Bus Request Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x01</td>
</tr>
<tr>
<td>9</td>
<td>BCFC</td>
<td>Bus Clear Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x01</td>
</tr>
<tr>
<td>8</td>
<td>BBFC</td>
<td>Bus Busy Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x01</td>
</tr>
<tr>
<td>7:5</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>AKFC</td>
<td>Acknowledge Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x00</td>
</tr>
<tr>
<td>3:1</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>STFC</td>
<td>Strobe Filter Control</td>
<td>R/W</td>
<td>P</td>
<td>0x00</td>
</tr>
</tbody>
</table>

ACKD (Acknowledge Delay): These bits define the delay time from when the VMEbus data strobes are negated until the acknowledge signals (DTACKO_, BERRO_, and RETRYO_) are
negated.

**Table 62: Acknowledge Delay Time**

<table>
<thead>
<tr>
<th>ACKD</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>Slow</td>
</tr>
<tr>
<td>01b</td>
<td>Medium</td>
</tr>
<tr>
<td>10b</td>
<td>Fast</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**BGFC (Bus Grant Filter Control):** When this bit is set, the VMEbus BG[3:0]IN_ and IACKIN_ signals are filtered with a digital filter to remove noise and glitches. When this bit is clear, the VMEbus BGIN[3:0]_ and IACKIN_ signals are not filtered.

**BRFC (Bus Request Filter Control):** When this bit is set, the VMEbus BR[3:0]I_ signals are filtered with a digital filter to remove noise and glitches. When this bit is clear, the VMEbus BR[3:0]I_ signals are not filtered.

**BCFC (Bus Clear Filter Control):** When this bit is set, the VMEbus BCLRI_ signal is filtered with a digital filter to remove noise and glitches. When this bit is clear, the VMEbus BCLRI_ signal is not filtered.

**BBFC (Bus Busy Filter Control):** When this bit is set, the VMEbus BBSYI_ signal is filtered with a digital filter to remove noise and glitches. When this bit is clear, the VMEbus BBSYI_ signal is not filtered.

**AKFC (Acknowledge Filter Control):** When this bit is set, filtering is applied to the VMEbus acknowledge signals (DTACKI_, BERRI_, and RETRYI_). When this bit cleared, no filtering is applied to the VMEbus acknowledge signals.

**STFC (Strobe Filter Control):** When this bit is set, filtering is applied to the VMEbus strobe signals (ASL_, DS0L_, and DS1L_). When this bit cleared, no filtering is applied to the VMEbus strobe signals.
### 8.4.38 VMEbus Exception Address Upper Register

This register captures VMEbus address bits 63 to 32 whenever the Tsi148 is VME Master and a VMEbus exception occurs. This register is only updated when the VES bit in the VMEbus Exception Attributes register is clear.

#### Table 63: VMEbus Exception Address Upper Register

<table>
<thead>
<tr>
<th>Register Name: VEAU</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x260</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VMEbus Exception Address Upper Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>VEAU</td>
<td>VMEbus Exception Address Upper</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.39 VMEbus Exception Address Lower Register

This register captures VMEbus address bits 31 to 1 whenever the Tsi148 is VME Master and a VMEbus exception occurs. This register is only updated when the VES bit in the VMEbus Exception Attributes register is clear.

Table 64: VMEbus Exception Address Lower Register

<table>
<thead>
<tr>
<th>Register Name: VEAL</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x264</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VMEbus Exception Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>VEAL</td>
<td>VMEbus Exception Address Lower</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.40 VMEbus Exception Attributes Register

Table 65: VMEbus Exception Attributes Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>VES</td>
<td>VEOF</td>
<td>VESCL</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td>2eOT</td>
<td>2eST</td>
<td>BERR</td>
<td>LWORD</td>
<td>WRITE</td>
<td>IACK</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>DS1</td>
<td>DS0</td>
<td>AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td>XAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PCI/X Control / Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>VES</td>
<td>VMEbus Exception Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>30</td>
<td>VEOF</td>
<td>VMEbus Exception Overflow</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>29</td>
<td>VESCL</td>
<td>VMEbus Exception Status Clear</td>
<td>C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>28:22</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>2eOT</td>
<td>2e Odd Termination</td>
<td>R</td>
<td>P/S/L</td>
<td>0x10</td>
</tr>
<tr>
<td>20</td>
<td>2eST</td>
<td>2e Slave Terminated</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>BERR</td>
<td>VMEbus Error</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>LWORD</td>
<td>LWORD</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>WRITE</td>
<td>WRITE</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>IACK</td>
<td>IACK</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>DS1</td>
<td>DS1</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>14</td>
<td>DS0</td>
<td>DS0</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>13:8</td>
<td>AM</td>
<td>AM</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>XAM</td>
<td>XAM</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
VES (VMEbus Exception Status): This bit is set when the VMEbus exception registers are updated. The VMEbus error diagnostic registers are updated when the VES bit is clear and the a VMEbus master transfer is terminated with an error condition, a 2eVME transfer is terminated by the slave or 2eSST transfer is terminated with the last word invalid. If an exception occurs and the VES bit is set, then the current status is retained and the VEOF bit is set. This bit is cleared by writing a one to the VESCL bit.

VEOF (VMEbus Exception Overflow): If the VES bit is clear and a VMEbus exception occurs, the VMEbus error diagnostic registers capture the VMEbus address and attributes. If another error occurs and the VES bit is set, then the VEOF bit is set and the registers are not updated. The VEOF and VES bits are cleared by writing a one to the VESCL bit.

VESCL (VMEbus Exception Status Clear): When this bit is set, the VES and VEOF bits are cleared. This bit always reads zero and writing a zero has no effect.

2eOT (2e Odd Termination): This bit is set when the error diagnostic registers are updated because a 2eSST transfer was terminated with a last word invalid exception. This bit is also set when a 2eVME transfer receives a slave termination or error termination on an odd beat. This bit is only updated when the VES bit is clear.

2eST (2e Slave Terminated): This bit is set when the error diagnostic registers are updated because a 2eVME or 2eSST transfer was terminated by the slave. This bit is only updated when the VES bit is clear.

BERR (VMEbus Error): This bit is set when the error diagnostic registers are updated because a VMEbus transfer was terminated with an error. This bit is only updated when the VES bit is clear.

LWORD (LWORD): This bit captures the state of the VMEbus LWORD_ signal when the Tsi148 is VME Master and an exception occurs. This bit is set when the LWORD_ signal is asserted. This bit is only updated when the VES bit is clear.

WRITE (WRITE): This bit captures the state of the VMEbus WRITE_ signal when the Tsi148 is VME Master and an exception occurs. This bit is set when the WRITEI_ signal is asserted. This bit is only updated when the VES bit is clear.

IACK (IACK): This bit captures the state of the VMEbus IACK_ signal when the Tsi148 is VME Master and an exception occurs. This bit is set when the IACK_ signal is asserted. This bit is only updated when the VES bit is clear.

DS1 (DS1): This bit captures the state of the VMEbus DS1_ signal when the Tsi148 is VME Master and an exception occurs. This bit is set when the DS1I_ signal is asserted. This bit is only updated when the VES bit is clear.
**DS0 (DS0):** This bit captures the state of the VMEbus DS0_ signal when the Tsi148 is VME Master and an exception occurs. This bit is set when the DS0L_ signal is asserted. This bit is only updated when the VES bit is clear.

**AM (AM):** These bits capture the state of the VMEbus AM signals when the Tsi148 is VME Master and an exception occurs. These bits are only updated when the VES bit is clear.

**XAM (XAM):** These bits captures the state of the VMEbus XAM signals when the Tsi148 is VME Master and an exception occurs. These bits are only updated when the VES bit is clear.
8. Registers

8.4.41 Error Diagnostic PCI/X Address Upper Register

This register captures PCI/X bus address bits 63 to 32 whenever PCI/X bus error occurs. This register is only updated when the EDPST bit in the Error Diagnostic PCI/X Attributes register is clear.

Table 66: Error Diagnostic PCI/X Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>EDPAU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Error Diagnostic PCI/X Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>EDPAU</td>
<td>Error Diagnostic PCI/X Address Upper</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.42 Error Diagnostic PCI/X Address Lower Register

This register captures PCI/X address bits 31 to 0 whenever a PCI/X bus error occurs. This register is only updated when the EDPST bit in the Error Diagnostic PCI/X Attributes register is clear.

Table 67: Error Diagnostic PCI/X Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EDPAL</td>
</tr>
</tbody>
</table>

Error Diagnostic PCI/X Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>EDPAL</td>
<td>Error Diagnostic PCI/X Address Lower</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### 8.4.43 Error Diagnostic PCI-X Attribute Register

This register captures the PCI-X bus AD bits 31 to 0 during the attribute phase whenever a PCI-X bus error occurs. This register is only updated when the EDPST bit is clear.

#### Table 68: Error Diagnostic PCI-X Attribute Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EDPXA</td>
</tr>
</tbody>
</table>

**Error Diagnostic PCI-X Attribute Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>EDPXA</td>
<td>Error Diagnostic PCI-X Attribute</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.44 Error Diagnostic PCI-X Split Completion Message Register

This register captures the PCI-X bus split completion message whenever a PC-X bus error occurs. This register is only updated when the EDPST bit in the Error Diagnostic PCI-X Attributes register is clear.

Table 69: Error Diagnostic PCI-X Split Completion Message Register

<table>
<thead>
<tr>
<th>Register Name: EDPXS</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x27C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:0</td>
<td>EDPXS</td>
<td></td>
</tr>
</tbody>
</table>

Error Diagnostic PCI-X Split Completion Message Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>EDPXS</td>
<td>Error Diagnostic PCI-X Split Completion Message</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### 8.4.45 Error Diagnostic PCI/X Attributes Register

**Table 70: Error Diagnostic PCI/X Attributes Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>EDPST</td>
<td>Error Diagnostic PCI/X Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>30:25</td>
<td>EDPOF</td>
<td>Error Diagnostic PCI/X Overflow</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>29:20</td>
<td>EDPCL</td>
<td>Error Diagnostic PCI/X Clear</td>
<td>C</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17:6</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>16: 7</td>
<td>SCD</td>
<td>Split Completion Discarded.</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15: 6</td>
<td>USC</td>
<td>Unexpected Split Completion</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>14: 5</td>
<td>SRT</td>
<td>Split Response Time-out</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>13: 4</td>
<td>SCEM</td>
<td>Split Completion Error Message</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>12: 3</td>
<td>DPED</td>
<td>Data Parity Error Detected.</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11: 2</td>
<td>DPE</td>
<td>Detected Parity Error</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10: 1</td>
<td>MRC</td>
<td>Maximum Retry Count</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9: 0</td>
<td>RMA</td>
<td>Received Master Abort</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8: 0</td>
<td>RTA</td>
<td>Received Target Abort</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7: 0</td>
<td>DTT</td>
<td>Delayed Transaction Time-out</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6: 0</td>
<td>CBEA0</td>
<td>CBE Attribute</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5: 0</td>
<td>CBEA1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: 0</td>
<td>CBEA2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: 0</td>
<td>CBEA3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Register Name: EDPAT
Reset Value: 0x00000000
Register Offset: CRG + 0x280
8. Registers

Error Diagnostic PCI/X Attributes Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:0</td>
<td>COMMx</td>
<td>Command</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**EDPST (Error Diagnostic PCI/X Status):** This bit is set when the PCI/X bus error diagnostic registers are updated. This bit is cleared by writing a one to the EDPCL bit.

**EDPOF (Error Diagnostic PCI/X Overflow):** If the EDPST bit is clear and a PCI/X bus error occurs, the PCI/X bus error diagnostic registers capture the PCI/X bus address and attributes. If another error occurs and the EDPST is set, then the EDPOF bit is set and the registers are not updated. The EDPOF bit is cleared by writing a one to the EDPCL bit.

**EDPCL (Error Diagnostic PCI/X Clear):** When this bit is set, all bits in the EDPAU, EDPAL and EDPAT registers are cleared. This bit always read zero and writing a zero has no effect.

**SCD (Split Completion Discarded):** This bit is set when a split completion is discarded. This bit is only updated when the EDPST bit is clear.

**USC (Unexpected Split Completion):** This bit is set when an unexpected split completion is received. This bit is only updated when the EDPST bit is clear.

**SRT (Split Response Time-out):** This bit is set when a split response time-out occurs. This bit is only updated when the EDPST bit is clear.

**SCEM (Split Completion Error Message):** This bit is set when a split completion error message is received. This bit is only updated when the EDPST bit is clear.

**DPED (Data Parity Error Detected):** This bit is set when three conditions are met: 1) the Tsi148 asserted PERR_ itself or observed PERR_ asserted; 2) the Tsi148 was the PCI/X Master for the transfer in which the error occurred; 3) the PERR bit in the CMMD register is set. This bit is only updated when the EDPST bit is clear.

**DPE (Detected Parity Error):** This bit is set when the PCI/X Master detects a data parity error during a read transaction or the PCI/X Target detects a parity error during a write transaction. This bit is only updated when the EDPST bit is clear.

**MRC (Maximum Retry Count):** This bit is set when the maximum retry count is exceeded. This bit is only updated when the EDPST bit is clear.
**RMA (Received Master Abort):** This bit is set when the master receives a master abort. This bit is only updated when the EDPST bit is clear.

**RTA (Received Target Abort):** This bit is set when the master receives a target abort. This bit is only updated when the EDPST bit is clear.

**DTT (Delayed Transaction Time-out):** This bit is set when there is a delayed transaction time-out. This bit is only updated when the EDPST bit is clear.

**CBEAx (CBE Attribute):** These bits capture the PCI-X bus CBE signals during the attribute phase whenever a PCI/X bus error occurs. These bits are only updated when the EDPST bit is clear.

**COMMx (Command):** These bits capture the PCI/X bus command whenever a PCI/X bus error occurs. These bits are only updated when the EDPST bit is clear.
8. Registers

8.4.46 Inbound Translation Starting Address Upper (0-7) Registers

The Inbound Translation Starting Address Upper Registers (ITSAU0-ITSAU7) contain address information associated with the mapping of VMEbus space to PCI/X space. The Inbound VMEbus address is decoded when the VMEbus address is greater than or equal to the start address and less than or equal to the end address.

Table 71: Inbound Translation Starting Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: ITSAUx</th>
<th>Register Offset: ITSAU0: CRG + 0x300 ITSAU1: CRG + 0x320 ITSAU2: CRG + 0x340 ITSAU3: CRG + 0x360 ITSAU4: CRG + 0x380 ITSAU5: CRG + 0x3A0 ITSAU6: CRG + 0x3C0 ITSAU7: CRG + 0x3E0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type</td>
<td></td>
<td>0x00</td>
</tr>
<tr>
<td>31:0</td>
<td>STAU</td>
<td>Start Address U</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**STAU (Start Address Upper):** This field determines the start address of a particular area on the VMEbus which is used to access local resources. The value of this field is compared with the incoming VMEbus address. If the VMEbus address is 64-bit, then the start address upper is compared with VMEbus address bit 63 to 32. This field is only used when the VMEbus address is 64-bits.
8. Registers

8.4.47 Inbound Translation Starting Address Lower (0-7) Registers

The Inbound Translation Starting Address Lower Registers (ITSAL0-ITSAL7) contain address information associated with the mapping of VMEbus space to PCI/X space. The inbound VMEbus address is decoded when the VMEbus address is greater than or equal to the start address and less than or equal to the end address.

Table 72: Inbound Translation Starting Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: IITSALx</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: ITSAL0: CRG + 0x304</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ITSAL1: CRG + 0x324</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL2: CRG + 0x344</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL3: CRG + 0x364</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL4: CRG + 0x384</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL5: CRG + 0x3A4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL6: CRG + 0x3C4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITSAL7: CRG + 0x3E4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
</tr>
<tr>
<td>7:0</td>
<td>STA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Res</td>
</tr>
</tbody>
</table>

Inbound Translation Starting Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>STAL</td>
<td>Start Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**STAL (Start Address Lower):** If the VMEbus address bus is 64-bit or 32-bit, then the start address lower bits 31 to 16 are compared with VMEbus address bits 31 to 16 and the granularity is 64 Kbytes. If the VMEbus address is 24-bits, then the start address lower bits 23 to 12 are compared with VMEbus address bits 23 to 12 and the granularity is 4 Kbytes. If the VMEbus address is 16-bits, then the start address lower bits 15 to 4 are compared with VMEbus address bits 15 to 4 and the granularity is 16 bytes.
8.4.48 Inbound Translation Ending Address Upper (0-7) Registers

The Inbound Translation Ending Address Upper Registers (ITEAU0-ITEAU7) contain address information associated with the mapping of VMEbus space to PCI/X space. The Inbound VMEbus address is decoded when the VMEbus address is greater than or equal to the start address and less than or equal to the end address.

Table 73: Inbound Translation Ending Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: ITEAUx</th>
<th>Register Offset: ITEAU0: CRG + 0x308</th>
<th>ITEAU1: CRG + 0x328</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>ITEAU2: CRG + 0x348</td>
<td>ITEAU3: CRG + 0x368</td>
</tr>
<tr>
<td></td>
<td>ITEAU4: CRG + 0x388</td>
<td>ITEAU5: CRG + 0x3A8</td>
</tr>
<tr>
<td></td>
<td>ITEAU6: CRG + 0x3C8</td>
<td>ITEAU7: CRG + 0x3E8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inbound Translation Ending Address Upper (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ENDU</td>
<td>End Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**ENDU (End Address Upper):** This field determines the end address of a particular area on the VMEbus which is used to access local resources. The value of this field is compared with the incoming VMEbus address. If the VMEbus address is 64-bit, then the end address upper is compared with VMEbus address bit 63 to 32. This field is only used when the VMEbus address is 64-bits.
8. Registers

8.4.49 Inbound Translation Ending Address Lower (0-7) Registers

The Inbound Translation Ending Address Lower Registers (ITEAL0-ITEAL7) contain address information associated with the mapping of VMEbus space to PCI/X space. The inbound VMEbus address is decoded when the VMEbus address is greater than or equal to the start address and less than or equal to the end address.

Table 74: Inbound Translation Ending Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: IITEALx</th>
<th>Register Offset: ITEAL0: CRG + 0x30C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>ITEAL1: CRG + 0x32C</td>
</tr>
<tr>
<td></td>
<td>ITEAL2: CRG + 0x34C</td>
</tr>
<tr>
<td></td>
<td>ITEAL3: CRG + 0x36C</td>
</tr>
<tr>
<td></td>
<td>ITEAL4: CRG + 0x38C</td>
</tr>
<tr>
<td></td>
<td>ITEAL5: CRG + 0x3AC</td>
</tr>
<tr>
<td></td>
<td>ITEAL6: CRG + 0x3CC</td>
</tr>
<tr>
<td></td>
<td>ITEAL7: CRG + 0x3EC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENDAL</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENDAL</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ENDAL</td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td>ENDAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Inbound Translation Ending Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>ENDAL</td>
<td>Start Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**ENDL (End Address Lower):** If the VMEbus address bus is 64-bit or 32-bit, then the end address lower bits 31 to 16 are compared with VMEbus address bits 31 to 16 and the granularity is 64 Kbytes. If the VMEbus address is 24-bits, then the end address lower bits 23 to 12 are compared with VMEbus address bits 23 to 12 and the granularity is 4 Kbytes. If the VMEbus address is 16-bits, then the end address lower bits 15 to 4 are compared with VMEbus address bits 15 to 4 and the granularity is 16 bytes.
8.4.50 Inbound Translation Offset Upper (0-7) Registers

The Inbound Translation Offset Upper Registers (ITOFU0-ITOFU7) contain information associated with the mapping of VMEbus space to PCI/X space.

Table 75: Inbound Translation Offset Upper (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: IITOFUx</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ITOFU0: CRG + 0x310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU1: CRG + 0x330</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU2: CRG + 0x350</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU3: CRG + 0x370</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU4: CRG + 0x390</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU5: CRG + 0x3B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU6: CRG + 0x3D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITOFU7: CRG + 0x3F0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>OFFU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inbound Translation Offset Upper (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>OFFU</td>
<td>Offset Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

OFFU (Offset Upper): This field contains the offset that is added to VMEbus address bits 63 to 32 to create the PCI/X bus address. If the VMEbus address is not 64-bit, then the internal VMEbus address bits 63 to 32 are zeroed before the offset is added.
8.4.51 Inbound Translation Offset Lower (0-7) Registers

The Inbound Translation Offset Lower Registers (ITOFL0-ITOFL7) contain information associated with the mapping of VMEbus space to PCI/X space.

Table 76: Inbound Translation Offset Lower (0-7) Register

<table>
<thead>
<tr>
<th>Register Name: IITOFL</th>
<th>Register Offset: ITOFL0: CRG + 0x314</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ITOFL1: CRG + 0x334</td>
</tr>
<tr>
<td></td>
<td>ITOFL2: CRG + 0x354</td>
</tr>
<tr>
<td></td>
<td>ITOFL3: CRG + 0x374</td>
</tr>
<tr>
<td></td>
<td>ITOFL4: CRG + 0x394</td>
</tr>
<tr>
<td></td>
<td>ITOFL5: CRG + 0x3B4</td>
</tr>
<tr>
<td></td>
<td>ITOFL6: CRG + 0x3D4</td>
</tr>
<tr>
<td></td>
<td>ITOFL7: CRG + 0x3F4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>OFFL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>OFFL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>OFFL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>OFFL</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inbound Translation Offset Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>OFFL</td>
<td>Offset Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**OFFL (Offset Lower):** This field contains the offset that is added to the lower VMEbus address bits to create the PCI/X bus address. If the VMEbus address is 24-bit, then the internal VMEbus address bits 31 to 24 are zeroed and then offset bits 31 to 12 are added. If the VMEbus address is 16-bit, then the internal VMEbus address bits 31 to 16 are zeroed and offset bits 31 to 4 are added.
8.4.52 Inbound Translation Attribute (0-7) Registers

The Inbound Translation Attribute Registers (ITAT0-ITAT7) contain information associated with the mapping of VMEbus space to PCI/X space.

Table 77: Inbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>EN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>TH</td>
<td>VFS1</td>
<td>VFS0</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>2eSSTM2</td>
<td>2eSSTM1</td>
<td>2eSSTM0</td>
<td>2eSSTB</td>
<td>2eSST</td>
<td>2eVME</td>
<td>MBLT</td>
</tr>
<tr>
<td>7:0</td>
<td>BLT</td>
<td>AS2</td>
<td>AS1</td>
<td>AS0</td>
<td>SUPR</td>
<td>NPRIV</td>
<td>PGM</td>
<td>DATA</td>
</tr>
</tbody>
</table>

Inbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>30:19</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>TH</td>
<td>Threshold</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>VFS1</td>
<td>Virtual FIFO Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>VFS0</td>
<td>Virtual FIFO Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>14</td>
<td>2eSSTM2</td>
<td>2eSSTM</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>13</td>
<td>2eSSTM1</td>
<td>2eSSTM</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>2eSSTM0</td>
<td>2eSSTM</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>2eSSTB</td>
<td>2eSSTB</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>2eSST</td>
<td>2eSST</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
Inbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2eVME</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>MBLT</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>BLT</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>AS2</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>AS1</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AS0</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SUPR</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NPRIV</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PGM</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DATA</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

**EN (Enable):** If set, the corresponding VME Slave window is enabled.

**TH (Threshold):** This field sets a threshold for when read-ahead prefetching resumes. If set, prefetching resumes once the FIFO is half empty. If cleared, prefetching resumes once the FIFO is completely empty.

**VFS (Virtual FIFO Size):** This field is used to set the FIFO size for inbound prefetch reads. The selection of a virtual FIFO size affects the number of initial prefetch read cycles and the number of subsequent prefetch read cycles.

### Table 78: Virtual FIFO Size

<table>
<thead>
<tr>
<th>VFS</th>
<th>FIFO Size</th>
<th>Initial Read</th>
<th>Subsequent Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bytes</td>
<td>Bytes</td>
<td>Bytes</td>
</tr>
<tr>
<td>00b</td>
<td>64</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>01b</td>
<td>128</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>10b</td>
<td>256</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>11b</td>
<td>512</td>
<td>512</td>
<td>256</td>
</tr>
</tbody>
</table>
**2eSSTM (2eSSTM):** These bits define the 2eSST transfer rates the corresponding VME Slave responds to. If SST320 is enabled, the VME Slave also responds to SST267 and SST160. If SST267 is enabled, the VME Slave also responds to SST160.

**Table 79: 2eSST Mode**

<table>
<thead>
<tr>
<th>2eSSTM</th>
<th>2eSST Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>SST160</td>
</tr>
<tr>
<td>001b</td>
<td>SST267</td>
</tr>
<tr>
<td>010b</td>
<td>SST320</td>
</tr>
<tr>
<td>011b-111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**2eSSTB (2eSSTB):** If set, the corresponding VME Slave responds to 2eSST broadcast cycles.

**2eSST (2eSST):** If set, the corresponding VME Slave responds to standard 2eSST cycles.

**2eVME (2eVME):** If set, the corresponding VME Slave responds to 2eVME cycles.

**MBLT (MBLT):** If set, the corresponding VME Slave responds to MBLT cycles.

**BLT (BLT):** If set, the corresponding VME Slave responds to BLT cycles.

**AS (Address Space):** These bits define the address space the corresponding VME Slave responds to.

**Table 80: VMEbus Address Space**

<table>
<thead>
<tr>
<th>AS</th>
<th>Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>A16</td>
</tr>
<tr>
<td>001b</td>
<td>A24</td>
</tr>
<tr>
<td>010b</td>
<td>A32</td>
</tr>
<tr>
<td>011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>100b</td>
<td>A64</td>
</tr>
<tr>
<td>101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
8. Registers

**SUPR (Supervisor):** If set, the corresponding VME Slave is enabled to respond to VMEbus supervisor access cycles.

**NPRIV (Non-privileged):** If set, the corresponding VME Slave is enabled to respond to non-privileged access cycles.

**PGM (Program):** If set, the corresponding VME Slave is enabled to respond to VMEbus program access cycles.

**DATA (Data):** If set, the corresponding VME Slave is enabled to respond to VMEbus data access cycles.
8.4.53 GCSR Base Address Upper Register

This field contains the VMEbus base address of the GCSR registers. The value in this register is compared with VMEbus address bits 63 to 32. This register is only used for during A64 cycles.

Table 81: GCSR Base Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBAU</td>
</tr>
</tbody>
</table>

---

GCSR Base Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>GBAU</td>
<td>GCSR Base Address Upper</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.54 GCSR Base Address Lower Register

This field contains the VMEbus base address of the GCSR registers. If the VMEbus address is A64 or A32, the value in this register is compared with VMEbus address bits 31 to 5. If the VMEbus address is A24, the value in this register is compared with VMEbus address bits 23 to 5. If the VMEbus address is A16, the value in this register is compared with VMEbus address bits 15 to 5.

Table 82: GCSR Base Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OFFL</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OFFL</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OFFL</td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

GCSR Base Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>OFFL</td>
<td>Offset Lower</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>4:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### 8.4.55 GCSR Attribute Register

#### Table 83: GCSR Attribute Register

<table>
<thead>
<tr>
<th>Register Name: GCSRAT</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x408</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>7:0</td>
<td>EN</td>
<td>AS2</td>
<td>AS1</td>
<td>AS0</td>
<td>SUPR</td>
<td>NPRIV</td>
<td>PGM</td>
<td>DATA</td>
</tr>
</tbody>
</table>

#### GCSR Base Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>AS2</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>AS1</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>AS0</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>SUPR</td>
<td>Supervisor</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>NPRIV</td>
<td>Non-privileged</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>PGM</td>
<td>Program</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>DATA</td>
<td>Data</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**EN (Enable):** If set, access to the GCSR registers is enabled.
AS (Address Space): These bits define the address space the GCSR decoder responds to.

Table 84: VMEbus Address Space

<table>
<thead>
<tr>
<th>AS</th>
<th>Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>A16</td>
</tr>
<tr>
<td>001b</td>
<td>A24</td>
</tr>
<tr>
<td>010b</td>
<td>A32</td>
</tr>
<tr>
<td>011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>100b</td>
<td>A64</td>
</tr>
<tr>
<td>101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

NPRIV (Non-privileged): If set, the GCSR decoder is enabled to respond to non-privileged access cycles.

SUPR (Supervisor): If set, the GCSR decoder is enabled to respond to VMEbus supervisor access cycles.

PGM (Program): If set, the GCSR decoder is enabled to respond to VMEbus program access cycles.

DATA (Data): If set, the GCSR decoder is enabled to respond to VMEbus data access cycles.
8. Registers

8.4.56 CRG Base Address Upper Register

This field contains the VMEbus base address of the CRG registers. The value in this register is compared with VMEbus address bits 63 to 32. This register is only used for during A64 cycles.

Table 85: CRG Base Address Upper Register

<table>
<thead>
<tr>
<th>Register Name: CBAU</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x40C</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRG Base Address Upper Register</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>CBAU</td>
<td>CRG Base Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.57 CRG Base Address Lower Register

This field contains the VMEbus base address of the CRG registers. If the VMEbus address is A64 or A32, the value in this register is compared with VMEbus address bits 31 to 12. If the VMEbus address is A24, the value in this register is compared with VMEbus address bits 23 to 12. If the VMEbus address is A16, the value in this register is compared with VMEbus address bits 15 to 12.

Table 86: CRG Base Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>CBAU</td>
<td>CRG Base Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R/W</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### 8.4.58 CRG Attribute Register

#### Table 87: CRG Attribute Register

<table>
<thead>
<tr>
<th>Register Name: CRGAT</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x414</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0 EN AS2 AS1 AS0</td>
<td>SUPR</td>
<td>NPRIV</td>
</tr>
</tbody>
</table>

#### CRG Base Address Lower (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>AS2</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>AS1</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>AS0</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>SUPR</td>
<td>Supervisor</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>NPRIV</td>
<td>Non-privileged</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>PGM</td>
<td>Program</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>DATA</td>
<td>Data</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**EN (Enable):** If set, access to the CRG is enabled.
8. Registers

**AS (Address Space):** These bits define the address space the CRG decoder responds to.

**Table 88: VMEbus Address Space**

<table>
<thead>
<tr>
<th>AS</th>
<th>Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>A16</td>
</tr>
<tr>
<td>001b</td>
<td>A24</td>
</tr>
<tr>
<td>010b</td>
<td>A32</td>
</tr>
<tr>
<td>011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>100b</td>
<td>A64</td>
</tr>
<tr>
<td>101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**NPRIV (Non-privileged):** If set, the CRG decoder is enabled to respond to non-privileged access cycles.

**SUPR (Supervisor):** If set, the CRG decoder is enabled to respond to VMEbus supervisor access cycles.

**PGM (Program):** If set, the CRG decoder is enabled to respond to VMEbus program access cycles.

**DATA (Data):** If set, the CRG decoder is enabled to respond to VMEbus data access cycles.
### 8.4.59 CR/CSR Offset Upper Register

This field contains the offset that is added to the internal VMEbus address bits 63 to 32 to create the PCI/X bus address. During CR/CSR cycles, the internal VMEbus address bits 63 to 32 are forced to zero.

Table 89: CR/CSR Offset Upper Register

<table>
<thead>
<tr>
<th>Register Name: CROU</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x418</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:0</td>
<td>CROU</td>
<td></td>
</tr>
</tbody>
</table>

**CR/CSR Offset Upper Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>CROU</td>
<td>CR/CSR Offset Upper</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.60 CR/CSR Offset Lower Register

This field contains the offset that is added to the internal VMEbus address bits 31 to 19 to create the PCI/X bus address. During CR/CSR cycles, the internal VMEbus address bits 31 to 19 are forced to zero.

Table 90: CR/CSR Offset Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CROL</td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CROL</td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

CR/CSR Offset Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:19</td>
<td>CROL</td>
<td>CR/CSR Base Address Lower</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>18:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.61 CR/CSR Attribute Register

Table 91: CRG Attribute Register

<table>
<thead>
<tr>
<th>Register Name: CRAT</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x420</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>EN</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CRG Attribute Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S</td>
<td>0xx</td>
</tr>
<tr>
<td>6:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

EN (Enable): If set, access to the CR/CSR registers are enabled. The initial value of this bit is determined the hardware configuration (see Section 5.4.2.1 on page 130).
8. Registers

8.4.62 Location Monitor Base Address Upper Register

This field contains the VMEbus base address of the Locations to be monitored. The value in this register is compared with VMEbus address bits 63 to 32. This register is only used for during A64 cycles.

Table 92: Location Monitor Base Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Location Monitor Base Address Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>LMBAU</td>
<td>Location Monitor Base Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.63 Location Monitor Base Address Lower Register

This field contains the VMEbus base address of the location to be monitored. If the VMEbus address is A64 or A32, the value in this register is compared with VMEbus address bits 31 to 5. If the VMEbus address is A24, the value in this register is compared with VMEbus address bits 23 to 5. If the VMEbus address is A16, the value in this register is compared with VMEbus address bits 15 to 5.

Table 93: Location Monitor Base Address Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td>LMBAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td>LMBAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td>LMBAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>LMBAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CR/CSR Offset Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>LMBAL</td>
<td>Location Monitor Base Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### 8.4.64 Location Monitor Attribute Register

Table 94: Location Monitor Register

<table>
<thead>
<tr>
<th>Register Name: LMAT</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x42C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>EN</td>
<td>AS2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>EN</td>
<td>Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>AS2</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>AS1</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>AS0</td>
<td>Address Space</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>SUPR</td>
<td>Supervisor</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>NPRIV</td>
<td>Non-privileged</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>PGM</td>
<td>Program</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>DATA</td>
<td>Data</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**EN (Enable):** If set, the location monitor is enabled.
AS (Address Space): These bits define the address space the location monitor responds to:

Table 95: VMEbus Address Space

<table>
<thead>
<tr>
<th>AS</th>
<th>Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>A16</td>
</tr>
<tr>
<td>001b</td>
<td>A24</td>
</tr>
<tr>
<td>010b</td>
<td>A32</td>
</tr>
<tr>
<td>011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>100b</td>
<td>A64</td>
</tr>
<tr>
<td>101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

NPRIV (Non-privileged): If set, the location monitor is enabled to respond to non-privileged access cycles.

SUPR (Supervisor): If set, the location monitor is enabled to respond to VMEbus supervisor access cycles.

PGM (Program): If set, the location monitor is enabled to respond to VMEbus program access cycles.

DATA (Data): If set, the location monitor is enabled to respond to VMEbus data access cycles.
8. Registers

8.4.65 64-bit Counter Upper

These bits are the most significant bits of the 64-bit counter. The 64-bit counter can be used to count events on the VMEbus IRQ[1]I_ or IRQ[2]I_ signal lines. Since the 64-bit counter is comprised of two 32-bit registers, it is possible that the lower counter may roll over between a read or write of the upper and lower portions. Software must consider this case.

Table 96: 64-bit Counter Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

64-bit Counter Upper Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>64BCU</td>
<td>64-bit Counter Upper</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.66 64-bit Counter Lower

These bits are the least significant bits of the 64-bit counter. The 64-bit counter can be used to count events on the VMEbus IRQ[1]I or IRQ[2]I signal lines. Since the 64-bit counter is comprised of two 32-bit registers, it is possible that the lower counter may roll over between a read or write of the upper and lower portions. Software must consider this case.

Table 97: 64-bit Counter Lower Register

<table>
<thead>
<tr>
<th>Register Name: 64BCL</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 0x434</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

31:0 64BCL

64-bit Counter Lower Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>64BCL</td>
<td>64-bit Counter Lower</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.67 Broadcast Pulse Generator Timer Register

The value in this register is compared to that of the internal Broadcast Pulse Generator Counter. When they are equal, a broadcast interrupt pulse is generated and the counter is reset to 0. The value in this register determines the broadcast interrupt pulse width in approximately 30-ns increments. Due to the required glitch filters on the VMEbus IRQ[1]I_ and IRQ[2]I_ signal lines, the value written to this register must be greater than 0x0003. Approximately, the broadcast interrupt pulse width is programmable from 120 ns to 1.97 ms. After power-up, this register is initialized to 0x0022 which produces a 1.02-µs pulse if broadcast pulse mode is enabled. Writing a value of all zeros to this register has no effect.

Table 98: Broadcast Pulse Generator Timer Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>15:0</td>
<td>BPGT</td>
<td>Broadcast Pulse Generator Timer</td>
<td>R/W</td>
<td>P/S</td>
<td>0x22</td>
</tr>
</tbody>
</table>
8.4.68 **Broadcast Programmable Clock Timer Register**

The value in this register is compared to that of the internal Broadcast Programmable Clock Counter. When they are equal, a broadcast interrupt clock is generated and the counter is reset to 0. The value in this register determines the broadcast interrupt clock rate in approximately 1.02-µs increments. Due to the required glitch filters on the VMEbus IRQ[1]I_ and IRQ[2]I_ signal lines, the value written to this register must be greater than 0x000001. Approximately, the broadcast interrupt clock rate is programmable from 2.04 µs to 17.11 seconds. After power-up, this register is initialized to 0x0003E8 which produces a 1.02-ms clock if broadcast programmable clock mode is enabled. Writing a value of all zeros to this register has no effect.

**Table 99: Broadcast Programmable Clock Timer Register**

<table>
<thead>
<tr>
<th>Register Name: BPCTR</th>
<th>Reset Value: 0x000003E8</th>
<th>Register Offset: CRG + 0x43C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>BPCT</td>
</tr>
<tr>
<td>BPCT</td>
<td>BPCT</td>
<td>BPCT</td>
</tr>
<tr>
<td>BPCT</td>
<td>BPCT</td>
<td>BPCT</td>
</tr>
</tbody>
</table>

**Broadcast Programmable Clock Timer Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>23:0</td>
<td>BPCT</td>
<td>Broadcast Programmable Clock Timer</td>
<td>R/W</td>
<td>P/S</td>
<td>0x3E8</td>
</tr>
</tbody>
</table>
8.4.69 VMEbus Interrupt Control Register

The VMEbus Interrupt Control Register is used to control the VMEbus interrupt function.

Table 100: VMEbus Interrupt Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>CNTS</td>
<td>Counter Source</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>29:28</td>
<td>EDGIS</td>
<td>Edge Interrupt Source</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>27:26</td>
<td>IRQIF</td>
<td>IRQ1 Function</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>25:24</td>
<td>IRQ2F</td>
<td>IRQ2 Function</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>23</td>
<td>BIP</td>
<td>Broadcast Interrupt Pulse</td>
<td>S</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>22</td>
<td>BIPS</td>
<td>Broadcast Interrupt Pulse Status</td>
<td>R</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>21:16</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>IRQC</td>
<td>VMEbus IRQ Clear</td>
<td>S</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>14:12</td>
<td>IRQLS</td>
<td>VMEbus IRQ Level Status</td>
<td>R</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>IRQS</td>
<td>VMEbus IRQ Status</td>
<td>R</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>10:8</td>
<td>IRQL</td>
<td>VMEbus IRQ Level</td>
<td>S</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>STID</td>
<td>STATUS/ID</td>
<td>R/W</td>
<td>P/S</td>
<td>0x0F</td>
</tr>
</tbody>
</table>
8. Registers

CNTS (Counter Source): These bits define input to the 64-bit counter.

Table 101: Counter Source

<table>
<thead>
<tr>
<th>CNTS</th>
<th>Counter Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>Counter Disable</td>
</tr>
<tr>
<td>01b</td>
<td>IRQ[1]</td>
</tr>
<tr>
<td>10b</td>
<td>IRQ[2]</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

EDGIS (Edge Interrupt Source): These bits define input to VMEbus edge interrupt logic.

Table 102: Edge Interrupt Source

<table>
<thead>
<tr>
<th>EDGIS</th>
<th>Edge Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>Edge Interrupt Disable</td>
</tr>
<tr>
<td>01b</td>
<td>IRQ[1]</td>
</tr>
<tr>
<td>10b</td>
<td>IRQ[2]</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

IRQ1F (IRQ1 Function): These bits define the function of the VMEbus IRQ[1]O signal line as an output.

Table 103: VMEbus IRQ[1]O Function

<table>
<thead>
<tr>
<th>IRQ1F</th>
<th>VMEbus IRQ[1]O Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>Normal</td>
</tr>
<tr>
<td>01b</td>
<td>Pulse Generator</td>
</tr>
<tr>
<td>10b</td>
<td>Programmable Clock</td>
</tr>
<tr>
<td>11b</td>
<td>1.02 μs Clock</td>
</tr>
</tbody>
</table>
8. Registers

**IRQ2F (IRQ2 Function):** These bits define the function of the VMEbus IRQ[2]O signal line as an output.

<table>
<thead>
<tr>
<th>IRQ2F</th>
<th>VMEbus IRQ[2]O Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>Normal</td>
</tr>
<tr>
<td>01b</td>
<td>Pulse Generator</td>
</tr>
<tr>
<td>10b</td>
<td>Programmable Clock</td>
</tr>
<tr>
<td>11b</td>
<td>1.02 µs Clock</td>
</tr>
</tbody>
</table>

**Table 104: VMEbus IRQ[2]O Function**

**BIP (Broadcast Interrupt Pulse):** When the broadcast interrupt pulse mode is enabled, setting this bit causes a pulse to be generated on the VMEbus IRQ[1]O or IRQ[2]O signal line. This bit always reads zero and writing a zero has no effect.

**BIPS (Broadcast Interrupt Pulse Status):** When this bit is high, the broadcast interrupt pulse is still being generated by the pulse generator. When this bit is low, the pulse generator has finished generating the broadcast interrupt pulse. This is a read-only status bit.

**IRQC (VMEbus IRQ Clear):** When this bit is set high, the IRQL bits are reset and the VMEbus interrupt is removed. This bit should only be used to recover from an error condition. Normally VMEbus interrupts should not be removed. This bit always reads zero and writing a zero has no effect.

**IRQLS (VMEbus IRQ Level Status):** These bits are read-only status bits and they define the current level of a pending VMEbus interrupt.

**IRQS (VMEbus IRQ Status):** When this bit is high, the VMEbus interrupt has not been acknowledged. When this bit is low, the VMEbus interrupt has been acknowledged. This is a read-only status bit.

**IRQL (VMEbus IRQ Level):** These bits define the level of the VMEbus interrupt generated by the Tsi148. A VMEbus interrupt is generated by writing the desired level to these bits. These bits always read 0 and writing a 0 to these bits has no effect. These bits are automatically cleared following the VMEbus interrupt acknowledge cycle.

**STID (STATUS/ID):** These bits define the VMEbus vector that is returned during an interrupt acknowledge cycle.
### 8.4.70 Interrupt Enable Register

Table 105: Interrupt Enable Register

<table>
<thead>
<tr>
<th>Register Name: INTEN</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CRG + 448</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:26</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>25</td>
<td>DMA1EN</td>
<td>DMAC 1 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>24</td>
<td>DMA0EN</td>
<td>DMAC 0 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23</td>
<td>LM3EN</td>
<td>Location Monitor 3 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>22</td>
<td>LM2EN</td>
<td>Location Monitor 2 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>LM1EN</td>
<td>Location Monitor 1 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>LM0EN</td>
<td>Location Monitor 0 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>MB3EN</td>
<td>Mail Box 3 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MB2EN</td>
<td>Mail Box 2 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>MB1EN</td>
<td>Mail Box 1 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>MB0EN</td>
<td>Mail Box 0 Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>13</td>
<td>PERREN</td>
<td>PCI/X Bus Error Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>VERREN</td>
<td>VMEbus Error Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>VIEEN</td>
<td>VMEbus IRQ Edge Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>IACKEN</td>
<td>Interrupt Acknowledge Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

### Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>SYSFLEN</td>
<td>System Fail Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>ACFLEN</td>
<td>AC Fail Interrupt Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>IRQ7EN</td>
<td>IRQ7 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>IRQ6EN</td>
<td>IRQ6 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>IRQ5EN</td>
<td>IRQ5 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>IRQ4EN</td>
<td>IRQ4 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>IRQ3EN</td>
<td>IRQ3 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>IRQ2EN</td>
<td>IRQ2 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>IRQ1EN</td>
<td>IRQ1 Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**DMA1EN (DMAC 1 Interrupt Enable):** When this bit is high, the DMA 1 controller interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the DMA 1 controller interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

**DMA0EN (DMAC 0 Interrupt Enable):** When this bit is high, the DMA 0 controller interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the DMA 0 controller interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

**LM3EN (Location Monitor 3 Interrupt Enable):** When this bit is high, the location monitor 3 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the location monitor 3 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.
8. Registers

LM2EN (Location Monitor 2 Interrupt Enable): When this bit is high, the location monitor 2 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the location monitor 2 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

LM1EN (Location Monitor 1 Interrupt Enable): When this bit is high, the location monitor 1 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the location monitor 1 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

LM0EN (Location Monitor 0 Interrupt Enable): When this bit is high, the location monitor 0 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the location monitor 0 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

MB3EN (Mail Box 3 Interrupt Enable): When this bit is high, the mail box 3 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the mail box 3 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

MB2EN (Mail Box 2 Interrupt Enable): When this bit is high, the mail box 2 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the mail box 2 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

MB1EN (Mail Box 1 Interrupt Enable): When this bit is high, the mail box 1 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the mail box 1 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

MB0EN (Mail Box 0 Interrupt Enable): When this bit is high, the mail box 0 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the mail box 0 interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

PERREN (PCI/X Bus Error Interrupt Enable): When this bit is high, the PCI/X bus error enabled. When the interrupt is enabled, the status bit indicates the state of the PCI/X buss error interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.
VERREN (VMEbus Error Interrupt Enable): When this bit is high, the VMEbus error interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus error interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

VIEEN (VMEbus IRQ Edge Interrupt Enable): When this bit is high, the VMEbus IRQ edge interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ edge interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

IACKEN (Interrupt Acknowledge Interrupt Enable): When this bit is high, the VMEbus interrupt acknowledge interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus interrupt acknowledge interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

SYSFLEN (System Fail Interrupt Enable): When this bit is high, the VMEbus system fail interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus system fail interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

ACFLEN (AC Fail Interrupt Enable): When this bit is high, the AC fail interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the AC fail interrupt. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

IRQ7EN (IRQ7 Enable): When this bit is high, the VMEbus IRQ7 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[7]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

IRQ6EN (IRQ6 Enable): When this bit is high, the VMEbus IRQ6 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[6]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

IRQ5EN (IRQ5 Enable): When this bit is high, the VMEbus IRQ5 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[5]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.
**IRQ4EN (IRQ4 Enable):** When this bit is high, the VMEbus IRQ4 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[4]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

**IRQ3EN (IRQ3 Enable):** When this bit is high, the VMEbus IRQ3 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[3]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

**IRQ2EN (IRQ2 Enable):** When this bit is high, the VMEbus IRQ2 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[2]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.

**IRQ1EN (IRQ1 Enable):** When this bit is high, the VMEbus IRQ1 interrupt is enabled. When the interrupt is enabled, the status bit indicates the state of the VMEbus IRQ[1]I_ signal line. A local bus interrupt is generated if the corresponding interrupt out bit is set. The interrupt can be polled by setting the enable bit and clearing the interrupt out bit.
### 8. Registers

#### 8.4.71 Interrupt Enable Out Register

**Table 106: Interrupt Enable Out Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:26</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>31:25</td>
<td>DMA1EO</td>
<td>DMAC 1 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:24</td>
<td>DMA0EO</td>
<td>DMAC 0 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:20</td>
<td>LM3EO</td>
<td>Location Monitor 3 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:19</td>
<td>LM2EO</td>
<td>Location Monitor 2 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:18</td>
<td>LM1EO</td>
<td>Location Monitor 1 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:17</td>
<td>LM0EO</td>
<td>Location Monitor 0 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:16</td>
<td>MB3EO</td>
<td>Mail Box 3 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:15</td>
<td>MB2EO</td>
<td>Mail Box 2 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:14</td>
<td>MB1EO</td>
<td>Mail Box 1 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>31:13</td>
<td>MB0EO</td>
<td>Mail Box 0 Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>15:13</td>
<td>PERREO</td>
<td>PCI/X Bus Error Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:12</td>
<td>VERREO</td>
<td>VMEbus Error Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:11</td>
<td>VIEEO</td>
<td>VMEbus IRQ Edge Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:10</td>
<td>IACKEO</td>
<td>Interrupt Acknowledge Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
Interrupt Enable Out Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>SYSFLEO</td>
<td>System Fail Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>ACFLEO</td>
<td>AC Fail Interrupt Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>IRQ7EO</td>
<td>IRQ7 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>IRQ6EO</td>
<td>IRQ6 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>IRQ5EO</td>
<td>IRQ5 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>IRQ4EO</td>
<td>IRQ4 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>IRQ3EO</td>
<td>IRQ3 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>IRQ2EO</td>
<td>IRQ2 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>IRQ1EO</td>
<td>IRQ1 Enable Out</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

DMA1EO (DMA 1 Interrupt Enable Out): When this bit is high, the DMA 1 controller interrupt is enabled to the one of the four INTx pins.

DMA0EO (DMA 0 Interrupt Enable Out): When this bit is high, the DMA 0 controller interrupt is enabled to the one of the four INTx pins.

LM3EO (Location Monitor 3 Interrupt Enable Out): When this bit is high, the location monitor 3 interrupt is enabled to the one of the four INTx pins.

LM2EO (Location Monitor 2 Interrupt Enable Out): When this bit is high, the location monitor 2 interrupt is enabled to the one of the four INTx pins.

LM1EO (Location Monitor 1 Interrupt Enable Out): When this bit is high, the location monitor 1 interrupt is enabled to the one of the four INTx pins.

LM0EO (Location Monitor 0 Interrupt Enable Out): When this bit is high, the location monitor 0 interrupt is enabled to the one of the four INTx pins.

MB3EO (Mail Box 3 Interrupt Enable Out): When this bit is high, the mail box 3 interrupt is enabled to the one of the four INTx pins.

MB2EO (Mail Box 2 Interrupt Enable Out): When this bit is high, the mail box 2 interrupt is enabled to the one of the four INTx pins.
8. Registers

**MB1EO (Mail Box 1 Interrupt Enable Out):** When this bit is high, the mail box 1 interrupt is enabled to the one of the four INTx pins.

**MB0EO (Mail Box 0 Interrupt Enable Out):** When this bit is high, the mail box 0 interrupt is enabled to the one of the four INTx pins.

**PERREO (PCI/X Bus Error Enable Out):** When this bit is high, the PCI/X bus error interrupt is enabled to the one of the four INTx pins.

**VERREO (VMEbus Error Interrupt Enable Out):** When this bit is high, the VMEbus error interrupt is enabled to the one of the four INTx pins.

**VIEEO (VMEbus IRQ Edge Interrupt Enable Out):** When this bit is high, the VMEbus IRQ edge interrupt is enabled to the one of the four INTx pins.

**IACKE0 (Interrupt Acknowledge Interrupt Enable Out):** When this bit is high, the VMEbus interrupt acknowledge interrupt is enabled to the one of the four INTx pins.

**SYSFLEO (System Fail Interrupt Enable Out):** When this bit is high, the VMEbus system fail interrupt is enabled to the one of the four INTx pins.

**ACFLEO (AC Fail Interrupt Enable Out):** When this bit is high, the AC fail interrupt is enabled to the one of the four INTx pins.

**IRQ7EO (IRQ7 Enable Out):** When this bit is high, the VMEbus IRQ[7] interrupt is enabled to the one of the four INTx pins.

**IRQ6EO (IRQ6 Enable Out):** When this bit is high, the VMEbus IRQ[6] interrupt is enabled to the one of the four INTx pins.

**IRQ5EO (IRQ5 Enable Out):** When this bit is high, the VMEbus IRQ[5] interrupt is enabled to the one of the four INTx pins.

**IRQ4EO (IRQ4 Enable Out):** When this bit is high, the VMEbus IRQ[4] interrupt is enabled to the one of the four INTx pins.

**IRQ3EO (IRQ3 Enable Out):** When this bit is high, the VMEbus IRQ[3] interrupt is enabled to the one of the four INTx pins.

**IRQ2EO (IRQ2 Enable Out):** When this bit is high, the VMEbus IRQ[2] interrupt is enabled to the one of the four INTx pins.

**IRQ1EO (IRQ1 Enable Out):** When this bit is high, the VMEbus IRQ[1] interrupt is enabled to the one of the four INTx pins.
### 8.4.72 Interrupt Status Register

Table 107: Interrupt Status Register

<table>
<thead>
<tr>
<th>Register Name: INTS</th>
<th>Register Offset: CRG + 450</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>CRG + 450</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>DMA1S</td>
<td>DMA0S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>LM3S</td>
<td>LM2S</td>
<td>LM1S</td>
<td>LM0S</td>
<td>MB3S</td>
<td>MB2S</td>
<td>MB1S</td>
<td>MB0S</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>PERRS</td>
<td>VERRS</td>
<td>VIES</td>
<td>IACKS</td>
<td>SYSFLS</td>
<td>ACFLS</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>IRQ7S</td>
<td>IRQ6S</td>
<td>IRQ5S</td>
<td>IRQ4S</td>
<td>IRQ3S</td>
<td>IRQ2S</td>
<td>IRQ1S</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Interrupt Enable Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:26</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>25</td>
<td>DMA1S</td>
<td>DMAC 1 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>24</td>
<td>DMA0S</td>
<td>DMAC 0 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23</td>
<td>LM3S</td>
<td>Location Monitor 3 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>22</td>
<td>LM2S</td>
<td>Location Monitor 2 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>LM1S</td>
<td>Location Monitor 1 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>LM0S</td>
<td>Location Monitor 0 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>MB3S</td>
<td>Mail Box 3 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MB2S</td>
<td>Mail Box 2 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>MB1S</td>
<td>Mail Box 1 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>MB0S</td>
<td>Mail Box 0 Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>13</td>
<td>PERRS</td>
<td>PCI/X Bus Error Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>VERRS</td>
<td>VMEbus Error Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>VIES</td>
<td>VMEbus IRQ Edge Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>IACKS</td>
<td>Interrupt Acknowledge Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
### Interrupt Enable Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>SYSFLS</td>
<td>System Fail Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>ACFLS</td>
<td>AC Fail Interrupt Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>IRQ7S</td>
<td>IRQ7 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>IRQ6S</td>
<td>IRQ6 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>IRQ5S</td>
<td>IRQ5 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>IRQ4S</td>
<td>IRQ4 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>IRQ3S</td>
<td>IRQ3 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>IRQ2S</td>
<td>IRQ2 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>IRQ1S</td>
<td>IRQ1 Enable Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**DMA1S (DMA 1 Interrupt Status):** When this bit is high, a DMA 1 controller interrupt is pending.

**DMA0S (DMA 0 Interrupt Status):** When this bit is high, a DMA 0 controller interrupt is pending.

**LM3S (Location Monitor 3 Interrupt Status):** When this bit is high, a location monitor 3 interrupt is pending.

**LM2S (Location Monitor 2 Interrupt Status):** When this bit is high, a location monitor 2 interrupt is pending.

**LM1S (Location Monitor 1 Interrupt Status):** When this bit is high, a location monitor 1 interrupt is pending.

**LM0S (Location Monitor 0 Interrupt Status):** When this bit is high, a location monitor 0 interrupt is pending.

**MB3S (Mail Box 3 Interrupt Status):** When this bit is high, a mail box 3 interrupt is pending.

**MB2S (Mail Box 2 Interrupt Status):** When this bit is high, a mail box 2 interrupt is pending.
8. Registers

**MB1S (Mail Box 1 Interrupt Status):** When this bit is high, a mail box 1 interrupt is pending.

**MB0S (Mail Box 0 Interrupt Status):** When this bit is high, a mail box 0 interrupt is pending.

**PERRS (PCI/X Bus Error Interrupt Status):** When this bit is high, a PCI/X bus error interrupt is pending.

**VERRS (VMEbus Error Interrupt Status):** When this bit is high, a VMEbus error interrupt is pending.

**VIES (VMEbus IRQ Edge Interrupt Status):** When this bit is high, a VMEbus IRQ edge interrupt is pending.

**IACKS (Interrupt Acknowledge Interrupt Status):** When this bit is high, an interrupt acknowledge interrupt is pending.

**SYSFLS (System Fail Interrupt Status):** When this bit is high, a VMEbus system fail interrupt is pending.

**ACFLS (AC Fail Interrupt Status):** When this bit is high, a VMEbus AC fail interrupt is pending.

**IRQ7S (IRQ7 Status):** When this bit is high, a VMEbus IRQ[7]I_ interrupt is pending.

**IRQ6S (IRQ6 Status):** When this bit is high, a VMEbus IRQ[6]I_ interrupt is pending.

**IRQ5S (IRQ5 Status):** When this bit is high, a VMEbus IRQ[5]I_ interrupt is pending.

**IRQ4S (IRQ4 Status):** When this bit is high, a VMEbus IRQ[4]I_ interrupt is pending.

**IRQ3S (IRQ3 Status):** When this bit is high, a VMEbus IRQ[3]I_ interrupt is pending.

**IRQ2S (IRQ2 Status):** When this bit is high, a VMEbus IRQ[2]I_ interrupt is pending.

**IRQ1S (IRQ1 Status):** When this bit is high, a VMEbus IRQ[1]I_ interrupt is pending.
## 8.4.73 Interrupt Clear Register

### Table 108: Interrupt Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>R</td>
<td>N/A 0x00</td>
</tr>
<tr>
<td>23:16</td>
<td>LM3C</td>
<td>Location Monitor 3 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>LM2C</td>
<td>Location Monitor 2 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>LM1C</td>
<td>Location Monitor 1 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td></td>
<td>LM0C</td>
<td>Location Monitor 0 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>MB3C</td>
<td>Mail Box 3 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MB2C</td>
<td>Mail Box 2 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>MB1C</td>
<td>Mail Box 1 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>MB0C</td>
<td>Mail Box 0 Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>R</td>
<td>N/A 0x00</td>
</tr>
<tr>
<td>13</td>
<td>PERRC</td>
<td>PCI/X Bus Error Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>VERRC</td>
<td>VMEbus Error Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>VIEC</td>
<td>VMEbus IRQ Edge Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>IACKC</td>
<td>Interrupt Acknowledge Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
</tbody>
</table>

## Interrupt Enable Clear Register
### Interrupt Enable Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>SYSFLC</td>
<td>System Fail Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>ACFLC</td>
<td>AC Fail Interrupt Clear</td>
<td>C</td>
<td>-</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**DMA1C (DMA 1 Interrupt Clear):** When this bit is set, the DMA 1 controller interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**DMA0C (DMA 0 Interrupt Clear):** When this bit is set, the DMA 0 controller interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**LM3C (Location Monitor 3 Interrupt Clear):** When this bit is set, the location monitor 3 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**LM2C (Location Monitor 2 Interrupt Clear):** When this bit is set, the location monitor 2 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**LM1C (Location Monitor 1 Interrupt Clear):** When this bit is set, the location monitor 1 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**LM0C (Location Monitor 0 Interrupt Clear):** When this bit is set, the location monitor 0 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**MB3C (Mail Box 3 Interrupt Clear):** When this bit is set, the mail box 3 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**MB2C (Mail Box 2 Interrupt Clear):** When this bit is set, the mail box 2 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**MB1C (Mail Box 1 Interrupt Clear):** When this bit is set, the mail box 1 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**MB0C (Mail Box 0 Interrupt Clear):** When this bit is set, the mail box 0 interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**PERRC (PCI/X Bus Error Interrupt Clear):** When this bit is set, the PCI/X bus error interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**VERRC (VMEbus Error Interrupt Clear):** When this bit is set, the VMEbus error interrupt is cleared. This bit always reads zero and writing a zero has no effect.
8. Registers

**VIEC (VMEbus IRQ Edge Interrupt Clear):** When this bit is set, the VMEbus IRQ edge interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**IACKC (Interrupt Acknowledge Interrupt Clear):** When this bit is set, the VMEbus interrupt acknowledge interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**SYSFLC (System Fail Interrupt Clear):** When this bit is set, the VMEbus system fail interrupt is cleared. This bit always reads zero and writing a zero has no effect.

**ACFLC (AC Fail Interrupt Clear):** When this bit is set, the AC fail interrupt is cleared. This bit always reads zero and writing a zero has no effect.
### 8.4.74 Interrupt Map 1 Register

#### Table 109: Interrupt Map 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DMA1M</td>
<td>DMA0M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>LM3M</td>
<td></td>
<td>LM2M</td>
<td></td>
<td>LM1M</td>
<td>LM0M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>MB3M</td>
<td>MB2M</td>
<td>MB1M</td>
<td>MB0M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Interrupt Map 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>19:18</td>
<td>DMA1M</td>
<td>DMA 1 Interrupt Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17:16</td>
<td>DMA0M</td>
<td>DMA 0 Interrupt Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>LM3M</td>
<td>Location Monitor 3 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>13:12</td>
<td>LM2M</td>
<td>Location Monitor 2 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11:10</td>
<td>LM1M</td>
<td>Location Monitor 1 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9:8</td>
<td>LM0M</td>
<td>Location Monitor 0 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7:6</td>
<td>MB3M</td>
<td>Mail Box 3 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5:4</td>
<td>MB2M</td>
<td>Mail Box 2 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3:2</td>
<td>MB1M</td>
<td>Mail Box 1 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1:0</td>
<td>MB0M</td>
<td>Mail Box 0 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**DMA1M (DMA 1 Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**DMA0M (DMA 0 Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.
**LM3M (Location Monitor 3 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**LM2M (Location Monitor 2 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**LM1M (Location Monitor 1 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**LM0M (Location Monitor 0 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**MB3M (Mail Box 3 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**MB2M (Mail Box 2 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**MB1M (Mail Box 1 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**MB0M (Mail Box 0 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.
8.4.75 Interrupt Map 2 Register

Table 110: Interrupt Map 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>19:18</td>
<td>DMA1M</td>
<td>DMA 1 Interrupt Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17:16</td>
<td>DMA0M</td>
<td>DMA 0 Interrupt Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>IRQ7M</td>
<td>IRQ7 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>13:12</td>
<td>IRQ6M</td>
<td>IRQ6 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11:10</td>
<td>IRQ5M</td>
<td>IRQ5Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9:8</td>
<td>IRQ4M</td>
<td>IRQ4 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7:6</td>
<td>IRQ3M</td>
<td>IRQ3 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5:4</td>
<td>IRQ2M</td>
<td>IRQ27 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3:2</td>
<td>IRQ1M</td>
<td>IRQ1 Map</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**PERRM (PCI/X Bus Error Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**VERRM (VMEbus Error Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.
8. Registers

**VIEM (VMEbus IRQ Edge Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IACKM (Interrupt Acknowledge Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**SYSFLM (System Fail Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**ACFLM (AC Fail Interrupt Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ7M (IRQ7 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ6M (IRQ6 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ5M (IRQ5 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ4M (IRQ4 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ3M (IRQ3 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ2M (IRQ2 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.

**IRQ1M (IRQ1 Map):** These bits indicate which INTx signal line the interrupt is routed to. The values 0 - 3 maps the interrupts to INTA_ - INTD_ respectively.
### 8.4.76 DMA Control (0-1) Registers

The DMA Control Register (DCTLx) provides the control fields for the DMA function.

#### Table 111: DMA Control (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ABT</td>
<td>PAU</td>
<td>DGO</td>
<td>Reserved</td>
</tr>
<tr>
<td>23:16</td>
<td>MOD</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>VBKS</td>
<td>Reserved</td>
<td>VBOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>PBKS</td>
<td>Reserved</td>
<td>PBOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### DMA Control (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>27</td>
<td>ABT</td>
<td>Abort</td>
<td>R/S</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>26</td>
<td>PAU</td>
<td>Pause</td>
<td>R/S</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>25</td>
<td>DGO</td>
<td>DMA Go</td>
<td>R/S</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>22:18</td>
<td>MOD</td>
<td>Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>VFAR</td>
<td>VME Flush on Aborted Read</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>PFAR</td>
<td>PCI/X Flush on Aborted Read</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>14:12</td>
<td>VBKS</td>
<td>VMEbus Block Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>10:8</td>
<td>VBOT</td>
<td>VMEbus Back-off Timer</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>6:4</td>
<td>PBKS</td>
<td>PCI/X Block Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

DMA Control (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>2:0</td>
<td>PBOT</td>
<td>PCI/X Back-off Timer</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**ABT (Abort):** Writing a one to this field aborts a DMA transaction. An abort is considered an unrecoverable operation to a DMA transaction, meaning that an aborted transaction may not be restarted. When issuing an abort, both the PCI/X and/or VMEbus masters are immediately stopped and all FIFO contents are invalidated. Once the abort has taken affect, the DSTA BSY bit is cleared and the DON and ERR bits is set. Reading this field always returns a zero.

**PAU (Pause):** Writing a one to this field pauses a DMA transaction. This bit is only applicable to Linked-List-Mode transactions. When pausing a DMA transaction, the DMA controller stops at the completion of the current linked-list transfer. If the pause took affect before the completion of a transaction, then the DTSA. The PAU field is set once the DMA Controller reaches the paused state. A paused transaction may be restarted by writing a one to the DGO field. Reading this field always returns a zero.

> Abort (ABT) has authority over Pause (PAU). If a commanded pause is followed by an commanded abort, then the DMA controller honors the commanded abort.

**DGO (DMA Go):** Writing a one to this field starts a DMA transaction. Reading this field always returns a zero.

**MOD (Mode):** This bit establishes the type of DMA transaction to be performed. If set, a Direct-Mode transaction is performed. A Direct-Mode transaction performs one transfer according to the contents of the DSAD, DSAT, DDAD, DDAT, and DCNT registers. If cleared, a Linked-List-Mode transaction is performed. A Linked-List-Mode transaction performs multiple transfers that are driven by a list of descriptors stored in PCI/X memory space. A Linked-List-Mode transaction obtains the first descriptor from the starting address placed within the DNLA register.

**VFAR (VME Flush on Aborted Read):** If this bit is set and a VMEbus cycle is terminated with an exception, any data remaining in the FIFO is transferred to the destination. If this bit is cleared and a VMEbus cycle is terminated with an exception, any data remaining in the DMA FIFO is discarded.
**PFAR (PCI/X Flush on Aborted Read):** If this bit is set and a PCI/X bus cycle is terminated with an exception, any data remaining in the FIFO is transferred to the destination. If this bit is cleared and a PCI/X bus cycle is terminated with an exception, any data remaining in the DMA FIFO is discarded.

**VBKS (VMEbus Block Size):** This field is used to control the VMEbus block size when the source is the VMEbus. The encoding of this field is shown in Table 112

*Table 112: DCTL BKS Encoding*

<table>
<thead>
<tr>
<th>VBKS</th>
<th>Transfer Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>32</td>
</tr>
<tr>
<td>001b</td>
<td>64</td>
</tr>
<tr>
<td>010b</td>
<td>128</td>
</tr>
<tr>
<td>011b</td>
<td>256</td>
</tr>
<tr>
<td>100b</td>
<td>512</td>
</tr>
<tr>
<td>101b</td>
<td>1024</td>
</tr>
<tr>
<td>110b</td>
<td>2048</td>
</tr>
<tr>
<td>111b</td>
<td>4096</td>
</tr>
</tbody>
</table>

**VBOT (VMEbus Back-off Timer):** The back-off timer determines how long the DMA controller waits before requesting the next block of data. This field controls the internal data flow between the DMA controller and the VME Master Module. The DMA does not attempt to read the next block until after the Back-off Timer has expired. To control the amount of time the VME Master is allowed to spend on the bus during DMA transfers please see the VME Master Control Register (Table 8.4.34 on page 205).
Table 113 shows the encoding for this field.

**Table 113: DCTL VBOT Encoding**

<table>
<thead>
<tr>
<th>VBOT</th>
<th>Back-off Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>0 µs</td>
</tr>
<tr>
<td>001b</td>
<td>1 µs</td>
</tr>
<tr>
<td>010b</td>
<td>2 µs</td>
</tr>
<tr>
<td>011b</td>
<td>4 µs</td>
</tr>
<tr>
<td>100b</td>
<td>8 µs</td>
</tr>
<tr>
<td>101b</td>
<td>16 µs</td>
</tr>
<tr>
<td>110b</td>
<td>32 µs</td>
</tr>
<tr>
<td>111b</td>
<td>64 µs</td>
</tr>
</tbody>
</table>

**PBKS (PCI/X Block Size):** This field is used to control the PCI/X bus block size when the source is the PCI/X bus. The encoding of this field is shown in Table 114.

**Table 114: DCTL PBKS Encoding**

<table>
<thead>
<tr>
<th>PBKS</th>
<th>Transfer Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bytes</td>
</tr>
<tr>
<td>000b</td>
<td>32</td>
</tr>
<tr>
<td>001b</td>
<td>64</td>
</tr>
<tr>
<td>010b</td>
<td>128</td>
</tr>
<tr>
<td>011b</td>
<td>256</td>
</tr>
<tr>
<td>100b</td>
<td>512</td>
</tr>
<tr>
<td>101b</td>
<td>1024</td>
</tr>
<tr>
<td>110b</td>
<td>2048</td>
</tr>
<tr>
<td>111b</td>
<td>4096</td>
</tr>
</tbody>
</table>

**PBOT (PCI/X Back-off Timer):** The back-off timer determines how long the DMA waits before requesting the next block of data. This field controls the internal data flow between the DMA controller and the PCI/X Master. The DMA does not attempt to read the next block until after the Back-off Timer has expired.
Table 115 shows the encoding for this field.

Table 115: DCTL PBOT Encoding

<table>
<thead>
<tr>
<th>PBOT</th>
<th>Back-off Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>0 µs</td>
</tr>
<tr>
<td>001b</td>
<td>1 µs</td>
</tr>
<tr>
<td>010b</td>
<td>2 µs</td>
</tr>
<tr>
<td>011b</td>
<td>4 µs</td>
</tr>
<tr>
<td>100b</td>
<td>8 µs</td>
</tr>
<tr>
<td>101b</td>
<td>16 µs</td>
</tr>
<tr>
<td>110b</td>
<td>32 µs</td>
</tr>
<tr>
<td>111b</td>
<td>64 µs</td>
</tr>
</tbody>
</table>
8. Registers

8.4.77 DMA Status (0-1) Registers

The DMA Status Register (DSTA) provides the status fields for the DMA function. The BSY field represents the current state of the DMA controller, and the remaining fields indicate completion status. When the DMA controller is starting a transaction (that is, the DGO field is set) the BSY field is asserted and all of the completion status fields are cleared. The BSY field remains asserted and the completion status fields remain cleared throughout the entire DMA transaction. Once the DMA Controller is finished, then the BSY field is cleared and only one of the completion status fields (DON, PAU, ABT, or ERR) is asserted. A functional interrupt is sent to the Exception module whenever the BSY field transitions to the deasserted state.

The completion status fields are prioritized from left to right, with the left most status field holding the highest priority. For example, if the DMA Controller incurs a simultaneous ERR error and an ABT, then the DSTA register only reflects the ERR completion status.

If the DMA Controller incurs multiple errors that are NOT simultaneously detected, then the DSTA register only reflects the status pertaining to the first occurring error. This is of particular importance to the PAU and ABT fields. If an error is detected before the pause or abort takes affect, then the DSTA register only reflects the status pertaining to the error.

Table 116: DMA Status (0-1) Register

<table>
<thead>
<tr>
<th>Register Name: DSTAx</th>
<th>Register Offset: DCSTA0: CRG + 0x504</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DCSTA1: CRG + 0x584</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>ERR</td>
<td>ABT</td>
<td>PAU</td>
<td>DON</td>
<td>BSY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>ERS</td>
<td>Reserved</td>
<td>ERT1</td>
<td>ERT0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DMA Status (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>28</td>
<td>ERR</td>
<td>Error</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>27</td>
<td>ABT</td>
<td>Abort</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>26</td>
<td>PAU</td>
<td>Pause</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

DMA Status (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>DON</td>
<td>Done</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>24</td>
<td>BSY</td>
<td>Busy</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23:21</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>ERRS</td>
<td>Error Source</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19:18</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>ERT1</td>
<td>Error Type</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>ERT0</td>
<td>Error Type</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**ERR (Error):** This read-only field is set if the DMA Controller receives an error signal from the PCI/X bus or VMEbus. Additional information is provided in the ERRS and ERT fields.

**ABT (Abort):** This read-only field is set if the DMA Controller has successfully completed a commanded abort. A successful command abort must meet the following criteria:

1. A write of a logic 1 to the ABT field.
2. The DMA Controller has not received any other errors (ERR) between the time the transaction was started and the time that the DMA Controller goes to the idle state.
3. The commanded abort took place before the DMA Controller was able to complete a transaction.

**PAU (Pause):** This read-only field is set if the DMA Controller has successfully completed a commanded pause. A successful command pause must meet the following criteria:

1. A write of a logic 1 to the PAU field.
2. The DMA Controller has not received any other errors (ERR) between the time the transaction was started and the time that the DMA Controller goes to the idle state.
3. The DMA Controller has not been issued a commanded abort.
4. The commanded pause took place before the DMA Controller was able to complete a transaction.
**DON (Done):** This read-only field is set if the DMA Controller has successfully completed a DMA transaction. A successful transaction must meet the following criteria:

1. The DMA Controller has not received any other errors (ERR) between the time the transaction was started and the time that the DMA Controller goes to the idle state.
2. If a commanded abort was issued, then it did not take effect before the transaction was completed.
3. If a commanded pause was issued, then it did not take effect before the transaction was completed.

**BSY (Busy):** This read-only field reflects the status of the DMA Controller. If set, the DMA Controller is currently processing a DMA transaction. If cleared, the DMA Controller has completed a previous transaction and is now idle.

**ERRS (Error Source):** When the ERR bit is set, this bit indicates the source of the error. When this bit is set, the PCI/X bus was the source of the error. When this bit is clear, the VMEbus was the source of the error.

**ERT (Error Type):** When the ERR bit is set, these bits indicate the type of error received.

**Table 117: DSTA ERT Encoding**

<table>
<thead>
<tr>
<th>ERS</th>
<th>ERT</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00b</td>
<td>Bus error: SCT, BLT, MBLT, 2eVME even data, 2eSST</td>
</tr>
<tr>
<td>0</td>
<td>01b</td>
<td>Bus error: 2eVME odd data</td>
</tr>
<tr>
<td>0</td>
<td>10b</td>
<td>Slave termination: 2eVME even data, 2eSST read</td>
</tr>
<tr>
<td>0</td>
<td>11b</td>
<td>Slave termination: 2eVME odd data, 2eSST read last word invalid</td>
</tr>
<tr>
<td>1</td>
<td>00b</td>
<td>PCI/X Bus Error</td>
</tr>
<tr>
<td>1</td>
<td>01b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>10b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
8.4.78 DMA Current Source Address Upper (0-1) Registers

This is a read-only register that contains the upper bits (63:32) of the current source address for a DMA transfer. If the source is VMEbus space, then this field represents a VMEbus address. If the source is PCI/X space, then this field represents a PCI/X address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

If VMEbus error occurs during a FIFO fill, with the VMEbus as the transfer source, this register represents the VMEbus address at which a read error occurred. If a PCI/X bus error occurs during a FIFO fill, with the PCI/X bus as the transfer source, this register represents the PCI/X address at which a read error occurred.

Table 118: DMA Current Source Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>31:0</th>
<th>DCSAUx</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCSAUx</td>
<td>DMA Current Source Address Upper</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.79 DMA Current Source Address Lower (0-1) Registers

This is a read-only register that contains the lower bits (31:0) of the current source address for a DMA transfer. If the source is VMEbus space, then this field represents a VMEbus address. If the source is PCI/X space, then this field represents a PCI/X address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

If a VMEbus error occurs during a FIFO fill, with the VMEbus as the transfer source, this register represents the VMEbus address at which a read error occurred. If a PCI/X bus error occurs during a FIFO fill, with the PCI/X bus as the transfer source, this register represents the PCI/X address at which a read error occurred.

Table 119: DMA Current Source Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCSALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DMA Current Source Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCSALx</td>
<td>DMA Current Source Address Lower</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.80 DMA Current Destination Address Upper (0-1) Registers

This is a read-only register that contains the upper bits (63:32) of the current destination address for a DMA transfer. If the destination is VMEbus space, then this field represents a VMEbus address. If the destination is PCI/X space, then this field represents a PCI/X address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

If a VMEbus error occurs during a FIFO empty, with the VMEbus as the transfer destination, this register represents the VMEbus address at which a read error occurred. If a PCI/X bus error occurs during a FIFO empty, with the PCI/X bus as the transfer destination, this register represents the PCI/X address at which a read error occurred.

Table 120: DMA Current Destination Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Register Name: DCDAUx</th>
<th>Reset Value: 0x00000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Offset:</td>
<td>DCDAU0: CRG + 0x510</td>
</tr>
<tr>
<td></td>
<td>DCDAU1: CRG + 0x590</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DMA Current Destination Upper (0-1) Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCDAUx</td>
<td>DMA Current Destination Address Upper</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.81 DMA Current Destination Address Lower (0-1) Registers

This is a read-only register that contains the lower bits (31:0) of the current destination address for a DMA transfer. If the destination is VMEbus space, then this field represents a VMEbus address. If the destination is PCI/X space, then this field represents a PCI/X address. Software can read this register after a DMA error to determine how far along a DMA transfer went before the error occurred.

If a VMEbus error occurs during a FIFO empty, with the VMEbus as the transfer destination, this register represents the VMEbus address at which a write error occurred. If a PCI/X bus error occurs during a FIFO empty, with the PCI/X bus as the transfer destination, this register represents the PCI/X address at which a write error occurred.

Table 121: DMA Current Destination Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** DMA Current Destination Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCDALx</td>
<td>DMA Current Destination Address Lower</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.82 DMA Current Link Address Upper (0-1) Registers

This is a read-only register that contains the upper bits (63:32) of the current Linked-List-Mode descriptor address for a DMA command. This always represents a PCI/X address. Software can read this register after a DMA error to determine which command in the linked list was being executed when the error occurred.

Table 122: DMA Current Link Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DMA Current Link Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCLAUx</td>
<td>DMA Current Link Address Upper</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.83 DMA Current Link Address Lower (0-1) Registers

This is a read-only register that contains the lower bits (31:6) of the current Linked-List-Mode descriptor address for a DMA command. This always represents a PCI/X address. Software can read this register after a DMA error to determine which command in the linked list was being executed when the error occurred.

Table 123: DMA Current Link Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>DCLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>DCLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>DCLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>DCLALx</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>DCLALx</td>
<td>DMA Current Link Address Lower</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.Registers

8.4.84 DMA Source Address Upper (0-1) Registers

This register contains the upper bits (63:32) of the source address for a DMA transfer. If the source is VMEbus space then this field represents a VMEbus address. If the source is PCI/X space then this field represents a PCI/X address.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the source address field of the current descriptor.

Table 124: DMA Source Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSAUx</td>
</tr>
</tbody>
</table>

DMA Source Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DSAUx</td>
<td>DMA Source Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.85 **DMA Source Address Lower (0-1) Registers**

This register contains the lower bits (31:0) of the source address for a DMA transfer. If the source is VMEbus space then this field represents a VMEbus address. If the source is PCI/X space then this field represents a PCI/X address.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the source address field of the current descriptor.

### Table 125: DMA Source Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Register Name: DCALx</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset:</th>
<th>DSAL0: CRG + 0x524</th>
<th>DSAL1: CRG + 0x5A4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DSALx</td>
</tr>
</tbody>
</table>

**DMA Source Address Lower (0-1) Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DSALx</td>
<td>DMA Source Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.86 DMA Destination Address Upper (0-1) Registers

This register contains the upper bits (63:32) of the destination address for a DMA transfer. If the destination is VMEbus space then this field represents a VMEbus address. If the destination is PCI/X space then this field represents a PCI/X address.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the destination address field of the current descriptor.

Table 126: DMA Destination Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DDAU</td>
</tr>
</tbody>
</table>

**DMA Destination Address Upper (0-1) Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DDAUx</td>
<td>DMA Destination Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.87 **DMA Destination Address Lower (0-1) Registers**

This register contains the lower bits (31:0) of the destination address for a DMA transfer. If the destination is VMEbus space then this field represents a VMEbus address. If the destination is PCI/X space then this field represents a PCI/X address.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the destination address field of the current descriptor.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DDAL</td>
<td>DMA Destination Address Lower</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**Table 127: DMA Destination Address Lower (0-1) Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Offset</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DDAL</td>
<td>CRG + 0x52C</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRG + 0x5AC</td>
<td></td>
</tr>
</tbody>
</table>
### 8.4.88 DMA Source Attribute (0-1) Registers

The DMA Source Attribute Register (DSAT) contains the source attributes for a DMA transfer. Not all fields are used for all transfer types. Fields that do not pertain to a particular transfer type are ignored.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the source attribute field of the current descriptor.

**Table 128: DMA Source Attribute (0-1) Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>29</td>
<td>TYP1</td>
<td>Type</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>28</td>
<td>TYP0</td>
<td>Type</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>27:26</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>25</td>
<td>PSZ</td>
<td>Pattern Size</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>24</td>
<td>NIN</td>
<td>No Increment</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>23:13</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>SSTM1</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>SSTM0</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>TM2</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9</td>
<td>TM1</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

DMA Source Attribute (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>TM0</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>DBW1</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>DBW0</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>SUP</td>
<td>VMEbus Supervisory Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>PGM</td>
<td>VMEbus Program Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>AMODE3</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>AMODE2</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>AMODE1</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>AMODE0</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**TYP (Type):** This field indicates the type of source to be used for a DMA transfer. Different fields within the DSAT register are used depending on the type of source selected. Table 129 shows the different source types and the associated fields within the DSAT register that apply.

**Table 129: DSAT TYP Encoding**

<table>
<thead>
<tr>
<th>TYP</th>
<th>DMA Source</th>
<th>Applicable Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NIN</td>
</tr>
<tr>
<td>00b</td>
<td>PCI/X bus</td>
<td>X</td>
</tr>
<tr>
<td>01b</td>
<td>VMEbus</td>
<td>X</td>
</tr>
<tr>
<td>1x</td>
<td>Data Pattern</td>
<td>X</td>
</tr>
</tbody>
</table>

**PSZ (Pattern Size):** If set, the data size used during Data Pattern transfers is bytes (8-bit). If cleared, the data size is words (32-bit). This field only applies to the generation of the data patterns used for a transfer. It does not specify how the patterns are actually placed into the destination space. (that is, selecting a byte pattern size does not result in a stream of single-beat bus cycles.)

**NIN (No Increment):** If set, source increment is disabled during a DMA transfer. If a VMEbus source is selected then the source address is not incremented. If the source is a data pattern, then the data pattern is not incremented. If cleared, the source is incremented.
SSTM (2eSST Mode): This field defines the 2eSST Transfer Rate.

Table 130: 2eSST Transfer Rate

<table>
<thead>
<tr>
<th>SSTM</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>160 MB/s</td>
</tr>
<tr>
<td>01b</td>
<td>267 MB/s</td>
</tr>
<tr>
<td>10b</td>
<td>320 MB/s</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

TM (Transfer Mode): This field defines the VMEbus transfer mode.

Table 131: VMEbus Transfer Mode

<table>
<thead>
<tr>
<th>TM</th>
<th>Transfer Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>SCT</td>
</tr>
<tr>
<td>001b</td>
<td>BLT</td>
</tr>
<tr>
<td>010b</td>
<td>MBLT</td>
</tr>
<tr>
<td>011b</td>
<td>2eVME</td>
</tr>
<tr>
<td>100b</td>
<td>2eSST</td>
</tr>
<tr>
<td>101b</td>
<td>2eSST Broadcast</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

DBW (VMEbus Data Bus Width): These bits define the maximum data bus width for VMEbus transfers initiated by the DMA controller. These bits apply to SCT and BLT transfers. MBLT, 2eVME and 2eSST transfers are always 64-bit.

Table 132: VMEbus Data Bus Width

<table>
<thead>
<tr>
<th>DWB</th>
<th>Data Bus Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>16 bit</td>
</tr>
<tr>
<td>01b</td>
<td>32 bit</td>
</tr>
<tr>
<td>10b</td>
<td>Reserved</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
SUP (VMEbus Supervisory Mode): When this bit is set the AM code indicates Supervisory Access. When this bit is cleared the AM code indicates Non-Privileged Access.

PGM (VMEbus Program Mode): When this bit is set the AM code indicates Program Access. When this bit is cleared the AM code indicates Data Access.

AMODE (Address Mode): This field defines the VMEbus Address mode.

Table 133: VMEbus Address Mode

<table>
<thead>
<tr>
<th>AMODE</th>
<th>Address Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000b</td>
<td>A16</td>
</tr>
<tr>
<td>0001b</td>
<td>A24</td>
</tr>
<tr>
<td>0010b</td>
<td>A32</td>
</tr>
<tr>
<td>0011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100b</td>
<td>A64</td>
</tr>
<tr>
<td>0101b</td>
<td>CR/CSR</td>
</tr>
<tr>
<td>0110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0111b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1000b</td>
<td>User1 (AM 0100xxb)</td>
</tr>
<tr>
<td>1001b</td>
<td>User2 (AM 0101xxb)</td>
</tr>
<tr>
<td>1010b</td>
<td>User3 (AM 0110xxb)</td>
</tr>
<tr>
<td>1011b</td>
<td>User4 (AM 0111xxb)</td>
</tr>
<tr>
<td>1100b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

When the User1-User4 modes are used, the AM[1] bit is defined by the SUP bit and the AM[0] bit is defined by the PGM bit.
8.4.89 DMA Destination Attribute (0-1) Registers

The DMA Destination Attribute Register (DDAT) contains the destination attributes for a DMA transfer. Not all fields are used for all transfer types. Fields that do not pertain to a particular transfer type are ignored.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the destination attribute field of the current descriptor.

Table 134: DMA Destination Attribute (0-1) Register

<table>
<thead>
<tr>
<th>Register Name: DDATx</th>
<th>Register Offset: DDATx: CRG + 0x534</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DDATx: CRG + 0x5B4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td>TYP</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>SST1</td>
<td>SST0</td>
<td>TM2</td>
<td>TM1</td>
<td>TM0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>DBW1</td>
<td>DBW0</td>
<td>SUP</td>
<td>PGM</td>
<td>AMODE3</td>
<td>AMODE2</td>
<td>AMODE1</td>
<td>AMODE0</td>
</tr>
</tbody>
</table>

Inbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>28</td>
<td>TYP</td>
<td>Type</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>27:13</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>12</td>
<td>SST1</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>11</td>
<td>SST0</td>
<td>2eSST Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>TM2</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>9</td>
<td>TM1</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>TM0</td>
<td>Transfer Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>DBW1</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>DBW0</td>
<td>VMEbus Data Bus Width</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>SUP</td>
<td>VMEbus Supervisory Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
Inbound Translation Attribute (0-7) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>PGM</td>
<td>VMEbus Program Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>AMODE3</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>AMODE2</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>AMODE1</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>0</td>
<td>AMODE0</td>
<td>Address Mode</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**TYP (Type):** This field indicates the type of destination to be used for a DMA transfer. Different fields within the DDAT register are used depending on the type of destination selected. Table 135 shows the different destination types and the associated fields within the DDAT register that apply.

**Table 135: DDAT TYP Encoding**

<table>
<thead>
<tr>
<th>TYP</th>
<th>DMA Destination</th>
<th>Applicable Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SSTM  TM  DBW  SUP PGM  AMODE</td>
</tr>
<tr>
<td>0</td>
<td>PCI/X bus</td>
<td>X     X   X     X     X   X</td>
</tr>
<tr>
<td>1</td>
<td>VMEbus</td>
<td>X     X   X     X     X   X</td>
</tr>
</tbody>
</table>

**NIN (No Increment):** If set, destination increment is disabled during a DMA transfer. If a VMEbus destination is selected then the destination address is not incremented. If cleared, the destination is incremented.

**SSTM (2eSST Mode):** This field defines the 2eSST Transfer Rate.

**Table 136: 2eSST Transfer Rate**

<table>
<thead>
<tr>
<th>SSTM</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>160 MB/s</td>
</tr>
<tr>
<td>01b</td>
<td>267 MB/s</td>
</tr>
<tr>
<td>10b</td>
<td>320 MB/s</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
8. Registers

**TM (Transfer Mode):** This field defines the VMEbus transfer mode.

### Table 137: VMEbus Transfer Mode

<table>
<thead>
<tr>
<th>TM</th>
<th>Transfer Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000b</td>
<td>SCT</td>
</tr>
<tr>
<td>001b</td>
<td>BLT</td>
</tr>
<tr>
<td>010b</td>
<td>MBLT</td>
</tr>
<tr>
<td>011b</td>
<td>2eVME</td>
</tr>
<tr>
<td>100b</td>
<td>2eSST</td>
</tr>
<tr>
<td>101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**DBW (VMEbus Data Bus Width):** These bits define the maximum data bus width for VMEbus transfers initiated by the DMA controller.

### Table 138: VMEbus Data Bus Width

<table>
<thead>
<tr>
<th>DWB</th>
<th>Data Bus Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td>16 bit</td>
</tr>
<tr>
<td>01b</td>
<td>32 bit</td>
</tr>
<tr>
<td>10b</td>
<td>Reserved</td>
</tr>
<tr>
<td>11b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**SUP (VMEbus Supervisory Mode):** When this bit is set the AM code indicates Supervisory Access. When this bit is cleared the AM code indicates Non-Privileged Access.

**PGM (VMEbus Program Mode):** When this bit is set the AM code indicates Program Access. When this bit is cleared the AM code indicates Data Access.
AMODE (Address Mode): This field defines the VMEbus Address mode.

Table 139: VMEbus Address Mode

<table>
<thead>
<tr>
<th>AMODE</th>
<th>Address Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000b</td>
<td>A16</td>
</tr>
<tr>
<td>0001b</td>
<td>A24</td>
</tr>
<tr>
<td>0010b</td>
<td>A32</td>
</tr>
<tr>
<td>0011b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0100b</td>
<td>A64</td>
</tr>
<tr>
<td>0101b</td>
<td>CR/CSR</td>
</tr>
<tr>
<td>0110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>0111b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1000b</td>
<td>User1 (AM 0100xxb)</td>
</tr>
<tr>
<td>1001b</td>
<td>User2 (AM 0101xxb)</td>
</tr>
<tr>
<td>1010b</td>
<td>User3 (AM0110xxb)</td>
</tr>
<tr>
<td>1011b</td>
<td>User4 (AM 0111xxb)</td>
</tr>
<tr>
<td>1100b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1101b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1110b</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111b</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

When the User1-User4 modes are used, the AM[1] bit is defined by the SUP bit and the AM[0] bit is defined by the PGM bit.
8.4.90 DMA Next Link Address Upper (0-1) Registers

These are the upper address bits (63:32) of the next descriptor when using Linked-List-Mode. This is a PCI/X address.

This register is not used when performing Direct-Mode transactions. When starting a Linked-List-Mode transaction, software programs this register with the address of the first Linked-List-Mode descriptor. When continuing a Linked-List-Mode transaction, the register is automatically loaded from the next link address field of the current descriptor.

Table 140: DMA Next Link Address Upper (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DNLAU</td>
<td>DMA Next Link Address Upper</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.91 DMA Next Link Address Lower (0-1) Registers

Table 141: DMA Next Link Address Lower (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>DNLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>DNLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>DNLALx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>DNLALx</td>
<td>Reserved</td>
<td>LLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DNLAL (DMA Next Link Address Lower):** These are the Lower address bits (31:3) of the next descriptor when using Linked-List-Mode. This is a PCI/X address.

This register is not used when performing Direct-Mode transactions. When starting a Linked-List-Mode transaction, software programs this register with the address of the first Linked-List-Mode descriptor. When continuing a Linked-List-Mode transaction, the register is automatically loaded from the next link address field of the current descriptor.

**LLA (Last Link Address):** If set, the current descriptor is the last descriptor of a Linked-List transaction. If cleared, the current descriptor is not the last descriptor.
8.4.92 DMA Count (0-1) Registers

This register contains the byte count for a DMA transfer. A zero value indicates that zero bytes are transferred. As the DMA transfer progresses, the DCNT register is decrement. When a DMA transfer completes with out errors, the final DCNT value is zero.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the count field of the current descriptor.

Table 142: DMA Count (0-1) Register

<table>
<thead>
<tr>
<th>Register Name: DCNTx</th>
<th>Register Offset: DCNTx: CRG + 0x540</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>DCNTx: CRG + 0x5C0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DCNTx</td>
</tr>
</tbody>
</table>

DMA Count (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DCNT</td>
<td>DMA Count Register</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.93 DMA Destination Broadcast Select (0-1) Registers

This register contains the 2eSST broadcast select bits. Each bit corresponds to one of the 21 possible slaves. The 2eSST master broadcasts this field during address phase three. Register bit 11 corresponds to VMEbus address line A21 and register bit 31 corresponds to VMEbus address line A1.

Software programs this register when performing Direct-Mode transactions. When performing Linked-List-Mode transactions, this register is automatically loaded from the broadcast select field of the current descriptor.

Table 143: DMA Destination Broadcast Select (0-1) Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>20:0</td>
<td>DDBS</td>
<td>DMA Destination Broadcast Select Register</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
</tbody>
</table>
8.4.94 GCSR Register Group

This section defines the Global Control and Status Registers (GCSR). These registers are accessible from the PCI/X bus or VMEbus. The VMEbus address and address space is programmable. RMW cycles from the VMEbus are not guaranteed indivisible.

8.4.95 Vendor ID / Device ID Registers

Table 144: Vendor ID / Device ID Register

<table>
<thead>
<tr>
<th>Register Name: DEVI/VENI</th>
<th>Reset Value: 0x014810E3</th>
<th>Register Offset: GCSR + 0x00 - CRG + 0x600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td>DEVI</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>DEVI</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>VENI</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>VENI</td>
<td></td>
</tr>
</tbody>
</table>

DEVI (Device ID): This register is a read only register that uniquely identifies this particular device. The Tsi148 always returns 0x0148.

VENI (Vendor ID): This register is a read-only register that identifies the manufacturer of the device. This identifier is allocated by the PCI/X Special Interest Group to ensure uniqueness. 0x10E3 has been assigned to Motorola and is hard wired as a read-only value.
8. Registers

8.4.96 Control and Status Register

Table 145: Control and Status Register

<table>
<thead>
<tr>
<th>Register Name: GCTRL</th>
<th>Register Offset: GCSR + 0x04 - CRG + 0x604</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>LRST</td>
<td>SFAILEN</td>
<td>BDFAILS</td>
<td>SCONS</td>
<td>MEN</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>LMI3S</td>
<td>LMI2S</td>
<td>LMI1S</td>
<td>LMI0S</td>
<td>MBI3S</td>
<td>MBI2S</td>
<td>MBI1S</td>
<td>MBI0S</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>GAP</td>
<td>GA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>Reserved</td>
<td>REVID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Control and Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LRST</td>
<td>Local Reset</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>30</td>
<td>SFAILEN</td>
<td>System Fail Enable</td>
<td>R/W</td>
<td>P/S</td>
<td>0xx</td>
</tr>
<tr>
<td>29</td>
<td>BDFAILS</td>
<td>Board Fail Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x01</td>
</tr>
<tr>
<td>28</td>
<td>SCONS</td>
<td>System Controller Status</td>
<td>R</td>
<td>P</td>
<td>0xx</td>
</tr>
<tr>
<td>27</td>
<td>MEN</td>
<td>Module Enable</td>
<td>R/W</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>26:24</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>23</td>
<td>LMI3S</td>
<td>Location Monitor Interrupt 3 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>22</td>
<td>LMI2S</td>
<td>Location Monitor Interrupt 2 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>21</td>
<td>LMI1S</td>
<td>Location Monitor Interrupt 1 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>LMI0S</td>
<td>Location Monitor Interrupt 0 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>MBI3S</td>
<td>Mail Box Interrupt 3 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>MBI2S</td>
<td>Mail Box Interrupt 2 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>17</td>
<td>MBI1S</td>
<td>Mail Box Interrupt 1 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>MBI0S</td>
<td>Mail Box Interrupt 0 Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>13</td>
<td>GAP</td>
<td>Geographic Address Parity</td>
<td>R</td>
<td>-</td>
<td>0xx</td>
</tr>
</tbody>
</table>
8. Registers

Control and Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:8</td>
<td>GA</td>
<td>Geographic Address</td>
<td>R</td>
<td>-</td>
<td>0xxx</td>
</tr>
<tr>
<td>7:0</td>
<td>REVID</td>
<td>Revision ID</td>
<td>R</td>
<td>-</td>
<td>0x01</td>
</tr>
</tbody>
</table>

**LRST (Local Reset):** When this bit is set, the LRSTO_ signal is asserted. When this bit is cleared, the LRSTO_ signal is not asserted. When this bit is set and cleared, the Tsi148 ensures the minimum pulse width for local reset is met.

**SFAILEN (System Fail Enable):** When this bit is set and the BDFAIL_ signal is asserted, the SFAILO signal line is asserted. When this bit is cleared, the SFAILO signal line is not asserted. The initial value of this bit is a configuration option.

The system fail enable bit is also accessible in the CR/CSR.

**BDFAILS (Board Fail Status):** Reading a one indicates the BDFAIL_ signal is asserted. Reading a zero indicates the BDFAIL_ signal is not asserted.

**SCONS (System Controller Status):** Reading a one indicates, the VMEbus system controller is enabled.

**MEN (Module Enable):** This is a read/write bit that can be used to indicate a ready condition. The READY is cleared by reset. Software may set this bit, after the board is initialized, to indicate the board is ready.

**LMI3S (Location Monitor Interrupt 3 Status):** When set, a location monitor 3 interrupt is pending.

**LMI2S (Location Monitor Interrupt 2 Status):** When set, a location monitor 2 interrupt is pending.

**LMI1S (Location Monitor Interrupt 1 Status):** When set, a location monitor 1 interrupt is pending.

**LMI0S (Location Monitor Interrupt 0 Status):** When set, a location monitor 0 interrupt is pending.

**MBI3S (Mail Box Interrupt 3 Status):** When set, a mail box 3 interrupt is pending.

**MBI2S (Mail Box Interrupt 2 Status):** When set, a mail box 2 interrupt is pending.

**MBI1S (Mail Box Interrupt 1 Status):** When set, a mail box 1 interrupt is pending.
8. Registers

MBIOS (Mail Box Interrupt 0 Status): When set, a mail box 0 interrupt is pending.

GAP (Geographic Address Parity): This bit is the parity bit for the Geographic Address. This bit is inverted from the VMEbus GAP* signal.

GA (Geographic Address): These bits represent the Geographic Address of the board. These bits are inverted from the VMEbus GA[4:0] signals.

REVID (Revision ID): This register identifies the Tsi148’s revision level.
8.4.97 Semaphore Registers (0-3)

Table 146: Semaphore Register (0-3)

<table>
<thead>
<tr>
<th>Register Offset: GCSR + 0x08 - CRG + 0x0608</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>31:24 SEMA0</td>
</tr>
<tr>
<td>23:16 SEMA1</td>
</tr>
<tr>
<td>15:8  SEMA2</td>
</tr>
<tr>
<td>7:0  SEMA3</td>
</tr>
</tbody>
</table>

Semaphore Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>SEMA0</td>
<td>Semaphore 0</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>23:16</td>
<td>SEMA1</td>
<td>Semaphore 1</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>15:8</td>
<td>SEMA2</td>
<td>Semaphore 2</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>SEMA3</td>
<td>Semaphore 3</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>

SEMA0 (Semaphore 0): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA1 (Semaphore 1): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA2 (Semaphore 2): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA3 (Semaphore 3): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.
8.4.98 Semaphore Registers (4-7)

Table 147: Semaphore Registers (0-4)

<table>
<thead>
<tr>
<th>Register Name: SEMAR1</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: GCSR + 0x0C - CRG + 0x60C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Semaphore Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>SEMA4</td>
<td>Semaphore 4</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>23:16</td>
<td>SEMA5</td>
<td>Semaphore 5</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>15:8</td>
<td>SEMA6</td>
<td>Semaphore 6</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>7:0</td>
<td>SEMA7</td>
<td>Semaphore 7</td>
<td>R/W</td>
<td>P/S</td>
<td>0x00</td>
</tr>
</tbody>
</table>

SEMA4 (Semaphore 4): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA5 (Semaphore 5): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA6 (Semaphore 6): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.

SEMA7 (Semaphore 7): Semaphore registers behave in the following way. A semaphore can only be written if the most significant bit in the register is 0 and the most significant bit of the write data is a one or the most significant bit of the write data is a 0.
8. Registers

8.4.99 Mail Box Registers (0-3)

The mail box register can be used to pass information between the local processor and other VME boards. When the least significant byte is written, an interrupt is sent to the interrupter. If the interrupt is enabled, an INTx signal is generated.

Table 148: Mail Box Registers (0-3)

<table>
<thead>
<tr>
<th>Register Name: MBOXx</th>
<th>Register Offset: MBOX0: GCSR + 0x10 - CRG + 0x610</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MBOX1: GCSR + 0x14 - CRG + 0x614</td>
</tr>
<tr>
<td></td>
<td>MBOX2: GCSR + 0x18 - CRG + 0x618</td>
</tr>
<tr>
<td></td>
<td>MBOX3: GCSR + 0x1C - CRG + 0x61C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>MBOX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mail Box Registers (0-3)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>MBOX</td>
<td>Mail Box</td>
<td>R/W P/S</td>
<td></td>
<td>0x00</td>
</tr>
</tbody>
</table>
8. Registers

8.4.100 CR/CSR Register Group Description

The Tsi148 implements a sub-set of the CR/CSR register set. This section describes the CSR registers that are included.

8.4.101 CR/CSR Bit Clear Register

Table 149: CR/CSR Bit Clear Register

<table>
<thead>
<tr>
<th>Register Name: CSRBCR</th>
<th>Register Offset: CR/CSR + 0x7FFF4 - CRG + 0xFF4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Value: 0x00000000</td>
<td>Bits 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>31:24</td>
<td>Reserved</td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
</tr>
<tr>
<td>7:0</td>
<td>LRSTC SFAILC BDFAILS MENC BERRSC Reserved</td>
</tr>
</tbody>
</table>

CR/CSR Bit Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>LRSTC</td>
<td>Local Reset Clear</td>
<td>C/R</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>SFAILC</td>
<td>System Fail Enable Clear</td>
<td>C/R</td>
<td>P/S</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>BDFAILS</td>
<td>Board Fail Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x01</td>
</tr>
<tr>
<td>4</td>
<td>MENC</td>
<td>Module Enable Clear</td>
<td>C/R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>BERRSC</td>
<td>Bus Error Status Clear</td>
<td>C/R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

LRSTC (Local Reset Clear): Writing a one to this bit clears the LRST bit. Reading a one indicates the LRST bit is set. Reading a zero indicates the LRST bit is cleared.

SFAILC (System Fail Enable Clear): Writing a one to this bit disables the SFAILO driver. Reading a one indicates the SFAILO driver is enabled. Reading a zero indicates the SFAILO driver is disabled. The initial value of this bit is a configuration option. The system fail enable bit is also accessible from the GCSR.
**BDFAILS (Board Fail Status):** Reading a one indicates the BDFAIL signal is asserted. Reading a zero indicates the BDFAIL signal is not asserted.

**MENC (Module Enable Clear):** Writing a one to this bit clears the module enable bit. Reading a one indicates the module enable bit is set. Reading a zero indicates the module enable bit is not set. The module enable bit can be used as a ready bit. The module enable bit is also accessible from the GCSR.

**BERRSC (Bus Error Status Clear):** This bit is set when the Tsi148 asserts the VMEbus BERR* signal. Writing a one to this bit clears the BERR status bit. Reading a one indicates that the Tsi148 has asserted the BERR* signal or that the BERRSS bit has been set. Reading a zero indicates that the Tsi148 has not asserted the BERR* signal and BERRS bit has not been set.
8. Registers

8.4.102 CR/CSR Bit Set Register

Table 150: CR/CSR Bit Set Register

<table>
<thead>
<tr>
<th>Register Name: CSRBSR</th>
<th>Reset Value: 0x</th>
<th>Register Offset: CR/CSR + 0x7FFF8 - CRG + 0xFF8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>31:24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>LRSTS</td>
<td>SFAILS</td>
</tr>
</tbody>
</table>

CR/CSR Bit Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7</td>
<td>LRSTS</td>
<td>Local Reset Set</td>
<td>S/R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>SFAILS</td>
<td>System Fail Enable Set</td>
<td>S/R</td>
<td>P/S</td>
<td>0xxx</td>
</tr>
<tr>
<td>5</td>
<td>BDFAILS</td>
<td>Board Fail Status</td>
<td>R</td>
<td>P/S/L</td>
<td>0x01</td>
</tr>
<tr>
<td>4</td>
<td>MENS</td>
<td>Module Enable Set</td>
<td>S/R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>3</td>
<td>BERRSS</td>
<td>Bus Error Status Set</td>
<td>S/R</td>
<td>P/S/L</td>
<td>0x00</td>
</tr>
<tr>
<td>2:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

**LRSTS (Local Reset Set):** Writing a one to this bit sets the LRST signal. This asserts the LRSTO_ signal and hold the board in reset until a one is written to the LRSTC bit. This bit should only be set from VMEbus. It should not be set from the CRG. Reading a one indicates the LRST bit is set. Reading a zero indicates the LRST bit is cleared.

**SFAILS (System Fail Enable Set):** Writing a one to this bit enables SFAILO driver. Reading a one indicates the SFAILO driver is enabled. Reading a zero indicates the SFAILO driver is disabled. The initial value of this bit is a configuration option. The system fail enable bit is also accessible from the GCSR.

**BDFAILS (Board Fail Status):** Reading a one indicates the BDFAIL_ signal is asserted. Reading a zero indicates the BDFAIL_ signal is not asserted.
MENS (Module Enable Set): Writing a one to this bit sets the module enable bit. Reading a one indicates the module enable bit is set. Reading a zero indicates the module enable bit is not set. The module enable bit can be used as a ready bit. The module enable bit is also accessible from the GCSR.

BERRSS (Bus Error Status Set): This bit is set when the Tsi148 asserts the VMEbus BERR* signal. Writing a one to this bit sets the BERR status bit. Reading a one indicates that the Tsi148 has asserted the BERR* signal or that the BERR status bit has been set. Reading a zero indicates that the Tsi148 has not asserted the BERR* signal and BERR status bit has not been set.
8. Registers

8.4.103 CR/CSR Base Address Register

The CBAR is used to select one of the 31 available CR/CSR regions (0x00 is reserved for use in Auto Slot ID). The CBAR values are in the range of 0x01 to 0x1F. Bits 7 to 3 of the CBAR are compared with VMEbus address bits 23 to 19. The initial value of CBAR is determined by the hardware configuration.

Table 151: CR/CSR Base Address Register

<table>
<thead>
<tr>
<th>Register Name: CBAR</th>
<th>Reset Value: 0x00000000</th>
<th>Register Offset: CR/CSR + 0x7FFFC - CRG + 0xFFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>
| 31:24              |    |    |    |    |    |    |    |    | Reserved
| 23:16              |    |    |    |    |    |    |    |    | Reserved
| 15:8               |    |    |    |    |    |    |    |    | Reserved
| 7:0                |    |    |    |    |    |    |    |    | CBAR

CR/CSR Bit Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Function</th>
<th>PCFS Space Type</th>
<th>Reset By</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
<tr>
<td>7:3</td>
<td>CBAR</td>
<td>CR/CSR Base Address</td>
<td>R/W</td>
<td>P/S</td>
<td>0xx</td>
</tr>
<tr>
<td>2:0</td>
<td>Reserved</td>
<td>N/A</td>
<td>R</td>
<td>N/A</td>
<td>0x00</td>
</tr>
</tbody>
</table>
A. Package Information

This appendix discusses Tsi148’s packaging (mechanical) features. The following topic is discussed:

- “Package Characteristics” on page 321

A.1 Package Characteristics

Tsi148’s package characteristics are summarized in the following table. Figure 36 illustrates the Bottom and Side views of the Tsi148 package. Figure 37 presents the Top view of the device.

Table 152: Package Characteristics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Type</td>
<td>456 PBGA</td>
</tr>
<tr>
<td>Package Body Size</td>
<td>27 x 27 mm</td>
</tr>
<tr>
<td>JEDEC Specification</td>
<td>MO-151 Solid State Product Outline</td>
</tr>
</tbody>
</table>
Figure 36: 456-Pin PBGA Package Diagram — Bottom and Side Views
A. Package Information

Figure 37: 456-Pin PBGA Package Diagram — Top View

A.1.1 Package Notes

1. Package conforms to JEDEC specification MO-151 solid state product outline.

2. Area reserved for ejector pin mark (typical four places). Corner shape to be chamfered or rounded within this area.

3. A1 ball location corresponds to gate release. Gate shape is for reference.
A. Package Information
## Bit Index

**Numerics**
- 133C 181
- 2eBS 190
- 2eOT 219
- 2eSST 235
- 2eSSTB 235
- 2eSSTM0 191, 235
- 2eSSTM1 191, 235
- 2eSSTM2 191, 235
- 2eST 219
- 2eVME 236
- 64BCU 254
- 64BCL 255
- 64D 181

**A**
- A64DS 201
- ABT 279, 284
- ACFAILS 209
- ACFLC 273
- ACFLEN 262
- ACFLEO 267
- ACFLS 270
- ACKD 215
- ADMODE0 192
- ADMODE1 192
- ADMODE2 192
- ADMODE3 192
- AKFC 215
- AM 219
- AMODE0 298, 302
- AMODE1 298, 302
- AMODE2 298, 302
- AMODE3 298, 302
- AS0 236, 241, 245, 252
- AS1 236, 241, 245, 252
- AS2 236, 241, 245, 252
- ATOEN 205

**B**
- BASEL 172
- BASEU 173
- BBFC 215
- BCFC 215
- BCLAS 169
- BDFAIL 209
- BDFAILS 209, 310, 316, 318
- BERR 219
- BERRSC 316
- BERRSS 318
- BGFC 215
- BID 205
- BIP 258
- BIPS 258
- BLT 236
- BN 181
- BPCT 257
- BPGT 256
- BRFC 215
- BSY 285

**C**
- CAPID 179
- CAPL 166
- CAPP 175
- CBAR 320
- CBAU 243, 244
- CBEAx 226
- CCTM 211
- CLSZ 170
- CNTS 258
- COMMx 227
- CPURST 209
- CROL 248
- CROU 247

**D**
- DATA 236, 241, 245, 252
- DBW0 192, 298, 301
- DBW1 192, 298, 301
- DC 181
- DCDALx 290
- DCDAUx 289
- DCLALx 292
- DCLAUx 291
- DCNT 307
- DCSALx 288
- DCSAUx 287
- DDAL 296
- DDAUx 295
- DDBS 308
- DEVI 165, 309
- DEVSELS 212
- DGO 279
- DHB 201
- DLT 205
- DMA0C 272
- DMA0EN 261
- DMA0EO 266
- DMA0M 275, 277
<table>
<thead>
<tr>
<th>Bit Index</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA0S</td>
<td>269</td>
</tr>
<tr>
<td>DMA1C</td>
<td>272</td>
</tr>
<tr>
<td>DMA1EN</td>
<td>261</td>
</tr>
<tr>
<td>DMA1EO</td>
<td>266</td>
</tr>
<tr>
<td>DMA1M</td>
<td>275, 277</td>
</tr>
<tr>
<td>DMA1S</td>
<td>269</td>
</tr>
<tr>
<td>DMCRS</td>
<td>181</td>
</tr>
<tr>
<td>DMMRC</td>
<td>181</td>
</tr>
<tr>
<td>DMOST</td>
<td>181</td>
</tr>
<tr>
<td>DN</td>
<td>181</td>
</tr>
<tr>
<td>DNLAx</td>
<td>306</td>
</tr>
<tr>
<td>DNLAU</td>
<td>305</td>
</tr>
<tr>
<td>DON</td>
<td>285</td>
</tr>
<tr>
<td>DPE</td>
<td>166, 226</td>
</tr>
<tr>
<td>DPED</td>
<td>166, 226</td>
</tr>
<tr>
<td>DPERE</td>
<td>179</td>
</tr>
<tr>
<td>DRQ</td>
<td>211</td>
</tr>
<tr>
<td>DS0</td>
<td>219</td>
</tr>
<tr>
<td>DS1</td>
<td>219</td>
</tr>
<tr>
<td>DSALx</td>
<td>294</td>
</tr>
<tr>
<td>DSAUX</td>
<td>293</td>
</tr>
<tr>
<td>DTT</td>
<td>226</td>
</tr>
<tr>
<td>DTTE</td>
<td>211</td>
</tr>
<tr>
<td>DTTT</td>
<td>211</td>
</tr>
<tr>
<td>DWB</td>
<td>201</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>EDGIS</td>
<td>258</td>
</tr>
<tr>
<td>EDPAL</td>
<td>223</td>
</tr>
<tr>
<td>EDPAU</td>
<td>222</td>
</tr>
<tr>
<td>EDPCL</td>
<td>226</td>
</tr>
<tr>
<td>EDPOF</td>
<td>226</td>
</tr>
<tr>
<td>EDPST</td>
<td>226</td>
</tr>
<tr>
<td>EDPXA</td>
<td>224</td>
</tr>
<tr>
<td>EDPXS</td>
<td>225</td>
</tr>
<tr>
<td>EN</td>
<td>191, 235, 241, 245, 249, 252</td>
</tr>
<tr>
<td>ENDAL</td>
<td>232</td>
</tr>
<tr>
<td>ENDL</td>
<td>187</td>
</tr>
<tr>
<td>ENDU</td>
<td>186, 231</td>
</tr>
<tr>
<td>ERO</td>
<td>179</td>
</tr>
<tr>
<td>ERR</td>
<td>284</td>
</tr>
<tr>
<td>ERRS</td>
<td>285</td>
</tr>
<tr>
<td>ERT0</td>
<td>285</td>
</tr>
<tr>
<td>ERT1</td>
<td>285</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FAST</td>
<td>166</td>
</tr>
<tr>
<td>FN</td>
<td>181</td>
</tr>
<tr>
<td>FRAMES</td>
<td>212</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>209, 311</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>HEAD</td>
<td>170</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>IACK</td>
<td>219</td>
</tr>
<tr>
<td>IACKC</td>
<td>272</td>
</tr>
<tr>
<td>IACKEN</td>
<td>261</td>
</tr>
<tr>
<td>IACKEO</td>
<td>266</td>
</tr>
<tr>
<td>IACKS</td>
<td>269</td>
</tr>
<tr>
<td>INTL</td>
<td>176</td>
</tr>
<tr>
<td>INTP</td>
<td>176</td>
</tr>
<tr>
<td>IO/MEM</td>
<td>172</td>
</tr>
<tr>
<td>IOSP</td>
<td>167</td>
</tr>
<tr>
<td>IRDYS</td>
<td>212</td>
</tr>
<tr>
<td>IRQ1EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ1EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ1M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ1S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ2EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ2EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ2F</td>
<td>258</td>
</tr>
<tr>
<td>IRQ2M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ2S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ3EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ3EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ3M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ3S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ4EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ4EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ4M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ4S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ5EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ5EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ5M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ5S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ6EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ6EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ6M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ6S</td>
<td>270</td>
</tr>
<tr>
<td>IRQ7EN</td>
<td>262</td>
</tr>
<tr>
<td>IRQ7EO</td>
<td>267</td>
</tr>
<tr>
<td>IRQ7M</td>
<td>277</td>
</tr>
<tr>
<td>IRQ7S</td>
<td>270</td>
</tr>
<tr>
<td>IRQC</td>
<td>258</td>
</tr>
<tr>
<td>IRQIF</td>
<td>258</td>
</tr>
<tr>
<td>IRQL</td>
<td>258</td>
</tr>
<tr>
<td>IRQLS</td>
<td>258</td>
</tr>
<tr>
<td>IRQS</td>
<td>258</td>
</tr>
</tbody>
</table>
**Bit Index**

**L**
- LLA 306
- LM0C 272
- LM0EN 261
- LM0EO 266
- LM0M 275
- LM0S 269
- LM1C 272
- LM1EN 261
- LM1EO 266
- LM1M 275
- LM1S 269
- LM2C 272
- LM2EN 261
- LM2EO 266
- LM2M 275
- LM2S 269
- LM3C 272
- LM3EN 261
- LM3EO 266
- LM3M 275
- LM3S 269
- LMBAL 251
- LMBAU 250
- LM1S 310
- LM1EO 310
- LM3S 310
- LRESET 205
- LRST 310
- LRSTC 316
- LRSTS 318
- LWORD 219

**M**
- M66ENS 212
- MB0C 272
- MB0EN 261
- MB0EO 266
- MB0M 275
- MB0S 269
- MB1C 272
- MB1EN 261
- MB1EO 266
- MB1M 275
- MB1S 269
- MB2C 272
- MB2EN 261
- MB2EO 266
- MB2M 275
- MB2S 269
- MB3C 272

MB3EN 261
MB3EO 266
MB3M 275
MB3S 269
MBI0S 310
MBI1S 310
MBI2S 310
MBI3S 310
MBLT 236
MBOX 315
MEMSP 167
MEN 310
MENC 316
MENS 318
MLAT 170
MMRBC 179
MNGN 176
MOD 279
MOST 179
MRC 211, 226
MRCE 212
MRCT 211
MRPFD 191
MSTR 167
MTYP0 172
MTYP1 172
MXLA 176

**N**
- NCAPP 179
- NELBB 205
- NIN 297
- NPRIV 236, 241, 245, 252

**O**
- OFFL 189, 234, 240
- OFFU 188, 233

**P**
- P66M 166
- PAU 279, 284
- PBKS 279
- PBOT 280
- PERR 167
- PERRC 272
- PERREN 261
- PERREO 266
- PERRS 269
- PFAR 279
- PFS0 191
- PFS1 191
- PGM 192, 236, 241, 245, 252, 298, 302
Bit Index

PIC 169
PRE 172
PSZ 297
PURSTS 209

R
RCVMA 166
RCVTA 166
REQ64S 212
REVI 169
REVID 311
RMA 226
RMWAL 197
RMWAU 196
RMWC 199
RMWEN 198, 201
RMWS 200
ROBIN 205
RSCEM 181
RTA 226

S
SBH 211
SCD 181, 226
SCEM 226
SCLAS 169
SCONS 209, 310
SETIM0 166
SETIM1 166
SEMA0 313
SEMA1 313
SEMA2 313
SEMA3 313
SEMA4 314
SEMA5 314
SEMA6 314
SEMA7 314
SERR 167
SFAILAI 205
SFAILC 316
SFAILEN 310
SFAILS 318
SIGSE 166
SIGTA 166
SRESET 205
SRT 226
SRTE 211
SRT0 211
SRTTS 211
SSTM0 297, 301
SSTM1 297, 301
STAL 185, 230

STAU 184, 229
STFC 215
STID 258
STOPS 212
SUBI 174
SUBV 174
SUP 192, 298, 301
SUPR 236, 241, 245, 252
SYSFLC 273
SYSFLEN 262
SYSFLEO 267
SYSFLS 209, 270

T
TH 235
TM0 192, 298, 301
TM1 192, 297, 301
TM2 191, 297, 301
TRSDYS 212
TYP 301
TYP0 297
TYP1 297

U
USC 181, 226

V
VBKS 279
VBOT 279
VEAL 218
VEAU 217
VENI 165, 309
VEOF 219
VERRC 272
VERREN 261
VERREO 266
VERRS 269
VES 219
VESCL 219
VFAIR 202
VFAIR 202
VFAR 279
VFS0 235
VFS1 235
VIACK 195
VIEC 272
VIEEN 261
VIEEO 266
VIES 269
VREL 202
VREQ 202
VREQ 202
VS 201
VSA 201
VTOFF 201
VTON 202

W
WRITE 219

X
XAM 219
Index

Numerics
2eSST Protocol 35
2eVME Protocol 35
64BCL register 255
64BCU register 254

B
BPCTR register 257
BPGTR register 256

C
CAPP register 175
CBAL register 244
CBAR register 320
CBAU register 243
CLAS/REVI register 169
clocks 121
contact information 5
CRAT register 249
CRGAT register 245
CROL register 248
CROU register 247
CSRBCR register 316
CSRBSR register 318
customer support information 5

D
DCALx register 294
DCDALx register 290
DCDAUx register 289
DCLALx register 292
DCLAUx register 291
DCNTx register 307
DCSALx register 288
DCSAUx register 287
DCTLx register 279
DDALx register 296
DDATx register 301
DDAUx register 295
DDBSx register 308
DEV/I/ENI register 165, 309
direct memory access (DMA) 101
DMA (direct memory access) 101
DMA Controller 102
Architecture 102
Buffers 102
Direction of Data Movement 105
Operating Modes 103
Direct mode 103
Linked-list mode 103
Transaction Termination 119
Commanded Abort 119
Commanded Stop 119
Detected Error Abort 120
Transfer Completion 119
DMA Controllers Overview 45
DNLALx register 306
DNLAUx register 305
document conventions 24
signals 24
documentation feedback 5
DSATx register 297
DSAUX register 293
DSTAx register 284

E
EDPAL register 223
EDPAT register 226
EDPAU register 222
EDPXA register 224
EDPXs register 225
e-mail 5
endian mapping 24

F
functional overview 29, 49, 73

G
GBAL register 240
GBAU register 239
Index

GCSRAT register 241
GCTRL register 310

H
HEAD/MLAT/CLSZ register 170

I
IITOFL register 234
INTC register 272
INTEN register 261
INTEO register 266
interrupts 135
INTM1 register 275
INTM2 register 277
INTS register 269
ITATx register 235
ITEALx register 232
ITEAUx register 231
ITOFUx register 233
ITSALx register 230
ITSAUx register 229

J
JTAG 47, 139
Instructions 140

L
Linkage Module 42
LMAT register 252
LMBAL register 251
LMBAU register 250
Local Interrupter 136

M
mailing address 5
MBAR register 172
MBARU register 173
MBOXx register 315
mechanical information 321
MXLA/MNGN/INTP/INTL register 176

P
packaging information 321
PCI Bus Exception Handling 86
Master 86
Target 86
PCI Master 85
Bandwidth Control 85
Buffers 85
Commands 85
PCI Target 74
Buffers 74
Commands 76
Read Transaction 77
Transaction Mapping 75
Write Transaction 82
PCI/X Bus Exception Handling 99
Master 99
Target 100
PCI/X Interface Overview 40
PCI-X Master 98
Bandwidth Control 98
Buffers 98
PCI-X Target 88
Buffers 88
Commands 90
Read Transaction 91
Transaction Mapping 89
Transactions 90
Write Transaction 95
PCIXCAP register 179
PCIXSTAT register 181
PCSR register 211
Power-up Options
PCI/X 127
VMEbus 129
power-up options 121
product benefits 32
product features 32

R
registers 143
Registers Overview 43
related documentation 27
resets 121
RMWAL register 197
RMWAU register 196
RMWC register 199
RMWEN register 198
RMWS register 200

S
SEMAR0 register 313
Index

SEMAR1 register 314
STAT/CMMMD register 166
SUBI/SUBV register 174

T
technical support 5
typical applications 33

V
VCTRL register 205
VEAL register 218
VEAT register 219
VEAU register 217
VIACKx register 195
VICR register 258
VMCTRL register 201
VME Master 61
  Bandwidth Control 64
  Buffers 62
  Read-Modify Write (RMW) Cycles 62
  Release Conditions 65
VME Master Interface Overview 37
VME Slave 50
  Buffers 50
  Read Transaction 54
  Read-Modify Write (RMW) Cycles 60
  Terminations 60
  Transaction Mapping 51
  Transactions 54
  Write Transaction 58
VME Slave Interface Overview 36
VMEbus
  Interrupter and Interrupt Handler 46
VMEbus Interface 50
VMEbus Interrupt Handler 137
VMEbus Interrupter 136
VMEbus System Controller
  Arbiter 38, 71
  Configuration 39, 72
  Global VMEbus Timer 39, 72
  SYSRESET Driver 39, 71
  System Clock Driver 39, 72
VMEFL register 215
VSTAT register 209