VIC068A
VAC068A

USER'S Guide

CYPRESS SEMICONDUCTOR

VIC068A
VAC068A
CYPRESS
USER'S Guide
CYPRESS SEMICONDUCTOR

VIC068A/VAC068A

User's Guide

Cypress Semiconductor, 3901 North First St., San Jose, CA 95134 (408) 943–2600
Telex: 821032 CYPRESS SNJ UD, TWX: 910 997 0753, FAX: (408) 943–2741

Published June 1992

VIC068A and VAC068A are trademarks of Cypress Semiconductor.
68K as it refers to the MC68000 family of CISC microprocessors is a trademark of Motorola, Inc.

Contents

Figures

Tables

How to Use This Guide

This guide provides the hardware and software designer with detailed information on the Cypress Semiconductor VIC068A VMEbus Interface Controller and the VAC068A VMEbus Address Controller. It may also be used to provide detailed information regarding the VIC068A for existing off-the-shelf VMEbus modules that utilize the VIC068A.

This document is not intended to instruct the reader on VMEbus standards and protocol. First-time VMEbus designers and users requiring such information are encouraged to refer to the VMEbus specification (IEEE P1014).

Throughout this specification, specific conventions are used when referring to VMEbus signals, terms, and register bit and bit fields.

- The terms High or H are used to specify actual $>V_{IH}$ or $>V_{OH}$ levels. The terms Low or L are used to specify actual $<V_{IL}$ or $<V_{OL}$ levels. See Chapter 24 for DC specifications.
- Active Low signals are followed by an asterisk (*).
- Active High signals, clock signals, and address/data buses do not have an asterisk.
- The terms *assertion* and *deassertion* are used to indicate the forcing of a signal to a particular state. Assertion means forcing a signal to its TRUE or active state. Deassertion refers to forcing a signal to its FALSE or inactive state. These terms are used independent of the actual voltage levels represented.
- Address and data buses (or portions thereof) are referred to using a bus[MSB:LSB] format. For example, the entire VMEbus data bus is referred to as D[31:0].
- An individual bit of an address or data bus is referred to using a bus[bit] format. For example, bit 0 of the local address bus is referred to as LA[0] or, where space was restrictive, LA0.
- When referring to address and data buses with a user-specific limit, a "+" character is used to indicate the limit. For example, to refer to the range from LA bit 0 to some user-specified or unknown limit, the term LA[+:0] is used. LA bit 31 to a lower user-specified or unknown limit is referred to as LA[31:+].
- When referring to one or more related signals or registers containing numbers, the lowercase letter "i" is used to indicate the signal(s). For example, when referring to one or

more of the VMEbus bus request signals (BR3*, BR2*, BR1*, and/or BR0*), the term BRi* is used. When referring to the SS0CR0 and/or the SS1CR0 register, the term SSiCR0 is used.

- When referring to a specified group of signals ending in a number, a slash (/) is used to separate the signals. For example, when referring to the SIZ1 and SIZ0 signals, the term SIZ1/0 is used.
- Specific bits of a VIC068A register are referred to in a register[bit] format. Ranges of bits are referred to in a register[upper:lower] format.
- Setting register bit or bits refers to writing a 1 (one) into the respective bits.
- Clearing register bit or bits refers to writing a 0 (zero) into the respective bits.
- The term *module* refers to a VMEbus circuit board. Depending on the context, module may or may not imply a VMEbus circuit board that utilizes the VIC068A.
- The terms *local* or *local side* refer to CPU, memory, or other resources that connect to the non-VMEbus signals of the VIC068A.
- The terms *master write* and *slave write* both imply a VMEbus write operation where data is transferred from a VMEbus master to a VMEbus slave. *Master read* and *slave read* both imply VMEbus read operations where data is transferred from a VMEbus slave to a VMEbus master.
- All hexadecimal values are preceded by a dollar sign ($).
- The term *byte* is used to indicate 8 bits. The term *word* is used to indicate 16 bits. The terms *longword* and *lword* are used to indicate 32 bits.
- The term 68K is used to indicate a member of the Motorola CISC family of microprocessors (i.e., MC68000 through MC68040).
- All times specified (except for propagation delays) assume the VIC068A is operating at 64 MHz (CLK64M = 64 MHz).
- The letter "T" is used to indicate CLK64M clock input period.
- The term *rescinding* is used to indicate a three-state output that is driven High before it is three-stated. See section 12.4.
- The letters "L" and "H" are used to indicate a High or Low value driven *by* the VIC068A. The numbers "1" and "0" are used to indicate a High or Low value driven *to* the VIC068A.

Section 1 contains information on VIC068A, Section 2 on VAC068A, and Section 3 contains DC parameters and package information, which is common to both parts.

Section 1

VIC068A

1 Introduction to the VIC068A

1.1 Description

The Cypress Semiconductor VMEbus Interface Controller, VIC068A, is a single, integrated circuit designed to minimize the cost and board-area requirements of VMEbus boards, while at the same time maximizing their performance. The VIC068A was designed using Cypress's high-performance standard cells on a 1-micron CMOS process. The VIC068A provides all VMEbus system controller functions plus many other features that simplify the development of VMEbus-based modules. The VIC068A utilizes Cypress's patented and military-approved high-drive CMOS drivers. These CMOS drivers connect directly to the VMEbus signal pins.

The VIC068A was developed through the joint efforts of Cypress Semiconductor and the VMEbus Technology Consortium under the auspices of the VMEbus International Trade Association (VITA). Because of this cooperation, the VIC068A offers an implementation that provides the broadest feature set and multi-vendor compatibility available on the market.

A block diagram of the VIC068A is shown in *Figure 1–1*. A typical 68030 application is shown in *Figure 1–2*.

1.2 Features Summary

The complete VMEbus Interface Controller and Arbiter includes

- PRI, SGL, and RRS arbitration
- the capability to drive arbitration signals directly
- arbitration timeout timer
- VMEbus timeout timer
- the capability to drive BGOUT*, IACK* daisy-chain

The complete VMEbus Master Interface includes

- five release modes
- write posting
- indivisible cycle support
- deadlock detection
- fair requesting
- user-defined AM code generation

The complete VMEbus Slave Interface includes

- write posting
- configurable local access timing
- slave block transfer support

Interleaved Block Transfer support includes

- block transfers with local DMA
- programmable transfer length, burst length, interleave period, and access timing
- "dual-path" option
- local, module-based DMA support

The compete VMEbus, Local Interrupt Handler/Generator includes

- seven local interrupt signals
- seven VMEbus interrupt signals
- seven-level local encoding
- error/status interrupts
- periodic "heartbeat" interrupt

Interprocessor Communication Support includes

- four global mailbox interrupts
- four module mailbox interrupts
- five mailbox registers

Other features include

- local DRAM refresh control
- local timeout timer
- "turbo" mode
- programmable metastability delay

The VIC068A meets the IEEE VMEbus Specification 1014 Rev C.1.

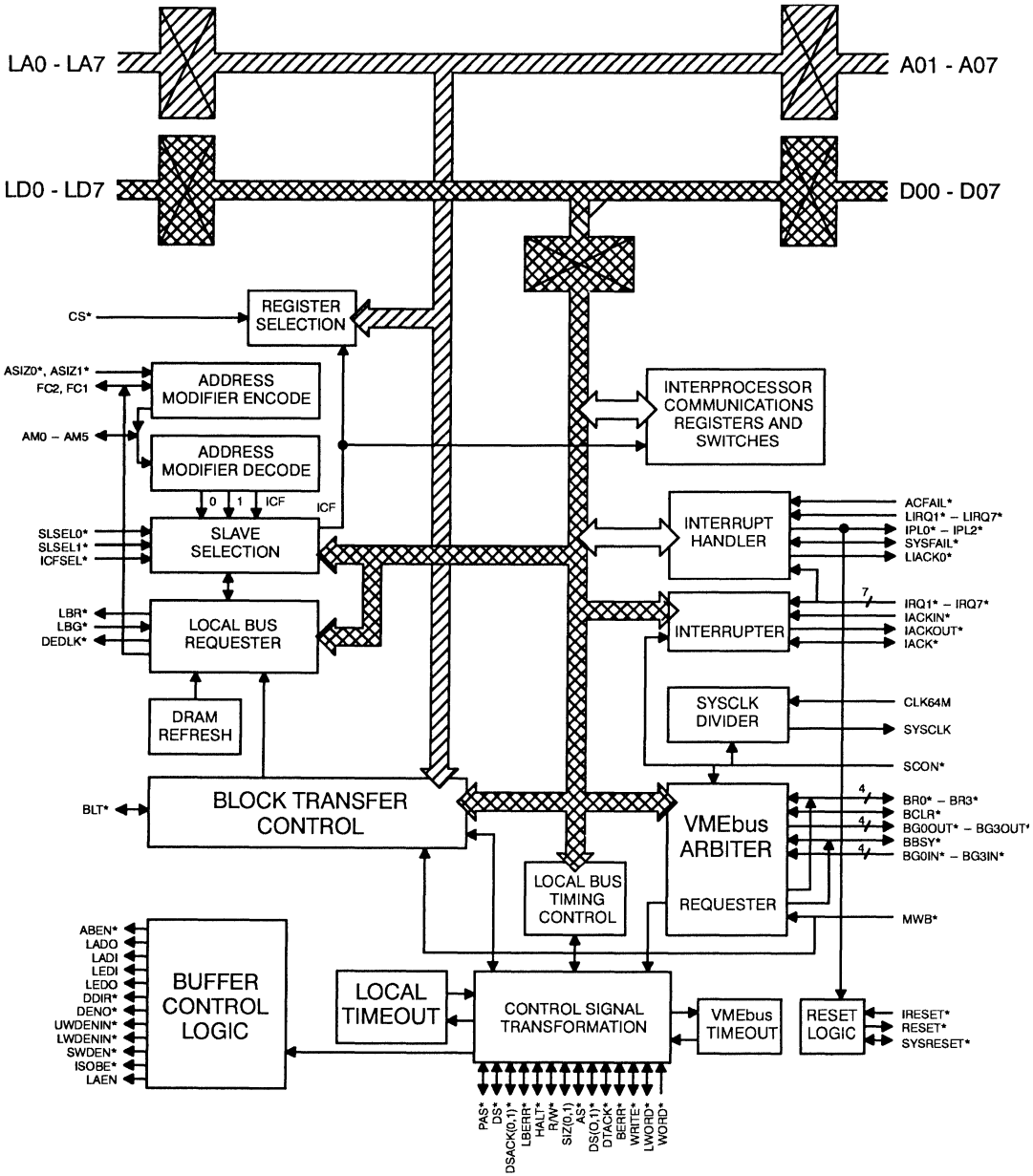


Figure 1–1. VIC068A Block Diagram

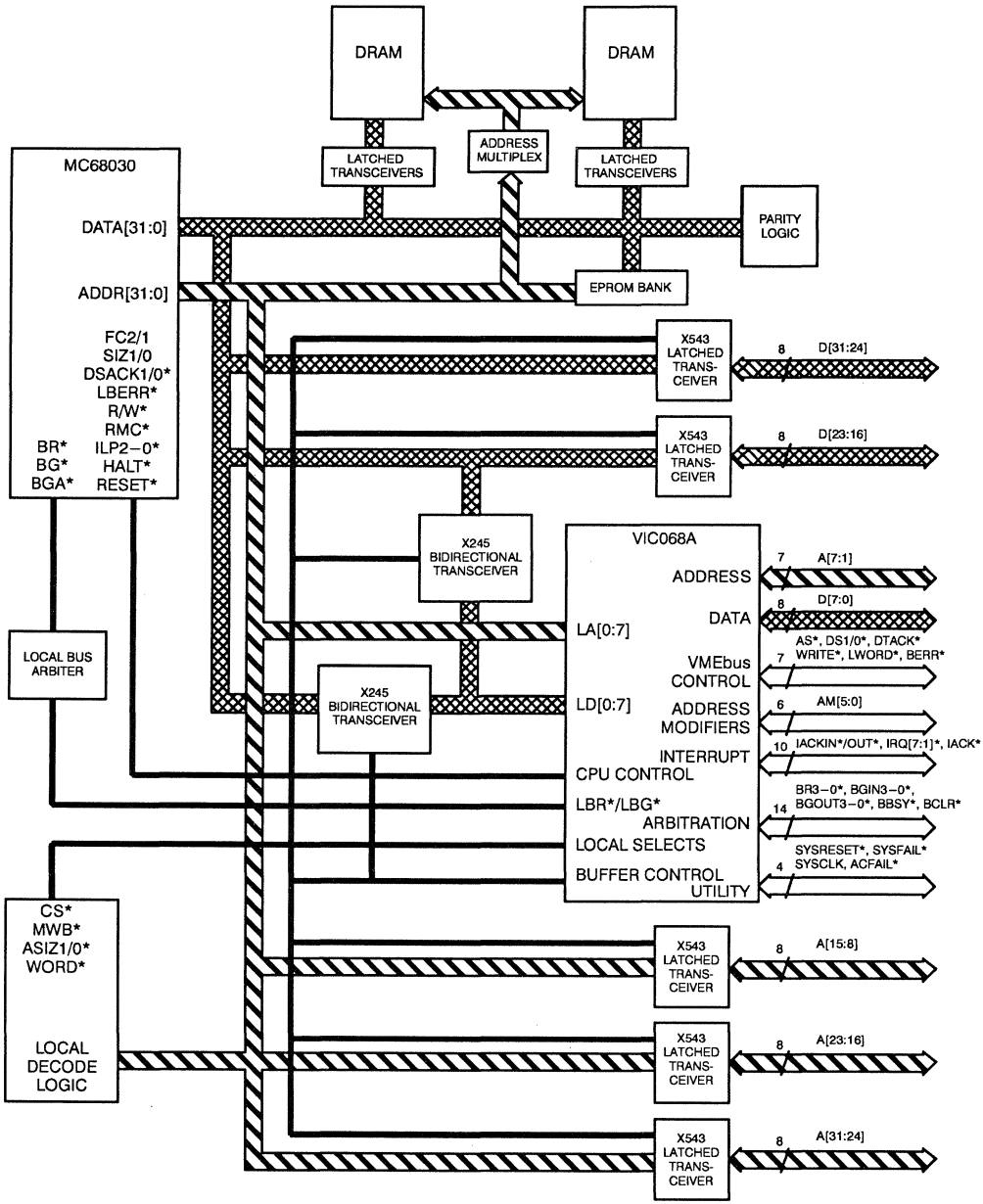


Figure 1–2. VIC068A on 68030 Board

2

VIC068A Signal Descriptions

2.1 VMEbus Signals

This chapter lists VMEbus-specified signals that are driven and received directly by the VIC068A. For complete definitions and descriptions of these signals, refer to the VMEbus specification (IEEE 1014).

SYSRESET*

Input: Yes
Output: Yes, open collector
Drive: 64 mA

This is the VMEbus system reset signal. A Low level on this signal resets the internal logic of the VIC068A and asserts the signals HALT* and RESET*. These signals remain asserted for a minimum of 200 ms. If the VIC068A is configured as VMEbus system controller, a Low level on IRESET* asserts SYSRESET* for a minimum of 200 ms.

ACFAIL*

Input: Yes
Output: No
Drive: None

This is the VMEbus AC fail signal. This signal should be driven by the VMEbus power monitor (if installed). The VIC068A can be enabled to provide a local interrupt when this signal is asserted.

SYSFAIL*

Input: Yes
Output: Yes, open collector
Drive: 64 mA

As an output, the SYSFAIL* signal is asserted when it detects that HALT* has been asserted for more than 4 ms by a source other than the VIC068A.

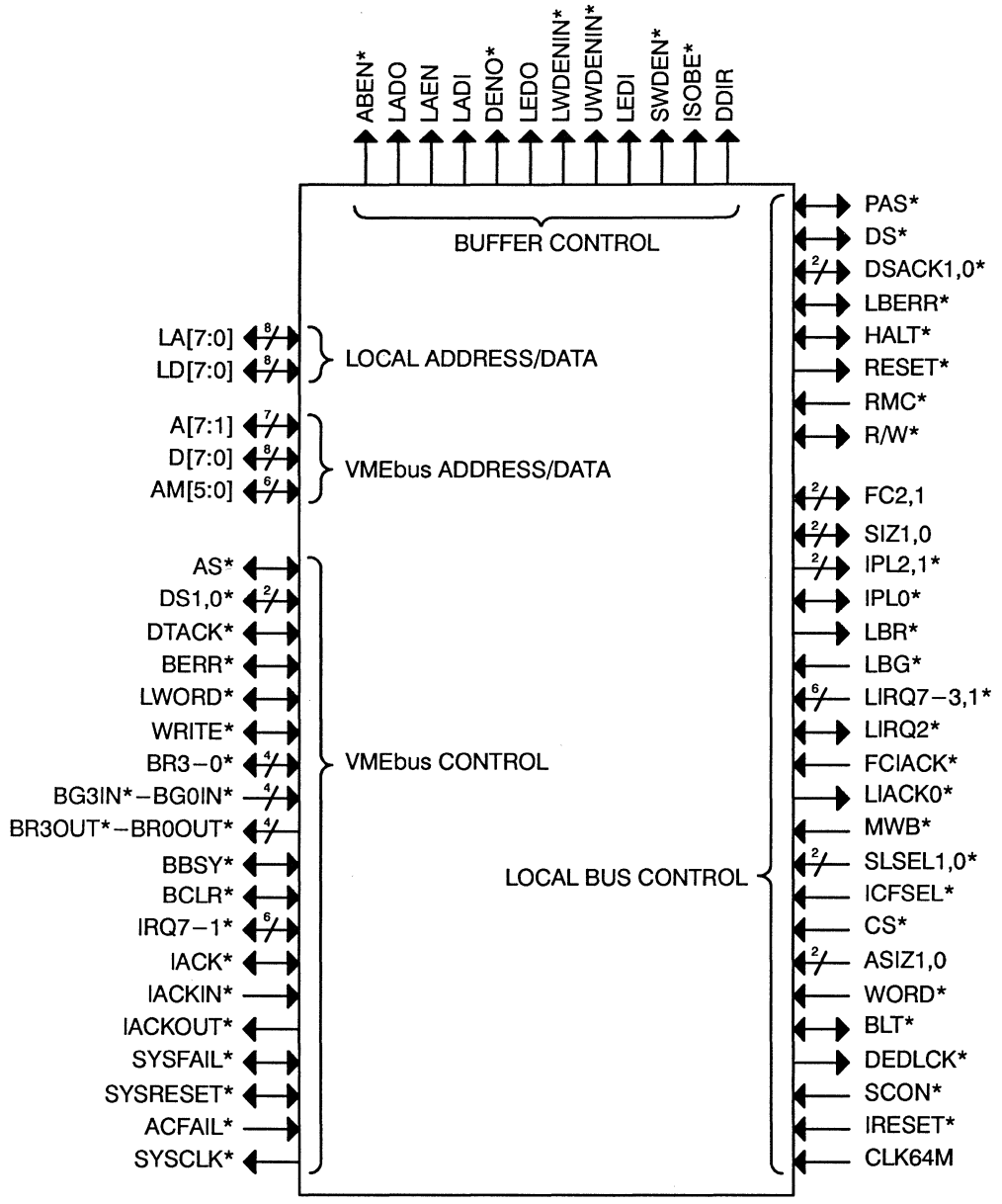


Figure 2–1. VIC068A Signal Diagram

This signal is asserted by the VIC068A after a global reset. It may be masked by clearing ICR6[6] or by setting ICR7[7]. The VIC068A can also be enabled to provide a local interrupt on the assertion of this signal.

SYSCLK

Input: No
Output: Yes, three-state
Drive: 64 mA

This is the VMEbus system clock signal. This signal is driven by the VIC068A when configured as system controller (SCON* asserted). The output frequency is one-fourth the frequency delivered to the VIC068A CLK64M signal. To deliver the required 16 MHz on this signal, the VIC068A must run at 64 MHz. The VIC068A does not use this signal internally.

BR3* – BR0*

Input: Yes
Output: Yes, open collector
Drive: 64 mA

These are the VMEbus Bus Request signals.

BG3IN* – BG0IN*

Input: Yes
Output: No
Drive: None

These are the VMEbus daisy-chained Bus-Grant-In signals.

BG3OUT* – BG0OUT*

Input: No
Output: Yes
Drive: 8 mA

These are the VMEbus daisy-chained Bus-Grant-Out signals.

BBSY*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Bus-Busy signal.

BCLR*

Input: Yes
Output: Yes, three-state
Drive: 64 mA

This is the VMEbus Bus-Clear signal.

D7 – D0

Input: Yes
Output: Yes, three-state
Drive: 64 mA

These are the VMEbus low-order data lines.

A7 – A1

Input: Yes
Output: Yes, three-state
Drive 64 mA

These are the VMEbus low-order address lines.

AS*

Input: Yes
Output Yes, rescinding
Drive: 64 mA

This is the VMEbus Address Strobe signal.

DS1* – DS0*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

These are the VMEbus Data Strobe signals.

DTACK*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Data-Transfer-Acknowledge signal.

BERR*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Bus-Error signal.

WRITE*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Data-Direction signal.

LWORD*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Longword signal.

AM5 – AM0

Input: Yes
Output: Yes, three-state
Drive: 64 mA

These are the VMEbus Address-Modifier signals.

IACK*

Input: Yes
Output: Yes, rescinding
Drive: 64 mA

This is the VMEbus Interrupt Acknowledge signal.

IACKIN*

Input: Yes
Output: No
Drive: None

This is the VMEbus daisy-chained Interrupt-Acknowledge-In signal.

IACKOUT*

Input: No
Output: Yes
Drive: 8 mA

This is the VMEbus daisy-chained Interrupt-Acknowledge-Out signal.

IRQ7* – IRQ0*

Input: Yes
Output: Yes, open collector
Drive: 64 mA

These are the VMEbus Interrupt request signals.

2.2 Local Signals

These signals define the local bus structure of the VIC068A. They are modeled after Motorola 68K signals.

LD7 – LD0

Input: Yes
Output: Yes, three-state
Drive: 8 mA

These are the Local Data 7–0 signals. These signals are typically connected to the local processor data lines D[7:0] through an isolation buffer. VIC068A register accesses are also made through these data signals.

LA7 – LA0

Input: Yes
Output: Yes, three-state
Drive: 8 mA

These are the Local Address 7–0 signals. These signals are typically connected to the local processor address lines. VIC068A registers are also addressed through these signals. When acting as the local bus master, the VIC068A drives these lines with the LAEN* signal to supply the local address.

CS*

Input: Yes
Output: No
Drive: None

This is the VIC068A chip select signal. This signal should be asserted whenever access to the VIC068A internal registers is required.

PAS*

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

This is the physical/processor address strobe. This signal is used to qualify an incoming address when performing VMEbus master operations or register operations. This signal is driven when performing slave transfers, DRAM refresh, slave block transfers and block transfers with local DMA. When acting as an output, the minimum assertion and negation timing for this signal is configured by the Local Bus Timing register.

DS*

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

This is the local data strobe. This signal is used to qualify incoming data when performing VMEbus master operations or register operations. This signal is driven when performing slave transfers, DRAM refresh, slave block transfers, and block transfers with local DMA. When acting as an output, the minimum assertion and negation timing for this signal is directed by the Local Bus Timing register.

DSACK1*, DSACK0*

Input: Yes
Output: Yes, three-state
Drive: 8 mA

These are the local data-size-acknowledge signals. One or both of these signals should be asserted to the VIC068A whenever the VIC068A is local bus master to acknowledge the successful completion of each cycle of a slave transfer, slave block transfer, or block transfers with local DMA. The VIC068A asserts one or both of these signals to acknowledge the

successful completion of a VMEbus master operation (after receiving the VMEbus DTACK* signal). The following should be noted about the DSACK1/0 signals:

- The VIC068A asserts a 16-bit DSACKi* code when the WORD* signal is asserted, indicating access to a D16 VMEbus resource is complete.
- The VIC068A treats the assertion of any DSACK1/0* signal as a 32-bit acknowledge for slave accesses.
- The VIC068A does not directly support 16- or 8-bit local bus sizes.
- The VIC068A always asserts both DSACKs for register accesses as well as for interrupt acknowledge cycles.

LBERR*

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

This is the local bus-error signal. This signal should be asserted to the VIC068A whenever the VIC068A is local bus master to acknowledge the unsuccessful completion of a slave transfer, slave block transfer, and block transfers with local DMA, in which case the VIC068A asserts the VMEbus BERR* signal. The VIC068A asserts this signal to acknowledge the unsuccessful completion of a VMEbus master operation (after receiving the VMEbus BERR* signal).

LBERR* may also be configured to assert with the HALT* signal to initiate a Motorola 68K retry sequence. LBERR* may also be configured to assert without HALT* for RMC cycle deadlocks.

RESET*

Input: No
Output: Yes, three-state
Drive: 8 mA

This is the local reset indication signal. This signal is asserted whenever the VIC068A is in a reset state. An internal, global, or system reset causes the VIC068A to start its 200-ms reset timer and to assert RESET* for a minimum of one rest timer period. If a reset condition is present at the end of the reset timer period (200 ms), the reset timer is retriggered for an additional 200-ms period and continues to assert RESET*. This reset timer retrigger operation repeats until the rest condition is not present when the reset timer period ends.

HALT*

Input: Yes
Output: Yes, three-state
Drive: 8 mA

This is the "halted" condition indication signal. This signal, along with RESET*, is asserted during reset conditions. An internal, global, or system reset causes the VIC068A to assert HALT* for a minimum of 200 ms. If the reset condition continues for longer than 200 ms, HALT* begins additional 200-ms timeouts until all reset conditions are cleared. Assertion of HALT* for more than 6 μs by anything other than the VIC068A causes the VIC068A to assert SYSFAIL*.

HALT* may be configured to assert during deadlock conditions along with LBERR* to initiate a retry sequence for Motorola 68K processors.

R/W*

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

This is the local data direction signal. This signal is driven while the VIC068A is a local bus master to indicate local data direction. As an input, R/W* indicates data direction for VMEbus master cycles. In this case, WRITE* reflects the value of R/W*. A Low condition indicates a write operation.

FC2, FC1

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

These are the local function code signals. These signals identify the type of local cycle in progress. As inputs, they should reflect the type of operations in terms of User/Supervisory Code/Data. They may be connected directly to the Motorola FC2/1 outputs for 68000-30 processors. For the 68040, the FC2/1 inputs may be connected to the TM2/1 outputs, respectively. Additional qualification may be required for 68040 applications because the 68040 uses previously reserved/unused function codes.

FC2	*FC1*	*Description*
0	0	User Data
0	1	User Program

1	0	Supervisor Data
1	1	Supervisor Program

As outputs, the VIC068A drives these signals whenever it is local bus master to indicate the type of local cycle the VIC068A is performing.

FC2	*FC1*	*Description*
0	0	Slave Block Transfer
0	1	Local DMA
1	0	Slave Access
1	1	DRAM Refresh

SIZ1, SIZ0

Input: Yes
Output: Yes, rescinding
Drive: 8 mA

These are the local data size signals. As inputs, these signals identify the width of the VME-bus data to be transferred. The SIZi signals should not be used to indicate the physical port size of the slave device (D16, or D32). This is done with the WORD* signal. As outputs, they are driven by the VIC068A as local bus master to identify the width of the incoming data.

SIZ1	*SIZ0*	*Data Width*
0	0	Longword
0	1	Byte
1	0	Word
1	1	3-Byte

LBR*

Input: No
Output: Yes
Drive: 8 mA

This is the local bus request signal. This signal is asserted whenever the VIC068A desires mastership of the local bus. This signal remains asserted for the entire bus tenure.

Local bus mastership is requested when each of the following operations is desired:

- Standard slave accesses

- Slave block transactions
- Block transfers with local DMA
- DRAM refresh

LBG*

Input: Yes
Output: No
Drive: None

This is the local bus grant signal. The signal is asserted by local resources in response to the LBR* signal. The VIC068A does not incorporate a local-bus-grant-acknowledge protocol, so the LBG* signal should remain asserted for the duration of LBR*.

MWB*

Input: Yes
Output: No
Drive: None

This is the "Module-Wants-Bus" signal. This signal is asserted by local resources to begin a VMEbus transaction. When qualified by the PAS* signal, the VIC068A asserts the VMEbus BRi* signal. This signal is usually asserted by local-to-VMEbus address decoders.

FCIACK*

Input: Yes
Output: No
Drive: None

This is the local interrupt acknowledge signal. This signal is asserted (qualified by PAS*) to acknowledge all VIC068A-generated local interrupts.

SLSEL1*, SLSEL0*

Input: Yes
Output: No
Drive: None

These are the slave select signals. These signals indicate the VIC068A has been selected to perform a VMEbus slave operation. When qualified by AS* and valid AM codes, the VIC068A requests the local bus to perform the slave cycle. These signals are usually asserted by VMEbus-to-local-address decoders.

The SLSEL1/0 signals may be used independently of each other to provide unique slave characteristics as defined by the Slave Select Control registers.

ICFSEL*

Input: Yes
Output: No
Drive: None

This is the Interprocessor Communication Facility (ICF) Select signal. This signal indicates that the ICF functions of the VIC068A have been selected. These include the ICF registers and the ICF switch interrupts. This signal is qualified with AS* and A16 AM codes (A16/Supervisory for global switches).

ASIZ1, ASIZ0

Input: Yes
Output: No
Drive: None

These are the VMEbus address size signals. These signals are driven to indicate the VMEbus address size of master VMEbus transfers. The address size information is issued on the VMEbus AM codes. User-defined address spaces may be accessed by asserting both ASIZ1/0 signals. In this case, the AM codes are issued according to the programming of the Address Modifier Source register.

ASIZ1	*ASIZ0*	*Address Size*
0	0	User defined
0	1	A32
1	0	A16
1	1	A24

The ASIZ1/0 signals are also used as cycle acknowledge signals for module-based DMA transfers. During a module-based DMA transfer, the ASIZ0 signal is used as a data-transfer-acknowledge signal (analogous to DTACK*). The ASIZ1 signal is used as a bus-error signal (analogous to BERR*).

WORD*

Input: Yes
Output: No
Drive: None

This is the VMEbus data width control signal. This signal, when asserted, indicates the requested VMEbus transaction should be treated as a D16 data path. When deasserted, the

VMEbus data path is assumed to be D32. This signal should be used to configure VMEbus data width for master cycles only. Data width for slave cycles is configured in the Slave Select Control registers.

This signal is also used to configure the data width for block transfers with local DMA. When this signal is asserted during the block transfer initiation cycle, the block transfer is assumed to be a D16 block transfer.

This signal may be changed dynamically for individual transfers, or strapped Low at power-up for permanent D16 operation. If WORD* is strapped Low at power-up, the VIC068A is configured as a D16 slave, independent of the slave configuration in the Slave Select Control registers.

WORD* should not be used to indicate data size (i.e., byte, word, or longword) only VMEbus data port size (i.e., D16 or D32).

BLT*

Input: Yes
Output: Yes, open collector
Drive: 8 mA

This is the block transfer with local DMA indication signal. This signal is used to indicate that a block transfer with local DMA is in progress. This signal remains asserted for the entire block transfer including interleave periods with the exception of local page boundary crossings. BLT* toggles during local boundary crossings to increment the external LA[+:8] counters.

If the BLT* signal is asserted simultaneously with the MWB* signal and BTCR[7] is set, a module-based DMA transfer is performed.

DEDLK*

Input: No
Output: Yes
Drive: 8 mA

This is the deadlock indication signal. This signal indicates that a deadlock condition has occurred. This signal should be used by local logic to remove its request for the VMEbus. DEDLK* remains asserted until the slave transaction is complete.

DEDLK* is also asserted to indicate that a VMEbus master cycle is being attempted during the interleave period of a block transfer with local DMA, without the dual-path feature en-

abled. In this case, DEDLK* is asserted while MWB* is asserted. If, during the interleave period, the MWB* signal is asserted after the VMEbus has been re-obtained, the VIC068A will assert DEDLK* for the duration of the burst.

IPL2*, IPL1*, IPL0*

Inputs: IPL0* only
Output: Yes, open collector
Drive: 8 mA

These are the local priority encoded interrupt request signals. These signals are asserted to interrupt the local processor. All local VIC068A interrupts are issued with these signals. These signals emulate the Motorola 68K interrupt mechanism. The assertion of one or more of these signals indicates a single interrupt with a priority given by the negative-logic value of the IPLi* signals. Level 7 is the highest priority. These signals are open collector to allow the wire-ORing of multiple interrupt sources.

During the assertion of IRESET*, IPL0* becomes an input. If IPL0* is asserted at this time, a global reset is performed.

LIRQ7* – LIRQ1*

Input: Yes
Output: LIRQ2* only
Drive: 8 mA (LIRQ2* only)

These are the local interrupt request signals. These signals serve as local interrupt request signals for the VIC068A. If enabled to handle the particular local interrupt, the VIC068A issues a processor interrupt with the IPLi* signals at the assertion of a LIRQi*. Configuration of local interrupts is allowed through the Local Interrupt Configuration registers.

LIRQ2* may also be configured to issue periodic "heartbeat" interrupts at user-defined intervals.

LIACKO*

Input: No
Output: Yes
Drive: 8 mA

This is the "autovectoring" indication signal. This signal is asserted when the VIC068A is configured to allow the interrupting device to place its status/ID vector on the local data bus

in response to a VIC068A-handled local interrupt acknowledge. This signal may be used to signal an autovectored interrupt acknowledge cycle for 68020/30/40 processors. This signal may be connected directly to the AVEC signal for these processors.

IRESET*

Input: Yes
Output: No
Drive: None

This is the internal reset signal. This signal is used to issue both internal and global resets to the VIC068A. If asserted with IPL0*, a global reset is performed. If asserted without IPL0*, an internal reset is performed. All internal state machines and selected register bits are reset during the assertion of IRESET*. HALT* and RESET* are both asserted during the assertion of IRESET*. If configured as system controller, SYSRESET* is also asserted during the assertion of IRESET*.

SCON*

Input: Yes
Output: No
Drive: None

This is the system controller enabling signal. This signal is used to configure the VIC068A as VMEbus system controller. This signal must be strapped Low at power-up and remain Low for VIC068A to reliably assume the role of VMEbus system controller.

CLK64M

Input: Yes
Output: No
Drive: None

This is the VIC068A master clock input. This 64-MHz clock input is used to clock internal arbitration, timing, and delay functions within the VIC068A. Clock speeds as low as 1 MHz may be used, but all synchronous delays as well as VMEbus and local timing are affected.

2.3 Buffer Control Signals

These signals control the latching and enabling of the external address and data latches and buffers. For block transfers with local DMA, some of these signals are used to control the counting and enabling of external counters required for page boundary crossing.

Figure 2–2 shows typical connections between the external latches/buffers and the buffer control signals.

ABEN*

Input: No
Output: Yes
Drive: 8 mA

This is the VMEbus Address Bus ENable signal. This signal is used to enable the external VMEbus address drivers for VMEbus master operations. It is typically connected to the OEAB input of a '543 address transceiver.

LAEN

Input: No
Output: Yes
Drive: 8 mA

This is the Local Address ENable signal. This signal is used to enable the external local address drivers for slave accesses. It is typically connected to the OEBA input of a '543 address transceiver through an inverter.

Note that this signal is an active-High signal.

LADO

Input: No
Output: Yes
Drive: 8 mA

This is the Latch ADdress Out signal. This signal is used to latch the outgoing VMEbus address for VMEbus master operations. When this signal is asserted (High), it is assumed that the latches are in a latched state. When deasserted, the latches should be in a flow-through state. This allows direct connection to the '543 address driver LEAB input. LADO is very

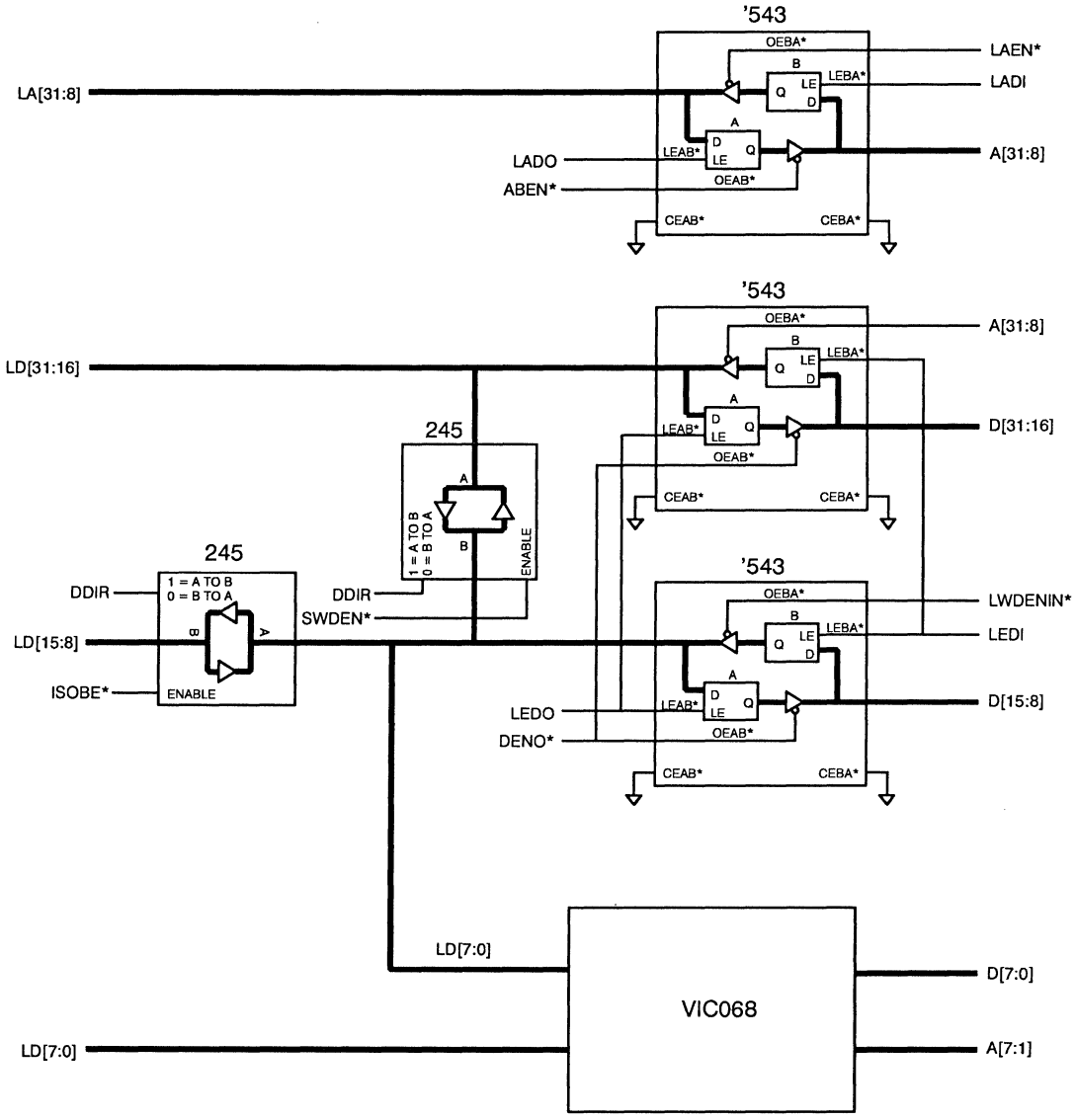


Figure 2–2. VIC068A Control Signals for Shared Memory Implementation

important for proper operation of master write posting and block transfers with interleave periods. For these operations, the VIC068A may use LADO in combination with LADI and ABEN* to temporarily store the contents of a VMEbus address during intervening slave accesses.

LADI

Input: No
Output: Yes
Drive: 8 mA

This is the Latch ADdress In signal. This signal is used to latch the incoming VMEbus address for slave accesses. When this signal is asserted (High), it is assumed that the latches are in a latched state. When deasserted, the latches should be in a flow-through state. This allows direct connection to the '543 address driver LEBA input. LADI is used in conjunction with LADO to temporarily store outgoing VMEbus master transaction addresses during intervening slave accesses.

DENO*

Input: No
Output: Yes
Drive: 8 mA

This is the Data ENable Out signal. This signal enables data onto the VMEbus data bus for master write and slave read cycles. This signal is typically connected to the OEAB input of the '543 data latches.

LWDENIN*

Input: No
Output: Yes
Drive: 8 mA

This is the Lower Word Data ENable IN signal. This signal enables data onto the lower word of the local data bus LD[15:8] for master read and slave write cycles. This signal is typically connected to the OEBA input of the '543 lower data latch.

UWDENIN*

Input: No
Output: Yes
Drive: 8 mA

This is the Upper Word Data ENable IN signal. This signal enables data onto the upper word of the local data bus LD[31:16] for master read and slave write cycles. This signal is typically connected to the OEBA input of the upper '543 data latches.

LEDO

Input: No
Output: Yes
Drive: 8 mA

The Latch Enable Data Out signal. This signal latches the outgoing VMEbus data for master write and slave read cycles. When this signal is asserted (High), it is assumed that the latches are in a latched state. When deasserted, the latches should be in a flow-through state. This allows direct connection to the '543 address driver LEAB input. This signal may be used in conjunction with LEDI to temporarily store outgoing master write post data (data switchback).

LEDI

Input: No
Output: Yes
Drive: 8 mA

This is the Latch Enable Data In signal. This signal latches the incoming VMEbus data for master read and slave write cycles. When this signal is asserted (High), it is assumed that the latches are in a latched state. When deasserted, the latches should be in a flow-through state. This allows direct connection to the '543 address driver LEBA input. This signal may be used in conjunction with LEDO to temporarily store outgoing master write post data.

ISOBE*

Input: No
Output: Yes
Drive: 8 mA

This is the ISOlation Buffer Enable signal. This signal, along with the SWDEN* signal, steers data from LD[31:16] to/from LD[15:0], which is referred to in this document as byte-lane switching. This signal is typically connected to the EN input of the '245 isolation buffer.

SWDEN*

Input: No
Output: Yes
Drive: 8 mA

This is the SWap Data ENable signal. This signal, along with the ISOBE* signal, provides byte-lane switching. It provides for swapping LD[31:16] to LD[15:0]. This signal is typically connected to the EN input of the '245 swap buffer.

DDIR

Input: No
Output: Yes
Drive: 8 mA

This is the Data DIRection signal. This signal provides the data direction (i.e., read/write) information to the isolation and swap buffers. When asserted, buffers should be configured in the local-to-VMEbus (A-to-B) direction. This signal is typically connected to the DIR input of the '245 isolation/swap buffers.

3

Overview of the VIC068A

The VIC068A provides an economical and convenient means to interface between a local CPU bus and the VMEbus. The local bus interface of the VIC068A emulates Motorola's family of 32-bit CISC processor interfaces (68K). Other processors can easily be adapted to interface to the VIC068A with appropriate logic. All of the following items are discussed in further detail in later sections of this manual.

3.1 Resetting the VIC068A

The VIC068A can be reset by any of three distinct reset conditions.

- Internal Reset. This reset is the most common means of resetting the VIC068A. It resets most register values and all mechanisms within the device. This reset is usually issued as a push-button reset.
- System Reset. This reset provides a means of resetting the VIC068A through the VME-bus backplane. The VIC068A may also signal a SYSRESET* by writing a configuration register.
- Global Reset. This is the most complete reset of the VIC068A. This resets all of the VIC068A's configuration registers. This reset should be used with caution since SYSCLK is not driven and the BG/IACK daisy-chains are disabled while a global reset is in progress (while it is system controller). This is usually issued as a power-up reset.

All three reset options are implemented in a different manner and have different effects on the VIC068A configuration registers. Refer to section 12.1 for additional VIC068A reset details.

3.2 The VIC068A VMEbus System Controller

The VIC068A is capable of operating as the VMEbus system controller. It provides VME-bus arbitration functions including:

- priority (PRI), round-robin (RRS), and single-level (SGL) arbitration schemes
- driving IACK* daisy-chain

- driving BGiOUT* daisy-chain (all four levels)
- driving SYSCLK output
- VMEbus arbitration timeout timer
- VMEbus transfer timeout timer

The system controller functions are enabled by the SCON* pin of the VIC068A. When strapped Low, the VIC068A functions as the VMEbus system controller. Refer to Chapter 4 for VIC068A system controller details.

3.3 VIC068A VMEbus Master Cycles

The VIC068A is capable of becoming the VMEbus master in response to a request from local resources. In this situation, the local resource requests that a VMEbus transfer is desired. The VIC068A then makes a request for the VMEbus. When the VMEbus is granted to the VIC068A, it then performs the transfer, acknowledges the local resource, and the cycle is complete. The VIC068A is capable of all four VMEbus request levels. The following release modes are supported:

- Release On Request (ROR)
- Release When Done (RWD)
- Release On Clear (ROC)
- Release under RMC* control
- Bus Capture And Hold (BCAP)

The VIC068A supports A32, A24, and A16 as well as user-defined address spaces.

3.3.1 Master Write-Posting

The VIC068A is capable of performing master write posting (bus-decoupling). In this situation, the VIC068A acknowledges the local resource immediately after the request to the VIC068A is made, thus freeing the local bus. The VIC068A latches the local data to be written and performs the VMEbus transfer without the local resource having to wait for VMEbus arbitration.

3.3.2 Indivisible Cycles

Read-modify-write cycles and Indivisible Multiple-Address Cycles (IMACs) are easily performed using the VIC068A. Significant control is allowed to:

- request the VMEbus on the assertion of RMC* independent of MWB* (this prevents any slave access from interrupting local indivisible cycles)
- stretch the VMEbus AS*
- make the above behaviors dependent on the local SIZi signals

3.3.3 Deadlock

If a master operation is attempted when a slave operation to the same module is in progress, a deadlock has occurred. The VIC068A signals a deadlock condition by asserting the DEDLK* signal. This should be used by the local resource requesting the VMEbus to try the transfer after the slave access has completed.

3.3.4 Self-Access

If the VIC068A is selected as the slave while it is VMEbus master, a self-access has occurred. The VIC068A asserts both BERR* and LBERR* in this situation.

BESR[2,1] also indicates when a self-access has occurred.

3.4 VIC068A VMEbus Slave Cycles

The VIC068A is capable of receiving slave accesses. The VIC068A contains a highly programmable environment to allow for a wide variety of slave configurations. The VIC068A allows for:

- D32 or D16 configuration
- A32, A24, A16, or user-defined address spaces
- programmable block transfer support including:
  - — accelerated block transfer (PAS* held asserted)
  - — non-accelerated-type block transfer (toggle PAS*)
  - — no support for block transfer
- programmable data acquisition delays
- programmable PAS* and DS* timing
- restricted slave accesses (supervisory accesses only)

When a slave access is required, the VIC068A requests the local bus. When local bus mastership is obtained, the VIC068A reads or writes the data to/from the local resource and asserts the DTACK* signal to complete the transfer.

3.4.1 Slave Write-Posting

The VIC068A is capable of performing a slave write post operation (bus-decoupling). When enabled, the VIC068A latches the data to be written and acknowledges the VMEbus (by asserting DTACK*) immediately thereafter. This prevents the VMEbus from having to wait for local bus access.

3.5 Address Modifier (AM) Codes

The VIC068A encodes and decodes the VMEbus address modifier codes. For VMEbus master accesses, the VIC068A encodes the appropriate AM codes through FCi status, ASI-Zi status, and the block transfer status. For slave accesses, the VIC068A decodes the AM Codes and checks the Slave Select Control registers to determine if the slave request is to be supported with regard to address spaces, supervisory accesses, and block transfers. The VIC068A also supports user-defined AM codes. That is, the VIC068A can be configured to assert and respond to user-defined AM codes.

3.6 VIC068A VMEbus Block Transfers

The VIC068A is capable of both performing (as master) and receiving (as slave) block transfers. The master VIC068A performs a block transfer in one of two modes:

- MOVEM-type block transfer
- master block transfer with local DMA

In addition to these VMEbus block transfers, the VIC068A is also capable of performing block transfers from one local resource to another in a DMA-like fashion. This is referred to as a module-based DMA transfer.

The VMEbus specification restricts block transfers from crossing 256-byte boundaries. The VIC068A works around this problem by simply toggling the AS* at VMEbus page boundaries. The VIC068A is also able to break the total transfer length into smaller bursts. The VIC068A allows for easy implementation of large block transfers by releasing the VMEbus and local bus between these bursts and, at the appropriate time, re-requesting the buses at a programmed time later. This in-between time is referred to as the interleave period. All of this is performed without processor/software intervention until the transfer is complete.

The VIC068A contains two separate address counters for the VMEbus and the local address buses. In addition, a separate address counter is provided for slave block transfers.

The VIC068A address counters are 8-bit up-counters that provide for transfers up to 256 bytes. For transfers that exceed the 256-byte limit, the Cypress VAC068A or external counters and latches are required.

The VIC068A allows slave accesses to occur during the interleave period. Master accesses are also allowed during interleave with programming and external logic. This is referred to as the dual-path option.

The Cypress Semiconductor VAC068A may be used in conjunction with the VIC068A to provide much of the external logic required for extended block transfer modes such as the 256-byte boundary crossing and dual path. The VAC068A extends the 8-bit counters in the VIC068A to support full 32-bit incrementing addresses on both the local bus and VMEbus. The VAC068A also contains the latches required for extended address block transfers as well as those required for supporting the dual-path option. The VAC068A enhances boards that support block transfers by greatly reducing the necessary support logic.

The Cypress Semiconductor CY7C964 may also be used to provide the latching and counting of upper data and addresses also reducing necessary support logic.

3.6.1 MOVEM Master Block Transfers

This mode of block transfer provides the simplest implementation of VMEbus block transfers. In this mode, the local resource configures the VIC068A for a MOVEM block transfer and proceeds with the consecutive-address cycles (such as a 68K MOVEM instruction). The local processor continues as the local bus master in this mode.

3.6.2 Master Block Transfers with Local DMA

In this mode, the VIC068A becomes the local bus master and reads or writes the local data in a DMA-like fashion. This provides a much faster interface than the MOVEM block transfer, but with less control and error detection.

3.6.3 Slave Block Transfers

The process of receiving a block transfer is referred to as a slave block transfer. The VIC068A is capable of decoding the address modifier codes to determine if a slave block transfer is desired. In this mode, the VIC068A captures the VMEbus address, and latches it into internal counters. For subsequent cycles, the VIC068A increments this counter for

each transfer. The local protocol for slave block transfers can be configured in a full handshake mode by toggling both PAS* and DS* and expecting DSACKi* to toggle, or in an accelerated mode in which only DS* toggles and PAS* is asserted throughout the cycle.

3.7 Module-Based DMA Transfers

The VIC068A is capable of acting as a DMA controller between two local resources. This mode is similar to that of master block transfers with local DMA except that a local I/O acts as the second source or destination.

3.8 VIC068A Interrupt Generation and Handling Facilities

The VIC068A is capable of generating and handling a seven-level prioritized interrupt scheme similar to that used by the Motorola 68K processors. These interrupts may be the result of the seven VMEbus interrupts, seven local interrupts, five VIC068A error/status interrupts, and eight interprocessor communication interrupts.

The VIC068A can be configured as an interrupt handler for any of the seven VMEbus interrupts. The VIC068A can generate the seven VMEbus interrupts as well as supplying a user-defined status/ID vector. The local priority level (IPL) for VMEbus interrupts is programmable. When configured as the system controller, the VIC068A drives the VMEbus IACK daisy-chain.

The following characteristics of local interrupts may be configured in VIC068A registers:

- user-defined local Interrupt Priority Level (IPL)
- option for VIC068A to provide the status/ID vector
- edge or level sensitivity
- polarity (rising/falling edge, active High/Low)

The VIC068A is also capable of generating local interrupts on certain error or status conditions. These include:

- ACFAIL* asserted
- SYSFAIL* asserted
- failed master write-post (BERR* asserted)
- local DMA completion for block transfers
- arbitration timeout
- VMEbus interrupter interrupt

The VIC068A can also issue interrupts by setting a module or global switch in the interprocessor communication facilities (mailbox interrupts).

3.9 Interprocessor Communication Facilities

The VIC068A includes interprocessor registers and switches that can be written and read through VMEbus accesses. These are the only registers that are directly accessible from the VMEbus. Included in the interprocessor communication facilities are:

- four general-purpose 8-bit registers
- four module switches
- four global switches
- VIC068A version/revision register (read-only)
- VIC068A Reset/Halt condition (read-only)
- VIC068A interprocessor communication register semaphores

When set through a VMEbus access, the switches can interrupt a local resource. The VIC068A includes module switches that are intended for a single module, and global switches that are intended to be used as a broadcast.

4

System Controller Operations

The VIC068A is able to assume the system controller functions (also known as slot 1 functions) by strapping the SCON* signal Low. For reliable operation, the SCON* signal must remain asserted for the duration of operation. As the system controller, the VIC068A performs the following functions:

- priority, round-robin, or single-level arbitration
- driving IACK* daisy-chain
- driving BGiOUT* daisy-chain (all four levels)
- driving SYSCLK output
- VMEbus arbitration timeout timer

The following VIC068A registers are used as the system controller:

- Transfer Timeout Register (TTR), bits 5–7
- Release Control Register (RCR), bits 6–7
- Error Group Interrupt Control Register (EGICR), bit 5

4.1 VMEbus Arbitration

The arbitration scheme is programmed by writing RCR[7:6]. In PRI mode, BR3 has the highest priority and BR0 has the lowest. In the RRS scheme, arbitration priority is assigned on a rotating basis. When the bus is granted to a requester on bus request line BR[n]*, then the highest priority for the next arbitration is assigned to bus request line BR[n–1]* (or BR3 if previous level was BR0). Single-level arbitration is obtained by programming the VIC068A for PRI and setting all requestors to level 3.

When the VIC068A is system controller, it senses the state of the BRi* inputs. One of the four BGiOUT* signals is asserted, corresponding to the highest pending request level during that arbitration cycle. If the VIC068A, as system controller, has a BRi* pending along with another potential master at the same request level, the VIC068A does not assert the BGiOUT* for itself.

An arbitration cycle begins with the deassertion of the BBSY* signal. The VIC068A waits a minimum of 3T after the deassertion of BBSY* before asserting the BGiOUT* signal. The VIC068A deasserts the BGiOUT* signal when the BBSY* is again reasserted.

The VIC068A asserts the BCLR* signal as part of its arbiter function when it senses a request at a higher priority than the level of the current VMEbus master. This may occur when the VIC068A is enabled for both PRI and RRS arbitration schemes. In either case, the VIC068A deasserts BCLR* when BBSY* is deasserted.

In systems containing many contending VMEbus masters, the use of RRS arbitration and fair requests is strongly recommended to prevent excessive bus latency to some of the VMEbus masters. To allocate an unequal share of bus bandwidth to a particular master, assign that master to a BR level shared with fewer masters.

4.2 The VMEbus Arbitration Timeout Timer

After the VIC068A has asserted the BGiOUT* signal, the VIC068A system controller monitors how long the grant is active. Failure to assert BBSY* within 8 μs causes the VIC068A to issue its own BBSY* for the VMEbus-required 90 ns. The EGICR can be used to generate an interrupt for a VMEbus arbitration timeout condition. This timeout feature may not be disabled.

4.3 The VMEbus Transfer Timeout Timer

The VIC068A contains a VMEbus transfer timeout timer. When the VIC068A is configured as the system controller, and the transfer timeout timer is enabled, the VIC068A starts this timer at the assertion of a DSi*. If the timer expires before the assertion of DTACK* or BERR*, BERR* is asserted by the system controller. BERR* remains asserted until the DSi*s are removed. The timer is configured in the TTR[7:5]. BESR[4] is set when this timeout condition occurs.

4.4 The BGi Daisy-Chain Driver

The VIC068A, as system controller, drives the BGiOUT* daisy-chain in response to VMEbus requests. When the VIC068A is the system controller, the BGiIN* lines are inactive, but need to be pulled High externally at the VIC068A (4.7 – 10KΩ).

4.5 The IACK* Daisy-Chain Driver

The VIC068A as system controller drives the IACK* daisy-chain. If the VIC068A has no VMEbus interrupt pending, the VIC068A asserts the IACKOUT* signal when IACKIN* and IACK* are asserted and approximately 40 ns (2T + t_{PD}) has elapsed since the assertion of DS*.

5

VIC068A VMEbus Master Operations

The transfer of data is initiated by a VMEbus master module. The master module controls the type of transfer (read, write, interrupt acknowledge, etc.) and provides the address and address modifiers for the transfer. The timing of the start of the transfer is also controlled by the master.

The following VIC068A registers are used for master operations (block transfer registers not included):

- Transfer Timeout Register (TTR), bits 1, 2–4
- Interface Configuration Register (ICR), bits 1–7
- Arbiter/Requester Configuration Register (ARCR), bits 0–3, 5, 6
- Address Modifier Source Register (AMSR)
- Bus Error Status Register (BESR), bits 0–3
- Slave Select 1 Control Register 0 (SS1CR0), bit 6
- Release Control Register (RCR), bits 6–7

See Chapter 13 for descriptions of these registers.

5.1 VMEbus Requests

There are many types of cycles in which the VIC068A requests the VMEbus. These include:

- SINGLE-cycle data transfer requests (SINGLE)
- status/ID fetches for Interrupt ACKnowledge cycles (IACK)
- Indivisible Single-Address Cycles (ISAC) such as read-modify-write cycles
- Indivisible Multiple-Address Cycles (IMAC)
- Block Transfer Requests (BLT)
- VMEBus Capture And Hold (BCAP) requests

The actual assertion of the BRi* signals are made in response to the following signals:

- assertion of MWB* qualified by PAS* for single-cycle and block-transfer accesses

- assertion of FCIACK* qualified by PAS* for VMEbus interrupt acknowledge cycles
- assertion of RMC* qualified by PAS* (when the ICR is appropriately programmed) for ISAC and IMAC cycles
- setting the BCAP bits in the ICR for BCAP and IMAC cycles

The request level is set in ARCR[6:5]. The default level is BR3.

5.2 Release Modes

The VIC068A supports the four VMEbus release modes:

- Release On Request (ROR)
- Release When Done (RWD)
- Release On Clear (ROC)
- VMEBus Capture And Hold (BCAP)

In addition to these, the VIC068A also allows an extension of the above items to provide for the use of the RMC* signal. This is referred to as Release Under RMC* Control. These modes are selected by writing RCR[7:6]. The Release Under RMC* Control mode is programmed by setting ICR[5].

5.2.1 Release On Request (ROR)

In this release mode, the VIC068A deasserts BBSY* when a BRi* is asserted by another VMEbus module and the VIC068A has no need for the VMEbus. The VIC068A does not assert the ABEN* signal if there is no data transfer in progress and the VIC068A is currently the VMEbus master.

5.2.2 Release When Done (RWD)

In this mode, the VIC068A deasserts the BBSY* signal as soon as the following conditions occur:

1. BBSY* has been asserted by the VIC068A for a minimum of 90 ns
2. AS* has been deasserted by the VIC068A
3. The VIC068A has no further need for the VMEbus (the VIC068A has not asserted BRi* for the last 2T)

4. BGiIN* is not asserted to the VIC068A

5.2.3 Release On Clear (ROC)

In this mode, the VIC068A continues to assert BBSY* until the BCLR* signal is asserted by the system controller.

5.2.4 VMEbus Capture and Hold (BCAP)

In this mode, the VIC068A asserts BBSY* continuously for as long as the BCAP mode is selected. The release of BBSY* occurs by programming the release control bits to another release mode. If RWD is selected, BBSY* is released immediately. If ROR is selected, BBSY* is released at a pending VMEbus request. The VIC068A deasserts BBSY* on the assertion of BCLR* if ROC is selected. Do not enter the BCAP mode if the VIC068A is currently the VMEbus master.

5.2.5 Release Under RMC* Control

In this mode, the VIC068A both requests and holds the VMEbus under control of the RMC* signal. When appropriately programmed by setting ICR[5], the assertion of RMC* and PAS* causes the VIC068A to request the VMEbus, accept the BGiIN*, and assert BBSY*. The deassertion of RMC* allows the deassertion of BBSY* based upon the release mode programmed.

5.3 VIC068A VMEbus Master Write Cycle

If the VIC068A is not the current VMEbus master, the VIC068A bids for access to the VMEbus when it receives the MWB* and PAS* signals asserted. When all of the following conditions occur:

1. AS* is deasserted from the previous cycle
2. DTACK* and BERR* are deasserted
3. the BGiIN* has been received
4. all appropriate metastability settling delays have elapsed

the VIC068A drives the D[7:0] data buffers onto the VMEbus and asserts DENO*, which should be used to enable the remaining data buffers. At the same time, the VIC068A enables the A[7:0] address lines onto the VMEbus in addition to asserting the ABEN* signal to drive the remaining VMEbus address lines. The VIC068A also drives AM[5:0], WRITE*, and LWORD* as required. At this time, the VIC068A initiates an internal delay to insure appropriate address set-up time before the assertion of the AS*. After AS* is asserted, the VIC068A latches the LA[7:0] and asserts the LADO signal, which should be used to latch the remaining local address lines.

After the AS* signal has been asserted, the VIC068A initiates an internal delay to assert the data strobes (DSi*). When this delay has elapsed, the VIC068A asserts the appropriate data strobes as determined by the size and alignment of the transfer. The DSi* signals remain asserted until either DTACK* or BERR* have been asserted to the VIC068A. If DTACK is asserted, the VIC068A asserts the DSACKi* signals according to the port size. That is, if the WORD* signal was deasserted, the VIC068A acknowledges this D32 operation by asserting both the DSACK0* and DSACK1* signals. If the WORD* signal was asserted, the VIC068A acknowledges this D16 transfer by asserting only the DSACK1* signal. For example, when performing a longword transfer to a D16 device, asserting only DSACK1* would notify the processor that the additional word of data needs to be transferred. This is consistent with the Motorola 68K dynamic bus sizing capabilities using DSACKi*.

When turbo mode is enabled by setting ICR[1], the VMEbus address and data set-up times are decreased by 1T.

Tables 5–1 through *5–4* show the buffer control signals for various master cycles.

Table 5–1. Buffer Control Signals: D32 VMEbus Master Write Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Longword	1	00	00	LL	LL	L	L	L	L	↑	L	L	L	H	H	L	H	H
	1	00	01	LL	HL	L	L	L	L	↑	L	L	L	H	H	L	H	H
	1	00	10	LL	LL	H	H	L	L	↑	L	L	L	H	H	L	H	H
	1	00	11	LL	HL	H	H	L	L	↑	L	L	L	H	H	L	H	H

Table 5–1. Buffer Control Signals: D32 VMEbus Master Write Operation (continued)

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Three-Byte	1	11	00	LL	LH	L	L	L	L	↑	L	L	L	H	H	L	H	H
	1	11	01	LL	HL	L	L	L	L	↑	L	L	L	H	H	L	H	H
	1	11	10	LL	LL	H	H	L	L	↑	L	L	L	H	H	L	H	H
	1	11	11	LL	HL	H	H	L	L	↑	L	L	L	H	H	L	H	H
Word	1	10	00	LL	LL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	1	10	01	LL	LL	H	L	L	L	↑	L	L	L	H	H	L	H	H
	1	10	10	LL	LL	H	H	L	L	↑	L	L	L	H	H	L	H	H
	1	10	11	LL	HL	H	H	L	L	↑	L	L	L	H	H	L	H	H
Byte	1	01	00	LL	LH	L	H	L	L	↑	L	L	L	H	L	H	H	H
	1	01	01	LL	HL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	1	01	10	LL	LH	H	H	L	L	↑	L	L	L	H	H	L	H	H
	1	01	11	LL	HL	H	H	L	L	↑	L	L	L	H	H	L	H	H

Table 5–2. Buffer Control Signals: D32 VMEbus Master Read Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Longword	1	00	00	LL	LL	L	L	L	L	↑	H	↑	L	L	H	L	L	L
	1	00	01	LL	HL	L	L	L	L	↑	H	↑	L	L	H	L	L	L
	1	00	10	LL	LL	H	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	00	11	LL	HL	H	H	L	L	↑	H	↑	L	L	L	L	L	H
Three-Byte	1	11	00	LL	LH	L	L	L	L	↑	H	↑	L	L	H	L	L	L
	1	11	01	LL	HL	L	L	L	L	↑	H	↑	L	L	H	L	L	L
	1	11	10	LL	LL	H	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	11	11	LL	HL	H	H	L	L	↑	H	↑	L	L	L	L	L	H

Table 5–2. Buffer Control Signals: D32 VMEbus Master Read Operation (continued)

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Word	1	10	00	LL	LL	L	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	10	01	LL	LL	H	L	L	L	↑	H	↑	L	L	H	L	L	L
	1	10	10	LL	LL	H	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	10	11	LL	HL	H	H	L	L	↑	H	↑	L	L	L	L	L	H
Byte	1	01	00	LL	LH	L	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	01	01	LL	HL	L	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	01	10	LL	LH	H	H	L	L	↑	H	↑	L	L	L	L	L	H
	1	01	11	LL	HL	H	H	L	L	↑	H	↑	L	L	L	L	L	H

Table 5–3. Buffer Control Signals: D16 VMEbus Master Write Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Longword	0	00	00	LH	LL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	00	01	LH	HL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	00	10	LH	LL	H	H	L	L	↑	L	L	L	H	L	H	H	H
	0	00	11	LH	HL	H	H	L	L	↑	L	L	L	H	L	H	H	H
Three-Byte	0	11	00	LH	LH	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	11	01	LH	HL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	11	10	LH	LL	H	H	L	L	↑	L	L	L	H	L	H	H	H
	0	11	11	LH	HL	H	H	L	L	↑	L	L	L	H	L	H	H	H
Word	0	10	00	LH	LL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	10	01	LH	LL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	10	10	LH	LL	H	H	L	L	↑	L	L	L	H	L	H	H	H
	0	10	11	LH	HL	H	H	L	L	↑	L	L	L	H	L	H	H	H
Byte	0	01	00	LH	LH	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	01	01	LH	HL	L	H	L	L	↑	L	L	L	H	L	H	H	H
	0	01	10	LH	LH	H	H	L	L	↑	L	L	L	H	L	H	H	H
	0	01	11	LH	HL	H	H	L	L	↑	L	L	L	H	L	H	H	H

Table 5–4. Buffer Control Signals: D16 VMEbus Master Read Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
Longword	0	00	00	LH	LH	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	00	01	LH	HL	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	00	10	LH	LL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	00	11	LH	HL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
Three-Byte	0	11	00	LH	LH	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	11	01	LH	HL	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	11	10	LH	LL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	11	11	LH	HL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
Word	0	10	00	LH	LL	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	10	01	LH	HL	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	10	10	LH	LL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	10	11	LH	HL	H	H	L	L	↑	H	↑	L	L	L	H	L	H
Byte	0	01	00	LH	LH	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	01	01	LH	HL	L	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	01	10	LH	LH	H	H	L	L	↑	H	↑	L	L	L	H	L	H
	0	01	11	LH	HL	H	H	L	L	↑	H	↑	L	L	L	H	L	H

5.4 VIC068A VMEbus Master Read Cycle

This cycle is identical to that of the master write cycle, as described in section 5.3, with the following exceptions:

- the VMEbus data buffers are not driven.
- DENO* is not asserted.
- the UWDENIN and LWDENIN are asserted.
- DDIR is not asserted. The address and AS* considerations are the same as well as the DSACKi* conventions.

5.5 Master Write Posting

The VIC068A is enabled for master write-posting by setting SS1CR0[6]. When enabled, the VIC068A captures the local address, data and control signals, requests the VMEbus, and

immediately acknowledges the local processor. This frees the local processor from waiting for VMEbus arbitration. When VMEbus mastership is obtained, the VIC068A performs the transfer according to normal VMEbus protocol. Further write-posts are disabled until a DTACK* or BERR* is asserted by the slave. If a BERR* is signaled, the VIC068A can be configured to issue a local interrupt by clearing EGICR[6].

If a slave read occurs after a write has been posted but not yet transferred, the latched write data is "toggled" to the B-to-A latch of the '543s by asserting the LEDI signal. The slave read then occurs normally without the write data being written over by the slave read data. After the VIC068A asserts DTACK*, the UWDENIN and LWDENIN signals are asserted to drive the write data to the A-to-B latch of the '543. Then LEDO is asserted to again latch the data for the write operation. This toggling of data is referred to as a Master Write-Post Data Switchback.

5.6 Indivisible Cycles

Indivisible cycles can be divided into two categories:

- Indivisible Single-Address Cycles (ISACs)
- Indivisible Multiple-Address Cycles (IMACs)

The VIC068A supports both ISACs and IMACs through many different protocols. Indivisible cycles can be configured as follows:

1. request the VMEbus on the assertion of RMC* independent of MWB* (this prevents any slave access from interrupting local indivisible cycles)

2. stretch the VMEbus AS*

3. make the above behaviors dependent on the local SIZi signals

These modes are summarized in *Table 5–5*.

Table 5–5. RMC Control Map

ICR[7:5]	First Operation	VMEbus Requested on RMC* Assertion	AS* Stretched	VMEbus Held During RMC* Assertion
X 0 0	Any	No	No	No
0 0 1	Any	Yes	No	Yes
0 1 0	Any	No	Yes	Yes
0 1 1	Any	Yes	Yes	Yes
1 0 1	Byte	No	No	No
1 0 1	Non-byte	Yes	No	Yes
1 1 0	Byte	No	Yes	Yes
1 1 0	Non-byte	No	No	No
1 1 1	Byte	No	Yes	Yes
1 1 1	Non-byte	Yes	No	Yes

Address strobe stretching is performed for ISACs in accordance with the VMEbus RMC specification. For IMACs, the address strobe is typically not stretched in order for slave modules to latch each address.

In the 68K family of processors, ISACs and IMACs are distinguished by the fact that the first read of an IMAC is never of byte size. This allows for AS* stretching for all RMCs, no RMCs, or only RMCs in which the first transfer was of byte size.

If a processor is not capable of generating indivisible cycles or does not distinguish ISACs from IMACs, cycle indivisibility may be guaranteed by using the BCAP release mode outlined below:

1. set the VIC068A to a BCAP release mode by setting RCR[7:6]
2. wait for VMEbus grant (BGiIN*)
3. perform indivisible cycles
4. release the VIC068A from BCAP mode in the RCR

When the VIC068A is the VMEbus slave to a ISAC, the VIC068A maintains the local bus by keeping LBR* asserted as long as AS* is asserted.

5.6.1 Indivisible Single-Address Cycles (ISACs)

The most common implementation of ISACs are of the read-modify-write (RMC) category. This is the only ISAC supported by the VMEbus. The Motorola TAS (Test And Set) instruc-

tion is an example of a read-modify-write cycle. The VMEbus specification requires that, for RMC cycles, the VMEbus address strobe be held asserted between the read and the write cycles. Motorola processors prior to the 68020 performed ISACs in the same manner by asserting their address strobes for the duration of the cycle. The Motorola processors such as the 68020/30/40 provide a signal (RMC* for the 68020/30, and LOCK* for the 68040) to indicate that a RMC is being performed. The VIC068A has an RMC* signal that is typically connected to these signals to control ISACs. For RMC cycles, the AS* should be programmed to be stretched.

5.6.2 Indivisible Multiple-Address Cycles (IMACs)

The Motorola CAS and CAS2 instructions are examples of IMACs. The VIC068A allows for the support of IMACs without using the BCAP protocol given in section 5.6. In this case, the RMC* signal should be set to request and hold the VMEbus. The local cycle is not allowed to complete until the VMEbus has been obtained. In addition, AS* stretching should be disabled so that address latching may be performed by the slave.

5.7 Deadlock

If the VMEbus is requested in response to MWB* or FCIACK* being asserted, and at the same time a valid slave select has been signaled, a deadlock has occurred. The VIC068A may be programmed in the ICR to signal a deadlock in the following ways:

- assert DEDLK*
- assert DEDLK*, LBERR*, and HALT*
- assert DEDLK* and LBERR* (without HALT*) for RMC deadlocks

The first option is typically used for non-Motorola processors without a retry capability. In that case, DEDLK* should be used to signal the processor to vacate the local bus (deassert MWB*).

For Motorola 68K applications, the second option may be used to signal the processor to retry the current bus cycle using the Motorola BERR/HALT retry mechanism.

For Motorola 68K processors, the HALT/BERR retry mechanism is disabled for RMC cycles. In this condition, the third option should be used to signal a generic BERR to the processor. The BERR exception processing routines should include software code that

checks the Special Status Word (SSW) of the 68K BERR exception stack frames to indicate RMC status.

In all of the above cases, DEDLK* is not deasserted until the slave access causing the deadlock is compete.

When a cycle is DEDLKed, the VMEbus is still requested. If the VMEbus is granted (after the slave access is complete) and is still available when MWB* is reasserted, the cycle proceeds as normal. If the VMEbus is granted and MWB* is not reasserted, the VIC068A asserts BBSY* for the 90 ns required by the VMEbus specification if it is configured as RWD. If configured for ROR, the VIC068A maintains the BBSY* until requested to release it.

5.7.1 Undetectable Deadlocks

Poor system design can lead to deadlocks that the VIC068A cannot detect and from which it cannot recover. Consider the following example:

1. Two boards, CPUa and CPUb, contain a local processor that has dual-ported memory connected to both the VMEbus and a VSBbus.
2. CPUa is local bus master and VSBbus master and desires data from CPUb's memory over the VSBbus.
3. CPUb is local bus master and VMEbus master and desires data from CPUa's memory over the VMEbus.
4. Because both CPUs are their own local bus masters, neither will be able gain access to the others local bus. DEADLOCK!

The VIC068A is not able to detect this deadlock because the VIC068A is only able to monitor the status of the local bus and the VMEbus. Deadlocks due to the existence of other buses, such as VSB, will not be detected.

The only way to recover from these types of deadlocks is to use bus timeout timers.

5.8 Self-Access

If a slave select is signaled while it is the VMEbus master, a self-access has occurred. The VIC068A signals a self-access by asserting both the LBERR* and BERR* signals. The BESR also indicates self-access status.

Self-accesses may be used to determine the slave address map of a module.

5.9 VMEbus/Local Bus Data and Port Size

A distinction should be made regarding the terms *transfer size* and *port size*. Transfer size indicates the size of the data in terms of bytes, words, and longwords. The port size indicates the physical size of the bus the data will be transferred on. Port sizes for the VMEbus are usually given in terms of D8, D16, and D32 for 8-bit, 16-bit, and 32-bit-wide buses respectively.

The transfer size of the master operation is indicated to the VIC068A by the SIZ1/0 signals according to the following table:

SIZ1	*SIZ0*	*Data Size*
0	0	Longword (32 bits)
0	1	Byte (8 bits)
1	0	Word (16 bits)
1	1	3-byte

This information insures proper VMEbus protocol in terms of LWORD*, A01, and DS1/0*. In addition, the VIC068A buffer control signals will be properly asserted for the size of the transfer.

The port size of the transfer is indicated by the WORD* signal. When asserted, the master transfer is treated as a D16 transfer. For D16 operations, the LWORD* signal is not asserted, and the DS1/0* signals behave appropriately. In addition, the SWDEN* and ISOBE* signal may be asserted differently in that the D16 VMEbus data located on D[15:0] may be swapped onto the LD[31:16]. This depends on the size of the transfer, the alignment of the transfer, and whether performing a read or a write. Refer to *Tables 5–3* to *5–6* for more details on these signals for particular cases.

The WORD* signal may be changed dynamically to enable the VIC068A to deal with both D16 and D32 slaves.

When performing D16 operations, the VIC068A asserts only the DSACK1* signal to indicate to the processor the port size is 16 bits. This would indicate to a 68K processor that when transferring a longword of data, two transfers are required. This is consistent with the Motorola 68K DSACKi* dynamic bus-sizing convention.

When using a 16-bit local processor, the WORD* signal must be asserted for all master transfers, or strapped Low at power-up to perform D16 transfers. The VIC068A does not support using a 16-bit local bus to a 32-bit VMEbus (D32).

5.10 Fair Request Timeout

A fair request timeout scheme may be used to prevent VMEbus starvation of any master in a bus request daisy-chain. When operating in a fair request mode, the VIC068A does not assert its BRi* signal until or unless that request level is in its deasserted state. If all boards in a system obey this fairness doctrine, VMEbus starvation will not occur.

To minimize starvation caused by unfair masters, the VIC068A is also equipped with a fair request timeout timer. If the VIC068A is unable to obtain VMEbus mastership within a programmed delay, the VIC068A stops using the fairness doctrine and asserts its BRi* without delay.

Fairness is controlled by writing the ARCR. Fairness is disabled by clearing bits ARCR[3:0]. Fairness is enabled (with no timeout) by setting bits ARCR[3:0]. The timeout timer is enabled by writing any other combination to these bits. The value of the timeout is 2 μs times the number written. A difference of 2 μs may exist between the value written and the actual delay observed.

5.11 Address-Only Cycles

The VIC068A will not perform address-only cycles. The VIC068A, as slave, can accept address-only cycles.

5.12 The Address Modifiers for Master Cycles

When the VIC068A performs master cycles, it examines the ASIZ1/0 and FC2/1 signals to determine the value of the AM[5:0] signals that will be driven. The information that the AM codes specify indicates address sizing and supervisory/user and program/data information. Under normal circumstances, the VIC068A outputs standard VMEbus AM codes. The

VIC068A may also be configured to output user-defined AM codes. This is done with the ASIZ1/0 signals and the AMSR. If the ASIZ1/0 signals are both Low, the VIC068A uses the AMSR to determine the value of the AM codes.

If AMSR[7] is clear, VIC068A issues the contents of AMSR[5:0] to the AM[5:0] signals. If AMSR[7] is set, the VIC068A issues AM codes based on AMSR[5:3] and the FC2/1 inputs.

Table 5–6 summarizes the AM codes for various VIC068A operations and configurations.

Table 5–6. Master Transfer AM Code Control Map

VIC068A Master Access Inputs				VIC068A AM Code Output	
ASIZ1/0	Address Size	Block Transfer	FC2/1	Operation Type	AM[5:0]
0 1	A32 Addressing	No	0 0 0 1 1 0 1 1	User Data User Program Supervisory Data Supervisory Program	$09 $0A $0D $0E
		Yes	0 X 1 X	User Block Supervisory Block	$0B $0F
1 1	A24 Addressing	No	0 0 0 1 1 0 1 1	User Data User Program Supervisory Data Supervisory Program	$39 $3A $3D $3E
		Yes	0 X 1 X	User Block Supervisory Block	$3B $3F
1 0	A16 Addressing	No	0 X 1 X	User Access Supervisory Access	$29 $2D
0 0	User Defined AMSR[7] = 0	Yes/No		User Defined	AMSR[5:0]
	User Defined AMSR[7] = 1	Yes/No	 0 0 0 1 1 0 1 1	User Defined	AMSR[5:3] + $01 $02 $05 $06

6

VIC068A VMEbus Slave Operations

The act of writing or retrieving data for a VMEbus master is referred to as a slave operation. The VIC068A is able to perform slave operations with extensive configuration options.

The following VIC068A registers are used in performing and configuring slave operations:

- Slave Select 0 Control Register 0 (SS0CR0), bits 0–5
- Slave Select 0 Control Register 1 (SS0CR1)
- Slave Select 1 Control Register 0 (SS0CR0), bits 0–5
- Slave Select 1 Control Register 1 (SS0CR1)
- Local Bus Timing Register (LBTR)
- Address Modifier Source Register (AMSR)

6.1 The Valid Slave Select

The VIC068A contains two separate signals that are used to indicate that the VIC068A has been selected for a slave access. These signals are SLSEL0* and SLSEL1*. These signals are usually the result of external VMEbus address decoding. When the VIC068A detects a SLSELi* asserted, the VIC068A waits for:

- AS* asserted
- DSi* asserted for the current cycle (not left over from previous cycle)
- DTACK* or BERR* deasserted from previous cycle

The VIC068A then checks the AM codes for:

- address sizing (A32/A24/A16)
- transfer type (supervisory/user)

If the VIC068A is configured, in SSiCR0, to accept the slave access as indicated by the AM codes, the VIC068A proceeds with the slave request by asserting the LBR* signal to obtain the local bus. If the VIC068A is not configured for the particular type of slave access, the VIC068A ignores the request and does not assert LBR*.

If the VIC068A has been selected for valid D32 slave access and the VIC068A is configured to accept only D16 operations, the VIC068A asserts BERR*. If the WORD* signal is asserted, the VIC068A will only accept D16 slave cycles, independent of how the VIC068A may be configured in the SSiCR0. If the VIC068A is not configured to accept block transfers, any block transfer request will be BERRed.

The VIC068A is also capable of bypassing the standard VMEbus AM codes and qualifying the SLSELi* with user-defined AM codes. If enabled for this operation (in SSiCR0[3:2]), the VIC068A compares the AM codes with the value contained in the AMSR[5:0]. If the values match, a valid slave select has occurred. The VIC068A may also be configured to only compare AM[5:3] to AMSR[5:3]. When enabled for this operation by setting AMSR[6], the address size is also qualified by the address size information in the SSiCR0. *Table 6–1* summarizes the AM code operations for VIC068A slave accesses.

In some situations, it is possible for both SLSEL1* and SLSEL0* to be asserted simultaneously. In this case, the VIC068A checks each SLSELi*'s register configuration with the AM codes to determine which, if any, valid slave select has occurred.

Also included in the AM codes is information specifying if the transfer is a block transfer. The VIC068A checks the SSiCR0 register to determine if the VIC068A is enabled to receive slave block transfers. Slave block transfers are discussed in detail in section 10.2.

Table 6–1. Slave Transfer AM Code Control Map

VIC068A AM Code Inputs		VIC068A Slave Access Outputs		
Operation Type	AM[5:0]	Address Size	Block Transfer	FC2/1
User Data User Program Supervisory Data Supervisory Program	\$09 \$0A \$0D \$0E	A32 Addressing	No	1 0
User Block Supervisory Block	\$0B \$0F		Yes	0 0
User Data User Program Supervisory Data Supervisory Program	\$39 \$3A \$3D \$3E	A24 Addressing	No	1 0
User Block Supervisory Block	\$3B \$3F		Yes	0 0
User Access Supervisory Access	\$29 \$2D	A16 Addressing	No	1 0

Table 6–1. Slave Transfer AM Code Control Map (continued)

VIC068A AM Code Inputs		VIC068A Slave Access Outputs		
Operation Type	AM[5:0]	Address Size	Block Transfer	FC2/1
User Defined	User Defined	User Defined	No	1 0
User Defined	User Defined	User Defined	Yes	0 0

6.2 The Local Bus Request

After a valid slave select has been signaled, the VIC068A bids for the local bus by asserting the LBR* signal. This signal should be input to a local bus arbiter, which in turn issues a bus grant (assert LBG*) to the VIC068A. The VIC068A does not perform local bus arbitration.

6.3 The Local Bus Grant

Once the VIC068A issues the LBR*, the VIC068A waits for the assertion of the local bus grant. If the local processor expects BGACK-type signaling in response to the assertion of BG*, this must be generated by external logic. This BGACK must continue to be asserted until LBR* is deasserted.

After the assertion of LBG*, the VIC068A waits for $3T+t_{PD}$ before enabling the local address drivers (and data drivers if writing). At this point, the VIC068A waits $1T+t_{PD}$ before driving PAS*. If local resources cannot be guaranteed to be off the local bus by these times after the assertion of LBG*, additional delay may need to be introduced between the assertion of a bus grant from the local arbiter and the assertion of LBG* to the VIC068A. See *Figure 14–11* in Chapter 14 for more details on local bus timing.

When the VIC068A begins driving the local bus, it also drives the FC2/1 signals with information on the type of local cycle it is performing. These function codes should not be confused with the function codes that are driven to the VIC068A when it is performing VMEbus master cycles. The function code outputs for the VIC068A are as follows:

FC2	*FC1*	*Description*
0	0	Slave Block Transfer
0	1	Standard Slave Access
1	0	Block Transfer with Local DMA
1	1	DRAM Refresh

6.4 Local Bus Timing

The VIC068A contains a Local Bus Timing Register (LBTR) for configuring the minimum PAS* assertion and deassertion and DS* deassertion times. For normal slave accesses, the minimum deassertion time is not considered in that PAS* and DS* are asserted according to the timing discussed in section 6.3. The minimum deassertion times for PAS* and DS* are usually used when performing multiple back-to-back local cycles (i.e., within a single assertion of LBR*), such as DRAM refresh cycles, slave block transfers, and master block transfers with local DMA. The minimum assertion time for PAS* should be set so that illegal memory access cycles can never occur for the particular memory devices being used. Minimum PAS* asserted time is usually dictated by the assertion of the DSACKi* signals, which start an additional delay circuit called the SAT delay (see section 6.8). *Figures 6–1* and *6–2* show an example of local reads and writes.

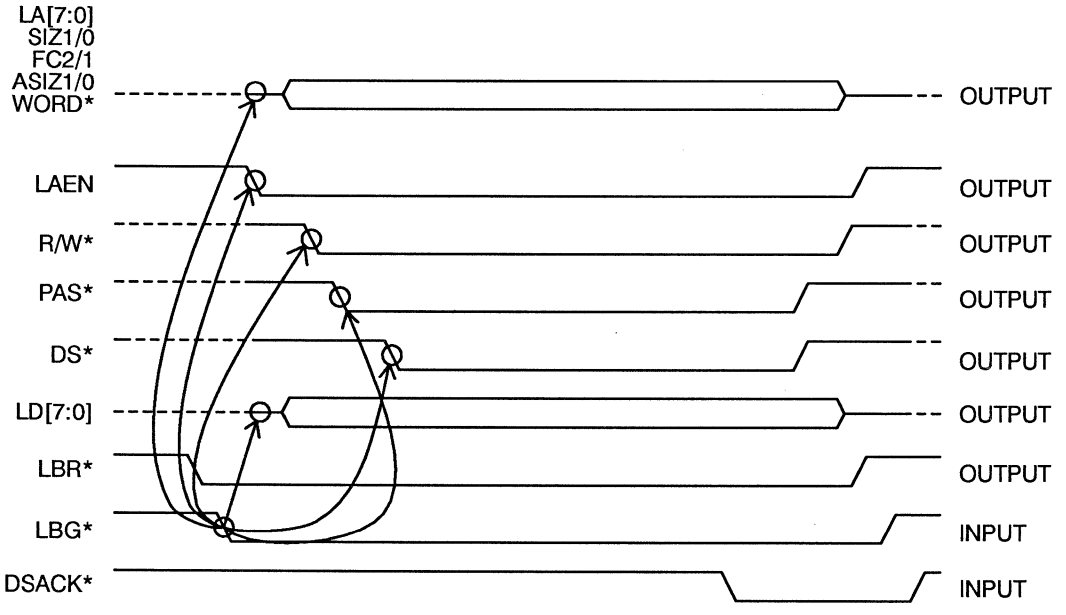


Figure 6–1. Local Bus Cycle—Write

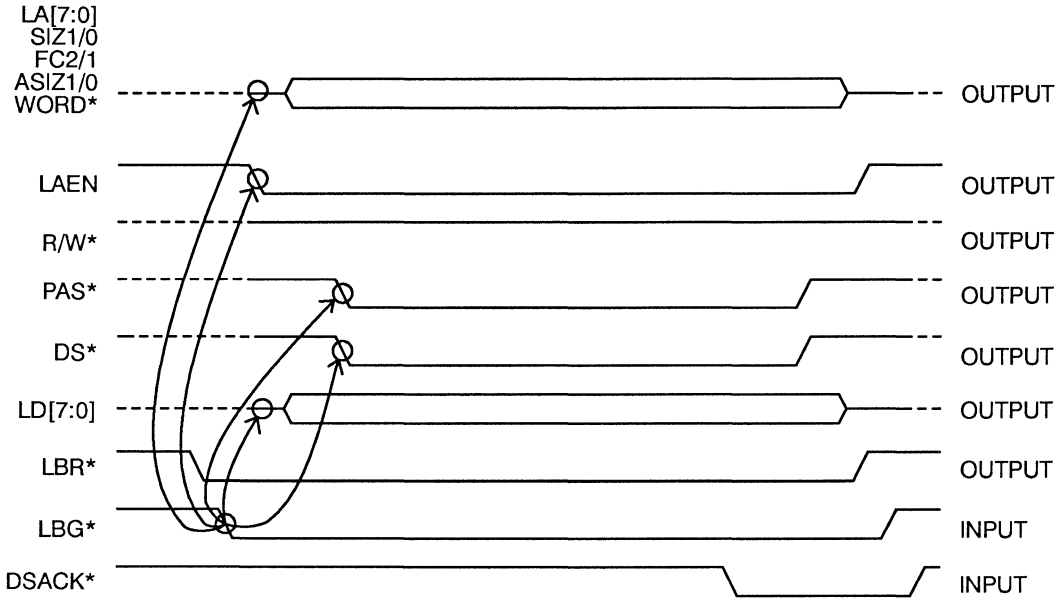


Figure 6–2. Local Bus Cycle—Read

6.5 VMEbus/Local Bus Data and Port Size

After receiving a valid slave select, the VIC068A examines the DS1/0*, A01, and LWORD* signals to determine the size and the alignment of the transfer. The VIC068A then drives the SIZ1/0 and LA[1:0] signals with the appropriate values. The SIZ1/0 codes are given below:

SIZ1	*SIZ0*	*Transfer Size*
0	0	Longword
0	1	Byte
1	0	Word
1	1	3-byte

The ability to handle D32 transfers is configured in the SSiCR0[4]. If configured as a D32 port, the VIC068A accepts any size VMEbus transfer. If programmed as a D16 port only,

the VIC068A will assert BERR* for all D32 requests. The VIC068A determines D32 and D16 from the LWORD* signal.

6.6 The Latched Bus Interface

When a slave read is performed, data is read from local memory and latched into the VMEbus data transceivers both internally and externally to the VIC068A. After DSACKi* is asserted, the VIC068A begins the SAT delay. After this delay times out, the VIC068A asserts both DTACK* and LEDO. These signals are held until both DS1/0* are deasserted.

During a normal (non-posted) slave write, the VIC068A asserts the LEDI signal immediately after receiving a valid slave select latching the data into the data transceivers.

6.7 Slave Write Posting

The VIC068A is able to perform slave write posting. In this mode, the VIC068A, after receiving a valid slave select, latches the incoming data (asserts LEDI) and immediately asserts DTACK*. The VIC068A requests the local bus and then performs the local write cycle independent of VMEbus activity.

If a slave write post is requested while a previous slave write post is currently being serviced, DTACK* is not asserted until the local write is complete. At this point, the VIC068A posts the write normally. Slave write posts should be used with caution. If there was a problem with the local portion of the cycle (i.e., LBERR* was asserted), the initiator of the cycle may never know the the cycle did not complete since DTACK* was already asserted.

6.8 Slave Acknowledge Timing (SAT)

Once DSACKi* is asserted, the delay defined by SSiCR1[3:0] begins. After this delay expires, DTACK* is asserted, and PAS* and DS* deassert. LEDO is also asserted if the slave cycle is a read. An assertion of any or both of the DSACKi* signals is considered an acknowledge and begins this timeout. The default value for these delays is 0. Usually delays need only be used for memory designs that use an advance acknowledge or late bus-error algo-

rithms. Once DTACK* is asserted, the VIC068A maintains DTACK* until the VMEbus DSi* signals are deasserted.

If the LBERR* signal is asserted, the VIC068A asserts BERR* until the DSi* signals are deasserted. If LBERR* is asserted after either DSACKi* has been asserted, and the DSACK- to-DTACK delay has not yet expired, DTACK* is inhibited and the BERR* signal will be asserted on the VMEbus without delay. If the user employs some delayed LBERR* algorithm (such as late parity check), the SAT delay must be programmed sufficiently long to allow for this delay of LBERR*.

Table 6–2. Buffer Control Signals: D32 VMEbus Slave Write Operation

Data Path Size	VMEbus Stimulus			Local Bus Response			Address Control			Data Control			Swap Control				
	DS1/0*	A01	LWORD*	SIZ1/0	LA[1:0]	DSACKi*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
D32	0 0	0	0	LL	LL	0	H	↑	L	H	↑	L	L	H	L	L	L
D32, UAT (0–2)	0 1	0	0	HH	LL	0	H	↑	L	H	↑	L	L	H	L	L	L
D32, UAT (1–3)	1 0	0	0	HH	LH	0	H	↑	L	H	↑	L	L	H	L	L	L
D32, UAT (1–2)	0 0	1	0	HL	LH	0	H	↑	L	H	↑	L	L	H	L	L	L
D16 (0–1)	0 0	0	1	HL	LL	0	H	↑	L	H	↑	L	L	L	L	L	H
D16 (2–3)	0 0	1	1	HL	HL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (0)	0 1	0	1	LH	LL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (1)	1 0	0	1	LH	LH	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (2)	0 1	1	1	LH	HL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (3)	1 0	1	1	LH	HH	0	H	↑	L	H	↑	L	L	L	L	L	H

Table 6–3. Buffer Control Signals: D32 VMEbus Slave Read Operation

Data Path Size	VMEbus Stimulus			Local Bus Response			Address Control			Data Control			Swap Control				
	DS1/0*	A01	LWORD*	SIZ1/0	LA[1:0]	DSACKi*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
D32	0 0	0	0	LL	LL	0	H	↑	L	L	L	↑	H	H	L	H	H
D32, UAT (0–2)	0 1	0	0	HH	LL	0	H	↑	L	L	L	↑	H	H	L	H	H
D32, UAT (1–3)	1 0	0	0	HH	LH	0	H	↑	L	L	L	↑	H	H	L	H	H
D32, UAT (1–2)	0 0	1	0	HL	LH	0	H	↑	L	L	L	↑	H	H	L	H	H

Table 6–3. Buffer Control Signals: D32 VMEbus Slave Read Operation (continued)

Data Path Size	VMEbus Stimulus			Local Bus Response			Address Control			Data Control			Swap Control				
	DS1/0*	A01	LWORD*	SIZ1/0	LA[1:0]	DSACKi*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
D16 (0–1)	0 0	0	1	HL	LL	0	H	↑	L	L	L	↑	H	L	H	H	H
D16 (2–3)	0 0	1	1	HL	HL	0	H	↑	L	L	L	↑	H	H	L	H	H
D8 (0)	0 1	0	1	LH	LL	0	H	↑	L	L	L	↑	H	L	H	H	H
D8 (1)	1 0	0	1	LH	LH	0	H	↑	L	L	L	↑	H	L	H	H	H
D8 (2)	0 1	1	1	LH	HL	0	H	↑	L	L	L	↑	H	H	L	H	H
D8 (3)	1 0	1	1	LH	HH	0	H	↑	L	L	L	↑	H	H	L	H	H

Table 6–4. Buffer Control Signals: D16 VMEbus Slave Read Operation

Data Path Size	VMEbus Stimulus			Local Bus Response			Address Control			Data Control			Swap Control				
	DS1/0*	A01	LWORD*	SIZ1/0	LA[1:0]	DSACKi*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
D16 (0–1)	0 0	0	1	HL	LL	0	H	↑	L	L	L	↑	H	L	H	H	H
D16 (2–3)	0 0	1	1	HL	HL	0	H	↑	L	L	L	↑	H	H	L	H	H
D8 (0)	0 1	0	1	LH	LL	0	H	↑	L	L	L	↑	H	L	H	H	H
D8 (1)	1 0	0	1	LH	LH	0	H	↑	L	L	L	↑	H	L	H	H	H
D8 (2)	0 1	1	1	LH	HL	0	H	↑	L	L	L	↑	H	H	L	H	H
D8 (3)	1 0	1	1	LH	HH	0	H	↑	L	L	L	↑	H	H	L	H	H

Table 6–5. Buffer Control Signals: D16 VMEbus Slave Write Operation

Data Path Size	VMEbus Stimulus			Local Bus Response			Address Control			Data Control			Swap Control				
	DS1/0*	A01	LWORD*	SIZ1/0	LA[1:0]	DSACKi*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR	SWDEN*	ISOBE*	LWDENIN*	UWDENIN*
D16 (0–1)	0 0	0	1	HL	LL	0	H	↑	L	H	↑	L	L	L	L	L	H
D16 (2–3)	0 0	1	1	HL	HL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (0)	0 1	0	1	LH	LL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (1)	1 0	0	1	LH	LH	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (2)	0 1	1	1	LH	HL	0	H	↑	L	H	↑	L	L	L	L	L	H
D8 (3)	1 0	1	1	LH	HH	0	H	↑	L	H	↑	L	L	L	L	L	H

7

VIC068A Control Register Access

The VIC068A contains 58 8-bit internal registers addressable from the local bus. These registers provide complete control and monitoring of the VIC068A. The seven Interprocessor Communications registers are also addressable from the VMEbus.

Although the registers are 8 bits wide, the VIC068A always acknowledges a register access with both DSACKs. This is because the VIC068A registers are addressed on longword boundaries and occupy 32 bits of address space.

When reading a register, the VIC068A delivers data on LD[7:0]. When writing data, the VIC068A must see data on LD[7:0]. Because of this, the VIC068A only acknowledges a register access that is addressed "correctly" according to the following table:

SIZ1, SIZ0	*VIC068A acknowledges if LA[1:0] is:*
0 0	0 0
0 1	1 1
1 0	1 0
1 1	1 1

This insures that data will be available to/from LD[7:0] according to 68K protocol.

The VIC068A addresses given in this user's guide are byte addresses located at the LA[1:0] = 1 1 position. This implies that if the registers are to be addressed through a longword access, LA[1:0] must be 0 0. If accessed though a word access, LA[1:0] must be 1 0.

For example, if the variable *vic* contains the VIC068A register base address for a particular application, the following Motorola instructions would identically move the DMASR data (address = $BF) to the Motorola D0 register:

```
move.b   (vic,$bf), d0
move.w   (vic,$be), d0
move.l   (vic,$bc), d0
```

8

Interprocessor Communication Facilities

The VIC068A contains three categories of interprocessor communication facilities (ICFs):

- Interprocessor Communication Registers (ICRs)
- Interprocessor Communication Global Switches (ICGSs)
- Interprocessor Communication Module Switches (ICMSs)

The ICRs are 8-bit registers that may be accessed from either the local bus or the VMEbus.

The ICGSs and the ICMSs are switches that may be set to interrupt the local processor.

These facilities are located in seven registers that are visible from both the local bus and the VMEbus. When accessed via the local bus, the registers are read/written by normal VIC068A register access methods. When accessed via the VMEbus, the ICFSEL* signal is used as a register select signal. The register addresses, when accessed from the local bus are not the same as when accessed from the VMEbus. The VIC068A contains an internal arbiter to arbitrate between local and VMEbus accesses to these facilities.

Additional registers used for the ICFs are as follows:

- ICGS Interrupt Control Register (ICGSICR)
- ICMS Interrupt Control Register (ICMSICR)
- ICGS Interrupt Vector Base Register (ICGSIVBR)
- ICMS Interrupt Vector Base Register (ICMSIVBR)
- Interprocessor Communication Switch Register (ICSR)

8.1 Valid ICF Selection

The ICFSEL* signal is used to signal that the VMEbus master desires access to the VIC068A interprocessor communication facilities. This signal is usually driven from VMEbus address decoders. When ICFSEL* is asserted, the VIC068A checks A[5:1] to determine what ICF is desired. Self-access to the ICF facilities is not detected by the VIC068A. The VIC068A then verifies the AM codes against the following table:

ICF	*AM code(s)*
ICR	$29, $2D (A16, user or supervisory data)
ICGS	$2D (A16, supervisory data)
ICMS	$29, $2D (A16, user or supervisory data)

Once a valid ICF select has occurred, the VIC068A then processes the request. The ICF VMEbus address is shown in *Table 8–1.*

Table 8–1. ICF VMEbus Address Map

A7	A6	A5	A4	A3	A2	A1	A0	Function
X	X	0	0	0	0	0	1	ICR0 Access
X	X	0	0	0	0	1	1	ICR1 Access
X	X	0	0	0	1	0	1	ICR2 Access
X	X	0	0	0	1	1	1	ICR3 Access
X	X	0	0	1	0	0	1	ICR4 Access
X	X	0	0	1	0	1	1	ICR5 Access
X	X	0	0	1	1	0	1	ICR6 Access
X	X	0	0	1	1	1	1	ICR7 Access
X	X	0	1	0	0	0	0	Clear ICGS0
X	X	0	1	0	0	0	1	Set ICGS0
X	X	0	1	0	0	1	0	Clear ICGS1
X	X	0	1	0	0	1	1	Set ICGS1
X	X	0	1	0	1	0	0	Clear ICGS2
X	X	0	1	0	1	0	1	Set ICGS2
X	X	0	1	0	1	1	0	Clear ICGS3
X	X	0	1	0	1	1	1	Set ICGS3
X	X	1	0	0	0	0	0	Clear ICMS0
X	X	1	0	0	0	0	1	Set ICMS0
X	X	1	0	0	0	1	0	Clear ICMS1
X	X	1	0	0	0	1	1	Set ICMS1
X	X	1	0	0	1	0	0	Clear ICMS2
X	X	1	0	0	1	0	1	Set ICMS2
X	X	1	0	0	1	1	0	Clear ICMS3
X	X	1	0	0	1	1	1	Set ICMS3
X	X	1	1	X	X	X	X	Undefined/Reserved

8.2 Interprocessor Communication Registers

The VIC068A contains seven interprocessor communication registers (ICRs). These registers are accessible from both the local bus and the VMEBUS. ICRs 4–0 are considered

general-purpose read/write registers. ICR5 is the VIC068A version/revision register. The value of this register indicates the mask revision of the device. ICR6 contains HALT and RESET status of the VIC068A. ICR7 provides semaphores for ICRs 5–0. These semaphores are set whenever ICRs 5–0 are written. In addition, ICR7 also indicates VMEbus mastership and a mask for SYSFAIL*. Refer to Chapter 13 for detailed register descriptions.

8.3 Interprocessor Communication Global Switches

The ICGSs are software switches that may be set over the VMEbus to interrupt a group of VMEbus modules.

When the VIC068A issues the global switches, it is performing a VMEbus byte-wide write to the pre-defined global switch address.

If the global switch interrupts are enabled in the ICGSICR of the VIC068A slave, a local interrupt is generated on a clear-to-set transition of the selected switch. When acknowledged (FCIACK* asserted), the slave VIC068A handles the interrupt by returning the status/ID value from the ICGSVBR. See Chapter 9 for details on VIC068A interrupt generation and handling. Once a switch is set, it must be cleared before it can be re-set.

Because the global switches are meant to be issued to several modules, the VIC068A as VMEbus master must assert DTACK* to complete the VMEbus cycle. The VIC068A issues DTACK* if the ICFSEL* signal is asserted while performing the VMEbus master cycle.

Notice that if a valid ICF select has occurred, A[2:1] selects the ICGS switch and A0 (i.e., DS1* and DS0*) indicates whether the switch is to be set or cleared. That is, a byte-wide write to an even address clears the selected switch and a byte-wide write to an odd address sets the selected switch.

The ICSR may be used to provide monitoring of the global switches.

8.4 Interprocessor Communication Module Switches

Like the ICGSs, the ICMSs are software switches that may be set over the VMEbus to interrupt the local processor. The module switches, however, are meant to be issued to a specific module.

As in the global switches, the VIC068A issuing the module switches performs a VMEbus byte-wide write to the pre-defined switch address.

Because the module switches are meant for a specific module, the VIC068A as VMEbus slave (the module whose switch was just set) must assert the DTACK*. The VIC068A issuing the ICMS need not have its ICFSEL* signal asserted.

The interrupt and addressing mechanisms are the same for ICMSs as they were for ICGSs.

The ICSR may be used to provide monitoring of the module switches. Unlike the global switches, this register may be written by local resources to interrupt the CPU.

9

Interrupts

The VIC068A offers complete VMEbus and local bus interrupt generation and handling functions. In addition, the VIC068A also offers error and status interrupts for various VIC068A features. Local interrupt 2 (LIRQ2) may also be used as a periodic "heartbeat" timer. Significant control over the VIC068A interrupt generation/handling capabilities through the control registers listed below (32 of the 59 VIC068A control registers are for interrupt generation/handling):

- VMEbus Interrupter Interrupt Control Register (VIICR)
- VMEbus Interrupt Control Registers 1–7 (VICR1–7)
- DMA Status Interrupt Control Register (DMASICR)
- Local Interrupt Control Registers 1–7 (LICR1–7)
- ICGS Interrupt Control Register (ICGSICR)
- ICMS Interrupt Control Register (ICMSICR)
- Error Group Interrupt Control Register (EGICR)
- ICGS Interrupt Vector Base Register (ICGSIVBR)
- ICMS Interrupt Vector Base Register (ICMSIVBR)
- Local Interrupt Vector Base Register (LIVBR)
- Error Group Interrupt Vector Base Register (EGIVBR)
- VMEbus Interrupt Request/Status Register (VIRSR)
- VMEbus Interrupt Vector Base Registers 1–7 (VIVBR1–7)

9.1 VMEbus Interrupter

The VIRSR controls the assertion of the VMEbus interrupts. VIRSR[7:1] control the assertion (and deassertion if desired) of IRQ7–1 signals respectively. VIRSR[0] enables the setting and clearing of these bits. To issue an interrupt, both the interrupt bit and VIRSR[0] must be set. To clear an interrupt, the interrupt bit must be set and VIRSR[0] must be cleared. VIRSR[7:1] may also be read to indicate the status of pending VMEbus interrupts. For example, if *vic* contains the base address of the VIC068A registers, the following 68K code could be used to assert IRQ4*:

```
move.b #$11,(vic,$83)
```

This code would clear the interrupt:

```
move.b #$10,(vic,$83)
```

Once the interrupt is issued, the handler for that interrupt proceeds with the interrupt acknowledge cycle. Once the VIC068A recognizes a valid interrupt acknowledge cycle (IACKIN* asserted), it places the status/ID vector, located in the VIVBRi, on the D[7:0] lines. The VIC068A is capable of issuing 8-bit status/IDs only. The VIC068A uses a Release-On-AcKnowledge (ROAK) for the normal deassertion of the IRQi* signals.

More than one VMEbus interrupt may be issued or pending simultaneously by the same VIC068A interrupter.

9.2 The VIC068A VMEbus Interrupt Handler

The VIC068A may be enabled to handle VMEbus interrupts. If VICR1–7[7] is clear, the VIC068A handles all pending interrupts on that respective level. Only one VMEbus module should be configured to handle any given interrupt level per VMEbus system.

The VIC068A performs D8 interrupt acknowledge cycles only.

When the VIC068A detects a pending VMEbus interrupt, it performs the following functions:

1. Assert the IPLi* signals with the value programmed in the VICRi.
2. Wait for the assertion of the FCIACK* signal, which indicates the local processor is acknowledging a local interrupt.
3. Once FCIACK* is asserted, sample LA[3:1] to determine the IPL value of local interrupt being acknowledged.
4. If matched, obtain VMEbus mastership, assert IACK*, drive A[3:1]=interrupt-level and enable D[7:0] to the local data bus LD[7:0].
5. Once DTACK* is asserted by the VMEbus interrupter indicating the status/ID byte is valid on D[7:0], the VIC068A asserts DSACKi* to complete the local interrupt acknowledge cycle and indicates that the status/ID is available on LD[7:0].

The LA[3:1] matching described in step 3 is discussed in section 9.3.

Table 9–1 summarizes the VMEbus interrupt acknowledge cycle.

Table 9–1. VMEbus Interrupt Acknowledge Cycle

Step	VIC068A Interrupter	VIC068A Handler	Local Processor
1	Generate IRQi* (write to VIRSR)		
2		Detect IRQi* asserted (VICRi enabled to handle interrupt)	
3		Assert IPLi value programmed in the VICRi	
4			Detect IPLi asserted
5			Encode IPL level of interrupt on LA[3:1]
6			Assert FCIACK*
7		Detect FCIACK* asserted	
8		Sample LA[3:1] for request level being acknowledged	
9		If matched, obtain VMEbus mastership	
10		Assert IACK*	
11		Enable D[7:0] in	
12	Detect IACKiIN* asserted		
13	Drive status/ID vector programmed in the VIVBRi		
14	Assert DTACK*		
15	Generate interrupt acknowledged interrupt if enabled in the VIICR	Assert DSACKi*	
16			Capture status/ID
17			Enter Interrupt Service Routine

9.3 Local Interrupt Handler

The VIC068A may be enabled to handle local interrupts in addition to VMEbus interrupts. Local interrupt handling is enabled through the LICRs 1–7. If bit 7 is cleared in any of these registers, the corresponding local interrupt is handled by the VIC068A. The local interrupts may be configured in a variety of ways including edge or level sensitivity, active High/Low levels, or rising/falling edges. The VIC068A may be configured to supply, or autovector, the status/ID. If VIC068A is configured to supply the status/ID vector, the vector is supplied from the LIVBR. Unlike the VIVBRs, this is a single register. The upper 5 bits are user-defined, and the lower 3 bits are dynamic to indicate the interrupt value (LIRQ7...LIRQ1, etc.). These lower 3 bits are place-holders only. They may not indicate the interrupt value if read, even during an interrupt acknowledge cycle. If the VIC068A is configured for autovectoring (VIC068A does not supply status/ID), the VIC068A asserts the LIACKO* signal after the interrupt is acknowledged (FCIACK* asserted) by the local processor. This should indicate to the interrupter to place the status/ID on D[7:0] signal lines. LIACK0* may be connected to the 68K AVEC signal for internal generation of the interrupt vector.

Once the VIC068A detects LIRQi* asserted, and the VIC068A is enabled to handle that interrupt, the VIC068A proceeds as follows:

1. Assert the IPLi* signals with the value programmed in the LICRi.

2. Wait for the assertion of the FCIACK* signal, which indicates the local processor is acknowledging a local interrupt.

3. Once FCIACK* is asserted, sample LA[3:1] to determine the IPL value of local interrupt being acknowledged.

4. If matched,
   - drive LD[7:0] with status/ID if enabled to supply vector and assert DSACKi*.
   - or, assert LIACK0* if not enabled to supply vector (autovectoring).

The LA[3:1] matching described in step 3 is required to distinguish an acknowledge from among multiple pending interrupts. If the value of LA[3:1] does not match the value of the IPL signals (see section 9.7 for positive/negative logic issues regarding the IPL signals), the VIC068A assumes the acknowledge is not for it. The 68K family of processors asserts LA[3:1], as described above, automatically. *Table 9–2* summarizes the local interrupt acknowledge cycle.

Table 9–2. Local Interrupt Acknowledge Cycle

Step	Local Interrupter	VIC068A Handler	Local Processor
1	Generate LIRQi*		
2		Assert IPLi value programmed in the VICRi	
3			Detect IPLi asserted
4			Encode IPL level of interrupt on LA[3:1]
5			Assert FCIACK*
6		Detect FCIACK* asserted	
7		Sample LA[3:1] for request level being acknowledged	
8a		If matched, drive status/ID on LD[7:0] programmed in the LICRi	
9a		Assert DSACKi*	
8b		Assert LIACK0*	
9b	Drive status/ID		
10			Capture status/ID
11			Enter ISR

9.4 The Error/Status Interrupts

The VIC068A is capable of generating local interrupts on certain error or status conditions. The error group interrupts include:

- SYSFAIL* assertion. An interrupt is generated when SYSFAIL* is detected by something other than the VIC068A.
- ACFAIL* assertion. An interrupt is generated when ACFAIL* is detected.
- Write post failure. An interrupt is generated when a master write post has failed due to a LBERR* or BERR* assertion.
- VMEbus arbitration failure. An interrupt is generated if the VIC068A is system controller and the VMEbus arbitration timeout timer expires.

The IPL value asserted for these error group interrupts is contained in the EGICR. There is only one IPL value for the error group interrupts.

The VIC068A contains two status interrupts:

- The DMA complete interrupt. An interrupt is generated when a block transfer with local DMA is complete (successful or unsuccessful). The IPL value issued is also contained in the DMAICSR.
- The Interrupter interrupt acknowledge. An interrupt is generated upon the acknowledgment of a previously issued VMEbus interrupt. The IPL value issued is contained in the VMEIICR.

Upon local acknowledgment of both the error and status group interrupts, the VIC068A issues the status/ID byte located in the EGIVBR. EGIVBR[7:3] are user defined. EGIVBR[2:0] are dynamic to indicate the particular interrupt being acknowledged.

9.5 Interrupt Priority Order

The 29 interrupt sources of the VIC068A are grouped into 19 priority categories. With multiple interrupts pending, the VIC068A issues the interrupts in the following order:

Priority	*Interrupt*
1	LIRQ7
2	Error Group
3	LIRQ6
4	LIRQ5
5	LIRQ4
6	LIRQ3
7	LIRQ2
8	LIRQ1
9	ICMS Group
10	ICGS Group
11	IRQ7
12	IRQ6
13	IRQ5
14	IRQ4
15	IRQ3
16	IRQ2
17	IRQ1
18	DMA Complete
19	VMEbus Interrupter

Notice that the priority of the interrupts within the VIC068A is not dependent on the IPL values programmed in the interrupt control registers. This, combined with the fact that

VMEbus interrupt priority is also governed by the position of the module within a VMEbus system, makes determining actual interrupt priority for a given processor in a given VMEbus system flexible.

If an interrupt is pending, and another higher-priority interrupt (according to the above table) is issued to the VIC068A, the VIC068A will change the state of the IPL lines to that of the higher-priority interrupt. The VIC068A, however, still handles the lower-priority interrupt if it is acknowledged before the higher one. If the lower-priority interrupt is being acknowledged (FCIACK* asserted) at the time of the assertion of the higher-priority interrupt, the IPL signals will not change until the interrupt acknowledge cycle is complete.

While an interrupt is pending, all lower-priority interrupts subsequently issued are queued within the VIC068A and issued at the completion of the pending interrupt's acknowledge cycle.

9.6 Clock-Tick Interrupt Generator

LIRQ2 may be enabled to function as a "heartbeat" interrupt generator issuing periodic interrupts. If the interrupt generator is enabled by configuring SS0CR0[7,6], the LIRQ2* pin is converted to an output and issues periodic interrupts in 50-, 100-, or 1000-Hz frequencies. This interrupt should be considered an edge-triggered interrupt since it may be deasserted before the interrupt is acknowledged. The VIC068A may also be configured to handle the interrupt by enabling LIRQ2 in LICR2. In this case, the interrupt is considered to be like any other interrupt issued to the VIC068A and normal interrupt acknowledge procedures apply.

9.7 Interrupt Control Registers

The IPL values programmed in the interrupt control registers are positive logic values. This is unlike the value of the IPL signals asserted to the processor by the VIC068A, which are negative logic. The VIC068A performs this complementing of values internally. That is, if a value of 010 (positive logic 2) is programmed into an interrupt control register, the value of 101 (negative logic 2) is driven on the IPL signals.

When local resources are acknowledging an interrupt and driving LA[3:1] with the level of the interrupt being acknowledged, LA[3:1] should contain the positive logic value of the interrupt.

Mask bits exist for every interrupt the VIC068A is capable of generating. These mask bits are set (interrupts disabled) after a global reset. When changing the level and/or polarities of the local interrupts, it is recommended that the interrupts be masked prior to this and re-enabled after.

10

VIC068A Block Transfer Functions

The ability to transfer large blocks of data at a high-sustained transfer rate is paramount in today's VMEbus market. When implemented properly, transfer rates exceeding 30 Mbyte/sec can be obtained using a high-speed processor, high-speed memory and high-speed VMEbus interfaces such as the VIC068A.

10.1 VIC068A Master Block Transfer

The VIC068A can be configured for block transfers while the local processor is bus master. This is referred to as a *MOVEM* block transfer. The term MOVEM is derived from the Motorola 68K MOVEM (move multiple registers) instruction. The VIC068A can also become the local bus master and implement block transfers using DMA on the local side. This is referred to as *block transfers with local DMA*. For master VMEbus block transfers with local DMA, The VIC068A contains two 8-bit address counters that can automatically increment the address for both the local or CPU side and the VMEbus side.

The VMEbus specification prohibits the crossing of 256-byte boundaries during block transfers without giving up the VMEbus or toggling the VMEbus AS*. The VIC068A allows, with external logic, implementing block transfers that exceed the 256-byte limit. The VIC068A is able to give up the bus at the 256-byte limit (or any limit), then re-arbitrate for the bus at a programmed time later. The time between sub-block transfers is called the *interleave period*. The total number of bytes to be transferred is called the *block transfer length*. The number of VMEbus cycles between interleave periods is called the *burst length*.

The VIC068A also allows for single-cycle VMEbus cycles to be performed during the interleave period. This feature is called the *dual-path* feature and requires external logic such as either the VAC068A or the CY7C964. As the name implies, the dual-path feature requires that a dual address path exist so that the block transfer address is not lost if a local interleave cycle is performed. Slave cycles can be interleaved between master block transfer bursts without external logic or programming.

The VIC068A only supports D32 and D16 block transfers. D8 block transfers are not supported.

VIC068A registers relevant to master block transfers are as follows:

- Block Transfer Control Register (BTCR)
- Block Transfer Definition Register (BTDR)
- Release Control Register (RCR)
- Block Transfer Length Registers (BTLRs)
- Local Bus Timing Register (LBTR)
- Slave Select 0 Control Register 1 (SS0CR1)
- DMA Status Register (DMASR)
- DMA Status Interrupt Control Register (DMASICR)

It is important to note that the BTLRs contain the number of bytes to be transferred, and the burst length in the RCR contains the number of VMEbus cycles, independent of the number of bytes per cycle.

10.1.1 Block Transfers with Local DMA

The block transfer with local DMA is a block transfer in which the VIC068A becomes not only the VMEbus master but the local bus master as well. Upon becoming the local bus master, the VIC068A accesses memory in a DMA-like fashion. As in any DMA operation, this increases the throughput of a system by making memory accesses memory-speed dependent, as opposed to processor-speed dependent. As in any block transfer, the VMEbus slave needs to be enabled for block transfers.

After the VIC068A 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 (since an actual VMEbus write is not performed) or the BLT initialization cycle. This pseudo cycle must be a VMEbus write. A read cycle would not place the correct data on the local data bus. *Figure 10–1* shows the BLT initiation cycle. This pseudo cycle only starts the block transfer mechanisms and loads the addresses. Since any assertion of MWB* after BTCR[6] is set is interpreted as the pseudo cycle, it is important that no normal VMEbus read/writes be performed until after the completion of the block transfer, or during the next interleave period if the dual-path feature (described later) is enabled. It also may be necessary to disable interrupts or have an interrupt service routine that saves the block transfer registers.

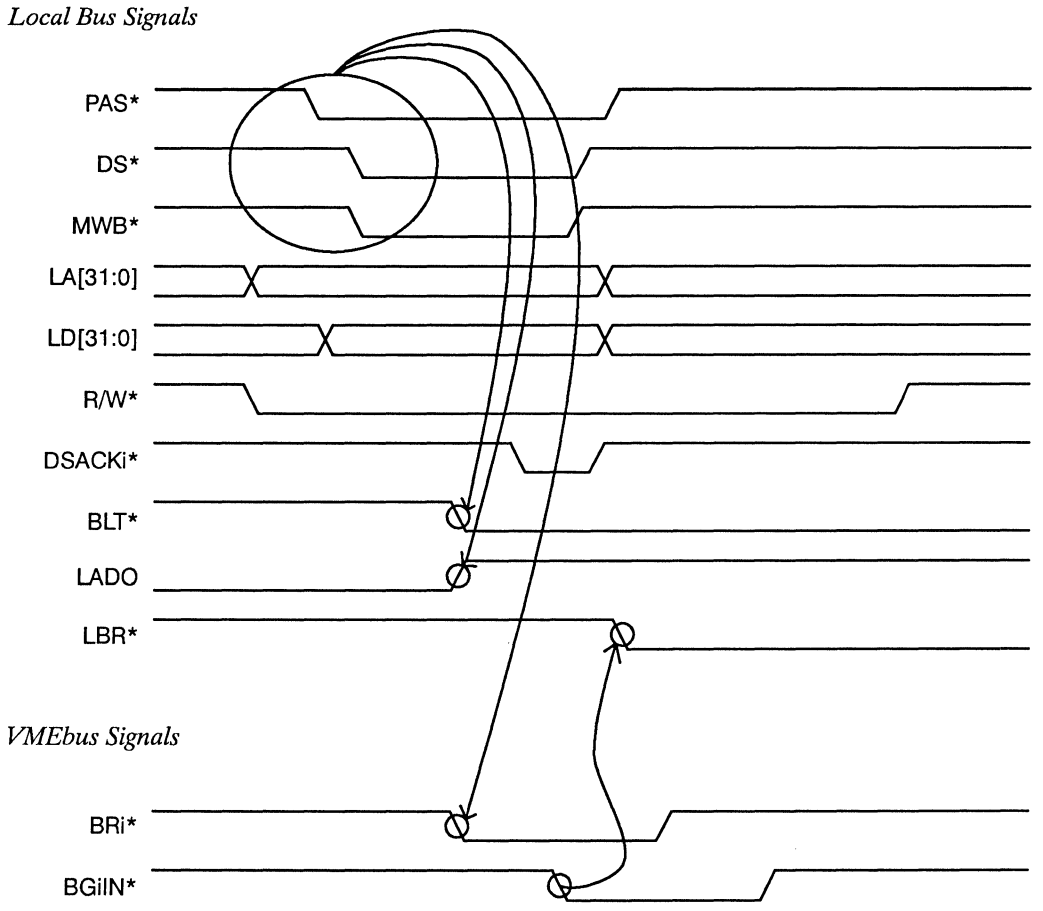


Figure 10–1. Master Block Transfer with Local DMA Initiation Cycle

During the pseudo write, the VIC068A loads the values on the LA[7:0] into its internal VMEbus address counter and asserts LADO to signal external latches to load the remainder of the LA[+:8] as the VMEbus address. The VIC068A at this time also loads the values of LD[7:0] into internal local address counters and asserts BLT* to latch the remainder of the LD[+:8] as the local address. The VIC068A then asserts the DSACKi* signals to terminate the local cycle, and requests the VMEbus. After the VMEbus is granted, the local bus is requested by asserting LBR*. After the local bus is granted, the VIC068A drives the local DMA address onto the local address bus. External logic should decode BLT*, LBG*, PAS*, LAEN, and the FCi signals to drive the upper portion of the address lines. See Section

10.1.1.8.1.3 for more details. The VIC068A then accesses the local data by asserting the local address and data strobes. The local resource then acknowledges the VIC068A that local data has been read or written by asserting the DSACKi* signals. Local data is held in the interface while the local address is incremented.

10.1.1.1 DMA Burst Length

Recall that the VIC068A is able to divide block transfers into a programmable number of bursts. The length of a burst is programmable from 1 to 64 cycles through writing RCR[5:0]. Clearing these bits (the default value) implies a burst length of 64, not 0. The burst count is independent of the number of bytes transferred per cycle.

10.1.1.2 Block Transfer Length

The total length of a block transfer is programmed separately from the burst length. The length is programmed by writing the BTLRs. The LSB of the number is programmed into register BTLR1. Since the VIC068A does not support D8 block transfers, if BTLR1[0] is set, the BTLRs are ignored and only a single burst will be performed.

10.1.1.3 256-Byte Boundary Crossing on the VMEbus

The VMEbus specification prohibits the crossing of 256-byte address boundaries. It is possible, when transferring large amounts of data, to simply deassert the AS* at the boundary crossing and reassert it after the address has incremented without releasing the VMEbus. In this case, BBSY* is held Low so that VMEbus mastership is not lost during the toggle of the AS*. Once the boundary has been crossed, the VIC068A releases BBSY* (if configured for RWD) after AS* is reasserted. The VIC068A, when enabled for VMEbus boundary crossing in the BTDR, does this without any user intervention. External circuitry is required to increment any address bits other than the lower 7 bits driven by the VIC068A. LADO, ABEN*, and the FCi function codes may be used to control the loading, holding, and incrementing of external latches and counters. During the VMEbus cycle before the boundary crossing, LADO will toggle. External counters should increment after two edge transitions of LADO. This is because LADO pulses Low if dual-path is not enabled and pulses High if dual-path is enabled. If the VAC068A or CY7C964 is used in conjunction with the VIC068A, no external hardware for boundary crossing is required.

10.1.1.4 256-Byte Boundary Crossing on the Local Bus

Just as the VIC068A allows for 256-byte boundary crossing on the VMEbus, similar provisions are made for the local address bus. Local boundary crossing is enabled in the BTDR.

External circuitry is again required to increment any address bits other than the lower 8 bits driven by the VIC068A. BLT* and the FCi function codes may be used to control the loading and incrementing of these upper address bits. During the local cycle before the boundary crossing, the BLT* will toggle. External counters should increment on the falling edge of BLT*. If the VAC068A or CY7C964 is used in conjunction with the VIC068A, no external hardware for boundary crossing is required.

10.1.1.5 Interleave Period

The time between bursts is known as the interleave period. The length of the interleave period is configurable in RCR[5:0]. During the interleave period, slave cycles can be performed. Master cycles are allowed if the dual-path feature is enabled.

10.1.1.6 Dual Path

Normally, during the interleave period no VMEbus master cycles are allowed by the VIC068A. All requests for VMEbus master cycles are DEDLKed by the VIC068A. If, however, the VIC068A is configured for dual-path operation, the VIC068A will allow master cycles. If the dual-path option is to be used, external logic such as the VAC068A or CY7C964 is required to maintain the local and VMEbus addresses at the completion of the last burst. If the VIC068A is enabled for local and VMEbus address boundary crossing, the external counters (assuming all 32 bits are handled by counters) may be used. If the VAC068A is used in conjunction with the VIC068A, no external hardware is required for the dual-path option.

All VMEbus interrupt acknowledge cycles are DEDLKed by the VIC068A whether dual-path is enabled or not.

10.1.1.7 The Block Transfer with Local DMA Enable Bit

When the BLT enable bit (BTCR[6]) is set, any assertion of MWB* (qualified by PAS* and DS*) will begin the block transfer. Special care must be taken to insure that this bit is set and cleared as close as possible to the actual block transfer. This means that the BLT enable bit must be cleared directly after the completion of the block transfer. This is important if retry logic is employed for deadlocked VMEbus transfers. For example, if an attempted VMEbus cycle is DEDLKed during an interleave period when dual-path is not enabled, the processor, such as a 68K processor, may continually retry the cycle waiting for the VIC068A to complete the block transfer. At the completion of the block transfer, this retry could initi-

ate another block transfer since the BLT enable bit has not been cleared. For this reason, retry logic may have to be disabled during block transfers with local DMA.

10.1.1.8 Example Configurations

The following paragraphs further explain some application details regarding the different modes of block transfer operation.

10.1.1.8.1 Boundary Crossing Disabled

This is the simplest form of the block transfer with local DMA. In this mode, block transfers are restricted to those that do not cross a 256-byte boundary on either the local bus or the VMEbus. Enabling dual path has no effect since there will not be an interleave. This cycle is initiated by writing the following registers on the master module:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA complete interrupt

To enable the slave module for block transfers, configure the SS0CR0 or SS1CR0 register in the slave's VIC068A as appropriate.

10.1.1.8.1.1 Sample 68K Code (Write)

If we assume that the VIC068A registers start at a base location of *vic_reg*, a sample of 68K code that would configure the VIC068A for a block transfer with local DMA could be as follows:

```
; Set up VIC068A to block transfer 128
; bytes in the BTLRs.
      move.b #$00, (vic_reg, $db)
      move.b #$80, (vic_reg, $df)
;
; Disable boundary crossing and dual
; path in the BTDR. This is the default configuration,
```

```
; but it is good to confirm the value.
      move.b #$00, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
; (These were probably set up at
; power-up configuration)
      move.b #$44, (vic_reg, $a7)
; Configure for:
; no interleave period
; write to VMEbus
; and enable block transfer w/ local
; DMA in the BTCR.
      move.b #$40, (vic_reg, $d7)
```

After the block transfer bit is set, the local processor performs a 32-bit write to the VMEbus location. This causes MWB* to be asserted to the VIC068A, which then requests the VMEbus. Any VMEbus access through the VIC068A then causes the VIC068A to enter the block transfer mode. It is important that this first transfer contain the starting address of local memory, not the data to be transferred.

If the 68K A0 register contains the local starting address and VME contains the VMEbus starting address, then the following instruction can be used to start the block transfer.

```
; Start the block transfer
      move.l vme, A1
      move.l local, A0
      move.l (A0), (A1)
```

The local processor can check for the end of the block transfer by checking the DMASR. The following code does this:

```
; Test the DMA bit of the DMASR.
      LOOP:
      btst #0, (vic_reg, $BF)
      bne LOOP
```

Also, the VIC068A can be programmed to issue an interrupt to the local processor to indicate the end of the transfer. Use the DMSICR to enable this operation.

10.1.1.8.1.2 Sample 68K Code (Read)

For this example, the set-up is the same as in the previous example with the exception that the BTCR would be set as follows:

```
; Configure for:
; no interleave period
; read from VMEbus
; and enable block transfer w/
; local DMA in the BTCR.
      move.b #$50, (vic_reg,$d7)
```

The transfer could be started with the same code fragment used for the write example. As before, the completion of the DMA could be indicated by testing DMASR[0], or by an interrupt.

10.1.1.8.1.3 External Hardware Implementation

Figure 10–2 shows the hardware required for implementing the block transfers described above. The only additional logic required is for latching the upper address bytes, LA[31:8]. Notice that for normal transfers, the '543 drives the address bus. For block transfers, the address bus is driven by the '373. The '373 is loaded from LD[31:8] at the assertion of BLT*.

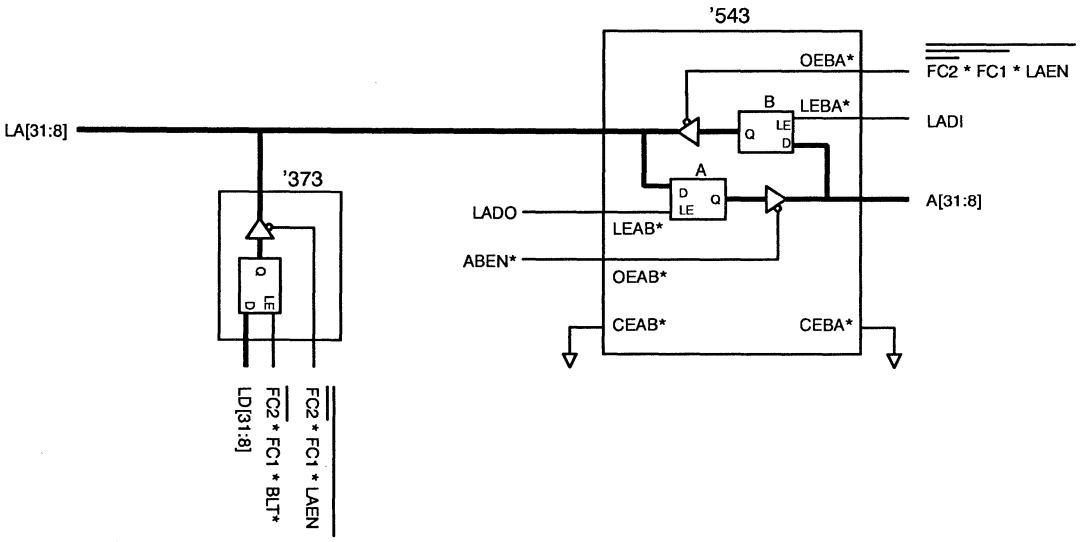


Figure 10–2. Minimum BLT Logic

10.1.1.8.2 Boundary Crossing Enabled, Dual-Path Feature Disabled

This mode allows implementing block transfers that are larger than the 256 bytes restricted by the VMEbus specification. The VIC068A does this by giving up the VMEbus after a 256-byte burst (or any size burst) and re-arbitrating for it at a later time (specified in the BTCR). The VIC068A system keeps track of all information required during the transfer so no additional software overhead is required. Some of the additional logic required for this operation consists of external counters that increment the LA[+:8] and A[+:8] counters as required. The VIC068A counts the lower 8 address bits only.

This cycle is initiated by writing the following registers on the master module:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- RCR, to specify burst length
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA-complete interrupt

Notice that the RCR must now be written to in order to specify the burst length.

The slave module must always be enabled for block transfers.

The software set-up for this operation is very similar to that of any block transfer with local DMA. The RCR must be written to specify the burst length.

10.1.1.8.2.1 Sample 68K Code (write)

If we assume that the VIC068A registers are at *vic_reg*, as in the previous example, a sample of 68K code that would configure the VIC068A for a block transfer with local DMA and boundary crossing could be as follows:

```
; Set up VIC068A to block transfer 512
; bytes in the BTLRs.
      move.b #$02, (vic_reg, $db)
      move.b #$00, (vic_reg, $df)
;
; Specify a burst length of 64 cycles
```

```
; in the RCR
; ($00 specifies 64 cycles)
      move.b #$00, (vic_reg, $d3)
;
; Enable boundary crossing for both
; the local bus and the VMEbus.
; Disable dual path. Use the BTDR
      move.b #$0c, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
      move.b #$44, (vic_reg, $a7)
;
; Configure for:
; interleave period = 0
; write to VMEbus
; and enable block transfer w/ local
; DMA using the BTCR.
      move.b #$40, (vic_reg, $d7)
```

As in any block transfer with local DMA, the VMEbus is requested at the assertion of MWB*. In the above example, two bursts of 256 bytes (64 cycles of longwords) are performed. Between these bursts, an interleave time of "0" was specified. In this case, the VMEbus is immediately requested after it is given up after the first burst and the local bus will not be released. If a VMEbus master cycle is attempted during the interleave period, it will be DEDLKed (dual-path is not enabled).

The block transfer could be started as in any of the previous examples. Program the VIC068A to issue an interrupt or check for the completion of the transfer with the DMASR.

Notice that no additional software overhead is required (with the exception of writing the RCR) to perform a boundary crossing block transfer.

10.1.1.8.3 Boundary Crossing Enabled, Dual-Path Enabled

This mode of operation is similar to that of the previous example except that master cycles are allowed during the interleave period. If a master cycle is desired (MWB* asserted), the

VMEbus is requested and the transfer takes place like any other master cycle. A request for a master cycle is DEDLKed if dual-path is not enabled.

The same registers that were used in the previous example are used in this example. Namely:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- RCR, to specify burst length
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA-complete interrupt

The required software set-up is no different from that of any block transfer with local DMA except that the dual-path bit (BTDR[0]) must be enabled.

10.1.1.8.3.1 Sample 68K Code (Write)

A sample of 68K code that would configure the VIC068A for a block transfer with local DMA with boundary crossing and dual-path could be as follows:

```
; Set up VIC068A to block transfer 512
; bytes in the BTLRs.
      move.b #$02, (vic_reg, $db)
      move.b #$00, (vic_reg, $df)
;                  ;
; Specify a burst length of 64 cycles
; in the RCR.
; ($00 specifies 64 cycles)
      move.b #$00, (vic_reg, $d3)
;
; Enable boundary crossing for both
; the local bus and the VMEbus.
; Disable dual path. Use the BTDR.
      move.b #$0d, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
```

```
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
      move.b #$44, (vic_reg, $a7)
;
; Configure for:
; interleave period = 1000ns
; write to VMEbus
; and enable block transfer w/ local
; DMA using the BTCR.
      move.b #$44, (vic_reg, $d7)
```

The VMEbus is requested at the assertion of MWB*.

In this scenario, a block transfer of 512 bytes is performed in two blocks of 256 bytes. During the interleave period (1000 ns), master cycles can be performed.

10.1.1.9 D16 Block Transfers

The VIC068A is capable of performing D16 block transfers. If the WORD* signal is asserted during the initiation cycle, the VIC068A performs the block transfer with D16 protocol on the VMEbus.

The SWDEN* and ISOBE* signals can be configured to alternately toggle between the lower and the upper word banks, or swap all data to the upper word bank. If SS0CR0[4] (D32 enable) is set, SWDEN* and ISOBE* will toggle for 32-bit memory. If it is cleared, the data will not switch between the upper and lower banks; only SWDEN* is asserted.

10.1.1.10 Data Acquisition Delays

The programmable delays in SS0CR1 take on different meanings depending on whether the system is reading, writing, or even whether it is the first transfer or subsequent transfers. During block transfer writes, the MBAT timing reflects the minimum delays. The actual delays will depend on the speed in which DTACK* for the previous cycle is asserted. Notice that the VIC068A asserts the DS* signals before the DTACK* signal has been asserted in an attempt to read-ahead the next data. During the read-ahead cycle, the VIC068A cannot deassert the LEDO and DS* signals before DTACK* is asserted from the previous cycle.

Master Write Block Transfer (MBAT1/0)

DSACKi*(L) & DS*(L) to DS*(H)
DSACKi*(L) & DS*(L) to LEDO(H)

DSACKi*(L) & DS*(L) to DSi*(L)
DSACKi*(L) & DS*(L) to LA(7:0)

Master Read Block Transfer (MBAT1/0)

DSACKi*(L) & DS*(L) to DS*(H)
DSACKi*(L) & DS*(L) to LEDI(H)
DSACKi*(L) & DS*(L) to DSi*(L)
DSACKi*(L) & DS*(L) to LA(7:0)

The terms MBAT refer to the following bit fields:

MBAT0 = SSiCR1[3:0]
MBAT1 = SSiCR1[7:4]

Figures 10–3 through *10–6* demonstrate these delays. See Chapter 14 for the actual AC performance specifications.

10.1.2 MOVEM Block Transfers

The MOVEM block transfer is one in which the local CPU continues to be the local bus master. The VMEbus block transfer protocol is the same as it was for block transfers with local DMA. This cycle is initiated by writing the registers:

- RCR, to configure burst length
- BTCR, to configure block transfer

Once BTCR[5] is set, the first assertion of MWB* qualified with the assertion of PAS*, causes the VIC068A to start the MOVEM transfer. The MOVEM block transfer will terminate if any of the following occur:

- BTCR[5] is cleared
- BERR* is asserted
- any local bus cycle occurs in which MWB* is not asserted

BTLR1/0 are not used for MOVEM block transfers. During MOVEM block transfers, there is no latching of addresses. The VIC068A passes the local address onto the VMEbus address lines directly.

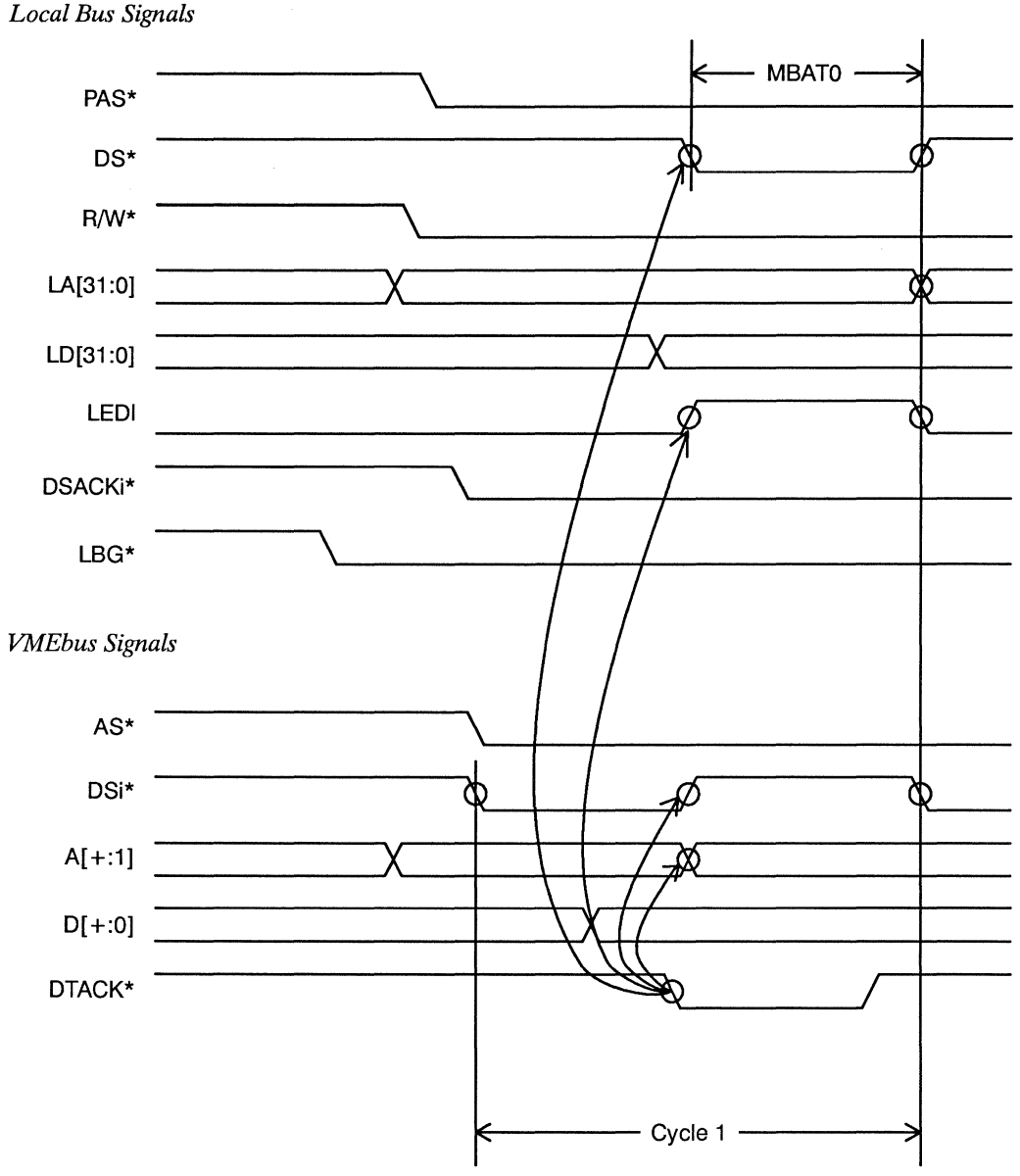


Figure 10–3. Master Block Transfer—Read, First Cycle

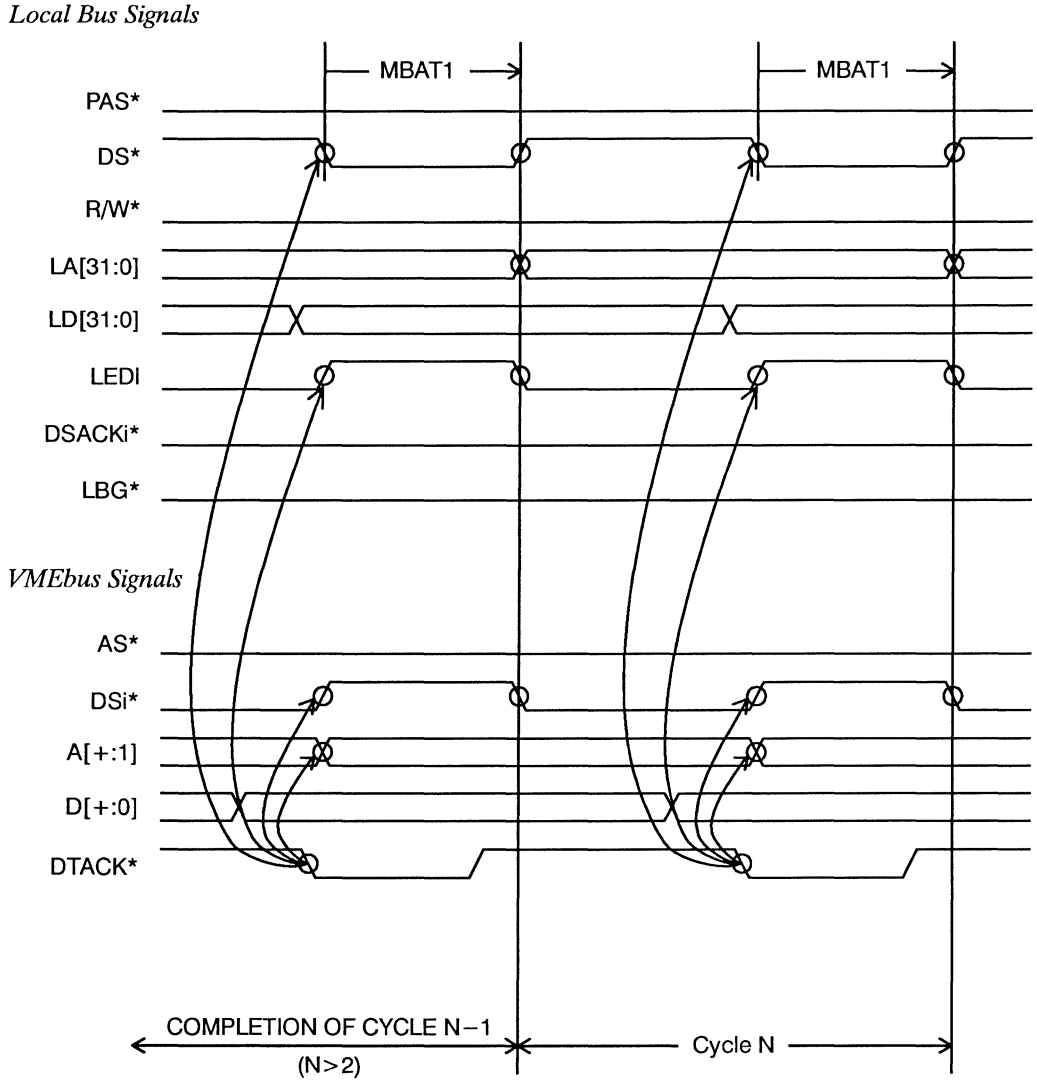


Figure 10−4. Master Block Transfer—Read, Second and Subsequent Cycles

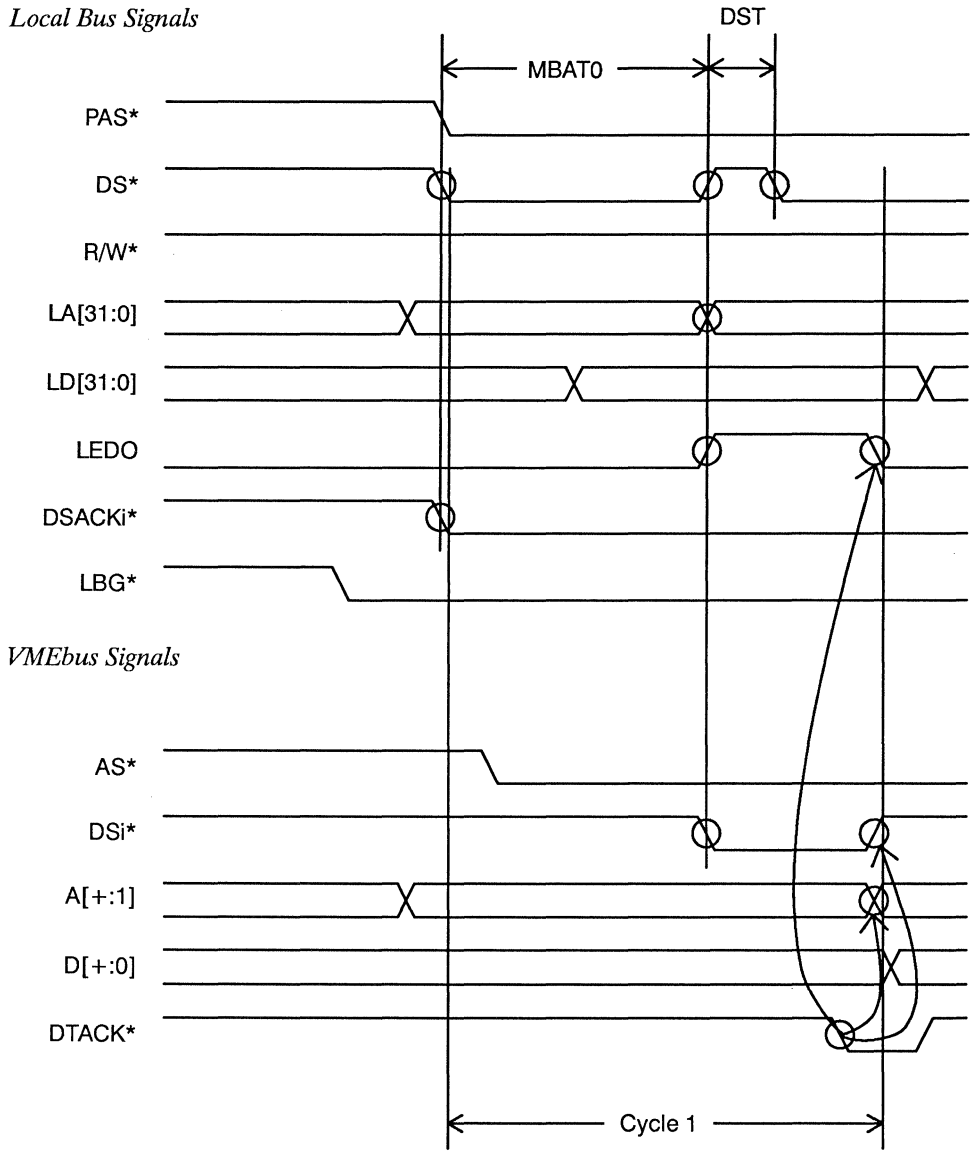


Figure 10–5. Master Block Transfer—Write, First Cycle

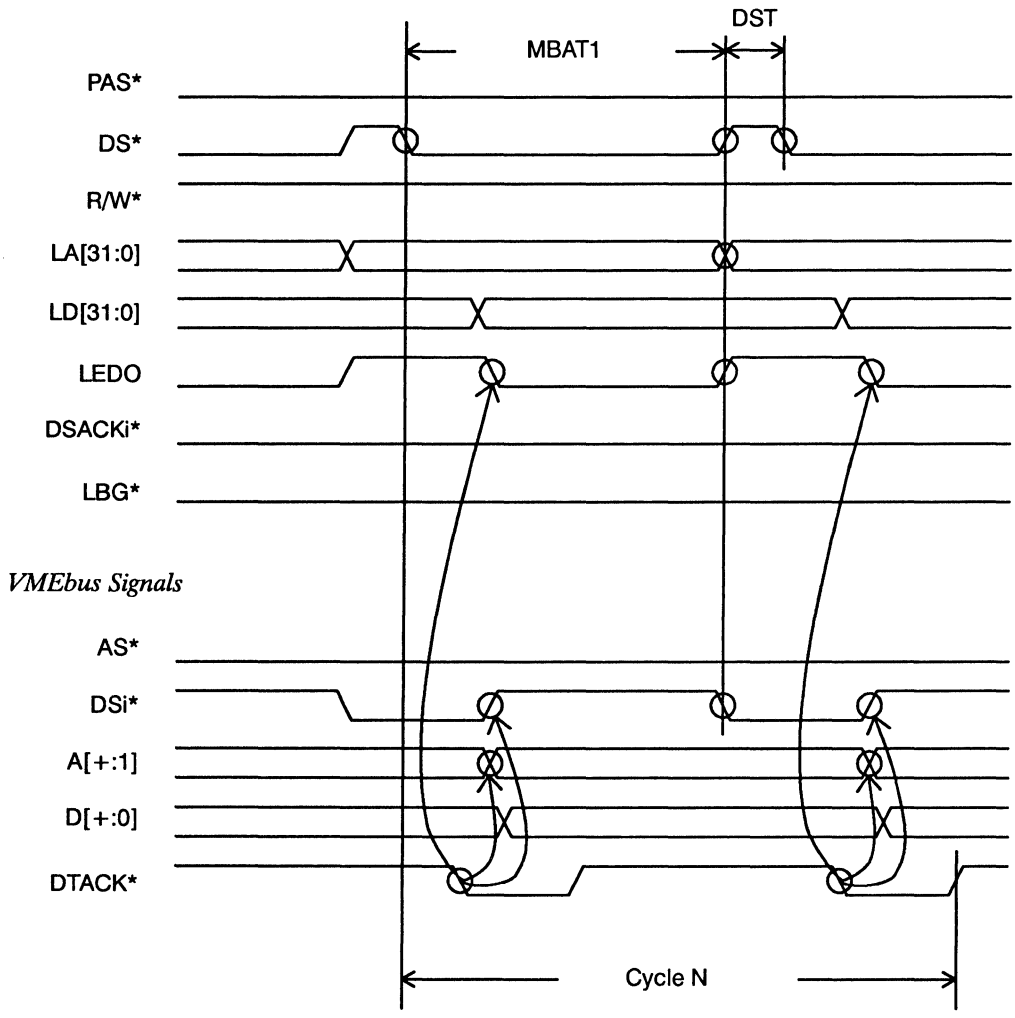


Figure 10–6. Master Block Transfer—Write, Second and Subsequent Cycles (Fast Slave)

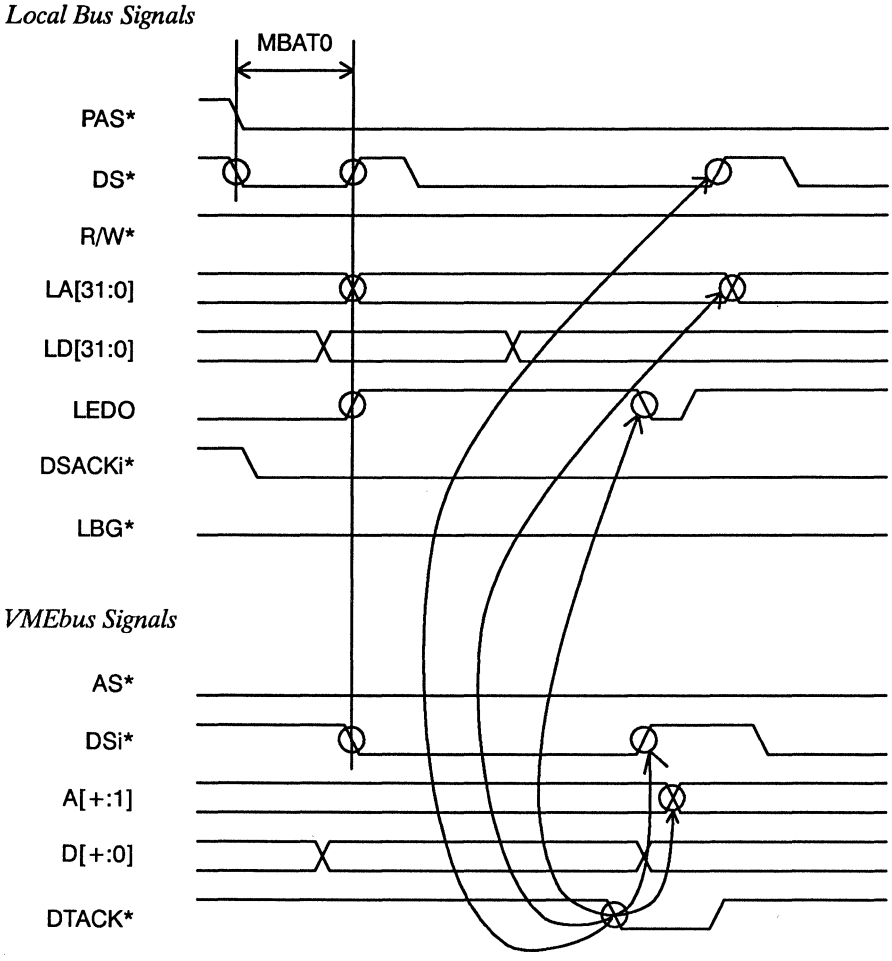


Figure 10–7. Master Block Transfer—Write (Slow Slave)

10.1.2.1 Sample MOVEM 68K Code

If, for example, we assume the VIC068A registers start at the base address *vic_reg*, a sample of 68K code that would configure a MOVEM block transfer and dump the word contents of the 68K d0 through d7 registers to memory could be as follows:

```
; Write burst length in to the RCR.
      move.b #$0f, (vic_reg, $d3)
;
; Set the MOVEM bit in the BTCR.
      move.b #$20, (vic_reg, $d7)
;
; Perform the MOVEM instruction.
; (a0 contains the destination
; address)
      movem.l 0xffff, (a0)
;
; Clear the MOVEM bit.
      move.b #$00, (vic_reg, $d7)
```

10.1.3 Buffer Control Signals During Master Block Transfers

The buffer control signals provide latching and bus-driving control for the address and data lines on both local and VMEbus sides. For MOVEM block transfers, the buffer control signals are identical in behavior to that of normal master transfers. For block transfers with local DMA, the behavior of the buffer control signals may act slightly differently. When using block transfers with local DMA, the loading, holding, and incrementing of the address counters and latches is controlled by LADO, ABEN* and the FC2/1 function codes.

Initialization Cycle

LADO is asserted as a result of PAS*, DS*, and MWB* being asserted for the initiation cycle.

VMEbus Read (Local Write)

LEDI is asserted as a result of the VMEbus DTACK* being asserted by the slave, indicating that valid data is available on the VMEbus. LEDI is deasserted as a result of MBAT1/0 expiring.

VMEbus Write (Local Read)

During block transfer writes, the MBAT timing reflects the minimum delays. The actual delays will depend on the speed in which DTACK* for the previous cycle is asserted. The

VIC068A asserts the DS* signal before the DTACK* signal has been asserted in an attempt to read-ahead the next data. During the read-ahead cycle, the VIC068A cannot deassert the LEDO and DS* signals before DTACK* is asserted from the previous cycle.

10.1.4 Performing Block Transfers to VMEbus Slaves Not Supporting Block Transfers

The VIC068A may be used to perform block transfers with local DMA to cards that do not accept block transfers. This is accomplished by performing the block transfer in bursts of 1 and using single-cycle AM codes (set BTDR[0]). This makes the VMEbus data appear to be single-cycle data, but data is transferred on the local bus by DMA. Because the VIC068A is in a block transfer mode, the LADO signal operates by deasserting after DS* is asserted (recall that the address for single-cycle transfers must be held for the duration of the cycle, but for block transfers the slave is responsible for latching the address at the beginning of the cycle). When the VIC068A is configured for bursts of 1, LADO may be configured to behave differently depending on certain register configurations. If BTDR[3] is set (VMEbus boundary crossing enabled), LADO is deasserted when DSi* is asserted for the final transfer. If BTDR[3] is clear, LADO will hold until the final DTACK* of the final transfer. Therefore, it is recommended that boundary crossing be disabled when performing these burst-of-1 transfers to non-block slaves.

10.2 VIC068A Slave Block Transfers

The VIC068A as a slave may be programmed in one of three modes for block transfers:

- does not support block transfers
- supports block transfers, but emulates a standard transfer on the local slave side (toggle PAS* and DSACKi* for each transfer)
- supports block transfers in a DMA-type mode where PAS* and DSACKi* are asserted throughout the cycle and not toggled

The VIC068A contains a slave block-transfer address counter separate from either of the address counters used for master block transfers. This counter, initialized by the VMEbus address, drives the local bus during the slave block transfer, and is incremented by the amount of data transferred with each local acknowledgement.

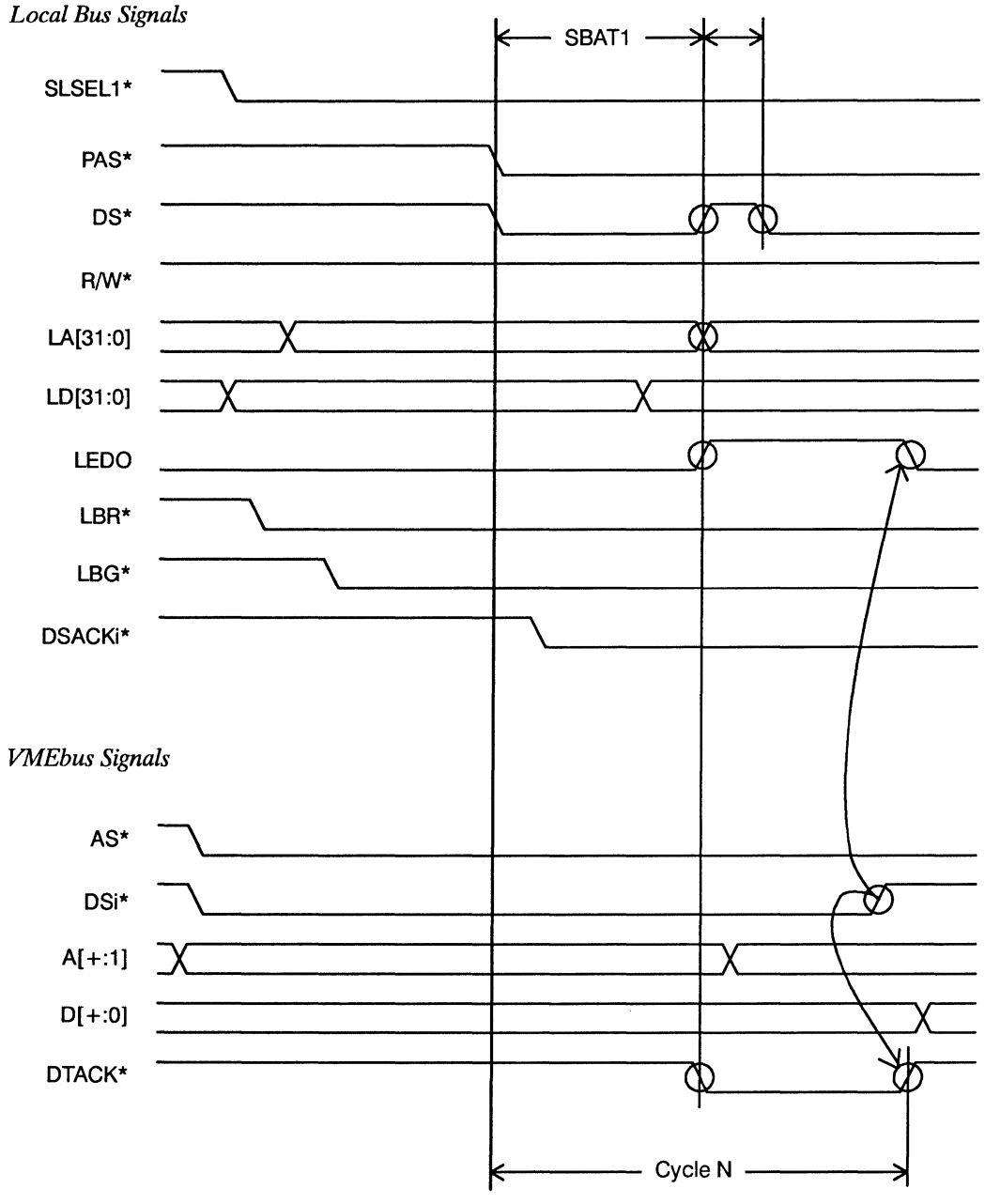


Figure 10–8. Slave Block Transfer—Read, First Cycle

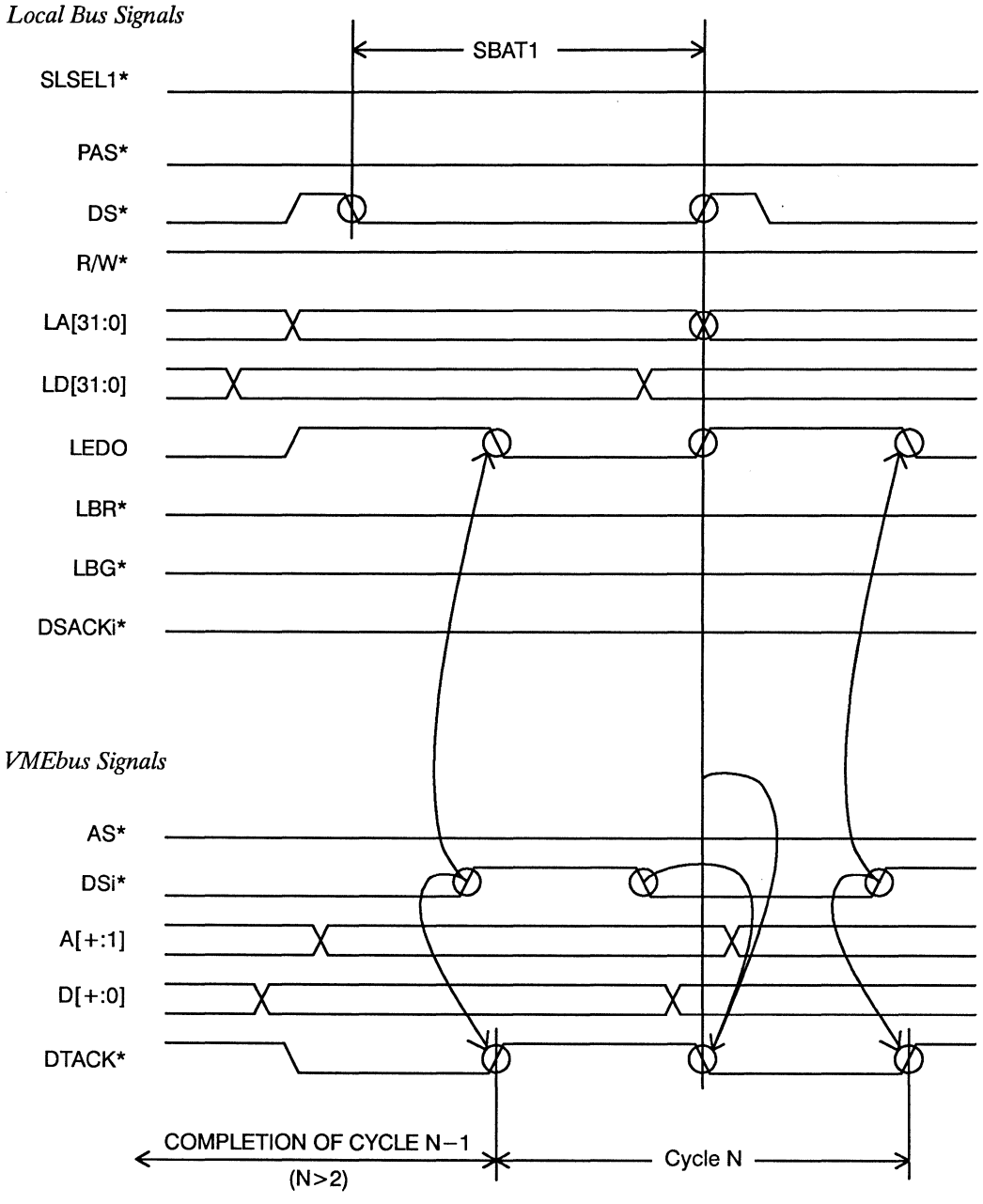


Figure 10−9. Slave Block Transfer—Read, Second and Subsequent Cycles

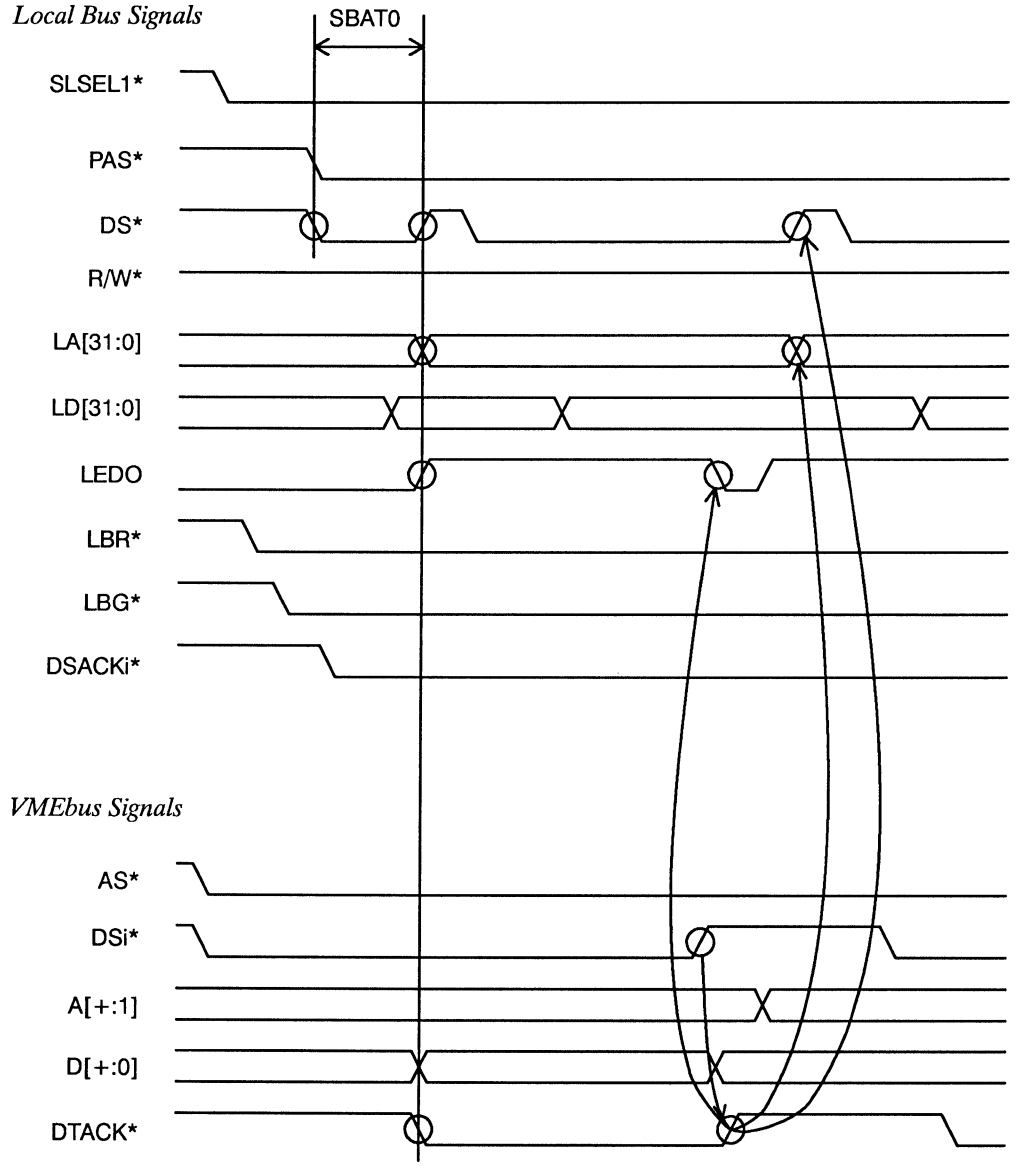


Figure 10–10. Slave Block Transfer—Read, Slow Master

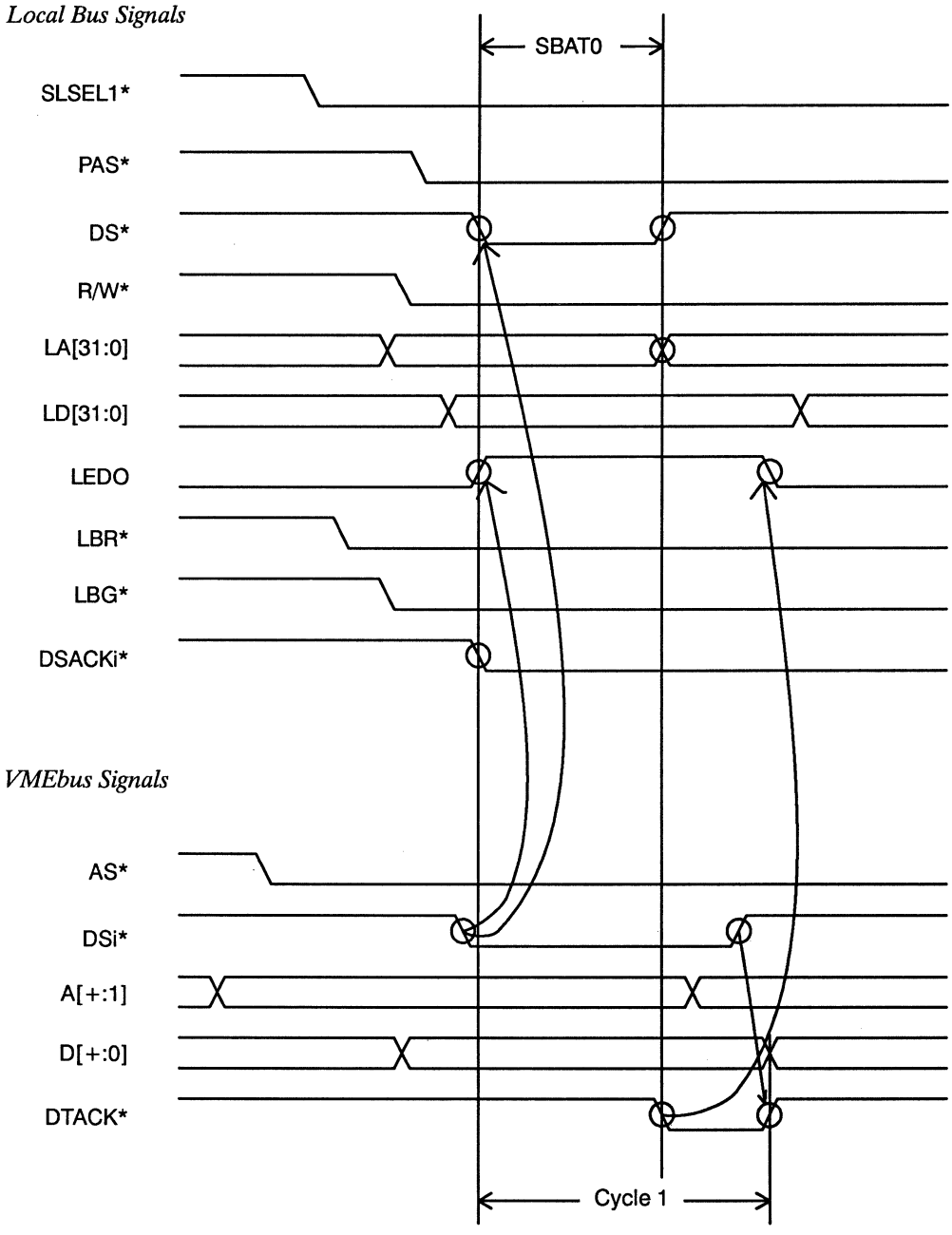


Figure 10–11. Slave Block Transfer—Write, First Cycle

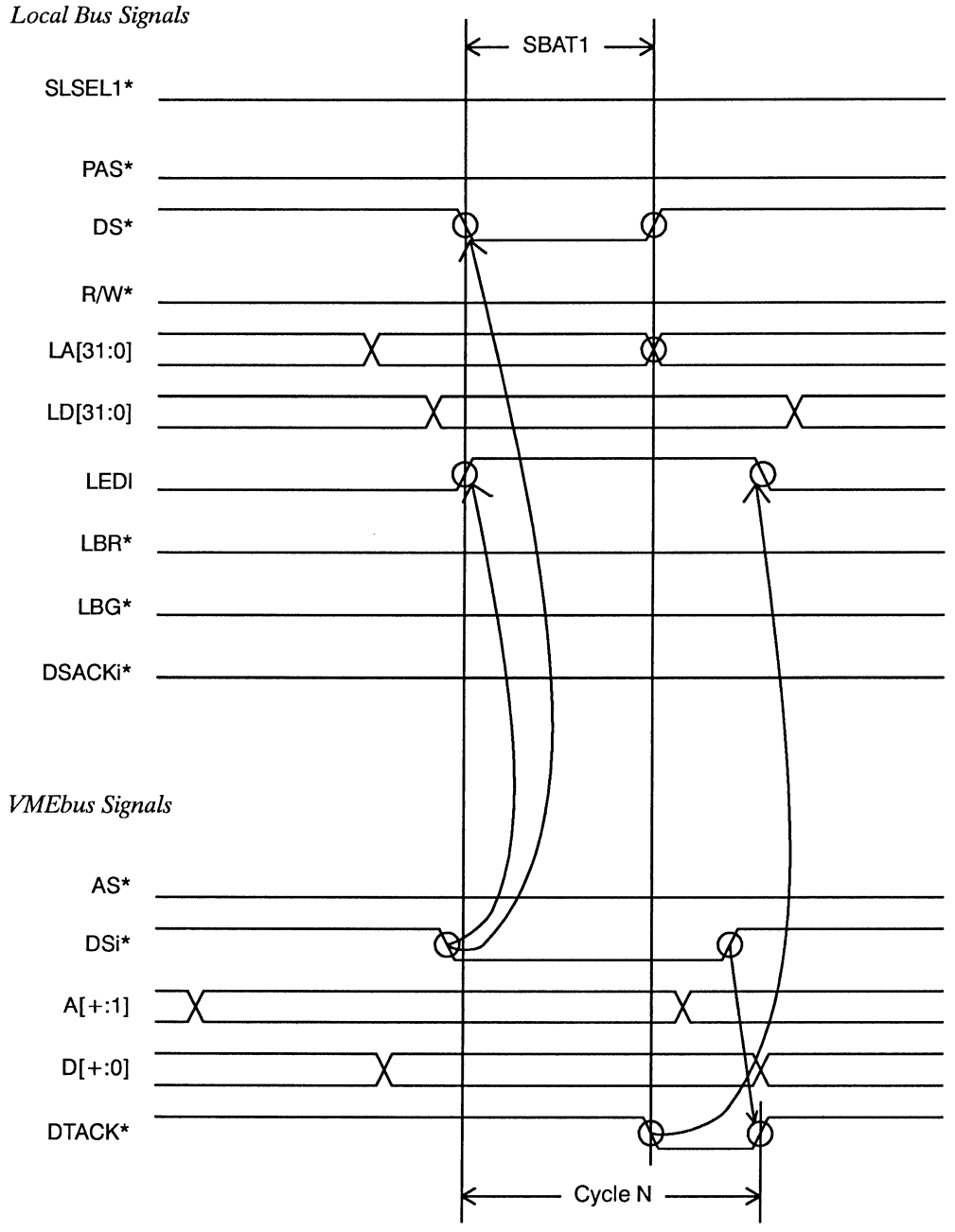


Figure 10–12. Slave Block Transfer—Write, Second and Subsequent Cycles

The timing for the slave's response to a block transfer is the same for that of a master block transfer with local DMA. VIC068A registers relevant to slave block transfers are as follows:

- Slave Select Control Registers (SS0CRi, SS1CRi)
- Local Bus Timing Register (LBTR)

The DMASR, DMASIR, RCR, BTCR, and the BTLRs are not used for slave block transfers. Slave block transfers that cross 256-byte boundaries are not supported in accordance with the VMEbus specification.

10.3 Buffer Control Signals During Slave Block Transfers

When the VIC068A is selected as a valid slave, LADI is asserted to latch the first address. From that point on, the VIC068A increments the lower 8 local address bits for each transfer. The data latches work as follows:

VMEbus Write (Local Write)

LEDI is asserted as a result of the VMEbus DSi* being asserted. LEDI is deasserted as a result of SBAT1/0 expiring.

VMEbus Read (Local Read)

Similar to master block transfer writes, the VIC068A attempts to read ahead the next data while the cycle is completing on the VMEbus. If DSi* from the previous cycle is not deasserted before SBAT1 expires for the read-ahead cycle, LEDO and DS* are deasserted at the deassertion of DSi*. If DSi* is deasserted before SBAT1 expires, LEDO and DS* are deasserted at that point.

10.4 Using the VAC068A for Additional Block Transfer Support

The VAC068A is ideally suited for applications that require the extended block transfer features. A VIC068A/VAC068A VMEbus interface requires no additional logic for incorporating

- local boundary crossing
- VMEbus boundary crossing

- dual-path feature

See the VAC068A section of this user's guide for further details on block transfer functions with the VAC068A.

10.5 Using the CY7C964 for Additional Block Transfer Support

The CY7C964s are perfect companion chips for designs for which all of the VAC068A's functionality is not required, or for designs that cannot incorporate the VAC068A's address mapping.

The CY7C964 provides all logic necessary for

- local boundary crossing
- VMEbus boundary crossing
- dual-path functionality

See the CY7C964 User's Guide for details on a VIC068A/CY7C964 interface.

11

Module-Based DMA Transfers

The VIC068A is capable of performing DMA transfers between local memory and another local resource such as an I/O device. This mode cannot be used for memory-to-memory DMA transfers without external logic. This is because the VIC068A only drives a single AS* and DS* and includes only one local address counter.

Module-based DMA transfer is very similar to the VMEbus block transfer with local DMA discussed in Chapter 10. Here is a summary of the main differences:

1. enabled by setting BTCR[7]
2. does not use the VMEbus as the source/destination
3. ASIZ1 is used as the acknowledge for the second resource
4. ASIZ0 is used as the bus-error signal for the second resource
5. after registers are initialized, the operation is started by asserting the BLT* signal (as opposed to the MWB* signal)

VIC068A registers relevant to module-based DMA transfers are:

- Block Transfer Control Register (BTCR)
- Block Transfer Definition Register (BTDR)
- Release Control Register (RCR)
- Block Transfer Length Registers (BTLRs)
- Local Bus Timing Register (LBTR)
- Slave Select 0 Control Register 1 (SS0CR1)
- DMA Status Register (DMASR)
- DMA Status Interrupt Control Register (DMASICR)

It is important to note that the BTLRs contain the number of bytes to be transferred, and the burst length in the RCR contains the number of VMEbus cycles, independent of the number of bytes per cycle.

11.1 BLT* Assertion Cycle

Once the DMA configuration is set up and BTCR[7] is set, an assertion of BLT* causes the VIC068A to enter the module-based DMA mode and the VIC068A captures the contents of LD[7:0] as the lower byte of the address. This assertion of BLT* should also cause external logic to latch the remainder of LD[31:8] as the local address. Once the DMA mode is entered, the VIC068A will request the local bus. Once the local bus is obtained, external logic should drive the local address bus with the newly captured address. The FC2/1* signals may be used to further qualify the driving of the local address bus. If the DMA transfer is a dual-address transfer, the assertion of BLT* should cause the LA[31:0] to be latched in external latches.

11.2 Module-Based DMA Transfer Flow Diagrams

The following paragraphs describe a typical flow for module-based DMA transfers. The term *device* refers to the I/O device (or possibly memory) that is to be the second source or destination (asserting ASIZ0/1). The term module refers to all other logic controlling such things as local arbitration, memory control, and VIC068A set-up.

11.2.1 DMA Memory Read

For a DMA memory read operation, the VIC068A acts as a DMA controller for a memory-to-device transfer. In this mode, the VIC068A asserts the DS* signal to signal a read from local memory. Once this data is available, the module asserts DSACKi*. The device, detecting DSACKi* asserted, captures the data and deasserts ASIZ*. This continues until the transfer is complete.

Once the DMA operation is configured and BTCR[7] is set, the following events occur for a DMA memory read.

Initialization:

The Module

1. place DMA target address on LA[31:0]
2. drive R/W* signal High for a read operation

3. assert PAS*
4. place memory address on LD[31:0]
5. asserts DS*
6. assert BLT*
7. capture remainder of LD[31:8] as local address

The VIC068A

8. detect BTCR[7] = 0 and BLT* asserted
9. capture LD[7:0] into internal address counter
10. assert BLT*
11. assert LBR*
12. assert DSACK1/0*
13. wait for LBG*

The Module

14. complete local cycle, assert LBG*
15. release BLT* (it remains asserted however because the VIC068A is still asserting it)

The VIC068A

16. detect LBG*, wait for local cycle to complete
17. start PAS* and DS* minimum High delays
18. drive LA[7:0] with DMA start address
19. drive R/W* High (according to BTCR[4])
20. drive FCi* and SIZ

The Module

21. detect LAEN*, FCi*
22. drive LA[31:8] with DMA start address

First read:

The VIC068A

23. assert PAS*, DS*

The Module

24. read data from memory

25. assert DSACKi*

The Device

26. wait until ready to transfer data

27. assert ASIZ0

The VIC068A

28. start and complete DMAAT0 delay

29. deassert DS*

The Device

30. detect DS* High

31. capture data

The VIC068A

32. update address and burst counters

The Device

33. deassert ASIZ0

Second and Subsequent Transfers:

The VIC068A

34. detect ASIZ0 High

35. wait for DS* High time

36. assert PAS*, DS*

The Module

37. read data from memory

38. assert DSACKi*

The Device

39. wait until ready to accept data

40. assert ASIZ0

The VIC068A

41. start and complete DMAAT1 delay

42. deassert DS*

The Device

43. detect DS* High

44. capture data

The VIC068A

45. update address and burst counters

The Device

46. deassert ASIZ0

47. continue if needed (go to 34)

For multiple burst transfers, the VIC068A will

48. toggle BLT* if at a page boundary crossing

49. release local bus and deassert LBR*

50. wait interleave period (in BTCR[3-0])

51. assert LBR*

52. continue if needed (go to 16)

Otherwise, if the DMA is complete, the VIC068A will

53. deassert BLT*

54. release local bus and deassert LBR*

11.2.2 DMA Memory Write

For a DMA memory write operation, the VIC068A acts as a DMA controller for a device-to-memory transfer. In this mode, the device must detect that a memory write is occur-

ring and make the data available on the local data signals. Once this data is available, the device asserts ASIZ0 signal. The VIC068A, detecting ASIZ0 asserted, asserts DS*, enabling the module to write the data.

Once the DMA operation is configured and BTCR[7] is set, the following events occur for a DMA memory write.

Initialization:

The Module

1. place DMA target address on LA[31:0]
2. drive R/W* Low for a write operation
3. assert PAS*
4. place memory address on LD[31:0]
5. assert DS*
6. assert BLT*
7. capture remainder of LD[31:8] as local address

The VIC068A

8. detect BTCR[7] = 0 and BLT* asserted.
9. capture LD[7:0] into internal address counter
10. assert BLT*
11. assert LBR*
12. assert DSACK1/0*
13. wait for LBG*

The Module

14. complete local cycle, assert LBG*
15. release BLT* (it remains asserted, however, because the VIC068A is still asserting it)

The VIC068A

16. Detect LBG*, wait for local cycle to complete

17. start PAS*/DS* minimum High delays

18. drive LA[7:0] with DMA start address

19. drive R/W* Low (according to BTCR[4])

20. drive FCi*, and SIZ

The Module

21. detect LAEN*, FCi*

22. drive LA[31:8] with DMA start address

First write:

The VIC068A

23. assert PAS*

The Device

24. access data to be written to local memory

25. assert ASIZ0

The VIC068A

26. detect ASIZ0 asserted

27. assert DS*

The Module

28. write data to memory

29. assert DSACKi*

The Device

30. detect DS* asserted

31. deassert ASIZ0

The VIC068A

32. detect ASIZ0 deasserted, DSACKi* asserted

33. start and complete DMAAT0 delay
34. deassert DS*
35. update address and burst counter

The Device

36. detect DS* deasserted

Second and subsequent cycles:

The Device

37. access data to be written to local memory
38. assert ASIZ0

The VIC068A

39. detect ASIZ0 asserted
40. assert DS*

The Module

41. write data to memory
42. assert DSACKi*

The Device

43. detect DS* asserted
44. deassert ASIZ0

The VIC068A

45. detect ASIZ0 deasserted, DSACKi* asserted
46. start and complete DMAAT0 delay
47. deassert DS*
48. update address and burst counters

The Device

49. detect DS* deasserted
50. continue if needed (go to 37)

For multiple burst transfers, the VIC068A will

51. toggle BLT*, if at a page boundary crossing
52. release local bus and deassert LBR*
53. wait interleave period (in BTCR[3-0])
54. assert LBR*
55. continue if needed (go to 16)

Otherwise, if the DMA is complete, the VIC068A will

56. deassert BLT*
57. release local bus and deassert LBR*

11.3 Deadlock Considerations

When transferring data using module-based DMA transfer, care should be taken to avoid possible deadlock situations. For example, if the DMA mechanism was used to transfer data from memory to the VSBbus, and the VSBbus attempted access to memory at the same time, a deadlock situation would exist. This deadlock could be resolved through external local timeout methods. Deadlock could, in this example, also be avoided by not asserting LBG* until the VSB access is complete.

11.4 Decode Considerations

For 68K-based systems, a convenient way to decode a module-based DMA transfer is to use the MOVES instruction. This allows a user-defined Function Code (FC) space to decode special accesses such as a module-based DMA transfer. See a 68K user's manual for further details on the MOVES instruction.

12

Miscellaneous Features

This chapter describes additional miscellaneous features of the VIC068A.

12.1 Resetting the VIC068A

The VIC068A is reset by any of three distinct reset conditions. These reset conditions are initiated by asserting various inputs or, in the case of a system reset, writing a VIC068A register. Since it is possible for more than one reset condition to be present simultaneously, the resets have the following priority:

1. global reset
2. internal reset
3. system reset

12.1.1 Internal Reset

The internal reset is the most common and easiest to implement of the VIC068A resets. This resets all the VIC068A internal circuitry and selected register bit fields. This reset is typically used as a push-button reset after power-up.

The internal reset is initiated by asserting the IRESET* signal for a minimum of 20 ns. Immediately after the assertion of IRESET*, the VIC068A asserts the LBR* signal. The VIC068A then waits 1 μs for the assertion of LBG*. If LBG* is asserted within the 1-μs timeout, the VIC068A asserts HALT* and RESET* immediately. If LBG* is not asserted within this timeout period, the VIC068A asserts HALT* and RESET* as if LBG* had been received. The VIC068A attempts this reset handshaking sequence to provide an orderly reset of local resources.

After the VIC068A asserts HALT* and RESET*, the VIC068A deasserts LBR*, places all three-state outputs to a high-Z state, and begins a 200-ms timeout period. If IRESET* is

still asserted after this timeout expires, the VIC068A begins an additional 200-ms timeout. The reset does not complete until the 200-ms timeout expires after IRESET* is deasserted.

If the VIC068A is the VMEbus system controller, the VIC068A also asserts SYSRESET* with the RESET* and HALT* signals.

After the conclusion of an internal reset, the HALT* and RESET* signals (and SYSRESET* if VMEbus is system controller) are deasserted and all outputs are brought to their quiescent states.

12.1.2 Global Reset

The global reset provides the most complete reset to the VIC068A. This resets all VIC068A internal circuitry and all register bit fields to predefined states. The global reset is typically issued as a "power-up" reset.

The global reset is initiated by asserting the IPL0* signal after the IRESET* signal is asserted. It is important that IPL0* be asserted after the IRESET* signal. This allows IRESET* to reverse the direction of the IPL0* line to an input. For a reliable global reset, the IPL0* signal should be asserted for a minimum of 50 ns.

Two notes should be observed when using global resets:

- SYSCLK is not driven during the global reset
- the BGiOUT* daisy-chain is disconnected during the global reset

The characteristics of the global reset are present only while IPL0* is asserted. This implies that the aforementioned behaviors are present only during the time IPL0* is asserted. Before the IPL0* signal is asserted, the VIC068A is in an internal reset state. After IPL0* is asserted, the VIC068A is in an internal reset state except that the registers have already had their bit fields set to their default global reset state.

If a global reset is needed when the VIC068A is configured as VMEbus system controller, assert IPL0* for a minimum time. This causes SYSCLK to be disabled only for the minimum time IPL0* is asserted. The same is true for the BGiOUT* daisy-chain behavior.

The global reset mechanism is identical to the internal reset (LBG* and HALT* or RESET* timeouts, etc.) with the following two exceptions:

- all register bit fields are set to their default states.

- if the IPL0* signal is deasserted after the IRESET* signal is deasserted, the reset concludes immediately instead of waiting for the 200-ms timeout to expire

SYSRESET* is asserted during a global reset when configured as VMEbus system controller, as it is during an internal reset.

12.1.3 System Reset

The system reset is a VMEbus-defined reset that is signaled by the assertion of the VMEbus signal SYSRESET*. The system reset is typically issued from another module to reset an entire VMEbus system. The system reset resets all the VIC068A internal circuitry and selected register bit fields. The VIC068A may both issue and receive a system reset. If the VIC068A issues a system reset, it is also resets itself.

There are two ways to issue a system reset:

- write the SRCR with the value of $F0.
- implement an internal or global reset while configured as a VMEbus system controller

The VIC068A receives a system reset by having its SYSRESET* signal asserted. When this occurs, the VIC068A begins the same procedures as if an internal reset had been performed.

The system reset is identical in function to the internal reset with the following exceptions:

- the effect on certain VIC068A register bit fields are different
- if SYSRESET* is asserted beyond the 200-ms timeout described by the internal reset, the reset completes immediately after SYSRESET* is deasserted

12.1.4 Power-On Reset

To reliably reset the VIC068A at power-on, a global reset must be performed. In addition, the VIC068A must be in a "stable" operating environment at the initiation of the reset. A stable operating environment is defined by the following conditions:

- V_{CC} has reached 5V dc
- The CLK64M signal is within the sxpecified operating range (see Chapter 24)
- All inputs are in a negated state (excluding address/data buses)

If the VIC068A is not in a stable environment when IRESET* is asserted, the VIC068A may not reset reliably. If the power-on reset is performed when the environment is not stabel, re-issue the global reset when the environement is stable.

Because a global reset causes certain VMEbus system controller operations to be suspended (see section 12.1.2), the power-on reset circuitry must be designed to assert the IPL0* signal for a "minimum" time while the VIC068A is the VMEbus system controller. Minimum is defined to be greater than the required 50 ns but less than the time the disabled features are required to be enabled for a particular application.

12.2 The Local Bus Timeout Timer

The VIC068A contains a local bus timeout timer that may be used to time certain local timeout conditions for the local bus. The timer begins on the assertion of DS*. If the timer period expires without an assertion of either the DSACKi* or LBERR* signals, the VIC068A asserts LBERR* and sets BESR[3].

The local timeout timer may be configured to include or not include time waiting for VMEbus acquisition. If enabled to include VMEbus acquisition time, BESR[0] is set when a timeout occurs. Once VMEbus mastership is obtained, the local timeout timer is reset and does not expire. At this point a VMEbus timeout timer (if one exists for the particular VMEbus system) is used to indicate timeout conditions. If the VIC068A is used as the system controller, this VMEbus timeout timer is located internal to the VIC068A.

If the local timeout timer is configured not to include VMEbus acquisition time, the timer resets at the assertion of BRi* and does not expire.

The local bus timeout periods are configurable for 4, 16, 32, 64, 128, 256, and 512 µs.

12.3 The DRAM Refresh Controller

The VIC068A contains a DRAM refresh controller. When enabled in the ARCR, the VIC068A increments a DRAM refresh counter every 15 µs. When the count of this counter reaches 4, the VIC068A requests the local bus by asserting LBR*. After the local bus is granted, the VIC068A performs a CAS-before-RAS refresh (that is, DS*-before-PAS*). The VIC068A increments the refresh counter until the local bus is granted so the actual number of cycles that are performed would equal the number in the refresh counter when the local bus is granted to the VIC068A. If a VMEbus slave request is pending, the VIC068A gives priority to the slave transfer and the DRAM refresh is performed after the slave transfer is complete, but within the same assertion of LBR*.

When the VIC068A receives the local bus, the VIC068A drives the FC2/1 signals with the DRAM refresh function codes. It then asserts PAS* and DS* according to the timing configured in the LBTR. Insure that the minimum High time specified for PAS* is greater then the minimum precharge time for the particular DRAMs being used. The VIC068A drives the LA[7:0] signals High and asserts LAEN when performing DRAM refresh. The SIZ1/0 signals are driven with a 32-bit code.

The refresh counter is a modulo-64 counter. If it reaches 64 without having the VIC068A obtain the local bus, DRAM refresh cycles may be lost. The VIC068A local bus timer may be used to insure that DRAM refresh cycles are not lost due to excessive local bus latency.

12.4 Rescinding Outputs

The VIC068A contains selected output signals that utilize a rescinding feature. A rescinding output is a three-state output driver that first drives the output High before three-stating. This is necessary for proper functionality of high-speed buses. The VIC068A rescinding outputs are as follows:

VMEbus	*Local bus*
DTACK*	PAS*
AS*	DS*
LWORD*	LBERR*
WRITE*	R/W*
BERR*	SIZ1/0*
IACK*	FC2/1*
DS1/0*	
BBSY*	

12.5 Turbo Mode

By setting ICR[1], it is possible to reduce certain delays within the VIC068A. When set, the VIC068A reduces the following by 1 CLK64M period:

- VMEbus address set-up time
- VMEbus data set-up time
- DTACK* asserted time for slave block transfers

Because VMEbus times may be violated with this mode enabled, this mode should be used with caution.

12.6 Metastability Delays

Because the VMEbus is an asynchronous bus (the local bus may also be), the VIC068A must insure that its synchronous logic is protected from possible metastable conditions. The VIC068A accomplishes this using a traditional double-sampling latch. The VIC068A samples an input, then a specified settling time later samples it again. This allows for any metastability that may have occurred to settle to a stable value. This settling time is programmable to 3T or 4T in the ICR.

13

VIC068A Register Map and Descriptions

This chapter describes the VIC068A internal configuration registers. These registers enable and disable various features of the VIC068A. (Refer to the specific sections of this guide for details on specific features.)

Table 13–1 provides information on the various reset states of the VIC068A registers. The following notes should be observed regarding this table:

- An asterisk (*) indicates a bit that is not affected by the particular reset.
- An "X" indicates a bit that is affected by that state of a particular VIC068A pin.
- ICR5 is the VIC068A-version register. Its contents will vary depending on the revision of the device being used.

Unless otherwise specified, the reset status is given in this chapter by the values located in the parentheses by each bit field. The (X/X/X) format indicates the Global/Internal/System reset state of each bit or bit field.

For compatibility with the VIC64, it is recommended that all reserved register bits be written with a 0.

Table 13–1. VIC068A Register Values After Reset Operations

Address (hex)	Name	Description	Global Reset	Internal Reset	System Reset
03	VIICR	VMEbus Interrupter Interrupt Control Register	11111000	11111***	11111***
07–1F	CICR1–7	VMEbus Interrupt Control Registers 1–7	11111000	11111***	11111***
23	DMASR	DMA Status Register	11111000	11111***	11111***
27–3F	LICR1–7	Local Interrupt Control Registers 1–7	1000X000	1***X***	1***X***
43	ICGSICR	ICGS Interrupt Control Register	11111000	11111***	11111***
47	ICMSICR	ICMS Interrupt Control Register	11111000	11111***	11111***
4B	EGICR	Error Group Interrupt Control Register	1111X000	1111X***	1111X***
4F	ICGSVBR	ICGS Vector Base Register	000011XX	000011XX	000011XX
53	ICMSVBR	ICMS Vector Base Register	000011XX	000011XX	000011XX
57	LIVBR	Local Interrupt Vector Base Register	00001XXX	00001XXX	00001XXX
5B	EGIVBR	Error Group Interrupt Vector Base Register	00001XXX	00001XXX	00001XXX
5F	ICSR	Interprocessor Communications Switch Register	00000000	****0000	00000000
63–73	ICR0–4	Interprocessor Communications Registers 0–4	00000000	00000000	00000000

Table 13–1. VIC068A Register Values After Reset Operations (continued)

Address (hex)	Name	Description	Global Reset	Internal Reset	System Reset
77	ICR5	Interprocessor Communications Register 5	Version	Version	Version
7B	ICR6	Interprocessor Communications Register 6	X11111XX	X1111111	X1111110
7F	ICR7	Interprocessor Communications Register 7	00X00000	*0XX****	00X00000
83	VIRSR	VMEbus Interrupt Request Status Register	00000000	*******0	00000000
87–9F	VIVBR1–7	VMEbus Interrupt Vector Base Registers 1–7	00001111	********	00001111
A3	TTR	Transfer Timeout Register	01101000	01101000	01101000
A7	LBTR	Local Bus Timing Register	00000000	********	********
AB	BTDR	Block Transfer Definition Register	11110000	11110000	11110000
AF	ICR	Interface Configuration Register	0000000X	0000000X	0000000X
B3	ARCR	Arbiter/Requester Configuration Register	01100000	011*0000	011*0000
B7	AMSR	Address Modifier Source Register	00000000	00000000	00000000
BB	BESR	Bus Error Status Register	X0000000	X0000000	X0000000
BF	DMASR	DMA Status Register	00000000	00000000	00000000
C3	SS0CR0	Slave Select 0 Control Register 0	00000000	00******	00******
C7	SS0CR1	Slave Select 0 Control Register 1	00000000	********	********
CB	SS1CR0	Slave Select 1 Control Register 0	00000000	00******	00******
CF	SS1CR1	Slave Select 1 Control Register 1	00000000	********	********
D3	RCR	Release Control Register	00000000	00000000	00000000
D7	BTCR	Block Transfer Control Register	00000000	00000000	00000000
DB	CTLR0	Block Transfer Length Register 0	00000000	00000000	00000000
DF	BTLR1	Block Transfer Length Register 1	00000000	00000000	00000000
E3	SRR	System Reset Register	11111111	11111111	11111111
EB–FF		Reserved Locations	11111111	11111111	11111111

VMEbus Interrupter Interrupt Control Register

Name: VIICR

Address: $03

Description: Provides enabling and IPL level encoding for the local interrupt issued when a VMEbus interrupt is acknowledged.

Bits 2–0: (0/*/*) IPL value. Value is inverted and driven onto the IPL lines when an interrupt is acknowledged.

Bits 6–3: (1/1/1) Undefined/Reserved. Bits will read as 1's.

Bit 7: (1/1/1) VMEbus interrupt mask. When clear, the VIC068A signals a local interrupt at the acknowledgement of a previously issued VMEbus interrupt. When set, the VIC068A will not issue a local interrupt.

VMEbus Interrupt Control Registers 1–7

Name: VICR1–7

Addresses: $07, $0B, $0F, $13, $17, $1B, $1F
For interrupt: 1 2 3 4 5 6 7

Description: Provides enabling of the VIC068A as VMEbus interrupt handler for any or all of the VMEbus interrupts. Seven registers exist to provide unique masking and IPL values for the seven VMEbus interrupts.

Bits 2–0: (0/*/*) IPL value. Value is inverted and driven onto the IPL signals when a VMEbus interrupt is acknowledged.

Bits 6–3: (1/1/1) Undefined/Reserved. Bits will read as 1's.

Bit 7: (1/1/1) VMEbus interrupt mask. When clear, the VIC068A acts as a VMEbus interrupt handler by signaling a local interrupt at the specified IPL level. When set, the VIC068A does not handle the VMEbus interrupt and no local interrupt is issued.

DMA Status Interrupt Control Register

Name: DMASICR

Address: $23

Description: Provides enabling and IPL-level encoding for the DMA-complete interrupt issued by the VIC068A when any VIC068A local DMA operation completes (successfully or unsuccessfully).

Bits 2–0: (0/*/*) IPL value. Value is inverted and driven onto the IPL lines when interrupt is acknowledged.

Bits 6–3: (1/1/1) Undefined/Reserved. Bits will read as 1's.

Bit 7: (1/1/1) DMA status interrupt mask. When clear, the VIC068A signals a local interrupt at the completion of any VIC068A local DMA operation. When set, the VIC068A will not issue a local interrupt.

Local Interrupt Control Registers 1–7

Name:	LICR1–7
Address: For LIRQ:	$27, $2B, $2F, $33, $37, $3B, $3F 1 2 3 4 5 6 7
Description:	Provides enabling, IPL level, and control of local interrupts 1–7 (LIRQ1–7*).
Bits 2–0: (0/*/*)	IPL value. Value is inverted and driven onto the IPL lines when a local interrupt is presented on the LIRQ1–7* signals and bit 7 of this register is clear (enabled).
Bit 3: (X/X/X)	LIRQ1–7* voltage state. A cleared bit indicates the LIRQ1–7* signal is asserted at the VIC068A.
Bit 4: (0/*/*)	Autovector enable. When set, the VIC068A will supply the interrupt status/ID vector for the local interrupt acknowledge cycle. When cleared, the VIC068A will assert the LIACK0* signal to indicate an 68K "autovector" condition or that the interrupting source should provide the Status/ID vector to the processor.
Bit 5: (0/*/*)	Edge/level enable. When cleared, the VIC068A responds to the LIRQ1–7* as a level-sensitive interrupt. When set, the VIC068A responds to LIRQ1–7* as an edge-sensitive interrupt.
Bit 6: 0/*/*)	Polarity set. When set, the VIC068A responds to interrupts as active High if bit 5 is set (level sensitive) or on a rising edge if bit 5 is cleared (edge sensitive). When cleared, the VIC068A responds to active Low or falling edges.
Bit 7: (1/1/1)	Local interrupt mask. When clear, the VIC068A is enabled to handle the corresponding local interrupt asserted on the LIRQ1–7* signals.

ICGS Interrupt Control Register

Name:	ICGSICR
Address:	$43
Description:	Provides enabling and IPL encoding for the four global switch interrupts.
Bits 2–0: (0/*/*)	IPL Value. Value is inverted and driven onto the IPL signals when a global switch is acknowledged.
Bit 3: (1/1/1)	Undefined/Reserved. Bit will read as a 1.
Bit 4: (1/1/1)	ICGS0 mask. When clear, the VIC068A will issue and handle a local interrupt when global switch 0 is set.
Bit 5: (1/1/1)	ICGS1 mask. When clear, the VIC068A will issue and handle a local interrupt when global switch 1 is set.
Bit 6: (1/1/1)	ICGS2 mask. When clear, the VIC068A will issue and handle a local interrupt when global switch 2 is set.
Bit 7: (1/1/1)	ICGS3 mask. When clear, the VIC068A will issue and handle a local interrupt when global switch 3 is set.

ICMS Interrupt Control Register

Name:	ICMSICR
Address:	$47
Description:	Provides enabling and IPL encoding for the four module switch interrupts.
Bits 2–0: (0/*/*)	IPL Value. Value is inverted and driven onto the IPL signals when a module switch is acknowledged.
Bit 3: (1/1/1)	Undefined/Reserved. Bit will read as a 1.
Bit 4: (1/1/1)	ICMS0 mask. When clear, the VIC068A will issue and handle a local interrupt when module switch 0 is set.
Bit 5: (1/1/1)	ICMS1 mask. When clear, the VIC068A will issue and handle a local interrupt when module switch 1 is set.
Bit 6: (1/1/1)	ICMS2 mask. When clear, the VIC068A will issue and handle a local interrupt when module switch 2 is set.
Bit 7: (1/1/1)	ICMS3 mask. When clear, the VIC068A will issue and handle a local interrupt when module switch 3 is set.

Error-Group Interrupt Control Register

Name:	EGICR
Address:	$4B
Description:	Provides enabling and IPL encoding for the error group interrupts.
Bits 2–0: (0/*/*)	IPL Value. Value is inverted and driven onto the IPL signals when an error group interrupt is acknowledged.
Bit 3: (X/X/X)	SYSFAIL* asserted. This bit is set whenever SYSFAIL* is detected asserted.
Bit 4: (1/1/1)	SYSFAIL* interrupt mask. When clear, the VIC068A generates a local interrupt when SYSFAIL* is asserted.
Bit 5: (1/1/1)	Arbitration timeout interrupt mask. When clear, the VIC068A generates a local interrupt when an arbitration timeout has occurred.
Bit 6: (1/1/1)	Write post fail interrupt mask. When clear, the VIC068A generates a local interrupt when a write post operation has failed due to a bus error. For master write posts, an assertion of BERR* will trigger an interrupt. For slave write posts, an assertion of LBERR* will trigger an interrupt.
Bit 7: (1/1/1)	AC Fail interrupt mask. When clear, the VIC068A generates a local interrupt when ACFAIL* is detected as asserted.

ICGS Interrupt Vector Base Register

Name:	ICGSIVBR
Address:	$4F
Description:	Provides the status/ID vector for the global switch interrupts. This register must be written after any VIC068A reset to enable identification encoding for bits 1–0. This register is reset to $F0 during any VIC068A reset.
Bits 1–0: (X/X/X)	Global switch number (read-only). This value indicates which global switch is pending during a global switch interrupt acknowledge cycle. These bits are used with bits 7–2 to provide a unique status/ID vector for each global switch. The numeric value of this field indicates the switch number. These bits are valid only during the interrupt acknowledge cycle.

Bits 7–2: Status/ID. These bits are user-definable and are used with bits 1–0 to provide a unique global switch interrupt status/ID vector.

ICMS Interrupt Vector Base Register

Name: ICMSIVBR

Address: $53

Description: Provides the status/ID vector for the module switch interrupts. This register must be written after any VIC068A reset to enable identification encoding for bits 1–0. This register is reset to $F0 during any VIC068A reset.

Bits 1–0: (X/X/X) Module switch number (read-only). This value indicates which module switch is pending during a module switch interrupt acknowledge cycle. These bits are used with bits 7–2 to provide a unique status/ID vector for each module switch. The numeric value of this field indicates the switch number. These bits are valid only during the interrupt acknowledge cycle.

Bits 7–2: Status/ID. These bits are user-definable and are used with bits 1–0 to provide a unique module switch interrupt status/ID vector.

Local Interrupt Vector Base Register

Name: LIVBR

Address: $57

Description: Provides the status/ID vector for the local interrupts. This register must be written after any VIC068A reset to enable identification encoding for bits 2–0. This register is reset to $F0 during any VIC068A reset.

Bits 2–0: (X/X/X) Local Interrupt number (read-only). This value indicates which local interrupt is pending during a local interrupt acknowledge cycle. These bits are used with bits 7–3 to provide a unique status/ID vector for each local interrupt. The numeric value of this field indicates the local interrupt number. These bits are valid only during the interrupt acknowledge cycle.

Bits 7–3: Status/ID. These bits are user-definable and are used with bits 1–0 to provide a unique local interrupt status/ID vector.

Error Group Interrupt Vector Base Register

Name: EGIVBR

Address: $5B

Description: Provides the status/ID vector for the error group interrupts. This register must be written after any VIC068A reset to enable identification encoding for bits 2–0. This register is reset to $F0 during any VIC068A reset.

Bits 2–0: (X/X/X) Error/Status Group Interrupt number (read-only). This value indicates which group interrupt is pending during the interrupt acknowledge cycle. These bits are used with bits 7–3 to provide a unique status/ID vector for each error group interrupt. These bits are valid only during the interrupt acknowledge cycle.

Bits 2–0	*Error/Status Interrupt*
0 0 0	ACFAIL* asserted
0 0 1	Write post failed
0 1 0	Arbitration timeout
0 1 1	SYSFAIL* asserted
1 0 0	VMEbus Interrupter interrupt acknowledge
1 0 1	DMA complete

Bits 7–3: Status/ID. These bits are user-definable and are used with bits 1–0 to provide a unique interrupt status/ID vector.

Interprocessor Communications Switch Register

Name: ICFSR

Address: $5F

Description: Provides setting, clearing, and monitoring of the interprocessor switch interrupts via the local bus. If the switch interrupts are enabled, setting these bits (more precisely, a clear-to-set transition) causes a local interrupt to occur in the same way as if the switch was set over the VMEbus.

Bits 3–0: (0/0/0) Module switches. Bits 0, 1, 2, and 3 correspond to ICMSs 0, 1, 2, and 3 respectively.

Bits 7–4: (0/*/0) Global switches. Bits 4, 5, 6, and 7 correspond to ICGSs 0, 1, 2, and 3 respectively.

Interprocessor Communication Registers 0–4

Name: ICR0–4

Addresses: $63, $67, $6B, $6F, $73
For registers: 0 1 2 3 4

Description: These are general-purpose read/write registers that can be accessed from either the local bus or the VMEbus. The addresses listed above are the local addresses. See Chapter 8 for details on accessing these registers from the VMEbus.

Bits 7–0: (0/0/0) Data field.

Interprocesor Communication Register 5

Name: ICR5

Address: $77

Description: This register provides the VIC068A version/revision number. The first VIC068A device contains a value of $F1. Contact your local Cypress Semiconductor sales office for current VIC068A revision status. The address listed above is the local address. See Chapter 8 for details on accessing this register from the VMEbus.

Bits 7–0: VIC068A version/revision (read-only).

Interprocessor Communication Register 6

Name: ICR6

Address: $7B

Description: This register provides local or remote reset and HALT*. The address listed above is the local address. See Chapter 8 for details on accessing this register from the VMEbus.

Bits 1–0: Reset/HALT* status (read-only from VMEbus). These bits provide reset/HALT* status of the VIC068A and local resources according to the following table:

Bits 1–0	Reset/HALT* Status.
0 1	HALT* has been asserted longer than 6 µs by a source other then the VIC068A. These bits may both be reset by the local CPU to indicate local resources are running and operational.
1 0	The VIC068A has performed a local reset function and the VIC068A is not the system controller. These bits may both be reset by the local CPU to indicate local resources are running and operational.
1 1	Indicates that the CPU has just been released from a system reset.
0 0	Local resources are running and operational. This pattern must be written by the local CPU after a reset condition to indicate that local resources are running and operational.

Bits 5–2: (1/1/1) Undefined/Reserved. Bits will read as 1's.

Bit 6: (1/1/1) IRESET* and HALT* status (read-only from VMEbus). This bit is set upon assertion of IRESET*, and/or HALT*. It is set whether HALT* is asserted by external sources or by the VIC068A. SYSFAIL* is asserted when this bit is set if the SYSFAIL* mask bit (ICR7, bit 7) is cleared.

Bit 7: (X/X/X) IRESET* status (read-only). On a VMEbus read, this bit indicates that the VIC068A is in a reset state. On a local bus read, this bit is set whenever ACFAIL* is asserted.

Interprocessor Communication Register 7

Name: ICR7

Address: $7F

Description: This register provides semaphores to the five general-purpose interprocessor communication registers (ICR4–0). The remaining bits indicate VMEbus master status, generate HALT* and RESET*, and mask SYSRESET*. The address listed above is the local address. See Chapter 8 for details on accessing this register from the VMEbus.

Bits 4–0: (0/*/0) ICR4–0 semaphores. These bits provide semaphores to the five interprocessor communication registers ICR4–0 respectively. Each bit is set when the corresponding ICR is written. These bits can be read or written from the local bus or the VMEbus.

Bit 5: (X/X/X) VMEbus master status (read-only). This bit is set whenever the VIC068A is the VMEbus master, and the VIC068A is asserting AS*. This bit is not set when the VIC068A is VMEbus master to an idle bus in ROR and BCAP release modes. Bit 7 of the BESR may be used to indicate that the VIC068A is VMEbus master when AS* is not asserted.

Bit 6: (0/0/0) HALT* and RESET* control. This bit may be used to assert the HALT* and RESET* pins via software. Whenever this bit is set, the VIC068A asserts HALT* and RESET* until this bit is cleared or any reset occurs.

Bit 7: (0/*/0) SYSFAIL* mask. When set, the VIC068A is prohibited from asserting SYSFAIL* in response to bit 6 of ICR6 being set (which, by default, is set after any reset).

VMEbus Interrupt Request/Status Register

Name: VIRSR

Address: $83

Description: This register provides status and control of the VMEbus interrupts 7–1.

Bit 0: (0/0/0) Register enable/disable. This bit provides enabling and disabling for the remainder of this register.

Bits 7–1: (0/0/0) VMEbus interrupt switches. Setting any of these bits asserts the VMEbus IRQi* signals corresponding to the bit positions, if bit 0 is set during the write. These bits are cleared by setting the appropriate bit and clearing bit 0.

VMEbus Interrupt Vector Base Registers 1–7

Name: VIVBR

Address: $87, $8B, $8F, $93, $97, $9B, $9F
for IRQ: 1 2 3 4 5 6 7

Description: Provides the status/ID vector for the VMEbus interrupts.

Bits 7–0: Status/ID Vector. These bits provide the status/ID vector for VMEbus interrupt acknowledge cycles. Address $87 corresponds to IRQ1*. These bits are set to a value of $0F for global and system resets and are unchanged by internal resets.

Transfer Timeout Register

Name: TTR

Address: $A3

Description: Provides control of the local and VMEbus timeout timers.

Bit 0: (0/0/0) Include VMEbus acquisition. When set, the local bus timer will include waiting for VMEbus acquisition. When clear, the local bus timer will stop and reset when the VMEbus is requested.

Bit 1: (0/0/0) Arbitration timeout. When set, the VIC068A as VMEbus arbiter has detected a VMEbus arbitration timeout. This is only used when configured as the VMEbus system controller (SCON* asserted).

Bits 4–2: Local bus timeout period. Defines the local bus timeout.

Bit 4	*Bit 3*	*Bit 2*	*Local Bus Timeout (µs)*
0	0	0	4
0	0	1	16
0	1	0	32 (default)
0	1	1	64
1	0	0	128
1	0	1	256
1	1	0	512
1	1	1	Infinite (timer disabled)

Bits 7–5: VMEbus timeout period. Defines the VMEbus timeout.

Bit 7	*Bit 6*	*Bit 5*	*VMEbus Timeout (µs)*
0	0	0	4
0	0	1	16
0	1	0	32 (default)
0	1	1	64
1	0	0	128
1	0	1	256
1	1	0	512
1	1	1	Infinite (timer disabled)

Local Bus Timing Register

Name: LBTR

Address: $A7

Description: Provides timing control for PAS* and DS* signals when the VIC068A is local bus master. In the following descriptions, *n* is the binary value specified in the bit fields, and *T* is one CLK64M clock period. Clock latency may add one additional clock period to these times.

Bits 3–0: (0/*/*) Minimum PAS* asserted time. This field specifies the *minimum* asserted time for the PAS* signal whenever the VIC068A is the local bus master. The time is specified by (n + 2)T. The actual asserted time depends on a number of factors including local and VMEbus acknowledge timing.

Bit 4: (0/*/*) Minimum DS* deasserted time. This field specifies the *minimum* deasserted time for the DS* signal whenever the VIC068A is the local bus master. A time of 1T is selected when this bit is clear; 2T is selected when this bit is set.

Bits 7–5: (0/*/*) Minimum PAS* deasserted time. This field specifies the *minimum* deasserted time for the PAS* signal whenever the VIC068A is the local bus master. The time is specified by (n + 1)T.

Block Transfer Definition Register

Name:	BTDR
Address:	$AB
Description:	Configures master block transfers (both MOVEM and block transfers with local DMA) for boundary crossings, dual-path, and user-defined address modifiers. See Chapter 10 for more details on implementing these features.
Bit 0: (0/0/0)	Dual-path enable. When set, the VIC068A is enabled with the dual-path feature during master block transfers with local DMA. External logic is required when this option is enabled.
Bit 1: (0/0/0)	AMSR Enable. When set, the VIC068A will issue the AM codes based in the address modifier source register for block transfers. This bit effects the AM codes for block transfers only.
Bit 2: (0/0/0)	When this bit is set, it enables local address 256-byte boundary crossings during DMA block transfer operations. External logic is required to increment latched address lines when this option is enabled.
Bit 3: (0/0/0)	When this bit is set, it enables VMEbus address 256-byte boundary crossings during DMA block transfer operations. External logic is required to increment latched address lines when this option is enabled.
Bit 7–4: (1/1/1)	Undefined/Reserved. Bits will be read as 1's.

Interface Configuration Register

Name: ICR

Address: $AF

Description: Controls various features of the VIC068A including RMCs, deadlock signaling, metastablility delays, and the "turbo" feature.

Bit 0: (X/X/X) SCON* value (read-only). Reads the value of the SCON* pin. When set, the VIC068A is not the VMEbus system controller. When clear, the VIC068A is the VMEbus system controller.

Bit 1: (0/0/0) Turbo enable. When set, the VIC068A accelerates VMEbus transfers by reducing selected timings by one CLK64M clock period. VMEbus protocols may be violated when the turbo mode is enabled (see section 12.5).

Bit 2: (0/0/0) Metastability interval. When set, the VIC068A adds one additional CLK64M clock period of metastability delay on asynchronous inputs (from 3 CLK64M periods to 4).

Bits 4, 3: (0/0/0) Deadlock signaling. These bits configure deadlock signaling. Bit 4 is used to enable the assertion of HALT* and LBERR* in addition to the DEDLK* signal in deadlock situations. If bit 4 is enabled, bit 3 may be used to prevent the assertion of HALT* for RMC deadlocks.

Bit 4	*Bit 3*	*Deadlock Signaling*
0	X	DEDLK* only (default)
1	0	HALT*, LBERR*, DEDLK*
1	1	HALT*, LBERR*, DEDLK* (HALT* is not asserted for RMC cycles)

Bit 5: (0/0/0) RMC control bit 1. When set, the VIC068A will request the VMEbus whenever the RMC* is asserted independent of the MWB* signal.

Bit 6: (0/0/0) RMC control bit 2. When set, the VMEbus AS* is stretched when RMC* is asserted for VMEbus transfers.

Bit 7: (0/0/0) RMC control bit 3. When set, the VIC068A qualifies the RMC control bits 1 and 2 with the SIZ1/0 signals. If RMC control bits 1 or 2 are set and the first cycle of the RMC transfer is of byte size, the "set" behaviors are not implemented.

Arbiter/Requester Configuration Register

Name: ARCR

Address: $B3

Description: This register provides configuration of the fairness timeout and DRAM refresh features. The VMEbus request level is also configured from this register.

Bits 3–0: (0/0/0) Fairness timer enable. The VMEbus fair requester is enabled in this bit field according to the following table:

Bits 3–0	*Timeout Period/Mode*
$0	Fairness disabled (default)
$F	Timeout disabled
All other patterns	2 µs times number

Bit 4: (0/*/*): DRAM refresh. When set, the VIC068A will perform CAS-before-RAS (DS* before PAS*) refresh functions.

Bits 6, 5: VMEbus request level. The VMEbus request level is set according to the following table:

Bit 6	*Bit 5*	*VMEbus Request Level*
0	0	BR0
0	1	BR1
1	0	BR2
1	1	BR3 (default)

Bit 7: (0/0/0) Arbitration mode. When set, the VIC068A performs priority VMEbus arbitration. When clear, the VIC068A performs round-robin arbitration. This bit is only relevant when the VIC068A is configured as the VMEbus system controller (SCON* asserted).

Address Modifier Source Register

Name: AMSR

Address: $B7

Description: This register provides the user-definable address modifiers (AM codes) that can be sourced by the VIC068A for VMEbus master cycles, or used in validating AM codes during VMEbus slave cycles.

Bits 5–0: (0/0/0) Address modifier code. The AM code that is issued during master cycles or used for qualifying slave cycles. This register is used only when enabled for user-defined AM codes. Otherwise, standard VMEbus AM codes are used.

Bit 6: (0/0/0) AM5–3 qualification. When set, the VIC068A uses bits 5–3 in qualifying for slave accesses in addition to the address space size information defined by bits 3 and 2 of the SSiCR0s. This bit is overridden if bits 3 and 2 of the SSiCR0s are both clear.

Bit 7: (0/0/0) AM2–0 generation. When set, the VIC068A issues the AM2–0 codes based on the FC2/1 signals. AM5–3 will be issued from bits 5–3 of this register. This bit is overridden if the VIC068A is configured to issue all AM bits from this register (i.e., ASIZ1 = ASIZ0 = Low).

Bus Error Status Register

Name:	BESR
Address:	$BB
Description:	This register provides BERR/LBERR*, self-access, VMEbus mastership, and timeout status. All bits except bit 7 are flags that must be cleared manually by the local processor after being set by status conditions. If these bits are to be used for a specific operation, it is important that they be cleared prior to starting that operation.
Bit 0: (0/0/0)	Local timeout during VMEbus acquisition. This bit, when set, indicates that a local bus timeout has occurred during an attempted acquisition of the VMEbus.
Bit 1: (0/0/0)	SLSEL1* self-access. This bit is set when the VIC068A is selected by the assertion on the SLSEL1* signal, while operating as VMEbus master.
Bit 2: (0/0/0)	SLSEL0* self-access. This bit is set when the VIC068A is selected by the assertion on the SLSEL0* signal, while operating as VMEbus master.
Bit 3: (0/0/0)	Local bus timeout. This bit, when set, indicates a local bus timeout occurred without qualification.
Bit 4: (0/0/0)	VMEbus timeout. This bit, when set, indicates the VIC068A has signaled a VMEbus timeout. This bit is relevant only if the VIC068A is system controller and the VMEbus timeout is enabled.
Bit 5: (0/0/0)	VMEbus bus error. This bit is set when a VMEbus bus error is signaled (BERR* asserted).
Bit 6: (0/0/0)	Local bus error. This bit is set when a local bus error is signaled by a source other then the VIC068A (LBERR* asserted to the VIC068A).
Bit 7: (X/X/X)	VMEbus mastership. This bit is set whenever the VIC068A is VMEbus master.

DMA Status Register

Name:	DMASR
Address:	$BF
Description:	This register provides status of a VIC068A DMA transfer. This includes the block transfer with local DMA function and the module-based DMA function. Status bits are included to show various BERR* and LBERR* statuses and DMA termination statuses.
Bit 0: (0/0/0)	Block transfer in progress. This bit, when set, indicates an interleaved block transfer is in progress. Once set, this bit must be cleared manually by writing a 0 (zero) to this bit location, or by resetting the VIC068A.
Bit 1: (0/0/0)	LBERR* during DMA transfer. This bit, when set, indicates a LBERR* was signaled during a DMA transfer. Once set, this bit must be cleared manually by writing a 0 (zero) to this bit location, or by resetting the VIC068A.
Bit 2: (0/0/0)	BERR* during DMA transfer. This bit, when set, indicates a BERR* was signaled during a DMA transfer. Once set, this bit must be cleared manually by writing a 0 (zero) to this bit location, or by resetting the VIC068A.
Bit 3: (0/0/0)	Local bus error (read-only). This bit is set when a local bus error is signaled by a source other then the VIC068A (LBERR* asserted to the VIC068A). This bit is a read-only copy of bit 6 of the BESR.
Bit 4: (0/0/0)	VMEbus bus error. This bit is set when a VMEbus bus error is signaled (BERR* asserted). This bit is a copy of bit 5 of the BESR.
Bits 5, 6: (1/1/1)	Undefined/Reserved. These bits will be read as 1's.
Bit 7: (0/0/0)	Master write post information stored. This bit is set whenever master write post information is stored.

Slave Select 0 Control Register 0

Name: SS0CR0

Address: $C3

Description: This register provides control of the slave selection 0 facilities of the VIC068A. Enabling of the LIRQ2 timer interrupt is also configured in this register.

Bits 1–0: (0/*/*) Local transfer mode. These bits set the local transfer mode when the VIC068A is local bus master for both slave and master block transfers.

Bit 1	*Bit 0*	*Mode*
0	0	No support is given for slave block transfers on SLSEL0. The VIC068A will BERR* any attempt to receive a VMEbus block transfer. Master block transfers with local DMA will not function in this mode.
0	1	Emulate single-cycle transfers on the local bus. In this mode, the VIC068A emulates single-cycle transfers when performing slave block transfers and master block transfers with local DMA. By emulating single-cycle transfers, the VIC068A toggles the PAS* for each cycle. DSACKi must toggle for each transfer and not be held asserted.
1	0	Accelerated transfers on the local bus. In this mode, the VIC068A asserts the PAS* signal for the entire slave block transfer and master block transfer with local DMA. The DSACKi* signals should be held asserted in this mode.
1	1	Undefined/Reserved.

Slave Select 0 Control Register 0 (continued)

Bits 3–2: (0/*/*) Address space configuration. The SLSEL0 address space is configured according to the following table:

Bit 3	*Bit 2*	*Address Space*
0	0	A32 (extended) (default)
0	1	A24 (standard)
1	0	A16 (short)
1	1	User defined, uses AMSR

Bit 4: (0/*/*) D32 enable. D32 slave operations are enabled for SLSEL0 when this bit is set. This bit has no effect for enabling D32 master accesses. This bit also controls byte-lane switching for D16 Block transfers. When set ISOBE* and SWDEN* alternate states thus alternating which D16 bus data is placed. When clear, only SWDEN* is asserted for D16 block transfers.

Bit 5: (0/*/*) Supervisory access. When set, SLSEL0* slave accesses are restricted to supervisory accesses. Other accesses are BERRed. Supervisory accesses are checked with the AM(2) signal.

Bits 7–6: (0/0/0) Periodic interrupt timer enable. These bits enable and determine the frequency of the periodic LIRQ2 interrupt. If the VIC068A is to handle this local interrupt, LICR2 must be enabled. The frequencies for this interrupt are given below:

Bit 7	*Bit 6*	*Timer Mode*
0	0	Timer disabled (default)
0	1	50-Hz output on LIRQ2*
1	0	1000-Hz output on LIRQ2*
1	1	100-Hz output on LIRQ2*

Slave Select 0 Control Register 1

Name: SS0CR1

Address: $C7

Description: This register provides the various access and acquisition timings for slave transfers and slave block transfers for SLSEL0* in addition to data acquisition timing for master block transfers with local DMA.

Bits 3–0: (0/*/*) Timing field 0. This bit field establishes the following data access/acquisition timings:

- single-cycle slave access timing for SLSEL0* (SAT)
- first cycle of a slave block transfer for SLSEL0* (SBAT0)
- first cycle of a master block transfer with local DMA (MBAT0)
- first cycle of a module-based DMA transfer (DMAAT0)

The delays are programmed in multiples of the CLK64M clock period according to the following table:

Bit 3	*Bit 2*	*Bit 1*	*Bit 0*	*CLK64M Clock Period Delay*
0	0	0	0	0
0	0	0	1	2.0
0	0	1	0	2.5
0	0	1	1	3.0
0	1	0	0	3.5
0	1	0	1	4.0
0	1	1	0	4.5
0	1	1	1	5.0
1	0	0	0	5.5
1	0	0	1	6.0
1	0	1	0	6.5
1	0	1	1	7.0
1	1	0	0	7.5
1	1	0	1	8.0
1	1	1	0	8.5
1	1	1	1	9.0

Slave Select 0 Control Register 1 (continued)

Bits 7–4: (0/*/*) Timing Field 1. This bit field establishes the following data access/acquisition timings:

- second and subsequent cycle of a slave block transfer for SLSEL0* (SBAT1)
- second and subsequent cycle of a master block transfer with local DMA (MBAT1)
- second and subsequent cycle of a module–based DMA transfer (DMAAT1)

The delays are programmed in multiples of the CLK64M clock period according to the following table:

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*CLK64M Clock Period Delay*
0	0	0	0	0
0	0	0	1	2.0
0	0	1	0	2.5
0	0	1	1	3.0
0	1	0	0	3.5
0	1	0	1	4.0
0	1	1	0	4.5
0	1	1	1	5.0
1	0	0	0	5.5
1	0	0	1	6.0
1	0	1	0	6.5
1	0	1	1	7.0
1	1	0	0	7.5
1	1	0	1	8.0
1	1	1	0	8.5
1	1	1	1	9.0

Slave Select 1 Control Register 0

Name: SS1CR0

Address: $CB

Description: This register provides control of the slave selection 1 facilities of the VIC068A. Master and slave write posting is enabled in this register as well.

Bits 1–0: 0/*/* Local Transfer Mode. These bits set the local transfer mode when the VIC068A is local bus master for both slave and master block transfers.

Bit 1	*Bit 0*	*Mode*
0	0	No support is given for slave block transfers on SLSEL1. The VIC068A will BERR* any attempt to receive a VMEbus block transfer.
0	1	Emulate single-cycle transfers on the local bus. In this mode, the VIC068A emulates single-cycle transfers when performing slave block transfers. By emulating single-cycle transfers, the VIC068A toggles PAS* for each cycle. DSACKi must toggle for each transfer and not be held asserted.
1	0	Accelerate transfers on the local bus. In this mode, the VIC068A asserts PAS* for the entire slave block transfer. The DSACKi* signals should be held asserted in this mode.
1	1	Undefined/Reserved.

Slave Select 1 Control Register 0 (continued)

Bits 3–2: (0/*/*) Address Space Configuration. The SLSEL1 address space is configured according to the following table:

Bit 3	*Bit 2*	*Address Space*
0	0	A32 (extended)
0	1	A24 (standard)
1	0	A16 (short)
1	1	User defined, uses AMSR

Bit 4: (0/*/*) D32 enable. D32 slave operations are enabled for SLSEL1 when this bit is set. This bit has no effect for enabling D32 master accesses.

Bit 5: (0/*/*) Supervisory access. When set, SLSEL1* slave accesses are restricted to supervisory accesses. Other accesses are BERRed. Supervisory accesses are checked with the AM(2) signal.

Bit 6: (0/0/0) Master write post enable. When set, master write posting is enabled.

Bit 7: (0/0/0) Slave write post enable. When set, slave write posting is enabled.

Slave Select 1 Control Register 1

Name: SS1CR1

Address: $CF

Description: This register provides the various access and acquisition timings for slave transfers and slave block transfers for SLSEL1*.

Bits 3–0: (0/*/*) Timing field 0. This bit field establishes the following data access/acquisition timings:

- single-cycle slave access timing for SLSEL1* (SAT)
- first cycle of a slave block transfer for SLSEL1* (SBAT0)

The delays are programmed in multiples of the CLK64M clock period according to the following table:

Bit 3	*Bit 2*	*Bit 1*	*Bit 0*	*CLK64M Clock Period Delay*
0	0	0	0	0
0	0	0	1	2.0
0	0	1	0	2.5
0	0	1	1	3.0
0	1	0	0	3.5
0	1	0	1	4.0
0	1	1	0	4.5
0	1	1	1	5.0
1	0	0	0	5.5
1	0	0	1	6.0
1	0	1	0	6.5
1	0	1	1	7.0
1	1	0	0	7.5
1	1	0	1	8.0
1	1	1	0	8.5
1	1	1	1	9.0

Bits 7–4: (0/*/*) Timing field 1. This bit field establishes the following data access/acquisition timing:

- Second and subsequent cycle of a slave block transfer for SLSEL1* (SBAT1)

Slave Select 1 Control Register 1 (continued)

The delays are programmed in multiples of the CLK64M clock period according to the following tables:

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*CLK64M Clock Period Delay*
0	0	0	0	0
0	0	0	1	2.0
0	0	1	0	2.5
0	0	1	1	3.0
0	1	0	0	3.5
0	1	0	1	4.0
0	1	1	0	4.5
0	1	1	1	5.0
1	0	0	0	5.5
1	0	0	1	6.0
1	0	1	0	6.5
1	0	1	1	7.0
1	1	0	0	7.5
1	1	0	1	8.0
1	1	1	0	8.5
1	1	1	1	9.0

Release Control Register

Name: RCR

Address: $D3

Description: This register configures the VMEbus release mode. The burst count for block transfers with local DMA is also configured in the RCR.

Bits 5–0: (0/0/0) Block transfer burst length. The burst length for both MOVEM block transfers and block transfers with local DMA are configured in this bit field. The value indicates the number of cycles per block transfer (not the number of bytes). A value of 0 in this bit field indicates the maximum 64 cycles per burst. All other values correspond directly to the burst count.

Bits 7,6: (0/0/0) Release mode. This bit field defines the release mode used by the VIC068A when releasing the VMEbus after the completion of a VMEbus transfer.

Bit 7	*Bit 6*	*Release Mode*
0	0	ROR—Release on Request (default)
0	1	RWD—Release When Done
1	0	ROC—Release on BCLR* assertion
1	1	BCAP—VMEbus Capture and Hold

Block Transfer Control Register

Name: BTCR

Address: $D7

Description: The BTCR provides control of the VIC068A block transfers. The local interleave periods and data direction are defined in this register. The enabling bits for all of the VIC068A's block transfer modes are located here as well. These enabling bits are mutually exclusive and more than one should not be set at the same time.

Bits 3–0: (0/0/0) Interleave period. The interleave period for block transfers is defined here. The interleave period is 250 ns times the value programmed in this bit field.

Bit 4: (0/0/0) Data direction. This bit defines the direction of a block transfer with local DMA (MOVEM data direction determined by the R/W* signal). When set, VMEbus block reads occur. When clear, VMEbus block writes occur.

Bit 5: (0/0/0) MOVEM enable. When set, MOVEM transfers are enabled. After this bit is set, the next VMEbus transfer is treated as the start of a VMEbus block transfer. Clearing this bit concludes a MOVEM block transfer in progress. It is important to set this bit immediately before and clear this bit immediately after the actual MOVEM transfer.

Bit 6: (0/0/0) Block transfer with local DMA enable. When set, block transfers with local DMA are enabled. After this bit is set, the next assertion of MWB* is considered the initiation cycle of a VMEbus block transfer with local DMA. It is important to set this bit immediately before and clear this bit immediately after the actual block transfer.

Bit 7: (0/0/0) Module-based DMA transfer enable. When set, module-based DMA transfers are enabled. After this bit is set, the next assertion of BLT* by external logic is considered the initiation cycle of a module-based DMA transfer. Clearing this bit concludes a module-based DMA transfer in progress. It is important to set this bit immediately before and clear this bit immediately after the actual DMA transfer.

Block Transfer Length Registers 1–0

Name: BTLR1–0

Addresses: $DB (BTLR1), $DF (BTLR0)

Description: These registers configure the byte count for block transfers with local DMA. BTLR1 is considered the most significant byte and BTLR0 the least significant. Bit 0 of BTLR0 must never be set because this implies at least one 8-bit transfer is required to complete the block transfer. Only D16 and D32 block transfers are supported. If bit 0 of BTLR0 is set, the block transfer length is ignored and only one burst is performed.

Bits 7–0: (0/0/0) Block transfer length. Defines the block transfer length in bytes. BTLR1 contains the most significant 8 bits of the length, and BTLR0 the least.

System Reset Register

Name: SRR

Address: $E3

Description: The system reset register provides the means to perform a VME-bus system reset (SYSRESET* asserted). Writing a value of $F0 causes this function to occur. A system reset is also performed within the VIC068A.

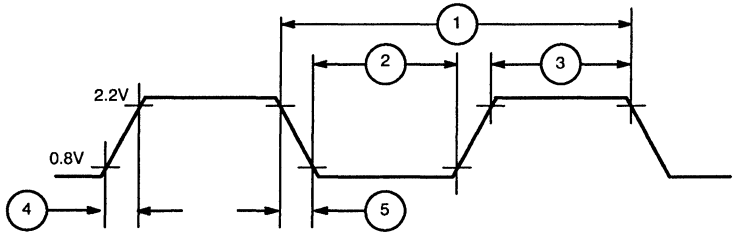
Bits 7–0: (1/1/1) System reset field. Writing this bit field with a value of $F0 causes SYSRESET* to be asserted for a minimum of 200 ms and a system reset to be performed within the VIC068A.

14

VIC068A AC Performance Specifications

Clock Input

Num.	Characteristic	Min.	Max.
	Frequency of Operation (MHz)	1	64
1	Cycle Time (ns)	15.6	1000
2, 3	Clock Pulse Width (Measured from 1.5V to 1.5V)	Note 1	Note 1
4, 5	Rise and Fall Time (ns)	—	5



Note:

1. A 60/40 to 40/60 duty cycle must be maintained.

AC Specifications[2]

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
ARBITRATION								
A1	BRi*[0] to BBSY*[H]	3	2½T+5	3T+25	2½T+4	3T+26	2½T+4	3T+31½
A2	BRi*[0] to BBSY*[L]	4	3T+8	3½T+28	3T+7	3½T+34	3T+7	3½T+35
A3	BRi*[0] to BGiOUT*[L]	4	2T+4	2T+25	2T+4	2T+26	2T+3	2T+28
A4	BRi*[0] to BCLR*[L]		2	16	2	16	2	19
A5	BGiIN*[0] to BGiOUT*[L]		2	17	2	17	2	19
A6	BGiIN*[0] to BBSY*[L]	5	4	3T+24	4	3T+25	3	3T+30
A7	BGiIN*[0] to BRi*[H]	5	5	3T+26	4	3T+27	4	3T+31
A8	BGiIN*[1] to BGiOUT*[H]		3	20	2	21	2	23

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
A9	BBSY*[0] to BGiOUT[H]	4	4	21	3	22	3	24
A10	BBSY*[1] to BGiOUT*[L]		3T+5	4T+25	3T+4	4T+26	3T+3	4T+29
A11	BBSY*[1] to BCLR*[H]		1T+4	2T+24	1T+4	2T+25	1T+3	2T+27
MASTER ACCESSES								
B1	BGiIN*[0] to DENO*[L]	5, 6	8	3T+36	7	3T+37	6	3T+42
B2	BGiIN*[0] to LADO[H]	5	14	3T+59	13	3T+61	12	3T+67
B3	BGiIN*[0] to AS*[L]	5	3T+5	6T+28	3T+5	6T+29	3T+4	6T+31
B4	BGiIN*[0] to A[7:1] Valid	5	6	3T+31	6	3T+32	5	3T+37
B5	BGiIN*[0] to LWORD*[H/L]	5	6	3T+31	6	3T+32	5	3T+37
B6	BGiIN*[0] to WRITE*[H/L]	5	6	3T+31	6	3T+32	5	3T+37
B7	BGiIN*[0] to ABEN*[L]	5	7	3T+29	6	3T+30	6	3T+30
B8	PAS*[0] & MWB*[0] to BRi*[L]		4	22	3	22	3	24
B9	PAS*[0] & MWB*[0] to ISOBE*[L]		4	22	3	23	3	25
B10	PAS*[0] & MWB*[0] to LADO[H]		15	60	13	62	12	68
B11	PAS*[0] & MWB*[0] to BBSY*[L]	7	7	32	5	33	5	36
B12	PAS*[0] & MWB*[0] to ABEN*[L]	7	2½T+8	3½T+36	2½T+7	3½T+37	2½T+6	3½T+41
B13	PAS*[0] & MWB*[0] to A[7:1]	7	7	32	6	34	5	38
B14	PAS*[0] & MWB*[0] to LWORD*[H/L]	7	7	32	6	34	5	38
B15	PAS*[0] & MWB*[0] to WRITE*[H/L]	7	7	32	6	34	5	38
B16	PAS*[0] & MWB*[0] & DS*[0] to DS1/0*[L]	7	5½T+10	6½T+46	5½T+9	6½T+47	5½T+9	6½T+57
B17	PAS*[0] & MWB*[0] to SWDEN*[L]		5	11	3	12	3	14
B18	PAS*[0] & MWB*[0] to LWDENIN*[L]	8	3	20	3	20	2	22
B19	PAS*[0] & MWB*[0] to UWDENIN*[L]	8	3	20	3	21	3	23
B20	PAS*[0] & MWB*[0] & DS*[0] to AS*[L]	7	5½T+6	6½T+28	5½T+5	6½T+29	5½T+5	6½T+32
B21	R/W*[0] to DDIR*[H]	6	4	22	3	23	2	25
B22	R/W*[1] to DDIR*[L]	6	2	14	1	14	1	15
B23	D[7:0] to LD[7:0] Valid	8	3	18	2	18	2	22
B24	DTACK*[0] to LEDI[H]	8	3T+6	4T+28	3T+4	4T+29	3T+4	4T+32

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
B25	DTACK*[0] to DSACKi*[L]		4	30	3	31	3	36
B26	PAS*[1] & DS*[1] to DSACKi*[H]		2	19	2	20	2	27
B27	PAS*[1] to AS*[H]		6	30	5	31	5	41
B28	DS*[1] to ISOBE*[H]		4	23	3	24	3	26
B29	DS*[1] to SWDEN*[H]		4	10	3	10	2	13
B30	DS*[1] to UWDENIN*[H]	8	3	19	3	20	2	22
B31	DS*[1] to LWDENIN*[H]	8	3	19	3	20	2	22
B32	DS*[1] to LD[7:0] Invalid	8	3	20	2	22	2	28
B33	DS*[1] to LD[7:0] Hi-Z	8	3	20	2	22	2	28
B34	DS*[0] to DSACKi*[L]	9	6	39	5	41	5	51
B35	DS*[0] to LADO[H]	9	8	38	7	39	7	43
B36	DS*[0] to LEDO[H]	9	4	22	3	23	3	25
LOCAL BUS TIMING (VIC068A AS LOCAL BUS MASTER)								
C1	LBG*[0] to PAS*[L]		5T+6	6T+31	5T+5	6T+33	5T+5	6T+44
C2	LBG*[0] to LA[7:0] Valid		3T+8	4T+36	3T+7	4T+37	3T+6	4T+46
C3	LBG*[0] to SIZ[1:0] Valid		1T+3	2T+20	1T+3	2T+21	1T+2	2T+28
C4	LBG*[0] to FC[2:1] Valid		1T+3	2T+20	1T+3	2T+21	1T+2	2T+27
C5	LBG*[0] to LD[7:0] Driven	6	3T+8	4T+38	3T+7	4T+39	3T+7	4T+48
C6	LBG*[0] to LAEN[H]		3T+10	4T+43	3T+9	4T+44	3T+8	4T+48
C7	LBG*[0] to ISOBE*[L]		3T+8	4T+37	3T+7	4T+39	3T+7	4T+42
C8	LBG*[0] to SWDEN*[L]		3T+9	4T+39	3T+8	4T+41	3T+7	4T+45
C9	LBG*[0] to DDIR*[H]	6	3T+8	4T+37	3T+7	4T+39	3T+7	4T+42
C10	LBG*[0] to UWDENIN*[L]	6	3T+7	4T+33	3T+6	4T+34	3T+6	4T+38
C11	LBG*[0] to LWDENIN*[L]	6	3T+7	4T+32	3T+6	4T+33	3T+5	4T+36
C12	LBG*[0] & DS1/0*[0] & WRITE[0] to R/W*[L]	6	3T+8	4T+38	3T+7	4T+40	3T+7	4T+47
C13	LBG*[0] & DS1/0*[0] to DS*[L]		5T+8	6T+39	5T+7	6T+42	5T+7	6T+56
C14	PAS*[0] to DS*[L]	10	0	12	0	15	0	15
C15	LBR*[H] to LBG*[1]	11		T		T		T
SLAVE ACCESSES								
D1	SLSELi*[0] & AS*[0] to LBR*[L]		7	35	6	36	6	40
D2	SLSELi*[0] & AS*[0] & DS1/0*[1] to LADI[H]		5	25	4	26	4	29
D3	LD[7:0] to D[7:0]	8	2	16	2	16	2	18

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
D4	DSACKI*[0] to LEDO*[H]	8	SAT+8	SAT+½T+35	SAT+7	SAT+½T+36	SAT+6	SAT+½T+39
D5	DSACKi*[0] to DTACK*[L]		SAT+10	SAT+½T+45	SAT+9	SAT+½T+47	SAT+9	SAT+½T+53
D6	DS1/0*[0] to DTACK*[L]	12	2T+5	3T+28	2T+5	3T+29	2T+4	3T+33
D7	DS1/0*[0] to LEDI[H]	12	9	41	8	43	8	47
D8	AS*[1] to LA[7:0], R/W* Invalid		5	38	4	42	4	55
D9	AS*[1] to LA[7:0], R/W* Hi−Z		5	38	4	42	4	55
D10	AS*[1] to FC2/1, Invalid		10	42	8	44	8	56
D11	AS*[1] & DSACKi*[1] to FC2/1, Hi-Z		10	42	8	44	8	56
D12	AS*[1] to SIZ1/0, Invalid		7	32	6	34	6	37
D13	AS*[1] & DSACKi*[1] to SIZ1/0, Hi-Z		3	1T+17	2	1T+19	2	1T+24
D14	AS*[1] to ISOBE*[H]		6	30	5	31	5	34
D15	AS*[1] to SWDEN*[H]		4	24	4	25	3	27
D16	AS*[1] to UWDENIN*[H]	6	5	27	4	28	4	30
D17	AS*[1] to LWDENIN*[H]	6	5	27	4	28	4	30
D18	AS*[1] & DSACKi*[1] to LBR*[H]		5	26	4	27	4	30
D19	AS*[1] to LAEN[L]		9	40	8	43	7	56
D20	DS*1/0[1] to LD[7:0] Invalid	6	2	27	2	30	2	39
D21	DS*1/0[1] to LD[7:0] Hi-Z	6	2	27	2	30	2	39
D22	DSACKi*[0] to PAS*[H]		SAT+10	SAT+½T+44	SAT+9	SAT+½T+46	SAT+8	SAT+½T+56
D23	DSACKi*[0] to DS*[H]		SAT+9	SAT+½T+40	SAT+8	SAT+½T+41	SAT+7	SAT+½T+48
D24	DSi*[1] to DTACK*[H]		3	27	3	28	3	35
INTERRUPT								
E1	IACKIN*[0] to IACKOUT*[L]		2	16	2	17	2	18
E2	IACKIN*[1] to IACKOUT*[H]		3	18	2	19	2	20
E3	FCIACK*[0] & PAS*[0] to BRi*[L]		5T+9	6T+41	5T+8	6T+42	5T+7	6T+48
E4	FCIACK*[0] & PAS*[0] to IACK*[L]	7	7½T+7	8½T+34	7½T+6	8½T+35	7½T+6	8½T+39
E5	FCIACK*[0] & PAS*[0] to LD[7:0] Driven	13	5T+12	6T+50	5T+10	6T+52	5T+10	6T+57

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
E6	FCIACK*[0] & PAS*[0] to LD[7:0] valid	14	10T+5	11T+27	10T+5	11T+28	10T+4	11T+37
E7	FCIACK*[0] & PAS*[0] to LIACK0*[L]	15	5T+7	6T+32	5T+6	6T+33	5T+5	6T+36
E8	IRQi*[0] to IPL*		5	33	5	34	4	37
E9	BGiIN*[0] to BBSY*[L]		7	32	5	33	5	36
E10	BGiIN*[0] to AS*[L]		3T+5	4T+27	3T+4	4T+28	3T+4	4T+31
E11	BGiIN*[0] to DS1/0*[L]		3T+10	4T+45	3T+9	4T+46	3T+8	4T+55
E12	BGiIN*[0] to IACK*[L]	14	39	7	40	7	44	
E13	PAS*[0] to ISOBE*[L]		5T+9	6T+39	5T+7	6T+40	5T+7	6T+44
E14	PAS*[0] to SWDEN*[L]		5T+8	6T+37	5T+7	6T+38	5T+6	6T+42
E15	IPLi* to IPLi*	10		10		12		12
MASTER BLOCK TRANSFER WITH LOCAL DMA (INITIATION CYCLE)								
F1	MWB*[0] & PAS*[0] & DS*[0] to BRi*[L]		4T+7	5T+32	4T+6	5T+33	4T+5	5T+38
F2	BGiIN*[0] to LBR*[L]		4T+10	5T+42	4T+8	5T+44	4T+8	5T+50
F3	MWB*[0] & PAS*[0] & DS*[0] to LBR*[L]	7	7½T+10	8½T+42	7½T+8	8½T+44	7½T+8	8½T+50
F4	MWB*[0] & PAS*[0] & DS*[0] to LADO[H]		2T+7	3T+35	2T+6	3T+36	2T+6	3T+39
F5	MWB*[0] & PAS*[0] & DS*[0] to BLT*[L]		2T+6	3T+28	2T+5	3T+29	2T+4	3T+37
MASTER BLOCK TRANSFER WITH LOCAL DMA (WRITE)								
** First cycle **								
G1	DSACKi*[0] and DS*[0] to DS*[H]		MBAT0+9	MBAT0+½T +41	MBAT0+8	MBAT0+½T +43	MBAT0+7	MBAT0+½T +52
G2	DSACKi*[0] and DS*[L] to LEDO[H]		MBAT0+8	MBAT0+½T +36	MBAT0+7	MBAT0+½T +37	MBAT0+6	MBAT0+½T +40
G3	DSACKi*[0] and DS*[L] to LA[7:0] valid		MBAT0+11	MBAT0+½T +49	MBAT0+9	MBAT0+½T +56	MBAT0+9	MBAT0+½T +80
G4	DSACKi*[0] and DS*[L] to DSi*[L]		MBAT0+6	MBAT0+½T +29	MBAT0+5	MBAT0+½T +30	MBAT0+5	MBAT0+½T +38
G5	DTACK*[0] to LEDO[L]		7	32	6	33	6	38
G6	DTACK*[0] to DSi*[H]		10	45	9	46	9	59
G7	DTACK*[0] to A[7:0] Valid		11	46	10	48	9	64
G8	DS*[H] to DS*[L]	16	DST+1½T −13	DST+1½T −6	DST+1½T −14	DST−1½T −5	DST+1½T −15	DST+1½T −4
** Second and subsequent cycles **								
G9	DSACKi*[0] and DS*[L] to DS*[H]		MBAT1+9	MBAT1+½T +41	MBAT1+8	MBAT1+½T +43	MBAT1+7	MBAT1+½T +52
G10	DSACKi*[0] and DS*[L] to LEDO[H]		MBAT1+8	MBAT1+½T +36	MBAT1+7	MBAT1+½T +37	MBAT1+6	MBAT1+½T +40

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
G11	DSACKi*[0] and DS*[L] to LA[7:0] Valid		MBAT1+11	MBAT1+½T +49	MBAT1+9	MBAT1+½T +56	MBAT1+9	MBAT1+½T +80
G12	DSACKi*[0] and DS*[L] to DSi*[L]		MBAT1+6	MBAT1+½T +29	MBAT1+5	MBAT1+½T +30	MBAT1+5	MBAT1+½T +38
G13	DTACK*[0] to LEDO[L]		7	32	6	33	6	38
G14	DTACK*[0] to DSi*[H]		10	45	9	46	9	59
G15	DTACK*[0] to A[7:0] Valid		11	46	10	48	9	64
G16	DTACK*[0] to DS*[H]	16,17	T + 15	1½T + 45	1½T + 14	1½T + 46	T + 13	1½T + 47
G17	LEDO[L] to LEDO[H]	16,17	T + 11	1½T + 25	1½T + 10	1½T + 26	T + 9	1½T + 27
MASTER BLOCK TRANSFER WITH LOCAL DMA (READ)								
** First Cycle **								
H1	DTACK*[0] to LEDI[H]		1½T+6	2½T+28	1½T+4	2½T+30	1½T+4	2½T+32
H2	DTACK*[0] to DSi*[H]		1½T+9	2½T+41	1½T+8	2½T+42	1½T+7	2½T+49
H3	DTACK*[0] to A[7:0] Valid		1½T+10	2½T+44	1½T+9	2½T+45	1½T+8	2½T+53
H4	DTACK*[0] to DS*[L]		1½T+8	2½T+38	1½T+7	2½T+40	1½T+7	2½T+47
H5	DSACKi*[0] and DS*[L] to DS*[H]		MBAT0+9	MBAT0+½T +41	MBAT0+8	MBAT0+½T +43	MBAT0+7	MBAT0+½T +52
H6	DSACKi*[0] and DS*[L] to LEDI[L]		MBAT0+8	MBAT0+½T +37	MBAT0+7	MBAT0+½T +38	MBAT0+6	MBAT0+½T +45
H7	DSACKi*[0] and DS*[L] to LA[7:0] Valid		MBAT0+11	MBAT0+½T +47	MBAT0+9	MBAT0+½T +48	MBAT0+9	MBAT0+½T +59
H8	DSACKi*[0] and DS*[L] to DSi*[L]		MBAT0+11	MBAT0+½T +50	MBAT0+10	MBAT0+½T +52	MBAT0+10	MBAT0+½T +68
** Second and subsequent cycles **								
H9	DTACK*[0] to LEDI[H]		6	28	4	30	4	32
H10	DTACK*[0] to DSi*[H]		9	41	8	42	7	49
H11	DTACK*[0] to A[7:0] Valid		10	44	9	45	8	53
H12	DTACK*[0] to DS*[L]		8	38	7	40	7	47
H13	DSACKi*[0] and DS*[0] to DS*[H]		MBAT1+9	MBAT1+½T +41	MBAT1+8	MBAT1+½T +43	MBAT1+7	MBAT1+½T +52
H14	DSACKi*[0] and DS*[0] to LEDI[L]		MBAT1+8	MBAT1+½T +37	MBAT1+7	MBAT1+½T +38	MBAT1+6	MBAT1+½T +45
H15	DSACKi*[0] and DS*[0] to LA[7:0] Valid		MBAT1+11	MBAT1+½T +47	MBAT1+9	MBAT1+½T +48	MBAT1+9	MBAT1+½T +59
H16	DSACKi*[0] and DS*[0] to DSi*[L]		MBAT1+11	MBAT1+½T +50	MBAT1+10	MBAT1+½T +52	MBAT1+10	MBAT1+½T +68
MASTER BLOCK TRANSFER WITH LOCAL DMA (BOUNDARY CROSSING)								
J1	DS*[L] to BLT*[H]		2	15	2	16	2	21
J2	DS*[H] to BLT[L]		12	52	11	55	10	73
J3	DSi*[L] to LEDO[H/L]		11	47	01	49	9	53

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
J4	DSi*[H] to LADO*[L/H]		15	62	13	64	13	69
SLAVE BLOCK TRANSFER (WRITE)								
** First Cycle **								
See: Local Bus Timing (VIC068A as local bus master)								
** Second and subsequent cycles **								
K1	DSACKi*[0] and DS*[L] to LEDI*[H]		SBAT+4	SBAT+½T+23	SBAT+4	SBAT+½T+24	SBAT+3	SBAT+½T+26
K2	DSi*[0] to DS*[L]		1T+6	1T+31	1T+6	1T+32	1T+5	1T+42
K3	DSACKi*[0] and DS*[L] to DS*[H]		SBAT+9	SBAT+½T+41	SBAT+8	SBAT+½T+42	SBAT+7	SBAT+½T+52
K4	DSACKi*[0] and DS*[L] to DTACK*[L]		SBAT+12	SBAT+½T+51	SBAT+11	SBAT+½T+53	SBAT+10	SBAT+½T+67
K5	DSACKi*[0] and DS*[L] to ISOBE*[H]		SBAT+13	SBAT+½T+54	SBAT+11	SBAT+½T+56	SBAT+11	SBAT+½T+62
K6	DSACKi*[0] and DS*[L] to SWDEN*[H]		SBAT+12	SBAT+½T+50	SBAT+10	SBAT+½T+52	SBAT+10	SBAT+½T+61
K7	DSACKi*[0] and DS*[L] to LA[7:0] Valid		SBAT+10	SBAT+½T+42	SBAT+8	SBAT+½T+44	SBAT+8	SBAT+½T+57
K8	DS1/0*[1] to LEDI*[L]		8	37	7	38	6	45
K9	DS1/0*[1] to DTACK*[H]		5	27	5	28	4	35
SLAVE BLOCK TRANSFER (READ)								
** First Cycle **								
See: Local Bus Timing (VIC068A as local bus master)								
** Second and subsequent cycles **								
L1	DS1/0*[1] to LEDO*[L]		4	23	3	24	3	30
L2	DS*[H] to DS*[L]	16	DST+1½T−13	DST+1½T−2	DST+1½T−14	DST+1½T−3	DST+1½T−15	DST+1½T−4
L3	DS1/0*[0] to DENO*[L]		3	20	3	20	3	22
L4	DSACKi*[0] and DS*[0] to LEDO*[H]		SBAT+8	SBAT+½T+36	SBAT+7	SBAT+½T+37	SBAT+6	SBAT+½T+41
L5	DSACKi*[0] and DS*[0] to DS*[H]		SBAT+9	SBAT+½T+41	SBAT+8	SBAT+½T+43	SBAT+7	SBAT+½T+52
L6	DSACKi*[0] and DS*[0] to DTACK*[L]		SBAT+11	SBAT+½T+46	SBAT+9	SBAT+½T+47	SBAT+9	SBAT+½T+57
L7	DSACKi*[0] and DS*[0] to LA[7:0] Valid		SBAT+9	SBAT+½T+42	SBAT+8	SBAT+½T+43	SBAT+8	SBAT+½T+57
L8	DS1/0*[1] to DENO*[H]		3	19	3	20	2	22
L9	DS1/0*[1] to DTACK*[H]		3	20	3	21	3	24
L10	LEDO[L] to LEDO[H]	16,18	T + 11	1½ + 25	T + 10	1½ + 26	1½ + 9	1½ + 27
L11	DTACK*[0] to DS*[H]	16,18	T + 15	1½ + 45	1½ + 14	1½ + 46	1½ + 13	1½ + 47

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
REGISTER ACCESS								
M1	PAS*[0] & DS*[0] & CS*[0] to DSACKi[L]		4T+5	4T+34	4T+5	4T+35	4T+4	4T+38
M2	PAS*[0] & DS*[0] & CS*[0] to LD[7:0] Valid	8	4T+5	4T+28	4T+5	4T+29	4T+4	4T+37
M3	AS*[0] & ICFSEL*[0] to DTACK*[L]		4T+6	4T+30	4T+5	4T+31	4T+5	4T+34
RESET								
N1	LBG*[0] to HALT*[L], RESET*[L]		8	36	7	37	6	48
N2	IRESET*[0] to LBR*[L]		6	29	5	30	5	33
N3	IRESET*[0] to IPL0[Z]		2	16	2	16	2	20
SET-UP TIMES								
P1	LA, ASIZ[1:0] Valid to PAS*[0]		−2T		−2T		−2T	
P2	SIZ[1:0], WORD*, FC[2:1] Valid to PAS*[0]		−2T		−2T		−2T	
P3	LD[7:0] Valid to DS*[0]		0		0		0	
HOLD TIMES								
Q1	PAS*[1] to LA, ASIZ[1:0] Invalid		0		0		0	
Q2	PAS*[1] to SIZ[1:0], WORD*, FC[2:1] Invalid		0		0		0	
Q3	DS*[1] to LD[7:0] Invalid		0		0		0	
Q4	DS1/0*[1] to DTACK*[H]		0		0		0	

Notes:

2. T = CLK64M clock period
   SAT = Slave Access Timing
   MBAT0 = Master Block Transfer Timing 0
   MBAT1 = Master Block Transfer Timing 1
   SBAT = Slave Block Transfer Timing
   DST = Data Strobe Timing
3. ROR mode.
4. While VMEbus system controller.
5. Synchronous delay depends on speed in which BGiIN is returned. If BGiIN is returned in zero time after request, synchronous delay will be maximum.
6. Write operation only.
7. While VMEbus master.
8. Read operation only.
9. Master write post only.
10. Skew.
11. Input requirement.
12. Slave write post only.
13. VMEbus interrupt only.
14. Local interrupt (LICR[4] = 1) only.
15. Local interrupt (LICR[4] = 0) only.
16. Timing specified but not tested.
17. "Slow" Slave.
18. "Slow" Master.

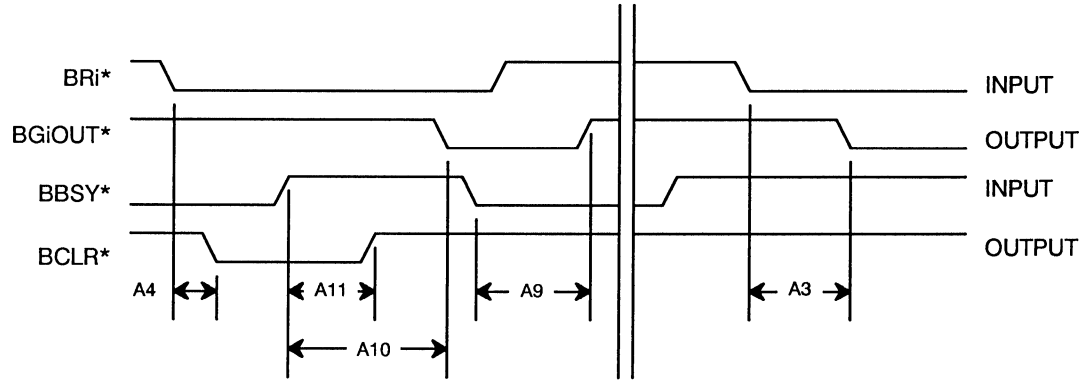


Figure 14–1. VMEbus Arbitration—Scenario 1

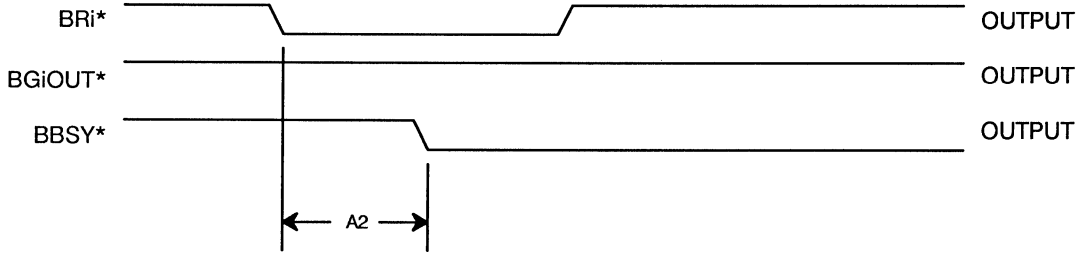


Figure 14–2. VMEbus Arbitration—Scenario 2

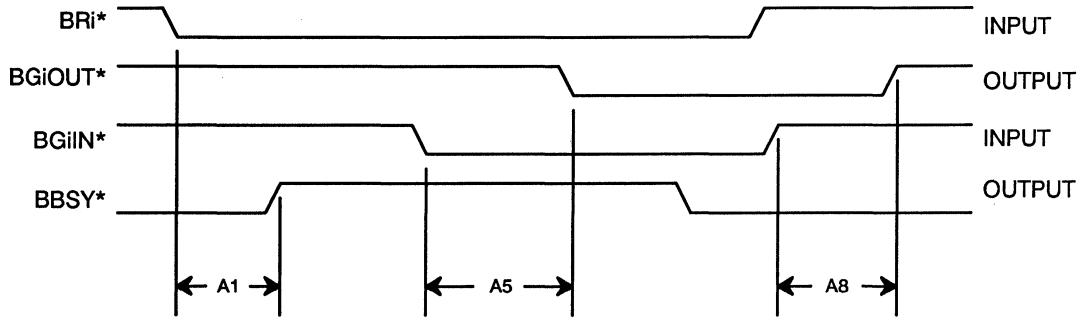


Figure 14–3. VMEbus Arbitration—Scenario 3

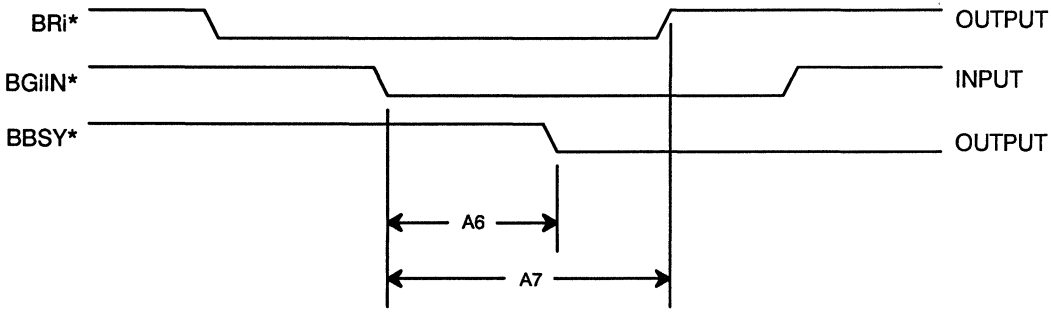


Figure 14–4. VMEbus Arbitration—Scenario 4

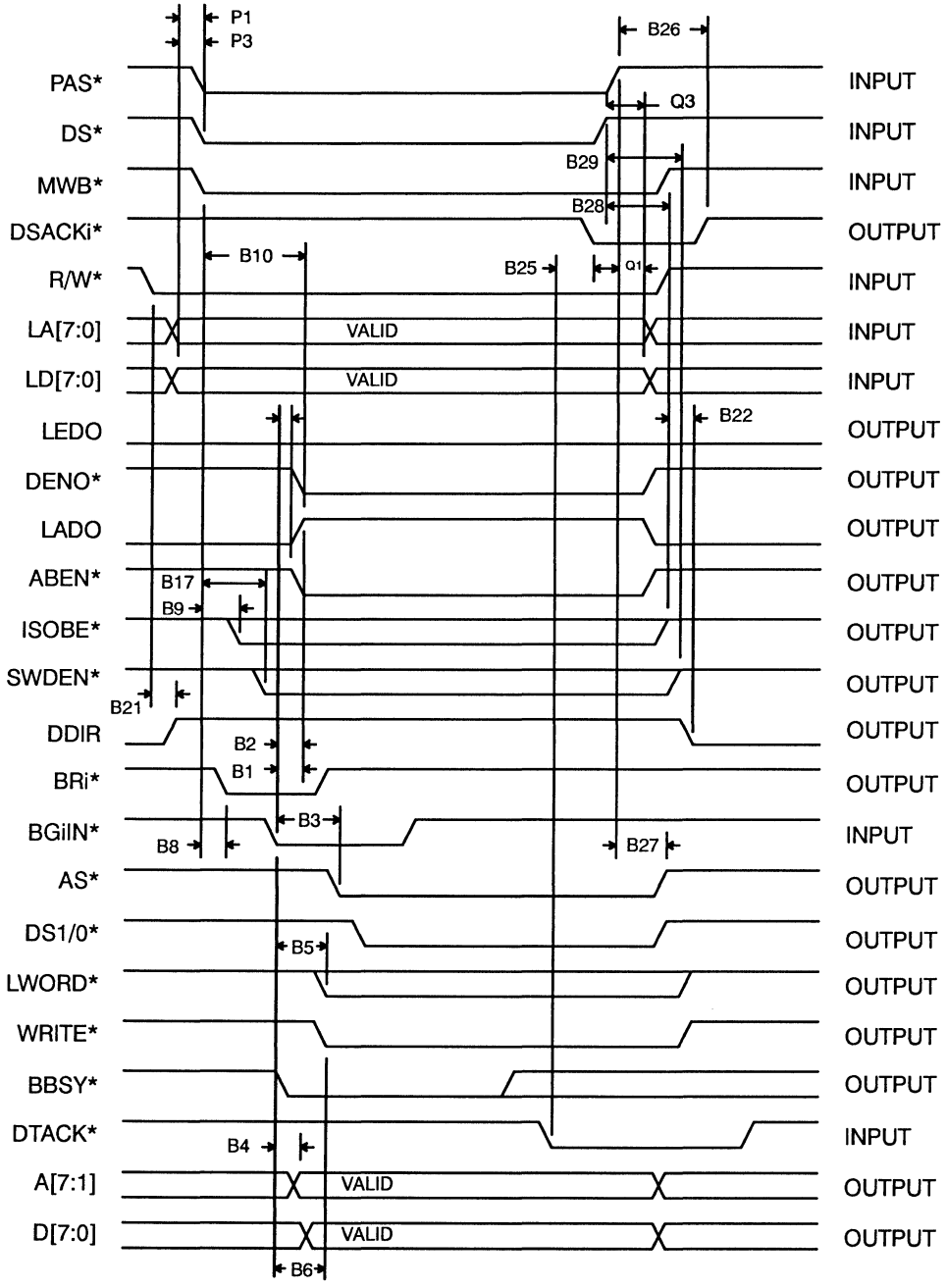


Figure 14–5. Master Write

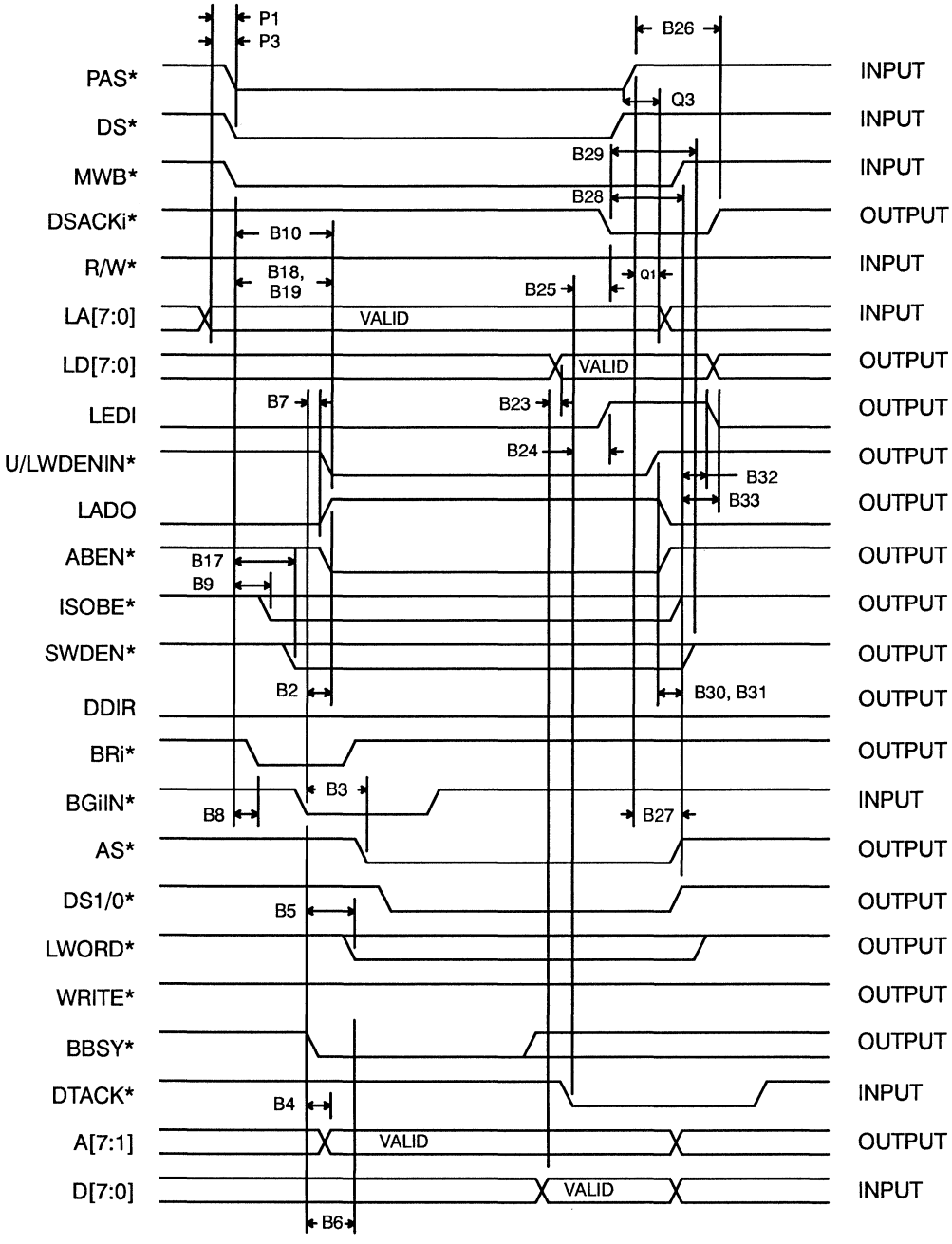


Figure 14–6. Master Read

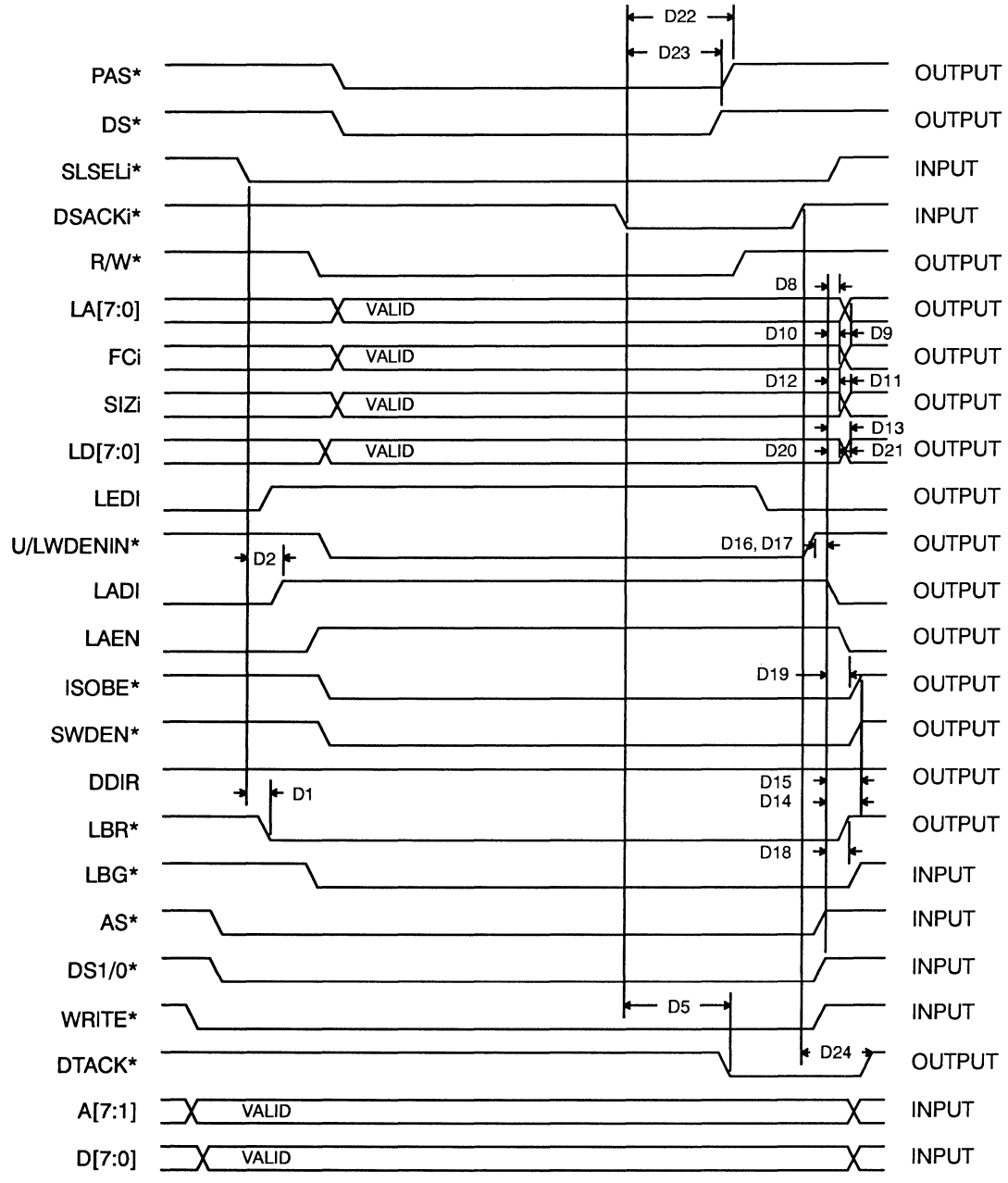


Figure 14–7. Slave Write

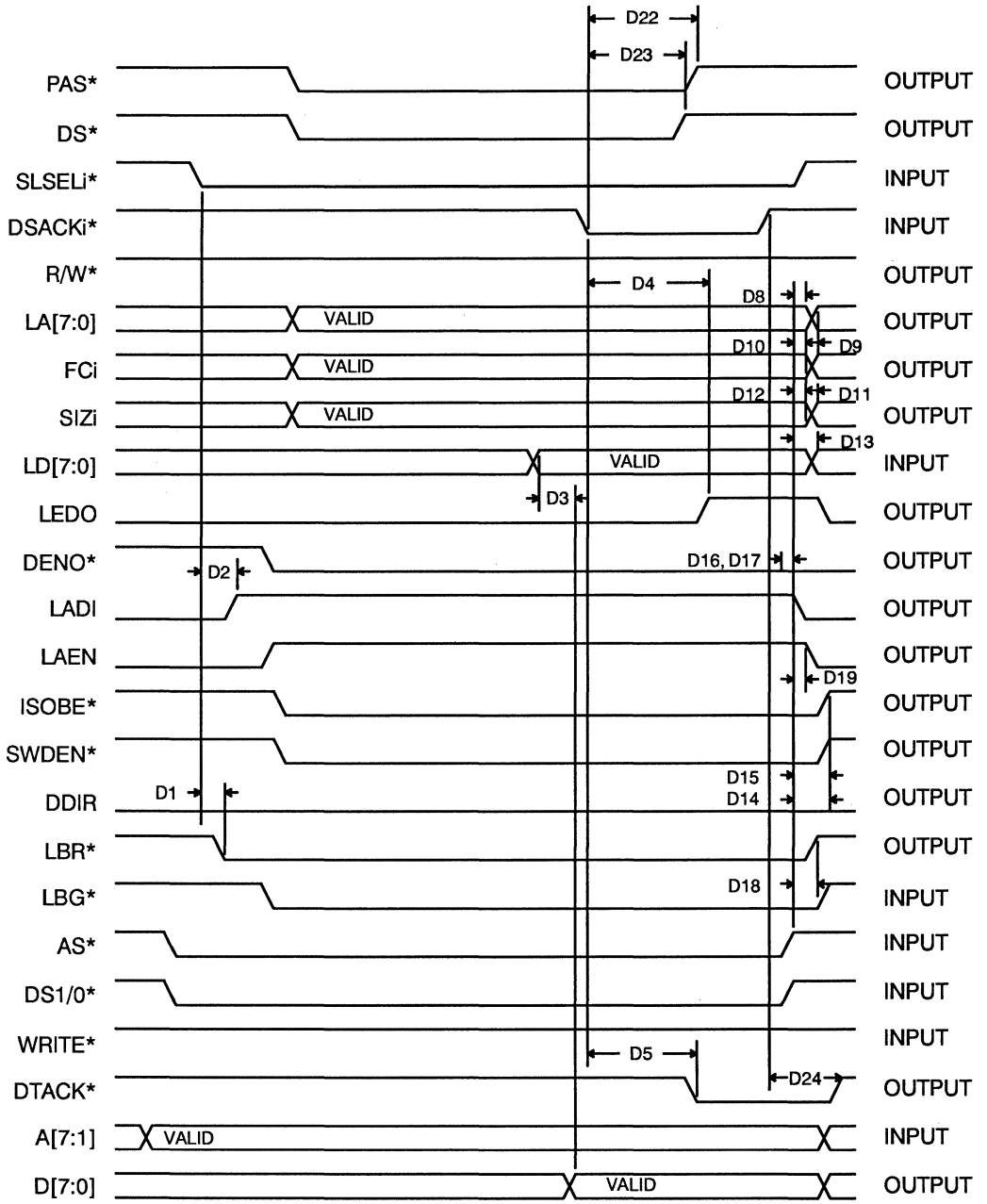


Figure 14–8. Slave Read

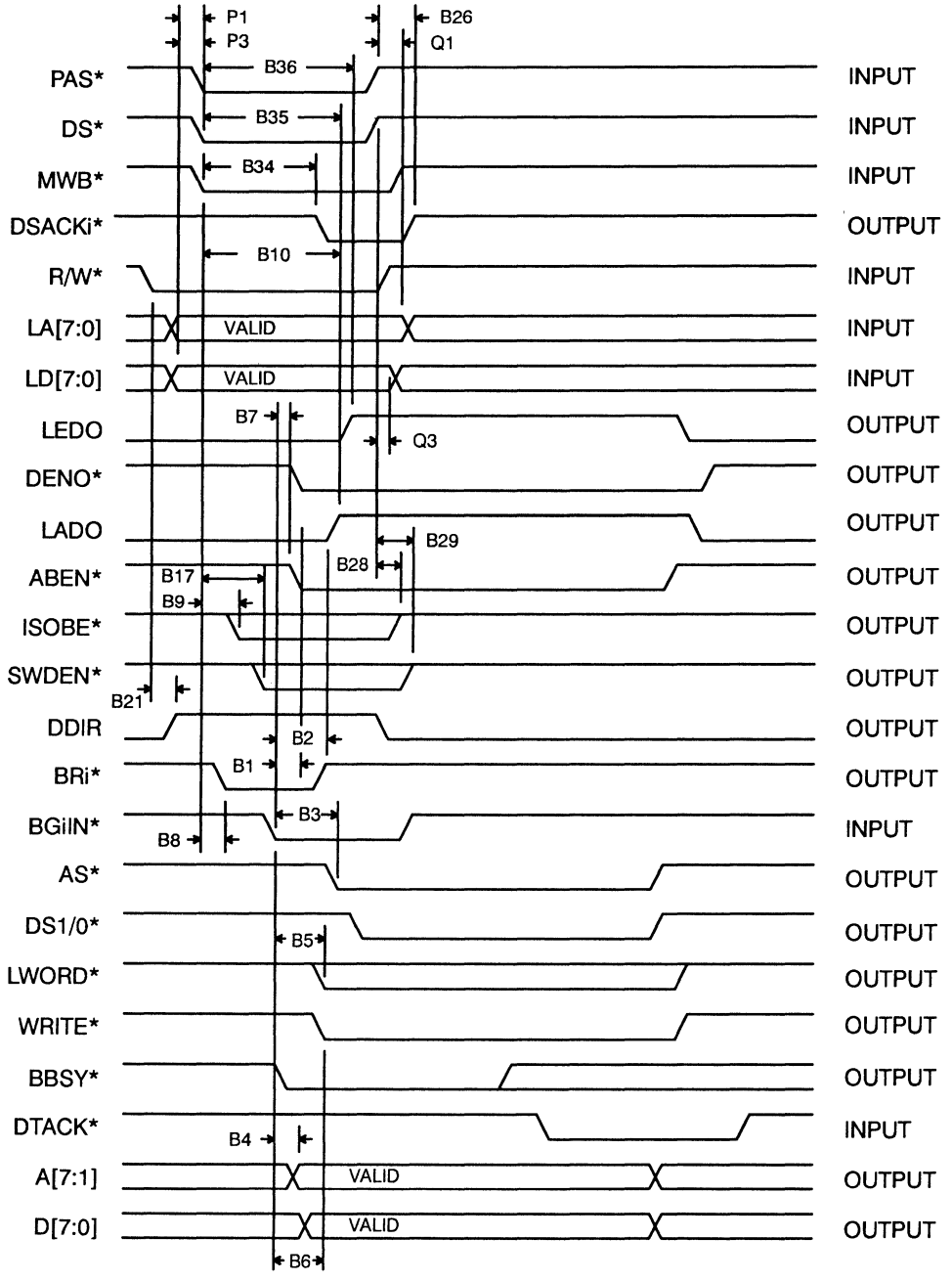


Figure 14–9. Master Write Post

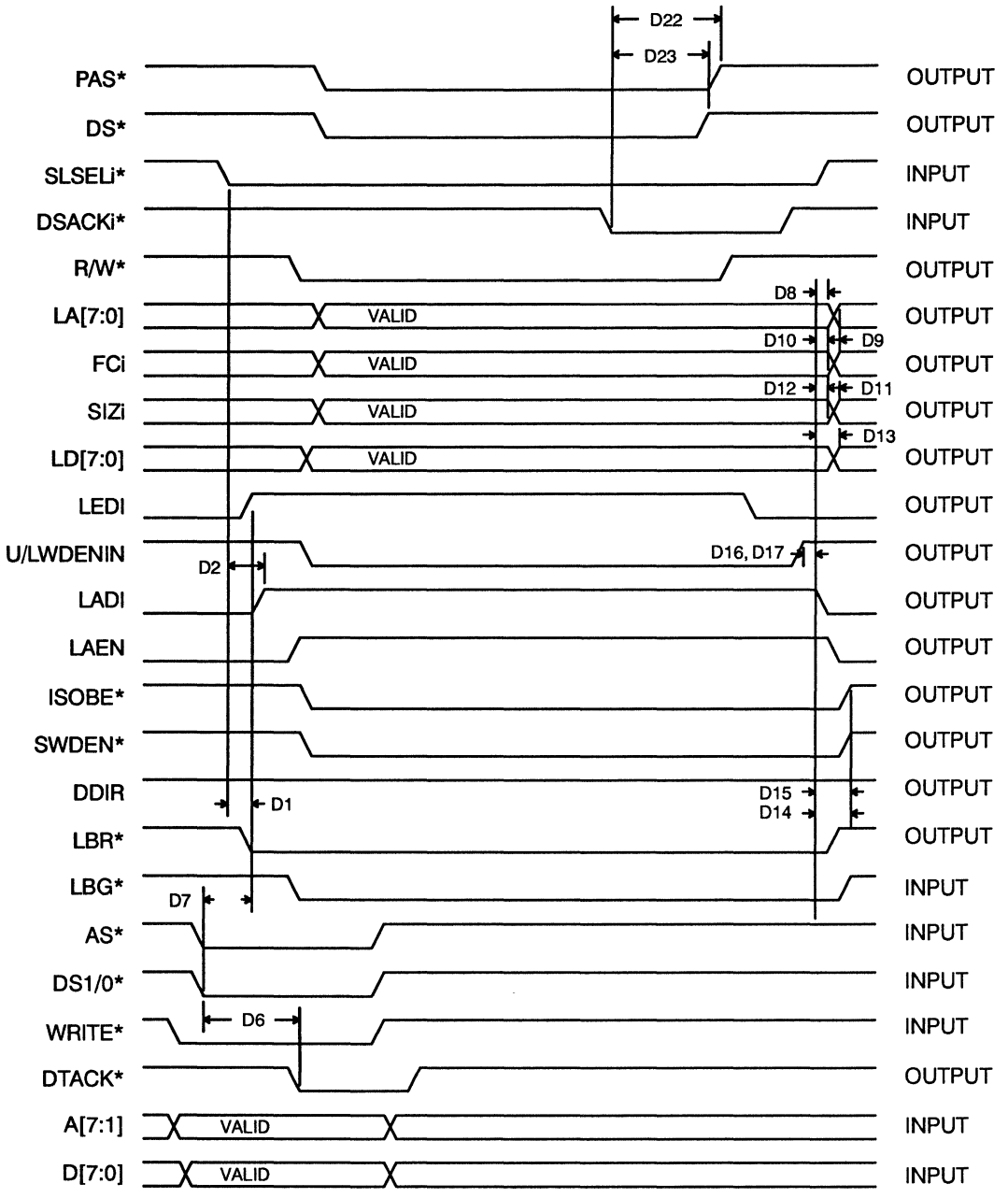


Figure 14–10. Slave Write Post

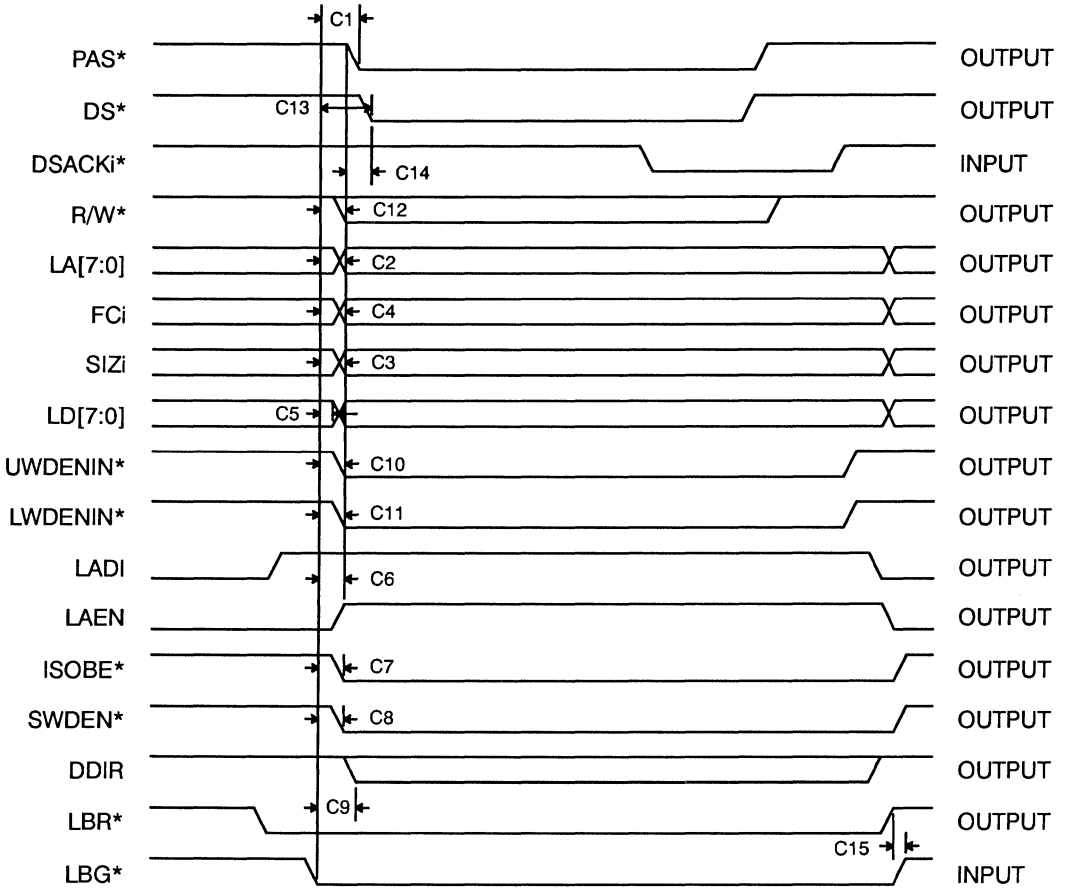


Figure 14–11. Local Bus

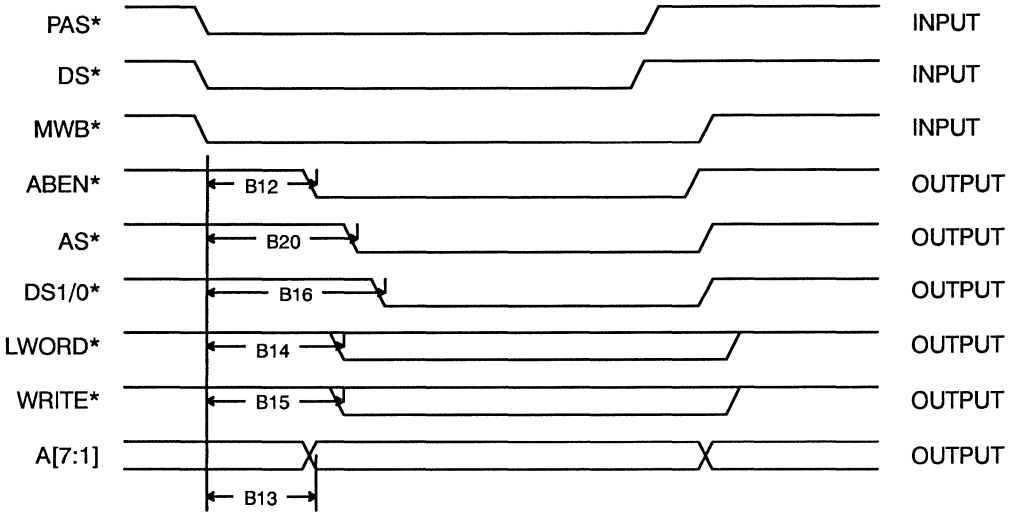


Figure 14–12. While VME Master

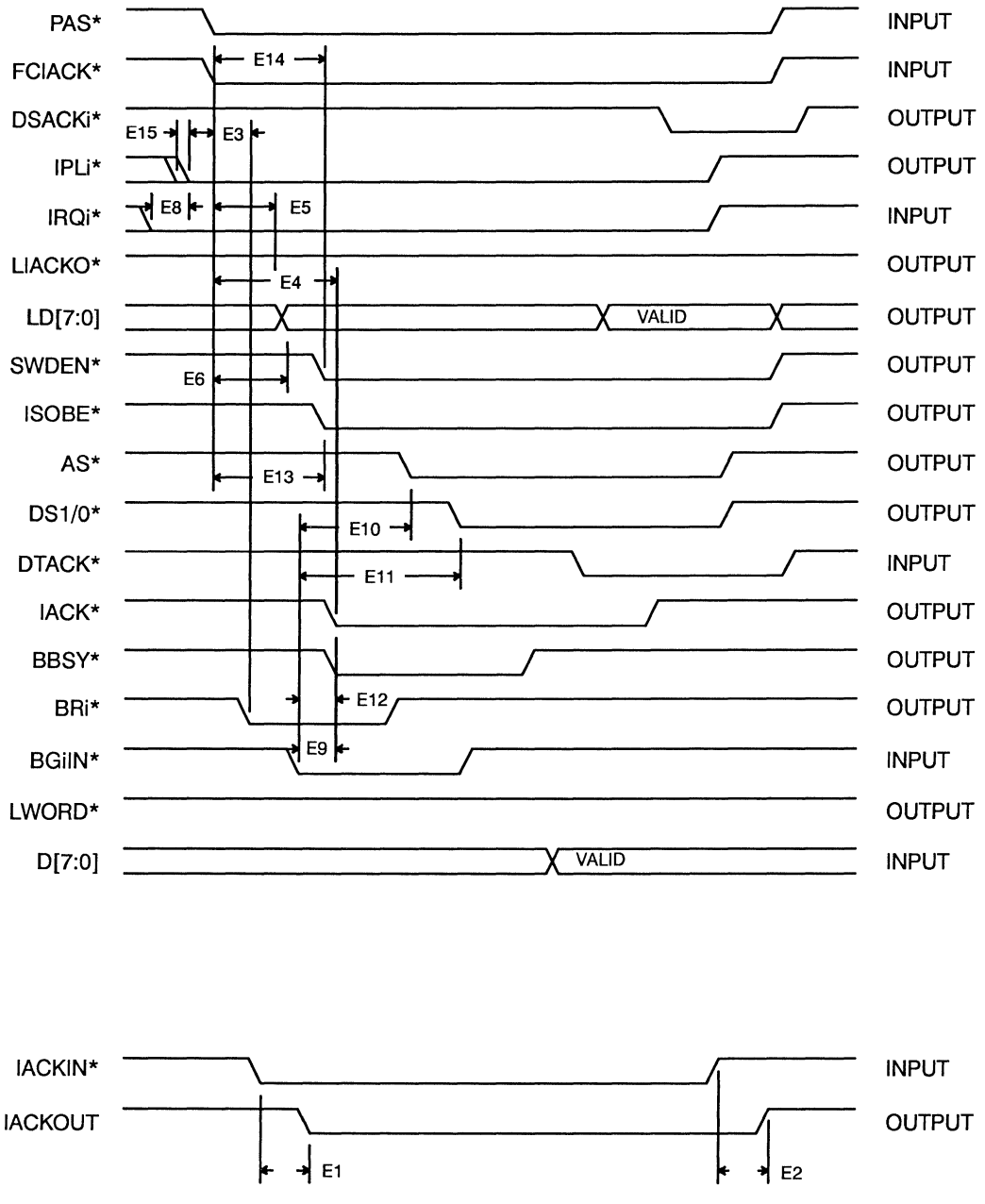


Figure 14–13. VME IACK

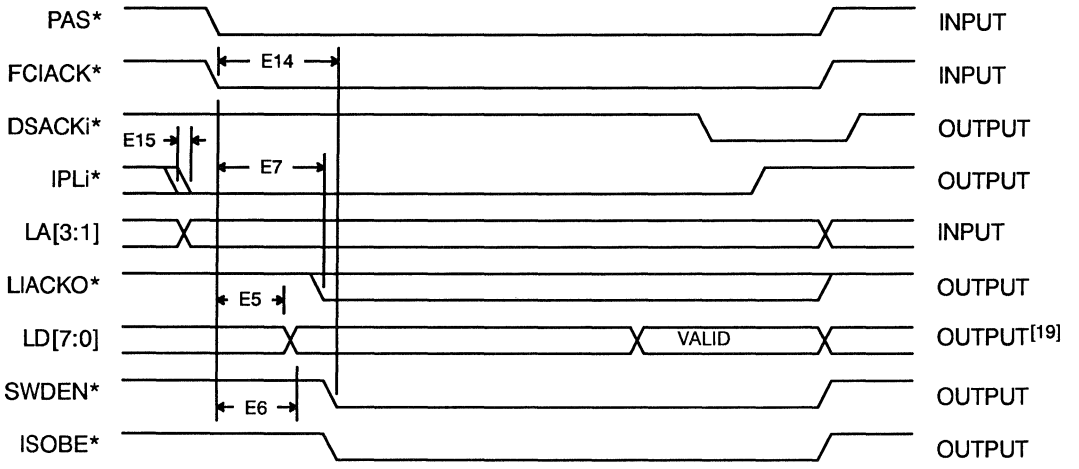


Figure 14–14. Local IACK

Note:

19. If VIC068A is configured to supply vector.

Initiation

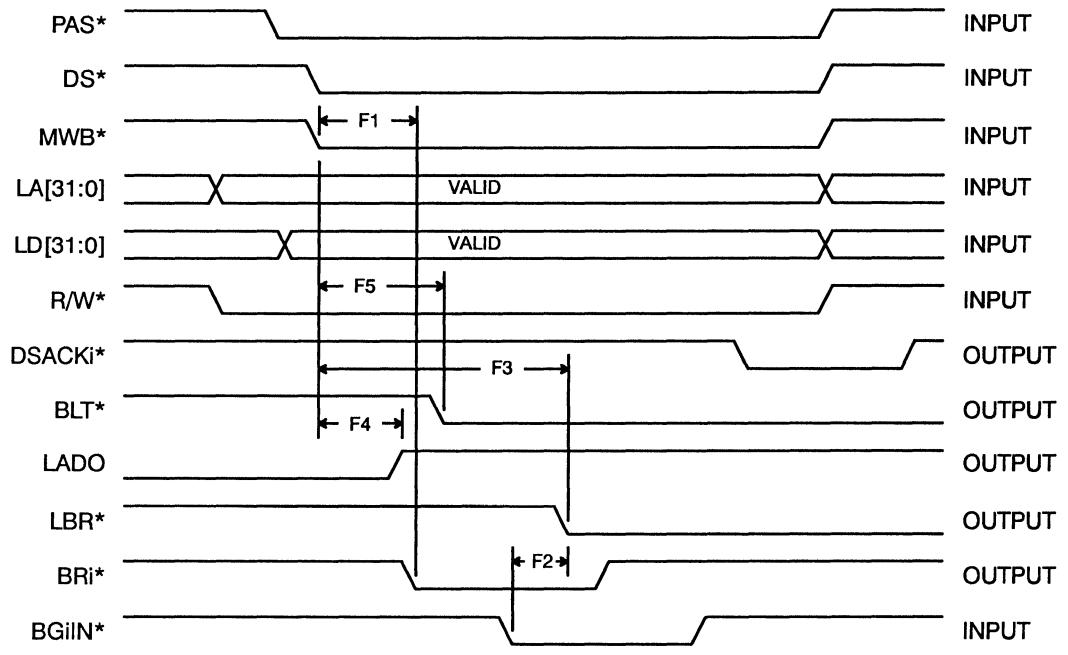


Figure 14–15. Initiation

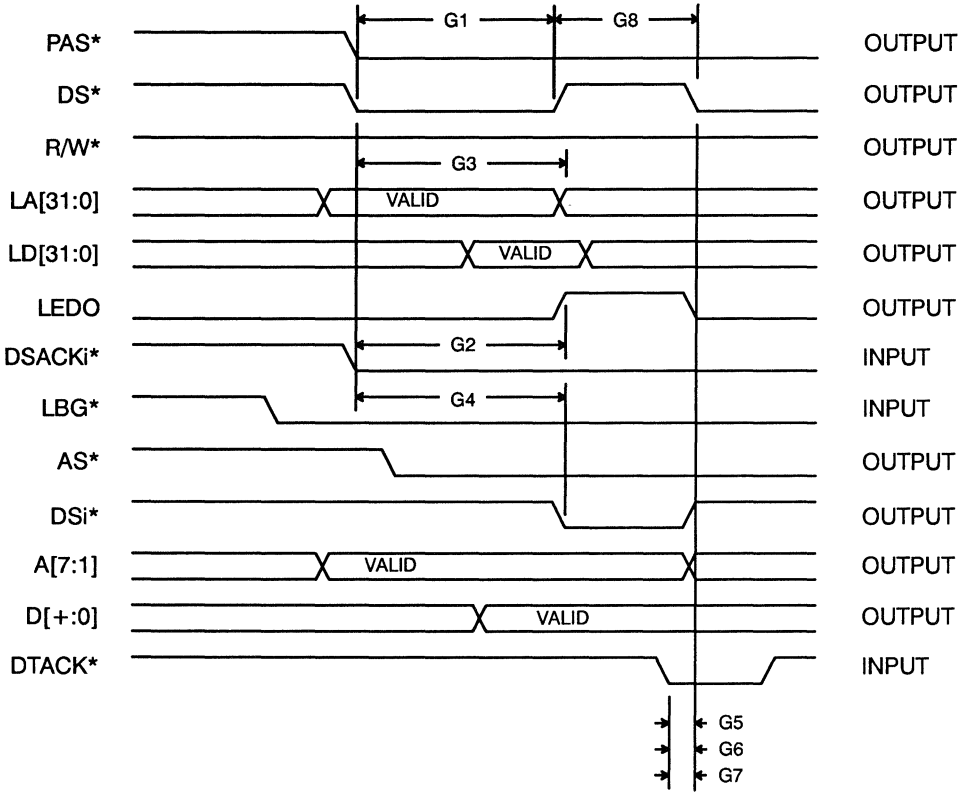


Figure 14–16. First MBLT Write

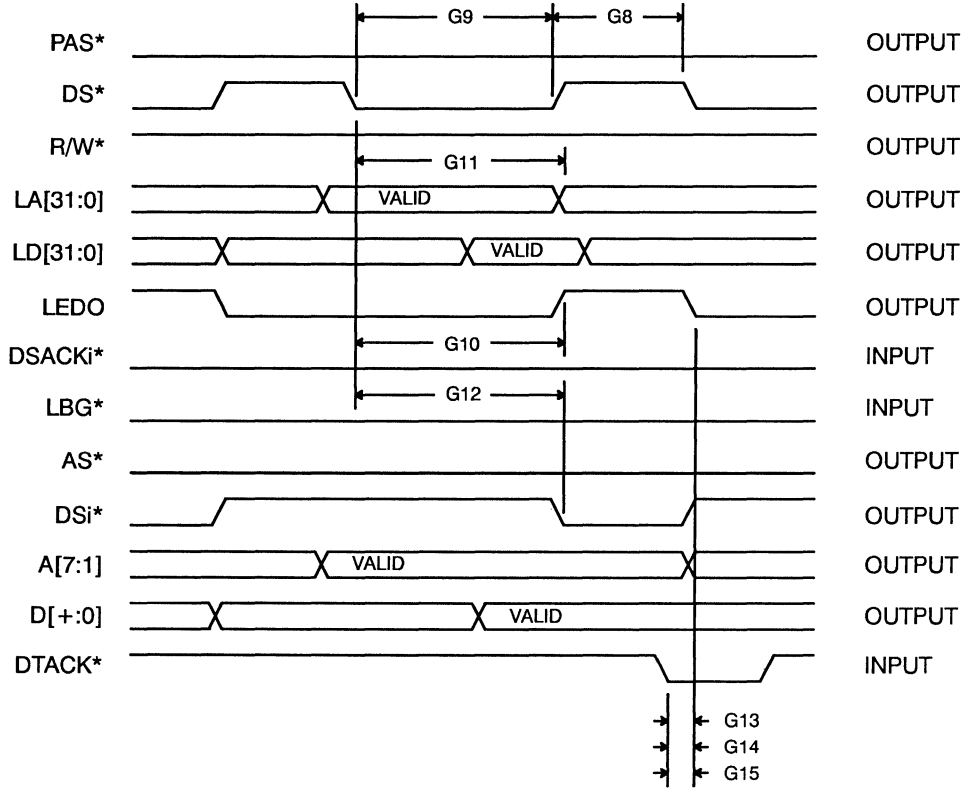


Figure 14–17. Second MBLT Write

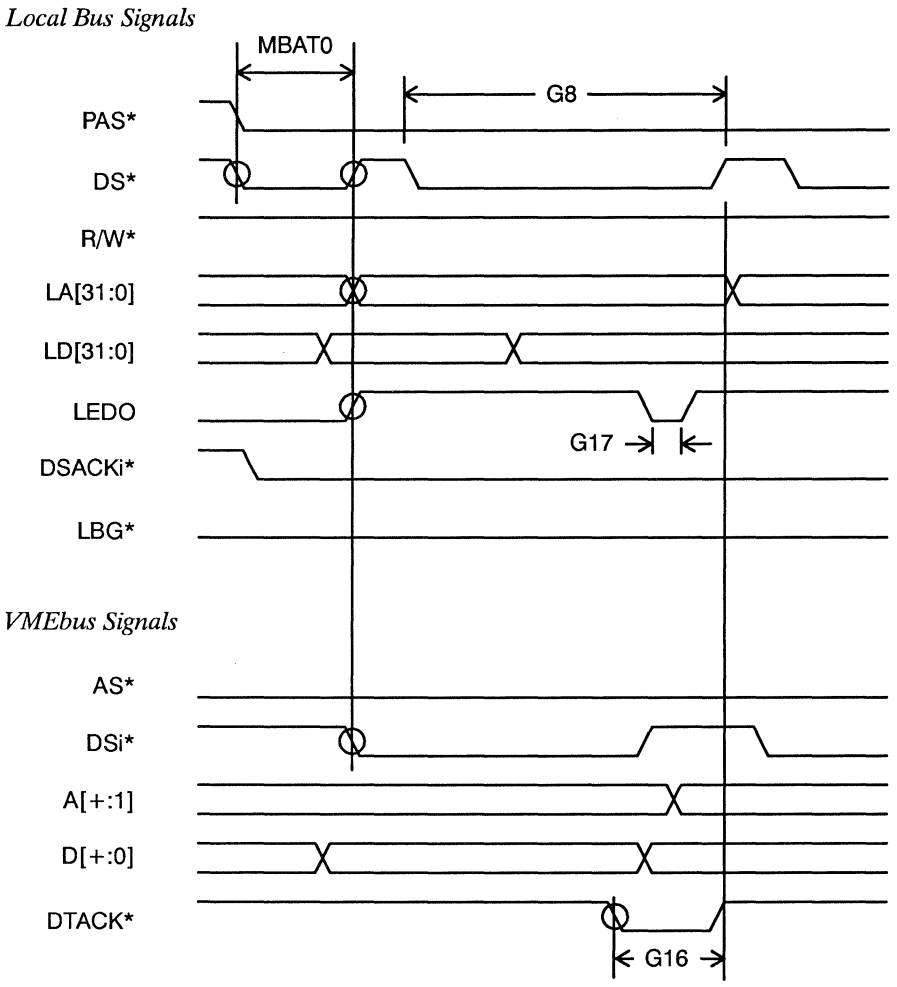


Figure 14–18. Master Block Transfer—Write (Slow Slave)

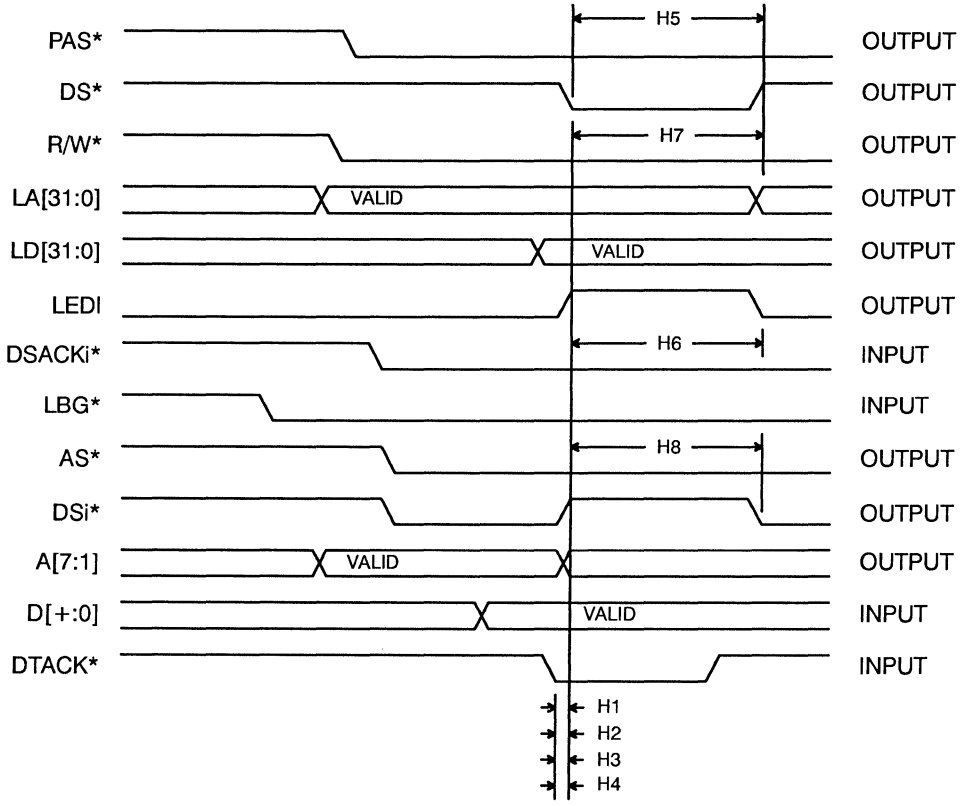


Figure 14–19. First MBLT Read

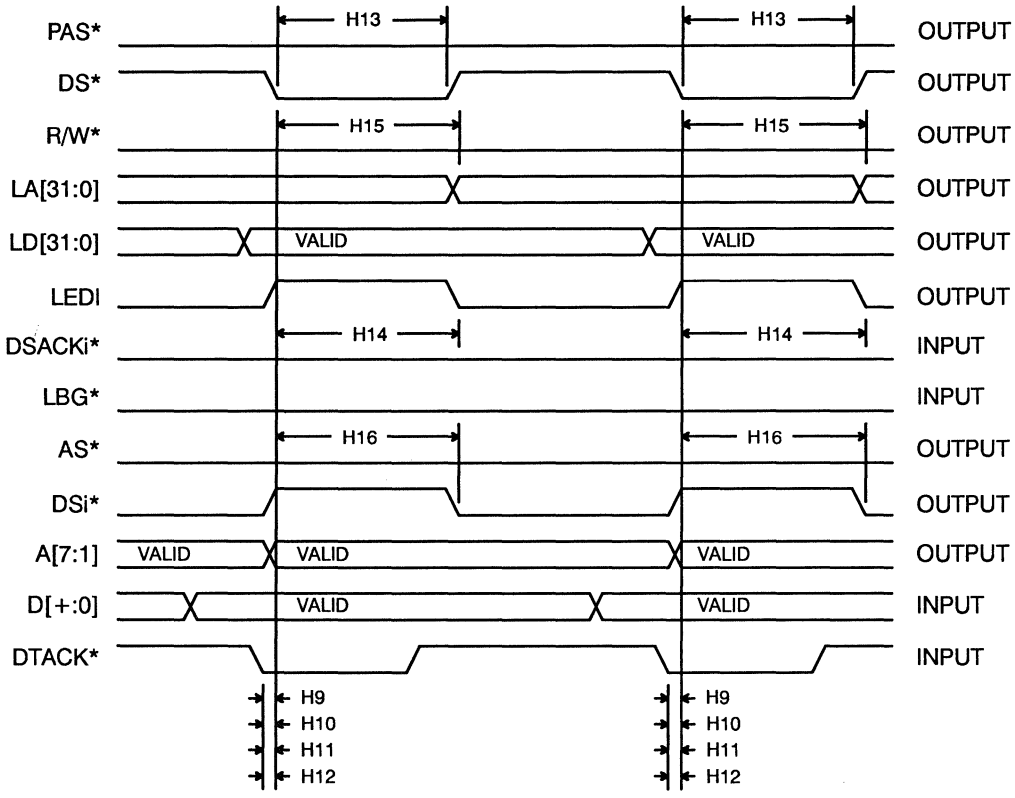
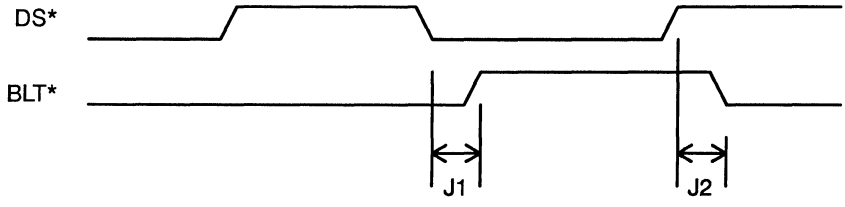


Figure 14–20. Second MBLT Read

Local Boundary Crossing



VMEbus Boundary Crossing

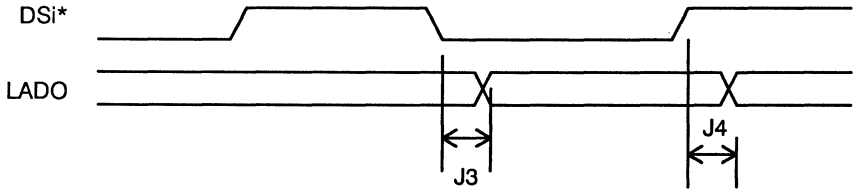


Figure 14–21. Boundary Crossing

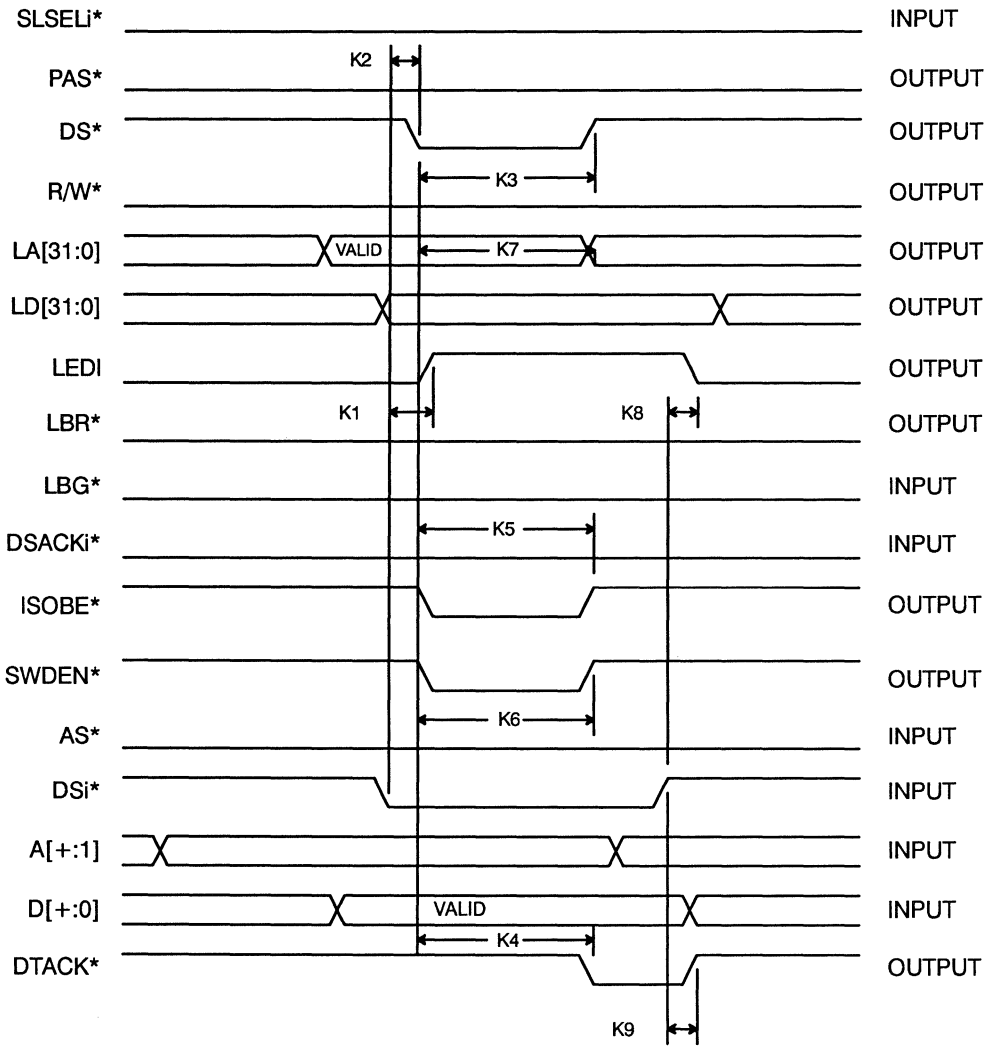


Figure 14–22. Slave Write BLT

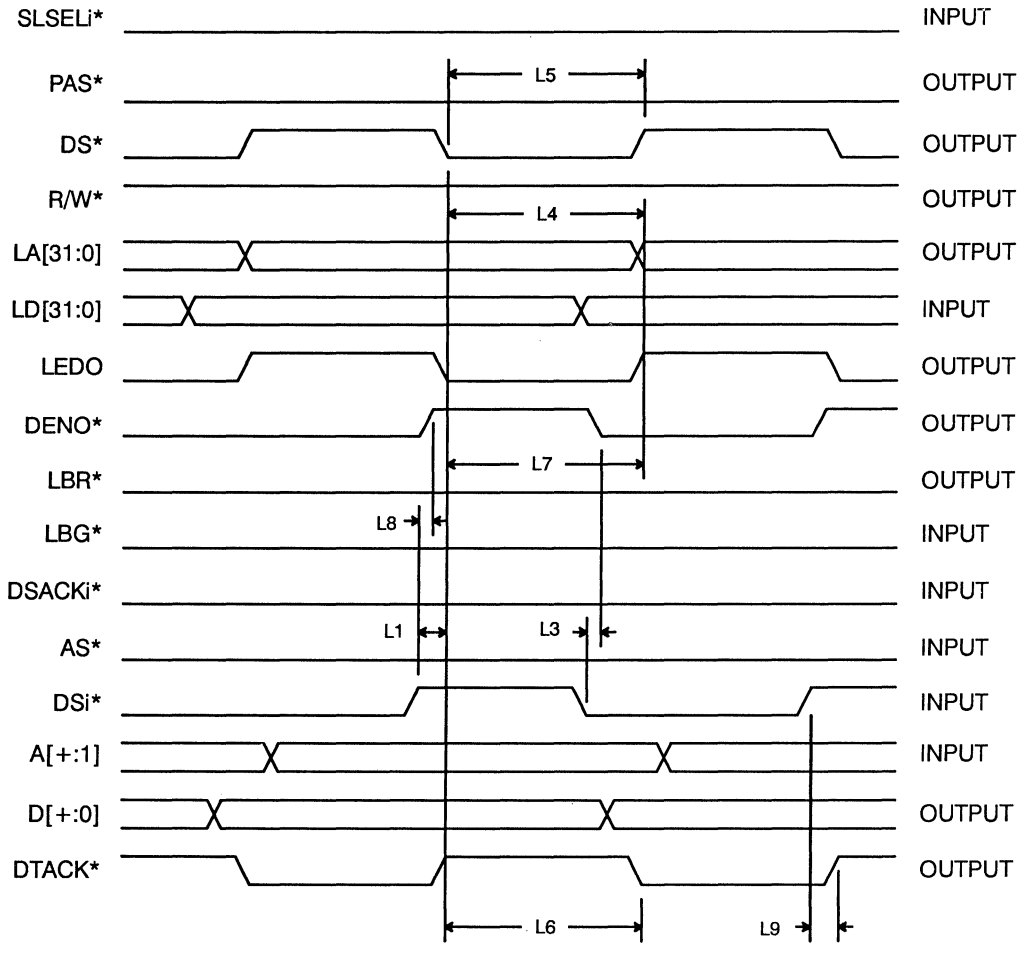


Figure 14–23. Slave Read BLT

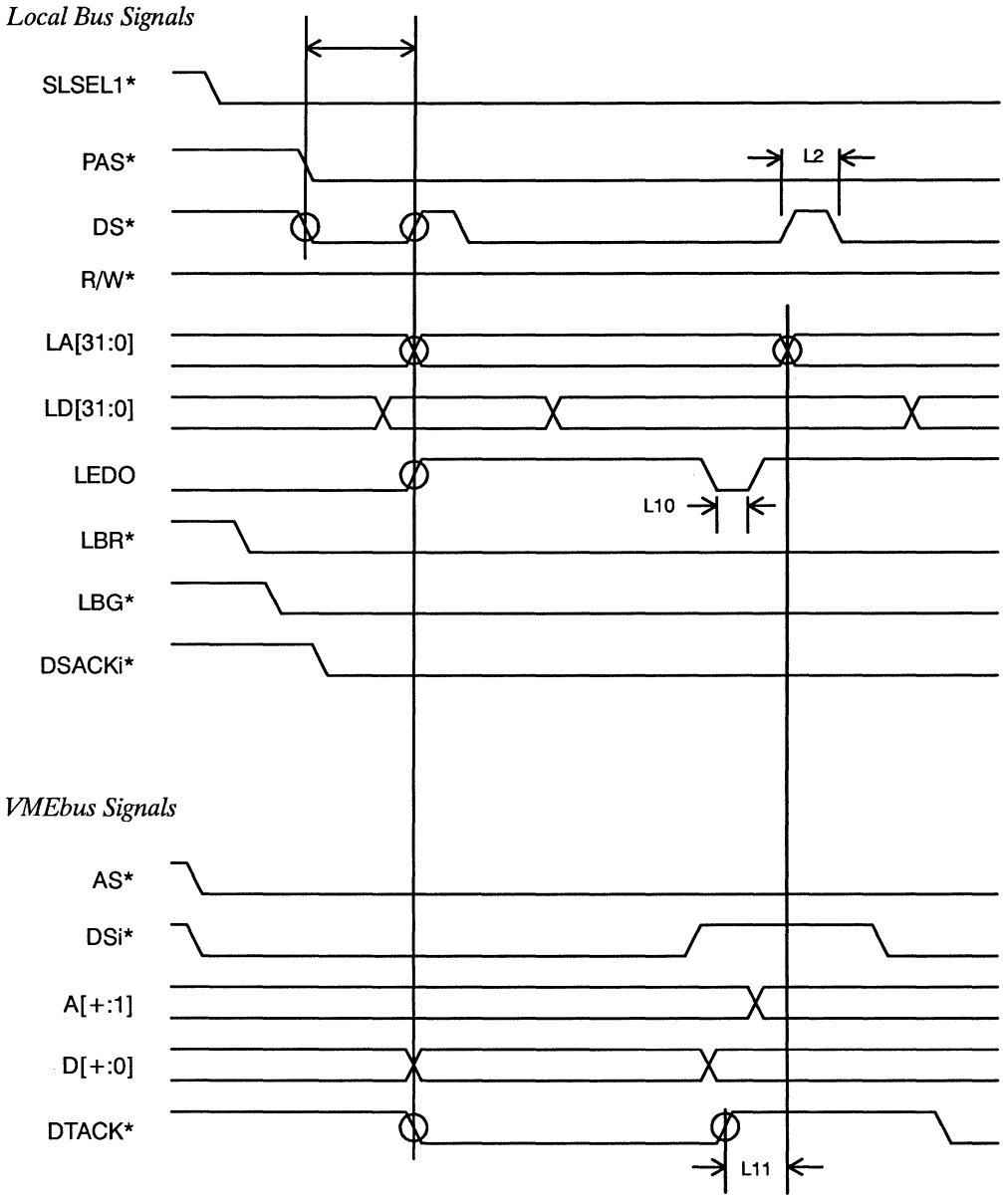
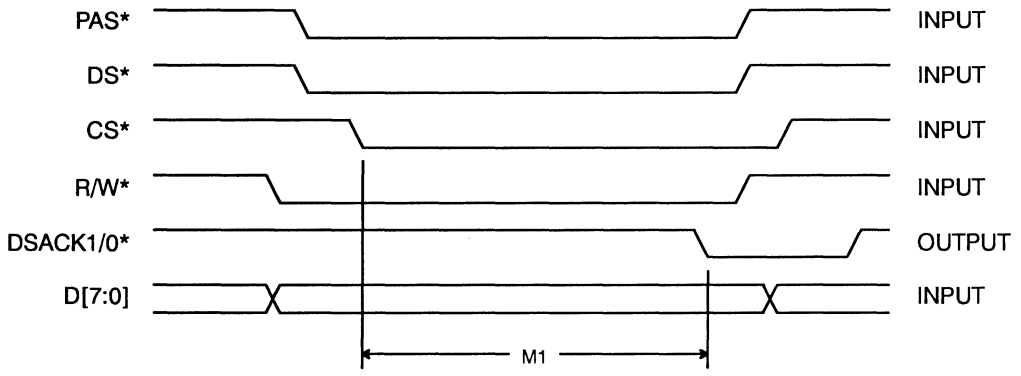
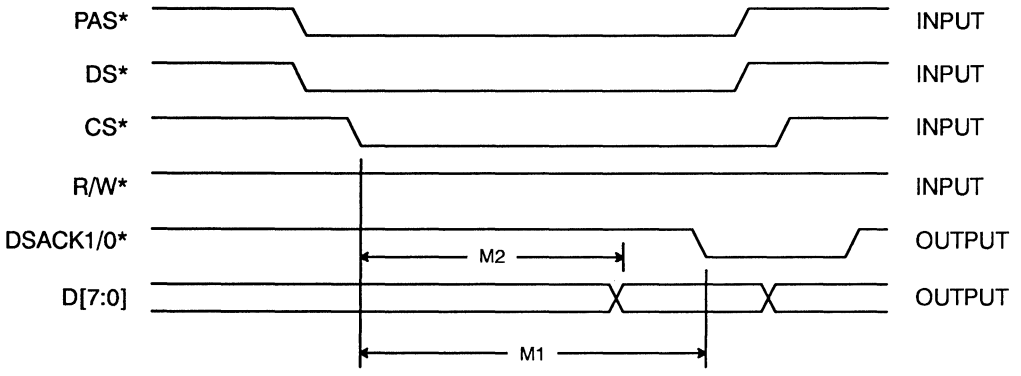


Figure 14–24. Slave Block Transfer—Read, Slow Master

Write



Read



ICF Select

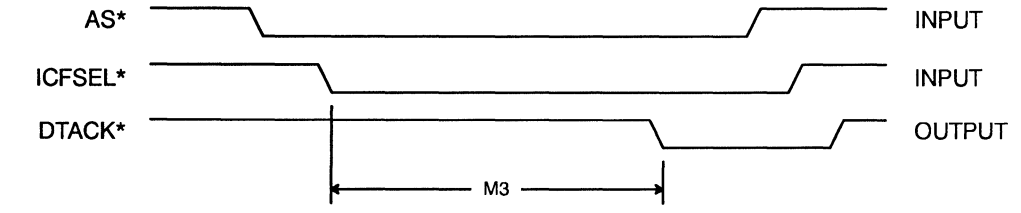


Figure 14–25. Register Operations

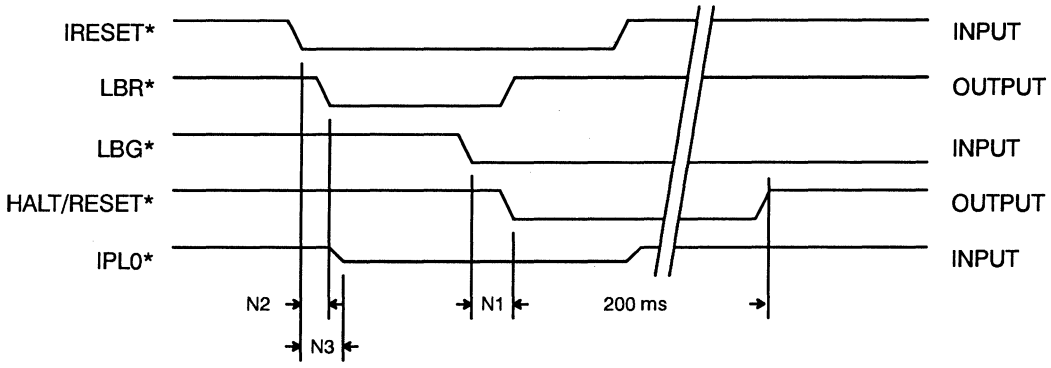


Figure 14–26. Global Reset

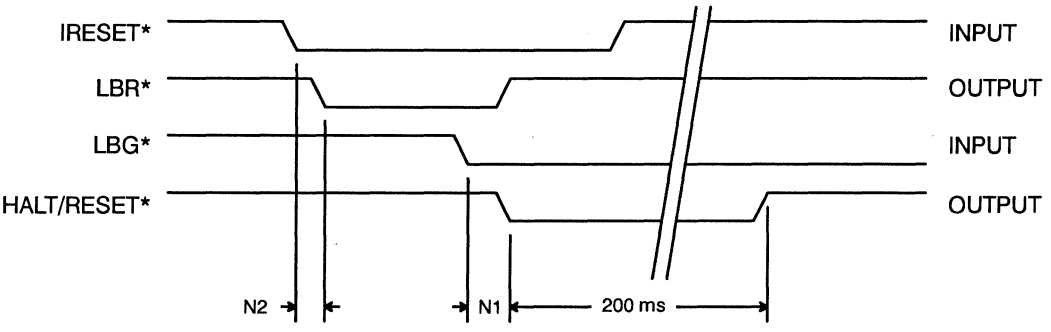


Figure 14–27. Internal Reset

15

VIC068A Signal List and Pinouts

Table 15–1. VMEbus Signals

Name	PGA Pin	QFP Pin	Type	Description
D07	H15	99	Three-State I/O	VMEbus Data
D06	H14	98	Three-State I/O	VMEbus Data
D05	J15	97	Three-State I/O	VMEbus Data
D04	K15	96	Three-State I/O	VMEbus Data
D03	J14	94	Three-State I/O	VMEbus Data
D02	L15	93	Three-State I/O	VMEbus Data
D01	K14	92	Three-State I/O	VMEbus Data
D00	K13	91	Three-State I/O	VMEbus Data
A07	N2	33	Three-State I/O	VMEbus Address
A06	L3	32	Three-State I/O	VMEbus Address
A05	M2	31	Three-State I/O	VMEbus Address
A04	M1	29	Three-State I/O	VMEbus Address
A03	L2	28	Three-State I/O	VMEbus Address
A02	L1	27	Three-State I/O	VMEbus Address
A01	K2	25	Three-State I/O	VMEbus Address
AM5	R10	63	Three-State I/O	VMEbus Address Modifier
AM4	R9	62	Three-State I/O	VMEbus Address Modifier
AM3	R8	59	Three-State I/O	VMEbus Address Modifier
AM2	P8	58	Three-State I/O	VMEbus Address Modifier
AM1	R7	57	Three-State I/O	VMEbus Address Modifier
AM0	R6	56	Three-State I/O	VMEbus Address Modifier
BG3IN*	N14	84	Input	VMEbus Bus Grant Input
BG3OUT*	M15	90	Output	VMEbus Bus Grant Output
BG2IN*	M13	83	Input	VMEbus Bus Grant Input
BG2OUT*	N15	88	Output	VMEbus Bus Grant Output
BG1IN*	P14	77	Input	VMEbus Bus Grant Input

Table 15–1. VMEbus Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
BG1OUT*	L13	87	Output	VMEbus Bus Grant Output
BG0IN*	N13	76	Input	VMEbus Bus Grant Input
BG0OUT*	M14	86	Output	VMEbus Bus Grant Output
BR3*	P13	73	Open Collector I/O	VMEbus Request
BR2*	N11	72	Open Collector I/O	VMEbus Request
BR1*	P12	71	Open Collector I/O	VMEbus Request
BR0*	R12	69	Open Collector I/O	VMEbus Request
IACK*	P6	52	Rescinding Three-State I/O	VMEbus Interrupt Acknowledge
IACKIN*	N6	51	Input	VMEbus Interrupt Acknowledge Input
IACKOUT*	R4	50	Output	VMEbus Interrupt Acknowledge Output
IRQ7*	R2	45	Open Collector I/O	VMEbus Interrupt Request
IRQ6*	P3	44	Open Collector I/O	VMEbus Interrupt Request
IRQ5*	N4	43	Open Collector I/O	VMEbus Interrupt Request
IRQ4*	R1	38	Open Collector I/O	VMEbus Interrupt Request
IRQ3*	P2	37	Open Collector I/O	VMEbus Interrupt Request
IRQ2*	N3	36	Open Collector I/O	VMEbus Interrupt Request
IRQ1*	M3	35	Open Collector I/O	VMEbus Interrupt Request
BBSY*	N12	75	Rescinding Three-State I/O	VMEbus Busy
BCLR*	R14	74	Three-State I/O	VMEbus Clear
AS*	P7	54	Rescinding Three-State I/O	VMEbus Address Strobe
DS1*	P11	68	Rescinding Three-State I/O	VMEbus DataStrobe
DS0*	R11	67	Rescinding Three-State I/O	VMEbus Data Strobe
DTACK*	R5	53	Rescinding Three-State I/O	VMEbus Data Transfer Acknowledge
BERR*	N10	66	Rescinding Three-State I/O	VMEbus Error
LWORD*	P9	64	Rescinding Three-State I/O	VMEbus Long–Word
WRITE*	P10	65	Rescinding Three-State I/O	VMEbus Data Direction
SYSCLK*	P15	85	Three-State Output	VMEbus Syatem Clock
SYSRESET*	P5	49	Open Collector I/O	VMEbus System Reset
ACFAIL*	R3	48	Input	VMEbus AC Fail
SYSFAIL*	N5	47	Open Collector I/O	VMEbus System Fail

Table 15–2. Local Signals

Name	PGA Pin	QFP Pin	Type	Description
LD7	C4	155	Three-State I/O	Local Data
LD6	A2	154	Three-State I/O	Local Data
LD5	B3	153	Three-State I/O	Local Data
LD4	C5	152	Three-State I/O	Local Data
LD3	B4	151	Three-State I/O	Local Data
LD2	A3	150	Three-State I/O	Local Data
LD1	A4	149	Three-State I/O	Local Data
LD0	B5	148	Three-State I/O	Local Data
LA7	A5	147	Three-State I/O	Local Address
LA6	C6	146	Three-State I/O	Local Address
LA5	B6	145	Three-State I/O	Local Address
LA4	B7	144	Three-State I/O	Local Address
LA3	A6	143	Three-State I/O	Local Address
LA2	A7	142	Three-State I/O	Local Address
LA1	A8	139	Three-State I/O	Local Address
LA0	B8	138	Three-State I/O	Local Address
SCON*	C13	116	Input	System Controller Enable
IRESET*	B14	117	Input	Internal Reset Input
RESET*	C10	131	Open Collector Output	Reset Output
HALT*	A12	130	Open Collector I/O	Halt Status
CLK64M	D13	115	Input	64-MHz Clock Input
PAS*	A10	136	Rescinding Three-State I/O	Physical/Processor Address Strobe
DS*	C9	137	Rescinding Three-State I/O	Processor Data Strobe
DSACK1*	B9	134	Three-State I/O	Data Size Acknowledge 1
DSACK0*	A11	133	Three-State I/O	Data Size Acknowledge 0
LBERR*	B10	132	Rescinding Three-State I/O	Local Bus Error
R/W*	B11	129	Rescinding Three-State I/O	Local Data Direction
RMC*	B12	126	Input	Read-Modify-Write
CS*	A9	137	Input	VIC068A Chip (Register) Select
SIZ1	A14	125	Rescinding Three-State I/O	Data Transfer Size 1
SIZ0	B13	124	Rescinding Three-State I/O	Data Transfer Size 0

Table 15–2. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
FC2	A13	128	Rescinding Three-State I/O	Function Code 2
FC1	C11	127	Rescinding Three-State I/O	Function Code 1
LBG*	A15	118	Input	Local Bus Grant
LBR*	C12	123	Output	Local Bus Request
IPL2*	B1	5	Output	Interrupt Priority Level 2
IPL1*	C2	4	Output	Interrupt Priority Level 1
IPL0*	D3	3	Three-State I/O	Interrupt Priority Level 0/Global Reset
BLT*	B2	157	Open Collector I/O	Block Transfer Status
DEDLK*	C3	156	Output	Deadlock Status
LIACKO*	C1	8	Output	Local Interrupt Autovector
LIRQ7*	G3	15	Input	Local Interrupt Request 7
LIRQ6*	G2	14	Input	Local Interrupt Request 6
LIRQ5*	E1	13	Input	Local Interrupt Request 5
LIRQ4*	F2	12	Input	Local Interrupt Request 4
LIRQ3*	F3	11	Input	Local Interrupt Request 3
LIRQ2*	D1	10	Three-State I/O (w/PU)	Local Interrupt Request 2
LIRQ1*	E2	40	Input	Local Interrupt Request 1
MWB*	J2	24	Input	Module Wants Bus
FCIACK*	K1	23	Input	Interrupt Acknowledge
WORD*	J1	22	Input	D16/D32 Control
SLSEL1*	H1	19	Input	Slave Select 1
SLSEL0*	J3	21	Input	Slave Select 0
ICFSEL*	H2	18	Input	ICF Select
ASIZ1*	F1	16	Input	Address Size 1
ASIZ0*	G1	17	Input	Address Size 0

Table 15–3. Buffer Control Signals

Name	PGA Pin	QFP Pin	Type	Description
ABEN*	B15	114	Output	VMEbus Address Buffer Enable
LADO	C14	113	Output	Latch Outgoing VMEbus Address
LADI	E13	112	Output	Latch Incomming VMEbus Address
LEDO	D15	109	Output	Latch Outgoing VMEbus Data
LEDI	D14	111	Output	Latch Incomming VMEbus Data
DDIR	E14	108	Output	Local Data Direction
UWDENIN*	E15	107	Output	Local Data Enable In (Upper Word)
LWDENIN*	F14	105	Output	Local Data Enable In (Lower Word)
SWDEN*	F15	103	Output	Swap Local Data Enable
DENO*	G14	104	Output	VMEbus Data Buffer Enable
ISOBE*	G15	102	Output	Isolation Buffer Enable
LAEN*	E3	7	Output	Local Address Buffer Enable

Table 15–4. Power Supplies[1]

Name	PGA Pin	Type	Description
V_{SS} Core	H3	Ground	Ground
V_{SS}	K3	Ground	Ground
V_{SS}	P1	Ground	Ground
V_{SS}	N7	Ground	Ground
V_{SS}	N8	Ground	Ground
V_{SS}	R13	Ground	Ground
V_{SS}	R15	Ground	Ground
V_{SS}	L14	Ground	Ground
V_{SS}	H13	Ground	Ground
V_{SS}	F13	Ground	Ground
V_{SS}	C7	Ground	Ground
V_{SS}	A1	Ground	Ground
V_{CC} Core	G13	Power	+5 Volts DC
V_{CC}	D2	Power	+5 Volts DC
V_{CC}	N1	Power	+5 Volts DC
V_{CC}	P4	Power	+5 Volts DC
V_{CC}	N9	Power	+5 Volts DC
V_{CC}	J13	Power	+5 Volts DC
V_{CC}	C15	Power	+5 Volts DC
V_{CC}	C8	Power	+5 Volts DC

Note:

1. For QFP power supply signals, see *Table 15–5*.

Table 15–5. Pinout for VIC068A Plastic and Ceramic Quad Flatpack (160-Pin): Cavity Up

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{SS}	32	A06	63	AM5	94	D03
2	V_{SS}	33	A07	64	LWORD*	95	V_{DD}
3	IPL0*	34	V_{SS}	65	WRITE*	96	D04
4	IPL1*	35	IRQ1*	66	BERR*	97	D05
5	IPL2*	36	IRQ2*	67	DS0*	98	D06
6	V_{DD}	37	IRQ3*	68	DS1*	99	D07
7	LAEN	38	IRQ4*	69	BR0*	100	V_{SS}
8	LIACKO*	39	V_{SS}	70	V_{SS}	101	V_{DD} CORE
9	LIRQ1*	40	V_{SS}	71	BR1*	102	ISOBE*
10	LIRQ2*	41	V_{DD}	72	BR2*	103	SWDEN*
11	LIRQ3*	42	V_{DD}	73	BR3*	104	DENO*
12	LIRQ4*	43	IRQ5*	74	BCLR*	105	LWDENIN*
13	LIRQ5*	44	IRQ6*	75	BBSY*	106	V_{SS}
14	LIRQ6*	45	IRQ7*	76	BGIN0*	107	UWDENIN*
15	LIRQ7*	46	V_{DD}	77	BGIN1*	108	DDIR*
16	ASIZ1*	47	SYSFAIL*	78	V_{SS}	109	LEDO
17	ASIZ0*	48	ACFAIL*	79	V_{DD}	110	V_{DD}
18	ICFSEL*	49	SYSRESET*	80	V_{DD}	111	LEDI
19	SLSEL1*	50	IACKOUT*	81	V_{SS}	112	LADI
20	V_{SS} CORE	51	IACKIN*	82	V_{SS}	113	LADO
21	SLSELO*	52	IACK*	83	BGIN2*	114	ABEN*
22	WORD*	53	DTACK*	84	BGIN3*	115	CLK64M
23	FCIACK*	54	AS*	85	SYSCLK	116	SCON*
24	MWB*	55	V_{SS}	86	BGOUT0*	117	IRESET*
25	A01	56	AM0	87	BGOUT1*	118	LBG*
26	V_{SS}	57	AM1	88	BGOUT2*	119	V_{SS}
27	A02	58	AM2	89	V_{SS}	120	V_{SS}
28	A03	59	AM3	90	BGOUT3*	121	V_{DD}
29	A04	60	V_{SS}	91	D00	122	V_{DD}
30	V_{DD}	61	V_{DD}	92	D01	123	LBR*
31	A05	62	AM4	93	D02	124	SIZ0

Table 15–5. Pinout for VIC068A Plastic and Ceramic Quad Flatpack: Cavity Up (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
125	SIZ1	134	DSACK1*	143	LA3	152	LD4
126	RMC*	135	DS*	144	LA4	153	LD5
127	FC1	136	PAS*	145	LA5	154	LD6
128	FC2	137	CS*	146	LA6	155	LD7
129	R/W	138	LA0	147	LA7	156	DEDLK*
130	HALT*	139	LA1	148	LD0	157	BLT*
131	RESET*	140	V_{DD}	149	LD1	158	V_{SS}
132	LBERR*	141	V_{SS}	150	LD2	159	V_{DD}
133	DSACK0*	142	LA2	151	LD3	160	V_{DD}

A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	
VSS	IPL2*	LIACKO*	LIRQ2*	LIRQ5*	ASIZ1*	ASIZ0*	SLSEL1*	WORD	FIACK*	A02	A04	VDD	VSS	IRQ4*	1
LD6	BLT*	IPL1*	VDD	LIRQ1*	LIRQ4*	LIRQ6*	ICFSEL*	MWB*	A01	A03	A05	A07	IRQ3*	IRQ7*	2
LD2	LD5	DEDLK*	IPL0*	LAEN	LIRQ3*	LIRQ7*	VSS	SLSEL0*	VSS	A06	IRQ1*	IRQ2*	IRQ6*	ACFAIL*	3
LD1	LD3	LD7	LOCATOR PIN									IRQ5*	VDD	IACKOUT*	4
LA7	LD0	LD4										SYSFAIL*	SYSRESET*	DTACK*	5
LA3	LA5	LA6										IACKIN*	IACK*	AM0	6
LA2	LA4	VSS										VSS	AS*	AM1	7
LA1	LA0	VCC7										VSS	AM2	AM3	8
CS*	DSACK1*	$\overline{DS}$										VDD	LWORD*	AM4	9
PAS*	LBERR*	RESET*										BERR*	WRITE*	AM5	10
DSACK0*	R/W*	FC1										BR2*	DS1*	DS0*	11
HALT*	RMC*	LBR*										BBSY*	BR1*	BR0*	12
FC2	SIZ0	SCON*	CLK64M	LADI	VSS	VDD	VSS8	VCC5	D00	BG1OUT*	BG2IN*	BG0IN*	$\overline{BR}$3	VSS	13
SIZ1	IRESET*	LADO	LEDI*	DDIR*	LWDENIN*	DENO*	D06	D03	D01	VSS7	BG0OUT*	BG3IN*	BG1IN*	BCLR*	14
LBG*	ABEN*	VDD	LEDO	UWDENIN*	SWDEN*	ISOBE*	D07	D05	D04	D02	BG3OUT*	BG2OUT*	SYSCLK	VSS	15

Figure 15–1. VIC068A Pin Grid Array (PGA), Bottom View

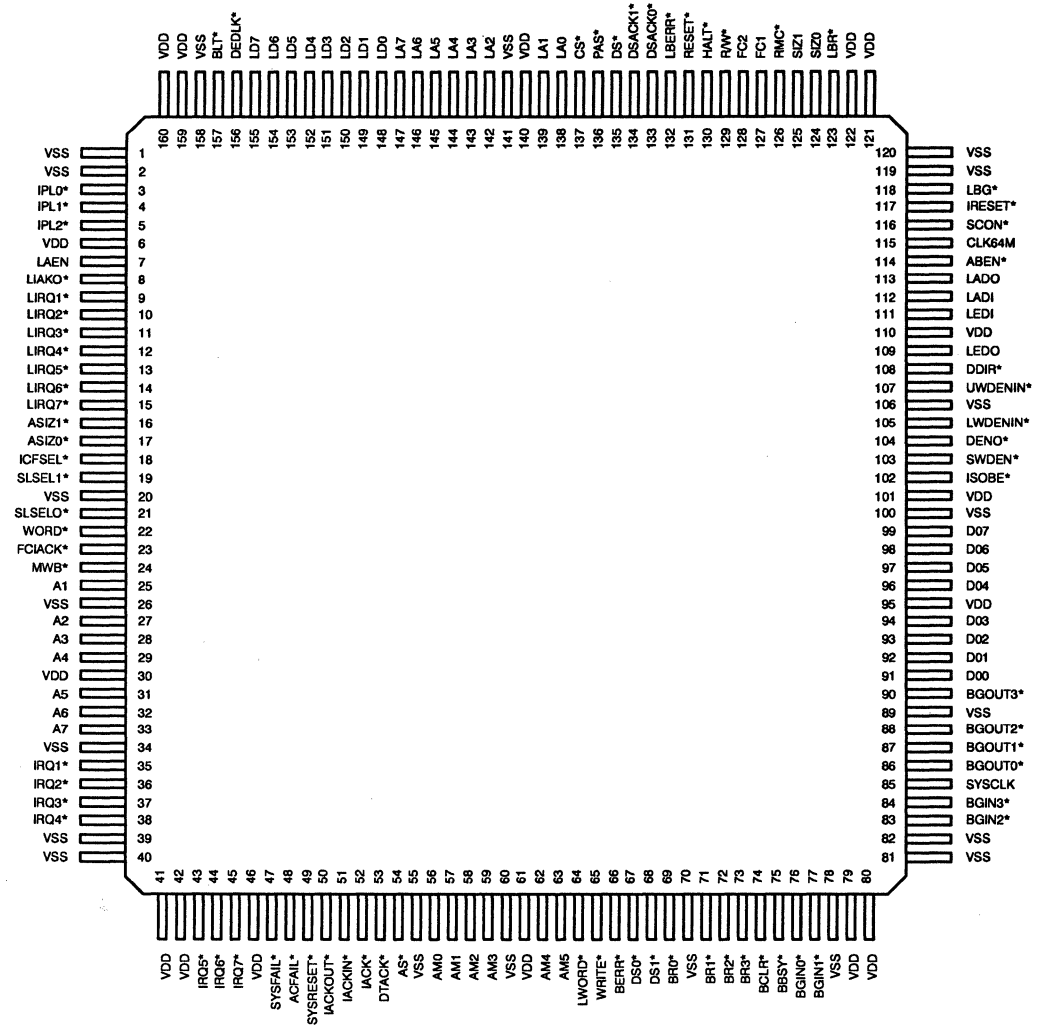


Figure 15–2. VIC068A Quad Flatpack (QFP), Top View

16

VIC068A Simulation Waveforms

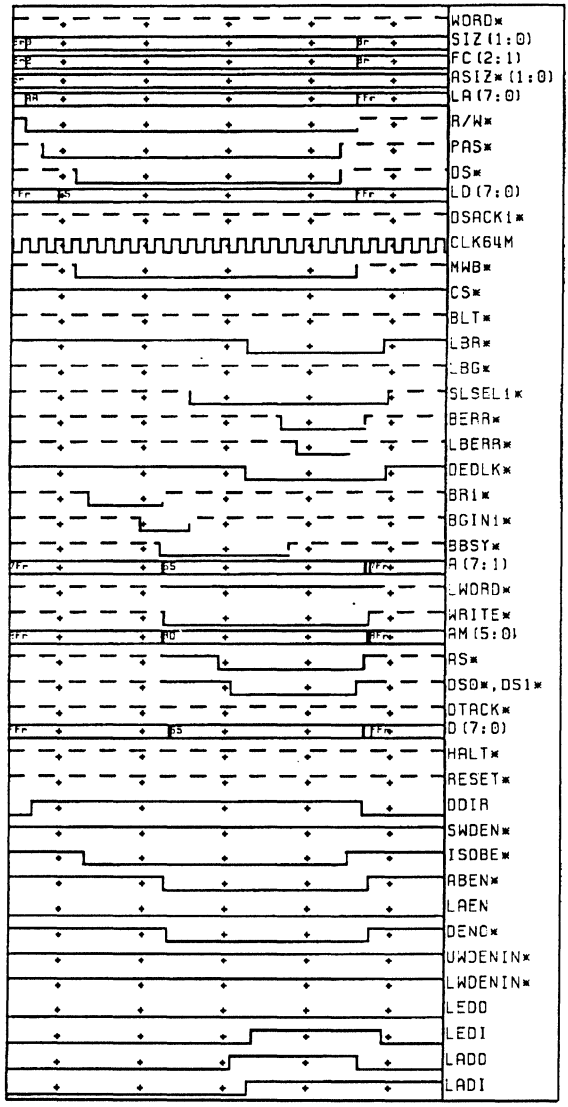


Figure 16–1. Master Self-Access

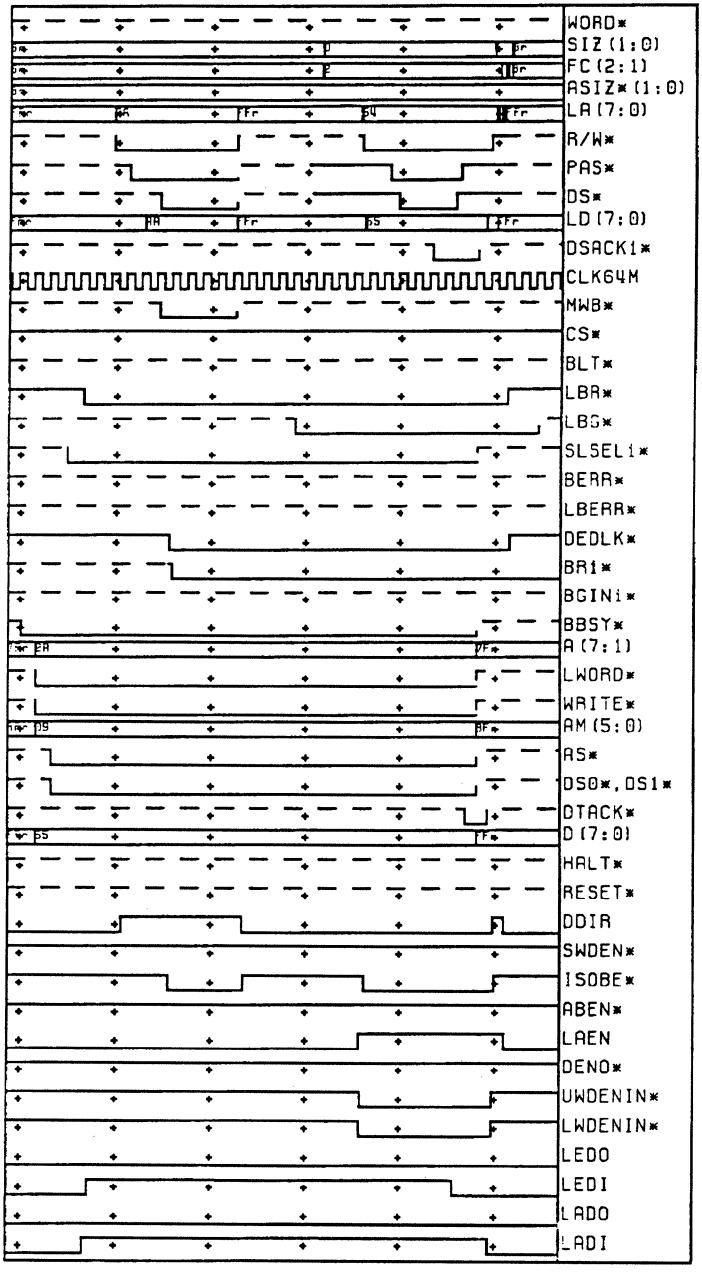


Figure 16–2. Master Deadlock Operation

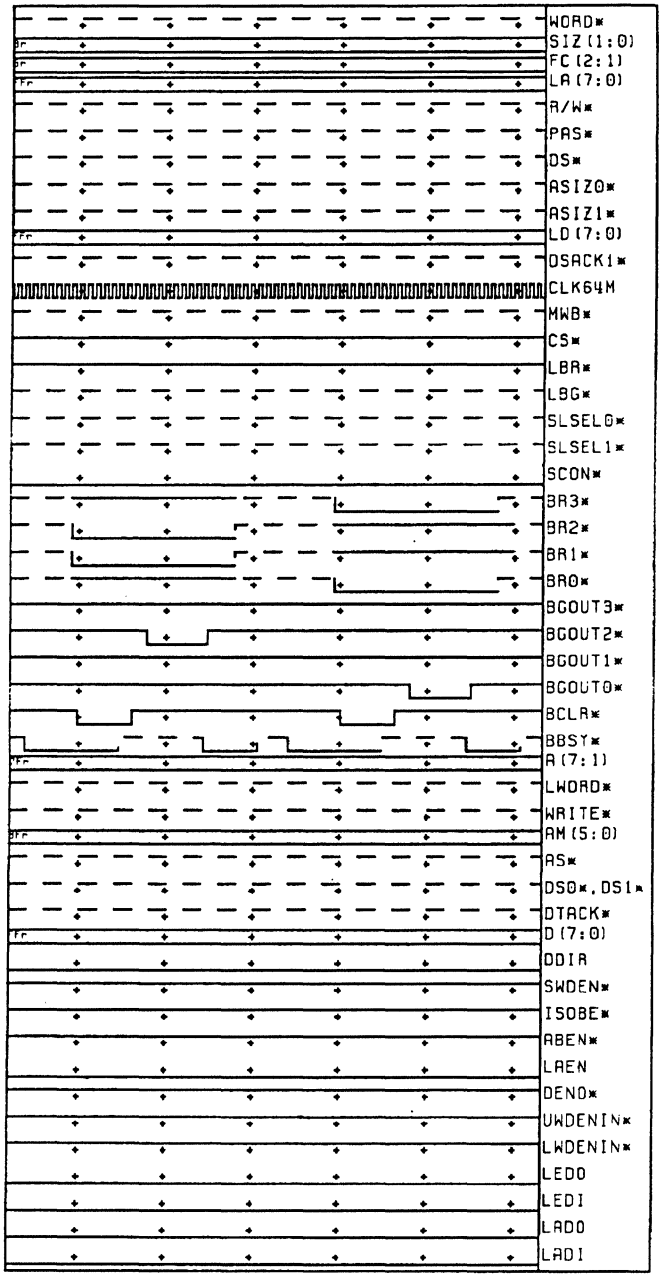


Figure 16–3. Arbitration Round Robin

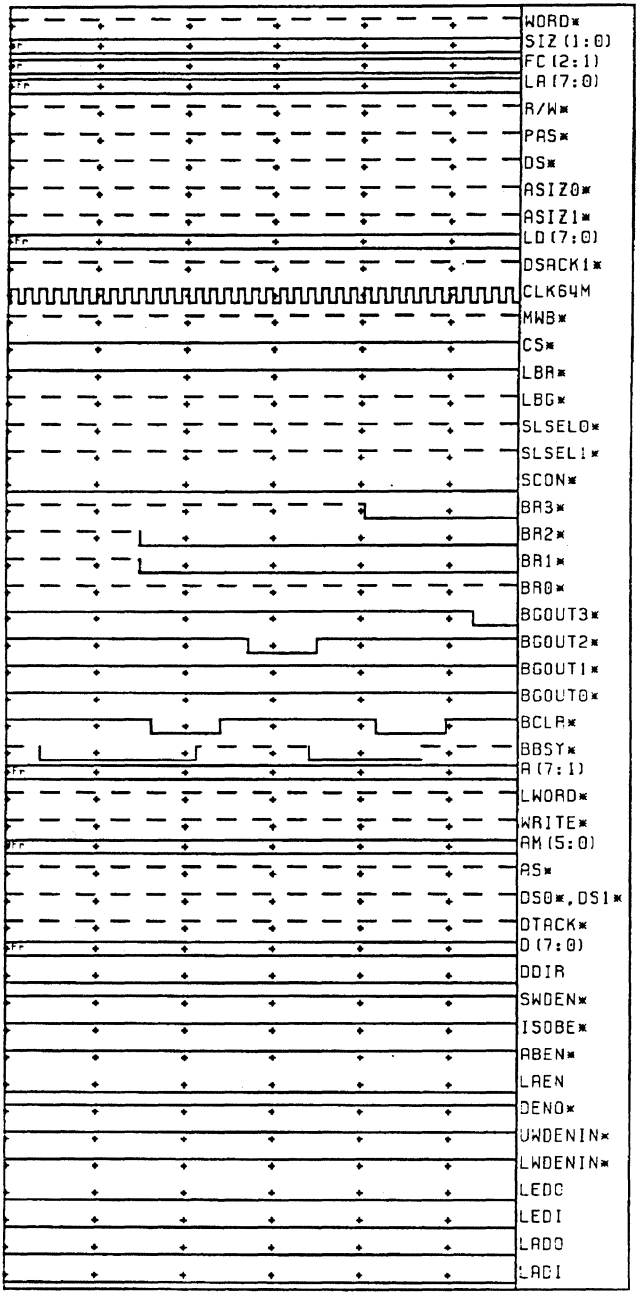


Figure 16–4. Arbitration Priority Mode

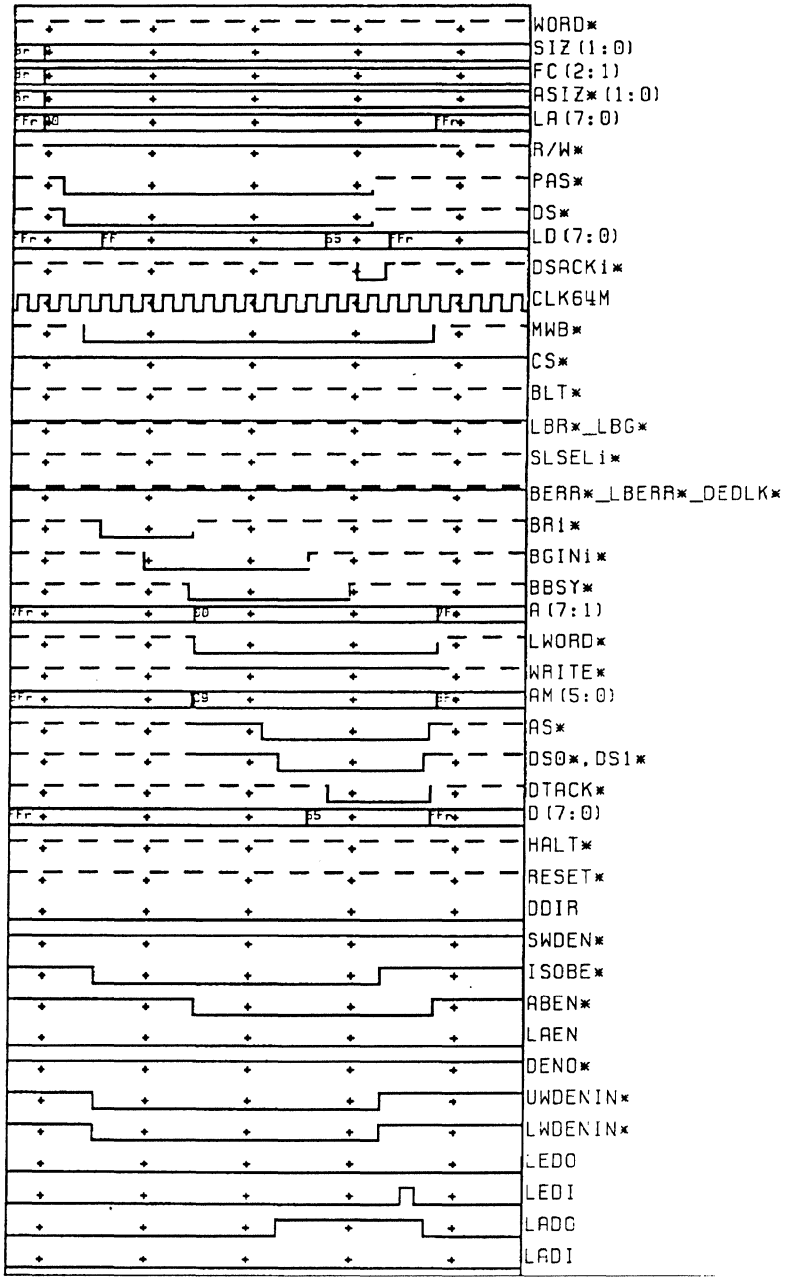


Figure 16–5. Master Read

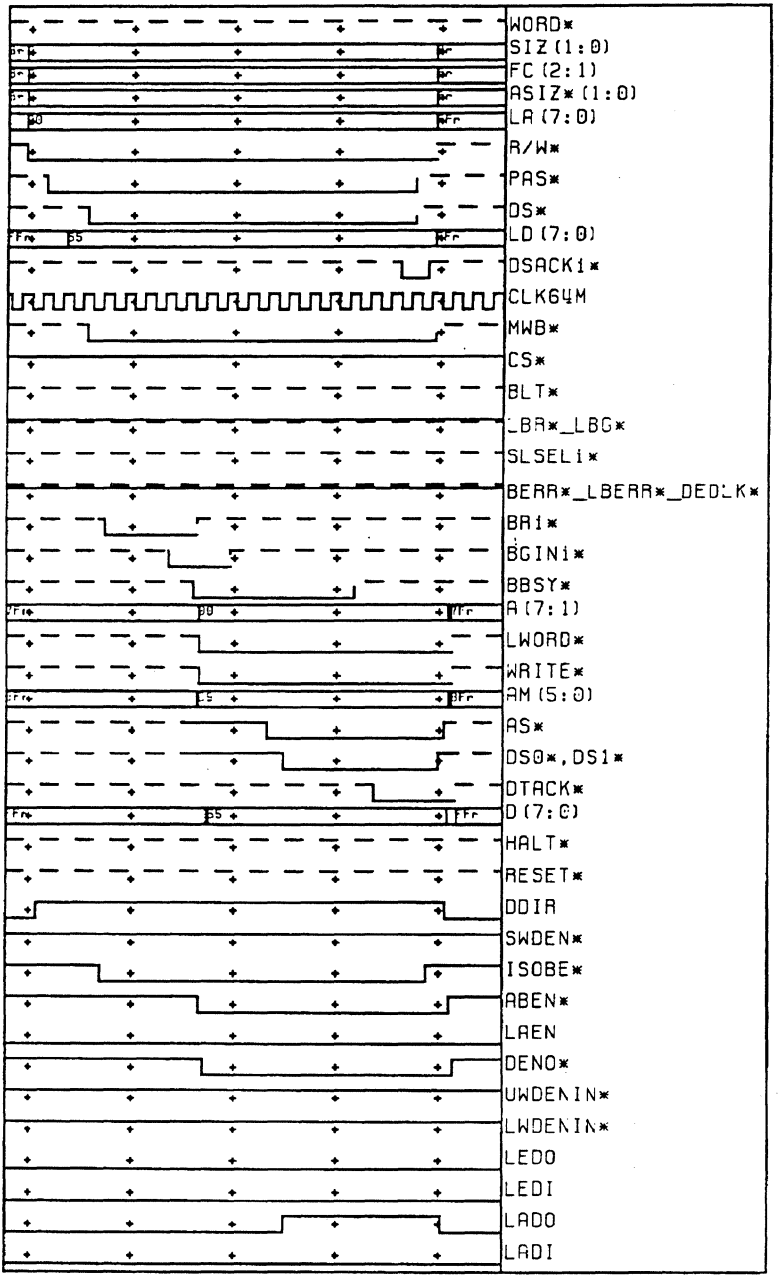


Figure 16–6. Master Write

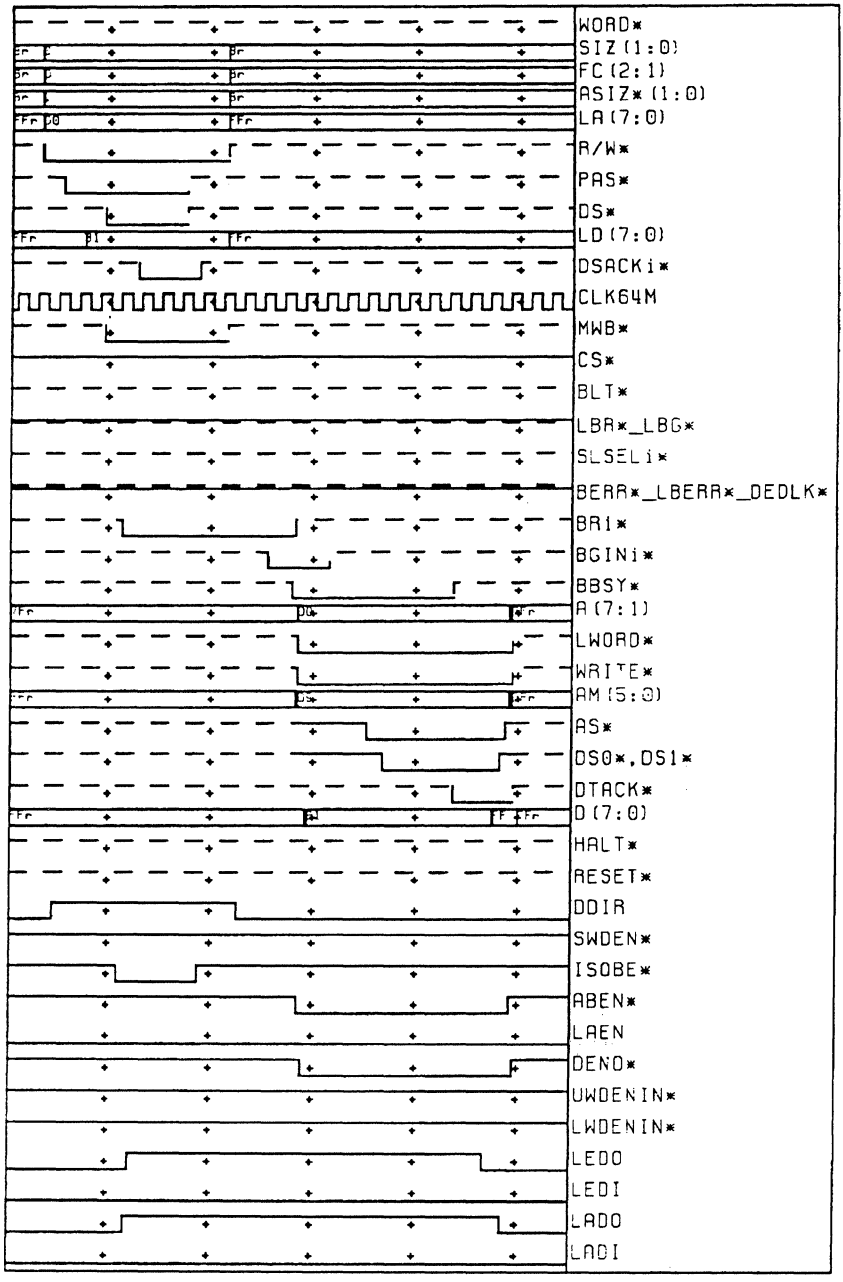


Figure 16–7. Master Write Post

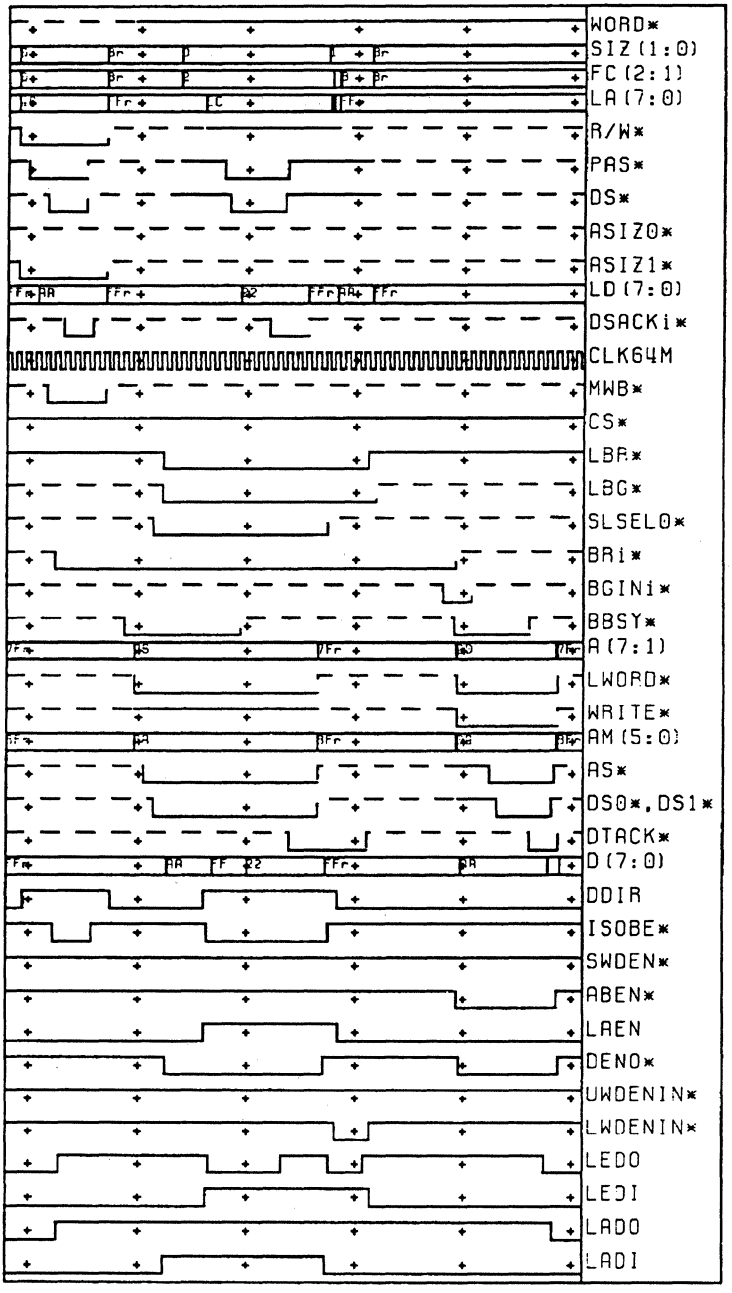


Figure 16–8. Master Write Post with Slave Read (shows data toggle)

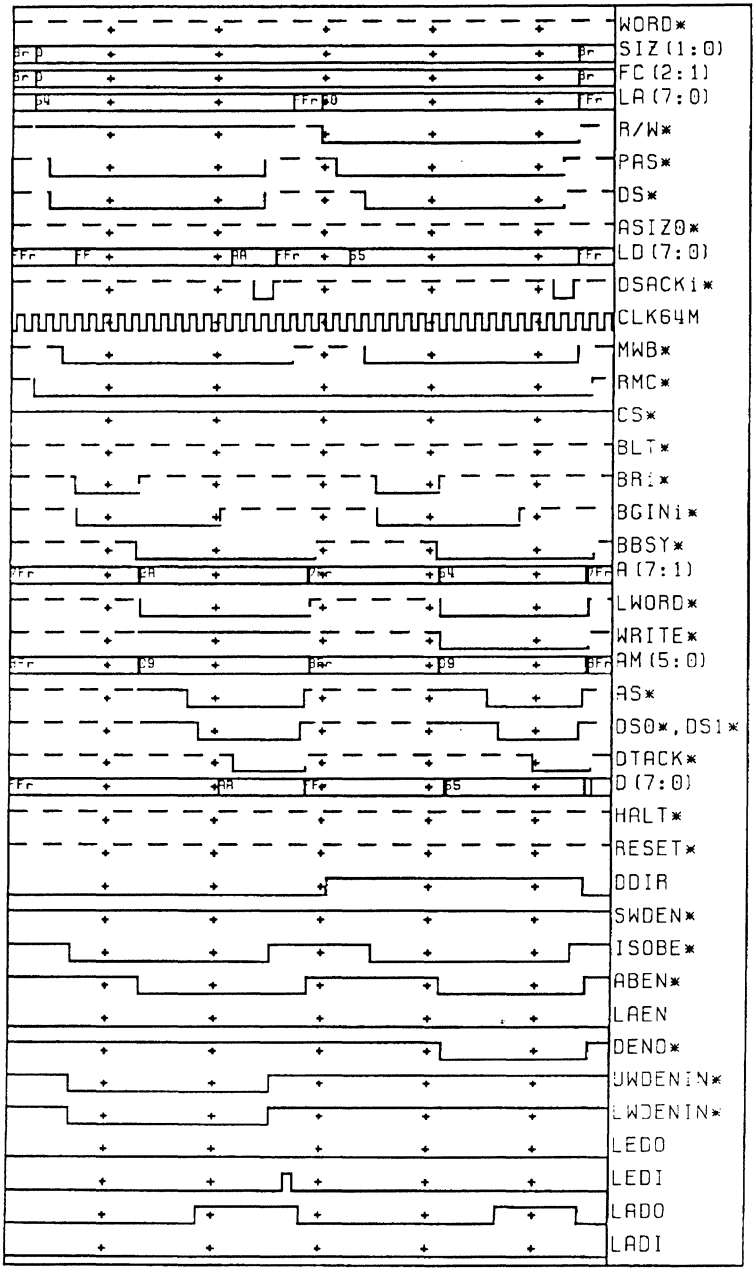


Figure 16−9. Master RMC1 ($AF[7:5] = 000)

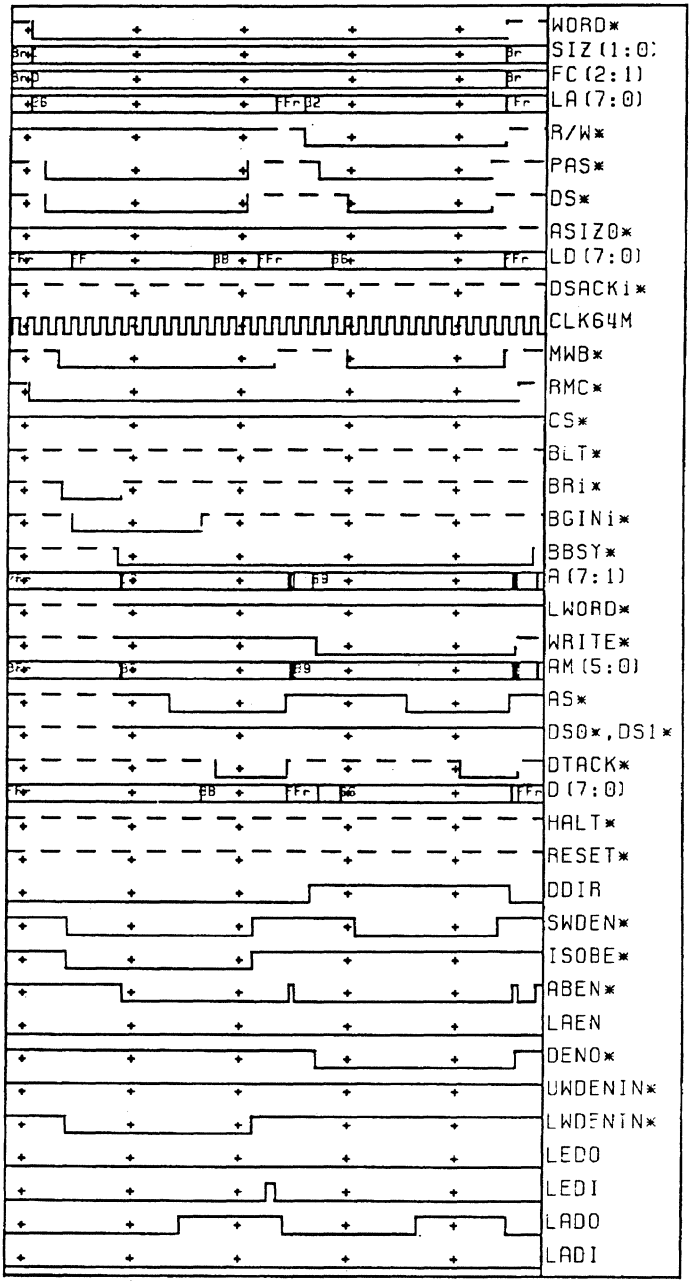


Figure 16−10. Master RMC2 ($AF[7:5] = 001)

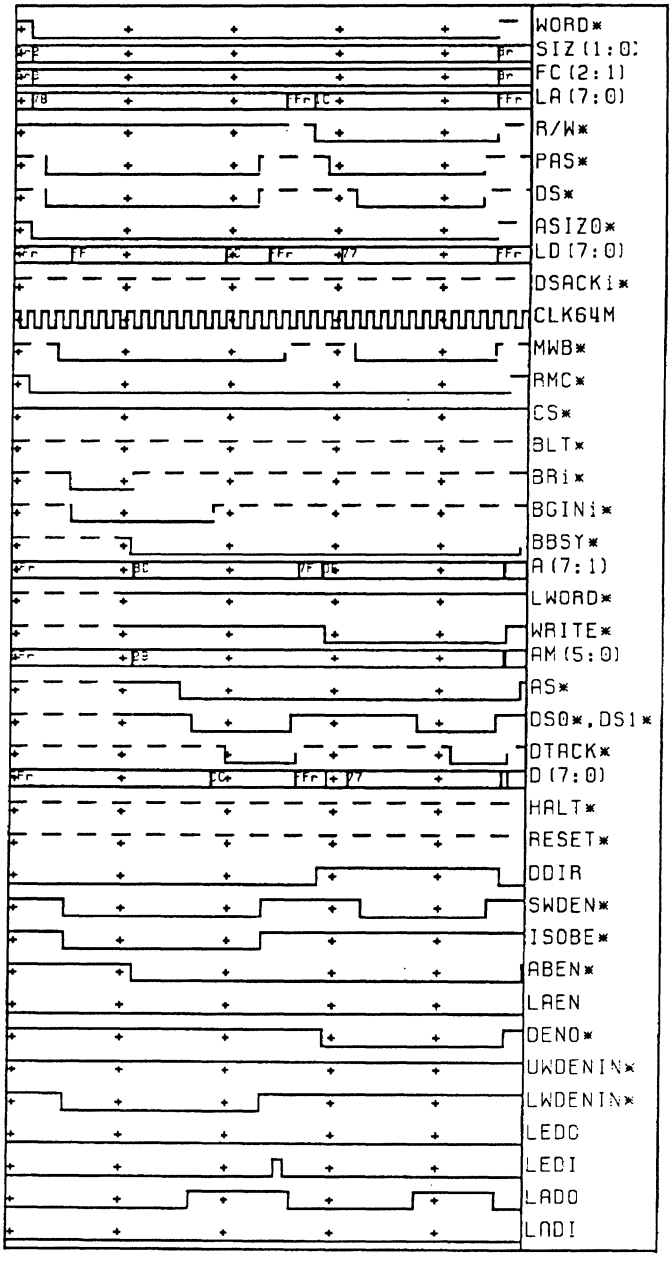


Figure 16−11. Master RMC3 ($AF[7:5] = 010)

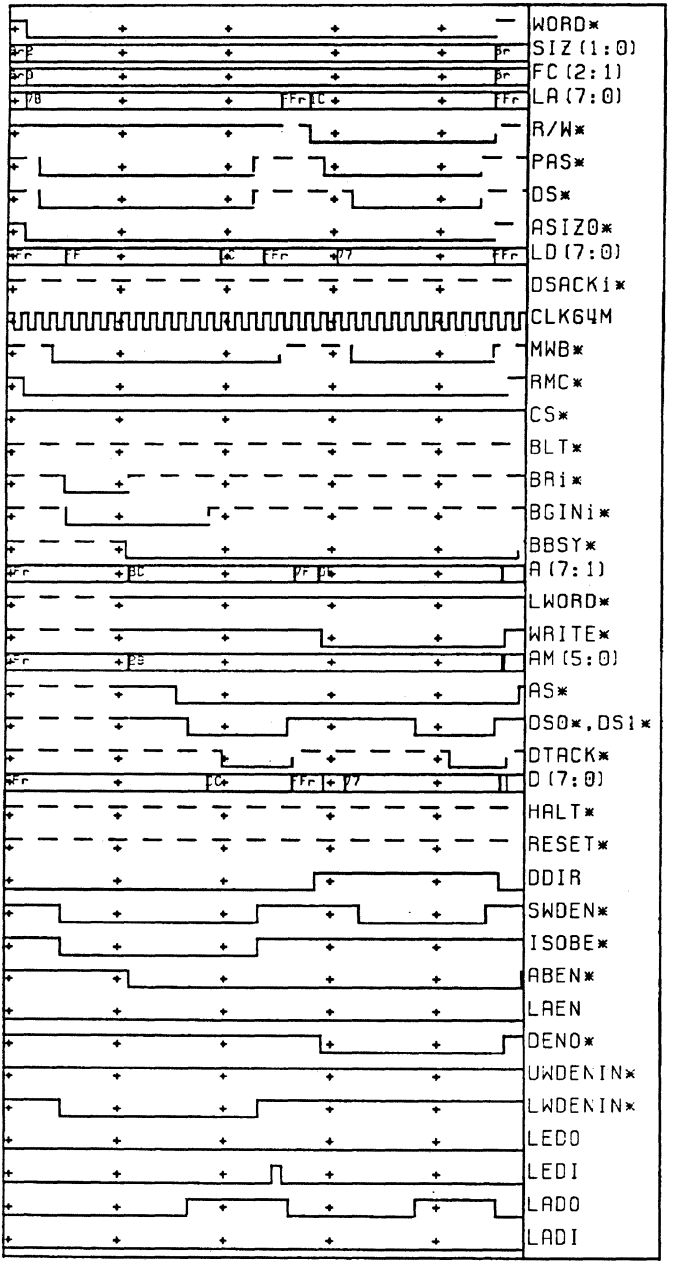


Figure 16−12. Master RMC4 ($AF[7:5] = 011)

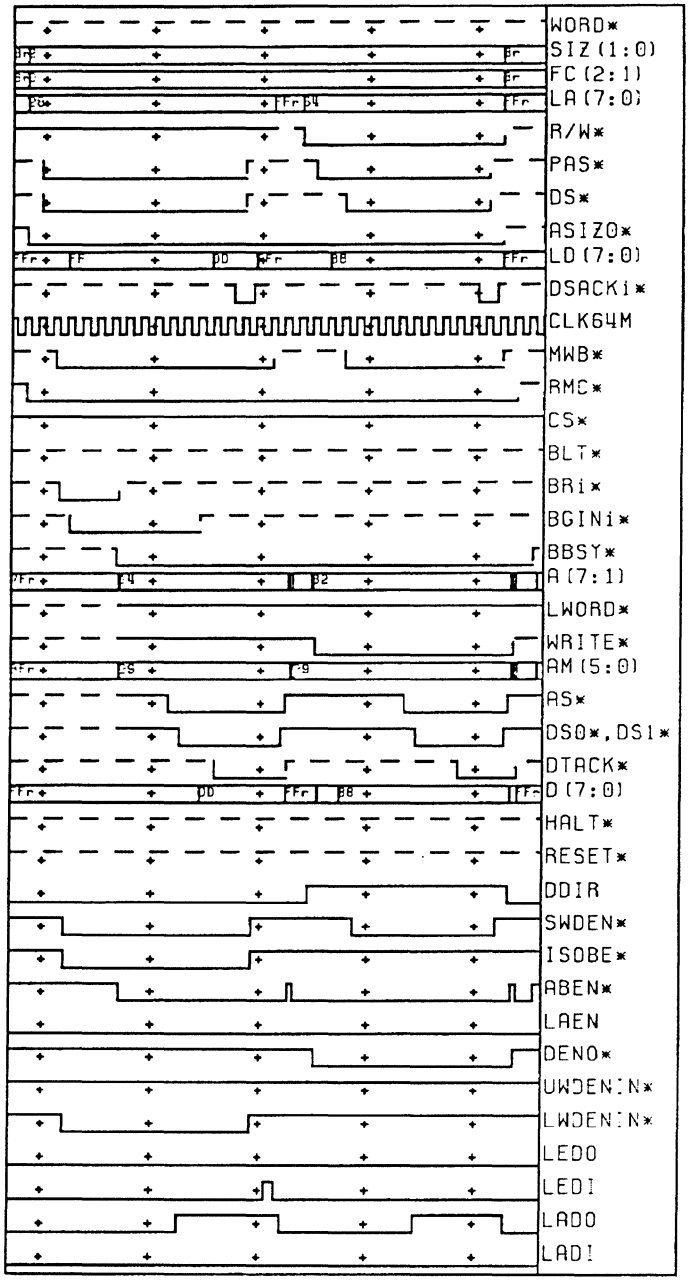


Figure 16–13. Master RMC5 ($AF[7:5] = 101)

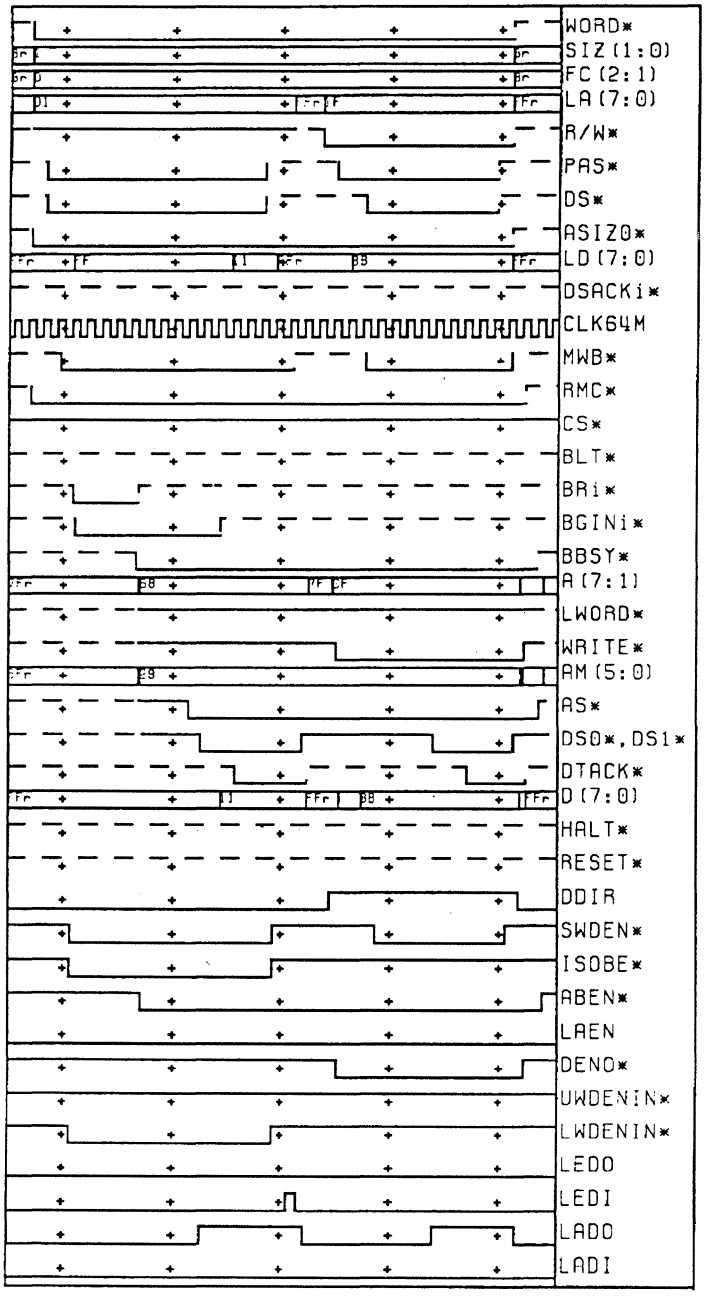


Figure 16–14. Master RMC6 Non-Byte ($AF[7:5] = 110)

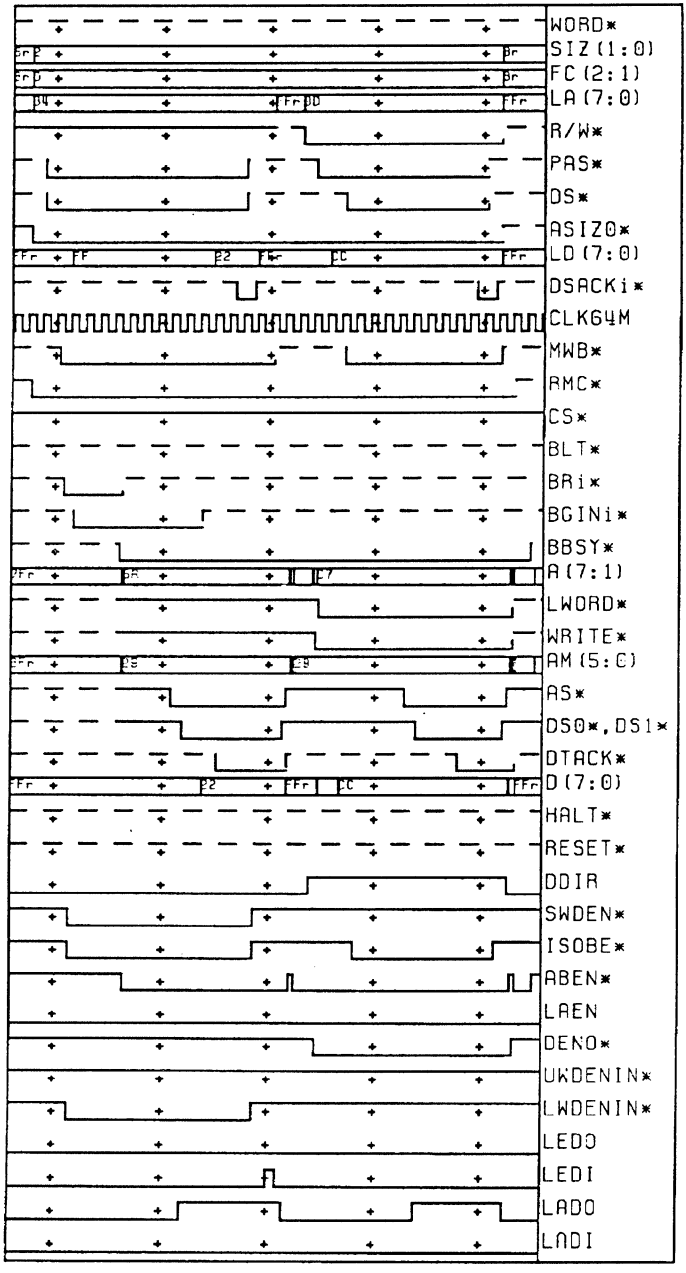


Figure 16−15. Master RMC7 ($AF[7:5] = 111)

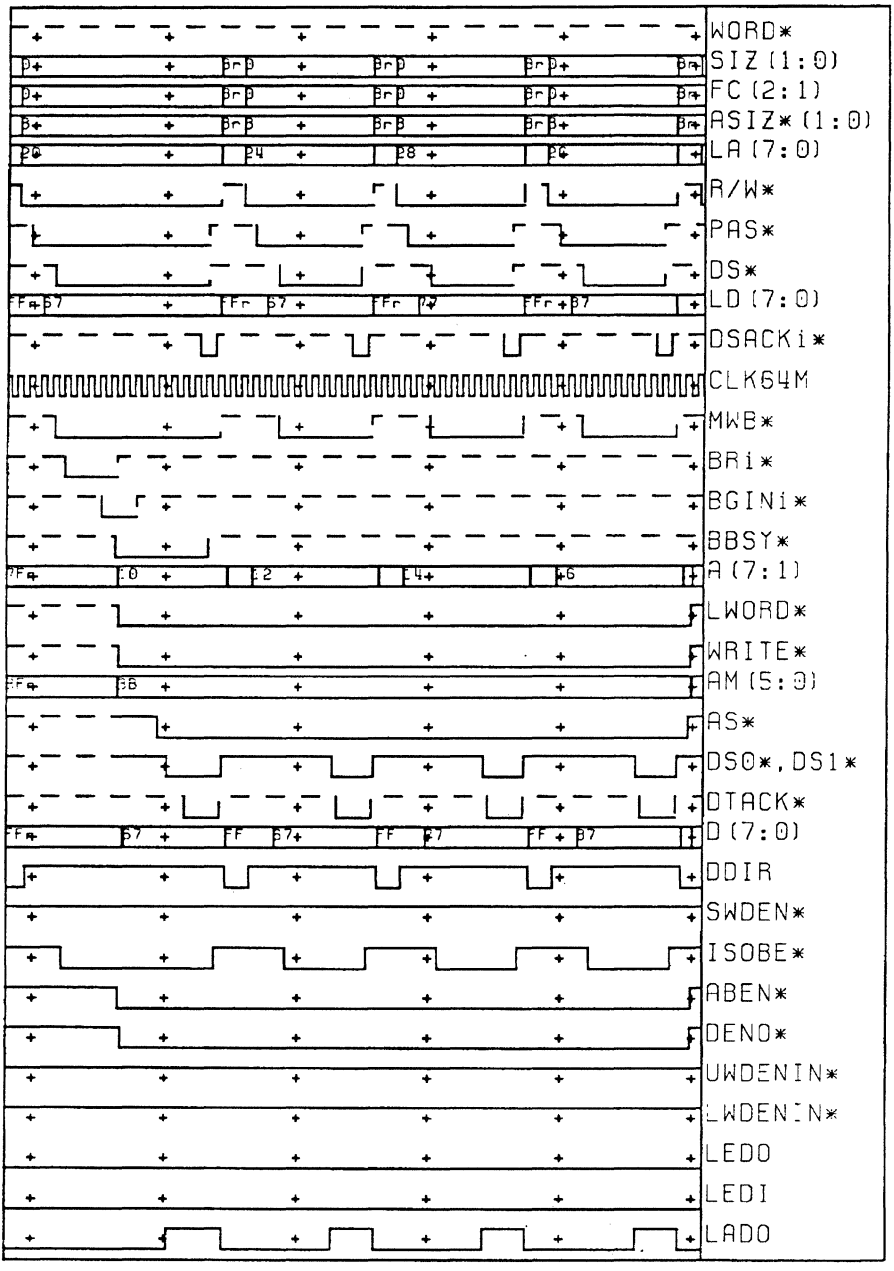


Figure 16–16. MOVEM Write Operation

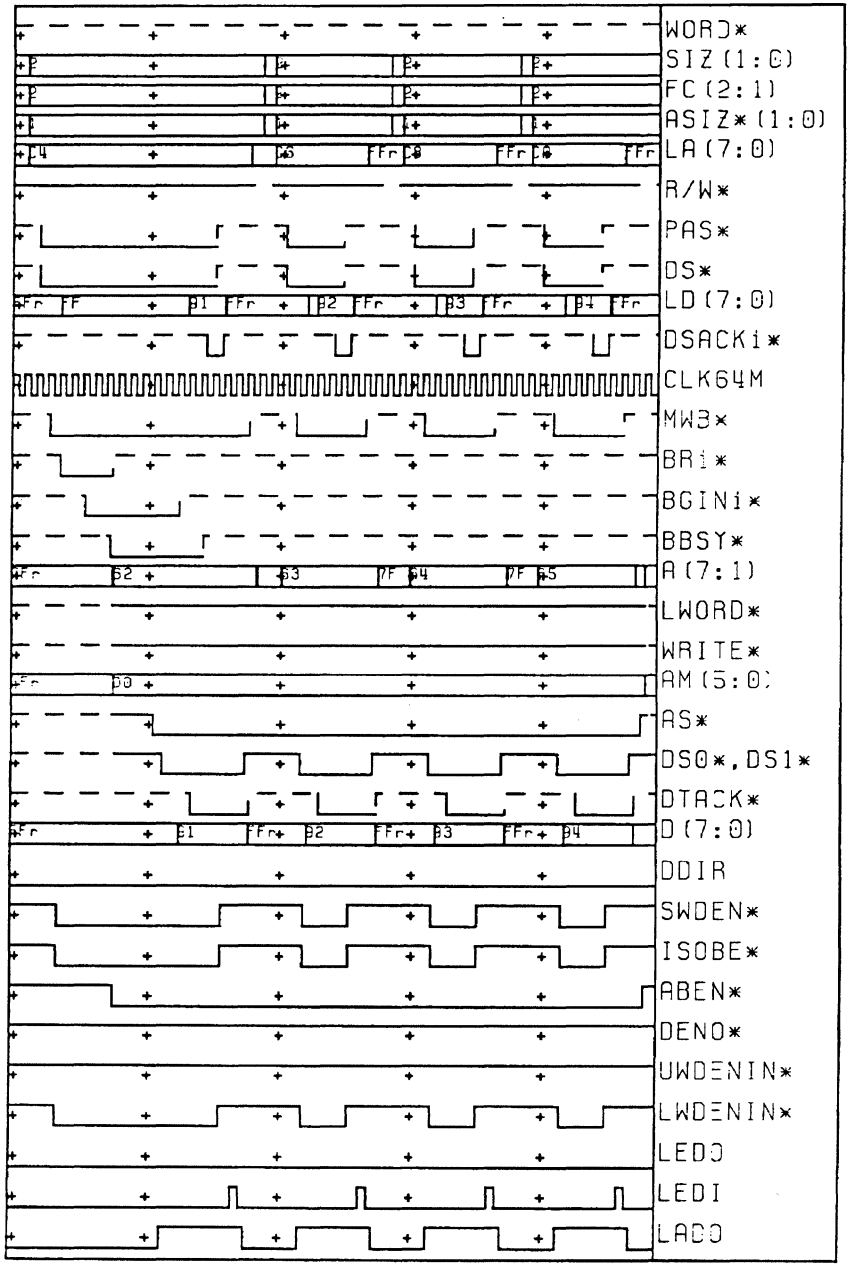


Figure 16–17. MOVEM Read Operation

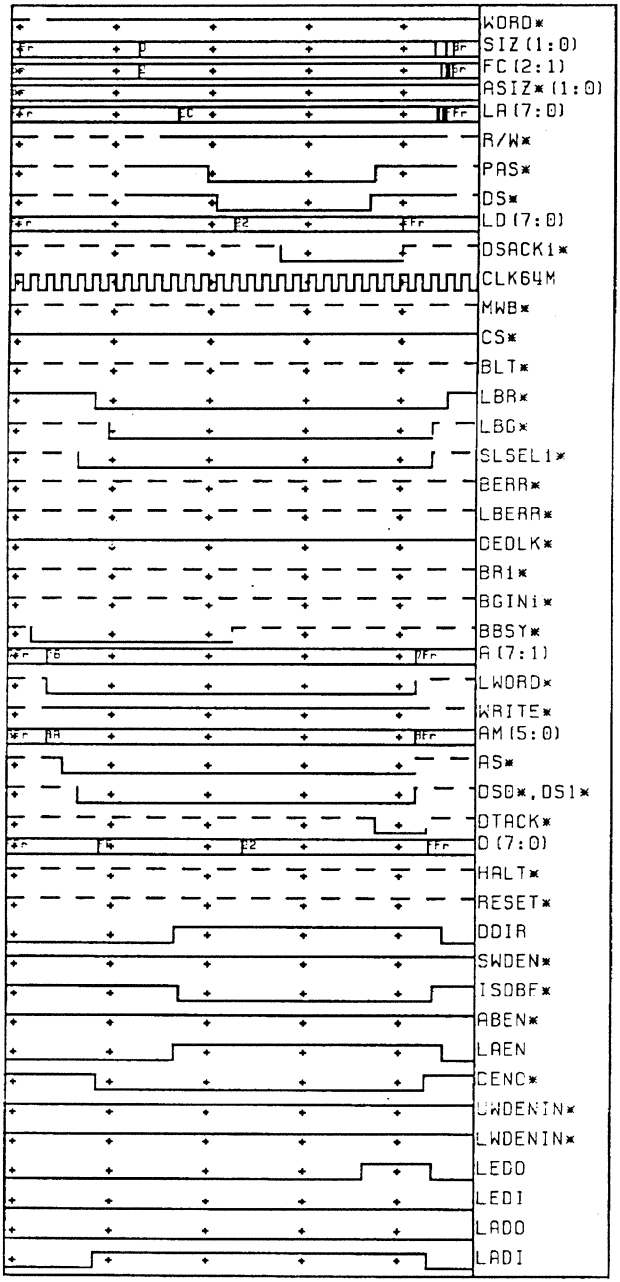


Figure 16–18. Slave Read

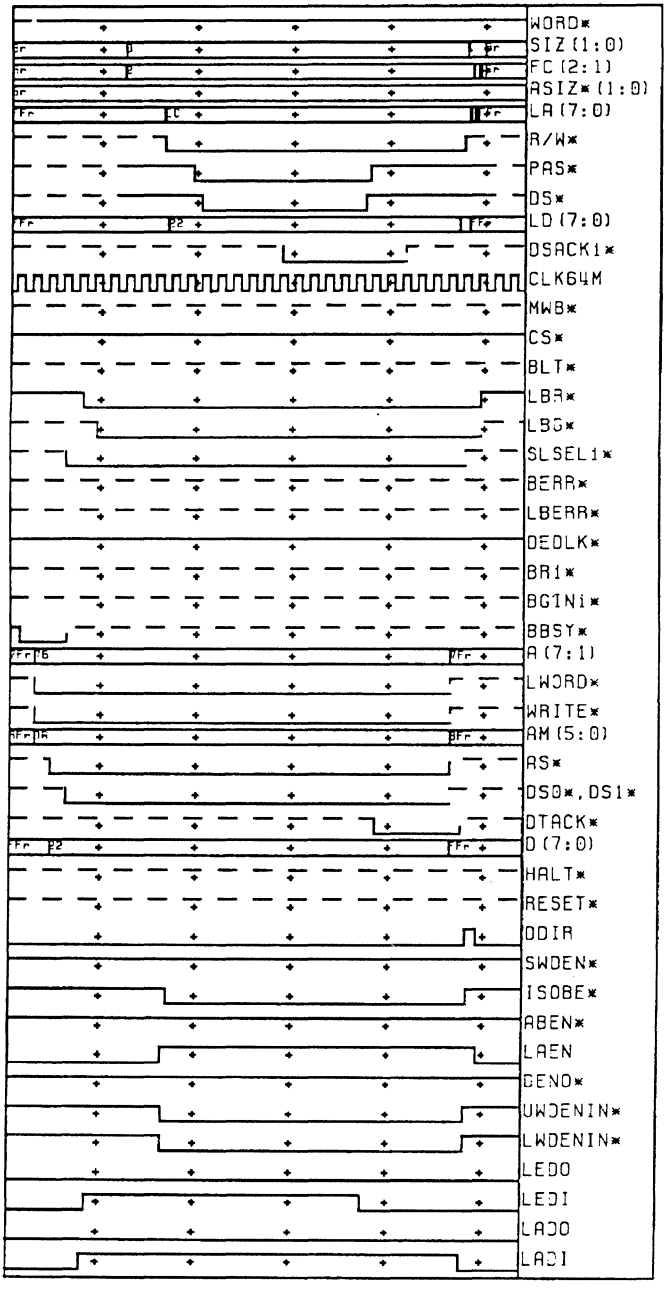


Figure 16–19. Slave Read (non-posted)

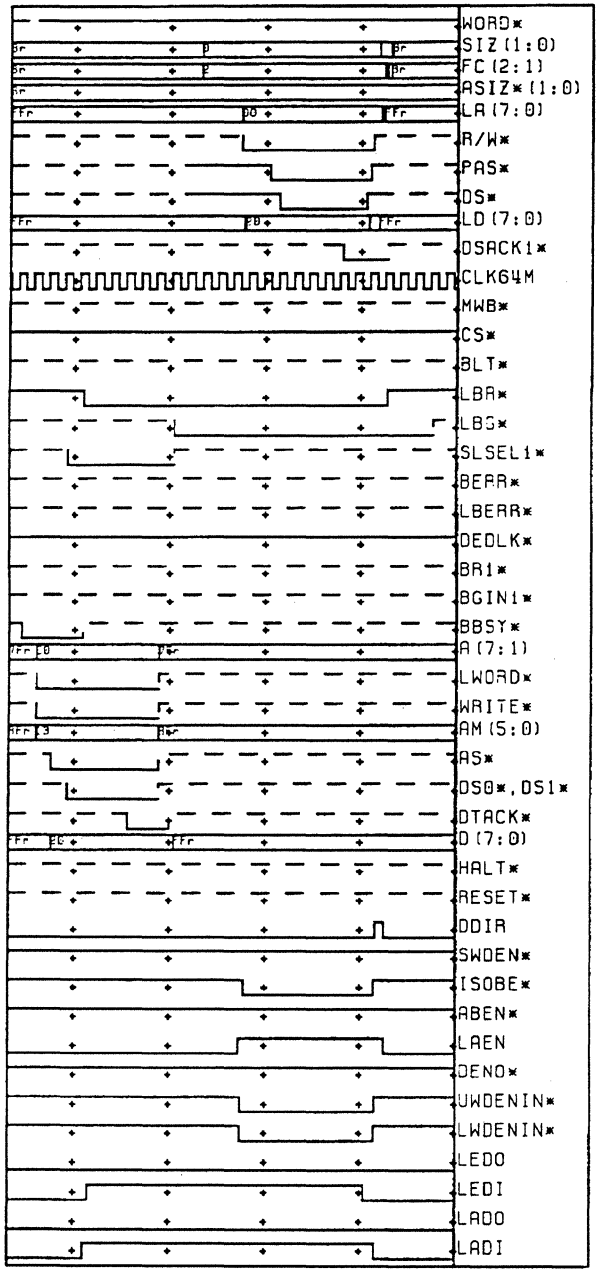
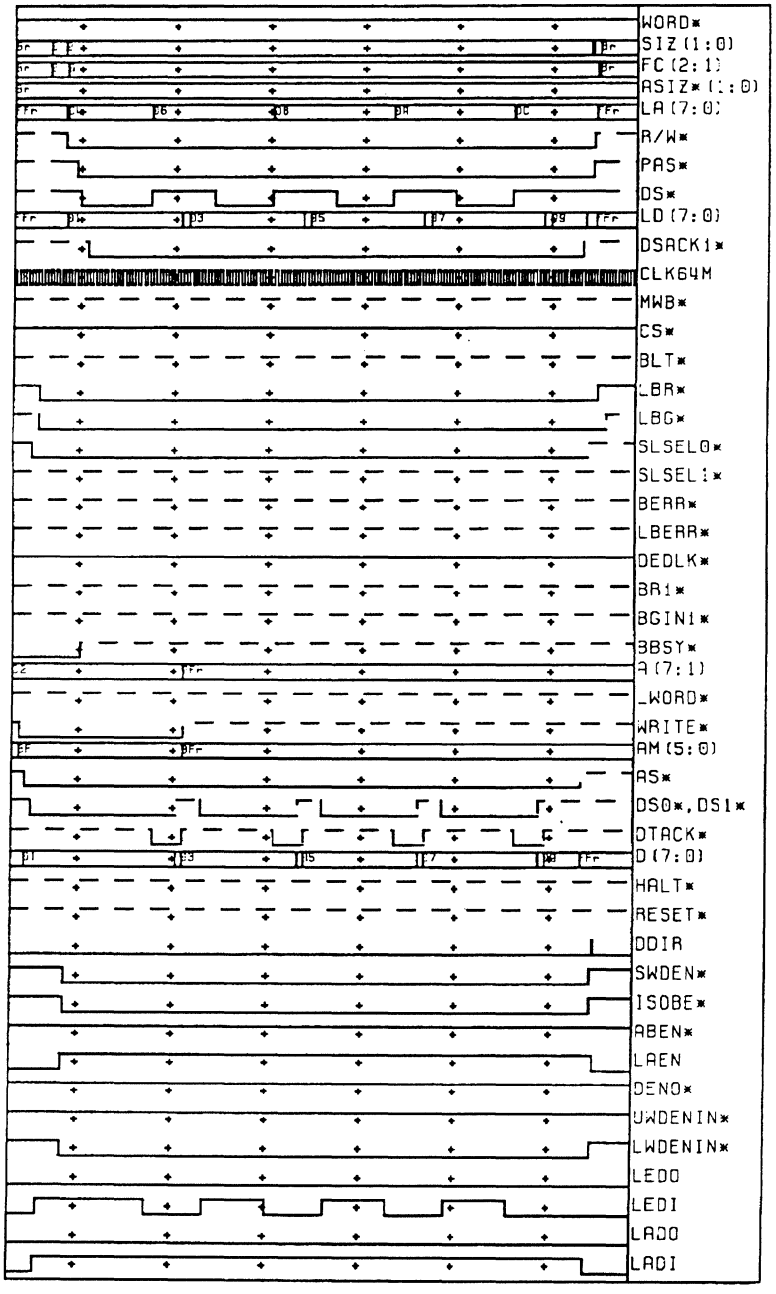


Figure 16–20. Slave Write Post



234 NS

Figure 16–21. Slave Write Block Transfer (DMA mode)

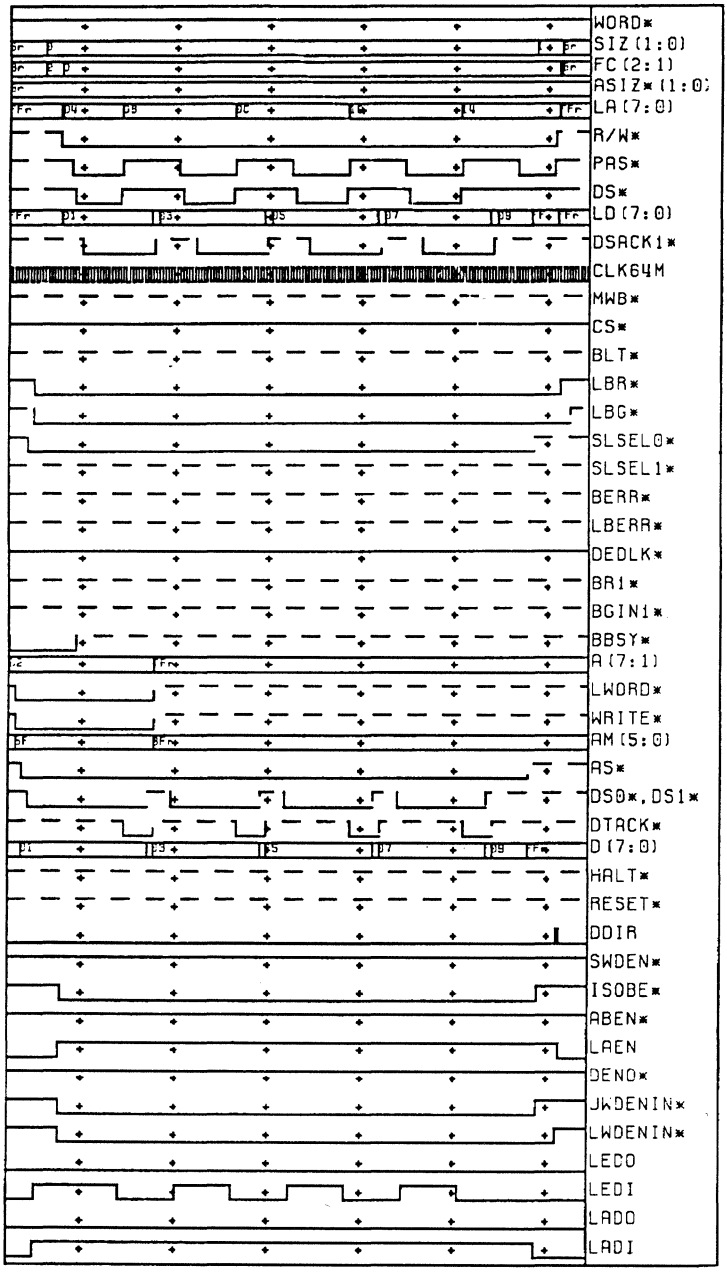


Figure 16−22. Slave Write Block Transfer (non-DMA mode)

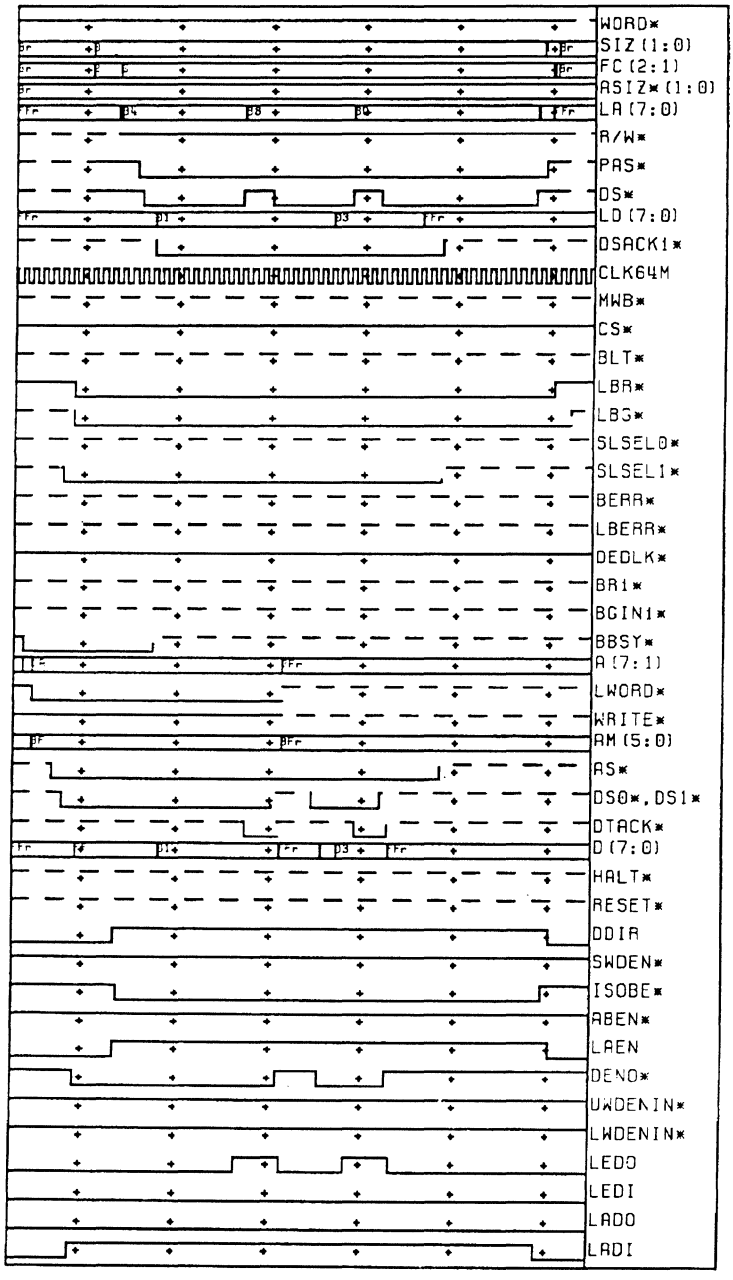


Figure 16−23. Slave Read Block Transfers (DMA mode)

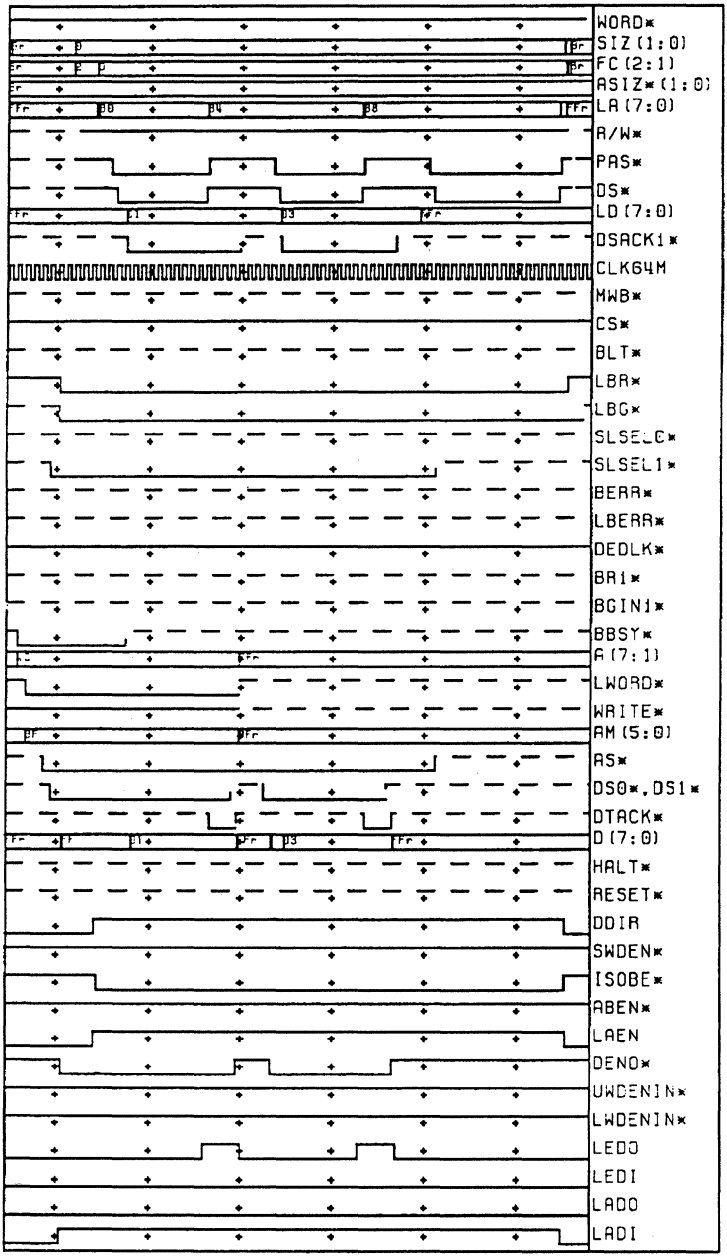


Figure 16–24. Slave Read Block Transfer (non-DMA mode)

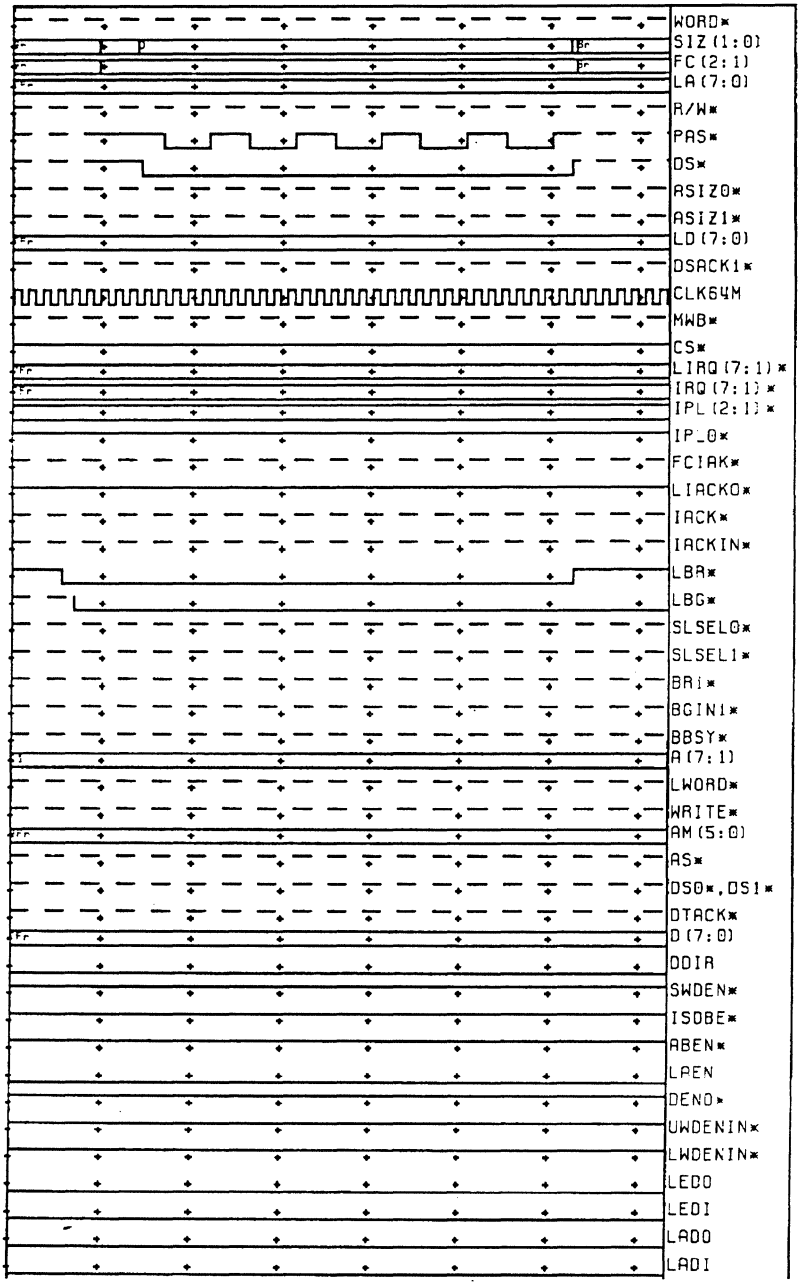


Figure 16–25. Refresh Timing

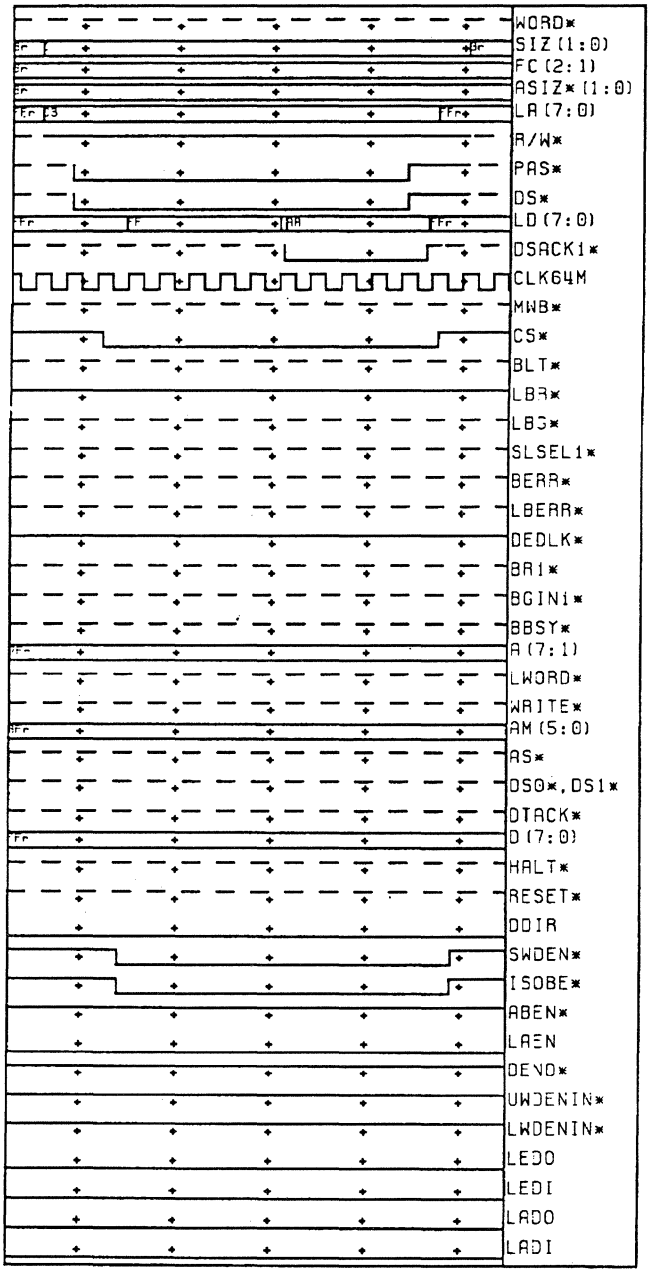


Figure 16−26. Register Read

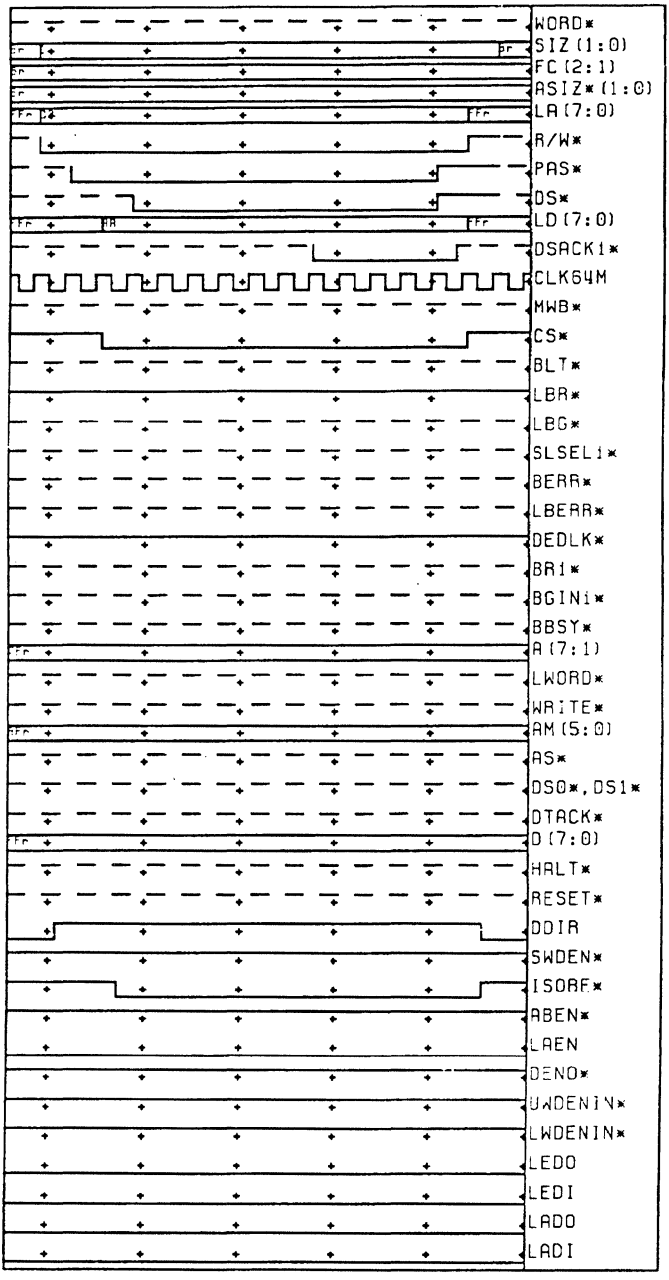


Figure 16–27. Register Write

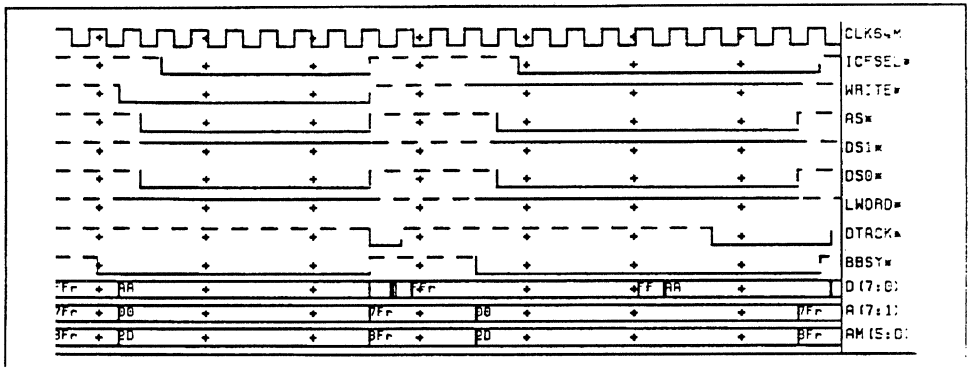


Figure 16–28. Interprocessor Communications Register Access Timing

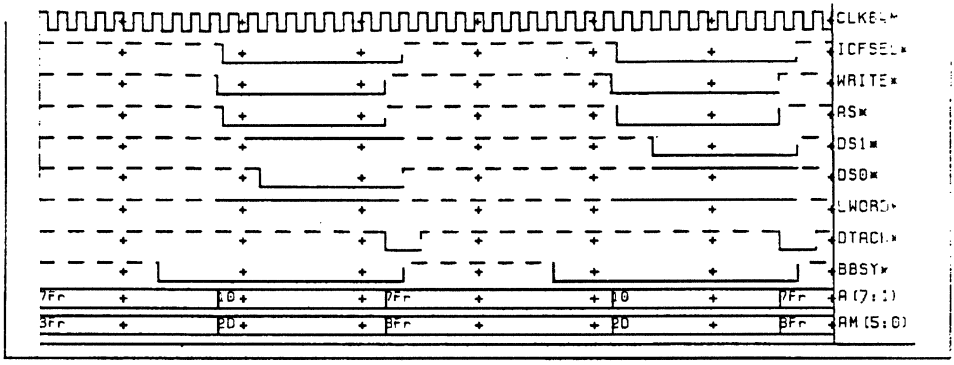


Figure 16–29. Interprocessor Communication Module Switch Access Timing

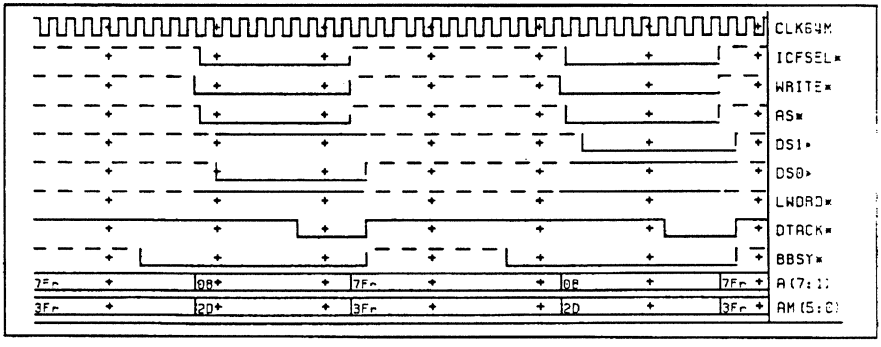


Figure 16–30. Interprocessor Communications Global Switch Access Timing

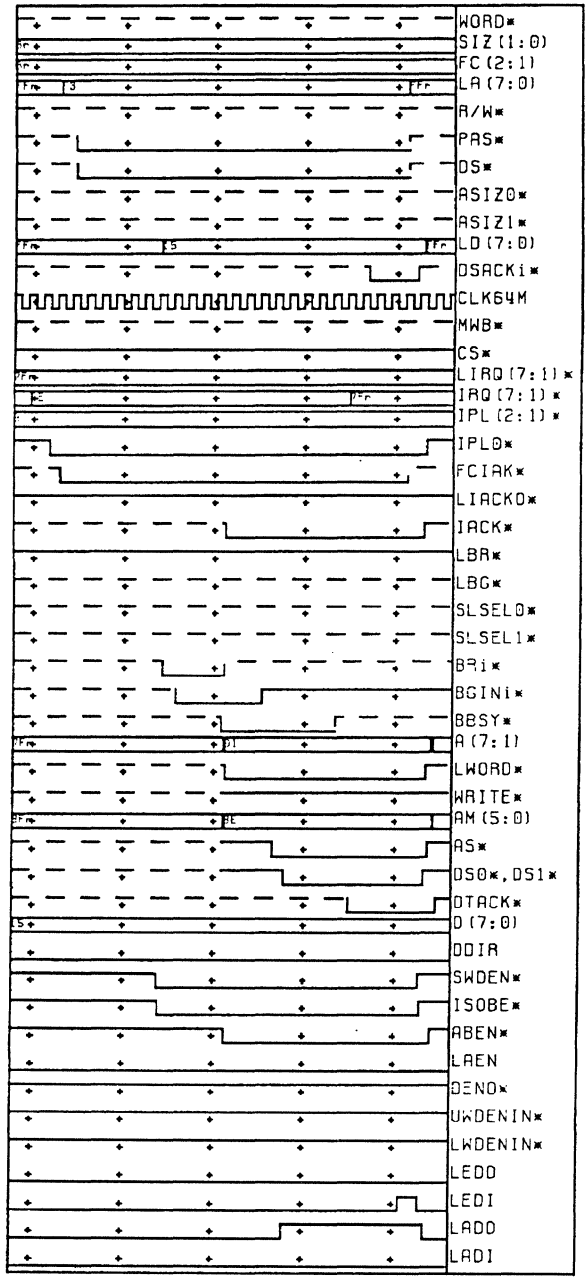


Figure 16–31. VMEbus Interrupt Acknowledge Cycle

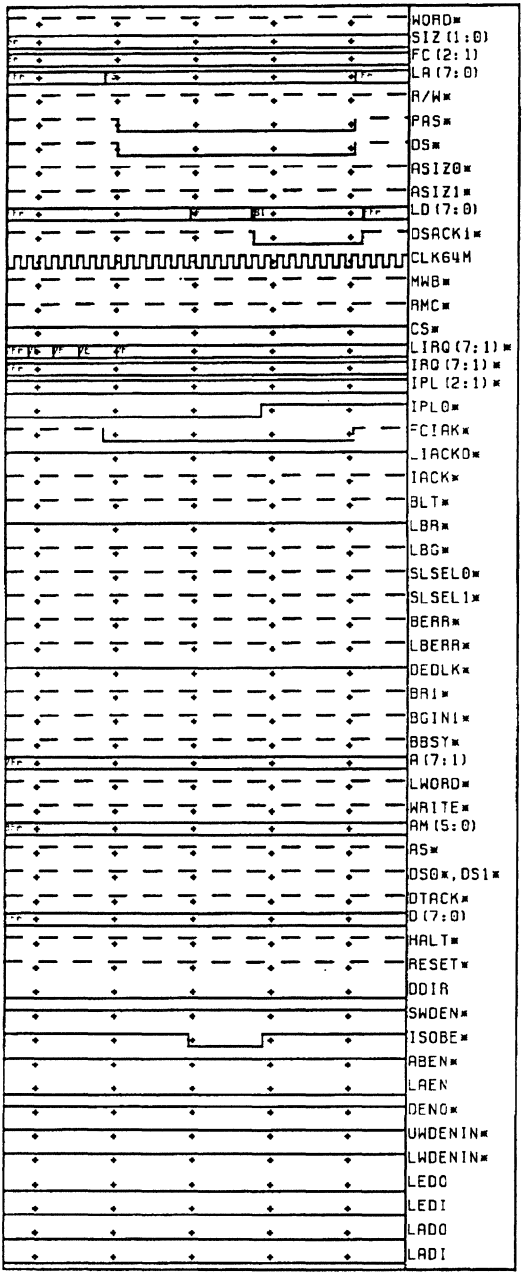


Figure 16–32. Local Interrupt Acknowledge Cycle

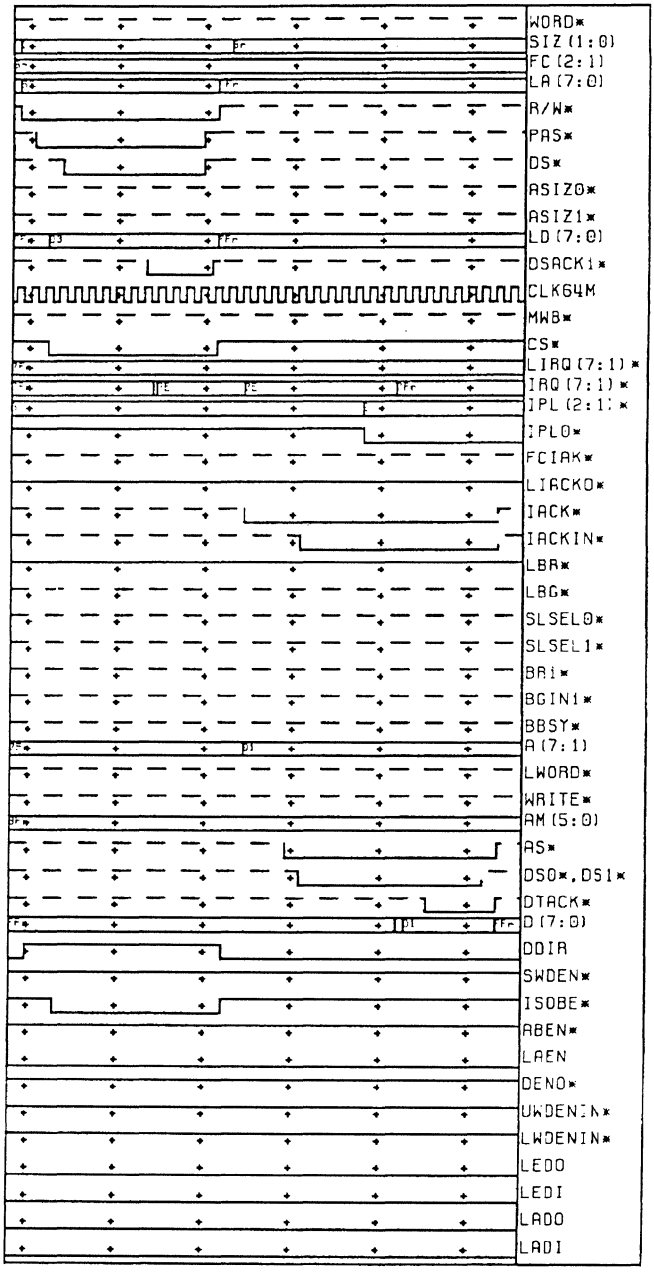


Figure 16–33. Interrupter Acknowledge Cycle

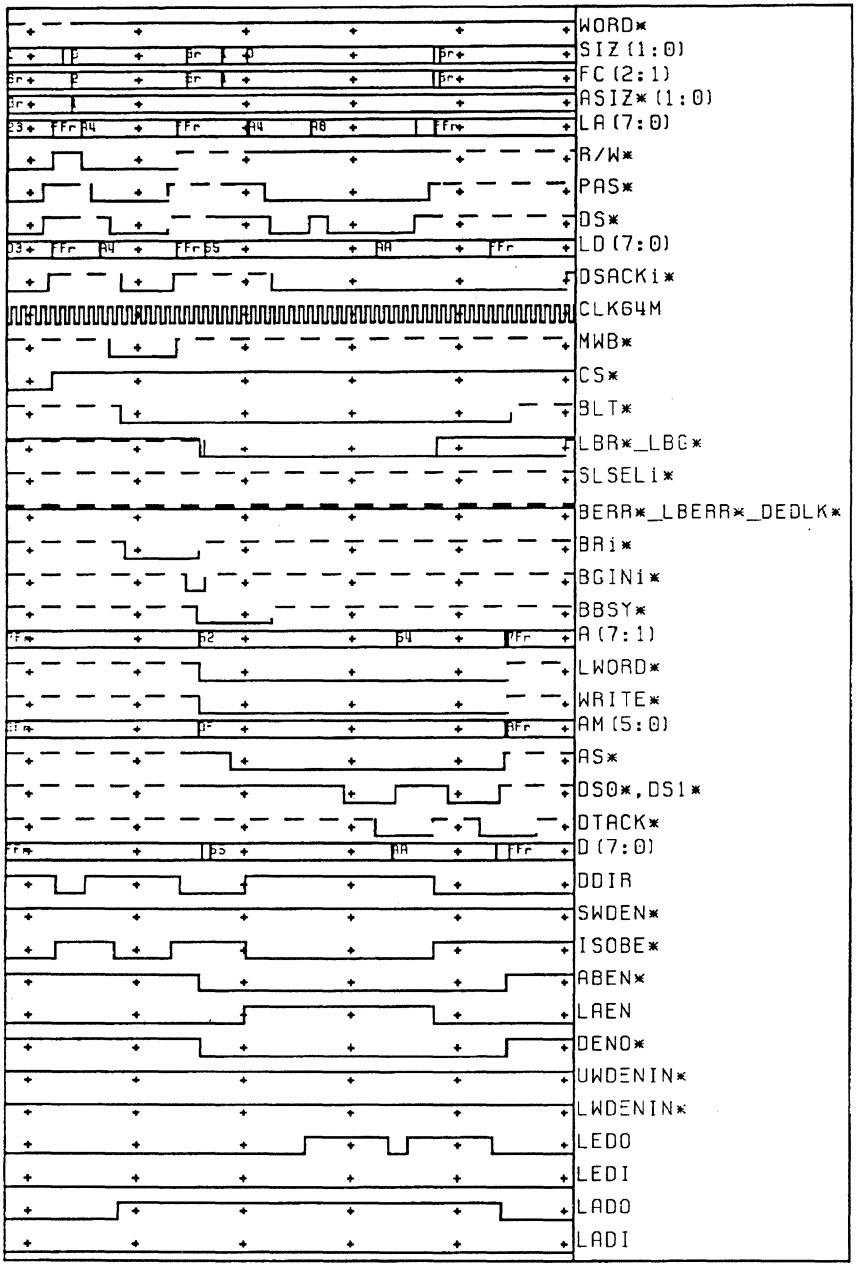


Figure 16–34. Block Transfer: VME Write: Burst of 2

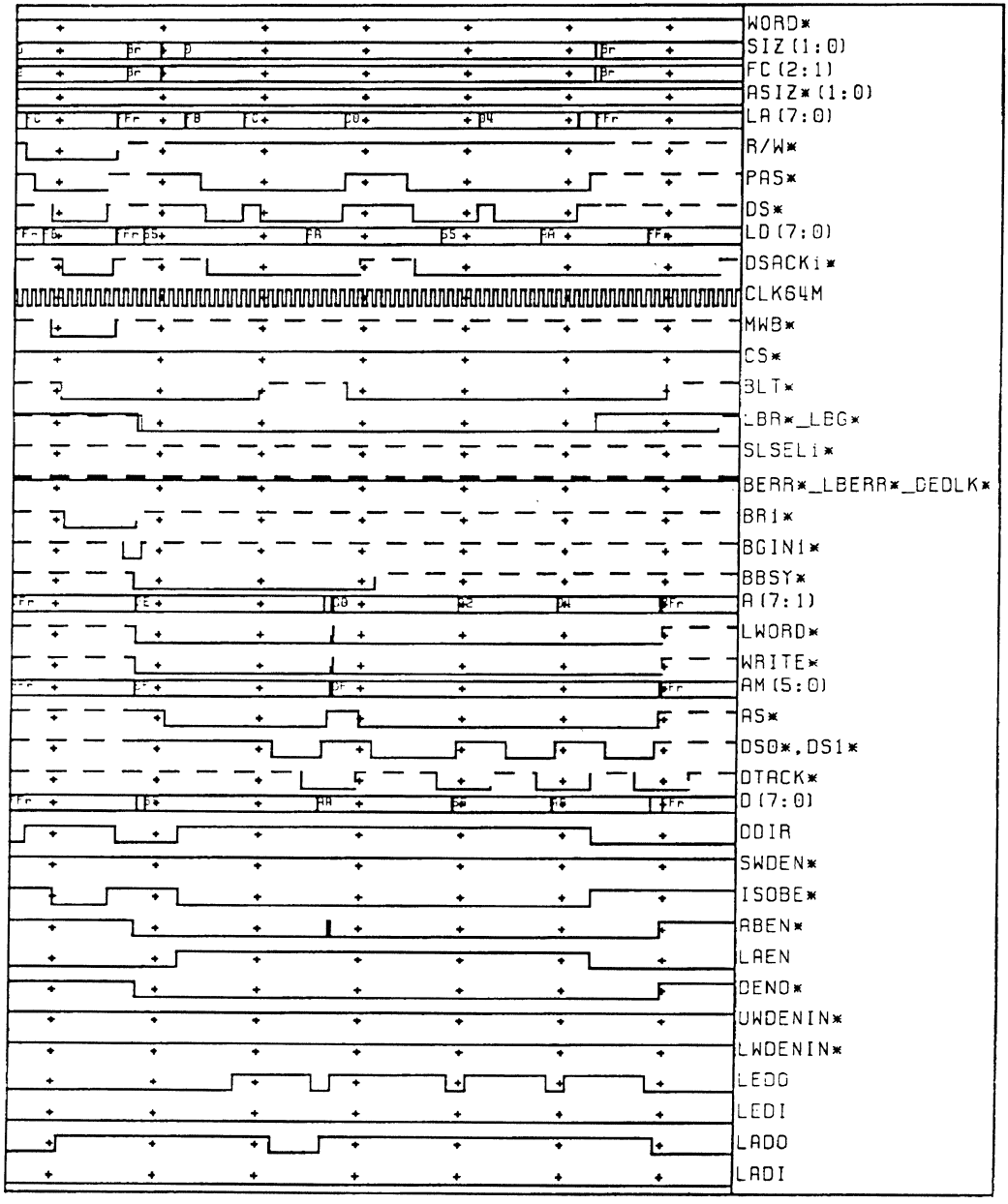


Figure 16–35. VME Boundary Crossing and Local Boundary Crossing

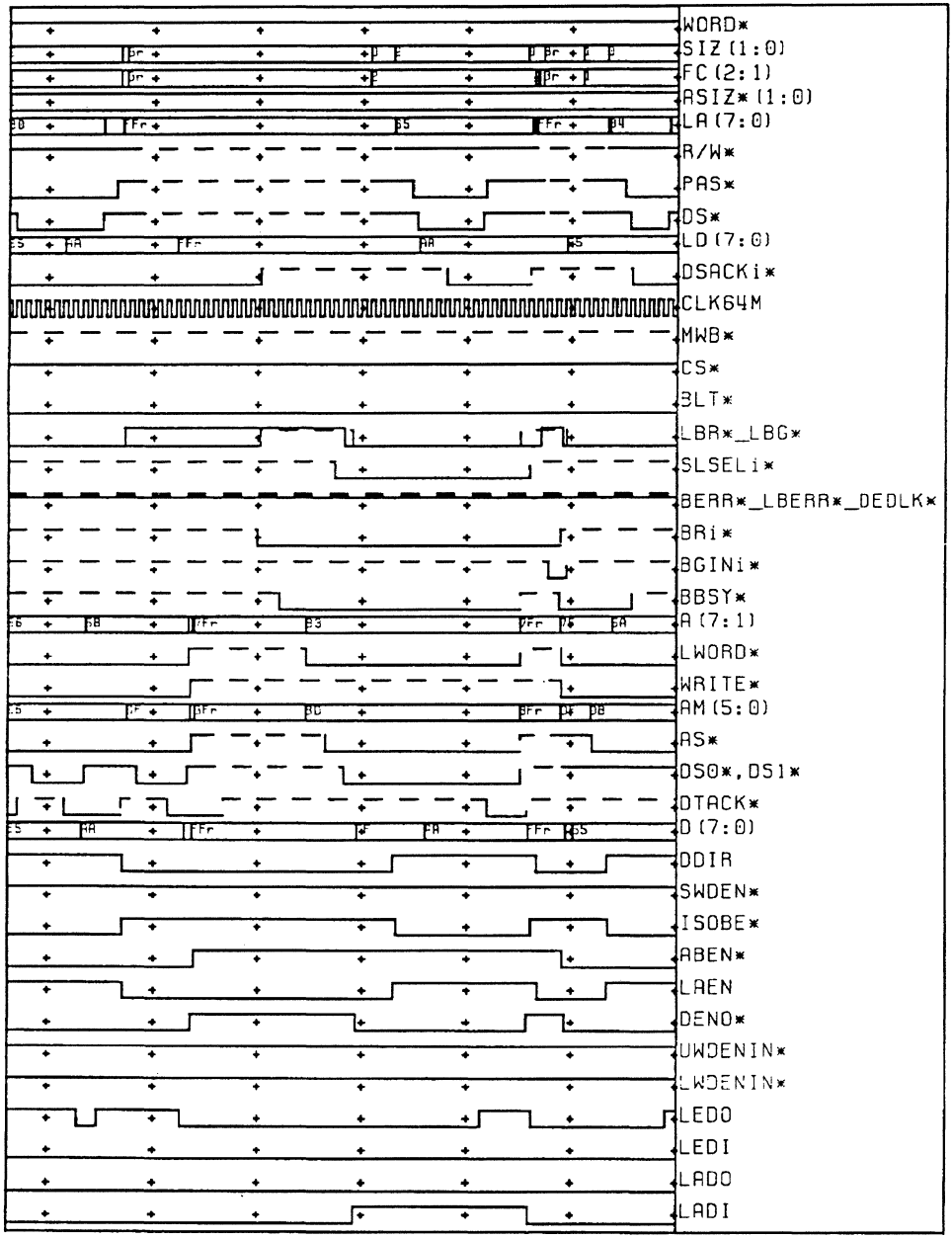


Figure 16–36. Slave Cycle During Interleave Period

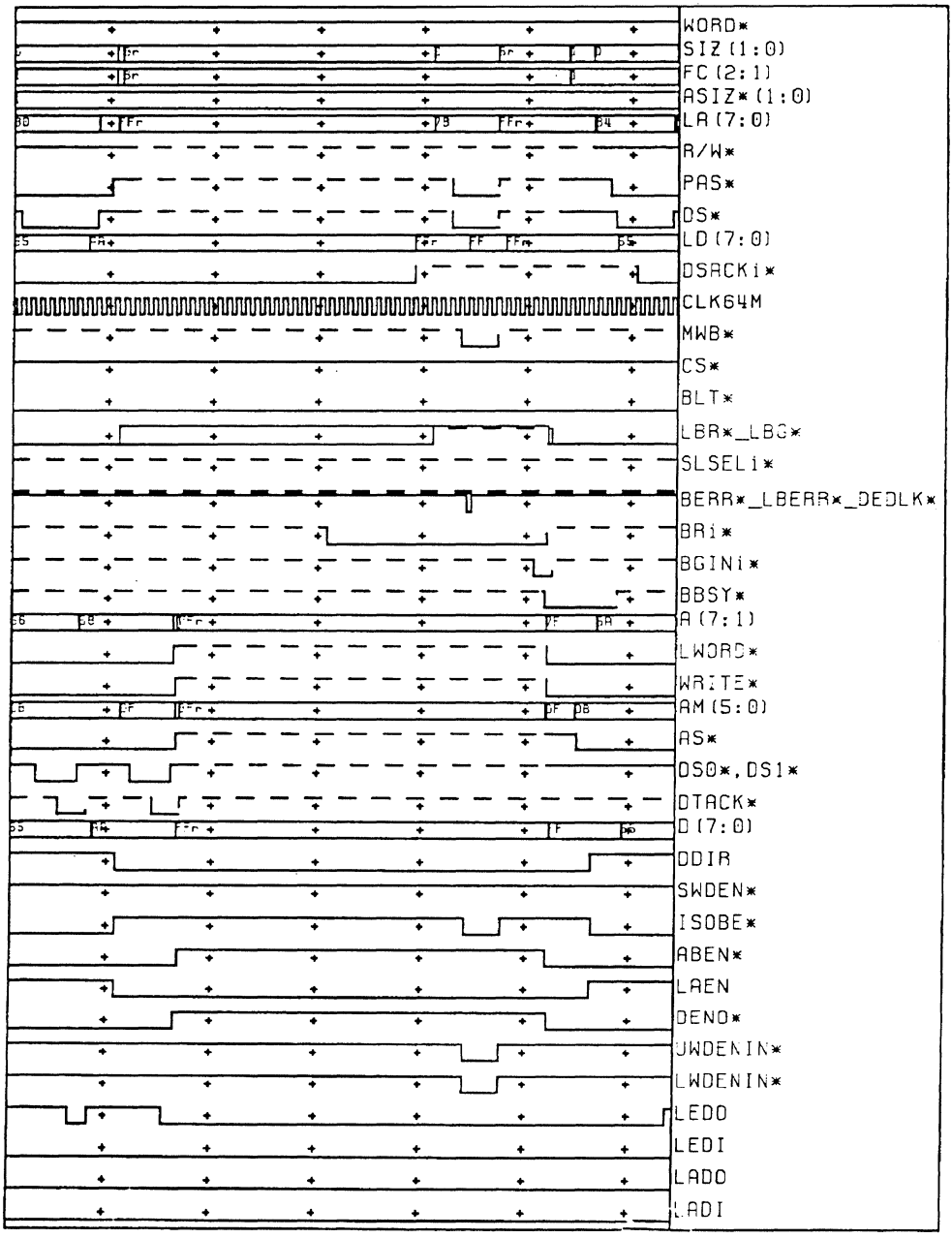


Figure 16–37. Master Cycle Deadlocked During Interleave

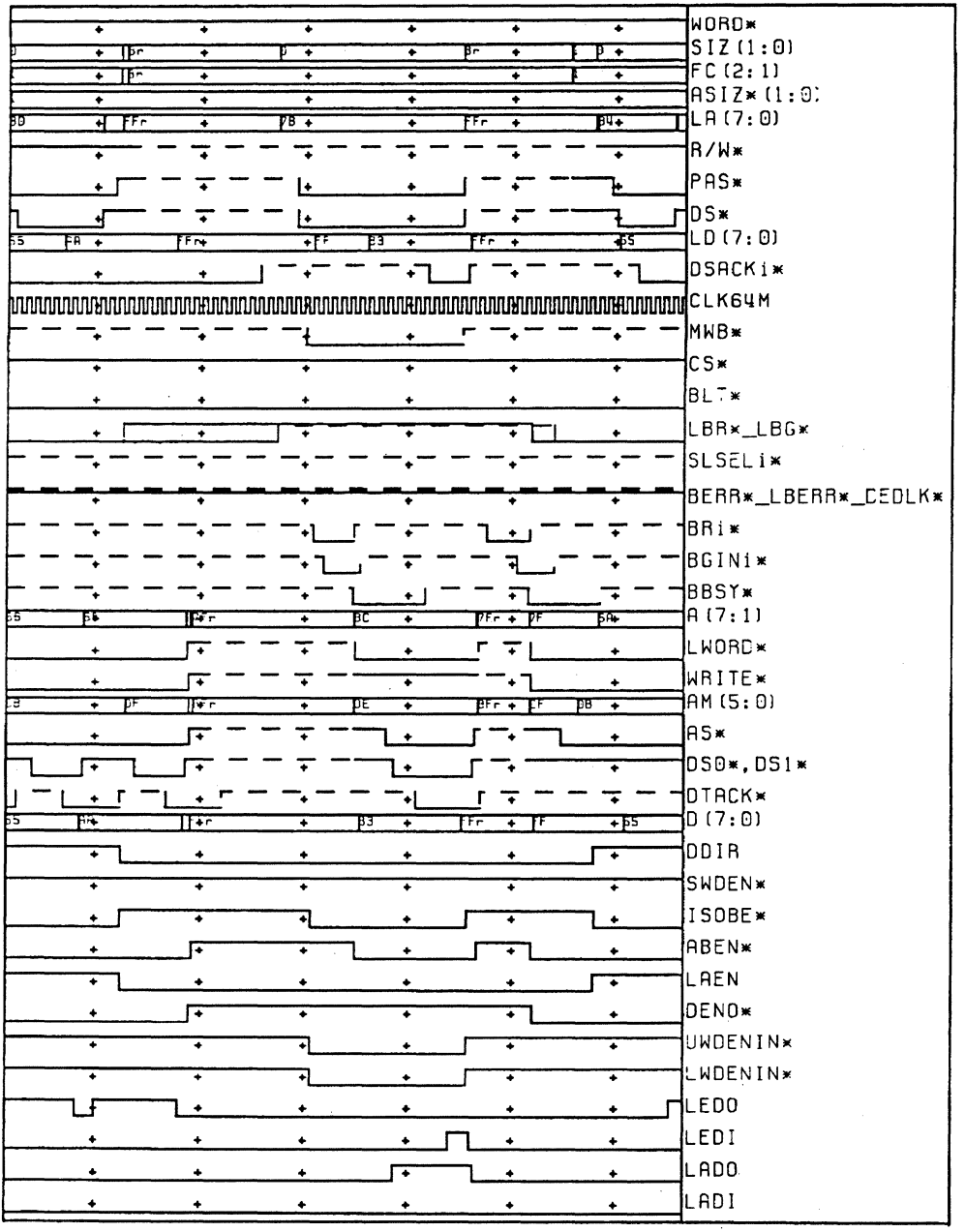


Figure 16–38. Master Cycle During Interleave with Dual-Path Option

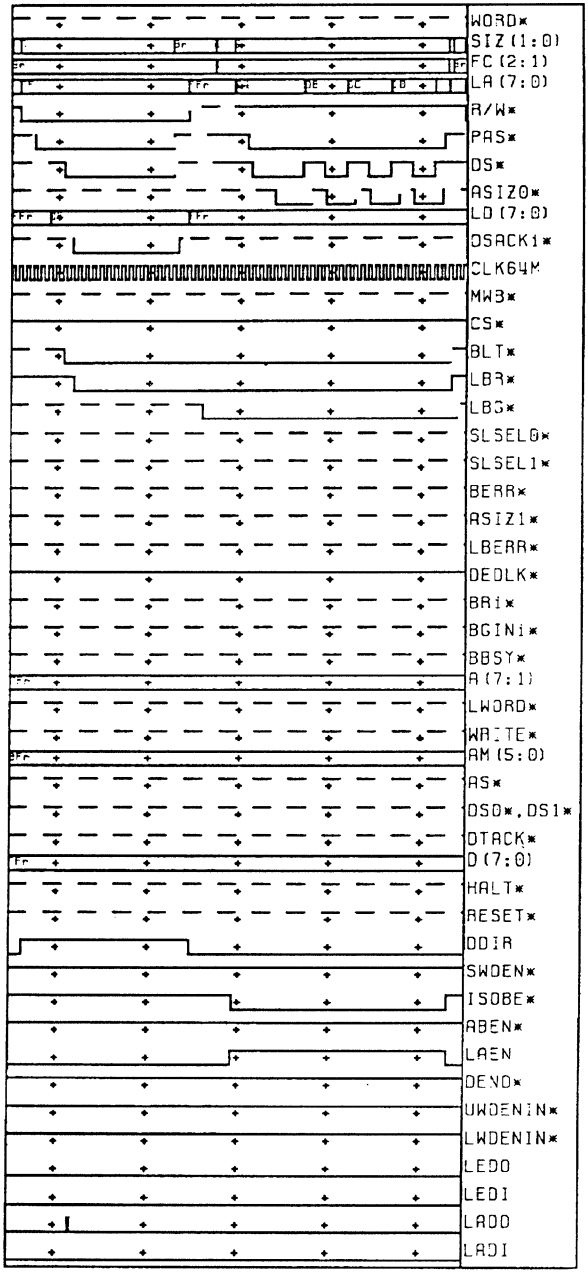


Figure 16–39. Read Operation Module-Based Block Transfers

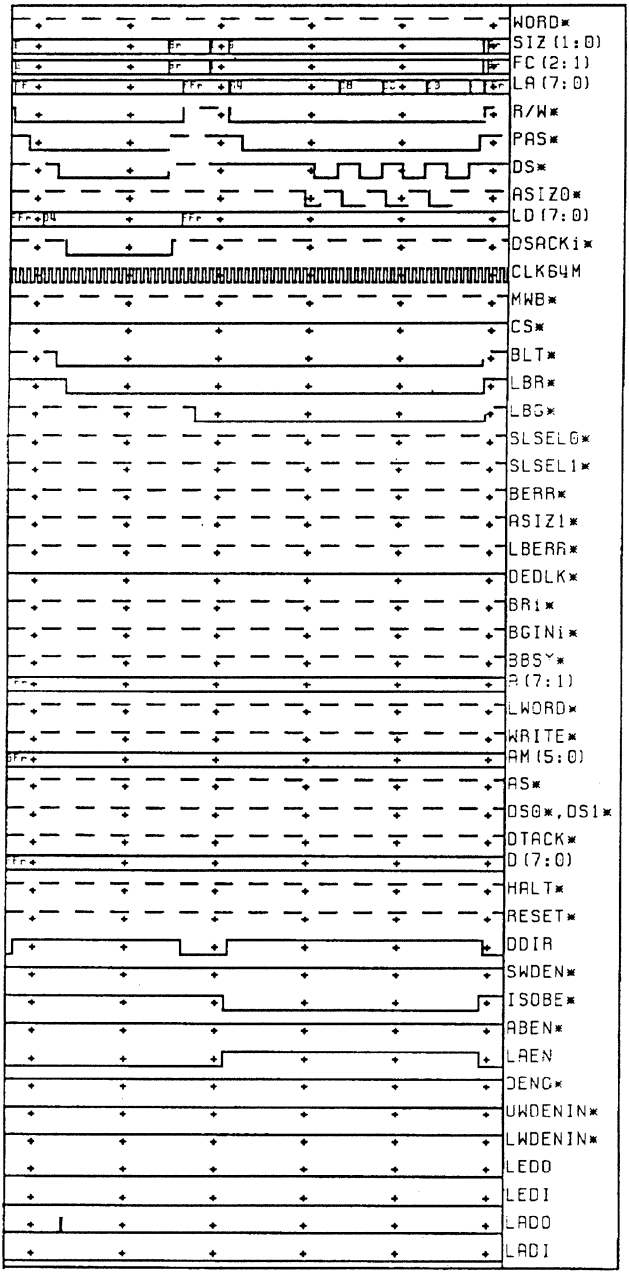


Figure 16−40. Write Operation Module-Based Block Transfer

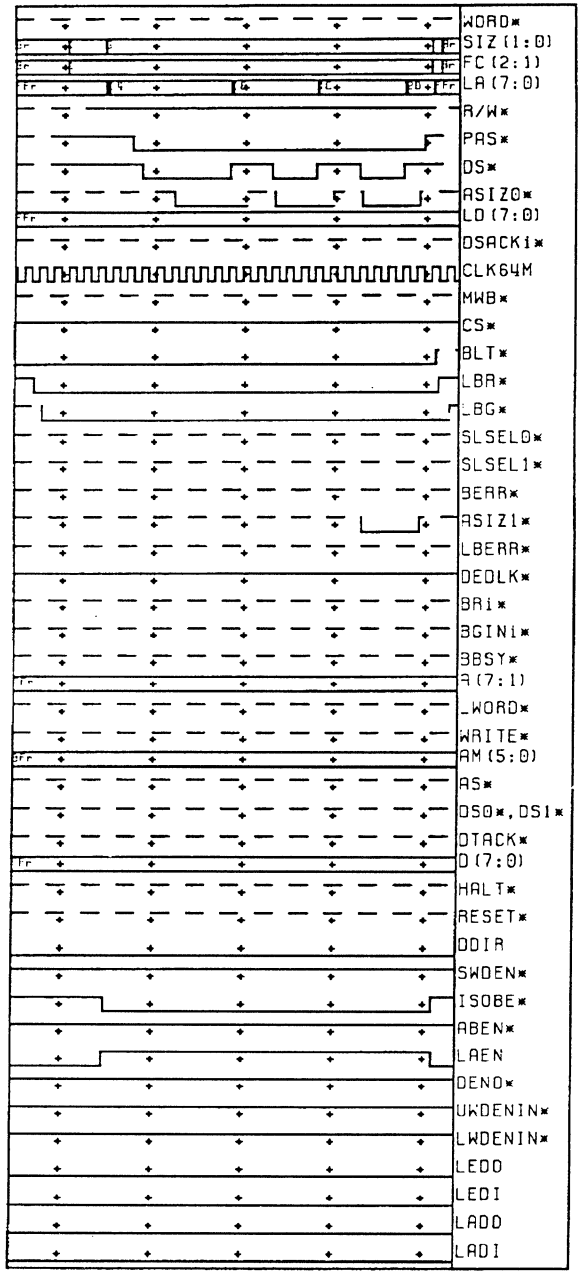


Figure 16−41. Suspension of Module-Based Transfer (caused by ASIZ1* = 0)

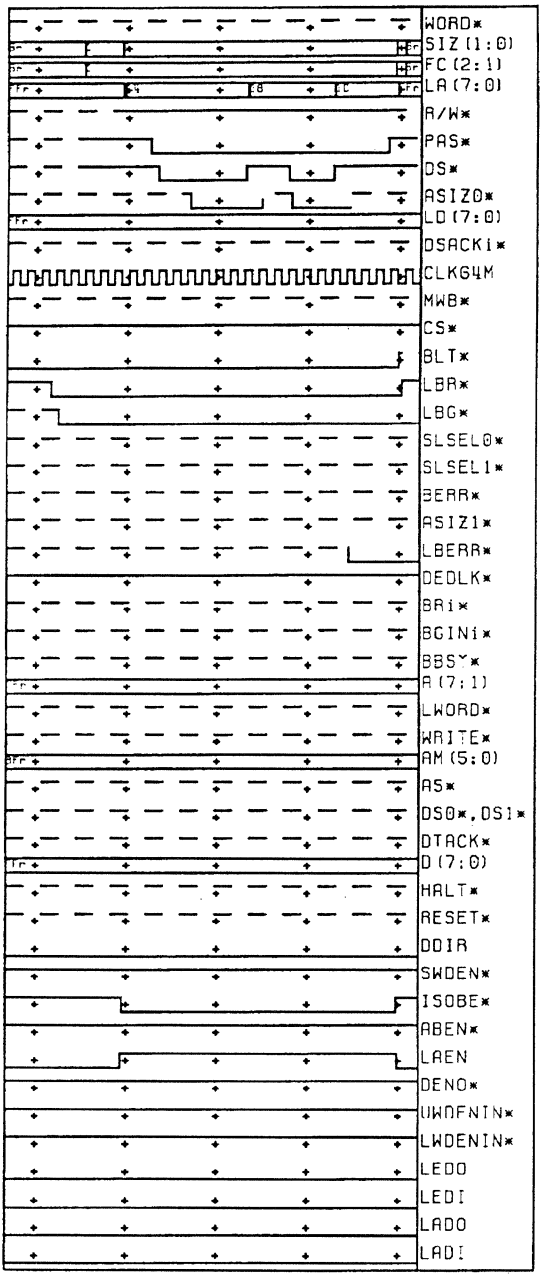


Figure 16−42. Suspension for Module-Based Transfer (caused by LBERR* = 0)

Section 2

VAC068A

17

Introduction to the VAC068A

17.1 Features Summary

When used with the VIC068A (VMEbus Interface Controller), the VAC068A(VMEbus Address Controller) forms a complete VMEbus master/slave interface solution. The VAC068A is intended for use solely with VIC068A. The following feature list is VAC068A specific but uses VIC068A implementations for items such as boundary crossing, dual-path, interprocessor communications facilities, and block transfers.

- Complete VMEbus and I/O DMA capability for 32-bit processors, including 42 programmable registers for configuration and control of:
  - — Slave address decode
  - — UARTs
  - — Programmable I/O signals
  - — Local I/O
  - — Interrupt source
- Provides complete local memory map decoding. Separate segments on local interface available for:
  - — DRAM
  - — VME Subsystem Bus (VSB)
  - — Shared Resource
  - — Local I/O
  - — EPROM
- Provides complete VMEbus memory map decoding. Separate segments are available for:
  - — Two VMEbus slave decodes
  - — Interprocessor communication facilities
- Supports block transfers over 256-byte boundaries:
  - — Address counters for VMEbus A[31:8] and local LA[31:8]
  - — Supports dual-path feature of the VIC068A

  - — Supports implementation of the VSB interface
  - — Includes local DMA capability
- Dual UARTs on chip:
  - — Double buffered on transmit, quintuple-buffered on receive
  - — Programmable baud rate from 300 to 9600 baud
- Miscellaneous:
  - — Supports unaligned transfers
  - — Programmable DSACKi* for local I/O
  - — Programmable timer and interrupts
  - — Programmable I/O signals (dual function)
  - — Buffer control signals for direct connection to 543s

17.2 General Description

The VAC068A is a programmable address decode controller and VMEbus DMA extension for the VIC068A. When used in conjunction with the VIC068A, the VAC068A maximizes performance of a master/slave VMEbus interface module. It also substantially reduces power consumption and board space when compared to discrete implementations. The VAC068A contains programmable registers that allow the user to easily define VMEbus address decoding. These are:

- A24 address space overlay register
- three programmable VMEbus boundary registers
- separate A16 address space with programmable D16 or D32 data size

The VAC068A reserves address space for local I/O resources, DRAM, and EPROM. Access to these address spaces forces the proper chip select signal on the VAC068A. The chip select outputs are CS*, IOSEL5–0*, DRAMCS*, and EPROMCS*. DRAM address space is hard coded to start at $0000 0000 to follow normal VMEbus address space conventions. Programmable address options also exist for the purpose of asserting VSB select (VSBSEL*) and shared resources chip select (SHRCS*).

The VAC068A contains address counters and control logic to allow for block transfer over 256-byte boundaries.

Additional features include 13 programmable input/output signals (PIO signals) that can be programmed for the following functions:

- 13 general-purpose I/O signals
- two serial I/O transmit and receive channels (A and B)
- three interrupt signals
- shared resource chip select
- I/O read and I/O write
- I/O selects 2–5 for slower 8-bit peripheral devices

Note: IOSEL0 and IOSEL1 have dedicated pins on the VAC068A.

The I/O selects have reserved address space and may use the local address bus or the IDbus. When the IDbus is used, programmable cycle end and DSACK control may be programmed in the corresponding DSACK control register.

Programmable DSACKi* control also exists for EPROM and shared resource selects. DSACKi*s may also be disabled if the user wishes to provide this function. There is also a programmable timer and interrupt mapping on either PIO7, PIO10, or PIO11 for the following interrupting functions:

- timer interrupt
- UART A and B interrupt
- mailbox interrupt
- PIO4, PIO7, PIO8, or PIO9

The VAC068A uses the VIC068A data direction (DDIR*) and swapping signals (SWDEN*) for direction control and unaligned transfers.

The VAC068A connects directly to the local address bus LA[31:8] and the VMEbus address A[31:8] signals. It also connects to the local data bus through the IDbus ID[15:8]. VAC068A uses the IDbus to provide VMEbus data signal connection D[15:8], although external buffers and line drivers are required. The VIC068A directly drives the VMEbus data signals D[7:0], address signals A[7:1], and local address LA[7:1] and data signals D[7:0].

Both parts utilize Cypress's patented output drivers and were designed with high-performance standard cells using a 1-micron CMOS process. Thirteen ground and nine power pins are provided.

18

VAC068A Signal Descriptions

18.1 VMEbus Signals

A[31:8]

These are the VMEbus address signals.

Drive: 64 mA (all)

Type: Three-state I/O

AS*

This is the VMEbus address strobe signal. It responds to both VIC068A- and VMEbus-generated address strobes.

Drive: None

Type: Input

ID[15:8]

These are the isolated data bus signals. They are used to interface local data [15:8] to the VMEbus D[15:8] in conjunction with transparent latching bidirectional I/O buffers. They also are used to interface with local 8-bit I/O peripherals via the Device Location and DSACKi* Control registers.

Drive: 16 mA

Type: Three-state I/O

18.2 CPU/Local Interface Signals

LD[31:16]

These are the local data bus signals. They are used to write or read the local data bus and for writing and reading the on-chip control registers.

Note: The IDbus connects to LD[15:8] and VIC068A connects to LD[7:0].

Drive: 16 mA

Type: Three-state I/O

LA[31:8]

These are the local address bus signals. They are used as inputs during a VMEbus master cycle and to access on-chip control registers. They are used for output during local or slave accesses.

Drive: 16 mA

Type: Three-state I/O

PAS*

This is the local-processor address strobe. It indicates to the VAC068A that a valid address is present on the address bus. This signal is typically driven by either VIC068A or the local processor.

Drive: None

Type: Input

R/W*

This is the local read/ write signal. When High, this signal indicates that the current cycle is a read. When Low, the current cycle is a write. This signal is typically driven by either the VIC068A or the local processor.

Drive: None

Type: Input

RESET*

This is the reset for the VAC068A. It is used alone or in conjunction with WORD* to reset the VAC068A internal registers. There are two reset types that may be implemented, and both of them are discussed in the reset section.

Drive: None

Type: Input

WORD*

This signal is active under programmable control from the appropriate region attribute register and controls the length of the data field. When it is asserted, the data path is 16 bits

wide. When deasserted, a 32-bit data path is set. It is also used as an input in conjunction with RESET* to set VAC068A registers. It is typically connected to the VIC068A as an output.

Drive: 16 mA

Type: Input/Three-state output

ASIZ1, ASIZ0

These are the address size signals. They are used to specify the address size of an access. They are active under programmable control from the appropriate region attribute register. These signals are typically driven to VIC068A along with WORD* to determine address and data path size.

ASIZ0	*ASIZ1*	*Addressing Mode*
0	0	User-defined
0	1	A32
1	0	A16
1	1	A24

Drive: 16 mA

Type: Three-state output

DSACK1/0*

These are the data sizing acknowledge signals. They are generated for any of the VAC068A device select outputs except CS* and VSBSEL* accesses. DSACK0* or DSACK1* can be selectively disabled or enabled in the DSACK1* Control register.

Drive: 16 mA

Type: Three-state I/O (rescinding)

FC2/0

These are the function code signals. They are used by the VAC068A to determine the local access type and are typically driven by the local processor or the VIC068A as shown in the following tables:

(Per 680X0 User's Guide)

FC2	*FC1*	*FC0*	*Cycle*
0	0	1	User Data Space

0	1	0	User Program Space
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

(Per section 2.2 of this User's Guide)

FC2	*FC1*	*Cycle*
0	0	Slave Block Transfer
0	1	Local DMA
1	0	Slave Access
1	1	DRAM Refresh

Drive: None

Type: Inputs

MWB*

This is the module-wants-bus signal. It is asserted under programmable control of the appropriate region attribute register and indicates that a VMEbus access is occurring. This signal is typically connected to the VIC068A.

Drive: 16 mA

Type: Output

FCIACK*

This is the local interrupt acknowledge signal. It indicates that the current cycle is an interrupt acknowledge cycle. This signal is typically connected to the VIC068A. It is asserted during local VAC068A interrupt cycles, or when HIACKEN is enabled in the PIO Direction register or when IOSEL5* address space is accessed when enabled in the PIO Function register.

Drive: 16 mA

Type: Output

DRAMCS*

This is the DRAM chip select signal. It is asserted when the local address maps into region 0 as defined by the DRAM Upper Limit Address register. It is also asserted when redirection is enabled in the VAC068A Decode Control register.

Drive: 16 mA

Type: Output

EPROMCS*

This is the EPROM chip select signal. It is asserted after a global reset, during a local access to EPROM address space, and during redirection of SLSEL1* on the local bus via the VAC068A Decode Control register.

Drive: 16 mA

Type: Output

FPUCS*

This is the floating-point-unit chip select signal. It is asserted when a floating-point coprocessor access is occurring. This is decoded from the processor function codes or under programmable control in the PIO Function register to be asserted in the IOSEL4* address range.

Drive: 16 mA

Type: Output

VSBSEL*

This is the VSB (VME Subsystem Bus) select signal. It is used to identify accesses to a daughterboard or VSB. It is asserted when enabled from the appropriate region attribute register.

Drive: 16 mA

Type: Output

REFGT*

This is the refresh grant signal. It is asserted during a DRAM refresh cycle and is typically decoded from the VIC068A function codes (FC1 and FC2).

Drive: 16 mA

Type: Output

LBR*

This is the VIC068A local bus request signal. It is used to signal the VAC068A when the VIC068A requests the local bus. It is typically connected to the VIC068A LBR* signal.

Drive: None

Type: Input

CS*

This is the VIC068A chip select signal. It is asserted when the fixed address of the VIC068A is present on the local address bus. This signal is typically connected to the VIC068A chip select signal (CS*).

Drive: 16 mA

Type: Output

BLT*

This is the block transfer signal. It is used to determine when a block transfer is in progress and to increment internal address counters during a boundary crossing. This signal is typically connected to the VIC068A.

Drive: None

Type: Input

CACHINH*

This is the cache inhibit signal. It is asserted when enabled in either the Region Attribute registers or in the A24 Space Base Address register. It is also asserted on access to the DRAM Mailbox and VMEbus A16 address space (Region 6). It may be connected to the CDIS signal on 680X0-type processors.

Drive: 16 mA

Type: Open Collector Output

LDMACK*

This is the local DMA activity signal. It is asserted when there is DMA activity mapped into a particular region. It is typically decoded from the VIC068A function codes (FC1 and FC2).

Drive: 16 mA

Type: Output

CPUCLK

This is the CPU clock signal. It is typically connected to the system CPU clock. Maximum frequency is 50 MHz.

Drive: None

Type: Input

SLSEL0*

This is the slave select 0 signal. It is asserted when enabled by a comparison of its base address register and the address on either the local or the VMEbus. It indicates to the VIC068A that a slave operation is pending.

Drive: 16 mA

Type: Output

SLSEL1*

This is the slave select 1 signal. It is asserted when enabled by a comparison of its base address register and the address on the VMEbus. It indicates to the VIC068A that a slave operation is pending.

Drive: 16 mA

Type: Output

ICFSEL*

This is the interprocessor communications signal. It is asserted under programmable control of a comparison of its base address register and the address on the VMEbus. It indicates a VIC068A interprocessor communication access.

Drive: 16 mA

Type: Output

IOSEL1/0*

These are 2 of the 6 I/O select signals. They are asserted when the local bus address matches their fixed memory location. They are also used in conjunction with the IDbus when programmed in the PIO Function register.

Drive: 16 mA

Type: Output

18.3 Parallel I/O-Shared Function Signals

The functions of these signals are programmed in the PIO Function register. When the corresponding bit is set in this register, the signal is the shared function. When the corresponding bit is cleared, the signals operate in the general-purpose parallel I/O mode (PIO).

PIO0–TXDA

The PIO0–TXDA signal is programmed to serve either as General-Purpose I/O pin bit 0, or as an output for the UART Channel-A Transmit signal.

Drive: 16 mA

Type: Input/Three-state output

PIO1–RXDA

The PIO1–RXDA signal is programmed to serve as either General-Purpose I/O pin bit 1, or as an input for the UART Channel-A Receiver signal.

Drive: 16 mA

Type: Input/Three-state output

PIO2–TXDB

The PIO2–TXDB signal is programmed to serve as either General-Purpose I/O pin bit 2, or as an output for the UART Channel-B Transmit signal.

Drive: 16 mA

Type: Input/Three-state output

PIO3–RXDB

The PIO3–RXDB signal is programmed to serve as either General-Purpose I/O pin bit 3, or as an input for the UART Channel-B Receiver signal.

Drive: 16 mA

Type: Input/Three-state output

PIO4–IORD*

The PIO4–IORD* signal is programmed to serve as either General-Purpose I/O pin bit 4, or as an output for the read enable signal (local I/O accesses).

Drive: 16 mA

Type: Input/Three-state output

PIO5–IOWR*

The PIO5–IOWR* signal is programmed to serve as either General-Purpose I/O pin bit 5, or as an output for the write enable signal (local I/O accesses).

Drive: 16 mA

Type: Input/Three-state output

PIO6–IOSEL3*

The PIO6–IOSEL3* signal is programmed to serve as either General-Purpose I/O pin bit 6, or as an output for the IOSEL3* enable signal (local fixed-map I/O select).

Drive: 16 mA

Type: Input/Three-state output

PIO7–Interrupt Request

The PIO7–Interrupt Request signal is used as either General-Purpose I/O pin bit 7, or as an output for interrupt requests on one of PIO 7, 10, or 11 (programmed in the Interrupt Control register).

Drive: 16 mA

Type: Input/Three-state output

PIO8–IOSEL4*

The PIO8–IOSEL4* signal is programmed to serve as either General-Purpose I/O pin bit 8, or as an output for the IOSEL4* enable signal (local fixed-map I/O select). IOSEL4* accesses also assert FPUCS* when so programmed in the PIO Function register.

Drive: 16 mA

Type: Input/Three-state output

PIO9–IOSEL5*

The PIO9–IOSEL5* signal is programmed to serve as either General-Purpose I/O pin bit 9, or as an output for the IOSEL5* enable signal (local fixed-map I/O select). IOSEL5* accesses also assert FCIACK* when so programmed in the PIO Function register.

Drive: 16 mA

Type: Input/Three-state output

PIO10–Interrupt Request

The PIO10–Interrupt Request signal is used as either General-Purpose I/O pin bit 10, or as a programmed interrupt request as programmed in the Interrupt Control register.

Drive: 16 mA

Type: Input/Three-state output

PIO11–Interrupt Request

The PIO11–Interrupt Request signal is used as either General-Purpose I/O pin bit 11, or as an output for interrupt requests as programmed in the Interrupt Control register.

Drive: 16 mA

Type: Input/Three-state output

PIO12–SHRCS*

The PIO12–SHRCS* signal is programmed to serve as either General-Purpose I/O pin bit 12, or as an output for shared resource chip select.

Drive: 16 mA

Type: Input/Three-state output

PIO13–IOSEL2*

The PIO13–IOSEL2* signal is programmed to serve as either General-Purpose I/O pin bit 13, or as an output for the IOSEL2* enable signal (local fixed-map I/O select).

Drive: 16 mA

Type: Input/Three-state output

18.4 Data Flow Control Signals

These signals are inputs to VAC068A and are connected to outputs from VIC068A.

SWDEN*

This is the swap data enable signal. It is used in conjunction with DDIR* to swap data to or from the Isolated Data bus signals ID[15:8] to the Local Data LD[15:8] bus. This signal is typically connected to the VIC068A.

Drive: None

Type: Input

DDIR

This is the data direction signal. It is typically connected to the VIC068A.

Drive: None

Type: Input

LADO

This is the latch address out signal. It is used to latch the local address out to the VMEbus. It is typically connected to the VIC068A. LADO is also used to increment internal address counters during a VMEbus boundary crossing.

Drive: None

Type: Input

LADI

This is the latch address in signal. It is used to latch the local address from the VMEbus.

Drive: None

Type: Input

LAEN

This is the local address bus enable signal. It is used by the VAC068A to indicate that the VIC068A has bus mastership of the local bus.

Drive: None

Type: Input

ABEN*

This is the VMEbus address enable signal. It is used to indicate that the VIC068A is driving the VMEbus address bus.

Drive: None

Type: Input

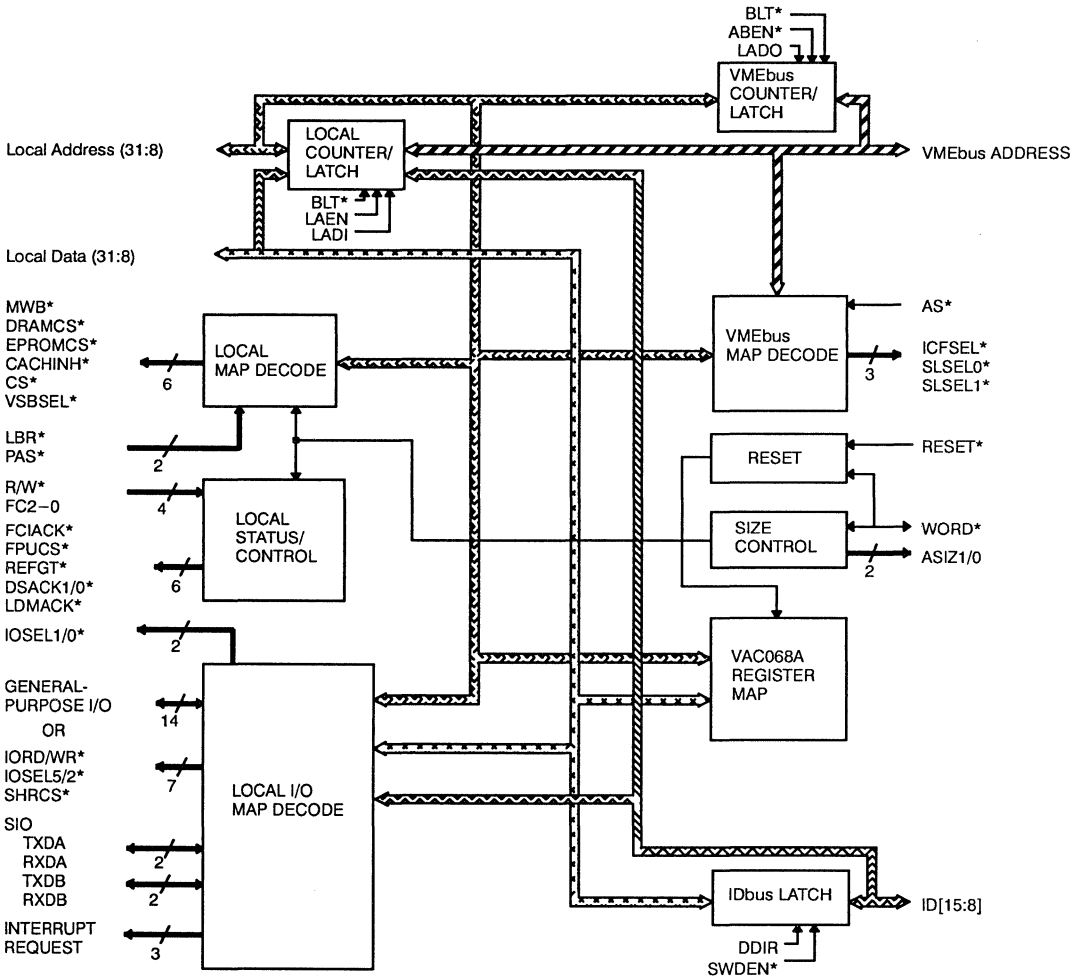


Figure 18–1. VAC068A Block Diagram

19

VAC068A Overview

19.1 Applications

The VAC068A is a complementary chip to Cypress's VIC068A VMEbus Interface Controller. As the VAC068A is intended to work exclusively with the VIC068A, the user should be familiar with VIC068A operation. The VIC068A section of this user's guide must be used in conjunction with the VAC068A section to fully understand the operation of the chip set.

The VAC068A includes drivers and receivers to interface directly to the local address and data bus. For connection to the local address bus, the VIC068A drives the lower local address bus signals LA[7:0] and the VAC068A drives the upper local address bus signals LA[31:8]. For the local data bus, the VIC068A drives LD[7:0] and the VAC068A drives LD[31:8].

For the VMEbus address signals, the VIC068A drives A[7:1] and the VAC068A drives A[31:8]. VMEbus data signals D[7:0] are driven directly by the VIC068A. The VAC068A uses an alternate IDbus to drive the next eight VMEbus data bits D[15:8] with an external '543. An external '245 is needed to swap between LD[15:8] and ID[15:8]. The upper VMEbus data signals D[31:16] are driven through external '543s and enabled by the VIC068A buffer control signals. Thus the only additional logic required with the VIC068A and the VAC068A for a complete VMEbus and local interface are three 543s and three 245s for direction and isolation of data signals from ID[15:8]. The VAC068A internally includes one of the two swap buffers required for unaligned transfers (see VAC068A block diagram). The internal swap buffer moves the ID bus data from ID[15:8] to LD[31:24].

The VAC068A IDbus also allows slower I/O devices to be utilized with system processor clock rates of 25 MHz or higher. This is accomplished by I/O read, I/O write, and DSACKi* programmable time delays. These parameters are set in the DSACKi* Control register along with assertion and recovery times for local I/O chip selects (IOSELi*).

When the IDbus is used to interface to slower 8-bit peripherals, certain registers must be configured. These are the DSACKi* Control register, PIO Function register, and the De-

vice Location register. Region 5 of the VAC068A address map defines each of six different address areas for these 8-bit peripherals with separate I/O select signals for each address area. Two of these select signals have dedicated pins and the other selects are shared with the programmable I/O signals. To enable these devices on the ID bus, the corresponding bit in the Device Location register must be set.

The VAC068A provides VMEbus address decoding when functioning as a slave. This is accomplished by two separate slave select base address registers. These registers are compared to the respective VMEbus address signals and, when a match occurs, the corresponding SLSELi* signal is asserted to the VIC068A. The address resolution of slave selects is 64 Kbytes; i.e., only the upper 16 bits are compared. For A16 slave selects, bits 26 and 27 of slave select 1 must be used. This is enabled in the Decode Control register.

In addition to VMEbus slave selects, the VAC068A can assert a variety of chip selects to the local module. These are activated by internal comparators that compare the local or the VMEbus address to the VAC068A address map. Two means exist for asserting the chip selects. One is when the local address matches the corresponding region address. The other is when redirection is enabled in the Decode Control register. When the VAC068A's own VMEbus slave select address is detected and redirection is enabled in the Decode Control register, the corresponding chip select is asserted. For example, the VAC068A can assert DRAMCS* (after VIC068A asserts LBR* and receives a LBG*) in response to the detection of its own slave select 0 address on the local bus. The slave select address is located in the SLSEL0 Base Address register. SLSEL1* may be redirected to EPROMCS*, DRAMCS*, SHRCS*, or VSBSEL* (if it is enabled in the Decode Control register) when the slave select 1 address is detected on the VMEbus address bus.

The VAC068A also provides address decode for local I/O devices (IOSEL0–5), the VIC068A (CS*), and local resources. The upper address range is reserved for VMEbus short address space (A16), and can be programmed for either automatic D32/D16 decode (based on A16), D16, or D32 data path.

The VAC068A includes local and VME address counters required to support VMEbus block transfers. These counters automatically increment when a 256-byte boundary is crossed. The VAC068A also contains a dual-address path that allows VIC068A master accesses to the VMEbus during the interleave period of a block transfer.

The VAC068A may be used in conjunction with the VIC068A for local DMA transfers. This includes transfers from memory to local I/O or across the VSB (VME Subsystem Bus). That is, the VIC068A/VAC068A architecture allows design of VSB and I/O LSI devices that utilize DMA capability.

The VAC068A decodes processor and VIC068A signals to identify the current state of the local module. For example, the function codes are decoded from the VIC068A to interpret whether the current local cycle is a slave block transfer, normal slave transfer, local DMA cycle, or DRAM refresh cycle. The result of this decode is used to determine the slave cycle type and to assert the signals LDMACK* (local DMA activity) or REFGT* (refresh grant).

FC2	*FC1*	*CYCLE*
0	0	Slave Block Transfer
0	1	Local DMA
1	0	Slave Access
1	1	DRAM Refresh

The VAC068A includes circuitry to decode the processor's function codes. This is used to supply floating-point-unit chip select (FPUCS*) to the modules coprocessor.

The VAC068A also identifies if the current cycle is an interrupt acknowledge cycle. This is done through the assertion of FCIACK* by the VAC068A when any one of these three conditions occur:

- function code bits 0–2 are all set to 1's
- bit 30 of the PIO Direction register is set and an access is made to $FFFF FFxx
- bit 31 of the PIO Function register is set and there is an access to local I/O 5 address space

The VAC068A local interrupts can occur on any of three interrupt-request output signals: PIO7, PIO10, or PIO11. These signals are enabled as output interrupt signals through the PIO Function register.

Enabling/disabling specific interrupts is accomplished in the Interrupt Control register. This register also specifies the mapping of interrupts on PIO7, 10, or 11. Normally, the PIO7, 10, or 11 signals are connected to the VIC068A LIRQi* signals. The Interrupt Status register may be read to determine which interrupt condition is causing the output to be asserted. The conditions causing these interrupts are:

- programmable timer
- mailbox access
- UART channel A or B service
- programmable I/O signals PIO4, PIO7, PIO8, or PIO9

The timer, mailbox, and PIO4, 7, 8, and 9 interrupts are active Low and edge triggered. The interrupt request goes inactive when the local processor clears its respective bit in the Interrupt Control register.

To disable the UART channel A and B interrupts, the user must first determine what caused the UART interrupt. The cause of the interrupt is located in the respected UART Serial I/O Channel Interrupt Status register. This register is read-only. Once the cause is determined, the respective bit in the UART Serial I/O Channel Interrupt Mask register must be disabled. Upon completion of this activity, the Interrupt Control register bit may be cleared.

Other signals the VAC068A decodes include:

- BLT* (local block transfer address counter)
- LBG* (VIC068A local bus activity)

The VAC068A also decodes the VIC068A buffer control signals. These include:

- SWDEN* (swap data enable)
- DDIR* (data direction)
- LAEN (local address enable)
- LADO (latch address out)
- LADI (latch address in)
- ABEN* (VME address bus enable)

The VAC068A also furnishes status and size signals to the VIC068A and the local processor. These include:

- WORD* for data size
- ASIZ1/0 for address size
- LDMACK* for local DMA activity
- CACHINH* cache inhibit signal
- REFGT* for refresh grant

The VAC068A also includes shared-function programmable I/O signals. These dual-function pins are defined in the PIO Function register. When not serving as general-purpose I/O signals they may be used for:

- dual UART channels (A and B)
- I/O read enable for local I/O
- I/O write enable for local I/O
- Local I/O chip selects (2–5)
- Shared-resource chip select

The VAC068A also includes a timer with prescaler, programmable DSACKi* control, and a chip select (ICFSEL*) for access to the VIC068A global and module switches from the VMEbus.

19.2 VMEbus Address Decoding

The VAC068A's VMEbus address map consists of an address region for VMEbus A16 master cycles. The A24 Space Base Address register specifies where the A24 address overlay is for A24 master cycles. This register also contains the bits to program the data size of the A16 and A24 master cycles. An automatic decode of the data size is accomplished through the VAC068A by monitoring LA[16] for A16 space and LA[24] for A24 space. If this address signal is High, a D16 data path is selected (WORD* asserted). If this address bit is Low, a D32 data path is selected (WORD* deasserted). A fixed data size option also exists.

There are three programmable regions that can be programmed to assert MWB*, VSBSEL*, or SHRCS*. MWB* signals the VIC068A to acquire the VMEbus for a master access. Any or all regions are capable of this operation. VSBSEL* and SHRCS* are chip selects to module resources.

The VAC068A also has two slave select base address registers, two slave select mask registers, and an interprocessor communications select address register. The slave select base address registers and mask registers are used to decode valid slave selects and generate SLSELi* to the VIC068A. Any of A32, A24, or A16 slave access can occur. The ICFSEL address register is used to access the VIC068A global and module switches from the VMEbus.

19.2.1 Master Access

There are five types of master accesses that may be accomplished with the VAC068A. They are:

- A32 address with programmable data size
- A24 address with D16
- A24 address with D32
- A16 address with D32
- A16 address with D16

19.2.2 Programmable VMEbus Space

Any or all of the three programmable regions can be used to set up VMEbus accesses by defining an address range and programming the attribute register to assert MWB*. The address range is configured by programming the Boundary 2 or 3 Address registers. When the

Region Attribute register is set for MWB* assertion, the VIC068A arbitrates for the VMEbus. The corresponding ASIZ1/0, WORD*, and CACHINH* settings are driven upon the assertion of the processor address strobe (PAS*). See the VIC068A section of this user's guide for details on VMEbus master transfers.

19.2.3 A24 VMEbus Space

The A24 Base Address register specifies an A24 address space overlay. This overlay is 32 Mbytes in size. The data path may be D16, D32, or automatically decoded. Bit 24 of the A24 Base Address register determines if the entire region is D16 or D32. Automatic decode is enabled through bit 20 of the A24 Base Address register. The A24 address space data path is determined by LA[24]. If LA[24] is High, the data path is D16. If LA[24] is Low, the data path is D32. WORD* is driven accordingly on these master cycles.

This address space can be divided into two 16-megabyte blocks if automatic decode of LA[24] is enabled. The first 16 Mbytes are D32 and the second 16 Mbytes are D16. If automatic data-path decoding is not used, bit 24 of the register determines the data path of the entire 32-Mbyte overlay. The A24 address space may be overlaid on the top of any of the three programmable regions (1, 2, or 3) regardless of what it is set for (i.e., VSBSEL*, MWB*, SHRCS*, or inactive).

The VAC068A compares the upper 7 bits [31:25] of the A24 Base Address register to local address bits LA[31:25] and, upon a match, overlays 32 Mbytes of A24 address space on the programmed region. When this overlay occurs, MWB* is asserted and an A24 VMEbus master cycle takes place. The VAC068A drives WORD*, ASIZ1/0, and CACHINH* as specified by the settings of the A24 Base Address register.

The requirements for forcing A24 address accesses are that it must fall between the address range of $02 and $FE00 0000 and that the A24 overlay address is above the DRAM region (region 0). This address space decode cannot be disabled.

As in a programmable master access, MWB* is asserted and the VIC068A requests the VMEbus and initiates the transfer per VMEbus protocol.

19.2.4 A16 VMEbus Space

The upper address space, starting at $FFFE 0000 and continuing to $FFFF FFF0, is reserved for A16 VMEbus accesses. This address space is fixed and should not be used for any

other purpose. There are two ways to set up the data size for A16 master accesses. The first is to allow the VAC068A to automatically decode LA[16]. This is similar to the automatic decode of the A24 VMEbus space. If LA[16] is High, the data path size is D16 (WORD* asserted). If LA[16] is Low, the data path size is D32 (WORD* deasserted). This decode reflects region 6 of the VAC068A address map as A16/D32. The other option is to set bit 22 in the A24 Space Base Address register. This enables bit 21 of the A24 Space Base Address register to control the data path size. If bit 21 is set, the data path is D32 (WORD* deasserted). If bit 21 is clear, the data path is D16 (WORD* asserted). CACHINH* is always asserted during VMEbus A16 access and is not programmable.

Region 6 and the programmable regions provide the only way for an A16 access to take place when using the VAC068A. As before, MWB* assertion triggers the VIC068A to acquire the VMEbus and initiate the VMEbus transfer.

19.3 VMEbus Slave Access

The VAC068A contains four registers to define VMEbus slave address decode. These are the Slave Select 1 and Slave Select 0 Base Address registers, and Slave Select 1 and Slave Select 0 Address Mask registers. The VAC068A also contains an Interprocessor Communications Select register. This is used to access the VIC068A global switches, module switches, and communication registers.

The base address registers are loaded with the address that determines a valid slave select to assert SLSEL0* or SLSEL1*. The mask registers are used to qualify which bits in the base address register are to be compared to its respective VMEbus address signal. When a bit is set in the mask register, it allows the base address register contents to be compared to the corresponding VMEbus address signal. If the compare matches, the corresponding SLSELi* is asserted. If clear, no compare is made and the VAC068A does not care what value is on that particular VMEbus address signal. It is important to use these mask bits with care. It is possible to get multiple slave selects if a small number of mask bits are set (i.e., 0's in the high-order mask bits).

The lower VMEbus address bits may also be compared for an A16 slave access. This is accomplished by setting bit 26 of the Decode Control register. When this bit is set, VMEbus A[15:8] is compared to the Slave Select 1 Base Address register bits [31:24]. This allows the user to decode VMEbus short address space and assert any of DRAM, EPROM, VSB, or shared resources chip selects. The chip select is determined using bits 28–29 in the Decode Control register.

The VAC068A address decoder constantly compares the register values to the VMEbus address, and when a match occurs it asserts the proper slave select signal. When SLSELi* is asserted to the VIC068A, the VIC068A asserts LBR* (qualified by VMEbus AS*), which is also connected to the VAC068A's LBR* signal. When LBG* is asserted by the modules local bus arbiter, the following sequence occurs:

1. the VIC068A asserts LAEN with valid FC2/1, SIZ1/0, and LA[7:0]
2. the VAC068A asserts LA[31:8]
3. the VIC068A asserts PAS* and DS*
4. the VAC068A drives ASIZ1/0 and WORD*, then asserts the proper chip select
5. the VAC068A asserts DSACK0/1 (if enabled)
6. the VIC068A times out SAT delay
7. the VIC068A asserts DTACK* and VAC068A deasserts DSACK0/1 with PAS* deassertion
8. the cycle completes, VAC068A three-states its address signals

Slave select 0 is always associated with DRAMCS*. This assumes that a local bus grant on assertion of SLSEL0* results in an access of local DRAM. Similarly, slave select 1 can be enabled to assert any of the following chip selects:

- SHRCS*
- EPROMCS*
- DRAMCS*
- VSBSEL*

This chip select is asserted according to the Decode Control register bits 28–29 (FFFD 14xx).

Slave select 1 is only asserted when slave select 0 is not asserted per an internal interlock. Slave select 0 is asserted for slave transfers matching its base address register (and when the proper mask bits are set) or redirection of the local address to DRAM and DRAMCS* assertion.

If both slave selects are in use and each is enabled for a different address space, it is possible for both SLSEL0* and SLSEL1* to be asserted simultaneously as in the following case:

When bit 26 is set in the Decode Control register, the VAC068A compares the VMEbus address signals A[15:8] to SLSEL1* Base Address register bits [31:24]. If a match occurs

with SLSEL0* address bits [31:24] and VME address signals A[31:24], qualified by the address mask register, then both selects are asserted. The slave select address mask register would also have to be enabled for a compare of these address bits. When this occurs, and the two slave selects are mapped to different regions, the local module must decide which slave select has precedence.

If both SLSEL0* and SLSEL1* are asserted and point to the same device (i.e., DRAMCS* via SLSEL1* redirection in the Decode Control register) or they are decoded at non-overlapping address ranges in the same address space, no conflict should arise. In this case, to access the full address space, slave select 0 should correspond to the larger address space for access to the Mailbox region.

For recognition of the slave select address on the local bus, additional comparators monitor the local bus for the SLSEL0 address range. If this function is enabled in the Decode Control register bit 19, the VAC068A asserts DRAMCS* when it recognizes a valid slave address on the local address bus. This allows the local processor to access data in DRAM at the same address as other modules on the VMEbus.

Additionally, the slave address may be determined by using the VIC068A's ability to assert LBERR* when it sees a qualified slave select. This is called self-access. The resulting operation is:

- VMEbus BERR is driven by the VIC068A
- LBERR is driven by the VIC068A
- the VIC068A Bus Error Status register indicates a self-access has occurred

The Interprocessor Communications Facility Select register is another type of slave select. It enables an A16 access to VIC068A internal global switches, module switches, and communication registers. This register compares VMEbus A[15:8] to each of the two bytes in the register. When a match occurs, the VAC068A asserts ICFSEL* to the VIC068A. The upper byte is normally used for global accesses to the four Interprocessor Communications Global switches. If used for this function, all VAC068A (and all other non-VAC068A modules) must be set to the same value. The lower byte is used to access the Interprocessor Communications Module switches and Interprocessor Communications registers. When used for this function, the values must be different for each VMEbus module. The VIC068A section of this user's guide should be referenced when accessing specific switches and registers.

19.4 Local Memory Map Decoding

The VAC068A segments the local processor's address space into a number of fixed- and variable-sized segments with a resolution of 64 Kbytes (i.e., bits [31:16]). Separate segments are available for

- DRAM
- VMEbus subsystem bus select (VSBSEL*)
- shared resource chip select (SHRCS*)
- EPROM
- local I/O (I/O selects 5–0, VIC068A and VAC068A register access)

The DRAM*, VSBSEL, and Shared Resource segments are programmable. These regions occupy address space from $0000 to $FFFD FFFF. DRAM is hard-coded to start at $0000 0000, thus making it region 0. The means for sectioning VSBSEL* or SHRCS* in regions 1, 2, or 3 are in the respective Region Attribute register. Boundary Address registers 2 and 3 divide the respective contiguous regions. Each of these regions has the following attribute settings associated with it:

- cache inhibit (CACHINH*)
- address size (ASIZ1/0)
- data word size (WORD*)

In addition to the above programmable segments that may be assigned boundaries and attributes, fixed-size attribute segments are mapped to EPROM and local I/O, which include IOSEL5–0* and the VIC068A and VAC068A register maps. Areas also exist for assertion of MWB* in any of the three programmable regions, VME A16 address space, and an A24 address overlay space (see VAC068A memory map for graphical representation).

19.4.1 DRAM Decode

DRAM is hard-coded to start at $0. This region continues to the value programmed into the DRAM Upper-Limit Mask register. The minimum value that can be programmed in this register is 0000 0001. A 256-byte region is always available for the DRAM mailbox area. This register value is also the starting address for region 1.

An interrupt can be enabled in the Interrupt Control register to indicate an access to the mailbox address space. DRAMCS* is asserted for $0000 00xx after the Force EPROM mode is exited.

The DRAM Upper-Limit Address register is NANDed with the local address bus LA[31:16] to determine the output of the compares. If any are Low, the access is not to DRAM. If all compares are High, then DRAM is being accessed and DRAMCS* is asserted.

There is another way to assert DRAMCS* and access DRAM address space using bits in the Decode Control register to redirect the Slave Select 0 address when it is present on the local address bus. Qualification of DRAMCS* assertion may be with or without PAS* assertion. This is programmable in the Decode Control register. DSACKi* control for DRAM access is also available in the Decode Control register. DRAM accesses may be set for a 32-bit data path, or VAC068A DSACKi* may be three-stated and external DSACKi* generation used. The three-state option is useful for processors that need synchronous termination of DRAM access as with the Motorola 68040.

For redirection of the SLSEL0* address, bit 19 in the Decode Control register must be set. The slave select base address register must be present on the local address bus. If this bit is clear, DRAMCS* is not asserted when the SLSEL0* address is present on the local bus. The correct mask bits must be set to enable a compare of the address bits.

The user may also elect to enable qualification of DRAMCS* with PAS* asserted by setting bit 30 in the Decode Control register. This bit is normally set to condition the VAC068A to look only at the address during the proper time period.

It should be noted that DRAMCS* is always asserted during a VMEbus SLSEL0* cycle unless redirection is disabled. DSACKi* may also be three-stated on LAEN assertion for VMEbus slave cycles. This is accomplished using bit 31 of the Decode Control register. The user may chose to set the DSACKi* delay for DRAM accesses in the Decode Control register using bits 17–18. The time intervals are in CPUCLK cycles, from 0 to 3.

The upper 256 bytes of DRAM serve as a mailbox area. If the mailbox interrupt is enabled via the Interrupt Control register, a write to the upper 256 bytes of DRAM via SLSEL0* interrupts the local processor with the specified interrupt signal.

19.4.2 Programmable Decode

Following the DRAM Upper-Limit Mask register are two Boundary Address registers. These registers define the next three regions of address space that may be assigned to any of the following chip selects or VMEbus access:

- SHRCS* (Shared Resource Chip Select)
- VSBSEL* (VMEbus Subsystem Bus Select)
- inactive (not programmed)

Normally SHRCS* is used as an SRAM chip select, although it can be used for any shared resource on the module including DRAM. When used in this fashion, it has no effect on the VIC068A block transfers with local DMA.

When configured for "inactive," the region attributes are still driven as programmed when the region is accessed. These attributes can be qualified with PAS* when enabled in the Decode Control register bit 20.

Since the DRAM Upper-Limit Mask register defines the upper address space of DRAM, it is also the starting address for region 1. This means that the Boundary 2 Address register specifies the upper limit of region 1 and the starting address of region 2. The Boundary 3 Address register specifies the upper address limit of region 2 and the starting address of region 3. Care should be exercised in programming region 3 so that it does not overlap the EPROM address space. If this occurs, EPROMCS* is asserted with the chip select programmed in the Region 3 Attribute register. If this situation arises, external logic must decode chip selects.

19.4.3 EPROM Decode

The EPROMCS* address starts at $FF00 0000 and continues to $FFEF FFFF. This gives the user 15 Mbytes of dedicated EPROM address space. When the user accesses this address space, the VAC068A asserts EPROMCS*. The default EPROM data size is 32 bits. Different EPROM data sizing is selected by setting signals ID8 or ID9 upon power-up reset. If ID9 is Low during RESET*, a 16-bit data path is selected. If ID8 is Low during RESET*, an 8-bit data path is selected. After reset, the user must configure the data size acknowledge in the EPROM DSACKi* Control register bits 27 and 28. It is assumed that the IORD, IOWR, and IOSEL assertion/recovery times are not to be used for EPROM accesses.

The DSACKi* Control register also allows for different-speed EPROM to be used on the module. Bits 29–31 are used to set proper DSACKi* assertion timing or disable DSACKi* assertion by VAC068A altogether.

The EPROMCS* timing should be determined upon power-up by loading the DSACKi* time in the DSACKi* Control register. There is also a "Force EPROM" mode that the VAC068A initially enters upon power-up. This mode is exited by any memory access in the EPROM address space ($FF00 0000 to $FFEF FFFF).

19.4.3.1 Forced EPROM Mode

The map decode function is overridden at power-up to force read accesses to EPROM independent of its mapping until the EPROM address appears on the local address bus. This implies a jump to EPROM address space ($FF00 0000 to $FFEF FFFF) should be one of the first instructions in the boot-up sequence. An initialization routine is given in Chapter 20. This routine or a similar one should be initiated after exiting from system reset. When the forced EPROM mode is exited, DRAMCS* is asserted for address $0000 00xx.

19.4.4 Local I/O Select Decode

The local I/O address space begins at $FFF0 0000 and ends at $FFFD FFFF. IOSEL5–0* accesses, VIC068A registers, and VAC068A registers reside in this address space. This address space is available to the local module only. The IOSELi*s each occupy 128 Kbytes of address space starting at $FFF0 0000 to $FFFA 0000. The VIC068A register accesses start at $FFFC 0000 and the VAC068A register accesses start at $FFFD 0000. Thus each register map consumes 64 Kbytes of address space. The upper limit for this region is $FFFD FFFF. Each IOSEL5–0* has an individual DSACKi* Control register associated with it. IOSEL5–0* also have a Device Location Register attribute that enables the particular IOSEL5–0* device to be located on the ID bus. IOSEL5/2* do not have their own dedicated signal on the VAC068A, instead they share a PIO signal function and must be enabled in the PIO Function register. IOSEL1/0* signals have their own pin and do not share functionality.

In the DSACKi* Control register, it is possible to set IORD* and IOWR* assertion and deassertion parameters. These are commonly called "cycle end control." Cycle end control is defined as a means to provide increased hold time to peripheral devices located on the local module. Qualification of IORD* or IOWR* deassertion is either with PAS* or when the DSACKi* assertion delay has elapsed. For assertion of IORD* or IOWR*, the delay is programmed in half CPUCLK cycles after PAS* assertion. The IORD* and IOWR* assertion and recovery delay is not used for EPROM or shared resources selects.

IOSEL5–0*s are available to the user in one of two modes of operation. The first mode is when they are located on the local address and data bus. This is the typical operating condition. The second mode is when they are used on the ID bus. This mode is used when slow peripherals reside on the local module.

When the ID bus is used, the Device Location register should be properly programmed. This register indicates to the VAC068A which device is located on the ID bus. Latching is

provided internal to the VAC068A. When using the IOSEL5–0*, no I/O device access is allowed to start until the select signal for the previously accessed I/O device has been deasserted for the number of clock cycles configured in the DSACKi* Control register. Programmable attributes for IOSEL5–0*s in the DSACKi* Control register are:

- assertion delay for the IOSEL5–0* from PAS* in half CPUCLK cycles (otherwise assertion is with PAS*)
- deassertion delay (recovery time) for IOSEL5–0* inactive in CPUCLK cycles

EPROMCS* and SHRCS* devices do not use these timing parameters.

The VAC068A registers are also located in region 5 address space. They are accessed at local address $FFFD 0000 through $FFFD 0029. Accesses to these registers occur on the local address and data bus. Acknowledge occurs with the assertion of DSACK1*. These registers are not byte-accessible and must be set upon power-up to configure the VAC068A operational mode. The undefined bits are read as 1's. When all registers are configured, the VAC068A ID register must be written to enable decode and control functions.

19.5 Local Decode Control/Status

The VAC068A decodes function codes FC2–0 to provide status and control signals to the local module. These signals include

- FCIACK* for local interrupt acknowledge cycle
- FPUCS* for floating-point coprocessor chip select
- REFGT* for refresh grant
- LDMACK* for local DMA activity
- CACHINH* for cache inhibit status

19.5.1 Function Code Decode

The VAC068A decodes FC2–0 from the processor and FC2–1 from the VIC068A. To determine when function codes from the VIC068A are to be decoded, the signal LAEN must be asserted. For local processor function decodes, LAEN is not asserted. The functions decoded on the local bus from the VIC068A are one of the following:

- Slave transfer (normal)
- Slave block transfer

- DRAM refresh cycle
- Local DMA

When these activities are detected by the VAC068A, the proper output signals are asserted. When the VIC068A drives the function codes (defined in the VIC068A section of this user's guide) the REFGT* and LDMACK* signals are decoded and asserted.

When decoding local processor cycles (LAEN deasserted), the VAC068A is capable of asserting FCIACK* and FPUCS*. These signals can only be generated when the local processor function codes specify CPU space (FC2–0 = 111). The FCIACK* signal, used for interrupt acknowledge to the VIC068A, is asserted when FC2–0 = 111 and LA[17:15] = 111. The FPUCS* signal, used to select a floating-point coprocessor, is asserted when FC2–0 = 111 and LA[17:13] = 10001.

Care should be taken when these signals are used in a robust system that makes use of coprocessors or other parts of CPU space as only those local address bus bits listed are decoded.

FPUCS* may also be asserted through an access to IOSEL region 4. This is enabled by setting bit 31 of the PIO Function register.

Assertion of FPUCS* is further qualified by bit 16 in the Decode Control register. This allows the assertion of FPUCS* to occur either when the CPUCLK goes Low or when PAS* is asserted.

Both FCIACK* and FPUCS* are inhibited when the VIC068A owns the local bus (LAEN is asserted).

19.6 Programmable Input/Output

The programmable I/O (PIO) signals of the VAC068A provide a general-purpose I/O facility that can alternatively be programmed to provide IORD* (I/O read), IOWR* (I/O write), IOSEL5/2* (I/O selects), interrupt, SHRCS*, or Serial I/O (asynchronous serial I/O) functions. Multiple registers are used to configure the operation of these pins and read status of a particular function. These include PIO Function, Data Out, Pin, and Direction registers.

When the PIO signals are configured in the PIO Function register as general-purpose I/O signals, the user may choose to use these signals to support the serial I/O function on VAC068A (i.e., RTS, CTS, DCD).

Bits 30 and 31 of the PIO Function register have special functions. Bit 30 enables a debounce delay circuit associated with PIO9 (see section 19.8.1). When bit 31 is set, it enables the user to assert FPUCS* on accesses to IOSEL4* address space and assert FCIACK* on accesses to IOSEL5* address space. This is useful for decoding an interrupt acknowledge cycle to obtain the status/ID of a service routine.

The PIO Data Output register contains data to be put out on the PIO pins that are defined as outputs. Reading this address causes the data bus to be driven with the contents of the register. Writing to this register causes the value in the PIO Data Output register to be driven on the PIO pins that are defined as outputs in the PIO Function register.

The PIO Pin register can be read to examine the data being presented on the PIO inputs. This register is a read-only register. Reading the PIO Pin register does not affect the contents of the PIO Data Output register. Writing to the Pin register causes a DSACK1* assertion, but has no effect on the data present on the PIO pins.

The PIO Direction register specifies the direction of the PIO signals. When set the signal is an output, when cleared it is an input. This register has no effect if the corresponding PIO Function Register bit is set to disable the general-purpose I/O capability of that pin. Bit 30 of the PIO Direction register has a special function (HIACKIN). It allows FCIACK* to be asserted on accesses to $FFFF FFxx. This is used to allow non-680x0 processors to assert FCIACK*.

19.6.1 Serial I/O

The VAC068A contains a dual, full-duplex UART. Each channel is double-buffered on transmit and quintuple-buffered on receive. The UARTs always transmit two stop bits and require a single stop bit on receive.

The UART Serial I/O Channel A and B Mode registers allow configuration for

- looping of transmitter or receiver
- break transmission or no break
- enable the transmitter/receiver pair
- transmitter/receiver reset and run
- baud rate selection from 300 to 9600 baud as well as intermediate frequencies (via the CPU Clock Divisor register)
- 7 or 8 data bits per character
- odd, even, or no parity generation and check

The CPU Clock Divisor register must be loaded with a value to produce the correct baud rate generation. Examples are given in the register description section for different CPUCLK rates.

Each UART contains a four-byte-deep FIFO (UART Channel A and B Receiver FIFO register) for receiving serial information. These FIFOs are accessed through the receiver FIFO registers. Included in these registers are indications of errors in parity, frame, and break detection. Once a character is read, the next character becomes available to read.

The Channel A and B Transmit Data registers are accessed from the local data bus. They are loaded with the data to be transmitted. When enabled to run, the next character may be written into the register.

The Interrupt Control register is configured to indicate which interrupt signal (PIO7, 10, or 11) is driven as a serial I/O interrupt. Serial I/O interrupts are not cleared by disabling the UART interrupt alone. The following sequence should be followed when handling a serial I/O interrupt:

1. determine which UART channel (A or B) interrupt is pending in the Interrupt Status register
2. determine the cause of the interrupt in the UART Channel (A or B) Interrupt Status register
3. mask the interrupt in the UART Channel (A or B) Interrupt Mask register
4. service the interrupt based on the status
5. clear the Interrupt Control register for the specific UART interrupt

The UART Channel A or B Interrupt Mask registers can be configured to interrupt the local module for other conditions. These conditions include:

- transmitter empty; transmitter ready
- single character received
- FIFO full
- break change
- parity error
- frame error
- overrun

19.6.2 I/O Select

The VAC068A incorporates individual local I/O select signals. IOSEL1 and 0 are individual signals and are not multiplexed with other function. IOSEL2 through IOSEL5 share func-

tions with PIO13, PIO6, PIO8, and PIO9 signals, respectively. As stated previously, these signals have the ability to communicate on the local data bus LD[31:16] or the ID bus ID[15:8].

The Device Location register specifies mapping of the IOSEL5–0* signals. When a bit is set, it indicates that the respective IOSEL5–0* is located on the ID bus instead of the local data bus.

The PIO Function register controls the multiplexed signal selection. When a bit is cleared, the signals are in the general-purpose I/O mode, otherwise they have the shared function.

IOSEL4* has a special function associated with it. When PIO Function register bit 31 is set, FPUCS* is asserted on accesses to IOSEL4* address space. Additionally FCIACK* is asserted on access to IOSEL5* address space. Another way to assert FCIACK* is to set bit 30 in the PIO Direction register (HIACKEN). When this bit is set, FCIACK is asserted on access to $FFFF FFxx.

The other signals of concern when using the IOSEL5–0* function are IORD* (I/O Read) and IOWR* (I/O Write). These signals are multiplexed with PIO3 and PIO4 respectively. Control for IORD* and IOWR* is located in the DSACKi* Control register. If these signals are to be used as IORD* and IOWR*, the PIO Function register must be set accordingly.

CACHINH* is enabled by bit 23 in the A24 Space Base Address register for accesses to region 5. This includes IOSEL5–0*, VIC068A, and VAC068A register accesses.

19.7 Interrupt Support

The VAC068A may be configured to generate up to three interrupt requests that are designed to connect to local interrupt request signals on the VIC068A. The interrupting functions are serial I/O, timer, mailbox, and PIO interrupt. The various interrupt requests can be multiplexed on up to three of the PIO signals, as configured in the Interrupt Control register. These interrupt requests are typically connected to the VIC068A LIRQi* signals.

19.7.1 Interrupt Status Register

The Interrupt Status register allows the processor to determine which of several possible events caused an interrupt. Except for the UARTs, a serviced interrupt request is withdrawn

when the processor clears its mapping bit in the Interrupt Control register ($FFFD 16xx). A separate Interrupt Status register is implemented for each of the serial I/O channels (channel A at $FFFD 25xx and channel B at $FFFD 26xx). These are cleared by determining the cause in the Interrupt Status register, then masking the interrupt in the Interrupt Mask register, then clearing the interrupt in the Interrupt Control register.

19.7.2 PIO Interrupt

The interrupt output signals share functions with PIO7, PIO10, and PIO11. These output signals are mappable from any of the following sources:

- timer (register-controlled)
- SIO Channel A and B (register-controlled)
- mailbox (access to upper 256 bytes of DRAM address space)
- PIO4 (user defined)
- PIO7 (user defined)
- PIO8 (user defined)
- PIO9 (user defined)

An interrupt request is generated if a falling edge is present on PIO4, 7, 8, or 9 input signals and output on PIO7, 10, or 11. The PIO interrupts are enabled in the Interrupt Control register. The interrupt sources are active Low and edge-triggered (except UART channel A and B). PIO4, PIO7, and PIO8 can be used for VSB write post failure, parity error, or any other user-defined interrupt. PIO9 has a special debounce delay when enabled in the PIO Function register. A change of state on this input is not recognized until it has been stable for 255 CPUCLK cycles. This provides a debounce delay of 26.7 ms when a 9600 baud rate is generated from the CPU Clock Divisor register.

19.7.2.1 Timer Interrupt

The timer interrupt is enabled by loading a value into the Timer Data register and configuring the Timer Control register per the user's requirement. The timer consists of a 16-bit programmable timer with a 6-bit prescaler. Interaction with the timer is by means of a Timer Control register and a Timer Data register. The control register contains a 6-bit prescaler, which is loaded with a count value. The carry of this counter clocks the value loaded in the Timer Data register. A run/load bit and a once/continuous bit are also included. A prescale value of 0 results in a DC prescale output. When the timer control register is read, LD[15:8] are driven with the instantaneous state of the prescaler counter and the value loaded in the

prescale counter. Whenever the timer overflows, or when it is disabled, the timer is loaded with the contents of the Timer Data register. When the Timer Data register is read, the data bus is driven with the instantaneous state of the timer (e.g., the 16-bit counter). The prescaler counter is clocked by CPUCLK.

19.7.2.2 Serial I/O Interrupt

The SIO channel A and B interrupts are enabled in the UART Serial I/O Channel Interrupt Mask registers. UART A and B interrupts are the only interrupts that are level-sensitive and not edge-triggered. All others are edge-triggered. Any of several interrupting conditions on the UART ports may cause an interrupt. These include transmit ready, transmit shift register empty, receiver FIFO full, received character ready, break change received, and error conditions such as overrun, frame, or parity. They can be masked in the UART Interrupt Mask registers (A at $FFFD 23xx and B at $FFFD 24xx) and monitored in the UART Interrupt Status register (A at $FFFD 25xx and B at $FFFD 26xx). The SIO interrupts are cleared in the UART Interrupt Mask registers.

19.7.2.3 Mailbox Interrupt

The mailbox interrupt is enabled in the Interrupt Control register. It is generated by writing to the top 256 bytes of local DRAM as defined by the DRAM Upper-Limit Address register. This mailbox region is always present in the VAC068A DRAM address space. The mailbox interrupt is activated by a slave access using SLSEL0* or when the local processor's access to the SLSEL0* address is redirected to local DRAM. It may also be generated by redirecting SLSEL1* to DRAM in the Decode Control register. The CACHINH* signal is asserted on accesses to the mailbox region.

19.8 Miscellaneous Features

Individual features of the VAC068A including the PIO9 Debounce, ID bus, DSACKi* control, local DMA support, and IORD* and IOWR* are discussed here.

19.8.1 PIO9 Debounce

PIO9 has a special debounce circuit associated with it. This makes it suitable for mechanical switch interrupt input. Switch debouncing is accomplished using a counter. The counter is clocked at the maximum rate of the baud rate timing chain. Because of the debounce circuit,

PIO9 must be held Low for 255 clock cycles of this counter in order to generate an interrupt. This provides a debounce circuit delay of 26.7 ms if a 9600 baud rate is generated from the CPU Clock Divisor register. The debounce delay is enabled by bit 30 of the PIO Function register.

19.8.2 Isolated Data Bus

The VAC068A pinout includes ID[15:8], the isolated data bus. On the module, a '245 is connected between LD[15:8] and ID[15:8] and a '543 is connected between ID[15:8] and D[15:8]. The VAC068A provides the data swap connection between D[15:8] and LD[31:24]. Analysis has shown that connecting peripheral controller chips directly to the 68030's data bus is difficult at higher processor-clock frequencies. Accordingly, systems using the VIC068A/VAC068A should be designed to connect low-speed peripheral devices to ID[15:8]. The VAC068A enables this data path on accesses to such I/O devices, as well as when SWDEN* is asserted by the VIC068A, and provides data latching and output enable control to ease interface timing. The DDIR* signal provides direction flow.

19.8.3 Programmable DSACKi* Timing

The VAC068A generates DSACKi*s with programmable timing for each of its device select outputs (i.e., IOSEL5–0*, SHRCS*, and EPROMCS*) except the VSBSEL, the VIC068A, and the VAC068A register accesses.

Upon power-up, DSACKi* sizing for EPROM space is as follows. For an EPROM data path of 16 bits, force ID[9] to a 0 at power-up. If an 8-bit data path is needed, force ID[8] to 0 at power-up. The default DSACKi* assertion for EPROM (prior to the DSACKi* EPROMCS* Control register being written) is for a 32-bit data path (ID8 and ID9 High).

The VAC068A asserts DSACKi*s on the processor access to DRAM with programmable wait state timing (set in the DSACKi* Control register). When used with the Motorola 68020, the VAC068A three-states its DSACKi* drivers on DRAM access to allow either the VAC068A or external logic to control their timing. When used with a synchronous local bus, the VAC068A allows for termination of DRAM accesses externally. On VME slave and VIC068A DMA cycles, the VAC068A asserts DSACKi* with minimum delay, following the assertion of local processor address strobe PAS*. This allows the high-reso[illegible] DSACKi*-to-DTACK* delay timer in the VIC068A to delay until data is [illegible] VIC068A registers $C7 and $A7.

19.8.4 VIC068A/VAC068A DMA Support

The VAC068A provides upper-address counters and control logic for both the VMEbus address bus A[31:8] and the local address bus LA[31:8] to extend the address range from 8 bits to 32 bits. This allows for crossing of 256-byte boundaries on either the local or VMEbus during block transfers. VIC068A/VAC068A DMA always has local memory as the source or destination of data. The data is transferred to either the VME interface or local I/O port in a pass-through mode. VMEbus block transfer DMA is a dual-address operation in which the source and destination are required to be on opposite sides of the VME interface. Data is transferred to/from some address in the local memory from/to some address accessed via the VMEbus. Memory-to-memory transfers, where both the source and sink address are local memory, are not supported.

In the VIC068A-supported module-based local DMA operation, DMA data transfers to/from the specified local memory address across an interface boundary to the local destination. This destination cannot be local memory. Thus, it allows the implementation of a VSB and/or daughterboard interface with DMA capability or fixed-address I/O operation as would be appropriate with a SCSI or Ethernet implementation.

There is also a dual-address path between the local bus and VMEbus that permits VMEbus accesses during periods of processor activity or during the interleave time between block transfers. The VAC068A also supports block transfers as a slave by latching incoming address information on the falling edge of AS*.

19.8.5 IORD* and IOWR*

The VAC068A generates IORD* and IOWR* as alternate I/O read and write signals. These outputs are synchronous to CPUCLK with a 0–2 clock delay from assertion IOSEL5–0* to the assertion of IOWR* or IORD* as well as the usual DSACKi* delay assertion. The IOSELi* outputs may be combinatorial with a minimum delay, or synchronous with a clock period delay to insure address set-up time. The IOSELi*s also have a programmable minimum deassertion time, since most peripheral controller chips cannot handle back-to-back accesses. These features significantly reduce the amount of support logic otherwise required to interface peripherals to the 68K.

19.8.6 I/O Recovery Timer

The VAC068A provides a programmable recovery time between individual I/O device accesses (IOSEL5–0* High). This timer function is provided when a value is written to the

Recovery Time bits in the DSACKi* Control registers. No IOSEL5–0* device access is allowed to start until the select signal for the previously accessed IOSEL5–0* device has been deasserted for the programmed number of clock cycles.

19.8.7 IACK Cycle Emulation for Non-680X0 Processors

The VAC068A provides two alternatives to 680X0 function code decoding for asserting the FCIACK* signal. This allows non-680X0-type processors to emulate the 680X0's interrupt acknowledge hardware protocol. If bit 31 of the PIO Function Register is set, accesses to IOSEL5* address space results in FCIACK* being asserted, while accesses to the IOSEL4* address space results in FPUCS* being asserted. If bit 30 of the PIO Direction register is set, then accesses to $FFFF FFxx results in FCIACK* being asserted, independent of the function codes.

19.8.8 Cache Inhibit Output

The VAC068A provides a CACHINH* (cache inhibit) signal typically connected to the corresponding input of the local processor. CACHINH* is asserted on any A16 access and is programmable in regions 1, 2, 3, 5, and in the A24 overlay region. For regions 1, 2, and 3, this is configured in the corresponding region attribute register. It is enabled for local I/O device access with bit 19 in the A24 Base Address register. For the A24 overlay, CACHINH* is enabled by setting bit 23 in the A24 Base Address register. CACHINH* is always asserted on redirected access of DRAM, and access to the top 256 bytes of DRAM if the mailbox interrupt is enabled.

CACHINH* can be asserted for all region 5 local I/O device selects (IOSEL5–0*), including VIC068A and VAC068A register accesses and region 6 (VMEbus A16) master accesses, and it may be asserted in the individual region attribute registers ($FFFD 09xx region 1, $FFFD 0Axx region 2, and $FFFD 0Bxx region 3), including the A24 address space overlay.

CACHINH* is deasserted in the remainder of DRAM and in region 4, the EPROM address space.

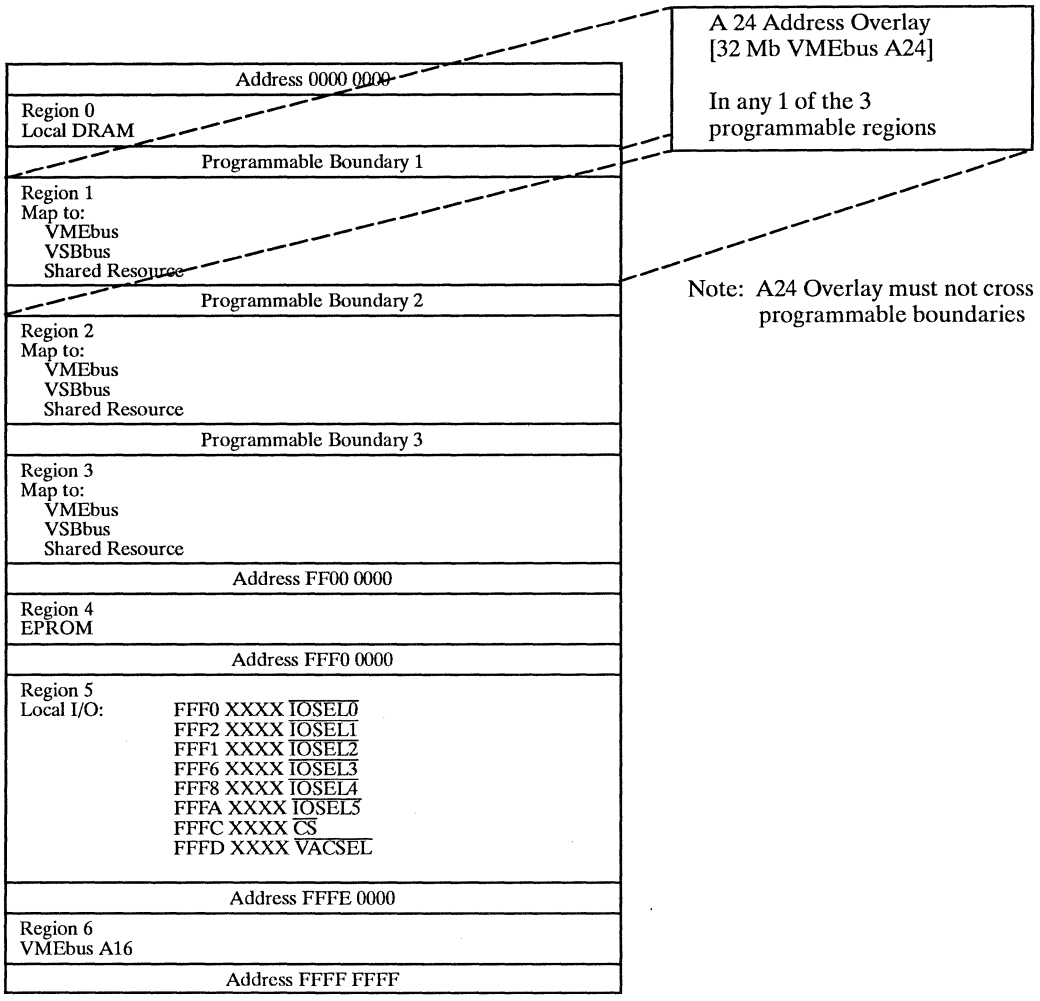


Figure 19–1. VAC068A Memory Map

20

VAC068A Operation

20.1 Resetting the VAC068A

There are two reset methods on the VAC068A. A global reset clears all registers and a soft reset (interrupt reset) masks all interrupt requests.

20.1.1 Global Reset

A global reset is initiated by either asserting the RESET* signal for 1K processor clock cycles or asserting RESET* in conjunction with WORD*. Both global reset types reset all VAC068A registers to their default values.

To execute the first type of global reset, the RESET* signal must be held Low for more than 1K CPUCLK cycles. If RESET* is held for less than 1K CPUCLK cycles, a soft reset occurs.

20.1.1.1 Power-On Reset

The other option to execute a reset of VAC068A registers is to assert RESET* for at least 5 CPUCLK cycles and then assert WORD* in conjunction with RESET* for at least 10 CPUCLK cycles. This also resets all VAC068A internal registers.

Following a global reset, memory map and slave address decodes (other than EPROM and Local I/O address space) are disabled until a local processor write occurs to the VAC068A ID register.

20.1.2 Soft Reset

A soft reset is initiated by asserting RESET* and holding it for less than 1K CPUCLK cycles. This function masks all interrupt requests but does not affect the memory map configurations.

20.1.3 RESET* Termination

Upon the termination of the RESET* operation, EPROMCS* is asserted for all processor read cycles, independent of the processor address. EPROMCS* DSACKi* timings default

to the maximum time and 32-bit data size. To modify the EPROM data size, hold either ID bus ID[9] Low upon power-up during RESET* for 16-bit data size, or hold IDbus ID[8] Low for 8-bit data size. Upon completion of the FORCE EPROM mode, the user must configure DSACKs in the DSACKi* Control register per the data size on the module. The Force EPROM mode is exited upon an access to the EPROM address space ($FF00 0000 to $FFEF FFFF). This dictates that a jump to EPROM should be one of the first instructions in the boot sequence. The default EPROM timing and data sizes are disabled by the first write to the EPROM DSACKi* Control register.

20.2 System Initialization

After reset and before VAC068A registers are initialized, only EPROM, the VIC068A, and the VAC068A may be accessed. As stated previously, processor reads are forced to EPROM and use the slowest DSACKi* timing. Once the EPROM address space is accessed on the local address bus, the force EPROM mode is exited. The user is expected to set DSACKi* timings for reliable module operation. The following sequence of events is anticipated after power-up reset:

1. The VAC068A samples ID[8,9] on the rising edge of RESET* to get the default EPROM data path width.
2. CPU reads the reset vector table entry (address $0000) and loads it into the interrupt stack pointer.
3. VAC068A asserts EPROMCS*, then DSACKi* is asserted after 7 CPUCLK cycles.
4. CPU captures $FF00 0008 (32-bit mode) from the data bus.
5. CPU reads address $0000 0004. The VAC068A asserts EPROMCS, then DSACKi* cycle.
6. CPU captures data from the data bus.
7. CPU reads $FF00 0008.
8. The VAC068A asserts EPROMCS*, then DSACKi* after 7 clocks.
9. CPU reads, writes to a VAC068A register other than EPROM DSACKi* Control to verify proper operation.
10. Write remainder of VAC068A registers to fully define memory map.
11. Write VAC068A ID register $FFFD 29xx to enable select and decode outputs.
12. Read/write sufficient addresses to verify map decoding.

20.3 Configuring the Local Memory Map

Sections 20.3 and 20.4 describe a sample memory map definition. The specific registers and configuration values are given.

The example assumes the module contains 4 Mbytes of DRAM, 16 Mbytes of VSB space in region 1, and 256 Kbytes of SRAM in the shared resource area. All are 32 bits in width.

20.3.1 DRAM Size

The module DRAM memory area (region 0) starts at address $0000 0000 and ends at the address configured in the DRAM Upper-Limit Mask register ($FFFD 05xx). For the 4 Mbytes required by this example, this register is loaded with $003F. Unlike regions 1 and 2, region 0 is required to be the DRAM memory area.

To support a local mailbox area, the DRAM area must contain at least 256 bytes of address space.

20.3.2 VSB Space

VSB Space is configured in region 1. This requires setting bits 27 and 26 of the Region 1 Attribute register ($FFFD 09XX) to 10 to assert VSBSEL* when the local address falls into the address range for region 1. The lower address limit for region 1 is specified as the first address beyond the DRAM area configured in the DRAM Upper-Limit Mask register. The upper address limit for this region is configured in the Boundary 2 Address register ($FFFD 06xx) with $013F to allocate a 16-Mbyte space directly above the DRAM area.

20.3.3 VMEbus A32, D32 Access

The VMEbus accessible address space is configured in region 2. This requires setting bits 27 and 26 of the Region 2 Attribute register ($FFFD 0Axx) to 11 to assert MWB* when the local address falls into the address range for region 2. The lower address limit for region 2 is specified as the first address beyond the VSB space configured in the Boundary 2 Address register. The upper address limit for this region is configured in the Boundary 3 Address register ($FFFD 07xx). This register is loaded with $FEFB to allocate all other address space—except 256 Kbytes for a shared resource area and the top 16 Mbytes for local resources and EPROM—to the VMEbus.

20.3.4 Shared Resource Area

The shared resource area is configured in region 3. This requires setting bits 27 and 26 of the Region 3 Attribute register ($FFFD 0Bxx) to 01 to assert SHRCS* when the local address falls into the address range for region 3. The lower address limit for region 3 is specified as the first address beyond the VMEbus area configured in Boundary Area 3 Address register. The upper address limit for region 3 is fixed at $FEFF FFFF.

20.3.5 EPROM Space

The VAC068A supports a fixed EPROM space from $FF00 0000 to $FFEF FFFF. These address limits may not be modified. When any address in this range is accessed, the VAC068A asserts EPROMCS*.

20.4 Configuring the VMEbus Address Map

The module address map for slave accesses from other VMEbus masters is configurable in multiple areas. For this example, the 4 Mbytes of DRAM and the 256 Kbytes of SRAM in the shared resource area are both mapped to other addresses on the VMEbus through the two slave select areas.

20.4.1 SLSEL0* Access

The SLSEL0 address range is used to define VMEbus access to the 4 Mbytes of module DRAM. SLSEL0* accesses can only be mapped to region 0 (DRAM). This memory is mapped into the $0880 0000 to $08BF FFFF address range using the SLSEL0* Base Address and Address Mask registers. The base address register is used to specify the required logic level of the upper address bits to be compared, while the mask register is used to specify which bits to compare. To map the specified space, the SLSEL0* Base Address register ($FFFD 03xx) is loaded with $0880 and the SLSEL0* Address Mask register ($FFFD 02xx) is loaded with $FFC0. This includes the A[31:22] address bus signals in the address comparison.

Because the SLSEL0* space is defined here to be A32 space, it is necessary to also configure the VIC068A Slave Select 0 Control register 0 (SS0CR0) for A32 address modifier codes and D32 address space.

By enabling redirection for the SLSEL0* region, it is possible for the local processor to access the 4 Mbytes of DRAM in both the region 0 and SLSEL0* areas of the address map.

Redirection is enabled for the SLSEL0* area by setting bit 19 in the Decode Control register ($FFFD 14xx). With this bit set, local processor accesses to either memory area will assert DRAMCS* and not assert SLSEL0*.

A VMEbus slave access to the SLSEL0* memory area (with proper AM codes) will assert both SLSEL0* (to the VIC068A) and DRAMCS*.

20.4.2 SLSEL1* Access

The SLSEL1* area can be configured to map VMEbus slave accesses into any of four areas on the module: EPROM, DRAM, VSB, or Shared Resource. The specific area is configured in the Decode Control register ($FFFD 14xx). For this example, the Shared Resource area is mapped into VMEbus space by setting bits 29 and 28 of the Decode Control register to 10.

The 256-Kbyte SRAM is mapped into the $C0 0000 to $CF FFFF address range using the SLSEL1* Base Address and Address Mask registers. The base address register is used to specify the required logic level of the upper address bits to be compared while the mask register is used to specify which bits to compare. To map a 256-Kbyte address space for SLSEL1 in the SHRCS region, the SLSEL1* Base Address register ($FFFD 01xx) is loaded with $xxC0 with the SLSEL1* Address Mask register ($FFFD 00xx) is loaded with $00FC. By clearing the upper 8 bits of the SLSEL1* Address Mask register, the A[31:24] address bus signals are excluded from the address comparison. This configures the shared resource to appear in A24/A32 address space by setting bit 27 of the Decode Control register.

Because the SLSEL1* space is defined here to be A24/A32 space, it is necessary to also configure the VIC068A Slave Select 1 Control register 0 (SS1CR0) for A24 or A32 address modifier codes and D32 address space.

When redirection for SLSEL1 is enabled by setting bit 20 of the Decode Control register, and an address in the SLSEL1 address mask range is present on the VMEbus, SHRCS* is asserted. Thus, a valid VMEbus slave access to SRAM occurs. If the SLSEL1* address is present on the local bus, no SLSEL1* assertion occurs.

If bit 20 in the Decode Control register is not set, no chip selects are asserted when the VMEbus address is within the SLSEL1 mask address range.

20.4.3 ICFSEL* Access

Decoding of VMEbus Interprocessor Communications Facility accesses are accomplished through the ICFSEL* Address register ($FFFD 04xx). The two bytes of this register are

both compared with A[15:8]. An exact match of these address bus signals with either byte causes the VAC068A to assert ICFSEL* to the VIC068A. This allows both global (or group) and module specific select addresses to be configured. The low address bits (A[7:0]) are used to select which specific module switches and registers are to be accessed.

Present convention is to use the upper byte of the ICFSEL* Address register to specify the global select address and the lower byte for the module specific address. Loading this register with $F00F would then specify that all A16 accesses to $F0xx are addressed globally, while a similar access to $0Fxx would only be responded to by a module.

These same ICF registers and switches are accessed by the local processor in the $FFFC xx5F through $FFFC xx7F range.

20.4.4 VME A24 Master Cycle

The A24 space for VMEbus master accesses may be mapped to any 32-Mbyte section of the programmable regions. This is configured in the A24 Base Address register bits [31:25]. The value placed in these bits is compared to the local address bus LA[31:25]. Because any value may be placed in these register bits, it is possible for the A24 space to overlay other regions of the local address map. Values should be greater than $02 and less than $FE.

For this example, the A24 Base Address register ($FFFD 08xx) is loaded with $F000. This assigns the local address range of $F000 0000 to $F1FF FFFF to A24 space. This exists in the local address space presently assigned as VMEbus space in region 2. Any local processor access to this space causes the VAC068A to drive ASIZ0/1 with 11 to force the VIC068A to assert the proper AM codes for A24 addressing.

The data bus size for these A24 master accesses may also be configured in the A24 Base Address register for either D16 or D32 operation.

20.4.5 VME A16 Master Cycle

VMEbus A16 master cycles are fixed at the top 128 Kbytes section of the address map ($FFFE 0000 to $FFFF FFFF). Other bits in the A24 Base Address register are used to configure the data bus width (D32 or D16) and cache inhibit control.

20.4.6 Decode Control Register

To completely specify the VAC068A behavior as a VME slave device, the Decode Control register ($FFFD 14xx) must be configured. Bits [29:28] must be set to 10 to select SHRCS*

for assertion in response to a LAEN when SLSEL1* is asserted and FC2/1 specifies a VME slave access. Bit 27 must be set for A24 (or A32) operation. Bits [25:23] must be set to 1 in the general case. If any of these bits is a 0, the slave select output is independent of VAS*. Otherwise, the output is only asserted when VAS* is asserted. Since the qualification of the slave selects is done midway through the address decode process, timing verification must demonstrate that it is done early enough to avoid glitches.

Bits [20:19] are normally set to 1. When this is the case, local CPU accesses to addresses within the SLSEL0* or SLSEL1* range, respectively, result in the device chip select, DRAMCS* for SLSEL0* or SHRCS* for SLSEL1*, being asserted instead of MWB*. Otherwise, MWB* is asserted and a VME access occurs. When these redirection bits are set to 0, the VIC068A should be programmed to assert BERR* and LBERR* when it sees a valid slave select when it is bus master. This allows software to determine its own slave select address. If the redirection bits are not set to 1, attempted access to a module's slave address results in a VME bus timeout. Bit 22 provides a means of qualifying decodes of the local address map with PAS*.

20.5 VME Master Access

For VAC068A-controlled VMEbus accesses, the local address must be set up 10 ns before PAS* is asserted. After the VAC068A decodes an address that maps to VMEbus, the VAC068A asserts MWB* and sets ASIZ0/1 and WORD* High or Low according to the appropriate region attribute register. The VIC068A then obtains bus mastership and asserts ABEN*. The VAC068A uses ABEN* to enable its VME address drivers. The access completes when the VIC068A asserts DSACKi* or LBERR*. When ABEN* is deasserted, the VMEbus address drivers must three-state before the VIC068A deasserts both BBSY* and AS*.

When write-posting is enabled, the VIC068A asserts LADO to freeze the address outputs. Accordingly, the address path through the VAC068A is transparent when LADO is Low and latched when LADO is High.

20.6 VME Slave Operation

The VAC068A continuously monitors the VMEbus address bus for any of the SLSEL0*, SLSEL1*, and ICFSEL* addresses. When it detects such an address, qualified by AS* if so

enabled, it asserts the proper select output. The LADI input is used to insure that the address presented to the local bus remains stable throughout the slave access. This is accomplished by latching the local address outputs when LADO is High. The VIC068A may respond to an asserted slave select output with a local bus request and subsequent slave transfer. The VIC068A qualifies each slave select with the address modifier field and data strobe.

20.6.1 Slave Transfer Sequence

The following sequence of events occur on a slave transfer:

1. The VAC068A asserts SLSEL0* (or SLSEL1* or ICFSEL*).
2. The VIC068A qualifies SLSEL0* with address modifiers and data strobe(s) and asserts VICLBR*.
3. Module logic arbitrates for the local bus and asserts LBG* to the VIC068A. The VIC068A waits for local-cycle-end plus 3 CLK64M clocks, then asserts LAEN, sets FC2/1 to 10 to distinguish slave transfer from DMA or refresh, drives LA[7:0] appropriately, and enables the data path (DDIR, SWDEN*, LWDENIN*, UWDENIN*, DENO*).
4. The VAC068A uses LAEN to enable its local address drivers and FC2/1 to select the VMEbus as the address source. Approximately 1 CLK64M clock from LAEN assertion, the VIC068A asserts PAS*.
5. Upon PAS* assertion, the VAC068A asserts DSACK0/1* and asserts the appropriate device select output (DRAMCS*, EPROMCS*, VSBSEL*, or SHRCS*).
6. The local memory reads/writes at the address on the local bus.
7. The VIC068A times out the DSACKi* to DTACK* delay, then asserts the LEDO output to capture the data in the bus interface latches, deasserts the local strobes, and asserts DTACK*. At some time previous to DTACK* assertion, the VIC068A asserts LADI to freeze the local address.
8. The VAC068A deasserts DSACK1/0* when PAS* is deasserted.
9. The VAC068A three-states its address drivers when LAEN is deasserted.

20.7 VME Master Block Transfer

The following sequence occurs on a master block transfer:

1. The local CPU initializes the VIC068A for a DMA block transfer, then asserts address and PAS* that maps to the VMEbus.
2. The VAC068A decodes the address and asserts MWB*, along with WORD* and ASIZ1/0 according to the appropriate Region Attribute registers.
3. The VIC068A detects MWB* asserted, requests the VMEbus, and asserts BLT*.
4. The VAC068A detects BLT* asserted and loads ID[31..8] into its DMA address counter for driving the local address and loads LA[31..8] into the BLT address counter for driving A[31..8].
5. The VIC068A receives VMEbus mastership and asserts LBR*.
6. Module logic arbitrates for the local bus and asserts LBG* to the VIC068A.
7. The VIC068A waits for local-cycle-end plus 3 CK64M clocks, then asserts LAEN, sets FC2/1 to 01 to distinguish DMA from a slave transfer or refresh, drives LA[7:0] appropriately, and enables the data path (DDIR, SWDEN*, LWDENIN*, UWDENIN*, DENO*).
8. The VAC068A uses LAEN to enable its local address drivers with the DMA address and uses FC2/1 to select the DMA counters as the address source.
9. The VAC068A asserts LDMACK* when LAEN is asserted and the function codes reach the DMA (VME or local) state.
10. Approximately 1 CLK64M clock from LAEN assertion, the VIC068A asserts PAS* and DS*.
11. Upon PAS* assertion, the VAC068A asserts DSACKi* and DRAMCS*.
12. The local memory reads/writes at the addresses on the local bus as they are incremented and strobed by the VIC068A and VAC068A.
13. After burst counter expiration in the VIC068A, PAS* is deasserted. If a 256-byte boundary crossing occurs, BLT* or LADO is pulsed to increment the appropriate counter. Next, LBR* and LAEN are deasserted. If BLT* remains asserted, the address counters are preserved. If LADO toggles twice while LBR* is asserted, the local address counter is incremented. If BLT* toggles twice while LBR* is asserted, the local address counter is incremented. When BLT* or LAEN deassert, the local address drivers are three-stated. The DSACKi* drivers three-state soon thereafter as permitted by their rescinding circuit. When both BLT* and LBR* are deasserted, the DMA block transfer is over.

20.8 Local (Non-VME) DMA

This functions the same as a VMEbus master block transfer except that

- the VMEbus is neither requested nor driven
- ASIZ1 and ASIZ0 are three-stated

In this mode, the VIC068A ASIZ1/0 signals serve as bus error and data acknowledge, respectively. Refer to the VIC068A section of this user's guide for more information on this operation.

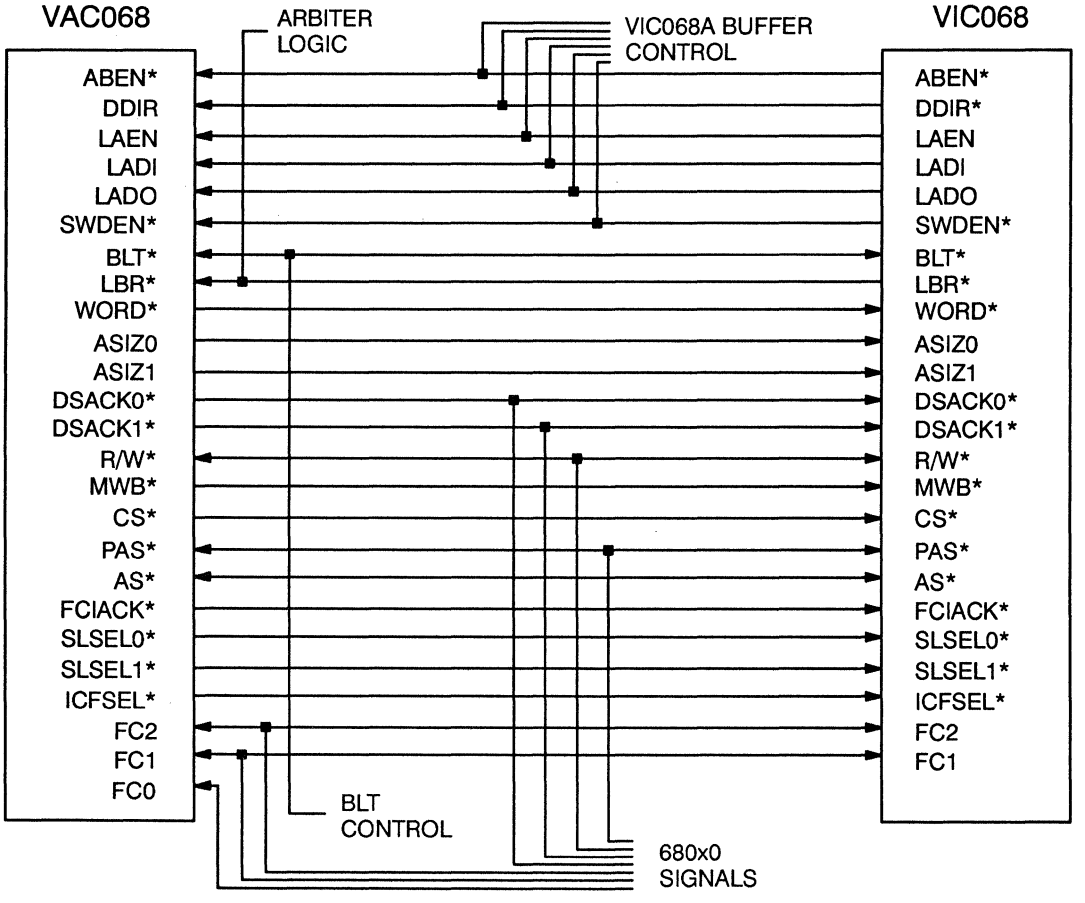


Figure 20–1. VIC068A/VAC068A Interconnect Diagram

21

VAC068A Register Map and Descriptions

Base address for the VAC068A register set is $FFFD 00xx. Register size is up to 16 bits wide and accesses are acknowledged by using DSACK1*. The 16-bit registers are NOT byte accessible. For single-byte registers, the unused bits are read as 1's. Register values are listed in *Table 21–1.*

Table 21–1. Register Values

Local Address	Register List	Size
FFFD 00xx	SLSEL1* Address Mask Register	16 bits
FFFD 01xx	SLSEL1* Base Address Register	16 bits
FFFD 02xx	SLSEL0* Base Address Register	16 bits
FFFD 03xx	SLSEL0* Address Mask Register	16 bits
FFFD 04xx	ICFSEL* Base Address Register	16 bits
FFFD 05xx	DRAM Upper-Limit Mask Register	16 bits
FFFD 06xx	Boundary 2 Address Register	16 bits
FFFD 07xx	Boundary 3 Address Register	16 bits
FFFD 08xx	A24 Base Address Register	13 bits
FFFD 09xx	Region 1 Attribute Register	6 bits
FFFD 0Axx	Region 2 Attribute Register	6 bits
FFFD 0Bxx	Region 3 Attribute Register	6 bits
FFFD 0Cxx	IOSEL4* DSACK Control Register	16 bits
FFFD 0Dxx	IOSEL5* DSACK Control Register	16 bits
FFFD 0Exx	SHRCS* DSACK Control Register	16 bits
FFFD 0Fxx	EPROMCS* DSACK Control Register	16 bits
FFFD 10xx	IOSEL0* DSACK Control Register	16 bits
FFFD 11xx	IOSEL1* DSACK Control Register	16 bits
FFFD 12xx	IOSEL2* DSACK Control Register	16 bits
FFFD 13xx	IOSEL3* DSACK Control Register	16 bits
FFFD 14xx	Decode Control Register	16 bits
FFFD 15xx	Interrupt Status Register	8 bits

Table 21–1. Register Values (continued)

Local Address	Register List	Size
FFFD 16xx	Interrupt Control Register	16 bits
FFFD 17xx	Device Location Register	6 bits
FFFD 18xx	PIO Data Out Register	14 bits
FFFD 19xx	PIO Pin Register	14 bits
FFFD 1Axx	PIO Direction Register	15 bits
FFFD 1Bxx	PIO Function Register	16 bits
FFFD 1Cxx	CPU Clock Divisor Register	8 bits
FFFD 1Dxx	UART Channel A Mode Register	12 bits
FFFD 1Exx	UART Channel A Transmit Data Register	8 bits
FFFD 1Fxx	UART Channel B Mode Register	12 bits
FFFD 20xx	UART Channel A Receiver FIFO	11 bits
FFFD 21xx	UART Channel B Receiver FIFO	11 bits
FFFD 22xx	UART Channel B Transmit Register	8 bits
FFFD 23xx	UART Channel A Interrupt Mask Register	6 bits
FFFD 24xx	UART Channel B Interrupt Mask Register	6 bits
FFFD 25xx	UART Channel A Interrupt Status Register	8 bits
FFFD 26xx	UART Channel B Interrupt Status Register	8 bits
FFFD 27xx	Timer Data Register	16 bits
FFFD 28xx	Timer Control Register	8 bits
FFFD 29xx	VAC068A ID Register	16 bits

The base address location for the VAC068A register set is $FFFD 00xx. All VAC068A registers are cleared during a global reset and remain intact during a soft reset. Only interrupts are masked during a soft reset. Unused or reserved bits may read as a 0 or a 1. The VAC068A ID register remains intact through all resets.

The VAC068A registers are accessed from the local address/data signals and acknowledged as a 16-bit access by DSACK1* assertion. They may only be accessed as a 16-bit word. CACHINH* is asserted during accesses to the VAC068A registers. The VAC068A Identification register must be written after reset to enable VAC068A operation.

SLSEL1* Address Mask Register

Local Address $FFFD 00xx

Bits 31:16 A set bit in any of the positions enables a comparison of the correspondingly numbered local and VMEbus address bits for the purpose of asserting SLSEL1*.

SLSEL1* Base Address Register

Local Address $FFFD 01xx

Bits 31:16 The contents of this register are compared under a bitwise mask compare to both the local and VMEbus address bits for the purpose of asserting SLSEL1*.

SLSEL0* Address Mask Register

Local Address $FFFD 02xx

Bits 31:16 A set bit in any of the positions enables a comparison of the correspondingly numbered local and VMEbus address bits for the purpose of asserting SLSEL0*.

SLSEL0* Base Address Register

Local Address $FFFD 03xx

Bits 31:16 The contents of this register are compared under a bitwise mask compare to both the local and VMEbus address bits for the purpose of asserting SLSEL0*.

ICFSEL* Base Address Register

Local Address $FFFD 04xx

Bits 31:24 The upper half of this register is compared to VMEbus address [15:8] for the purpose of asserting ICFSEL*. When a match occurs between register bits [31:24] and the VMEbus address signals A[15:8], ICFSEL* is asserted.

Bits 23:16 The lower half of this register is compared to VMEbus address [15:8] for the purpose of asserting ICFSEL*. When a match occurs between register bits [23:16] and VME address signals A[15:8], ICFSEL* is asserted.

When either bits A[31:24] or A[23:16] match the VMEbus address bits A[15:8], ICFSEL* is asserted. These two different 8-bit compares are used for asserting ICFSEL* to the VIC068A for differentiating between module-based or global functions.

DRAM Upper-Limit Mask Register

Local Address $FFFD 05xx

This register contains an address mask used to specify the upper address limit of the DRAM memory area located in region 0. The local address bits LA[31:16] are logically ANDed with the NOT of their respective bits in this register and, if all AND outputs are Low, DRAMCS* is asserted. A simpler way to view this is if any 1 is present on the address bus where a corresponding 0 exists in the DRAMCS* Upper-Limit Mask register, DRAMCS* is not asserted.

The logic function used for this compare is a masking operation rather than a full magnitude compare. It is advised that only exact binary multiples be specified for the DRAM memory size (i.e., 1 Mbyte, 2 Mbytes, 4 Mbytes, etc.) to avoid having holes in the local address map. While it is not necessary to fully populate a defined area, DRAMCS* will be asserted if an access is attempted to the area, even if there is no memory present.

This register is cleared up on power-up and on global resets. Following exit from the force EPROM mode, accesses to $0000 00xx will assert DRAMCS* even if no value has been loaded into the mask register.

Boundary 2 Address Register

Local Address $FFFD 06xx

This register contains the lower address limit for region 2 and the upper address limit for region 1. Its contents are compared to local address bits LA[31:16]. If the address is less than the value of this register, and neither DRAM nor A24 space (configured in the A24 Base Address register) access occurs, the access is valid for this region.

Boundary 3 Address Register

Local Address $FFFD 070xx

This register contains the upper address limit for region 2 and the lower limit for region 3. The upper address limit for region 3 is the EPROM address space ($FF00 0000). Its contents are compared to the local address bits LA[31:16]. If the address is less than the value in this register and neither DRAM or A24 space is being accessed, the access is valid for this region. If the address is greater than or equal to the value in this register yet less than EPROM space, a region 3 access occurs.

A24 Base Address Register

Local Address $FFFD 08xx

Bits 31:25 — These are compared to local address bits LA[31:25] for the purpose of overlaying an A24 address space in any one of the three regions described by their respective boundaries. Local access to this address space forces a master access to VMEbus A24 space. The area specified must be above the DRAM upper limit. *Note:* Valid values for bits [31:25] are greater than $02 and less than $FE

Bit 24 — This bit is decoded along with bit 20 to determine the A24 data path size for the entire 32-Mbyte range. If bit 24 is cleared, a D16 data path is selected. If bit 24 is set, a D32 data path is selected. This bit is only interpreted if bit 20 is cleared.

Bit 23 — When set, this bit selects A24 CACHINH* (cache inhibit). When cleared, no CACHINH* for A24 address space accesses.

Bit 22 — When set, this bit enables bit 21 to determine the data path size of the A16 address space (region 6). When cleared, this bit enables the local address bit LA[16] to decode data path size. If LA[16] is High, a D16 data path is enabled (WORD* asserted). If LA16 is Low, a D32 data path is enabled (WORD* deasserted).

Bit 21 — When set, this bit along with bit 22 causes region 6 (VMEbus A16 address space) to have a D32 data path (WORD* deasserted). When cleared, this bit causes region 6 to have a D16 data path (WORD asserted).

Bit 20 — When set, this bit enables the local address bit LA[24] to determine the data path size for A24 master accesses. When LA[24] is High, the data path size is D16. When LA[24] is Low, the data path is D32. When bit 20 is cleared, bit 24 decodes the data path size for the entire A24 address space.

Bit 19 — When set, this bit enables CACHINH* on accesses to VIC068A register accesses, VAC068A register accesses, and any of the six IOSEL0–5 local I/O address areas. When cleared, CACHINH* is not asserted on access to these address spaces.

Region 1–3 Attribute Registers

Local Address — $FFFD 09xx Region 1 Attribute register.

Local Address $FFFD 0Axx Region 2 Attribute register.

Local Address $FFFD 0Bxx Region 3 Attribute register.

Bit 31 When set, this bit enables WORD* to be asserted upon access.

Bit 30 When set, this bit enables ASIZ1 to be driven Low upon access.

Bit 29 When set, this bit enables ASIZ0 to be driven Low upon access.

Bit 28 When set, this bit enables CACHINH* to be asserted upon access.

Bits 27:26 Follow this table:

Bit 27	*Bit 26*	*Mode*
0	0	Inactive
0	1	Shared resources chip select
1	0	VSB resource chip select
1	1	MWB select (VMEbus request)

DSACKi* Control Registers

Local Address $FFFD 0Cxx – IOSEL4* DSACKi* Control register (I/O Select Address = $FFF8 0000 to $FFF9 FFFF).

Local Address $FFFD 0Dxx – IOSEL5* DSACKi* Control register (I/O Select Address = $FFFA 0000 to $FFFB FFFF).

Local Address $FFFD 0Exx – SHRCS* DSACKi* Control register (I/O Select Address is programmable).

Local Address $FFFD 0Fxx – EPROMCS* DSACKi* Control register (I/O Select Address = $FF00 0000 to $FFEF FFFF).

Local Address $FFFD 10xx – IOSEL0* DSACKi* Control register (I/O Select Address = $FFF0 0000 to $FFF1 FFFF).

Local Address $FFFD 11xx – IOSEL1* DSACKi* Control register (I/O Select Address = $FFF2 0000 to $FFF3 FFFF).

Local Address $FFFD 12xx – IOSEL2* DSACKi* Control register (I/O Select Address = $FFF4 0000 to $FFF5 FFFF).

Local Address $FFFD 13xx – IOSEL3* DSACKi* Control register (I/O Select Address = $FFF6 0000 to $FFF7 FFFF).

Bits 31:29 These bits determine the delay from PAS* assertion to assertion of DSACKi* in CPUCLK cycles per the following table:

000 = 1 cycle
001 = 2 cycles
.
.
111 = 8 cycles

Bit 28 When set, this bit enables DSACK1* on slave accesses. When cleared, DSACK1* is inactive.

Bit 27 When set, this bit enables DSACK0* on slave accesses. When cleared, DSACK0* is inactive.

Bits 26:24 These bits determine the recovery time for IOSELi* in integer multiples of the CPUCLK as follows:

000 = 1 CPUCLK cycle
001 = 2 CPUCLK cycles
.
.
111 = 8 CPUCLK cycles

The VAC068A recovery time (time between assertions of device select outputs) is controlled by two separate timers; one for even-numbered IOSEL5–0* device select outputs and one for odd-numbered IOSEL5–0* device select outputs. Because of the shared usage of these counters, the user must insure that an IOSEL5–0* access to a device that uses the same counter (odd or even IOSEL5–0* address) is not allowed to start (not issued by the local processor) until the previous access has been deasserted for the required number of CPUCLK cycles. These counters operate only on IOSEL5–0* accesses. It is assumed that accesses to EPROMCS* or SHRCS* do not require a recovery time and should set the value of these bits of their DSACKi* Control register to 000.

Bits 23:22 — These bits determine the assertion delay for IORD* from PAS* in 1/2 CPUCLK cycles (i.e., 0.5, 1, 1.5, 2).

Bits 21:20 — These bits determine the assertion delay for IOWR* from PAS* in 1/2 CPUCLK cycles (i.e., 0.5, 1, 1.5, 2).

Bits 19:18 — These bits determine the assertion delay for IOSEL5–0* from PAS* in 1/2 CPUCLK cycles (i.e., 0.5, 1, 1.5, 2).

Bit 17 — When cleared, the IORD* signal is deasserted when PAS* is deasserted. When set, IORD* is deasserted when the time specified in the DSACKi* Assertion Delay (bits 31:29) has elapsed. This is used to provide additional hold time for the peripheral device.

Bit 16 — When cleared, the IOWR* signal is deasserted when PAS* is deasserted. When set, IOWR* is deasserted when the time specified in the DSACKi* Assertion Delay (bits 31:29) has elapsed. This used to provide additional hold time for the peripheral device.

If bits 18 and 19 are cleared, IOSEL5–0* is always deasserted when PAS* is deasserted. To use early cycle end control on read cycles, the data must be latched until captured by the local processor, independent of the deassertion of the control signals. This latching function is provided in the VAC068A for devices located on the ID bus signals ID[15:8].

No odd-numbered I/O device access is allowed to start until the select signal for the previously accessed odd-numbered I/O device has been deasserted for the programmed number of clock cycles. The same is true for even-numbered device selects. Recovery time is metered only for IOSEL5–0*. All other devices are assumed to not need a recovery time. Accordingly, the recovery time field for SHRCS* and EPROMCS* should be set to 0.

Decode Control Register

Local Address — $FFFD 14xx

Bit 31 — When set, assert DSACKi* on slave accesses during VIC068A local bus cycles (except DRAM Refresh). When cleared, three-state DSACKi* on slave accesses during VIC068A local bus cycles.

Bit 30 — When set, qualify DRAMCS* assertion with PAS* assertion. When cleared, assert DRAMCS* on address space match.

Bits 29:28 — These bits specify what local resource select is asserted on SLSEL1* assertion when redirection of SLSEL1* is enabled (bit 20).

00 = EPROMCS*
01 = VSBSEL*
10 = SHRCS*
11 = DRAMCS*

Bit 27 — When set, compare VMEbus A[31:16] to SLSEL1* base address register [31:16]. When cleared, no compare. This is used to allocate SLSEL1* in an A32/A24 space.

Bit 26 — When set, compare VMEbus A[15:8] to SLSEL1* base address register [31:24]. When cleared, no compare. This is used to allocate SLSEL1* in an A16 space.

Bit 25 — When set, qualify SLSEL0* decode with VMEbus AS*. When cleared, no qualification.

Bit 24 — When set, qualify SLSEL1* decode with VMEbus AS*. When cleared, no qualification.

Bit 23 — When set, qualify ICFSEL* decode with VMEbus AS*. When cleared, no qualification.

Bit 22 — When sets, qualify boundary decodes (except DRAM) with PAS* or DS*. When cleared, no qualification.

Bit 21 — When set, acknowledge DRAM access as 32-bit port (both DSACK0/1 asserted). When cleared, three-state DSACK0/1*s on DRAMCS*.

Bit 20 — When set, redirect SLSEL1* area accesses on local bus to local resource specified in bits 29:28. When cleared, no redirect for SLSEL1*.

Bit 19 — When set, redirect SLSEL0* area accesses on local bus to DRAM. When disabled, no redirect.

Bits 18:17 — Assertion delay for DSACKi* upon access to DRAM (assertion of DRAMCS*) in CPU clock cycles per the following table:

00 = 0 CPUCLK cycles
01 = 1 CPUCLK cycles
10 = 2 CPUCLK cycles
11 = 3 CPUCLK cycles

Bit 16 When set, FPUCS* is asserted on assertion of PAS*. When cleared, FPUCS* is asserted on CPUCLK.

Interrupt Status Register

Local Address $FFFD 15xx

Bit 31 When set, this bit indicates that a PIO9 interrupt is pending.

Bit 30 When set, this bit indicates that a PIO8 interrupt is pending.

Bit 29 When set, this bit indicates that a PIO7 interrupt is pending.

Bit 28 When set, this bit indicates that a PIO4 interrupt is pending.

Bit 27 When set, this bit indicates that a mailbox interrupt is pending.

Bit 26 When set, this bit indicates that a timer interrupt is pending.

Bit 25 When set, this bit indicates that a UART A interrupt is pending.

Bit 24 When set, this bit indicates that a UART B interrupt is pending.

This register is read-only. Bits of this register are cleared by accessing the interrupt control register and clearing the control bits for that particular interrupt.

Interrupt Control Register

Local Address $FFFD 16xx

Bits 31:30 These bits specify the mapping of PIO9 interrupt to one of the three signals as detailed in the following table.

Bits 29:28 These bits specify the mapping of PIO8 interrupt to one of the three signals as detailed in the following table.

Bits 27:26 These bits specify the mapping of PIO7 interrupt to one of the three signals as detailed in the following table.

Bits 25:24 These bits specify the mapping of PIO4 interrupt to one of the three signals as detailed in the following table.

Bits 23:22 These bits specify the mapping of the mailbox interrupt to one of the three signals as detailed in the following table.

Bits 21:20 These bits specify the mapping of the UART A interrupt to one of the three signals as detailed in the following table.

Bits 19:18 These bits specify the mapping of the UART B interrupt to one of the three signals as detailed in the following table.

Bits 17:16 These bits specify the mapping of the timer interrupt to one of the three signals as detailed in the following table.

Odd bit	*Even bit*	*Function*
0	0	Disabled
0	1	Enable to PIO7
1	0	Enable to PIO10
1	1	Enable to PIO11

Note that each interrupt service routine should clear its interrupt's map bits momentarily in order to clear the interrupt request output. Each interrupt is active Low and edge-triggered except UART A and B, which are event triggered and hold until cleared by clearing the interrupt in the Int Mask register. An interrupt request output is asserted only if a falling edge on an interrupt request input occurs while its map bits are non-zero. PIO9 must be held Low for at least 2.8 ms in order to generate an interrupt request.

Device Location Register

Local Address $FFFD 17xx

Bit 21 When set, IOSEL5* active on the ID bus.

Bit 20 When set, IOSEL4* active on the ID bus.

Bit 19 When set, IOSEL3* active on the ID bus.

Bit 18 When set, IOSEL2* active on the ID bus.

Bit 17 When set, IOSEL1* active on the ID bus.

Bit 16 When set, IOSEL0* active on the ID bus.

This register specifies mapping of the input/output select device on the ID bus. If any bit is set, it indicates that the corresponding device is located on ID[15:8]. This allows the VAC068A to control the internal Buffer and Latch on the ID bus when these devices are accessed. SWDEN* swaps the data from/to ID[31:24] to ID[15:8] and DDIR* controls the data direction.

PIO Data Out Register

Local Address $FFFD 18xx

Bit 29 PIO13 or LD[29] signal output value.

Bit 28 PIO12 or LD[28] signal output value.

Bit 27 PIO11 or LD[27] signal output value.

Bit 26 PIO10 or LD[26] signal output value.

Bit 25 PIO9 or LD[25] signal output value.

Bit 24 PIO8 or LD[24] signal output value.

Bit 23 PIO7 or LD[23] signal output value.

Bit 22 PIO6 or LD[22] signal output value.

Bit 21 PIO5 or LD[21] signal output value.

Bit 20 PIO4 or LD[20] signal output value.

Bit 19 PIO3 or LD[19] signal output value.

Bit 18 PIO2 or LD[18] signal output value.

Bit 17 PIO1 or LD[17] signal output value.

Bit 16 PIO0 or LD[16] signal output value.

This register is used for writing to the PIO signals [13:0] defined as outputs. PIO[13:0] correspond directly to LD[29:16]. When read, the value in the register is driven onto the local data bus LD[29:16]. When written, the value in the register is driven onto those PIO[13:0] signals defined as outputs in the PIO Function register. To set or clear a single PIO[13:0] bit, the register must be read and a logical AND or OR operation performed on the bit, then the value is written back into the register.

PIO Pin Register

Local Address $FFFD 19xx

Bits 29:16 Reflect the status of PIO signals [13:0] respectively (i.e., bit 29 = PIO 13, etc.).

This register is read-only and reflects the instantaneous value on those PIO[13:0] signals configured as inputs. Reading this register takes the logic value at the PIO[13:0] signal and drives it onto the local data bus LD[29:16]. Writing to this register causes a DSACK1* assertion and has no effect on the contents of the register.

PIO Direction Register

Local Address $FFFD 1Axx

Bit 30 When set, this bit enables FCIACK* assertion upon access to $FFFF FFxx independent of the function codes. This is useful for interrupt acknowledge emulation for non-68K processors.

Bits 29:16 These bits correspond directly to PIO signals [13:0]. When set, the direction of the PIO signals are output from VAC068A. When cleared, the direction of the PIO signals [13:0] are input to VAC068A. These register bits have no effect if the corresponding PIO Function register bits ($FFFD 1Bxx) are set.

PIO Function Register

Local Address $FFFD 1Bxx

Bit 31 When set, this bit asserts FCIACK* upon access to IOSEL5* address space independent of the function codes. Also, access to IOSEL4* ad-

dress space asserts FPUCS*. When cleared, access to IOSEL4* and IOSEL5* address space does not affect the FCIACK* and FPUCS* signals.

Bit 30 — When set, this bit enables the debounce delay associated with PIO9 (i.e., 26.7-ms debounce circuit delay). See PIO9 Debounce delay description for further details. When cleared, the debounce delay is disabled.

Bits 29:16 — These bits select whether the shared function of the PIO pins are enabled. If set, the signal is always an output and operates with the shared function per the following table. If cleared, the signals operate in the PIO mode.

Bit	*General Purpose*	*Shared Function*
29	PIO signal 13	IOSEL2* address range $FFF4 0000 select
28	PIO signal 12	Shared resources chip select output
27	PIO signal 11	Interrupt request pin 11 (output)
26	PIO signal 10	Interrupt request pin 10 (output)
25	PIO signal 9	IOSEL5* address range $FFFA 0000 select
24	PIO signal 8	IOSEL4* address range $FFF8 0000 select
23	PIO signal 7	Interrupt request pin 7 (output)
22	PIO signal 6	IOSEL3* address range $FFF6 0000 select
21	PIO signal 5	I/O write signal
20	PIO signal 4	I/O read signal
19	PIO signal 3	UART B receive data signal
18	PIO signal 2	UART B transmit data signal
17	PIO signal 1	UART A receive data signal
16	PIO signal 0	UART A transmit data signal

Interrupt request functions (bits 27, 26, and 23) are mapped in the Interrupt Control register ($FFFD 16xx).

CPU Clock Divisor Register

Local Address $FFFD 1Cxx

Bits 31:24 — These bits set the 16X baud rate clock of 153.6 kHz for use with the VAC068A UART. This register is loaded into an up-counter that continuously counts from the loaded value to $FF and reloads on the next clock. The table below gives examples of some CPU clock frequencies and the respective divisor to generate a baud rate of 9600.

CPU Clock	*Register Divisor*
16 MHz	105

16.67 MHz	108
20 MHz	131
25 MHz	164
30 MHz	196
33 MHz	216

Note: Baud rate = CPUCLK/(Divisor * 16)

UART Channel A and B Mode Register

Local Address $FFFD 1Dxx Channel A Mode register.

Local Address $FFFD 1Fxx Channel B Mode register.

Bit 31 When set, parity check and generate are disabled. When cleared, parity generate and check are enabled.

Bit 30 When set, even parity check and generate are enabled. When cleared, odd parity check and generate are enabled.

Bit 29 When set, 8 data bits per character are enabled. When cleared, 7 data bits per character.

Bit 28:26 These bits set the baud rate for both the transmitter and receiver. The highest baud rate is derived from the CPU Clock Divisor register. The subsequent baud rates are a division of 2 from the previous baud rate. An example follows:

111 = baud rate of 9600
110 = baud rate of 4800
.
.
000 = baud rate of 75

Bit 25 When set, it allows the character receiver to run. When cleared, the receiver is reset.

Bit 24 When set, it allows the character transmitter to run. When cleared, the transmitter is reset.

Bit 23 When set, it enables the transmitter. When cleared, the transmitter is disabled.

Bit 22 When set, it enables the receiver. When cleared, the receiver is disabled.

Bit 21 When set, a continuous break is sent. When cleared, break is disabled.

Bit 20 When set, this bit enables looping of the transmitter output to the receiver FIFO register. When cleared, looping is disabled.

UART Channel A and B Transmit Data Register

Local Address $FFFD 1Exx Channel A Transmit Data register.

Local Address $FFFD 22xx Channel B Transmit Data register.

Bits 31:24 These bits are loaded with data to be transmitted via the TXD* output when configured in the PIO Function register and enabled in the UART Mode register.

UART Channel A and B Receiver FIFO Register

Local Address $FFFD 20xx Channel A Receiver FIFO register.

Local Address $FFFD 21xx Channel B Receiver FIFO register.

Bit 26 When set, this bit indicates that a break error for this byte was detected; otherwise no break error.

Bit 25 When set, this bit indicates that a frame error for this byte was detected; otherwise no frame error.

Bit 24 When set, this bit indicates that a parity error for this byte was detected; otherwise no parity error.

Bits 23:16 Received characters.

The A and B Receiver FIFO registers are read-only.

UART Channel A and B Interrupt Mask Register

Local Address $FFFD 23xx Channel A Interrupt Mask register.

Local Address $FFFD 24xx Channel B Interrupt Mask register.

Bit 31 When set, enable interrupt on single character. When cleared, disable interrupt.

Bit 30 When set, enable interrupt on receiver FIFO full. When cleared, disable interrupt.

Bit 29 When set, enable interrupt on break change. When cleared, disable interrupt.

Bit 28 When set, enable interrupt on overrun, framing, or parity error. When cleared, disable interrupt.

Bit 27 When set, enable interrupt on transmitter ready. When cleared, disable interrupt.

Bit 26 When set, enable interrupt on transmitter empty. When cleared, disable interrupt.

The pending interrupt must be disabled in this register, serviced, and then cleared in the Interrupt Control register.

UART Channel A and B Interrupt Status Register

Local Address $FFFD 25xx Channel A Interrupt Status register.

Local Address $FFFD 26xx Channel B Interrupt Status register.

Bit 31 When set, this bit indicates that an interrupt has occurred because a character in the receiver is ready to be read.

Bit 30 When set, this bit indicates that an interrupt has occurred because the receiver FIFO is full.

Bit 29 When set, this bit indicates that an interrupt has occurred because a break change was detected.

Bit 28 When set, this bit indicates that an interrupt has occurred because a parity error was detected.

Bit 27 When set, this bit indicates that an interrupt has occurred because a framing error was detected.

Bit 26 When set, this bit indicates that an interrupt has occurred because a overrun error was detected.

Bit 25 When set, this bit indicates that an interrupt has occurred because the transmitter is ready for another character.

Bit 24 When set, this bit indicates that an interrupt has occurred because the transmitter is empty.

This read-only register contains the interrupt status conditions causing the interrupt generated.

Timer Data Register

Local Address $FFFD 27xx

Bits 31:16 This register contains the data for loading the VAC068A internal watchdog timer. This data is loaded into a 16-bit up-counter when RUN/LOAD is Low as well as under control of the reload circuitry when ONCE/CONTINUOUS is Low. When the contents of this register are read, the value of the timer is driven onto the data bus, not the value loaded into the register. The counter clock input is driven from the carry out of the prescale counter.

Note: Refer to the Timer Control register for more information on RUN/LOAD and ONCE/CONTINUOUS.

Timer Control Register

Local Address $FFFD 28xx

Bit 31 ONCE/CONTINUOUS: When cleared, the timer counts continuous and interrupt at the end of expiration. If this bit is set, the timer counts once and stops.

Bit 30 RUN/LOAD: When set, the count is enabled and dependant on bit 31 for control of count cycles. When cleared, the counter is disabled.

Bits 29:24 Prescale Load Value: These bits are loaded into the prescale counter. Bits 29 through 24 correspond directly to D5 through D0 respectively.

The upper two bits of the prescale output (D6, D7) are tied High and not displayed in the register. The prescale counter carry out clocks the count value loaded into the Timer Data register.

Bits 23:16 Prescaler Value: These bits are read-only. They contain the instantaneous value of the prescale counter. Bits 23 through 16 correspond directly to the prescaler counter output Q7 through Q0 respectively.

VAC068A Identification Register

Local Address $FFFD 29xx

Bits 31:20 Constant: These bits are predefined and cannot be changed. A read or write to this register does not affect these bits.

Bits 19:16 Revision number: These bits contain the chip revision number.
VAC068–F5 – 1AC0
VAC068A – 1AC1

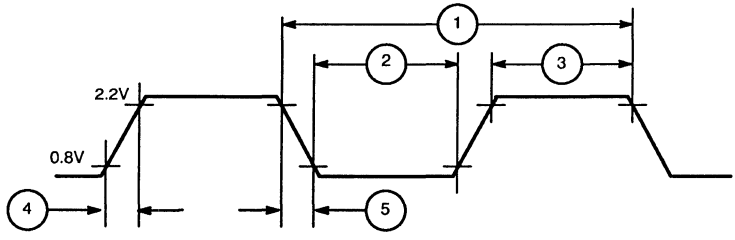
Note: After a global reset and the completion of loading all other registers, this register must be written in order for the VAC068A to enable its decode and compare functions.

22

VAC068A AC Performance Specifications

Clock Input

Num.	Characteristic	Commercial		Military	
		Min.	Max.	Min.	Max.
	Frequency of Operation (MHz)	1	50	1	40
1	Cycle Time (ns)	20	1000	2.5	1000
2, 3	Clock Pulse Width (Measured from 1.5V to 1.5V)			11.25	
4, 5	Rise and Fall Time (ns)		5		5



AC Specifications

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
GLOBAL RESET								
1	RESET*[0] to WORD*[L]		5T		5T		5T	
2	WORD*[0] to RESET* High, WORD*[H]		10T		10T		10T	
REGISTER WRITE								
1	LA[31:8], FCi, R/W* Valid to PAS*[L] (Set-Up Time)		10		10		10	
2	LD[31:16] Valid to DSACKi*[L] (Set-Up Time)		5		5		5	
3	PAS*[0] to DSACKi*[L]		6 + 1T	32 + 2T	5 + 1T	33 + 2T	5 + 1T	35 + 2T
4	PAS*[1] to DSACKi*[H]		5	28	4	29	4	31
5	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
6	PAS*[1] to LD[31:16] Invalid		6	33	5	34	5	36
REGISTER READ								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)		10		10		10	
2	PAS*[0] to DSACKi*[L]		6	32 + 1T	5	33 + 1T	5	35 +1T
3	PAS*[0] to LD[31:16] Valid		9	43	8	45	7	48
4	PAS*[1] to DSACKi*[H]		5	28	4	29	5	31
5	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	
6	PAS*[1] to LD[31:16] Invalid		6	33	5	34	5	36
LOCAL ACCESS VIA LOCAL BUS								
1	LA[31:8], FCi, R/W* to PAS*[0] (Set-Up Time)		10		10		10	
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	39	6	40	6	43
3	PAS*[0] to Chip Select[L]	1, 2	6	34	5	35	5	37
4	PAS*[0] to DSACKi*[L]	3	PI1 + 6	PI1+35+1T	PI1 + 5	PI1+36+1T	PI1 + 5	PI1+38+1T
5	PAS*[0] to IORD*/IOWR*[L]	4	PI3 + 6	PI3+35+1T	PI3 + 6	PI3+36+1T	PI3 + 5	PI3+38+1T
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	
7	PAS*[1] to ASIZ1/0 WORD* Invalid		7	40	7	42	6	44
8	PAS*[1] to Chip Select*[H]	1	4	25	3	26	3	28
9	PAS*[1] to DSACKi*[H]		4	27	4	28	4	29
10	PAS*[1] to IORD*[H] /IOWR*[H]	4	PI3 + 4	PI3+25+1T	PI3 + 3	PI3+26+1T	PI3 + 3	PI3+27+1T
LOCAL ACCESS VIA VMEbus								
1	PAS*[0] to ASIZ1/0, WORD Valid		7	39	6	40	6	43
2	PAS*[0] to Chip Select[L]	5	6	34	5	35	5	37
3	PAS*[0] to DSACKi*[L]	6	PI2 + 6	PI2+35+1T	PI2 + 5	PI2+36+1T	PI2 + 5	PI2+38+1T
4	PAS*[0] to IORD*[L], IOWR*[L]	4	PI3 + 6	PI3 + 3 + 1T	PI3 + 6	PI3+36+1T	PI3 + 5	PI3+38+1T
5	PAS*[1] to ASIZ1/0, WORD* Invalid		7	40	7	42	6	44
6	PAS*[1] to Chip Select[H]	4	4	25	3	26	3	28
7	PAS*[1] to DSACKi*[H]	7	5	28	4	29	4	31
8	PAS*[1] to IORD*[H], IOWR*[H]	4	PI3 + 4	PI3 +25+1T	PI3 + 3	PI3+26+1T	PI3 + 3	PI3+27+1T

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
VMEbus SLAVE/SLAVE BLOCK ACCESS								
1	AS*[0] to SLSELi*[L] or ICFSEL[L]		3	23	3	24	2	26
2	LAEN[1] to LA[31:8] Valid		4	28	3	29	3	31
3	AS*[1] to SLSELi*[H] or ICFSEL[L]		6	33	5	34	5	36
4	LAEN[0] to LA[31:0] Valid		12	63	10	65	10	69
VMEbus MASTER ACCESS								
1	LA[31:8], FCi, R/W* to PAS*[0] (Set-Up Time)		10		10		10	
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	39	6	40	6	43
3	PAS*[0] to MWB*[L]		11	57	10	59	10	63
4	ABEN*[0] to A[31:8] Valid		3	21	2	22	2	21
5	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	
6	PAS*[1] to ASIZ1/0, WORD* Invalid		7	40	7	42	6	44
7	PAS*[1] to MWB*[L]		3	19	2	20	2	21
8	ABEN*[1] to A[31:8] Invalid		2	13	1	13	1	14
MASTER BLOCK TRANSFER INITIATION CYCLE								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)		10		10		10	
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	39	6	40	6	43
3	PAS*[0] to DSACKi[L]		6 + 1T	32 + 2T	5 + 1T	33 + 2T	5 + 1T	35 + 2T
4	PAS*[0] to MWB*[L]		11	57	10	59	10	63
5	PAS*[1] to ASIZ1/0, WORD* (Hold Time)		7	40	7	42	6	44
6	PAS*[1] to DSACKi*[H]		5	28	4	29	4	31
7	PAS*[1] to MWB*[H]		3	19	2	20	2	21
8	ABEN*[0] to A[31:8] Valid		3	21	2	22	2	24
9	LAEN[1], FCi Valid to LA[31:8] Valid		4	28	3	29	3	31
10	LAEN[1], FCi Valid to LDMACK*[1]		1	13	1	13	1	17
BOUNDARY CROSSING								
1	BLT*[1] to LA[31:8] Incremented		1	13	1	13	1	17
2	LADO[0] to A[31:8] Incremented		7	21	5	21	4	26

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
PIO OUTPUT								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)		10		10		10	
2	LD[31:16] to PAS*[0] (Set-Up Time)		5		5		5	
3	PAS*[0] to PIO[13:0] Valid		4	24	3	25	3	26
4	PAS*[0] to DSACKi*[L]		6 + 1T	32 + 2T	5 + 1T	33 + 2T	5 + 1T	35 + 2T
5	PAS*[1] to DSACKi*[H]		4	27	4	28	4	29
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	
7	PAS*[1] to LD[31:16] Invalid		6	33	5	34	5	36
PIO INPUT								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)		10		10		10	
2	PIO[13:0] to PAS*[0] (Set-Up Time)		5		5		5	
3	PAS*[0] to DSACKi*[L]		6	32 + 1T	5	33 + 1T	5	35 + 1T
4	PAS*[0] to LD[31:16] Valid		9	43	8	45	7	48
5	PAS*[1] to DSACKi*[H]		4	27	4	28	4	29
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)		5		5		5	
7	PAS*[1] to LD[31:16] Invalid		6	33	5	34	5	36

Notes:

1. Chip select can be any of DRAMCS*, EPROMCS*, SHRCS*, VSBSEL*, FPUCS*, CS*, or IOSELi*.
2. The Decode Control register provides facilities to condition DRAMCS* or boundary decodes with the assertion of PAS*.
3. PI1 is the programmable interval for EPROMCS*, SHRCS*, and IOSELi* in the DSACKi* Control register.
4. PI3 is the programmable interval for IORD and IOWR in the Decode Control register.
4. Chip select can be any of DRAMCS*, EPROMCS*, SHRCS*, or VSBSEL*.
6. PI2 is the programmable interval for EPROMCS*, SHRCS*, DRAMCS*, or VSBSEL* in the Decode Control register.
7. SLSELi* redirection is enabled in the Decode Control register.

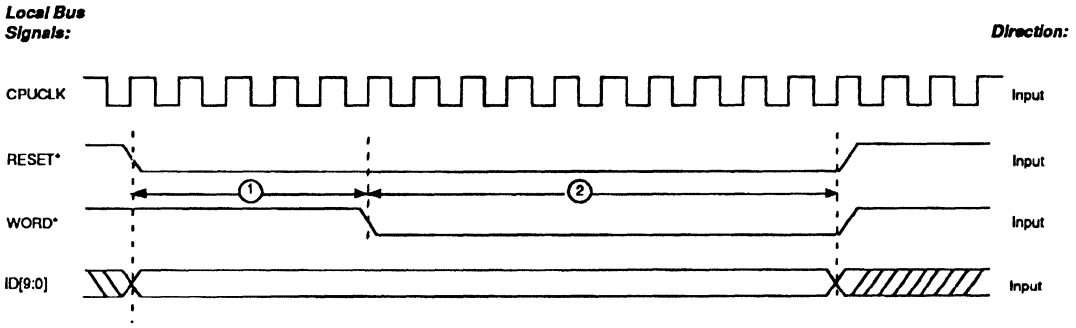


Figure 22–1. VAC068A Global Reset

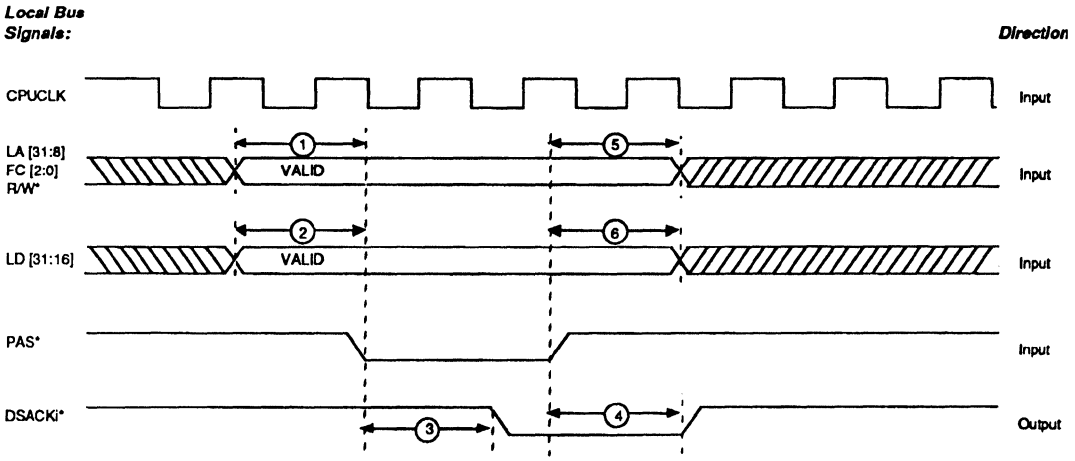


Figure 22–2. VAC068A Register Write

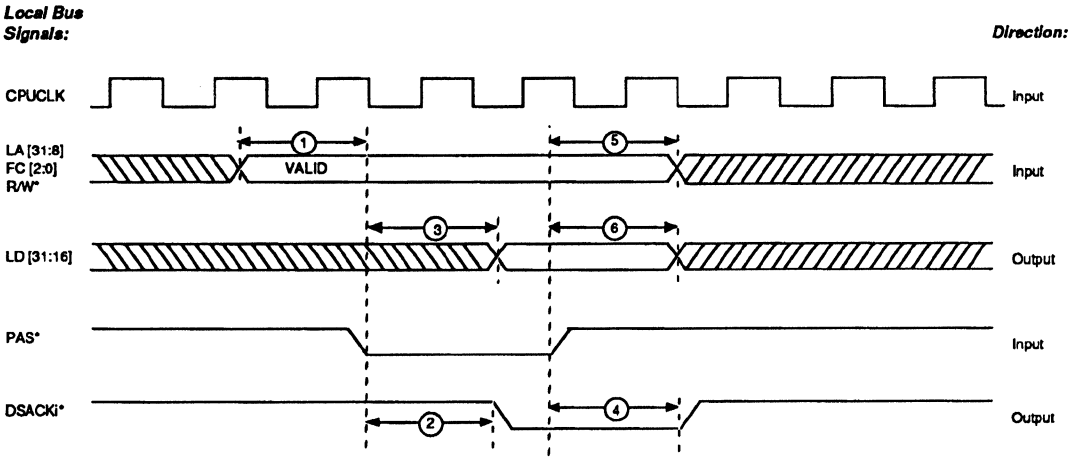


Figure 22–3. VAC068A Register Read

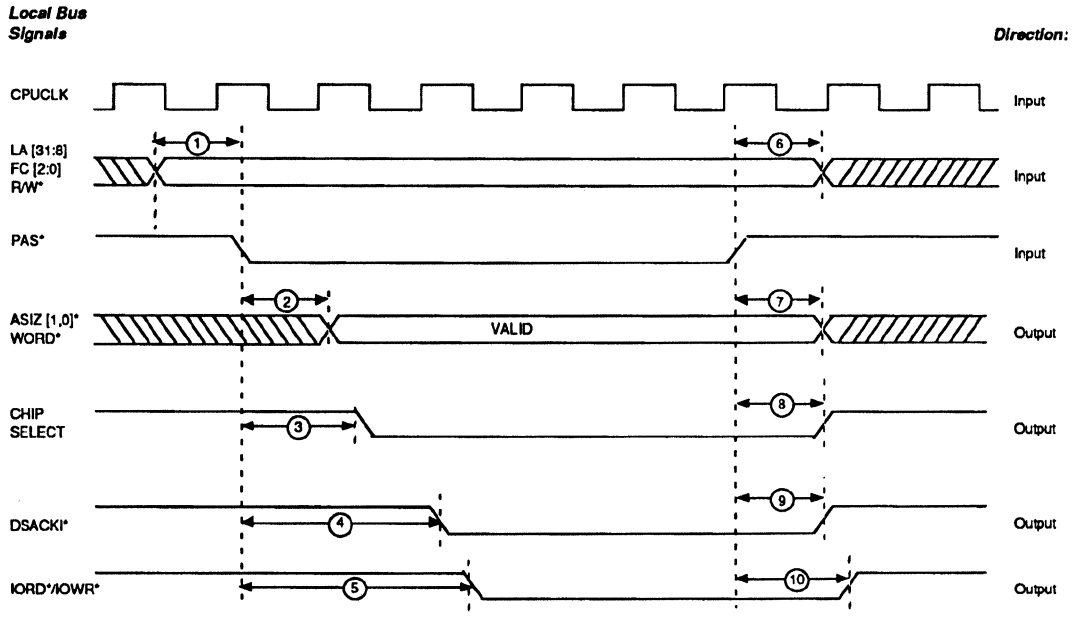


Figure 22–4. Local Resource Access via Local Bus

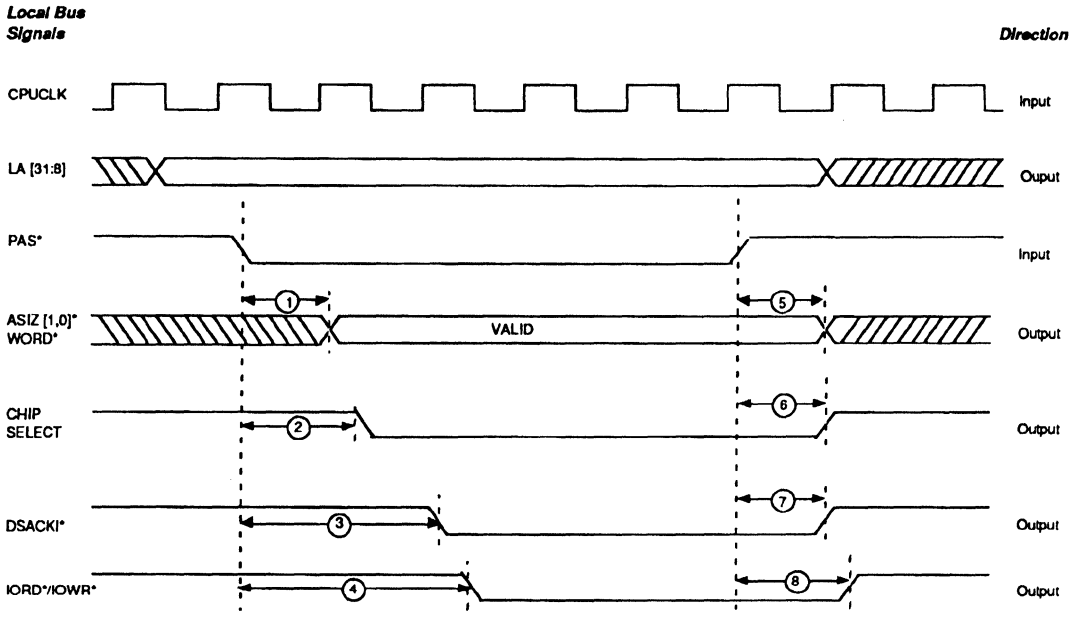


Figure 22–5. Local Resource Accesses via VMEbus

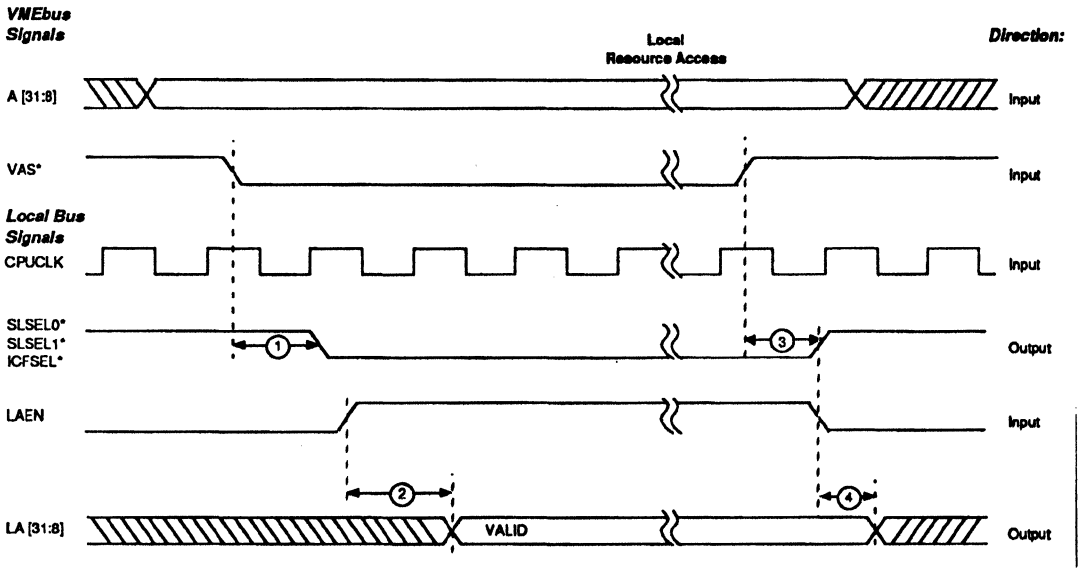


Figure 22–6. VMEbus Accesses – Slave

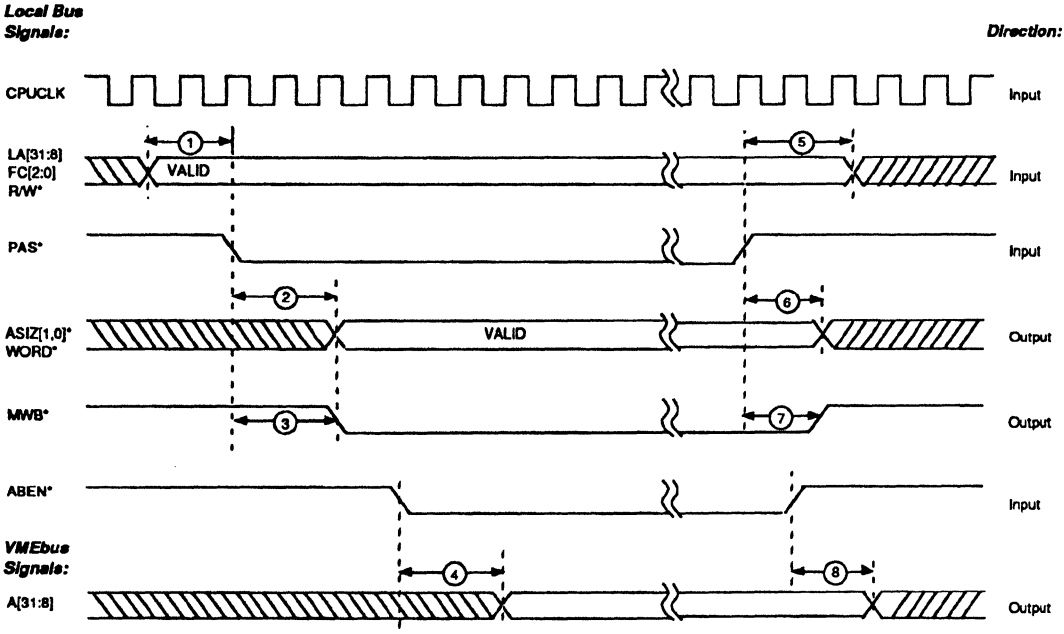


Figure 22–7. VMEbus Accesses – Master

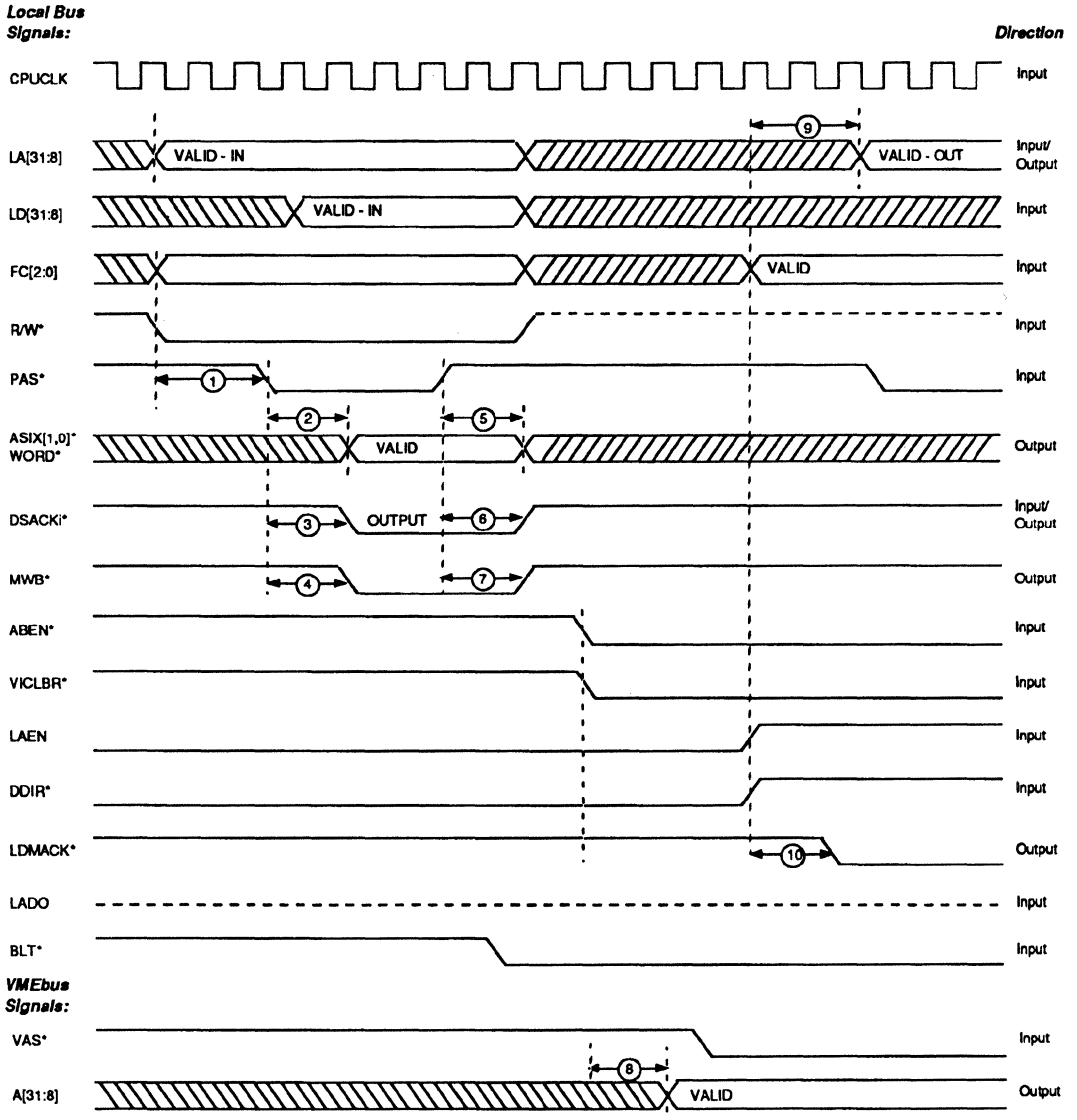


Figure 22–8. Master Block Transfer – Initialization Cycle

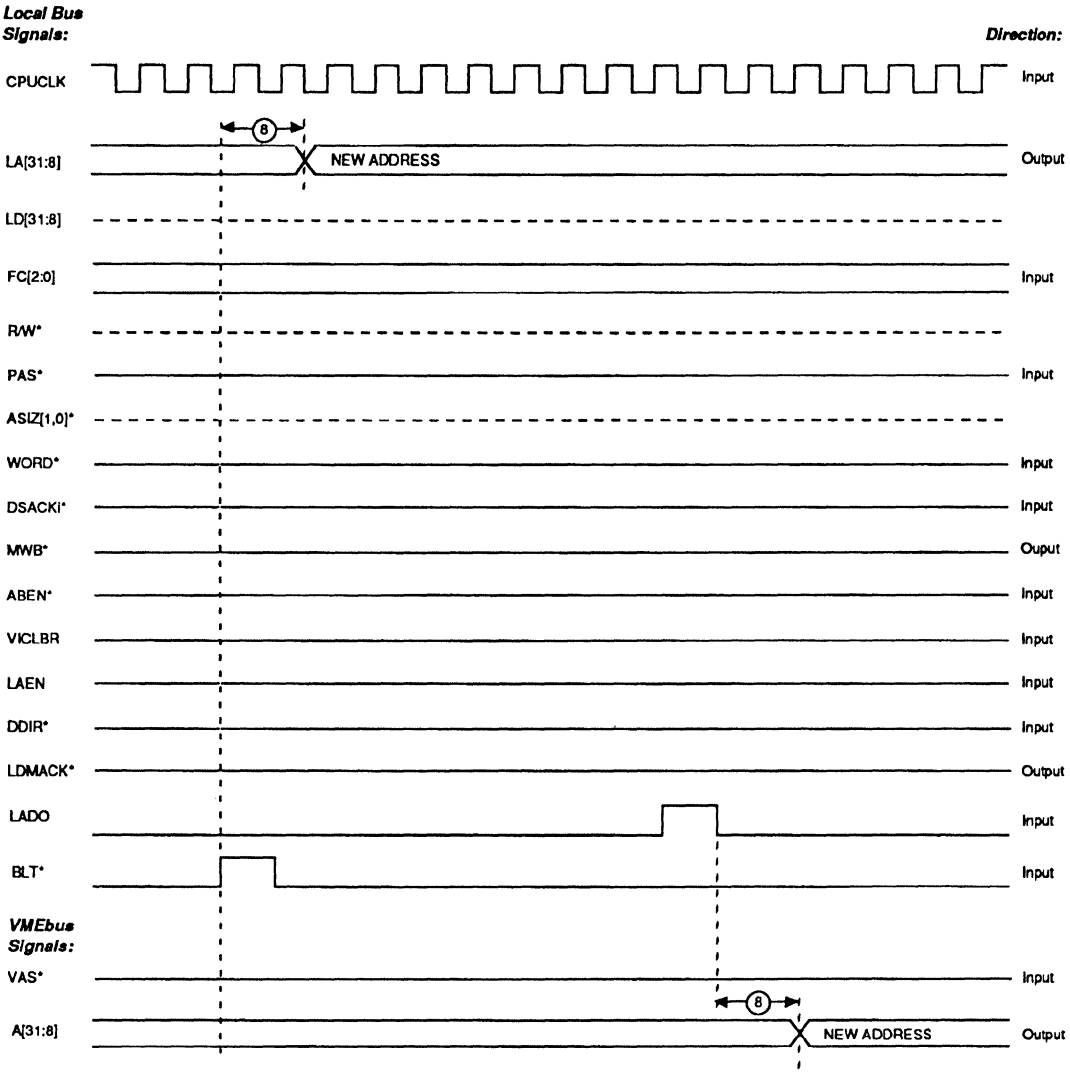


Figure 22–9. Master Block Transfer – Local and VME Boundary Crossing

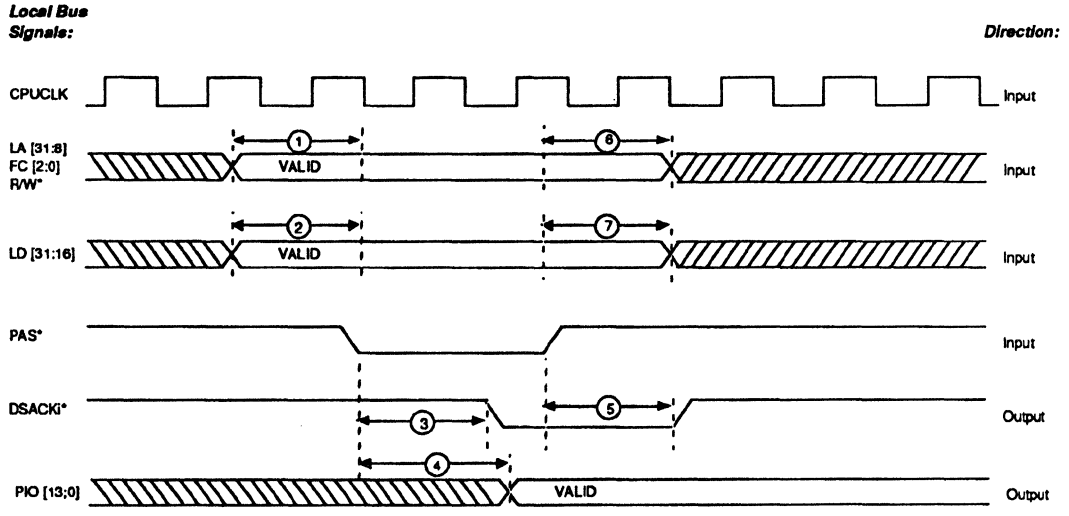


Figure 22–10. PIO Operation – Output

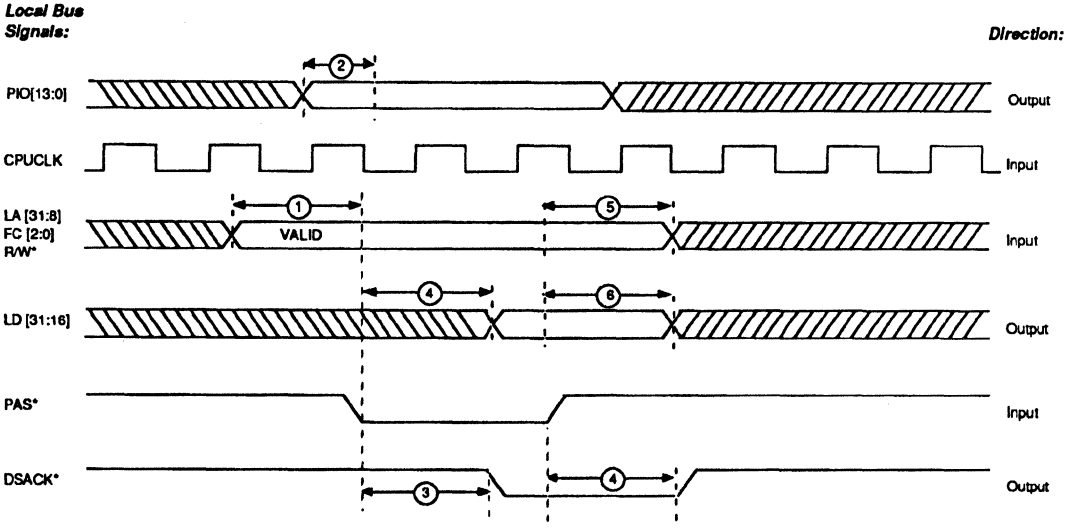


Figure 22–11. PIO Operation – Input

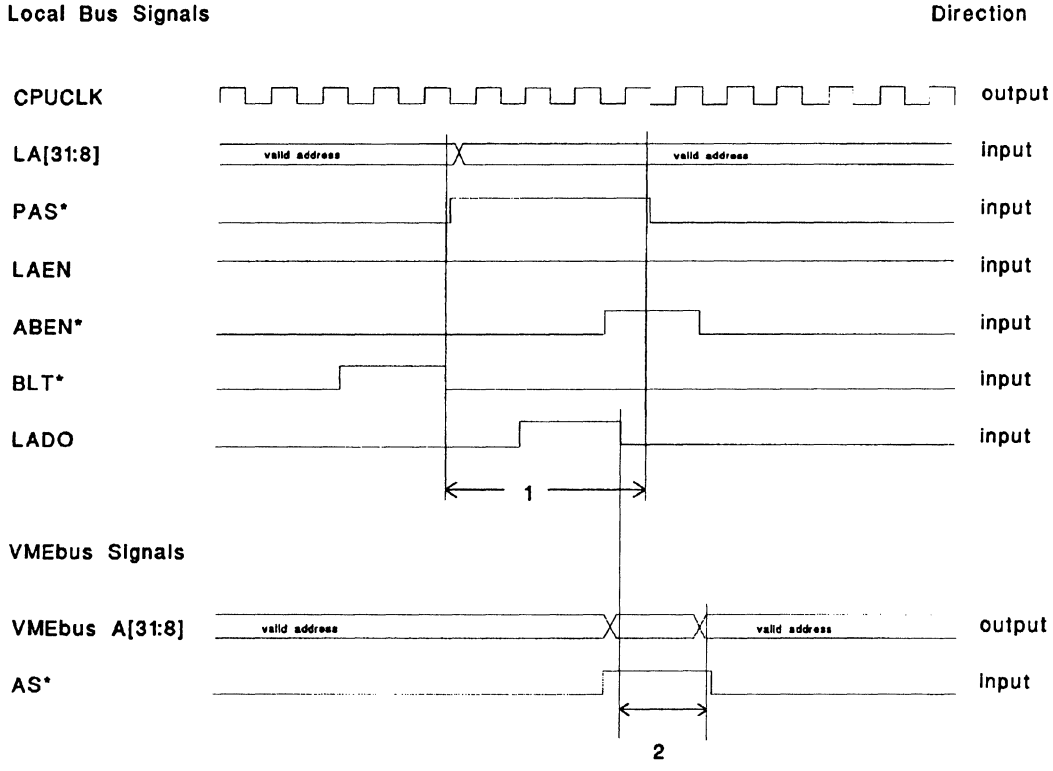


Figure 22–12. Master Block Transfer – Local and VMEbus Boundary Crossing (DMA Write Cycle)

23

VAC068A Signal List and Pinouts

Table 23–1. VMEbus Signals

Name	PGA Pin	QFP Pin	Type	Description
A08	A8	139	Three-State I/O	VMEbus Address Signals
A09	B8	138	Three-State I/O	VMEbus Address Signals
A10	A7	142	Three-State I/O	VMEbus Address Signals
A11	B7	144	Three-State I/O	VMEbus Address Signals
A12	A6	143	Three-State I/O	VMEbus Address Signals
A13	B6	145	Three-State I/O	VMEbus Address Signals
A14	A5	147	Three-State I/O	VMEbus Address Signals
A15	B5	148	Three-State I/O	VMEbus Address Signals
A16	A4	149	Three-State I/O	VMEbus Address Signals
A17	A3	150	Three-State I/O	VMEbus Address Signals
A18	B4	151	Three-State I/O	VMEbus Address Signals
A19	B3	153	Three-State I/O	VMEbus Address Signals
A20	A2	154	Three-State I/O	VMEbus Address Signals
A21	C3	156	Three-State I/O	VMEbus Address Signals
A22	B2	157	Three-State I/O	VMEbus Address Signals
A23	A1	158	Three-State I/O	VMEbus Address Signals
A24	B9	134	Three-State I/O	VMEbus Address Signals
A25	A9	137	Three-State I/O	VMEbus Address Signals
A26	B10	132	Three-State I/O	VMEbus Address Signals
A27	A10	136	Three-State I/O	VMEbus Address Signals
A28	B11	129	Three-State I/O	VMEbus Address Signals
A29	A11	133	Three-State I/O	VMEbus Address Signals
A30	A13	128	Three-State I/O	VMEbus Address Signals
A31	A12	130	Three-State I/O	VMEbus Address Signals
AS*	D2	6	Input	VMEbus Address Strobe Signal
ABEN*	E2	9	Input	VMEbus Address Bus Enable Signal

Table 23–2. Local Signals

Name	PGA Pin	QFP Pin	Type	Description
ASIZ0	P1	34	Three-State I/O	Identifies VMEbus Address Size
ASIZ1	M3	35	Three-State I/O	Identifies VMEbus Address Size
BLT*	F1	16	Input	DMA Control Signal
CACHINH*	P13	73	Open Collector Input	Data Cache Inhibit to Processor
CPUCLK	N3	36	Input	CPU Clock Input
CS*	D14	111	Output	Chip Select to VIC068A Signal
DDIR	C1	8	Input	Swap Data Direction Buffer
DRAMCS*	N11	72	Output	DRAM Chip Select
DSACK0*	P5	49	Three-State I/O	Data and Size Acknowledge
DSACK1*	R3	48	Three-State I/O	Data and Size Acknowledge
EPROMCS*	P12	71	Output	EPROM Chip Select
FC0	P2	37	Input	CPU, VIC068A Function Code Input
FC1	R1	38	Input	CPU, VIC068A Function Code Input
FC2	N4	43	Input	CPU, VIC068A Function Code Input
FCIACK*	N2	33	Output	Interrupt Acknowledge Cycle
FPUCS*	R13	70	Output	Floating-Point Coprocessor Select
ICFSEL*	H1	19	Output	Interprocessor Communications Select
ID08	K1	23	Three-State I/O	Isolated Local Data Signals
ID09	K2	25	Three-State I/O	Isolated Local Data Signals
ID10	J2	24	Three-State I/O	Isolated Local Data Signals
ID11	L1	27	Three-State I/O	Isolated Local Data Signals
ID12	L2	28	Three-State I/O	Isolated Local Data Signals
ID13	M1	29	Three-State I/O	Isolated Local Data Signals
ID14	N1	30	Three-State I/O	Isolated Local Data Signals
ID15	L3	32	Three-State I/O	Isolated Local Data Signals
IOSEL0*	R15	78	Output	Local I/O Device Chip Select
IOSEL1*	J14	94	Output	Local I/O Device Chip Select
LA08	P14	77	Three-State I/O	Local Address Signals
LA09	M13	83	Three-State I/O	Local Address Signals
LA10	P15	85	Three-State I/O	Local Address Signals
LA11	N13	76	Three-State I/O	Local Address Signals

Table 23–2. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
LA12	N14	84	Three-State I/O	Local Address Signals
LA13	L13	87	Three-State I/O	Local Address Signals
LA14	M14	86	Three-State I/O	Local Address Signals
LA15	L14	89	Three-State I/O	Local Address Signals
LA16	N15	88	Three-State I/O	Local Address Signals
LA17	K14	92	Three-State I/O	Local Address Signals
LA18	M15	90	Three-State I/O	Local Address Signals
LA19	K15	96	Three-State I/O	Local Address Signals
LA20	L15	93	Three-State I/O	Local Address Signals
LA21	J15	97	Three-State I/O	Local Address Signals
LA22	H14	98	Three-State I/O	Local Address Signals
LA23	H15	99	Three-State I/O	Local Address Signals
LA24	G14	104	Three-State I/O	Local Address Signals
LA25	G15	102	Three-State I/O	Local Address Signals
LA26	F14	105	Three-State I/O	Local Address Signals
LA27	F15	103	Three-State I/O	Local Address Signals
LA28	E15	107	Three-State I/O	Local Address Signals
LA29	F13	106	Three-State I/O	Local Address Signals
LA30	C15	110	Three-State I/O	Local Address Signals
LA31	E14	108	Three-State I/O	Local Address Signals
LD16	P6	52	Three-State I/O	Local Data Signals
LD17	R6	56	Three-State I/O	Local Data Signals
LD18	P7	54	Three-State I/O	Local Data Signals
LD19	R4	50	Three-State I/O	Local Data Signals
LD20	R7	57	Three-State I/O	Local Data Signals
LD21	R5	53	Three-State I/O	Local Data Signals
LD22	P8	58	Three-State I/O	Local Data Signals
LD23	N7	55	Three-State I/O	Local Data Signals
LD24	N8	60	Three-State I/O	Local Data Signals
LD25	R8	59	Three-State I/O	Local Data Signals
LD26	R9	62	Three-State I/O	Local Data Signals

Table 23–2. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
LD27	P9	64	Three-State I/O	Local Data Signals
LD28	R10	63	Three-State I/O	Local Data Signals
LD29	P10	65	Three-State I/O	Local Data Signals
LD30	R11	67	Three-State I/O	Local Data Signals
LD31	P11	68	Three-State I/O	Local Data Signals
LADO	D3	3	Input	Latch VMEbus Address Out Signal
LADI	E1	13	Input	Latch Local Address In Signal
LAEN	P3	44	Input	Local Address Bus Enable Signal
LBR*	G3	15	Input	Local Bus Request to VIC068A Signal
LDMACK*	E3	7	Output	Local DMA is in Progress
MWB*	R12	69	Output	Module Wants Local Bus Signal
PAS*	R2	45	Input	Processor Address Strobe
PIO0/TXDA	C11	127	Three-State I/O	General-Purpose I/O or UART A Transmit Signal
PIO1/RXDA	B12	126	Three-State I/O	General-Purpose I/O or UART A Receive Signal
PIO2/TXDB	A14	125	Three-State I/O	General-Purpose I/O or UART B Transmit Signal
PIO3/RXDB	B13	124	Three-State I/O	General-Purpose I/O or UART B Receive Signal
PIO4/ IORD*	F2	12	Three-State I/O	General-Purpose I/O or I/O Read Signal
PIO5/ IOWR*	C12	123	Three-State I/O	General-Purpose I/O or I/O Write Signal
PIO6/ IOSEL3*	B14	117	Three-State I/O	General-Purpose I/O or I/O Select 3 Signal
PIO7/ Interrupt	C13	116	Three-State I/O	General-Purpose I/O or Interrupt Request Signal
PIO8/ IOSEL4*	D13	115	Three-State I/O	General-Purpose I/O or I/O Select 4 Signal
PIO9/ IOSEL5*	B15	114	Three-State I/O	General-Purpose I/O or I/O Select 5 Signal
PIO10/ Interrupt	C14	113	Three-State I/O	General-Purpose I/O or Interrupt Request Signal

Table 23–2. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
PIO11/ Interrupt	D1	10	Three-State I/O	General-Purpose I/O or Interrupt Request Signal
PIO12/ SHRCS*	D15	109	Three-State I/O	General-Purpose I/O or Shared Resources Chip Select Signal
PIO13/ IOSEL2*	B1	5	Three-State I/O	General-Purpose I/O or I/O Select 2 Signal
REFGT*	G1	17	Output	Refresh Grant Signal
RESET*	R14	74	Input	System Reset Signal
R/W*	P4	46	Input	Read/Write Signal
SLSEL0*	H2	18	Output	Slave Select 0 Signal
SLSEL1*	J1	22	Output	Slave Select 1 Signal
SWDEN*	C2	4	Input	Swap Data Enable Signal
VSBSEL*	G2	14	Output	VSB Chip Select Signal
WORD*	M2	31	Output	16-Bit Data Access Signal

Table 23–3. Power Supply Signals[1]

Name	PGA Pin	Type	Description
V_{DD}	A15	Input	Power Input
V_{DD}	C5	Input	Power Input
V_{DD}	C7	Input	Power Input
V_{DD}	C9	Input	Power Input
V_{DD}	H3	Input	Power Input
V_{DD} Core	H13	Input	Power Input
V_{DD}	J13	Input	Power Input
V_{DD}	N5	Input	Power Input
V_{DD}	N10	Input	Power Input
V_{DD}		Input	Power Input
V_{DD}		Input	Power Input
V_{DD}		Input	Power Input
V_{DD}		Input	Power Input

Table 23–3. Power Supply Signals (continued)

Name	PGA Pin	Type	Description
V_{DD}		Input	Power Input
V_{DD}		Input	Power Input
V_{SS}	C4	Input	Ground
V_{SS}	C6	Input	Ground
V_{SS}	C8	Input	Ground
V_{SS}	C10	Input	Ground
V_{SS}	E13	Input	Ground
V_{SS}	F3	Input	Ground
V_{SS}	G13	Input	Ground
V_{SS} Core	J3	Input	Ground
V_{SS}	K3	Input	Ground
V_{SS}	N6	Input	Ground
V_{SS}	N9	Input	Ground
V_{SS}	N12	Input	Ground
V_{SS}	K13	Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground
V_{SS}		Input	Ground

Note:

1. For QFP power supply signals, see *Table 23–4*.

Table 23–4. Pinout for VAC068A Plastic and Ceramic Quad Flatpack (160-Pin): Cavity Up

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{SS}	32	ID15	63	LD28	94	IOSEL1*
2	V_{SS}	33	FCIACK*	64	LD27	95	V_{DD}
3	LADO*	34	ASIZ0*	65	LD29	96	LA19
4	SWDEN*	35	ASIZ1*	66	V_{DD}	97	LA21
5	PIO13–IOSEL2*	36	CPUCLK	67	LD30	98	LA22
6	AS*	37	FC0	68	LD31	99	LA23
7	LDMACK*	38	FC1	69	MWB*	100	V_{DD} Core
8	DDIR	39	V_{SS}	70	FPUCS*	101	V_{SS}
9	ABEN*	40	V_{SS}	71	EPROMCS*	102	LA25
10	PIO11	41	V_{DD}	72	DRAMCS*	103	LA27
11	V_{SS}	42	V_{DD}	73	CACHINH*	104	LA24
12	PIO4–IORD*	43	FC2	74	RESET*	105	LA26
13	LADI	44	LAEN	75	V_{SS}	106	LA29
14	VSBSEL*	45	PAS*	76	LA11	107	LA28
15	LBR*	46	R/W*	77	LA8	108	LA31
16	BLT*	47	V_{DD}	78	IOSEL0*	109	PIO12–SHRCS*
17	REFGT*	48	DSACK1*	79	V_{DD}	110	LA30
18	SLSEL0*	49	DSACK0*	80	V_{DD}	111	CS*
19	ICFSEL*	50	LD19	81	V_{SS}	112	V_{SS}
20	V_{DD}	51	V_{SS}	82	V_{SS}	113	PIO10
21	V_{SS} Core	52	LD16	83	LA9	114	PIO9–IOSEL5*
22	SLSEL1*	53	LD21	84	LA12	115	PIO8–IOSEL4*
23	ID8	54	LD18	85	LA10	116	V_{DD}
24	ID10	55	LD23	86	LA14	117	PIO6–IOSEL3*
25	ID9	56	LD17	87	LA13	118	V_{DD}
26	V_{SS}	57	LD20	88	LA16	119	V_{SS}
27	ID11	58	LD22	89	LA15	120	V_{SS}
28	ID12	59	LD25	90	LA18	121	V_{DD}
29	ID13	60	LD24	91	V_{SS}	122	V_{DD}
30	ID14	61	V_{SS}	92	LA17	123	PIO5–IOWR*
31	WORD*	62	LD26	93	LA20	124	PIO3–RXDB

Table 23–4. Pinout for VAC068A Plastic and Ceramic Quad Flatpack: Cavity Up (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
125	PIO2–TXDB	134	A24	143	A12	152	V_{DD}
126	PIO1–RXDA	135	V_{DD}	144	A11	153	A19
127	PIO0–TXDA	136	A27	145	A13	154	A20
128	A30	137	A25	146	V_{SS}	155	V_{SS}
129	A28	138	A9	147	A14	156	A21
130	A31	139	A8	148	A15	157	A22
131	V_{SS}	140	V_{SS}	149	A16	158	A23
132	A26	141	V_{DD}	150	A17	159	V_{DD}
133	A29	142	A10	151	A18	160	V_{DD}

A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	
A23	PIO13/ IOSEL2*	DDIR	PIO11	LADI	BLT*	REFGT*	ICFSEL*	SLSEL1*	ID8	ID11	ID13	ID14	ASIZ0*	FC1	1
A20	A22	SWDEN*	VAS*	ABEN*	PIO4/ IORD*	VSBSEL*	SLSEL0	ID10	ID9	ID12	WORD*	FCIACK*	FC0	PAS*	2
A17	A19	A21	LADO	LDMACK*	VSS	LBR*	VDD	VSS	VSS	ID15	ASIZ1*	CPUCLK	LAEN	DSACK1*	3
A16	A18	VSS	LOCATOR PIN									FC2	R/W*	LD19	4
A14	A15	VDD										VDD	DSACK0*	LD21	5
A12	A13	VSS										VSS	LD16	LD17	6
A10	A11	VDD										LD23	LD18	LD20	7
A08	A09	VSS										LD24	LD22	LD25	8
A25	A24	VDD										VSS	LD27	LD26	9
A27	A26	VSS										VDD	LD29	LD28	10
A29	A28	PIO0/ TXDA										DRAMCS*	LD31	LD30	11
A31	PIO1/ RXDA	PIO5/ IOWR*										VSS	EPROMCS*	MWB*	12
A30	PIO3/ RXDB	PIO7	PIO8/ IOSEL4*	VSS	LA29	VSS	VDD	VDD	VSS	LA13	LA9	LA11	CACHINH*	FPUCS*	13
P102/ TXDB	PIO6/ IOSEL3*	PIO10	CS*	LA31	LA26	LA24	LA22	IOSEL1*	LA17	LA15	LA14	LA12	LA8	RESET*	14
VDD	PIO9/ IOSEL5*	LA30	PIO12/ SHRCS*	LA28	LA27	LA25	LA23	LA21	LA19	LA20	LA18	LA16	LA10	IOSEL0*	15

Figure 23–1. VAC068A Pin Grid Array (PGA), Bottom View

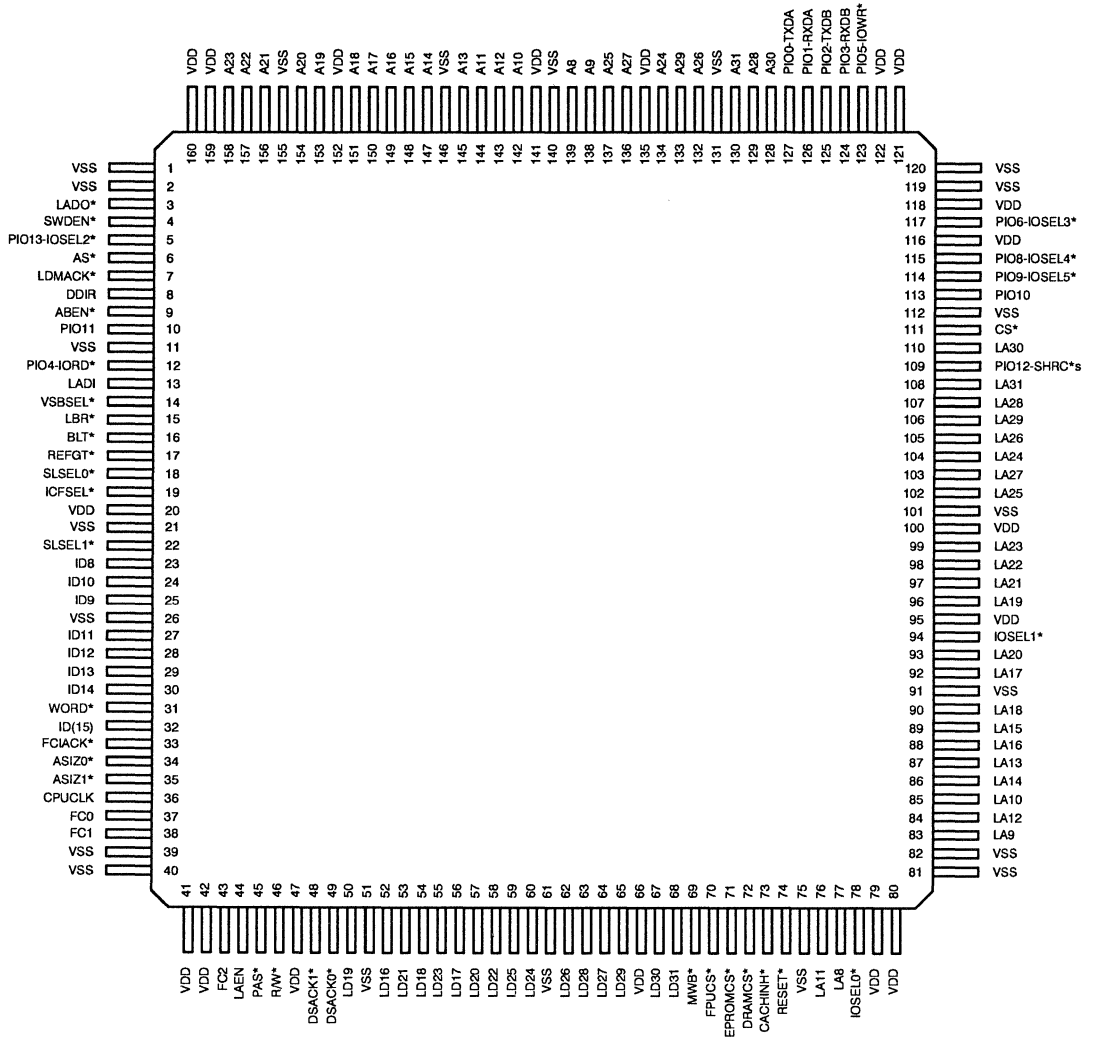


Figure 23–2. VAC068A Quad Flatpack (QFP), Top View

Section 3

VIC068A and VAC068A

Specifications

24

DC Performance Specifications

Table 24–1. VMEbus Signals (AS*, DS1*, DS0*, BCLR*, SYSCLK)

Parameter	Description	Test Conditions		Military	Industrial	Comm.	Units
V_{IH}	Minimum High-Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low-Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High-Level Output Voltage	V_{CC} = Min., I_{OH} = −3 mA		2.4	2.4	2.4	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 64 mA, 56 mA, 48 mA		0.6	0.6	0.6	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., V_{IN} = 0.6–2.4		±5	±5	±5	µA
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	I_{IN} = −18 mA	−1.2	−1.2	−1.2	V
			I_{IN} = 18 mA	V_{CC}+1.2	V_{CC}+1.2	V_{CC}+1.2	V
I_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. GND ≤ V_{OUT} ≤ V_{CC} All Outputs Disabled		±10	±10	±10	µA

Table 24–2. VMEbus Signals (Low Drive. All VMEbus Daisy-Chain Signals.)

Parameter	Description	Test Conditions		Military	Industrial	Comm.	Units
V_{IH}	Maximum High-Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low-Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High-Level Output Voltage	V_{CC} = Min., I_{OH} = −8 mA		2.4	2.4	2.4	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 8 mA		0.6	0.6	0.6	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., V_{IN} = 0.6–2.4		±5	±5	±5	µA
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	I_{IN} = −18 mA	−1.2	−1.2	−1.2	V
			I_{IN} = 18 mA	V_{CC}+1.2	V_{CC}+1.2	V_{CC}+1.2	V
I_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. V_{OUT} = 0.6/2.4V All Outputs Disabled		±10	±5	±5	µA

Table 24–3. VMEbus Signals (Medium Drive. All non-High, non-Low Drive Signals, All VAC068A VMEbus Signals.)

Parameter	Description	Test Conditions		Military	Industrial	Comm.	Units
V_{IH}	Maximum High-Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low-Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High-Level Output Voltage	V_{CC} = Min., I_{OH} = −3 mA		2.4	2.4	2.4	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 48 mA		0.6	0.6	0.6	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., V_{IN} = 0.6–2.4		±5	±5	±5	µA
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	I_{IN} = −18 mA	−1.2	−1.2	−1.2	V
			I_{IN} = 18 mA	V_{CC}+1.2	V_{CC}+1.2	V_{CC}+1.2	V
I_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. V_{OUT} = 0.6/2.4V All Outputs Disabled		±10	±5	±5	µA

Table 24–4. Non-VMEbus Signals

Parameter	Description	Test Conditions		Military	Industrial	Comm.	Units
V_{IH}	Maximum High-Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum Low-Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum High-Level Output Voltage	V_{CC} = Min., I_{OH} = −8 mA		2.4	2.4	2.4	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 8 mA		0.6	0.6	0.6	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., V_{IN} = 0.00/V_{CC}		±5	±5	±5	µA
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	I_{IN} = −18 mA	−1.2	−1.2	−1.2	V
			I_{IN} = 18 mA	V_{CC}+1.2	V_{CC}+1.2	V_{CC}+1.2	V
I_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. GND ≤ V_{OUT} ≤ V_{CC} All Outputs Disabled		±10		±5	µA

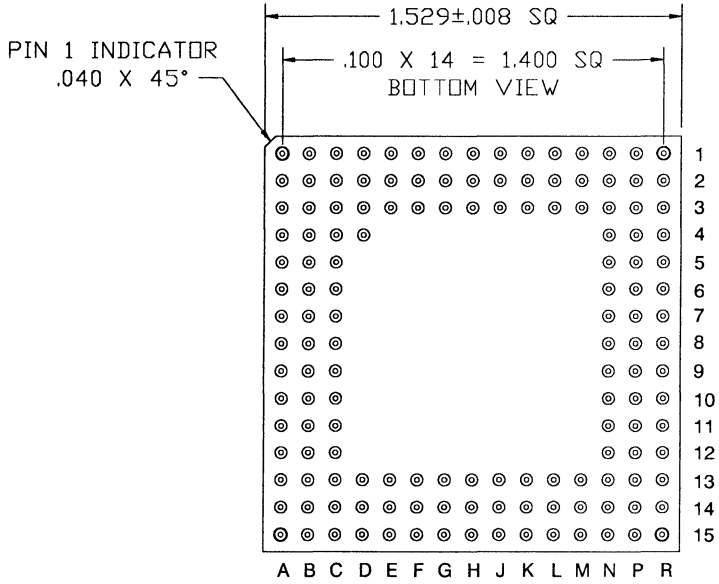
Table 24–5. Capacitance

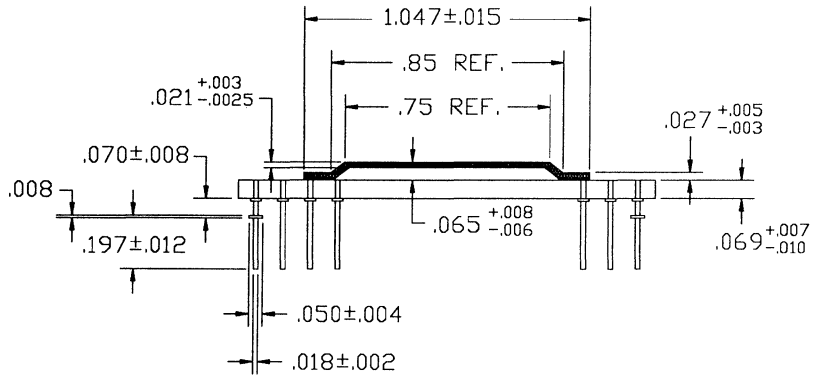
Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 64 MHz, V_{CC} = 5.0V	5	pF
C_{OUT}	Output Capacitance		7	pF

25

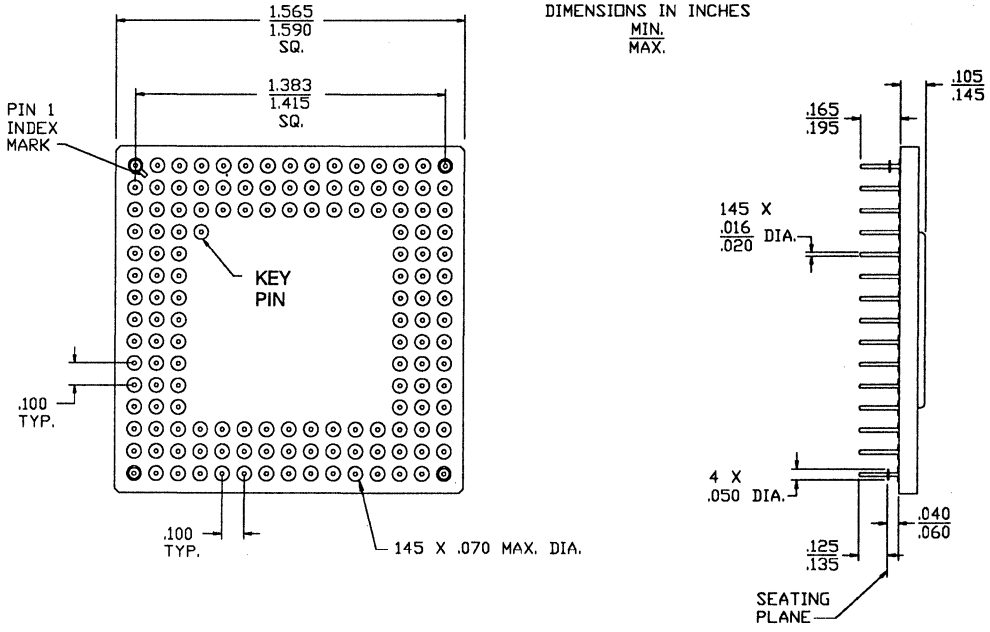
Package Information

145-Pin Plastic Grid Array (Cavity Up) B144

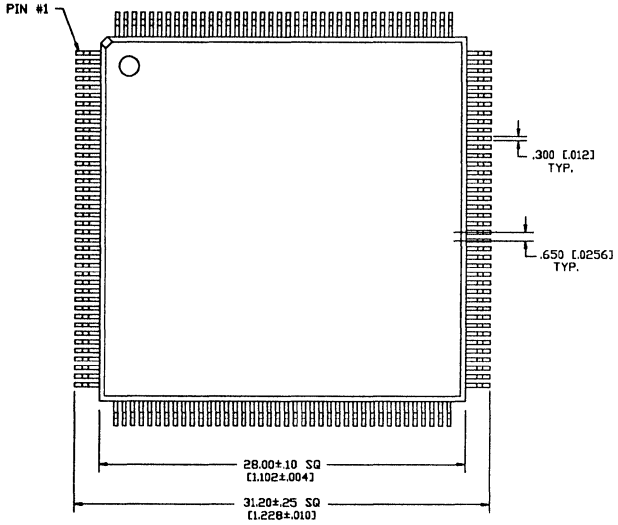


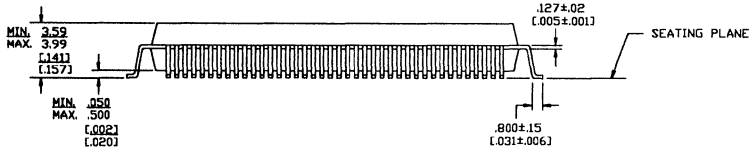


145-Pin Grid Array (Cavity Up) G145

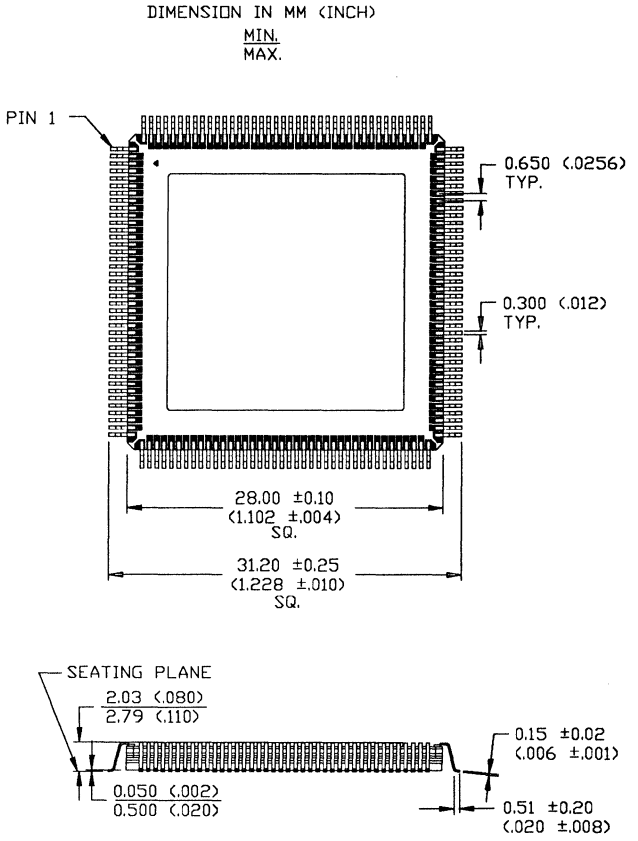


160-Lead Plastic Flatpack N160





160-Lead Ceramic Flatpack U162



Glossary

245 7400-family part type that is an 8-bit (octal) bus transceiver with controls for direction and enabling of drivers.

543 7400-family part type that is an 8-bit (octal) bus transceiver with controls for latch enable, output enable, and direction flow.

68K A Motorola 68000, 68010, 68020, 68030 or 68040 microprocessor.

accelerated mode A mode of operation where the VIC068A will continuously assert the PAS* signal for the duration of a DMA operation. The VIC068A expects the DSACK* signal to be continuously asserted for the duration of the DMA operation as well.

arbitration timeout A timeout that occurs when no module responds to a VMEbus bus grant.

assertion The forcing of a signal to its TRUE state.

base address The starting point of an address region defined by the mask register.

block transfer length The total length, in bytes, of a block transfer. In terms of the VIC068A, a block transfer may or may not contain more then one burst.

block transfer w/DMA A VIC068A block transfer mode in which the VIC068A obtains local bus mastership and performs a VMEbus block transfer utilizing DMA on the local bus.

boundary crossing The crossing of a 256-byte local or VMEbus boundary during a block transfer.

buffer control signals VIC068A signals which control the operation of external address and data latches/buffers.

burst length The length, in VMEbus transfers, of a VMEbus block transfer burst. In terms of the VIC068A, there may or may not be more then one burst per block transfer.

byte An 8-bit unit of data.

clock-tick interrupt An optional, periodic interrupt issued by the VIC068A.

daisy-chain A type of VMEbus signal in which a signal level is propagated from board to board starting from slot 1 and ending with the last occupied slot.

data size The size of a VMEbus data transfer independent of the physical bus size (byte, word, etc.).

deadlock In the context of the VIC068A, this is a condition where the local bus requires the use of the VMEbus and the VMEbus requires the use of the local bus. In this condition, the VIC068A requires the current local bus master remove its bus tenure to let the VMEbus access proceed.

deassertion The forcing of a signal to its FALSE state.

DMA Direct Memory Access. With the VAC068A, this refers to either DMA to the VMEbus or DMA to another interface controller.

DMAAT0 Module-based DMA Transfer Access Timing. The data acquisition timing the VIC068A uses for the first transfer of a module-based DMA transfer. This timing is programmed in bits 3–0 in the SSiCR1.

DMAAT1 Module-based DMA Transfer Access Timing. The data acquisition timing the VIC068A uses for the second and subsequent transfers of a module-based DMA transfer. This timing is programmed in bits 7–4 in the SSiCR1.

DST The local data strobe minimum assertion timing. This timing is programmed in bit 4 of the LBTR.

dual-path A mode of operation which allows a single-cycle master operation to be performed by the VIC068A during interleave periods.

fair requester A VMEbus requester who waits until all requests on its particular VMEbus request level are inactive before requesting the VMEbus.

Force EPROM A VAC068A mode of operation that asserts EPROMCS* after reset.

global switch A local interrupt issued by a VMEbus module to multiple VMEbus slaves.

IMAC Indivisible Multiple Address Cycle.

initiation cycle The local cycle which initiates a VMEbus block transfer with local DMA.

interleave period The period of time between block transfer bursts.

interprocessor communication facilities Various VIC068A register and facilities available by VMEbus accesses.

IPL Interrupt priority level.

ISAC Indivisible Single Address Cycle

local (local side) Resources that connect to the non-VMEbus signals of a VIC068A or VAC068A.

longword (lword) A 32-bit unit of data.

mail box An area of memory reserved for passing messages.

mask The comparison of only selected address bits for the purpose of specifying a range of don't-care conditions.

master block transfer A mode of operation where the VIC068A performs a VMEbus block transfer.

master read The act of transferring data from a VMEbus slave to a VMEbus master.

master write The act of transferring data from a VMEbus master to a VMEbus slave

master write posting A VMEbus master operation where the VIC068A captures outgoing VMEbus write data and acknowledges the local side immediately. This removes the VMEbus access time from local resources.

MBAT0 Master Block Transfer Access Timing. The data acquisition timing the VIC068A uses for the first transfer of a master block transfer with local DMA. This timing is programmed in bits 3–0 in the SSiCR1.

MBAT1 Master Block Transfer Access Timing. The data acquisition timing the VIC068A uses for the second and subsequent transfers of a master block transfer with local DMA. This timing is programmed in bits 7–4 in the SSiCR1.

module A VMEbus circuit card.

module-based DMA A mode of operation where the VIC068A transfers data from one local resource to another utilizing DMA.

module switch A local interrupt issued by a VMEbus module to a single VMEbus slave.

MOVEM block transfer A VIC068A block transfer mode in which the local resource maintains local bus mastership while having the VIC068A perform transfers using block transfer protocol on the VMEbus.

non-accelerated mode A mode of operation where the VIC068A will toggle the PAS* signal for each transfer of a DMA operation. The VIC068A expects the DSACK* signal to toggle for each transfer of the DMA operation as well.

port size The physical size of the VMEbus modules data bus (D8, D16, D32, etc.)

pseudo cycle A block transfer with local DMA, initiation cycle

redirection Re-mapping VMEbus slave select address ranges to a specific local chip select output.

region An area of memory defined by one of three boundary registers in the VAC068A.

rescinding output A three-state output which is first driven High before it is three-stated.

RMC	An indivisible read-modify-write cycle.
SAT	Slave Access Timing. The data acquisition timing the VIC068A uses while performing a slave transfer. This timing is programmed in bits 3-0 in the SSiCR1.
SBAT0	Slave Block Transfer Access Timing. The data acquisition timing the VIC068A uses for the first transfer of a slave block transfer. This timing is programmed in bits 3-0 in the SSiCR1.
SBAT1	Slave Block Transfer Access Timing. The data acquisition timing the VIC068A uses for the second and subsequent transfers of a slave block transfer. This timing is programmed in bits 7-4 in the SSiCR1.
self-access	A condition where the VIC068A, as the VMEbus master, has selected itself as the VMEbus slave.
slave block transfer	A mode of operation where the VIC068A is slave to a VMEbus block transfer.
slave read	The act of transferring data from a VMEbus slave to a VMEbus master.
slave write	The act of transferring data from a VMEbus master to a VMEbus slave
slave write posting	A VMEbus operation where the VIC068A captures incoming VMEbus write data and acknowledges the VMEbus immediately. This removes the local access time from VMEbus resources.
transfer timeout	A timeout that occurs when no module responds with an acknowledge to a data transfer.
turbo	A mode of operation in which the VIC068A reduces certain delays including set-up times.
UART	Universal Asynchronous Receiver Transmitter.
VAC068A	VMEbus Address Controller.
valid slave select	A fully qualified request for slave operations.
VIC068A	VMEbus Interface Controller.
VITA	VMEbus International Trade Association.
VSB	VMEbus Subsystem Bus.
word	A 16-bit unit of data.

Sales Offices

Direct Sales Offices (Domestic)

California

Cypress Semiconductor
Corporate Headquarters
3901 N. First Street
San Jose, CA 95134
(408) 943–2600
Telex: 821032 CYPRESS SNJ UD
TWX: 910 997 0753
FAX: (408) 943–6860

Cypress Semiconductor
23586 Calabasas Rd., Ste. 201
Calabasas, CA 91302
(818) 222–3800
FAX: (818) 222–3810

Cypress Semiconductor
2 Venture Plaza, Suite 460
Irvine, CA 92718
(714) 753–5800
FAX: (714) 753–5808

Cypress Semiconductor
12526 High Bluff Dr., Ste. 300
San Diego, CA 92130
(619) 755–1976
FAX: (619) 755–1969

Alabama

Cypress Semiconductor
303 Williams Ave., Ste. 125
Huntsville, AL 35801
(205) 533–2794
FAX: (205) 533–2796

Colorado

Cypress Semiconductor
4704 Harlan St., Suite 360
Denver, CO 80212
(303) 433–4889
FAX: (303) 433–0398

Florida

Cypress Semiconductor
10014 N. Dale Mabry Hwy. 101
Tampa, FL 33618
(813) 968–1504
FAX: (813) 968–8474

Cypress Semiconductor
255 South Orange Avenue
Suite 1255
Orlando, FL 32801
(407) 422–0734
FAX: (407) 422–1976

Illinois

Cypress Semiconductor
1530 E. Dundee Rd., Ste. 190
Palatine, IL 60067
(708) 934–3144
FAX: (708) 934–7364

Maryland

Cypress Semiconductor
5457 Twin Knolls Rd., Ste. 103
Columbia, MD 21045
(410) 740–2087
FAX: (410) 997–2571

Minnesota

Cypress Semiconductor
14525 Hwy. 7, Ste. 360
Minnetonka, MN 55345
(612) 935–7747
FAX: (612) 935–6982

New Hampshire

Cypress Semiconductor
61 Spit Brook Road, Ste. 110
Nashua, NH 03060
(603) 891–2655
FAX: (603) 891–2676

New York

Cypress Semiconductor
244 Hooker Ave., Ste. B
Poughkeepsie, NY 12603
(914) 485–6375
FAX: (914) 485–7103

Cypress Semiconductor
Hauppauge Exec. Center
300 Vanderbilt Motor Pkwy., #2100
Hauppauge, NY 11788
(516) 231–0238
FAX: (516) 544–4359

North Carolina

Cypress Semiconductor
7500 Six Forks Rd., Suite G
Raleigh, NC 27615
(919) 870–0880
FAX: (919) 870–0881

Oregon

Cypress Semiconductor
12225 SW 2nd Street, Ste. 200
Beaverton, OR 97005
(503) 626–6622
FAX: (503) 626–6688

Pennsylvania

Cypress Semiconductor
Two Neshaminy Interplex, Ste. 206
Trevose, PA 19053
(215) 639–6663
FAX: (215) 639–9024

Texas

Cypress Semiconductor
333 West Campbell Rd., Ste. 240
Richardson, TX 75080
(214) 437–0496
FAX: (214) 644–4839

Cypress Semiconductor
Great Hills Plaza
9600 Great Hills Trail, Ste. 150W
Austin, TX 78759
(512) 338–0204
FAX: (512) 338–0865

Cypress Semiconductor
20405 SH 249, Ste. 216
Houston, TX 77070
(713) 370–0221
FAX: (713) 370–0222

Virginia

Cypress Semiconductor
3151C Anchorway Court
Falls Church, VA 22042
(703) 849–1733
FAX: (703) 849–1734

Sales Representatives (Domestic)

Alabama

Group 2000 Sales Inc.
109C Jefferson St.
Huntsville, AL 35801
(205) 536–2000
FAX: (205) 533–5525

Arizona

Thom Luke Sales, Inc.
9700 North 91st St., Suite A–200
Scottsdale, AZ 85258
(602) 451–5400
FAX: (602) 451–0172

California

TAARCOM
451 N. Shoreline Blvd.
Mountain View, CA 94043
(415) 960–1550
FAX: (415) 960–1999

Canada

bbd Electronics, Inc.
6685–1 Millcreek Dr.
Mississauga, Ontario L5N 5M5
(416) 821–7800
FAX: (416) 821–4541

bbd Electronics, Inc.
298 Lakeshore Rd., Ste. 203
Pointe Claire, Quebec H9S 4L3
(514) 697–0801
FAX: (514) 697–0277

bbd Electronics, Inc. — Ottawa
(613) 564–0014
FAX: (416) 821–4092

bbd Electronics, Inc. — Winnipeg
(204) 942–2977
FAX: (416) 821–4092

Mirika
84 Woodland Dr.
Delta, British Columbia V4C 3C1
(604) 943–5020
FAX: (604) 943–8184

Connecticut

HLM
3 Pembroke Rd.
Danbury, CT 06810
(203) 791–1878
FAX: (203) 791–1876

Florida

CM Marketing
445 Douglas Ave., Suite 2155–20
Altamonte Springs, FL 32714
(407) 682–7709
FAX: (407) 682–7995

CM Marketing
1435–D Gulf to Bay Blvd.
Clearwater, FL 34615
(813) 443–6390
FAX: (813) 443–6312

CM Marketing
3108 NE 26th St.
Ft. Lauderdale, FL 33305
(305) 566–6386
FAX: (305) 537–4725

Georgia

Group 2000 Sales Inc.
5390 Peachtree Industrial Blvd.
Suite 210B
Norcross, GA 30071
(404) 729–1889
FAX: (404) 729–1896

Illinois

Micro Sales Inc.
901 W. Hawthorn Drive
Itasca, IL 60143
(708) 285–1000
FAX: (708) 285–1008

Indiana

Technology Mktg. Corp.
599 Industrial Dr.
Carmel, IN 46032
(317) 844–8462
FAX: (317) 573–5472

Technology Mktg. Corp.
4630–10 W. Jefferson Blvd.
Ft. Wayne, IN 46804
(219) 432–5553
FAX: (219) 432–5555

Technology Marketing Corp.
1214 Appletree Lane
Kokomo, IN 46902
(317) 459–5152
FAX: (317) 457–3822

Iowa

Midwest Technical Sales
463 Northland Ave., N.E.
Suite 101
Cedar Rapids, IA 52402
(319) 377–1688
FAX: (319) 377–2029

Kansas

Midwest Technical Sales
21901 La Vista
Goddard, KS 67052
(316) 794–8565

Midwest Technical Sales
15301 W. 87 Parkway, Ste. 200
Lenexa, KS 66219
(913) 888–5100
FAX: (913) 888–1103

Kentucky

Technology Marketing Corp.
4012 DuPont Circle, Ste. 414
Louisville, KY 40207
(502) 893–1377
FAX: (502) 896–6679

Michigan

Techrep
2200 North Canton Center Rd.
Suite 110
Canton, MI 48187
(313) 981–1950
FAX: (313) 981–2006

Missouri

Midwest Technical Sales
514 Earth City Expwy., #239
Earth City, MO 63045
(314) 298–8787
FAX: (314) 298–9843

New Jersey

HLM
333 Littleton Rd.
Parsippany, NJ 07054
(201) 263–1535
FAX: (201) 263–0914

New Mexico

Techni-Source, Inc.
1101 Cardenas NE #103
Albuquerque, NM 87110
(505) 268–4232
FAX: (505) 268–0451

New York

HLM
P.O Box 328
Northport, NY 11768
(516) 757–1606
FAX: (516) 757–1636

Reagan/Compar
37A Brook Hill Lane
Rochester, NY 14625
(716) 271–2230
FAX: (716) 381–2840

Reagan/Compar
214 Dorchester Ave., #3C
Syracuse, NY 13203
(315) 432–8232
FAX: (315) 432–8238

Reagan/Compar
3301 Country Club Road
Ste. 2211
P.O. Box 135
Endwell, NY 13760
(607) 754–2171
FAX: (607) 754–4270

Ohio

KW Electronic Sales, Inc.
8514 North Main Street
Dayton, OH 45415
(513) 890–2150
TWX: 510 601 2994
FAX: (513) 890–5408

KW Electronic Sales, Inc.
3645 Warrensville Center Rd. #244
Shaker Heights, OH 44122
(216) 491–9177
TWX: 62926868
FAX: (216) 491–9102

Pennsylvania

L. D. Lowery
2801 West Chester Pike
Broomall, PA 19008
(215) 356–5300
FAX: (215) 356–8710

KW Electronic Sales, Inc.
4068 Mt. Royal Blvd., Ste. 110
Allison Park, PA 15101
(412) 492–0777
FAX: (412) 492–0780

Puerto Rico

Electronic Technical Sales
P.O. Box 10758
Caparra Heights Station
San Juan, P.R. 00922
(809) 798–1300
FAX: (809) 798–3661

Utah

Sierra Technical Sales
4700 South 900 East, 30–150
Salt Lake City, UT 84117
(801) 566–9719
FAX: (801) 565–1150

Washington

Electronic Sources
1603 116th Ave. NE, Ste. 115
Bellevue, WA 98004
(206) 451–3500
FAX: (206) 451–1038

Wisconsin

Micro Sales Inc.
210 Regency Court
Suite L101
Waukesha, WI 53186
(414) 786–1403
FAX: (414) 786–1813

Direct Sales Offices (International)

Cypress Semiconductor International—Europe

Avenue Ernest Solvay, 7
B–1310 La Hulpe, Belgium
Tel: (32) 2–652–0270
Telex: 64677 CYPINT B
FAX: (32) 2–652–1504

France

Cypress Semiconductor France
Miniparc Bât. no 8
Avenue des Andes, 6
Z.A. de Courtaboeuf
91952 Les Ulis Cedex, France
Tel: (33) 1–69–07–55–46
FAX: (33) 1–69–07–55–71

Germany

Cypress Semiconductor GmbH
Munchner Str. 15A
W-8011, Zorneding, Germany
Tel: (49) 8106–2855
FAX: (49) 8106–20087

Cypress Semiconductor GmbH
Büro Nord
Matthias-Claudius-Str. 17
W–2359 Henstedt-Ulzburg, Germany
Tel: (49) 4193-77217
FAX: (49) 4193-78259

Italy

Cypress Semiconductor
Via del Poggio Laurentino 118
00144 Rome, Italy
Tel: (39) 65–920–723
FAX: (39) 65–920–924

Cypress Semiconductor
Via Quintino 28
10121 Torino, Italy
Tel: (39) 11–562–55–22
FAX: (39) 11–562–86–12

Japan

Cypress Semiconductor Japan K.K.
Fuchu-Minami Bldg., 2F
10-3, 1-Chome, Fuchu-machi,
Fuchu-shi, Tokyo, Japan 183
Tel: (81) 423–69–82–11
FAX: (81) 423–69–82–10

Sweden

Cypress Semiconductor Scandinavia AB
Marknadsvagen 15
S–18311 Taby, Sweden
Taby Centrum, Ingang S
Tel: (46) 8 638 0100
FAX: (46) 8 792 1560

United Kingdom

Cypress Semiconductor U.K., Ltd.
3, Blackhorse Lane, Hitchin,
Hertfordshire, U.K., SG4 9EE
Tel: (44) 462–42–05–66
FAX: (44) 462–42–19–69

Cypress Semiconductor Manchester
27 Saville Rd. Cheadle
Gatley, Cheshire, U.K.
Tel: (44) 614–28–22–08
FAX: (44) 614–28–0746

Sales Representatives (International)

Australia

Braemac Pty. Ltd.
Unit 6, 111 Moore St.
Leichhardt, N.S.W. 2040, Australia
Tel: (61) 2–564–1211
FAX: (61) 2–564–2789

Braemac Pty. Ltd.
10–12 Prospect Street, Box Hill
Melbourne, Victoria, 3128, Australia
Tel: (61) 3–899–1272
FAX: (61) 3–899–1276

Austria

Hitronik Vertriebsge GmbH
St. Veitgasse 51
A–1130 Wien, Austria
Tel: (43) 222–824–199
Telex: 133404 HIT A
FAX: (43) 222–828–55–72

Belgium

Sonetech
Limburg Stirum 243
1780 Wemmel, Belgium
Tel: (32) 2–460–0707
FAX: (32) 2–460–1200

Denmark

Nortec Elektronik A/S
Transformervej 17
DK–2730 Herlev, Denmark
Tel: (45) 42–84–20–00
Telex: 35200 NORDEL DK
FAX: (45) 44–92–15–52

France

Arrow Electronics
73/79, Rue des Solets
Silic 585
94653 Rungis Cedex
Tel: (33) 1 49 78 49 00
FAX: (33) 1 49 78 05 99

Arrow Electronics
Les Jardins d'Entreprises
Betiment B3
213, Rue Gerland
69007 Lyon
Tel: (33) 78 72 79 42
FAX: (33) 78 72 80 24

Arrow Electronics
Centreda
Avenue Didier Daurat
31700 Blagnac
Tel: (33) 61 15 75 18
FAX: (33) 61 30 01 93

Arrow Electronics
Immeuble St. Christophe
Rue de la Frebardiere
Zi Sud Est
35135 Chantepie
Tel: (33) 99 41 70 44
FAX: (33) 99 50 11 28

Newtek
Rue de L'Esterel, 8, Silic 583
F–94663 Rungis Cedex, France
Tel: (33) 1–46–87–22–00
Telex: 263046 F
FAX: (33) 1–46–87–80–49

Newtek
Rue de l'Europe, 4
Zac Font–Ratel
38640 Claix, France
Tel: (33) 16–76–98–56–01
FAX: (33) 16–76–98–16–04

Scaib, SA
80 Rue d'Arcueil Silic 137
9 4523 Rungis, Cedex, France
Tel: (33) 1–46–87–23–13
FAX: (33) 1–45–60–55–49

Germany

API Electronik GmbH
Lorenz–Brarenstr 32
W–8062 Markt, Indersdorf
Germany
Tel: (49) 8136 7092
Telex: 527 0505
FAX: (49) 8136 7398

Astek GmbH
Gottlieb–Daimler Str. 7
W–2358 Kaltenkirchen
Germany
Tel: (49) 41 91–80 07–0
Telex: 2180120 ASK D
FAX: (49) 41 91–80 07–33

Metronik GmbH
Leonhardsweg 2, Postfach 1328
W–8025 Unterhaching,
Germany
Tel: (49) 89 611080
Telex: 17 897434 METRO D
FAX: (49) 89 6116468

Metronik GmbH
Laufamholzstrasse 118
W–8500 Nürnberg,
Germany
Tel: (49) 911 544966
Telex: 6 26 205
FAX: (49) 911 542936

Metronik GbmH
Löwenstrasse 37
W–7000 Stuttgart 70
Germany
Tel: (49) 711 764033
Telex: 7–255–228
FAX: (49) 711 7655181

Metronik GmbH
Siemensstrasse 4–6
W–6805 Heddesheim, Germany
Tel: (49) 6203 4701
Telex: 465 035
FAX: (49) 6203 45543

Metronik GmbH
Zum Lonnenhohl 38
W–4600 Dortmund 13, Germany
Tel: (49) 231 217041
FAX: (49) 231 210799

Metronik GmbH
Buckhorner Moor 81
W–2000 Norderstedt, Germany
Tel: (49) 40 5228091
Telex: 2162488
FAX: (49) 40–522 80 93

Metronik Halle
Thalmannplatz 16/0904
O–4020 Halle, Germany

Spoerle Electonic
Kackertstrasse 10
W–5100 Aachen 1, Germany
Tel: (49) 241 / 81032
FAX: (49) 241 / 81162

Spoerle Electronic
Rudower Strasse 27–29
W–1000 Berlin 47, Germany
Tel: (49) 30 / 60 60 11
Telex: 186 029
FAX: (49) 30 / 6 01 40 57

Spoerle Electronic
Hildebrandstrasse 11
W–4600 Dortmund 13, Germany
Tel: (49) 231 / 21801–0
Telex: 822 555
FAX: (49) 231 / 2180167

Spoerle Electronic
IM Gefierth 11
W–6072 Dreieich Bei
Frankfurt, Germany
Tel: (49) 6103 / 3040
FAX: (49) 6103 / 304201

Spoerle Electronic
Hans–Bunte–Strasse 2
W–7800 Freiburg i.Br., Germany
Tel: (49) 7 61 / 5 10 45–0
Telex: 7 721 994
FAX: (49) 7 61 / 50 22 33

Spoerle Electronic
Rodeweg 18
W–3400 Gottingen, Germany
Tel: (49) 5 51 / 9 04–0
Telex: 96 733
FAX: (49) 5 51 / 9 04 46/48

Spoerle Electronic
Winsbergring 42
W–2000 Hamburg 54, Germany
Tel: (49) 40 / 85 31 34–0
Telex: 2 164 536
FAX: (49) 40 / 85 31 34 91

Spoerle Electronic
Roscher Str. 31
0–7010 Leipzig, Germany
Tel: (37) (9) 41 / 592322
or (37) (9) 41 / 591732
FAX: (37) (9) 41 / 52087

Spoerle Electronic
Fohringer Allee 17
W–8043 Unterfohring, Germany
Tel: (49) 89 / 9 50 99–0
Telex: 5 216 379
FAX: (49) 89 / 9 50 99 99

Spoerle Electronic
Rathsberg Str. 17
W–8500 Nurnberg 10, Germany
Tel: (49) 9 11 / 5 21 56–0
Telex: 622 996
FAX: (49) 9 11 / 5 21 56 35

Spoerle Electronic
Hopfigheimer Strasse 5
W–7120 Bietigheim–Bissingen, Germany
Tel: (49) 71 42 / 70 03–0
Telex: 724 287
FAX: (49) 71 42 / 70 03 60

Hong Kong

Tekcomp Electronics, Ltd.
514 Bank Centre
636, Nathan Road
Kowloon, Hong Kong
Tel: (852) 3–880–629
Telex: 38513 TEKHL
FAX: (852) 7–805-871

India

Spectra Innovations Inc.
Manipal Centre, Unit No. S–822
47, Dickenson Rd.
Bangalore–560,042
Tel: 812–566 630 x3808
Telex: 845 2696 or 8055
(Attn: ICTP–705)
FAX: 812–261 468 (IC FAX 217)

Israel

Talviton Electronics
P.O. Box 21104, 9 Biltmore Street
Tel Aviv 61 210, Israel
Tel: (972) 3–544–2430
Telex: 33400 VITKO
FAX: (972) 3–544–2085

Italy

Dott. Ing. Guiseppe De Mico s.p.a.
V. Le Vittorio Veneto, 8
I–20060 Cassina d'Pechi
Milano, Italy
Tel: (39) 29–53––43–600
Telex: 330869 DEMICO I
FAX: (39) 29–52–22–27

Silverstar Ltd. SPA
Viale Fulvio Testi, 280
20126 Milano, Italy
Tel: (39) 2 661251
Teles: 33 2189 SIL 7I
FAX: (39) 2 66101359

Japan

Tomen Electronics Corp.
2-1-1 Uchisaiwai-Cho, Chiyoda-Ku
Tokyo, 100 Japan
Tel: (81) 3–3506–3673
Telex: 23548 TMELCA
FAX: (81) 3–3506–3497

CTC Components Systems Co. Ltd.
4–8–1, Tsuchihashi,
Miyamae–Ku, Kawasaki–Shi,
Kanagawa, 213 Japan
Tel: (81) 44–852–5121
Telex: 3842272 CTCEC J
FAX: (81) 44–877–4268

Fuji Electronics Co., Ltd.
Ochanomizu Center Bldg.
3–2–12 Hongo, Bunkyo–Ku
Tokyo, 113 Japan
Tel: (81) 3–3814–1411
Telex: J28603 FUJITRON
FAX: (81) 3–3814–1414

N.D.A. Co. Ltd.
Cuctus Iidabashi Bldg.
4–8–3 Iidabashi Chiyoda–Ku
Tokyo, 102 Japan
Tel: (81) 3–3264–1321
Telex: J29503 ISI JAPAN
FAX: (81) 3–3264–3419

Fujitsu Devices, Inc.
Osaki West Bldg.
8–8, Osaki 2–Chome, Shinagawa–ku
Tokyo 141, Japan
Tel: (81) 3–3490–3321
FAX: (81) 3–3490–7274

Korea

Hanaro Corporation
Hana Bldg.
122–30 Chung Dam Dong
Kangnam–ku
Seoul, Korea
Tel: (82) 2–516–1144
FAX: (82) 2–516–1151

Netherlands

Semicon B.V.
Gulberg 33, NL–5674
Te Nuenen
The Netherlands
Tel: (31) 40–83–70–75
Telex: 59418 INTRA NL
FAX: (31) 40–83–86–35

Norway

Nortec Electronics A/S
Smedsvingen 4, P.O. Box 123
1364 Hvalstad, Norway
Tel: (47) 2–84–62–10
FAX: (47) 2–84–65–45

Singapore

Serial Systems Marketing
21 Moonstone Lane
Pohleng Building #0201
Singapore 1232
Tel: (65) 29–38–830
FAX: (65) 29–12–673

Spain

Comelta s.a.
Emilio Munoz, 41 Nave 1–1–2
28037 Madrid, Spain
Tel: (34) 1–327–0614
Telex: 42007 CETA–E
FAX: (34) 1–327–0540

Comelta s.a.
Pedro IV, 8–4–5 Planta
08005 Barcelona, Spain
Tel: (34) 3–007–7712

Sweden

TH:s Elektronik AB
P.O. Box 3027
Arrendevägen 36
S163 03 SPÅNGA, Sweden
Tel: (46) 8 362 970
Telex: 111 45 tenik s
FAX: (46) 8 761 3065

Switzerland

Basix für Elektronik A. G.
Hardturmstrasse 181
CH–8010 Zurich, Switzerland
Tel: (41) 1–276–11–11
Telex: 822762 BAEZ CH
FAX: (41) 1–276–12–34

Taiwan R.O.C.

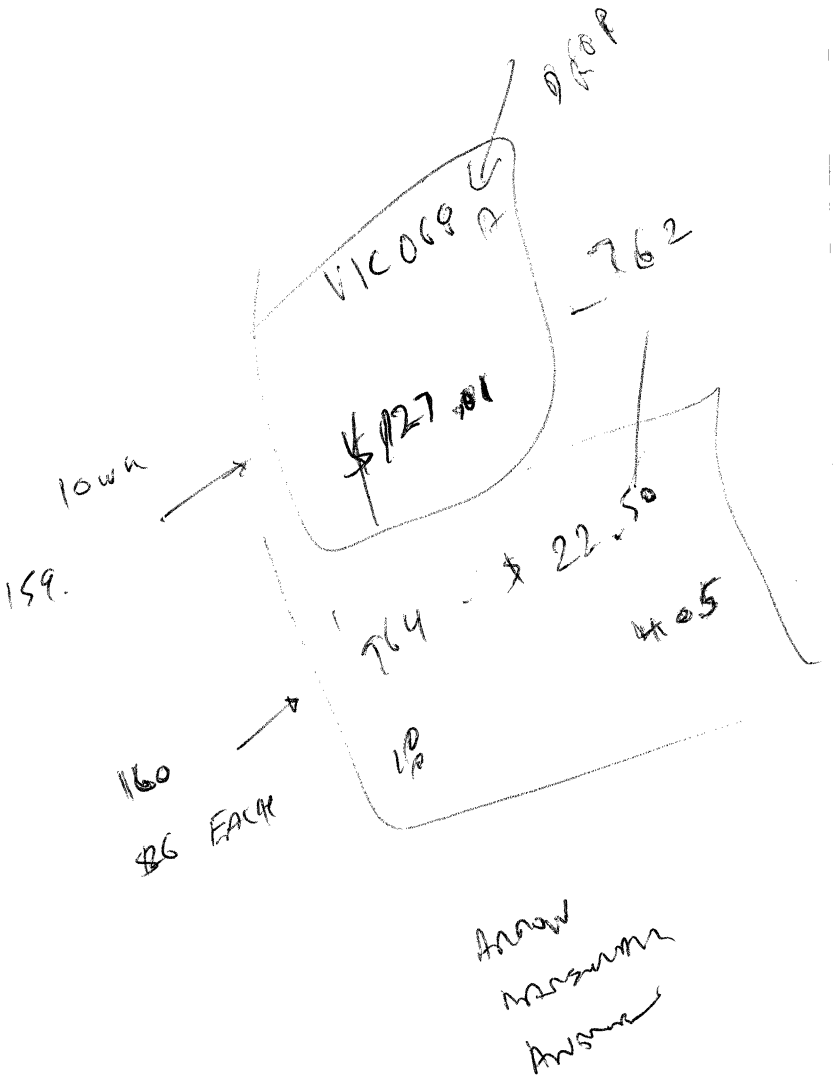
Prospect Technology Corp.
5F, No. 348, Pai-Ling 5th Road
Taipei, Taiwan
Tel: (886) 2–820–5353
FAX: (886) 2–820–5731

United Kingdom

Ambar Components Ltd.
17 Thame Park Road
Thame, Oxfordshire
England, OX9 3XD
Tel: (44) 844–26–11–44
Telex: 837427
FAX: (44) 844–26–17–89

Arrow Electronics (UK) Ltd.
St. martins Business Centre
Cambridge Road
Bedford MK42 OLF
Tel: (44) 234 270272
FAX: (44) 234 214674

Pronto Electronic System Ltd.
City Gate House
Eastern Avenue, 399–425
Gants Hill, Ilford,
Essex, U. K. IG2 6LR
Tel: (44) 81–554–62–22
Telex: 8954213 PRONTO G
FAX: (44) 81–518–32–22


VIC069
$127.01
762
10 un
159.
764 - $ 22.50
405
160
$86 EACH
1/2

Index

Page numbers listed in italics in this index indicate that the entry refers to a figure. Please keep in mind while using this index that Chapters 1 through 16 discuss the VIC068A, Chapters 17 through 23 discuss the VAC068A, and Chapters 24 and 25 discuss both.

CYPRESS
SEMICONDUCTOR

3901 North First Street, San Jose, CA 95134 (408) 943-2600