Migrating a VIC64/VIC068A Design to the SCV64™

Document Number: 8091078_AN001_01
Document Type: Design Note
Document Status:Final
Release Date: February 2003

Tundra Semiconductor Corporation

Trademarks

Copyright

Published in Canada

1. Migrating a VIC64/VIC068A Design to the SCV64

1.1 Overview

This document outlines the differences between Cypress VIC*x* (VIC64 and VIC068A) and the Tundra SCV64 device. The VIC*x* and the SCV64 devices are both VME64 compliant devices. However, they both have distinct features that are discussed in detail in this document.

The Single Chip VME64 interface component (SCV64) is one of Tundra Semiconductor's VMEbus System Interconnect Component. It provides a high performance 64-bit VMEbus interface in a single device. The SCV64 results from a four year development effort in VMEbus interface design which produced a VME64 Specification compliant device. Major Strengths of the SCV64 include a built in DMA controller, local bus burst modes and a FIFO architecture which allows the chip to decouple VMEbus transfers from the local bus. Other strengths of the SCV64 include a rich pool of features and operational modes which allow the user to tailor the VMEbus interface to a variety of data passing environments. The SCV64 is a full fledged VMEbus component with integrated VMEbus system controller capability, interrupt controller, DMA controller, and other features inherent to VME64.

The VIC64 and VIC068A are members of the industry standard Cypress VIC family of VMEbus interface products which are now obsolete and are not recommended for new designs[1]. The VIC64 provides a 64-bit interface, in addition to the 32-bit and 16-bit data bus cycles provided by the VIC068A. The primary benefit of using the VIC64 is that it provides a single chip 64-bit interface in addition to some feature enhancements over the VIC068A. Like the VIC068A, the VIC64 contains all the features required to function either as a slave, a master, or system controller. It provides a built in interrupt controller, a DMA controller, and DRAM refresh controller. Throughout this discussion, the VIC64 and VIC068A will be discussed hand in hand and referenced as the VICx.

This document assumes that the reader is familiar with the Motorola Processor 68K bus protocol and the VME64 bus protocol as defined by the ANSI/VITA VMEbus specification.

References for this Design Note can be obtained from the *Cypress VMEbus Interface Handbook* published in May 1996 and the *SCV64 VMEbus Interface Components Manual (*document number 8091078_MA001_01).

1. Refer to the Cypress website for details at http://www.cypress.com

1.2 Feature Differences

The Tundra SCV64 and Cypress VIC*x* devices are both VMEbus to MC68030 local bus bridge devices however, they have the following feature and operational differences:

- Data path: The SCV64's internal architecture is FIFO based. Write posting to a FIFO on both the transmit and receive data paths allows decoupled operation. The VIC*x* only provides a single register for write posting. Decoupled architecture is a significant advantage as opposed to coupled architectures. Decoupled mode provides a mechanism whereby the initiating master does not control the transfer. Data is stored into the SCV64's internal FIFO's while the processing master is free to perform other tasks. In a coupled architecture, any initiating master must wait for a data transfer to complete before initiating another data transfer. Decoupled mode maximizes throughput and reduces bus utilization on both the source and target card involved in the transfer.
- Register Access: The SCV64 registers can be accessed from the VMEbus and the local bus. The VIC*x* registers can be accessed from the local bus; only the Inter-processor Communication Facilities registers can be accessed from the VMEbus through the ICFSEL* signal.
- Block Transfer generation: In the VIC64, the CPU can control block transfers to the VMEbus via software. The SCV64 does not have this capability however this can be done via the SCV64's DMA controller.
- Reflected Cycles: The SCV64 supports reflected cycles. This allows the card to write to a VME address on itself. The SCV64 will reflect that cycle back onto the card. The VIC*x* does not support this and signals a VMEbus BERR*.
- Rescinding DTACK*: The VIC64 supports rescinding of the DTACK* signal whereby the signal is actively driven high at the end of a data transfer cycle. The SCV64 does not support this feature. Note that rescinding of DTACK* is an optional feature within the VME64 specification.
- Address Decode: The SCV64 has a built in VMEbus address decode. The VIC*x* requires external address decode. Both the SCV64 and the VIC*x* devices require external address decode on the local bus.
- Auto Syscon detection: During Power up, the SCV64 will monitor the BGIN3* pin (as per the VME64 specification) do determine system controller functionality. The VIC64 requires a jumper to be installed prior to slot 1 insertion to enable system controller functionality.

1.3 Signal Description

The following tables show all the VMEbus, local bus, and miscellaneous signals in both the VIC*x* and SCV64 devices. Note that all signals appended with "*" or with "_" indicate an active low signal. N/A indicates that it is not applicable in either the VIC devices or the SCV64.

1.3.1 VMEbus Signals

Table 1: VMEbus Signals

Signal Name	VIC068A/VIC64	Signal Name	SCV64
	Description		Description
D[7:0]	VMEbus Data Lines	VDATA[31:0]	VMEbus Data Lines
A[7:0]	VMEbus Address Lines	VADDR [31:0]	VMEbus Address Lines
AM[5:0]	VMEbus Address Modifiers	VAM[5:0]	VMEbus Address Modifiers
BG[3:0]OUT*	VMEbus Grant Output	BG[3:0]OUT*	VMEbus Grant Output
BG[3:0]IN*	VMEbus Grant Input	BG[3:0]IN*	VMEbus Grant Input
BR[3:0]*	VMEbus Request Lines	BR[3:0]*	VMEbus Request Lines
IACK*	VMEbus Interrupt Acknowledge	IACK*	VMEbus Interrupt Acknowledge
IACKIN*	VMEbus Interrupt Acknowledge Input	IACKI*	VMEbus Interrupt Acknowledge Input
IACKOUT*	VMEbus Interrupt Acknowledge Output	IACKO*	VMEBus Interrupt Acknowledge Output
IRQ[7:1]*	VMEbus Interrupt Requests	IRQ[7:1]*	VMEbus Interrupt Request
BBSY*	VMEbus Busy	BBSY*	VMEbus Busy
BCLR*	VMEbus Clear	BCLR*	VMEBus Clear
AS*	VMEbus Address Strobe	VAS_	VMEbus Address Strobe
DS[1:0]*	VMEbus Data Strobes	VDS[1:0]_	VMEbus Data Strobes
DTACK*	VMEbus Data Transfer Acknowledge	DTACK*	VMEbus Data Transfer Acknowledge
BERR*	VMEbus Error	BERR*	VMEbus Error
LWORD*	VMEbus Long Word	VLWORD_	VMEbus Longword Data Transfer Size
WRITE*	VMEbus Data Direction	VWR_	VMEbus read/write line
SYSCLK	VMEbus System Clock	SYSCLK	VMEbus System clock
SYSRESET*	VMEbus System Reset	SYSRST*	VMEbus System Reset
ACFAIL*	VMEbus ACFAIL	$\overline{\text{L7IACF}}$	Vmebus ACFAIL*
SYSFAIL*	VMEbus SYSFAIL	SYSFAIL*	VMEbus SYSFAIL

Table 1: VMEbus Signals

Signal Name	VIC068A/VIC64	Signal Name	SCV64
	Description		Description
	N/A	VADDROUT	VMEbus Address Transceiver Control
	N/A	VDATAOUT	VMEbus Data Transceiver Direction Control
	N/A	VSTRBOUT	VMEbus Address and Data Strobe Direction Control
	N/A	RETRY*/ VRMC_	Proprietary VMEbus retry output and VMEbus Retry input.

1.3.2 Local Signals

Table 2: Local Signals

Signals	VIC068A/VIC64	Signals	SCV64
	Description		Description
LD[7:0]	Local Data lines	KDATA[31:0]	Local Data bus lines
LA[7:0]	Local Address lines	KADDR[31:0]	Local Address lines
RESET*	Reset Output	LRST_	Local Reset Output
HALT*	Halt Status	KHALT_	CPU Halt Signal
PAS*	Physical/Processor Address Strobe	KAS_	Local Address Strobe
DS*	Processor Data Strobe	KDS_	Local Data Strobe
DSACK[1:0]*	Processor Data Size Acknowledge	KDSACK[1:0]_	Local Bus data transfer and size acknowledge
LBERR*	Local Bus Error	KBERR_	Local Bus Error
R/W*	Local Direction Signal	KWR_	Local Write signal/Direction Signal
RMC*	Read/Modify/Write	KRMC_	Local Read/Modify/Write
CS*	VIC Chip Register Select	SCV64SEL_	SCV64 Chip Register Select
SIZ[1:0]	Data Transfer Size	KSIZE[1:0]	Local Bus Data Transfer
FC[2:1]	Function Code	KFC[2:0]	Local Function codes
LBR*	Local Bus Request	KBRQ_	CPU Bus Request
LBG*	Local Bus Grant	KBGR_	CPU Bus grant
IPL[2:0]	Interrupt Priority Level	KIPL[2:0]_	CPU Interrupt Priority Levels
LIACKO*	Local Interrupt Autovector	KAVEC_	Local Interrupt Autovector signal
LIRQ[7:1]*	Local Interrupt Requests	LIRQ[5:0_]	Local Interrupt Requests
MWB*	Module wants bus	VMEOUT	Local VMEbus Chip Select
FCIACK*	Interrupt Acknowledge	LIRQ2_/KIACK_	Local Interrupt request 2 or Local IACK Decoding
ICFSEL*	Interprocessor Communications Facilities Chip Select		N/A
ASIZ[1:0]	Local Address Size	KSIZE[1:0]	Local Address Size

Table 2: Local Signals

Signals	VIC068A/VIC64	Signals	SCV64
	Description		Description
SCON*/D64	System Controller Status		N/A
IRESET*	Internal Reset Input		N/A
CLK64M	64 Mhz Clock Input		N/A
WORD*	D16/D32 Control		N/A
SLSEL[1:0]	Slave Select		N/A
DEDLK*	Deadlock Status		N/A
BLT*	Block Transfer Status		N/A
ABEN*	VMEbus Address Buffer Enable		N/A
LADO*	Latch Outgoing VMEbus Address		N/A
LADI*	Latch Incoming VMEbus Address		N/A
LEDO	Latch Outgoing VMEbus Data		N/A
LEDI	Latch Incoming VMEbus Data		N/A
DDIR	Local Data Direction		N/A
DENIN1*	Local Data Enable (Upper Word)		N/A
DENIN*	Local Data Enable (Lower Word)		N/A
SWDEN*	Swap Local Data Enable		N/A
DENO*	VMEbus Data Buffer Enable		N/A
ISOBE*	Isolation Buffer Enable		N/A
LAEN*	Local Address Buffer Enable		N/A
	N/A	KBGACK_	Local Bus Grant Acknowledge
	N/A	EXTRST_	External Reset Input used to reset device
	N/A	KCLK	Local clock Input
	N/A	L7IMEM_	Local Level 7 Interrupt
	N/A	L7INMI_	Local Level 7 Interrupt
	N/A	LBGR1_	Local Bus Grant when in Arbiter Mode
	N/A	LBRQ1_	Local Bus Request from a local requester

Table 2: Local Signals

Signals	VIC068A/VIC64	Signals	SCV64
	Description		Description
	N/A	LIACK[5:4]_	Local Interrupt Acknowledge
	N/A	LMINT_	Location Monitor Interrupt
	N/A	VSBSEL_	VSB Bus Select
	N/A	RAMSEL_	Memory Select Signal
	N/A	SYSFLED_	SYSFAIL LED Driver
	N/A	TICK_	Tick Timer
	N/A	TMODE[1:0]	Test Mode Inputs (Not Used/Tied to Ground)
	N/A	VMEINT_	VMEbus Activity Interrupt
	N/A	WDOG_	Watch Dog Timer
	N/A	BAUDCLK	Baud Clock output used to generate baud rate Clocks
	N/A	BIMODE	Bi Mode Signal
	N/A	BIREL_	Bi Mode Release Signal
	N/A	BITRIG_	Bi Mode Trigger Signal
	N/A	C14US	14us clock generated from the C32Mhz clock
	N/A	C8MHZ	8 Mhz general purpose clock output
	N/A	C32MHZ	32 Mhz clock used to clock signals and generate VMEbus state machine timing
	N/A	JTCLK	JTAG Test Clock (Not Used)
	N/A	JTDI	JTAG Test Data Input (Not Used)
	N/A	JTDO	JTAG Test Data Output (Not Used)
	N/A	JTMS	JTAG Test Mode Select (Not Used)
	N/A	PWRRST*	Power On Reset

1.4 Registers

Both the SCV64 and the VIC*x* devices have unique control and status registers which are accessed using two distinct methods. The VIC*x* internal registers can be accessed from the local bus. Only the Inter-processor Communications Facilities (ICF) registers can be accessed from the VMEbus by means of the ICFSEL* signal.

The SCV64 has an added feature which allows all the device's internal registers to be accessed from the VMEbus.

For more information on specific bit locations and bit functionality refer to the *SCV64 VMEbus Component Interface Manual* and the *Cypress VMEbus Interface Handbook*.

1.4.1 Device Register Maps

The following sections independently discuss the Control and Status Register (CSR) space for the VIC*x* device and the SCV64 device.

The VIC068A is added to the description because the VIC64 is an extension to the VIC068A device to allow 64-bit operation on the VMEbus.

1.4.1.1 VIC64/VIC068A Register Map

The VIC068A contains 58, 8-bit internal registers addressable from the local bus interface only. Although registers in the VIC068A are 8-bit, the VIC068A always acknowledges a register access with both DSACK0* and DSACK1* asserted because the registers need to be addressed on longword boundaries.

There are minor differences between the VIC64 and the VIC068A registers to differentiate the 32-bit only VIC068A and the 64-bit VIC64 device. To allow compatibility with the VIC64, within the VIC068A all reserved bits must have a 0 written to them.

Table 3: VIC64/VIC068A Register Map

Register Address	Name	Description
0x03	VIICR	VMEbus Interrupter Interrupt Control Register
0x07-0x1F	CICR1-7	VMEbus Interrupt Control Register 1-7
0x23	DMASR	DMA Status Register
0x27-0x3F	LICR1-7	Local Interrupt Control Registers 1-7
0x43	ICGSICR	ICGS Interrupt Control Register
0x47	ICMSICR	ICMS Interrupt Control Register
0x4B	EGICR	Error Group Interrupt Control Register
0x4F	ICGSVBR	ICGS Vector Base Register
0x53	ICMSVBR	ICMS Vector Base Register
0x57	LIVBR	Local Interrupt Vector Base Register
0x5B	EGIVBR	Error Group Interrupt Vector Base Register
0x5F	ICSR	Interprocessor Communications Switch Register
0x63-0x73	ICR0-4	Interprocessor Communications Registers 0-4
0x77	ICR5	Interprocessor Communications Register 5
0x7B	ICR6	Interprocessor Communications Register 6
0x7F	ICR7	Interprocessor Communications Register 7
0x83	VIRSR	VMEbus Interrupt Request Status Register
0x87-9F	VIVBR1-7	VMEbus Interrupt Vector Base Register 1-7
0xA3	TTR	Transfer Time-out Register

Table 3: VIC64/VIC068A Register Map

Register Address	Name	Description
0xA7	LBTR	Local Bus Timer Register
0xAB	BTDR	Block Transfer Definition Register
0xAF	ICR	Interface Configuration Register
0xB3	ARCR	Arbiter/Requester Configuration Register
0xB7	AMSR	Address Modifier Source Register
0xBB	BESR	Bus Error Status Register
0xBF	DMAISR	DMA Interrupt Status Register
0xC3	SS0CR0	Slave Select 0 Control Register 0
0xC7	SS0CR1	Slave Select 0 Control Register 1
0xCB	SS1CR0	Slave Select 1 Control Register 0
0xCF	SS1CR1	Slave Select 1 Control Register 1
0xD3	RCR	Release Control Register
0xD7	BTCR	Block Transfer Control Register
0xDB	BTLR1	BLock Transfer Length Register 1
0xDF	BTLR0	Block Transfer Length Register 0
0xE3	SRR	System Reset Register
0xEB-0xFF		Reserved Locations

1.4.1.2 SCV64 Register Map

The SCV64 has 45, 32-bit registers that are addressable from the local bus as well as the VMEbus. The SCV64 always responds as a 32-bit slave when its internal control and status registers are accessed (both KDSACK1_ and KDSACK0_ are asserted).

The SCV64 decodes access to its registers using local address lines KADDR08 to KADDR00. Registers in the address range 0x000 to 0x04C respond only to aligned longword transfers; the SCV64 will assert KBERR_ with any other access. Registers in the address range 0x080 to 0x0BF accept any size transfer, but ignore all data lines except KDATA07 and KDATA00. Registers in the address range 0x0C0 to 0x0E0 respond only to aligned longword transfers; the SCV64 asserts KBERR_ with any other access.

Ranges 0x050 to 0x07F and 0x0E4 to 0x1FC are reserved locations and the SCV64 will assert KBERR_ if accessed. The SCV64 register map is shown in Table 4.

Table 4: SCV64 Register Map

Register address	Name	Description
0x000	DMALAR	DMA local Address Register
0x004	DMAVAR	DMA VMEbus Address Register
0x008	DMATC	DMA Transfer Count
0x00C	DCSR	DMA Control and Status Register
0x010	VMEBAR	VMEbus Slave Base Address
0x014	RXDATA	Rx(Receive) FIFO Data
0x018	RXADDR	Rx(Receive) FIFO Address Register
0x01C	RXCTL	Rx(Receive) Control Register
0x020	BUSSEL	VMEbus/VSB Bus Select
0x024	IVECT	VMEbus Interrupt Vector
0x028	APBR	Access Protect Boundary
0x02C	TXDATA	Tx (Transmit) FIFO Output Latch
0x030	TXADDR	Tx(Transmit) FIFO Address Output Latch
0x034	TXCTL	Tx(Transmit) FIFO AM code and Control bit latch
0x038	LMFIFO	Location Monitor FIFO Read port
0x03C	MODE	SCV64 Mode Control Register
0x040	SA64BAR	Slave A64 Base Address
0x044	MA64BAR	Master A64 Base Address
0x048	LAG	Local Address Generator
0x04C	DMAVTC	DMA VMEbus Transfer Count
0x050 to 0x07F		Reserved
0x080	STAT0	Status Register 0
0x084	STAT1	Status Register 1
0x088	GENCTL	General Control Register
0x08C	VINT	VMEbus Interrupt Register
0x090	VREQ	VMEbus Request Register
0x094	VARB	VMEbus Arbiter Register

Table 4: SCV64 Register Map

Register address	Name	Description
0x098	ID	ID Register
0x09C	CTL2	Control and Status Register
0x0A0	7IS	Level 7 Interrupt Enable Register
0x0A4	LIS	Local Interrupt Status Register
0x0A8	7IE	Level 7 interrupt Enable Register
0x0AC	LIE	Local Interrupt Enable Register
0x0B0	VIE	VMEbus Interrupt Enable Register
0x0B4	IC10	Local Interrupts 1 and 0 Control Register
0x0B8	IC32	Local Interrupts 3 and 2 Control Register
0x0BC	IC54	Local Interrupts 5 and 4 Control Register
0x0C0	MISC	Miscellaneous Control Register
0x0C4	DLCT	Delay Line Control Register
0x0C8	DLST1	Delay Line Status Register 1
0x0CC	DLST2	Delay Line Status Register 2
0x0D0	DLST3	Delay Line Status Register 3
0x0D4	MBOX0	Mailbox Register 0
0x0D8	MBOX1	Mailbox Register 1
0x0DC	MBOX2	Mailbox Register 2
0x0E0	MBOX3	Mailbox Register 3
0x0E4 to 0x1FC		Reserved

1.4.2 Register Access

The following sections describe how registers are accessed when both using the VIC*x* and SCV64 devices.

1.4.2.1 Accessing the VIC*x* Internal Registers

Below is a list of all the control and bus signals involved in a VICx (VIC64/VIC068A) register access:

- PAS*: Physical/Processor Address Strobe
- DS*: Processor Data Strobe
- CS*: VIC068A/VIC64 Register Chip Select
- R/W*: Processor Read/Write signal
- DSACK[1:0]*: Data size acknowledge signal
- LD[7:0]: Processor Local Data
- LA[7:0]: Processor Local Address

Access to the VICx internal registers is determined by the assertion of PAS*, DS*, and CS*. R/W* will determine if the access is a read or a write. Although the registers are 8 bits wide, the VICx always acknowledges a register access with DSACK*s asserted. This is due to the fact that the registers are addressed on longword boundaries. When reading a register, data is placed on the LD[7:0] lines. When writing to a register, data must always be place on the LD[7:0] lines. A register access will complete when either PAS*, DS*, or CS* deasserts.

Accesses to the Interprocessor Communications Facilities registers are accessed by using the ICFSEL* signal. This signal is driven from the VMEbus address decoders. When it is asserted, the VIC device checks A[5:1] to determine which ICF register is accessed.

For a detailed timing diagram refer to the *Cypress VMEbus Interface handbook*.

1.4.2.2 Accessing the SCV64 Internal Registers

Below is a list of all the control and bus signals involved in a SCV64 register access:

- SCV64SEL_: SCV64 chip select used by a local address decoder for register accesses.
- KADDR[0:8]: Local address lines.
- KSIZ[1:0]: Local bus data transfer size.
- KDATA[31:0]: Local bus data lines.
- KAS_: Local Address Strobe.
- KDS_: Local Data Strobe.
- KWR_: Local Read/Write signal.
- KDSACK[1:0_]: Local bus Data Transfer and size acknowledge signals.

Note: The KFC[2:0] signals are not decoded during a register access as they would be during a local slave access. This will discussed further in section 1.7 when the SCV64 is a local slave.

A register access occurs when the SCV64SEL_ signal (the SCV64 register access chip select) is asserted. This indicates to the SCV64 that the pending local cycle is an access to its internal registers.

The sequence of events for a register access are as follows:

- A local master drives the address on KADDR[0:8] and KSIZ[1:0] and KWR_

During a register access, the SCV64 only monitors local address lines KADDR[8:0]. It ignores the upper data lines.

- The local master asserts KAS_ and KDS_.
- The local address decoder asserts SCV64SEL_
- The local master places data on the KDATA[31:0] for a write access
- The SCV64 latches data into the appropriate register in the case of a write access
- The SCV64 drives the data onto the KDATA[31:0] lines in case of a read access
- The SCV64 asserts both KDSACK[1:0]_.
- The local master negates the control signals and stops driving the address bus.
- The SCV64 negates the KDSACK[1:0]_ signals.

1.4.3 Register Accesses Differences Overview

The following section outlines the differences of the protocol differences between the two register accesses. This section is not a bit by bit comparison of each of the device's CSR space. As a standard notation, all active low signals (logic state is zero when asserted) are appended with an "*" or "_".

1.4.3.1 Address Phase

In both devices, a register access is determined by the assertion of a chip select. The VIC*x* requires assertion of CS* (or ICFSEL*) and the SCV64 requires the assertion SCV64SEL_. In both devices, only the lower 8-bit local address lines are decoded, LA[7:0] for the VICx and KADDR[7:0] for the SCV64. The PAS* signal for the VIC*x* and the KAS_ signal for the SCV64 indicate the beginning of the address phase.

The address and transfer type (read or write) is latched. When accessing the SCV64 registers from the VMEbus, the SCV64 must first gain access of the local bus by asserting RAMSEL_. The local address decoder then redirects the cycle to the SCV64 registers by asserting SCV64SEL_.

1.4.3.2 Data phase and Termination

In both devices, once the address phase is complete the cycle is accepted and the data latched.

In the case of a write, the WR* signal for the VIC*x* or the KWR_ signal for the SCV64 is asserted. Data is placed on the local data lines by the master following assertion of the DS* signal for the VIC*x* and the KDS_ signal for the SCV64. The cycle is terminated with both KDSACK[0:1]* asserted with the VICx since it will always respond as 32 bit device even though registers are 8 bit wide. The SCV64 will respond with both KDSACK[0:1]_ asserted. Assertion of the data transfer acknowledge signals will prompt the VICx devices and SCV64 that data has been accepted and to remove all control signals.

In the case of a read (WR* or KWR_ high), data is returned when the Data Transfer Acknowledge signals are asserted by the target device; either the VIC*x* or the SCV64. Data latching by the read master is indicated by the removal of the data strobe signals (DS* and KDS_) and other control signals. The Data Transfer Acknowledge signals are then negated by the VIC*x* and SCV64 devices.

For detailed timing information refer to the *SCV64 VMEbus Component Interface Manual* and the *Cypress VMEbus Interface Handbook*.

1.5 DMA Operation

Both the SCV64 and VIC*x* devices have built in DMA capability to allow them to become master on both the local bus and the VMEbus. DMA capability is key to higher performance. Both DMA engines handle the segmentation of the data transfer to ensure that the boundary rules set by the VMEbus specification are not violated. The use of the DMA engine increases the system performance by off-loading the processor from having to transfer the data.

When the VIC*x* devices are performing block transfers, they become both the local bus master and the VMEbus master.

The following sections describe programming and operation of the DMA engine for each of the devices.

1.5.1 VIC*x* DMA Engine

Performing block transfers using the DMA engine means the VIC*x* becomes the VMEbus master and the local master.

The VIC*x* can perform block transfers without using the DMA engine. Refer to “VME Master Block Operation” on page 38.

Performing DMA block transfers as a VMEbus master requires the following steps:

1. Define the transfer length to the VME byte length registers, BTLR0 and BTLR1 in the VICx device.
   - — Within the VIC64, D64 block mode operations can be distinguished by writing to bit 4 of the VIC64’s block transfer definition register (BTDR). The transfer length must be even because D08 block transfers are not supported.

The VIC068A can perform D16 and D32 block transfers. The VIC64 can perform D16, D32, and D64 block transfers.

2. Define the transfer direction by setting (for reads) or clearing (for writes) the DMA direction bit and set the DMA enable bit of the VICx block transfer control register (BTCR)
3. Define the source and destination addresses within the CPU memory map.
4. Enable DMA transfers by clearing the DMA enable bit in the BTCR register.

For description of the VIC*x* chip register set, refer to the *Cypress VMEbus Interface handbook*.

5. Once the previous steps are complete, and when the VIC*x* registers are initialized, a VMEbus write must be performed to the VMEbus destination address with the local starting address as the data.
   - This is referred to as a *Pseudo Write Cycle (PWC)*. This cycle must be a write because a read cycle does not place the correct data on the local bus. This cycle starts the block transfer mechanism and loads the addresses. Since any assertion of the MWB* (Module wants Bus) signal after the BTCR[6] bit is set (Block Transfer with Local DMA Enabled) indicates a PWC.
   - No normal read or write cycles should be performed until completion of the block transfer.
   - During this PWC, the VIC*x* loads the local address and signals by asserting the LADO (Latch Outgoing VMEbus address) signal to the local bus in order to load the VMEbus address in its internal registers. The VIC*x* loads the local data value by signaling the BLT* (Block Transfer) signal. The VICx terminates the local cycle and requests the VMEbus. After the VMEbus is obtained, the VIC*x* acquires the local bus by generating the LBR* (Local bus request) signal and drives the local DMA address.
   - It is important when using the DMA that local logic decode the BLT*, LBG*,PAS*, LAEN. For more details on these signals refer to the *Cypress VMEbus Interface handbook*.
6. Wait for completion of the DMA. The VICx can notify completion of the transfer if the completion interrupt is enable in the VICx status register (DMAICR) and its vector is generated by the VICx error group interrupt vector address register (EGIVBR).

The SCV64 DMA does not require a *Pseudo Write Cycle* to be performed. The source and destination addresses are pre-programmed within the device prior to launching the DMA operation

1.5.2 SCV64 DMA Engine

The SCV64 performs all VMEbus block transfers (BLT and MBLT) through its DMA engine only. The VIC*x* devices can perform block transfers without the use of its DMA engine (see "Interfacing to the Local Bus" on page 30)

Single cycles can also be performed the SCV64 DMA engine. DMA cycles are always decoupled transactions. Decoupled transactions mean the source access and destination access operate independently.

The following set of registers involved in the programming of the SCV64's DMA engine:

- DMA Control and Status Register (DCSR)
- DMA Transfer Count Register (DMATC)
- DMA Local Address Register (DMALAR)
- DMA VMEbus address register (DMAVAR)
- Mode register (MODE)

For more information on these registers refer to the *SCV64's User's Manual*.

The following steps are used for initiating the SCV64 DMA:

1. Clear the DCSR register
2. Enable decoupled cycles by clearing either the RXATOM bit (VMEbus reads) or the TXATOM bit (VMEbus writes) in the MODE register.
3. Set the proper address and data modes.
   - This includes programming of the DMALAR and DMAVAR registers. The DMALAR and DMAVAR values are either the source or destination address depending on whether the DMA transfer is a VMEbus read or VMEbus write.
   - Data modes are set in the MODE register including: D8,D16,D32, BLT, or MBLT transfers. Address space is programmed in the MODE register as well to determine A24, A32, or A64 accesses.
4. If local burst mode is required on the SCV64 local bus, setting the DMABEN bit in the MODE register enables local bursting.
5. Set the direction of the DMA transaction in the DMARD bit in the MODE register.
6. Program the DMA transfer count in the DMATC register.
7. Set the GO bit in the DCSR register

The SCV64's DMA acquires the source address depending on whether the transfer is a read or write. Once the DMA completes, the SCV64 internally sets the DONE bit in the DCSR register and, optionally, asserts the VMEINT signal to the CPU to indicate normal completion of the DMA transfer.

For more information on Addressing and Data Transfer mode, please refer to the *SCV64's User's manual*.

1.5.3 DMA Engine Differences

Both the VIC*x* and the SCV64 require that internal registers be programmed prior to initiating a DMA transfer. Direction of the transfer, length of the transfer, source and destination addresses need to be set up before a DMA can be performed. The main difference between implementing DMA transfers are the steps required before a cycle starts, the types of cycles supported, and configuration.

1.5.3.1 DMA Cycle Initiation

In order to initiate a DMA transfer with SCV64, all that is required is to program five registers. The SCV64 contains registers for the source and destination addresses (DMALAR and DMAVAR), the length of the DMA transfer (DMATC register). All other transfer attributes can be programmed in the DCSR (DMA control and Status register) and MODE registers. The DMA is then launched by setting the GO bit.

The VICx devices requires more steps to initiate a transfer. Information on the DMA transfer is contained both in the internal registers and in the CPU's local memory map. Internal registers contain transfer attributes, size of the transfer. The source and destination addresses are programmed within local CPU memory. Unlike the SCV64 where the DMA is initiated by setting an internal bit, the VIC*x* DMA is initiated by performing what is called a Pseudo Write Cycle.

1.5.3.2 DMA Cycle Support

The additional advantage of using the SCV64's internal DMA is that it allows single cycle operations to be performed. The VIC*x*'s internal DMA will only perform block transfers when programmed. Single read and write cycles can be performed when using the SCV64's internal DMA engine.

1.5.3.3 DMA Configuration

Configuration of the DMA within the SCV64 (see "SCV64 DMA Engine" on page 23) allows for easier implementation of application software and also provides a simple approach when designing a slave VME application.

1.6 Interfacing to the VMEbus

Both the SCV64 and the VICx devices have similar VMEbus signals. The differences between the two devices are the buffer control logic. Both devices require external transceivers and buffer logic to operate on the VMEbus. The difference between the VIC*x* devices and the SCV64 device is the VIC*x* requirement to have address and data multiplexing buffers to perform D16, D32, and D64 (available only with the VIC64) transfers, since the device only provides eight address and data lines as on chip signals.

For diagram descriptions of the physical interface refer to the *SCV64 Vmebus Component Interface Manual* and the *Cypress VMEbus Interface Handbook*.

1.6.1 VICx VMEbus Hardware Guideline

The VIC*x* (VIC068A/VIC64) interface to the VMEbus is composed of direct connect signals to the VMEbus backplane as well as buffered signals. The VIC*x* devices requires specific logic to allow multiplexing and latching of incoming and outgoing Data Bus Transfers (DBT).

The VIC*x* devices are 8-bit interface devices that offer the full VMEbus D32 bus width and D64 bus width (available only with the VIC64) when interfaced with the proper logic. The VMEbus address and data decoding is accomplished through latches and transceivers that are shared with the local address and data lines. The VIC*x* devices require external address decode logic for local or VME slave transfers.

The following list shows the VIC*x* direct connect signals to the VMEbus backplane:

- BG[3:0]IN*
- BG[3:0]OUT*
- BR[3:0]*
- IRQ[1:7]*
- IACK*
- IACKIN*
- IACKOUT*
- BBSY*
- BCLR*
- BERR*
- DTACK*

- SYSCLK*
- SYSFAIL*
- SYSRST*
- AS*
- DS[1:0]*
- LWORD*
- WRITE*
- ACFAIL*

The VMEbus address, data, and address modifier signals are required to interface with their appropriate control signals to external logic.

The VMEbus address and data signals are connected to buffer latches enabled and controlled by the buffer control signals outlined in section 1.6.3. Such signals include LADO, LADI, LEDO, LEDI, ABEN*, DENO*, D64, BLT*, LAEN*, DENIN*, and DENIN1*.

Typical VICx applications are composed of x245 Bidirectional transceivers as well as x543 latched transceivers for the data and address groups [15:8], [23:16], and [31:24]. Cypress' CY7C964 device can interface the VIC*x* devices on the VMEbus since all buffer control signals, address, and data signals connect directly to it. Only four of these devices are required as opposed to 6 x542 latches and 2 x245 transceivers. The A[7:0], D[7:0] transceivers on the CY7C964 provided the high drive strength, allowing direct connection to the respective address and data signals on the VMEbus backplane.

A typical VMEbus interface using the VIC*x* devices can be seen in the *Cypress VMEbus Interface Handbook*.

1.6.2 SCV64 VMEbus Hardware Guideline

The SCV64 interface to the VMEbus is composed of direct connect signals to the VMEbus backplane as well as buffered signals. The buffers in this interface are used to allow the device to meet the *VMEbus Specification* electrical requirements. As in all buffered design, control and direction signals are required to enable proper flow of data transfers when in slave mode or in master mode. The SCV64 performs VMEbus address decoding and data latching internally and does not require latches since it possesses a full 32 bit address and data bus.

The following list shows the SCV64 direct connect signals to the VMEbus backplane:

- BG[3:0]IN*
- BG[3:0]OUT*
- BR[3:0]*
- IRQ[1:7]*
- IACK*
- IACKI*
- IACKO*
- BBSY*
- BCLR*
- BERR*
- DTACK*
- SYSCLK
- SYSFAIL*
- SYSRST*
- VRMC*/RETRY_
- $\overline{\text{L7IACF}}$

All other VMEbus signals require buffering to the VMEbus backplane to allow electrical characteristics of the signals to meet drive strength requirement. The signals remaining are:

- VA[31:0]
- VD[31:0]

- VDATAOUT
- VLWORD_
- VAM[5:0]
- VSTRBOUT
- VAS_
- VWR_
- VDS[1:0]_

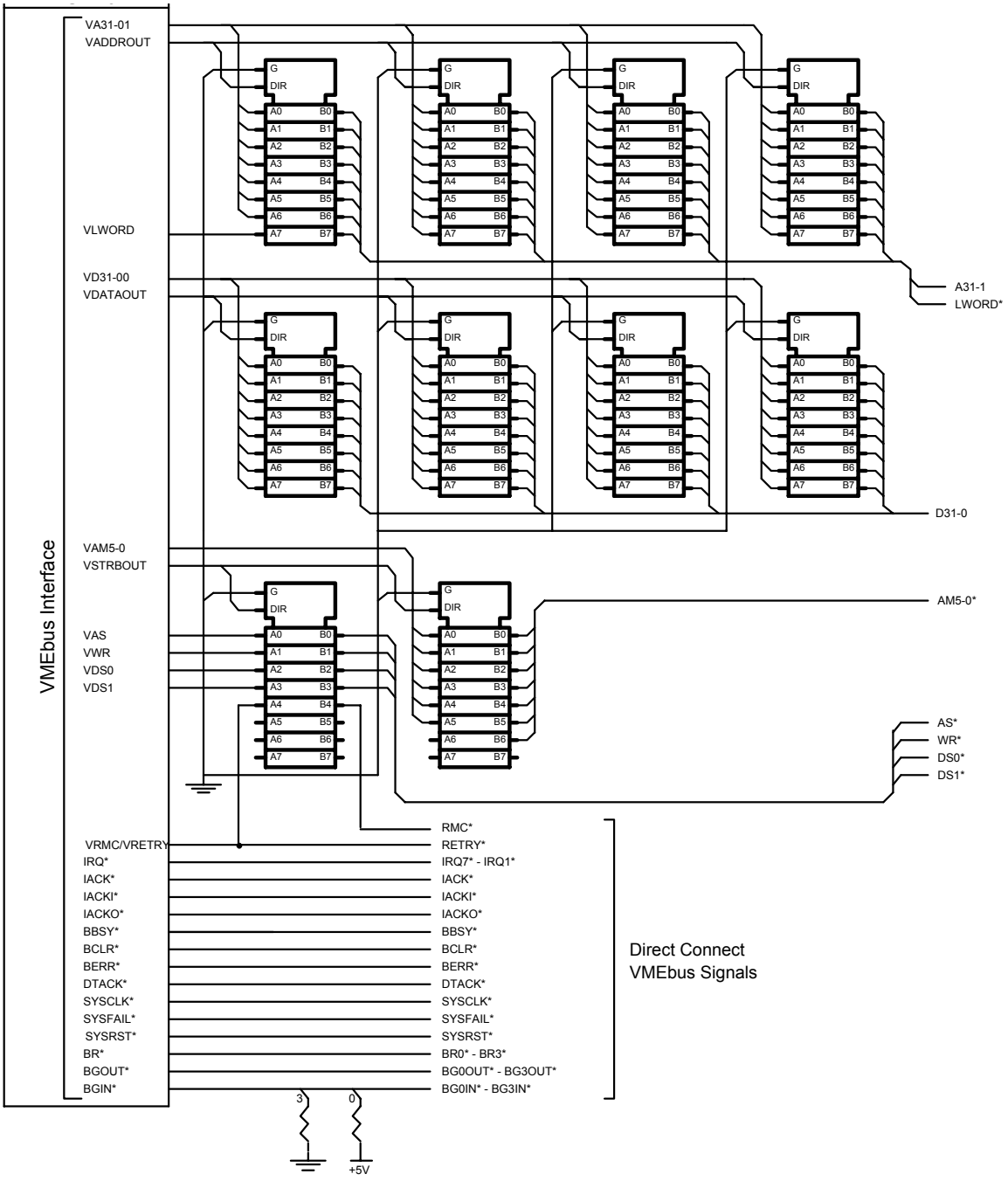
VADDROUT is the control signal required to determine the direction of the address lines during read or write transfers. VDATAOUT is the control signal required to determine the direction of the data lines during read or write transfers. VSTRBOUT is the control signal required to determine the direction of all the data transfer bus control signals. The buffer required for these signals is an ABT245 buffer.

1.6.3 SCV64 VMEbus Interface

Interfacing to the VMEbus using the SCV64 requires less space than the VIC*x* because the device has the 32-bit and 64-bit interface implemented internally. The VIC*x* devices require a multitude of multiplexers and latches to build the 32-bit address and data bus and the 64-bit data bus in order to perform 32-bit and/or 64-bit operations.

A typical SCV64 to VMEbus interface is shown in Figure 1.

Figure 1: SCV64 VMEbus Interface



1.7 Interfacing to the Local Bus

Both the SCV64 and the VIC*x* devices have Motorola MC68030 type local bus interfaces. Both devices require address decoding and arbitration circuitry on the local bus. The difference between the two devices, from a hardware perspective, is how address decoding and address latching is performed. The VIC*x* devices have only 8-bit on-chip local address and data lines. The SCV64 has 32-bit on-chip local address and data lines. The local arbitration is performed either by local logic or the SCV64 can perform the local arbitration. The VIC*x* devices do not perform local arbitration.

The SCV64's local interface require less hardware to perform 32-bit transfers since it internally possesses 32-bit address and data buses.

In the sections below, common direct connect signals to the 68030 device are outlined. All other signals require additional logic for proper operation.

For diagram descriptions of the physical interface refer to the *SCV64 Vmebus Component Interface Manual* and the *Cypress VMEbus Interface Handbook.*

1.7.1 VIC*x* Local Hardware Guideline

The VIC*x* devices can have their local signals categorized into the following groups: local address and data lines, the CPU control signals, request and grant signals, local select signals, buffer control signals, and local interrupt sources.

1.7.1.1 Local Address and Data Lines

The local address signals are required to be connected to latched transceivers to create a 32 bit address bus. The local Address lines LA[7:0] are connected to a local address decode block with the local select signals.Note the local address latches are also connected to the VMEbus address lines since 32-bit addresses are built and passed through these latches.

The local select signals are:

- CS*
- MWB*
- ASIZ[1:0]
- WORD*

The local data signals are connected to transceiver buffers to build the 32-bit local data bus. The local data transceivers are also connected to the VMEbus data lines because data is passed through these latches for 32 bit transfers.

1.7.1.2 CPU Control Signals

The following VIC*x* CPU control signals are direct connect to the MC68030 device:

- FC[2:1]
- SIZ[1:0]
- DSACK[1:0]*
- LBERR*
- R/W*
- RMC*
- IPL[2:0]
- HALT*
- RESET*
- PAS*
- DS*
- LIACKO*

1.7.1.3 Buffer Control Signals

The buffer control signals are connected to all latches and transceivers to determine direction of the address and data lines for local and VMEbus master cycles. VMEbus address and data lines share the same decode logic. The buffer control signals are describe in Table 2 on page 9.

1.7.1.4 Arbitration Signals

The arbitration signals LBR* and LBG* are connected to an external arbiter block which connects to the MC68030's BR*, BG* and BGACK* signals. The local interrupt sources can be connected directly to the CPU or routed through a local interrupt controller device.

For diagram descriptions of the physical interface refer to the *Cypress VMEbus Interface Handbook.*

1.7.2 SCV64 Local Hardware Guideline

The SCV64 local bus is composed of direct connect signals to the MC68030 and signals that interface to local address decode logic. Local address decoding is required but is not shared with VMEbus address decoding. The SCV64 performs VMEbus address decoding and data latching internally and does not require latches since it possesses a full 32-bit address and data bus.

When designing with the SCV64, designers have the option of using the SCV64's internal arbiter or using arbitration bypass. In this last case, local arbitration logic is required similar to the local arbitration logic of the VIC*x* devices. The local interrupt sources can be connected directly to the CPU or routed through a local interrupt controller device.

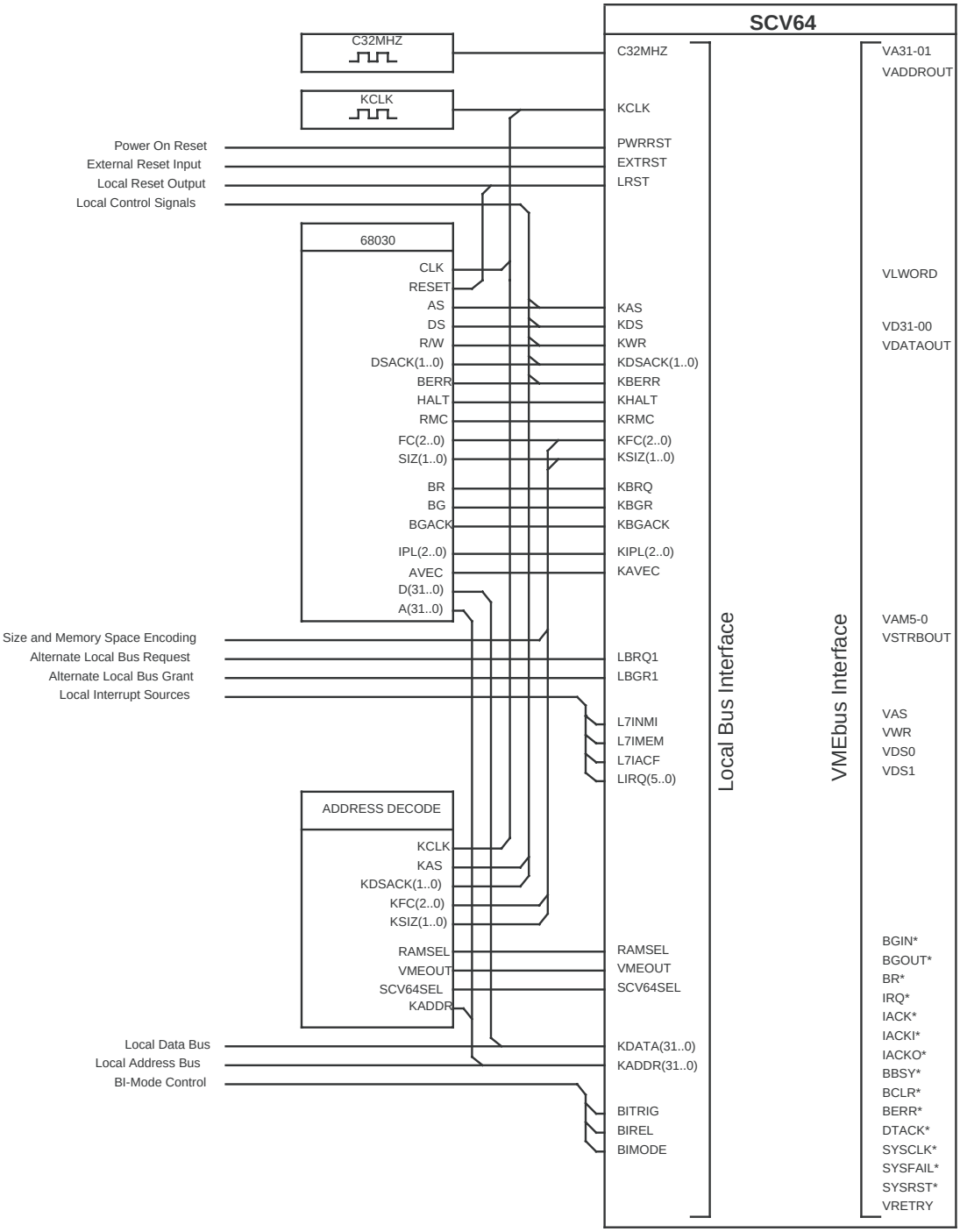
The following SCV64 signals are direct connect signals to the MC68030:

- KAS_
- KDS_
- KWR_
- KDSACK[1:0]_
- KBERR_
- KHALT_
- KRMC_
- KFC[2:0]
- KSIZ[1:0]
- KBRQ_
- KBGR_
- KBGACK_
- KIPL[2:0]_

The address decode logic monitors the KADDR[31:0], KAS_, KDSACK[1:0]_, KFC[2:0], KSIZ[1:0] signals for function type and transfer size. The SCV64 chip select's VMEOUT_, and SCV64SEL_ are generated from the local address decode logic.

A typical SCV64 to VMEbus interface is shown in Figure 2.

Figure 2: Connections for 68030 Design



1.8 Feature Comparisons

This section compares VMEbus, interrupt, and miscellaneous features between the two devices.

1.8.1 VMEbus Features

The VIC*x* and SCV64 devices can be programmed to perform arbitration and interrupt handling functions in a VMEbus system. Each device also has requester capability to acquire VMEbus mastership.

1.8.1.1 Interrupts and Interrupt Handling

The SCV64 and VIC*x* devices offer the capability of generating VMEbus interrupts and also the handling of VMEbus interrupts. In a VMEbus system, only one VMEbus interrupt handler should be present for each interrupt level.

Both devices generate VMEbus interrupts by means of a register access. The VIC*x* device's VIRSR register controls the assertion and deassertion of the seven VMEbus interrupts signals. Once the interrupt is generated, the handler proceeds in acquiring the interrupt vector which is programmed in the VIVBRx register. The VIC*x* devices only return 8-bit vectors.

The SCV64 can be programmed to generate VMEbus interrupts on any one of the seven VMEbus interrupt levels. The VINT register controls both determination of the level as well as assertion and clearing of the interrupt. Upon acknowledgement of the interrupt, the SCV64 drives an 8-bit vector from the IVECT register onto the VMEbus.

As an interrupt handler, the VIC068A device performs 8-bit interrupt acknowledge cycles. The VIC64 can be programmed to perform 8-bit, 16-bit, and 32-bit interrupt acknowledge cycles on the VMEbus.

For more information on how each interrupt handler is programmed, refer to the *Cypress VMEbus Interface Handbook* and the *SCV64 User's manual*.

1.8.1.2 Requester

The SCV64's VMEbus requester can be configured to be in Fair mode or in Demand mode by setting the REQ bit in the VREQ register. In fair mode, the SCV64 waits until there are no requests pending at its programmed level and BBSY* is inactive. In Demand mode, the SCV64 requests the bus if the following situations occur: entries exist within its transmit FIFO, a coupled cycle is initiated, or a VMEbus Interrupt Acknowledge cycle is pending.

The priority of each interrupt level is programmed by the LVL1 and LVL0 bits in the VREQ register. The SCV64 supports Release on Request (ROR) and Release When Done (RWD) modes. The SCV64 can be programmed to monitor the VMEbus clear (BCLR*) signal by setting the BCEN bit in the VREQ register.

The VIC*x* devices operate in Demand mode where a request is issued if assertion of MWB* is qualified by PAS* for single and block transfer accesses, a VMEbus interrupt acknowledge cycle is pending, and for coupled cycles.

The release modes supported by the VIC*x* are Release on Request (ROR), Release When Done (RWD), Release on Clear (ROC), Release under RMC* (Read Modify Correct) mode, and Bus Capture and Hold (BCAP) mode. In ROC mode, the VIC*x* device holds the bus until the VMEbus Clear (BCLR*) signals is asserted. In BCAP mode, the VIC*x* releases the bus only when a different release mode is programmed. In RMC mode, the assertion of RMC* and PAS* causes the VICx to request the VMEbus. Only when RMC* is released is BBSY* be removed. RMC mode is programmed by setting the ICR[5] bit in the ICR register. The VIC*x* devices can be programmed in fair mode if fairness is enabled in the ARCR[3:0] register.

For more information on how each Requester is programmed, refer to the *Cypress VMEbus Interface Handbook* and the *SCV64 User's manual*

1.8.1.3 Arbitration

Both the VIC*x* and SCV64 have the capability of being VMEbus arbiter when assuming system controller functions.

When the VIC*x* devices are the system controller, they drive the SCON* signal at all times. The VIC*x* devices support Priority mode (PRI) where BR3* has the highest priority. In Round Robin Scheme (RRS), priority is assigned on a rotating basis.

Single level arbitration can be obtained by programming the VIC*x* devices to priority mode and setting all priority levels to the same level. The VIC*x* arbiter asserts BCLR* to relinquish bus ownership of a device in PRI and RRS mode. The VIC*x* arbiter has a time-out mechanism of 8 us. This ensures that BBSY* is driven for the VMEbus required 90ns.

The SCV64 provides all arbitration modes required by the *VMEbus Specification*: Full Priority mode (PRI), Round Robin Scheme(RRS), and Single level Arbiter. Two additional mixed priority modes are provided, priority level 3, and priority levels 2 and 3.

The mode of arbitration is programmed in the VARB register by setting the ARB[1:0] bits.The SCV64 possesses an arbitration time-out of 16us, this is x2 the time allowed by the VIC*x* devices. This timer can be disabled by clearing the ATEN bit in the VARB register.

Both devices have a system time-out mechanism which ensures that if no DTACK is asserted, a BERR* is issued. The system time-out is configured in the VIC*x* devices in the TTR register. The system time-out is configured in the SCV64 in the VARB register by setting the

For more information on how each arbitration mode is programmed, refer to the *Cypress VMEbus Interface Handbook* and the *SCV64 User's manual*

1.8.2 Interrupt Features

1.8.2.1 Tick Timer

Both the SCV64 and the VICx devices offer a Tick Timer which can be enabled to interrupt the local CPU at regular intervals.

In the case of the SCV64, the TICK* pin is driven low each time the timer expires. It remains low until software resets it though the CLRTIK bit in the STAT0 register. The TICK* signal is typically connected to a general purpose interrupt on the local bus.

For the VIC*x* devices, the LIRQ2* signal can be enabled to function as the interrupt generator issuing periodic interrupts. This can be configured in the SS0CR0 register. If the Tick Timer is not enabled, LIRQ2* acts as general purpose local interrupt.

For more detailed information refer to the *SCV64 User's Manual* and the *Cypress VMEbus Interface Handbook*.

1.8.2.2 Location Monitor Accesses and Mailboxes

The VIC*x* devices do not have mailbox or location monitor capabilities.

The SCV64 has a location monitor capability which allows it to interrupt the local bus when data is written to that location.

The SCV64 also has mailbox registers but they are used to store data only; they do not generate interrupts on the local or VMEbus.

The location monitor resides at the top longword of each A32 and A24 slave images and can be accessed from both the local bus and the VMEbus. The bottom word of the location monitor is used for 16-bit messages. Since the location monitor replaces the upper longword of memory, reads to that location result in a VMEbus error and writes do not generate local accesses. A write to the location causes LMINT* to be asserted to the local CPU.

For more information on programming and usage of the Location monitor, refer to the *SCV64 User's manual*.

1.8.2.3 Local Interrupts

The VIC*x* devices can be configured through its internal registers to generate local interrupts. User defined local interrupts can be generated using the IPL lines.

The VIC*x* devices can provide the status ID vector or generate LIACKO* to signal and auto vectored interrupt. The local interrupt signals can be programmed to be level or edge sensitive and the polarity can be programmed. The VIC*x* devices can also generate local interrupts caused by internal VMEbus events. VMEbus internal events can include: VMEbus ACFAIL*, SYSFAIL*, BERR*, interrupter interrupt, and arbitration time-out. Completion of a DMA transfer can also be mapped to local interrupts. The VIC*x* device can be programmed to handle incoming interrupts.

The SCV64 provides a full set of interrupt features on the local bus through its KIPL lines, LIRQ lines, and level seven local interrupts.

The KIPL signals encode seven possible CPU interrupt levels. Level seven local interrupts, local, and interrupts from VMEBus sources are all mapped to a common set of CPU interrupt levels. Level seven interrupts such as L7NMI*, L7IMEM*, and L7IACF* are automatically mapped to the highest CPU level while VMEBus interrupts are all pre-mapped to their corresponding CPU interrupt levels. The local interrupts can be programmed to different CPU levels via the ICx registers.

The SCV64 propagates any incoming interrupt to the CPU via its KIPL lines. The CPU generates an IACK cycle which the SCV64 can decode. The SCV64 is capable of handling either internal or external local IACK decoding.

For more information on how to program the SCV64's interrupts, refer to the S*CV64 User's Manual*.

1.8.3 Miscellaneous Features

This section compares miscellaneous features between the SCV64 and VIC*x*.

1.8.3.1 Deadlock resolution

In a VMEbus system, deadlock conditions occur when the local CPU is trying to access the VMEbus while there is an incoming VME slave cycle that has been properly signaled. This condition usually occurs during coupled accesses to the VMEbus such as incoming VME read, coupled master write cycles to the VMEbus or Interrupt Acknowledge Cycles (IACK). The SCV64 has the capability of handling these conditions using local bus terminations. If these terminations are not successful, deadlock conditions can be resolved by allowing the VMEbus time-out mechanism to signal a BERR* condition.

When the VIC*x* devices encounter a deadlock scenario, they can be programmed in the ICR register to perform one of the following operations:

- Assert DEDLK*
- Assert DEDLK*, LBERR*, and HALT*
- Assert DEDLK*, and LBERR with HALT* during local Read Modify Write Cycles (RMC).

The SCV64 only has to assert KBERR* and KHALT* to resolve this condition. The KBERR* and KHALT* assertion is compliant to the Motorola 68K protocol which interprets this termination as a Retry mechanism.

1.8.3.2 VME Master Block Operation

As a VME master, the VIC*x* devices are capable of generating VMEbus master block transfers using the MOVEM command. In this mode, the local resource configures the VIC*x* for a 68K type MOVEM block transfer and proceeds with consecutive- address cycles. The CPU remains local bus master until the cycle completes. This operation is similar to a DMA transaction, but the burst length is programmed in the RCR register and the block transfer is configured in the BTCR register.

As a VME master, the SCV64 perform block transfers only when using its internal DMA. For more details on how to use the SCV64's DMA, refer "DMA Operation" on page 21.

1.8.3.3 DRAM Refresh Controller

The VIC*x* devices contain a DRAM refresh controller. The VIC*x* increments a DRAM refresh counter every 15us, when the refresh is enabled in the ARCR register.

The SCV64 does not have a DRAM refresh controller.

1.8.3.4 Additional SCV64 features

This section describes features available with the SCV64 but not supported by the VICx devices.

RETRY/VRMC Function*

The SCV64 RETRY*/VRMC pin can 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, is should be routed through the strobes buffer (along with VAS, VDS0, VDS1, etc.) with the direction controlled by VSTRBOUT.

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 following:

- 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 can be any frequency up to the specified limit.

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 enters BI-mode whenever one of the BI-mode initiators is active, but automatically returns to an operational state when the initiator is removed. Initiators of BI-mode include:

- Any reset
- IRQ1*: when configured as a BI-mode initiator
- 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 remains 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 can see erratic entry into BI-mode until IRQ1* is re-configured as a VME interrupt source using the VI1BI bit in the GENCTL register.

Auto-ID Mechanism

The SCV64 has a proprietary Auto-ID mechanism which identifies the relative position of each board 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.

The benefits of bypassing the use of jumpers through Auto-ID include:

- Increased speed of system level repairs in the field
- Reduced possibility of incorrect configurations
- Reduced number of unique spare cards that must be stocked

The SCV64 Auto ID mechanism is not compliant to the *VME64 Specification* Auto-ID mechanism.

1.9 Resets

1.9.1 VIC*x* Resets

The VIC*x* devices are reset by any of the following conditions: global reset, internal reset, system reset, or power-on reset. A system reset is performed by accessing one the VIC*x*'s internal register.

1.9.1.1 Global Reset

Global reset provides the most complete reset; all of the VICx circuitry is reset. A global reset is initiated when ILP0 is asserted after the IRESET* signal is asserted. During a global reset, SYSCLK is not driven and the daisy chain is disconnected.

VIC*x* devices must be globally reset at power-up for proper operation.

1.9.1.2 Internal Reset

The internal reset is initiated by asserting IRESET* for a minimum of 20ns. The VICx then asserts LBR* and waits 1us for the assertion of LBG*. If LBG* is asserted within the 1us, the VIC*x* asserts HALT* and RESET* immediately.

During the internal reset, the VICx places all three state outputs to a high-Z state and begins a 200ms time-out period. If IRESET* is still asserted, the VIC*x* begins an additional 200 ms time-out. If the VIC*x* device is system controller, it also asserts SYSRESET* with RESET* and HALT* for 200ms (required by the *VMEbus Specification*).

1.9.1.3 System Reset

A VMEbus system reset is signaled by the assertion of the VMEbus signal SYSRESET*. The system reset is identical in function to the internal reset, except for the following: the reset state of some registers are different, if SYSRESET* is still asserted after 200 ms, the reset completes once de-asserted and does not perform another internal reset.

1.9.1.4 Power-on Reset

To reliably reset the VIC*x* at power on, a global reset must be performed. The VIC*x* device must be in a stable operating environment at the initiation of reset. The VIC*x* is considered to be in a stable operating environment when VCC is stable at 5 V, the input clocks are within their operating range, and all inputs are inactive. The power-on reset circuitry must be designed to assert the IPL0 signal for a minimum of 50ns.

For more information on the internal reset scheme of the VIC*x* device, refer to the *Cypress VMEbus Interface handbook*.

1.9.2 SCV64 Resets

The SCV64 can be reset through the following sources: PWRRST*, EXTRST*, SYSRST*, a software reset, and, when programmed System controller, with the BG0IN* signal.

The PWRRST* input directly resets all circuits in the SCV64 and should be held for a minimum of 100 ms once the SCV64 is in stable operating conditions. The SCV64 is considered to be in a stable operating environment when VCC is stable at 5V, the input clocks are within their operating range, and all inputs are inactive. Reset through the PWRRST* pin set the PWRUP bit in the STAT1 register. This bit is cleared under all other reset conditions.

The SCV64 asserts SYSRST* if PWRRST* is asserted, the BG0IN* pin is asserted while System controller or if the SWRST bit in the GENCTL register is set. SYSRST* will be asserted for a minimum of 0.25 seconds.

The local reset LRST* output is asserted to the local bus when PWRRST*, EXTRST*, SYSRST*, a software reset, and if BG0IN* is asserted.

1.10 VIC*x* and SCV64 Comparison Summary

Table 5 outlines the main functional features supported by the SCV64 and VIC*x* devices.

Table 5: Functional Overview

Functional Feature Set Supported	VIC*x*	SCV64	Comments
Internal DMA	Yes	Yes	Both DMA engines can be programmed to perform VMEbus block transfers. The SCV64's DMA can also perform single cycles. Section 1.6 reviews this in detail.
VMEbus Block Transfers (Reads/Writes)	Yes	Yes	These are performed using the DMA engine. The VIC64 also supports a MOVE block command from the CPU which allows it to perform BLTs as a local slave, the SCV64 does not support this.
Data Path FIFO based	No	Yes	The SCV64's internal architecture is FIFO based. This allows for decoupled accesses. The VIC64 only provides a single register for write posting.
Register Accesses from the VMEbus	Partially	Yes	The SCV64 and VICx registers can be accessed via the local bus. The SCV64 has the added functionality to access registers from the VMEbus. The VIC ICF registers can be accessed from the VMEbus.
Reflected Cycle	No	Yes	The SCV64 supports reflected cycles where the SCV64 will respond to its own VMEbus base address. This is useful for loopback diagnostics.
Rescinding DTACK*	Yes	No	The VIC64 supports rescinding DTACK* whereby the signal is actively driven high.
VMEbus Address decode	No	Yes	The SCV64 has a built in address decoder on the VMEBus interface. This simplifies a board design since no address decode logic is required externally.
Auto System Controller detection	No	Yes	The SCV64 will automatically detect system controller functionality by monitoring its BGIN3* input.
VMEbus Interrupt Handling and Generating	Yes	Yes	Both devices comply to the VME64 specification. The VICx devices can support 16, and 32 bit vectors. The SCV64 will provide 8 bit vectors.

Table 5: Functional Overview

Functional Feature Set Supported	VIC*x*	SCV64	Comments
VMEbus Requester	Yes	Yes	Both devices comply to the VME64 specification. Refer to Section 1.8.1.1 of this document
VMEbus arbiter	Yes	Yes	Both devices comply to the VME64 specification. Refer to section 1.8.1.3 of this document
Local Bus Interrupts	Yes	Yes	Both devices support the IPL interrupt lines as defined by the MC68030 device. Refer to section 1.8.2.3 of this document.
Mailbox Registers	No	Yes	The SCV64 Mailbox register will store data only and will not generate interrupts. The VICx does not possess mailbox registers.
Location Monitors	No	Yes	In the SCV64, the location monitor resides at the top longword of each of the A32 and A24 images.
Tick Timer	Yes	Yes	Refer to section 1.8.2.1 of this document.
Watchdog timer	No	Yes	The SCV64 has a watchdog timer which runs for 2 sec. and then asserts the WDOG_ signal. This timer can be disabled by clearing the ENDOG bit in the MISC register.
Bus Isolation Mode	No	Yes	The SCV64 can be placed in Bus Isolation mode. The VIC devices do not support this feature. Refer to section 1.8.3.4 of this document.
DRAM Refresh Controller	Yes	No	The VICx can perform a DRAM refresh every 15us.
Retry	No	Yes	Refer to section 1.8.3.4 of this document.
Auto ID	No	Yes	The SCV64 has a proprietary Auto-ID mechanism. Refer to section 1.8.3.4 for more information.
VME64 Auto-ID	No	No	Neither devices implemented this VME64 feature.

1.11 Conclusion

Although the SCV64 and the VIC*x* (VIC068A/VIC64) have similar bus interfaces, it does not mean that the handling of all operations are performed identically. The VIC*x* and SCV64 are complete VMEbus and 68030 interface solutions, but both require unique hardware and software design approaches.

This document outlined the major feature differences, signal descriptions, and hardware implementations to allow the SCV64 to replace aVIC device in an existing design. The main advantage of the SCV64 from a hardware approach is that less external logic is required due to its full 32-bit local and VME buses. The only external logic required with the SCV64 is for local address decoding.

The *SCV64 User's Manual* offers a comprehensive description of all timing and functional features of the device. Tundra Semiconductor's Applications Engineering team is available for any additional questions through support@tundra.com or by visiting www.tundra.com.