

## **SCV64™ User Manual**



<http://www.tundra.com>

The information in this document is subject to change without notice and should not be construed as a commitment by Tundra Semiconductor Corporation. While reasonable precautions have been taken, Tundra Semiconductor Corporation assumes no responsibility for any errors that may appear in this document.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Tundra Semiconductor Corporation.

Tundra® products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Tundra product could create a situation where personal injury or death may occur. Should Buyer purchase or use Tundra products for any such unintended or unauthorized application, Buyer shall indemnify and hold Tundra and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Tundra was negligent regarding the design or manufacture of the part.

The acceptance of this document will be construed as an acceptance of the foregoing conditions.

SCV64™ User Manual

Copyright 2001, Tundra Semiconductor Corporation  
All rights reserved.

Document: 8091078\_MA001\_01

Printed in Canada

Tundra and Tundra logo are registered trademarks of Tundra Semiconductor Corporation. SCV64 is a trademark of Tundra Semiconductor Corporation. BI-Mode is a registered trademark of DY-4 Systems, Inc.

# Table of Contents

<b>1</b>	<b>General Information .....</b>	<b>1-1</b>
1.1	Introduction .....	1-1
1.2	Product Overview .....	1-1
1.2.1	Flexibility and Features .....	1-1
1.3	Using This Document .....	1-4
1.4	Conventions .....	1-5
1.4.1	Signals .....	1-5
1.4.2	Symbols .....	1-5
1.4.3	Mathematical Notation .....	1-5
<b>2</b>	<b>Functional Description.....</b>	<b>2-1</b>
2.1	Introduction .....	2-1
2.1.1	Organization of the Functional Description .....	2-1
2.1.2	Functional Overview .....	2-2
2.1.2.1	Data Path .....	2-2
2.1.2.2	VMEbus Interface .....	2-5
2.1.2.3	Local Bus Interface .....	2-6
2.2	VMEbus Requester .....	2-8
2.2.1	Function.....	2-8
2.2.2	Bus Request Modes .....	2-9
2.2.2.1	Fair and Demand Modes .....	2-9
2.2.2.2	VMEbus Request Levels .....	2-9
2.2.3	Bus Release Modes .....	2-10
2.2.3.1	Bus Clear Enabling .....	2-10
2.2.3.2	Release On Request and Release When Done .....	2-10
2.2.3.3	Ownership Timer .....	2-11
2.2.4	Other Bus Release Mechanisms .....	2-11
2.2.4.1	Local Memory Interrupt.....	2-11
2.2.4.2	BI-Mode .....	2-12

	2.2.4.3	Local and System Reset .....	2-12
2.3		Interrupter .....	2-13
	2.3.1	VMEbus Interrupts .....	2-13
	2.3.1.1	Interrupt Generation.....	2-13
	2.3.1.2	BI-mode Effects .....	2-14
	2.3.1.3	Reset Effects .....	2-14
	2.3.2	Local Bus Interrupts.....	2-15
2.4		Interrupt Handler .....	2-18
	2.4.1	Interrupt Enabling and Status .....	2-20
	2.4.1.1	Local Interrupt Level Mapping .....	2-21
	2.4.2	Interrupt Acknowledge Cycles .....	2-22
	2.4.2.1	Auto-Vectored Interrupts .....	2-25
	2.4.2.2	Vectored Interrupts .....	2-26
	2.4.2.3	BI-Mode Effects .....	2-29
	2.4.2.4	Reset Effects .....	2-29
2.5		System Controller Functions .....	2-30
	2.5.1	Syscon Determination.....	2-30
	2.5.2	IACK Daisy Chain Driver .....	2-31
	2.5.3	VMEbus Arbiter .....	2-31
	2.5.3.1	Arbitration Modes.....	2-31
	2.5.3.2	Arbitration Time-out.....	2-33
	2.5.3.3	Reset Effects .....	2-33
	2.5.4	Bus Timer .....	2-34
	2.5.5	System Clock Driver.....	2-34
	2.5.6	External Inputs .....	2-34
	2.5.6.1	External Status .....	2-34
	2.5.6.2	Off-Board Reset Input .....	2-35
	2.5.7	Reset Effects on Syscon Functions.....	2-35
2.6		Data Path.....	2-36
	2.6.1	SCV64 as VME Slave.....	2-39
	2.6.1.1	Coupled Mode.....	2-39

	2.6.1.2	Decoupled Mode .....	2-39
	2.6.2	SCV64 as VME Master.....	2-41
	2.6.2.1	Coupled Mode .....	2-41
	2.6.2.2	Decoupled Mode .....	2-42
	2.6.2.3	DMA Transfers .....	2-43
2.7		Memory Mapping .....	2-45
	2.7.1	CPU Memory Map.....	2-45
	2.7.2	VME Slave Memory Map.....	2-49
	2.7.2.1	Automatic Base Address Programming .....	2-51
	2.7.2.2	Access Protection .....	2-52
2.8		VMEbus Interface .....	2-53
	2.8.1	SCV64 as VME Master.....	2-53
	2.8.1.1	Address Translation .....	2-53
	2.8.1.2	Byte Lane Translation .....	2-54
	2.8.1.3	VMEbus Mastership.....	2-56
	2.8.1.4	RMW Cycles.....	2-57
	2.8.1.5	Termination of a Master Cycle with RETRY* .....	2-58
	2.8.2	SCV64 as VME Slave.....	2-58
	2.8.2.1	Address Translation .....	2-58
	2.8.2.2	Byte Lane Translation .....	2-59
	2.8.2.3	Local Bus Mastership.....	2-60
	2.8.3	DMA Transfers .....	2-61
	2.8.4	Master/Slave Deadlock Resolution .....	2-61
	2.8.5	Location Monitor Access .....	2-62
	2.8.6	Bus Busy Glitch .....	2-63
	2.8.7	BI-Mode Effects.....	2-63
	2.8.8	Bus Error Handling .....	2-64
2.9		Local Bus Interface .....	2-66
	2.9.1	Local Bus Arbitration.....	2-66
	2.9.1.1	Local Arbiter Bypassed.....	2-66
	2.9.1.2	Local Arbiter Active .....	2-67

2.9.2	Local Cycles – Overview .....	2-73
2.9.2.1	Cycle Initiation .....	2-73
2.9.2.2	Data Transfer .....	2-74
2.9.2.3	Cycle Termination Signals.....	2-76
2.9.2.4	Bus Error Handling .....	2-77
2.9.3	SCV64 as Local Slave .....	2-78
2.9.4	SCV64 as Local Master .....	2-83
2.9.5	Register Access .....	2-87
2.9.6	Burst Cycles .....	2-89
2.9.6.1	Burst Reads .....	2-89
2.9.6.2	Burst Writes .....	2-90
2.9.6.3	Local Address Incrementation .....	2-96
2.9.7	Read-Modify-Write Cycles.....	2-97
2.9.8	Master/Slave Deadlock Resolution.....	2-98
2.9.8.1	Deadlock Resolution in Decoupled Mode .....	2-98
2.9.9	Location Monitor Access .....	2-99
2.9.10	Reflected Cycles .....	2-99
2.9.11	VSBbus Access.....	2-100
2.9.12	BI-Mode.....	2-100
2.10	Location Monitor and LMFIFO.....	2-101
2.11	DMA Controller.....	2-102
2.11.1	DMA Initialization.....	2-102
2.11.2	Addressing and Data Transfer Modes .....	2-104
2.11.3	Data Transfer Counts .....	2-107
2.11.4	FILL Option.....	2-107
2.11.5	No Release Option .....	2-108
2.11.6	DMA Completion and Error Checking.....	2-108
2.12	Resets, Clocks and Timers.....	2-110
2.12.1	Resets .....	2-110
2.12.1.1	Local Reset .....	2-110
2.12.1.2	System Reset.....	2-111

2.12.1.3	Power-up Reset .....	2-111
2.12.1.4	Reset Effects on SCV64 Internal Resources .....	2-112
2.12.2	Clocks.....	2-113
2.12.2.1	Clocks Required .....	2-113
2.12.2.2	Clocks Generated .....	2-114
2.12.3	Timers .....	2-115
2.12.3.1	Local Bus Timer.....	2-115
2.12.3.2	Tick Timer.....	2-115
2.12.3.3	Watchdog Timer.....	2-116
2.12.3.4	VMEbus Timers .....	2-116
2.13	Power-up Modes .....	2-117
2.13.1	Local IACK Cycle Decoding .....	2-117
2.13.2	Auto SYSCON select.....	2-118
2.13.3	Local Arbiter Bypass.....	2-118
2.13.4	Automatic Base Address Programming .....	2-118
2.14	BI-Mode.....	2-119
2.15	Auto-ID .....	2-121
2.15.1	The Auto-ID Cycle.....	2-121
2.15.2	Auto-ID Self Test.....	2-123
2.16	Internal Delay Line Calibration .....	2-124
2.17	Test and Diagnostic Modes.....	2-126
2.17.1	Decoupled Write Diagnostics .....	2-126
2.17.2	Loopback Diagnostics .....	2-126
2.17.2.1	Loopback Mode .....	2-126
2.17.2.2	DMA Loopback .....	2-128
2.17.2.3	Interrupt Loopback.....	2-128
2.17.2.4	JTAG Support .....	2-128
<b>3</b>	<b>Description of SCV64 Signals .....</b>	<b>3-1</b>
3.1	VMEbus Signals .....	3-1
3.2	Local Signals .....	3-5

<b>4</b>	<b>Signals and DC Characteristics .....</b>	<b>4-1</b>
4.1	Terminology .....	4-1
4.2	DC Characteristics .....	4-2
4.3	Capacitive Loading .....	4-10
4.4	Pin Configuration.....	4-11
<b>App-A</b>	<b>Registers.....</b>	<b>App A-1</b>
A.1	Control and Status Registers.....	App A-1
<b>App-B</b>	<b>SCV64 Timing Characteristics .....</b>	<b>App B-1</b>
B.1	Timing Parameters.....	App B-1
B.2	Capacitive Loading .....	App B-13
B.3	Timing Diagrams .....	App B-14
<b>App-C</b>	<b>Performance .....</b>	<b>App C-1</b>
C.1	Coupled Cycles .....	App C-1
C.1.1	SCV64 as VME Master .....	App C-2
C.1.2	SCV64 as VME Slave.....	App C-5
C.2	Decoupled Cycles .....	App C-7
C.2.1	SCV64 as VME Master .....	App C-7
C.2.2	SCV64 as VME Slave.....	App C-7
C.3	Daisy Chains .....	App C-8
C.4	Arbiter Functions .....	App C-8
C.5	Summary .....	App C-9
<b>App-D</b>	<b>VMEbus-Local Bus Cycle Mapping.....</b>	<b>App D-1</b>
D.1	Cycle Translation.....	App D-1
D.2	Address Space Mapping .....	App D-12
<b>App-E</b>	<b>Applications.....</b>	<b>App E-1</b>
E.1	VMEbus Interface.....	App E-1
E.1.1	Buffered Signals .....	App E-1
E.1.2	Layout Issues .....	App E-1



	E.1.3	Decoupling VDD and VSS .....	App E-3
	E.1.4	BGxIN*[3:0] Signals .....	App E-3
	E.1.5	RETRY*/VRMC Pin .....	App E-6
	E.1.6	BI-Mode and IRQ1* .....	App E-7
E.2		Local Bus Interface for Slave Only Applications .....	App E-8
	E.2.1	Initialization .....	App E-8
	E.2.2	VMEbus Programming .....	App E-8
E.3		Local Bus Interface for MC68030 Application .....	App E-11
E.4		Local Bus Interface for MC68040 Application .....	App E-12
	E.4.1	Design Philosophy .....	App E-12
	E.4.2	Design Overview.....	App E-13
	E.4.3	Bus Adapter Operation.....	App E-16
	E.4.4	Cycle Translation State Machine .....	App E-17
	E.4.5	Dynamic Bus Sizing.....	App E-20
	E.4.6	Transfer Attributes .....	App E-21
	E.4.7	Interrupts .....	App E-24
<b>App-F</b>		<b>Initialization.....</b>	<b>App F-1</b>
F.1		Hardware Configuration .....	App F-1
	F.1.1	Power-Up Modes .....	App F-1
	F.1.2	Test Mode Pins.....	App F-2
	F.1.3	BI-Mode .....	App F-2
	F.1.4	JTAG .....	App F-2
	F.1.5	Clocks.....	App F-2
	F.1.6	Resets .....	App F-3
	F.1.7	RETRY*/VRMC.....	App F-3
F.2		Software .....	App F-4
	F.2.1	Delay Line Initialization .....	App F-5
	F.2.2	Byte Lane Mapping and Cycle Conversion .....	App F-6
	F.2.3	Slave Address Programming.....	App F-6
	F.2.4	Memory Map Overlay Initialization .....	App F-7
	F.2.5	BI-Mode .....	App F-7

F.2.6	RETRY*/VRMC Configuration .....	App F-7
F.2.7	Other Local Options.....	App F-8
F.2.8	Local Bursts .....	App F-8
F.2.9	Local Arbiter.....	App F-8
F.2.10	Interrupts .....	App F-8
F.2.11	Local Timer Functions .....	App F-9
F.2.12	SYSFAIL* .....	App F-9
F.2.13	VME Requester.....	App F-9
F.2.14	VME Arbiter .....	App F-9
<b>App-G</b>	<b>Reliability prediction .....</b>	<b>App G-1</b>
G.1	Device Description .....	App G-1
G.1.1	Physical characteristics .....	App G-1
G.1.2	Thermal characteristics .....	App G-1
G.1.3	Latch-up current.....	App G-2
G.2	Parameters.....	App G-2
<b>App-H</b>	<b>Environmental and Operating Parameters.....</b>	<b>App H-1</b>
<b>App-I</b>	<b>Revision History .....</b>	<b>App I-1</b>
<b>App-J</b>	<b>Mechanical and Ordering Information .....</b>	<b>App J-1</b>
J.1	Mechanical Information.....	App J-1
J.2	Ordering Information.....	App J-3
<b>Index</b>	<b>.....</b>	<b>Index-1</b>

# List of Figures

Figure 1.1 :	Coupled versus Decoupled Cycles .....	1-2
Figure 2.1 :	Functional Block Diagram.....	2-3
Figure 2.2 :	Interrupt Handler Block Diagram .....	2-19
Figure 2.3 :	Connections Resulting from Changing Local IACK Decoding .....	2-24
Figure 2.4 :	Local Interrupt Cycle - Auto-Vectored.....	2-26
Figure 2.5 :	Timing for Local Interrupts .....	2-27
Figure 2.6 :	Vectored Interrupt Cycle - with Local and VME Interrupter .....	2-28
Figure 2.7 :	Coupled Cycles on the VMEbus.....	2-37
Figure 2.8 :	Decoupled Cycles on the VMEbus .....	2-38
Figure 2.9 :	Sample CPU Memory Map as Determined by Local Address Decoder.....	2-46
Figure 2.10 :	SCV64 VME Access Overlay for the CPU Memory Map .....	2-47
Figure 2.11 :	Sample CPU Memory Map with SCV64 VME and VSB Space Overlay .....	2-48
Figure 2.12 :	Sample VME Slave Memory Map (Accessed as A32 and A24)....	2-51
Figure 2.13 :	Local to VME Byte Lane Translation with BUSSIZ=1 .....	2-55
Figure 2.14 :	Local to VME Byte Lane Translation with BUSSIZ=0 .....	2-56
Figure 2.15 :	Local Bus Arbitration with Local Arbiter Bypass .....	2-67
Figure 2.16 :	Local Bus Arbiter State Machine.....	2-68
Figure 2.17 :	Local Arbitration for Internal Requester .....	2-70
Figure 2.18 :	Local Arbitration for External Device - With Acknowledgment ...	2-72
Figure 2.19 :	Local Arbitration for External Device - Without Acknowledgment .....	2-72
Figure 2.20 :	Example of Longword Transfer to a 16-Bit Port .....	2-76
Figure 2.21 :	Flowchart for a Local Cycle with SCV64 as Local Slave .....	2-79
Figure 2.22 :	Decoupled Write Cycle – SCV64 as Local Slave.....	2-82
Figure 2.23 :	Flowchart for a Local Cycle with SCV64 as Local Master .....	2-84
Figure 2.24 :	Write Cycle – SCV64 as Local Master .....	2-85

Figure 2.25 :	Flowchart for a Burst Read Cycle - SCV64 as Local Master.....	2-91
Figure 2.26 :	Burst Read Cycle .....	2-92
Figure 2.27 :	Flowchart for a Burst Write Cycle - SCV64 as Local Master.....	2-94
Figure 2.28 :	Burst Write Cycle .....	2-95
Figure 2.29 :	Local interface for maximum burst length of 4 longwords .....	2-96
Figure 2.30 :	Procedural Steps in Normal DMA Reads and Writes. ....	2-103
Figure 2.31 :	Sample Power-up Reset Circuit.....	2-112
Figure 2.32 :	Connections with Different Local IACK Decoding .....	2-117
Figure 2.33 :	Timing for Auto-ID Cycle.....	2-122
Figure 2.34 :	Functional Block Diagram for Delay Lines .....	2-125
Figure 4.1 :	Pin Configuration for 299-Pin CPGA Package .....	4-11
Figure 4.2 :	Pin Configuration for 304-Pin PQFP Package .....	4-12
Figure B.1 :	Clocks and Timers .....	App B-14
Figure B.2 :	Reset Timing.....	App B-15
Figure B.3 :	Local Interrupts - Generation .....	App B-16
Figure B.4 :	Local Interrupts-Acknowledgment.....	App B-17
Figure B.5 :	Register Access.....	App B-18
Figure B.6 :	Register Access Effects .....	App B-19
Figure B.7 :	Decoupled Write Cycle - SCV64 as Local Master.....	App B-20
Figure B.8 :	Decoupled Write Cycle - SCV64 as Local Slave .....	App B-21
Figure B.9 :	Coupled Cycle - SCV64 as Local Master.....	App B-22
Figure B.10 :	Coupled Cycle - SCV64 as Local Slave .....	App B-23
Figure B.11 :	Burst Write Cycle - SCV64 as Local Master .....	App B-24
Figure B.12 :	Burst Read Cycle - SCV64 as Local Master .....	App B-25
Figure B.13 :	Slave Image/VSB/Location Monitor Access.....	App B-26
Figure B.14a :	Local Requester – Arbiter Bypassed .....	App B-27
Figure B.14b :	Local Requester/Arbiter: Request by SCV64.....	App B-27
Figure B.14c :	Local Requester/Arbiter: Request by an External Device – with Acknowledge.....	App B-28
Figure B.14d :	Local Requester/Arbiter: Request by an External Device – without Acknowledge.....	App B-28
Figure B.15a :	Local Bus Error Termination - Write or Read Cycle	

	with BERRCHK bit=0 (SCV64 as Local Master) .....	App B-29
Figure B.15b :	Local Bus Error Termination - Read Cycle with BERRCHK bit=1 (SCV64 as Local Master) .....	App B-29
Figure B.15c :	Deadlock Retry Termination (SCV64 as Local Slave).....	App B-29
Figure B.16 :	Daisy Chain Driver Timing .....	App B-30
Figure B.17a :	VMEbus Requester - Normal Release .....	App B-31
Figure B.17b :	VMEbus Requester - Release with BCLR* .....	App B-31
Figure B.18 :	VMEbus Arbiter Timing.....	App B-32
Figure B.19 :	VMEbus Interrupter Timing .....	App B-33
Figure B.20 :	VMEbus Interrupt Handler .....	App B-34
Figure B.21 :	Decoupled Write Cycle - SCV64 as VME Master .....	App B-35
Figure B.22 :	Decoupled Write Cycle - SCV64 as VME Slave .....	App B-36
Figure B.23 :	Coupled Cycle - SCV64 as VME Master .....	App B-37
Figure B.24 :	Coupled Cycle - SCV64 as VME Slave .....	App B-38
Figure B.25 :	BLT/MBLT Writes - SCV64 as VME Master.....	App B-39
Figure B.26 :	BLT/MBLT Writes - SCV64 as VME Slave.....	App B-40
Figure B.27 :	BLT/MBLT Read Cycle - SCV64 as VME Master.....	App B-41
Figure B.28 :	BLT Read Cycle - SCV64 as VME Slave .....	App B-42
Figure B.29 :	MBLT Read Cycle - SCV64 as VME Slave.....	App B-43
Figure C.1 :	SCV64 as VME master - RWD mode .....	App C-3
Figure C.2 :	SCV64 as VME master - ROR mode .....	App C-4
Figure C.3 :	Coupled Cycle - SCV64 as VME Slave (local arbiter active)..	App C-6
Figure C.4 :	Coupled Cycle-SCV64 as VME Slave (local arbiter bypassed)	App C-6
Figure D.1 :	Single Longword Aligned Byte Transfer with SWAP = 0 .....	App D-2
Figure D.2 :	Single Longword Aligned Byte Transfer with SWAP = 1 .....	App D-2
Figure D.3 :	Tri-byte Transfer with BUSSIZ = 1 .....	App D-4
Figure D.4 :	Tri-byte Transfer with BUSSIZ = 0.....	App D-4
Figure E.1 :	Orientation of the SCV64 (CPGA and PQFP) to Minimize Distance to the Connector .....	App E-2
Figure E.2 :	SCV64 Interface to VMEbus .....	App E-5
Figure E.3 :	Bus Grant Daisy Chain Usage in System .....	App E-6
Figure E.4 :	Implementation of VRMC and RETRY* .....	App E-7

Figure E.5 :	Connections for Slave Only Design .....	App E-10
Figure E.6 :	Connections for 68030 Design .....	App E-11
Figure E.7 :	Shared Local Bus Structure .....	App E-13
Figure E.8 :	Hierarchical Local Bus Structure .....	App E-13
Figure E.9 :	MC68040 to SCV64 Bus Adapter Block Diagram .....	App E-15
Figure E.10 :	MC68040 Bus Transfer Control .....	App E-16
Figure E.11 :	SCV64 Bus Transfer Control .....	App E-17
Figure E.12 :	Cycle Translation Unit State Machine.....	App E-19
Figure E.13 :	Cycle Translation Timing Diagram.....	App E-20
Figure E.14 :	Local Bus Arbiter State Machine .....	App E-24
Figure E.15 :	KIACK Generation.....	App E-25
Figure E.16 :	IACK Cycle Interrupt Level Generation .....	App E-25
Figure E.17 :	Reset Circuit .....	App E-26
Figure J.1 :	299-pin Cavity-down CPGA Package.....	App J-1
Figure J.2 :	Mechanical Drawing for 304-Pin PQFP Package .....	App J-2

# List of Tables

Table 1.1 :	Signals Used in this Manual .....	1-5
Table 1.2 :	Technical Support Documentation References.....	1-6
Table 2.1 :	VMEbus Request and Release Modes .....	2-8
Table 2.2 :	Setting VMEbus Request Levels .....	2-9
Table 2.3 :	Setting the VMEbus Ownership Timer .....	2-11
Table 2.4 :	Encoding for VMEbus Interrupt Levels .....	2-13
Table 2.5 :	KIPL2-KIPL0 Encoding for CPU Interrupt Levels .....	2-15
Table 2.6 :	Mapping Bits for General Local Interrupts.....	2-16
Table 2.7 :	Encoding for General Local Interrupt Level Mapping .....	2-16
Table 2.8 :	Properties of Interrupt Sources .....	2-18
Table 2.9 :	Control Bits for Interrupt Sources .....	2-20
Table 2.10 :	Status Bits for Interrupt Sources .....	2-21
Table 2.11 :	Mapping Bits for General Local Interrupts.....	2-22
Table 2.12 :	Encoding for General Local Interrupt Level Mapping .....	2-22
Table 2.13 :	KADDR3-1 Encoding for CPU Interrupt Levels .....	2-24
Table 2.14 :	Correlation Between KFC1 and Local IACK Decoding .....	2-25
Table 2.15 :	Setting VMEbus Arbitration Modes .....	2-32
Table 2.16 :	Setting VMEbus Time-out Period .....	2-34
Table 2.17 :	Functions for Different Settings of KFC2 and KFC0.....	2-52
Table 2.18 :	TXCTL Register Bits Description .....	2-64
Table 2.19 :	KFC Encodings for Transfer Type .....	2-73
Table 2.20 :	Transfer Size Encoding.....	2-74
Table 2.21 :	Port Size as Indicated by BUSSIZ and Cycle Termination .....	2-74
Table 2.22 :	Byte Lanes for Different Port Sizes on the Local Bus.....	2-75
Table 2.23 :	Encoding for Address Offset on the Local Bus .....	2-75
Table 2.24 :	A Summary of SCV64 Register Accesses .....	2-87
Table 2.25 :	SCV64 Register Map .....	2-88
Table 2.26 :	Local Bus Frequency and Allowable Wait States During Burst Writes .....	2-93

Table 2.27 :	Bit Settings for DMA Transfer Modes .....	2-105
Table 2.28 :	Setting the Tick Timer Interval.....	2-116
Table 2.29 :	VMEbus Signal Behavior in BI-mode.....	2-120
Table 4.1 :	DC Electrical Characteristics .....	4-2
Table 4.2 :	Pin List and DC Characteristics for SCV64 Signals (-55°C to 125°C) .....	4-3
Table 4.3 :	VMEbus Address and Data Input and Output Signal Bits.....	4-7
Table 4.4 :	Local Bus Address and Data Input and Output Signal Bits.....	4-8
Table 4.6 :	Pin Assignments for Power.....	4-9
Table 4.5 :	Pin Assignments for Ground.....	4-9
Table 4.7 :	Input Capacitive Loading.....	4-10
Table A.1 :	A Summary of SCV64 Register Accesses .....	App A-1
Table A.2 :	SCV64 Register Map .....	App A-2
Table A.3 :	DMA Local Address Register (DMALAR).....	App A-4
Table A.4 :	DMA VMEbus Address Register (DMAVAR).....	App A-4
Table A.5 :	DMA Transfer Count Register (DMATC) .....	App A-5
Table A.6 :	Control and Status Register (DCSR) .....	App A-6
Table A.7 :	VMEbus Slave Base Address Register (VMEBAR) .....	App A-9
Table A.8 :	A24 Slave Image Programming .....	App A-10
Table A.9 :	A32 Slave Image Size (4 Bit Field) .....	App A-12
Table A.10 :	A32 Slave Image Base Address (5 Bit Field) .....	App A-12
Table A.11 :	RXFIFO Data Register (RXDATA) .....	App A-13
Table A.12 :	RXFIFO Address Register (RXADDR) .....	App A-13
Table A.13 :	RXFIFO Control Register (RXCTL) .....	App A-14
Table A.14 :	VMEbus/VSB Bus Select (BUSSEL).....	App A-15
Table A.15 :	VMEbus Interrupter Vector (IVECT).....	App A-16
Table A.16 :	Access Protect Boundary Register (APBR).....	App A-17
Table A.17 :	TXFIFO Data Output Latch Register (TXDATA) .....	App A-18
Table A.18 :	TXFIFO Address Output Latch Register (TXADDR) .....	App A-18
Table A.19 :	Transmit FIFO AM Code and Control Bit Latch (TXCTL)...	App A-19
Table A.20 :	Location Monitor FIFO Read Port (LMFIFO) .....	App A-20



Table A.21 :	Mode Control (MODE) .....	App A-21
Table A.22 :	Slave A64 Base Address Register (SA64BAR) .....	App A-25
Table A.23 :	Master A64 Base Address Register (MA64BAR) .....	App A-26
Table A.24 :	Local Address Generator (LAG) .....	App A-26
Table A.25 :	DMA VMEbus Transfer Count (DMAVTC) .....	App A-27
Table A.26 :	Status Register 0 (STAT0).....	App A-28
Table A.27 :	Status Register 1 (STAT1).....	App A-29
Table A.28 :	General Control Register (GENCTL).....	App A-30
Table A.29 :	VMEbus Interrupter Requester (VINT).....	App A-31
Table A.30 :	VMEbus Requester Register (VREQ) .....	App A-32
Table A.31 :	VMEbus Arbiter Register (VARB) .....	App A-33
Table A.32 :	ID Register (ID) .....	App A-34
Table A.33 :	Control and Status Register (CTL2) .....	App A-35
Table A.34 :	Level 7 Interrupt Status Register (7IS) .....	App A-36
Table A.35 :	Local Interrupt Status Register (LIS).....	App A-37
Table A.36 :	Level 7 Interrupt Enable Register (7IE) .....	App A-38
Table A.37 :	Local Interrupt Enable Register (LIE) .....	App A-39
Table A.38 :	VMEbus Interrupt Enable Register (VIE) .....	App A-40
Table A.39 :	Local Interrupts 1 and 0 Control Register (IC10).....	App A-41
Table A.40 :	Local Interrupts 3 and 2 Control Register (IC32).....	App A-41
Table A.41 :	Local Interrupts 5 and 4 Control Register (IC54).....	App A-42
Table A.42 :	Miscellaneous Control Register (MISC) .....	App A-43
Table A.43 :	Delay Line Control Register (DLCT) .....	App A-44
Table A.44 :	Delay Line Status Register 1 (DLST1).....	App A-45
Table A.45 :	Delay Line Status Register 2 (DLST2).....	App A-46
Table A.46 :	Delay Line Status Register 3 (DLST3)) .....	App A-47
Table A.47 :	Mailbox Register 0 (MBOX0).....	App A-48
Table A.48 :	Mailbox Register 1 (MBOX1).....	App A-48
Table A.49 :	Mailbox Register 2 (MBOX2).....	App A-48
Table A.50 :	Mailbox Register 3 (MBOX3).....	App A-48
Table B.1 :	SCV64 AC Characteristics .....	App B-1

Table B.2 :	Capacitive Loading from Output Signals .....	App B-13
Table C.1 :	Timing Parameters .....	App C-9
Table C.2 :	Data Transfer - SCV64 as VME Master .....	App C-10
Table C.3 :	Data Transfer - DMA.....	App C-10
Table C.4 :	Data Transfer - SCV64 as VME Slave .....	App C-11
Table C.5 :	Daisy Chains .....	App C-11
Table C.6 :	VMEbus Arbiter .....	App C-11
Table D.1 :	Effect of SWAP Bit in the MODE Register on Byte Lane Translation .....	App D-2
Table D.2 :	Port Size as Indicated by BUSSIZ and Cycle Termination .....	App D-3
Table D.3 :	Transfer Size Encoding.....	App D-5
Table D.4 :	Read Cycles - SCV64 as Local Slave/VME Master (SWAP=0) .....	App D-6
Table D.5 :	Read Cycles-SCV64 as Local Slave/VME Master (SWAP=1)	App D-7
Table D.6 :	Write Cycles - SCV64 as Local Slave/VME Master (SWAP=0) .....	App D-8
Table D.7 :	Write Cycles-SCV64 as Local Slave/VME Master (SWAP=1) .....	App D-9
Table D.8 :	Read/Write Cycles - SCV64 as Local Master/VME Slave (SWAP =x, BUSSIZ=x) .....	App D-11
Table D.9 :	KFC Signal Translation to VME Address Space (SCV64 as VME Master).....	App D-12
Table D.10 :	VME AM Code Translation to KFC Signals (SCV64 as VME Slave) .....	App D-13
Table E.1 :	68040 Terminations .....	App E-16
Table E.2 :	SCV64 Terminations .....	App E-17
Table E.3 :	68040 Transfer Type Encoding .....	App E-21
Table E.4 :	Local and VME bus Space Mapping .....	App E-22
Table F.1 :	Pin Levels for Specific Power-up Modes .....	App F-1
Table F.2 :	Software Initialization Summary .....	App F-4
Table F.3 :	Programming for VME Slave Address .....	App F-6
Table F.4 :	Address Space Enabling .....	App F-7
Table H.1 :	Recommended Operating Conditions .....	App H-1

Table H.2 :	Absolute Maximum Ratings .....	App H-1
-------------	--------------------------------	---------



# 1 General Information

## 1.1 Introduction

The Single Chip VMEbus interface chip (SCV64, CA91C078A) is a member of Tundra's growing line of Backplane Interface Components, and provides a reliable high performance 64-bit VMEbus interface in one device. The SCV64 results from a four year development effort in VMEbus interface design and support, an investment which originally produced the Advanced VMEbus Interface Chip Set (AVICS).

Designed in collaboration with DY-4 Systems, the SCV64 is compliant with both the VME64 and IEEE 1014 Rev C Specifications. In addition, the SCV64 provides the high reliability of Motorola Semiconductors' manufacturing processes. The SCV64 is manufactured with a CMOS process and is available in 299-pin CPGA and 304-pin PQFP packages.

The SCV64 is ideally suited for CPU boards which assume both master and slave roles in the VME system. Under this working environment, the user can make full use of the SCV64's multiple features and options.

## 1.2 Product Overview

### 1.2.1 Flexibility and Features

One of the major strengths of the SCV64 is its functional flexibility. A rich pool of features and operational modes allows the user to tailor a VMEbus interface for a variety of data-passing environments. The flexibility provided by the SCV64 derives from the numerous options available to the user at each major functional layer: the VMEbus interface, the data path, and the local bus interface.

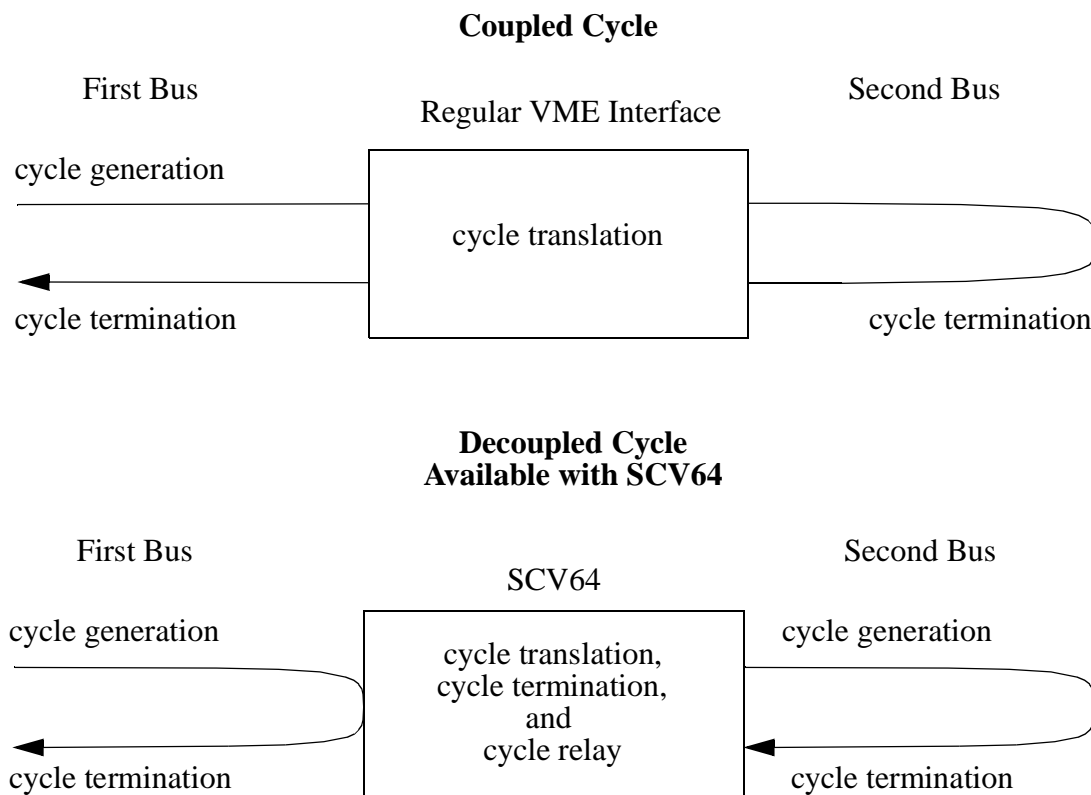
At the VMEbus interface, the user may enable and/or configure all of the following features:

- VMEbus system controller functions,
- auto-ID VME slot identification,
- automatic VMEbus Syscon identification,
- multiple VMEbus request and release options,
- multiple VMEbus arbitration schemes,

- VMEbus interrupter and interrupt handler for all seven VMEbus interrupts,
- master/slave A64/D64, A32/D64, A24/D32, D16, D08 (EO) interface,
- master A16/D16, D08 (EO) capabilities,
- VME slave image access protection, and
- automatic VME base address programming.

Data transfer between the local bus and the VMEbus normally involves coupling handshake protocols used by separate buses. This means that mastership of three buses (master local bus, VMEbus, and slave local bus) must be maintained even though data transfer only occurs on one bus at a time. The bus that originated the data transfer is only released when the last bus in the data transfer chain completes its cycle. The idle time created by cycle coupling across different buses constitutes a loss of effective bandwidth.

The SCV64 decouples data transfer by acting as a cycle shuttle at the local bus-VMEbus interface. When a cycle arrives from one bus (local or VME), it is queued in an internal SCV64 FIFO and the originating bus is notified of cycle completion. While the SCV64 acquires the other bus and completes the data transfer, the original bus is released and is free to conduct other business (see Figure 1.1 below).



**Figure 1.1 : Coupled versus Decoupled Cycles**

Along with the unique decoupling function, the SCV64 offers the following data path options:

- integral A64, A32, A24/D64, D32, D16 Direct Memory Access Controller (DMAC),
- A64/A32/A24 Multiplexed Block Transfers (MBLT),
- A32/A24 Block Transfers (BLT),
- location monitor with 31 deep message FIFO for inter-process communication, and
- a Bus Isolation (BI-mode<sup>®</sup>) controller unique to the SCV64.

Like the VMEbus interface, the SCV64's local interface may be adapted by the user to a variety of operational modes. The SCV64 local interface offers:

- a local bus requester and arbiter,
- local bus burst mode to complement BLT and MBLT cycles on the VMEbus,
- local interrupt handler functions,
- compatibility with bus-sizing or nonbus-sizing CPUs, and
- local clocks and timers.

From the list of features and options available with the VMEbus interface, the data path, and the local bus interface, it is clear that the SCV64 offers a configuration to suit most VME interface requirements.

## 1.3 Using This Document

The SCV64 Manual is organized into the following sections:

- Chapter 1- General Information,
- Chapter 2 - Functional Description,
- Chapter 3 - Signal Description,
- Chapter 4 - DC Characteristics,
- Appendix A - Registers,
- Appendix B - Timing Characteristics,
- Appendix C - SCV64 Performance,
- Appendix D - VMEbus-Local Bus Cycle Mapping,
- Appendix E - Typical Applications,
- Appendix F - Reliability Prediction
- Appendix G - Initialization,
- Appendix H- Environmental and Operating Parameters,
- Appendix I - Revision History,
- Appendix J - Mechanical and Ordering Information, and
- Index.

Chapter 1 introduces the SCV64 and provides the reader with information about the necessary concepts and conventions required to use the manual.

Chapters 2 to 4 describe SCV64 function, beginning with overall functionality in Chapter 2 to detailed signal descriptions in Chapters 3 and 4.

The Appendices are reference sources necessary for the implementation of the SCV64 (for example, register programming, timing characteristics etc.). In addition, the Appendices contain application information to aid the user in system design. Appendix I points out substantial revisions in this current version of the SCV64 User Manual and is important for users with previous versions of the manual.

An index is provided to assist the reader in accessing information using key words. Entries which refer to definitive passages in the text are highlighted with bold type face.

The Sales Network lists our current team of national and international sales representatives and distributors.



## 1.4 Conventions

### 1.4.1 Signals

Signals on the VMEbus and those on the local bus, may be active high or active low. Active low signals are defined as being true (asserted) when they are at logic low. Similarly, active high signals are defined as being true when they are at a logic high. Signals are said to be 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.

**Table 1.1 : Signals Used in this Manual**

SIGNALS	asterisk suffix to direct connect VMEbus active low signals
<u>OVERBAR</u>	on-card active low signals that do not connect directly to the VMEbus
V	prefix for nondirect-connect VMEbus signals
L	prefix for general local signals
K	prefix for signals connected to the local CPU

### 1.4.2 Symbols



*Caution: This symbol alerts the reader to procedures or operating levels which may result in misuse of or damage to the SCV64.*



*Note: This symbol directs the reader's attention to useful information or suggestions.*

### 1.4.3 Mathematical Notation

'0x' will be used to prefix hexadecimal digits.

For general technical information, please consult the references listed in Table 1.2.

**Table 1.2 : Technical Support Documentation References**

Document Number	Institution	Description
ISBN 0-13-567017-9	Motorola	MC68020 User's Manual
ISBN 0-13-566969-3	Motorola	MC68030 User's Manual
—	VITA	The VMEbus Handbook, 1991
IEEE 1014-1987	IEEE	VMEbus Specification
VME64	VITA	VME64 Specification

## 2 Functional Description

### 2.1 Introduction

#### 2.1.1 Organization of the Functional Description

The functional description of the SCV64 is organized into the following sections:

- 2.1 Introduction
- 2.2 VMEbus Requester
- 2.3 Interrupter
- 2.4 Interrupt Handler
- 2.5 System Controller Functions
- 2.6 Data Path
- 2.7 Memory Mapping
- 2.8 VMEbus Interface
- 2.9 Local Bus Interface
- 2.10 Location Monitor and LMFIFO
- 2.11 DMA Controller
- 2.12 Resets, Clocks and Timers
- 2.13 Power-up Modes
- 2.14 BI-Mode
- 2.15 Auto-ID
- 2.16 Internal Delay Line Calibration
- 2.17 Test and Diagnostic Modes

Sections 2.3 to 2.5 address VMEbus ownership, interrupts and interrupt handling, and VMEbus system controller functions provided by the SCV64. Since the control and monitoring of VMEbus and local interrupts share many characteristics within the SCV64, both types of interrupts are discussed in the Interrupter and Interrupt Handler sections.

Sections 2.6 to 2.11 focus on the architectural concepts for data transfer as performed by the SCV64. Sections 2.6 and 2.7 provide important theoretical background for the user, while sections 2.8 and 2.9 examine cycle translation in detail at the VMEbus and local bus interfaces.

Sections 2.12 to 2.17 cover a variety of additional functions and features found in the SCV64. The influence of these additional functions and features on the VMEbus interface, the data path, and the local bus interface is described in detail in each of these sections.

## 2.1.2 Functional Overview

The following overview uses the functional block diagram in Figure 2.1 on page 2-3 to organize the different components of the SCV64. The blocks which comprise the three major functional layers (data path, VMEbus interface, local bus interface) are examined in general terms below. These brief descriptions of the various functional components are intended to introduce the reader to important concepts and point to more detailed sections within the main text of this chapter.

### 2.1.2.1 Data Path

#### Receive and Transmit FIFOs

When the SCV64 is in decoupled mode, incoming cycles (i.e. VMEbus to local bus) are queued in the receive FIFO (RXFIFO) and the SCV64 terminates the cycle on the VMEbus. On the local side, the SCV64 requests the local bus as soon as entries appear in the RXFIFO.

Outgoing cycles (i.e. local bus to VMEbus) are queued in the transmit FIFO (TXFIFO). When a cycle is entered in the TXFIFO, the SCV64 terminates the local cycle by asserting  $\overline{\text{KDSACKx}}$ , and requests access to the VMEbus. The SCV64 can be programmed to request the VMEbus as soon as any entries appear in the TXFIFO or only when the TXFIFO is filled. Once the SCV64 has obtained the VMEbus, it will maintain ownership according to its bus release mode (see “VMEbus Interface” below).

For further information, see “Data Path” on page 2-36.

#### DMA Controller

For optimum bandwidth usage, the SCV64 provides a DMA controller suited to the transfer of large batches of data. With this in mind, the SCV64 is designed to generate all outgoing BLT and MBLT cycles through the DMA controller. In addition, all DMA transfers are decoupled and make use of the RXFIFO or TXFIFO.

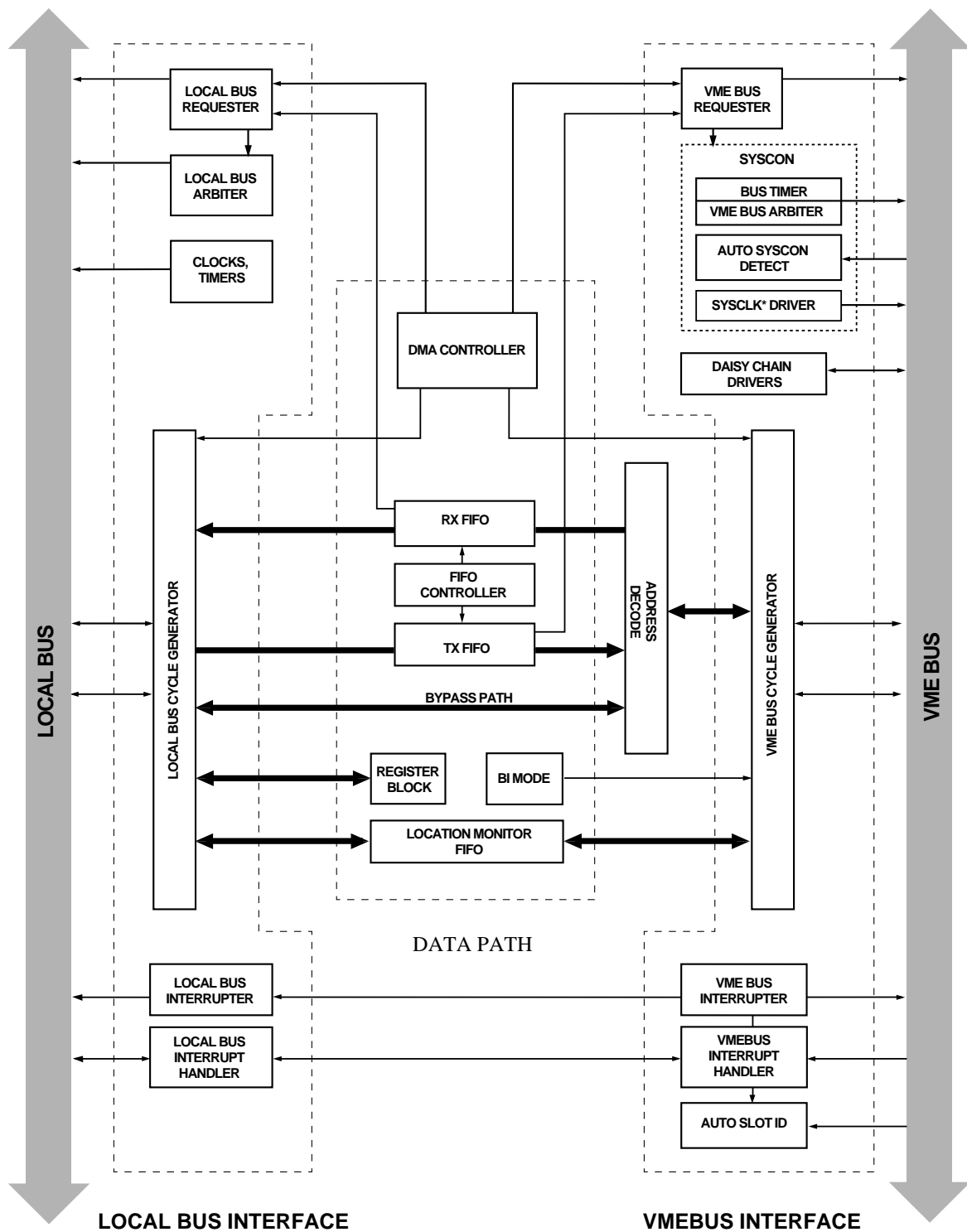


Figure 2.1 : Functional Block Diagram

The DMA controller in the SCV64 is software configurable for a variety of address and data modes (MBLT, BLT, A64, A32, A24, A16, D32, D16). Various control and status registers allow the user to direct and monitor DMA transfers. A transfer count register allows users to program up to 4 MBytes in one DMA operation, and several error checking bits provide valuable tools for data recovery in the event of bus errors.

To further maximize the rate of data transfer for DMA operations or for incoming BLT and MBLT cycles, the SCV64 provides a local burst mode. In burst mode, multiple blocks of data are transferred with only one address phase. If local burst reads are enabled, then the DMA controller will initiate a local burst cycle to fill the TXFIFO. Similarly, the SCV64 can be programmed to initiate local burst writes whenever a series of BLT or MBLT entries appear in the RXFIFO.

For further information see “DMA Controller” on page 2-102.

### Register Access

A wide range of status and control registers in the SCV64 provide a window to SCV64 operations and allow the user to configure the device for a variety of functional modes. Registers may be accessed both from the local and the VME side, allowing local and other masters control over the board’s VME interface. The SCV64 always responds as a 32-bit port when a master accesses the address range occupied by its control and status registers.

For further information see “Register Access” on page 2-87.

### Location Monitor

The location monitor allows for inter-process communication between VME masters. Writes to the location monitor generate a specific local interrupt ( $\overline{\text{LMINT}}$ ) which is usually tied to a local interrupt relayed to the CPU.

Data written to the location monitor enters a 32 bit wide, 31 stage deep location monitor FIFO (LMFIFO). The oldest entry in the LMFIFO is automatically written to the LMFIFO register.

For further information see “Location Monitor and LMFIFO” on page 2-101).

### Bus Isolation Mode (BI-Mode®)

BI-mode is a unique feature offered by the SCV64 which allows the user to functionally isolate a board from VMEbus transactions. Several BI-mode initiators are available to the user, allowing for different local designs. The user can reactivate an isolated board at any time.

Some potential applications for BI-mode are:

- hot standby systems,
- system diagnostics for routine maintenance, or
- fault isolation in the case of a board failure.

For further information see “BI-Mode” on page 2-119.

## 2.1.2.2 VMEbus Interface

### VMEbus Requester

The SCV64 can be programmed to request the VMEbus at any of the four (BR3\*, BR2\*, BR1\*, and BR0\*) levels given in the VMEbus specification, and is configurable for either Fair or Demand mode. In Fair mode, the SCV64 will not issue a bus request until the bus is clear and there are no other requests pending. In Demand mode, the SCV64 will request the bus irrespective of requests at other levels.

Once the SCV64 acquires the VMEbus, its release of the bus depends upon whether it is programmed for Release on Request (ROR) or Release When Done (RWD). With ROR, the SCV64 releases the bus only if another request is pending. In RWD mode, the SCV64 releases the VMEbus only when the current cycle is complete and the TXFIFO is empty.

To support priority arbitration, the VMEbus specification provides a bus clear signal (BCLR\*) to force low level requesters off the bus in favour of higher level requesters. The SCV64 can be programmed to accept or ignore the BCLR\* signal.

For further information see “VMEbus Requester” on page 2-8.

### VMEbus Interrupter

The SCV64 is a single level interrupter, software configurable for all seven VMEbus interrupt levels. When the SCV64 detects an IACK cycle on the level of its current interrupt, it places an 8-bit interrupt vector on the bus and terminates the cycle (the SCV64 is strictly Release on Acknowledgment, ROAK).

While all VMEbus interrupt levels are supported by the SCV64, the level one VMEbus interrupt (IRQ1\*) may be reconfigured as a BI-mode initiator. When IRQ1\* is set to this state, it is enabled and monitored through specific control and status registers.

For further information see “VMEbus Interrupts” on page 2-13.

### VMEbus Interrupt Handler

All seven VMEbus interrupt levels are accepted by the SCV64, but since they are software maskable, the user is free to control which interrupt levels the SCV64 receives. When a VME interrupt is enabled and received by the SCV64, it is translated to a corresponding local interrupt and relayed to the local CPU via the  $\overline{\text{KIPL2-0}}$  lines.

When the CPU generates an interrupt acknowledgment for a VME interrupt, the SCV64 generates an IACK cycle on the VMEbus. In what is essentially a coupled read cycle, the SCV64 accepts the interrupt vector from the VME interrupter and places it on the local bus for the CPU.

For further information see “Interrupt Handler” on page 2-18.

### **System Controller Functions**

The SCV64 automatically becomes VMEbus system controller (Syscon) if it resides at the top of the bus grant daisy chain. The following Syscon functions are provided by the SCV64:

- arbiter,
- bus timer,
- system clock driver.

All arbitration modes defined in the VMEbus specification are provided by the SCV64, namely:

- Full Priority,
- Round Robin, and
- Single Level Arbiter (a subset of Full Priority).

In addition, two mixed priority modes are provided:

- Priority Level 3, and
- Priority Levels 2 and 3.

The different arbitration modes are software configurable. There is also an arbitration time-out function to ensure that bus arbitration resumes in the event of an unanswered bus grant signal.

Similar to the arbitration timer, the SCV64 provides a VME bus timer which times out unacknowledged cycles with bus error (BERR\*) unless the VME slave responds within a specific time. The VME bus timer is also software configurable and, if activated, can be set to 16, 32, or 64  $\mu$ s.

The SCV64 provides a 16MHZ system clock within VME specifications.

For further information see “System Controller Functions” on page 2-30.

### **2.1.2.3 Local Bus Interface**

#### **Local Arbiter/Requester**

The SCV64 provides an internal local arbiter which relays bus requests from a single sub-requester to the CPU or other external arbiters. The SCV64 arbiter can be configured to either give priority to the external requester, or arbitrate between itself and the external requester at an equal priority level. The user may also entirely bypass the SCV64 arbiter at power-up.

The SCV64 arbiter allows the external requester to maintain bus ownership with either bus grant acknowledgment or the bus request signal. By using bus grant acknowledgment to hold the bus, the bus request line is available to other local devices before the current cycle is complete.



For further information see “Local Bus Arbitration” on page 2-66.

### Local Interrupter

Interrupts to the local CPU can be generated at any one of 7 levels using the  $\overline{\text{KIPL2-0}}$  lines. In its function as local interrupt handler, the SCV64 uses the  $\overline{\text{KIPL2-0}}$  lines to relay interrupts from various sources (VME and local) to the CPU.

Two other local interrupts,  $\overline{\text{LMINT}}$  and  $\overline{\text{VMEINT}}$ , are provided by the SCV64 and are used to notify the CPU of specific SCV64 activity.  $\overline{\text{LMINT}}$  informs the CPU of writes to the location monitor, while  $\overline{\text{VMEINT}}$  is asserted whenever specific error and status bits become set.

For further information see “Local Bus Interrupts” on page 2-15.

### Local Interrupt Handler

The SCV64 provides a local interrupt management function in that it accepts interrupts from different local sources, prioritizes the interrupts and then relays the signals to the local CPU via the  $\overline{\text{KIPL2-0}}$  lines. The handling of the subsequent CPU-generated IACK cycle depends upon whether the original interrupt source is vectored or auto-vectored. If the local interrupter is vectored, the SCV64 communicates the interrupt acknowledgment to the interrupt source which then places its interrupt vector on the local data bus. If the local interrupter is auto-vectored, then the SCV64 signals the CPU to generate its own interrupt vector internally.

For further information see “Interrupt Handler” on page 2-18.

### Local Clocks and Timers

Three different clocks are made available by the SCV64. BAUDCLK provides a 2.4615 MHz signal for use by local devices generating baud rate clocks. A 14  $\mu\text{s}$  clock signal, derived from 32 MHz input, is constantly synchronized with the CPU clock. For additional general purpose applications, the SCV64 generates an 8 MHz clock.

Along with the local clocks provided by the SCV64, there are three local timers. These are: a local bus timer (synchronized to the CPU clock), the watchdog timer and the tick timer.

For further information see “Resets, Clocks and Timers” on page 2-110.

## 2.2 VMEbus Requester

### 2.2.1 Function

The SCV64 VMEbus requester requests VMEbus ownership if:

- an outgoing cycle is initiated (see “Memory Mapping” on page 2-45),
- under the direction of its DMA controller (see “DMA Controller” on page 2-102), or
- the CPU accesses its own VME slave image and the SCV64 is in loopback mode (see “Loopback Diagnostics” on page 2-126).

The SCV64 releases the bus if:

- the VMEbus signal BCLR\* is received (due to the arbiter receiving a higher priority VMEbus request, ROR mode),
- data transfer is complete (RWD mode), or
- its programmed ownership period has expired.

While the SCV64 owns the VMEbus, it asserts BBSY\* (as indicated by the MYBBSY bit in the CTL2 register, Table A.33).

Several request and release modes are listed in Table 2.1 and described in the following sections.



*Note: since the following initiators of request and release act independently of each other, some functional overlap may occur.*

**Table 2.1 : VMEbus Request and Release Modes**

<b>Request Mode</b>	<b>Description</b>
Fair or Demand	Controls the interaction of the requester with BBSY*
Levels 3 to 0	VMEbus Request Level
<b>Release Mode</b>	<b>Description</b>
Bus Clear Enable or Disable	Controls the interaction of the requester with BCLR*
Release on Request	Gives up bus ownership with any request
Release When Done	Gives up bus ownership only when transaction is complete
Ownership Time-out	An optional timer can be used to limit the requester's ownership of the VMEbus

## 2.2.2 Bus Request Modes

### 2.2.2.1 Fair and Demand Modes

The requester can be programmed for either the Fair mode or the Demand mode by setting the REQ control bit in the VREQ Register (see Table A.30).

In Fair mode, the SCV64 waits until there are no requests pending at its programmed level and BBSY\* is high. This mode ensures that every requester on a given level has access to the bus.

In Demand mode (the default setting), the requester asserts its bus request under the following conditions:

- entries exist in the Transmit FIFO,
- coupled read or write is initiated by local bus, or
- a VMEbus IACK cycle is pending.

By requesting the bus frequently, requesters far down the daisy chain may be prevented from ever obtaining bus ownership. This is referred to as “starving” those requesters.

### 2.2.2.2 VMEbus Request Levels

The level used by the requester is selected through the LVL1 and LVL0 bits in the VREQ Register (Table A.30, see Table 2.15 below). Level 3 is the default setting after reset.

**Table 2.2 : Setting VMEbus Request Levels**

Request Level	Setting for LVL1,0
BR0*	00
BR1*	01
BR2*	10
BR3*	11

During PRI arbitration, level 3 has the highest priority and level 0 the lowest (see “VMEbus Arbiter” on page 2-31).



*Caution: The LVL1 and LVL0 bits must not be changed while the VMEbus is being arbitrated; that is, while the requester is propagating any BGnOUT\* signals. Changing these bits during arbitration may introduce pulses on the BGnOUT\* signals of the current and next request levels involved. It is recommended that the bits be changed when the VMEbus is not in use (such as after a reset when all potential masters are performing self tests and configuration, or while the system is in BI-mode).*

If a request for the data transfer bus is initiated and then withdrawn, the SCV64:

1. continues its request for VMEbus ownership according to its programmed modes,
2. asserts BBSY\* for at least the minimum specified time, and then
3. releases the bus.

The protocol above is in compliance with the VMEbus specification. The requester may initiate the protocol during the resolution of deadlock cycles when the CPU attempts a coupled access to the VMEbus while the VMEbus is already attempting a coupled access to the local bus.

### 2.2.3 Bus Release Modes

#### 2.2.3.1 Bus Clear Enabling

Bus Clear (BCLR\*) is asserted by the VMEbus arbiter if a request is asserted that has higher priority than the current owner (see “VMEbus Arbiter” on page 2-31) or if bus ownership has timed out (see “Bus Timer” on page 2-34). Release on BCLR\* is enabled by setting the BCEN bit in the VREQ register (Table A.30, and below). If Release on BCLR\* is enabled, the requester in the SCV64 releases the bus when BCLR\* becomes asserted. The default setting after reset is with BCLR\* ignored.

#### 2.2.3.2 Release On Request and Release When Done

The SCV64 can be programmed to either Release on Request (ROR) or Release When Done (RWD). These two modes are controlled by REL bit in the VREQ register (Table A.30).

When the REL bit is set to ROR, the BBSY\* signal is released by the requester only if a bus request is pending.

In RWD mode (the default setting), the requester only releases the data transfer bus when:

- there are no further entries in the TXFIFO,
- the coupled cycle is complete, or
- the IACK cycle is complete.



*Caution: If the SCV64 is programmed in Fair and ROR mode, and some other board is using the bus on the same request level, the CPU and SCV64 may toggle endlessly, requesting bus grant and discarding it. To avoid this configuration, use different request levels, but not the Fair and ROR modes at the same time.*

### 2.2.3.3 Ownership Timer

VMEbus ownership time can be limited using an ownership timer. The timer is enabled using the OTEN bit in the VREQ register. The time-out period is selected using the OT1 and OT0 bits in the same register (Table A.30, and see Table 2.3 below).

If the Ownership Timer is enabled (the default setting), the requester releases the VMEbus after it has owned it for the programmed time (8  $\mu$ s is the default setting). Programming the OT1 and OT0 bits for zero time limits the SCV64 to only one or two transfers on the bus. If the ownership timer is not enabled the ownership period is unlimited.

**Table 2.3 : Setting the VMEbus Ownership Timer**

Timer Duration	Setting for OT1,0
0 time	00
2 $\mu$ s	01
4 $\mu$ s	10
8 $\mu$ s (default)	11

The ownership timer permits rapid bus mastership switching by allowing other VME masters predictable bus access latencies.

## 2.2.4 Other Bus Release Mechanisms

### 2.2.4.1 Local Memory Interrupt

Asserting the  $\overline{L7IMEM}$  pin causes the SCV64 to release the VMEbus when the  $\overline{L7IMEM}$  pin is enabled (see “Interrupt Enabling and Status” on page 2-20). In addition, no further bus requests will be issued while  $\overline{L7IMEM}$  is asserted. If  $\overline{L7IMEM}$  is asserted after a request has been issued, the SCV64:

1. obtains the bus,
2. asserts BBSY\* for at least the minimum required time and releases the bus without performing any transfers.

The  $\overline{L7IMEM}$  input can be programmed to indicate a local memory failure. Using this feature, the requester will prevent a device such as DMA from transferring unreliable memory contents to the bus until the local CPU has resolved the memory fault and cleared  $\overline{L7IMEM}$ .

### 2.2.4.2 BI-Mode

BI-mode has the same effect on the requester as  $\overline{L7IMEM}$ . The requester releases the bus and will not generate any new bus requests. Requests in progress are completed and then aborted with minimum BBSY\* assertion. BI-mode functionality is described in “BI-Mode” on page 2-119.

### 2.2.4.3 Local and System Reset

Local or system reset causes the requester to immediately deassert all of its outputs including:

- any Bus Requests,
- any Bus Grants, and
- BBSY\*.

Note that deasserting a request in progress through a local or system reset may generate improper timing of a Bus Grant propagating on the level programmed for use by the requester.

## 2.3 Interrupter

### 2.3.1 VMEbus Interrupts

#### 2.3.1.1 Interrupt Generation

The SCV64 can be programmed to generate VMEbus interrupts on any one of the seven VME interrupt levels. The interrupt level and interrupt initiation are both controlled by the VINT register (Table A.29). Interrupt level is set using the IL2-IL0 bits, see Table 2.4 for a complete listing of VMEbus interrupt levels and IL2-IL0 encoding. Setting the INT bit in the same register initiates a VMEbus interrupt at the level programmed with the IL0-IL2 bits. The SCV64 clears the INT bit after the interrupter has placed its interrupt vector on the data transfer bus. Note that an interrupt can only be deasserted by writing 0 to the IL2-IL0 bits in the VINT register. Writing 0 to the INT bit has no effect on interrupt assertion.

**Table 2.4 : Encoding for VMEbus Interrupt Levels**

VMEbus Interrupt Level	Setting for IL2-IL0
7	111
6	110
5	101
4	100
3	011
2	010
1	001
0 (no interrupt)	000

The IRQ1\* signal can be configured either as an interrupt or as a BI-mode initiator. By default, the IRQ1\* signal serves as a BI-mode initiator. Under this configuration, setting the ABI bit in the GENCTL register (Table A.28) asserts IRQ1\* as a BI-mode initiator. Alternatively, IRQ1\* can be configured as a VMEbus interrupt by clearing the VI1BI bit in the GENCTL register.



*Caution: Changing the interrupt level while INT is state 1 may cause improper operation in the IACK DCD logic.*

When the SCV64 detects an IACK cycle on its interrupt level, it places its 8-bit interrupt vector on the data lines VDATA 07-00 and asserts DTACK\* (the SCV64 is strictly release on acknowledgment, ROAK). The interrupt vector is held on the bus until DS0\* is negated.

The interrupt vector is programmed into the lower 8-bits of the IVECT register (Table A.15). Although the upper 3 bytes of the IVECT register are unused, the register must still be accessed as a longword port. In addition, since the SCV64 is strictly an 8-bit interrupter, the data strobe and LWORD\* signals are not decoded for information about data transfer size.

### 2.3.1.2 BI-mode Effects

Entry into BI-mode causes any SCV64-generated interrupts to be immediately deasserted. If the SCV64 is asserting an interrupt, then the IACK Daisy Chain will be sensitive to BI-mode entry if it is generating IACKOUT\*. Under these circumstances, the SCV64 will either rescind the signal or not generate IACKOUT\*. Typically, a VME data transfer timer will then end the IACK cycle by asserting BERR\*.

The Interrupter can be programmed while it is in BI-mode. Although no interrupts are asserted while the SCV64 is in BI-mode, programmed interrupts will be asserted when the SCV64 exits BI-mode. To cancel a pending interrupt, the IL2-IL0 bits in the VINT register must be cleared. When the SCV64 exits BI-mode, the Interrupt logic completes the clearing operation by resetting the INT bit in the VINT register.

### 2.3.1.3 Reset Effects

Any reset (power reset, system reset, external reset or software reset, see “Resets” on page 2-110) will return the SCV64 registers to their default settings. Therefore, after reset the user can expect that:

- all VMEbus interrupts are deasserted (the IL2-IL0 bits in the VINT register have been cleared), and
- the IRQ1\* signal has returned to its default configuration as a BI-mode initiator.



### 2.3.2 Local Bus Interrupts

As part of its function as Interrupt Handler, the SCV64 generates interrupts to the CPU using the KIPL2-KIPL0 signals (see Figure 2.2 on page 2-19). The KIPL2-KIPL0 signals encode 7 possible CPU interrupt levels (level 7 to 1) according to Table 2.5. Interrupt requests from level 7, local, and VME sources are all channelled via prioritizing logic to a common set of CPU interrupt levels. Level 7 interrupts are automatically mapped to the highest CPU interrupt level, while VMEbus interrupts are all pre-mapped to their corresponding CPU interrupt levels. However, the general local interrupts ( $\overline{\text{LIRQ5}}$  -  $\overline{\text{LIRQ0}}$ ) can be programmed to different CPU interrupt levels.

The KIPL2-KIPL0 signals assert asynchronously upon the assertion of any local interrupt source, and similarly only negate ( $\text{KIPL} [2:0] = 111$ ) once the source interrupt is negated. If a higher priority interrupt occurs, then the KIPL2-KIPL0 signals immediately change to reflect the new interrupt level. Local interface logic should be designed to latch the interrupt level on the KIPL [2:0] lines when generating the level for an interrupt acknowledge cycle.

**Table 2.5 : KIPL2-KIPL0 Encoding for CPU Interrupt Levels**

CPU Interrupt Level	Setting for KIPL2-KIPL0
7	000
6	001
5	010
4	011
3	100
2	101
1	110
0 (no interrupt)	111

Local interrupt level mapping is performed through the IC10 register (Table A.39), the IC32 register (Table A.40), and IC54 register (Table A.41) (see Table 2.11 below for a complete listing of general local interrupts and their corresponding mapping bits). Each interrupt source has a group of 3 mapping bits which are used to assign that particular interrupt pin a CPU interrupt level between 0 and 7. For example, writing 001 to bits 3L2, 3L1 and 3L0 in the IC32 register maps  $\overline{\text{LIRQ3}}$  to CPU interrupt level 1 (Table 2.12 below provides the encoding for different interrupt levels). All general local interrupts default to CPU interrupt level 0 after reset.

**Table 2.6 : Mapping Bits for General Local Interrupts**

Local Interrupt	Register	Mapping Bits
$\overline{\text{LIRQ5}}$	IC54	5L2-5L0
$\overline{\text{LIRQ4}}$		4L2-4L0
$\overline{\text{LIRQ3}}$	IC32	3L2-3L0
$\overline{\text{LIRQ2}}$		2L2-2L0
$\overline{\text{LIRQ1}}$	IC10	1L2-1L0
$\overline{\text{LIRQ0}}$		0L2-0L0

**Table 2.7 : Encoding for General Local Interrupt Level Mapping**

CPU Interrupt Level	Setting for Mapping Bits
7	111
6	110
5	101
4	100
3	011
2	010
1	001
0 (no interrupt)	000

Two other local interrupt sources are available by using the  $\overline{\text{LMINT}}$  and  $\overline{\text{VMEINT}}$  pins. These two interrupts can typically be tied to any of the general local interrupt inputs ( $\overline{\text{LIRQ5}}$  -  $\overline{\text{LIRQ0}}$ ).  $\overline{\text{LMINT}}$  is held asserted while there are entries in the location monitor FIFO, as indicated by the LMHD bit in the DCSR register (see “Location Monitor and LMFIFO” on page 2-101). While the  $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin is in  $\overline{\text{KIACK}}$  mode (see page 2-25), the  $\overline{\text{LMINT}}$  signal is tied to the  $\overline{\text{LIRQ2}}$  line.

The  $\overline{\text{VMEINT}}$  signal is asserted when any of the following flags in the DCSR register become set:

- CERR (indicating a DMA configuration error, see “DMA Completion and Error Checking” on page 2-108),
- RMCERR (indicating a lockup in a read modify write cycle, see “During burst transfers, the SCV64 presents addresses only at burst length boundaries. Therefore, for some applications, a local address counter has to be provided as the SCV64 does not increment the addresses during the burst.” on page 2-96),
- DLBERR (indicating a local bus error while the DMA controller owns the local bus, see “DMA Completion and Error Checking” on page 2-108),
- LBERR (indicating a local bus error during data transfer from the RXFIFO, see “DMA Completion and Error Checking” on page 2-108),
- VBERR (indicating a VMEbus error during data transfer from the TXFIFO, see “DMA Completion and Error Checking” on page 2-108), and
- DONE (indicating that the DMA data transfer is complete, see “DMA Completion and Error Checking” on page 2-108).

## 2.4 Interrupt Handler

The Interrupt Handler module (shown in Figure 2.2 on page 2-19) accepts interrupt requests from three groups of interrupt sources. These are listed below according to their priority:

1. level 7 interrupts,
2. general local interrupts, and
3. VMEbus interrupts.

Figure 2.2 shows how the different interrupt sources are prioritized and channelled to interrupt levels recognized by the CPU (as encoded by the  $\overline{\text{KIPL2-0}}$  signals). Not only are all interrupt sources channelled through a common interrupt handler module, but all interrupt acknowledge cycles are processed by the same acknowledge module (see Figure 2.2). The acknowledge module output depends upon the interrupt level and the original interrupt source. This section first describes how to enable interrupts and monitor their status, and then discusses the different types of IACK cycles.

**Table 2.8 : Properties of Interrupt Sources**

Interrupt	Sensitivity	Maskable	Latch	Latch Readable	Source Readable	Vectored	Auto-vectored
BIMODE	Level	Yes	Yes	Yes	Yes	No	Yes
$\overline{\text{L7NMI}}$	Edge	No	Yes	Yes	Yes	No	Yes
$\overline{\text{L7IMEM}}$	Level	Yes	Yes	Yes	Yes	No	Yes
SYSFAIL*	Level	Yes	Yes	Yes	Yes	No	Yes
$\overline{\text{L7IACF}}$	Level	Yes	Yes	Yes	Yes	No	Yes
$\overline{\text{LIRQ5-4}}$	Level	Yes	No	No	Yes	Yes	Yes
$\overline{\text{LIRQ3-0}}$	Level	Yes	No	No	Yes	No	Yes
IRQ7-2*	Level	Yes	No	No	No	Yes	No
IRQ1*	Level	Yes	No	No	Yes	Yes (as IRQ1*)	YES (as BI-mode signal)

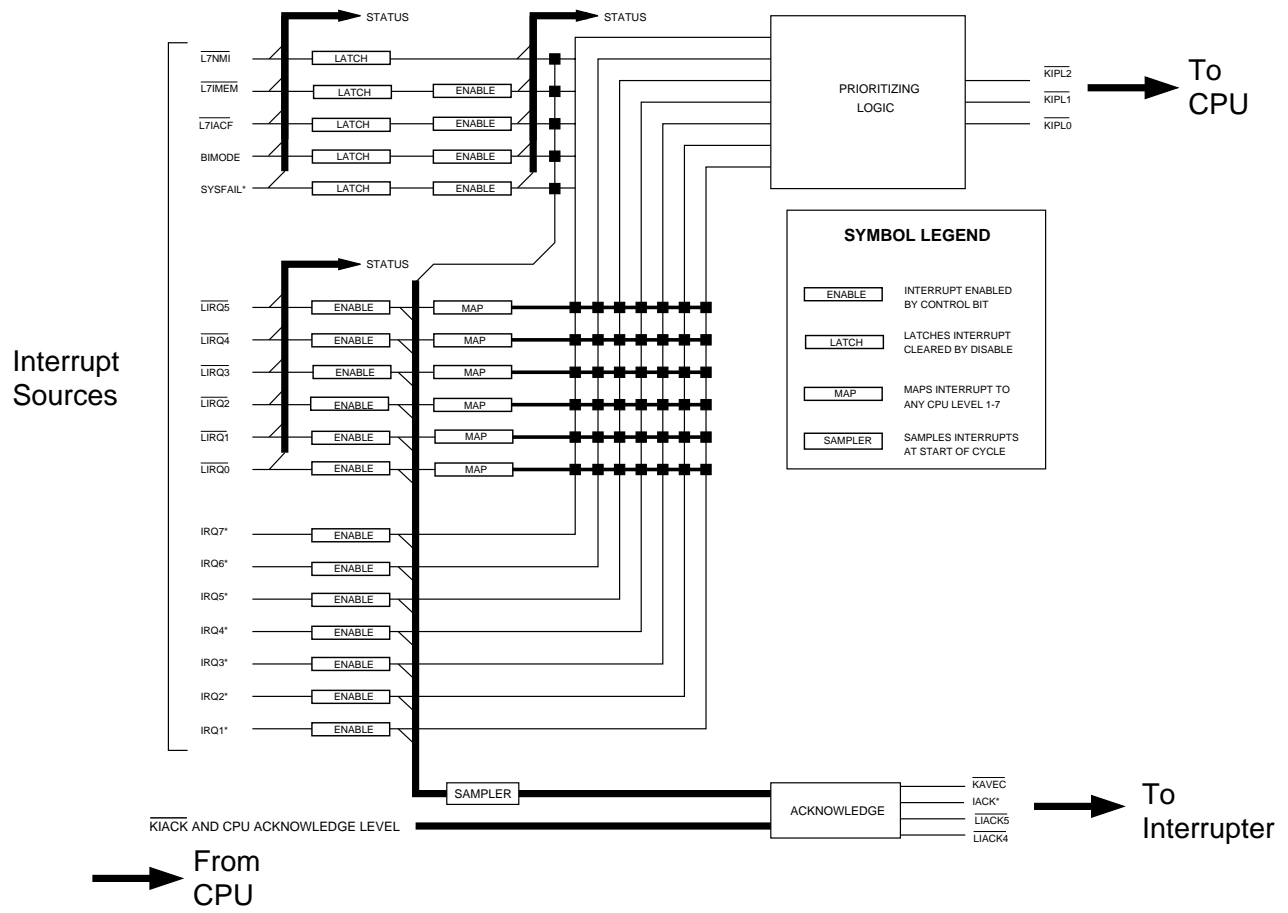


Figure 2.2 : Interrupt Handler Block Diagram

## 2.4.1 Interrupt Enabling and Status

Each interrupt group has its own control register for enabling. The 7IE register (Table A.36) enables level 7 interrupts, the LIE register (Table A.37) enables general local interrupts, and the VIE register (Table A.38) enables VMEbus interrupts. The interrupt source is enabled by setting its corresponding control bit (see Table 2.9 for a complete listing). Note that  $\overline{L7INMI}$  is always enabled and its control bit in the 7IE register only serves to clear the interrupt.

**Table 2.9 : Control Bits for Interrupt Sources**

Level 7 Interrupts	Control Bits in 7IE Register	General Local Interrupts	Control Bits in LIE Interrupts	VMEbus Interrupts	Control Bits in VIE Registers
$\overline{L7NMI}$	NMICLR	$\overline{LIRQ5}$	L5E	IRQ7*	V7E
$\overline{L7IMEM}$	MEMIE	$\overline{LIRQ4}$	L4E	IRQ6*	V6E
$\overline{L7IACF}$	ACFIE	$\overline{LIRQ3}$	L3E	IRQ5*	V5E
BIMODE	BIE	$\overline{LIRQ2}$	L2E	IRQ4*	V4E
SYSFAIL*	SYFIE	$\overline{LIRQ1}$	L1E	IRQ3*	V3E
		$\overline{LIRQ0}$	L0E	IRQ2*	V2E
				IRQ1*	V1E



*Note that IRQ1\* defaults as a BI-mode initiator. To use IRQ1\* as a VME interrupt signal, the user must clear the VIIBI bit in the GENCTL register (Table A.28). Do not assert IRQ1\* via the ABI bit in the GENCTL register when IRQ1\* is configured as a VME interrupt.*

All level 7 interrupts (except  $\overline{L7INMI}$ ) are reset by disabling and then re-enabling them through their corresponding control bit.  $\overline{L7INMI}$  is reset by clearing the NMICLR bit in the 7IE register.

The status of level 7 interrupts and general local interrupts is monitored through interrupt status registers. As shown in Figure 2.2 on page 2-19, the status of level 7 interrupts can be read both at the pin level (STAT0 register, Table A.26) and the latch level (7IS register, Table A.34). The status of general local interrupts is monitored only at the pin level through the LIS register (Table A.35). If the interrupt's corresponding status bit is set, then the pin is asserted (or the interrupt is latched in the case of the 7IS register). See Table 2.10 below for a complete list of level 7 and VMEbus interrupts with their corresponding status bits.



*Note the status of VMEbus interrupts cannot be monitored. The exception is  $\overline{IRQ1}^*$ , which can serve as a BI-mode initiator and is monitored through the  $\overline{VI1}$  bit in the  $\overline{STAT0}$  register (Table A.26).*

**Table 2.10 : Status Bits for Interrupt Sources**

Level 7 Interrupts - Pin Status	Status Bits in $\overline{STAT0}$ Register	Level 7 Interrupts - Latch Status	Status Bits in $\overline{7IS}$ Register	General Local Interrupts	Status Bits in LIS Interrupts
$\overline{L7NMI}$	NMIIP	$\overline{L7NMI}$	NMIIS	$\overline{LIRQ5}$	LI5
$\overline{L7IMEM}$	MEMIP	$\overline{L7IMEM}$	MEMIS	$\overline{LIRQ4}$	LI4
$\overline{L7IACF}$	ACFIP	$\overline{L7IACF}$	ACFIS	$\overline{LIRQ3}$	LI3
BIMODE	$\overline{VI1}$	BIMODE	BIS	$\overline{LIRQ2}$	LI2
$\overline{SYSFAIL}^*$	SYFIP	$\overline{SYSFAIL}^*$	SYFIS	$\overline{LIRQ1}$	LI1
				$\overline{LIRQ0}$	LI0

### 2.4.1.1 Local Interrupt Level Mapping

By definition, Level 7 interrupts are all mapped to the highest CPU interrupt level ( $\overline{KIPL2-0} = 000$ ), while VMEbus interrupts are all pre-mapped to their corresponding CPU interrupt levels. However, the general local interrupts ( $\overline{LIRQ5} - \overline{LIRQ0}$ ) can be programmed to different CPU interrupt levels. Local interrupt level mapping is performed through the IC10 register (Table A.39), the IC32 register (Table A.40), and IC54 register (Table A.41) (see Table 2.11 below for a complete listing of general local interrupts and their corresponding mapping bits). Each general local interrupt source has a group of 3 mapping bits which are used to assign that particular interrupt pin a CPU interrupt level between 0 and 7. For example, writing 001 to bits 3L2, 3L1 and 3L0 in the IC32 register maps  $\overline{LIRQ3}$  to CPU interrupt level 1 (Table 2.12 below provides the encoding for different interrupt levels). All general local interrupts default to level 0 after reset.

Since  $\overline{\text{LIRQ5}}$  and  $\overline{\text{LIRQ4}}$  can be vectored or auto-vectored, the IC54 register contains bits (5AV and 4AV) which are used to set these interrupts to either of these states. The interrupt is auto-vectored if its corresponding auto-vector control bit is set. For example, clearing the 5AV bit in the IC54 register programs  $\overline{\text{LIRQ5}}$  as a vectored interrupt. Interrupts  $\overline{\text{LIRQ3}}$  -  $\overline{\text{LIRQ0}}$  can only be auto-vectored. The auto-vector control bits default to 0 after reset ( $\overline{\text{LIRQ5}}$  and  $\overline{\text{LIRQ4}}$  are vectored).

**Table 2.11 : Mapping Bits for General Local Interrupts**

Local Interrupt	Register	Mapping Bits
$\overline{\text{LIRQ5}}$	IC54	5L2-5L0
$\overline{\text{LIRQ4}}$		4L2-4L0
$\overline{\text{LIRQ3}}$	IC32	3L2-3L0
$\overline{\text{LIRQ2}}$		2L2-2L0
$\overline{\text{LIRQ1}}$	IC10	1L2-1L0
$\overline{\text{LIRQ0}}$		0L2-0L0

**Table 2.12 : Encoding for General Local Interrupt Level Mapping**

CPU Interrupt Level	Setting for Mapping Bits
7	111
6	110
5	101
4	100
3	011
2	010
1	001
0 (no interrupt)	000

## 2.4.2 Interrupt Acknowledge Cycles

Figure 2.2 on page 2-19 shows that when the interrupt handler module receives an interrupt request (either vectored or auto-vectored), it propagates the interrupt to the CPU using the  $\overline{\text{KIPL2-0}}$  signals. When the local CPU receives the interrupt, it generates an interrupt acknowledge cycle. This IACK cycle can be either decoded by an external device or decoded internally by the SCV64. In either case, the IACK cycle generated by the CPU leads to the SCV64 receiving  $\overline{\text{KIACK}}$ . The  $\overline{\text{KIACK}}$  input signals the SCV64 to generate an IACK cycle.



$\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  are functionally defined at power-up (see Figure 2.3), and the mode for this pin depends upon whether the SCV64 uses internal or external local IACK decoding. The IACK cycle level need not match the current level reflected on the KIPL lines. Any active and enabled interrupt level may be acknowledged.

For interrupts mapped to the same level, the acknowledge priority is:

- any auto-vectored interrupts on that level,
- $\overline{\text{LIRQ5}}$  if vectored and programmed for that level,
- $\overline{\text{LIRQ4}}$  if vectored and programmed for that level,
- the VME interrupt on that level.

For internal IACK decoding ( $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin serves as  $\overline{\text{LIRQ2}}$ ) the SCV64 requires:

- $\text{KFC} = 111$ ,
- $\text{KADDR3-1}$  = level of interrupt being acknowledged (see Table 2.13 below),
- $\text{KADDR0} = 1$
- $\overline{\text{KAS}} = 0$ ,
- $\text{KADDR19-16} = 1111$ , and
- $\overline{\text{KDS}} = 0$ .

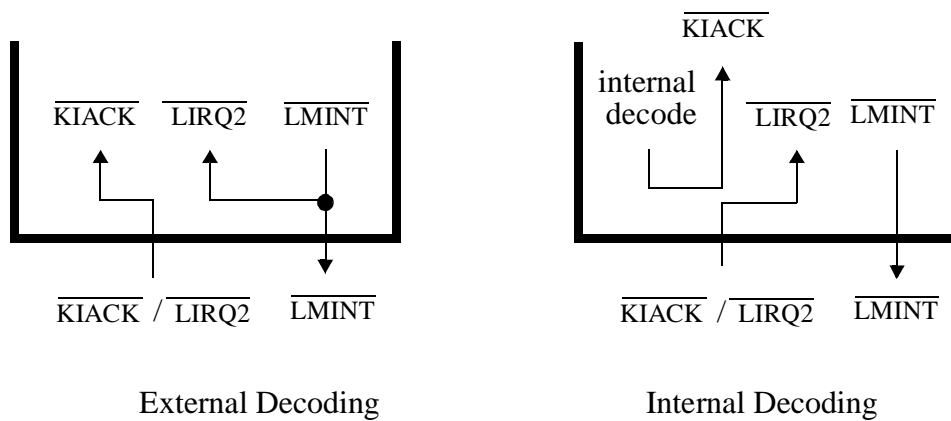
For external IACK decoding ( $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin serves as  $\overline{\text{KIACK}}$ ) the SCV64 requires:

- $\overline{\text{KIACK}} = 0$ ,
- $\text{KADDR3-1}$  = level of interrupt being acknowledged (see Table 2.13 below),
- $\text{KADDR0} = 1$ ,
- $\overline{\text{KAS}} = 0$ , and
- $\overline{\text{KDS}} = 0$ .

**Table 2.13 : KADDR3-1 Encoding for CPU Interrupt Levels**

CPU Interrupt Level	Signal on KADDR3-1
7	111
6	110
5	101
4	100
3	011
2	010
1	001

In  $\overline{\text{KIACK}}$  mode, the  $\overline{\text{LMINT}}$  signal is internally tied to the  $\overline{\text{LIRQ2}}$  line (see Figure 2.3).

**Figure 2.3 : Connections Resulting from Changing Local IACK Decoding**

**Table 2.14 : Correlation Between KFC1 and Local IACK Decoding**

KFC1	Mode
high	$\overline{\text{KIACK}}$
low	$\overline{\text{LIRQ2}}$

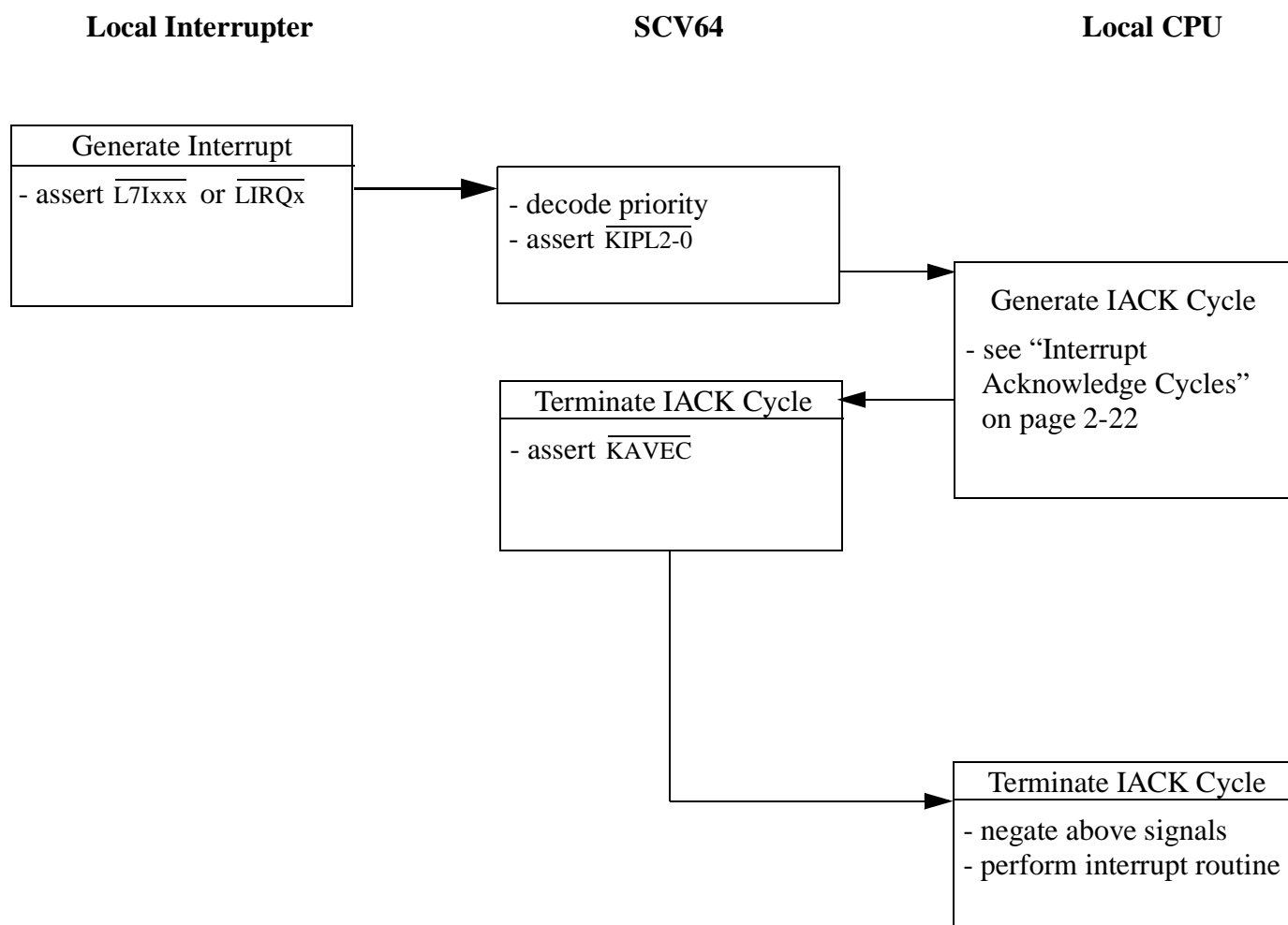
### 2.4.2.1 Auto-Vectored Interrupts

If the interrupter cannot supply an interrupt vector, then the IACK cycle is terminated with  $\overline{\text{KAVEC}}$  (see Figure 2.4 and Figure 2.5 below).  $\overline{\text{KAVEC}}$  signals the CPU to generate its own interrupt vector internally. The SCV64 acknowledge module shown in Figure 2.2 on page 2-19 asserts  $\overline{\text{KAVEC}}$  if it samples any of the following interrupts:

- all local dedicated level 7 interrupts,
- $\overline{\text{LIRQ5}}$  and  $\overline{\text{LIRQ4}}$  (if configured as auto-vector), or
- $\overline{\text{LIRQ3}}$  to  $\overline{\text{LIRQ0}}$ .



*Note: Since the SCV64 auto-vectors all level 7 interrupts, expanding any of the  $\overline{\text{L7Ixxx}}$  pins to more sources is easily accomplished by wire OR-ing several signals and adding a pull-up resistor to any of the  $\overline{\text{L7Ixxx}}$  pins.*



**Figure 2.4 : Local Interrupt Cycle - Auto-Vectored**

### 2.4.2.2 Vectored Interrupts

Vectored interrupts may originate from either local sources (using  $\overline{LIRQ5}$  or  $\overline{LIRQ4}$ ) or VMEbus interrupters. The flow for these two types of IACK cycles are similar and are both described in Figure 2.6 on page 2-28. However, because the IACK cycle for a VMEbus interrupter is essentially a coupled read, its timing differs significantly from a local vectored IACK cycle (see Figure 2.5 on page 2-27). See "Local Cycles – Overview" on page 2-73 and "SCV64 as Local Slave" on page 2-78 for further information on local bus cycles.

The SCV64 signals a local vectored interrupter about the IACK cycle by asserting  $\overline{\text{LIACK5}}$  or  $\overline{\text{LIACK4}}$ , depending upon the original interrupt. When the local interrupter receives notification of IACK, it places the interrupt vector on the data bus and asserts  $\overline{\text{KDSACKx}}$ . The CPU latches the vector and negates  $\overline{\text{KDS}}$  and  $\overline{\text{KAS}}$ .

In a similar manner, the SCV64 notifies a VME interrupter of the IACK cycle by asserting IACK\*. When the VMEbus interrupter receives IACKIN\* on its IACK daisy chain driver, it places its 8 bit vector on the VMEbus and asserts DTACK\*. As in any coupled read, the SCV64 transfers the data from the VMEbus to the local data bus and asserts  $\overline{\text{KDSACKx}}$ . The CPU latches the data and negates  $\overline{\text{KDS}}$  and  $\overline{\text{KAS}}$ .

Vectored IACK cycles with a VME interrupter differ from a strictly local IACK cycle in that the SCV64 places the vector on the local data bus. In addition, since the IACK cycle for a VME interrupter is coupled, the VMEbus is only released when the local IACK cycle is complete.

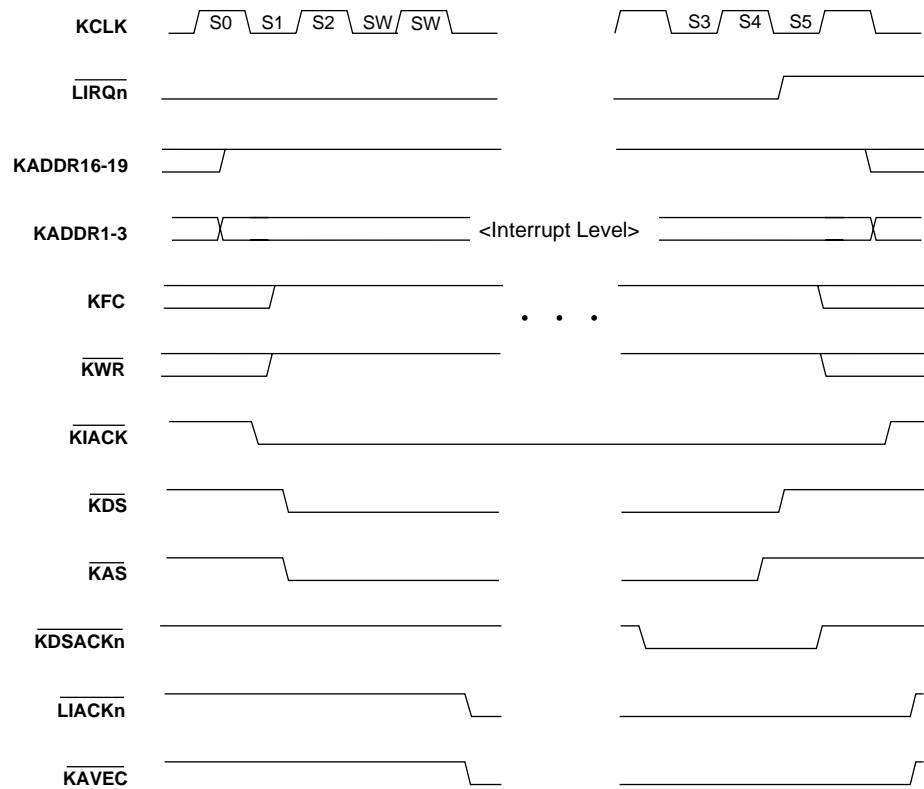


Figure 2.5 : Timing for Local Interrupts

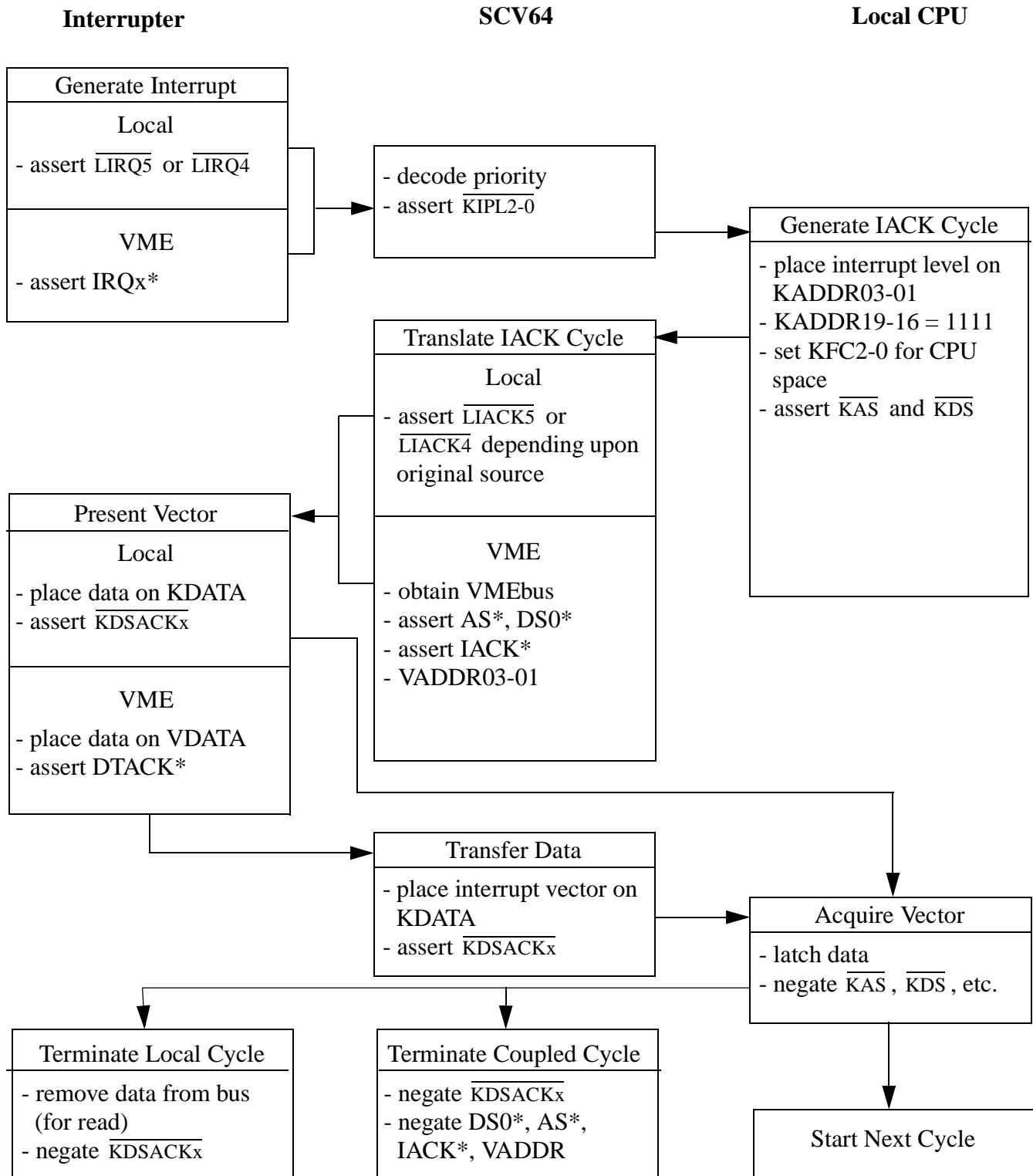


Figure 2.6 : Vectored Interrupt Cycle - with Local and VME Interrupter

### 2.4.2.3 BI-Mode Effects

If the SCV64 is in BI-mode, it will still decode VMEbus interrupts to the CPU. However, local IACK cycles generated by the CPU in response to VMEbus interrupts will not be propagated to the VMEbus. The local bus timer is expected to time out the local IACK cycle with  $\overline{\text{KBERR}}$ .

### 2.4.2.4 Reset Effects

Resetting the SCV64 returns its registers to their default settings. Therefore, all interrupts (except  $\overline{\text{L7INMI}}$ ) will be disabled (see “Interrupt Enabling and Status” on page 2-20). The user must re-initialize all Interrupt control registers after reset.

## 2.5 System Controller Functions

### 2.5.1 Syscon Determination

The SCV64 samples BG3IN\* 500 ns after the  $\overline{\text{PWRRST}}$  pin is deasserted, and at every rising edge of SYSRST\*. If BG3IN\* is sampled low, the SCV64:

- becomes the Syscon,
- enables its Syscon functions, and
- sets the SYSC bit in the STAT1 register (Table A.27).

If BG3IN\* is high, the SCV64 does not become, or ceases to be, the Syscon, and clears the SYSC bit in the STAT1 register. If desired, the SYSC bit may be used to either enable or disable Syscon functions.



*Caution: Software designers should ensure that there is no activity on the VMEbus before enabling or disabling the Syscon through this bit.*

To ensure automatic Syscon determination, the board in slot 1 must have a 10K $\frac{3}{4}$  pull-down resistor from BG3IN\* to ground.

If the host board is not at the top of the bus grant daisy chain, then the preceding board in the chain drives the BGnIN\* signals. During system reset, the VMEbus specification requires that boards reset and drive their BGnOUT\* signals high. If the SCV64 host board is at the top of the bus grant daisy chain, the pull-down resistor will drive BG3IN\* low, ensuring that the SCV64 in slot 1 becomes Syscon.



*Caution: The SCV64 requires that the board preceding it in the bus grant daisy chain adhere to the VMEbus specification that all Bus Grant outputs be driven high during reset. If the preceding board is driving its BG3OUT<sub>T</sub> low at the end of reset, then more than one SCV64 could become system controller, causing a system malfunction.*



## 2.5.2 IACK Daisy Chain Driver

When the SCV64 is Syscon, it generates IACKOUT\* which is passed along the IACK daisy chain. When the interrupter receives IACKIN\* on the same level as it is currently requesting, it will respond to the interrupt acknowledge cycle and not propagate the falling edge down the IACK daisy chain. The interrupter places its STATUS/ID onto the data bus and terminates the IACK cycle with DTACK\*. Otherwise, the IACK daisy chain driver asserts IACKOUT\* and the next board receives its opportunity to respond to the interrupt acknowledgment.



*Note that due to the use of the IACK daisy chain driver in an Auto-ID system on the SCV64 (see “Auto-ID” on page 2-121), IACKOUT\* assertion will be delayed by an extra 5 clock cycles for the first level one interrupt after system reset.*

## 2.5.3 VMEbus Arbiter

The Arbiter module is enabled whenever the SCV64 is the system controller. There are four programmable arbitration modes as well as an arbitration time-out function.

### 2.5.3.1 Arbitration Modes

The SCV64 provides all arbitration modes defined in the VMEbus Specification. These are:

- Full Priority (PRI),
- Round Robin, and
- Single Level Arbiter (a subset of Full Priority).

In addition, two mixed priority modes are provided:

- priority level 3, and
- priority levels 2 and 3.

All arbitration modes are determined by the ARB1 and ARB0 bits in the VARB register (Table A.31, and below). Since arbiter logic continuously samples the VARB register, any changes to ARB1 and ARB0 will immediately be reflected in the current arbitration mode. The arbiter defaults to Full Round Robin mode after reset.

**Table 2.15 : Setting VMEbus Arbitration Modes**

Arbitration Mode	Setting for ARB1,0
full round robin	00
priority 3, round robin 2,1,0	01
priority 3, 2; round robin 1,0	10
full priority	11

### Full Priority Arbitration

In the Full Priority mode, the order of priority is BR3\*, BR2\*, BR1\*, BR0\*, as defined by the VMEbus specification. The arbiter samples these lines every 31.25 ns and issues a Bus Grant to the highest requesting level.

If a Bus Request of a higher priority than the current bus owner becomes asserted, the arbiter asserts BCLR\* until the owner releases the bus (BBSY\* released).

### Full Round Robin Arbitration

This mode arbitrates all levels in a round robin mode, repeatedly scanning from levels 3 to 0. Only one grant is issued per level and one owner is never forced from the bus in favour of another requester (BCLR\* is never asserted).

Since only one grant is issued per level on each round robin cycle, several scans will be required to service a queue of requests at one level. If each requester on a round robin level is in the Fair mode (see “Fair and Demand Modes” on page 2-9), then every board eventually receives bus ownership.

### Single Level Arbiter

A single level arbiter can be implemented by programming the arbiter to Full Priority mode, and then connecting all requesters to level 3. Leaving the arbiter in default Round Robin mode wastes time by scanning unused levels. There is no provision for masking off-bus request levels.

### **BR3\* Priority**

This version of mixed mode arbitration gives immediate priority to BR3\* while BR2\*, BR1\* and BR0\* are allocated to a round robin group.

The arbiter repeatedly scans levels 2, 1 and 0 in a circular fashion and if a request is detected on one of those levels, then a bus grant is issued. The scan continues only after the owner releases the bus by deasserting BBSY\*.

Round robin arbitration is superseded if a request on level 3 is made. With a level 3 request, BCLR\* is asserted to clear the bus of any lower priority owner. After all pending requests on level 3 have been granted, the scanning of the round robin group continues at the point of interruption. Note that multiple requests on level 3 may prevent levels 2 through 0 from obtaining the bus.

### **BR3\*, BR2\* Priority**

This mixed arbitration mode is similar to the BR3\* Priority mode, in that levels 1 and 0 are allocated to a round robin group, while requests at levels 3 or 2 are given priority. The arbiter scans levels 1 and 0, issuing one bus grant per level per scan. Round robin arbitration is interrupted if a request occurs at level 3 or 2. The arbiter asserts BCLR\* to clear the bus for the highest priority (level 3 or 2) request.

Level 3 has priority over all other levels, and all requests pending on that level are served first. Requests on level 2 are also all granted before round robin group scanning is resumed. Unless there is heavy bus traffic on levels 3 or 2, all requesters on levels 1 and 0 eventually gain access to the bus if they request in the Fair mode (see “Fair and Demand Modes” on page 2-9).

## **2.5.3.2 Arbitration Time-out**

Arbitration time-out ensures that bus arbitration will resume if BBSY\* is not asserted within 16  $\mu$ s of a bus grant signal. Time-out returns the arbiter to its previous mode and outstanding requests. Arbitration Time-out is enabled by default and is deactivated by clearing the ATEN bit in the VARB register (Table A.31).

### **2.5.3.3 Reset Effects**

Local reset does not disrupt arbiter operation, although the arbiter will revert to its default settings (the VARB register is re-initialized).

System reset and Power reset immediately cause deassertion of all arbiter signals.

Additional information concerning reset effects is provided in “Resets” on page 2-110.

## 2.5.4 Bus Timer

The SCV64 uses BERR\* to terminate VMEbus data transfers if a slave does not respond within a specific programmed time. The time-out period before BERR\* is driven low can be 16  $\mu$ s, 32  $\mu$ s, 64  $\mu$ s (the default setting), or Never, depending upon the value of the VXL1 and VXL0 bits in the VARB register (Table A.31). The Never setting should only be used during system debug, since the VMEbus cannot recover from errors (such as incorrect addressing) in this mode. Note also that the Auto-ID function depends on bus time-out.

The timer is held reset while DS1\* and DS0\* are high, and begins running when either data strobe becomes asserted. At time-out, the SCV64 asserts and maintains BERR\* until both DS1\* and DS0\* are high. Note that a race condition can occur if a slave takes a long time to respond and asserts DTACK\* coincidentally with the SCV64 assertion of BERR\*. Under these conditions, the master may see DTACK\* and BERR\* simultaneously. On-board logic may be required to respond to this conflict.

**Table 2.16 : Setting VMEbus Time-out Period**

Time-out Period	Setting for VXL1,0
Never	00
16 $\mu$ s	01
32 $\mu$ s	10
64 $\mu$ s (default)	11

## 2.5.5 System Clock Driver

When the SCV64 is the Syscon, the SYSCLK driver supplies a 16 MHz, 50% duty cycle clock, meeting VMEbus specifications. The SCV64 provides this clock before SYSRST\* deasserts at power-up. If two SCV64s exchange Syscon roles at a SYSRST\* deassertion, then an invalid level may occur for one or two clock periods while one SYSCLK driver is enabled and the other is disabled.

## 2.5.6 External Inputs

### 2.5.6.1 External Status

The BG2IN\* and BG1IN\* signals are available as user defined, off-board input status bits when the SCV64 is Syscon. The input from these two pins is read from the BG2IN and BG1IN bits in the STAT1 register (Table A.27).

The BG2IN and BG1IN bits indicate the true state of the BG2IN\* and BG1IN\* pins, which in turn may indicate different system modes defined by the user.



*Note: The BG2IN and BG1IN bits are not sampled and can change while being read.*

### 2.5.6.2 Off-Board Reset Input

When the SCV64 is Syscon, a low level on the BG0IN\* pin for more than 70 ms will initiate system reset. The SYSRST\* signal will remain asserted for 0.25 sec beyond the release of BG0IN\*.

Sensitivity to noise on the BG0IN\* line is reduced by a digital pulse filter. BG0IN\* is sampled every 14.3 ms, and four successive low samples must be recorded before SYSRST\* is driven low. The digital pulse filter introduces a latency period of 43 to 57 ms between BG0IN\* assertion and SYSRST\* assertion.



*Caution: The BG0IN\* pin does not provide a Schmitt trigger input circuit. Therefore, the pin must be pulled up with a resistor if it is unused and the SCV64 is in Slot 1.*

### 2.5.7 Reset Effects on Syscon Functions

Local reset may alter Arbiter function by resetting the VARB register to its default settings (see “VMEbus Arbiter” on page 2-31). However, the Arbiter is designed to accept mode changes during operation, so re-initialization of the VARB register will not generate any improper responses.

Similarly, arbitration time-out will be returned to its “enabled” default setting after local reset (see “Arbitration Time-out” on page 2-33). If the “disabled” option is required, the user will have to re-initialize the VARB register.

Since local reset re-initializes all registers to their default settings, the user can expect the data transfer timer to return a setting of 64  $\mu$ s. The counter will be cleared correctly when DS1\* and DS0\* are both high, but the reset may result in improper operation if the timer has just expired. If the timer has asserted BERR\* just before the reset (when programmed to 16 or 32  $\mu$ s), BERR\* is deasserted until the 64  $\mu$ s mark is reached, and then re-asserted.

## 2.6 Data Path

The SCV64 can transfer data across the local/VMEbus interface in either a coupled or decoupled mode. The coupled mode is the conventional VMEbus cycle (see Figure 2.7 below). Because mastership of three buses is maintained during a coupled transfer (master local bus, VMEbus, and slave local bus), two buses are idle while the third bus performs its cycle(s). This idle time constitutes a loss in effective bus bandwidth.

Decoupled mode uses FIFOs to buffer the local bus from the VMEbus. Data from the VMEbus is written to a 15 stage deep, 71-bit wide receive FIFO (RXFIFO). Once the incoming cycles are recorded and queued within the RXFIFO, the SCV64 asserts DTACK\* and ends the VMEbus cycle. Having freed the VMEbus for other operations, the RXFIFO then obtains the local bus and completes the data transfer. Note that in decoupled mode the SCV64 gives a VME slave response as close to the minimum permitted by the VME specification. The RXFIFO is reset using the RXRST bit in the DCSR register (Table A.6).

Data from the local bus is written to a 15 stage deep, 72-bit wide transmit FIFO (TXFIFO) in the SCV64. Once the outgoing cycles are recorded and queued within the TXFIFO, the SCV64 asserts  $\overline{\text{KDSACKx}}$  and ends the local cycle. Having freed the local bus for other operations, the TXFIFO obtains the VMEbus and completes the data transfer. This decoupled operation permits single wait state writes to the VMEbus. The TXFIFO is reset using the TXRST bit in the DCSR register (Table A.6).

For both types of FIFOs, different cycle types from different sources may be interleaved and are individually managed (e.g. CPU cycle interleaved with DMA cycles in the RXFIFO).

In both incoming (VME to local) and outgoing (local to VME) transfers, the FIFOs decouple the local and VMEbus by allowing the bus cycles to be sequential and independent. Decoupling the local and VMEbus in this manner maximizes VMEbus throughput (less time is lost waiting for completion of local bus cycles) and minimizes local bus loading.

Although incoming and outgoing transfers are similar, the function of the RXFIFO and TXFIFO differ in some respects. The detailed operation of data transfers in these two directions will be discussed separately below (“SCV64 as VME Slave” on page 2-39 and “SCV64 as VME Master” on page 2-41). DMA transfers, which are strictly decoupled, will be discussed separately, since DMA entails a different application of the receive and transmit FIFOs.

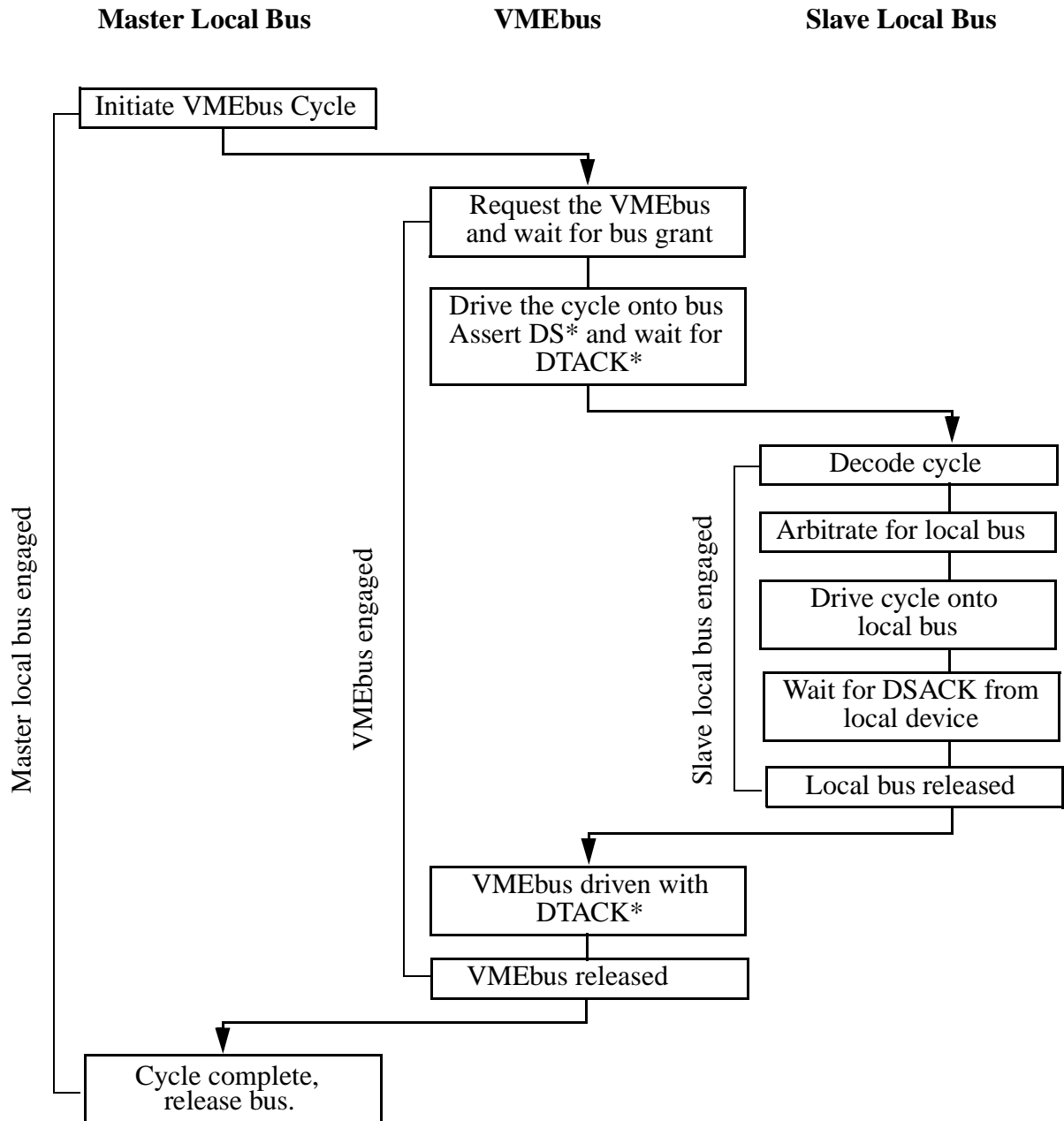
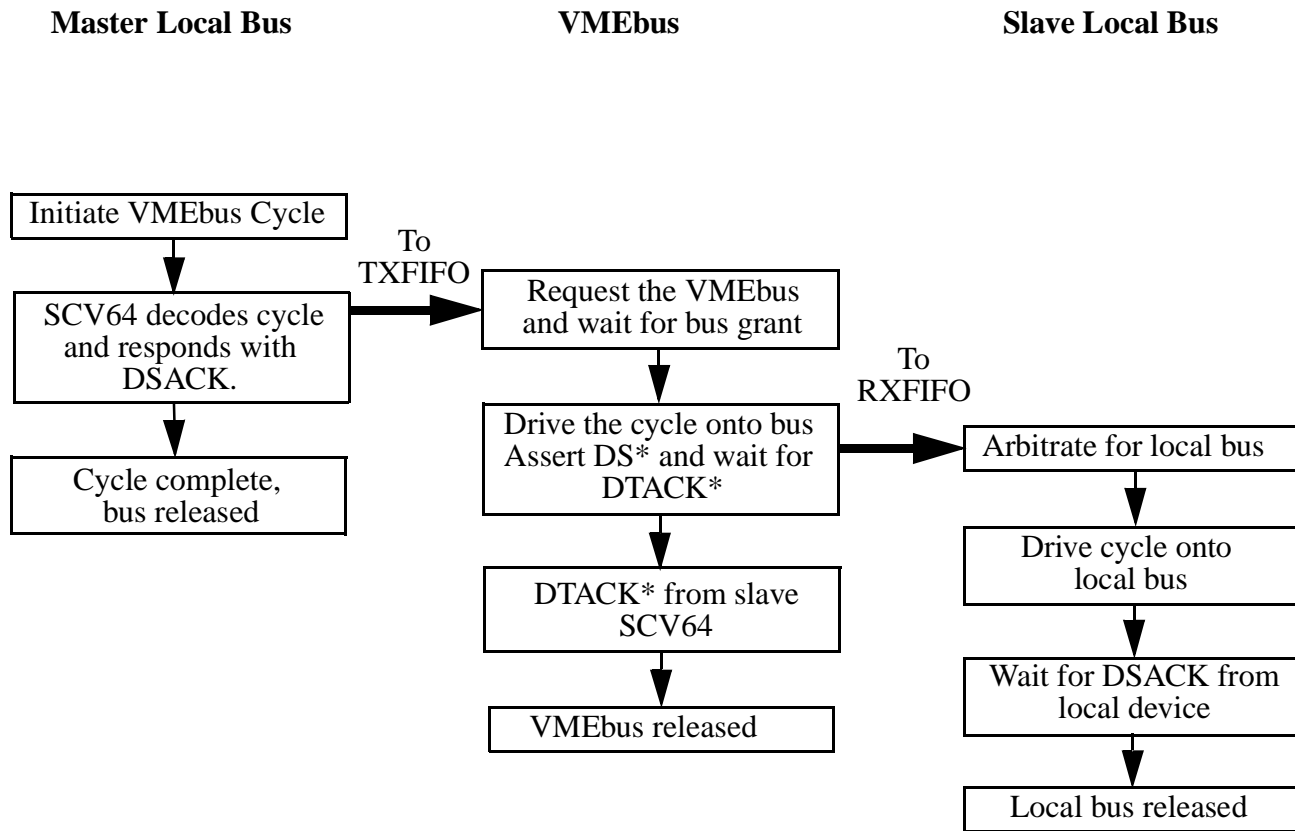


Figure 2.7 : Coupled Cycles on the VMEbus



**Figure 2.8 : Decoupled Cycles on the VMEbus**



## 2.6.1 SCV64 as VME Slave

### 2.6.1.1 Coupled Mode

Reads from a slave SCV64 are always coupled, although writes may be either coupled or decoupled. Coupled mode for VME slave write cycles is programmed by setting the RXATOM bit in the MODE register (Table A.21). If the bit is set while entries still exist in the RXFIFO, then the SCV64 completes all pending cycles in the RXFIFO before entering coupled mode. Clearing this bit returns the SCV64 to decoupled mode (its default setting).

While in coupled mode, the local bus is released after each local cycle. The exception is during MBLT slave reads in which the local bus is released every two local cycles. The SCV64 samples  $\overline{\text{KDSACK}}_n$  and  $\overline{\text{KBERR}}$  on the falling edge of the local clock and ends the cycle when  $\overline{\text{KDSACK}}_n$  or  $\overline{\text{KBERR}}$  are double sampled low. When the local cycle is terminated, the SCV64 asserts DTACK\* to end the VMEbus cycle.

If a slave read is initiated while there are entries in its RXFIFO, the cycles in the RXFIFO will be completed before the coupled read is allowed. Similarly, the SCV64 will complete all entries in its TXFIFO before initiating a coupled master read cycle. In both cases the master CPU will execute wait states until the FIFO is empty. Completing pending write cycles before allowing reads ensures sequential consistency; that is, the data being read is the most current.



*Caution: The RXATOM bit should not be modified by the local CPU while there is the possibility of an incoming VME slave cycle because this might prevent the SCV64 from decoding the cycle. In order to ensure that there is no incoming VME cycle, the user can program the SCV64 to obtain ownership of the VMEbus. Once the SCV64 has ownership of the VMEbus, the state of the RXATOM bit may be changed. (See page 2-22).*

### 2.6.1.2 Decoupled Mode

Writes to a slave SCV64 may be either coupled or decoupled. The SCV64 is programmed for decoupled mode by clearing the RXATOM bit in the MODE register (see Table A.21). The SCV64 defaults to decoupled mode. The RXATOM bit in the MODE register should not be modified by the local CPU while there is the possibility of an incoming VMEbus slave cycle since the SCV64 may not be able to decode the incoming cycle. If you need to modify the RXATOM bit when there is the possibility of an incoming VMEbus slave cycle, see the programming solution described on page A-24.

After the slave address decoder and protection logic validate the access, write cycles from the VMEbus are loaded into the RXFIFO and acknowledged with DTACK\* on the VMEbus. If the access is protected (see “Access Protection” on page 2-52), the SCV64 asserts BERR\*.

If the RXFIFO is full, the SCV64 does not accept writes until at least one is cleared from the RXFIFO, making space available. Note that the VMEbus data transfer rate is limited by the rate at which the slave local bus completes its cycles. Therefore, a full RXFIFO effectively couples the buses. If the RXFIFO cannot perform its writes on the local bus, the VMEbus transfer times out with a BERR\* signal.

Each 71-bit entry in the RXFIFO is made up of 3 components - control, address, and data. The 3 components for the most current or last completed cycle (the oldest entry in the RXFIFO) are read from:

- the RXFIFO Control register (RXCTL, Table A.13),
- the RXFIFO Address register (RXADDR, Table A.12), or
- the RXFIFO Data register (RXDATA, Table A.11).

The RXCTL register contains 6 control bits which qualify the RXFIFO entry. These control bits:

- identify the entry as the beginning of a block transfer,
- indicate whether the entry is a BLT or MBLT cycle,
- provide the size of the cycle (longword, byte, word, or triple byte), and
- give the local function code for non-privileged or supervisory program and data space.

The RXADDR register is 32 bits and gives the RXFIFO output address. During MBLT cycles, this register provides the top 32 bits of the 64-bit data transfer. The RXDATA register is also 32 bits wide and provides the RXFIFO output data.

The local bus is requested by the RXFIFO whenever entries appear. Arbitration of the local bus follows these priorities:

1.  $\overline{\text{LBRQ1}}$  (if SCV64 is not in default fair mode),
2. the RXFIFO, and
3. the DMA controller.

The RXFIFO is given priority over the DMA controller to ensure sequential consistency (the most current data is available).

Entries in the RXFIFO are transferred to their indicated address by single local cycles or burst mode. Using zero wait state memory on the local bus, the SCV64 can transfer one longword every 3 clock cycles (see “Local Bus Interface” on page 2-66).

The SCV64 can be programmed to perform burst double clock write cycles. With a 33 MHz local clock, burst writes allow data transfer rates up to 62 MBytes/sec. Burst writes are enabled by setting the FIFOBEN bit in the MODE register (Table A.21). The BLEN1 and BLEN0 bits in the MODE register set the maximum burst length (4, 8, 16, or 32 longwords). The default setting for maximum burst length is 4 longwords.

If the FIFOBEN bit is set, then the SCV64 will initiate burst writes when there are 2 sequential MBLT entries or 4 sequential BLT entries in the RXFIFO. If a sufficient number of BLT or MBLT entries are queued, the SCV64 initiates a burst write. The burst write is terminated if the SCV64:

- encounters a different entry type (e.g. a BLT entry after a series of MBLT entries),
- completes its maximum burst length, or
- empties the RXFIFO.

Unlike the TXFIFO, which is disabled by a VMEbus error, the RXFIFO is not disabled if a local bus error occurs. However, the RXFIFO can be disabled by setting the DISRX bit in the MODE register (Table A.21) (by default, the RXFIFO is enabled). If the RXFIFO is disabled, then it can only be emptied by using the RXSHFT bit in the DCSR register. Writing 1 to the RXSHFT bit discards the oldest entry and shifts the RXFIFO queue forward by one step. The VMEbus can still write to the RXFIFO if it is disabled, but doing so will set the LBERR bit in the DCSR register and cause  $\overline{\text{VMEINT}}$  to be asserted. If the RXFIFO fills while it is disabled, then the pending VME cycles will time out.

## 2.6.2 SCV64 as VME Master

### 2.6.2.1 Coupled Mode

Master read cycles are always coupled, although writes may be either coupled or decoupled. Coupled mode for VME master cycles is programmed by setting the TXATOM bit in the MODE register (Table A.21). If the bit is set while entries still exist in the TXFIFO, then the SCV64 completes all pending cycles in the TXFIFO before entering coupled mode. Clearing this bit returns the SCV64 to decoupled mode (its default setting).

In coupled mode, the CPU transfer is not queued by the TXFIFO but is directly linked to the VMEbus. The CPU enters wait states until a DTACK\* or BERR\* signal is generated by the VMEbus slave. The SCV64 copies the cycle termination through to the local master, which then ends the master local cycle. If BERR\* is received during the coupled cycle, the VBERR flag is set in the DCSR register (Table A.6), and  $\overline{\text{VMEINT}}$  is asserted.

If the SCV64 initiates a coupled slave read while there are still entries in its TXFIFO, the cycles in the TXFIFO will be completed before the coupled read is allowed. Similarly, the SCV64 will complete all entries in its RXFIFO before initiating a coupled master write cycle. In both cases the master CPU must execute wait states until the FIFO is empty. Completing pending cycles before allowing subsequent coupled cycles ensures sequential consistency; that is, the data being read is the most current.

### 2.6.2.2 Decoupled Mode

Writes from a master SCV64 to a slave may be either coupled or decoupled. The SCV64 is programmed for decoupled write cycles by clearing the TXATOM bit in the MODE register (see Table A.21). The SCV64 defaults to decoupled mode.

In decoupled mode, writes to the VMEbus are loaded into the TXFIFO. The TXFIFO queues the entries and asserts  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KDSACK0}}$  to terminate the master local cycle. In normal operation, the SCV64 requests the VMEbus as soon as entries appear in the TXFIFO.

However, while the device is in DMA mode, the SCV64 can be programmed to request the VMEbus only when the TXFIFO is filled. This option is programmed by setting the FILL bit in the MODE register (see Table A.21).



*Note that to ensure sequential consistency, read and IACK cycles are blocked until all TXFIFO entries are processed.*

If the TXFIFO is full, the SCV64 does not accept writes until an entry is unloaded to the VMEbus, making space available. Note that the master local bus data transfer rate is limited by the rate at which the slave bus completes its cycles. Therefore, a full TXFIFO effectively couples the buses. If the TXFIFO cannot perform its writes on the VMEbus, the master local bus transfer times out with a  $\overline{\text{KBERR}}$  signal. The user should consider the capabilities of slave buses and alternate between cycles for slower and faster buses.

Each 72-bit entry in the TXFIFO is made up of three components - control, address, and data. The three components for the most current or last completed cycle are read from the following three registers:

- the TXFIFO Control register (TXCTL, Table A.19),
- the TXFIFO Address register (TXADDR, Table A.18), and
- the TXFIFO Data register (TXDATA, Table A.17).

The TXCTL register contains 8 control bits which qualify the TXFIFO entry. These control bits:

- identify the cycle as MBLT or BLT,
- provide information about transfer size (longword, byte, word, or triple byte),

- identify the address space (A32, A24, A16),
- distinguish whether the cycle is supervisory or non-privileged, and
- identify the data type as program or data.

The TXADDR register is 32 bits and gives the TXFIFO output address. During MBLT cycles, this register provides the top 32 lines of the 64-bit data transfer. The TXDATA register is also 32 bits wide and provides the TXFIFO output data.

When a VMEbus BERR\* signal is received by the SCV64, it releases the bus, sets the VBERR bit in the DCSR register (Table A.6), asserts  $\overline{\text{VMEINT}}$  and negates BRn\*.

While the VBERR flag is set, entries may still be queued in the TXFIFO until it reaches a full condition. After the TXFIFO is full, any local cycles will time out on the local bus. The TXFIFO is unable to dequeue cycles to the VMEbus until the VBERR flag is cleared. The TXCTL, TXADDR and TXDATA registers in the SCV64 record the values that were present at the output stage of the TXFIFO when the error occurred. They may be read by the CPU to obtain the address, data and control codes in the failed write cycle. This information can be examined by software to determine the fault, or used to rerun the cycle. The failed cycle will not remain in the TXFIFO when the VBERR bit is cleared, and must be regenerated in order for it to be retried.

### 2.6.2.3 DMA Transfers

The SCV64 performs all master BLT and MBLT cycles through the DMA controller. Since master DMA cycles are always decoupled, the operation of the DMA controller is closely associated with the RXFIFO and TXFIFO (see “DMA Controller” on page 2-102).

In either DMA reads or DMA writes, the DMA controller always reads from one bus and writes to a FIFO. The fundamental differences between a DMA read and write depend upon,

- which bus (local or VME) is read from, and
- whether the DMA controller writes to a RXFIFO or TXFIFO.

The various addressing and data transfer modes used in DMA operations are selected using the MODE register (Table A.21). The required bit settings for the different modes are given in “Addressing and Data Transfer Modes” on page 2-104. The available address modes are A64, A32, and A24. All address modes can be combined with the two data modes, D32 and D16. However, only A64 and A32 may be combined with MBLT (D64).

Note that for A64 transfers, the MA64BAR register (Table A.23) is used for the upper 32 bits of the VME A64 address.

The SCV64 provides a burst mode which optimizes DMA operations involving BLT or MBLT cycles. For DMA reads the SCV64 can initiate burst write cycles from the RXFIFO to local memory (see page 2-41). For DMA writes, the DMA controller can initiate burst read cycles from local memory to the TXFIFO. All burst start addresses will be longword aligned for BLTs and double-longword aligned for MBLTs. In order to keep local address registers updated, burst cycles are automatically terminated and restarted at every burst-length boundary. For example, for a programmed burst of 8 longwords starting at address 0x08, the SCV64 will terminate and restart the burst at addresses 0x20. Bursts may terminate on any longword aligned address for BLTs and any double-longword aligned address for MBLTs.

Burst read cycles are terminated if:

- the maximum burst length is reached,
- the DMA transfer count is completed (see “Data Transfer Counts” on page 2-107),
- local bus ownership is granted to a higher priority request, or
- the TXFIFO fills.

If the TXFIFO fills, the SCV64 will release the local bus before any additional cycles are initiated. During DMA transfers, the local bus is automatically released when the TXFIFO contains 15 entries (the TXFIFO is 15 stages deep). During MBLT cycles, the local bus is automatically released when the TXFIFO contains 14 entries (since each MBLT cycle requires two local bus cycles).

## 2.7 Memory Mapping

There are two types of memory maps involved in the VME-local bus interface. One memory map, the CPU memory map, is from the perspective of the local bus looking onto local and VME addresses. The other memory map, the VME slave image, is from the perspective of the VMEbus looking into the local address space. This section describes the properties and function of these two memory maps: the CPU memory map, and the VME slave image memory map.

### 2.7.1 CPU Memory Map

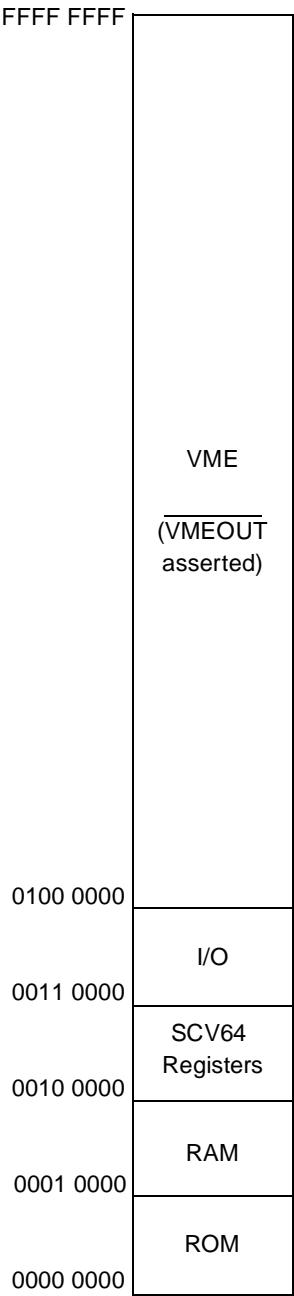
Local devices and memory are accessed through the CPU memory map with a local address decoder and chip select signals (see Figure 2.9 below for a sample CPU memory map). VME address space is also allocated a position within the CPU memory map. VME address space is defined by a local address decoder asserting the  $\overline{\text{VMEOUT}}$  signal. If a VME space is accessed, a local address decoder asserts  $\overline{\text{VMEOUT}}$  which signals the SCV64 to initiate a VMEbus cycle or queue the cycle in the TXFIFO. The SCV64 subdivides the CPU memory map with a VME access overlay (see Figure 2.10 on page 2-47). This VME access overlay is divided into 32 pages, 128 MBytes per page and defines the accesses to various VME address and data spaces.

Pages 0 to 30 are by default defined as A32:D32/D16/D08 (referred to as A32:D32 for the sake of brevity, see Figure 2.10 on page 2-47). Therefore, any VME address space accessed on these pages of the CPU memory map will be automatically accessed as A32:D32.

Similarly, A24 VME accesses must be made in the lower 32 MBytes of page 31 in the CPU memory map (see Figure 2.10). By default, any VME address space assigned to this region of the CPU memory map will be accessed as A24. Access to an address with the lower 16 MByte portion of the A24 region results in an A24:D16 VMEbus cycle. Accesses to the upper 16 MByte portion of the A24 region results in A24:D32 VMEbus cycles. The A24 VME address space can be disabled by setting the A24DI bit in the MODE register (Table A.21). Alternatively, the VME A24 space can be moved to the lower 32 MBytes of page 0 in the CPU memory map. Moving the A24 region to page 0 converts the default A24 region to A32:D32 VME space (see Figure 2.10).

A16:D16 access can be made only to VME address space located in the top 64 KBytes of the CPU memory map (see Figure 2.10). Any VME address space assigned to this region (using the  $\overline{\text{VMEOUT}}$  signal) will automatically be accessed as A16:D16. The A16:D16 region can be disabled by setting the A16DI bit in the MODE register (Table A.21).

Figure 2.9 shows the effect of the SCV64 VME overlay on the sample CPU memory map previously shown in Figure 2.9.



**Figure 2.9 : Sample CPU Memory Map as Determined by Local Address Decoder**



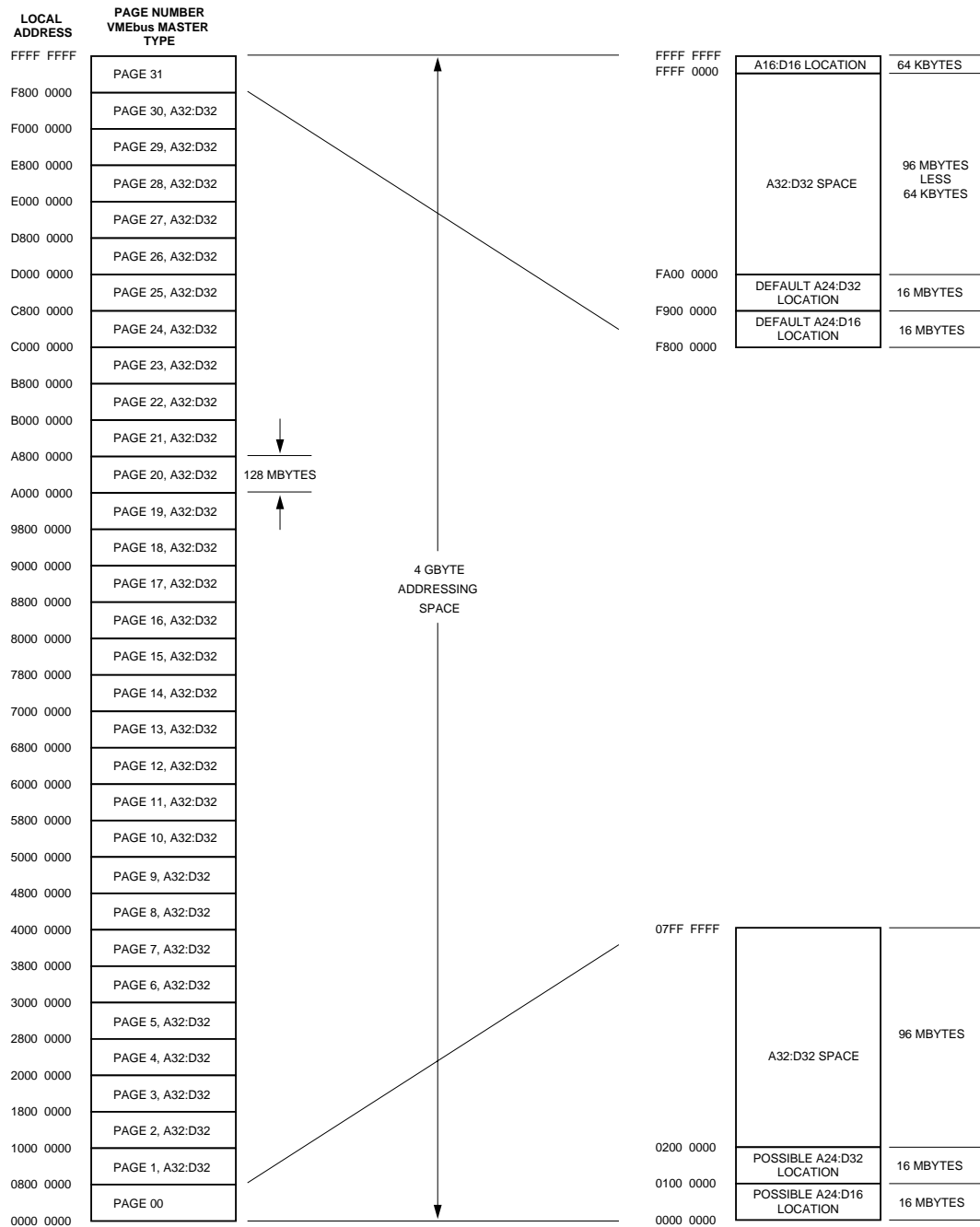


Figure 2.10 : SCV64 VME Access Overlay for the CPU Memory Map

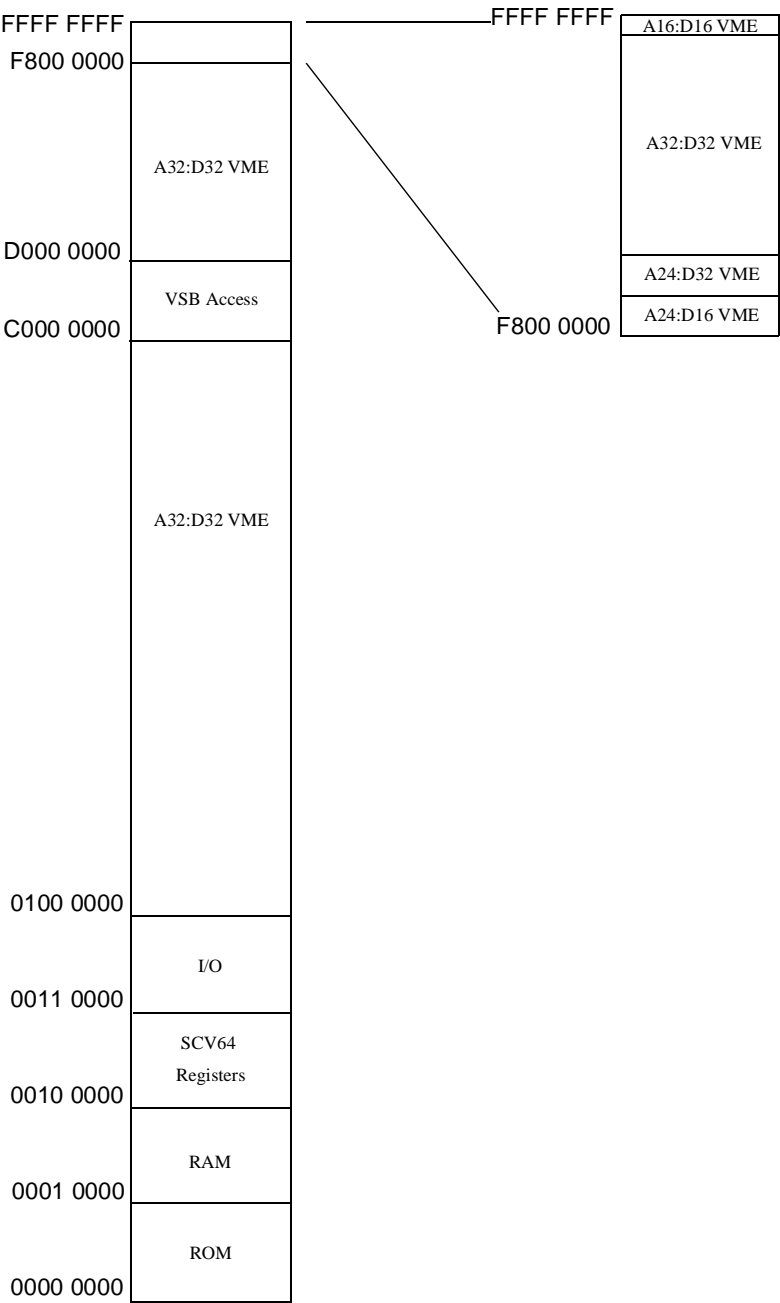


Figure 2.11 : Sample CPU Memory Map with SCV64 VME and VSB Space Overlay

Data transfers using A64 mode must be made with the DMA controller. Note that all DMA addressing modes are entirely independent of the CPU memory map and must be determined within the MODE register. A64 mode is initiated by setting the DMAA64 bit in the MODE register (Table A.21, see “DMA Controller” on page 2-102). Data modes for DMA transfer are also programmed using the MODE register (see Table 2.27 on page 2-105 for a full listing of address and data modes for DMA).

There are two special conditions under which the SCV64 will not generate a VMEbus cycle when  $\overline{\text{VMEOUT}}$  is asserted.

1. **If the CPU accesses VSB address space:** The SCV64 uses the same 32 page map described above to define VSB space in the CPU memory map. Any of the 32 pages can be designated VSB space using the BUSSEL register (Table A.14), where each of the 32 bits in the BUSSEL register corresponds to one of the 32 pages overlaying the CPU memory map. For example, setting bits 24 and 25 in the BUSSEL register designates pages 24 and 25 (256 MBytes) of the CPU's memory map as VSB space (Figure 2.9 on page 2-46). If the CPU accesses VSBbus address space, the local address decoder asserts  $\overline{\text{VMEOUT}}$  and the SCV64 (after decoding the address) asserts  $\overline{\text{VSBSEL}}$ . An external address decoder will use the  $\overline{\text{VSBSEL}}$  signal to initiate a transfer to the VSBbus. Note that no VMEbus or FIFO activity is caused by this access.
2. **If the CPU accesses its own image in VME address space:** If the CPU attempts access to its own VME address range, a local address decoder still asserts  $\overline{\text{VMEOUT}}$ , but the SCV64 decodes the address as being the board's own VME address space and asserts  $\overline{\text{RAMSEL}}$ .  $\overline{\text{RAMSEL}}$  is used by a local address decoder to select a set of addresses for local devices and memory accessible from the VMEbus (this set of addresses is defined as the VME slave image). Note that no VMEbus or FIFO activity is caused by this access.

### 2.7.2 VME Slave Memory Map

The address range of the local bus from the perspective of the VMEbus is termed the board's VME slave image. The SCV64  $\overline{\text{RAMSEL}}$  signal is used by the local address decoder to select a set of addresses for local devices and memory which are accessible from the VMEbus.  $\overline{\text{RAMSEL}}$  is also asserted during DMA transfers when accessing the local bus, and during CPU accesses to its own VME slave image.

The VME slave image memory map may be accessed as either A24 or A32 (see Figure 2.12 below), but not A16. The base addresses for the A24 and A32 slave images are programmed using the VMEBAR register (Table A.7).

The A32 image base address can be programmed to any 128 MByte boundary and to any size from 4 KBytes to 128 MBytes. The A24 image base address can be programmed to any half megabyte boundary or multiple of its programmed size (512 KB, 1, 2 or 4 MBytes), whichever is larger, within the 16 MByte A24 addressing space. For example, if the A24 slave image is 4 MBytes, then its possible base addresses are 0x00 0000, 0x40 0000, 0x80 0000, or 0xC0 0000.

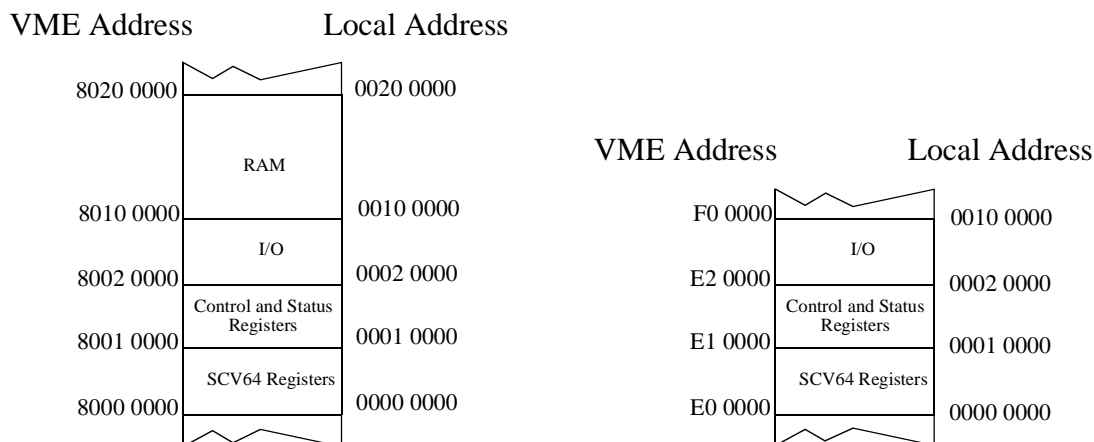
The upper 32 bits of the A64 base address are set using the SA64BAR register (Table A.22). The SCV64 will set the A64BARDY bit in the DCSR register (Table A.6) when both the master and slave A64 base addresses have been programmed.

On the local bus, the SCV64 drives the low address lines relevant to the programmed size of the VME slave image to a state matching the low address bits used as the VME address. KADDR31 - KADDR27 are driven to an undefined state, while all other address lines are driven low. For example, if the slave image size is programmed for 2 MBytes, the SCV64 drives KADDR20-KADDR0 to match the bits on the VMEbus; KADDR26-KADDR21 will be zero, while KADDR31-KADDR27 will be undefined. With this in mind, the local address decoding logic should be designed in such a way that it ignores the upper five address lines while the SCV64 is driving the local address. A common way to implement this is to qualify the local address lines into the decoder with the RAMSEL signal.

In A64 accesses, the upper 32 bits of the A64 base address are not carried onto the local bus, and the lower 32 bits are handled like any A32 access.



*Note that when SCV64 registers are accessed from the VMEbus, the SCV64 treats itself as it would any other local device. The local bus is requested and data is transferred through a local cycle to the appropriate register address.*



### A32 Memory Map

A32 Base Address = 0x8000 0000

A32 Size = 2 MBytes

### A24 Memory Map

A24 Base Address = 0xE0 0000

A24 Size = 1 MByte

Programming for (Table A.7) would be = 0x003C 0130

**Figure 2.12 : Sample VME Slave Memory Map (Accessed as A32 and A24)**

### 2.7.2.1 Automatic Base Address Programming

A slave boot mode (AUTOBAR) is available which facilitates design when no local intelligence is available to initialize the SCV64 base address. The AUTOBAR mode allows the SCV64 to power up with its slave image accessible from the VMEbus, facilitating the design of slave-only cards.

On the rising edge of  $\overline{\text{PWRRST}}$ , the VME base address register (see Table A.7 for a description of the base address register) is loaded with the current local bus value. Setting KFC2 high and KFC0 low during a power on reset sets the enabling bits that allow slave images to be defined (see Table 2.17). By use of an external address decoder, the VMEbus can then access SCV64 registers on power up.

The AUTOBAR mode is indicated by a 1 value in the AUTOBAR bit in the MISC register.

**Table 2.17 : Functions for Different Settings of KFC2 and KFC0**

KFC2	KFC0	Function
0	x	Reserved
1	1	VINEN and A24SLVEN cleared
1	0	AUTOBAR MODE: Set VINEN and A24SLVEN



*Note: SYSRST\* does not return the initial base address programmed at power-up. Instead, the initial base address will remain at the address given prior to the system reset with both A24 and A32 images enabled.*

### 2.7.2.2 Access Protection

Access protection allows the user to mask specific ranges of its VME slave image from the VME masters. The type of access protection is specified using the PROT bit in the MODE register (Table A.21). The setting defaults to write protection; setting the bit programs read and write protection. The lower 4 bits in the APB register are used to specify the extent of slave image protection (values between 16 KBytes to the entire 128 MBytes are available), where the protected range always begins at the base address and proceeds upwards. The APB register defaults to 0 after reset (no protection).

If access to a protected area is attempted from the VMEbus, the SCV64 generates a BERR\* signal. Protection is not applied to accesses by the local CPU to its own slave images. If write protection has been programmed and a read-modify-write operation is attempted on the protected memory, the RMW read cycles complete successfully but the write cycles are terminated with BERR\*.

Protecting the entire 128 MByte slave image will still leave the location monitor unprotected. The location monitor (see “Location Monitor and LMFIFO” on page 2-101) resides in the upper longword (or even word of the upper longword for 16 bit accesses) of the slave image and is used in interprocess communication and for BI-mode release (see “BI-Mode” on page 2-119).

## 2.8 VMEbus Interface

The SCV64 offers a flexible high performance master/slave VMEbus interface adaptable to a variety of applications. As discussed elsewhere (see “Data Path” on page 2-36), data can be transferred using either a coupled or decoupled architecture, where decoupling the VMEbus and local bus optimizes bus bandwidth usage. Although the coupling or decoupling of bus cycles greatly affects data transfer rates, it has no effect on the translation of address and control information between the VMEbus and local bus. However, the translation process does vary depending upon whether the transfer is a master cycle (SCV64 is the VME master) or a slave cycle (SCV64 is the VME slave). This section describes the translation process between VMEbus cycles and local bus cycles when the SCV64 is the VME master or VME slave.

### 2.8.1 SCV64 as VME Master

As VME master, the SCV64 generates read, write and read-modify-write cycles (but does not generate address only cycles). Outgoing cycles pass through external buffers before reaching the VMEbus. The SCV64 controls the direction of these buffers through the following signals:

- VADDROUT (for address lines A31-A00 and VLWORD\*),
- VDATAOUT (for data lines VDATA31-VDATA00), and
- VSTRBOUT (for  $\overline{VAS}$ ,  $\overline{VDS1}$ ,  $\overline{VDS0}$ ,  $\overline{VWR}$ , and VAM5-VAM0).

The buffers are placed in transmit configuration (SCV64 is driving VMEbus) when these signals are driven high. By default, the SCV64 sets the buffers to receive (VMEbus to local bus). Note that the direction of the data buffers will change depending upon whether the cycle is a read or write (transmit for a write and receive for a read, assuming the SCV64 is VME master).

#### 2.8.1.1 Address Translation

In transfers not involving the DMA controller, the SCV64 uses the VME address range in the CPU memory map to determine address and data modes (see “CPU Memory Map” on page 2-45). Local function codes (KFC2-KFC0) are used to specify the address space (supervisory or non-privileged, program or data). Based upon the address and function codes, the SCV64 will generate the appropriate address modifier code on the VMEbus. The 32 address lines on the local bus are mapped directly to the 31 VME address lines. The mapping between function codes, VME address space and AM codes is given in Appendix-D.

BLT and MBLT master cycles must be performed using DMA transfers. The address size and privilege type for DMA transfers is specified in the MODE register (Table A.21, see “Addressing and Data Transfer Modes” on page 2-104). For A64 master cycles (which can only be performed with DMA), the MA64BAR register (Table A.23) is used to program the top longword of the master A64 address.

The VME specification limits BLT cycles to 256 Bytes and MBLT cycles to transfer sizes up to 2 KBytes. For transfers larger than the specified limit, the SCV64 will automatically insert a new address at transfer size boundaries.

### 2.8.1.2 Byte Lane Translation

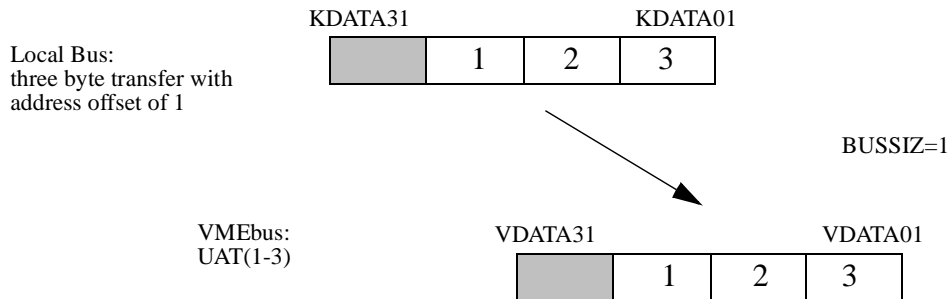
The control signals on the VMEbus relevant to data sizing (DS1\*, DS0\*, A01, LWORD\*, and A02) indicate both data transfer size as well as the byte lanes used in the transfer. Ultimately, this information specifies the address offset at which data is written to or read from. A BYTE(n) convention is used in the VMEbus specification to indicate the position of the data in memory, where ‘n’ is the address offset from a longword boundary. In the SCV64 local interface, information about data transfer size and address offset is provided with 4 bits, KSIZE1, KSIZE0, KADDR00, and KADDR01.

During a read cycle with the SCV64 as VME master, the SCV64 decodes the VMEbus byte lanes and translates them to specific byte lanes on the local bus. Byte translation depends upon the SWAP bit setting in the MODE register, the transfer size and address offset. When SWAP is set, then the byte lanes are mapped directly from the VMEbus to the local bus. If SWAP is cleared then byte lane translation follows the protocol specified in Appendix-D. For example, a single byte transfer with no address offset on VMEbus data lines VDATA15-VDATA08 will be translated to local bus data lines KDATA31-KDATA24.

During a write cycle with the SCV64 as VME master, there are two possible byte lane translation options depending upon the setting of the SWAP bit in the MODE register (Table A.21). When the SWAP bit is cleared, byte lane translation follows the protocol specified in Appendix-D. When the SWAP bit is set, then byte lanes are directly mapped from the local to the VMEbus: that is, data lines KDATA31-24 are mapped to VDATA31-24, KDATA23-16 are mapped to VDATA23-16, etc. (see Appendix-D).

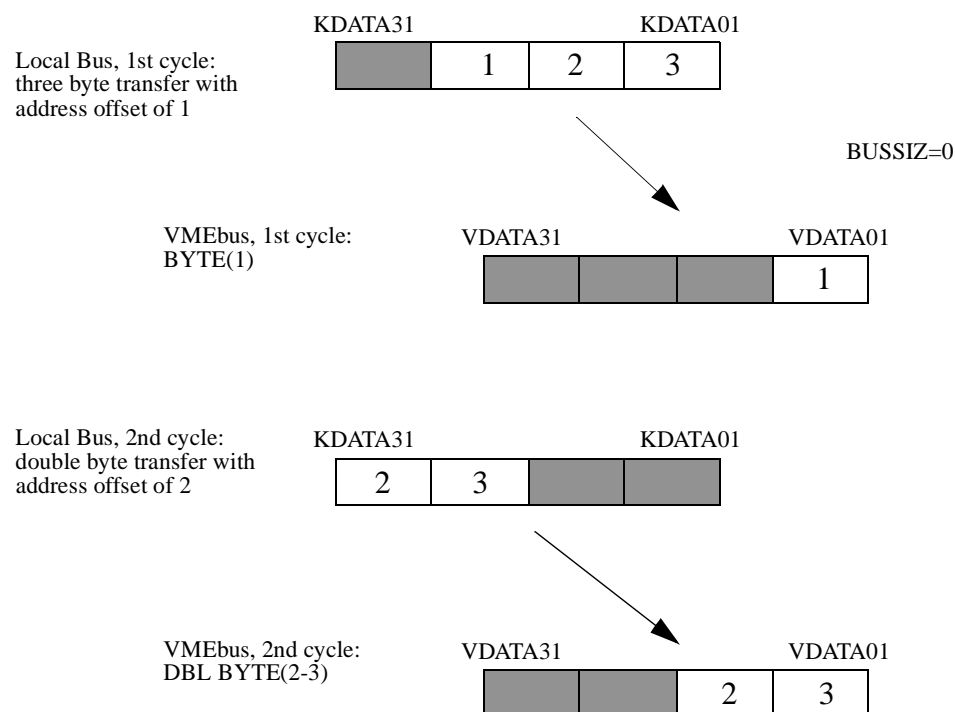
The SCV64 accepts unaligned transfers from the VMEbus, but will only generate unaligned transfers if the BUSSIZ bit in the MODE register (Table A.21) is set (see Appendix-D). When the BUSSIZ bit is set, the SCV64 always responds to the CPU as a 32-bit port and will use all 32 VMEbus data lines for unaligned transfers on the VMEbus. For example, if the CPU sends the SCV64 a three byte transfer with an address offset of 1, the SCV64 responds as a 32-bit port and generates a UAT(1-3) transfer on the VMEbus (see Figure 2.13). Since the SCV64 responds as a 32-bit port and all data is accepted from the local bus, the CPU is not required to perform dynamic bus sizing.





**Figure 2.13 : Local to VME Byte Lane Translation with BUSSIZ=1**

If the BUSSIZ bit is cleared, the SCV64 may respond to the CPU as a 16-bit port, depending upon the VME access (see “Memory Mapping” on page 2-45). Under this configuration, the CPU will need to dynamically bus size in order to complete unaligned transfers. Revisiting the example above, the CPU initiates a three byte transfer with address offset of 1, but in this case the device responds as a 16-bit port. In the first cycle, the SCV64 generates a single byte transfer on the VMEbus (transferring the top byte of the three byte transfer) and responds to the CPU as a 16-bit port. The CPU must then send the remaining 2 bytes with an address offset of two (the original offset of 1 plus the byte already sent). The SCV64 translates this second local cycle to a VMEbus DBL BYTE(2-3) transfer (see Figure 2.14). Note that it is the dynamic bus sizing capacity of the CPU that allows the SCV64 to accept unaligned transfers from the local bus and send aligned transfers on the VMEbus.



**Figure 2.14 : Local to VME Byte Lane Translation with  $BUSSIZ=0$**

### 2.8.1.3 VMEbus Mastership

While in coupled mode, the local bus will only be released by the CPU when the VMEbus is released (see Figure 2.7 on page 2-37) by the SCV64. In decoupled operation, the SCV64 requests the VMEbus as soon as entries appear in the TXFIFO. However, while the device is in DMA mode, the SCV64 can be programmed to request the VMEbus only when the TXFIFO is filled. This option is programmed by setting the FILL bit in the MODE register (see Table A.21).

Ownership and release of the VMEbus by the SCV64 will depend upon whether the SCV64 is set as ROR or RWD (see “VMEbus Requester” on page 2-8). If it is set as ROR, then the SCV64 will only release BBSY\* if another request is pending. In RWD mode, the SCV64 only releases the VMEbus when:

- there are no further entries in the TXFIFO,
- the coupled cycle is complete, or
- the IACK cycle is complete.

VMEbus ownership can be throttled using the bus ownership timer (see “Ownership Timer” on page 2-11). The ownership timer (enable by default) can be set for 0, 2, 4, or 8 $\mu$ s (the default setting). The SCV64 will give up ownership of the VMEbus within 1 or 2 cycles after the ownership timer expires (provided it is enabled).

VMEbus release modes can also be programmed through the NOREL bit in the MODE register (Table A.21). Setting this bit causes the SCV64 to not release the bus unless otherwise instructed by the arbiter (see “VMEbus Arbiter” on page 2-31). If the NOREL bit is cleared, the SCV64 will only release the VMEbus when no entries remain in the TXFIFO. The bus ownership timer can be used to control the duration of No Release mode.



*Caution: If the TXFIFO is set for No Release and the ownership timer is disabled, then the SCV64 will only release the bus if it receives BCLR\* and BCLR\* is enabled.*

The VMEbus will also be released under these conditions:

- $\overline{\text{L7IMEM}}$  interrupt,
- BI-mode, or
- local or system reset (see “Other Bus Release Mechanisms” on page 2-11).

#### 2.8.1.4 RMW Cycles

The VME read-modify-write cycle when SCV64 is VME master can be configured in two ways. In one configuration, the assertion of  $\overline{\text{KRMC}}$  on the local bus causes the SCV64 to hold AS\* ( $\overline{\text{VAS}}$ ) low on the VMEbus. Setting the TASCN bit in the MODE register (Table A.21) programs the SCV64 to hold  $\overline{\text{VAS}}$  asserted when the CPU asserts  $\overline{\text{KRMC}}$ .

In the other RMW configuration, the VMEbus RETRY\* line is used as a proprietary VMEbus RMC signal (this is programmed by clearing the RMCPIN bit in the MODE register). When the CPU asserts  $\overline{\text{KRMC}}$ , the SCV64 asserts RETRY\*/ $\overline{\text{VRMC}}$ . If the slave is programmed to accept RETRY\*/ $\overline{\text{VRMC}}$  as an RMW signal, then the slave local bus and the VMEbus will be locked until the RMW cycle is complete and RETRY\*/ $\overline{\text{VRMC}}$  is released.  $\overline{\text{KRMC}}$  will be asserted on the slave local bus during this cycle.

As a VME bus slave, VME read-modify-write (RMW) cycles prevent re-arbitration of the SCV64's local bus until  $AS^*$  is negated. This effectively locks the local resource until the VME bus RMW cycle is complete.

Note that when  $RETRY^*$  is used as a proprietary RMW signal, the  $AS^*$  line is free for multiple addressing during the RMW cycle. However, if  $AS^*$  is used to indicate RMW, then only single address RMW cycles may be performed.

### 2.8.1.5 Termination of a Master Cycle with $RETRY^*$

If the SCV64 (as VME master) receives  $RETRY^*$  and  $BERR^*$  from the VME slave during a coupled cycle, it translates the error signal to  $\overline{KBERR}$  on the local bus, asserts  $\overline{VMEINT}$  and sets the VBERR and RETRY bits in the DCSR register (Table A.6).

If the SCV64 receives  $RETRY^*$  and  $BERR^*$  during a decoupled cycle, then it asserts  $\overline{VMEINT}$  and sets the VBERR and RETRY bits, but does not assert  $\overline{KBERR}$ . The TXFIFO will be locked until the CPU clears the VBERR bit. Note that the  $RETRY^*$  line is only an input signal.

## 2.8.2 SCV64 as VME Slave

As VME slave, the SCV64 receives read, write and read-modify-write cycles (but does not respond to address only cycles). Incoming cycles pass through external buffers before reaching the SCV64. The SCV64 controls the direction of these buffers through the following signals:

- VADDROUT (for address lines A31-A00) and VLWORD\*,
- VDATAOUT (for data lines VDATA31-VDATA00), and
- VSTRBOUT (for  $\overline{VAS}$ ,  $\overline{VDS1}$ ,  $\overline{VDS0}$ ,  $\overline{VWR}$ , and VAM5-VAM0).

The buffers are placed in receive configuration (buffers are driving SCV64 pins) when these signals are driven low. By default, the SCV64 sets the buffers to receive. Note that the direction of the data buffers will change depending upon whether the cycle is a read or write (transmit for a read and receive for a write, assuming the SCV64 is VME slave).

### 2.8.2.1 Address Translation

During A32 accesses, the SCV64 drives the low address lines on the local bus relevant to the programmed size of the VME slave image to a state matching the low address bits used as the VME address. KADDR31-KADDR27 are driven to an undefined state, while all other address lines are driven low. For example, if the slave image size is programmed for 1 MByte, the SCV64 will drive KADDR20-KADDR0 to match the bits on the VMEbus; KADDR26-KADDR21 will be zero, while KADDR31-KADDR27 will be undefined. During A24 accesses, the SCV64 drives KADDR31-KADDR27 undefined, at least KADDR26-

KADDR22 low depending on the programmed A24 slave image size, and the remaining address bits straight through from the VMEbus. Because the top two bits of the A24 image (KADDR23 and KADDR22) are always driven low, the A32 and A24 slave images will share common local addresses in the lower 16 MBytes of the A32 image (within this range there are no bits available to distinguish between an A24 or A32 address). See “VME Slave Memory Map” on page 2-49 for more information about A32 and A24 slave images.

The SCV64 monitors the AM codes to determine the VME cycle’s address mode and transfer type (supervisory or non-privileged, data or program). The address size is not translated into an equivalent message on the local bus, but the SCV64 does translate the transfer type into local function codes (KFC2-KFC0). A complete translation table for AM codes to local function codes is provided in Appendix-D.

### 2.8.2.2 Byte Lane Translation

The control signals on the VMEbus relevant to data sizing (DS1\*, DS0\*, A01, LWORD\*, and A02) indicate both data transfer size as well as the byte lanes used in the transfer. Ultimately, this information specifies the address offset at which data is written to or read from. A BYTE(n) convention is used in the VMEbus specification to indicate the position of the data in memory, where ‘n’ is the address offset from a longword boundary. In the SCV64 local interface, information about data transfer size and address offset is provided with 4 bits, KSIZ1, KSIZ0, KADDR00, and KADDR01.

During a write cycle with the SCV64 as VME slave, the SCV64 decodes the VME byte lanes and translates them to specific byte lanes on the local bus. Byte translation depends upon the transfer size and address offset. For example, a single byte transfer with no address offset on VMEbus data lines VDATA15-VDATA08 will be translated to local bus data lines KDATA31-KDATA24 (see Appendix-D).

During a read cycle with the SCV64 as VME slave, byte lane translation is from the local bus to the VMEbus. The SCV64 will expect the data to be presented on the appropriate local data lanes as determined by the transfer size and address offset. The setting of the SWAP bit does not affect the byte lane translation when the SCV64 is the VME slave. See Table D.8 for further information on byte lane translation when the SCV64 is the VME slave.

The SCV64 accepts unaligned transfers from the VMEbus, but will only generate unaligned transfers if the BUSSIZ bit in the MODE register (Table A.21) is set (see Appendix-D). When the BUSSIZ bit is set, the SCV64 always responds to the CPU as a 32-bit port and will use all 32 VMEbus data lines for unaligned transfers on the VMEbus. For example, if the CPU sends the SCV64 a three byte transfer with an address offset of 1, the SCV64 responds as a 32-bit port and generates a UAT(1-3) transfer on the VMEbus (see Figure 2.13 on page 2-55). Since the SCV64 responds as a 32-bit port and all data is accepted from the local bus, the CPU is not required to perform dynamic bus sizing.

If the BUSSIZ bit is cleared, the SCV64 may respond to the CPU as a 16-bit port, depending upon the VME access (see “Memory Mapping” on page 2-45). Under this configuration, the CPU will need to dynamically bus size in order to complete unaligned transfers. Revisiting the example above, the CPU initiates a three byte transfer with address offset of 1, but in this case the device responds as a 16-bit port. In the first cycle, the SCV64 generates a single byte transfer on the VMEbus (transferring the top byte of the three byte transfer) and responds to the CPU as a 16-bit port. The CPU must then use dynamic bus sizing to send the remaining 2 bytes with an address offset of two (the original offset of 1 plus the byte already sent). The SCV64 translates this second local cycle to a VMEbus DBL BYTE(2-3) transfer (see Figure 2.14 on page 2-56).

### 2.8.2.3 Local Bus Mastership

When the SCV64 receives and decodes an incoming VME cycle, it requests and obtains the local bus using its local bus arbitration protocol (see “Local Bus Arbitration” on page 2-66). The means for obtaining the local bus does not vary with the type of incoming cycle. However, the type of incoming cycle does affect bus release.

During coupled read and write cycles, the SCV64 will release the bus after each single cycle. The exception to this is MBLT reads or writes, where the SCV64 will only release the bus after every 2 cycles (in order to transfer the necessary 64 bits of data).

During decoupled cycles, the local bus is released only after the RXFIFO is emptied or local bus grant is removed. In addition, burst mode can be initiated during decoupled BLT and MBLT cycles. If the RXFIFO has 2 consecutive MBLT entries, or 4 consecutive BLT entries, then the SCV64 will initiate a local burst write. The local burst write only stops when it:

- encounters a different entry type (e.g. a BLT entry following an MBLT entry),
- completes its maximum burst length (see “SCV64 as VME Slave” on page 2-39), or
- empties the RXFIFO.

Local bus mastership is controlled differently during read-modify-write (RMW) cycles compared to normal read and write cycles. There are two types of RMW slave cycles. In one type, holding AS\* low locks the VMEbus and local bus until AS\* is deasserted.  $\overline{\text{KBRQ}}$  is held asserted on the local bus as long as AS\* is held asserted, thus disabling re-arbitration of the local bus.

In the other type of RMW cycle, the VMEbus RETRY\* signal is used as a proprietary VMEbus RMC signal (this is programmed by setting the RMCPIN bit in the MODE register). If RETRY\*/ $\overline{\text{VRMC}}$  becomes asserted in this configuration, then the SCV64 asserts  $\overline{\text{KRM C}}$  to indicate an RMW cycle on the local bus. Under these conditions, the local bus is only released when RETRY\*/ $\overline{\text{VRMC}}$  is deasserted.

Note that when  $\overline{\text{RETRY}}^*$  is used as a proprietary RMW signal, the  $\text{AS}^*$  line is free for multiple addressing during the RMW cycle. However, if  $\text{AS}^*$  is used to indicate RMW, then only single address RMW cycles will be compliant with VME specifications.

### 2.8.3 DMA Transfers

During DMA transfers, the SCV64 is both the VME and local master. The DMA Local Address Register (DMALAR, Table A.3) is used for the source address in DMA writes, and the DMA VME Address Register (DMAVAR, Table A.4) is for the source address in DMA reads (see Figure 2.30). Data from the source address is then transferred to the RXFIFO in the case of a DMA read, and to the TXFIFO in the case of a DMA write. Once the DMA cycle is entered in either of the FIFOs, it is managed independently of other cycles queued in the same FIFO.

The various addressing and data transfer modes used in DMA operations are selected using the MODE register (Table A.21). The required bit settings for the different modes are given in Table 2.27 on page 105. The available address modes are A64, A32, and A24. All address modes can be combined with the two data modes, D32 and D16. However, only A64 and A32 may be combined with MBLT (D64).

Note that for A64 transfers, the MA64BAR register (Table A.23) is used for the upper 32 bits of the destination address for a DMA read, or the upper 32 bits of the source address for a DMA write.

On the local side, the SCV64 can initiate burst reads and writes which will optimize data transfer during BLT and MBLT cycles. In a DMA read, the SCV64 will initiate a local burst write if the RXFIFO contains 2 consecutive MBLT entries or 4 consecutive BLT entries. For BLT or MBLT writes, the DMA controller will initiate local burst reads from local memory. For more information on local burst mode, see “DMA Controller” on page 2-102 and “Burst Cycles” on page 2-89.

### 2.8.4 Master/Slave Deadlock Resolution

If the SCV64 is waiting for VMEbus ownership in order to perform a coupled master cycle (or a master write cycle to a full TXFIFO) while there is an incoming coupled slave cycle (or an incoming decoupled write cycle to a full RXFIFO), then the SCV64 asserts either  $\overline{\text{KBERR}}$ , or  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  (depending upon the cycle type and the setting of the RMCRETRY bit in the MODE register). This is intended to force the CPU to retry its master cycle and allows the incoming cycle to be completed first, thus resolving the deadlock.

The cycles which can result in deadlock include:

- coupled write,
- decoupled write cycle from the VMEbus when the RXFIFO is full,
- read or RMW, or
- IACK cycles.

The assertion of  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  in response to the master/slave deadlock is controlled by the RMCRETRY bit in the MODE register (Table A.21). By default this bit is cleared, which means that the SCV64 will assert only  $\overline{\text{KBERR}}$ . Setting RMCRETRY will cause the SCV64 to resolve the deadlock condition by asserting both  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$ .

## 2.8.5 Location Monitor Access

The Location Monitor resides at the top longword (32 bits) of each A32 and A24 slave image and is accessible from both the local and VMEbus (see “Location Monitor and LMFIFO” on page 2-101). The bottom (even) word of the Location Monitor is used for 16-bit messages. Since the Location Monitor replaces the longword of memory which otherwise exists at that address, writes to the Location Monitor are not turned into local cycles or entered into the RXFIFO. In addition, read accesses to the Location Monitor result in a bus error.

Writes to the Location Monitor are queued in the LMFIFO (32 bits wide and 31 stages deep) and read from the LMFIFO register (Table A.20). While there are entries in the LMFIFO, the SCV64 asserts  $\overline{\text{LMINT}}$  and sets the LMHD bit in the DCSR register (Table A.6). Since  $\overline{\text{LMINT}}$  is only asserted while there are entries in the LMFIFO, an interrupt service routine or polling of the  $\overline{\text{LMINT}}$  signal can be used to test for entries in the LMFIFO and support various Location Monitor functions. Note that  $\overline{\text{LMINT}}$  may be tied directly to  $\overline{\text{LIRQ2}}$  during external local IACK decoding (see “Interrupt Acknowledge Cycles” on page 2-22).

The CPU may write to its own Location Monitor through its slave image as it would any address on the VMEbus (i.e. asserting  $\overline{\text{VMEOUT}}$  with the address). The SCV64 address decoder responds by redirecting the message to the LMFIFO. From there the message is handled in the same manner as a write from the VMEbus side. If the condition occurs in which the local CPU and VMEbus write to the Location Monitor at the same time, the race condition is resolved and the writes queued without loss of either of the messages (providing space is available in the LMFIFO).

One function of the Location Monitor is as a command to exit BI-mode. When there is a write to the Location Monitor, the SCV64 is removed from BI-mode and the BIMODE signal is cleared (see “BI-Mode” on page 2-119).



### 2.8.6 Bus Busy Glitch

The VME bus busy glitch is a common characteristic of the VMEbus. The ACC and SCV64 have BBSY glitch filters that should filter out problems by double sampling with C32MHz. To help avoid the occurrence of this type of glitch, it is recommended that the backplane be designed with at least 12 to 14 layers. Backplane designs with only 2 or 3 layers are prone to have this glitch.

### 2.8.7 BI-Mode Effects

While the SCV64 is in BI-mode, its VMEbus transceivers are pointed inwards and it ceases any master or slave activity on the VMEbus. Even though its slave images may have been programmed, it does not respond to any accesses and the VMEbus times out if such an access is attempted. Syscon functions, location monitor access, VMEbus daisy chain logic and all functions related to internal card activity are not affected.

The internal registers of the SCV64 are fully accessible to the local CPU while in BI-mode. The local memory slave image, at the programmed base address and size, is also accessible to the local CPU. Such an access does not use the VMEbus or FIFOs unless the SCV64 is in loopback mode (see “Test and Diagnostic Modes” on page 2-126). Address decoding for the VSB also remains active, for those pages that have been mapped over to that bus (see “CPU Memory Map” on page 2-45), though the card design may externally block such access while in BI-mode.

## 2.8.8 Bus Error Handling

A BERR\* encountered by the SCV64, whether as a result of a coupled cycle, a decoupled write, or a DMA cycle, disables the SCV64's VME master interface. In each case, the SCV64 asserts the  $\overline{\text{VMEINT}}$  pin, and sets the VBERR bit. The VME master interface remains disabled until the VBERR bit is cleared.

While the interface is disabled, CPU initiated coupled cycles time-out on the local bus. Cycles from a CPU initiated decoupled write (or a DMA write) may continued to be written into the TXFIFO but they will not progress beyond the TXFIFO. Once the TXFIFO fills, all future cycles time-out on the local bus and the DMA will stop.

The transmit FIFO registers always contain information about the most current or recently attempted cycle including the failed cycle. The TXADDR and TXDATA registers contain the attempted address and the data being written. The TXCTL register contains information on the type of cycle attempted and the information to be used to determine the cycle's address modifier code (AM code).

**Table 2.18 : TXCTL Register Bits Description**

Bit	Description
TYPE	If part of a CPU cycle, this bit indicates program or data transfer
SUPER	This bit indicates supervisory or user privilege level
SPC	The SPC bit indicates the size of address space being accessed
SIZ	The SIZ bit indicates the size of data being transferred
BLT	This bit indicates whether this is part of a BLT transfer
MBLT	This bit indicates whether this is part of an MBLT transfer

Decoupled writes which terminate with BERR\* on the VMEbus cause  $\overline{\text{VMEINT}}$  to be asserted, the transmit FIFO to be locked up, and the VBERR bit in the DCSR register to be set. VBERR will be set and  $\overline{\text{VMEINT}}$  will be asserted in response to BERR\* during a DMA initiated read or write cycle as well. The cycle that failed will still be in the TXFIFO registers TXADDR, TXDATA, and TXCTL. The local CPU's interrupt service routine for the  $\overline{\text{VMEINT}}$  pin should examine the DCSR register to determine if the cause of the interrupt was a VMEbus error. If either the DCSR registers' DONE bit is set or the DMATC registers' contents are non-zero, then the error occurred during a DMA transfer. The DONE bit is set if the DMA has just completed its transfers, although some cycles may still be queued in either FIFO.

A DMA read operation which terminates with BERR\* on the VMEbus results in the SCV64 halting further reads from the VMEbus; however, the SCV64 will transfer any cycles already queued in the receive FIFO onto the local bus.

Failed cycles can be queued back into the FIFO except for cycles that were parts of block transfers (BLT and MBLT). Block transfers can only be queued back into the FIFO as standard transfers, not as block transfers. To queue a cycle back into the FIFO, recreate the cycle using the information contained in the FIFO registers. It may be necessary to unlock the transmit FIFO if it is full by clearing the error bits in the DCSR. The transmit FIFO should then recommence the cycles that have been previously queued.

If sequential consistency of write cycles is a concern, the FIFO contents can be recirculated to put the failed cycle back at the FIFO head. With the FIFO still disabled, manually shift out the contents of the FIFO using the TXSHFT bit in the DCSR register. This bit will shift the contents of the FIFO one entry at a time into the transmit FIFO registers. Examine the contents of these entries as they are shifted out and store them into memory. Continue this process until the FIFO is empty as indicated by the TXHD bit in the DCSR register being cleared. No other local devices should be allowed onto the local bus during this period to perform writes to the VMEbus. Once the contents of the FIFO have been determined and stored, unlock the FIFO by clearing the relevant error bit in the DCSR register. Re-queue the original write cycles, including the modified one, by duplicating them using the information gathered earlier from the FIFO registers.

If the failed cycle does not need to be re-attempted, the FIFO may be unlocked at any time. Any cycles previously queued up within it will, in their turn, be carried out on the VMEbus.

## 2.9 Local Bus Interface

### 2.9.1 Local Bus Arbitration

The SCV64 provides a local arbiter which may be bypassed if required. Bypassing the local arbiter causes the SCV64 to function only as a local requester. If the local arbiter is active, then three arbitration conditions arise:

- an SCV64 request,
- a request from an external device using bus grant acknowledgment to maintain bus ownership, and
- a request from an external device not using bus grant acknowledgment to maintain bus ownership.

The SCV64 asserts  $\overline{\text{KBRQ}}$  to request the bus (from the CPU or external arbiter), and accepts  $\overline{\text{KBGR}}$  as its bus grant signal. As the arbiter, the SCV64 accepts  $\overline{\text{LBRQ1}}$  as a bus request from an external device (e.g. a DMA controller), and generates  $\overline{\text{LBGR1}}$  as a bus grant signal to the local requester. For internal requests (e.g. from the internal DMA, or the RXFIFO), the internal bus request and bus grant signals will be referred to as  $\overline{\text{INTREQ}}$  and  $\overline{\text{INTGR}}$ , respectively. These internal signals cannot be monitored but are included in the following description of local arbiter operation.

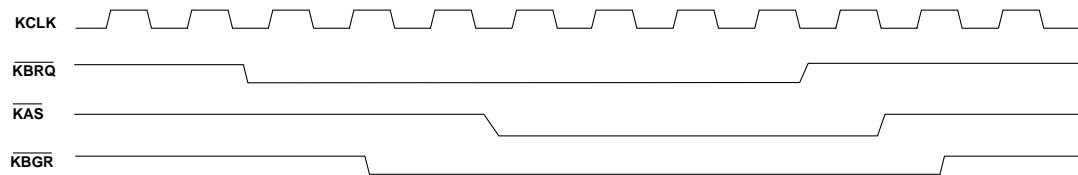
#### 2.9.1.1 Local Arbiter Bypassed

The local arbiter in the SCV64 can be bypassed by holding  $\overline{\text{KBGACK}}$  low during a low to high transition on  $\overline{\text{PWRST}}$ . Bypass mode disables request level 1, making  $\overline{\text{LBRQ1}}$  and  $\overline{\text{LBGR1}}$  undefined, and sets the ARBBYP bit in the MISC register (see Table A.42).

With the local arbiter bypassed, any bus requests from internal requesters (e.g. DMA controller) are routed directly to  $\overline{\text{KBRQ}}$ . A local bus grant ( $\overline{\text{KBGR}}$ ) must be issued to the SCV64 by the external arbiter when the bus is released by the previous bus master. The SCV64 retains bus mastership while it holds  $\overline{\text{KBRQ}}$  or  $\overline{\text{KAS}}$  asserted (see Figure 2.15). The grant ( $\overline{\text{KBGR}}$ ) should not remain asserted to the SCV64 once it has negated both  $\overline{\text{KBRQ}}$  and  $\overline{\text{KAS}}$ . To force the SCV64 to relinquish bus mastership, the local bus may negate  $\overline{\text{KBGR}}$  at any point. The SCV64 gives up the bus within one or two transfers except under the following conditions:

- during MBLT slave writes or DMA MBLT reads when bursts are disabled (FIFOBEN bit cleared), or
- during burst reads.

In the first instance, the SCV64 gives up the local bus once the RXFIFO is emptied or a block transfer boundary is reached. In the second instance, the SCV64 releases the bus once the TXFIFO is filled.



**Figure 2.15 : Local Bus Arbitration with Local Arbiter Bypass**



*Note that  $\overline{\text{KBGACK}}$  should be pulled to a defined logic level while the SCV64 local arbiter is bypassed. Transitions of  $\overline{\text{KBGACK}}$  should be expected but can be ignored.*

## 2.9.1.2 Local Arbiter Active

### Internal Requests

Internal requests are received by the arbiter on the INTREQ line and enter the local arbiter state machine (see Figure 2.16) which runs off the KCLK clock input. The numerical value of the different states (e.g. S0, S1, etc.) reflect the internal accounting system of the state machine and do not reflect any underlying ordering scheme. The timing for internal arbitration is shown in Figure 2.17 on page 2-70.

#### State 0

The state machine idles in S0 until a request is received on  $\overline{\text{LBRQ1}}$  or INTREQ. When either of these requests are asserted, the SCV64 drives  $\overline{\text{KBRQ}}$  low and enters S1 on the next rising clock edge.

#### State 1

The SCV64 holds  $\overline{\text{KBRQ}}$  low and remains in S1 for one clock cycle before entering S3.

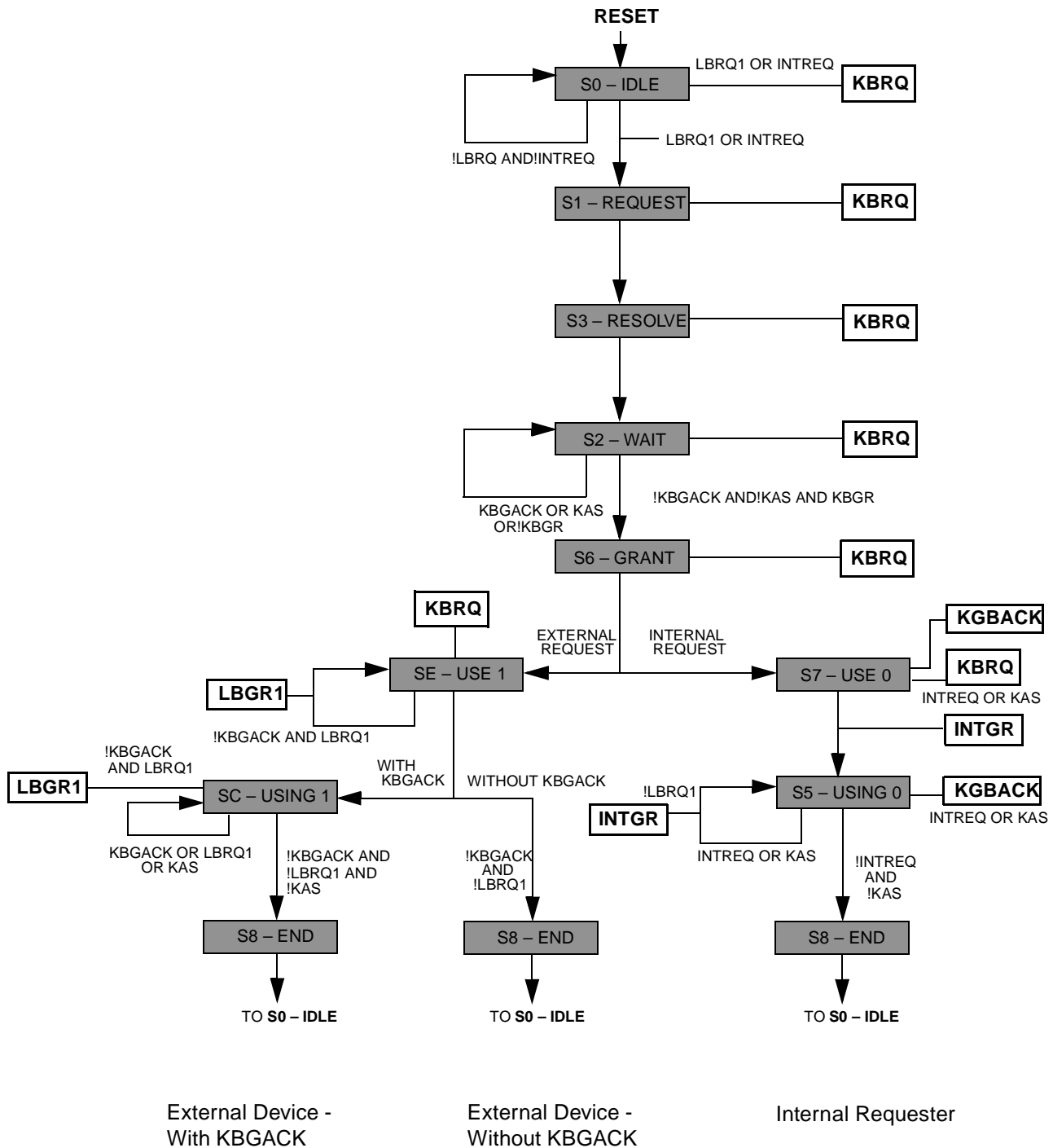


Figure 2.16 : Local Bus Arbiter State Machine

### State 3

During this state, the SCV64 resolves priority between  $\overline{\text{LBRQ1}}$  and  $\text{INTREQ}$ . If the  $\text{LBRAM}$  bit in the  $\text{CTL2}$  register is set, then  $\overline{\text{LBRQ1}}$  will receive bus grant before the internal request.  $\text{LBRAM}$  defaults to fair arbitration (SCV64 given fair arbitration level as external requester). The priority resolution lasts one clock cycle and the SCV64 enters S2.

### State 2

The state machine idles in S2 until these following conditions are met:

- $\overline{\text{KBGACK}}$  high,
- $\overline{\text{KAS}}$  high, and
- $\overline{\text{KBGR}}$  asserted.

These signals are sampled on the falling edge of S2. If the above conditions are met, then the state machine enters S6.

### State 6

If arbitration is for an external request, see “External Requests (With or Without Acknowledgment)” on page 2-71, then the SCV64 enters State E. If arbitration is for an internal request (i.e. for the SCV64 itself), then the SCV64 enters State 7.

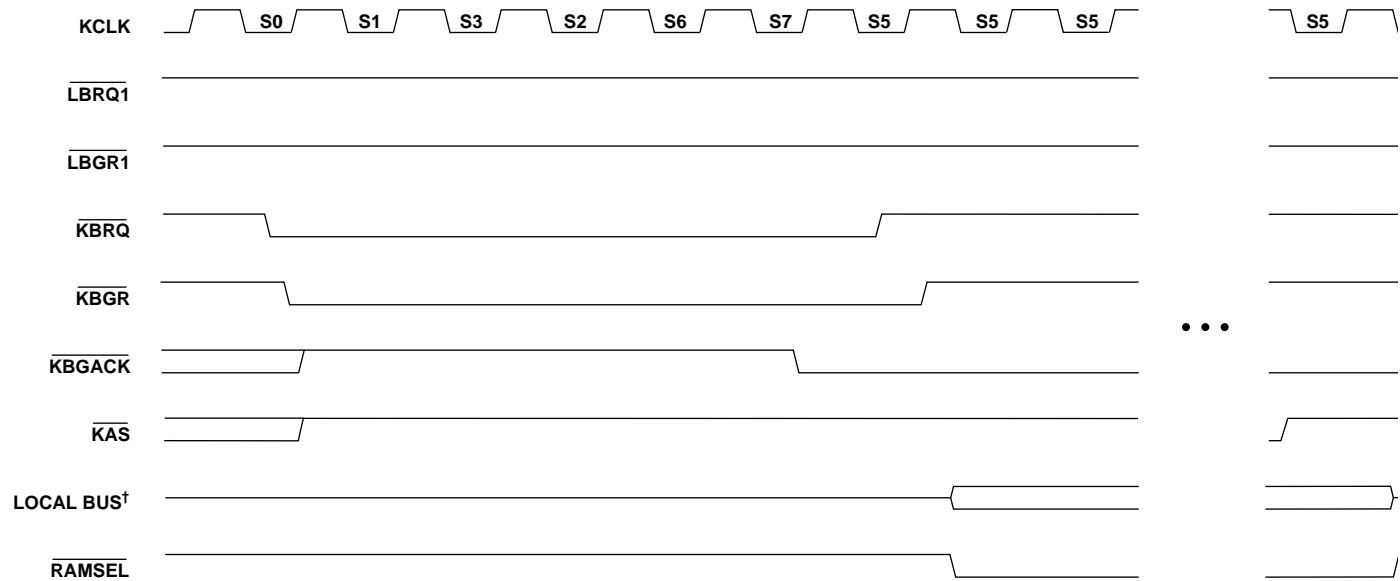
### State 7

$\overline{\text{KBGACK}}$  is asserted during S7, indicating that the SCV64 is local master. At the end of S7, the arbiter asserts the internal bus grant signal,  $\text{INTGR}$ , and enters S5.  $\overline{\text{KBRQ}}$  is released at the end of S7, allowing other devices the opportunity to request the bus while the SCV64 is master.

### State 5

The state machine will idle in S5, holding  $\overline{\text{KBGACK}}$  low, until the internal request and  $\overline{\text{KAS}}$  are negated. When  $\overline{\text{KAS}}$  is deasserted,  $\overline{\text{KBGACK}}$  is released after S5. If  $\overline{\text{LBRQ1}}$  (an external request) is asserted at any time while the SCV64 is master, then  $\overline{\text{KAS}}$  will be negated and  $\overline{\text{KBGACK}}$  released after S5. (The state machine terminates the cycle in S8 and then returns to S0.)

$\text{INTGR}$  is maintained until the SCV64 removes its internal request or  $\overline{\text{KAS}}$ , or  $\overline{\text{LBRQ1}}$  is asserted while the arbiter is in demand mode. Once  $\text{INTGR}$  is removed, the SCV64 removes its  $\text{INTRQ}$  and  $\overline{\text{KAS}}$ ;  $\overline{\text{KBGACK}}$  is negated by the arbiter, and the arbiter moves to S8. If  $\text{INTRQ}$  is removed first, then  $\text{INTGR}$  is removed immediately,  $\overline{\text{KBGACK}}$  is negated at end of S5 and arbiter goes to S8.



<sup>†</sup>**NOTE:** Local bus signals include: KADDR, KSIZ, KAS, KDS, KWR, and KRMC

**Figure 2.17 : Local Arbitration for Internal Requester**



### External Requests (With or Without Acknowledgment)

External bus requests are asserted on the  $\overline{\text{LBRQ1}}$  line. When the local arbiter is active, both external and internal bus requests enter the local arbiter state machine (see Figure 2.16). The numerical value of the different states (e.g. S0, S1, etc.) reflect the internal accounting system of the state machine and do not reflect any underlying ordering scheme. Arbitration timing for external requests is shown in Figure 2.18 and Figure 2.19 on page 2-72.

#### State 0

The state machine idles in S0 until a request is received on  $\overline{\text{LBRQ1}}$  or INTREQ. When either of these requests are asserted, the SCV64 asserts  $\overline{\text{KBRQ}}$  and enters S1 on the rising clock edge.

#### State 1

The SCV64 holds  $\overline{\text{KBRQ}}$  low and remains in S1 for one clock cycle before entering S3.

#### State 3

During this state, the SCV64 resolves priority between  $\overline{\text{LBRQ1}}$  and INTREQ. If LBRAM in the CTL2 register is set, then  $\overline{\text{LBRQ1}}$  will receive bus grant before the internal request. By default, LBRAM is cleared and the SCV64 is given equal priority to external requesters. The priority resolution lasts one clock cycle and the SCV64 enters S2.

#### State 2

The state machine idles in S2 until these following conditions are met:

- $\overline{\text{KBGACK}}$  high,
- $\overline{\text{KAS}}$  high, and
- $\overline{\text{KBGR}}$  asserted.

These signals are sampled during S2. If the above conditions are met, then the state machine enters S6.

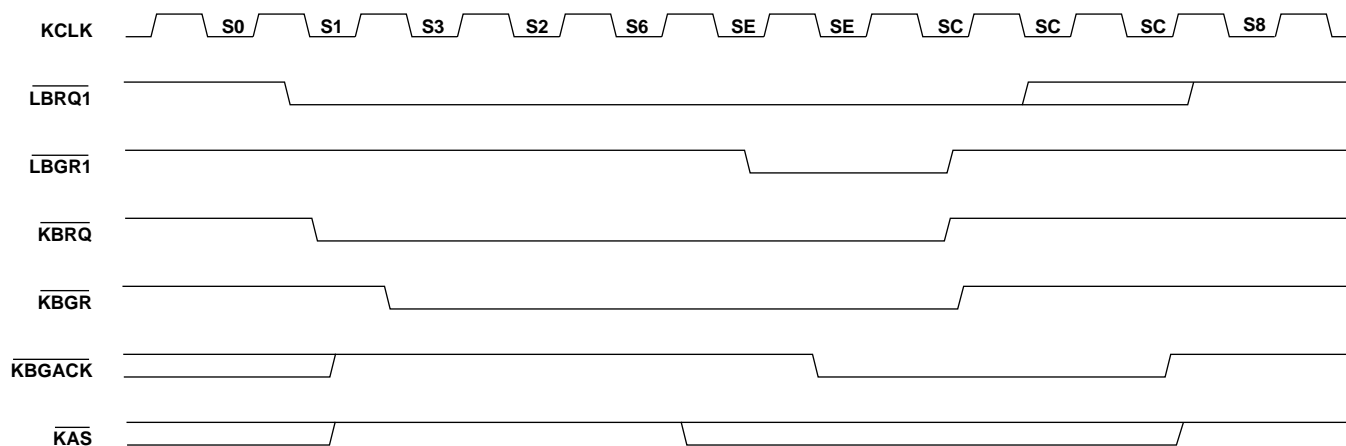
#### State 6

If arbitration is for an internal requester (see “Internal Requests” on page 2-67), then the SCV64 enters State 7. If arbitration is for an external requester, then the SCV64 enters State E.

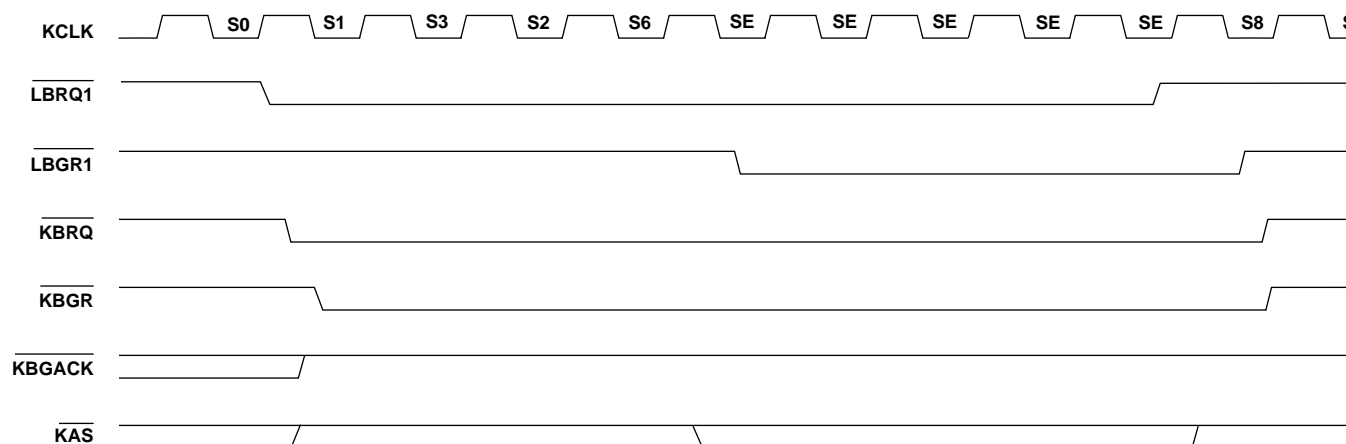
## State E and State C

The state machine can branch from SE along two directions. In one direction (with acknowledgment), the local requester asserts  $\overline{\text{KBGACK}}$  on the falling edge of SE, the arbiter releases  $\overline{\text{KBRQ}}$ , and the cycle enters State C. In SC, the local requester remains bus master as long as  $\overline{\text{KBGACK}}$  is asserted. Since  $\overline{\text{KBRQ}}$  has been negated, other devices can request the bus while the external device has ownership.

In the other direction (without acknowledgment), the local requester does not assert  $\overline{\text{KBGACK}}$ . Instead, the requester owns the bus as long as it asserts  $\overline{\text{LBRQ1}}$ . Since  $\overline{\text{KBRQ}}$  is maintained by the SCV64 throughout the cycle, no other device can request the bus until the cycle is complete. Under these conditions, the local arbiter state machine remains in SE until  $\overline{\text{LBRQ1}}$  is negated.



**Figure 2.18 : Local Arbitration for External Device - With Acknowledgment**



**Figure 2.19 : Local Arbitration for External Device - Without Acknowledgment**

## 2.9.2 Local Cycles – Overview

SCV64 local cycles are synchronous, meaning that bus and control input signals are externally synchronized to the CPU clock (KCLK). However, the  $\overline{\text{KDSACKx}}$  and  $\overline{\text{KBERR}}$  signals may be provided asynchronously and will be double sampled by the SCV64. In the following discussion, bus cycles are divided into three phases:

1. cycle initiation,
2. data transfer, and
3. cycle termination.

The signals and general operation for each of these three phases will be described in the overview below.

### 2.9.2.1 Cycle Initiation

Local bus cycles are initiated by driving the address, function codes, size, and read/write signals onto the bus. The function code signals (KFC0-KFC2) select the transfer type (supervisory or nonprivileged, program or data, Table 2.19). The KSIZ0-KSIZ1 signals indicate the number of bytes to be transferred in the cycle. The write signal ( $\overline{\text{KWR}}$ ) is negated for reads and asserted for write operations. The  $\overline{\text{KRMC}}$  signal is asserted for read-modify-write cycles. These signals are all qualified with the local address strobe ( $\overline{\text{KAS}}$ ).

**Table 2.19 : KFC Encodings for Transfer Type**

KFC2	KFC1	KFC0	Transfer Type
0	0	0	–
0	0	1	Nonprivileged Data Space
0	1	0	Nonprivileged Program Space
0	1	1	–
1	0	0	–
1	0	1	Supervisory Data Space
1	1	0	Supervisory Program Space
1	1	1	–

**Table 2.20 : Transfer Size Encoding**

KSIZE1	KSIZE0	Size
0	1	Byte
1	0	Word
1	1	3 bytes
0	0	Long Word

### 2.9.2.2 Data Transfer

When the SCV64 is local slave, it signals its port size to the CPU using the  $\overline{\text{KDSACKx}}$  signals at cycle termination (see Table 2.21). As discussed earlier (see “CPU Memory Map” on page 2-45), the address and data mode used during VME cycles depends upon the VME address range accessed by the CPU. The data mode for the VME cycle determines the port size that the SCV64 signals to the CPU. For example, if the CPU accesses A16:D16 VME space, then the SCV64 will respond to the CPU as a 16-bit port. If the BUSSIZ bit in the MODE register (Table A.21) is cleared, then the SCV64 will vary its apparent port size, depending upon the current cycle. Note that the SCV64 cannot generate unaligned VME transfers while the BUSSIZ bit is cleared.

If the BUSSIZ bit is set, then the SCV64 always responds as a 32-bit port. In addition, setting this bit also allows the SCV64 to generate unaligned transfers on the VMEbus (see Table 2.21 for effect of BUSSIZ on indicated port size).

**Table 2.21 : Port Size as Indicated by BUSSIZ and Cycle Termination**

Data Space	BUSSIZ	$\overline{\text{KDSACK1}}$	$\overline{\text{KDSACK0}}$	Port Size
D32	0*	0	0	32 bit
	0	1	0	16 bit
	1	0	0	32 bit
D16	0	1	0	16 bit
	1	0	0	32 bit

\*Longword and longword aligned accesses only.

Dynamic bus sizing by the CPU means that the CPU will perform more than one cycle for a specific transfer, depending upon the initial transfer size and the local slave port size. For example, a longword transfer to an 16-bit port will require more than one cycle. A CPU that bus sizes will automatically generate additional cycles when it is notified of the slave's port size.

Bus sizing requires that certain byte lanes on the data bus be used for ports of certain sizes. For example, an 16-bit port resides on data bus bits 31-16 (see Table 2.22 for the protocol used by the SCV64). However, data can be routed to different byte lanes through the use of an address offset. Address offset is specified with the A00 and A01 address lines (see Table 2.23).

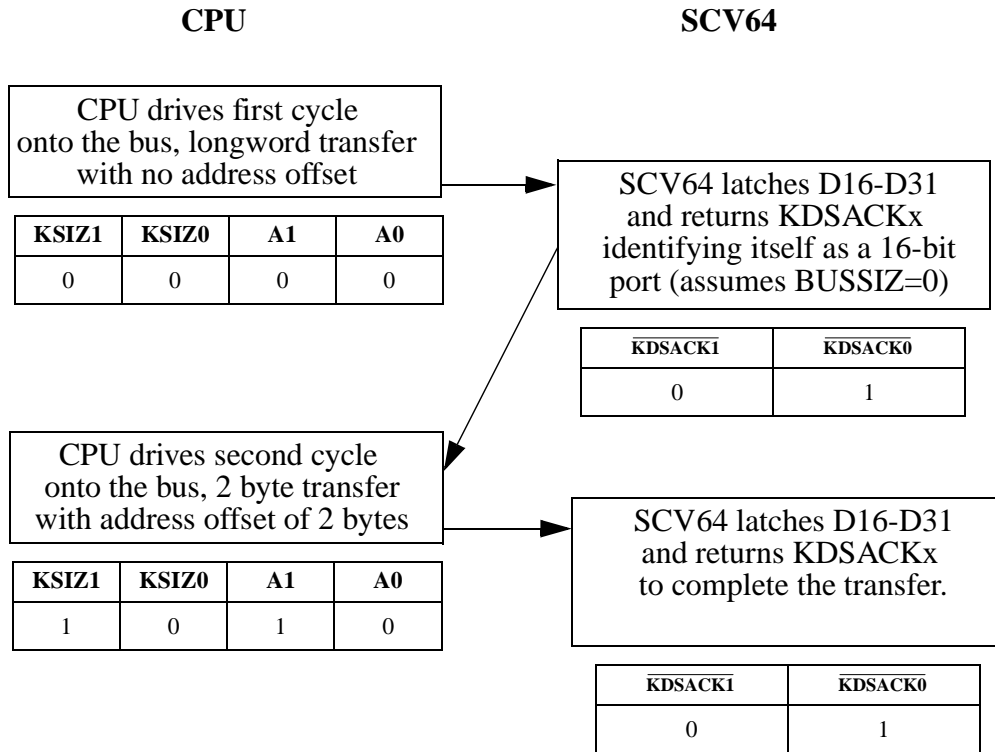
**Table 2.22 : Byte Lanes for Different Port Sizes on the Local Bus**

Port Size	Data Lines
32	31-0
16	31-16
08	31-24

**Table 2.23 : Encoding for Address Offset on the Local Bus**

A1	A0	Address Offset
0	0	+ 0 bytes
0	1	+ 1 byte
1	0	+ 2 bytes
1	1	+ 3 bytes

Figure 2.20 below gives an example of a CPU using the SCV64  $\overline{\text{KDSACKx}}$  signals for bus sizing. The CPU drives address, function codes and transfer size onto the bus, indicating that the transfer is longword with an address offset of 0. The CPU is accessing D16 space, so the SCV64 latches only the first two bytes of the transfer and identifies itself as a 16-bit port with  $\overline{\text{KDSACKx}}$  encoding ( $\overline{\text{KDSACKx}} = 01$ ). The CPU must then run another cycle to transfer the remaining 2 bytes. The transfer size in this second cycle is 2 bytes and the address offset is also 2 bytes (to allow for the 2 bytes already transferred). The final two bytes are latched by the SCV64 and it responds with  $\overline{\text{KDSACKx}}$  to complete the cycle.



**Figure 2.20 : Example of Longword Transfer to a 16-Bit Port**

Besides indicating port size, the  $\overline{\text{KDSACKx}}$  signals are also used to generate wait states. The SCV64 signals a wait state by holding  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KDSACK0}}$  high. The exception to this is in burst cycles, where  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KBERR}}$  are held high to signal a wait state.

### 2.9.2.3 Cycle Termination Signals

$\overline{\text{KDSACKx}}$  signals are used by the SCV64 to asynchronously terminate local transfers and to indicate port size (see above). During a write cycle,  $\overline{\text{KDSACKx}}$  signals also indicate that the data has been stored and the cycle can be terminated. During a read cycle, the  $\overline{\text{KDSACKx}}$  signals tell the local master to latch the data and terminate the transfer.

As local master, the SCV64 must sample  $\overline{\text{KDSACKx}}$  on two consecutive falling edges before terminating the cycle. After the first successful sample of  $\overline{\text{KDSACKx}}$  (on the falling edge of S2) the SCV64 enters S3 and samples  $\overline{\text{KDSACKx}}$  again on the falling edge of S4. If  $\overline{\text{KDSACKx}}$  is low on this second sample, then the cycle is terminated. The SCV64 does not perform dynamic bus sizing and accepts any combination of  $\overline{\text{KDSACKx}}$  as a signal to terminate the cycle.

$\overline{\text{KBERR}}$  is sampled in the same manner as  $\overline{\text{KDSACKx}}$ . However, if the BERRCHK bit in the MODE register is set, then the SCV64 will sample for  $\overline{\text{KBERR}}$  up to two clock cycles after cycle termination.

$\overline{\text{KBERR}}$  is also used as a termination signal (in combination with  $\overline{\text{KHALT}}$ ) during master/slave deadlocks (see “Master/Slave Deadlock Resolution” on page 2-61). The combination of  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  is a signal from the SCV64 indicating that the CPU retry its cycle.

Local interrupt acknowledge cycles can be terminated with  $\overline{\text{KAVEC}}$ , which signals the CPU to generate its own vector for an interrupt service routine (see “Interrupt Handler” on page 2-18).

## 2.9.2.4 Bus Error Handling

When the SCV64 is the local master, it can encounter bus errors due to the following operations:

- coupled read or write cycles (SCV64 is the VME slave)
- decoupled write cycles (SCV64 is the VME slave)
- DMA initiated read cycles (SCV64 is the VME master)

During coupled read or write cycles, the bus error is propagated back to the VME master where it should be handled normally. During decoupled write cycles (SCV64 as VME slave) and DMA initiated read cycles (SCV64 as VME master), the SCV64 is the initiator of the cycle either because of a previously enqueued write cycle or because the DMA was performing local reads to fill the transmit FIFO. In either case, it is not possible to propagate the error information back to the appropriate CPU.

Bus errors encountered during local write cycles initiated from the receive FIFO do not have any external effect, the  $\overline{\text{VMEINT}}$  pin is not asserted and the FIFO does not stop emptying its contents; however, the LBERR bit in the DCSR register is set. If more information is required, local logic must latch the relevant information when a bus error occurs, and then act upon it appropriately.

When the DMA is the initiator of a local read cycle that fails with a bus error, the DMA will continue on with its operations, assert the  $\overline{\text{VMEINT}}$  pin, and set the DLBERR bit in the DCSR register. The failed cycle will still be enqueued in the transmit FIFO. Local logic must respond appropriately and if necessary, take action to recover from the error.

### 2.9.3 SCV64 as Local Slave

The SCV64 becomes local slave if the CPU accesses

- SCV64 registers (see “Register Access” on page 2-87), or
- VME address space.

This section will deal only with the latter case where the CPU accesses VME space; SCV64 register access is described in “Register Access” on page 2-87. As described elsewhere (see “Data Path” on page 2-36), transfers between the local bus and VMEbus can be coupled or decoupled, where decoupling involves the use of a RXFIFO or TXFIFO. However, coupling or decoupling does not alter the local bus protocol used by the SCV64. From the perspective of the local bus, coupled cycles are essentially identical to decoupled cycles, except for the inclusion of several wait states while VMEbus arbitration and data transfer is completed. Therefore, local bus cycles with the SCV64 as local slave will be described together with unique properties of particular transfers mentioned where necessary.

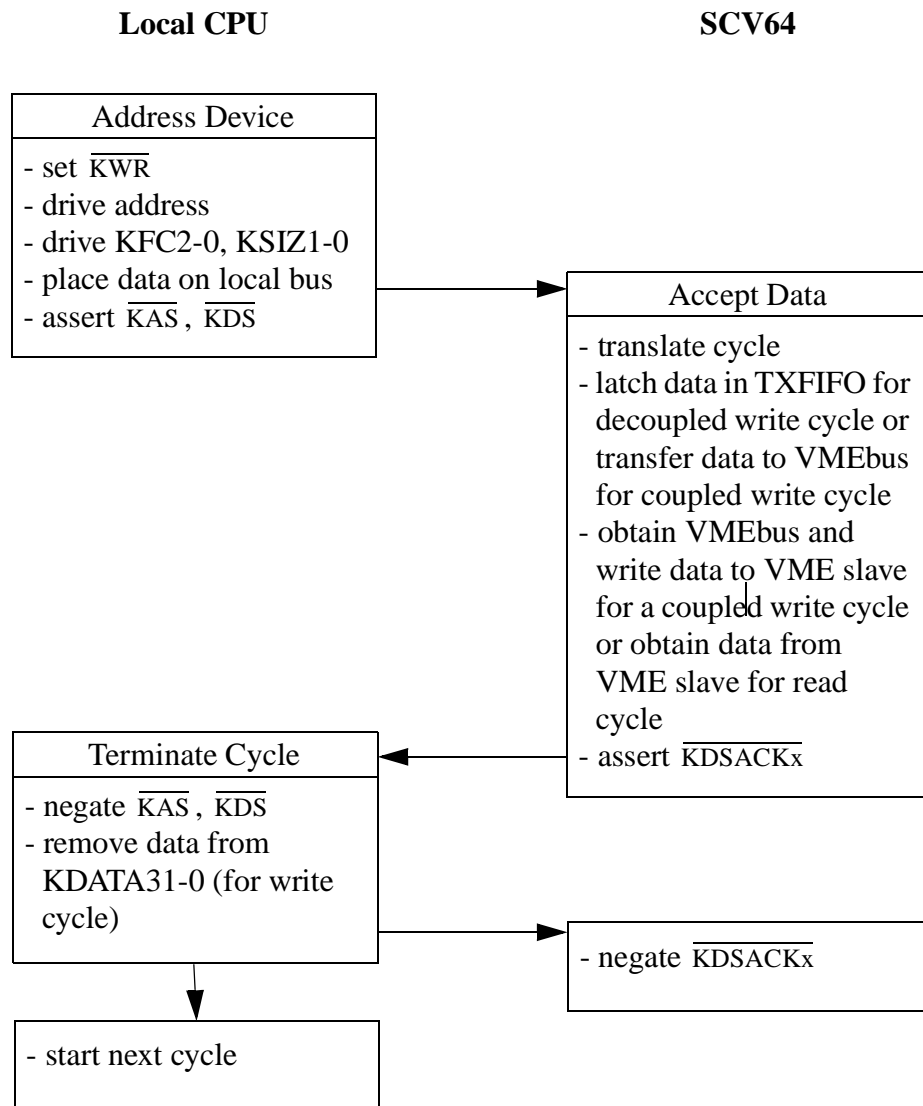
For further information about how the following cycles are combined with VMEbus transfers, see “Data Path” on page 2-36, “VMEbus Interface” on page 2-53, and “DMA Controller” on page 2-102. For information concerning byte lane translation, see “Byte Lane Translation” on page 2-59.

A flow chart describing a local cycle with SCV64 as local slave is provided in Figure 2.23 on page 2-84.



*When the local CPU performs a coupled access to the VMEbus, the cycle must compete on the VMEbus before it completes on the local bus. The cycle is not allowed to time-out (i.e., KBERR asserted) on the local bus before the cycle is terminated on the VMEbus.*





**Figure 2.21 : Flowchart for a Local Cycle with SCV64 as Local Slave**

**State 0**

Typically, the processor drives the address, function codes, and size information onto the local bus during S0. In addition,  $\overline{KWR}$  will be driven to indicate either a read or write.

**State 1**

During this period, the local address decoder asserts  $\overline{VMEOUT}$  to the SCV64. Together with  $\overline{KAS}$ , this signals the SCV64 to initiate a VME cycle. After putting address information on the bus, the SCV64 expects  $\overline{KAS}$  to be asserted indicating that the address on the bus is valid. If the transfer is a register access, the CPU must assert  $\overline{KDS}$  during this time as well.

**State 2**

During a coupled write cycle, the data is placed on the data bus by the end of S2 and qualified with  $\overline{KDS}$ . Note that the events in S0 and S1 do not have to occur in the order given. The only restriction is that the various signals be stable by the falling edge of S2. If the necessary signals are present by S2, then the SCV64 recognizes the cycle.

**State 3**

If neither of the  $\overline{\text{KDSACKx}}$  are recognized by the beginning of S3, then the cycle enters wait states. The different cycle types will typically have wait states inserted as follows:

- indeterminate number of wait states for coupled cycles),
- 1 wait states for a decoupled cycle (see Figure 2.22 on page 2-82),
- 1 wait state for register, slave image, or VSB access, and
- 2 wait states for location monitor access.

Both  $\overline{\text{KDSACKx}}$  signals must be negated in order for wait states to be inserted. When either  $\overline{\text{KDSACKx}}$  is recognized, the cycle enters S3 (first sample of  $\overline{\text{KDSACKx}}$ ).

**State 4**

$\overline{\text{KDSACKx}}$  signals are sampled a second time on the falling edge of S4. The second positive sampling of  $\overline{\text{KDSACKx}}$  indicates that the data can either be latched by local memory (for a read cycle) or that the VME slave has successfully latched the data (for a write cycle).

**State 5**

$\overline{\text{KAS}}$  is negated during S5.

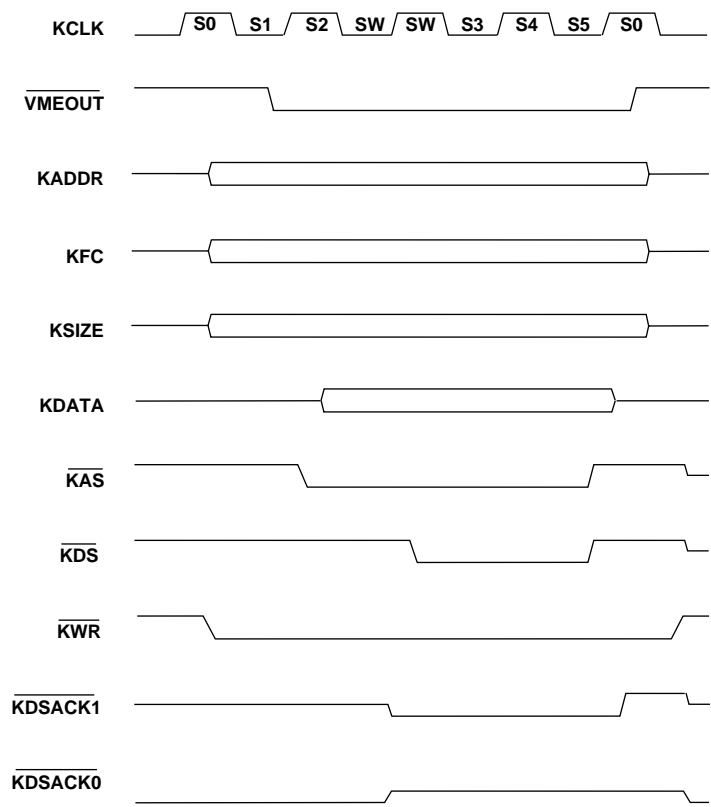


Figure 2.22 : Decoupled Write Cycle – SCV64 as Local Slave

## 2.9.4 SCV64 as Local Master

The SCV64 becomes local master:

- during VME accesses to local memory (which makes the SCV64 VME slave), or
- during data transfers involving the SCV64 DMA controller (which makes the SCV64 both local master and VME master, see “DMA Controller” on page 2-102).

As discussed elsewhere (“Data Path” on page 2-36), access from the VMEbus may be either coupled or decoupled, where decoupling involves the use of a RXFIFO or TXFIFO. Read cycles are always coupled, but write cycles can be either coupled or decoupled. However, coupling or decoupling does not alter the local bus protocol used by the SCV64. Coupled and decoupled cycles are essentially identical from the perspective of the local bus. Therefore, local bus cycles with the SCV64 as local master will be described together with unique properties of particular transfers mentioned where necessary.

For further information about how the following cycles are combined with VMEbus transfers, see “Data Path” on page 2-36, “VMEbus Interface” on page 2-53, and “DMA Controller” on page 2-102. For information concerning byte lane translation, see “Byte Lane Translation” on page 2-54.



*When the SCV64 initiates a local master cycle as a result of a coupled VME cycle, the cycle must complete on the local bus before it completes on the VME bus. It is not acceptable for a coupled access to time-out on the VMEbus with BERR\* while the cycle is still waiting to receive its termination on the local bus.*

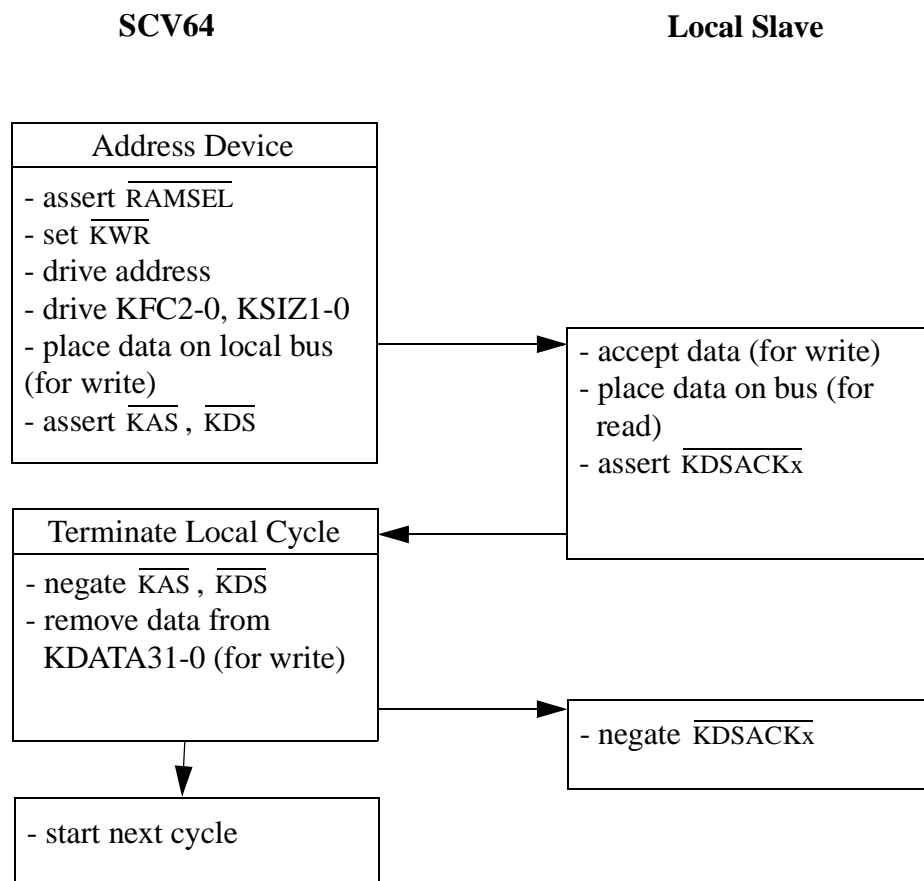
A flow chart describing a local cycle with SCV64 as local master is provided in Figure 2.23 on page 2-84. Timing for a decoupled write cycle with SCV64 as local master is given in Figure 2.24 on page 2-85.

### State 0

The SCV64 drives the address, function codes, and size information onto the local bus.  $\overline{\text{RAMSEL}}$  is asserted by the SCV64 when its VME slave image is accessed (see “VME Slave Memory Map” on page 2-49).  $\overline{\text{KWR}}$  is set during S0 to indicate a read or write.

### State 1

One half clock after putting address information on the bus, the SCV64 asserts  $\overline{\text{KAS}}$  indicating that the address on the bus is valid. If the transfer is a read cycle, the SCV64 asserts  $\overline{\text{KDS}}$  during this time as well.



**Figure 2.23 : Flowchart for a Local Cycle with SCV64 as Local Master**

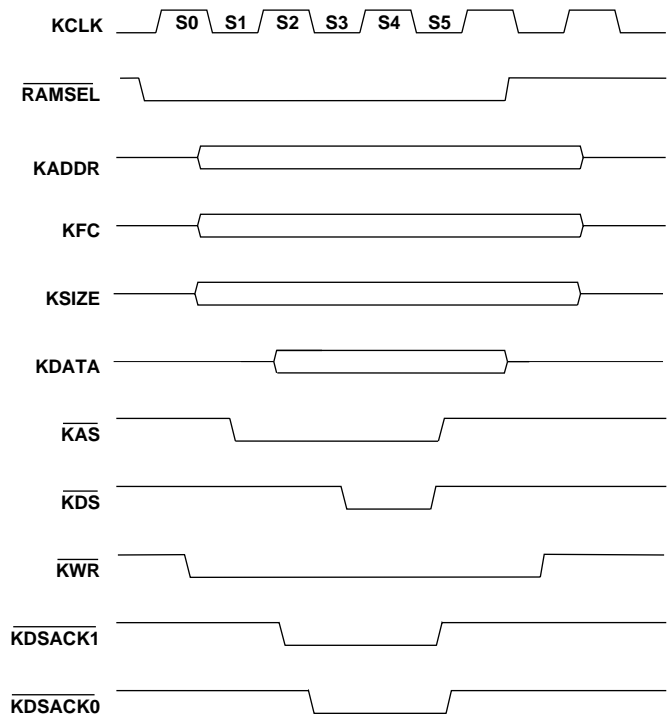


Figure 2.24 : Write Cycle – SCV64 as Local Master

**State 2**

During a write cycle, the data is placed on the data bus by the end of S2 and qualified with  $\overline{\text{KDS}}$ .

**State 3**

If neither of the  $\overline{\text{KDSACKx}}$  are recognized by the beginning of S3, then the SCV64 inserts wait states. Both  $\overline{\text{KDSACKx}}$  signals must be negated in order for wait states to be inserted. When either  $\overline{\text{KDSACKx}}$  is recognized by the SCV64, the cycle enters S3 (first sample of  $\overline{\text{KDSACKx}}$ ).

**State 4**

If  $\overline{\text{KDSACKx}}$  signals are still asserted (second sample of  $\overline{\text{KDSACKx}}$ ), then the data is latched on the falling edge of S4 for a read cycle. During a write cycle, the second positive sampling of  $\overline{\text{KDSACKx}}$  indicates that the local slave has successfully latched the data.

**State 5**

$\overline{\text{KAS}}$ ,  $\overline{\text{KDS}}$  and  $\overline{\text{KDSACKx}}$  are negated during S5. Address, functions code, and size information are held valid until after S5 to ensure hold time for memory systems.



## 2.9.5 Register Access

The SCV64 registers are accessed in order to:

- read status or information, or
- write to data registers or control bits.

The protocol for read and write cycles to registers are as described in “SCV64 as Local Slave” on page 2-78, except that the SCV64 chip select ( $\overline{\text{SCV64SEL}}$ ) is asserted instead of  $\overline{\text{VMEOUT}}$ . In addition, the SCV64 always responds as a 32-bit port during register accesses (both  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KDSACK0}}$  are asserted).

If SCV64 registers are accessed from the VMEbus, then the SCV64 treats itself as it would any other local device by asserting  $\overline{\text{RAMSEL}}$  and initiating a cycle on the local bus. A local address decoder asserts  $\overline{\text{SCV64SEL}}$  and directs the cycle to the SCV64 registers. Once the transfer has completed by the SCV64 asserting both  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KDSACK0}}$ , the SCV64 releases the local bus. The cycle completes normally on the local and the VMEbus.

The SCV64 decodes access to its registers using local address lines KADDR08 to KADDR00. Registers in address range 0x000 to 0x04C respond only to aligned longword transfers; the SCV64 will assert  $\overline{\text{KBERR}}$  with any other access. Registers in address range 0x080 to 0x0BF accept any size transfer, but ignore all data lines except KDATA07 to KDATA00 (these are effectively one byte registers in this address range, see Table 2.24 below and Table 2.25 on page 88).

The Master A64 Base Address Register (MA64BAR) and the Slave A64 Base Address Register (SA64BAR) can only be programmed from the VMEbus by using a coupled cycle (see “Coupled Mode” on page 2-39). Also, if the VMEbus Slave Base Address Register (VMEBAR) is programmed from the VMEbus using a decoupled cycle, the VMEbus slave base address will not be updated immediately.

**Table 2.24 : A Summary of SCV64 Register Accesses**

Address Range	32-bit Access	16-bit Access	8-bit Access
000 to 04C	yes	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
050 to 07F	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
080 to 0BF	yes	yes	yes
0C0 to 0E0	yes	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
0E4 to 1FF	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$

**Table 2.25 : SCV64 Register Map**

Longword Register Address	Register	Name
xxxx x000	DMA Local Address	DMALAR
xxxx x004	DMA VMEbus Address	DMAVAR
xxxx x008	DMA Transfer Count	DMATC
xxxx x00C	Control and Status	DCSR
xxxx x010	VMEbus Slave Base Address	VMEBAR
xxxx x014	RXFIFO Data	RXDATA
xxxx x018	RXFIFO Address Register	RXADDR
xxxx x01C	RXFIFO Control Register	RXCTL
xxxx x020	VMEbus/VSB Bus Select	BUSSEL
xxxx x024	VMEbus Interrupter Vector	IVECT
xxxx x028	Access Protect Boundary	APBR
xxxx x02C	TXFIFO Data Output Latch	TXDATA
xxxx x030	TXFIFO Address Output Latch	TXADDR
xxxx x034	TXFIFO AM Code and Control Bit Latch	TXCTL
xxxx x038	Location Monitor FIFO Read Port	LMFIFO
xxxx x03C	SCV64 Mode Control	MODE
xxxx x040	Slave A64 Base Address	SA64BAR
xxxx x044	Master A64 Base Address	MA64BAR
xxxx x048	Local Address Generator	LAG
xxxx x04C	DMA VMEbus Transfer Count	DMAVTC
xxxx 050 to xxxx 07F	reserved	
xxxx x080	Status Register 0	STAT0
xxxx x084	Status Register 1	STAT1
xxxx x088	General Control Register	GENCTL
xxxx x08C	VMEbus Interrupter Requester	VINT
xxxx x090	VMEbus Requester Register	VREQ
xxxx x094	VMEbus Arbiter Register	VARB
xxxx x098	ID Register	ID
xxxx x09C	Control and Status Register	CTL2
xxxx x0A0	Level 7 Interrupt Status Register	7IS
xxxx x0A4	Local Interrupt Status Register	LIS
xxxx x0A8	Level 7 Interrupt Enable Register	7IE
xxxx x0AC	Local Interrupt Enable Register	LIE
xxxx x0B0	VMEbus Interrupt Enable Register	VIE
xxxx x0B4	Local Interrupts 1 and 0 Control Register	IC10

**Table 2.25 : SCV64 Register Map (Continued)**

Longword Register Address	Register	Name
xxxx x0B8	Local Interrupts 3 and 2 Control Register	IC32
xxxx x0BC	Local Interrupts 5 and 4 Control Register	IC54
xxxx x0C0	Miscellaneous control register	MISC
xxxx x0C4	Delay line control register	DLCT
xxxx x0C8	Delay line status register 1	DLST1
xxxx x0CC	Delay line status register 2	DLST2
xxxx x0D0	Delay line status register 3	DLST3
xxxx x0D4	Mailbox register 0	MBOX0
xxxx x0D8	Mailbox register 1	MBOX1
xxxx x0DC	Mailbox register 2	MBOX2
xxxx x0E0	Mailbox register 3	MBOX3
xxxx x0E4 to xxxx x1FC	reserved	

## 2.9.6 Burst Cycles

The SCV64 can be programmed to initiate burst cycles to increase data transfer rate on the local bus, thus increasing the efficiency of BLT and MBLT transfers on the VMEbus. The SCV64 can perform burst read cycles (data from local memory to the TXFIFO) during DMA write operations, and burst write cycles (data from the RXFIFO to local memory) during DMA read operations. The SCV64 also supports burst mode for slave D32BLT write cycles. Burst mode is not supported D08 or D16BLT cycles.

All burst start addresses must be longword aligned for BLTs and double-longword aligned for MBLTs. Burst cycles are automatically terminated and restarted at every burst-length boundary. For example, a programmed burst of 8 longwords starting at address 0x08 terminates and restarts at address 0x20.

### 2.9.6.1 Burst Reads

Burst reads are enabled by setting the DMABEN bit in the MODE register (Table A.21). The maximum length for a burst read can be 4, 8, 16 or 32 longwords, and is set using the BLEN1-0 bits in the same register. If burst mode is enabled, the DMA controller will initiate a burst cycle by placing address, size, and function code information on the bus, and asserting  $\overline{\text{KAS}}$  and  $\overline{\text{KDS}}$  (see Figure 2.25). A function code of 3 identifies the transfer as a burst cycle.

On the second rising clock edge, the SCV64 begins to sample  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KBERR}}$ , and repeats this sampling on every subsequent rising edge (see Figure 2.26 on page 2-92). If  $\overline{\text{KDSACK1}}$  is asserted on a rising clock edge, then the SCV64 will latch the data on the subsequent falling clock edge. However, if  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KBERR}}$  are sampled high, then the SCV64 will insert wait states until one of these two signals are sampled low. One longword is transferred per clock cycle. Upon assertion of  $\overline{\text{KBERR}}$  during a burst read, the SCV64 sets the  $\overline{\text{DLBERR}}$  bit in the DCSR register and asserts  $\overline{\text{VMEINT}}$ . However, the burst does not terminate unless  $\overline{\text{KBERR}}$  is asserted for two falling edges of  $\text{KCLK}$ . When this occurs, the burst does not immediately terminate. Instead, up to four more longwords may be transferred.

Burst read cycles are automatically terminated by negating  $\overline{\text{KAS}}$  and  $\overline{\text{KDS}}$  at burst-length boundaries and restarting with a new address phase. Burst read cycles are terminated but not restarted if:

- the DMA transfer count is completed (see “Data Transfer Counts” on page 2-107),
- local bus ownership is granted to a higher priority request (bus grant negated), or
- the TXFIFO fills.

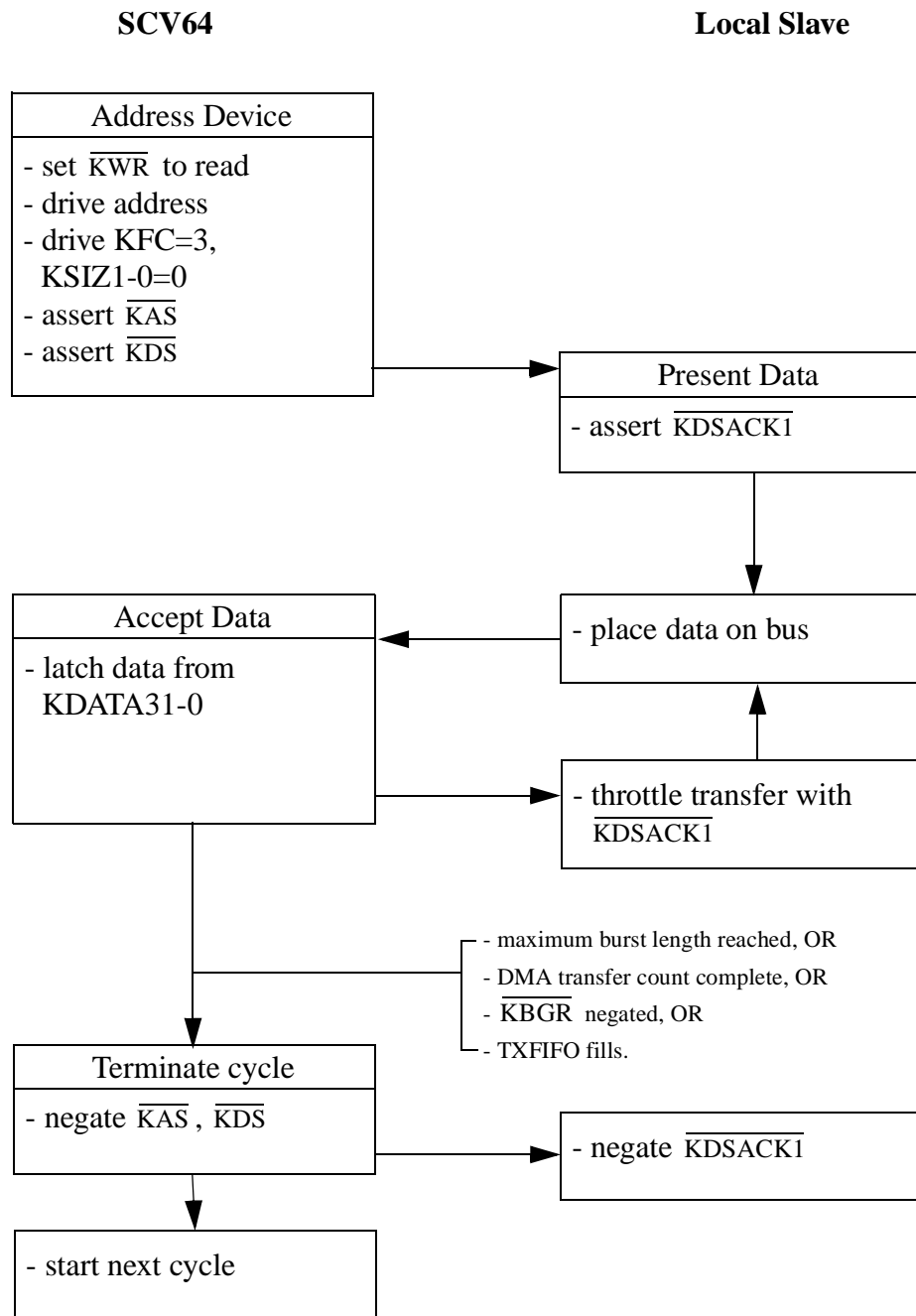
If the TXFIFO fills, the SCV64 will release the local bus before any additional cycles are completed. During BLT transfers, the local bus is automatically released when the TXFIFO contains 15 entries (the TXFIFO is 15 stages deep). During MBLT cycles, the local bus is automatically released when the TXFIFO contains 14 entries (since each MBLT cycle requires two local bus cycles).



*Note that  $\overline{\text{KAS}}$  is negated on the first falling edge after the last  $\overline{\text{KDSACKx}}$  sample. In some designs this may cause the slave to put data onto the bus which will be ignored by the SCV64.*

### 2.9.6.2 Burst Writes

Burst writes are enabled by setting the FIFOBEN bit in the MODE register (Table A.21). The maximum length for a burst write can be 4, 8, 16 or 32 longwords, and is set using the  $\text{BLEN1-0}$  bits in the same register. If FIFOBEN is set, the SCV64 will initiate a burst cycle if there are 2 sequential MBLT entries or 4 sequential BLT entries in the RXFIFO. The SCV64 begins the burst cycle by placing address, size, and function code information on the bus, and asserting  $\overline{\text{KAS}}$  and  $\overline{\text{KDS}}$  (see Figure 2.27). A function code of 3 identifies the transfer as a burst cycle.



**Figure 2.25 : Flowchart for a Burst Read Cycle - SCV64 as Local Master**

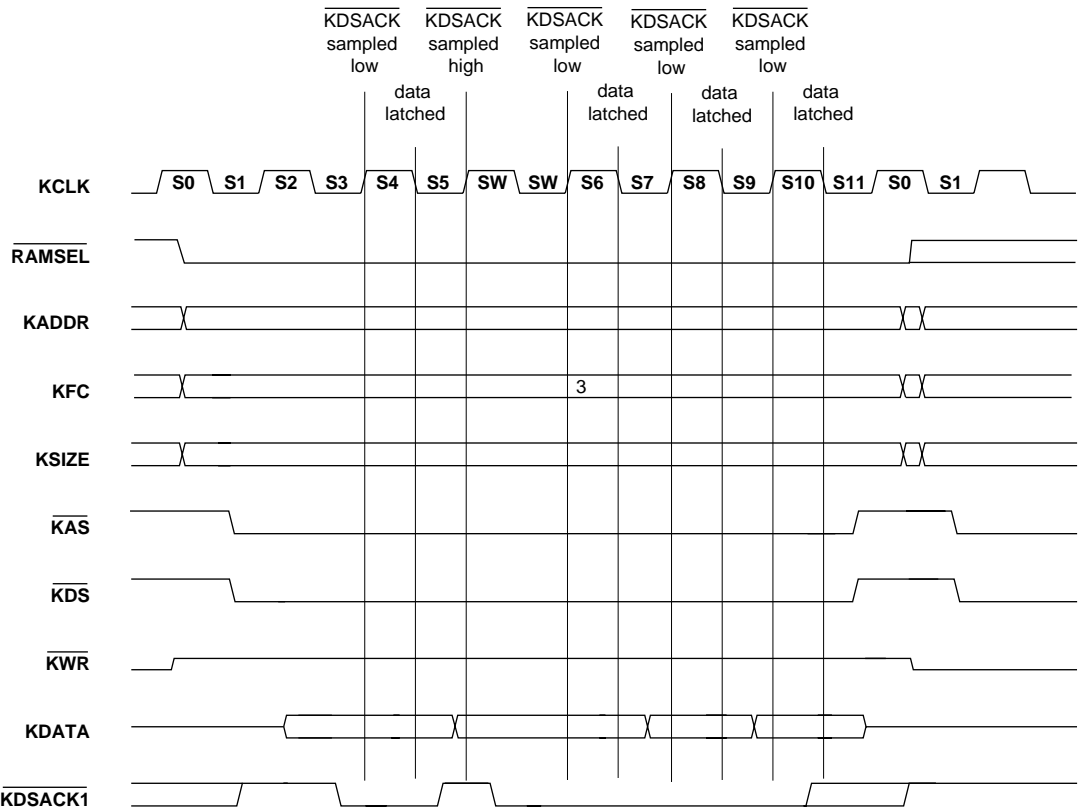


Figure 2.26 : Burst Read Cycle

$\overline{\text{KDSACK1}}$  is sampled on the second rising clock edge after cycle initiation to determine whether the slave device is ready to accept data. If  $\overline{\text{KDSACK1}}$  is asserted, then data is written on the next rising clock edge. This cycle is repeated until the cycle ends or  $\overline{\text{KDSACK1}}$  is sampled high. If  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KBERR}}$  are high, then the SCV64 will insert wait states and sample  $\overline{\text{KDSACK1}}$  and  $\overline{\text{KBERR}}$  on every rising clock edge. Data transfer resumes when  $\overline{\text{KDSACK1}}$  or  $\overline{\text{KBERR}}$  is sampled low. Assertion of  $\overline{\text{KBERR}}$  does not terminate the burst write, or activate any error logic. Unlike burst reads, where one longword is transferred every clock cycle, burst writes allow one longword transfer every 2 clock cycles (see Figure 2.28 on page 2-95).



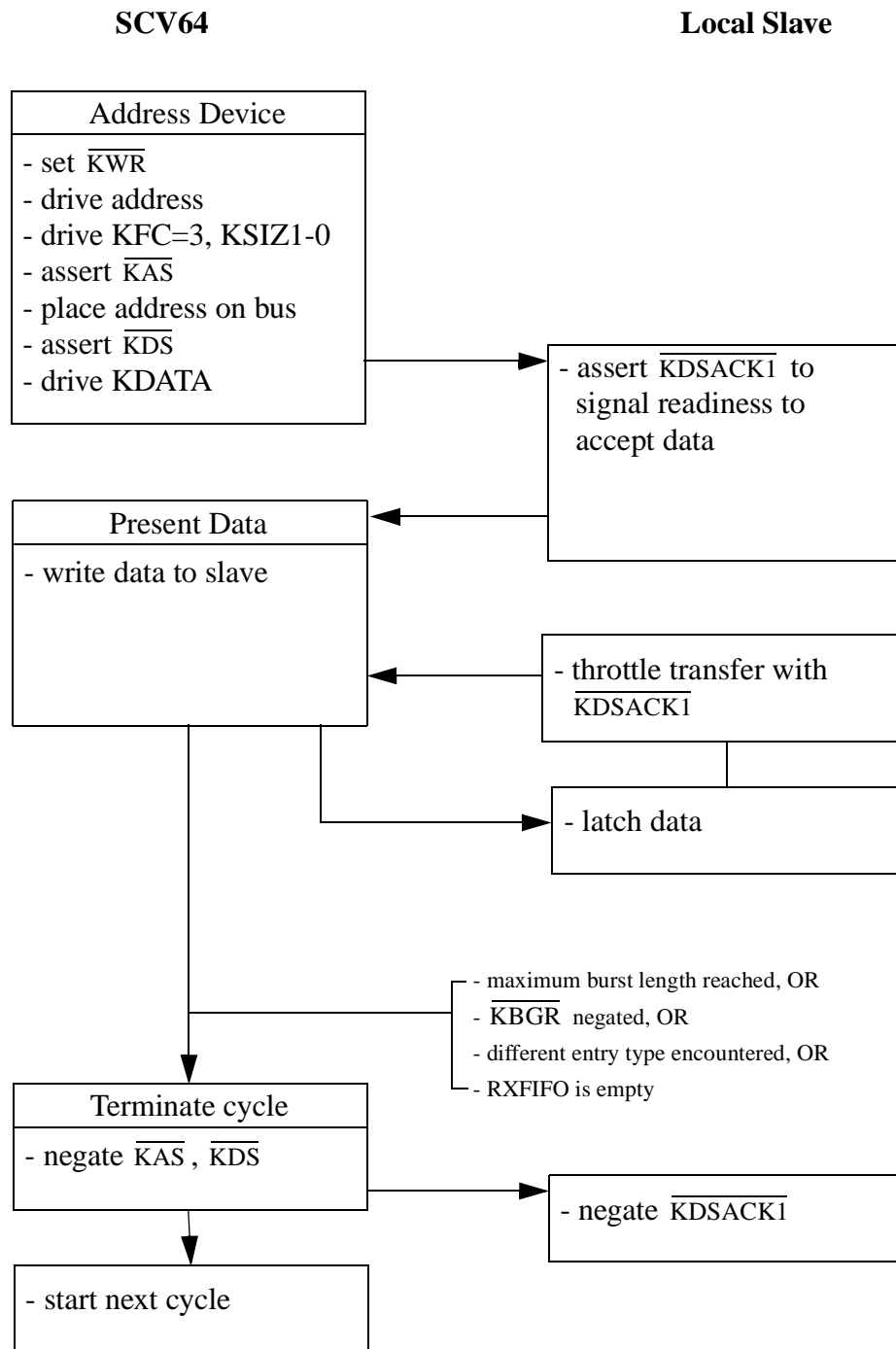
*Caution: Except during the first two burst data beats of local bus tenure, the number of waits that may be inserted during burst writes is limited by the local bus frequency (see Table 2.26 below). If not performing MBLT operations, any number wait states may be inserted. During the first data beat of a burst that is not the first burst of a local tenure, the maximum number of wait states that may be inserted during the first data beat is two fewer than that shown in Table 2.26. If more wait states must be inserted during the first data beat, remove the SCV64 from local bus ownership at the end of each burst cycle (through negation of  $\overline{\text{KBGR}}$  in arbiter bypass mode.) The grant signal ( $\overline{\text{KBGR}}$ ) must be negated with a setup of -2ns (AC parameter  $t_{598}$ ) to the next falling edge of  $\overline{\text{KCLK}}$  once  $\overline{\text{KAS}}$  has been negated by the SCV64 (See Figure B.11). This ensures that the first data beat of a burst is always the first one of a bus tenure.*

**Table 2.26 : Local Bus Frequency and Allowable Wait States During Burst Writes**

KCLK Frequency (MHz)	Maximum Number of Wait States
20	2
25	3
33	4
40	5

The burst is terminated ( $\overline{\text{KAS}}$  and  $\overline{\text{KDS}}$  negated) if the SCV64:

- encounters a different entry type (e.g. a BLT entry following a series of MBLT),
- $\overline{\text{KBGR}}$  is negated (in which case the burst will terminate on the next burst length boundary),
- completes its maximum burst length, or
- empties the RXFIFO.



**Figure 2.27 : Flowchart for a Burst Write Cycle - SCV64 as Local**



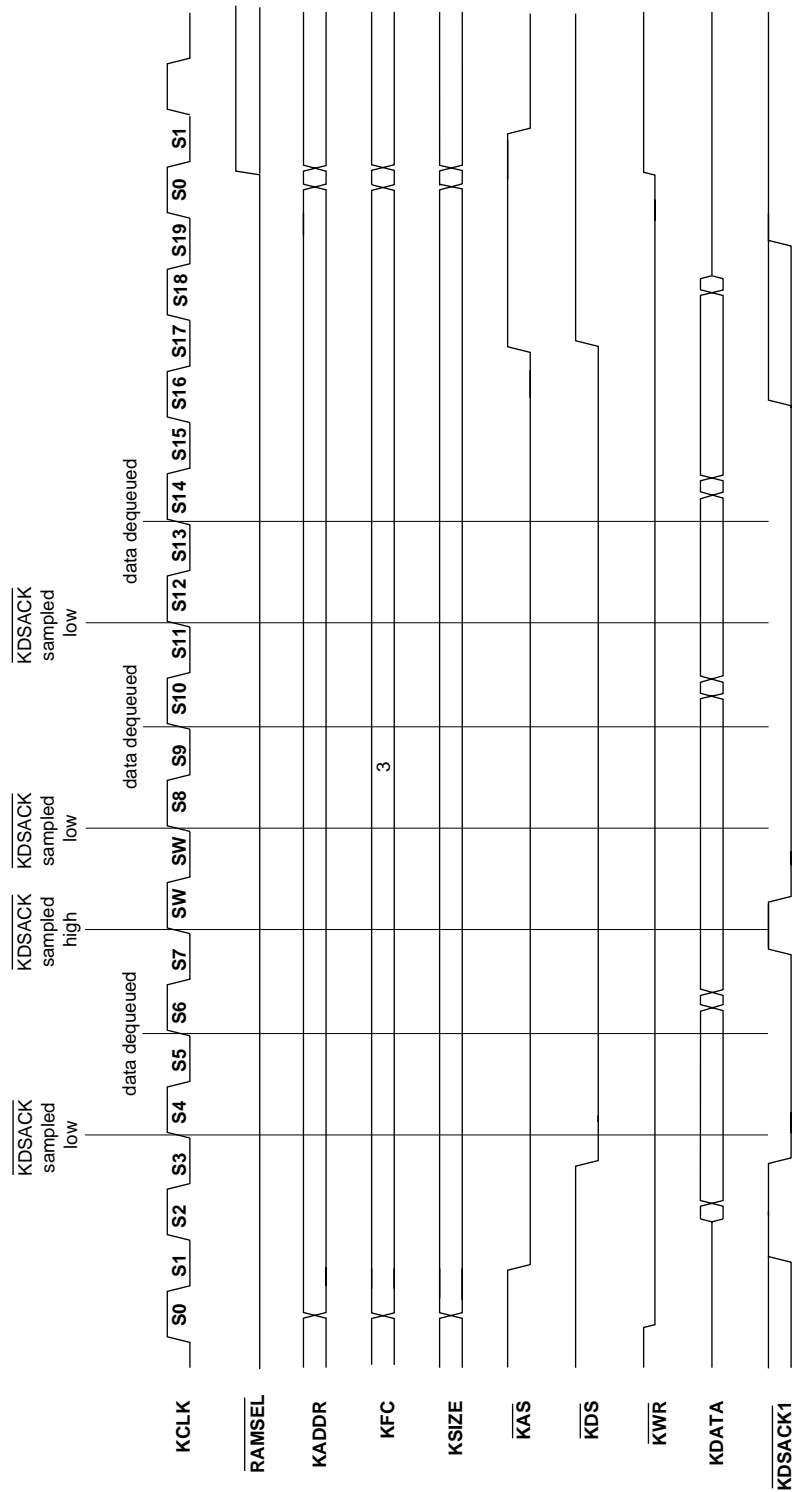


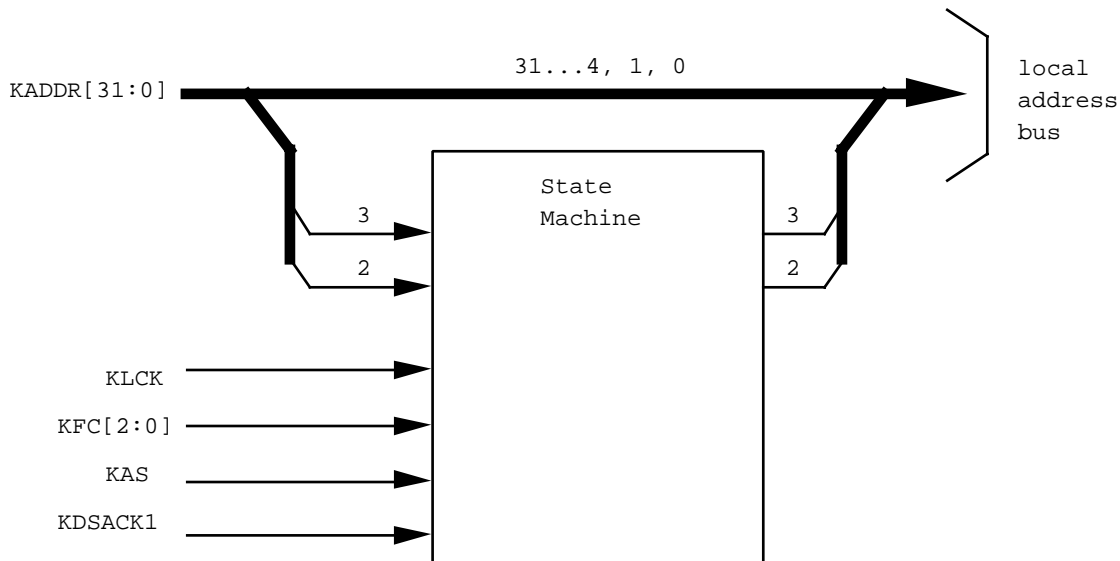
Figure 2.28 : Burst Write Cycle

### 2.9.6.3 Local Address Incrementation

During burst transfers, the SCV64 presents addresses only at burst length boundaries. Therefore, for some applications, a local address counter has to be provided as the SCV64 does not increment the addresses during the burst.

For example, if the burst length is programmed to be 4 longwords, then a 2-bit counter is required to increment address bits 2 and 3. Bits 0 and 1 remain low because burst transfers must be longword-aligned. Burst boundary conditions would occur at addresses 0x00, 0x10, 0x20, etc. If the programmed burst starts at address 0x04, then the SCV64 would perform 3 data beats before terminating the burst at the next burst boundary condition, 0x10. Provided there was still data available in the Rx FIFO in the case of a burst write, or room available in the Tx FIFO in the case of a burst read, then the SCV64 would begin the next burst at address 0x10.

In the example above, KADDR3 and KADDR2 can be routed through some PLD (see Figure 2.29). Between burst boundaries, the SCV64 does not tristate KADDR3 and KADDR2 so these pins should not be directly connected to the local bus. The counter can be controlled by using KCLK, KAS, KFC2-0 and KDSACK1.



**Figure 2.29 : Local interface for maximum burst length of 4 longwords**

### 2.9.7 Read-Modify-Write Cycles

On the VMEbus, AS\* is typically used to lock the bus during a RMW operation. Setting the TASCN bit in the MODE register (Table A.21) will program the SCV64 to assert AS\* for multiple cycles on the VMEbus while  $\overline{\text{KRMC}}$  is asserted on the local bus. Note that this approach allows only single address RMW cycles.

As a VME bus slave, the SCV64 does not allow re-arbitration of the local bus until AS\* is negated. This effectively locks the local resource until the VME bus RMW cycle is complete.

Alternatively, the VMEbus RETRY\* line can be used by the SCV64 as a proprietary VMEbus RMC signal (by clearing the RMCPIN bit in the MODE register). This is only possible if the VME slave is programmed to accept RETRY\*/ $\overline{\text{VRMC}}$  for this function. When the SCV64 receives  $\overline{\text{KRMC}}$  from the local master, it asserts RETRY\*/ $\overline{\text{VRMC}}$ , thus locking the local and VMEbus during the RMW cycle. Using this method, the AS\* line is free to qualify multiple addresses during the RMW transfer. If the VME slave is equipped with an SCV64 on board, and is configured to accept RETRY\* as a proprietary VMEbus RMC line, then the assertion of RETRY\*/ $\overline{\text{VRMC}}$  will cause  $\overline{\text{KRMC}}$  to be asserted on the local bus. Under these conditions, the local bus is only released when RETRY\*/ $\overline{\text{VRMC}}$  is negated.

## 2.9.8 Master/Slave Deadlock Resolution

If the SCV64 is waiting for VMEbus ownership in order to perform a coupled master cycle (or a master write cycle to a full TXFIFO) while there is an incoming coupled slave cycle (or an incoming write cycle to a full RXFIFO), then the SCV64 asserts either  $\overline{\text{KBERR}}$ , or  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$ . This is intended to force the CPU to retry its local master cycle and allows the incoming cycle to be completed first (or a RXFIFO entry to be completed), thus resolving the deadlock. The cycles which can result in deadlock include:

- coupled write,
- decoupled write cycle from the VMEbus when the RXFIFO is full,
- read or RMW, or
- IACK cycles.

The assertion of  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  in response to the master/slave deadlock is controlled by the RMCRETRY bit in the MODE register (Table A.21). During deadlocked RMW cycles (i.e. with  $\overline{\text{KRMC}}$  asserted), the SCV64 will assert only  $\overline{\text{KBERR}}$  by default. However, setting RMCRETRY programs the SCV64 to assert both  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  during a master/slave deadlock. The LBERR bit in DCSR register is not set during deadlocked cycles.

### 2.9.8.1 Deadlock Resolution in Decoupled Mode



*Caution: There is a potential deadlock resolution problem which users should avoid. The problem occurs when the SCV64 is in decoupled mode (TXFIFO). The SCV64 card performs a VMEbus write cycle that terminates in a bus error; then performs a coupled VME cycle before clearing the VBERR bit in the DCSR register. If that coupled cycle encounters a deadlock condition with an incoming coupled cycle, then the SCV64 will fail to issue  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  to resolve this. Instead, both bus cycles will terminate with time-out bus errors. One way of dealing with this issue is by implementing this hardware mechanism: if KBRQ from the SCV64 and VMEOUT are both asserted for at least 1 us, then assume a deadlock condition and generate BERR and HALT with external logic to the local processor.*

## 2.9.9 Location Monitor Access

The location monitor resides at the top longword of each A32 and A24 slave image (see “VME Slave Memory Map” on page 2-49) and is accessible from both the local and VMEbus. The bottom (even) word of the location monitor is used for 16-bit messages. Writes to the location monitor from the VMEbus are recorded in the LMFIFO, a 31 longword deep message queue. Whenever an entry appears in the LMFIFO, the SCV64 generates a local interrupt ( $\overline{\text{LMINT}}$ ) and sets the LMHD bit in the DCSR register (Table A.6).  $\overline{\text{LMINT}}$  is only negated when the LMFIFO is emptied.

Messages in the LMFIFO are read from the LMFIFO register (Table A.6). Reads from the LMFIFO itself returns only undefined data.

The CPU may write to its own location monitor as it would any other VME address. The SCV64 address decoder responds by redirecting the message to the LMFIFO.  $\overline{\text{LMINT}}$  is only asserted after the write to the LMFIFO is complete. From there, the message is handled in the same manner as any location monitor write from the VMEbus ( $\overline{\text{LMINT}}$  asserted and LMHD bit set).



*Note that a location monitor write involves 2 more wait states (one more clock cycle) than a register access or reflected cycle.*

If the condition occurs where the CPU and VMEbus write to the location monitor at the same time, priorities are resolved and both writes are queued provided there is sufficient space in the LMFIFO.

## 2.9.10 Reflected Cycles

If the CPU accesses its own VME slave image (see “VME Slave Memory Map” on page 2-49), the SCV64 redirects the access onto the local bus by asserting  $\overline{\text{RAMSEL}}$ . No VMEbus cycle is initiated (although  $\overline{\text{VMEOUT}}$  is still asserted by a local address decoder).  $\overline{\text{RAMSEL}}$  is asserted one clock cycle after the local address strobe, and is only negated after  $\overline{\text{KAS}}$  is deasserted. The SCV64 only supports reflected cycles once the VMEBAR register has been written to. This is independent of whether the device has powered up with AUTOBAR mode enabled.

When the SCV64 is in loopback mode, accesses to the slave image will result in the SCV64 generating a VME cycle (see “Test and Diagnostic Modes” on page 2-126).

### 2.9.11 VSBbus Access

When the CPU accesses a VSB address space (see Figure 2.9 in “CPU Memory Map” on page 2-45), a local address decoder asserts  $\overline{\text{VMEOUT}}$  as it would for any VME access. However, instead of initiating a VME cycle, the SCV64 decodes the access by asserting  $\overline{\text{VSBSEL}}$ . This signal directs the cycle to an external device which handles the VSBbus/local bus interface.  $\overline{\text{VSBSEL}}$  is deasserted only after  $\overline{\text{KAS}}$  and  $\overline{\text{VMEOUT}}$  are negated.



*Note that  $\overline{\text{VSBSEL}}$  may be assigned other device or memory select functions.*

### 2.9.12 BI-Mode

BI-mode completely isolates the local bus from the VMEbus, except for the location monitor (see “BI-Mode” on page 2-119 for BI-mode initiation). If the SCV64 enters BI-mode while there are entries in the RXFIFO, the RXFIFO will still complete all pending write cycles. However, all TXFIFO entries will be lost.

If the CPU attempts a VME access during BI-mode, the SCV64 ignores the cycle and the local bus times out with  $\overline{\text{KBERR}}$ . However, the SCV64 registers and the board’s own VME slave image are still accessible to the CPU. Address decoding for VSB access will also remain active, though the board design may externally block such an access during BI-mode.

## 2.10 Location Monitor and LMFIFO

The Location Monitor is provided to assist with inter-processor or inter-process communication. It incorporates a Location Monitor message FIFO (LMFIFO) which allows either 16 or 32-bit messages to be sent between processes. While there are entries in the LMFIFO, the SCV64 sets the LMHD bit in the DCSR register (Table A.6) and generates an interrupt to the local CPU by asserting  $\overline{\text{LMINT}}$ .

The Location Monitor resides at the top longword (32 bits) of each A32 and A24 slave image and is accessible from both the local and VMEbus. The bottom (even) word of the Location Monitor is used for 16-bit messages. Since the Location Monitor replaces the longword of memory which otherwise exists at that address, writes to the Location Monitor are not entered into memory. In addition, read accesses to the Location Monitor result in a bus error.

The LMFIFO is a 32-bit wide and 31 longword deep message queue for incoming data. Longwords from the Location Monitor are written to the LMFIFO as space becomes available. If an even word has been written to the Location Monitor, then the upper 16 bits in the LMFIFO entry are set to one (1). If the LMFIFO is full when a Location Monitor write is made, the cycle receives no response. The write will be accepted any time before bus time-out (see “Bus Timer” on page 2-34) if the local CPU reads the oldest entry in the FIFO thus opening a position for a new message. However, if a position does not become available before bus time-out, then the cycle ends with a bus error.

A local interrupt is generated ( $\overline{\text{LMINT}}$  asserted) as long as there are entries in the LMFIFO.  $\overline{\text{LMINT}}$  is only negated when the LMFIFO is empty at the end of the LMFIFO read cycle. The status of the LM FIFO is indicated by the state of the LMHD bit in the DCSR register (see Table A.6). The LMHD bit is set while there is data in the LMFIFO.



*Note that if the LMFIFO is read while it is empty and a message arrives from the VMEbus, then unstable data may be put onto the local bus. Do not read the LMFIFO unless the LMHD bit is set and  $\overline{\text{LMINT}}$  is asserted.*

Messages in the LMFIFO are read from the LMFIFO Register (see Table A.20). Reads to the empty LMFIFO return only undefined data (with  $\overline{\text{KDSACK}}_n$ ). Since  $\overline{\text{LMINT}}$  is only asserted while there are entries in the LMFIFO, an interrupt service routine or polling of the  $\overline{\text{LMINT}}$  signal can be used to test for entries in the LMFIFO.

The CPU may write to its own Location Monitor through its slave image as it would any address on the VMEbus (i.e. asserting  $\overline{\text{VMEOUT}}$  with the address). The SCV64 address decoder responds by redirecting the message to the LMFIFO. From there the message is handled in the same manner as a write from the VMEbus side.

If the race condition occurs in which the local CPU and VMEbus write to the Location Monitor at the same time, the race condition is resolved and the writes queued without loss of either of the messages (providing space is available in the LMFIFO).

One function of the Location Monitor is as a command to exit BI-mode. When there is a write to the Location Monitor and BI-mode initiators are absent, the SCV64 is removed from BI-mode and the BIMODE signal is cleared (see “BI-Mode” on page 2-119).

## 2.11 DMA Controller

The SCV64 performs all BLT and MBLT cycles through the DMA controller; single cycle transfers may be accomplished with or without DMA. Since DMA cycles are always decoupled, the operation of the DMA controller is closely associated with the RXFIFO and TXFIFO (see “Data Path” on page 2-36).

In either DMA reads or DMA writes, the DMA controller always reads from a source address and writes to a FIFO (the FIFO entry will contain the destination address). The fundamental differences between a DMA read and write depend upon,

- which bus (local or VME) is required in order to access memory, and
- whether the DMAC writes to a RXFIFO or TXFIFO.

### 2.11.1 DMA Initialization

The steps for initiating both DMA reads and writes are given in Figure 2.30 on page 2-103. In summary, the steps are:

1. clear the DCSR register (so that a new DMA transfer may begin, see “DMA Completion and Error Checking” on page 2-108),
2. ensure that the SCV64 is in decoupled mode,
3. set the appropriate address and data modes (see “Addressing and Data Transfer Modes” below),
4. fix the direction of transfer (DMA read or write),
5. provide the requisite source and destination addresses, and
6. program the transfer count.



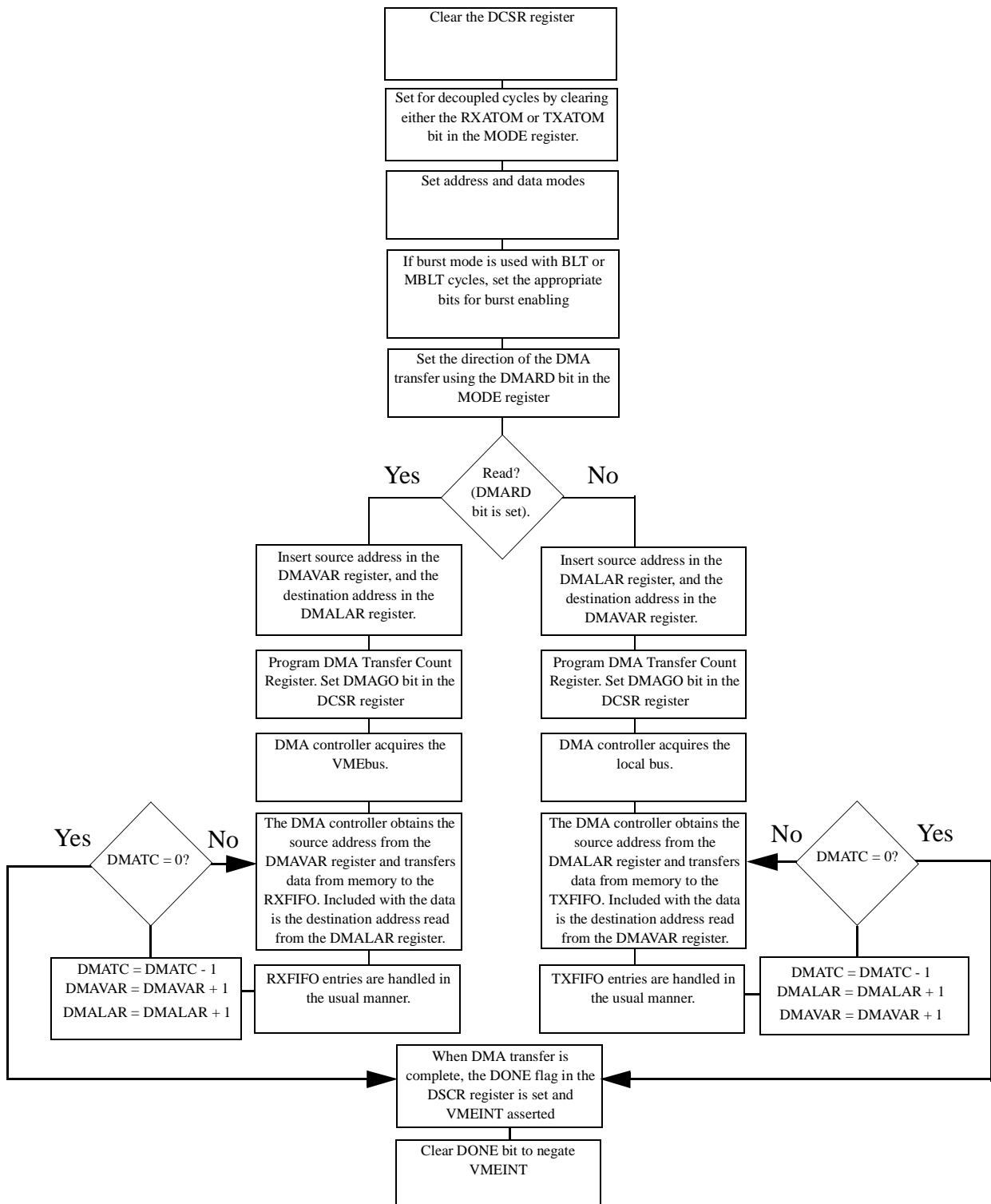


Figure 2.30 : Procedural Steps in Normal DMA Reads and Writes.

Setting the decoupled mode and selecting the direction of transfer are done through the single bits given in Figure 2.30 on page 2-103. However, many options exist for the addressing and data transfer modes, and address registers will be used differently depending upon the direction of transfer and the selected address mode. These various considerations are discussed below.

### 2.11.2 Addressing and Data Transfer Modes

The source and destination addresses for a DMA data transfer are obtained from two registers: the DMA Local Address Register (DMALAR, Table A.3), and the DMA VME Address Register (DMAVAR, Table A.4). DMALAR is the starting local address and DMAVAR is the starting VME address for the DMA transfer (see Figure 2.30). The association of each register (DMALAR or DMAVAR) with the source or destination address is dependant on the direction of the data transfer (DMA read or DMA write).

The various addressing and data transfer modes used in DMA operations are selected using the MODE register (see Table A.21). The required bit settings for the different modes are given below in Table 2.27. The available address modes are A64, A32, and A24. All address modes can be combined with the two data modes, D32 and D16. However, only A64 and A32 may be combined with MBLT (D64). The DPRIV bit in the MODE register can be used to switch between non privileged and supervisory AM codes. When performing DMA transfers the SCV64 always uses a variant of a data access AM code (i.e there is no mechanism to switch between program and data variant AM codes).

Note that for A64 transfers, the MA64BAR register (Table A.23) is used for the upper 32 bits of the master's (SCV64) address.

The table below lists the different operational modes available for the DMA controller, and the necessary programming to obtain that mode. Note that some modes are mutually exclusive. For example, the BLT bit cannot be set at the same time as the MBLT bit when attempting to perform a MBLT cycle.



*Caution: The DMA MODE register bits must be programmed before any changes are made to the DMALAR, DMAVAR, or DMATC registers.*



*Caution: The DMATC or DMAVTC registers cannot be read while a DMA transfer is taking place.*



*Caution: If a DMA BLT/MBLT transfer finishes at or within the last data transfer address before a block boundary condition as defined by the subsequent DMA transfer, and the subsequent DMA BLT/MBLT transfer begins at an address that is one data beat before the address boundary condition, then you must clear the DMAVAR register before programming the VME address for the subsequent DMA cycle. For example, if a DMA D16 BLT transfer completed its last word transfer at address 7f6, 7f8, 7fa, or 7fc, and the next DMA cycle to be performed was an MBLT transfer beginning at address 7f8 (one double longword before the address boundary) then you must clear the DMAVAR register before beginning the programming sequence for the MBLT DMA transfer.*

**Table 2.27 : Bit Settings for DMA Transfer Modes**

Mode	Pertinent Bits in MODE Register (Table A.21)									
	DMARD	DMAA64	DMA24	MBLT	DMAD16	BLT	DPRIV	DISRX	RXATOM	TXATOM
DMA read	1	x	x	x	x	x	x	0 for DMA Read	0 for DMA Read	0 for DMA Write
DMA write	0	x	x	x	x	x	x			
A64	x	1	0	1	x	0	0			
A32	x	0	0	x	x	x	x			
A24	x	0	1	x	x	x	x			
MBLT(D64)	x	any address mode		1	0	0	x			
D32	x	any address mode		0	0	x	x			
D16	x	any address mode		0	1	x	x			
BLT	x	any address mode		0	x	1	x			
non-privileged	x	x	x	x	x	x	0			
supervisory	x	x	x	x	x	x	1			

When block transfers are received from the VMEbus, the SCV64 increments the local address for each transfer using the LAG register (Table A.24). Each cycle within a block transfer entering the RXFIFO receives a unique address from the LAG register.

The SCV64 provides a burst mode which optimizes DMA operations involving BLT or MBLT cycles. For DMA reads, the SCV64 can initiate burst write cycles from the RXFIFO to local memory. With a 25 MHz local clock, burst writes allow data transfer rates up to 50 MBytes/sec. Burst writes are enabled by setting the FIFOBEN bit in the MODE register (Table A.21). The BLEN1 and BLEN0 bits in the MODE register set the maximum burst length (4, 8, 16, or 32 longwords). The default setting for maximum burst length is 4 longwords. The SCV64 only performs longword bursts. If you have programmed the DMA controller to perform D16 block transfers with bursts enabled, then the SCV64 will initiate single word cycles on the local bus.

If the FIFOBEN bit is set, then the SCV64 will initiate burst writes when there are 2 sequential MBLT entries or 4 sequential BLT entries in the RXFIFO. The SCV64 checks the BLKST bit in RXCTL register to test for the beginning of a block. If this bit is set, the SCV64 counts the number of BLT or MBLT entries appearing in series. A BLT or MBLT entry is indicated by the BLT and MBLT bits in the RXCTL register.

If a sufficient number of BLT or MBLT entries are queued, the SCV64 initiates a burst write. The burst write is terminated if the SCV64:

- encounters a different entry type,
- $\overline{\text{KBGR}}$  is negated,
- completes its maximum burst length, or
- empties the RXFIFO.

For DMA writes, the DMA controller can initiate burst read cycles from local memory to the TXFIFO. Burst reads are enabled by setting the DMABEN bit in the MODE register (Table A.21). The maximum burst length (4, 8, 16, or 32 longwords) is set using the BLEN1-0 bits in the same register. If burst mode has been set, the DMA initiates a burst read by asserting  $\overline{\text{KAS}}$  and issuing a local function code of 3.  $\overline{\text{KDSACK1}}$  is sampled on the second rising edge of KCLK after the cycle is initiated and on every subsequent rising edge. If  $\overline{\text{KDSACK1}}$  is asserted on a rising clock edge, then data is transferred on the next falling clock edge. One longword is transferred per clock cycle.

All burst start addresses must be longword aligned for BLTs and double-longword aligned for MBLTs. Data enqueued into the RxFIFO from an MBLT transfer is dequeued on the local bus in the following order: data from the VMEbus address lines (bytes 0-3) is presented first; then the data from the VMEbus data lines (bytes 4-7 of the MBLT transfer) is presented. In order to keep local address registers updated, burst cycles are automatically terminated and restarted at every burst-length boundary. For example, for a programmed burst of 8 longwords starting at address 0x08H, the SCV64 will terminate and restart the burst at addresses 0x20H. Bursts may terminate on any longword aligned address for BLTs and any double-longword aligned address for MBLTs.

Burst read cycles are terminated if:

- the maximum burst length is reached,
- the DMA transfer count is completed (see “Data Transfer Counts” on page 2-107), or
- the TXFIFO fills.

If the TXFIFO fills, the SCV64 will release the local bus before any additional cycles are completed. During BLT transfers, the local bus is automatically released when the TXFIFO contains 15 entries (the TXFIFO is 15 stages deep). During MBLT cycles, the local bus is automatically released when the TXFIFO contains 14 entries (since each MBLT cycle requires two local bus cycles).

### 2.11.3 Data Transfer Counts

The size of the data transfer is controlled by both the DMATC register (Table A.5) and the DTCSIZ bit in the MODE register. When the DTCSIZ bit is cleared (the default setting), the DMATC register will count up to 4096 longword transfers (16 Kbytes). If the DTCSIZ bit is set, then the DMATC register will count up to 1,048,576 longword transfers (4 Mbytes).

Transfer count for VMEbus cycles is recorded in the DMAVTC register (Table A.25). The difference between the value in this register and the value in the DMATC register is equal to the number of entries in the RXFIFO or TXFIFO.



*Note that the DMA Local and VMEbus Address Registers and the Transfer Count Register may change while being read, unless the DMA is stopped before reading.*

### 2.11.4 FILL Option

During a DMA write, the DMA controller transfers data from local memory to the TXFIFO. Past this point the DMA controller is no longer involved in data transfer. The TXFIFO entries are handled by the VME cycle generator (see “Data Path” on page 2-36). In normal operation, the SCV64 requests the VMEbus as soon as entries appear in the TXFIFO. However, while the device is in DMA mode, the SCV64 can be programmed to request the VMEbus only when the TXFIFO is filled. This option is programmed by setting the FILL bit in the MODE register (see Table A.21). The FILL mode is useful in applications involving the transfer of large batches of data.

### 2.11.5 No Release Option

Once the TXFIFO has obtained the VMEbus, it can be programmed using the NOREL bit in the MODE register (Table A.21) to not release the bus unless otherwise instructed by the requester block (much like ROR mode, see “Release On Request and Release When Done” on page 2-10). Clearing the NOREL bit causes the SCV64 to release the VMEbus when there are no entries remaining in the TXFIFO (much like RWD). The bus ownership timer (see “Bus Timer” on page 2-34) can be used to control the duration of No Release mode.



*Caution: If the TXFIFO is set for No Release and the Requester is set for RWD (see “VMEbus Requester” on page 2-8), then the SCV64 will not release the VMEbus unless it times out or receives BCLR\*.*

### 2.11.6 DMA Completion and Error Checking

When the DMA transfer is completed:

- the DONE flag in the DCSR register (Table A.6) is set,
- the DMAGO bit in the DCSR register is cleared, and
- the  $\overline{\text{VMEINT}}$  pin is asserted.

The DMA can be stopped before completion of data transfer by clearing the DMAGO bit. Under this circumstance, the DONE bit and the  $\overline{\text{VMEINT}}$  pin cannot be monitored for completion of the DMA transfer. When stopped, the DMA controller will complete its transfer on the source bus within one or two cycles. Before reprogramming another DMA transfer, ensure that the previous DMA transfer completed on the local bus.

The DMA transfer will also stop before a complete data transfer if:

- there is a local bus error,
- there is a VMEbus error, or
- the SCV64 receives a late local bus error.

A late local bus error is detected only if the SCV64 has been programmed by setting the BERRCHK bit in the MODE register (Table A.21). Setting this bit allows the SCV64 to check for local bus errors one local clock cycle after completion of DMA.

The CERR bit in the CSR register will be set if incompatible options are programmed in the MODE register. The CERR will be set if any of the following register bits are both programmed to “1”:

- DPRIV and DMA64,
- DMARD and DISRx,
- MBLT and BLT,
- DMA64 and MBLT,
- DMA64 and DMA24,
- DMARD & RXATOM,
- DMARD and LPBK, or
- TXATOM

In the event of one of the local bus errors listed above, the DMAC clears the DMAGO bit, asserts the  $\overline{\text{VMEINT}}$  pin, but does not set the DONE flag. In the event of a VMEbus error, the DMAGO bit may or may not be cleared. The cleared DONE flag indicates that a bus error has terminated data transfer. The type of bus error is indicated by the following bits in the DCSR register:

- LBERR is set for a local bus error when the RXFIFO has the bus,
- VBERR is set for a VMEbus error when the TXFIFO has the bus, and
- DLBERR is set for a local bus error when the DMA is performing local reads.

The error bits indicate which registers (TXFIFO, RXFIFO or DMA) the user should read for data recovery. Note that the DONE bit will be set after DMA completion even while there are entries remaining in the FIFO. Therefore, a bus error can occur after DMA completion but during data transfer. The user should always poll the error bits to ensure that no errors have occurred. The  $\overline{\text{VMEINT}}$  pin will be asserted (and the DONE bit set) when the transfer has completed. These completed conditions apply:

- during DMA Reads or DMA block writes when the transfer has completed on the VMEbus, and
- during single DMA writes when the cycle has finished on the local bus.

## 2.12 Resets, Clocks and Timers

### 2.12.1 Resets

#### 2.12.1.1 Local Reset

The SCV64 asserts Local Reset ( $\overline{\text{LRST}}$ ) under any one of the following conditions:

- power-up reset ( $\overline{\text{PWRRST}}$ ),
- external reset ( $\overline{\text{EXTRST}}$ ),
- VMEbus system reset ( $\text{SYSRST}^*$ ),
- software reset using the  $\text{SWRST}$  bit in the  $\text{GENCTL}$  register (see Table A.28).
- $\text{BG0IN}^*$  asserted while the SCV64 is Syscon.

All circuitry in the SCV64 is held reset while  $\overline{\text{PWRRST}}$  is asserted. When this signal is released, the SCV64 keeps  $\overline{\text{LRST}}$  asserted a further 0.25 seconds.



*Caution: The SCV64's KCLK input must be clocked in order for LRST to negate.*

The  $\overline{\text{EXTRST}}$  pin is intended to be driven by any on-board logic that needs to initiate a local reset. The  $\overline{\text{LRST}}$  signal is asserted immediately and kept asserted for another 0.25 seconds after  $\overline{\text{EXTRST}}$  is negated.

Local reset is also asserted for the duration of any VMEbus  $\text{SYSRST}^*$  assertion. Since the VMEbus specification ensures that  $\text{SYSRST}^*$  be driven for at least 200 ms, the SCV64 does not extend  $\overline{\text{LRST}}$  assertion as it does in the reset conditions described above.

Software reset is controlled through the  $\text{SWRST}$  bit in the  $\text{GENCTL}$  register (Table A.28). The reset circuitry captures the high value of  $\text{SWRST}$  and asserts  $\overline{\text{LRST}}$  and  $\text{SYSRST}^*$  for 0.25 seconds.

Local reset only deasserts after  $\text{SYSRST}^*$  has been deasserted, since  $\text{SYSRST}^*$  directly asserts  $\overline{\text{LRST}}$ .  $\text{SYSRST}^*$  is an open collector based line on the backplane, so  $\overline{\text{LRST}}$  does not actually negate until the last board in the system releases  $\text{SYSRST}^*$ .



### 2.12.1.2 System Reset

The SCV64 asserts system reset (SYSRST\*) under any one of the following conditions:

- power-up reset pin ( $\overline{\text{PWRRST}}$ ) is asserted,
- the VMEbus BG0IN\* pin is asserted while the SCV64 is Syscon,
- software reset using the SWRST bit in the GENCTL register (Table A.28).

As with Local reset, power-up causes SYSRST\* to be asserted for 0.25 seconds.

Since the BGnIN\* lines are not driven by the arbiter when the SCV64 is in slot 1 (i.e. Syscon), the BGnIN\* pin functions may be redefined. In the SCV64, the BG0IN\* pin is used for off-board reset inputs (see “External Inputs” on page 2-34). Noise sensitivity is reduced with a digital pulse filter, which must record four successive low samples on BG0IN\* before SYSRST\* is asserted. Therefore, there will be a 43 to 57 ms latency between BG0IN\* assertion and system reset.

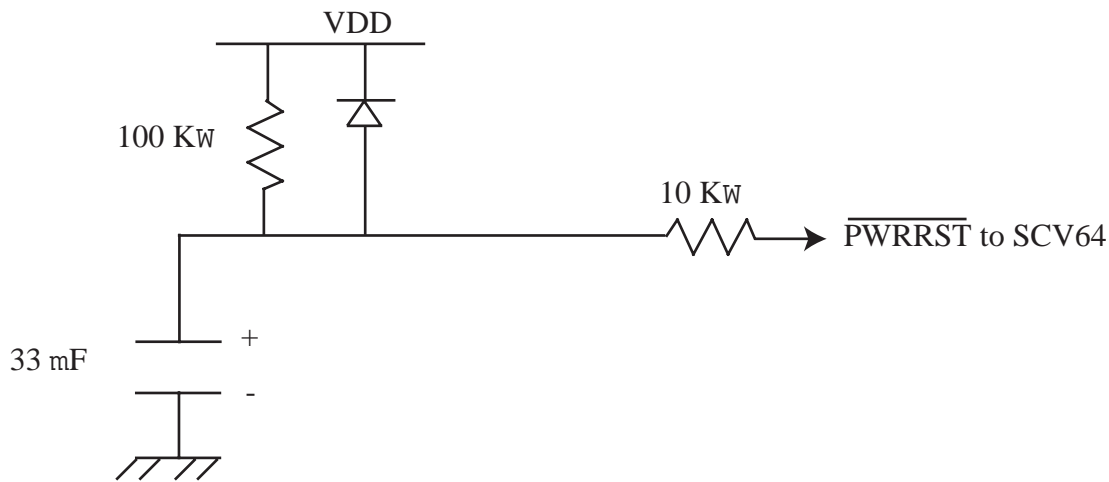
The same control bit, SWRST, in the GENCTL register (Table A.28) that initiates a local reset also causes SYSRST\* to be asserted for 0.25 seconds.

### 2.12.1.3 Power-up Reset

The  $\overline{\text{PWRRST}}$  input is passed through a Schmitt trigger input circuit so that noise expected from RC timing circuits is rejected. The  $\overline{\text{PWRRST}}$  pin directly resets all circuits in the SCV64, and should be held low for a minimum of 100 ns after power becomes stable.

Reset through the  $\overline{\text{PWRRST}}$  pin sets the PWRUP bit in the STAT1 register (Table A.27), thus indicating to the user that the last reset was due to power-up. This bit is cleared under all other reset conditions.

Figure 2.31 depicts a typical power-up reset circuit.



**Figure 2.31 : Sample Power-up Reset Circuit**

#### 2.12.1.4 Reset Effects on SCV64 Internal Resources

Most modules in the SCV64 are affected by  $\overline{\text{LRST}}$  or  $\text{SYSRST}^*$ , but not both. The effect of  $\overline{\text{PWRRT}}$ ,  $\text{SYSRST}^*$ , and  $\overline{\text{LRST}}$  on the pertinent functional blocks is described below. A summary of the  $\overline{\text{LRST}}$  and  $\text{SYSRST}^*$  initiators is also given.

- $\overline{\text{PWRRT}}$  (input pin)

Resets every circuit in the SCV64.

Causes  $\text{SYSFAIL}^*$  and  $\overline{\text{SYSFLED}}$  to be asserted. These signals can be negated by clearing the  $\text{SYSFAIL}$  bit in the  $\text{MISC\_CTL}$  register.

- $\overline{\text{LRST}}$  (by  $\overline{\text{PWRRT}}$ ,  $\overline{\text{EXTRST}}$ ,  $\text{SYSRST}^*$ , or software)

Resets:

- all control registers
- the local bus requester
- local bus timer
- tick timer
- the interrupt handler, and
- the VMEbus interrupt generator

Does not affect any clocks or VMEbus services

Does not affect the watchdog timer, unless the initiator is  $\overline{\text{PWRRST}}$

Places the board in BI-mode.

Causes  $\text{SYSFAIL}^*$  and  $\overline{\text{SYSFLED}}$  to be asserted. These signals can be negated by clearing the  $\text{SYSFAIL}$  bit in the  $\text{MISC\_CTL}$  register.

- $\text{SYSRST}^*$  (by  $\overline{\text{PWRRST}}$ ,  $\text{BG0IN}^*$ , software or another board)

Resets the following VMEbus related modules:

arbiter

requester

IACK daisy chain driver

Resets all modules specified under  $\overline{\text{LRST}}$ , because it is also an  $\overline{\text{LRST}}$  initiator.

Places the board in BI-mode.

Causes  $\text{SYSFAIL}^*$  and  $\overline{\text{SYSFLED}}$  to be asserted. These signals can be negated by clearing the  $\text{SYSFAIL}$  bit in the  $\text{MISC\_CTL}$  register.

Clocks and the watchdog timer remain unaffected

## 2.12.2 Clocks

### 2.12.2.1 Clocks Required

The SCV64 operation requires two clocks:

- a constant 32 MHz input to generate clocks of specified frequency, and
- a CPU clock for sections that must be synchronous to the CPU.

The 32 MHz input:

- generates all timers,
- synchronizes sampling of several local and VME signals,
- synchronizes operation of internal state machines, and
- is used for auto-calibration of internal delay lines.

The  $\text{SYSCLK}$  frequency is defined as 16 MHz, although the SCV64 does not depend on this value.

Some logic in the register local bus interface depends on timing between CPU signals and  $\text{KCLK}$ . These timing requirements usually necessitate a direct connection between the CPU and the SCV64  $\text{KCLK}$  pin (with its related signal pins  $\overline{\text{KAS}}$  and  $\overline{\text{KDS}}$ ) without intervening buffers or signal skew.

### 2.12.2.2 Clocks Generated

The clocks generated by the SCV64 are:

• VMEbus system clock	SYSCLK
• master baud rate clock	BAUDCLK
• 14 $\mu$ s clock	C14US
• 8 MHz clock	C8MHZ
• three timers	
local bus timer	no direct pin
tick timer	$\overline{\text{TICK}}$
watchdog timer	$\overline{\text{WDOG}}$

The clocks are described as follows:

#### **SYSCLK**

A 16 MHz, 50% duty cycle signal generated while the SCV64 is the Syscon. The clock stops only while  $\overline{\text{PWRRST}}$  is asserted, and is driven only when the SCV64 is the Syscon.

#### **BAUDCLK**

A 2.4615 MHz (32 MHz divided by 13) clock, with a 6/13 duty cycle, provided for use by a baud rate generator, which must divide it down to standard baud rates. The clock stops only while  $\overline{\text{PWRRST}}$  is asserted.

#### **C14US**

A 14  $\mu$ s clock (32 MHz  $\div$  448), 50% duty cycle (approximately) with  $\pm 1$  clock sampling jitter, synchronized by KCLK to the local CPU. The relationship between KCLK and C14US is shown in the timing specifications in Appendix B. This clock is typically used by dynamic memory refresh control circuits. The clock runs as long as  $\overline{\text{PWRRST}}$  is not asserted and KCLK is running.

#### **C8MHz**

General purpose 8 MHz clock output.

## 2.12.3 Timers

### 2.12.3.1 Local Bus Timer

The Local Bus Timer defaults to enabled after reset and begins counting as soon as the local data strobe,  $\overline{\text{KDS}}$ , is asserted. If  $\overline{\text{KDS}}$  is still asserted after 512  $\mu\text{s}$ , the SCV64 asserts  $\overline{\text{KBERR}}$  ( $\overline{\text{VMEINT}}$  is not asserted). The  $\overline{\text{KBERR}}$  signal remains low until  $\overline{\text{KDS}}$  is deasserted, then goes high until the falling edge of the next KCLK pulse, when it is tristated. The assertion of  $\overline{\text{KBERR}}$  is not synchronized to KCLK.

The Local Bus Timer does not use either of the local data strobe acknowledgment signals ( $\overline{\text{KDSACK0}}$  or  $\overline{\text{KDSACK1}}$ ). Therefore, if some device asserts one of these signals when the timer is about to generate  $\overline{\text{KBERR}}$ , both of the signals will be ignored and may be asserted to the CPU. This occurrence may create an illegal timing condition if the local bus master is not a 680x0 CPU.

The Local Bus Timer is disabled by the LTOEN bit in the GENCTL register (Table A.28). The timer should only be disabled in a system development environment, but not in any application. If a VMEbus lockout occurs when the timer is disabled, the local CPU hangs and the board stops accepting accesses from the VMEbus.

### 2.12.3.2 Tick Timer

The Tick Timer is used to schedule interrupts or other functions. The  $\overline{\text{TICK}}$  pin is driven low each time the timer expires, and stays low until software resets it through the CLRTIK bit in the STAT0 register (Table A.26). Clearing the CLRTIK bit only resets the TICK pin's output back to the inactive state and does not internally reset the timer. To use the tick signal, connect it externally to the destination device, usually a general purpose local interrupt. Tick timer output can only be read if the pin is connected to one of the interrupts.

The Tick Timer mode (normal or fast) is set using the TICKM bit in the CTL2 register (Table A.33).

Within normal and fast mode, various tick intervals can be selected using the TLEN1 and TLEN0 bits in the GENCTL register. In normal mode, the tick interval can be set to one of 5, 10, 50, or 100 ms (the default setting). The fast mode, with settings of 0.2, 0.4, 2, and 4 ms, offers tick intervals 25 times briefer than the normal mode.

**Table 2.28 : Setting the Tick Timer Interval**

Interval		Setting for TLEN 1,0
Normal Mode	Fast Mode	
5 ms	0.2 ms	00
10 ms	0.4 ms	01
50 ms	2 ms	10
100 ms	4 ms	11

### 2.12.3.3 Watchdog Timer

The watchdog timer is another general purpose timer supplied by the SCV64. In the absence of any power resets, the watchdog timer will continuously count a two second interval, assert the  $\overline{\text{WDOG}}$  pin for 200 ms, and reset its counter. The watchdog timer is only affected by three sources: the  $\overline{\text{PWRRST}}$  pin, the ENDOG bit in the MISC register (Table A.42) and the CLRDOG bit in the STAT0 register.

When the  $\overline{\text{PWRRST}}$  pin is asserted, counter reset is held. Neither local reset nor system reset have this effect, so the watchdog timer may time-out after these resets. The significance of a watchdog time-out will depend on the user-defined application for the timer. Writing to the CLRDOG bit will clear and restart the counter. If the CLRDOG bit is already 1 and the CPU writes another 1, the watchdog counter is not affected. This edge sensitivity helps protect against software failure while clearing the watchdog.

### 2.12.3.4 VMEbus Timers

The SCV64 provides a VMEbus ownership timer (see “Ownership Timer” on page 2-11) a VMEbus data transfer time-out (see “Bus Timer” on page 2-34) and a VMEbus arbitration timer (see “Arbitration Time-out” on page 2-33).

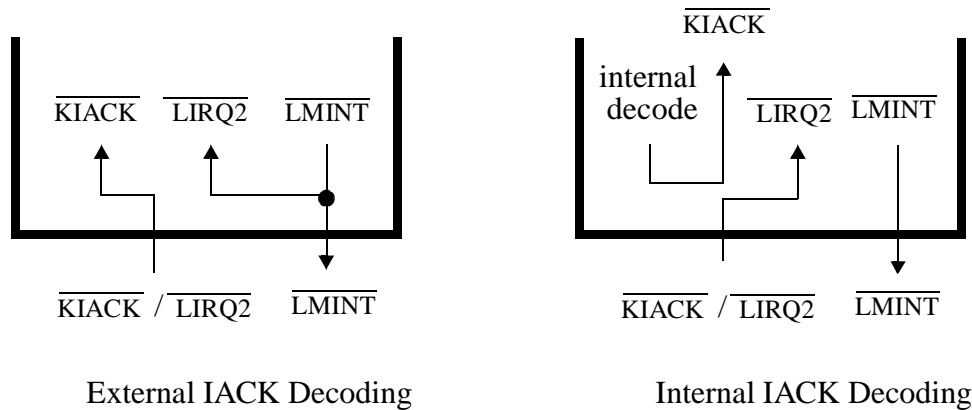
## 2.13 Power-up Modes

### 2.13.1 Local IACK Cycle Decoding

The function of the  $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin is determined by the state of the KFC1 pin immediately following a power reset condition (a low to high transition on  $\overline{\text{PWRRST}}$ ). If KFC1 is sampled high, the  $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin implements the  $\overline{\text{KIACK}}$  function; if KFC1 is sampled low, the  $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  pin implements the  $\overline{\text{LIRQ2}}$  function.

In  $\overline{\text{KIACK}}$  mode, the SCV64 accepts IACK decoding from an external device (see “Interrupt Acknowledge Cycles” on page 2-22), and the internal resources dedicated to the  $\overline{\text{LIRQ2}}$  interrupt are assigned to the location monitor interrupt ( $\overline{\text{LMINT}}$ ).

In  $\overline{\text{LIRQ2}}$  mode, local interrupt acknowledge cycles are decoded directly from the 68030/020 bus and  $\overline{\text{KIACK}}$  is generated internally.



**Figure 2.32 : Connections with Different Local IACK Decoding**

The  $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$  mode is indicated after reset by the EXTKIACK bit in the MISC register (Table A.42).

### 2.13.2 Auto SYSCON select

The SCV64 will become the VMEbus system controller if B<sub>Gin3</sub>\* is low upon a low to high transition on  $\overline{\text{PWRRST}}$  or  $\overline{\text{SYSRST}}$  (see “Syscon Determination” on page 2-30).

### 2.13.3 Local Arbiter Bypass

The local arbiter in the SCV64 may be bypassed by holding  $\overline{\text{KBGACK}}$  low during a low to high transition on power reset. This causes request level 1 to the SCV64 arbiter to be disabled, and decreases the arbitration delay for SCV64 requests to the local bus.  $\overline{\text{LBRQ1}}$  is ignored and  $\overline{\text{LBGR1}}$  is undefined in this mode. Any SCV64 request is mapped directly to  $\overline{\text{KBRQ}}$  and grants from the external arbiter come via  $\overline{\text{KBGR}}$ .

### 2.13.4 Automatic Base Address Programming

A slave boot mode (AUTOBAR) is available which facilitates design when no local intelligence is available to initialize the SCV64 base address. The AUTOBAR mode allows the SCV64 to power up with its slave image accessible from the VMEbus, facilitating the design of slave-only cards (see “Automatic Base Address Programming” on page 2-51).

If KFC2 is held high and KFC0 is held low during power reset, then the SCV64 will automatically load and enable its base address. The automatic base addressing function can be verified by reading the AUTOBAR bit in the MISC register (Table A.42). Note that if the function codes are not set as described above, then the base address will still be loaded but not enabled.



*Note:  $\overline{\text{SYSRST}}$ \* does not return the initial base address programmed at power-up. Instead, the initial base address will remain at the address given prior to the system reset with both A24 and A32 images enabled.*



## 2.14 BI-Mode

Bus isolation mode (BI-mode) disables communication between the local bus and the VMEbus. Daisy chain and Syscon functions remain operative, but the board ceases all master and slave activity on the VMEbus until BI-mode is released. By isolating boards from the VMEbus, the user can implement:

- hot-standby systems
- system diagnostics for routine maintenance, or
- fault isolation in the event of a card failure.

BI-mode is initiated (the BIMODE output pin becomes asserted) under any of these conditions:

- during resets ( $\overline{\text{EXTRST}}$ ,  $\overline{\text{PWRRST}}$ , or  $\text{SYSRST}^*$ ),
- when  $\text{IRQ1}^*$  is asserted (using ABI bit in the GENCTL register, Table A.28) and configured as a BI-mode initiator,
- when the SBI bit in the GENCTL register (Table A.28) is set, or
- after asserting  $\overline{\text{BITRIG}}$ .

While in BI-mode, all transceivers point inwards and all accesses to the board's VME slave image will time out on the VMEbus ( $\text{DTACK}^*$  is disabled). Similarly, if the local CPU attempts to access the VMEbus during BI-mode, the local cycle will time out. However, the SCV64 VME slave image and its internal registers are still fully accessible to the CPU. Address decoding for VSB accesses will also remain active.

The board will remain in BI-mode while any of the initiating conditions listed above are present. Once initiating conditions are cleared, BIMODE is released by either writing to the Location Monitor or asserting  $\overline{\text{BIREL}}$ . By tying  $\overline{\text{BIREL}}$  low, the SCV64 can be configured to automatically come out of BI-mode when initiating signals are removed.



*Note that BI-mode signals have higher priority than BI-mode release signals. Therefore, the SCV64 will not come out of BI-mode unless all initiating signals are absent.*

**Table 2.29 : VMEbus Signal Behavior in BI-mode**

VMEbus Signal Name	Normal Behaviour	BI-Mode Behaviour
A31 – A01	Bi-directional address lines	Ignored
ACFAIL*	Input signal for power failure	Unaffected
AM0 – AM5	Bi-directional address modifier	Input only
AS*	Bi-directional address strobe	Input only
BBSY*	Bi-directional requester signal	Tristate
BERR*	Input signal for VMEbus error	Disabled
BR0* – BR3*	Bi-directional bus request	Disabled
D31 – D00	Bi-directional data lines	Input only
DS1*, DS0*	Bi-directional data strobes	Input only
DTACK*	Bi-directional acknowledge	Disabled
IACK*	Interrupt handler output and decode input	Input only
IACKIN*	Input for IACK daisy chain	Unaffected
IACKOUT*	Output for IACK daisy chain	Unaffected
IRQ7* – IRQ2*	Bi-directional interrupt request lines	Disabled
IRQ1*	Bi-directional interrupt	Input only
LWORD*	Bidirectional signal to indicate transfer size	Input only
SERCLK, SERDAT	Not used	Unaffected
SYSCLK	Bidirectional signal for system clock	Unaffected
SYSFAIL*	Bi-directional signal for system failure	Unaffected
SYSRESET*	Bi-directional signal for system reset	Unaffected
WRITE*	Bi-directional write signal	Input only

## 2.15 Auto-ID

The Auto-ID function identifies the relative position of each board in the system, without using jumpers or on-board information. The ID number generated by Auto-ID can then be used to determine the board's base address.

Bypassing the use of jumpers through Auto-ID:

- increases the speed of system level repairs in the field,
- reduces the possibility of incorrect configurations, and
- reduces the number of unique spare cards that must be stocked.

In addition, if the system includes a VSB bus (which means no user defined pins are available), then the Auto-ID system provided by the SCV64 becomes even more useful.

### 2.15.1 The Auto-ID Cycle

After any system reset (assertion of SYSRST\*), the Auto-ID logic responds to the first level one IACK cycle on the VMEbus. The level one IACK\* signal can either be generated normally in response to an IRQ1\* signal or it can be synthesized without the use of IRQ1\*.

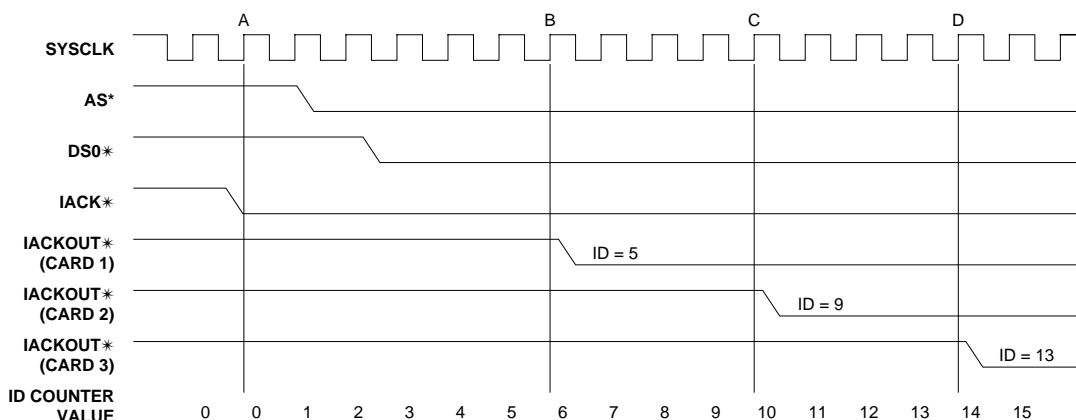
If the IRQ1\* signal is asserted to run the Auto-ID cycle, care must be taken to ensure that the asserting module is physically last in the IACK daisy chain. If the asserting module is in any other position, then the Auto-ID cycle will be halted at that location when the IACK cycle is accepted (see "IACK Daisy Chain Driver" on page 2-31).

Alternatively, a member of the 680x0 CPU family can synthesize a level one IACK without asserting IRQ1\* by running the following instructions:

```
MOVE.L      #0,D0
MOVEC. L    D0,SFC
MOVES.B     $xxxx xxx2,D0
```

Since the synthesized level one IACK\* entails no interrupt source requiring acknowledgment, the Auto-ID cycle will run through all modules in the system until the cycle terminates in BERR\*.

After the level one IACK\* signal has been asserted (either through IRQ1\* or with a synthesized version), the SCV64 in slot 1 counts five clocks from the start of the cycle and then asserts IACKOUT\* to the second board in the system (see Figure 2.33). All other boards continue counting until they receive IACKIN\*, then count four more clocks and assert IACKOUT\* to the next board. Finally, the last board asserts IACKOUT\* and the bus pauses until the data transfer time-out circuit ends the bus cycle by asserting BERR\*.



**Figure 2.33 : Timing for Auto-ID Cycle**

Because all boards are four clocks “wide”, the value in the clock counter is divided by four to identify the slot in which the board is installed; any remainder is discarded. Note that since the start of the IACK cycle is not synchronized to SYSCLK, a one count variation from the theoretical value of the board can occur. However, in all cases the ID value of a board is greater than that of a board in a lower slot number.

When the Auto-ID cycle is completed, the IDGOT bit in the STAT1 register (see Table A.27) is set, and the ID count is written to the ID register (see Table A.32). The value in the ID register is unique for each SCV64 equipped board. Note that synchronization errors and variations in IACKIN\* and IACKOUT\* delay may cause the ID value to vary between Auto-ID cycles.

If all boards contain SCV64 Auto-ID circuits, the board position can be directly identified from the ID register.

After the ID value has been determined and placed in the ID register, each board can either identify itself to a master CPU board which has access to system configuration information, or calculate its own base address using information in local EPROM.

If the boards identify themselves to some master CPU, then this master board must provide an empty table with a capacity for 256 entries. The software then:

1. verifies that the proper boards are present,
2. enters the base addresses and other configuration information into the table, and
3. signals the other boards that the table is ready to be used.

The Auto-ID function is compatible with any VMEbus board with IACKIN\* to IACKOUT\* delay less than 750 ns. Any delay greater than 750 ns may cause the cycle to be terminated with a BERR\*.

Boards without Auto-ID can be interspersed in the VMEbus system with boards having Auto-ID. Such a mixed system results in ID numbers that do not numerically correspond to slot numbers, but which can still be used to determine Auto-ID locations.



*Caution: If a read of the ID register is the first cycle which occurs on the SCV64's local bus after the Auto-ID cycle completes, then an incorrect value will be returned. The user must ensure that a register access (read or write, from either interface) is made to any of the other SCV64 registers before this register is accessed.*

### 2.15.2 Auto-ID Self Test

The Auto-ID logic is tested using the IDTST bit in the GENCTL register (see Table A.28). Auto-ID control logic is functioning correctly if writing 1 to the IDTST bit increments the ID value in the ID register.

The IDTST bit defaults to zero after reset.

## 2.16 Internal Delay Line Calibration

The SCV64 contains five individual digital delay lines providing delay values of 20, 30 and 40 ns. Each delay line is a series of logic elements, with access points or taps between each element. The delay lines are calibrated to automatically compensate for internal gate delay variations due to changes in the SCV64 supply voltage, operating temperature and semiconductor processing. Voltage and temperature change compensation is achieved using control circuits internal to the SCV64, while software control (see Appendix-F) is used to compensate for semiconductor processing. The following is a brief description of delay line function, but an understanding of their operation is not required by the user. Appendix-F provides all the necessary information required to implement delay function.

The calibration unit, seen in Figure 2.3, uses the 32 MHz reference clock to produce a digital value for the 20, 30 and 40 ns delays. The calibration targets indicate the appropriate tap select for the digital delay lines. The CAL\_xx fields of the DLST1 register give the values of these calibration unit targets. Each of the five delay lines is loaded with its target upon the negation of  $\overline{\text{LRST}}$ . Subsequent changes to the calibration targets are then tracked by the individual delay lines. This is autocalibrate mode, and is the default.



*Caution: All voltage, temperature and process compensation is lost if the offset fields are modified without factory instruction (see Appendix-F).*

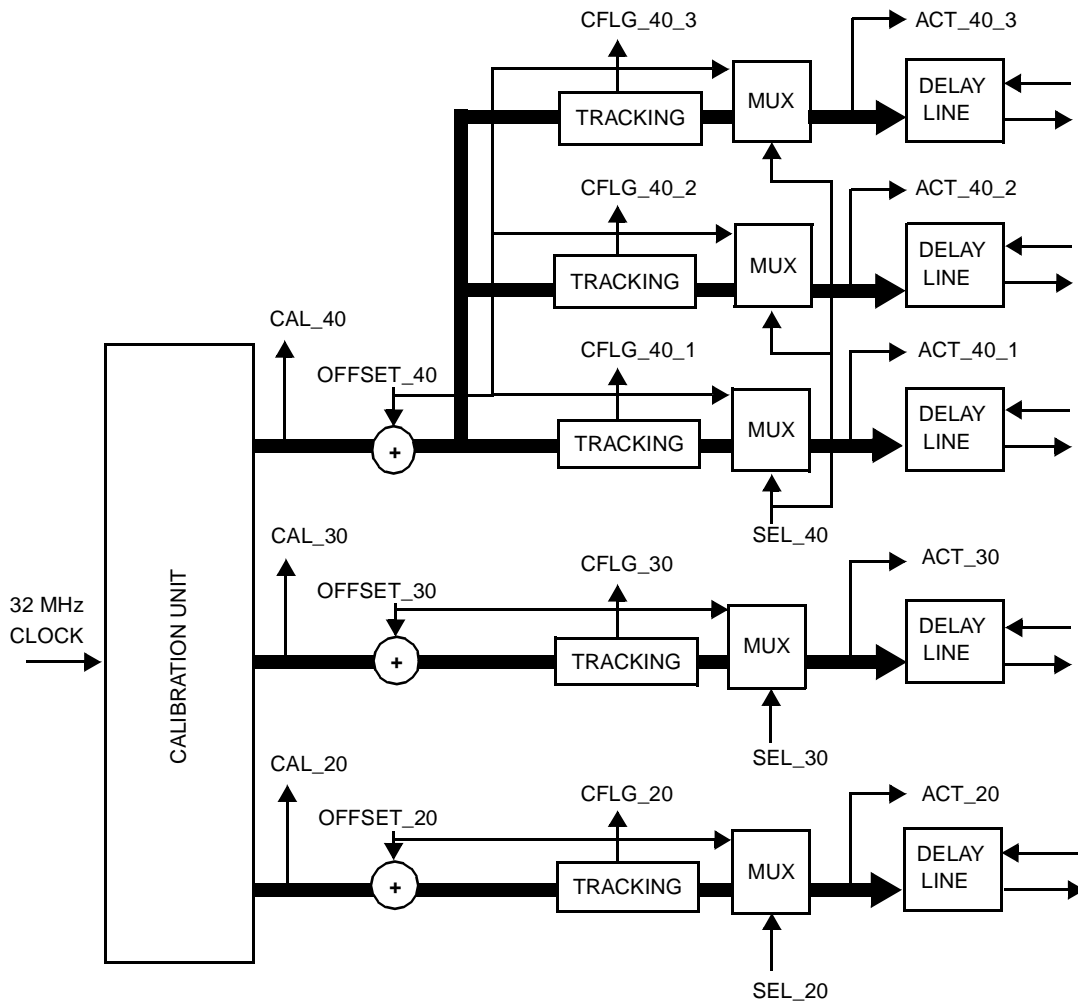


Figure 2.34 : Functional Block Diagram for Delay Lines

## 2.17 Test and Diagnostic Modes

### 2.17.1 Decoupled Write Diagnostics

TXFIFO data, address and control signals for a current write cycle may be read from the TXDATA, TXADDR, and TXCTL registers, respectively. Note that although these registers can be used to recover a current write cycle terminated by a VMEbus error, they do not show the subsequent cycle at the output stage of the TXFIFO. However, the subsequent cycle in the TXFIFO can be moved to the registers described above using the TXSHFT bit in the DCSR register (Table A.6).

The failed cycle read back from the RXFIFO registers may be retried by the CPU, but the retry will occur only after other writes queued ahead in the TXFIFO are completed. Note that the cycle is not automatically retried by the SCV64 when the VBERR flag is cleared; instead, the SCV64 proceeds with the next cycle in the TXFIFO.

### 2.17.2 Loopback Diagnostics

#### 2.17.2.1 Loopback Mode

By placing the SCV64 in loopback mode, any write cycle by the local CPU to its own VME slave image does not go directly to memory (see “Reflected Cycles” on page 2-99) but is instead sent out to the VMEbus and from there into the SCV64 RXFIFO. Once the cycle loops back into the RXFIFO, DTACK\* on the VMEbus is both asserted and received by the SCV64 to end the cycle.



*Note that VMEbus address and data transceivers drive the cycle onto the VMEbus, but the SCV64 loads the cycle into its RXFIFO from its own output pins, so the loopback mode does not test the external VMEbus buffers.*

Read cycles are not looped back, because the SCV64 needs local bus mastership to complete the read and the CPU is the current owner. Read cycles are always directed into memory with the RAMSEL pin in the same manner as a reflected cycle.

Loopback mode can be used to test:

- the base address and size of the slave image (see “VME Slave Memory Map” on page 2-49),
- the access protection (“Access Protection” on page 2-52), and
- the integrity of the RXFIFO and TXFIFO.



Address and data faults either on the VMEbus or in the VMEbus transceivers cannot be detected, since address and data signals are received by the SCV64 from its own output pins.

The base and size of the slave image can be tested in two ways. The local slave image detector can be verified while still in BI-mode. Without using loopback, reads or writes to memory are completed just as to the direct image. The second slave image detector exists on the VMEbus side and loopback mode or interactive testing using another VMEbus master can be used to verify that circuit.

Access protection can be verified as follows:

1. write to the slave image with no protection,
2. allow time for the write to occur, and then verify that the write succeeded,
3. enable protection, and check that the VBERR bit in the DCSR register is set when the write is repeated.

Integrity of the RXFIFO can be checked by disabling the RXFIFO and then shifting and unloading each cycle in the queue. As each cycle is shifted to the end of the RXFIFO, it can be read from the RXDATA, RXADDR, RXCTL registers. Note that this shift and read process can be done only when no other card is accessing the SCV64. The RXFIFO is disabled by setting the DISRX bit in the MODE register; setting the RXSHFT bit in the DCSR register shifts the RXFIFO ahead by one entry. The SCV64 clears RXSHFT after completing the shift.

Loopback mode is initiated by setting the LPBK bit in the MODE register. LPBK can be set or cleared at any time, since it only affects the path of CPU cycles to its own slave image (therefore, an access will not be in progress during programming). However, loopback mode only operates when the SCV64 is set to decoupled mode, and cannot be used in BI-mode.



*Note that data coherency is no longer assured during loopback mode. For example, the CPU can dispatch a decoupled write to its own memory through the SCV64, and then immediately begin a read to the same address on the local bus. The local read cycle would likely start before the write cycle is propagated through the VMEbus and back through the RXFIFO. Therefore, the local read would not return the most current data.*

### 2.17.2.2 DMA Loopback

The SCV64 can be programmed to perform a DMA loopback which is an effective way to test the functionality of the chip. During a DMA write, it is possible to program the VME address to correspond to the programmed slave image. The SCV64 reads the data from the local bus and writes it to the TXFIFO, which then completes the cycle on the VMEbus to its own slave image by latching the data into the RXFIFO and asserting DTACK\*. The SCV64 will then take local bus control away from the DMA controller, and write the data back to local memory.

DMA read loopbacks are similar to the DMA write loopbacks. The DMA controller reads from the VMEbus and the cycle is reflected back to the local side to read from local memory. The data is read from local memory and passed back through the SCV64's coupled path to the VMEbus. DTACK is asserted and received by the SCV64 and the data is queued into the RXFIFO. When the RXFIFO gains control of the local bus, the SCV64 writes the data back to the local bus. Although this technique can be used to DMA data between local resources, it is slow and involves tying up the VME.

### 2.17.2.3 Interrupt Loopback

The SCV64 can perform an interrupt loopback by asserting a VME interrupt using the VINT register. If that interrupt is enabled, it causes an interrupt to be asserted on the KIPL lines. A local IACK cycle is performed which causes the SCV64 to perform a VMEbus IACK cycle. The SCV64 supplies its own vector from the VMEbus Interrupter Vector Register (IVECT) and asserts DTACK\* and the VMEbus IACK cycle completes. The vector is then passed back to the local bus and the local bus IACK cycle is terminated by the SCV64 asserting  $\overline{\text{KDSACK}}_n$ .

### 2.17.2.4 JTAG Support

The SCV64 provides the following JTAG (IEEE P1149.1) pins:

- JTMS,
- JTCLK,
- JTDI,
- and JTDO.

The SCV64 presently has no internal JTAG controller. JTMS and JTCLK may be connected to ground or to the corresponding board level signals. JTDI and JTDO are internally connected, allowing subsequent implementation of a complete JTAG board level test.

## 3 Description of SCV64 Signals

The following detailed description of the SCV64 signals is organized according to these functional groups:

- VMEbus Signals
- Local Signals

### 3.1 VMEbus Signals

<b>BBSY*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECT</b>
VMEbus Bus Busy – signal driven by the SCV64 while it owns the bus.	
<b>BCLR*</b>	<b>ACTIVE LOW TRISTATE HIGH CURRENT BIDIRECT</b>
<p>VMEbus Bus Clear – requests that the current owner release the bus.</p> <p>The SCV64 Requester circuit can be configured to use or ignore this line. If the SCV64 is the Syscon, the Arbiter circuit asserts this line when higher priority requests are pending.</p>	
<b>BERR*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECT</b>
VMEbus Bus Error – a low level signal indicates that the addressed slave has not responded, or is signalling an error.	
<b>BG3IN*-BG0IN*</b>	<b>ACTIVE LOW INPUTS</b>
<p>VMEbus Bus Grant Inputs – The VME arbiter awards use of the data transfer bus by driving these bus grant lines low. The signal propagates down the bus grant daisy chain and is either:</p> <ul style="list-style-type: none"> <li>• accepted by a requester if it requesting at the appropriate level, or</li> <li>• passed on as a BGnOUT* to the next board in the bus grant daisy chain.</li> </ul> <p>If the SCV64 is the Syscon, then the BG0IN* signal can be used to trigger system reset (see “Off-Board Reset Input” on page 2-35).</p>	

### 3.1 VMEbus Signals<sup>(Continued)</sup>

<b>BG3OUT*_BG0OUT*</b>	<b>ACTIVE LOW OUTPUTS</b>
<p>VMEbus Bus Grant Outputs – Only one output is asserted at any time, according to the level at which the VMEbus is being granted.</p> <p>Three lines are propagated to BGnOUT* while the fourth BGnIN* may be terminated on the SCV64, if the SCV64 is requesting the bus at this level.</p> <p>If the SCV64 is configured as the VMEbus Syscon (i.e. the card is in slot 1), then a low level on BG0IN* causes the SCV64 to generate local and system reset.</p> <p>BG3IN* is used for <math>\overline{\text{SYSCON}}</math> determination. If BG3IN* is sampled low after <math>\overline{\text{PWRST}}</math>, then the SCV64 is enabled as the VMEbus Syscon</p> <p>If the SCV64 is enabled as the Syscon, then BG2IN* and BG1IN* can be read by software. Therefore, it is recommended that 10K<math>\frac{3}{4}</math> external pull-down resistors be used on BG2IN* and BG1IN* if the SCV64 is Syscon.</p>	
<b>BR 3-0*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECTS</b>
<p>VMEbus Bus Request Lines – these signals are asserted by the SCV64 requester when requesting use of the data transfer bus. The level of the request is programmed using the LVL1 and LVL0 bits in the VREQ register.</p> <p>If the SCV64 is the Syscon, the Arbiter logic monitors these signals and generates the appropriate Bus Grant signals.</p>	
<b>DTACK*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECT</b>
<p>VMEbus Data Transfer Acknowledge – DTACK* driven low indicates that the addressed slave has responded to the transfer.</p>	
<b>IACK*</b>	<b>TRISTATE HIGH CURRENT BIDIRECT</b>
<p>VMEbus Interrupt Acknowledge — indicates that the cycle just beginning is an interrupt acknowledge cycle.</p>	
<b>IACKIN*</b>	<b>ACTIVE LOW INPUT</b>
<p>VMEbus Interrupt Acknowledge In— Input for IACK daisy chain driver. If interrupt acknowledge is at same level as interrupt currently generated by the SCV64, then the cycle is accepted. If interrupt acknowledge is not at same level as current interrupt or SCV64 is not generating an interrupt, then the SCV64 propagates IACKOUT*.</p>	
<b>IACKOUT*</b>	<b>ACTIVE LOW OUTPUT</b>
<p>VMEbus Interrupt Acknowledge Out— generated by the SCV64 if it receives IACKIN* and is not currently generating an interrupt at the level being acknowledged.</p>	
<b>IRQ 7-2*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECTS</b>
<p>VMEbus Interrupts 7 through 2 – these interrupts map directly onto CPU interrupt levels 7 through 2.</p> <p>IRQ7-2* are individually maskable, but cannot be read. Any of them can be asserted by the SCV64 under software control.</p>	

### 3.1 VMEbus Signals<sup>(Continued)</sup>

<b>IRQ1*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECT</b>
<p>VMEbus Interrupt 1 – this signal can be as either a VME interrupt or a BI-mode initiator.</p> <p>If configured as a BI-mode initiator, then IRQ1* is asserted by writing to the ABI bit in the GENCTL register. The status of IRQ1* (asserted or negated) is monitored through the VI1 bit in the STAT0 register.</p>	
<b>SYSCLK</b>	<b>HIGH CURRENT BIDIRECT</b>
<p>VMEbus System Clock – generated by the SCV64 when it is the Syscon, and used by the Auto-ID state machine and ID counter.</p>	
<b>SYSFAIL*</b>	<b>OPEN DRAIN BIDIRECT</b>
<p>VMEbus System Failure – this signal is asserted using the SYSFAIL bit in the MISC register. When SYSFAIL* is asserted by the SCV64, <math>\overline{\text{SYSFLED}}</math> is also asserted. When SYSFAIL* is asserted on the VMEbus the SCV64 can signal a level 7 interrupt to the local CPU (See “Interrupt Handler” on page 2-18.)</p>	
<b>SYSRST*</b>	<b>ACTIVE LOW OPEN DRAIN HIGH CURRENT BIDIRECT</b>
<p>VMEbus System Reset – the SCV64 asserts SYSRST* under the following conditions:</p> <ul style="list-style-type: none"> <li>• on power-up, if <math>\overline{\text{PWRRST}}</math> is asserted (the SCV64 drives SYSRST* for 250 ms),</li> <li>• when BG0IN* is asserted and the SCV64 is the Syscon, or</li> <li>• if the software reset bit (SWRST) in the GENCTL register is set.</li> </ul>	
<b>VADDR 31 – 01</b>	<b>BIDIRECTS</b>
<p>VMEbus Address Lines 31 to 01 – during MBLT transfers, VADDR31-01 serve as data bits D63-D33. VADDR03-01 are used to indicate interrupt level on the VMEbus.</p>	
<b>VADDRROUT</b>	<b>ACTIVE HIGH OUTPUT</b>
<p>VMEbus Address Transceiver Direction Control – the SCV64 controls the direction of the address (VADDR31-01, VLWORD*) transceivers as required for master, slave and bus isolation modes.</p>	
<b>VAM 5 – 0</b>	<b>BIDIRECTS</b>
<p>VMEbus Address Modifier Codes – these codes indicate the address space being accessed (A16, A24, A32, A64), the privilege level (user, supervisor), the cycle type (standard, BLT, MBLT) and the data type (program, data).</p>	
<b><math>\overline{\text{VAS}}</math></b>	<b>ACTIVE LOW BIDIRECT</b>
<p>VMEbus Address Strobe – the falling edge of <math>\overline{\text{VAS}}</math> indicates a valid address on the bus. By continuing to assert <math>\overline{\text{VAS}}</math>, ownership of the bus is maintained during a RMW cycle.</p>	
<b>VDATA 31 – 00</b>	<b>BIDIRECTS</b>
<p>VMEbus Data Lines 31 through 00.</p>	

### 3.1 VMEbus Signals<sup>(Continued)</sup>

<b>VDATAOUT</b>	<b>ACTIVE HIGH OUTPUT</b>
VMEbus Data Transceiver Direction Control – the SCV64 controls the direction of the data transceivers as required for read, write and bus isolation modes.	
<b><math>\overline{\text{VDS}} 1 - 0</math></b>	<b>ACTIVE LOW BIDIRECTS</b>
VMEbus Data Strobes: <ul style="list-style-type: none"> <li>• During write cycles, the falling edge indicates valid data on the bus.</li> <li>• During read cycles, assertion indicates a request to a slave to provide data.</li> </ul>	
<b><math>\overline{\text{VLWORD}}</math></b>	<b>ACTIVE LOW BIDIRECT</b>
VMEbus Longword Data Transfer Size Indicator – this signal is used in conjunction with the two data strobes $\overline{\text{VDS}}$ and $\text{VADDR } 01$ to indicate the number of bytes (1 – 4) in the current transfer. During MBLT transfers, $\overline{\text{VLWORD}}$ , serves as data bit D32.	
<b>VSTRBOUT</b>	<b>ACTIVE HIGH OUTPUT</b>
VMEbus Address and Data Strobe Transceiver Direction Control – the SCV64 points the control strobe transceivers outward when it is the bus master. In all other cases, the transceivers are pointed inward, presenting a high impedance to the VMEbus.	
<b><math>\text{RETRY}^*/\overline{\text{VRMC}}</math></b>	<b>ACTIVE LOW BIDIRECT</b>
<p><math>\text{RETRY}^*</math>: VMEbus Retry – input indicating that current data transfer cannot occur, but should be retried.</p> <p><math>\overline{\text{VRMC}}</math>: Proprietary VMEbus RMW signal – Bidirectional signal used to signal a RMW cycle over the VMEbus. Using this signal allows multiple addressing during a RMW cycle.</p>	
<b><math>\overline{\text{VWR}}</math></b>	<b>ACTIVE LOW BIDIRECT</b>
VMEbus Write signal — timing of this signal is defined relative to the data strobes and indicates the direction of data transfer.	

## 3.2 Local Signals

BAUDCLK	OUTPUT
Baud Clock – this is a 2.4615 MHz clock signal generated from the 32 MHz input ( $32\text{ MHz} \div 13$ and 38.6% duty cycle) used by devices to generate baud rate clocks.	
BIMODE	ACTIVE HIGH OUTPUT
BI-mode Signal – generated by the SCV64 to indicate that logic on the card must isolate itself from the bus.	
$\overline{\text{BIREL}}$	ACTIVE LOW INPUT
BI-mode Release – $\overline{\text{BIREL}}$ releases the SCV64 from BI-mode and negates BIMODE if both VMEbus IRQ1*, $\overline{\text{BITRIG}}$ , and the SBI bit in the GENCTL register are not asserted. Otherwise, the BI-mode state is maintained.	
$\overline{\text{BITRIG}}$	ACTIVE LOW INPUT
BI-mode Trigger – asserting $\overline{\text{BITRIG}}$ puts the SCV64 into BI-mode in a similar manner as IRQ1* when IRQ1* is configured as a BI-mode initiator.	
C14US	OUTPUT
The 14 $\mu\text{s}$ clock signal is generated from the 32 MHz input ( $32\text{ MHz} \div 448$ and 50% duty cycle) and sampled by the falling edge of KCLK to synchronize C14US to CPU logic. This clock is typically used by DRAM refresh logic.	
C8MHZ	OUTPUT
8 MHz general purpose clock output.	
C32MHZ	INPUT
32 MHz clock signal used to generate clock signals and run state machines related to the VMEbus. This signal is used to generate the 16.0 MHz SYSCLK signal, and delay line calibration.	
$\overline{\text{EXTRST}}$	ACTIVE LOW INPUT
External Reset – this reset can be driven by any on-board logic requiring local reset. External reset affects sections of the SCV64 related to local functions, <i>but not those sections related to VMEbus functions</i> . SYSRST* is not generated.	
JTCLK	INPUT
JTAG test clock - not used	
JTDI	INPUT
JTAG test data - not used, connected internally to JTDO	
JTD0	OUTPUT
JTAG test data - not used, connected internally to JTDI	

## 3.2 Local Signals<sup>(Continued)</sup>

JTMS	INPUT
JTAG test mode select - not used	
KADDR 31 – 00	BIDIRECTS
Local Address Lines – KADDR03-01 are used to signal the interrupt level during interrupt acknowledge cycles.	
$\overline{\text{KAS}}$	ACTIVE LOW BIDIRECT
Local Address Strobe – used to qualify address on the local bus.	
$\overline{\text{KAVEC}}$	ACTIVE LOW OUTPUT
Auto-vector – asserted to the CPU during an interrupt acknowledge cycle indicating that the cycle be auto-vector by the CPU.	
$\overline{\text{KBERR}}$	ACTIVE LOW BIDIRECT
Local Bus Error – asserted to the local CPU to indicate a local bus time-out or invalid transfer. This signal requires an external pull-up resistor of approximately 4.7K $\frac{3}{4}$ .	
$\overline{\text{KBGACK}}$	ACTIVE LOW OPEN DRAIN BIDIRECT
Local Bus Grant Acknowledge – driven by the SCV64 during mastership of local bus. In arbiter active mode, it indicates ownership of local bus by the device driving the signal. In arbiter bypass mode, it may be driven to undefined state. This signal is used during power-up to set the arbiter mode. This signal requires an external pull-up resistor of approximately 4.7K $\frac{3}{4}$ to use the local arbiter or a pull-down resistor of a similar value if the local arbiter is bypassed (see “Local Bus Arbitration” on page 2-66).	
$\overline{\text{KBGR}}$	ACTIVE LOW INPUT
CPU Bus Grant – a signal from the CPU indicating that the bus is available. When $\overline{\text{KBGR}}$ is received and the SCV64 local arbiter is active, the SCV64 acknowledges bus grant (if it is the requester) or generates $\overline{\text{LBGR1}}$ to the local requester.	
$\overline{\text{KBRQ}}$	ACTIVE LOW OUTPUT
CPU Bus Request – asserted to the CPU from the SCV64 when $\overline{\text{LBRQ1}}$ is asserted.	
KCLK	INPUT
Local Clock – derived from the local CPU.	
KDATA 31 – 00	BIDIRECTS
Local Bus Data Lines 31 to 00 – data lines 31 to 00 are used to transfer data during SCV64 register accesses.	




### 3.2 Local Signals<sup>(Continued)</sup>

$\overline{\text{KDS}}$	ACTIVE LOW BIDIRECT		
Local Data Strobe – used in local cycles to indicate valid data during a write and a request for data during a read. The SCV64 does not monitor $\overline{\text{KDS}}$ when it is the local slave except during accesses to SCV64 registers.			
$\overline{\text{KDSACK1-0}}$	ACTIVE LOW BIDIRECTS		
Local bus data transfer and size acknowledge. The slave for the current cycle asserts one or both of these signals to acknowledge the cycle and indicates the size of transfer accepted.			
	$\overline{\text{KDSACK1}}$	$\overline{\text{KSACK0}}$	Data Transfer Size
	0	0	4 bytes
	0	1	2 bytes
	1	0	1 byte
	1	1	wait states inserted
$\text{KFC2-0}$	BIDIRECTS		
Local Function Codes – used to encode transfer type on the local bus. These signals are used during power-up to determine AUTOBAR configuration, and $\overline{\text{LIRQ2}}/\overline{\text{KIACK}}$ pin definition.			
$\overline{\text{KHALT}}$	ACTIVE LOW BIDIRECT		
CPU Halt Signal – may be used in conjunction with $\overline{\text{KBERR}}$ to resolve a Master/Slave deadlock.			
$\overline{\text{KIPL2-0}}$	ACTIVE LOW OUTPUTS		
CPU Interrupt Priority Level – The SCV64 channels all interrupt sources to 7 CPU interrupt levels. The $\overline{\text{KIPL2-0}}$ lines are used to encode the 7 CPU interrupt levels.			
$\overline{\text{KRM}\overline{\text{C}}}$	ACTIVE LOW BIDIRECT		
Local Read Modify Write – this signal indicates that the current cycle is part of a read-modify-write cycle.			
$\text{KSIZE 1 – 0}$	BIDIRECTS		
Local Bus Data Transfer Size – this code indicates the number of bytes (including current cycle) to be transferred to complete the operation.			
	$\text{KSIZE1}$	$\text{KSIZE0}$	Data Transfer Size
	0	0	4 bytes
	1	1	3 bytes
	1	0	2 bytes
	0	1	1 byte
$\overline{\text{KWR}}$	ACTIVE LOW BIDIRECT		
Write signal – used to indicate a read ( $\overline{\text{KWR}}$ negated) or write ( $\overline{\text{KWR}}$ asserted)			
$\overline{\text{L7IACF}}$	ACTIVE LOW INPUT		
AC Failure Signal– may be connected directly to ACFAIL*			

## 3.2 Local Signals(Continued)

$\overline{L7MEM}$	ACTIVE LOW INPUT
Local Level 7 Interrupt – this interrupt is intended to come from local memory fault detection circuitry. The interrupt is level sensitive, latched and maskable. The VMEbus Requester circuit requests that the SCV64 release the VMEbus while keeping $\overline{LBGR1}$ negated and not initiate any new VMEbus requests while $\overline{L7MEM}$ is asserted. However, the SCV64 or other device can remain the local bus owner as long as $\overline{LBRQ1}$ remains asserted.	
$\overline{L7INMI}$	ACTIVE LOW INPUT
Local Level 7 Interrupt – this interrupt is not maskable by the SCV64. The interrupt is edge sensitive and latched and can be cleared by the CPU.	
$\overline{LBGR1}$	ACTIVE LOW OUTPUT
Local Bus Grant – this signal is asserted to a local requester when the local bus is available.	
$\overline{LBRQ1}$	ACTIVE LOW INPUT
Local Bus Request – this request is generated by a local requester to the SCV64. $\overline{LBGR1}$ is asserted when the bus is available.	
$\overline{LIACK5-4}$	ACTIVE LOW OUTPUTS
Local Interrupt Acknowledgment – this signal is sent to the device connected to either the $\overline{LIRQ5}$ or $\overline{LIRQ4}$ line.	
$\overline{LIRQ5-0}$	ACTIVE LOW INPUTS
Local Interrupts – each interrupt may be mapped to any of the seven CPU interrupt levels. All six interrupts are level sensitive, not latched, individually maskable and can be read.	
$\overline{LIRQ2} / \overline{KIACK}$	ACTIVE LOW INPUTS
External Local IACK Decoding – this pin must be in $\overline{KIACK}$ mode Internal Local IACK Decoding – this pin is available for $\overline{LIRQ2}$	
$\overline{LMINT}$	ACTIVE LOW OUTPUT
Location Monitor Interrupt – this interrupt indicates that there are entries in the Location Monitor FIFO. $\overline{LMINT}$ is negated when the FIFO becomes empty. This signal is usually connected to the SCV64 as one of the auto-vectored interrupt inputs.	
$\overline{LRST}$	ACTIVE LOW OUTPUT
Local Reset – ends any cycles being performed by the SCV64 and clears various internal registers.	
$\overline{PWRST}$	ACTIVE LOW CMOS THRESHOLD SCHMITT TRIGGER INPUT
Power-on-Reset – this reset is typically supplied by a resistor-capacitor network. All circuitry in the SCV64 is directly reset by this signal. During power reset, the SCV64 generates local and system reset.	

## 3.2 Local Signals<sup>(Continued)</sup>

$\overline{\text{RAMSEL}}$	ACTIVE LOW OUTPUT
Memory Select – this signal is asserted by the SCV64: <ul style="list-style-type: none"> <li>to select local memory during accesses from the VMEbus,</li> <li>when the SCV64 DMA is accessing memory, or</li> <li>when the local CPU is accessing its own VME slave image.</li> </ul>	
$\overline{\text{SCV64SEL}}$	ACTIVE LOW INPUT
SCV64 Chip Select – used by a local address decoder for SCV64 register accesses.	
$\overline{\text{SYSFLED}}$	ACTIVE LOW OUTPUT
SYSFAIL LED Driver – this signal can be used to directly control an onboard LED, typically a faceplate indicator of the board's status. Asserted when SYSFAIL* is asserted.	
$\overline{\text{TICK}}$	ACTIVE LOW OUTPUT
Tick Timer – the tick timer period is determined by the TLEN0 and TLEN1 bits in the GENCTL register, and can be set to 5, 10, 50 or 100 ms in Normal mode, or 200, 400 $\mu\text{s}$ and 2 or 4 ms in Fast mode. Fast mode is determined using the TICKM bit in the CTL2 register (see “Tick Timer” on page 2-115).	
$\overline{\text{TMODE1-0}}$	ACTIVE HIGH INPUTS
Input signals used to test the SCV64 during chip fabrication. <div style="display: flex; align-items: center; margin-top: 20px;">  <div> <p><i>Caution: These inputs must be connected to ground for proper SCV64 operation.</i></p> </div> </div>	
$\overline{\text{VMEINT}}$	ACTIVE LOW OUTPUT
VMEbus Activity Interrupt – this interrupt indicates an event related to VMEbus use has occurred (such as DMA finished or a bus error received). The signal is negated when appropriate flags in the SCV64 status register are cleared. $\overline{\text{VMEINT}}$ can also be used as an interrupt to the local CPU and is usually connected via one of the auto-vectored interrupt inputs.	

## 3.2 Local Signals (Continued)

$\overline{\text{VMEOUT}}$	ACTIVE LOW INPUT
VMEbus Select– this signal is generated by the local address decoder to indicate that the SCV64 should attempt to initiate a VMEbus cycle. The address map within the SCV64 may indicate that the access is to the board's VME slave image or a VSB bus cycle and assert $\overline{\text{RAMSEL}}$ or $\overline{\text{VSBSEL}}$ respectively. Otherwise, a VMEbus cycle is initiated.	
$\overline{\text{VSBSEL}}$	ACTIVE LOW OUTPUT
VSB Select – this signal is asserted by the SCV64 when a local CPU access to the VMEbus has been mapped onto the VSB (by SCV64 internal registers). The SCV64 does not perform a VMEbus cycle, but expects a VSB interface to complete the transfer.	
$\overline{\text{WDOG}}$	ACTIVE LOW OUTPUT
Watchdog Timer – watchdog runs for 2 sec, asserts $\overline{\text{WDOG}}$ low for 200 ms, then begins the 2 sec time-out again. Only power-up reset or clearing the watchdog (by writing to the CLRDOG bit in the STAT0 register) can reset the timer. Clearing the ENDOG bit in the MISC register disables the watchdog timer.	

## 4 Signals and DC Characteristics

### 4.1 Terminology

The input and output types have all been abbreviated with a letter code. For example, the VDATA31-00 signals are shown as input type CTTL (which are CMOS inputs with normal TTL voltage thresholds) and output type TS (which are tri-stateable outputs). The following is a list of abbreviations:

CMOS	CMOS input with CMOS thresholds
CMOS SCH	Schmitt trigger input with CMOS thresholds
CTTL SCH	Schmitt trigger input with TTL thresholds
CTTL	CMOS input with TTL thresholds
I	Input
I/O	Input/Output
O	Output
OD	Open drain output
TP	Totem pole output
TS	Tri-state totem pole output
VOD	VMEbus specification open drain output
VTs	VMEbus specification tri-state totem pole output

## 4.2 DC Characteristics

**Table 4.1 : DC Electrical Characteristics**

Symbol	Parameter	Signal Type	Test Conditions	Tested at -40°C, 25°C, 85°C $V_{DD} = 5V \pm 5\%$		Tested at -55°C, 125°C $V_{DD} = 5V \pm 10\%$	
				Min	Max	Min	Max
$V_{IH}$	Min. high-level input	CTTL	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	2.0 V	-	2.0 V	-
		CMOS	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	$0.7V_{DD}$	-	$0.7V_{DD}$	-
$V_{IL}$	Max. low-level input	CTTL	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	-	0.8V	-	0.8V
		CMOS	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	-	$0.3V_{DD}$	-	$0.3V_{DD}$
$V_{T+}$	Positive going Schmitt trigger voltage	CTTL/SC H	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	1.2 V	2.4 V	1.2 V	2.4 V
		CMOS/SC H	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	$0.42V_{DD}$	$0.94V_{DD}$	$0.4V_{DD}$	$1.03V_{DD}$
$V_{T-}$	Negative going Schmitt trigger voltage	CTTL/SC H	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	0.8 V	2.0 V	0.8 V	2.0 V
		CMOS/SC H	$V_{OUT} = 0.1V$ or $V_{DD} - 0.1V$ ; $[I_{OUT}] = 20 \mu A$	$0.31V_{DD}$	$0.69V_{DD}$	$0.29 V_{DD}$	$0.76 V_{DD}$
$V_{Hysteresis}$	Schmitt trigger hysteresis voltage	CTTL/SC H	$V_{T+}$ to $V_{T-}$	$0.06V_{DD}$	$0.12V_{DD}$	$0.05 V_{DD}$	$0.14 V_{DD}$
		CMOS/SC H	$V_{T+}$ to $V_{T-}$	$0.11V_{DD}$	$0.25V_{DD}$	$0.11 V_{DD}$	$0.27 V_{DD}$
$I_{IN}$	Maximum input leakage current	CMOS and CTTL	With no pull-up resistor ( $V_{IN} = V_{SS}$ or $V_{DD}$ )	-5.0 $\mu A$	5.0 $\mu A$	-5.0 $\mu A$	5.0 $\mu A$
$I_{OZ}$	Maximum output leakage current	TS	$(V_{OUT} = V_{SS} \text{ or } V_{DD})$	-10.0 $\mu A$	10.0 $\mu A$	-10.0 $\mu A$	10.0 $\mu A$
		OD	$(V_{OUT} = V_{DD})$	-10.0 $\mu A$	10.0 $\mu A$	-10.0 $\mu A$	10.0 $\mu A$

**Table 4.2 : Pin List and DC Characteristics for SCV64 Signals  
(-55°C to 125°C)**

Signal Name	Type	Pin Number		In Type	Out Type	$I_{OL}$ (mA) $V_{OL}=0.4$ , * 0.6 V Min	$I_{OH}$ (mA) $V_{OH}=3.5V$ Min	Signal Description
		CPGA	PQFP					
BAUDCLK	O	E4	88	-	TP	10	-10	Baud clock
BBSY*	I/O	C20	295	CTTL/ SCH	V0D	48*	-	VMEbus BBSY* signal
BCLR*	I/O	N17	252	CTTL/ SCH	VTS	64*	-50	VMEbus BCLR* signal
BERR*	I/O	E19	291	CTTL/ SCH	VOD	48*	-	VMEbus error
BG0IN*	I	P19	251	CTTL	-	-	-	VMEbus bus grant in
BG1IN*	I	N19	253	CTTL	-	-	-	VMEbus bus grant in
BG2IN*	I	L19	262	CTTL	-	-	-	VMEbus bus grant in
BG3IN*	I	L17	264	CTTL	-	-	-	VMEbus bus grant in
BG0OUT*	O	U9	182	-	TP	10	-10	VMEbus bus grant out
BG1OUT*	O	T9	183	-	TP	10	-10	VMEbus bus grant out
BG2OUT*	O	V9	184	-	TP	10	-10	VMEbus bus grant out
BG3OUT*	O	W10	186	-	TP	10	-10	VMEbus bus grant out
BIMODE	O	D2	87	-	TP	10	-10	Bi-Mode output
$\overline{BIREL}$	I	A13	25	CTTL/ SCH	-	-	-	Bi-Mode release
$\overline{BITRIG}$	I	D11	36	CTTL/ SCH	-	-	-	Bi-Mode trigger
BR0*	I/O	D13	24	CTTL/ SCH	VOD	48*	-	VMEbus bus request
BR1*	I/O	B12	35	CTTL/ SCH	VOD	48*	-	VMEbus bus request
BR2*	I/O	D8	54	CTTL/ SCH	VOD	48*	-	VMEbus bus request
BR3*	I/O	B17	11	CTTL/ SCH	VOD	48*	-	VMEbus bus request
C14US	O	D7	58	-	TP	10	-10	Clock out - 14us period
C32MHZ	I	L3	114	CMOS	-	-	-	32MHz clock input
C8MHZ	O	L4	116	-	TP	10	-10	Clock output - 8MHz
DTACK*	I/O	N20	255	CTTL/ SCH	VOD	48*	-	VMEbus DTACK* signal
$\overline{EXTRST}$	I	D6	64	CTTL/ SCH	-	-	-	External reset
IACK*	I/O	U19	239	CTTL	VTS	48*	-40	VMEbus IACK* signal
IACKI*	I	N18	254	CTTL	-	-	-	VMEbus IACKIN* signal
IACKO*	O	W19	233	-	TP	10	-10	VMEbus IACKOUT* signal

**Table 4.2 : Pin List and DC Characteristics for SCV64 Signals  
(-55°C to 125°C) (Continued)**

Signal Name	Type	Pin Number		In Type	Out Type	$I_{OL}$ (mA) $V_{OL}=0.4$ , * 0.6 V Min	$I_{OH}$ (mA) $V_{OH}=3.5V$ Min	Signal Description
		CPGA	PQFP					
IRQ1*	I/O	W4	163	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ2*	I/O	W6	167	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ3*	I/O	U8	176	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ4*	I/O	W11	195	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ5*	I/O	V14	206	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ6*	I/O	W15	215	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
IRQ7*	I/O	Y18	219	CTTL/ SCH	VOD	48*	-	VMEbus interrupt request
JTCLK	I	J1	119	CTTL	-	-	-	JTAG test clock
JTDI	I	K4	110	CTTL	-	-	-	JTAG test data
JTDO	O	J5	111	-	TS	10	-10	JTAG test data
JTMS	I	M3	120	CTTL	-	-	-	JTAG test mode select
KADDR (31:0)	I/O	See Table 4.4	See Table 4.4	CTTL	TS	5	-5	CPU address bus
$\overline{KAS}$	I/O	K2	112	CTTL	TS	10	-10	CPU address strobe
$\overline{KAVEC}$	O	C5	70	-	TP	5	-5	AVEC interrupt termination
$\overline{KBERR}$	I/O	A3	69	CTTL	TS	10	-10	CPU bus error
$\overline{KBGACK}$	I/O	B4	67	CTTL/ SCH	OD	10	-	CPU bus grant acknowledge
$\overline{KBGR}$	I	B8	55	CTTL	-	-	-	CPU bus grant
$\overline{KBRQ}$	O	B9	53	-	TP	5	-5	CPU bus request
KCLK	I	K3	108	CMOS	-	-	-	CPU clock
KDATA (31:0)	I/O	See Table 4.4	See Table 4.4	CTTL	TS	5	-5	CPU data bus
$\overline{KDS}$	I/O	C8	52	CTTL	TS	5	-5	CPU data strobe
$\overline{KDSACK0}$	I/O	E10	47	CTTL	TS	10	-10	CPU data transfer and size acknowledge
$\overline{KDSACK1}$	I/O	D9	48	CTTL	TS	10	-10	CPU data transfer and size acknowledge
KFC0	I/O	C10	40	CTTL	TS	5	-5	CPU function code
KFC1	I/O	A11	41	CTTL	TS	5	-5	CPU function code
KFC2	I/O	D10	42	CTTL	TS	5	-5	CPU function code
$\overline{KHALT}$	I/O	A10	43	CTTL	TS	10	-10	CPU halt



**Table 4.2 : Pin List and DC Characteristics for SCV64 Signals  
(-55°C to 125°C) (Continued)**

Signal Name	Type	Pin Number		In Type	Out Type	$I_{OL}$ (mA) $V_{OL}=0.4$ , * 0.6 V Min	$I_{OH}$ (mA) $V_{OH}=3.5V$ Min	Signal Description
		CPGA	PQFP					
$\overline{LIRQ2} / \overline{KIACK}$	I	W8	175	CTTL	-	-	-	Local interrupt acknowledge
$\overline{KIPL0}$	O	R2	139	-	TP	5	-5	CPU interrupt priority level
$\overline{KIPL1}$	O	R4	140	-	TP	5	-5	CPU interrupt priority level
$\overline{KIPL2}$	O	T2	141	-	TP	5	-5	CPU interrupt priority level
$\overline{KRMC}$	I/O	B10	44	CTTL	TS	2	-2	CPU RMC
KSIZE0	I/O	A12	37	CTTL	TS	5	-5	CPU transfer size code
KSIZE1	I/O	C11	38	CTTL	TS	5	-5	CPU transfer size code
$\overline{KWR}$	I/O	C9	46	CTTL	TS	10	-10	CPU write
$\overline{L7IACF}$	I	F17	292	CTTL	-	-	-	Level 7 interrupt (ACFAIL)
$\overline{L7IMEM}$	I	E18	294	CTTL	-	-	-	Level 7 interrupt (Memory error)
$\overline{L7INMI}$	I	C16	12	CTTL	-	-	-	Level 7 interrupt (non-maskable)
$\overline{LBGR1}$	O	B11	34	-	TP	5	-5	Local bus grant (external DMA)
$\overline{LBRQ1}$	I	V16	218	CTTL	-	-	-	Local bus request (external DMA)
$\overline{LIACK4}$	O	V7	174	-	TP	5	-5	Local interrupt acknowledge 4
$\overline{LIACK5}$	O	V8	178	-	TP	5	-5	Local interrupt acknowledge 5
$\overline{LIRQ0}$	I	V5	164	CTTL	-	-	-	Local interrupt
$\overline{LIRQ1}$	I	U6	166	CTTL	-	-	-	Local interrupt
$\overline{LIRQ2} / \overline{KIACK}$	I	W8	175	CTTL	-	-	-	Local interrupt
$\overline{LIRQ3}$	I	U11	194	CTTL	-	-	-	Local interrupt
$\overline{LIRQ4}$	I	W12	205	CTTL	-	-	-	Local interrupt
$\overline{LIRQ5}$	I	U15	216	CTTL	-	-	-	Local interrupt
$\overline{LMINT}$	O	Y8	177	-	TP	2	-2	Location monitor interrupt
$\overline{LRST}$	O	U7	172	-	TP	20	-20	Local reset
$\overline{PWRRST}$	I	R17	240	CTTL/ SCH	-	-	-	Power-up reset
$\overline{RAMSEL}$	O	C13	26	-	TP	5	-5	Local memory select
RETRY*/ $\overline{VRMC}$	I/O	B6	63	CTTL/ SCH	TS	2	-2	VMEbus RETRY* (I), or VMEbus RMC signal (I/O)
$\overline{SCV64SEL}$	I	G19	283	CTTL	-	-	-	SCV64 chip select
SYSCLK	I/O	K16	263	CTTL	VTS	64*	-50	VMEbus SYSCLK

**Table 4.2 : Pin List and DC Characteristics for SCV64 Signals  
(-55°C to 125°C) (Continued)**

Signal Name	Type	Pin Number		In Type	Out Type	$I_{OL}$ (mA) $V_{OL}=0.4$ , * 0.6 V Min	$I_{OH}$ (mA) $V_{OH}=3.5V$ Min	Signal Description
		CPGA	PQFP					
SYSFAIL*	I/O	R18	242	CTTL/ SCH	VOD	48*	-	VMEbus SYSFAIL
$\overline{\text{SYSFLED}}$	O	C12	32	-	OD	24*	-	SYSFAIL LED driver
SYSRST*	I/O	H17	282	CTTL/ SCH	VOD	48*	-	VMEbus SYSRESET
$\overline{\text{TICK}}$	O	D12	30	-	TP	5	-5	Tick clock
TMODE0	I	J2	117	CMOS /SCH	-	-	-	Test mode enable connect to ground
TMODE1	I	L5	118	CMOS /SCH	-	-	-	Test mode enable connect to ground
VADDR (31:1)	I/O	See Table 4.3	See Table 4.3	CTTL	TS	2	-2	VMEbus address bus
VADDRROUT	O	C14	22	-	TP	10	-10	VMEbus address direction control
VAM0	I/O	M17	258	CTTL	TS	2	-2	VMEbus address modifier 0
VAM1	I/O	M19	259	CTTL	TS	2	-2	VMEbus address modifier 1
VAM2	I/O	M18	260	CTTL	TS	2	-2	VMEbus address modifier 2
VAM3	I/O	L18	266	CTTL	TS	2	-2	VMEbus address modifier 3
VAM4	I/O	K18	268	CTTL	TS	2	-2	VMEbus address modifier 4
VAM5	I/O	L20	269	CTTL	TS	2	-2	VMEbus address modifier 5
$\overline{\text{VAs}}$	I/O	U18	235	CTTL/ SCH	TS	5	-5	VMEbus address strobe
VDATA (31:0)	I/O	See Table 4.3	See Table 4.3	CTTL	TS	2	-2	VMEbus data bus
VDATAOUT	O	D14	20	-	TP	10	-10	VMEbus data direction
$\overline{\text{VDS0}}$	I/O	V18	231	CTTL	TS	5	-5	VMEbus data strobe
$\overline{\text{VDS1}}$	I/O	U17	232	CTTL	TS	5	-5	VMEbus data strobe
$\overline{\text{VLWORD}}$	I/O	W9	187	CTTL	TS	2	-2	VMEbus LWORD* signal
$\overline{\text{VMEINT}}$	O	C15	18	-	TP	2	-2	VMEbus activity interrupt
$\overline{\text{VMEOUT}}$	I	V20	241	CTTL	-	-	-	VMEbus select
RETRY*/ $\overline{\text{VRMC}}$	I/O	B6	63	CTTL/ SCH	TS	2	-2	VMEbus RETRY* (I), or VMEbus RMC signal (I/O)
$\overline{\text{VSBSEL}}$	O	B13	23	-	TP	5	-5	VSBbus select
VSTRBOUT	O	B14	21	-	TP	5	-5	VMEbus strobe direction
$\overline{\text{VWR}}$	I/O	T17	234	CTTL	TS	2	-2	VMEbus WRITE* signal
$\overline{\text{WDOG}}$	O	E12	31	-	TP	5	-5	Watchdog interrupt output

**Table 4.3 : VMEbus Address and Data Input and Output Signal Bits**

Signal	CPGA Pin	PQFP Pin	Signal	CPGA Pin	PQFP Pin
VADDR1	U10	188	VDATA0	K17	270
VADDR2	Y10	189	VDATA1	K19	271
VADDR3	V10	190	VDATA2	J18	272
VADDR4	V11	192	VDATA3	J17	274
VADDR5	Y11	193	VDATA4	K20	275
VADDR6	V12	196	VDATA5	J16	276
VADDR7	U12	198	VDATA6	J19	279
VADDR8	T11	199	VDATA7	H18	280
VADDR9	V13	200	VDATA8	H19	281
VADDR10	U13	204	VDATA9	G18	284
VADDR11	W13	207	VDATA10	F20	285
VADDR12	T13	208	VDATA11	G17	286
VADDR13	W14	209	VDATA12	F18	288
VADDR14	U14	210	VDATA13	F19	289
VADDR15	V15	212	VDATA14	D19	293
VADDR16	Y15	213	VDATA15	C19	297
VADDR17	W16	217	VDATA16	F16	298
VADDR18	W17	221	VDATA17	D18	299
VADDR19	T15	222	VDATA18	E17	300
VADDR20	W18	223	VDATA19	C18	301
VADDR21	U16	224	VDATA20	E16	302
VADDR22	V17	225	VDATA21	B19	3
VADDR23	T16	226	VDATA22	D17	4
VADDR24	T18	236	VDATA23	B18	5
VADDR25	V19	237	VDATA24	E15	6
VADDR26	T19	243	VDATA25	C17	7
VADDR27	R19	245	VDATA26	D16	8
VADDR28	P17	246	VDATA27	A18	9
VADDR29	N16	248	VDATA28	B16	13
VADDR30	R20	249	VDATA29	D15	14
VADDR31	P18	250	VDATA30	B15	15
			VDATA31	A15	17

**Table 4.4 : Local Bus Address and Data Input and Output Signal Bits**

Signal	CPGA Pin	PQFP Pin	Signal	CPGA Pin	PQFP Pin
KADDR0	C7	56	KDATA0	M4	122
KADDR1	B7	57	KDATA1	L2	123
KADDR2	C6	60	KDATA2	N3	124
KADDR3	A6	61	KDATA3	L1	127
KADDR4	B5	65	KDATA4	N4	128
KADDR5	E7	66	KDATA5	M2	129
KADDR6	C4	71	KDATA6	P3	130
KADDR7	D5	72	KDATA7	N2	131
KADDR8	B3	73	KDATA8	N5	132
KADDR9	C3	74	KDATA9	P2	133
KADDR10	B2	79	KDATA10	P4	134
KADDR11	E5	80	KDATA11	R3	136
KADDR12	C2	81	KDATA12	R1	137
KADDR13	D4	82	KDATA13	T3	142
KADDR14	D3	83	KDATA14	U2	143
KADDR15	F5	84	KDATA15	V1	145
KADDR16	C1	85	KDATA16	T4	146
KADDR17	E3	89	KDATA17	V2	147
KADDR18	F4	90	KDATA18	U3	148
KADDR19	E2	91	KDATA19	W2	149
KADDR20	F3	93	KDATA20	T5	150
KADDR21	G4	94	KDATA21	V3	155
KADDR22	G3	96	KDATA22	U4	156
KADDR23	F2	97	KDATA23	W3	157
KADDR24	H4	98	KDATA24	T6	158
KADDR25	F1	99	KDATA25	V4	159
KADDR26	H3	100	KDATA26	U5	160
KADDR27	G2	101	KDATA27	Y3	161
KADDR28	J4	102	KDATA28	W5	165
KADDR29	H2	103	KDATA29	Y6	169
KADDR30	J3	106	KDATA30	V6	170
KADDR31	H1	107	KDATA 31	W7	173

**Table 4.5 : Pin Assignments for Ground**

<b>V<sub>SS</sub> Pins</b>					
<b>CPGA</b>			<b>PQFP</b>		
A2	A4	A7	76	68	59
A9	A14	A17	50	45	39
A19	B1	B20	33	28	19
D1	D20	E8	10	1	304
E11	E14	G1	296	287	278
G5	G16	G20	273	267	261
J20	K1	K5	256	247	238
L16	M1	P1	229	228	220
P5	P16	P20	211	202	197
T7	T10	T14	191	185	180
U1	U20	W1	171	162	153
W20	Y2	Y4	152	144	135
Y7	Y12	Y14	126	121	115
Y17	Y19		109	104	95
			86	77	

**Table 4.6 : Pin Assignments for Power**

<b>V<sub>DD</sub> Pins</b>					
<b>CPGA</b>			<b>PQFP</b>		
A1	A5	A8	75	62	51
A16	A20	E1	49	29	27
E9	E13	E20	16	2	303
H5	H16	H20	290	277	265
M5	M16	M20	257	244	230
N1	R5	R16	227	214	203
T1	T8	T12	201	181	179
T20	Y1	Y5	168	154	151
Y9	Y13	Y16	138	125	113
Y20			105	92	78

## 4.3 Capacitive Loading

**Table 4.7 : Input Capacitive Loading**

Signal Type	Signal Name	Input Capacitance (pF)
Input	BGIN[3:0], BIREL, BITRIG, C32MHZ, EXTRST, JTCLK, JTDI, JTMS, KBGR, KCLK, L7IACF, L7IMEM, L7INMI, LBRQ, LIRQ[5:0], PWRRST, TMODE[1:0], IACKIN, VMEOUT	18
Input/Output	IACK, KADDR[31:0], KAS, KBERR, KDATA[31:0], KDS, KDSACK[1:0], KFC[2:0], KHALT, KSIZE[1:0], KWR, KRMC, SCV64SEL, VADDR[31:1], VAM[5:0], VAS, VDATA[31:0], VDS[1:0], VLWORD, VRETRY, VWR	18
	BBSY, BR[3:0], BERR, DTACK, IRQ[7:1], SYSFAIL, SYSRST	24
	BCLR, SYSCLK	30
Output	JTDO	18

## 4.4 Pin Configuration

The 299-pin CPGA configuration is shown here in Figure 4.1 while the 304 pin PQFP package is illustrated in Figure 4.2.

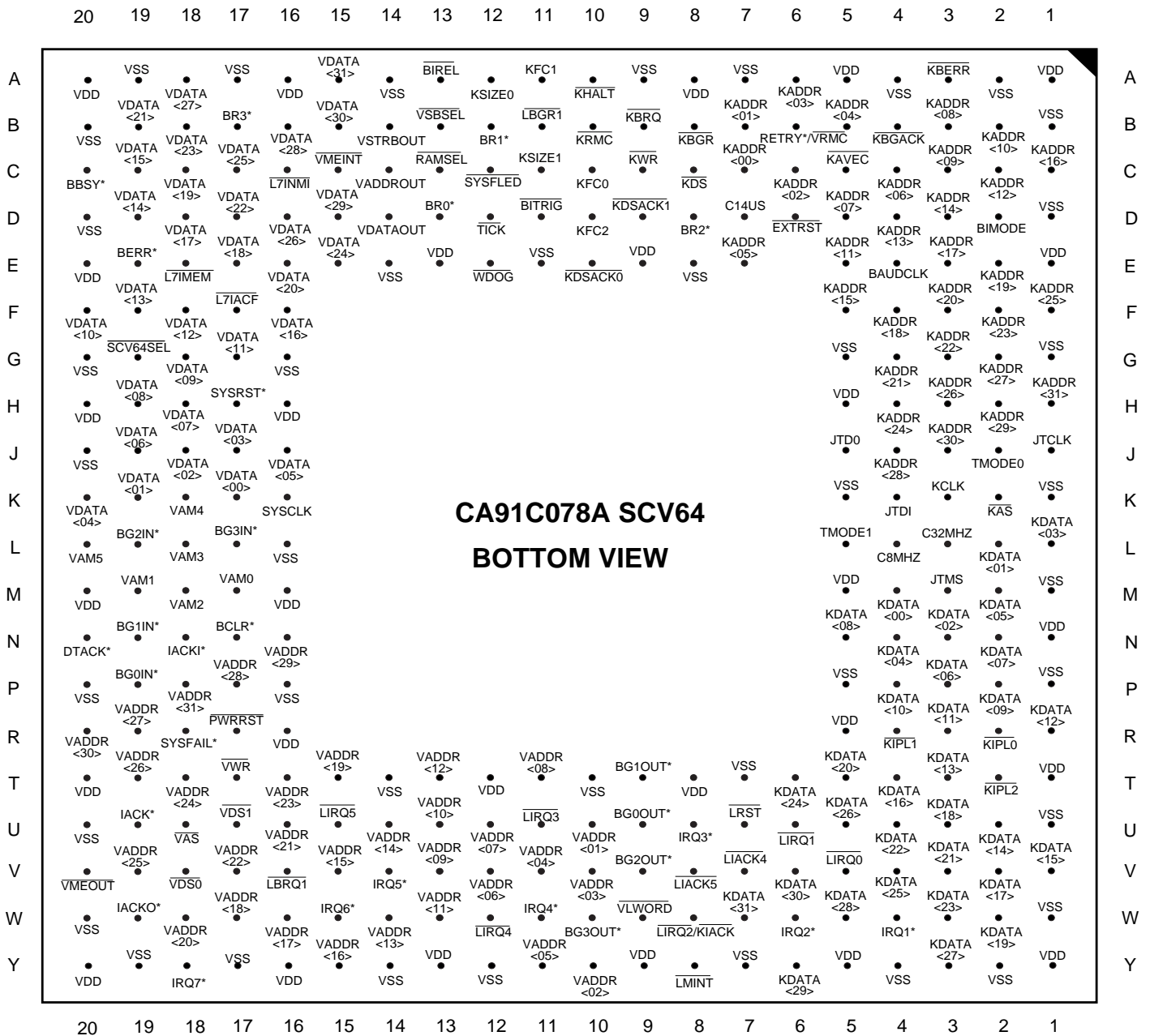


Figure 4.1 : Pin Configuration for 299-Pin CPGA Package

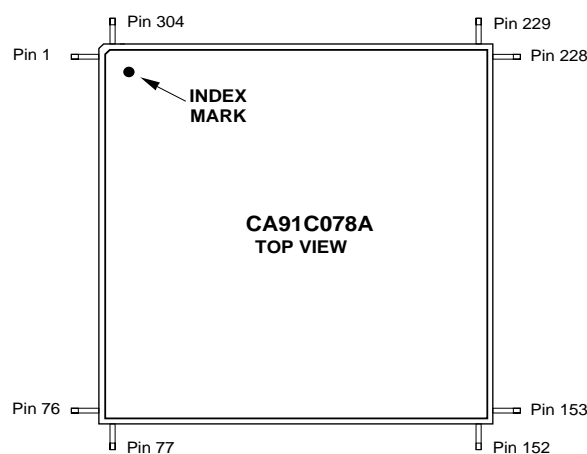


Figure 4.2 : Pin Configuration for 304-Pin PQFP Package

## 304 PIN PLASTIC QFP

1. V <sub>SS</sub>	39. V <sub>SS</sub>	77. V <sub>SS</sub>	115. V <sub>SS</sub>	153. V <sub>SS</sub>	191. V <sub>SS</sub>	229. V <sub>SS</sub>	267. V <sub>SS</sub>
2. V <sub>DD</sub>	40. KFC0	78. V <sub>DD</sub>	116. C8MHZ	154. V <sub>DD</sub>	192. VADDR4	230. V <sub>DD</sub>	268. VAM4
3. VDATA21	41. KFC1	79. KADDR10	117. TMODE0	155. KDATA21	193. VADDR5	231. $\overline{\text{VDS0}}$	269. VAM5
4. VDATA22	42. KFC2	80. KADDR11	118. TMODE1	156. KDATA22	194. $\overline{\text{LIRQ3}}$	232. $\overline{\text{VDS1}}$	270. VDATA0
5. VDATA23	43. $\overline{\text{KHALT}}$	81. KADDR12	119. JTCLK	157. KDATA23	195. IRQ4*	233. IACKO*	271. VDATA1
6. VDATA24	44. $\overline{\text{KRM}\overline{\text{C}}}$	82. KADDR13	120. JTMS	158. KDATA24	196. VADDR6	234. $\overline{\text{VWR}}$	272. VDATA2
7. VDATA25	45. V <sub>SS</sub>	83. KADDR14	121. V <sub>SS</sub>	159. KDATA25	197. V <sub>SS</sub>	235. $\overline{\text{V}\overline{\text{A}}\overline{\text{S}}}$	273. V <sub>SS</sub>
8. VDATA26	46. $\overline{\text{KWR}}$	84. KADDR15	122. KDATA0	160. KDATA26	198. VADDR7	236. VADDR24	274. VDATA3
9. VDATA27	47. $\overline{\text{KDSACK0}}$	85. KADDR16	123. KDATA1	161. KDATA27	199. VADDR8	237. VADDR25	275. VDATA4
10. V <sub>SS</sub>	48. $\overline{\text{KDSACK1}}$	86. V <sub>SS</sub>	124. KDATA2	162. V <sub>SS</sub>	200. VADDR9	238. V <sub>SS</sub>	276. VDATA5
11. BR3*	49. V <sub>DD</sub>	87. BIMODE	125. V <sub>DD</sub>	163. IRQ1*	201. V <sub>DD</sub>	239. IACK*	277. V <sub>DD</sub>
12. $\overline{\text{L7INMI}}$	50. V <sub>SS</sub>	88. BAUDCLK	126. V <sub>SS</sub>	164. $\overline{\text{LIRQ0}}$	202. V <sub>SS</sub>	240. $\overline{\text{PWRRST}}$	278. V <sub>SS</sub>
13. VDATA28	51. V <sub>DD</sub>	89. KADDR17	127. KDATA3	165. KDATA28	203. V <sub>DD</sub>	241. $\overline{\text{VMEOUT}}$	279. VDATA6
14. VDATA29	52. $\overline{\text{KDS}}$	90. KADDR18	128. KDATA4	166. $\overline{\text{LIRQ1}}$	204. VADDR10	242. SYSFAIL*	280. VDATA7
15. VDATA30	53. $\overline{\text{KBRQ}}$	91. KADDR19	129. KDATA5	167. IRQ2*	205. $\overline{\text{LIRQ4}}$	243. VADDR26	281. VDATA8
16. V <sub>DD</sub>	54. BR2*	92. V <sub>DD</sub>	130. KDATA6	168. V <sub>DD</sub>	206. IRQ5*	244. V <sub>DD</sub>	282. SYSRST*
17. VDATA31	55. $\overline{\text{KBGR}}$	93. KADDR20	131. KDATA7	169. KDATA29	207. VADDR11	245. VADDR27	283. $\overline{\text{SCV64SEL}}$
18. $\overline{\text{VMEINT}}$	56. KADDR0	94. KADDR21	132. KDATA8	170. KDATA30	208. VADDR12	246. VADDR28	284. VDATA9
19. V <sub>SS</sub>	57. KADDR1	95. V <sub>SS</sub>	133. KDATA9	171. V <sub>SS</sub>	209. VADDR13	247. V <sub>SS</sub>	285. VDATA10
20. VDATAOUT	58. C14US	96. KADDR22	134. KDATA10	172. $\overline{\text{LRST}}$	210. VADDR14	248. VADDR29	286. VDATA11
21. VSTRBOUT	59. V <sub>SS</sub>	97. KADDR23	135. V <sub>SS</sub>	173. KDATA31	211. V <sub>SS</sub>	249. VADDR30	287. V <sub>SS</sub>
22. VADDRROUT	60. KADDR2	98. KADDR24	136. KDATA11	174. $\overline{\text{LIACK4}}$	212. VADDR15	250. VADDR31	288. VDATA12
23. $\overline{\text{VSBSEL}}$	61. KADDR3	99. KADDR25	137. KDATA12	175. $\overline{\text{LIRQ2}} / \overline{\text{KIACK}}$	213. VADDR16	251. BG0IN*	289. VDATA13
24. BR0*	62. V <sub>DD</sub>	100. KADDR26	138. V <sub>DD</sub>	176. IRQ3*	214. V <sub>DD</sub>	252. BCLR*	290. V <sub>DD</sub>
25. BIREL	63. $\overline{\text{RETRY}} / \overline{\text{VRMC}}$	101. KADDR27	139. $\overline{\text{KIPL0}}$	177. $\overline{\text{LMINT}}$	215. IRQ6*	253. BG1IN*	291. BERR*
26. RAMSEL	64. $\overline{\text{EXTRST}}$	102. KADDR28	140. $\overline{\text{KIPL1}}$	178. $\overline{\text{LIACK5}}$	216. $\overline{\text{LIRQ5}}$	254. IACKI*	292. $\overline{\text{L7IACF}}$
27. V <sub>DD</sub>	65. KADDR4	103. KADDR29	141. $\overline{\text{KIPL2}}$	179. V <sub>DD</sub>	217. VADDR17	255. DTACK*	293. VDATA14
28. V <sub>SS</sub>	66. KADDR5	104. V <sub>SS</sub>	142. KDATA13	180. V <sub>SS</sub>	218. $\overline{\text{LBRQ1}}$	256. V <sub>SS</sub>	294. $\overline{\text{L7IMEM}}$
29. V <sub>DD</sub>	67. $\overline{\text{KBGACK}}$	105. V <sub>DD</sub>	143. KDATA14	181. V <sub>DD</sub>	219. IRQ7*	257. V <sub>DD</sub>	295. BBSY*
30. $\overline{\text{TICK}}$	68. V <sub>SS</sub>	106. KADDR30	144. V <sub>SS</sub>	182. BG0OUT*	220. V <sub>SS</sub>	258. VAM0	296. V <sub>SS</sub>
31. $\overline{\text{WDOG}}$	69. $\overline{\text{KBERR}}$	107. KADDR31	145. KDATA15	183. BG1OUT*	221. VADDR18	259. VAM1	297. VDATA15
32. $\overline{\text{SYSFLED}}$	70. KAVEC	108. KCLK	146. KDATA16	184. BG2OUT*	222. VADDR19	260. VAM2	298. VDATA16
33. V <sub>SS</sub>	71. KADDR6	109. V <sub>SS</sub>	147. KDATA17	185. V <sub>SS</sub>	223. VADDR20	261. V <sub>SS</sub>	299. VDATA17
34. $\overline{\text{LBGR1}}$	72. KADDR7	110. JTDI	148. KDATA18	186. BG3OUT*	224. VADDR21	262. BG2IN*	300. VDATA18
35. BR1*	73. KADDR8	111. JTDO	149. KDATA19	187. $\overline{\text{VLWORD}}$	225. VADDR22	263. SYSCCLK	301. VDATA19
36. $\overline{\text{BITRIG}}$	74. KADDR9	112. $\overline{\text{KAS}}$	150. KDATA20	188. VADDR1	226. VADDR23	264. BG3IN*	302. VDATA20
37. KSIZE0	75. V <sub>DD</sub>	113. V <sub>DD</sub>	151. V <sub>DD</sub>	189. VADDR2	227. V <sub>DD</sub>	265. V <sub>DD</sub>	303. V <sub>DD</sub>
38. KSIZE1	76. V <sub>SS</sub>	114. C32MHZ	152. V <sub>SS</sub>	190. VADDR3	228. V <sub>SS</sub>	266. VAM3	304. V <sub>SS</sub>



# Appendix A Registers

## A.1 Control and Status Registers

The SCV64 decodes local bus access to its addressable internal registers using the  $\overline{\text{SCV64SEL}}$  pin and the least significant 9 local address lines,  $\text{KADDR}[8:0]$ . Address range 000 to 04C will respond only to aligned 32-bit transfers, and will assert  $\overline{\text{KBERR}}$  otherwise.

Address range 080 to 0BF responds to any size transfer with any alignment but always transfers data using the least significant data lines  $\text{KDATA}[7:0]$ . Since the upper 24 data lines are ignored on writes to these registers and are always driven to zero, Table A.26 to Table A.41 will include only the 8 least significant bits. However, note that the SCV64 always responds as a 32-bit port, i.e. both  $\overline{\text{KDSACK0}}$  and  $\overline{\text{KDSACK1}}$  are asserted.

Any attempted access to a reserved register addresses will terminate with  $\overline{\text{KBERR}}$  (see Table A.1 and Table A.2).

**Table A.1 : A Summary of SCV64 Register Accesses**

Address Range (hexadecimal)	32-bit Access	16-bit Access	8-bit Access
000 to 04C	yes	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
050 to 07F	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
080 to 0BF	yes	yes	yes
0C0 to 0E0	yes	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$
0E4 to 1FF	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$	$\overline{\text{KBERR}}$

**Table A.2 : SCV64 Register Map**

Longword Register Address	Register	Name
xxxx x000	DMA Local Address	DMALAR
xxxx x004	DMA VMEbus Address	DMAVAR
xxxx x008	DMA Transfer Count	DMATC
xxxx x00C	Control and Status	DCSR
xxxx x010	VMEbus Slave Base Address	VMEBAR
xxxx x014	Rx FIFO Data	RXDATA
xxxx x018	Rx FIFO Address Register	RXADDR
xxxx x01C	Rx FIFO Control Register	RXCTL
xxxx x020	VMEbus/VSB Bus Select	BUSSEL
xxxx x024	VMEbus Interrupter Vector	IVECT
xxxx x028	Access Protect Boundary	APBR
xxxx x02C	Tx FIFO Data Output Latch	TXDATA
xxxx x030	Tx FIFO Address Output Latch	TXADDR
xxxx x034	Tx FIFO AM Code and Control Bit Latch	TXCTL
xxxx x038	Location Monitor FIFO Read Port	LMFIFO
xxxx x03C	SCV64 Mode Control	MODE
xxxx x040	Slave A64 Base Address	SA64BAR
xxxx x044	Master A64 Base Address	MA64BAR
xxxx x048	Local Address Generator	LAG
xxxx x04C	DMA VMEbus Transfer Count	DMAVTC
xxxx 050 to xxxx 07F	reserved	
xxxx x080	Status Register 0	STAT0
xxxx x084	Status Register 1	STAT1
xxxx x088	General Control Register	GENCTL
xxxx x08C	VMEbus Interrupter Requester	VINT
xxxx x090	VMEbus Requester Register	VREQ
xxxx x094	VMEbus Arbiter Register	VARB
xxxx x098	ID Register	ID
xxxx x09C	Control and Status Register	CTL2
xxxx x0A0	Level 7 Interrupt Status Register	7IS
xxxx x0A4	Local Interrupt Status Register	LIS
xxxx x0A8	Level 7 Interrupt Enable Register	7IE
xxxx x0AC	Local Interrupt Enable Register	LIE
xxxx x0B0	VMEbus Interrupt Enable Register	VIE
xxxx x0B4	Local Interrupts 1 and 0 Control Register	IC10
xxxx x0B8	Local Interrupts 3 and 2 Control Register	IC32

**Table A.2 : SCV64 Register Map (Continued)**

Longword Register Address	Register	Name
xxxx x0BC	Local Interrupts 5 and 4 Control Register	IC54
xxxx x0C0	Miscellaneous control register	MISC
xxxx x0C4	Delay line control register	DLCT
xxxx x0C8	Delay line status register 1	DLST1
xxxx x0CC	Delay line status register 2	DLST2
xxxx x0D0	Delay line status register 3	DLST3
xxxx x0D4	Mailbox register 0	MBOX0
xxxx x0D8	Mailbox register 1	MBOX1
xxxx x0DC	Mailbox register 2	MBOX2
xxxx x0E0	Mailbox register 3	MBOX3
xxxx x0E4 to xxxx x1FC	reserved	

**Table A.3 : DMA Local Address Register (DMALAR)**

Register Name: DMALAR		Register Number: xxxx x000	
Bits	Function		
31-24	Not Used (5 bits)		DLA (DMA Local Address)
23-16	DLA (DMA Local Address) - Byte 2		
15-08	DLA (DMA Local Address) - Byte 1		
07-00	DLA (DMA Local Address) - Byte 0 (Total 27 bits)		0

**DMALAR Description**

Name	Type	Condition after Reset	State	Function
DLA 26 – 01	R/W	U		DMA local address bits 26 – 01
DLA 00	R	0	0	DMA local address bit 00 is always 0
DLA 01	R/W	U	–	DMA local address bit 01 is always 0 for D32 BLT, or MBLT transfers
DLA 02	R/W	U	–	DMA local address 02 is always 0 for MBLT transfers

**Table A.4 : DMA VMEbus Address Register (DMAVAR)**

Register Name: DMAVAR		Register Address: xxxx x004	
Bits	Function		
31-24	DVA (DMA VME bus Address) - Byte 3		
23-16	DVA (DMA VME bus Address) - Byte 2		
15-08	DVA (DMA VME bus Address) - Byte 1		
07-00	DVA (DMA VME bus Address) - Byte 0		0

**DMAVAR Description**

Name	Type	Condition after Reset	State	Function
DVA 31 – 01	R/W	U		DMA VMEbus address bits 31 – 01
DVA 00	R	0	0	DMA VMEbus address bit 00 is always 0
DVA 01	R/W	U	–	DMA VMEbus address bit 01 is always 0 for D32 BLT, or MBLT transfers
DVA 02	R/W	U	–	DMA VMEbus address bit 02 is always 0 for MBLT transfers

**Table A.5 : DMA Transfer Count Register (DMATC)**

Register Name: DMATC		Register Address: xxxx x008	
Bits	Function		
31-24	Not Used		
23-16	Not used	DTC (DMA Transfer Count)	
15-08	DTC (DMA Transfer Count)	DTC (DMA Transfer Count)	
07-00	(DMA Transfer Count) (Total 20 bits)		

**DMATC Description**

Name	Type	Condition after Reset	State	Function	
DTC19 – 00 (Note 1)	R/W	U		DMA transfer count: specifies the number of local bus transfers to perform, summarized as follows:.	
				DMA Mode	Register Value
				Non-block, D16	Words
				Non-block, D32	Longwords
				BLT, D16	Words
				BLT, D32	Longwords
				MBLT	Longwords, (Note 2)

**Notes:**

1. DTC19-12 are only used when DTCSIZ (Bit 31 in Mode Control Register) is set.
2. Bit 00 is read-only value 0, as MBLT VMEbus transfers are double longwords.

Table A.6 : Control and Status Register (DCSR)

Register Name: DCSR					Register Address: xxxx x00C			
Bits	Function							
31-24	Not Used				reserved			
23-16	Not Used							CERR
15-08	TXSHIFT	RETRY	RMCERR	A64BARDY	reserved	RXSHIFT	RXRST	TXRST
07-00	RXHD	TXHD	DLBERR	LMHD	LBERR	VBERR	DONE	DMAGO

### DCSR Description

Name	Type	Condition after Reset	State	Function
CERR	R/W	0	0 1	Configuration error: asserts $\overline{\text{VMEINT}}$ while 1. Clear by writing 0 to this bit. See DMA Completion and Error Checking on page 2-108.  No conflicting configurations set Incompatible options are set
TXSHIFT	R/W	0	R W0 W1	Transmit FIFO shift Clears self; always reads zero No effect TX FIFO shifts one forward
RETRY	R/W	0	0 1	State of VMEbus RETRY* signal during last failed cycle Not asserted Asserted
RMCERR	R/W	0	R0 R1 W0 W1	RMC cycle lockup flag Last $\overline{\text{KBERR}}$ was <i>not</i> issued due to RMW lockup Last $\overline{\text{KBERR}}$ was issued due to RMW lockup Clear no effect
A64BARDY	R	0	0 1	A64 base address ready flag MA64BAR and SA64BAR registers not programmed yet Master and slave A64 base addresses are programmed
RXSHIFT	R/W	0	R W0 W1	Receive FIFO shift Clears self; always reads zero No effect RX FIFO shifts one forward

**DCSR Description (Continued)**

Name	Type	Condition after Reset	State	Function
RXRST	R/W	O	R W0 W1	Receive FIFO reset Clears self; always reads zero No effect Resets entire receive FIFO
TXRST	R/W	O	R W0 W1	Transmit FIFO reset Clears self; always reads zero No effect Resets entire transmit FIFO and any VME-out cycle in progress
RXHD	R	O	0 1	Receive FIFO status Receive FIFO is empty Receive FIFO has entries
TXHD	R	O	0 1	Transmit FIFO status Transmit FIFO is empty Transmit FIFO has entries
DLBERR	R/W	O	R0 R1 W0 W1	DMA Local Bus Error indicator; asserts $\overline{\text{VMEINT}}$ pin while 1 No error indicated DMA received a local bus error Clears DLBERR indicator No effect
LMHD	R	O	0 1	Location Monitor FIFO status; asserts $\overline{\text{LMINT}}$ pin while 1 LM FIFO is empty LM FIFO has entries
LBERR	R/W	O	R0 R1 W0 W1	Local $\overline{\text{KBERR}}$ received while in coupled or decoupled mode; asserts $\overline{\text{VMEINT}}$ pin while 1 No error indicated Local bus error received Clears LBERR indicator No effect.
VBERR	R/W	O	R0 R1 W0 W1	VMEbus BERR* received while in coupled or decoupled mode; causes the SCV64 to halt VME master cycles and assert the $\overline{\text{VMEINT}}$ pin while 1 No error indicated VMEbus BERR* received Clears BERR* indicator No effect.

**DCSR Description (Continued)**

Name	Type	Condition after Reset	State	Function
DONE	R/W	O		DMA Done indicator; asserts $\overline{\text{VMEINT}}$ pin while 1
			R0	DMA not done yet
			R1	DMA finished or stopped by CPU; not set if stopped due to BERR*
			W0	Clear DONE bit
DMAGO	R/W	O	W1	No effect
				DMA Go bit
			R0	DMA is stopped, by self or CPU
			R1	DMA is running
			W0	DMA stop request
			W1	Starts DMA



**Table A.7 : VMEbus Slave Base Address Register (VMEBAR)**

Register Name: VMEBAR			Register Address: xxxx x010		
Bits	Function				
31-24	Not Used				
23-16	Not Used	A24SIZ (2 bits)	A24BA (5 bits)		
15-08	Not Used				A32SIZ
07-00	A32SIZ (total 4 bits)		A32BA (5 bits)		

**VMEBAR Description**

Name	Type	Condition after Reset	State	Function
A24SIZ	R/W	U	0 1 2 3	Sets size of A24 slave image 512K 1M 2M 4M
A24BA	R/W	U	See Table A.8	Sets base address of A24 slave image. Programmed size of the image will force the low three order A24BA address bits (register bits 18, 17, and 16) to zero as appropriate. Bits 20 – 16 will be compared against VMEbus address bits A23 – A19.
A32SIZ	R/W	U	See Table A.9	Sets size of A32 slave image
A32BA	R/W	U	See Table A.10	Sets base address of A32 image



*Note: If VMEBAR is programmed from the VMEbus using a decoupled cycle, the VMEbus slave base address will not be updated immediately*

**Table A.8 : A24 Slave Image Programming**

A24SIZ State	A24 Slave Image Size	Acceptable A24 Slave Image Base Addresses	A24BA State
00	512 Kbytes	00 0000	00
		08 0000	01
		10 0000	02
		18 0000	03
		20 0000	04
		28 0000	05
		30 0000	06
		38 0000	07
		40 0000	08
		48 0000	09
		50 0000	0A
		58 0000	0B
		60 0000	0C
		68 0000	0D
		70 0000	0E
		78 0000	0F
		80 0000	10
		88 0000	11
		90 0000	12
		98 0000	13
		A0 0000	14
		A8 0000	15
		B0 0000	16
		B8 0000	17
		C0 0000	18
		C8 0000	19
		D0 0000	1A
		D8 0000	1B
		E0 0000	1C
		E8 0000	1D
		F0 0000	1E
		F8 0000	1F

**Table A.8 : A24 Slave Image Programming (Continued)**

A24SIZ State	A24 Slave Image Size	Acceptable A24 Slave Image Base Addresses	A24BA State
01	1Mbyte	00 0000	00
		10 0000	02
		20 0000	04
		30 0000	06
		40 0000	08
		50 0000	0A
		60 0000	0C
		70 0000	0E
		80 0000	10
		90 0000	12
		A0 0000	14
		B0 0000	16
		C0 0000	18
		D0 0000	1A
		E0 0000	1C
		F0 0000	1E
10	2 Mbytes	00 0000	00
		20 0000	04
		40 0000	08
		60 0000	0C
		80 0000	10
		A0 0000	14
		C0 0000	18
		E0 0000	1C
11	4 Mbytes	00 0000	00
		40 0000	08
		80 0000	10
		C0 0000	18

**Table A.9 : A32 Slave Image Size (4 Bit Field)**

State	A32 Slave Image Size	State	A32 Slave Image Size	State	A32 Slave Image Size	State	A32 Slave Image Size
0	4K	4	64K	8	1M	C	16M
1	8K	5	128K	9	2M	D	32M
2	16K	6	256K	A	4M	E	64M
3	32K	7	512K	B	8M	F	128M

**Table A.10 : A32 Slave Image Base Address (5 Bit Field)**

State	Acceptable A32 Slave Image Base Address	State	Acceptable A32 Slave Image Base Address	State	Acceptable A32 Slave Image Base Address	State	Acceptable A32 Slave Image Base Address
0	0000.0000	8	4000.0000	10	8000.0000	18	C000.0000
1	0800.0000	9	4800.0000	11	8800.0000	19	C800.0000
2	1000.0000	A	5000.0000	12	9000.0000	1A	D000.0000
3	1800.0000	B	5800.0000	13	9800.0000	1B	D800.0000
4	2000.0000	C	6000.0000	14	A000.0000	1C	E000.0000
5	2800.0000	D	6800.0000	15	A800.0000	1D	E800.0000
6	3000.0000	E	7000.0000	16	B000.0000	1E	F000.0000
7	3800.0000	F	7800.0000	17	B800.0000	1F	F800.0000

**Table A.11 : RXFIFO Data Register (RXDATA)**

Register Name: RXDATA		Register Address: xxxx x014	
Bits	Function		
31-24	Data Lane at Output Stage of Receive FIFO - Byte 3		
23-16	Data Lane at Output Stage of Receive FIFO - Byte 2		
15-08	Data Lane at Output Stage of Receive FIFO - Byte 1		
07-00	Data Lane at Output Stage of Receive FIFO - Byte 0		

**RXDATA Description**

Name	Type	Condition after Reset	State	Function
RDATA 31 – 00	R	U		Receive FIFO output data

**Table A.12 : RXFIFO Address Register (RXADDR)**

Register Name: RXADDR		Register Address: xxxx x018	
Bits	Function		
31-24	Address Lane at Output Stage of Receive FIFO - Byte 3		
23-16	Address Lane at Output Stage of Receive FIFO - Byte 2		
15-08	Address Lane at Output Stage of Receive FIFO - Byte 1		
07-00	Address Lane at Output Stage of Receive FIFO - Byte 0		

**RXADDR Description**

Name	Type	Condition after Reset	State	Function
RADDR 31 – 00	R	U		Receive FIFO output address, except during multiplexed block data transfers in which case D63-D32 are provided here. Upper address bits may be zeroed according to the size of slave image programmed that accepted the transfer. No zeroing effect on the data received during MBLT transfers.

Table A.13 : RXFIFO Control Register (RXCTL)

Register Name: RXCTL					Register Address: xxxx 01C			
Bits	Function							
31-24	Not Used							
23-16	Not Used							
15-08	Not Used							0
07-00	0	BLKST	MBLT	BLT	SIZ1	SIZ0	SPC1	SPC0

### RXCTL Description

Name	Type	Condition after Reset	State	Function
BLKST	R	U	0 1	Flags the start of a block in the FIFO This cycle is <i>not</i> the start of a block This cycle is the start of a block
MBLT	R	U	0 1	Identifies the entry as being part of an MBLT block This cycle is <i>not</i> part of an MBLT block This cycle is part of an MBLT block
BLT	R	U	0 1	Identifies the entry as being part of a BLT block This cycle is <i>not</i> part of a BLT block This cycle is part of a BLT block
SIZ0, 1	R	U	0,0 0,1 1,0 1,1	Data size Longword Byte Word Tri-byte
SPC0, 1	R	U	0,0 0,1 1,0 1,1	Receive FIFO TC1, 0 output bits User program space User data space Supervisor program space Supervisor data space

**Table A.14 : VMEbus/VSB Bus Select (BUSSEL)**

Register Name: BUSSEL		Register Address: xxxx x020
Bits	Function	
31-24	VSBEN (VMEbus/VSB Data Transfer Cycle Routing Bits) - Byte 3	
23-16	VSBEN (VMEbus/VSB Data Transfer Cycle Routing Bits) - Byte 2	
15-08	VSBEN (VMEbus/VSB Data Transfer Cycle Routing Bits) - Byte 1	
07-00	VSBEN (VMEbus/VSB Data Transfer Cycle Routing Bits) - Byte 0	

**BUSSEL Description**

Name	Type	Condition after Reset	State	Function
VSBEN 31 – 00	R/W	0		VMEbus/VSB select bits, one for each of the 32 by 128 Mbyte pages (refer to Address Range Map below)
			0	VMEbus selected
			1	VSB selected

Bit	Address Range Mapped		Bit	Address Range Mapped
0	0000.0000 - 07FF.FFFF		16	8000.0000 - 87FF.FFFF
1	0800.0000 - 0FFF.FFFF		17	8800.0000 - 8FFF.FFFF
2	1000.0000 - 17FF.FFFF		18	9000.0000 - 97FF.FFFF
3	1800.0000 - 1FFF.FFFF		19	9800.0000 - 9FFF.FFFF
4	2000.0000 - 27FF.FFFF		20	A000.0000 - A7FF.FFFF
5	2800.0000 - 2FFF.FFFF		21	A800.0000 - AFFF.FFFF
6	3000.0000 - 37FF.FFFF		22	B000.0000 - B7FF.FFFF
7	3800.0000 - 3FFF.FFFF		23	B800.0000 - BFFF.FFFF
8	4000.0000 - 47FF.FFFF		24	C000.0000 - C7FF.FFFF
9	4800.0000 - 4FFF.FFFF		25	C800.0000 - CFFF.FFFF
10	5000.0000 - 57FF.FFFF		26	D000.0000 - D7FF.FFFF
11	5800.0000 - 5FFF.FFFF		27	D800.0000 - DFFF.FFFF
12	6000.0000 - 67FF.FFFF		28	E000.0000 - E7FF.FFFF
13	6800.0000 - 6FFF.FFFF		29	E800.0000 - EFFF.FFFF
14	7000.0000 - 77FF.FFFF		30	F000.0000 - F7FF.FFFF
15	7800.0000 - 7FFF.FFFF		31	F800.0000 - FFFF.FFFF

**Table A.15 : VMEbus Interrupter Vector (IVECT)**

Register Name: IVECT		Register Address: xxxx x024	
Bits	Function		
31-24	Not Used		
23-16	Not Used		
15-08	Not Used		
07-00	IVECT (Interrupt Vector)		

**IVECT Description**

Name	Type	Condition after Reset	State	Function
IVECT 07 – 00	R/W	U		VMEbus interrupt vector bits



**Table A.16 : Access Protect Boundary Register (APBR)**

Register Name: APBR		Register Address: xxxx x028				
Bits	Function					
31-24	Not Used					
23-16	Not Used					
15-08	Not Used					
07-00	Not Used	APB03	APB02	APB01	APB00	

**APBR Description**

Name	Type	Condition after Reset	State	Function
APB03 – 00	R/W	0		Access protection boundary; protection enforced below
			0	No protection
			1	Lower 64 KB
			2	Lower 128 KB
			3	Lower 256 KB
			4	Lower 512 KB
			5	Lower 1 MB
			6	Lower 2 MB
			7	Lower 4 MB
			8	Lower 8 MB
			9	Lower 16 MB
			A	Lower 32 MB
			B	Lower 64 MB
			C	Entire 128 MB
			D	Entire 128 MB
			E	Entire 128 MB
			F	Entire 128 MB

**Table A.17 : TXFIFO Data Output Latch Register (TXDATA)**

Register Name: TXDATA		Register Address: xxxx x02C
Bits	Function	
31-24	LM (Data at Output Stage of Location Monitor FIFO) - Byte 3	
23-16	LM (Data at Output Stage of Location Monitor FIFO) - Byte 2	
15-08	LM (Data at Output Stage of Location Monitor FIFO) - Byte 1	
07-00	LM (Data at Output Stage of Location Monitor FIFO) - Byte 0	

**TXDATA Description**

Name	Type	Condition after Reset	State	Function
TDATA 31 – 00	R	U		Data at transmit FIFO data lane output stage. During D64 cycles this will be D31 – D00.

**Table A.18 : TXFIFO Address Output Latch Register (TXADDR)**

Register Name: TXADDR		Register Address: xxxx x030
Bits	Function	
31-24	Address Lane at Output Stage of Transmit FIFO - Byte 3	
23-16	Address Lane at Output Stage of Transmit FIFO - Byte 2	
15-08	Address Lane at Output Stage of Transmit FIFO - Byte1	
07-00	Address Lane at Output Stage of Transmit FIFO - Byte0	

**TXADDR Description**

Name	Type	Condition after Reset	State	Function
TADDR 31 – 00	R	U		Address at transmit FIFO stage for discrete or non-block DMA transfers. D63 – D32 data during multiplexed block transfers.

**Table A.19 : Transmit FIFO AM Code and Control Bit Latch (TXCTL)**

Register Name: TXCTL					Register Address: xxxx x034			
Bits	Function							
31-24	Not Used							
23-16	Not Used							
15-08	Not Used							
07-00	MBLT	BLT	SIZ1	SIZ0	SPC1	SPC0	SUPER	TYPE

**TXCTL Description**

Name	Type	Condition after Reset	State	Function
MBLT	R	U	0 1	MBLT transfer type flag This transfer is <i>not</i> part of an MBLT block This transfer is part of an MBLT block
BLT	R	U	0 1	BLT transfer type flag This transfer is <i>not</i> part of a BLT block This transfer is part of a BLT block
SIZ0-1	R	U	0,0 0,1 1,0 1,1	Transfer size Longword Byte Word Tri-byte
SPC0-1	R	U	0,0 0,1 1,0 1,1	Address space A32 Reserved A16 A24
SUPER	R	U	0 1	Privilege level of transfer User (non privileged) Supervisor
TYPE	R	U	0 1 0 1	Data type is CPU Transfer: Program Data Data type is DMA Transfer: MBLT BLT or non-block

**Table A.20 : Location Monitor FIFO Read Port (LMFIFO)**

Register Name: LMFIFO		Register Address: xxxx x038	
Bits	Function		
31-24	LM (Location Monitor FIFO Output Stage Data) - Byte 3		
23-16	LM (Location Monitor FIFO Output Stage Data) - Byte 2		
15-08	LM (Location Monitor FIFO Output Stage Data) - Byte 1		
07-00	LM (Location Monitor FIFO Output Stage Data) - Byte 0		

**LMFIFO Description**

Name	Type	Condition after Reset	State	Function
LM 31 – 00	R	U		Data at output stage of Location Monitor FIFO

**Table A.21 : Mode Control (MODE)**

Register Name: MODE				Register Address xxxx x03C				
Bits	Function							
31-24	DCTSI	Reserved	RMCRETRY	DMABEN	RMCPIN	BUSSIZ	SWAP	0
23-16	DPRIV	NOREL	FILL	DMA64	MBLT	BLT	DMAD16	DMARD
15-08	FIFOBEN	BLEN (2 bits)		RXATOM	TXATOM	A24SLVEN	DMA24	BERRCHK
07-00	TASCON	A24PO	LPBK	DISRX	A24DI	A16DI	PROT	VINEN

**MODE Description**

Name	Type	Condition after Reset	State	Function
DCTSI	R/W	0	0 1	Sets the DMA transfer count size 12 bit transfer count (4K) 20 bit transfer count (1 M)
RMCRETRY	R/W	0	0 1	Response to a RMW cycle deadlock Indivisible (RMW) cycles that encounter deadlock conditions are terminated with $\overline{KBERR}$ . Indivisible (RMW) cycles that encounter deadlock conditions are terminated with $\overline{KBERR}$ and $\overline{KHALT}$ .
DMABEN	R/W	0	0 1	DMA local bus burst enable Burst mode disabled for DMA Burst mode enabled for DMA
RMCPIN	R/W	0	0 1	RMC pin configuration RETRY/VRMC pin in VRMC mode RETRY/VRMC pin is in RETRY mode
BUSSIZ	R/W	0	0 1	Dynamic bus sizing request enable Enable sizing requests to CPU, implement D16 spaces, disable UAT cycles Disable sizing requests to CPU, set VMEbus to all D32, enable UAT cycles
SWAP	R/W	0	0 1	Word swap enable Disable word swapping Enable word swapping
DPRIV	R/W	0	0 1	DMAC privilege type for VMEbus cycles DMAC uses non-privileged AM codes DMAC uses supervisory AM codes

**MODE Description (Continued)**

Name	Type	Condition after Reset	State	Function
NOREL	R/W	0	0 1	VMEbus no-release mode SCV64 will release VMEbus when done SCV64 will maintain VMEbus ownership until forced off
FILL	R/W	0	0 1	Transmit FIFO fill mode Request VMEbus when any entries are in the FIFO If DMA is running, wait for FIFO full before requesting VMEbus. If DMA is not running, same as FILL = 0
DMAA64	R/W	0	0 1	A64 mode control for the DMA DMAC does not use A64 mode DMA uses A64 mode and Master A64 Base Addr Reg
MBLT	R/W	0	0 1	Multiplexed block transfer mode DMAC control DMAC does not use MBLT mode DMAC uses MBLT mode
BLT	R/W	0	0 1	Non-multiplexed block transfer mode control for DMAC DMAC does not use BLT mode DMAC uses BLT mode
DMAD16	R/W	0	0 1	Data size for DMAC in non-block and BLT block modes DMAC uses D32 transfers DMAC uses D16 transfers
DMARD	R/W	0	0 1	DMAC read or write mode DMAC transfers from local memory to VMEbus DMAC transfers from VMEbus into local memory
FIFOBEN	R/W	0	0 1	Receive FIFO burst enable Disable burst mode for receive FIFO Enable burst mode for receive FIFO
BLN 1-0	R/W	0	0 1 2 3	Local bus maximum burst length 4longwords 8longwords 16longwords 32longwords
RXATOM	R/W	0	0 1	Receive FIFO Coupled and Decoupled RX FIFOs used in decoupled mode RX FIFOs bypassed (Coupled mode)

**MODE Description (Continued)**

Name	Type	Condition after Reset	State	Function
TXATOM	R/W	0	0 1	Transmit FIFO Coupled and Decoupled TX FIFOs used in decoupled mode TX FIFOs bypassed (Coupled mode)
A24SLVEN	R/W	0	0 1	A24 slave image enable Image will not respond Image is enabled
DMA24	R/W	0	0 1	DMA destination address size DMA operates as A32 master DMA operates as A24 master
BERRCHK	R/W	0	0 1	Local late bus error enable No extra delay inserted. 2clock local delay inserted to check for late BERR from RAM
TASCON	R/W	0	0 1	AS* modifier for RMW cycles AS* negated between every VMEbus cycle AS* not negated between cycles while CPU RMC is asserted
A24PO	R/W	0	0 1	VMEbus A24 space address A24 space located at F800.0000 A24 space located at 0000.0000
LPBK	R/W	0	0 1	Loopback enable bit No loopback Loopback through FIFOs enabled
DISRX	R/W	0	0 1	Receive FIFO disable bit Normal operation FIFO emptied by RXSHFT control bit only
A24DI	R/W	0	0 1	Page 0 VMEbus A24 disable bit A24 responds per A24P0 bit A24 disabled if in page 0
A16DI	R/W	0	0 1	VMEbus A16 disable bit VMEbus A16 located at FFFF.0000 A16 space disabled
PROT	R/W	0	0 1	Access protection type Write protection only Read and write protection

**MODE Description (Continued)**

Name	Type	Condition after Reset	State	Function
VINEN	R/W	0		Slave images enabling
			R 0	All slave images are disabled
			R 1	All programmed images enabled
			W 0	Disable all slave images
			W 1	Enable all programmed images



*Caution: The RXATOM bit in the MODE register should not be modified by the local CPU while there is the possibility of an incoming VMEbus slave cycle since the SCV64 may not be able to decode the incoming cycle. If you need to modify the RXATOM bit when there is the possibility of an incoming VMEbus slave cycle, have the SCV64 obtain and hold the VMEbus before changing the bit, as follows:*

- Clear the OTEN and BCEN bits in the VREQ register.
- Set the NOREL bit in the MODE register.
- Have the local CPU perform a read of VME memory (forces the SCV64 to obtain the VME bus).
- Change the status of the RXATOM bit in the MODE register
- Clear the NOREL bit in the MODE register
- Restore the OTEN and BCEN bits to their original state.

If a VME host would like to program a slave SCV64 board into coupled mode (i.e. RXATOM changing from 0 to 1), then a VME read of any resource on the same slave board should immediately follow the VME write cycle. The read cycle forces the write to the MODE register to complete on the local bus and therefore the RXATOM bit to be changed without any adverse effects. The VME host must maintain ownership of the VMEbus between the write and read cycles.



**Table A.22 : Slave A64 Base Address Register (SA64BAR)**

Register Name: SA64BAR		Register Address: xxxx040	
Bits	Function		
31-24	A63 - A56 Address Bits for Slave A64 Base Address		
23-16	A55 - A48 Address Bits for Slave A64 Base Address		
15-08	A47 - A40 Address Bits for Slave A64 Base Address		
07-00	A39 - A32 Address Bits for Slave A64 Base Address		

**SA64BAR Description**

Name	Type	Condition after Reset	State	Function
SA64B31 – 00	R/W	U	x	This parameter defines the base address for slave A64 accesses.



*SA64BAR can only be programmed from the VMEbus by using a coupled cycle (see Coupled Mode on page 2-39).*

**Table A.23 : Master A64 Base Address Register (MA64BAR)**

Register Name: MA64BAR		Register Address: xxxx x044	
Bits	Function		
31-24	A63 - A56 Address Bits for Master A64 Base Address		
23-16	A55 - A48 Address Bits for Master A64 Base Address		
15-08	A47 - A40 Address Bits for Master A64 Base Address		
07-00	A39 - A32 Address Bits for Master A64 Base Address		

**MA64BAR Description**

Name	Type	Condition after Reset	State	Function
MA64B31 – 00	R/W	U	x	A64 - A32 address bits during master A64 DMA transfers



*MA64BAR can only be programmed from the VMEbus by using a coupled cycle (see Coupled Mode on page 2-39).*

**Table A.24 : Local Address Generator (LAG)**

Register Name: LAG		Register Address: xxxx x048	
Bits	Function		
31-24	Not Used (5 bits)		Local Address Generator Value
23-16	Local Address Generator Value		
15-08	Local Address Generator Value		
07-00	Local Address Generator Value (Total 27 bits)		

**LAG Description**

Name	Type	Condition after Reset	State	Function
LAG26 – 00	R	U	x	This is the next local address to be used in block mode transfers received from the VMEbus, either as a slave or by DMAC transfers from the VMEbus to local memory.

**Table A.25 : DMA VMEbus Transfer Count (DMAVTC)**

Register Name: DMAVTC		Register Address: xxxx x04C	
Bits	Function		
31-24	Not Used		
23-16	Not Used	DMA VMEbus Transfer Count	
15-08	DMA VMEbus Transfer Count		
07-00	DMA VMEbus Transfer Count (Total 20 bits)		

**DMAVTC Description**

Name	Type	Condition after Reset	State	Function
DMAVTC19-00	R	U	0	The number of VMEbus cycles remaining to perform in the BLT or non-block DMA transfer. If an MBLT transfer is being performed then the number of cycles remaining is half the value read from this register.

Table A.26 : Status Register 0 (STAT0)

<b>Register Name: STAT0</b>	<b>Register Address: xxxx x080</b>
-----------------------------	------------------------------------

Bits	Function							
07-00	CLRDOG	CLRTIK	LTO	VI1	SYFIP	ACFIP	MEMIP	NMIIP

### STAT0 Description

Name	Type	Condition after Reset	State	Function
CLRDOG	R/W	0	W0 W1	Watchdog clear and restart No effect If 0 before write, resets; If 1 before write, no effect
CLRTIK	R/W	0	R W 0 W 1	Clear TICK signal Always reads as zero TICK pin cleared to high No effect
LTO	R/W	0	R 0 R 1 W 0 W 1	Local Bus Timed Out flag No time-out has occurred Local bus time-out occurred Clears LTO bit (does not reset counter) No effect
VI1	R	X	0 1	VMEbus IRQ1* (BI-mode line) IRQ1* is not asserted IRQ1* is asserted
SYFIP	R	1	0 1	<b>SYSFAIL</b> * Pin Pin is not asserted Pin is asserted
ACFIP	R	X	0 1	$\overline{L7IACF}$ Interrupt Pin Pin is not asserted Pin is asserted
MEMIP	R	X	0 1	$\overline{L7IMEM}$ Interrupt Pin Pin is not asserted Pin is asserted
NMIIP	R	X	0 1	$\overline{L7INMI}$ Interrupt Pin Pin is not asserted Pin is asserted

**Table A.27 : Status Register 1 (STAT1)**

Register Name: STAT1					Register Address: xxxx x084			
Bits	Function							
07-00	1	0	SYSC	BI	PWRUP	IDGOT	BG2IN	BG1IN

**STAT1 Description**

Name	Type	Condition after Reset	State	Function
SYSC	R/W	X		VMEbus SYSCON mode bit
			R0	SCV64 is not SYSCON
			R1	SCV64 is SYSCON
			W0	SCV64 ceases to be SYSCON
			W1	SCV64 becomes SYSCON
BI	R	1		BI-mode Indicator
			0	SCV64 is not in BI-mode
			1	SCV64 is in BI-mode
PWRUP	R	X		Power-up indicator
			0	Last reset was not by power up
			1	Last reset was due to power up
IDGOT	R	X		Auto-ID Completion indicator
			0	Auto-ID cycle not done yet
			1	Auto-ID cycle completed
BG2IN	R	X		Bus Grant 2 In pin
			0	BG2IN* is low or SCV64 isn't SYSCON
			1	BG2IN* is high and SCV64 is SYSCON
BG1IN	R	X		Bus Grant 1 In pin
			0	BG1IN* is low or SCV64 isn't SYSCON
			1	BG1IN* is high and SCV64 is SYSCON

Table A.28 : General Control Register (GENCTL)

<b>Register Name: GENCTL</b>	<b>Register Address: xxxx x088</b>
------------------------------	------------------------------------

Bits	Function							
07-00	SWRST	ABI	SBI	TLEN1	TLEN0	LTOEN	IDTST	VI1BI

### GENCTL Description

Name	Type	Condition after Reset	State	Function											
SWRST	R/W	0	0 1	Software Initiated System Reset No effect Initiates reset											
ABI	R/W	0	0 1	Assert BI-mode line (IRQ1*) De-assert IRQ1* Assert IRQ1*											
SBI	R/W	0	0 1	Self BI-mode control bit De-assert software BI-mode trigger Set self into BI-mode											
TLEN1 TLEN0	R/W	3	0 1 2 3	Tick timer length code <table><tr><td>Norm</td><td>Fast</td></tr><tr><td>5 ms</td><td>0.2 ms</td></tr><tr><td>10 ms</td><td>0.4 ms</td></tr><tr><td>50 ms</td><td>2 ms</td></tr><tr><td>100 ms</td><td>4 ms</td></tr></table>		Norm	Fast	5 ms	0.2 ms	10 ms	0.4 ms	50 ms	2 ms	100 ms	4 ms
Norm	Fast														
5 ms	0.2 ms														
10 ms	0.4 ms														
50 ms	2 ms														
100 ms	4 ms														
LTOEN	R/W	1	0 1	Local Bus Time-out Enable Disable time-out Enable time-out (512 μs)											
IDTST	R/W	0	W 0 W 1 W 1	Auto-ID Test Bit No effect If previously 0, increment ID If previously 1, no effect											
VI1BI	R/W	1	0 1	IRQ1* Configuration Control Bit IRQ1* as interrupt only IRQ1* as BI-mode line only											

**Table A.29 : VMEbus Interrupter Requester (VINT)**

Register Name: VINT					Register Address: xxxx x08C			
Bits	Function							
07-00	0	0	0	0	INT	IL2	IL1	IL0

**VINT Description**

Name	Type	Condition after Reset	State	Function
INT	R/W	0	R 0 R 1 W 0 W 1	Interrupt Control and Status bit No interrupt is pending Interrupt is presently asserted No effect; does not change to 0 Assert interrupt
IL2 IL1 IL0	R/W	0	0 1-7	Interrupt Level Interrupter clears itself Sets interrupt level



*Caution: Changing the interrupt level while INT is state 1 may cause improper operation in the IACK DCD logic.*

**Table A.30 : VMEbus Requester Register (VREQ)**

<b>Register Name: VREQ</b>	<b>Register Address: xxxx x090</b>
----------------------------	------------------------------------

Bits	Function							
07-00	OTEN	BCEN	REL	REQ	OT1	OT0	LVL1	LVL0

**VREQ Description**

Name	Type	Condition after Reset	State	Function
OTEN	R/W	1	0 1	VMEbus Ownership Timer Enable Disable timer Enable timer
BCEN	R/W	0	0 1	Bus Clear Recognition Control Ignore BCLR* signal Release bus if BCLR* asserted
REL	R/W	1	0 1	VMEbus Release Mode Control Release on request (ROR) Release when done (RWD)
REQ	R/W	1	0 1	VMEbus Request Mode Control Fair Demand
OT1 OT0	R/W	3	0 1 2 3	VMEbus Ownership Timer (Time-out Period) Zero 2 $\mu$ s 4 $\mu$ s 8 $\mu$ s
LVL1 LVL0	R/W	3	0 1 2 3	VMEbus Request Level Level 0 Level 1 Level 2 Level 3



**Table A.31 : VMEbus Arbiter Register (VARB)**

Register Name: VARB					Register Address: xxxx x094				
Bits	Function								
07-00	0	0	VXL1	VXL0	0	ATEN	ARB1	ARB0	

**VARB Description**

Name	Type	Condition after Reset	State	Function
VXL1 VXL0	R/W	3	0 1 2 3	Data Transfer Time-out Period Never 16 $\mu$ s 32 $\mu$ s 64 $\mu$ s
ATEN	R/W	1	0 1	Arbitration Time-out Enable Disable arbitration time-out Enable arbitration time-out
ARB1 ARB0	R/W	0	0 1 2 3	Arbitration Mode Round Robin all four levels Priority 3, Round Robin 2, 1, 0 Priority 3, 2, Round Robin 1, 0 Priority on all four levels

Table A.32 : ID Register (ID)

Register Name: ID					Register Address: xxxx x098			
Bits	Function							
07-00	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

ID Description

Name	Type	Condition after Reset	State	Function
ID	R	0	n	Auto-ID Card Slot ID Value of ID counter



*Caution: If a read of the Auto-ID register is the first cycle which occurs on the SCV64’s local bus after the ID cycle completes, then an incorrect value will be returned. The user must ensure that a register access (read or write, from either interface) is made to any of the other SCV64 registers before this register is accessed.*

**Table A.33 : Control and Status Register (CTL2)**

Register Name: CTL2					Register Address: xxxx x09C			
Bits	Function							
07-00	0	0	MYBBSY	$\overline{\text{BITRIG}}$	0	LBRAM	MEMIM	TICKM

**CTL2 Description**

Name	Type	Condition after Reset	State	Function
MYBBSY	R	0	0 1	SCV64 BBSY* Output BBSY* is not asserted BBSY* is driven low
$\overline{\text{BITRIG}}$	R	U	0 1	BI-Mode Trigger $\overline{\text{BITRIG}}$ is not asserted $\overline{\text{BITRIG}}$ is asserted
LBRAM	R/W	0	0 1	Local Bus Requester Mode Channels 0 and 1 fair Channel 1 pre-empts channel 0
MEMIM	R/W	1	0 1	$\overline{\text{L7IMEM}}$ effect on requester $\overline{\text{L7IMEM}}$ does not affect requester $\overline{\text{L7IMEM}}$ forces requester to release VMEbus
TICKM	R/W	0	0 1	Tick Speed Normal rates Fast rates

**Table A.34 : Level 7 Interrupt Status Register (7IS)**

Register Name: 7IS					Register Address: xxxx x0A0			
Bits	Function							
07-00	0	0	0	BIS	SYFIS	ACFIS	MEMIS	NMIIS

**7IS Description**

Name	Type	Condition after Reset	State	Function
BIS	R	X	0 1	BI-mode Interrupt Interrupt is not asserted Interrupt is asserted
SYFIS	R	0	0 1	SYSFAIL* Pin Interrupt Interrupt is not asserted Interrupt is asserted
ACFIS	R	0	0 1	$\overline{L7IACF}$ Pin Interrupt Interrupt is not asserted Interrupt is asserted
MEMIS	R	0	0 1	$\overline{L7IMEM}$ Pin Interrupt Interrupt is not asserted Interrupt is asserted
NMIIS	R	(Note)	0 1	$\overline{L7INMI}$ Pin Interrupt Interrupt is not asserted Interrupt is asserted

**Note:**

The bits in this status register represent the current state of the interrupt latches corresponding to the five interrupts. The interrupt inputs themselves can be read through Status Register 0. The initial state of NMIIS is dependent on the circuit card design.

**Table A.35 : Local Interrupt Status Register (LIS)**

Register Name: LIS				Register Address: xxxx x0A4				
Bits	Function							
07-00	0	0	LI5	LI4	LI3	LI2	LI1	LI0

**LIS Description**

Name	Type	Condition after Reset	State	Function
LI5	R	X	0 1	Local Interrupt Five ( $\overline{\text{LIRQ5}}$ ) $\overline{\text{LIRQ5}}$ pin is not asserted Interrupt pin is asserted
LI4	R	X	0 1	Local Interrupt Four ( $\overline{\text{LIRQ4}}$ ) $\overline{\text{LIRQ4}}$ pin is not asserted Interrupt pin is asserted
LI3	R	X	0 1	Local Interrupt Three ( $\overline{\text{LIRQ3}}$ ) $\overline{\text{LIRQ3}}$ pin is not asserted Interrupt pin is asserted
LI2	R	X	0 1	Local Interrupt Two ( $\overline{\text{LIRQ2}}$ ) $\overline{\text{LIRQ2}}$ pin is not asserted Interrupt pin is asserted
LI1	R	X	0 1	Local Interrupt One ( $\overline{\text{LIRQ1}}$ ) $\overline{\text{LIRQ1}}$ pin is not asserted Interrupt pin is asserted
LI0	R	X	0 1	Local Interrupt Zero ( $\overline{\text{LIRQ0}}$ ) $\overline{\text{LIRQ0}}$ pin is not asserted Interrupt pin is asserted

Table A.36 : Level 7 Interrupt Enable Register (7IE)

Register Name: 7IE					Register Address: xxxx x0A8			
Bits	Function							
07-00	0	0	0	BIE	SYFIE	ACFIE	MEMIE	NMICLR

### 7IE Description

Name	Type	Condition after Reset	State	Function
BIE	R/W	0	0 1	BI-mode Interrupt Enable Disable interrupt Enable interrupt
SYFIE	R/W	0	0 1	SYSFAIL* Pin Interrupt Enable Disable interrupt Enable interrupt
ACFIE	R/W	0	0 1	$\overline{L7IACF}$ Pin Interrupt Enable Disable interrupt Enable interrupt
MEMIE	R/W	0	0 1	$\overline{L7IMEM}$ Pin Interrupt Enable Disable interrupt Enable interrupt
NMICLR	R/W	1	R W 0 W 1	$\overline{L7INMI}$ Pin Interrupt Clear Always reads one Clear interrupt; remains enabled No effect; always enabled

**Table A.37 : Local Interrupt Enable Register (LIE)**

Register Name: LIE					Register Address: xxxx x0AC			
Bits	Function							
07-00	0	0	L5E	L4E	L3E	L2E	L1E	L0E

**LIE Description**

Name	Type	Condition after Reset	State	Function
L5E	R/W	0	0 1	Local Interrupt 5 ( $\overline{\text{LIRQ5}}$ ) Enable Disable interrupt Enable interrupt
L4E	R/W	0	0 1	Local Interrupt 4 ( $\overline{\text{LIRQ4}}$ ) Enable Disable interrupt Enable interrupt
L3E	R/W	0	0 1	Local Interrupt 3 ( $\overline{\text{LIRQ3}}$ ) Enable Disable interrupt Enable interrupt
L2E	R/W	0	0 1	Local Interrupt 2 ( $\overline{\text{LIRQ2}}$ ) Enable Disable interrupt Enable interrupt
L1E	R/W	0	0 1	Local Interrupt 1 ( $\overline{\text{LIRQ1}}$ ) Enable Disable interrupt Enable interrupt
L0E	R/W	0	0 1	Local Interrupt 0 ( $\overline{\text{LIRQ0}}$ ) Enable Disable interrupt Enable interrupt

**Table A.38 : VMEbus Interrupt Enable Register (VIE)**

Register Name: VIE					Register Address: xxxx x0B0			
Bits	Function							
07-00	0	V7E	V6E	V5E	V4E	V3E	V2E	V1E

**VIE Description**

Name	Type	Condition after Reset	State	Function
V7E	R/W	0	0 1	VMEbus Interrupt 7 Enable Interrupt disabled Interrupt enabled
V6E	R/W	0	0 1	VMEbus Interrupt 6 Enable Interrupt disabled Interrupt enabled
V5E	R/W	0	0 1	VMEbus Interrupt 5 Enable Interrupt disabled Interrupt enabled
V4E	R/W	0	0 1	VMEbus Interrupt 4 Enable Interrupt disabled Interrupt enabled
V3E	R/W	0	0 1	VMEbus Interrupt 3 Enable Interrupt disabled Interrupt enabled
V2E	R/W	0	0 1	VMEbus Interrupt 2 Enable Interrupt disabled Interrupt enabled
V1E	R/W	0	0 1	VMEbus Interrupt 1 Enable Interrupt disabled Interrupt enabled



**Table A.39 : Local Interrupts 1 and 0 Control Register (IC10)**

Register Name: IC10					Register: xxxx x0B4			
Bits	Function							
07-00	1	1L2	1L1	1L0	1	0L2	0L1	0L0

**IC10 Description**

Name	Type	Condition after Reset	State	Function
1L2	R/W	0		Local Interrupt 1 ( $\overline{\text{LIRQ1}}$ ) Level
1L1			0	Masks interrupt
1L0			1-7	Maps pin to level 1-7
0L2	R/W	0		Local Interrupt 0 ( $\overline{\text{LIRQ0}}$ ) Level
0L1			0	Masks interrupt
0L0			1-7	Maps pin to level 1-7

**Table A.40 : Local Interrupts 3 and 2 Control Register (IC32)**

Register Name: IC32					Register Address: xxxx x0B8			
Bits	Function							
07-00	1	3L2	3L1	3L0	1	2L2	2L1	2L0

**IC32 Description**

Name	Type	Condition after Reset	State	Function
3L2	R/W	0		Local Interrupt 3 ( $\overline{\text{LIRQ3}}$ ) Level
3L1			0	Masks interrupt
3L0			1-7	Maps pin to level 1-7
2L2	R/W	0		Local Interrupt 2 ( $\overline{\text{LIRQ2}}$ ) Level
2L1			0	Masks interrupt
2L0			1-7	Maps pin to level 1-7

**Table A.41 : Local Interrupts 5 and 4 Control Register (IC54)**

Register Name: IC54					Register Address: xxxx x0BC			
Bits	Function							
07-00	5AV	5L2	5L1	5L0	4AV	4L2	4L1	4L0

**IC54 Description**

Name	Type	Condition after Reset	State	Function
5AV	RW	0	0 1	Local Interrupt 5 ( $\overline{\text{LIRQ5}}$ ) Vector Vectored Auto-vectored
5L2 5L1 5L0	R/W	0	0 1-7	Local Interrupt 5 ( $\overline{\text{LIRQ5}}$ ) Level Masks interrupt Maps pin to level 1-7
4AV	R/W	0	0 1	Local Interrupt 4 ( $\overline{\text{LIRQ4}}$ ) Vector Vectored Autovectored
4L2 4L1 4L0	R/W	0	0 1-7	Local Interrupt 4 ( $\overline{\text{LIRQ4}}$ ) Level Masks interrupt Maps pin to level 1-7

**Table A.42 : Miscellaneous Control Register (MISC)**

Register Name: MISC		Register Address: xxxx x0C0			
Bits	Function				
31-24	reserved				
23-16	reserved				
15-08	reserved	EXTKIACK	ARBBYP	AUTOBAR	
07-00	reserved			ENDOG	SYSFAIL

**MISC Description**

Name	Type	Condition after Reset	State	Function
EXTKIACK	R	configurable	0	internal KIACK decode
			1	external KIACK decode
ARBBYP	R	configurable	0	local arbiter active
			1	local arbiter bypassed
AUTOBAR	R	configurable	0	AUTOBAR disabled
			1	AUTOBAR enabled
ENDOG	R/W	0	0	Watchdog timer disabled
			1	Watchdog timer enabled
SYSFAIL	R/W	1	0	release SYSFAIL* and $\overline{\text{SYSFLED}}$
			1	assert SYSFAIL* and $\overline{\text{SYSFLED}}$

**Table A.43 : Delay Line Control Register (DLCT)**

<b>Register Name: DLCT</b>	<b>Register Address: xxxx x0C4</b>
----------------------------	------------------------------------

Bits	Function				
31-24	SEL_40	OFFSET_40			
23-16	SEL_30	OFFSET_30			
15-08	SEL_20	OFFSET_20			
07-00	see caution below		MKEY	KEY_20	KEY_30    KEY_40

**DLCT Description**

Name	Type	Condition after Reset	State	Function
SEL_40	R/W	0	0 1	add offset to calibrated value only use offset value
OFFSET_40	R/W	0	3F 00 40	calibrated value + 3F taps calibrated value + 0 taps calibrated value - 40 taps
SEL_30	R/W	0	0 1	add offset to calibrated value only use offset value
OFFSET_30	R/W	0	3F 00 40	calibrated value + 3F taps calibrated value + 0 taps calibrated value -40 taps
SEL_20	R/W	0	0 1	add offset to calibrated value only use offset value
OFFSET_20	R/W	0	3F 00 40	calibrated value + 3F taps calibrated value + 0 taps calibrated value - 40 taps
see caution below	read and write	0	N/A	user must write 0
MKEY	R/W	1	0 1	permanently disables writes to keys until next $\overline{\text{LRST}}$ . enable writes to keys
KEY_20	R/W	1	0 1	disables writes to OFFSET_20 field enables writes to OFFSET_20 field
KEY_30	R/W	1	0 1	disables writes to OFFSET_30 field enables writes to OFFSET_30 field
KEY_40	R/W	1	0 1	disables writes to OFFSET_40 field enables writes to OFFSET_40 field



*Caution: Always write 0 to these bits. Writing values other than 0 to this area will cause unpredictable SCV64 operation.*

**Table A.44 : Delay Line Status Register 1 (DLST1)**

Register Name: DLST1			Register Address: xxxx x0C8	
Bits	Function			
31-24	reserved			
23-16	reserved	CAL_40		
15-08	reserved	CAL_30		
07-00	reserved	CAL_20		

Name	Type	Condition after Reset	State	Function
CAL_40	R	N/A	N/A	internal calibration target for 40ns delay line
CAL_30	R	N/A	N/A	internal calibration target for 30ns delay line
CAL_20	R	N/A	N/A	internal calibration target for 20ns delay line



*The CAL fields in the DLST1 register will be updated to reflect their new values once the OFFSET fields in the DLCT register are programmed.*

**Table A.45 : Delay Line Status Register 2 (DLST2)**

Register Name: DLST2			Register Address: xxxx x0CC	
Bits	Function			
31-24	reserved			
23-16	reserved			
15-08	C Flag-30	ACT-30		
07-00	C Flag-20	ACT-20		
Name	Type	Condition after Reset	State	Function
CFLG_30	R	N/A	0 1	30ns delay line not calibrated 30ns delay line calibrated
CFLG_20	R	N/A	0 1	20ns delay line not calibrated 20ns delay line calibrated
ACT_30	R	N/A	N/A	actual selected tap for 30ns delay lines
ACT_20	R	N/A	N/A	actual selected tap for 20ns delay lines

**Table A.46 : Delay Line Status Register 3 (DLST3))**

Register Name: DLST3			Register Address: xxxx x0D0	
----------------------	--	--	-----------------------------	--

Bits	Function			
31-24	reserved			
23-16	CFLG_40_3	ACT_40_3		
15-08	CFLG_40_2	ACT_40_2		
07-00	CFLG_40_1	ACT_40_1		

Name	Type	Condition after Reset	State	Function
CFLG_40_3	R	N/A	0	40_3 ns delay line not calibrated
			1	40_3 ns delay line calibrated
CFLG_40_2	R	N/A	0	40_2 ns delay line not calibrated
			1	40_2 ns delay line calibrated
CFLG_40_1	R	N/A	0	40_1 ns delay line not calibrated
			1	40_1 ns delay line calibrated
ACT_40_3	R	N/A	N/A	actual tap offset for 40 ns delay line 3
ACT_40_2	R	N/A	N/A	actual tap offset for 40 ns delay line 2
ACT_40_1	R	N/A	N/A	actual tap offset for 40 ns delay line 1

**Table A.47 : Mailbox Register 0 (MBOX0)**

Register Name: MBOX0		Register Address: xxxx x0D4	
Bits	Function		
31-24	user defined		
23-16			
15-08			
07-00			

**Table A.48 : Mailbox Register 1 (MBOX1)**

Register Name: MBOX1		Register Address: xxxx x0D8	
Bits	Function		
31-24	user defined		
23-16			
15-08			
07-00			

**Table A.49 : Mailbox Register 2 (MBOX2)**

Register Name: MBOX2		Register Address: xxxx x0DC	
Bits	Function		
31-24	user defined		
23-16			
15-08			
07-00			

**Table A.50 : Mailbox Register 3 (MBOX3)**

Register Name: MBOX3		Register Address: xxxx x0E0	
Bits	Function		
31-24	user defined		
23-16			
15-08			
07-00			



# Appendix B SCV64 Timing Characteristics

## B.1 Timing Parameters

Test conditions for the timing parameters in Table B.1 are:

Commercial (Com)      0°C to 70°C, 5V ± 0.25V, and

Extended (Ext)        -55°C to 125°C, 5V ± 0.5V.

**Table B.1 : SCV64 AC Characteristics**

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
–	KCLK frequency (note 29)	20	33	20	40	25	MHz
t <sub>10</sub>	C32MHz high time (note 10)	10	–	10	–	–	ns
t <sub>11</sub>	C32MHz low time (note 10)	10	–	10	–	–	ns
t <sub>12</sub>	KCLK high time Note: Extended at 25 MHz is a minimum 15.5 ns	14.5	–	12.0	–	–	ns
t <sub>13</sub>	KCLK low time Note: Extended at 25 MHz is a minimum 16.5 ns	14.5	–	12.0	–	–	ns
t <sub>14</sub>	C32MHz, KCLK fall time	–	5	–	5	5	ns
t <sub>15</sub>	C32MHz, KCLK rise time	–	5	–	5	5	ns
t <sub>16</sub>	KCLK to C14US low	5	17	5	14	19	ns
t <sub>17</sub>	KCLK to C14US high	5	18	5	15	20	ns
t <sub>18</sub>	C14US low time	6.99	7.01	6.99	7.01	7.01	µs
t <sub>19</sub>	C14US high time	6.99	7.01	6.99	7.01	7.01	µs
t <sub>20</sub>	C14US period	14	14	14	14	14	µs
t <sub>21</sub>	C32MHz to SYSCLK high	3	10	3	8	11	ns
t <sub>22</sub>	C32MHz to SYSCLK low	3	11	3	9	12	ns
t <sub>23</sub>	SYSCLK high time	31	33	31	33	33	ns
t <sub>24</sub>	SYSCLK low time	30	30	30	30	31	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>25</sub>	SYSCLK period (note 1)	62.5	62.5	62.5	62.5	62.5	ns
t <sub>26</sub>	C32MHZ to C8MHZ high	4	13	4	11	14	ns
t <sub>27</sub>	C32MHZ to C8MHZ low	4	13	4	11	14	ns
t <sub>28</sub>	C8MHZ high time	62.3	62.5	62.3	62.5	62.4	ns
t <sub>29</sub>	C8MHZ low time	62.3	62.4	62.3	62.4	62.5	ns
t <sub>30</sub>	C8MHZ period (note 1)	124.8	124.8	124.8	124.8	124.8	ns
t <sub>31</sub>	C32MHZ to BAUDCLK high	4	13	4	11	14	ns
t <sub>32</sub>	C32MHZ to BAUDCLK low	4	12	4	10	13	ns
t <sub>33</sub>	BAUDCLK high time	186	187	186	187	187	ns
t <sub>34</sub>	BAUDCLK low time	218	220	218	220	220	ns
t <sub>35</sub>	BAUDCLK period (note 1)	406	406	406	406	406	ns
t <sub>36</sub>	$\overline{\text{WDOG}}$ low time	2	2	2	2	2	s
t <sub>38</sub>	$\overline{\text{PWRRST}}$ asserted to $\overline{\text{WDOG}}$ high	4	13	4	11	14	ns
t <sub>39</sub>	$\overline{\text{LRST}}$ asserted to $\overline{\text{WDOG}}$ high	9	27	9	22	30	ns
t <sub>45</sub>	BG0IN* asserted to SYSRST* (when SYSCON) (note 1)	56	70	56	70	70	ms
t <sub>46</sub>	BG0IN* minimum low time (note 1)	84	–	84	–	–	ms
t <sub>50</sub>	$\overline{\text{KIPL}}$ change from $\overline{\text{LIRQn}}$ asserted	6	26	6	21	27	ns
t <sub>51</sub>	$\overline{\text{KIPL}}$ change from $\overline{\text{L7Ixxx}}$ asserted (notes 1 and 7)	66	83	66	80	85	ns
t <sub>52</sub>	$\overline{\text{KIPL}}$ change from SYSFAIL* asserted (notes 1 and 7)	66	83	66	80	85	ns
t <sub>53</sub>	$\overline{\text{KIPL}}$ change from IRQn* asserted	4	17	4	14	19	ns
t <sub>54</sub>	$\overline{\text{KIPL}}$ change from KCLK (unmasking interrupt)	9	31	9	25	34	ns
t <sub>55</sub>	SYSRST* minimum assertion time (note 1, 15)	224	238	224	238	238	ms
t <sub>56</sub>	$\overline{\text{L7Ixxx}}$ minimum assertion time (note 1)	62	94	62	94	94	ns
t <sub>57</sub>	SYSFAIL* minimum assertion time (note 1)	62	94	62	94	94	ns
t <sub>58</sub>	$\overline{\text{PWRRST}}$ minimum low time	50	–	50	–	–	ns
t <sub>59</sub>	$\overline{\text{EXTRST}}$ minimum low time	50	–	50	–	–	ns
t <sub>60</sub>	$\overline{\text{LMINT}}$ asserted from start of S5 (LM write)	8	26	8	21	29	ns
t <sub>61</sub>	$\overline{\text{LMINT}}$ negated from $\overline{\text{KAS}}$ negated (LM FIFO read)	7	25	7	20	27	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>62</sub>	BIMODE negated from KCLK (LM write)	6	21	6	17	23	ns
t <sub>70</sub>	$\overline{\text{LIACKn}}$ asserted from KCLK	5	18	5	14	20	ns
t <sub>71</sub>	$\overline{\text{LIACKn}}$ negated from $\overline{\text{KDS}}$ (notes 4 and 6)	0	16	0	13	17	ns
t <sub>72</sub>	$\overline{\text{KAVEC}}$ asserted from KCLK	6	19	6	16	21	ns
t <sub>73</sub>	$\overline{\text{KAVEC}}$ negated from $\overline{\text{KDS}}$ (notes 4 and 6)	0	17	0	14	19	ns
t <sub>74</sub>	$\overline{\text{LIACKn}}$ negated from $\overline{\text{KIACK}}$ negated (notes 4, 6)	4	14	4	12	16	ns
t <sub>75</sub>	$\overline{\text{KAVEC}}$ negated from $\overline{\text{KIACK}}$ negated (notes 4, 6)	5	16	5	13	17	ns
t <sub>76</sub>	$\overline{\text{LIACKn}}$ negated from $\overline{\text{KAS}}$ (note 5)	5	16	5	13	17	ns
t <sub>77</sub>	$\overline{\text{KAVEC}}$ negated from $\overline{\text{KAS}}$ (note 5)	6	19	6	16	21	ns
t <sub>78</sub>	$\overline{\text{LIACKn}}$ negated from $\overline{\text{LIRQn}}$ negated	4	18	4	15	20	ns
t <sub>79</sub>	$\overline{\text{KAVEC}}$ negated from $\overline{\text{LIRQn}}$ negated	4	19	4	16	21	ns
t <sub>81</sub>	$\overline{\text{KIACK}}$ setup to KCLK	10	–	8	–	–	ns
t <sub>85</sub>	SYSRST asserted from $\overline{\text{PWRRST}}$ asserted	5	16	5	13	18	ns
t <sub>86</sub>	$\overline{\text{LRST}}$ asserted from $\overline{\text{PWRRST}}$ asserted	6	20	6	16	22	ns
t <sub>87</sub>	$\overline{\text{EXTRST}}$ to $\overline{\text{LRST}}$ asserted	6	18	6	15	20	ns
t <sub>88</sub>	SYSRST* asserted to $\overline{\text{LRST}}$ asserted (note 14)	6	18	6	15	20	ns
t <sub>89</sub>	BIMODE asserted from $\overline{\text{LRST}}$ asserted (note 14)	1	6	1	5	7	ns
t <sub>95</sub>	$\overline{\text{PWRRST}}$ held low from V <sub>DD</sub> stable (V <sub>DD</sub> = 4.5V)	100	–	100	–	–	ns
t <sub>97</sub>	BG3IN* hold from $\overline{\text{PWRRST}}$ negated (note 1)	750	–	750	–	–	ns
t <sub>98</sub>	BG3IN* hold from SYSRST*	20	–	20	–	–	ns
t <sub>99</sub>	KFC, $\overline{\text{KGBACK}}$ , KDATA hold from $\overline{\text{PWRRST}}$ negated	25	–	25	–	–	ns
t <sub>100</sub>	$\overline{\text{SCV64SEL}}$ setup to end of S2	11	–	9	–	–	ns
t <sub>101</sub>	$\overline{\text{SCV64SEL}}$ hold from $\overline{\text{KAS}}$	9	–	8	–	–	ns
t <sub>103</sub>	$\overline{\text{SCV64SEL}}$ setup to $\overline{\text{KAS}}$ (notes 31 and 34)	-(2T-5)	–	-(2T-5)	–	–	ns
t <sub>105</sub>	KADDR, KFC, KSIZE setup to KCLK	12	–	10	–	–	ns
t <sub>106</sub>	KADDR, KSIZE hold from KCLK	0	–	0	–	–	ns
t <sub>107</sub>	KADDR, KSIZE hold from $\overline{\text{KAS}}$	0	–	0	–	–	ns
t <sub>108</sub>	KDATA setup to KCLK	5	–	3	–	–	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>109</sub>	KDATA hold from KCLK	12	–	9	–	–	ns
t <sub>111</sub>	$\overline{KAS}$ held high from end of S0	5	–	4	–	–	ns
t <sub>112</sub>	$\overline{KAS}$ , $\overline{KDS}$ setup to KCLK	5	–	4	–	–	ns
t <sub>113</sub>	$\overline{KAS}$ , $\overline{KDS}$ hold from KCLK	7	–	6	–	–	ns
t <sub>116</sub>	$\overline{KWR}$ setup to KCLK	2	–	2	–	–	ns
t <sub>117</sub>	$\overline{KWR}$ hold from $\overline{KAS}$ or $\overline{KDS}$	10	–	8	–	–	ns
t <sub>118</sub>	$\overline{KWR}$ hold from KCLK	5	–	4	–	–	ns
t <sub>120</sub>	$\overline{KDSACK}$ asserted from end of S2	0	31	0	25	34	ns
t <sub>121</sub>	$\overline{KDSACK}$ negated from KAS negated	5	18	5	15	20	ns
t <sub>122</sub>	$\overline{KDSACK}$ tri-state from KCLK	0	34	0	28	37	ns
t <sub>123</sub>	$\overline{KBERR}$ asserted from end of S2	8	31	8	25	34	ns
t <sub>124</sub>	$\overline{KBERR}$ negated from $\overline{KAS}$ negated	7	25	7	20	28	ns
t <sub>125</sub>	$\overline{KBERR}$ tri-state from $\overline{KBERR}$ negated	10	32	10	26	35	ns
t <sub>127</sub>	$\overline{KHALT}$ asserted from end of S2	6	20	6	16	22	ns
t <sub>128</sub>	$\overline{KHALT}$ negated from end of S5	4	16	4	13	17	ns
t <sub>129</sub>	$\overline{KHALT}$ tri-state from $\overline{KHALT}$ negated	5	16	5	13	18	ns
t <sub>130</sub>	KDATA driven from end of S2	8	32	8	26	35	ns
t <sub>131</sub>	KDATA valid from end of S2	8	32	8	26	35	ns
t <sub>136</sub>	$\overline{KAS}$ negated to KDATA invalid (Note 2)	0	31	0	25	33	ns
t <sub>137</sub>	$\overline{KAS}$ negated to KDATA tri-state (Note 2)	0	31	0	25	34	ns
t <sub>138</sub>	KDATA invalid from end of S5 (Note 2)	3	29	3	24	31	ns
t <sub>139</sub>	KDATA released from end of S5 (Note 2)	4	29	4	24	32	ns
t <sub>144</sub>	$\overline{VMEOUT}$ setup to KCLK	10	–	8	–	–	ns
t <sub>145</sub>	$\overline{VMEOUT}$ hold from $\overline{KAS}$	10	–	8	–	–	ns
t <sub>147</sub>	$\overline{KRMC}$ asynchronous setup to KCLK	0	–	0	–	–	ns
t <sub>148</sub>	$\overline{KRMC}$ hold from $\overline{KDSACKx}$	0	–	0	–	–	ns
t <sub>150</sub>	$\overline{TICK}$ timer cleared, reset from start of S3	6	20	6	16	22	ns
t <sub>152</sub>	$\overline{LRST}$ asserted from start of S3 (SWRST bit)	10	31	10	25	34	ns
t <sub>153</sub>	BIMODE asserted from start of S3 (SBI)	8	26	8	21	29	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

**Table B.1 : SCV64 AC Characteristics** (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>154</sub>	SYSRST* asserted from start of S3 (SWRST bit)	9	27	9	22	30	ns
t <sub>157</sub>	$\overline{\text{LRST}}$ negated from KCLK	5	16	5	13	18	ns
t <sub>159</sub>	IRQ1* asserted from start of S3 (ABI bit)	8	25	8	21	27	ns
t <sub>160</sub>	IRQ1* negated from start of S3 (ABI bit)	10	32	10	26	35	ns
t <sub>161</sub>	$\overline{\text{VMEINT}}$ asserted from KCLK (CERR bit)	8	26	8	21	28	ns
t <sub>162</sub>	$\overline{\text{VMEINT}}$ negated from KCLK (CERR bit)	7	23	7	19	25	ns
t <sub>163</sub>	$\overline{\text{VMEINT}}$ negated from KCLK (clearing DMA DO)	5	18	5	15	19	ns
t <sub>164</sub>	SYSFAIL asserted from KCLK (SYSFAIL bit)	6	21	6	17	22	ns
t <sub>165</sub>	$\overline{\text{SYSFLED}}$ asserted from KCLK	4	15	4	12	17	ns
t <sub>166</sub>	SYSFAIL negated from KCLK	8	26	8	21	28	ns
t <sub>167</sub>	$\overline{\text{SYSFLED}}$ negated from KCLK	5	16	5	13	18	ns
t <sub>168</sub>	SYSFAIL asserted from KCLK (SWRST bit)	14	42	14	34	46	ns
t <sub>169</sub>	$\overline{\text{SYSFLED}}$ asserted from start of S3 (SWRST bit)	12	36	12	29	40	ns
t <sub>170</sub>	DTACK* to KDATA driven	46	73	46	68	77	ns
t <sub>171</sub>	DTACK* to KDATA valid	46	73	46	68	78	ns
t <sub>172</sub>	DTACK* to $\overline{\text{KDSACKx}}$ asserted	47	71	47	67	75	ns
t <sub>175</sub>	$\overline{\text{KBERR}}$ asserted from BERR* asserted	11	35	11	28	39	ns
t <sub>180</sub>	KDATA asynchronous setup to KCLK (coupled write)	0	-	0	-	-	ns
t <sub>181</sub>	KDATA hold from $\overline{\text{KDSACKx}}$ (coupled write)	0	-	0	-	-	ns
t <sub>200</sub>	$\overline{\text{RAMSEL}}$ asserted from KCLK	5	19	5	14	21	ns
t <sub>201</sub>	$\overline{\text{RAMSEL}}$ negated from KCLK	5	17	5	12	19	ns
t <sub>202</sub>	$\overline{\text{VSBSEL}}$ asserted from $\overline{\text{VMEOUT}}$ asserted, KADDR valid	5	18	5	15	19	ns
t <sub>203</sub>	$\overline{\text{VSBSEL}}$ negated from $\overline{\text{VMEOUT}}$ negated, or KADDR invalid	4	15	4	12	16	ns
t <sub>204</sub>	$\overline{\text{KWR}}$ driven from KCLK	5	20	5	14	21	ns
t <sub>205</sub>	$\overline{\text{KWR}}$ tri-state from KCLK	10	33	10	24	37	ns
t <sub>206</sub>	$\overline{\text{RAMSEL}}$ negated from $\overline{\text{KAS}}$ negated	5	16	5	13	17	ns
t <sub>210</sub>	KCLK to KADDR, KFC, KSIZE driven	5	23	5	19	26	ns
t <sub>211</sub>	KCLK to address valid	8	29	8	21	31	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>212</sub>	KCLK to address invalid	7	29	7	20	30	ns
t <sub>213</sub>	address invalid from $\overline{\text{KAS}}$ negated	16	27	16	22	28	ns
t <sub>214</sub>	KADDR, KFC, KSIZE tri-state from KCLK	8	25	8	20	28	ns
t <sub>215</sub>	address invalid from KCLK	0	–	0	–	–	ns
t <sub>218</sub>	$\overline{\text{KRCM}}$ asserted from KCLK	6	21	6	17	22	ns
t <sub>219</sub>	$\overline{\text{KRCM}}$ negated from KCLK	0	–	0	–	–	ns
t <sub>220</sub>	KDATA driven from start of S2	6	26	6	21	28	ns
t <sub>221</sub>	KDATA valid from start of S2	6	26	6	21	29	ns
t <sub>222</sub>	KDATA invalid from end of S5	8	27	8	22	30	ns
t <sub>223</sub>	KDATA released from end of S5	8	28	8	23	30	ns
t <sub>225</sub>	$\overline{\text{KAS}}$ negated to KDATA released	17	26	17	21	27	ns
t <sub>230</sub>	$\overline{\text{KAS}}$ asserted from KCLK	6	19	6	14	21	ns
t <sub>231</sub>	$\overline{\text{KAS}}$ negated from KCLK	5	18	5	13	19	ns
t <sub>232</sub>	$\overline{\text{KAS}}$ tri-state from KCLK	10	33	10	27	36	ns
t <sub>233</sub>	$\overline{\text{KAS}}$ , $\overline{\text{KDS}}$ tri-state from KCLK	4	33	4	27	36	ns
t <sub>235</sub>	$\overline{\text{KDS}}$ asserted from KCLK	4	14	4	10	15	ns
t <sub>236</sub>	$\overline{\text{KDS}}$ negated from KCLK	4	14	4	10	15	ns
t <sub>240</sub>	$\overline{\text{KDSACK}}/\overline{\text{KBERR}}$ setup KCLK (note 3)	5	–	2	–	–	ns
t <sub>241</sub>	$\overline{\text{KDSACK}}/\overline{\text{KBERR}}$ hold from KCLK (note 3)	5	–	2	–	–	ns
t <sub>242</sub>	$\overline{\text{KDSACK}}$ hold from $\overline{\text{KAS}}$ negated	5	–	4	–	–	ns
t <sub>245</sub>	$\overline{\text{KDSACK}}$ asserted to $\overline{\text{KBERR}}$ asserted	5	–	4	–	–	ns
t <sub>248</sub>	Late $\overline{\text{KBERR}}$ setup to KCLK (note 12)	5	–	4	–	–	ns
t <sub>249</sub>	Late $\overline{\text{KBERR}}$ hold from KCLK (note 12)	5	–	4	–	–	ns
t <sub>250</sub>	KDATA driven from start of S2	6	20	6	14	22	ns
t <sub>251</sub>	KDATA valid from KCLK	8	37	8	30	41	ns
t <sub>252</sub>	KDATA invalid from KCLK	10	36	10	27	39	ns
t <sub>253</sub>	KDATA released from KCLK	9	28	9	21	30	ns
t <sub>260</sub>	$\overline{\text{LBRQ1}}$ setup to KCLK	5	–	4	–	–	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

**Table B.1 : SCV64 AC Characteristics** (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>261</sub>	$\overline{\text{KBRQ}}$ asserted from $\overline{\text{LBRQ1}}$ asserted	4	15	4	12	16	ns
t <sub>263</sub>	$\overline{\text{LBGR1}}$ asserted from KCLK	5	17	5	14	18	ns
t <sub>264</sub>	$\overline{\text{LBGR1}}$ negated from KCLK	5	15	5	12	17	ns
t <sub>265</sub>	$\overline{\text{KBRQ}}$ asserted from KCLK (SCV64 request)	8	26	8	21	28	ns
t <sub>266</sub>	$\overline{\text{KBRQ}}$ negated from KCLK	7	23	7	19	25	ns
t <sub>268</sub>	$\overline{\text{KBGR}}$ setup to KCLK (note 9)	5	–	4	–	–	ns
t <sub>270</sub>	$\overline{\text{KBRQ}}$ asserted from KCLK (SCV64 request - bypassed) (note 30)	5	16	4	13	17	ns
t <sub>271</sub>	$\overline{\text{KBRQ}}$ negated from KCLK (SCV64 request - bypassed) (note 30)	5	16	4	13	17	ns
t <sub>272</sub>	$\overline{\text{KBGR}}$ setup to KCLK (SCV64 request - bypassed) (note 9)	5	–	4	–	–	ns
t <sub>273</sub>	KCLK to local bus driven (note 11)	0	23	0	19	26	ns
t <sub>274</sub>	KCLK to local bus released (note 11)	0	33	0	27	36	ns
t <sub>275</sub>	$\overline{\text{KBGACK}}$ asserted from KCLK (SCV64 request)	6	20	6	16	21	ns
t <sub>276</sub>	$\overline{\text{KBGACK}}$ negated from KCLK (SCV64 request bypassed)	6	20	6	16	21	ns
t <sub>277</sub>	$\overline{\text{KBGACK}}$ setup to KCLK	5	–	4	–	–	ns
t <sub>278</sub>	$\overline{\text{KAS}}$ setup to KCLK	5	–	4	–	–	ns
t <sub>279</sub>	$\overline{\text{KAS}}$ hold from KCLK	5	–	4	–	–	ns
t <sub>301</sub>	$\overline{\text{VAS}}$ , $\overline{\text{VDS}}$ , DTACK*, BERR*, IACKIN* asynchronous setup to C32MHZ (note 16)	0	–	0	–	–	ns
t <sub>302</sub>	IACKOUT* asserted from C32MHZ (note 16)	4	14	4	14	15	ns
t <sub>303</sub>	IACKOUT* negated from $\overline{\text{VAS}}$ negated	3	10	3	10	10	ns
t <sub>304</sub>	IACK daisy chain delay (note 17)	77	108	77	108	109	ns
t <sub>308</sub>	Bus Grant daisy chain delay (assertion) (note 32)	4	13	4	13	15	ns
t <sub>309</sub>	Bus Grant daisy chain delay (negation)	4	14	4	14	15	ns
t <sub>310</sub>	BRn* asserted from C32MHZ	5	18	5	18	20	ns
t <sub>311</sub>	BRn* negated from C32MHZ (note 19)	7	23	7	23	25	ns
t <sub>312</sub>	BGnIN* asynchronous setup to C32MHZ	10	–	10	–	–	ns
t <sub>315</sub>	BBSY* asserted from C32MHZ (note 18)	6	18	6	18	19	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>316</sub>	BBSY* hold from BGnIN* negated	31	–	31	–	–	ns
t <sub>317</sub>	BBSY* negated from C32MHZ (note 20)	8	26	8	26	28	ns
t <sub>319</sub>	BCLR* asynchronous setup to C32MHZ	10	–	10	–	–	ns
t <sub>320</sub>	BRn* asynchronous setup to C32MHZ	10	–	10	–	–	ns
t <sub>321</sub>	BGnOUT* asserted from C32MHZ (note 21)	5	18	5	18	18	ns
t <sub>322</sub>	BGnOUT* negated from C32MHZ (note 22)	5	17	5	17	19	ns
t <sub>324</sub>	BBSY* asynchronous setup to C32MHZ	10	–	10	–	–	ns
t <sub>325</sub>	BBSY* minimum assertion time (note 1)	73	–	73	–	–	ns
t <sub>328</sub>	BCLR* asserted from C32MHZ (note 23)	4	17	4	17	18	ns
t <sub>329</sub>	BCLR* negated from C32MHZ (note 24)	4	13	4	13	14	ns
t <sub>350</sub>	IRQn* asserted from end of State 5	5	20	5	20	21	ns
t <sub>351</sub>	IRQn* negated from C32MHZ (note 27)	9	31	9	31	31	ns
t <sub>360</sub>	VADDOUT high from KCLK	6	18	6	18	20	ns
t <sub>361</sub>	VSTRBOUT high from KCLK	6	18	6	18	20	ns
t <sub>362</sub>	VDATOUT high from KCLK	4	15	4	15	16	ns
t <sub>363</sub>	KCLK to VDATOUT high (A64 only)	7	23	7	23	25	ns
t <sub>364</sub>	VADDR, VAM, $\overline{\text{VLWORD}}$ released from VADDOUT low	0	3	0	3	3	ns
t <sub>365</sub>	VSTRBOUT low from $\overline{\text{VAS}}$ negated	6	18	6	18	20	ns
t <sub>366</sub>	DTACK* negated to VASOUT low (MBLT)	11	33	11	33	36	ns
t <sub>367</sub>	VDATA released from VDATOUT low	0	3	0	3	3	ns
t <sub>369</sub>	VADDOUT low from $\overline{\text{VDSb}}$	7	22	7	22	24	ns
t <sub>404</sub>	VADDR, VAM, $\overline{\text{VLWORD}}$ setup to $\overline{\text{VAS}}$	10	–	10	–	–	ns
t <sub>405</sub>	$\overline{\text{VAS}}$ minimum high time	30	–	30	–	–	ns
t <sub>408</sub>	data setup to $\overline{\text{VDSa}}$ asserted (VADDR and VDATA)	10	–	10	–	–	ns
t <sub>410</sub>	$\overline{\text{VAS}}$ to $\overline{\text{VDS}}$ skew	-10	–	-10	–	–	ns
t <sub>411</sub>	$\overline{\text{VDS}}$ minimum high time	30	–	30	–	–	ns
t <sub>412</sub>	$\overline{\text{VWR}}$ setup to $\overline{\text{VDSa}}$ asserted	10	–	10	–	–	ns
t <sub>413</sub>	$\overline{\text{VDS}}$ skew	–	20	–	20	20	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value



**Table B.1 : SCV64 AC Characteristics** (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>414</sub>	VADDR, $\overline{\text{VLWORD}}$ hold from DTACK* asserted	0	–	0	–	–	ns
t <sub>416</sub>	VAM hold from DTACK* asserted	0	–	0	–	–	ns
t <sub>418</sub>	$\overline{\text{VAS}}$ hold from DTACK* asserted	0	–	0	–	–	ns
t <sub>420</sub>	$\overline{\text{VDSa}}$ hold from DTACK* asserted	0	–	0	–	–	ns
t <sub>421</sub>	$\overline{\text{VDSb}}$ hold from DTACK* asserted	0	–	0	–	–	ns
t <sub>422</sub>	data hold from DTACK* asserted (VADDR & VDATA)	0	–	0	–	–	ns
t <sub>423</sub>	$\overline{\text{VWR}}$ hold from $\overline{\text{VDS}}$ negated	0	–	0	–	–	ns
t <sub>428</sub>	SCV64 slave response (note 28)	38	59	38	59	65	ns
t <sub>429</sub>	VDATA invalid from $\overline{\text{VDSb}}$ negated	6	25	6	25	26	ns
t <sub>430</sub>	DTACK* negated from $\overline{\text{VDSb}}$ negated	8	27	8	27	29	ns
t <sub>431</sub>	VDATA invalid TO DTACK* negated	32	37	32	37	39	ns
t <sub>432</sub>	VADDR, VAM, VLWORD* setup to $\overline{\text{VDS}}$	10	–	10	–	–	ns
t <sub>450</sub>	VDATA valid from KCLK rising (note 25)	5	16	5	16	17	ns
t <sub>451</sub>	DTACK* asserted from KCLK (note 26)	4	16	4	16	17	ns
t <sub>458</sub>	VADDR setup to $\overline{\text{VDSa}}$ asserted	-10	–	-10	–	–	ns
t <sub>480</sub>	DTACK* negated from $\overline{\text{VDSb}}$ negated	40	61	40	61	64	ns
t <sub>481</sub>	BERR* released from $\overline{\text{VDSb}}$ negated	6	20	6	20	22	ns
t <sub>490</sub>	VDATA driven from KCLK	5	21	5	21	23	ns
t <sub>491</sub>	VDATA (and VADDR for MBLT) valid from start of S5	6	23	6	23	26	ns
t <sub>493</sub>	DTACK*, BERR* asserted from end of S5 (note 28)	7	23	7	23	25	ns
t <sub>504</sub>	VADDR, $\overline{\text{VLWORD}}$ , VAM valid to $\overline{\text{VAS}}$ asserted	26	37	26	37	39	ns
t <sub>505</sub>	$\overline{\text{VAS}}$ minimum high time	39	50	39	50	53	ns
t <sub>508</sub>	VDATA setup to $\overline{\text{VDSa}}$ asserted	33	35	33	35	36	ns
t <sub>509</sub>	DTACK* negated to next $\overline{\text{VDS}}$ asserted (writes)	7	23	7	23	25	ns
t <sub>510</sub>	$\overline{\text{VAS}}$ to $\overline{\text{VDS}}$ skew	–	6	–	6	6	ns
t <sub>511</sub>	$\overline{\text{VDS}}$ minimum high time	49	67	49	67	72	ns
t <sub>513</sub>	$\overline{\text{VDS}}$ skew	0	1	0	1	1	ns
t <sub>514</sub>	VADDR, VLWORD* invalid from DTACK* asserted (writes)	11	42	11	42	45	ns
t <sub>516</sub>	IACK* negated from DTACK* asserted	53	89	53	89	95	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

Table B.1 : SCV64 AC Characteristics (Continued)

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>518</sub>	DTACK* asserted to $\overline{\text{VAS}}$ negated (writes)	7	23	7	23	25	ns
t <sub>519</sub>	$\overline{\text{VAS}}$ minimum assertion time (writes)	44	59	44	59	65	ns
t <sub>520</sub>	$\overline{\text{VDSa}}$ negated from DTACK* asserted (writes)	7	21	7	21	23	ns
t <sub>521</sub>	$\overline{\text{VDSb}}$ negated from DTACK* asserted (writes)	7	23	7	23	25	ns
t <sub>522</sub>	data invalid from DTACK* asserted	11	36	11	36	39	ns
t <sub>523</sub>	$\overline{\text{VWR}}$ hold from $\overline{\text{VDS}}$ negated	6	21	6	21	22	ns
t <sub>527</sub>	VDATA setup to DTACK* asserted	-25	–	-25	–	–	ns
t <sub>528</sub>	minimum slave response time	30	–	30	–	–	ns
t <sub>529</sub>	VDATA hold from $\overline{\text{VDSa}}$ negated	0	–	0	–	–	ns
t <sub>530</sub>	DTACK* hold from $\overline{\text{VDS}}$ negated	0	–	0	–	–	ns
t <sub>533</sub>	$\overline{\text{VDS}}$ low time	42	59	42	59	63	ns
t <sub>550</sub>	VADDR driven from KCLK	5	16	5	16	17	ns
t <sub>552</sub>	$\overline{\text{VWR}}$ driven from KCLK	4	20	4	20	22	ns
t <sub>553</sub>	KCLK to IACK* asserted	10	32	10	32	35	ns
t <sub>555</sub>	first $\overline{\text{VAS}}$ asserted from KCLK	31	62	31	62	67	ns
t <sub>557</sub>	VDATA driven from KCLK	7	27	7	27	30	ns
t <sub>559</sub>	DTACK* negated to next $\overline{\text{VDS}}$ asserted (reads)	47	68	47	68	69	ns
t <sub>560</sub>	$\overline{\text{VAS}}$ negated from DTACK* asserted (last cycle)	17	51	17	51	55	ns
t <sub>564</sub>	address invalid from DTACK* asserted (reads)	50	83	50	83	89	ns
t <sub>567</sub>	$\overline{\text{VAS}}$ negated from DTACK* asserted	54	92	54	92	97	ns
t <sub>568</sub>	$\overline{\text{VAS}}$ negated from DTACK* asserted (reads)	44	64	44	64	67	ns
t <sub>569</sub>	$\overline{\text{VAS}}$ minimum low time (reads)	82	94	82	94	102	ns
t <sub>570</sub>	$\overline{\text{VDSa}}$ negated from DTACK* asserted (reads)	44	62	44	62	66	ns
t <sub>571</sub>	$\overline{\text{VDSb}}$ negated from DTACK* asserted (reads)	44	64	44	64	67	ns
t <sub>572</sub>	VDATA tri-state from DTACK* asserted	14	43	14	43	47	ns
t <sub>574</sub>	VADDR, VAM, $\overline{\text{VLWORD}}$ released from DTACK* asserted	14	44	14	44	48	ns
t <sub>580</sub>	$\overline{\text{VDSa}}$ negated from BERR* asserted	8	28	8	28	31	ns
t <sub>581</sub>	$\overline{\text{VDSb}}$ negated from BERR* asserted	9	28	9	28	31	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

**Table B.1 : SCV64 AC Characteristics (Continued)**

Timing Parameter	Description	33 MHz (All variants)		40 MHz (Com)		25 MHz (Ext)	Units
		Min	Max	Min	Max	Max	
t <sub>584</sub>	address invalid from BERR* asserted	15	47	15	47	51	ns
t <sub>588</sub>	$\overline{\text{V}}\text{AS}$ negated from BERR* asserted	18	57	18	57	62	ns
t <sub>590</sub>	$\overline{\text{V}}\text{RMC}$ asserted from KCLK	11	34	11	34	37	ns
t <sub>594</sub>	VADDR / VDATA invalid/hi-z from DTACK* asserted (BLT)	45	65	45	65	68	ns
t <sub>597</sub>	$\overline{\text{V}}\text{AS}$ negated from DTACK* asserted (for last read cycle)	54	93	54	93	98	ns
t <sub>598</sub>	Setup of $\overline{\text{K}}\text{BGR}$ to KCLK (note 33)	-2	—	-2	—	—	ns

\* Use the minimum value in the 33 MHz column for the 25 MHz Extended minimum timing value

*Notes:*

- These parameters are based upon a C32MHZ frequency of 32 MHz.
- Use the worse case of t<sub>136</sub>/t<sub>138</sub> and t<sub>137</sub>/t<sub>139</sub> for register access.
- These are synchronous acknowledgments for burst cycles. For all other cycles,  $\overline{\text{KDSACK}}$  are double sampled by the falling edge of KCLK.
- When using the  $\overline{\text{K}}\text{IACK}$  power-up option.
- When using the interrupt acknowledge decode power-up option,  $\overline{\text{L}}\text{IACKn}$ ,  $\overline{\text{K}}\text{AVEC}$  are negated on the first of  $\overline{\text{K}}\text{AS}$  negated or  $\overline{\text{L}}\text{IRQn}$  negated.
- $\overline{\text{L}}\text{IACKn}$ ,  $\overline{\text{K}}\text{AVEC}$  are negated on the first of  $\overline{\text{K}}\text{IACK}$  negated,  $\overline{\text{K}}\text{DS}$  negated,  $\overline{\text{L}}\text{IRQn}$  negated.
- This signal is double sampled by rising edge of C32MHZ.
- This includes 40 ns internal delay line.
- Signal is double sampled by KCLK.
- t<sub>10</sub> + t<sub>11</sub> = 31.25 ns ± 0.50 ns.
- In arbiter bypass mode, the local bus is driven after  $\overline{\text{K}}\text{BGR}$  has been sampled low on consecutive falling then rising edges of KCLK, and released on the rising edge after  $\overline{\text{K}}\text{BRQ}$  and  $\overline{\text{K}}\text{AS}$  are negated. In internal arbiter mode, it is driven on the rising edge after  $\overline{\text{K}}\text{BRQ}$  is released.
- Late  $\overline{\text{K}}\text{BERR}$  is setup and held from the rising edge of KCLK two clocks after a cycle completes.
- $\overline{\text{L}}\text{RST}$  goes high two falling edges of KCLK after SYSRST\* is negated, and 224 to 238 ms after  $\overline{\text{P}}\text{WRRST}$  and  $\overline{\text{E}}\text{XTRST}$  are negated, and 224 to 238 ms after a software reset.
- Due to SYSRST\* asserted externally or SYSRST\* asserted by  $\overline{\text{P}}\text{WRRST}$ , software reset, or BG0IN\* (as SYSCON).
- SYSRST\* is negated 224 to 238 ms after BG0IN\* or  $\overline{\text{P}}\text{WRRST}$  are released or a software reset.
- IACKOUT\* is asserted on the third rising edge of C32MHZ after:
  - IACKIN\* is low,
  - $\overline{\text{V}}\text{AS}$  is low,
  - either  $\overline{\text{V}}\text{DS0}$  or  $\overline{\text{V}}\text{DS1}$  is low, and
  - DTACK\* and BERR\* are high.
- This is derived from parameters t<sub>301</sub> and t<sub>302</sub> and a 32 MHZ C32MHZ frequency.
- BBSY\* is asserted on the first edge after BGnIN\* is sampled low twice by C32MHZ.
- BRn\* is negated on the second edge after BGnIN\* is sampled low twice by C32MHZ.

20. BBSY\* is negated once BCLR\* is sampled low twice by C32MHZ.
21. BGnOUT\* is asserted on first edge after BRn\* has been sampled low on two consecutive edges and BBSY\* has been sampled high on two consecutive edges of C32MHZ.
22. BGnOUT\* is negated on the first edge of C32MHZ after BBSY\* has been sampled low on two consecutive edges of C32MHZ.
23. BCLR\* is asserted once a higher level request has been sampled low twice.
24. BCLR\* is negated once BBSY\* has been sampled high twice.
25. VDATA is driven on the first rising edge of KCLK after the IACK conditions have been true for three consecutive C32MHZ edges.
26. DTACK\* is asserted on the third clock edge after the IACK conditions have been true for 3 consecutive C32MHZ edges.
27. IRQn\* is negated on the next rising edge of C32MHZ after IACKIN\*,  $\overline{VAS}$ , and  $\overline{VDS}$  are double sampled low by C32MHZ, and VADDR03-01 are at the appropriate level, and DTACK\* and BERR\* are high.
28. For the address phase of MBLT cycles, DTACK\* is asserted three KCLKs (rising edges) after  $\overline{VDSa}$  is asserted. DTACK\*/BERR\* are asserted at the end of S5 when BERRCHK=0. During reads, with BERRCHK=1, DTACK\*/BERR\* will be asserted two clock cycles later.
29. Contact the factory for operation at frequencies lower than 20 MHz.
30.  $\overline{KBRQ}$  is negated for a minimum of 3 KCLK periods before the next assertion of  $\overline{KBRQ}$ . If  $\overline{KBRQ}$  is negated during the current bus cycle, it will not be reasserted until at least 2 KCLK cycles after KAS has been negated.
31. T represents one clock period.
32. If the SCV64 is passing a bus grant on the same level that it is requesting on, then the SCV64 double samples the BGnIN\* signal with the C32MHz clock before asserting BGnOUT\*.
33. This parameter need only be implemented if your logic will be exceeding the maximum number of wait states that can be tolerated by the SCV64 during a burst write cycle which resulted from a MBLT transfer on the VMEbus (see "Burst Writes" on page 2 90).
34. This timing parameter allows the  $\overline{KAS}$  signal to precede the  $\overline{SCV64SEC}$  signal by no more than 2 KCLK cycles less 5 ns.

## B.2 Capacitive Loading

The timing parameters in Table B.1 were determined assuming the values for capacitive loading in Table .

**Table B.2 : Capacitive Loading from Output Signals**

Signal Name	Typical Capacitance (pF)
BAUDCLK, C14US, C8MHZ, JTDO, KAVEC, KBGACK, KBRQ, KDS, KIPL2-0, LBGR1, LIACK5, LIACK4, LMINT, SYSFLED, TICK, VADDOUT, VADDR31-1, VAM5-0, VAS, VASOUT, VDATA31-0, VDATAOUT, VDS1, VDS0, VLWORD, VMEINT, VRETRY, VWR, WDOG	30.0
RAMSEL	40.0
BGOUT3-0, IACKOUT,	50.0
BIMODE, KFC2-0, KSIZE1, KSIZE0, VSBSEL	60.0
KADDR31-0, KDATA31-0, KRMC	80.0
KAS, KBERR, KDSACK1, KDSACK0, KHALT, KWR, LRST	130.0
BBSY, BCLR, BERR, BR3-0, DTACK, IACK, IRQ7-1, SYSCLK, SYSFAIL, SYSRST	200.0

## B.3 Timing Diagrams

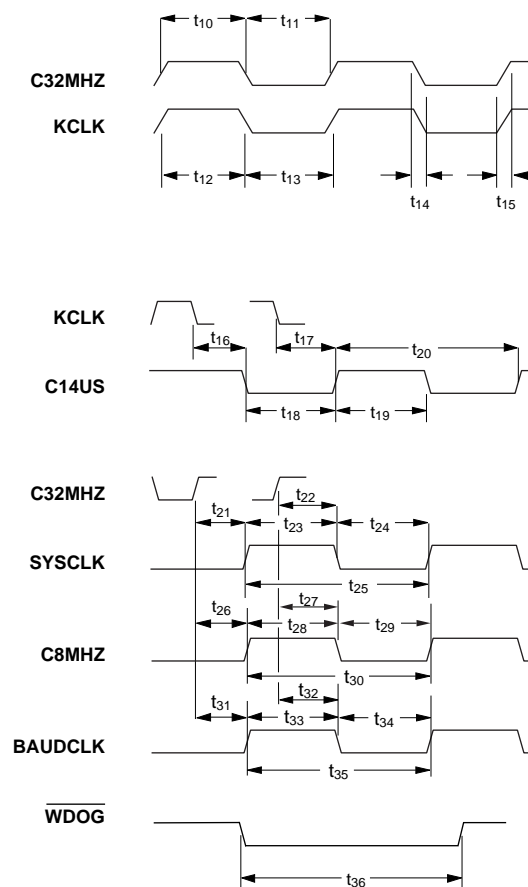
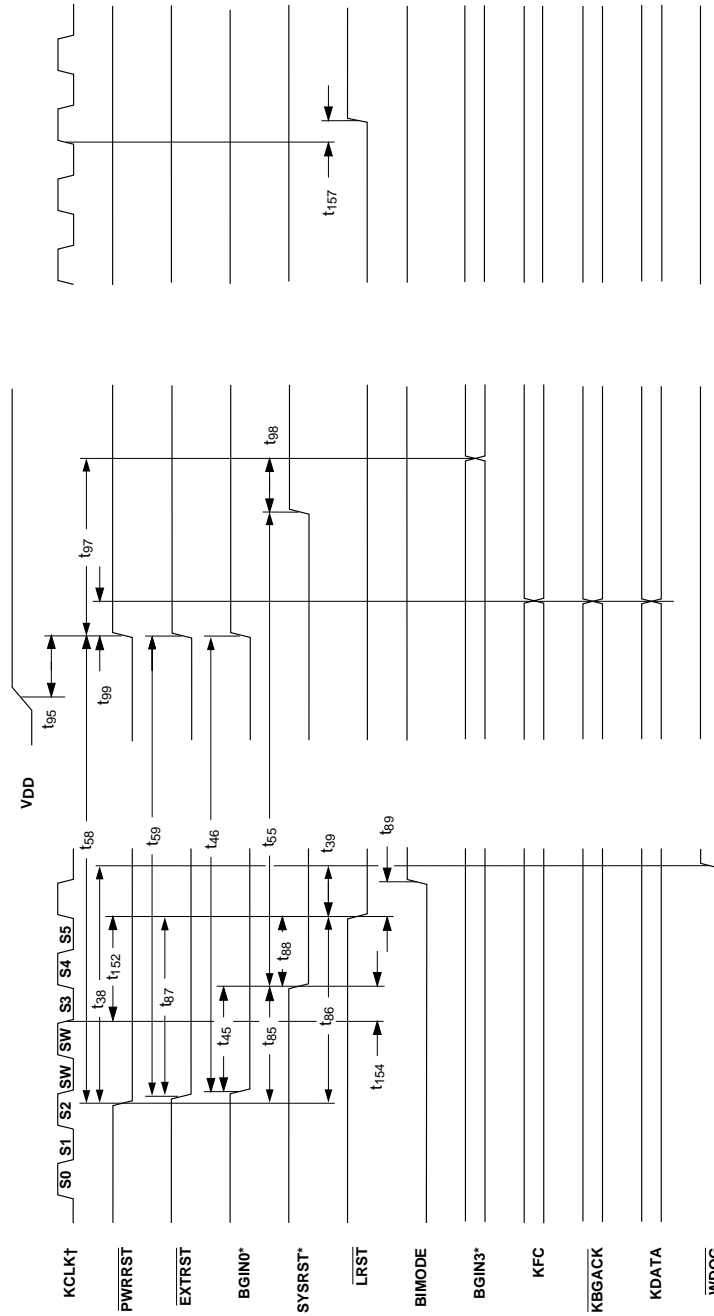
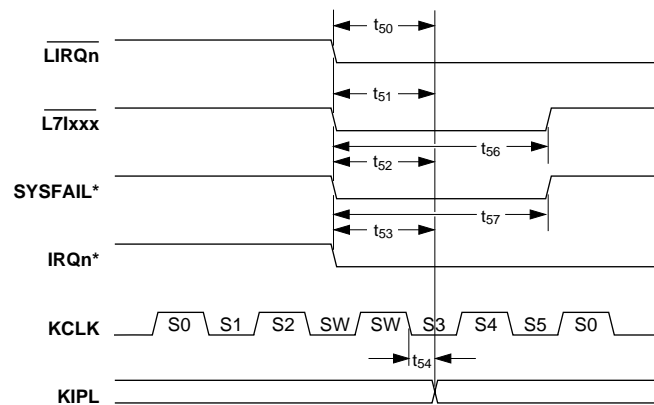


Figure B.1 : Clocks and Timers

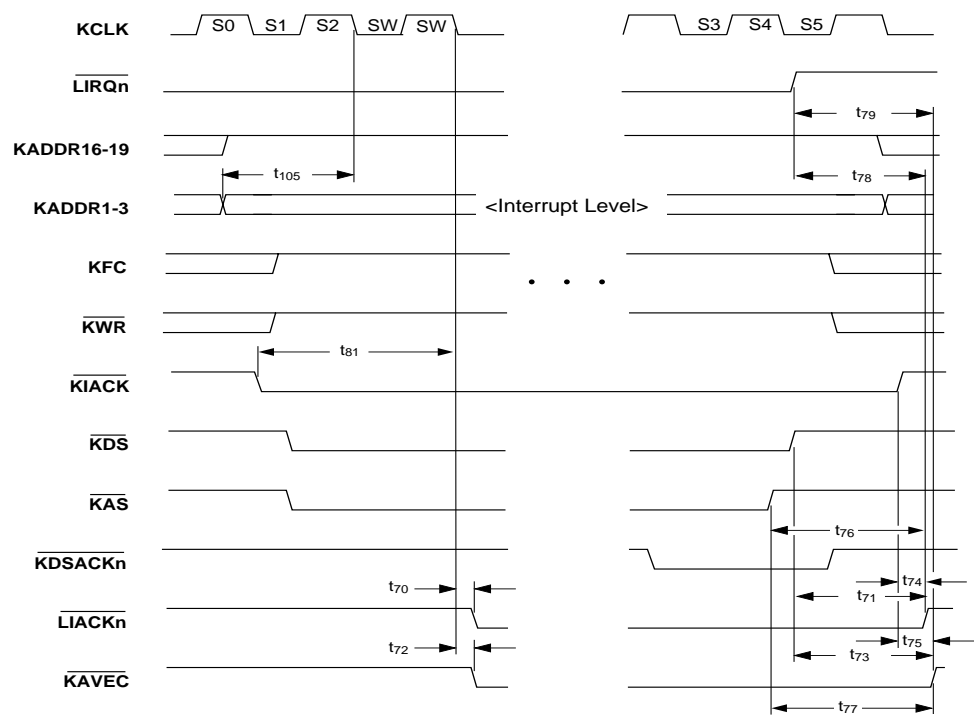


<sup>†</sup>**NOTE:** Shown to reference software reset effects, and LRST negation

### Figure B.2: Reset Timing

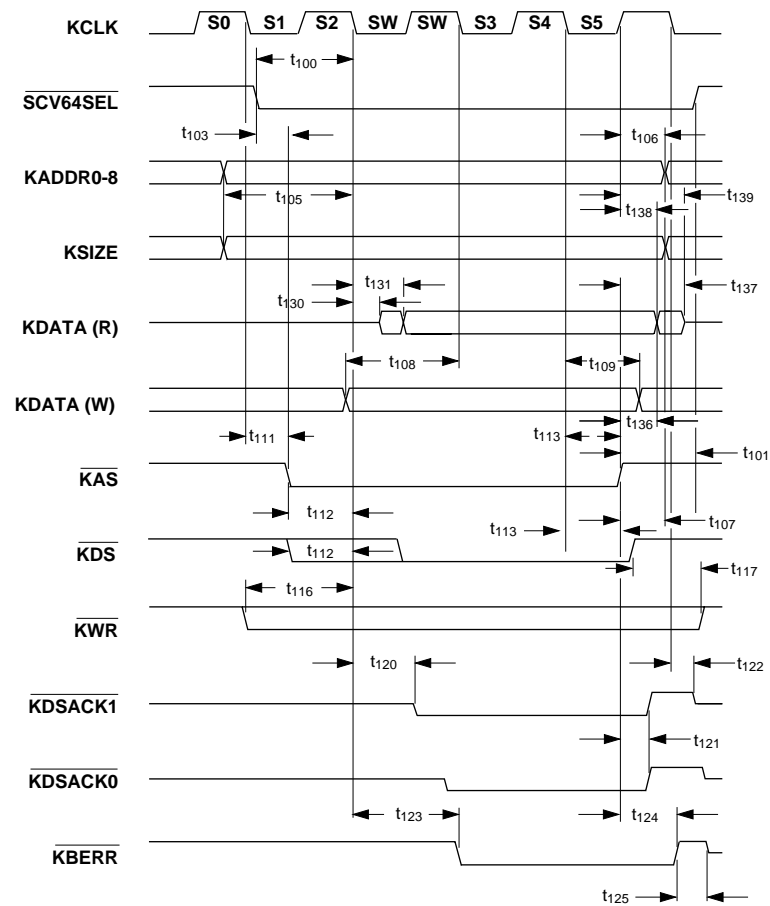
**Figure B.3 : Local Interrupts - Generation**





<sup>†</sup>NOTE: The timing for KDSACK when obtaining the vector from the VMEbus can be obtained from Figure B.10.

Figure B.4 : Local Interrupts-Acknowledgment



<sup>†</sup>NOTE: The KFC signals are not decoded by the SCV64 during register accesses

**Figure B.5 : Register Access**

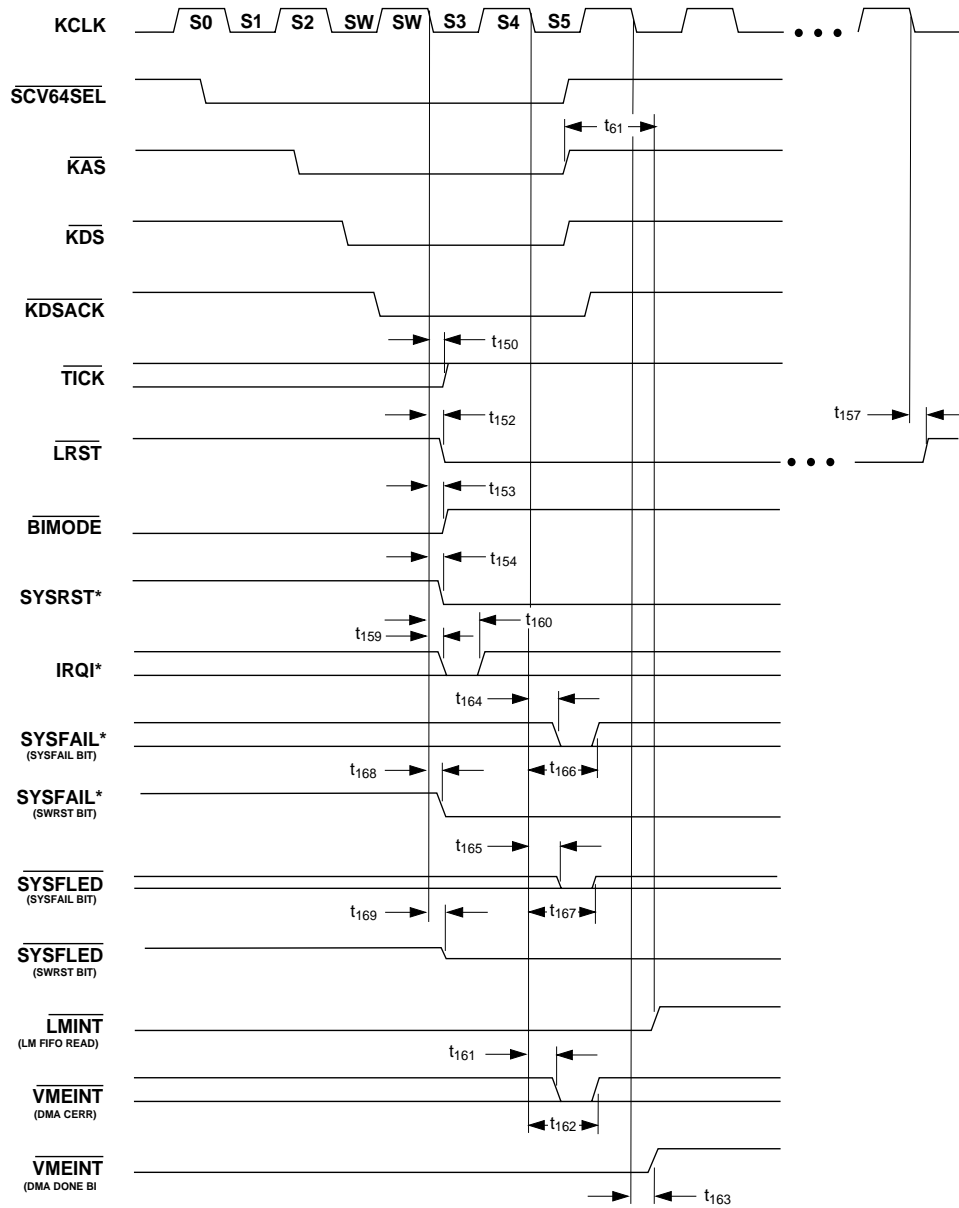


Figure B.6 : Register Access Effects

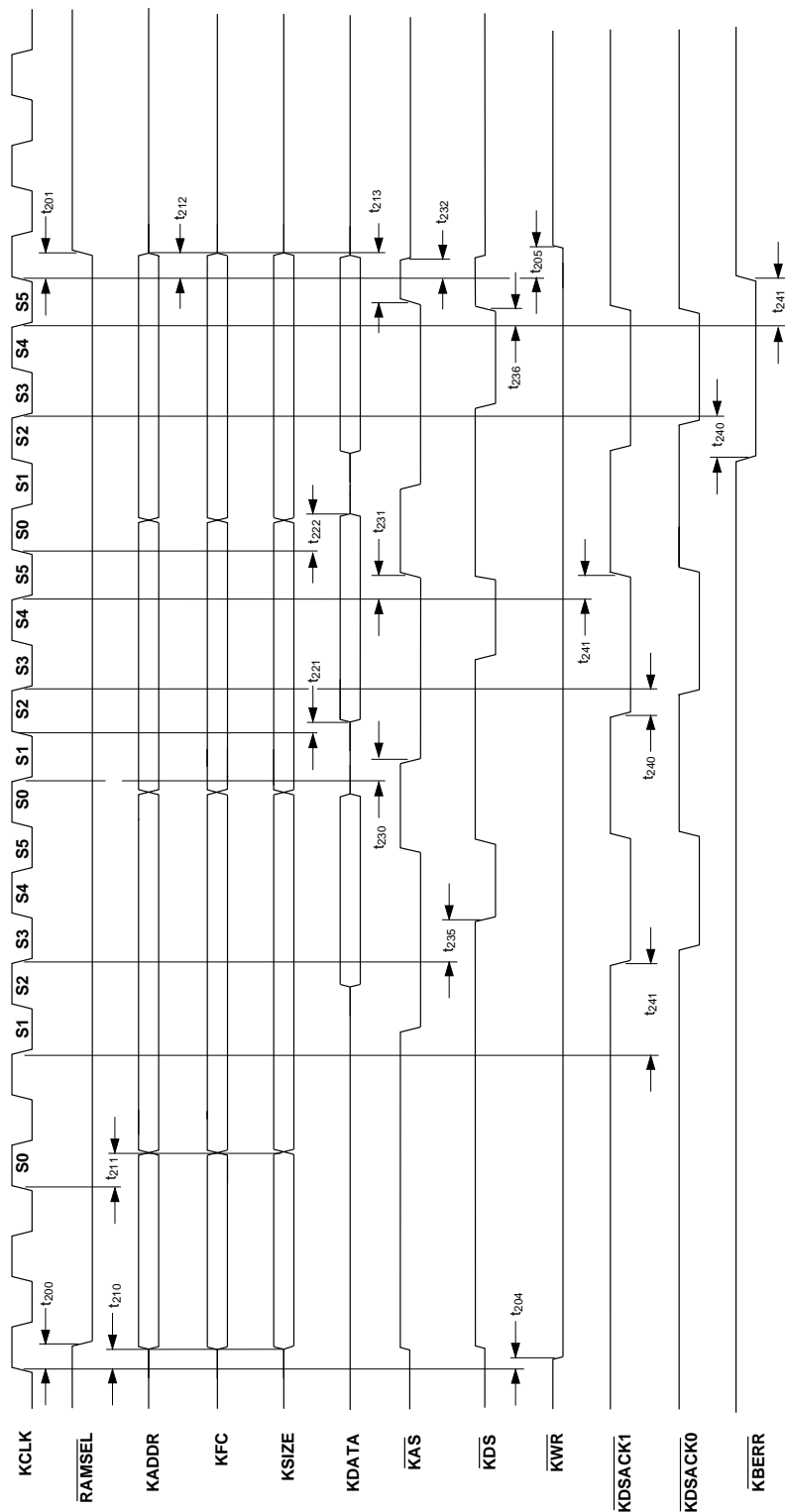
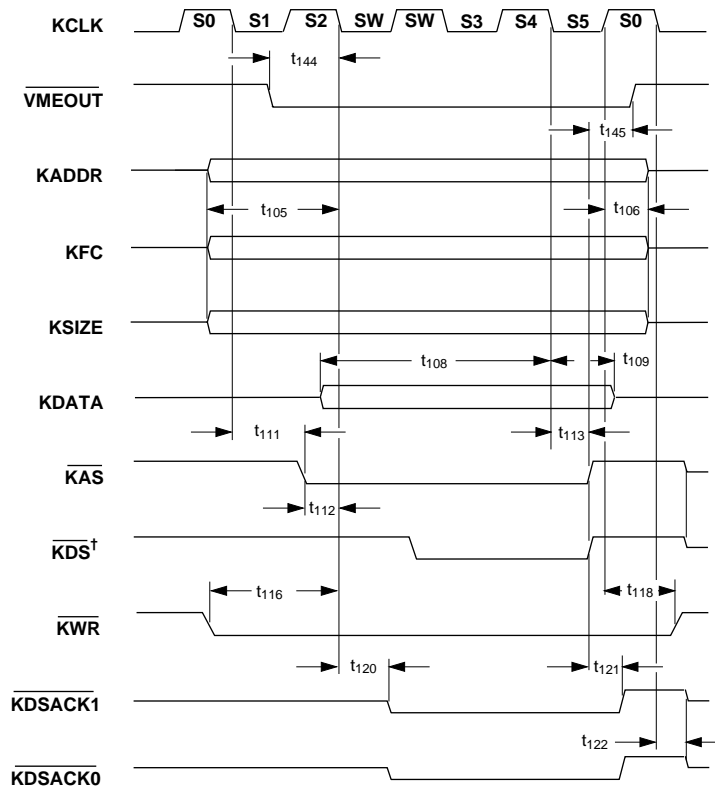


Figure B.7 : Decoupled Write Cycle - SCV64 as Local Master



<sup>†</sup>NOTE: KDS is not monitored by the SCV64 during this operation. It is shown here only for completeness

Figure B.8 : Decoupled Write Cycle - SCV64 as Local Slave

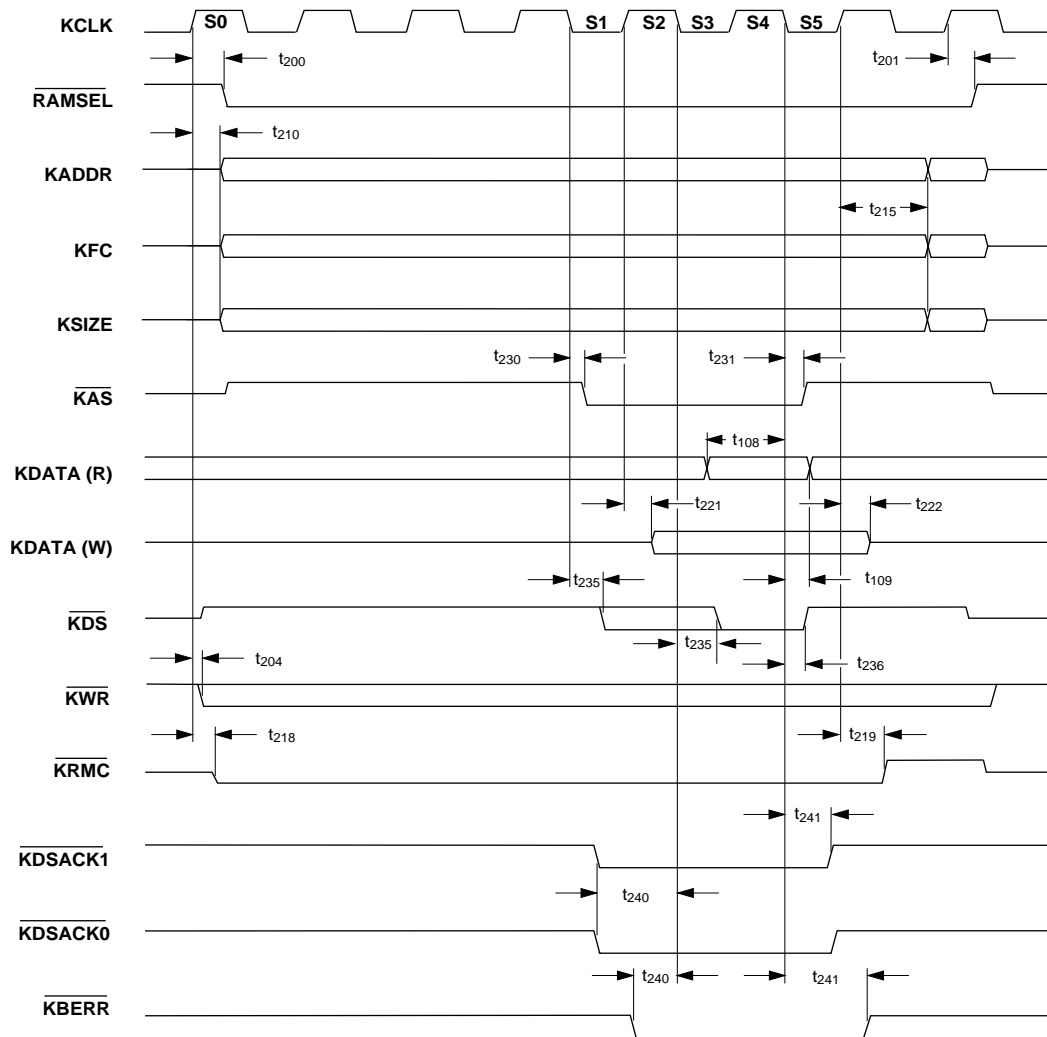
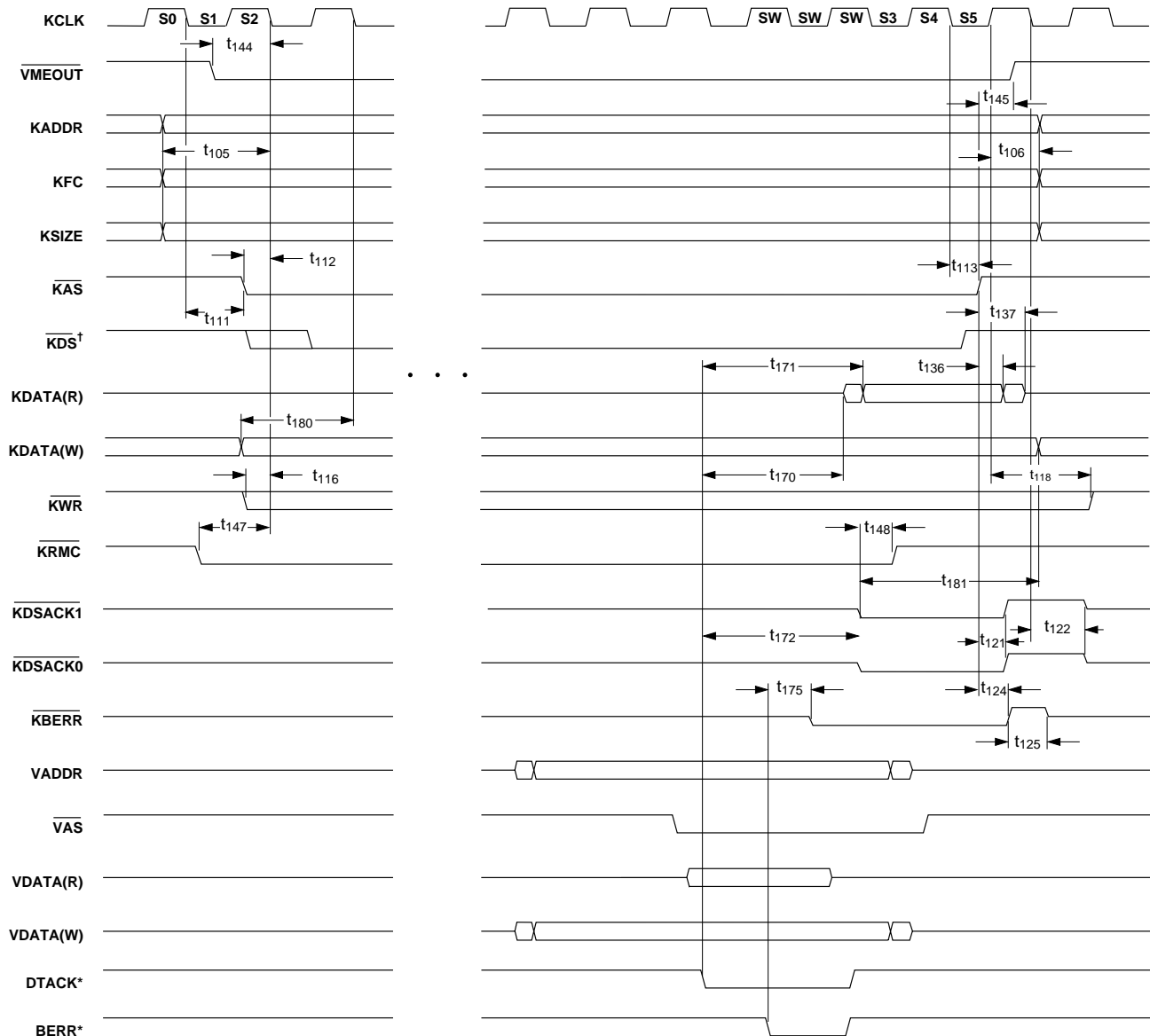


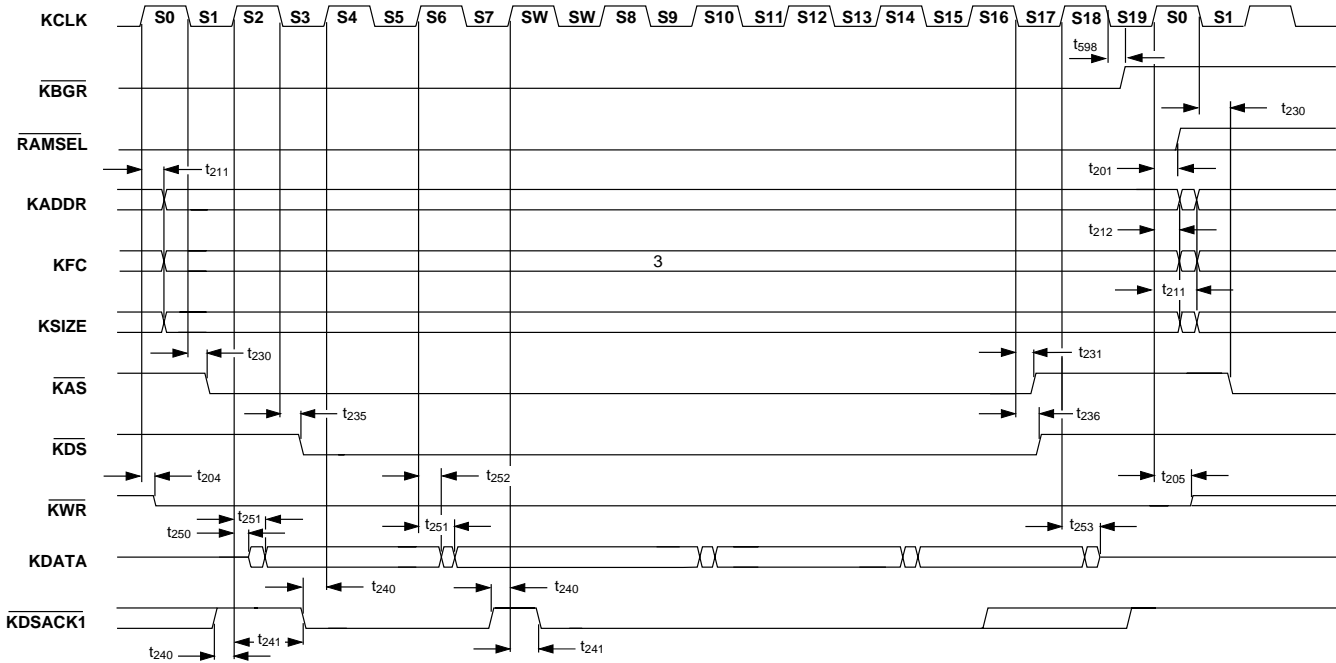
Figure B.9 : Coupled Cycle - SCV64 as Local Master



<sup>†</sup>NOTE: KDS is not monitored by the SCV64 during this operation.  
It is shown here only for completeness

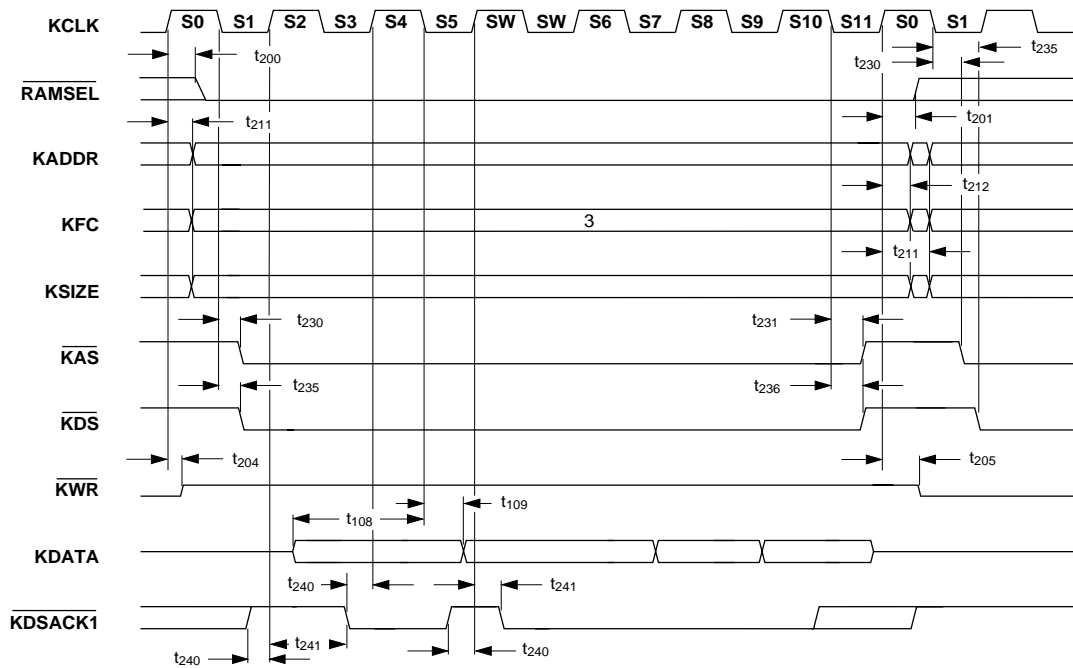
**Figure B.10 : Coupled Cycle - SCV64 as Local Slave**

(see Figure B.23 for VME timing)

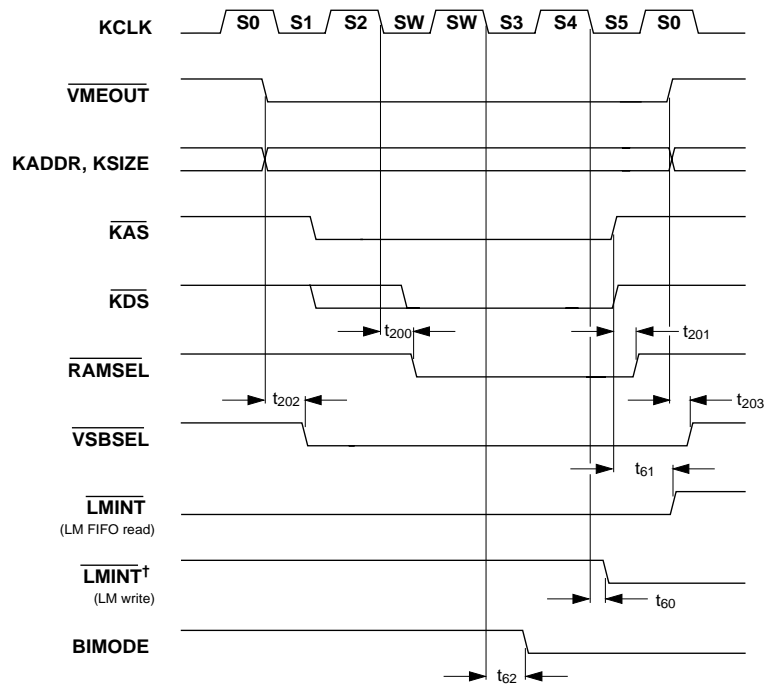


**Figure B.11 : Burst Write Cycle - SCV64 as Local Master**  
(see Figure B.14a and Figure B.14b for full start/stop timing)



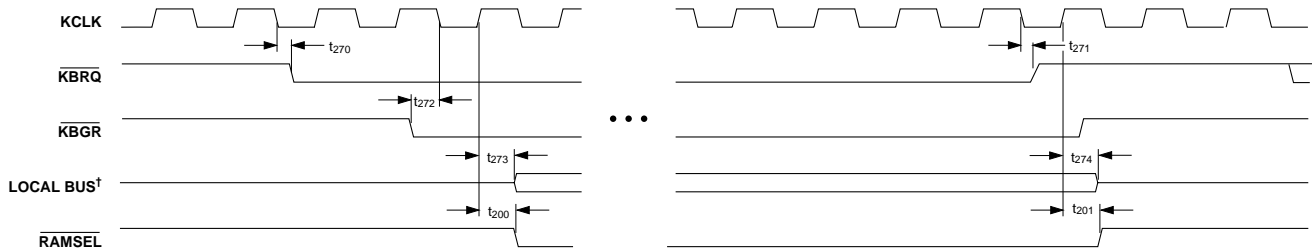


**Figure B.12 : Burst Read Cycle - SCV64 as Local Master**



<sup>†</sup>NOTE: Location monitor writes involve two wait states  
(i.e. one additional CLK)

Figure B.13 : Slave Image/VSB/Location Monitor Access



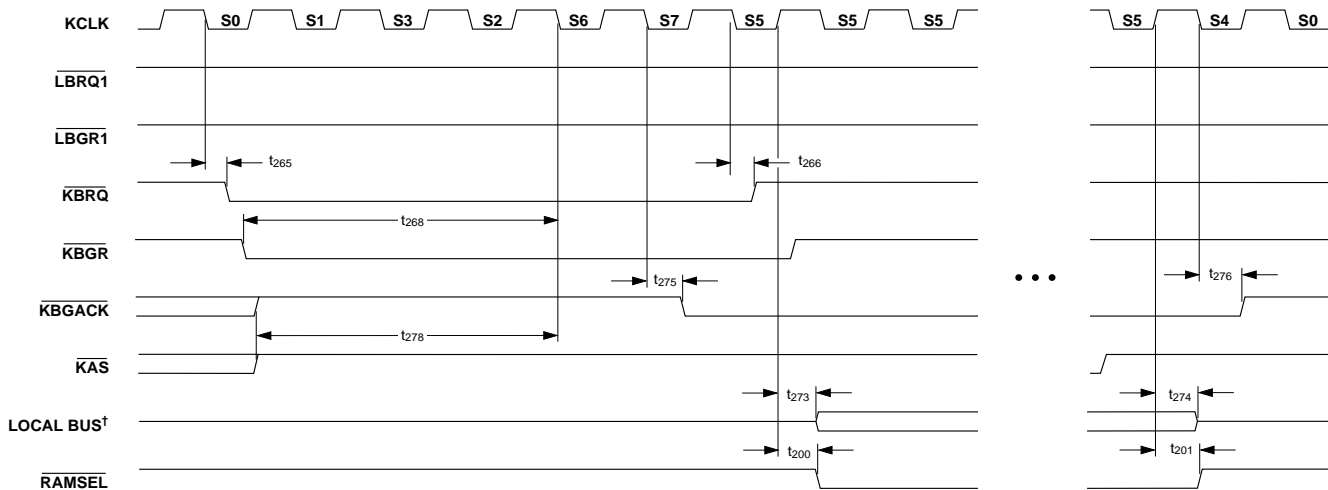
<sup>†</sup>NOTE: Local bus signals include: KADDR, KFC,  $\overline{KSIZ}$ ,  $\overline{KAS}$ ,  $\overline{KDS}$ ,  $\overline{KWR}$ ,  $\overline{KRMC}$  tristated off the rising edge of S5 and KAS, KDS tristated off the falling edge of S4.

**Figure B.14a : Local Requester – Arbiter Bypassed**



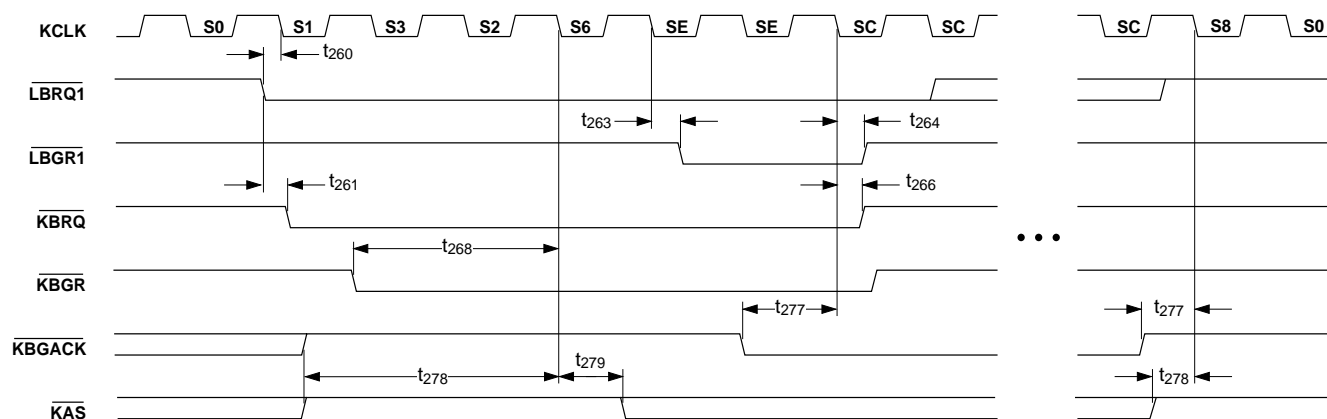
*Caution: The SCV64 may negate its request early. Therefore local arbitration logic should be implemented in a manner which keeps the grant asserted to the SCV64 while either KAS or KBRQ is asserted.*

In order to force the SCV64 off the bus during a decoupled write after only 1 transfer, the (KBGR) must be negated in order to meet a setup time to the falling edge of S2.

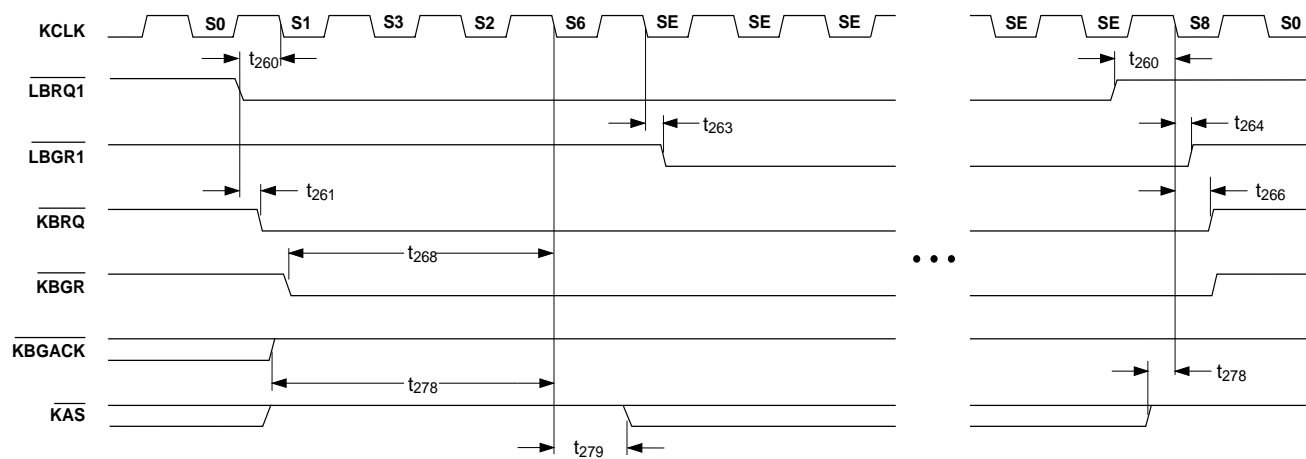


<sup>†</sup>NOTE: Local bus signals include: KADDR, KFC,  $\overline{KSIZ}$ ,  $\overline{KAS}$ ,  $\overline{KDS}$ ,  $\overline{KWR}$ , and  $\overline{KRMC}$

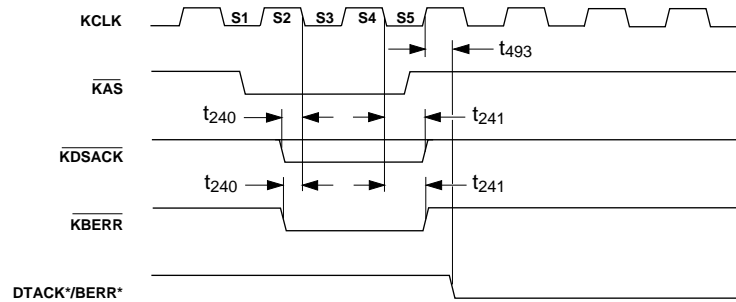
**Figure B.14b : Local Requester/Arbiter: Request by SCV64**



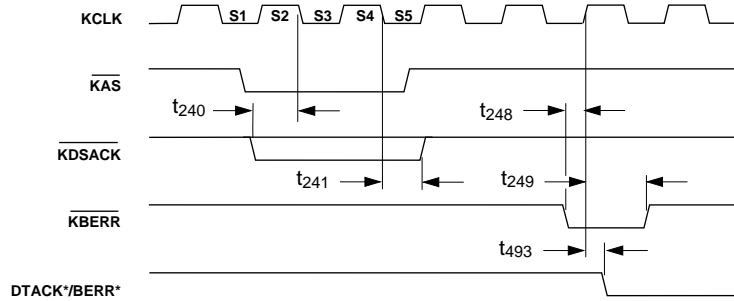
**Figure B.14c : Local Requester/Arbiter: Request by an External Device  
– with Acknowledge**



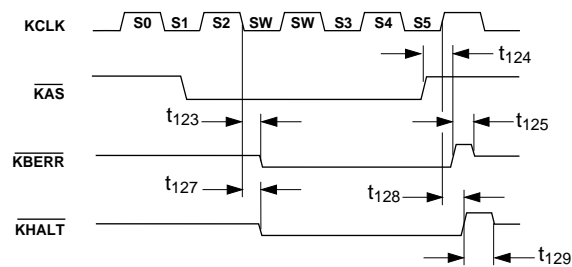
**Figure B.14d : Local Requester/Arbiter: Request by an External Device  
– without Acknowledge**



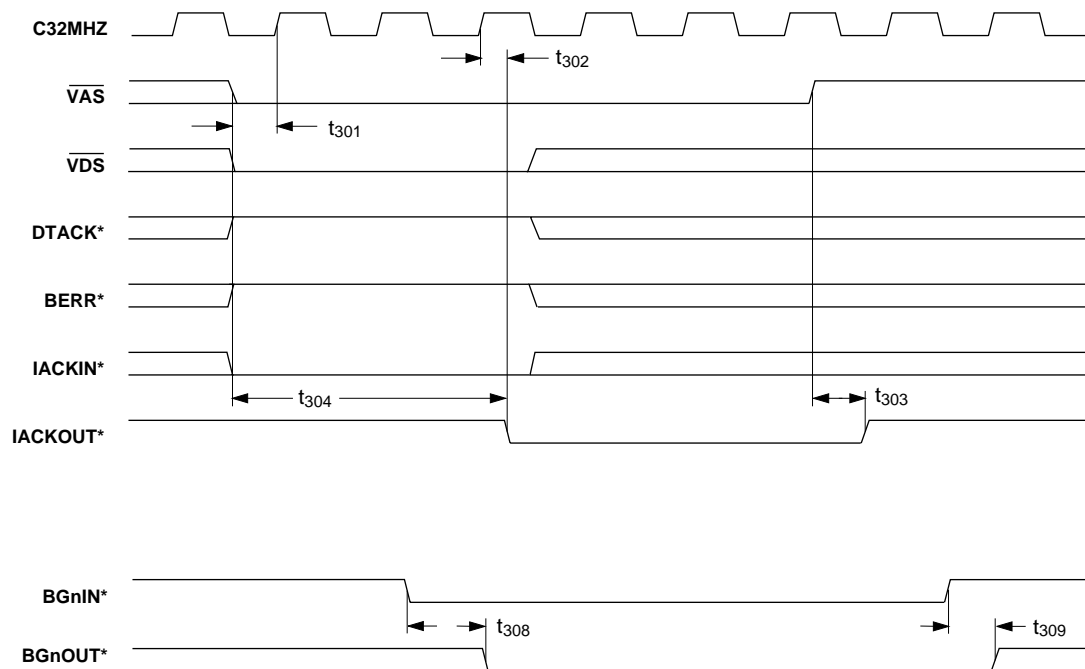
**Figure B.15a : Local Bus Error Termination - Write or Read Cycle with BERRCHK bit=0 (SCV64 as Local Master)**



**Figure B.15b : Local Bus Error Termination - Read Cycle with BERRCHK bit=1 (SCV64 as Local Master)**



**Figure B.15c : Deadlock Retry Termination (SCV64 as Local Slave)**

**Figure B.16 : Daisy Chain Driver Timing**

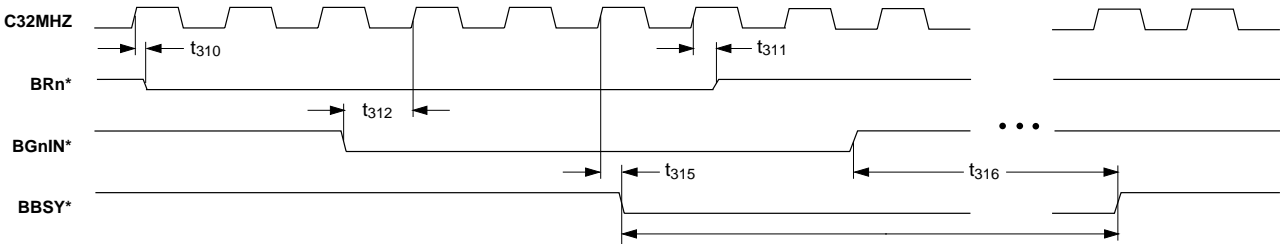


Figure B.17a : VMEbus Requester - Normal Release

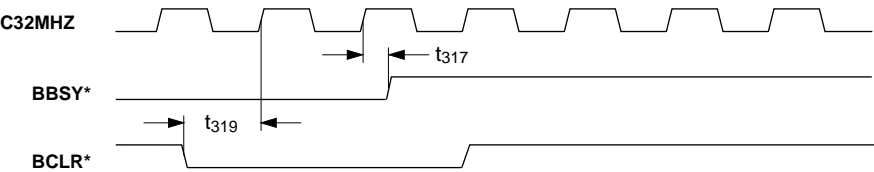


Figure B.17b : VMEbus Requester - Release with BCLR\*

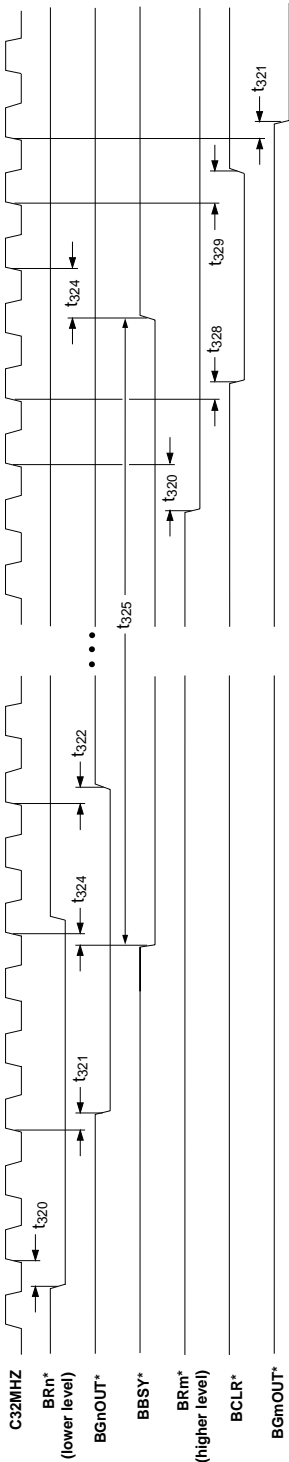


Figure B.18 : VMEbus Arbitration Timing



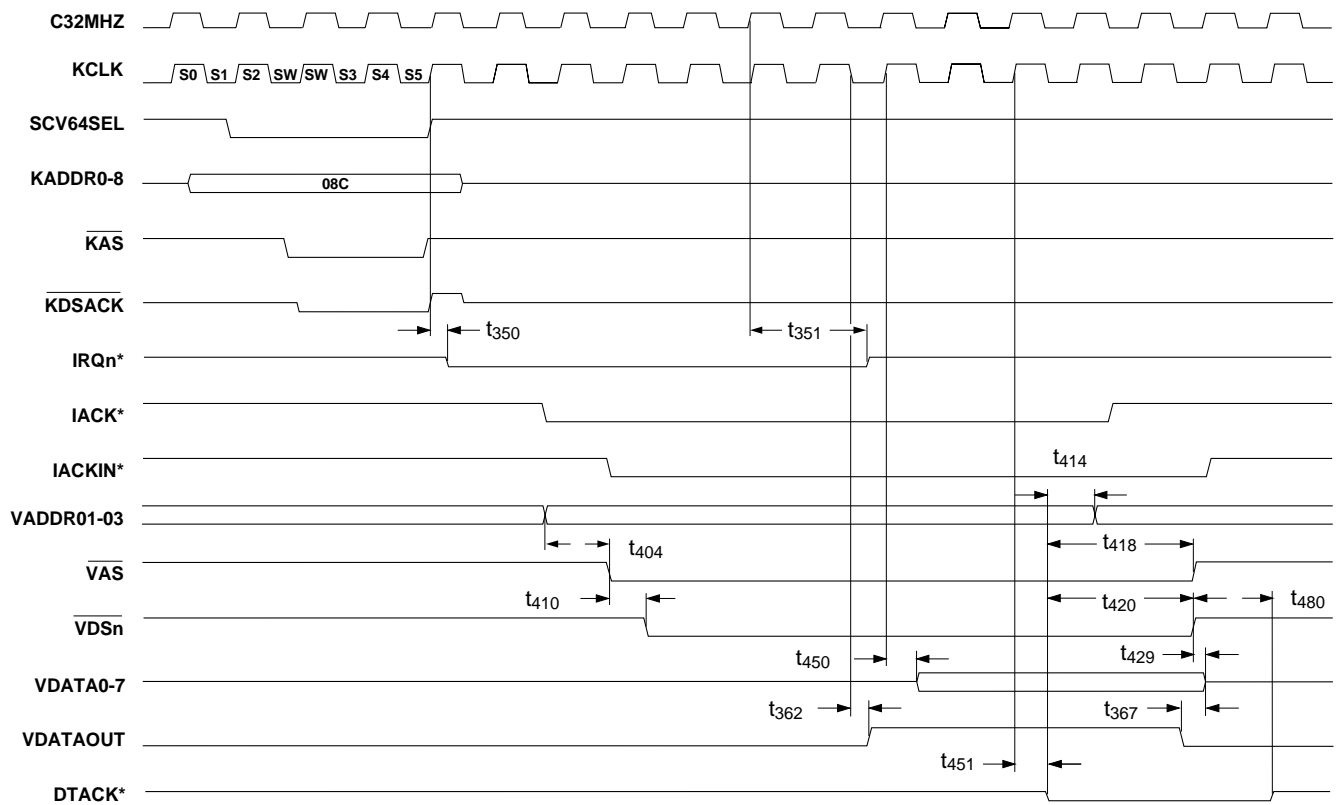


Figure B.19 : VMEbus Interrupter Timing

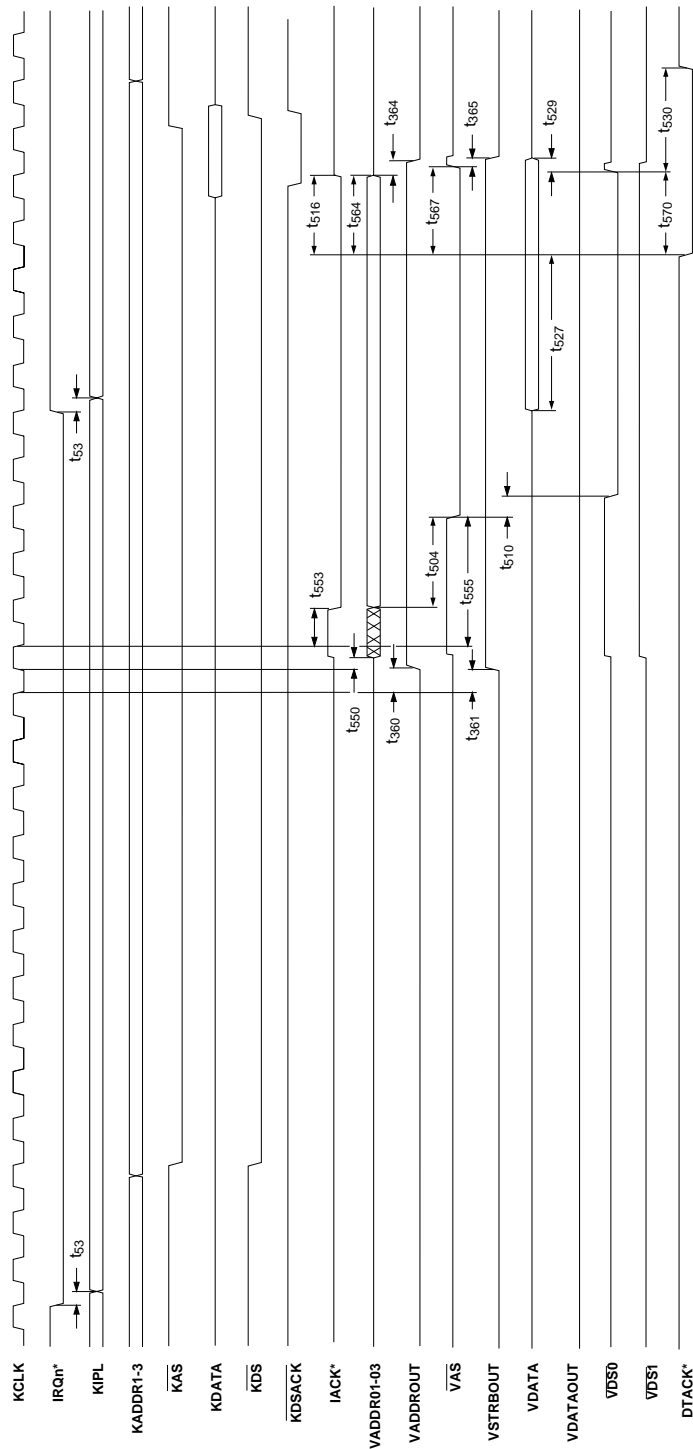
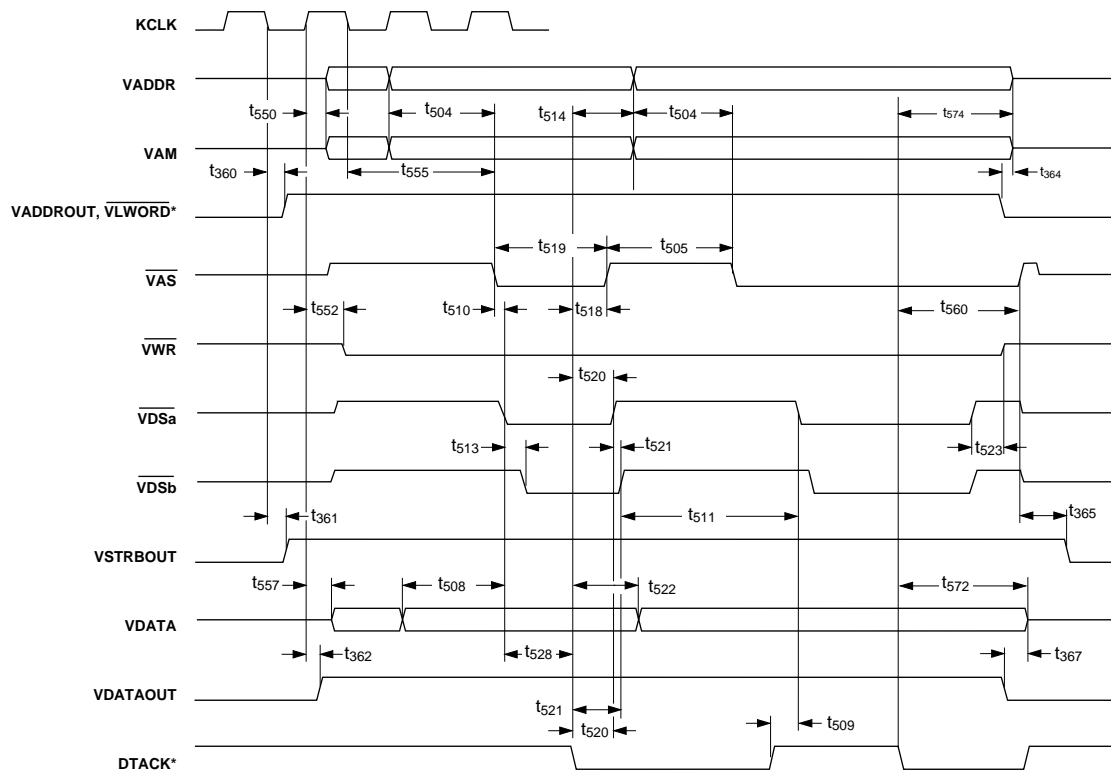


Figure B.20 : VMEbus Interrupt Handler



**Figure B.21 : Decoupled Write Cycle - SCV64 as VME Master**

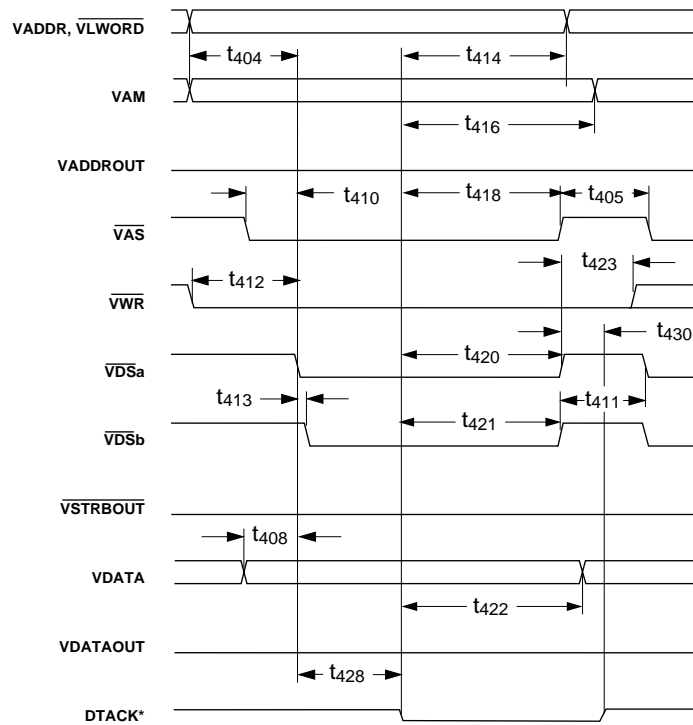
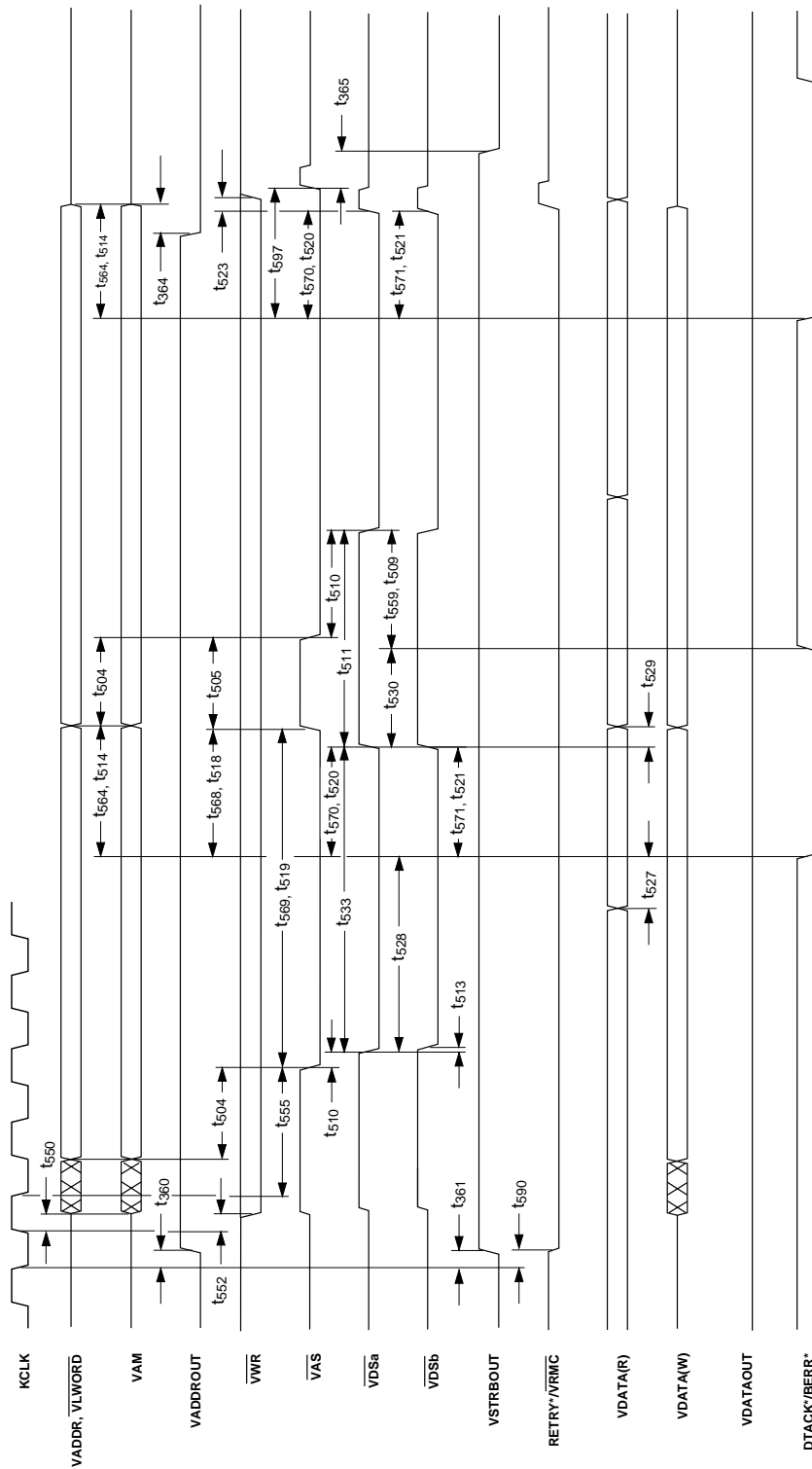
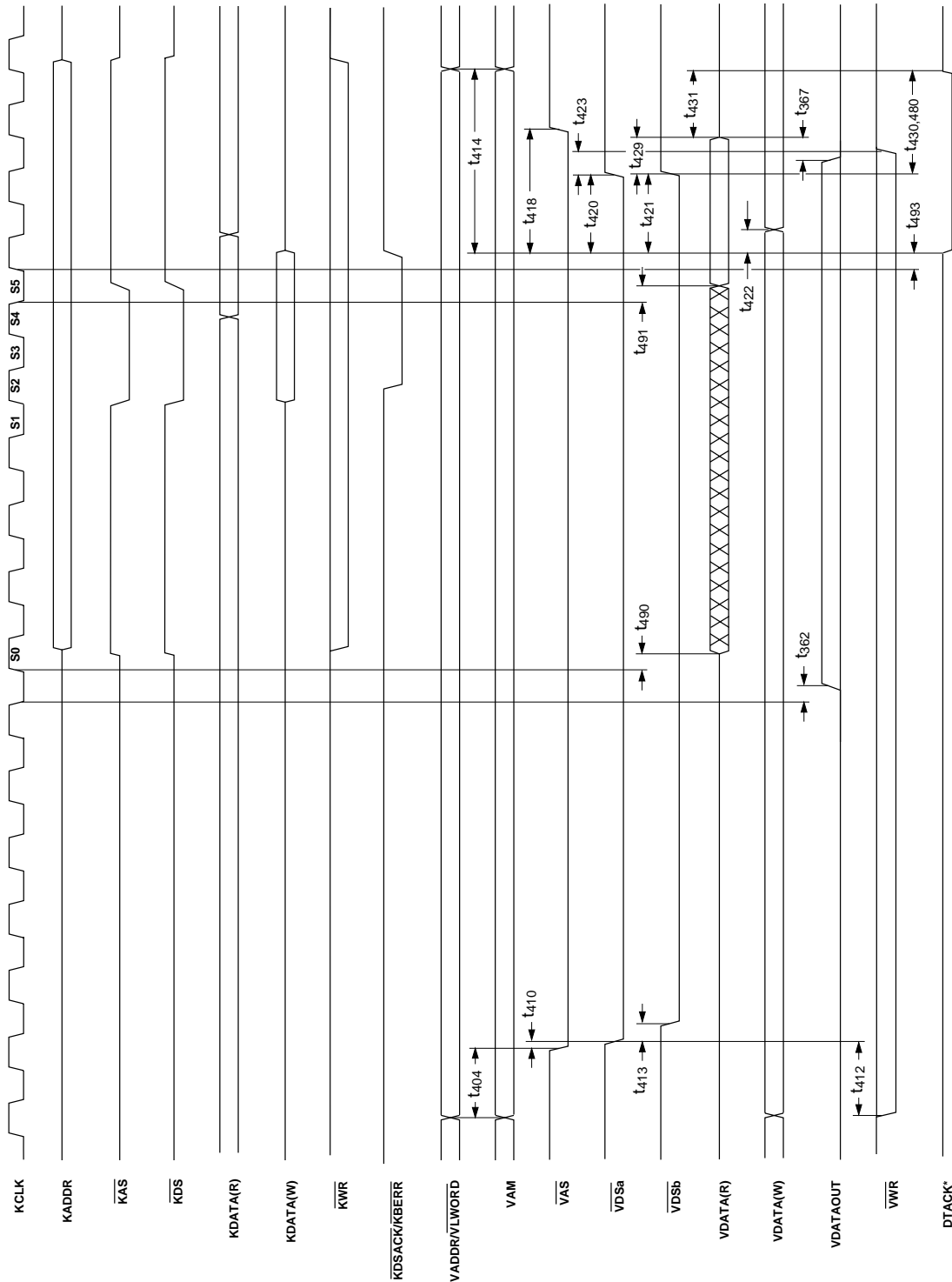


Figure B.22 : Decoupled Write Cycle - SCV64 as VME Slave

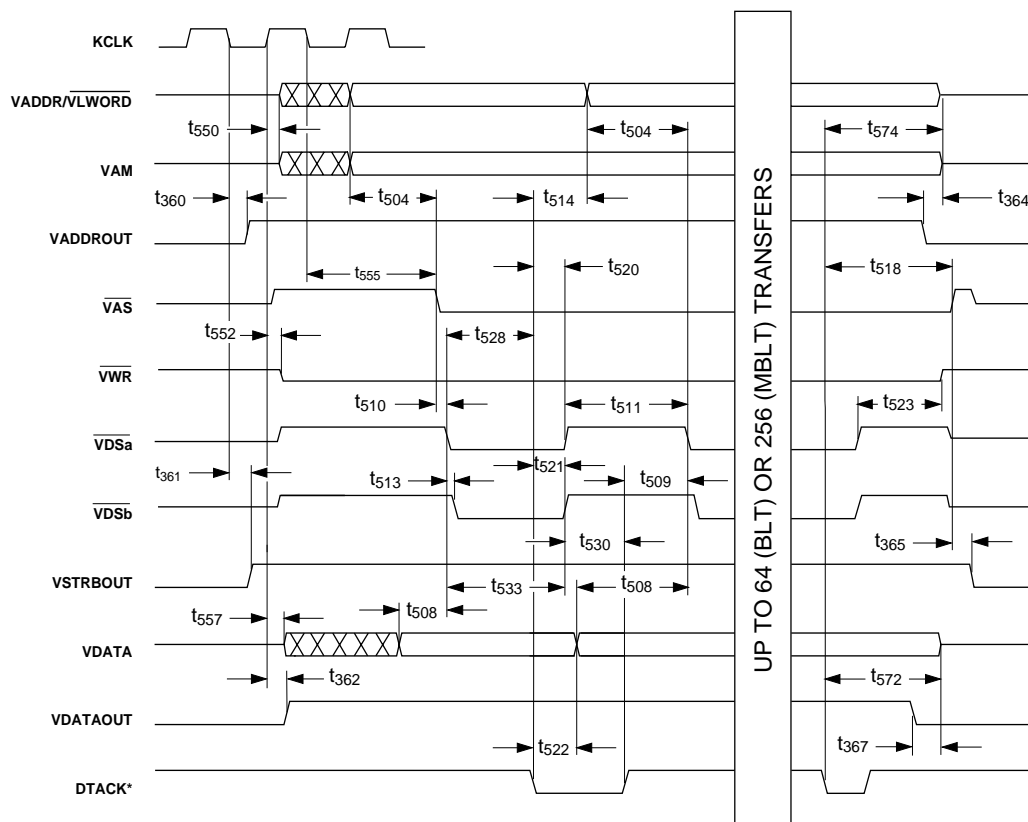


**†NOTE:** RETRY\*/VPMC asserted low during RMW cycles when the VRETRY line is configured as the SCV64 proprietary Read-Modify-Write pin

**Figure B.23 : Coupled Cycle - SCV64 as VME Master**



**Figure B.24 : Coupled Cycle - SCV64 as VME Slave**  
(see Figure 9 for Local Timing)



**Figure B.25 : BLT/MBLT Writes - SCV64 as VME Master**

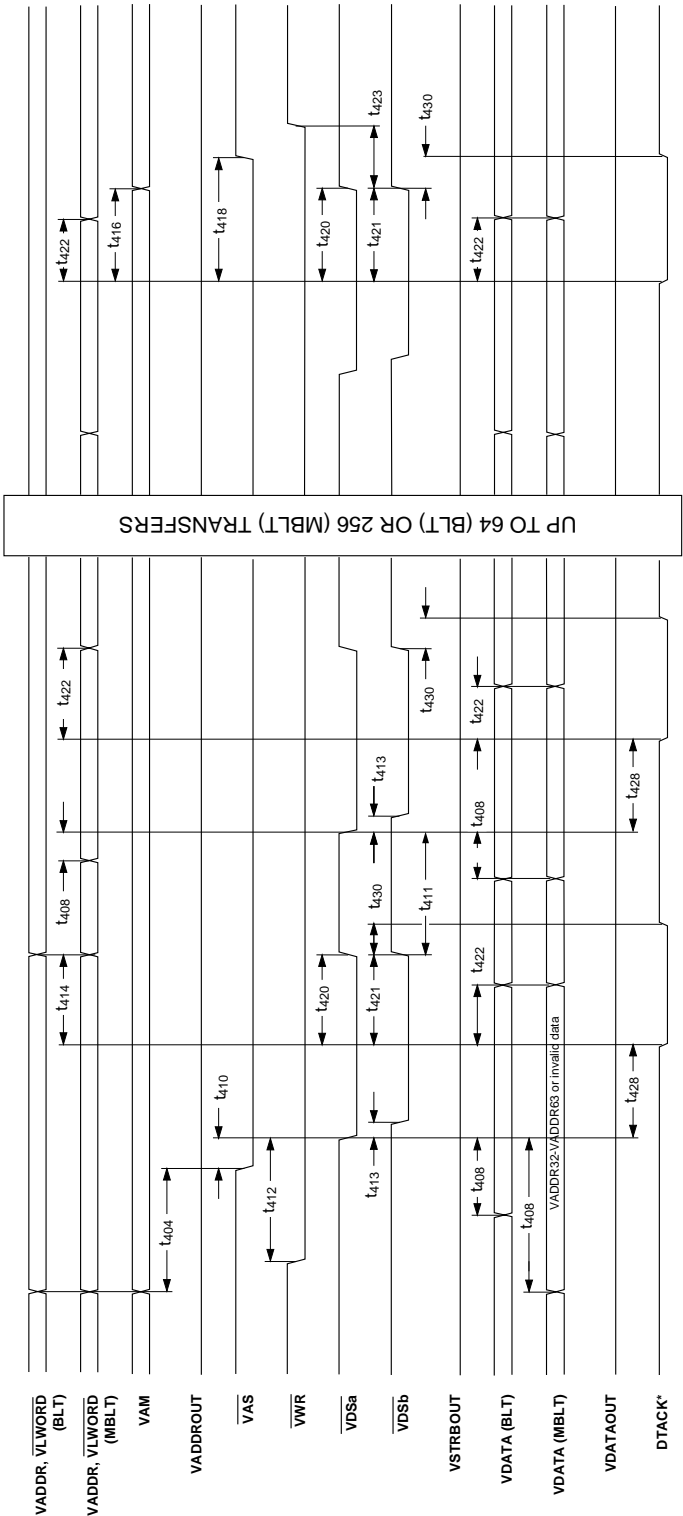
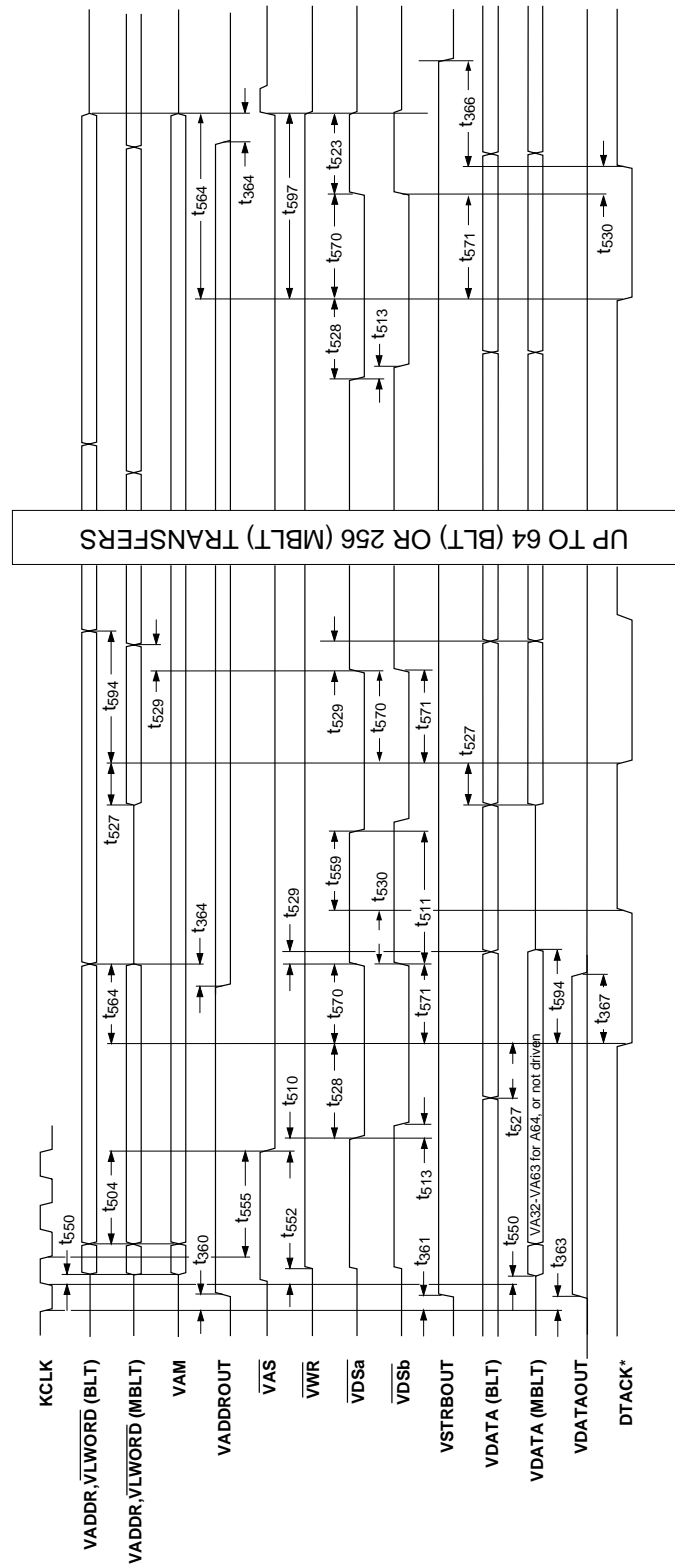
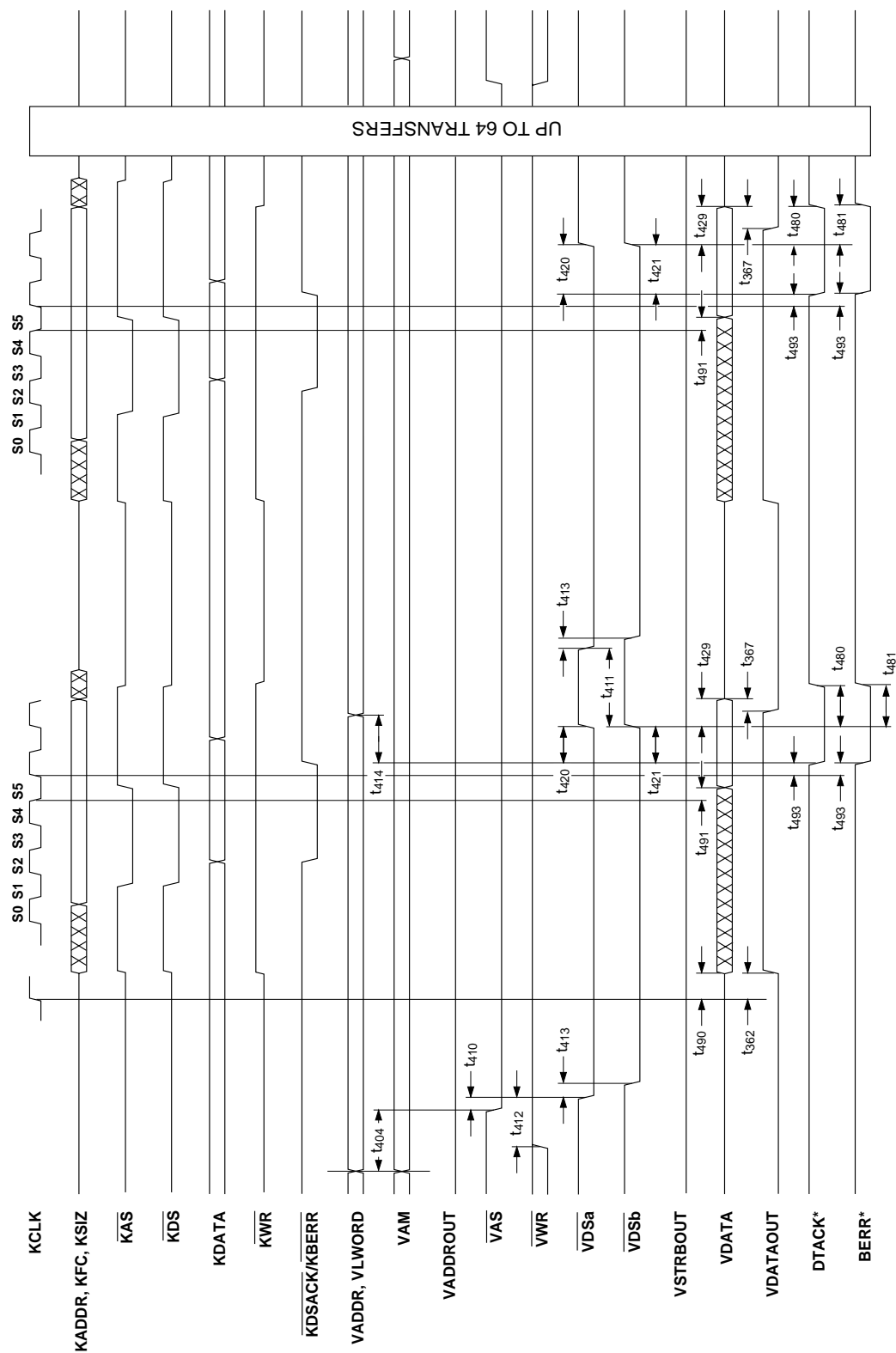


Figure B.26 : BLT/MBLT Writes - SCV64 as VME Slave



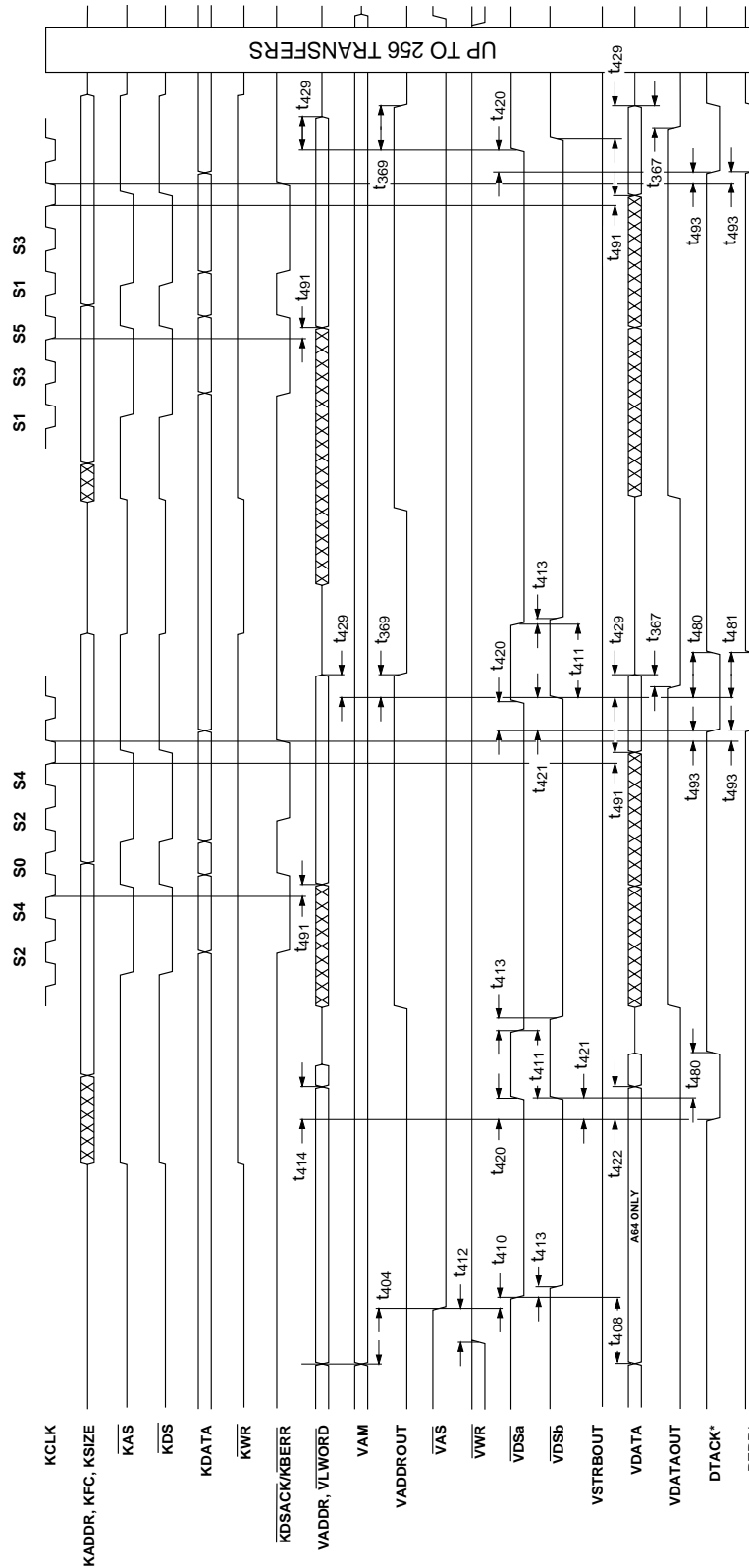


**Figure B.27 : BLT/MBLT Read Cycle - SCV64 as VME Master**



<sup>†</sup>NOTE: See Figure B.9 for local side timing

### Figure B.28: BLT Read Cycle - SCV64 as VME Slave



<sup>†</sup>**NOTE:** See Figure B-9 for local side timing

**Figure B.29: MBLT Read Cycle - SCV64 as VME Slave**



## Appendix C Performance

The SCV64 provides a complete high performance VMEbus interface for a variety of applications from high end multi-processing CPU cards, to low end slave only I/O and I/O manipulation cards. The following discussion outlines what performance can be expected from the SCV64 during data transfer, arbitration and daisy chaining.

Data transfer performance is handled in two sections: one covering coupled performance; the other covering decoupled performance. The operation and performance of these two modes varies considerably between the two. Generally coupled mode, because of its relative performance inefficiency would be used primarily for barrier transactions. These are transactions where the CPU does not terminate until the full transfer has completed. The CPU enters wait-states either waiting for data to be returned during a read, or for data to be successfully stored in the final destination.

Coupled (barrier) transactions are increasingly the exception on the VMEbus, as raw processing moves off the bus, and onto multi-processing cards. Instead, the VMEbus is being used to transfer large blocks of data between independent processes. An example might be a video frame grabber which transfers frame blocks over the VMEbus to a DSP card, which synthesizes data based upon the video frames. The data transfer would typically be handled in decoupled mode to maximize bus bandwidth. The DSP card may then report to a main controller either with barrier transactions or not, depending upon the application. To perform these block data transfers in coupled mode, would quickly saturate the system resources, whereas decoupling frees up resources and CPU processing power to perform other essential operations.

The following analyses show the SCV64's performance only up to its pins. Buffer delays and backplane skew were not incorporated into the calculations.

### C.1 Coupled Cycles

Coupled cycles are performed by the SCV64 during all read cycles (except DMA initiated reads), VME interrupt acknowledge cycles, and during write cycles when the FIFOs have been coupled.

When evaluating the performance of the SCV64 during coupled cycles, the characteristics of both the VME and local operations must be examined. Because of the involvement of both buses during the cycle, and the required arbitration for one bus, coupled cycles have considerably lower performance than decoupled ones.

### C.1.1 SCV64 as VME Master

During a coupled cycle when the SCV64 is the VME master, the following operations occur

- the local cycle is decoded,
- the VME bus is requested and granted,
- the cycle is driven onto the VME bus,
- the VME slave responds to and terminates the cycle, and
- the SCV64 terminates the local cycle.

The bus release mechanism programmed for the SCV64 affects performance. If the SCV64 is in Release When Done (RWD) mode, it must re-arbitrate for the bus during each cycle. In Release On Request (ROR) mode, the bus need only be arbitrated for once, and then all subsequent cycles may proceed at a higher rate.

The following figures illustrate the SCV64 performing coupled cycles in RWD and ROR mode. Each figure shows the relationship between operations on the two buses, and the two clock inputs, KCLK, and C32MHZ. KCLK is used mostly for the local synchronous bus, while C32MHZ is used for VMEbus request and ownership operations. Because it is coupled to the local bus, and because the local bus operates synchronous to KCLK, the VMEbus also tends to operate synchronous to KCLK.

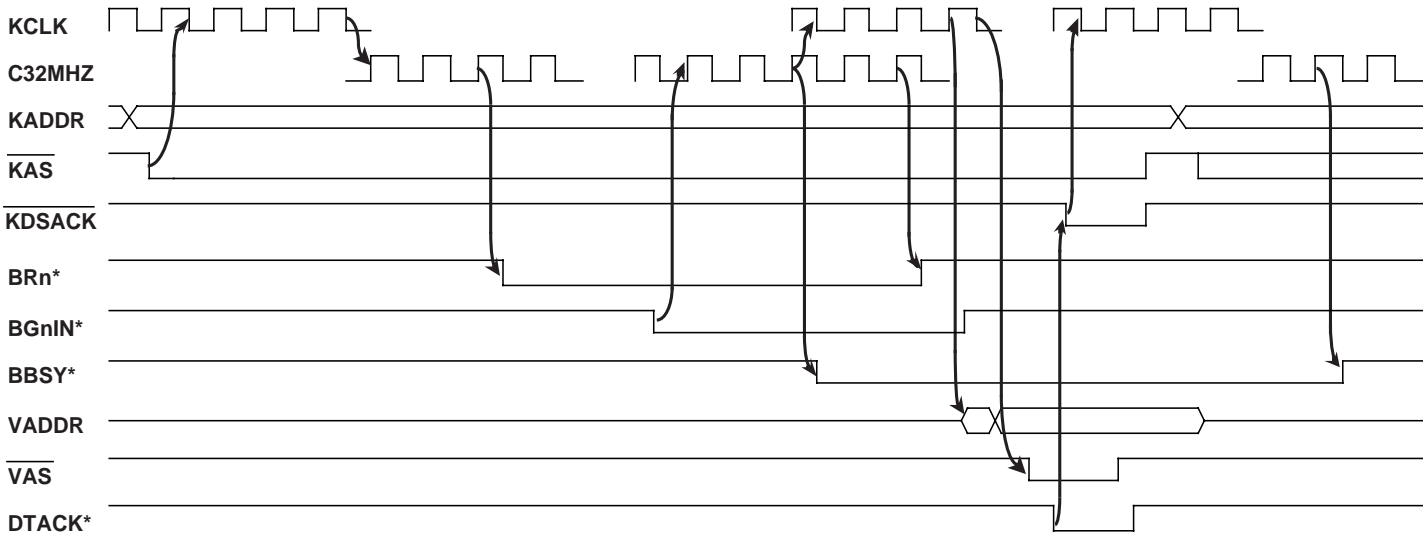
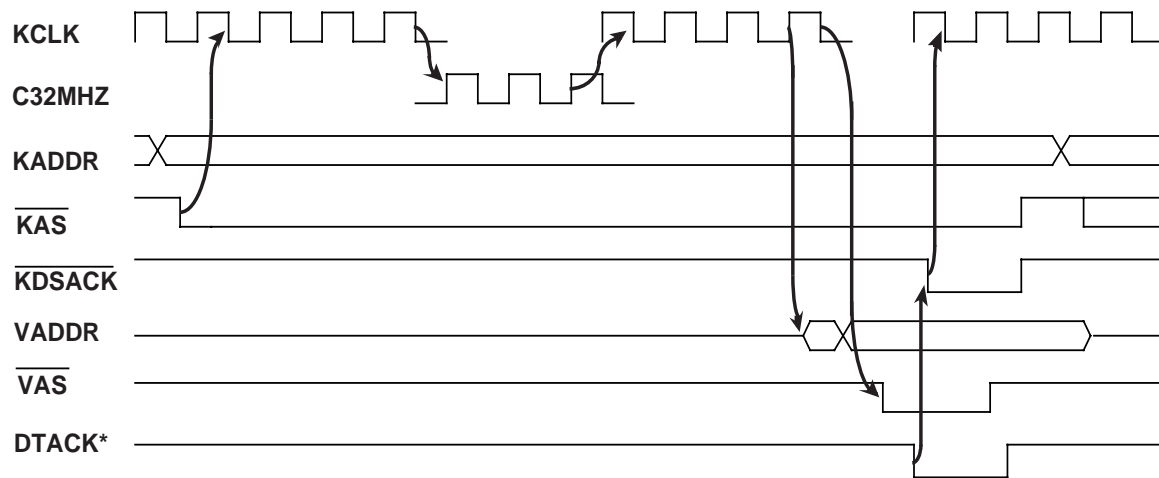


Figure C.1 : SCV64 as VME master - RWD mode



**Figure C.2 : SCV64 as VME master - ROR mode**

On the falling edge of each KCLK, the SCV64 examines the state of  $\overline{\text{KAS}}$ , and  $\overline{\text{VMEOUT}}$ . Once both these pins are asserted, it initiates internal state machines running off of both KCLK, and C32MHZ.

The typical measurements one may expect for coupled master cycles are:

RWD Mode:

$\overline{\text{KAS}}$  to  $\text{BRn}^*$ :  $3\text{k CLK} + 2.5 \text{ C32MHz} + t_{310}$

$\text{BGnIN}^*$  to  $\overline{\text{VAS}}$ :  $3.5\text{k CLK} + 2.5 \text{ C32MHz} + t_{555}$

$\text{DTACK}^*$  to  $\overline{\text{KDSACK}}$ :  $t_{172}$

ROR Mode:

$\overline{\text{KAS}}$  to  $\overline{\text{VAS}}$ :  $6\text{k CLK} + 2.5 \text{ C32MHz} + t_{555}$

$\text{DTACK}^*$  to  $\overline{\text{KDSACK}}$ :  $t_{172}$



A complete cycle (from S0 through to S5) will take:

RWD Mode:

$$10k \text{ CLK} + 5 \text{ C32MHz} + t_{310} + (\text{bus grant time}) + t_{555} + (\text{slave response}) + t_{172}$$

ROR Mode:

$$10k \text{ CLK} + 2.5 \text{ C32MHz} + t_{555} + (\text{slave response}) + t_{172}$$

### C.1.2 SCV64 as VME Slave

During a coupled cycle when the SCV64 is the VME slave, the following operations occur:

- the cycle is decoded to determine if it is relevant to the SCV64,
- the local bus must be requested and granted,
- the cycle is driven onto the local bus,
- the local slave responds to and terminates the local cycle, and
- the SCV64 terminates the VME cycle.

The SCV64 provides a local bus arbiter which may be used to arbitrate between the SCV64, and another device for access on the bus. This function uses an internal state machine to perform the arbitration. However, if the internal arbiter is not required such as when the SCV64 is the only master on the bus, or when an external arbiter is implemented, the designer may bypass the internal arbiter, and as such increase performance.

The figures on the following page show coupled cycles (reads, and writes perform identically) both when the internal arbiter is active, and when it is bypassed. The net effect is the addition of several clock cycles when the arbiter is active. Both figures assume zero wait-state memory.

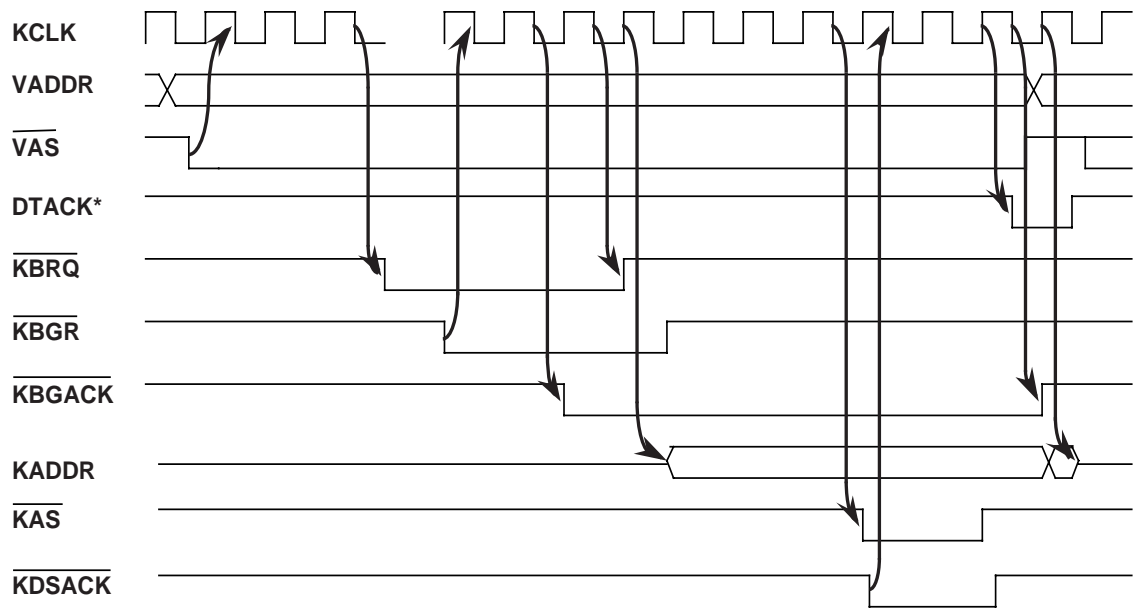


Figure C.3 : Coupled Cycle - SCV64 as VME Slave (local arbiter active)

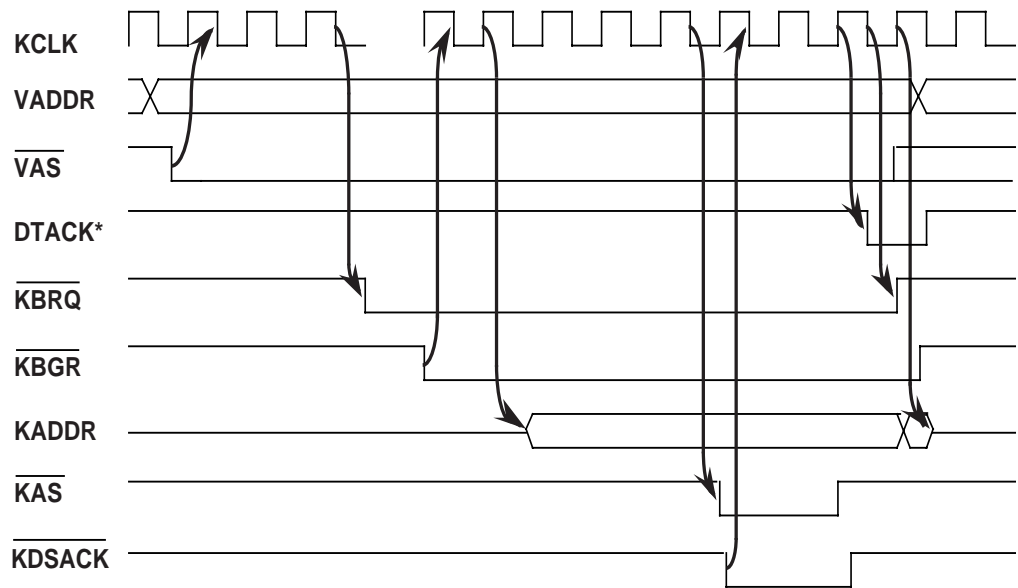


Figure C.4 : Coupled Cycle-SCV64 as VME Slave (local arbiter bypassed)

From the previous figures, the typical slave response ( $\overline{\text{VAS}}$  asserted to DTACK\*) can be calculated.

Arbiter Active:

$$15\text{k CLK} + t_{493}$$

Arbiter Bypassed:

$$10\text{k CLK} + t_{493}$$

These figures assume the minimum local bus grant time, which for arbiter active is 4 KCLK periods from the edge where  $\overline{\text{KBRQ}}$  is asserted, and for arbiter bypassed, 1 clock period from the same edge.

## C.2 Decoupled Cycles

The SCV64 operates at its peak performance, when in decoupled mode. Since the local and VME buses operate independently of each other, arbitration latencies have no effect on performance.

### C.2.1 SCV64 as VME Master

As a VME master, when the SCV64 is accessing an ideal VME slave (slave response = 30 ns), each data transfer will take 103 ns. This is the measurement taken from  $\overline{\text{VAS}}$  asserted to the next  $\overline{\text{VAS}}$  asserted. It does not take into account the initial arbitration time. However, so long as the transmit FIFO contains entries, the SCV64 will not re-arbitrate for the bus, and as such, the initial arbitration time becomes negligible.

When performing D32 standard writes, the above data transfer period translates to a peak transfer rate of 39 MB/s -- close to the theoretical VME maximum of 50MB/s (or an 80 ns period). BLT transfers occur at the same rate, since the SCV64 takes advantage of address pipelining during standard transfers. The transfer rate doubles to 78MB/s for MBLTs.

### C.2.2 SCV64 as VME Slave

When the SCV64 is being written to, and its receive FIFO has been enabled, it has a slave response time of 56 ns (VDS\* to DTACK\* asserted)

### C.3 Daisy Chains

The SCV64 propagates three of the four BGnOUT\* lines directly from the BGnIN\* lines. The fourth line, the level that the SCV64 is performing its bus requests on is double sampled by the 32MHz clock before being propagated. As such there are two performance numbers to be considered for the bus grant propagation delays:

Bus Grant Propagation Delay:  $t_{308}$  (not requesting level)

$1.5 \text{ C32MHz} + t_{307}$  (requesting level)

As with bus grant propagation delay for the SCV64's requesting level, the IACK daisy chain driver is also double sampled by the C32MHZ clock before being propagated:

IACK Daisy Chain Delay:  $1.5 \text{ C32MHz} + t_{302}$

### C.4 Arbiter Functions

When the SCV64 is acting as the system arbiter, it uses the C32MHZ clock to double sample each bus request before issuing a bus grant to propagate down the daisy chain. When configured as a priority (PRI) arbiter, the grant is issued on the next edge of the C32MHZ clock, if the bus is currently not in use ( $\overline{\text{VAS}}$ , and BBSY\* high). As such, the bus request to grant delay when the bus is free will typically be:

Bus Grant Arbitration Delay:  $2.5 \text{ C32MHz} + t_{321}$  (PRI mode)

When configured for round robin (RR) arbitration, the SCV64 cycles through each of the request levels issuing a grant to any request on the currently examined level. If a request comes in while the SCV64 is monitoring that level, it will respond with the grant in the same time as in PRI mode above. However, the request may come in just as that level has passed in the request level ring. In this case, the grant take a further 6 C32MHz clocks before being issued (two clocks per level), assuming no other requests are pending on other levels. On average then, the grant will typically take 3 clocks longer than in PRI mode:

Bus Grant Arbitration Delay:  $5.5 \text{ C32MHz} + t_{321}$  (RR mode)

## C.5 Summary

The following tables describe the SCV64's performance under the following conditions:

- KCLK = 33MHz
- C32MHZ = 32MHz
- $V_{DD} = 5\text{ V}$
- temp = 25°C

**Period** refers to the time between consecutive cycles. For standard cycles this would refer to the time from address strobe to the next address strobe. For block transfer, it refers to the time from data strobe to data strobe. In the case of BLT, and MBLT transfers this refers to the data beat of the transfer, since the effect of the address phase diminishes with increasing block length. **Transfer rate** is the expected data transfer frequency with a 32 bit data bus width (64 bit for MBLT transfers).

Bus request to bus grant delays are assumed to be that which can be expected when the SCV64 is the SYSCON, and there is no bus contention. Also, when calculating it's master performance, a slave response of 30ns (the VME minimum) is used.

Under typical conditions<sup>†</sup>, the following parameters are used:

**Table C.1 : Timing Parameters**

Timing Parameter	Description	Typical (ns)
$t_{172}$	DTACK* to $\overline{\text{KDSACK}}$ asserted	61
$t_{302}$	IACKOUT* asserted from C32MHz	8
$t_{308}$	Bus Grant daisy chain delay	8
$t_{310}$	BRn* asserted from C32MHz	10
$t_{493}$	DTACK* asserted from KCLK	13
$t_{321}$	BGnOUT* asserted from C32MHz	11
$t_{555}$	first $\overline{\text{VAS}}$ asserted from KCLK	59

<sup>†</sup> Typical conditions are:  $V_{DD} = 5.0\text{ V}$

temp = 25°C

**Table C.2 : Data Transfer - SCV64 as VME Master**

Cycle Type	Period (ns)	Rate (MB/s)
Coupled Standard Read - RWD mode	693	5.8
Coupled Standard Read - ROR mode	528	7.6
Coupled Standard Write - RWD mode	693	5.8
Coupled Standard Write - ROR mode	528	7.6
Decoupled Write	103	38.8

**Table C.3 : Data Transfer - DMA**

Cycle Type	Period (ns)	Rate (MB/s)
Standard Read	145	27.6
Standard Write	103	38.8
BLT Read	145	27.6
BLT Write	103	38.8
MBLT Read	145	55.2
MBLT Write	103	77.6
Local burst reads	30	133.3
Local burst writes	60	66.7
Local single cycle reads	90	44.4
Local single cycle writes	90	44.4

**Table C.4 : Data Transfer - SCV64 as VME Slave**

Cycle Type	Slave Response (ns)
Coupled D32 Read - arbiter active	463
Coupled D32 Read - arbiter bypassed	313
Coupled D32 Write - arbiter active	463
Coupled D32 Write - arbiter bypassed	313
Coupled MBLT Cycle - arbiter active	553
Coupled MBLT Cycle - arbiter bypassed	403
Decoupled Write	56
Decoupled BLT Write	56
Decoupled MBLT Write	56

Note: the above measurements assume a  $\overline{\text{KBRQ}}$  to  $\overline{\text{KBGR}}$  delay, of less than 10ns for arbiter bypassed, or 90ns for arbiter active, and zero wait state memory.

**Table C.5 : Daisy Chains**

Daisy Chain	Propagation Delay (ns)
BGnIN* to BGnOUT* (not requesting level)	8
BGnIN* to BGnOUT* (requesting level)	56
IACKIN* to IACKOUT*	86

**Table C.6 : VMEbus Arbiter**

Cycle Type	Delay (ns)
Bus Request to Bus Grant - PRI or next RR request in ring	87
Bus Request to Bus Grant - typical RR	181
Bus Clear Delay - higher bus request to BCLR asserted	56





# Appendix D VMEbus-Local Bus Cycle Mapping

## D.1 Cycle Translation

The VME/local cycle mapping functionality provided by the SCV64, while targeted at 68020/030 CPU's, can be used under a variety of architectures. This appendix outlines how the SCV64 maps local cycles into VME cycles, and VME cycles into local ones, as well as which byte lanes are active during the cycle. Cycle type and byte lane translation are based upon:

- type of cycle (read / write),
- direction of cycle (VME to local, or local to VME),
- the value of the SWAP bit in the MODE register, and
- the value of the BUSSIZ bit in the MODE register

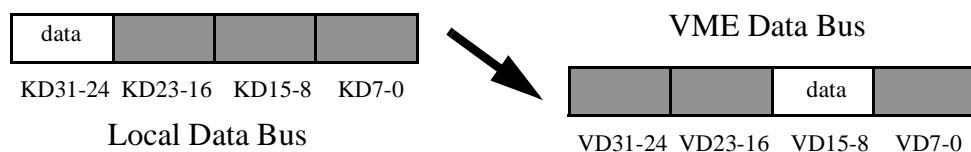
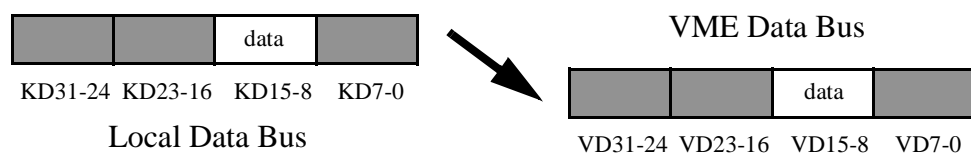
The tables below are grouped primarily upon type and direction of the cycle. They are further differentiated by whether the SCV64 is acting as the local master (VME slave), or local slave (VME master). When acting as the local master, the SCV64 will be asserting  $\overline{\text{RAMSEL}}$ , while as the local slave,  $\overline{\text{VMEOUT}}$  will be asserted by a local device.

Separate tables exist for cases where the SWAP bit (see MODE register) is cleared or set. When SWAP is cleared and the SCV64 is the local slave, the SCV64 will use local byte lanes in accordance with a 68020/030 architecture. When SWAP is set during these accesses, the SCV64 will use byte lanes matching directly with the relevant byte lanes on the VMEbus. The type of transfer generated onto the VMEbus is not affected by this bit.

**Table D.1 : Effect of SWAP Bit in the MODE Register on Byte Lane Translation**

SWAP	Effect
0	map local big endian byte lanes to VME byte lanes
1	map local byte lanes directly to VME byte lanes

For example, when performing a single byte, longword aligned (BYTE (0)) transfer, the SCV64 will use the data in the highest byte lane (KDATA31-24) to generate a single byte transfer on the VMEbus. However, with the SWAP bit set the SCV64 will use the second byte lane (KDATA 15-8) to generate the same transfer.

**Figure D.1 : Single Longword Aligned Byte Transfer with SWAP = 0****Figure D.2 : Single Longword Aligned Byte Transfer with SWAP = 1**

*Note: Byte lane requirements are not affected by the SWAP bit during accesses from the VMEbus. (i.e. when the SCV64 is acting as the VME slave.)*

As well as showing the type of cycle generated on one bus as a result of an incoming cycle on the other, the following tables also outline the byte lane requirements. For this purpose, a BYTE(n) convention is used, where n represents the address offset from an even 32-bit boundary (longword aligned). Each table shows either where the SCV64 presents data, or where data is expected to be present for the SCV64 to transfer.

In several cases, when the SCV64 is driving the local data lanes, it presents the same data in duplicate byte lanes. This is compatible with 68020/030 architectures, and local devices may pick up the data from either byte position.

As well, required data presentation of data on the byte lanes varies depending on the state of the BUSSIZ bit in the MODE register. This bit affects whether or not unaligned transfers (UATs) will be generated on the VMEbus by the SCV64. When set, the SCV64 will generate UAT cycles, and will always respond with a 32-bit  $\overline{\text{KDSACKx}}$ . This implies that the user-defined D16 areas of the memory map will become D32. The size of the data transfer will be uniquely defined by the state of the KSIZ lines.

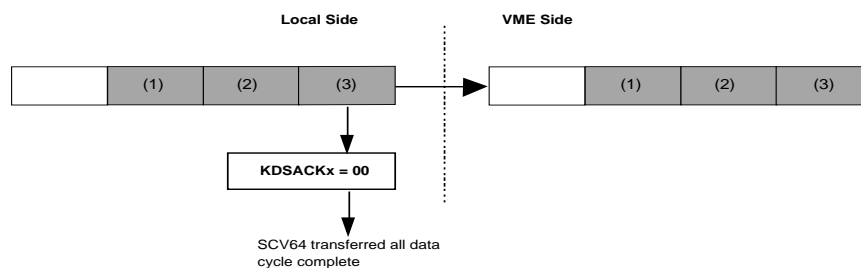
**Table D.2 : Port Size as Indicated by BUSSIZ and Cycle Termination**

Data Space	BUSSIZ	$\overline{\text{KDSACK1}}$	$\overline{\text{KDSACK0}}$	Port Size
D32	0*	0	0	32 bit
	0	1	0	16 bit
	1	0	0	32 bit
D16	0	1	0	16 bit
	1	0	0	32 bit

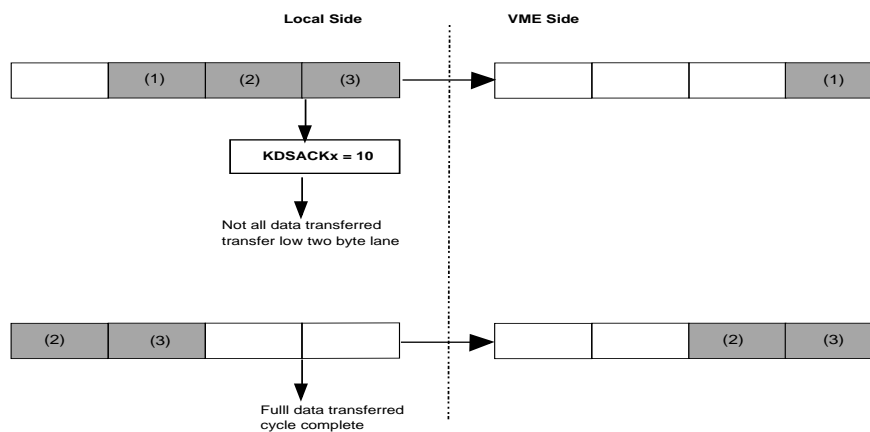
*\*Longword and longword aligned accesses only.*

When cleared, the SCV64 disables UAT generation on the VMEbus and will now terminate cycles with either a 32-bit  $\overline{\text{KDSACKx}}$  or a 16-bit  $\overline{\text{KDSACKx}}$ . It terminates longword aligned transfers with a 32-bit  $\overline{\text{KDSACKx}}$ . All others are terminated with a 16-bit  $\overline{\text{KDSACKx}}$ . A 16-bit termination implies that only data in the uppermost two byte lanes (KDATA31-24, and KDATA23-16) are relevant to the cycle. Any data on the other two byte lanes must be transferred in a later cycle.

For example, consider a tri-byte write at a longword offset 1. With BUSSIZ = 1, the SCV64 will pick up data in the low three byte lanes, and terminate the cycle with a 32 bit DSACK, and generate a unaligned tri-byte write on the VMEbus. With BUSSIZ = 0, the SCV64 will terminate the cycle with a 16-bit DSACK, and generate a single byte write at offset 1 (BYTE(1)).



**Figure D.3 : Tri-byte Transfer with BUSSIZ = 1**



**Figure D.4 : Tri-byte Transfer with BUSSIZ = 0**

The state of the BUSSIZ bit can also determine where relevant data will be expected by the SCV64, even when the cycle performed on the VMEbus is identical. When BUSSIZ = 0, the SCV64 will generally pick up, and present data on the uppermost byte lanes - the exception being longword, longword aligned transfers where all four byte lanes are used. When BUSSIZ = 1, byte lane use is determined by the offset from the bytes being transferred relative to a longword boundary. BYTE(0) always appears in the uppermost byte lane, BYTE(1) always appears in the next byte lane, and so on. Both states can be supported by duplicating the data in the uppermost two byte lanes with data in the lowermost two byte lanes when the transfer is either byte size, or word size.

When performing read cycles where the SCV64 is the local slave, it is not necessary to know the state of the BUSSIZ bit. The byte lanes relevant to the cycle can be uniquely determined through knowledge of the transfer size, the address offset, and the KDSACK returned by the SCV64.



*Note: During cycles where the SCV64 is the local master (VME slave), the BUSSIZ bit has no effect either on type of cycle generated on the local bus, or on the byte lanes in use during the cycle.*

**Table D.3 : Transfer Size Encoding**

KSIZE1	KSIZE0	Size
0	1	Byte
1	0	Word
1	1	3 bytes
0	0	Long Word

**Table D.4 : Read Cycles - SCV64 as Local Slave/VME Master  
(SWAP=0)**

KSIZE	KA01-00	KDSACK	Data Lanes Driven by SCV64				VME Cycle Generated
			KD31-24	KD23-16	KD15-08	KD07-00	
00	00	00	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	QUAD(0-3)
00	00	01	BYTE(0)	BYTE(1)	BYTE(0)	BYTE(1)	DBL BYTE(0-1)
00	01	00		BYTE(1)	BYTE(2)	BYTE(3)	UAT(1-3)
00	01	01		BYTE(1)		BYTE(1)	BYTE(1)
00	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
00	10	01	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)
00	11	00				BYTE(3)	BYTE(3)
00	11	01		BYTE(3)		BYTE(3)	BYTE(3)
01	00	00	BYTE(0)		BYTE(0)		BYTE(0)
01	00	01	BYTE(0)		BYTE(0)		BYTE(0)
01	01	00		BYTE(1)		BYTE(1)	BYTE(1)
01	01	01		BYTE(1)		BYTE(1)	BYTE(1)
01	10	00			BYTE(2)		BYTE(2)
01	10	01	BYTE(2)		BYTE(2)		BYTE(2)
01	11	00				BYTE(3)	BYTE(3)
01	11	01		BYTE(3)		BYTE(3)	BYTE(3)
10	00	00	BYTE(0)	BYTE(1)	BYTE(0)	BYTE(1)	DBL BYTE(0-1)
10	00	01	BYTE(0)	BYTE(1)	BYTE(0)	BYTE(1)	DBL BYTE(0-1)
10	01	00		BYTE(1)	BYTE(2)		UAT(1-2)
10	01	01		BYTE(1)		BYTE(1)	BYTE(1)
10	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
10	10	01	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)
10	11	00				BYTE(3)	BYTE(3)
10	11	01		BYTE(3)		BYTE(3)	BYTE(3)
11	00	00	BYTE(0)	BYTE(1)	BYTE(2)		UAT(0-2)
11	00	01	BYTE(0)	BYTE(1)	BYTE(0)	BYTE(1)	DBL BYTE(0-1)
11	01	00		BYTE(1)	BYTE(2)	BYTE(3)	UAT(1-3)
11	01	01		BYTE(1)		BYTE(1)	BYTE(1)
11	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
11	10	01	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)
11	11	00				BYTE(3)	BYTE(3)
11	11	01		BYTE(3)		BYTE(3)	BYTE(3)

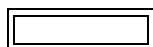
**Table D.5 : Read Cycles-SCV64 as Local Slave/VME Master (SWAP=1)**

KSIZE	KA01-00	KDSACK	Data Lanes Driven by SCV64				VME cycle generated
			KD31-24	KD23-16	KD15-08	KD07-00	
00	00	00	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	QUAD(0-3)
00	00	01			BYTE(0)	BYTE(1)	DBL BYTE(0-1)
00	01	00		BYTE(1)	BYTE(2)	BYTE(3)	UAT(1-3)
00	01	01				BYTE(1)	BYTE(1)
00	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
00	10	01			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
00	11	00				BYTE(3)	BYTE(3)
00	11	01				BYTE(3)	BYTE(3)
01	00	00			BYTE(0)		BYTE(0)
01	00	01			BYTE(0)		BYTE(0)
01	01	00				BYTE(1)	BYTE(1)
01	01	01				BYTE(1)	BYTE(1)
01	10	00			BYTE(2)		BYTE(2)
01	10	01			BYTE(2)		BYTE(2)
01	11	00				BYTE(3)	BYTE(3)
01	11	01				BYTE(3)	BYTE(3)
10	00	00			BYTE(0)	BYTE(1)	DBL BYTE(0-1)
10	00	01			BYTE(0)	BYTE(1)	DBL BYTE(0-1)
10	01	00		BYTE(1)	BYTE(2)		UAT(1-2)
10	01	01				BYTE(1)	BYTE(1)
10	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
10	10	01			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
10	11	00				BYTE(3)	BYTE(3)
10	11	01				BYTE(3)	BYTE(3)
11	00	00	BYTE(0)	BYTE(1)	BYTE(2)		UAT(0-2)
11	00	01			BYTE(0)	BYTE(1)	DBL BYTE(0-1)
11	01	00		BYTE(1)	BYTE(2)	BYTE(3)	UAT(1-3)
11	01	01				BYTE(1)	BYTE(1)
11	10	00			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
11	10	01			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
11	11	00				BYTE(3)	BYTE(3)
11	11	01				BYTE(3)	BYTE(3)

**Table D.6 : Write Cycles - SCV64 as Local Slave/VME Master  
(SWAP=0)**

		Data Lanes Driven by Local Device				VME Cycle Generated	
KSIZE	KA01-00	KD31-24	KD23-16	KD15-08	KD07-00	BUSSIZ = 0	BUSSIZ = 1
00	00	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	QUAD(0-3)	QUAD(0-3)
00	01		BYTE(1)	BYTE(2)	BYTE(3)	BYTE(1)	UAT(1-3)
00	10	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
00	11		BYTE(3)		BYTE(3)	BYTE(3)	BYTE(3)
01	00	BYTE(0)				BYTE(0)	BYTE(0)
01	01		BYTE(1)			BYTE(1)	BYTE(1)
01	10	BYTE(2)		BYTE(2)		BYTE(2)	BYTE(2)
01	11		BYTE(3)		BYTE(3)	BYTE(3)	BYTE(3)
10	00	BYTE(0)	BYTE(1)			DBL BYTE(0-1)	DBL BYTE(0-1)
10	01		BYTE(1)	BYTE(2)		BYTE(1)	UAT(1-2)
10	10	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
10	11		BYTE(3)		BYTE(3)	BYTE(3)	BYTE(3)
11	00	BYTE(0)	BYTE(1)	BYTE(2)		DBL BYTE(0-1)	UAT(0-2)
11	01		BYTE(1)	BYTE(2)	BYTE(3)	BYTE(1)	UAT(1-3)
11	10	BYTE(2)	BYTE(3)	BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
11	11		BYTE(3)		BYTE(3)	BYTE(3)	BYTE(3)

*Notes:*



*indicates byte lanes applicable only to BUSSIZ = 0*



*indicates byte lanes applicable only to BUSSIZ = 1*

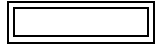
*all other byte lanes applicable to both BUSSIZ = 0, and BUSSIZ = 1*



**Table D.7 : Write Cycles-SCV64 as Local Slave/VME Master  
(SWAP=1)**

KSIZE	KA01-00	Data Lines Driven by Local Device				VME Cycle Generated	
		KD31-24	KD23-16	KD15-08	KD07-00	BUSSIZ = 0	BUSSIZ = 1
00	00	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	QUAD(0-3)	QUAD(0-3)
00	01		BYTE(1)	BYTE(2)	(1) (3)	BYTE(1)	UAT(1-3)
00	10			BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
00	11				BYTE(3)	BYTE(3)	BYTE(3)
01	00			BYTE(0)		BYTE(0)	BYTE(0)
01	01				BYTE(1)	BYTE(1)	BYTE(1)
01	10			BYTE(2)		BYTE(2)	BYTE(2)
01	11				BYTE(3)	BYTE(3)	BYTE(3)
10	00			BYTE(0)	BYTE(1)	DBL BYTE(0-1)	DBL BYTE(0-1)
10	01		BYTE(1)	BYTE(2)	(BYTE(1))	BYTE(1)	UAT(1-2)
10	10			BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
10	11				BYTE(3)	BYTE(3)	BYTE(3)
11	00	BYTE(0)	BYTE(1)	(0) (2)	BYTE(1)	DBL BYTE(0-1)	UAT(0-2)
11	01		BYTE(1)	BYTE(2)	(1) (3)	BYTE(1)	UAT(1-3)
11	10			BYTE(2)	BYTE(3)	DBL BYTE(2-3)	DBL BYTE(2-3)
11	11				BYTE(3)	BYTE(3)	BYTE(3)

*Notes:*



*indicates byte lanes applicable only to  $BUSSIZ = 0$*



*indicates byte lanes applicable only to  $BUSSIZ = 1$*

*all other byte lanes applicable to both  $BUSSIZ = 0$ , and  $BUSSIZ = 1$*

**Table D.8 : Read/Write Cycles - SCV64 as Local Master/VME Slave**  
**(SWAP =x, BUSSIZ=x)**

KSIZE	KA01-00	Data Lines Driven by Local Device				VME Cycle Accepted
		KD31-24	KD23-16	KD15-08	KD07-00	
00	00	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)	QUAD(0-3)
00	01	-	-	-	-	-
00	10	-	-	-	-	-
00	11	-	-	-	-	-
01	00	BYTE(0)		BYTE(0)		BYTE(0)
01	01		BYTE(1)		BYTE(1)	BYTE(1)
01	10			BYTE(2)		BYTE(2)
01	11				BYTE(3)	BYTE(3)
10	00	BYTE(0)	BYTE(1)	BYTE(0)	BYTE(1)	DBL BYTE(0-1)
10	01		BYTE(1)	BYTE(2)		UAT(1-2)
10	10			BYTE(2)	BYTE(3)	DBL BYTE(2-3)
10	11	-	-	-	-	-
11	00	BYTE(0)	BYTE(1)	BYTE(2)		UAT(0-2)
11	01		BYTE(1)	BYTE(2)	BYTE(3)	UAT(1-3)
11	10	-	-	-	-	-
11	11	-	-	-	-	-

*Note: Duplicated entries for VMEIN writes are given heavy outlines in the table above.*

## D.2 Address Space Mapping

**Table D.9 : KFC Signal Translation to VME Address Space  
(SCV64 as VME Master)**

KFC2	KFC1	KFC0	AM codes	VME address space
0	0	0	0A, 29, 3A	non-privileged program access
0	0	1	09, 29, 39	non-privileged data access
0	1	0	0A, 29, 3A	non-privileged program access
0	1	1	09, 29, 39	non-privileged data access
1	0	0	0E, 2D, 3E	supervisory program access
1	0	1	0D, 2D, 3D	supervisory data access
1	1	0	0E, 2D 3E	supervisory program access
1	1	1	0D, 2D, 3D	supervisory data access

**Table D.10 : VME AM Code Translation to KFC Signals (SCV64 as VME Slave)**

AM code	VME address space	KFC2	KFC1	KFC0
	–	0	0	0
09, 39 0B, 3B 00	non-privileged data access non-privileged block transfer A64 64-bit block transfer	0	0	1
0A, 3A 08, 38 00	non-privileged program access non-privileged 64-bit block transfer A64 64-bit block transfer	0	1	0
	BLT, MBLT (FIFOBEN set)	0	1	1
	–	1	0	0
0D, 3D 0F, 3F	supervisory data access supervisory block transfer	1	0	1
0E, 3E 0C, 3C	supervisory program access supervisory 64-bit block transfer	1	1	0



# Appendix E Applications

This appendix covers the following topics:

- “VMEbus Interface” on page E-1
- “Local Bus Interface for Slave Only Applications” on page E-8,
- “Local Bus Interface for MC68030 Application” on page E-11, and
- “Local Bus Interface for MC68040 Application” on page E-12.

## E.1 VMEbus Interface

### E.1.1 Buffered Signals

The SCV64 requires ten 8-bit transceivers (or five 16-bit transceivers) to buffer the VME address, data, address modifiers, and strobes. While always enabled, the direction of each of these transceivers is controlled by one of three signals generated by the SCV64 (VADDRROUT, VDATAOUT, and VSTRBOUT) as shown in Figure E.2 on page E-5.

Tundra recommends the use of '245 transceivers in the VMEbus interface. In particular 'F245, 'FCTAT245, and 'ABT245 may be used. The most commonly used component is the 'F245. These buffers are available in a wide range of packages from many vendors with up to 36 transceivers per package. The low profile of the TSOP package allows the mounting of these components on the backside of VME boards, so this package may be of particular interest to designers with limited real estate.

### E.1.2 Layout Issues

As with all high frequency designs, proper layout of your board is critical in order to eliminate crosstalk and minimize noise on the board. Glitches or crosstalk can occur on many of the VMEbus signals which may cause the SCV64 to behave in an undesirable manner. The following SCV64 signals are particularly sensitive to noise: the bus grant inputs, bus busy signal and the five highest VME address lines.

Noise or glitches which occur on the bus grant inputs usually originate from crosstalk induced on the board. In order to minimize crosstalk these tracks should be as short as possible (see Figure E.1) and should be routed on a separate layer. If a glitch occurs on the bus grant inputs the SCV64 can interpret this glitch as an additional bus grant, in which case you could end up with contention on the VMEbus. The SCV64 does not always de-glitch these signals in order to accelerate the propagation of a bus grant down the bus grant daisy chain.

The VME bus BBSY\* signal is inherently very noisy. The SCV64 internally de-glitches this signal on a rising and a subsequent falling edge of the C32MHz clock; however, care should still be taken to minimize the length of this track.

The third group of signals in which noise can impact the SCV64 is on the upper 5 VME address lines VADDR[31-27]. These signals are used by the SCV64 in decoding an A32 slave access and reside on the P2 connector along with other user defined signals. If your design involves the use of a secondary bus (i.e. the VSB bus) then you must take precaution in ensuring that the switching of these user-defined signals doesn't induce noise onto these five VME address lines, otherwise the SCV64 may not decode a slave access that was intended for it.

All of these potential problems mentioned above can be eliminated with proper layout techniques which begin with the proper placement of the SCV64 device (See Figure E.1). This orientation of the SCV64 placed closer to the P1 connector than the P2 connector will minimize the length of the VME signals to the SCV64's pins and should allow you to meet the "2-inch" rule of the VMEbus specification.



**Figure E.1 : Orientation of the SCV64 (CPGA and PQFP) to Minimize Distance to the Connector**

The following board stackup has been used successfully without any glitches or noise occurring on the above mentioned signals.

---Ground Shield

Signal These two layers are used to route the VME data lines from the connectors to the transceivers.

Signal

===VCC

Signal These two layers are used to route BBSY\*, BCLR\*, BGxIN\*, BGxOUT\*, BRx\*

Signal

---Ground



Signal    These two layers used to route the data lines from the SCV64 to the transceivers.  
 Signal  
 ---Ground Shield

As noted before, the BBSY\* signal is especially sensitive to picking up crosstalk on the PCB and therefore it is suggested that the BBSY\* track be routed at the very outside of its layer and never underneath any data buffers. Similarly, the four bus grant inputs should not be routed underneath any of the VME buffers. The BG3IN\* signal will likely have a pulldown resistor (refer to the next section) attached to this track. Care should be taken to minimize the length of the stub to this resistor in order to alleviate any noise on this bus grant input. The BG0IN\* signal can be used as an external reset input if the SCV64 is the SYSCON. If the SCV64 is being used in this manner then the length of the stub connection from this reset logic to the BG0IN\* track should be minimized as well.

A way to minimize the effect of the ground bounce which can be induced on the VME address lines by the undefined pins on the P2 connector switching at the same time is to separate the routing of these undefined pins from the VME address pins on your board. This can be done by routing the VME signals on one side of a power plane and the undefined P2 signals on the other side of the power plane. Other possibilities to reduce ground bounce would be to investigate using the new 5 row connectors which have added rows "z" and "d" to the connector. If your board was plugged into a backplane with the 5 row receptacles, these additional ground pins will reduce the ground bounce, and therefore the noise on the VME address lines.

### E.1.3    Decoupling VDD and VSS

In order to decouple low frequencies, a 22  $\mu$ F capacitor should be placed near the SCV64 between the  $V_{DD}$  and  $V_{SS}$  power layers. In order to decouple high frequencies, it is recommended that at least six 0.1  $\mu$ F bypass capacitors be distributed around the pairs of  $V_{DD}$  and  $V_{SS}$  pins on the device.

### E.1.4    BGxIN\*[3:0] Signals

While the BGxIN\*[3:0] signals are direct connects to the backplane bus grant daisy chain, these signals also serve special purposes for system controller configuration and determination. The SCV64's Auto-Syscon Detect mechanism relies upon BG3IN\* to determine its system controller status at system reset. If BG3IN\* is low immediately after system reset, then the SCV64 will automatically become the system controller. Otherwise, the system controller functions are disabled. This feature is part of the new VME64 specification, and is fully compliant with revision C systems.

To make use of Auto-Syscon Detect feature, the SCV64 requires a pull-down resistor of approximately  $4.7k\frac{3}{4}$  on the BG3IN\* signal. This will not in any way impact the performance of the bus grant daisy chain. With this resistor in place, no jumpers are required to configure system controller enabling.

When a card is configured as the system controller, the other three bus grant in signals, BGIN\*[2:0] , are typically unused. However, the SCV64 uses the BG0IN\* signal as an off card reset, while the two other signals may be used as off-card configuration inputs, monitored through internal registers.

**Figure E.2 : SCV64 Interface to VMEbus**

BG0IN\* will act as an active-low system reset signal when the SCV64 is the system controller. If the BG0IN\* signal is used in this capacity, then all SCV64 equipped cards must put a pull-up resistor on the BG0IN\* signal of approximately  $4.7k\Omega$ .

BG1IN\* and BG2IN\* are configuration inputs only, monitored through the STAT1 register. Since they are non-functional signals, these signals do not need to have pull-ups or pull-downs on them, and may be left directly connected to the VMEbus. Alternatively, they may be connected to chassis mounted switches.

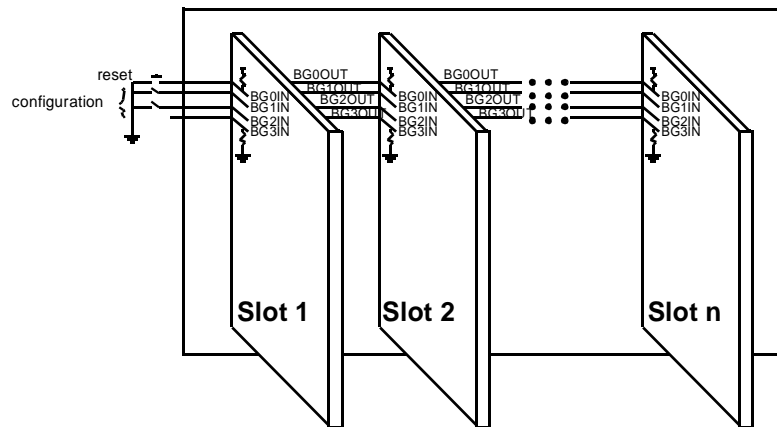


Figure E.3 : Bus Grant Daisy Chain Usage in System

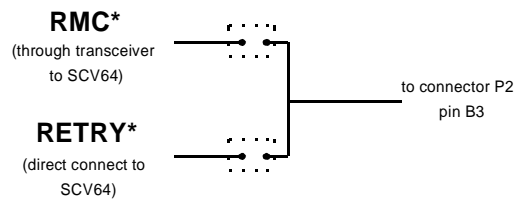
### E.1.5 RETRY\*/ $\overline{\text{VRMC}}$ Pin

The SCV64 has a dual purpose pin, RETRY\*/ $\overline{\text{VRMC}}$ , which may be used as either the VME64 specification RETRY\* line, or may be used as a Tundra proprietary read-modify-write (RMW) cycle pin. The configuration of the RETRY\*/ $\overline{\text{VRMC}}$  pin is controlled by the RMCPIN bit in the MODE register, and defaults to use as the  $\overline{\text{VRMC}}$  pin.

When configured as the  $\overline{\text{VRMC}}$  pin, the signal is a bi-direct and must be buffered through a transceiver with direction controlled by VSTRBOUT. One of the unused inputs on the strobe transceiver may be used for this purpose. When configured as RETRY\*, the signal acts only as an input. It may be directly connected to the backplane.

RETRY\* is defined in the new VME64 specification as occupying the B3 (formerly reserved), on the P2 connector. However, many current implementations of  $\overline{\text{VRMC}}$  use that same pin. To maintain compatibility with other board vendors using Tundra's VME interface components, it may be necessary to continue to use this pin, in violation of the new definition of this pin in the VME64 specification. Note that this will only be possible if P2 pin B3 is not being used as the RETRY\* in other parts of the system.

To provide as much flexibility as possible, many board vendors choose to use a jumper option to provide the capability of having the pin operate as either the  $\overline{\text{VRMC}}$  pin on P2 pin B3, or as RETRY\*. This jumper either takes the buffered version of the signal through the transceiver for  $\overline{\text{VRMC}}$ , or the direct connect for RETRY\* (See Figure E.4 below).



**Figure E.4 : Implementation of  $\overline{\text{VRMC}}$  and RETRY\***

Note that if either  $\overline{\text{VRMC}}$  or RETRY\* is implemented on P2 pin B3, that this signal should be properly terminated on the backplane. VME64 style backplanes will already have provision for the termination of this line. If termination is not provided on the backplane, ensure that there is termination on-board to pull RETRY\*/ $\overline{\text{VRMC}}$  to the inactive state.

If neither  $\overline{\text{VRMC}}$  or RETRY\* is to be implemented on the board, ensure that the RETRY\*/ $\overline{\text{VRMC}}$  is pulled to the inactive state (high) to ensure proper function.

### E.1.6 BI-Mode and IRQ1\*

The SCV64's BI-Mode feature may be triggered through four mechanisms:

- reset,
- assertion of the  $\overline{\text{BITRIG}}$  pin,
- software (through the SBI bit), and
- assertion of IRQ1\* (when configured as such).

IRQ1\* defaults upon reset to being a BI-Mode initiator, which means that the SCV64 will enter BI-Mode whenever IRQ1\* is asserted. When configured as BI-mode initiator, IRQ1\* triggers a local interrupt on level 7 (if so enabled through the BIE bit in the 7IE register). When configured as an interrupt source, a local interrupt on level 1 is generated upon assertion of IRQ1\* if that level is enabled in the VIE register.

To reconfigure this signal as an interrupt only, the VI1BI bit in the GENCTL register should be cleared, typically as part of a reset routine.

## **E.2 Local Bus Interface for Slave Only Applications**

The SCV64 is intended for use both in applications where a local processor is provided and in applications where no local intelligence processor is provided. This is achieved primarily through a power-up option enabling the device to automatically initialize all internal registers required to enable access to the VMEbus.

### **E.2.1 Initialization**

All internal register bits required for VMEbus operation are set automatically when the AUTOBAR power-up option is enabled. This option is enabled by pulling KFC2 high and KFC0 low at the rising edge of  $\overline{\text{PWRRST}}$ . At this time, the VMEBAR register is loaded with the value on the local data bus and all required internal bits are set to enable slave access to the device.

### **E.2.2 VMEbus Programming**

In conjunction with the automatic initialization of the device, additional register accesses may be required to complete initialization of the device or to reprogram functionality. This is achieved by programming any additional internal registers from the VMEbus. Access to the internal registers is made through the slave image programmed during automatic initialization of the device. The SCV64 will generate a cycle on the local bus of the SCV64 in the same way as an access to local memory or any other local bus device. This cycle must be decoded to generate a chip select signal for the SCV64 ( $\overline{\text{SCV64SEL}}$ ). The SCV64 accepts the register access from the local bus and terminates the local cycle with  $\overline{\text{KDSACK}}$ . Access to the local bus for register programming may be through either A24 or A32 addressing spaces.

An example circuit diagram is provided. In this circuit, an addressing decoding block decodes the select signal for the SCV64 ( $\overline{\text{SCV64SEL}}$ ) in addition to any other select signals required on the local bus. A transceiver in conjunction with user defined programming (resistors or jumpers) provides A24 programming bits for the VMEBAR register during the rising edge of PWRRST. 7 bits are shown which select the A24 base address and size. All unused A32 base address and size bits are left undefined in this example. VMEBAR register bits may be left undefined provided that system design and initialization are unaffected by undefined A24 or A32 images. In this case, all initialization must be completed using the defined A24 address space. Alternatively, logic could be provided to program both A24 and A32 base addresses and sizes during power-up.

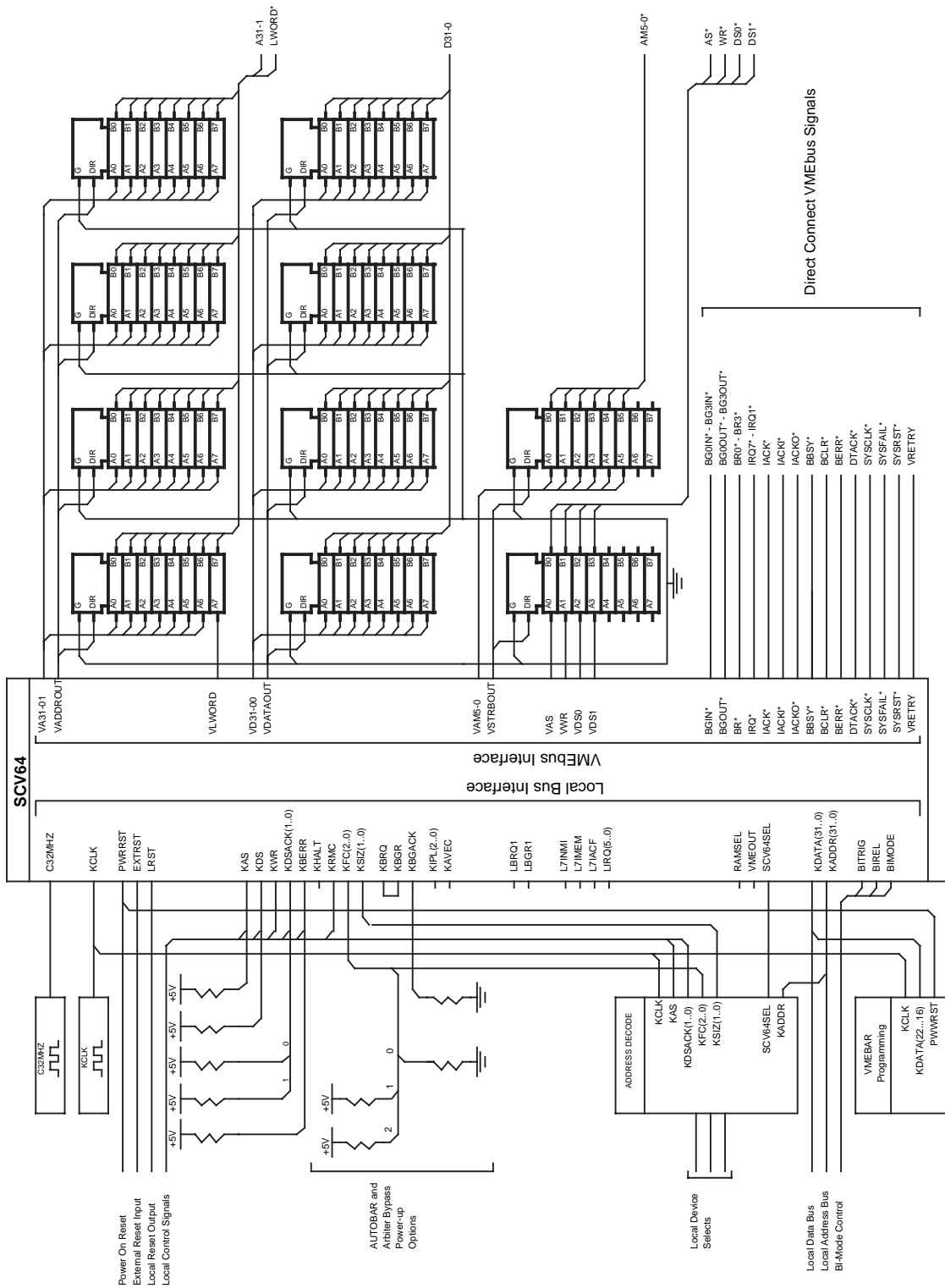


Figure E.5 : Connections for Slave Only Design



## E.3 Local Bus Interface for MC68030 Application

A circuit diagram is provided in Figure E.6 below which illustrates local connections for a MC68030 application.

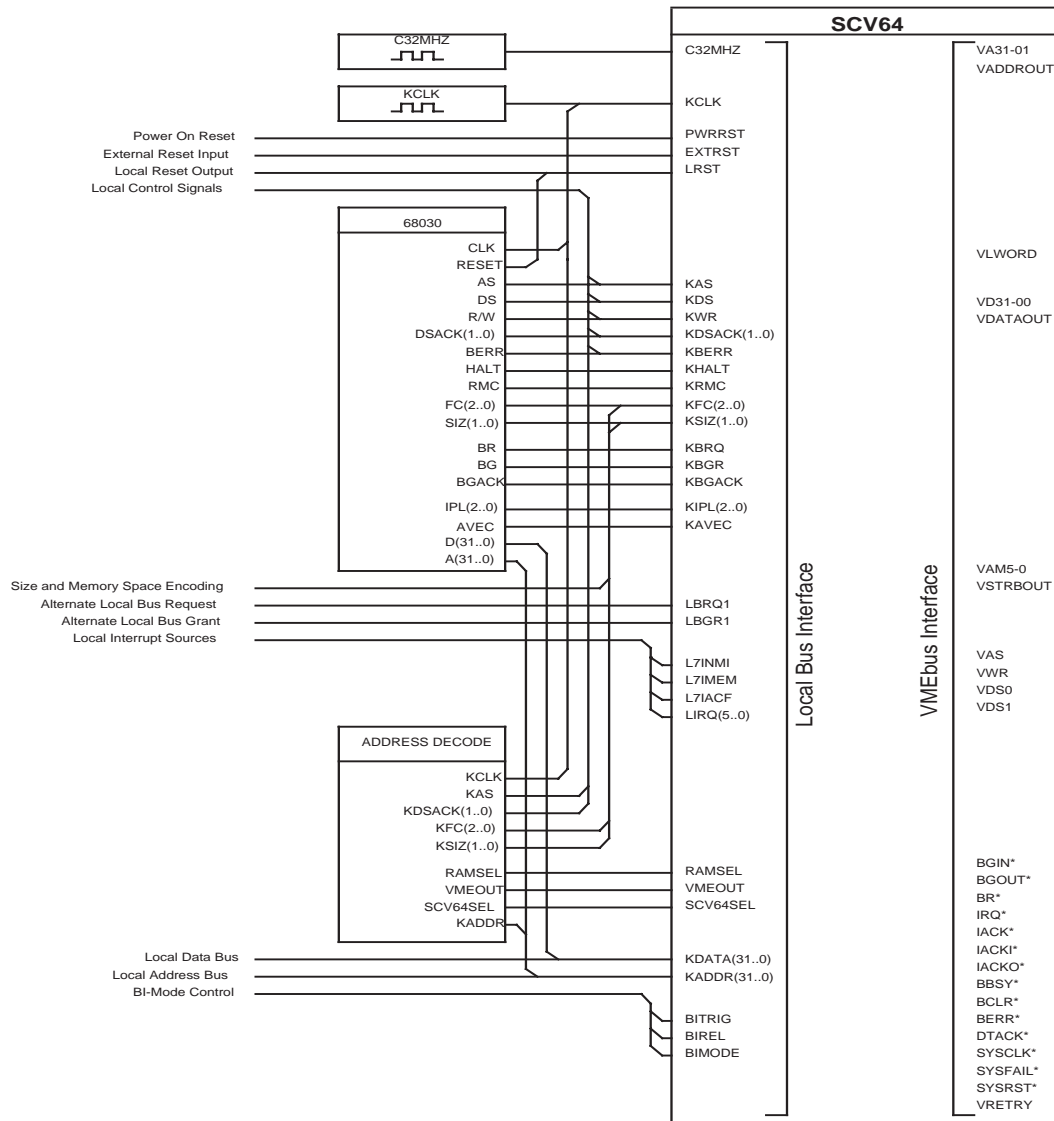


Figure E.6 : Connections for 68030 Design

## E.4 Local Bus Interface for MC68040 Application

The SCV64, while designed with a MC68030 compatible interface, lends itself well to a variety of other processing environments. Because of its similarity to the MC68030, the MC68040 is particularly suited among processors for use with the SCV64 in a multi-processing VME environment. This application note provides an overview of one possible implementation of a SCV64/MC68040 interface design.

Over the past decade, the MC68000 family of processors has dominated VME board design. This is in large part due to Motorola®'s market dominance in the field, but also to the similarities between the VMEbus and a 680x0 style bus. While other processors, particularly RISC processors, are growing in importance within VME applications, 680x0 boards still predominate because of their relative affordability and availability.

The prevalence of 680x0 hardware entails the prevalence of 680x0 software in VME applications. Taking advantage of this investment in software is a major factor in choosing processors for a new VME design. Every member of the 680x0 family, including the 68040, is object-code compatible with each other. Therefore, the significant investment in software need not be duplicated when upgrading a system's performance. Upgrading a 68030 system to 68040 significantly increases performance at minimum cost.

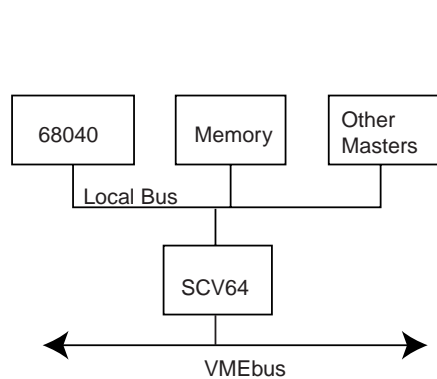
Although upgrading to a 68040 improves a board's raw processing power, these gains may be lost on communication and data transfer unless the system bus is optimized. VMEbus can be a powerful tool in optimizing the system throughput, but the bus is only as powerful as its interface.

The SCV64 provides that powerful interface. Its internal FIFO architecture effectively decouples bus performance, allowing both the VMEbus and local CPU bus to operate at their peak capacities. In addition, the integral DMA allows high speed data transfer between boards at close to the theoretical bus limit.

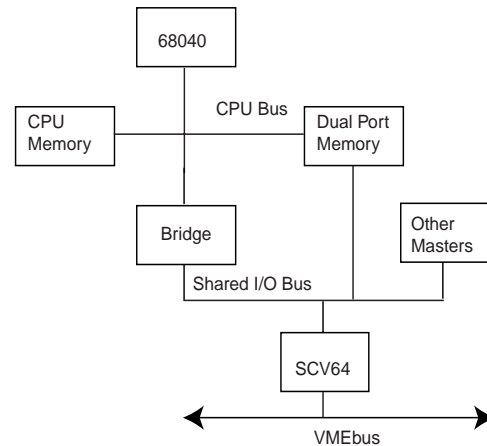
### E.4.1 Design Philosophy

In this design, the SCV64 and the CPU occupy bandwidth on the same local bus. This implies that there are two masters (the SCV64 and the CPU) on the local bus, and possibly more if the application demands it. The address, data, control, and interrupt facilities on the local bus are shared. As such, the SCV64 must compete with other masters for access to local resources. The basic layout for this architecture is shown in Figure E.7 below.

A different approach involves a hierarchical structure, as in Figure E.8, where the CPU resides on its own bus. A bridge allows the CPU to access an I/O bus from which the CPU can access the VMEbus. The advantage of such a scheme is that activities occur simultaneously on the I/O bus and on the CPU bus. The implementation of a hierarchical system is not discussed in this manual. The present discussion focuses on the scheme shown in Figure E.7.



**Figure E.7 : Shared Local Bus Structure**



**Figure E.8 : Hierarchical Local Bus Structure**

To make this application as general as possible, cache coherency protocols are not discussed. Therefore, the flow of cycles is either from the CPU to the VMEbus or memory, or from the SCV64 to memory. SCV64 cycles are never translated into equivalent CPU cycles for snooping requirements. Thus, the adapter design translates cycles only in the direction of the CPU to the SCV64.

## E.4.2 Design Overview

There are four major components of the bus adapter design that are discussed here: arbitration, interrupts, address space mapping, and the cycle translation unit.

The arbitration unit provides the design for a typical arbiter. It arbitrates only between the SCV64 and the CPU, although other masters may be added. The arbitration follows a simple priority scheme, where the CPU has the greatest priority on the bus. Ownership of the bus is granted to the SCV64 only when the bus is idle. If the SCV64 has ownership of the bus when the CPU requests it, the SCV64's ownership is immediately rescinded. The SCV64 typically gives up the local bus within one or two cycles after ownership rescission. The priority scheme also implies that locking capability is built-in. The arbiter will never remove ownership of the bus from the CPU, and during lock sequences, the SCV64 will hold the bus despite rescission of its grant.

The interrupt unit adapts the MC68040 interrupt acknowledgment cycle to an equivalent cycle for the SCV64. Note that the SCV64's internal interrupt handler can be used to provide efficient interrupt management. The SCV64 can monitor up to six general purpose interrupt sources as vectored or auto-vectored, all seven VME interrupts, as well as a non-maskable interrupt, SYSFAIL\*, and ACFAIL\*. All of these interrupt sources can be seamlessly adapted to MC68040's interrupt capabilities.

The address mapping and decode unit provide access to multiple VME address spaces. Access to A16, A24 and A32 spaces is provided through the SCV64's internal address decode mechanisms, while access to program/data and supervisory/non-privilege address spaces is mapped directly from the CPU's equivalent address spaces indicated on the Transfer Mode lines. The address decode unit also provides access to the SCV64's internal registers.

The fourth unit within the bus adapter is the cycle translation unit, providing mapping of MC68040 cycles into equivalent SCV64 cycles. The unit implements a state machine to map CPU read and write cycles into single wait state VME accesses. It handles bus errored, deadlock, as well as auto-vectored terminations. While dynamic bus sizing is provided as a programmable option within the SCV64, it is not implemented by the MC68040. Therefore, it is not available with this design. If dynamic bus sizing is a requirement in the system, then the designer may wish to refer to the *MC68150 Dynamic Read/Write Bus Sizer* documentation available from Motorola®.

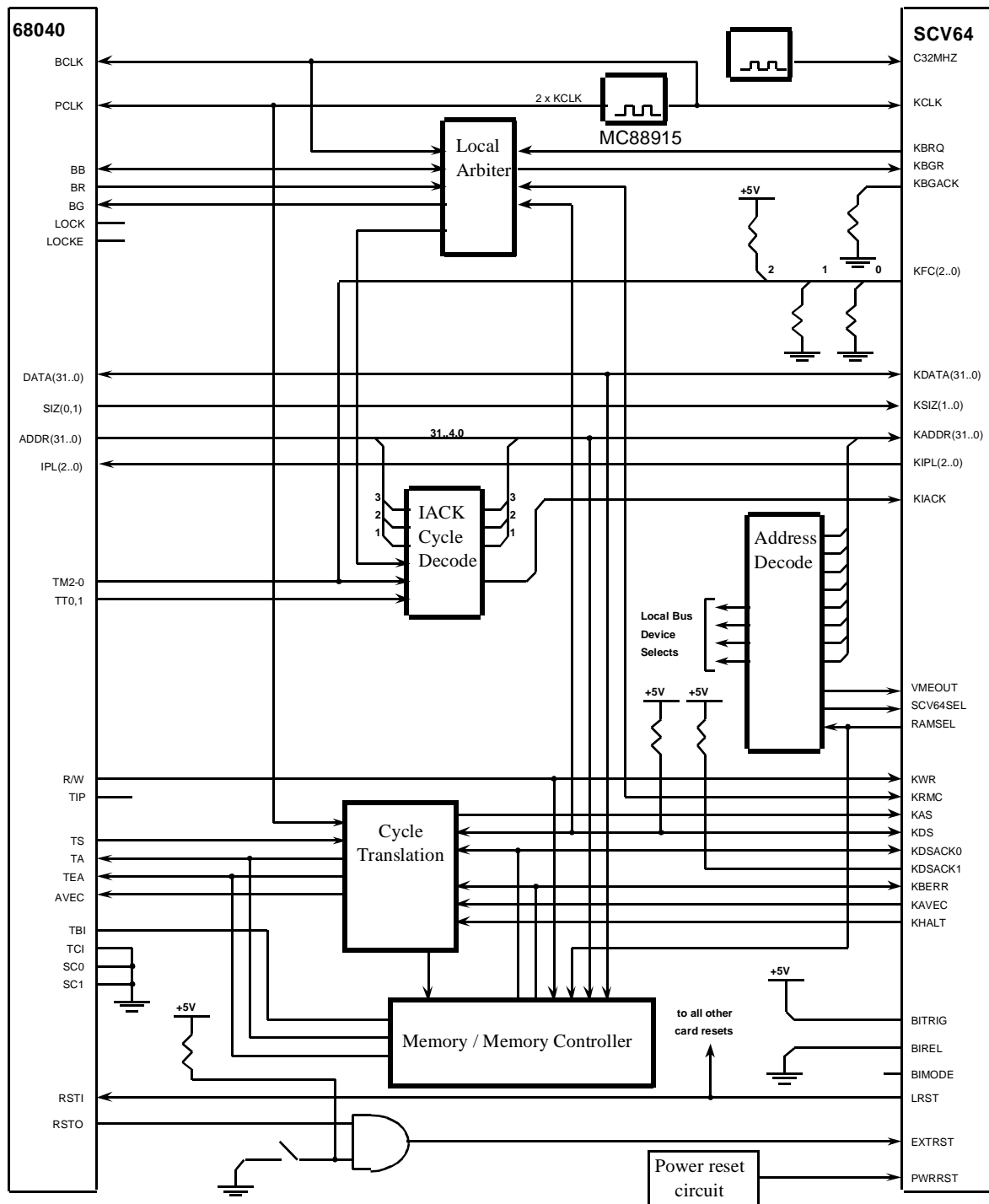


Figure E.9 : MC68040 to SCV64 Bus Adapter Block Diagram

### E.4.3 Bus Adapter Operation

#### Cycle Translation Unit

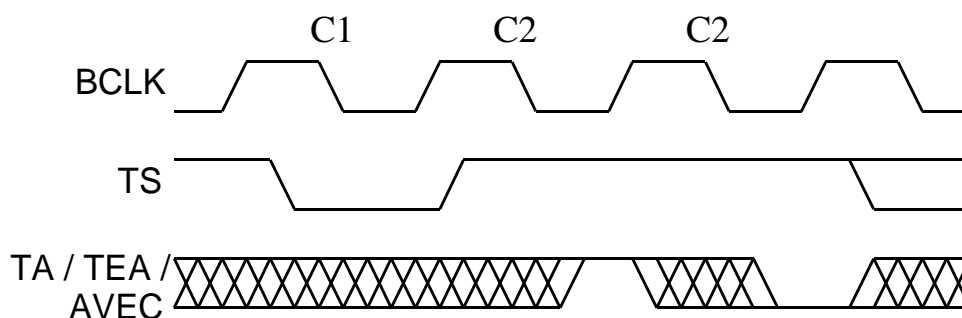
The main unit of the bus adapter maps the 68040 strobe and termination signals to the equivalent SCV64 signals. This mapping is performed through a state machine running off the rising edge of the processor clock, PCLK.

#### SCV64 and MC68040 Read/Write Cycle Overview

The 68040 has a minimum two clock cycle composed of states C1 and C2. In C1, TS is asserted by the 68040, with setup to the falling edge of BCLK. The processor then goes into state C2, sampling the termination signals TA, TEA, and AVEC on the falling edge of BCLK. Data is considered valid by the 68040 on the edge that TA is asserted. If none of these signals is sampled low, the processor enters wait states. The significance of each of the termination combinations is shown in Table E.1.

**Table E.1 : 68040 Terminations**

TA	TEA	AVEC	Termination Description
0	1	1	successful
1	0	1	bus error
0	0	1	deadlock retry
0	1	0	auto-vectored IACK cycle



**Figure E.10 : MC68040 Bus Transfer Control**

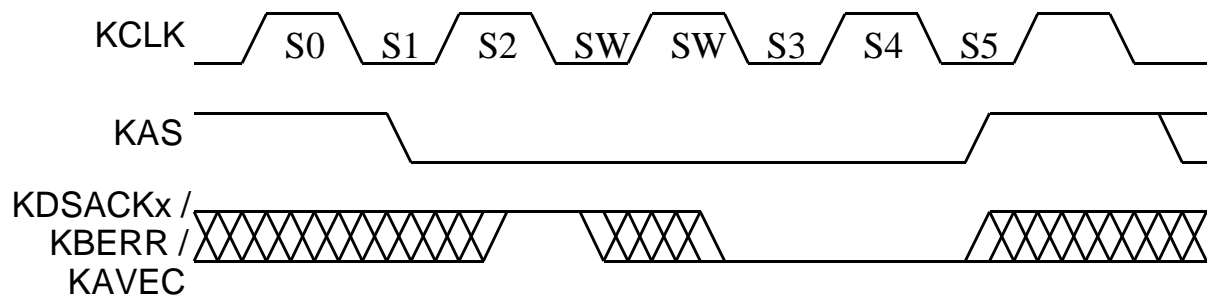
The SCV64 has a minimum three clock cycle composed of states S0 through S5, following the 68030 convention. The address strobe, AS, is typically asserted in S1. However, as a slave (i.e. when the CPU is accessing internal registers or the VMEbus) the SCV64 only examines its strobe, KAS, on the falling edge of S2. If KAS and either of  $\overline{\text{SCV64SEL}}$  or VMEOUT are asserted, the SCV64 becomes the local slave.

The SCV64 terminates slave accesses by asserting a combination of KDSACK0, KDSACK1, KBERR, KAVEC, and KHALT. Each combination indicates a different type of termination (See Table E.2). If none of KDSACKx, KBERR, or KAVEC is asserted, the CPU is expected to insert wait states.

**Table E.2 : SCV64 Terminations**

KDSACK 0	KDSACK 1	KBERR	KAVEC	KHALT	termination description
0	1	1	1	1	16-bit successful
0	0	1	1	1	32-bit successful
1	1	0	1	1	bus error
1	1	0	1	0	deadlock retry
1	1	1	0	1	auto-vectored

Each of the termination signals may be considered asynchronous. As such, the local master should double-sample the termination signals on the falling edge of KCLK. Data is generally considered valid during the second sample, although during register write cycles the SCV64 may latch the data on the first sample.



**Figure E.11 : SCV64 Bus Transfer Control**

#### E.4.4 Cycle Translation State Machine

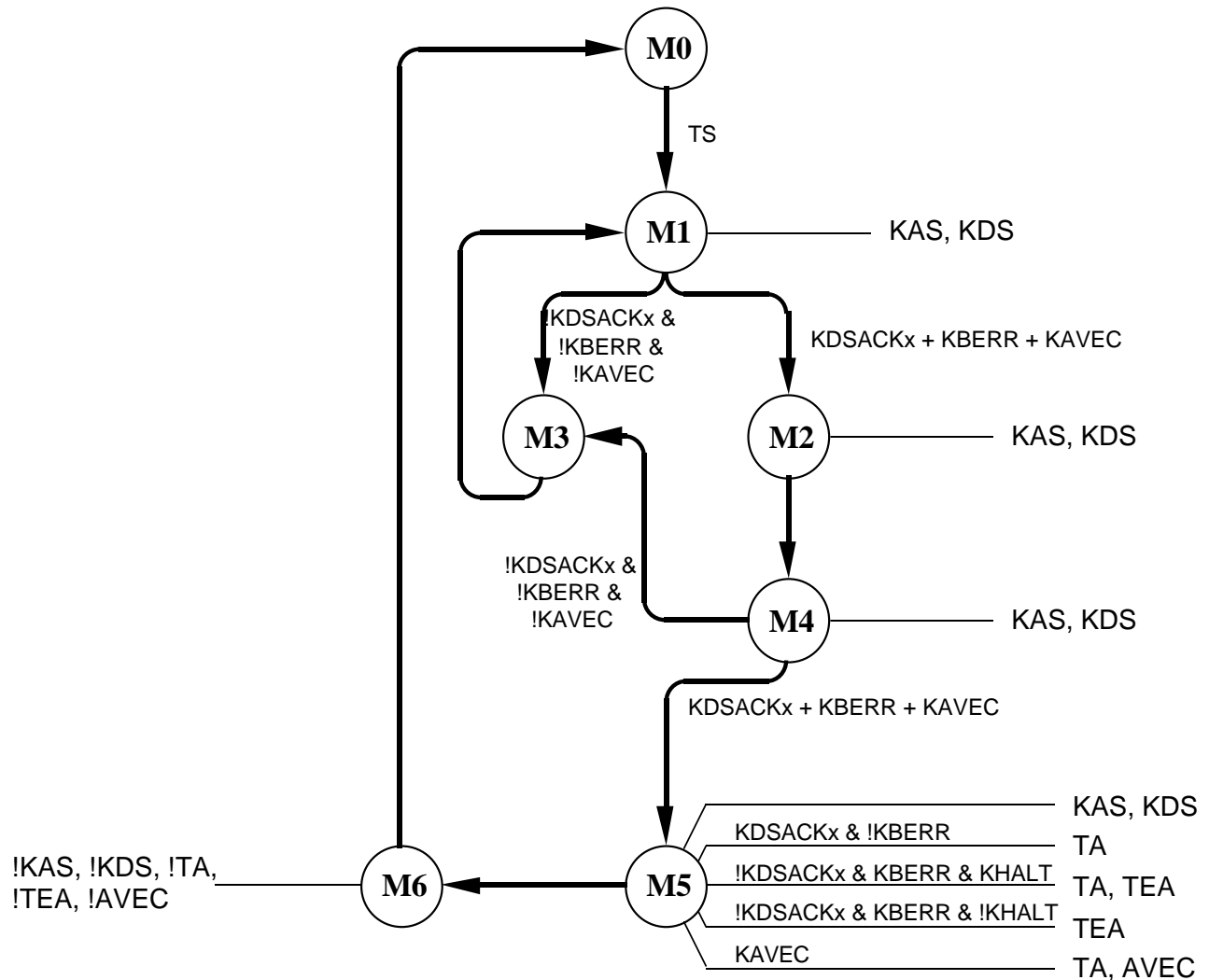
The 68040 read and write cycles are mapped to SCV64 cycles by the state machine in Figure E.12. While both the SCV64 and the 68040 use the same bus clock (KCLK and BCLK), the 68040 processor clock (PCLK) is used as the state machine clock. PCLK is double the frequency of the bus clock. The processor clock is used since state transitions may occur on either edge of the bus clock.

The Cycle Translation state machine sits in the idle state, M0, until the 68040 asserts its TS signal, when it enters M1. In this state, the machine starts the cycle to the SCV64 by asserting KAS and KDS. Unlike the standard 68030 protocol, KDS is not required on the next clock during write cycles. The SCV64 ignores that signal until later in the cycle.

If none of the terminations is returned from the SCV64 before the next state transition, then the machine progresses to state M3, and then back to M1. This process continues until one of the termination signals has been sampled low on the falling edge of KCLK/BCLK. When this occurs, the machine progresses to state M2, and then M4 on the next transition.

If the terminations are not still active at the end of M4, then the machine moves back to state M1 to begin the double sampling process again. Otherwise, one of the terminations has been active for two sample periods, and the state machine moves to state M5, where it terminates the 68040 cycle by asserting the appropriate cycle acknowledge to the 68040. The type of acknowledge generated depends upon the state of the SCV64 termination lines (see Figure E.12 below).





**Figure E.12 : Cycle Translation Unit State Machine**

The SCV64 latches the data on the data lines during the second sample of the terminations at the end of M4. The 68040 will latch the data at end of the next state M5 when TA is asserted. Note that to maintain valid data until the end of M5, the state machine keeps KAS asserted an extra half clock. (This violates the MC68030 specification.)

After M5, the state machine enters M6, where it drives all outputs high, including TA, TEA, AVEC, KAS, and KDS. It is not until it moves back to the idle state during the next transition that these signals are tri-stated. Some signals may not need to be tri-stated if they are not being used as open-collector signals.

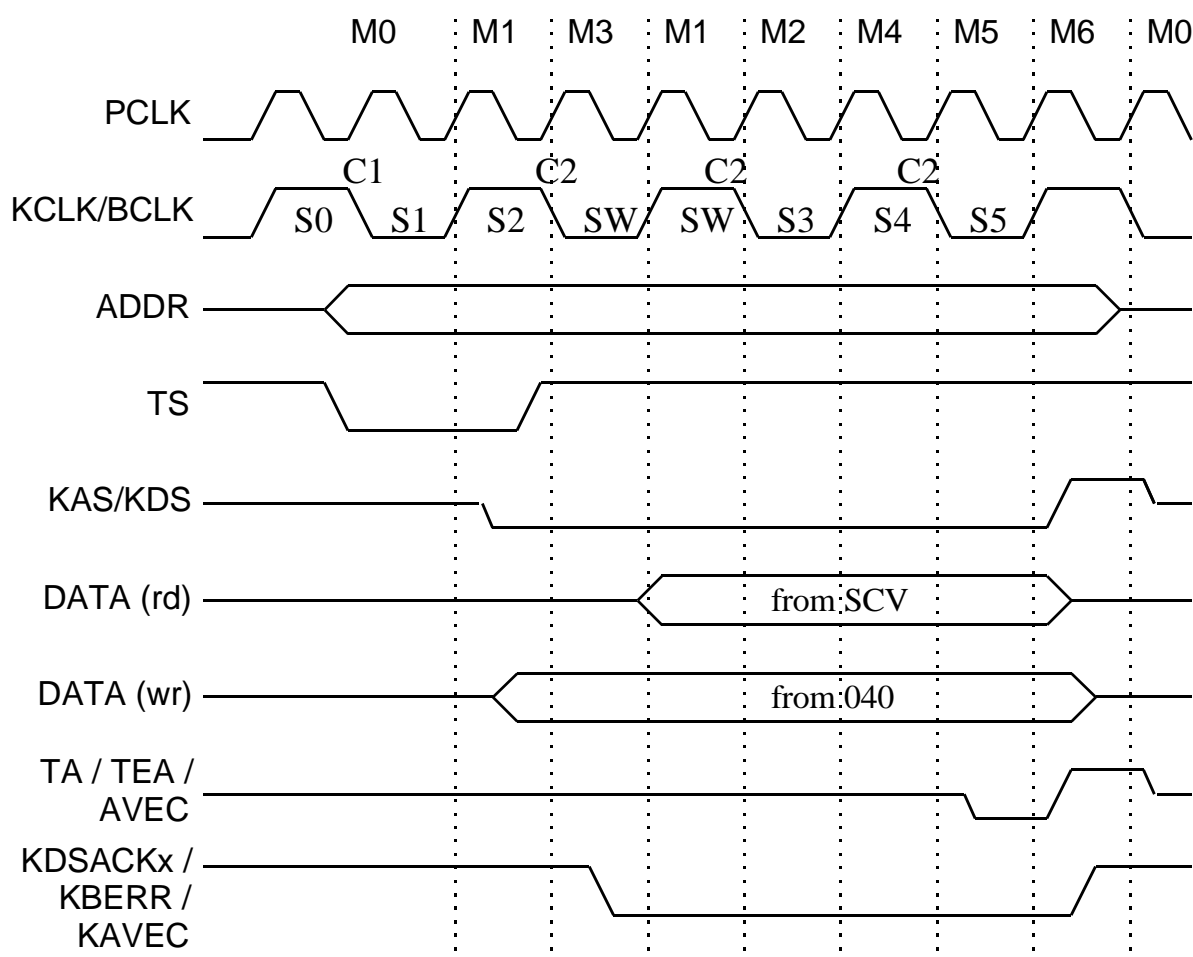


Figure E.13 : Cycle Translation Timing Diagram

### E.4.5 Dynamic Bus Sizing

Dynamic bus sizing is the capability of a CPU to transparently break a data transfer into multiple transfers based upon the size of the destination port. While the SCV64 does provide a capability to request the CPU to break-up a cycle, the 68040 does not recognize this protocol. The SCV64 can request that the CPU break an unaligned transfer into aligned 16- or 8-bit transfers, or to force 16-bit transfers to D16 VME boards. However, the 68040 expects that the addressed port is capable of accepting the current transfer, thus transferring the responsibility for bus sizing onto software.

The SCV64 can be programmed to disable its dynamic bus sizing capability, matching it with the 68040. This is done through setting the BUSSIZ bit in the MODE register. When the BUSSIZ bit is set, the SCV64 responds with a 32-bit KDSACK (both KDSACK lines asserted).

With dynamic bus sizing disabled (BUSSIZ = 1), only one of the two KDSACK lines need to be fed into the cycle translation state machine: KDSACK0.

## E.4.6 Transfer Attributes

### Transfer Type

The Transfer Type lines on the 68040 are driven by the CPU to indicate the type of access of the current bus cycle (see Table E.3). Only two of the four possible encodings are of interest in a SCV64 application. These are Normal Access, and Acknowledge Access. The Acknowledge Access can be used to generate the SCV64's interrupt acknowledge signal (see the section on Interrupts). Normal Access bus cycles should be used exclusively for SCV64 accesses. As such, the address decoder (see the section on the Address Decoder) can use the TT1,0 lines to generate  $\overline{\text{SCV64SEL}}$  and VMEOUT.

**Table E.3 : 68040 Transfer Type Encoding**

TT1	TT0	Transfer Type
0	0	Normal Access
0	1	MOVE16 Access
1	0	Alternate Logical Function Code Access
1	1	Acknowledge Access

Since the SCV64 can not accept burst cycles as the local slave, all MOVE16 bus cycles should be either bus errored, or allowed to time-out with the bus timer.

As an option, the Alternate Logical Function Code Access may be used by the address decoder to redefine the address map. One application may be to use this code exclusively for VMEbus accesses.

### Transfer Modifier / Function Codes

The Transfer Modifier lines on the 68040 are driven to indicate the data space being accessed. The definition of these lines change depending upon whether or not the current bus cycle is an Alternate Function Codes Access. In either case, the lines may be mapped one for one to the SCV64 Function Code lines KFC[2:0].

The Function Code lines are used by the SCV64 in determining the Address Modifiers on the VMEbus. With the TM2-0 lines connected directly to the KFC2-0 lines, address spaces on the 68040 bus are mapped to equivalent spaces on the VMEbus (See Table E.4).

**Table E.4 : Local and VME bus Space Mapping**

68040 Space	VMEbus Space
User Data	Non-privileged Data
User Program	Non-privileged Program
Supervisor Data	Supervisory Data
Supervisor Program	Supervisory Program

## Transfer Size

The Transfer Size lines on the 68040 are defined almost identically to those on the SCV64 (KSIZ0, KSIZ1): both indicate the size of the current bus cycle. The SCV64 and the MC68040 differ in the case where KSIZ1 = 1 and KSIZ0 = 1. In this case, the SCV64 indicates this is a 3-byte transfer and the 68040 indicates this is a line transfer. The Transfer Size lines may be directly connected between the two devices; however, the SCV64 does not support line read or line write cycles from the 68040.

## Bus Arbitration

The local bus arbiter of the bus adapter provides arbitration between the SCV64 and the 68040. The arbiter implemented here is a priority arbiter, where the 68040 has priority over the SCV64. The SCV64 is forced to give up the local bus either when it is done, or when the 68040 has a request pending.

The SCV64 has two arbitration protocols which are selected at power-up. The three-wire protocol provides 68030 style arbitration, while the two-wire protocol is more general, and intended for non-68030 applications. The bus arbiter described here uses the two-wire protocol (KBGACK must be low during power-up).

The arbiter starts in state M0, the idle state. If it receives a request from the SCV64, and there are no requests from the 68040, the arbiter progresses to state M3 where it grants the bus to the SCV64 and asserts Bus Busy to the 68040.

The grant to the SCV64 can not be removed until the first cycle has started. This is to ensure that the SCV64's request line will negate when the grant is negated later. Therefore, the SCV64 is only recognized as bus owner once it has asserted KAS. Once in state M4, the arbiter continues SCV64 bus ownership until one of two events occur: either the SCV64 no longer requires the bus (as indicated by its request line and KAS negating), or the 68040 requires the bus (indicated by its request line asserted).

If the SCV64 no longer requires the bus, the arbiter proceeds back to the idle state, waiting for the next request. This transition may be removed if desired, implementing a release-on-request scheme instead.

If the 68040 requests the bus and the SCV64 is not performing a read-modify-write cycle, the SCV64's grant is removed, and the arbiter proceeds to state M5. In state M5, the arbiter grants the bus to the 68040 and waits for the SCV64 to complete its bus tenure. Note that the 68040 does not actually begin its tenure until the arbiter releases the bus busy (BB) signal and proceeds to state M1.

Once in M1, the 68040 obtains bus ownership when the arbiter releases the BB signal. Unless a deadlock condition is indicated by the SCV64 (KHALT asserted), the arbiter maintains the grant to the CPU while the CPU's request signal is still valid. While the CPU has bus ownership, the arbiter asserts 040\_master, which is used by the IACK logic block to determine when to drive address lines.

Since the CPU maintains ownership until it has no further cycles to perform, there is no requirement to implement the 68040's Lock signals, LOCK and LOCKE in this arbiter design. If some other design is to be implemented, these signals may be required.

Deadlock situations are handled through state M2. If the 68040 is the current bus master (i.e. the arbiter is in state M1) attempting a VME access, and KHALT is asserted by the SCV64, the arbiter removes the CPU's bus grant and proceeds to state M2. The arbiter remains in M2 until the SCV64 requests the bus and the CPU releases the bus.

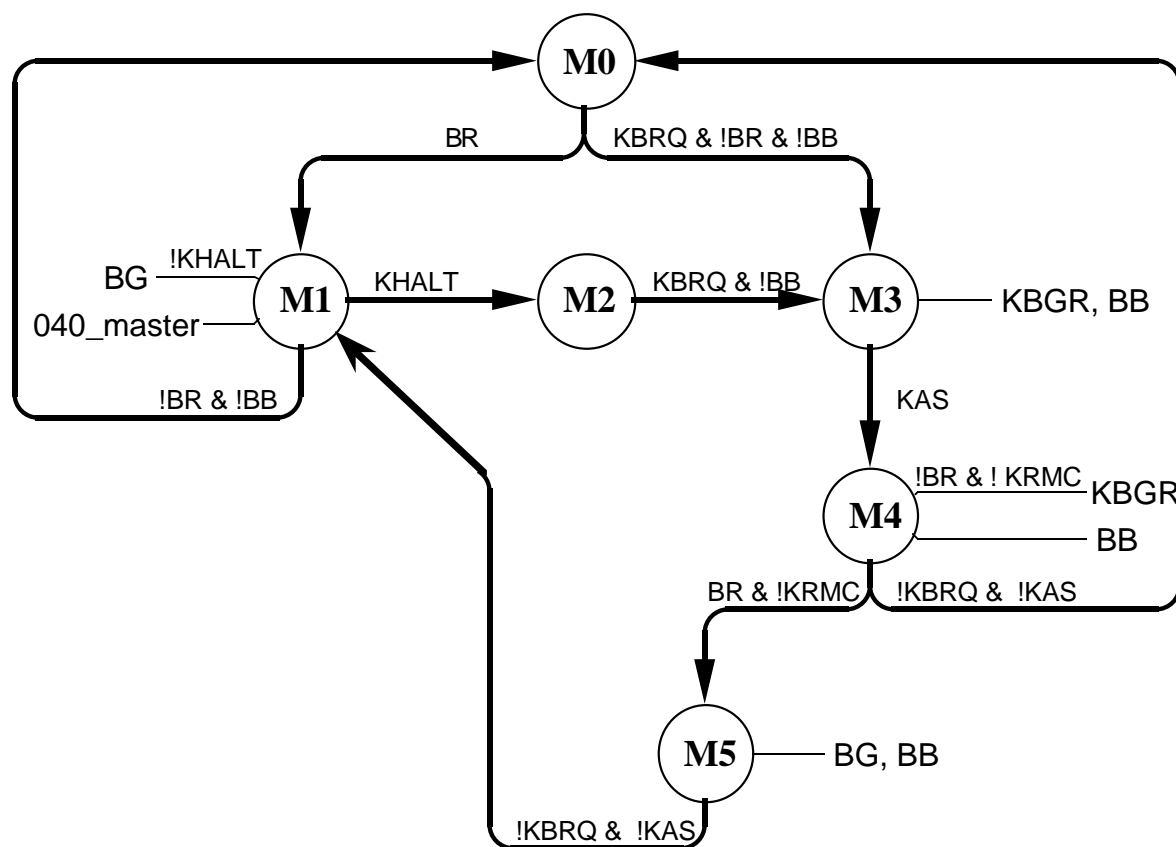


Figure E.14 : Local Bus Arbiter State Machine

## E.4.7 Interrupts

### Generation

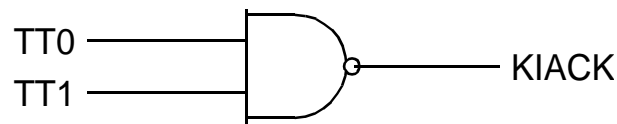
Since both the 68040 and the SCV64 use the same encoding of interrupts on the Interrupt Priority Level lines (IPL0-2), these lines can be directly connected between the two devices.

### Acknowledge Cycles

The major difficulty in interfacing a 68040 interrupt acknowledge cycle is identifying the interrupt level being acknowledged. The SCV64 expects the interrupt level to be encoded on KADDR[3:1], but the 68040 encodes the level on its Transfer Modifier lines, TM0-2.

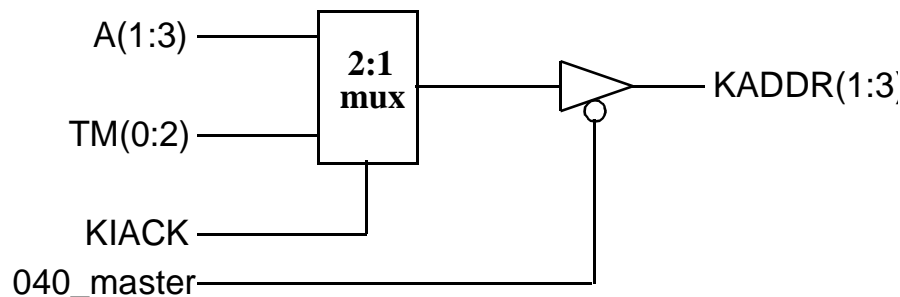
The Interrupt Acknowledge block of the bus adapter must detect that a 68040 IACK cycle is being performed and indicate that to the SCV64. The Transfer Type signals on the 68040 indicate that an IACK cycle is being performed, while the KIACK input performs the same function from the SCV64's perspective. Note that the SCV64 has two IACK modes for internal and external decoding. The internal IACK decoding works with 68030 style IACK cycles. For non-68030 IACK cycles, the SCV64 is set for external decoding mode (KFC1 held high at power-up). In external mode, the KIACK pin is used to select IACK cycles..

The generation of KIACK to the SCV64 can be generated directly from a decoding of the TT0-1 as in Figure E.15. It is not necessary to qualify KIACK with any other signal since the SCV64 performs its own qualification of the signal with KAS, generated by the cycle translation state machine.



**Figure E.15 : KIACK Generation**

The low address lines (KADDR[3:1]) must be buffered in order to be used to encode the interrupt level. KADDR[3:1] must be driven with the 68040 address lines (A[3:1]) during standard bus cycles, but with the Transfer Modifier lines (TM[2:0]) during IACK cycles. A signal from the arbiter is also required in order to indicate when the 68040 is the current bus master.



**Figure E.16 : IACK Cycle Interrupt Level Generation**

## Address Decoder

The address decoder provides chip-selects to various devices on the local bus, including the SCV64. The SCV64 requires two chip-select signals:  $\overline{\text{VMEOUT}}$ , and  $\overline{\text{SCV64SEL}}$ . Asserting  $\overline{\text{VMEOUT}}$  initiates VME master accesses, while  $\overline{\text{SCV64SEL}}$  selects the SCV64's internal registers. While many chip-selects are typically required to be qualified by an address strobe, this is not the case for the SCV64. It internally qualifies  $\overline{\text{VMEOUT}}$ ,  $\overline{\text{SCV64SEL}}$ , and  $\text{KIACK}$  with  $\text{KAS}$ . Hence these signals may be directly decoded from the address and possibly the Transfer Type and Transfer Modifier lines. (See the section on Interrupts for details on generation of  $\text{KIACK}$ .)

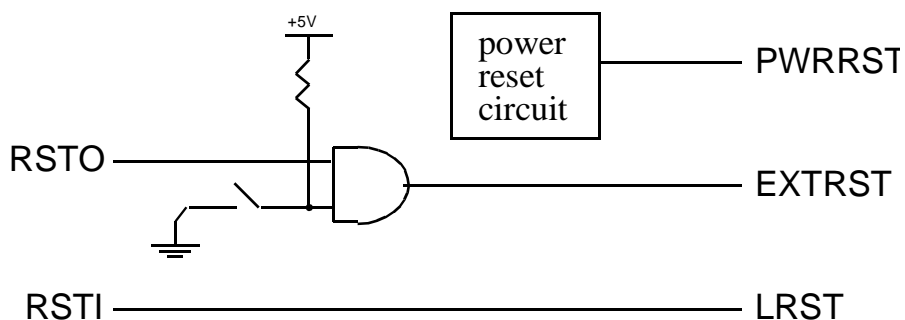
Details on decoding these signals are not provided here since they are application dependent. However,  $\overline{\text{VMEOUT}}$  is usually asserted whenever no other device on the local bus is selected. This ensures the local bus has maximum access to VMEbus resources.

$\text{RAMSEL}$  is asserted by the SCV64 when it is accessing local resources. It can be used as part of the decode scheme to provide a different perspective on the local memory map from the VMEbus. See “Memory Mapping” on page 2-45 for more information about address decoding

## Resets

The SCV64 has an internal reset mechanism for sorting several resets on the board. It has a  $\text{PWRRST}$  pin for reset on power-up,  $\text{EXTRST}$  for locally generated resets,  $\text{SYSRST}$  for VMEbus generated resets, and  $\text{LRST}$ , an output for resetting all local devices.

The 68040 has a reset output pin,  $\text{RSTO}$ , and a reset input pin,  $\text{RSTI}$ . The various elements of the reset scheme can be connected as in Figure E.17 below.



**Figure E.17 : Reset Circuit**



### BI-mode Mechanisms

Many designers may find that they do not require use of the SCV64's BI-mode feature. If this is the case, disable BI-mode, by tying  $\overline{\text{BITRIG}}$  high, and BIREL low. This ensures that the SCV64 will come out of BI-mode immediately after reset. The VI1BI bit in the GENCTL register should also be cleared to reconfigure IRQ1\* as a VME interrupt instead of a BI-mode initiator. (See “BI-Mode Effects” on page 2-63 and on page 2-119 for more details on BI-mode.



## Appendix F Initialization

### F.1 Hardware Configuration

The following describes various hardware issues that require attention during initialization.

#### F.1.1 Power-Up Modes

The following SCV64 pins must be at the levels given in Table F.1 during the rising edge of PWRRST to ensure that the device comes up in a defined state.

**Table F.1 : Pin Levels for Specific Power-up Modes**

Pin	Sample Point	Level	Power-up Mode
KBGACK	PWRRST	low high	internal arbiter bypassed internal arbiter implemented
KFC2, KFC0	PWRRST	low, low  low, high high, high high, low	reserved  reserved VMEBAR loaded with KDATA, not enabled VMEBAR loaded with KDATA, enabled
KFC1	PWRRST	low high	KIACK/LIRQ2 configured as LIRQ2 pin KIACK/LIRQ2 configured as KIACK pin
KDS <sup>a</sup>	PWRRST, SYSRST, EXTRST, LRST	high	must be high to ensure normal operation
BG3IN	PWRRST, SYSRST	low high	SCV64 becomes VME Syscon SCV64 does not become VME Syscon
KDATA	PWRRST	x	KDATA is latched into VMEBAR register

*a. The SCV64's KDS pin must be sampled as a logic high on the rising edge of the SCV64's LRST output which can occur up to 250 ms after a reset initiator (PWRRST, SYSRST, EXTRST, SWRST bit) has negated. One possible solution to ensure that the KDS pin is always sampled as a logic high is to use the SCV64's LRST output to control the local CPU's reset input. This will prevent the local CPU from driving the KDS line active when LRST negates.*



*Note all of these pins may be driven at some later point during normal operation, by the SCV64. For this reason, all pins specified as bidirectional, must be pulled to an appropriate logic level through an external resistor.*

### F.1.2 Test Mode Pins

These pins are used by the factory during manufacturing and test. Both must be grounded during normal operation. If in-circuit testing is to be performed on the board, pins TMODE1 and TMODE0 should be pulled low through 4.7 K $\Omega$  resistors. Contact Tundra's Applications Department for more detail.

### F.1.3 BI-Mode

If BI-Mode is not to be used in the system, tie BIREL to ground, and BITRIG to a high level. The SCV64 will enter BI-mode whenever one of the BI-mode initiators is active, but will automatically return to an operational state when the initiator is removed. Initiators are:

- any reset,
- IRQ1\* (when configured as a BI-mode initiator),
- the SBI bit in the GENCTL register,
- BITRIG

IRQ1\* defaults as a BI-mode initiator, meaning that if this signal is low after reset, the SCV64 will remain in BI-mode, until it is released. Normally, all IRQ lines are pulled high by the VME backplane, however, some backplanes are shipped without terminating resistors. Systems where IRQ1\* is floating may see erratic entry into BI-mode until IRQ1\* is reconfigured as a VME interrupt source using the VI1BI bit in the GENCTL register.

### F.1.4 JTAG

The SCV64 presently has no internal JTAG controller. JTMS and JTCLK may be connected to ground or to the corresponding board level signals. JTDI and JTDO are internally connected, allowing subsequent implementation of a complete JTAG board level test.

### F.1.5 Clocks

The C32MHZ clock input on the SCV64 controls several internal timing functions. Besides generating the SCV64 clock/timer outputs (C8MHZ, C14US, BAUDCLK, TICK, and WDOG, SYSCLK), it also affects:

- the local bus timer
- the VMEbus ownership timer

- the VMEbus time-out timer
- calibration of internal delay lines

To ensure reliable operation of the SCV64, this pin must be tied to a 32MHz clock.

The KCLK clock input should be synchronous to the local CPU clock, and may be any frequency up to the specified limit.

### F.1.6 Resets

The SCV64 can be reset through one of several sources: PWRRST, EXTRST, SYSRST\*, a software reset, and, when configured as the Syscon, BG0IN\*. Some backplanes do not terminate BG0IN\* on slot 1. Ensure that this signal is tied to an appropriate level, or to an external reset switch.



*Note: For slave applications, where the VME slave image is configured using the AUTOBAR process, the slave image will not be reset or reloaded during any reset except PWRRST. As such, a designer may wish to route the SYSRST pin to the PWRRST pin, to guarantee that the slave base addresses are reprogrammed to their initial values.*

### F.1.7 RETRY\*/VRMC

The SCV64 RETRY\*/VRMC pin may be configured as either the RETRY\* pin, or as a proprietary read-modify-write pin. The use of this pin is configured with the RMCPIN bit in the MODE register. When configured as RETRY\*, it may be direct connected to the VMEbus connector. When configured as the VRMC pin, it should be routed through the strobes buffer (along with VAS, VDS0, VDS1, etc.) with the direction controlled by VSTRBOUT.

## F.2 Software

The following describes various software issues for the programmer during initialization.

**Table F.2 : Software Initialization Summary**

Functional Group	Register	Bit	Description
Delay Lines	DLST1	CAL20 CAL30 CAL40	20 ns process calibration value 30 ns process calibration value 40 ns process calibration value
	DLCT	OFFSET20 OFFSET30 OFFSET40	20 ns delay line process offset 30 ns delay line process offset 40 ns delay line process offset
Cycle generation	MODE	BUSSIZ SWAP	unaligned cycle generation / bus sizing VME to local byte lane mapping
Slave Image	VMEBAR	A24BA A24SIZ	A24 slave image base address A24 slave image size
		A32BA A32SIZ	A32 slave image base address A32 slave image size
	MODE	A24SLVEN VINEN	A24 slave image enable A32 slave image enable
	SA64BAR	-	A64 slave image enable
Local Memory Map	MODE	A16DI A24DI A24PO	disable A16 accesses disable A24 accesses A24 page location
	VSBSSEL	-	VSb decode area
BI-mode	GENCTL	VI1BI	IRQ1* BI-mode configuration
RMC/RETRY	MODE	RMCPIN	RETRY*/VRMC pin use
Burst cycles	MODE	FIFOBEN	enable RX FIFO bursts to local side
Local Timers	MISC	ENDOG	enable watchdog timer
	CTL2	TICKM TLEN1,0	tick timer speed tick timer length
	GENCTL	LTOEN	local bus timer enable
Other local options	MODE	RMCRETRY BERRCHK	read-modify-write deadlock termination check for late bus error on VMEIN reads
Local Arbiter	CTL2	LBRAM	arbitration mode (SCV64 vs. LBRQ1)
Interrupts	IC54		LIRQ5, 4 interrupt level mapping
	IC32		LIRQ3,2 interrupt level mapping
	IC10		LIRQ1, 0 interrupt level mapping
	IC54	5AV, 4AV	LIRQ5, 4 auto-vector configuration
SYSFAIL	MISC	SYSFAIL	assertion of VME SYSFAIL*

**Table F.2 : Software Initialization Summary (Continued)**

Functional Group	Register	Bit	Description
VME Requester	VREQ	OTEN	VME ownership timer enable
		OT1,0	VME ownership timer length
		BCEN	Bus clear (BCLR*) recognition
		REL	release mode (ROR or RWD)
		REQ	request mode (Fair or Demand)
		LVL1,0	request level
	CTL2	MEMIM	L7IMEM effect of VME requester
VME Arbiter	VARB	ARB1,0	arbitration mode (PRI, RR, mixed)
		ATEN	arbitration time-out enable
		VXL1,0	arbitration time-out length

### F.2.1 Delay Line Initialization

The SCV64 has five internal delay lines (three 40ns delay lines and one each of 30ns and 20ns) which provide for necessary timing to satisfy conditions of the VME specification. These delay lines are continuously calibrated for temperature and voltage. Initially, to ensure optimum performance, they must also be calibrated for processing. This is done through reading a value from a status register for each delay line, and altering another based upon this value.

Three register fields must be read. These are CAL40, CAL30, and CAL20, all in the DLST1 register. A new value based upon the original contents of these registers must be written to the OFFSET\_40, OFFSET\_30, and OFFSET\_20 fields in the DLCT register (the SEL\_40, SEL\_30, and SEL\_20 bits must be 0 during the write). Once written, the OFFSET fields need not be re-calculated until the next reset of the SCV64, and can be locked until the next reset through writes to the KEY fields in the same register. Follow the setting of the KEY\_20, KEY\_30, and KEY\_40 bits with setting of the MKEY bit to lock the OFFSET fields.

Perform the following operations (truncating the results to integral values):

$$\text{OFFSET\_40} = (\text{CAL40} \div 40) + 3,$$

$$\text{OFFSET\_30} = (\text{CAL30} \div 6) + 4, \text{ and}$$

$$\text{OFFSET\_20} = \text{CAL20} \div 35.$$

Write these above values to the appropriate fields in the DLCT register.

These values can be locked in through setting bits KEY\_40, KEY\_30, and KEY\_20. A subsequent setting of bit MKEY permanently locks all these values until the next reset.



*Note: the SEL bits in the DLCT register must not be set.*

## F.2.2 Byte Lane Mapping and Cycle Conversion

Two bits in the MODE register control the form that the local bus interface to the VMEbus takes on. These are the BUSSIZ bit and the SWAP bit. Essentially, the BUSSIZ bit controls how local CPU cycles are converted into VME cycles through enabling the generation of unaligned cycles on the VMEbus, while the SWAP bit controls what byte lanes are relevant to a particular cycle.

## F.2.3 Slave Address Programming

The SCV64's VME slave address may be automatically set and enabled at power up, or configured and enabled at any other point in time through programming a series of registers as follows:

**Table F.3 : Programming for VME Slave Address**

Requirement	Register	Field
A24 slave image base address	VMEBAR	A24BA
A24 slave image size	VMEBAR	A24SIZ
A24 slave image enable	MODE	A24SLVEN+VINEN
A32 slave image base address	VMEBAR	A32BA
A32 slave image size	VMEBAR	A32SIZ
A32 slave image enable	MODE	VINEN
A64 slave image base address	SA64BAR	-
A64 slave image size	always 4GB	
A64 slave image enable	enabled by write to SA64BAR+VINEN bit in MODE register	

Access protection may also be applied to any portion or all of the A24 or A32 slave images through programming the APBR register. This protection may be either write protection only, or read and write protection, as set by the PROT bit in the MODE register.



## F.2.4 Memory Map Overlay Initialization

The SCV64's memory map overlay can be configured to enable or disable A24 and A16 VME access. In addition, A24 accesses may be configured as either in the first page or last page of CPU addresses. This is done through bits in the MODE register:

**Table F.4 : Address Space Enabling**

MODE Register Bit	Effect When Set	Effect When Cleared
A16DI	disable A16 accesses	enable A16 accesses
A24DI	disable A24 accesses	enable A24 accesses
A24PO	A24 accesses in top page	A24 accesses in page 0

When A24 or A16 accesses are disabled, accesses in these areas become A32.

A portion of the memory map overlay may also be set aside for VSB or other device selects. This is programmed through the BUSSEL register.

## F.2.5 BI-Mode

If  $\overline{\text{BIREL}}$  was not tied low, and  $\overline{\text{BITRIG}}$  tied high as described above (see “Hardware Configuration” on page App F-1), then the SCV64 will come out of reset in BI-mode. It can be removed from BI-mode through a single write to the SCV64 Location Monitor, located in the upper longword of the A24 slave image and the A32 slave image. It may be written to from either the VME side (the location monitor is still accessible in BI-mode), or from the local side.

If BI-mode is not to be used in a system, IRQ1\* should be reconfigured to be a VME interrupt, instead of its default as a BI-mode initiator. This is done through clearing the VI1BI bit in the GENCTL register.

## F.2.6 RETRY\*/VRMC Configuration

The RETRY\*/ $\overline{\text{VRMC}}$  pin defaults as a proprietary read-modify-write pin, allowing multiple locked reads and writes at multiple addresses. Instead, it may be configured as the VME RETRY\* pin. Setting the RMCPIN bit in the MODE register will configure this pin as the RETRY\* pin.

If this pin is configured as the RETRY\* pin, VME style read-modify-writes may still be enabled through setting the TASCN bit in the MODE register. Setting this bit forces  $\overline{\text{VAS}}$  low for multiple cycles while the local  $\overline{\text{KMRC}}$  pin is held low. Note that read-modify-write using this method can only be performed on a single address per cycle.

## F.2.7 Other Local Options

During deadlocked cycles (due to a coupled cycle coming from the VME bus while another is attempting to go out), the SCV64 will force the local cycle to terminate through asserting  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$ . Read-modify-write cycles have the option of terminating with  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$  or just with  $\overline{\text{KBERR}}$ . The SCV64 will default to asserting only  $\overline{\text{KBERR}}$  during read-modify-write lockups, but by setting the  $\text{RMCRETRY}$  bit in the  $\text{MODE}$  register, these lockups will terminate with  $\overline{\text{KBERR}}$  and  $\overline{\text{KHALT}}$ .

During reads initiated from the VMEbus the designer may wish to check for data parity allowing the cycle to terminate. When the  $\text{BERRCHK}$  in the  $\text{MODE}$  register is set, the SCV64 will wait a further two clock cycles after  $\text{DSACK}$  has been asserted before terminating the cycle on the VMEbus with either  $\text{BERR}^*$  or  $\text{DTACK}^*$ .

## F.2.8 Local Bursts

Local burst of data from the  $\text{RXFIFO}$  may be enabled by setting the  $\text{FIFOBEN}$  bit in the  $\text{MODE}$  register. When there are more than two  $\text{MBLT}$  or four  $\text{BLT}$  transfers queued in the  $\text{RXFIFO}$ , and this bit is set, the  $\text{BLT}$  and  $\text{MBLT}$  data will generate local burst cycles.

Use the  $\text{BLEN}$  bits to limit the length of local burst to 4, 8, 16, or 32 longwords.

## F.2.9 Local Arbiter

The SCV64's internal arbiter, when not bypassed (in the arbiter bypass power-up mode), may be configured to use fair or demand arbitration between the SCV64 and the external request,  $\overline{\text{LBRQ1}}$ . When in demand mode, the external request has highest priority. Local arbitration mode is controlled through the  $\text{LBRAM}$  bit in the  $\text{CTL2}$  register.

## F.2.10 Interrupts

Local and VME interrupt sources may be individually enabled using the  $\text{LIE}$ ,  $\text{VIE}$ , and  $\text{7IE}$  registers. All local general interrupt sources (  $\overline{\text{LIRQ0}}$  -  $\overline{\text{LIRQ5}}$  ) may be mapped to interrupt levels using registers  $\text{IC54}$ ,  $\text{IC32}$ , and  $\text{IC10}$ . VME interrupt levels are permanently mapped to the corresponding level on the local bus. As well, two general interrupt sources,  $\overline{\text{LIRQ4}}$  and  $\overline{\text{LIRQ5}}$ , may be configured as vectored, or auto-vectored sources through the  $\text{IC54}$  register.

### F.2.11 Local Timer Functions

The SCV64 has a variety of timers, both for the VME and local buses. The watchdog timer,  $\overline{\text{WDOG}}$ , is enabled using the ENDOG bit in the MISC register. The TICK timer rate is set through the TICKM bit in the CTL2 register and the TLEN bits in the GENCTL register. Finally, the local bus timer, used to time out local cycles, may be disabled through the LTOEN bit in the GENCTL register.

### F.2.12 SYSFAIL\*

Assertion of the SYSFAIL\* pin is controlled through the SYSFAIL bit in the MISC register. It defaults to set, driving the SYSFAIL\* line. After completing diagnostics, the designer will want to clear this bit.

### F.2.13 VME Requester

The VME requester module may be configured for several options. The options available in the VREQ register are:

- Release On Request vs. Release When Done
- Fair or Demand requesting
- ownership timer
- request level
- recognition of the BCLR\* signal

As well, the L7IMEM interrupt source may be configured, through the MEMIM bit in the CTL2 register, to force the requester to release the VMEbus or have no effect.

### F.2.14 VME Arbiter

The SCV64's VMEbus arbiter may be configured for round robin, priority, or mixed mode arbitration with the VARB register. This register also controls the arbitration time-out period: never, 16us, 32us, 64us, or disabled.



# Appendix G Reliability prediction

This section is designed to help the user to estimate the inherent reliability of the SCV64, as based on the requirements of MIL-HDBK-217F. This information is recommended for personnel who are familiar with the methods and limitations contained in MIL-HDBK-217F. The information serves as a guide only; meaningful results will be obtained only through careful consideration of the device, its operating environment, and its application.

## G.1 Device Description

### G.1.1 Physical characteristics

- CMOS gate array
- 54,184 gates
- 0.7 $\mu$ m feature size
- 9.931mm x 9.931mm scribed die size

### G.1.2 Thermal characteristics

#### Thermal Resistance (Junction to Case) $\theta_{JC}$

299-pin Pin Grid Array (PGA)	2.0°C/W
304-pin Plastic Quad Flat Pack (PQFP)	2.0°C/W

#### Thermal Resistance (Junction to ambient) $\theta_{JA}$

299 pin Pin Grid Array (PGA)	19.3°C/W
304-pin Plastic Quad Flat Pack (PQFP)	21.30°C/W

**Power Dissipation (worst case)**

1.50W

Worst case power dissipation is defined as

- Fully loaded VMEbus backplane
- Vdd = 5.5Volts
- Worst case data pattern (alternate FFFF FFFF, 0000 0000 on consecutive data transfer cycles)
- Device is operating as a system controller.

**Power Dissipation (typical)**

System Controller enabled	0.75W
System Controller disabled	0.50W

**G.1.3 Latch-up current**

- Greater than 500mA per JEDEC Standard No. 17

**G.2 Parameters**

The equation used for reliability predictions is found in section 5 of MIL-HDBK-217F, in the subsection pertaining to microcircuits, gate/logic arrays and microprocessors. This equation incorporates the following parameters:

**MOS Digital and Linear Gate/Logic Array Die Complexity Failure Rate**

This parameter is derived from the number of gates in the SCV64 (54,184).

**Temperature Factor**

Derive this parameter from the tables provided using the worst case junction temperature of the device as above.

**Package Failure Rate**

The number of functional pins (which equals power and ground pins subtracted from total number of pins) on the 299 pin PGA is 230. We recommend using a value of 0.099 based upon the formula supplied for the PGA package and the number of pins.

**Environmental Factor**

This number is derived from the tables supplied, and requires knowledge of the device operating environment. Operating environments vary with the application, and selection of this parameter can only be determined by the end user.

**Learning Factor**

The CA91C078A device has been in production since December 1998.

# Appendix H      Environmental and Operating Parameters

**Table H.1 : Recommended Operating Conditions**

DC Supply Voltage ( $V_{DD}$ )	5 V
Power Dissipation	1.5 W
Ambient Operating Temperature ( $T_A$ Commercial)	0°C to +70°C
Ambient Operating Temperature ( $T_A$ Military)	-55°C to +125°C

**Table H.2 : Absolute Maximum Ratings**

DC Supply Voltage	-0.5 to 7.0 V
Input Voltage ( $V_{IN}$ )	-1.5 to $V_{DD}+1.5$ V
DC Current Drain per Pin, Any Single Input or Output	100 mA
DC Current Drain per Pin, Any Paralleled Outputs	100 mA
DC Current Drain $V_{DD}$ and $V_{SS}$ Pins	75 mA
Storage Temperature, ( $T_{STG}$ )	-65°C to +150°C



**WARNING:** Stresses beyond those listed above may cause permanent damage to the devices. These are stress ratings only, and functional operation of the devices at these or any other conditions beyond those indicated in the operational sections of this document is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.



**WARNING:** Plastic packages should not be out of drypack for more than 48 hours at standard conditions of 45% RH and 25°C. Any packages out of drypack for extended time periods should be redried by baking in air for 12 hours at 125°C. Higher temperatures and longer times may cause solderability failures.



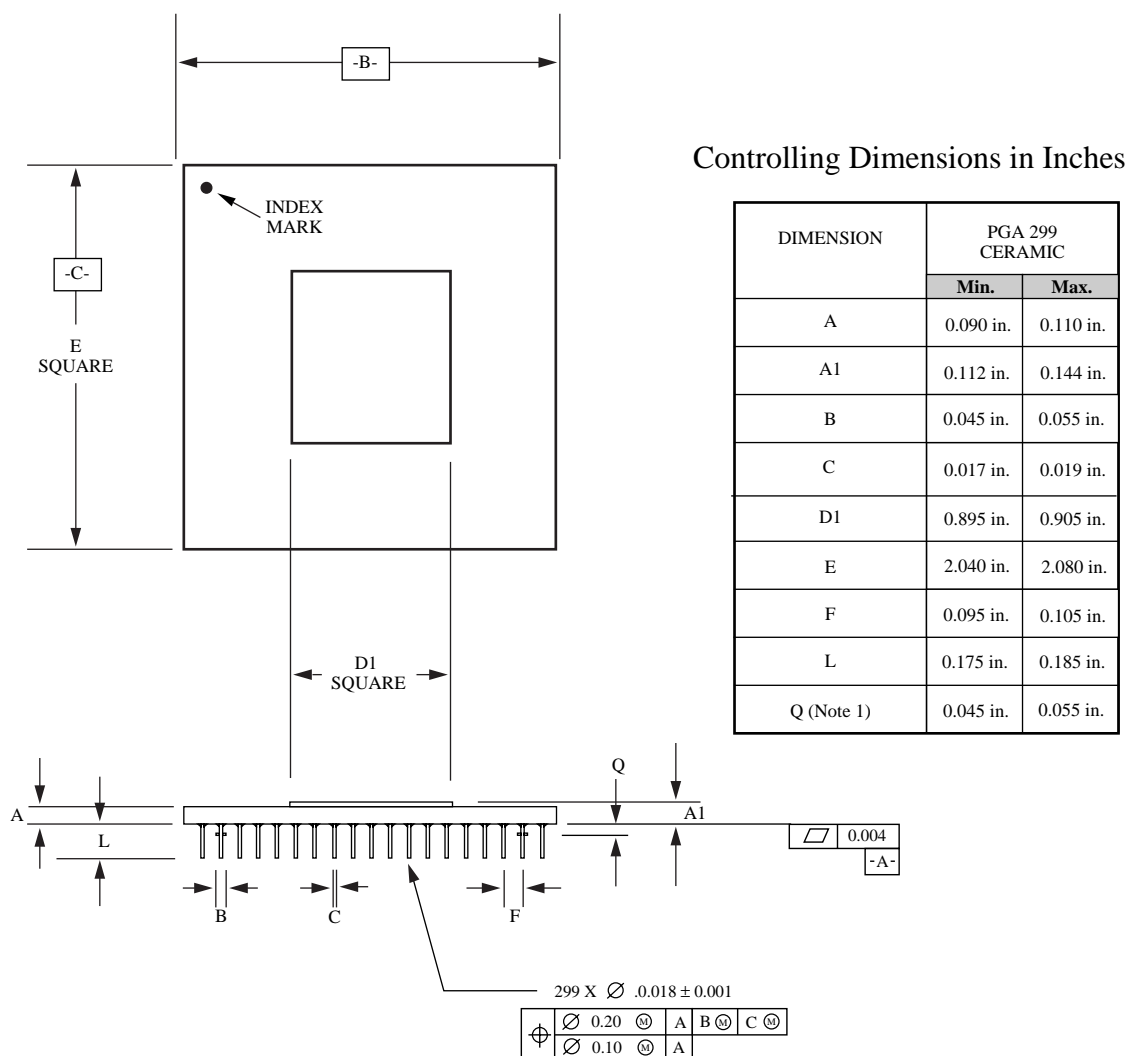
# Appendix I      Revision History

This revision of the SCV64 User's Manual incorporates minor modifications from the previous revision. The changes reflect the correction of errata and the addition of new information. Modifications are indicated by "change bars"—i.e., horizontal lines in the left margin of the affected lines of text.



# Appendix J Mechanical and Ordering Information

## J.1 Mechanical Information



Note 1. This dimension applies to the 4 ceramic disks on pins B2, B19, W2, and W19, which are used to hold the device above the printed circuit board.

**Figure J.1 : 299-pin Cavity-down CPGA Package**

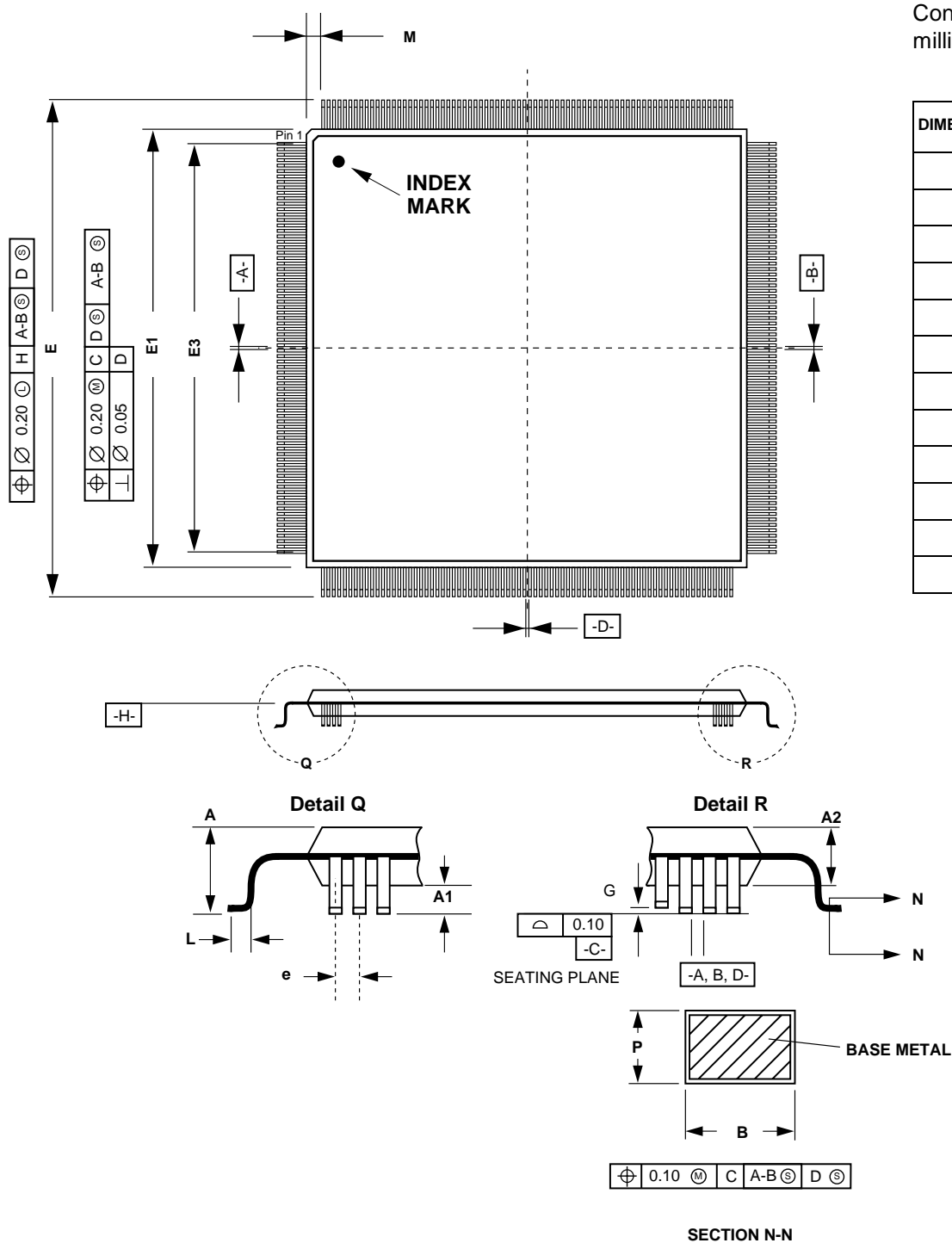
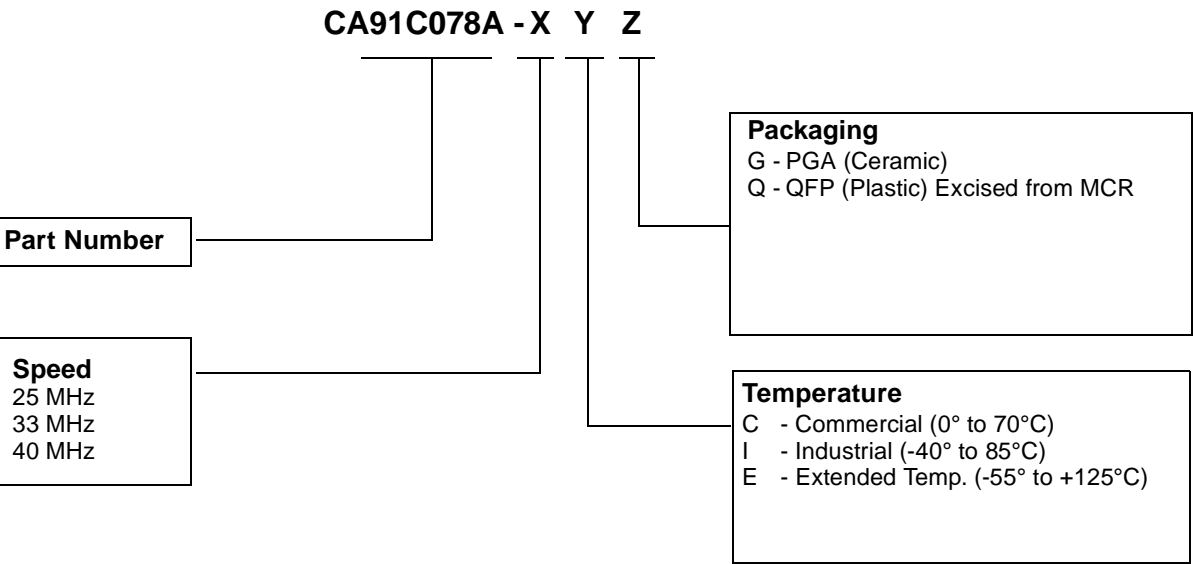


Figure J.2 : Mechanical Drawing for 304-Pin PQFP Package

J.2      Ordering Information

Tundra products are designated by a Product Code. When ordering, refer to products by their full code. For detailed mechanical drawings or alternative packaging requirements, please contact our factory directly.



**Valid Suffixes for CA91C078A**

X	Y	Z
25	I	Q
25	E	G
33	C	G, Q
33	I	Q
33	E	G
40	C	G, Q



# Index

Entries in **bold type-face** indicate definitions.

## Numerics

122 2-89

## A

A16 2-45, 2-49, 2-74

A16DI 2-45

A24 2-45, 2-51, 2-61, 2-99, 2-104, 2-118

A24DI 2-45

A24SLVEN 2-52

A32 2-45, 2-49, 2-51, 2-52, 2-61, 2-101, 2-104, 2-118

A64 2-43, 2-49, 2-50, 2-54, 2-61, 2-104

A64BARDY 2-50

ABI 2-13, 2-20, 2-119

Access Protection **2-52**  
specifying 2-52

ACFIE 2-20

ACFIP 2-21

ACFIS 2-21

Address Offset 2-75

Address Translation

SCV64 as VME master 2-53

SCV64 as VME slave 2-58

APBR 2-88

ARB0 2-32

ARB1 2-32

ARBBYP 2-66

Arbitration Modes 2-31

BR3\* priority 2-33

BR3\*, BR2\* priority 2-33

full priority arbitration 2-32

full round robin 2-32

AS\* 2-57

ATEN 2-33

AUTOBAR 2-51, 2-118

Auto-ID **2-121**

self test 2-123

Automatic Base Address Programming **2-51**,  
2-118

## B

BAUDCLK **3-5**

BBSY\* 2-9, 2-10, 2-33, **3-1**

BCEN 2-10

BCLR 2-57

BCLR\* 2-8, 2-10, 2-32, 2-33, **3-1**

BERR\* 2-39, 2-43, **3-1**

BERRCHK 2-77, 2-108

BG0IN\* 2-35

BG1IN 2-34

BG2IN 2-34

BG2IN\* and BG1IN\*  
external inputs 2-34

BG3IN\* 2-30

BG3IN\*-BG0IN\* **3-1**

BG3OUT\*-BG0OUT\* **3-2**

BGnIN\* 2-111

BGnOUT\* 2-9

BIE 2-20

BIMODE **3-5**

BI-Mode **2-4**, 2-100, 2-119

BI-Mode Effects 2-14, 2-29, 2-63  
 BIREL **3-5**  
 BIS 2-21  
 BITRIG 2-119, **3-5**  
 BLEN 2-41, 2-89, 2-106  
 BLEN1 2-90  
 BLKST 2-106  
 BLT 2-44, 2-54, 2-61, 2-102, 2-104  
 BLT and MBLT  
     DMA transfers 2-43  
 BR 3-0\* **3-2**  
 BR0\* 2-32  
 BR1\* 2-32  
 BR2\* 2-32  
 BR3\* 2-32  
 Burst Mode **2-4**  
     DMA transfers 2-106  
     local bus mastership 2-60  
     maximum burst length 2-41  
     reads 2-44, 2-60, 2-89  
     termination 2-89  
     writes 2-60, 2-90  
 Burst Reads  
     termination 2-89  
 Burst Writes 2-41, 2-44  
     termination 2-93  
 BUSSEL 2-49, 2-88  
 BUSSIZ 2-55, 2-59, 2-74  
 Byte Lane Translation 2-54  
     SCV64 as VME master 2-54  
     SCV64 as VME Slave 2-59  
 Byte Lanes 2-54  
 byte lanes 2-59

## C

C14US **3-5**  
 C32MHZ **3-5**

C8MHZ **3-5**  
 CERR 2-17  
 Clocks 2-113  
 CLRDOG 2-116  
 CLRTIK 2-115  
 Coupled Mode 2-36  
 CPU Memory Map **2-45**  
 CTL2  
     VMEbus busy 2-8

## D

D16 2-45, 2-61, 2-104  
 D32 2-45, 2-61, 2-104  
 D64 2-61, 2-104  
 D64(MBLT) 2-104  
 DCSR 2-36, 2-58, 2-62, 2-64, 2-77, 2-88, 2-99, 2-101, 2-108, 2-126, 2-127  
 Decoupled Mode 2-36  
 DISRX 2-41, 2-105, 2-127  
 DLBER 2-17, 2-78, 2-109  
 DLCT 2-89  
 DLST1 2-89, 2-124  
 DLST2 2-89  
 DLST3 2-89  
 DMA Controller **2-2**  
 DMA Transfers  
     completion and error checking 2-108  
     data transfer counts 2-107  
     initialization 2-102  
 DMA24 2-105  
 DMAA64 2-49, 2-105  
 DMABEN 2-89, 2-106  
 DMAD 2-105  
 DMAGO 2-108  
 DMALAR 2-61, 2-88  
 DMARD 2-103, 2-105



DMATC 2-64, 2-88, 2-104, 2-107  
 DMAVAR 2-61, 2-88, 2-104  
 DMAVTC 2-88, 2-104, 2-107  
     VMEbus DMA transfer count 2-107  
 DONE 2-17, 2-64, 2-108  
 DPRIV 2-105  
 DTACK\* 2-31, 2-39, **3-2**  
 DTCSIZ 2-107  
 Dynamic Bus Sizing **2-74**

## E

ENDOG 2-116  
 External Inputs 2-34  
     off-board reset 2-35  
 EXTKIACK 2-117  
 EXTRST 2-110, **3-5**

## F

Fair and Demand Modes **2-5**, 2-9  
 FIFOBEN 2-41, 2-90, 2-106  
 FILL 2-42, 2-56, 2-107  
 FILL Mode 2-56, 2-107

## G

GENCTL  
     TICK timer 2-116  
 General Local Interrupts  
     level mapping 2-21

## I

IACK Daisy Chain Driver 2-31  
 IACK\* 2-121, **3-2**

IACKIN\* 2-31, 2-122  
 IACKOUT\* 2-31, 2-122  
 IC10 2-16, 2-21, 2-22, 2-88  
     local interrupt level mapping 2-16  
 IC32 2-16, 2-21, 2-89  
     local interrupt level mapping 2-16  
 IC54 2-16, 2-21, 2-89  
     local interrupt level mapping 2-16  
 ID Register 2-88  
 ID register 2-122, 2-123  
 IDGOT 2-122  
 IDTST 2-123

Interrupt Acknowledge Cycles  
     auto-ID 2-121  
     auto-vectored 2-25  
     local decoding 2-22, 2-117  
     vectored 2-26

Interrupt Vector  
     local interrupter 2-27  
     VME interrupter 2-27

Interrupter **2-13**  
     local interrupts 2-15  
     VMEbus interrupts 2-13

Interrupts  
     auto-vectored 2-25  
     local timing 2-27  
     vectored 2-26  
     VME timing 2-28

IP 2-21  
 IRQ 7-2\* **3-2**  
 IRQ1\* 2-13, 2-119, 2-121  
 IVECT 2-14, 2-88

## J

JTCLK **3-5**  
 JTD0 **3-5**  
 JTD1 **3-5**

JTMS **3-6**

## K

KADDR 31 – 00 **3-6**

KAS 2-69, 2-73, 2-80, 2-83, 2-89, 2-93, 2-99, **3-6**

KAVEC 2-77, **3-6**

KBERR 2-77, **3-6**

KBGACK 2-69, 2-71, **3-6**

KBGR 2-66, 2-69, 2-71, **3-6**

KBRQ 2-66, 2-71, **3-6**

KCLK **3-6**

KDATA 31 – 00 **3-6**

KDS 2-73, 2-80, 2-83, 2-89, 2-93, **3-7**

KDSACK1 2-89, 2-93

KDSACK1-0 **3-7**

KDSACKx 2-80, 2-83

KFC2-KFC0 2-73

KHALT 2-77

KIPL2-0 **3-7**

KRMC **3-7**

KSIZE0-KSIZE1 2-73

KSIZE 1 – 0 **3-7**

KWR **3-7**

## L

L7IMEM 2-11, **3-8**

L7INMI **3-8**

LBERR 2-17, 2-41, 2-77, 2-109

LBGR1 2-66, **3-8**

LBRAM 2-69, 2-71

LBRQ1 2-66, 2-71, **3-8**

LIACK5-4 **3-8**

LIE 2-20, 2-88

interrupt enabling 2-20

LIRQ5-0 **3-8**

LIRQ5-LIRQ0 2-21

LIS

interrupt status 2-20

LMFIFO 2-1, **2-4**, 2-88, 2-99, 2-101

LMFIFO entries 2-62

LMHD 2-16, 2-62, 2-99, 2-101

LMINT 2-7, 2-16, 2-62, 2-99, 2-101, **3-8**

Local 2-115

local 2-7, 2-29, 2-114

Local Arbiter Bypass 2-66, 2-118

Local Bus Arbitration 2-66

arbiter state machine 2-67, 2-71

external requests 2-71

internal requests 2-67

Local Bus Cycles

data transfer 2-74

initiation 2-73

termination 2-76

Local Bus Mastership 2-60, 2-61

arbitration 2-66

local bus timer 2-7, 2-29, 2-114

Local Reset 2-110

Location Monitor **2-4**, 2-101

access 2-62, 2-99

LMFIFO 2-62

LMINT 2-62

LPBK 2-127

LRST 2-110, **3-8**

LTOEN 2-115

LVL0 2-9

LVL1 2-9

LWORD\* 2-14

## M

Master/Slave Deadlock 2-61, 2-77, 2-98

MBLT     **1-3**, 2-2, 2-39, 2-43, 2-61, 2-89, 2-102, 2-104, 2-106  
 MBOX0    2-89  
 MBOX1    2-89  
 MBOX2    2-89  
 MBOX3    2-89  
 MEMIE    2-20  
 MEMIP    2-21  
 MEMIS    2-21  
 MYBBSY   2-8

## N

NMICLR    2-20  
 NMIIS     2-21  
 NMIP       2-21  
 No Release Option   2-57, 2-108  
 NOREL     2-57, 2-108

## O

OT0     2-11  
 OT1     2-11  
 OTEN    2-11

## P

Power-up Reset    2-111  
 PROT     2-52  
 PWRRST   2-30, 2-111, **3-8**  
 PWRUP    2-111

## R

r        2-115  
 RAMSEL   2-83, 2-87, 2-99, **3-9**

Read Modify Write   2-57, 2-60  
     SCV64 as VME master   2-57  
     SCV64 as VME slave    2-60  
 Receive FIFO (RXFIFO)   **2-2**, 2-36  
     shift   2-41  
 Reflected Cycles    **2-49**, 2-99  
 Register Access      2-87  
 Registers  
     7IE  
         interrupt enabling   2-20  
     7IS  
         interrupt status    2-20  
     APBR  
         access protection   2-52  
     BUSSEL  
         VSB space   2-49  
     CTL2  
         local arbitration    2-69  
         TICK speed   2-115  
     DCSR  
         DMA completion and error checking   2-108  
         DMA initialization    2-102  
         error bits   2-16  
         location monitor access   2-99, 2-101  
         RXFIFO shift   2-41  
         VMEbus error   2-43  
     DMATC  
         DMA transfer count    2-107  
     GENCTL  
         Auto-ID test   2-123  
         BI-mode   2-119  
         IRQ1\* programming    2-13  
         local bus timer   2-115  
         software reset   2-110, 2-111  
     IVECT  
         interrupt vector   2-14  
     LAG  
         local address generator   2-105  
     MA64BAR  
         DMA addressing   2-43  
         DMA transfers   2-61, 2-104

- MODE
    - A16 disabling 2-45
    - A24 disabling 2-45
    - access protection 2-52
    - coupled mode 2-39
    - decoupled mode 2-39
    - DMA transfers 2-104
    - FILL mode 2-56
    - FILL option 2-107
    - no release option 2-57, 2-108
    - RMW control 2-57
    - RMW retry 2-62, 2-98
  - RXCTL, RXADDR, RXDATA
    - RXFIFO entry 2-40
  - SA64BAR
    - A64 base address 2-50
  - STAT1
    - auto-ID 2-122
    - BG2IN\*, BG1IN\* status 2-34
    - external status 2-34
    - power-up reset 2-111
    - syscon indicator 2-30
  - TXCTL, TXADDR, TXDATA
    - TXFIFO entry 2-42
  - VARB
    - ownership timer 2-34
  - VREQ
    - demand mode 2-9
    - fair mode 2-9
    - release modes 2-10, 2-11
    - request levels 2-9
    - VMEbus ownership timer 2-11
  - REL 2-10
  - Release on Acknowledgment (ROAK) **2-5**, 2-14
  - Release on Request (ROR) **2-5**, 2-10, 2-57
  - Release When Done (RWD) **2-5**, 2-10, 2-57
  - Request and Release Modes 2-8
  - Reset Effects 2-14, 2-29, 2-33, 2-35, 2-112
  - RETRY 2-57, 2-60, 2-97
  - RETRY\* 2-57, 2-58
  - RETRY\*/VRMC **3-4**
  - RMCCERR 2-17
  - RMCPIN 2-57, 2-60, 2-97
  - RMCRETRY 2-61, 2-98
  - RXADDR 2-40, 2-88, 2-127
  - RXATOM 2-39, 2-103, 2-105
  - RXCTL 2-40, 2-88, 2-106, 2-127
  - RXDATA 2-40, 2-88, 2-127
  - RXRST 2-36
  - RXSHFT 2-41, 2-127
- ## S
- SBI 2-119
  - SCV64 as Local Master
    - RXFIFO 2-83
    - TXFIFO 2-83
  - SCV64 as Local Slave
    - Local Bus Interface 2-78
    - RXFIFO 2-78
    - TXFIFO 2-78
  - SCV64 as VME Master
    - address translation 2-53
    - coupled mode 2-41
    - decoupled mode 2-42
  - SCV64 as VME Slave 2-39, 2-58
    - address translation 2-58
    - coupled mode 2-39
    - decoupled mode 2-39
  - SCV64SEL 2-87, **3-9**
  - Sequential Consistency **2-40**
  - SIZ 2-64
  - SPC 2-64
  - STAT0 2-20, 2-88, 2-115
  - Status/ID 2-31
  - SUPER 2-64
  - SWAP 2-54, 2-59
  - SWRST 2-110

SYFIE 2-20  
 SYFIP 2-21  
 SYFIS 2-21  
 SYSC 2-30  
 Syscon Determination 2-30, 2-118  
 SYSFAIL 2-18, 2-20, 2-21, 2-120  
 SYSFAIL\* **3-3**  
 SYSFLED **3-9**  
 SYSRST\* 2-30, 2-35, 2-52, 2-110, 2-111, 2-118, **3-3**  
 System Clock Driver 2-34

## T

TASCAN 2-57, 2-97  
 TICK **3-9**  
 TICKM 2-115  
 Timers 2-115  
     arbitration 2-33  
     TICK timer 2-115  
     VMEbus ownership 2-34  
     watchdog timer 2-116  
 TLEN 2-116  
 TMODE1-0 **3-9**  
 Transmit FIFO (TXFIFO) **2-2**, 2-36  
 TXADDR 2-42, 2-64, 2-88, 2-126  
 TXATOM 2-41, 2-103, 2-105  
 TXCTL 2-42, 2-64, 2-88, 2-126  
 TXDATA 2-42, 2-64, 2-88, 2-126  
 TXHD 2-65  
 TXRST 2-36  
 TXSHFT 2-65, 2-126  
 TYPE 2-64

## V

VADDR 31 – 01 **3-3**

VADDROUT 2-53, 2-58, **3-3**  
 VAM 5 – 0 **3-3**  
 VAS 2-57, **3-3**  
 VBERR 2-17, 2-41, 2-58, 2-64, 2-109, 2-126, 2-127  
 VDATAOUT 2-53, 2-58, **3-4**  
 VDS **3-4**  
 VI1 2-13, 2-20  
 VIE 2-20, 2-88  
 VINEN 2-52  
 VINT 2-13, 2-88, 2-128  
 VLWORD **3-4**  
 VME Accesses  
     A16 2-45  
     A24 2-45  
     A32 2-45  
     A64 2-49  
     D16 2-45  
     D32 2-45  
 VME Base Addressing  
     A24 2-50  
     A32 2-50  
     A64 2-50  
 VME Slave Image **2-49**  
 VMEBAR 2-49, 2-51, 2-88  
 VMEbus Arbiter **2-31**  
 VMEbus Interrupts  
     generation 2-13  
 VMEbus Mastership 2-56  
 VMEbus Ownership Timer 2-11  
 VMEbus Requester  
     **2-8**  
 VMEINT 2-7, 2-16, 2-41, **3-9**  
 VMEOUT 2-80, 2-99, 2-101, **3-10**  
 VSB Bus Accesses 2-49, 2-100  
 VSBSSEL 2-49, 2-100, **3-10**  
 VSTRBOUT 2-53, 2-58, **3-4**  
 VWR **3-4**

VXL0 2-34  
VXL1 2-34

## **W**

WDOG **3-10**