VMEBOOK

Table of Contents

Introduction

Thank you for your interest in Cypress's line of VMEbus Interface Products! Cypress provides a wide range of solutions to help you design almost any VMEbus interface. This Handbook explains the use of each product individually. Diagrams and examples are shown where needed to help clarify the operation of each part. This book is broken into five sections as follows:

Section 1: The VIC068A VMEbus Interface Controller

Section 2: The VIC64 VMEbus Interface Controller

Section 3: The CY7C960/961 Slave VMEbus Interface Controllers

Section 4: The CY7C964 Bus Interface Logic Circuit

Section 5: The VAC068A VMEbus Address Controller

We also offer the *Cypress Applications Handbook*, which contains design examples using our VMEbus products. Although these examples may not show the exact solution you need, they can be used as building blocks to create an interface that fits your design.

Cypress also manufactures high speed SRAMs, Programmable Logic Devices (PLDs), Clock devices, and many Datacom devices as an aid for your design. Call (800)858-1810 to obtain a copy of one of our data books or the *Cypress Applications Handbook*.

For further help using any Cypress device, to download datasheets or application notes, or for general information about Cypress Semiconductor, check out our web page at www.cypress.com. Datasheets or applications notes can be sent directly to your fax machine by calling (800)213-5120. For direct technical assistance call (408)943-2821 to reach our applications hotline or email us at cyapps@cypress.com.

How to Use This Book

This guide provides the hardware and software designer with detailed information on the Cypress Semiconductor VMEbus Interface Products. It may also be used to provide detailed information regarding existing off-the-shelf VMEbus modules that utilize the Cypress line of interface products.

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 (ANSI/VITA-1-1994).

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.
- 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 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.
- The terms *local* or *local side* refer to CPU, memory, or other resources that connect to the non-VMEbus signals of the VMEbus interface device.
- 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).
- The letter “T” is used to indicate the 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 1.11.4.
- The letters “L” and “H” are used to indicate a High or Low value driven *by* the VMEbus interface device. The numbers “1” and “0” are used to indicate a High or Low value driven *to* the VMEbus interface device.

Section 1

The VIC068A VMEbus
Interface Controller

1.1 Introduction to the VIC068A

1.1.1 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 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.*

1.1.2 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

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.

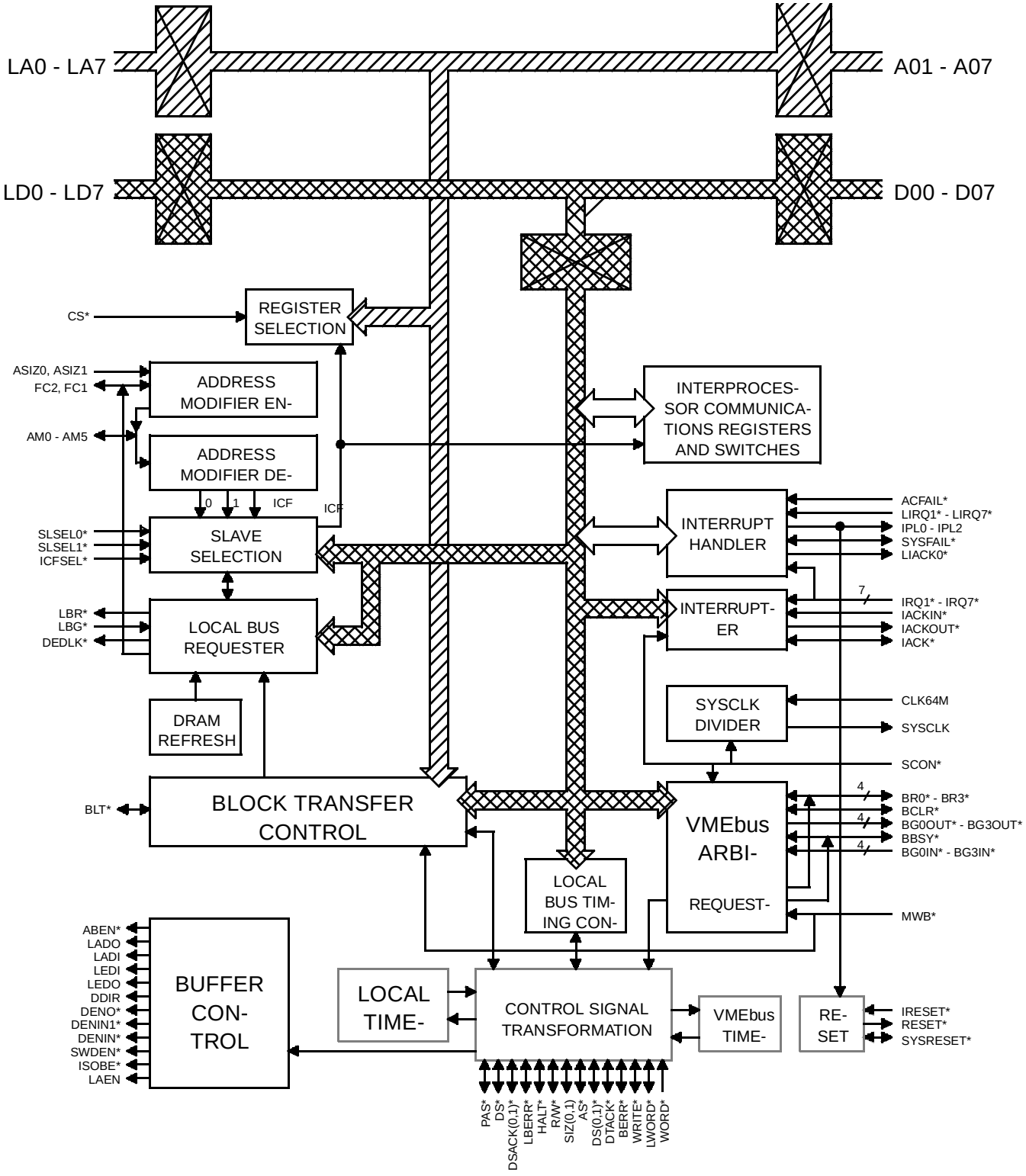


Figure 1-1. VIC068A Block Diagram

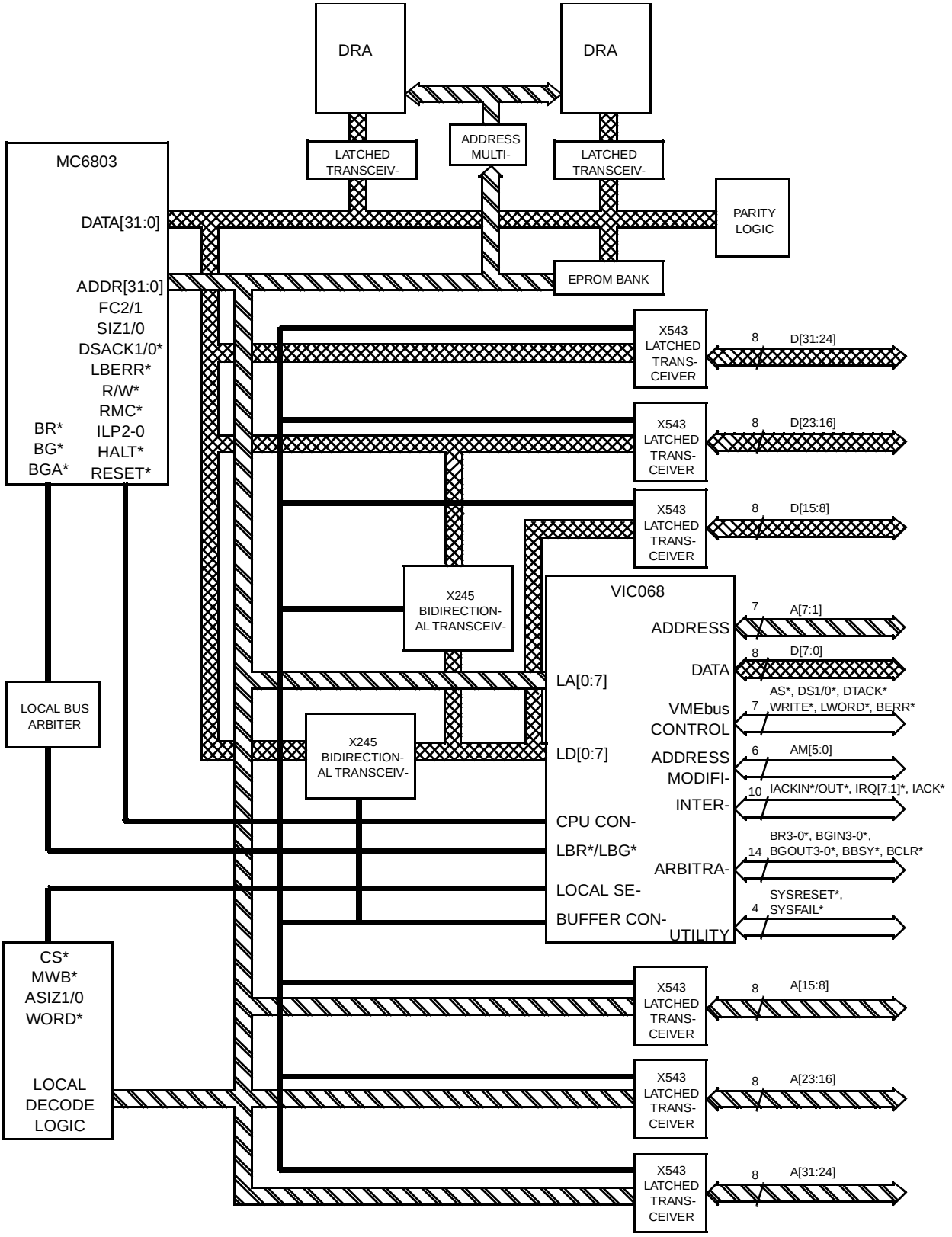


Figure 1-2. VIC068A on 68030 Board

1.2 VIC068A Signal Descriptions

1.2.1 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: 48 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. See section 1.11.1.

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), not the VIC068A. The VIC068A can be enabled to provide a local interrupt when this signal is asserted. See section 1.9.5.

SYSFAIL*

Input: Yes
Output: Yes, open collector
Drive: 48 mA

As an output, the SYSFAIL* signal is asserted when it detects that HALT* has been asserted for more than 6 μs by a source other than the VIC068A.

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. See section 1.9.5.

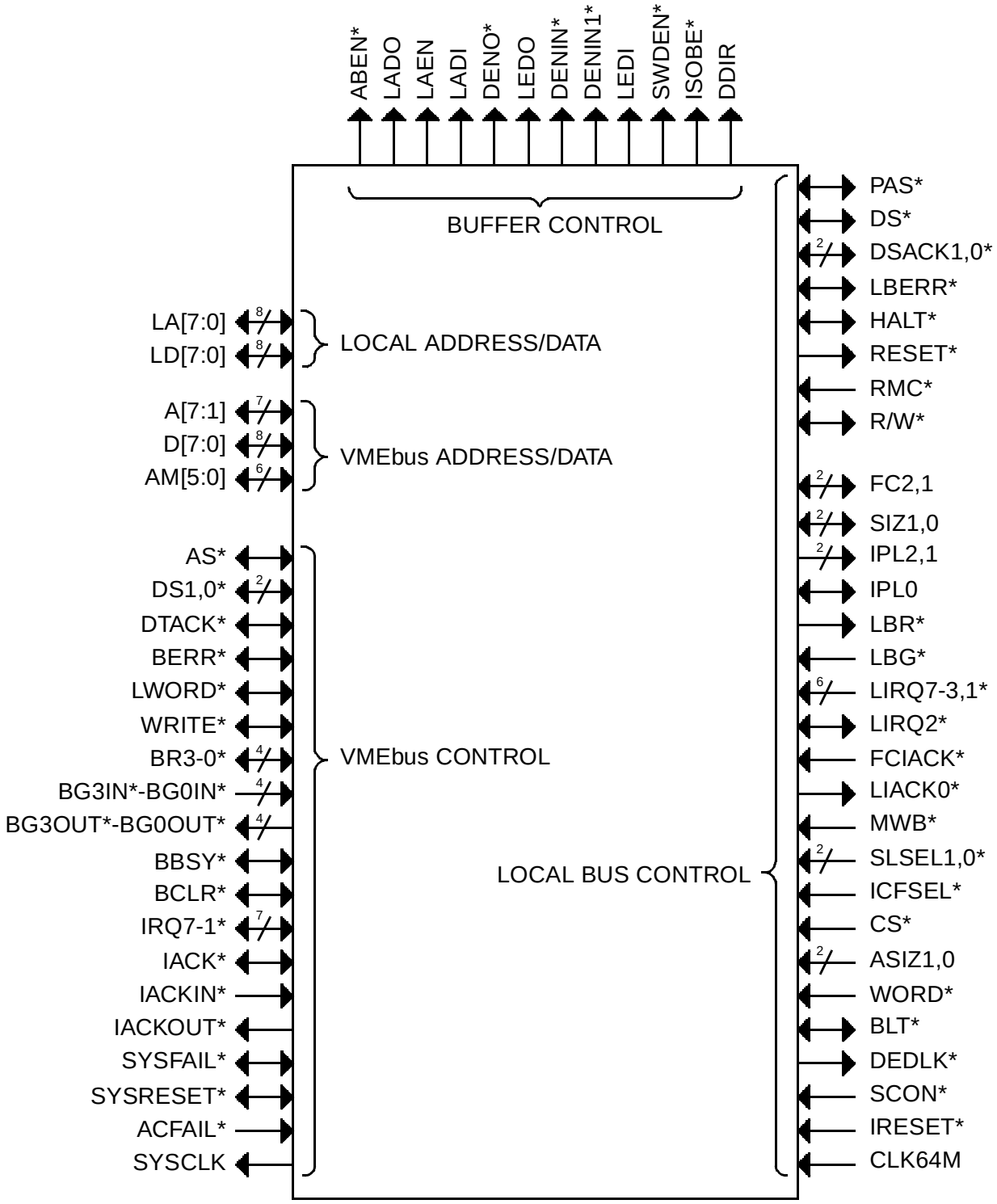


Figure 1-3. VIC068A Signal Diagram

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: 48 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: 48 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: 48 mA

These are the VMEbus low-order data lines.

A7–A1

Input: Yes
Output: Yes, three-state
Drive 48 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: 48 mA

This is the VMEbus Data-Transfer-Acknowledge signal.

BERR*

Input: Yes
Output: Yes, rescinding
Drive: 48 mA

This is the VMEbus Bus-Error signal.

WRITE*

Input: Yes
Output: Yes, rescinding
Drive: 48 mA

This is the VMEbus Data-Direction signal.

LWORD*

Input: Yes
Output: Yes, rescinding
Drive: 48 mA

This is the VMEbus Longword signal.

AM5–AM0

Input: Yes
Output: Yes, three-state
Drive: 48 mA

These are the VMEbus Address-Modifier signals.

IACK*

Input: Yes
Output: Yes, rescinding
Drive: 48 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* - IRQ1*

Input: Yes
Output: Yes, open collector
Drive: 48 mA

These are the VMEbus Interrupt request signals.

1.2.2 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 (active High) 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. See section 1.7.2.

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 (LBTR).

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 (LBTR).

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. See section 1.5.3.
- 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 DSACK*s 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).

During deadlocks, 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. See section 1.5.7.

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 reset 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 reset condition is not present when the reset timer period ends. Once the VIC068A stops driving RESET* Low, this pin is three-stated. Since the VIC068A does not actively drive RESET* to its inactive state, a pull-up resistor should be used on this signal to ensure that any device monitoring the RESET* signal will see its removal. See section 12.1.

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. See section 1.5.7.

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, the VMEbus signal 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. See section 1.6.3.

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 VMEbus 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. See sections 1.5.9, 1.6.5, and 1.6.8.

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 must 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 DS*) to acknowledge all VIC068A-generated local interrupts. See Chapter 1.9.

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. See section 1.6.1.

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). See Chapter 1.8.

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

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 cross-

ings. BLT* toggles during local boundary crossings to increment the external LA[+:8] counters. See section 1.10.1.1.

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 enabled. 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. See section 1.5.7.

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. See Chapter 1.9.

During the assertion of IRESET*, IPL0 becomes an input. If IPL0 is asserted at this time, a global reset is performed. See section 1.11.1.2.

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. See section 1.9.3.

LIRQ2* may also be configured to issue periodic “heartbeat” interrupts at user-defined intervals. See section 1.9.6.

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. See section 1.9.3.

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*. See Chapter 1.12.

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, otherwise this signal should be tied High. See Chapter 1.4.

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.

RMC*

Input: Yes
Output: No
Drive: None

This is the Read-Modify-Write control signal. This signal may be used to control indivisible cycles on the VMEbus. Its operation is controlled with the Interface Configuration register, bits 5–7. See section 1.5.6.

1.2.3 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. These signals can be directly connected to Cypress CY7C964s which simplifies the VME interface by replacing 8 bit wide external latches, buffers and counters with one CY7C964. A complete 32 bit wide VME interface would consist of the VIC068A and three CY7C964s. See Section 4, The CY7C964 Bus Interface Logic Circuit, for more information.

For simple VME designs (i.e. single-cycle only) the VIC068A can directly drive the control lines of discrete buffers and latches (*Figure 1-4*).

Figure 1-4 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.

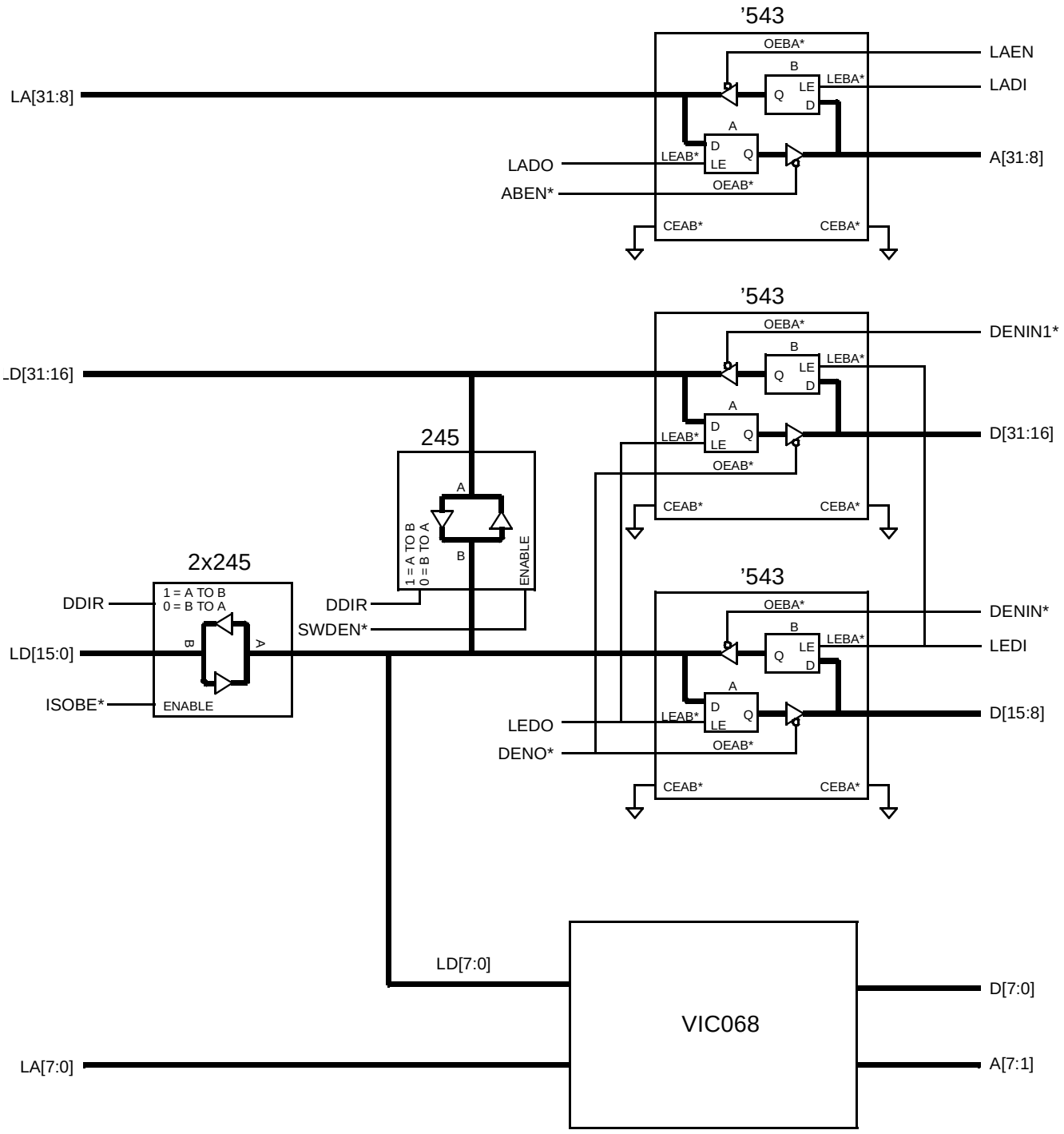


Figure 1-4. VIC068A Control Signals for Shared Memory Implementation

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 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.

DENIN* (formerly 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.

DENIN1* (formerly 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.

1.3 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.

1.3.1 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 VMEbus 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. See section 1.11.1.

1.3.2 The VIC068A VMEbus System Controller

The VIC068A is capable of operating as the VMEbus system controller. It provides VMEbus 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. See Chapter 1.4.

1.3.3 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 (see section 1.5.1). The following release modes are supported (see section 1.5.2):

- 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.

1.3.3.1 Master Write-Posting

The VIC068A is capable of performing master write-posting (bus-decoupling) during both block and single-cycle transactions. 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 the VMEbus. See section 1.5.5.

1.3.3.2 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

See section 1.5.6.

1.3.3.3 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. See section 1.5.7.

1.3.3.4 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.

1.3.4 VIC068A VMEbus Slave Cycles

The VIC068A is capable of receiving slave accesses (see Chapter 1.6). 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.

1.3.4.1 Slave Write-Posting

The VIC068A is capable of performing a slave write-post operation (bus-decoupling) during single cycle transactions. 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. See section 1.6.7.

1.3.5 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, ASIZi 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. See section 1.6.1.

1.3.6 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

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. See section 1.10.1.1.

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, Cypress CY7C964s, 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. See section 1.10.1.1.6.

The Cypress Semiconductor CY7C964s or 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. Three CY7C964s extend the 8-bit counters in the VIC068A to support full 32-bit incrementing addresses on both the local bus and VMEbus. The CY7C964s also contain the latches required for extended address block transfers as well as those required for supporting the dual-path option. The CY7C964 enhances boards that support block transfers by greatly reducing the necessary support logic.

The Cypress Semiconductor VAC068A may also be used to provide the latching and counting of upper data and addresses also reducing necessary support logic.

1.3.6.1 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. See section 1.10.1.2.

1.3.6.2 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. See section 1.10.1.1.

1.3.6.3 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.

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.

1.3.7 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).

1.3.8 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.

1.4 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
- driving SYSRESET* output
- driving BCLR*
- VMEbus arbitration timeout timer

The following VIC068A registers are used as the system controller:

- Transfer Timeout Register (TTR), bits 5–7
- Arbiter/Requestor Control Register (ARCR), bit 7
- Error Group Interrupt Control Register (EGICR), bit 5

1.4.1 VMEbus Arbitration

The arbitration scheme is programmed by writing ARCR[7]. In PRI (priority) mode, BR3* has the highest priority and BR0* has the lowest. Higher priority bus requests will be handled before lower priority bus requests when in PRI mode. In the RRS (round robin) 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 the same level.

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.

1.4.2 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 ms 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. See section 1.9.5.

1.4.3 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.

1.4.4 The BGi Daisy-Chain Driver

The VIC068A, as system controller, drives the BGiOUT* daisy-chain in response to VME-bus 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).

1.4.5 The IACK* Daisy-Chain Driver

The VIC068A, as system controller, is the first device to drive the IACK* daisy chain (*Figure 1-5*). When the VIC068A is performing duties as the system controller, the IACK* input is internally tied to the IACKIN* input. When a VMEbus interrupt handler drives IACK* Low

on the VMEbus, the system controller VIC068A will see this as a Low on its IACK* input and will react just like a VIC068A located elsewhere on the VMEbus would when its IACKIN* is driven Low. See section 1.9.2.

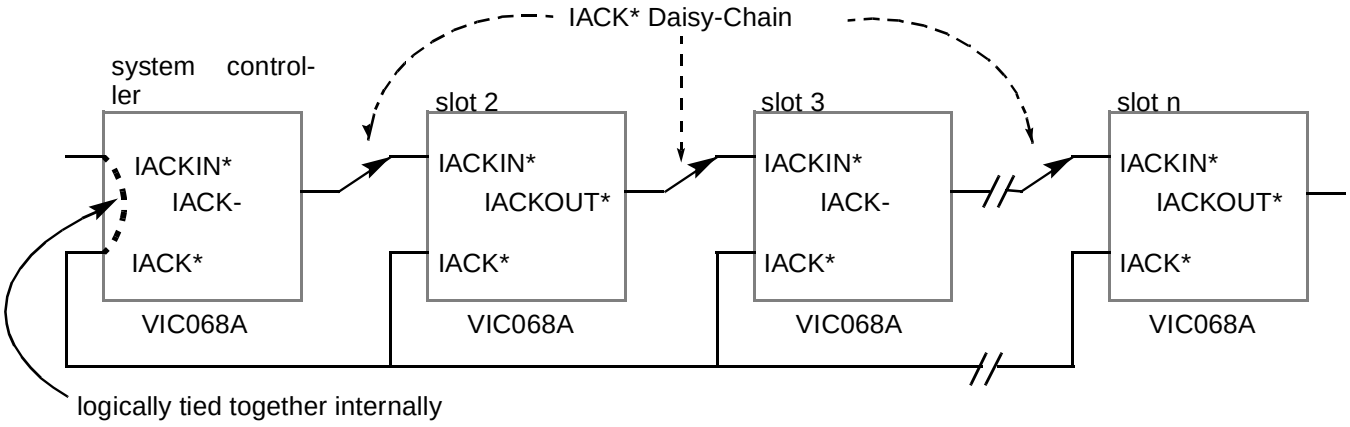


Figure 1-5. IACK* Daisy-Chain

1.5

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 1.12 for descriptions of these registers.

1.5.1 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*.

1.5.2 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].

1.5.2.1 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.

1.5.2.2 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

1.5.2.3 Release On Clear (ROC)

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

1.5.2.4 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.

1.5.2.5 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.

1.5.3 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 1-1 through *1-4* show the buffer control signals for various master cycles.

Table 1-1. 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	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL HL LL HL	L L H H	L L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	H H H H	L L L L	H H H H	H H H H
Three-Byte	1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LL HL	L L H H	L L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	H H H H	L L L L	H H H H	H H H H
Word	1 1 1 1	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL LL LL HL	L H H H	H L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	L H H H	H L L L	H H H H	H H H H
Byte	1 1 1 1	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	L L H H	H H L L	H H H H	H H H H

Table 1-2. 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	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL HL LL HL	L L H H	L L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	H H L L	L L L L	L L L L	L L H H
Three-Byte	1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LL HL	L L H H	L L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	H H L L	L L L L	L L L L	L L H H
Word	1 1 1 1	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL LL LL HL	L H H H	H L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L H L L	L L L L	L L L L	H L H H
Byte	1 1 1 1	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	L L L L	L L L L	H H H H

Table 1-3. 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	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL HL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Three-Byte	0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Word	0 0 0 0	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL LL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Byte	0 0 0 0	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H

Table 1-4. 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	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Three-Byte	0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Word	0 0 0 0	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Byte	0 0 0 0	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H

1.5.4 VIC068A VMEbus Master Read Cycle

This cycle is identical to that of the master write cycle, as described in section 1.5.3, with the following exceptions:

- The VMEbus data buffers are not driven.
- DENO* is not asserted.
- The DENIN1* and DENIN* are asserted.
- DDIR is not asserted. The address and AS* considerations are the same as well as the DSACKi* conventions.

1.5.5 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 DENIN1* and DENIN* 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.

1.5.6 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 1-5*.

Table 1-5. 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.

1.5.6.1 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) instruction 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.

1.5.6.2 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 1.5.6. 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.

1.5.7 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.

1.5.7.1 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.

1.5.8 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.

1.5.9 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 1-3* to *1-6* 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).

1.5.10 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.

1.5.11 Address-Only Cycles

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

1.5.12 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 1-6 summarizes the AM codes for various VIC068A operations and configurations.

Table 1-6. 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
0 1	A32 Addressing	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
1 1	A24 Addressing	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]
0 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

1.6

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)

1.6.1 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 1-7* 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 1.10.2.

Table 1-7. 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	A32 Addressing	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	A24 Addressing	Yes	0 0
User Access Supervisory Access	\$29 \$2D	A16 Addressing	No	1 0
User Defined	User Defined	User Defined	No	1 0
User Defined	User Defined	User Defined	Yes	0 0

1.6.2 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.

1.6.3 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 1-30* in Chapter 1.13 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 (see section 1.5.12). The function code outputs for the VIC068A are as follows:

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

1.6.4 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 single cycle slave accesses, the minimum deassertion time is not considered in that PAS* and DS* are asserted according to the timing discussed in section 1.6.3. The minimum deassertion times for PAS* and DS* are usually used when performing multiple back-to-back local cycles 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 1.6.8). *Figures 1-6* and *1-7* show an example of local reads and writes.

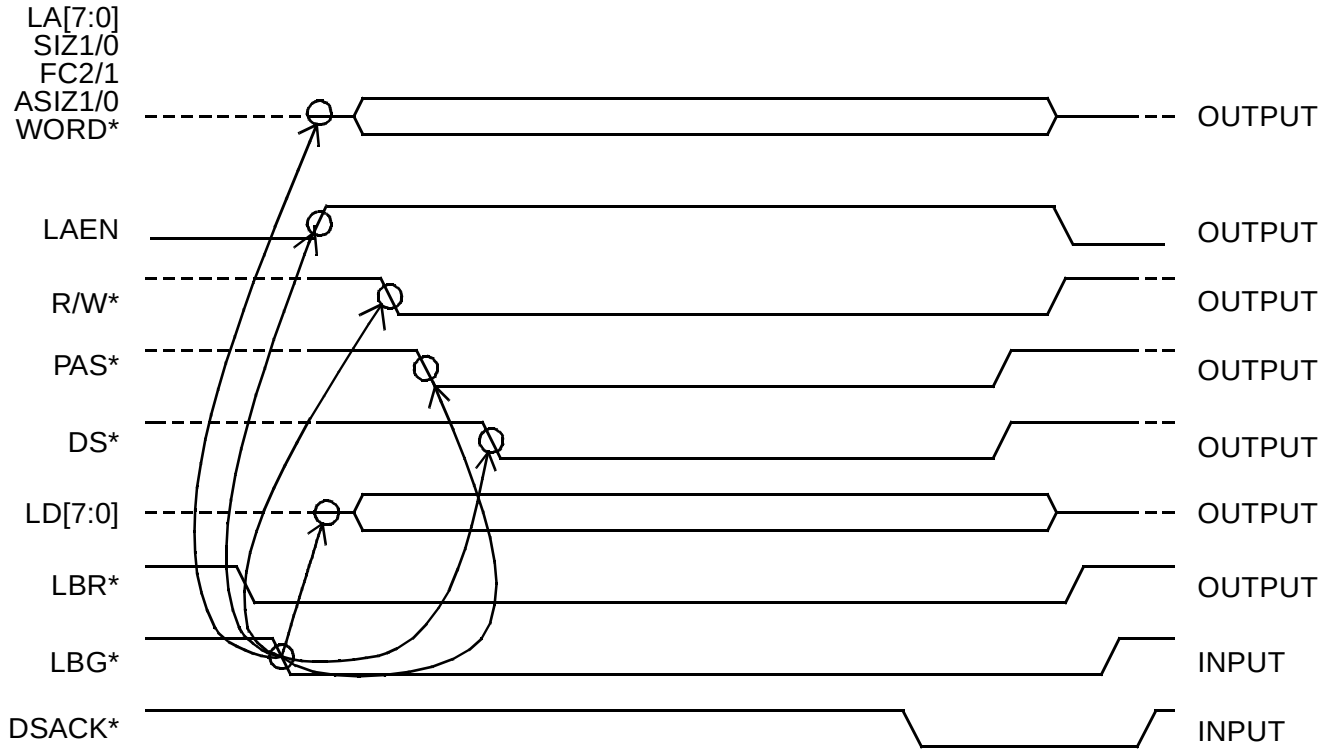


Figure 1-6. Local Bus Cycle—Write

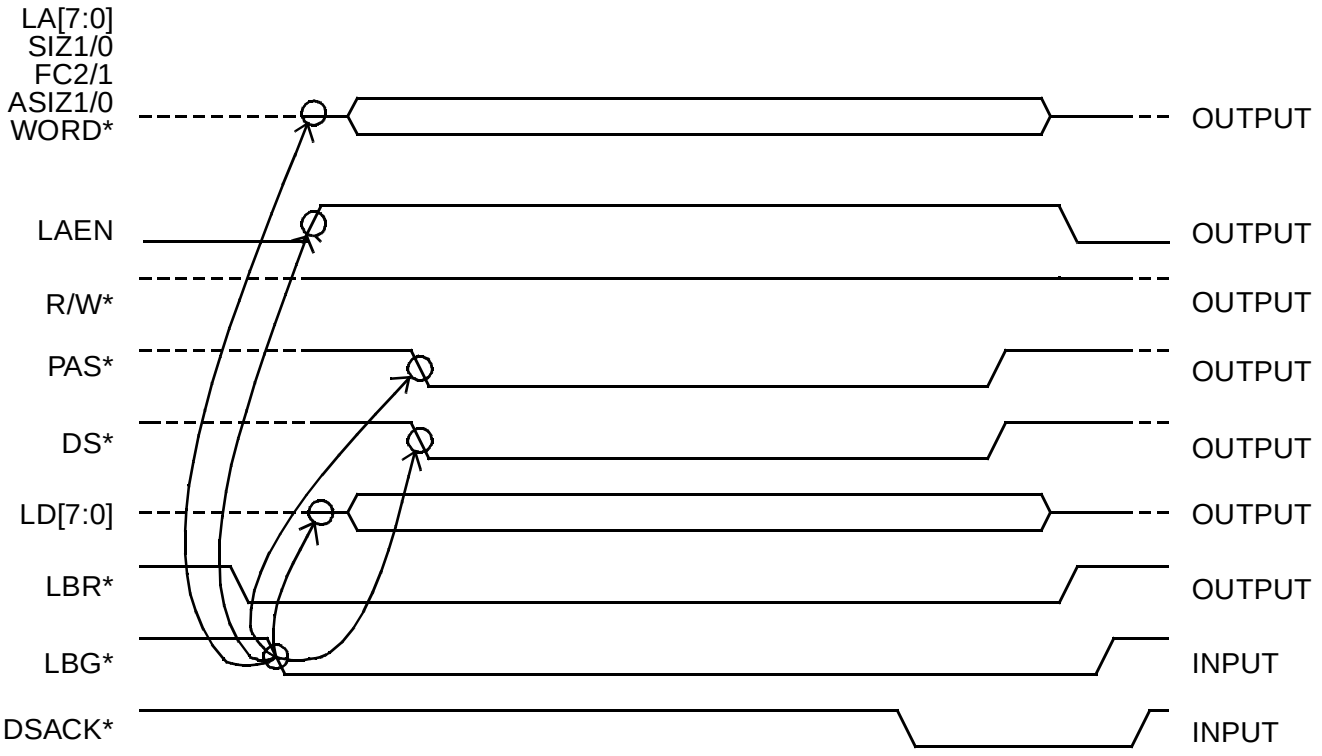


Figure 1-7. Local Bus Cycle—Read

1.6.5 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.

1.6.6 The Latched Bus Interface

When a slave read is performed, data is read from local memory and latched into the VMEbus data transceivers. LD[7:0] is latched into the VIC068A and LD[31:8] is latched into external devices such as the CY7C964s. After DSACKi* is asserted, the VIC068A begins the SAT delay(SSiCR1[3:0]). After this delay times out, the VIC068A asserts both DTACK* and LEDO. These signals are held until both DS1/0* are deasserted.

During a slave write access, the VIC068A asserts the LEDI signal immediately after receiving a valid slave select latching the data into the data transceivers.

1.6.7 Slave Write Posting

The VIC068A is able to perform slave write posting, when SS1CR0[7] is set. 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 that the cycle did not complete since DTACK* was already asserted.

1.6.8 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 algorithms. 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 1-8. Buffer Control Signals: D32 VMEbus Slave Write Operation

Data Path Size	VMEbus Stimulus			Local Bus 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	DDIR1	SWDEN*	ISOBE*	DENIN*
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 1-9. Buffer Control Signals: D32 VMEbus Slave Read Operation

Data Path Size	VMEbus Stimulus			Local Bus 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	DDIR1	SWDEN*	ISOBE*	DENIN*
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
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 1-10. Buffer Control Signals: D16 VMEbus Slave Read Operation

Data Path Size	VMEbus Stimulus			Local Bus 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	DDIR1	SWDEN*	ISOBE*	DENIN*
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 1-11. Buffer Control Signals: D16 VMEbus Slave Write Operation

Data Path Size	VMEbus Stimulus			Local Bus 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	DDIR1	SWDEN*	ISOBE*	DENIN*
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

1.7 VIC068A Control Register Access

1.7.1 Control Registers

The VIC068A contains 58 8-bit internal registers addressable from the local bus. These registers provide complete control and monitoring of the VIC068A.

1.7.2 Control Register Access

Access to internal registers of the VIC068A is accomplished through the assertion of CS*, PAS* and DS*. It is the value of R/W* that determines if the access is a read or a write.

When CS*, PAS* and DS* are all driven Low (in any order) to the VIC068A, access to an internal register is initiated (i.e., REGISTER_ACCESS* = 0 when (CS*[0] and PAS*[0] and DS*[0]). Register access completes when either CS*, PAS*, or DS* deassert (i.e., REGISTER_ACCESS* = 1 when (CS*[1] or PAS*[1] or DS*[1]). *Figure 1-8* shows a timing diagram of the VIC068A internal register accesses. The timing values are explained in *Table 1-12.*

Table 1-12. Register Access Timing Values

Operation		Commercial		Industrial		Military	
		Min.	Max.	Min.	Max.	Min.	Max.
REGISTER ACCESS							
M1	PAS*[0] & DS*[0] & CS*[0] to DSACKi*[L]	4T+5	5T+34	4T+5	5T+35	4T+4	5T+38
M2	PAS*[0] & DS*[0] & CS*[0] to LD[7:0] Valid	3T+5	4T+28	3T+5	4T+29	3T+4	4T+37
M3	AS*[0] & ICFSEL*[0] to DTACK*[L]	4T+6	4T+30	4T+5	4T+31	4T+5	4T+34
M4	PAS*[0] & DS*[0] & CS*[0] to LD[7:0], LA[7:0] Valid	2T+6	3T+30	2T+5	3T+31	2T+5	3T+34

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

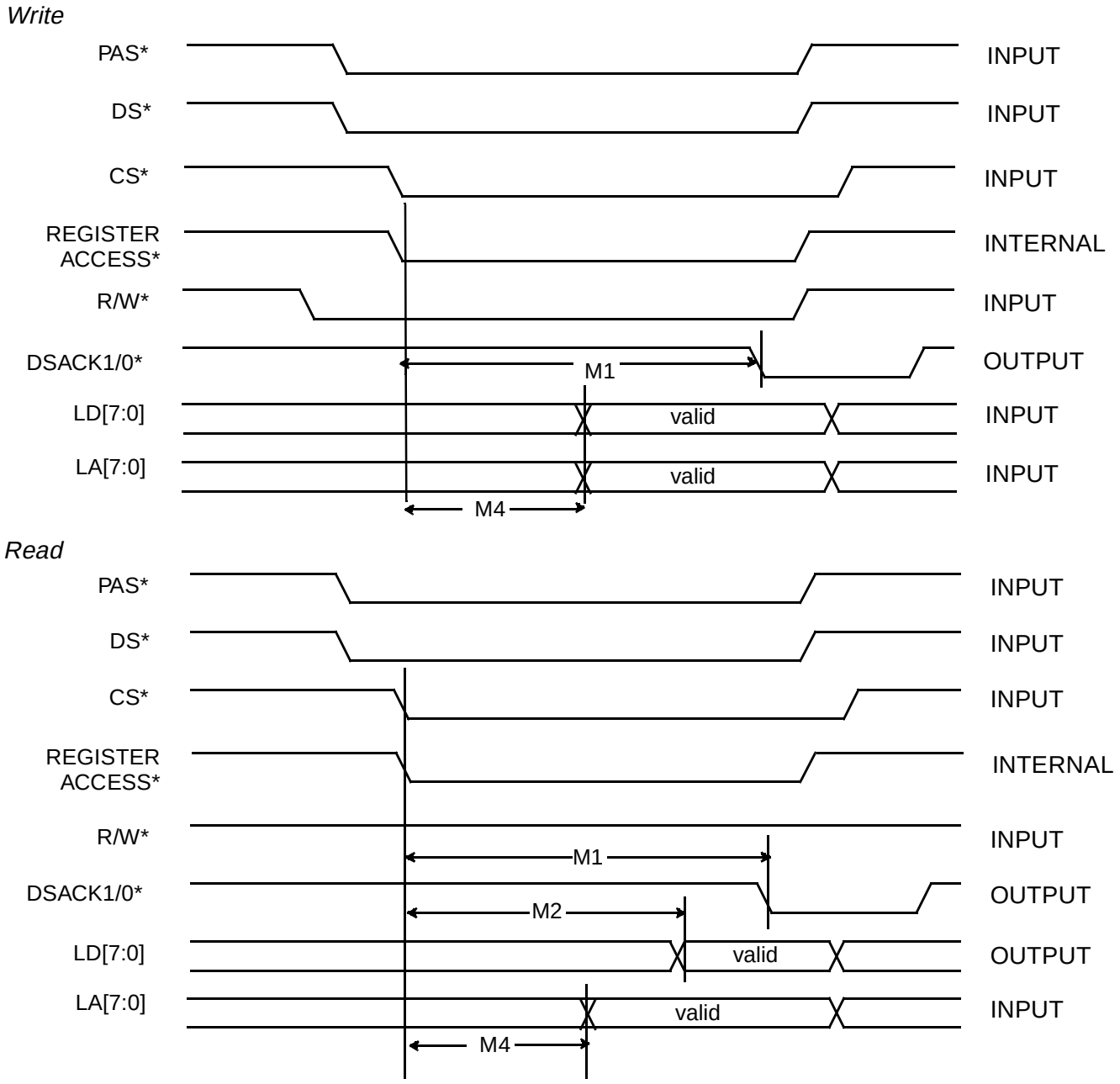


Figure 1-8. VIC068A Internal Register Access Timing

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
```

1.8 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)

1.8.1 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 1-13*.

Table 1-13. ICF VMEbus Address Map

A7	A6	A5	A4	A3	A2	A1	LWORD	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

1.8.2 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 sema-

phores are set whenever ICRs 5–0 are written. In addition, ICR7 also indicates VMEbus mastership and a mask for SYSFAIL*. Refer to Chapter 1.12 for detailed register descriptions.

1.8.3 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 1.9 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.

1.8.4 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.

1.9 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)

1.9.1 VMEbus Interrupter

The VIRSR controls the assertion of the VMEbus interrupts. VIRSR[7:1] control the assertion (and deassertion if desired) of IRQ*7–1 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.

1.9.2 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 1.9.3.

Table 1-14 summarizes the VMEbus interrupt acknowledge cycle.

Table 1-14. 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 inverted version of IPLi value as programmed in the VICRi	
4			Detect IPLi asserted
5			Place inverted IPL level of interrupt being handled on LA[3:1]
6			Assert FCIACK*, PAS*, DS*
7		Detect FCIACK* asserted	
8		Sample LA[3:1] for request level being acknowledged	
9		If matched, obtain VMEbus mastership	
10		Assert IACK* and place number of IRQ being acknowledged on A[3:1]	
11		Enable D[7:0] to capture status/ID vector	
12	Detect IACKiIN* asserted and sample A[3:1]		
13	If A[3:1] matches IRQ number requested, 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

1.9.3 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 and assert DSACKi*. 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.

- 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 1.9.8 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 1-15* summarizes the local interrupt acknowledge cycle.

Table 1-15. Local Interrupt Acknowledge Cycle

Step	Local Interrupter	VIC068A Handler	Local Processor
1	Generate LIRQi*		
2		Assert inverted version of IPLi value programmed in the LICRi	
3			Detect IPLi asserted
4			Place inverted IPL level of interrupt being handled on LA[3:1]
5			Assert FCIACK*, PAS*, DS*
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, DSACKi*		
10			Capture status/ID
11			Enter ISR

1.9.4 The FCIACK Cycle

When the VIC068A detects an interrupt (either VME or local) that it has been programmed to handle, the VIC068A begins a FCIACK cycle. The IPL value associated with that interrupt (as programmed in VIICR, VICR1–7, DMASICR, LICR1–7, ICGSICR and ICMSICR) is placed into an internal lookup table and driven (inverted) onto the IPL lines. The value placed onto LA[3:1] by the local processor during a FCIACK cycle is compared with the lookup table within the VIC068A. If a match is found, the interrupt that the local processor has agreed to service is handled.

Since interrupts occur asynchronous to each other, it is possible for the value of the IPL lines to change without warning. If the IPL lines were not sampled prior to them changing, that interrupt will not be seen by the local processor until all higher priority interrupts have been handled. Once FCIACK* is asserted to the VIC068A, the IPLi lines will be frozen until the Interrupt Acknowledge cycle is complete. Since the value placed onto LA[3:1] by the local processor is compared against a lookup table and not the current value of the

IPL lines the VIC068A will handle any pending interrupt that the local processor agrees to handle during a FCIACK cycle.

1.9.5 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.

1.9.6 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.

1.9.7 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.

1.9.8 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.

1.10

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.

1.10.1 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 (such as CY7C964s), 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 number of VMEbus cycles between interleave periods is called the *burst length*. The total number of bytes to be transferred is called the *block transfer 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.

1.10.1.1 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 1-9* shows the BLT initiation cycle (pseudo 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.

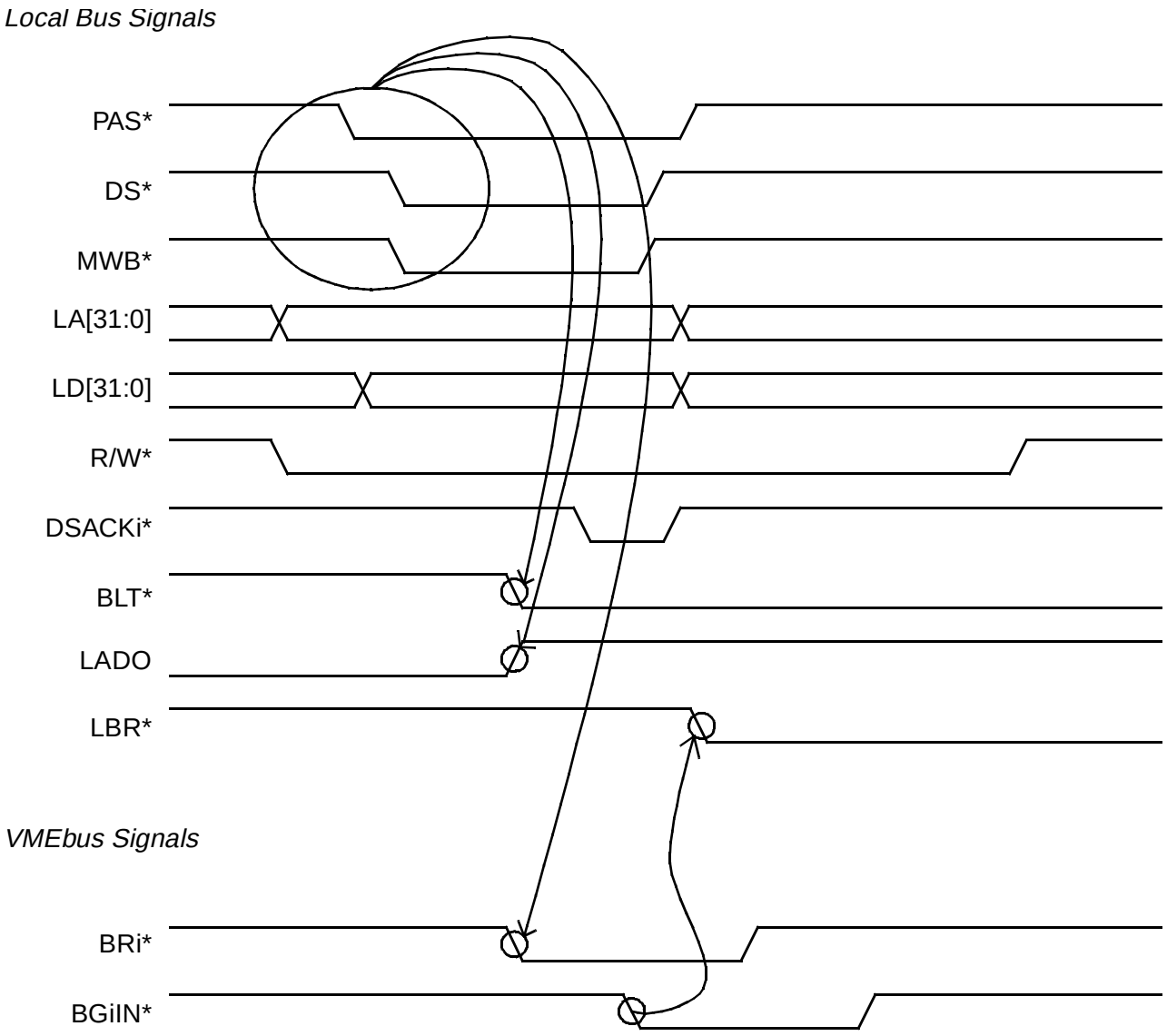


Figure 1-9. Master Block Transfer with Local DMA Initiation Cycle (pseudo cycle)

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. CY7C964s, or external logic, should decode BLT*, LBG*, PAS*, LAEN, and the FCi signals to drive the upper portion of the address lines. See section 1.10.1.1.8.1.3 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.

1.10.1.1.1 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.

1.10.1.1.2 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 BTLR0[0] is set, the BTLRs are ignored and only a single burst, as defined in RCR[5:0], will be performed.

1.10.1.1.3 256-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.

1.10.1.1.4 256-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.

1.10.1.1.5 Interleave Period

The time between bursts is known as the interleave period. The length of the interleave period is configurable in BTCR[3:0]. During the interleave period, slave cycles can be performed. Master cycles are allowed if the dual-path feature is enabled.

1.10.1.1.6 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.

1.10.1.1.7 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 once the block transfer begins the BLT enable bit must be cleared. It is not necessary to wait for the completion of the block transfer before clearing BTCR[6], this bit can be cleared during an interleave period. 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 initiate 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.

1.10.1.1.8 Example Configurations

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

1.10.1.1.8.1 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.

1.10.1.1.8.1.1 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.

1.10.1.1.8.1.2 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.

1.10.1.1.8.1.3 External Hardware Implementation

Figure 1-10 shows the hardware required for implementing the block transfers described above. The 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*.

The simplest solution is to use CY7C964s in place of using discrete components (’373s and ’543s).

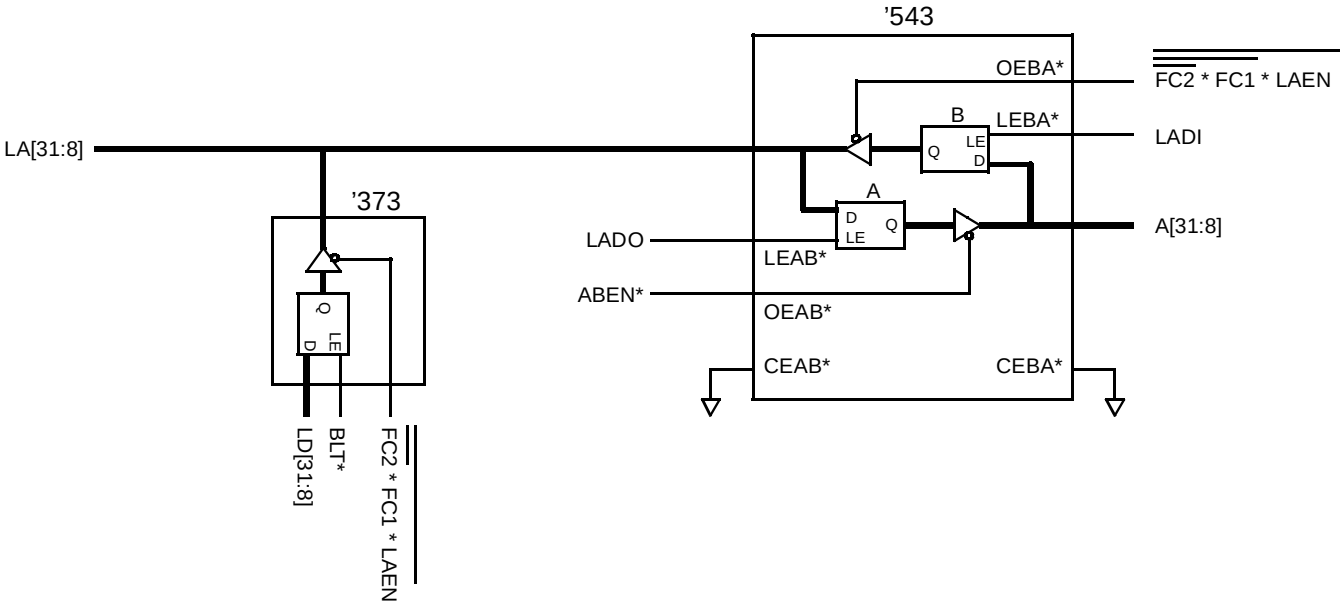


Figure 1-10. Minimum BLT Logic

1.10.1.1.8.2 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 configured 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 configured to specify the burst length.

1.10.1.1.8.2.1 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.

1.10.1.1.8.3 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.

1.10.1.1.8.3.1 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.

1.10.1.1.9 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.

1.10.1.1.10 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 1-11 through *1-15* demonstrate these delays. See Chapter 1.13 for the actual AC performance specifications.

1.10.1.2 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.

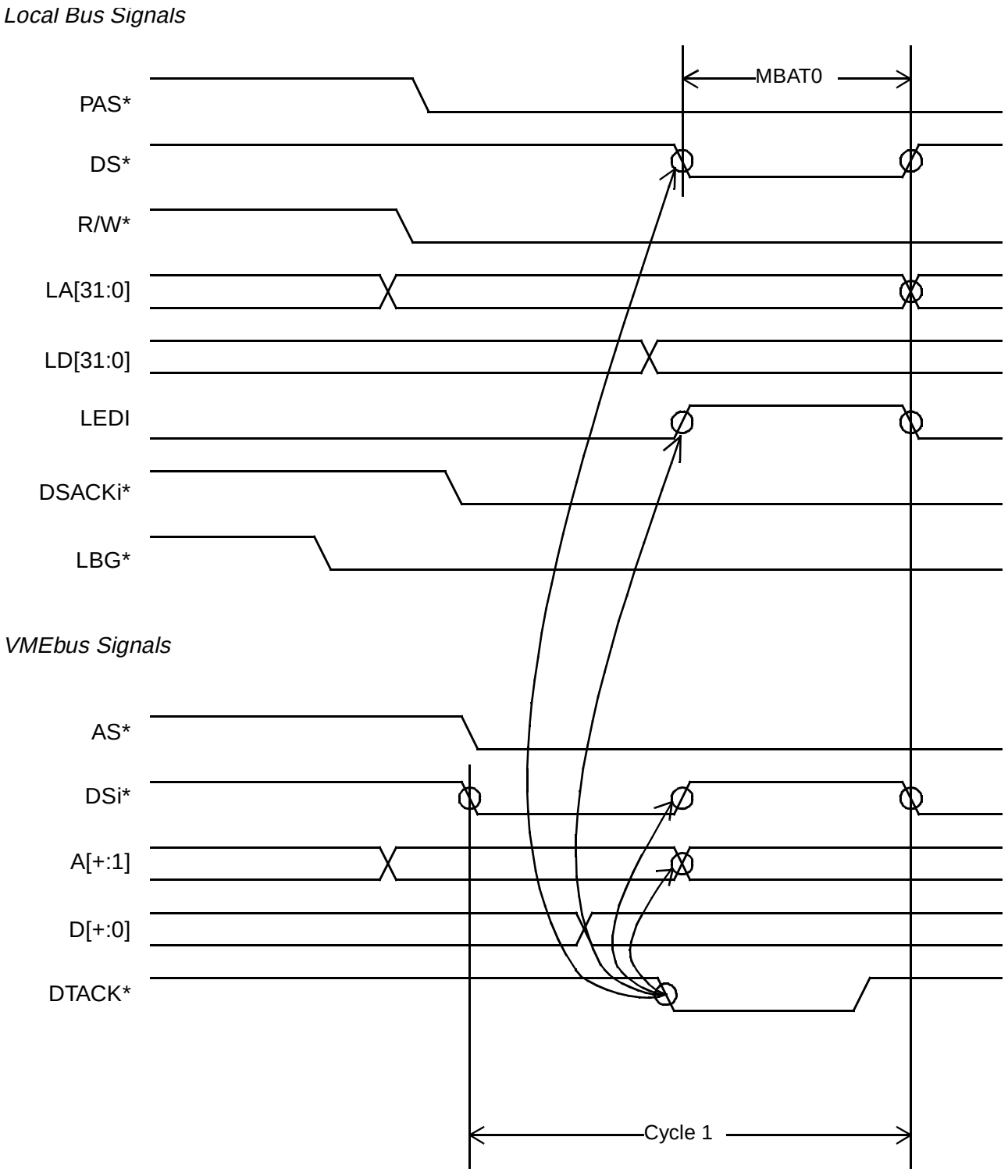


Figure 1-11. Master Block Transfer—Read, First Cycle

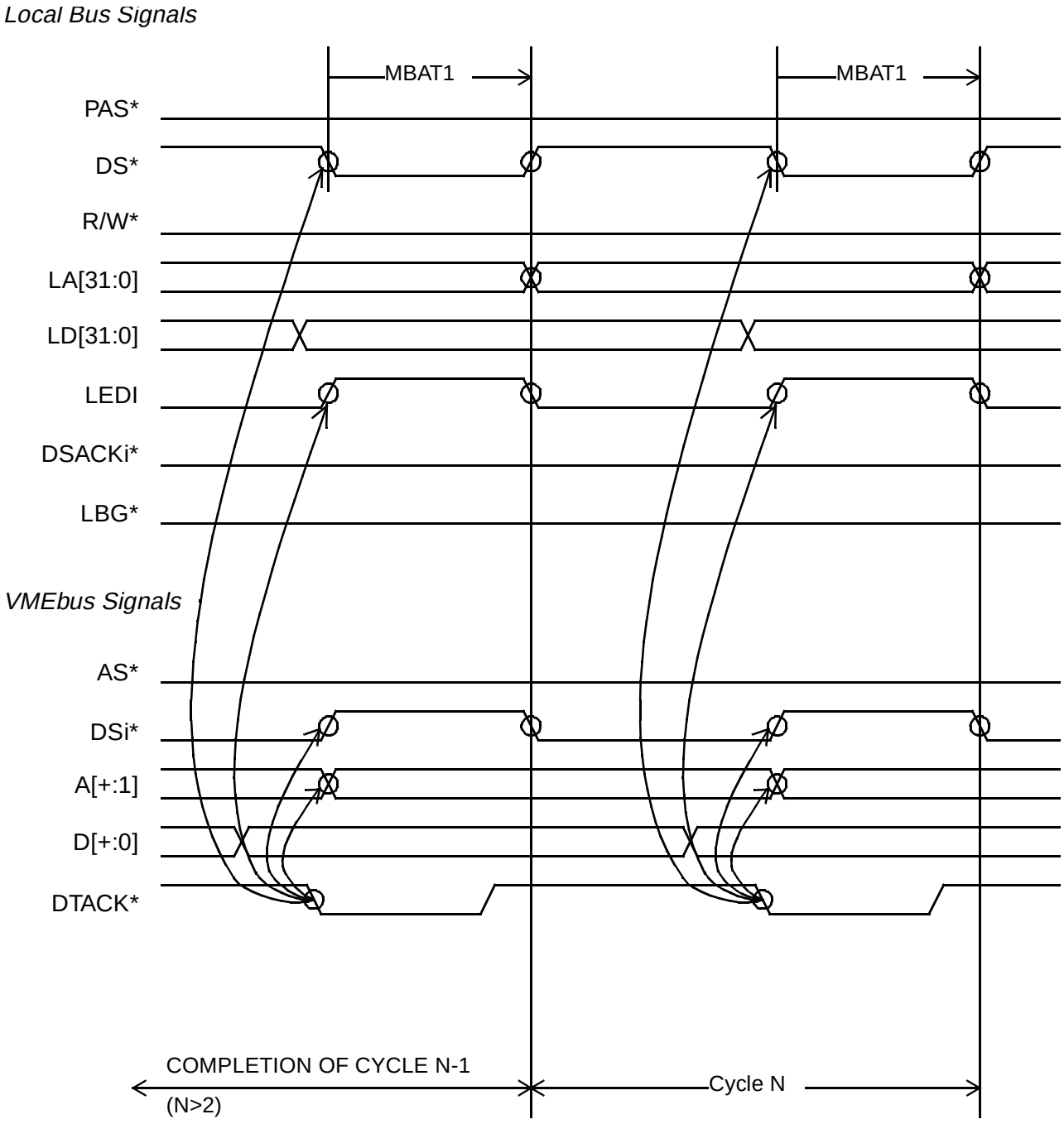


Figure 1-12. Master Block Transfer—Read, Second and Subsequent Cycles

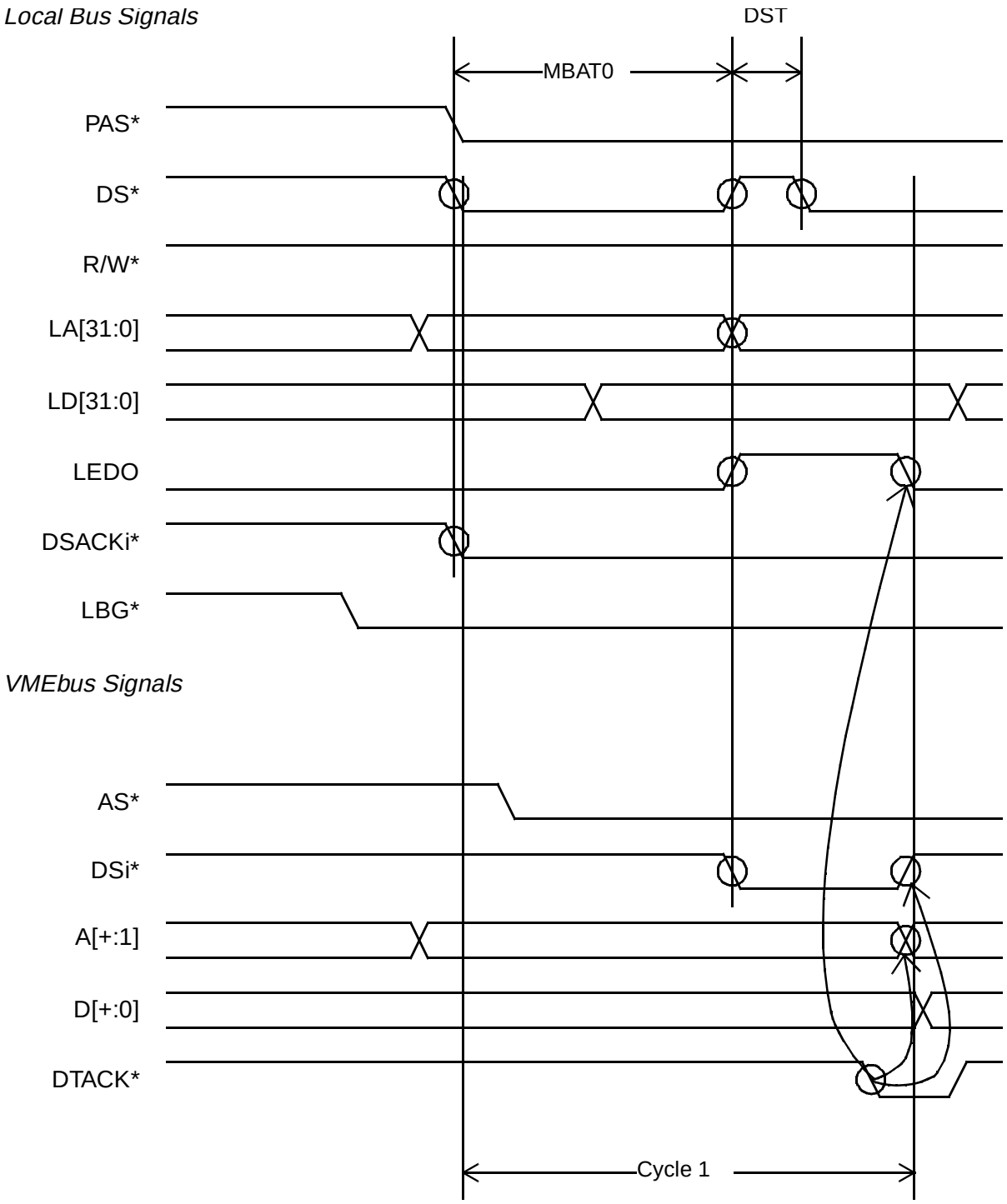


Figure 1-13. Master Block Transfer—Write, First Cycle

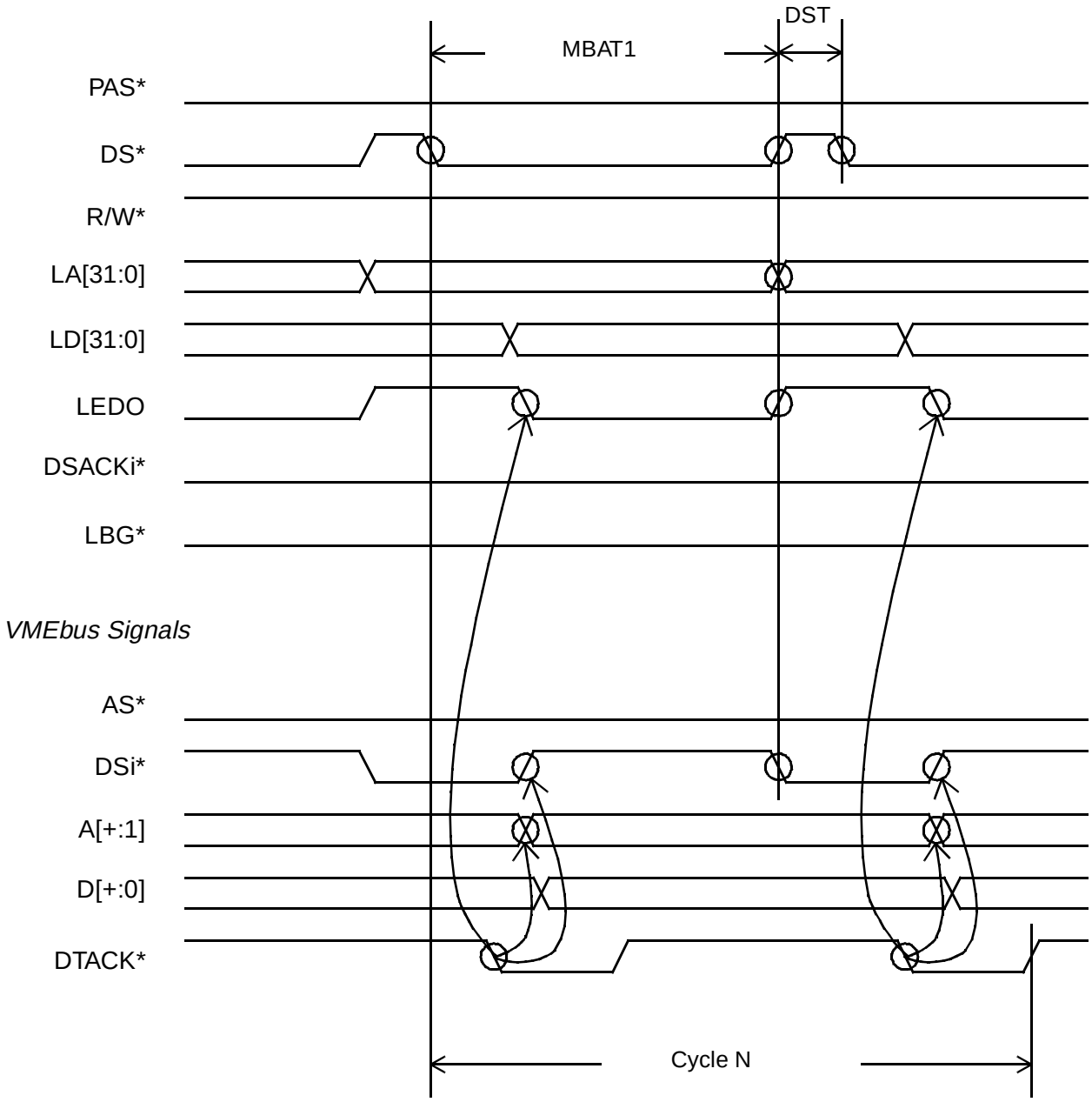


Figure 1-14. Master Block Transfer—Write, Second and Subsequent Cycles (Fast Slave)

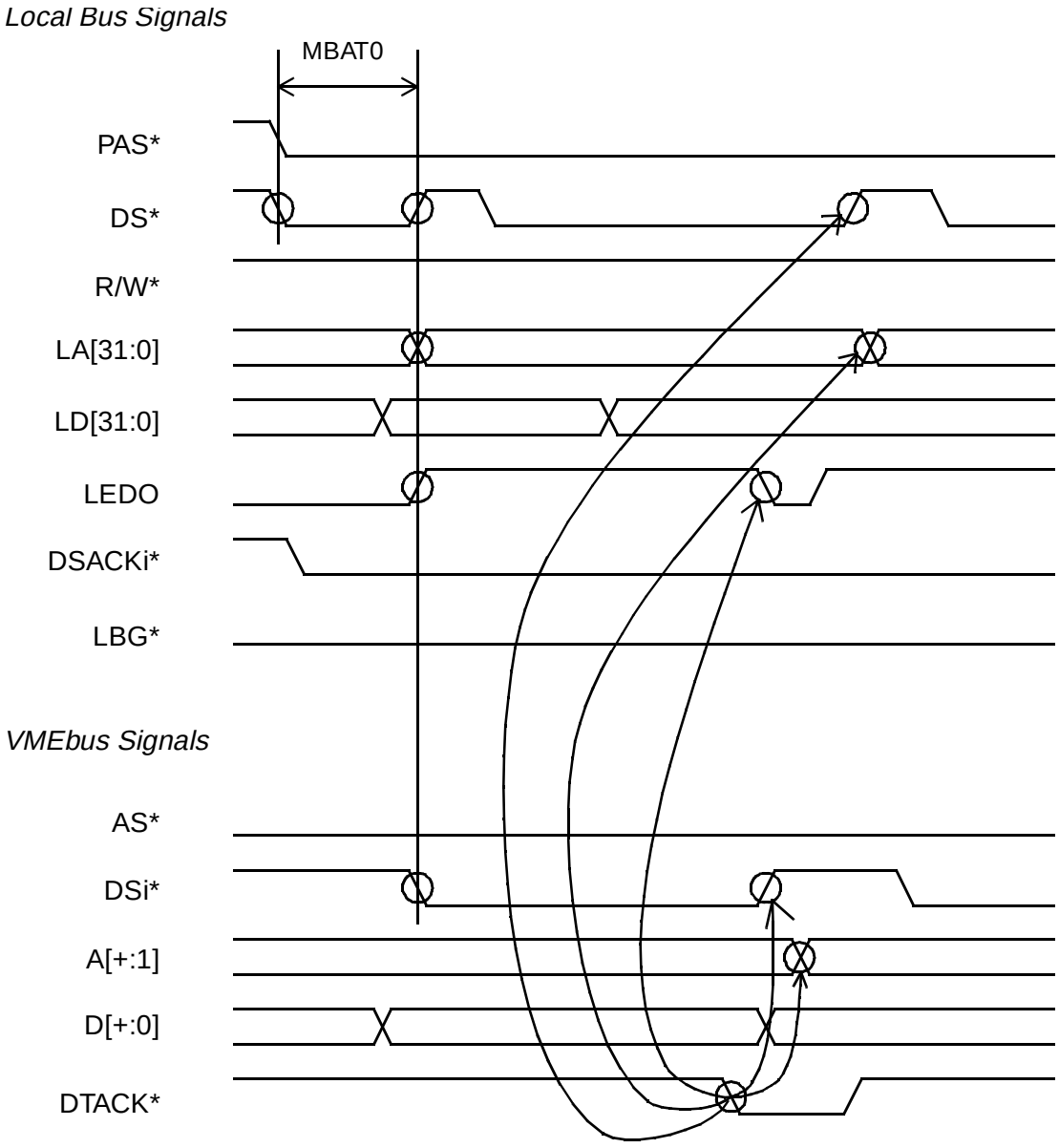


Figure 1-15. Master Block Transfer—Write (Slow Slave)

1.10.1.2.1 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)
```

1.10.1.3 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.

1.10.1.4 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[1]). 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.

1.10.2 VIC068A Slave Block Transfers

The VIC068A as a slave may be configured, in SSiCR0, to perform in one of three modes for block transfers:

- does not support block transfers
- supports block transfers, but emulates single-cycle transactions 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 (accelerated transfers)

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 acknowledgment.

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.

1.10.3 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.

1.10.4 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 Section 4, The CY7C964 Bus Interface Logic Circuit, for details on a VIC068A/CY7C964 interface.

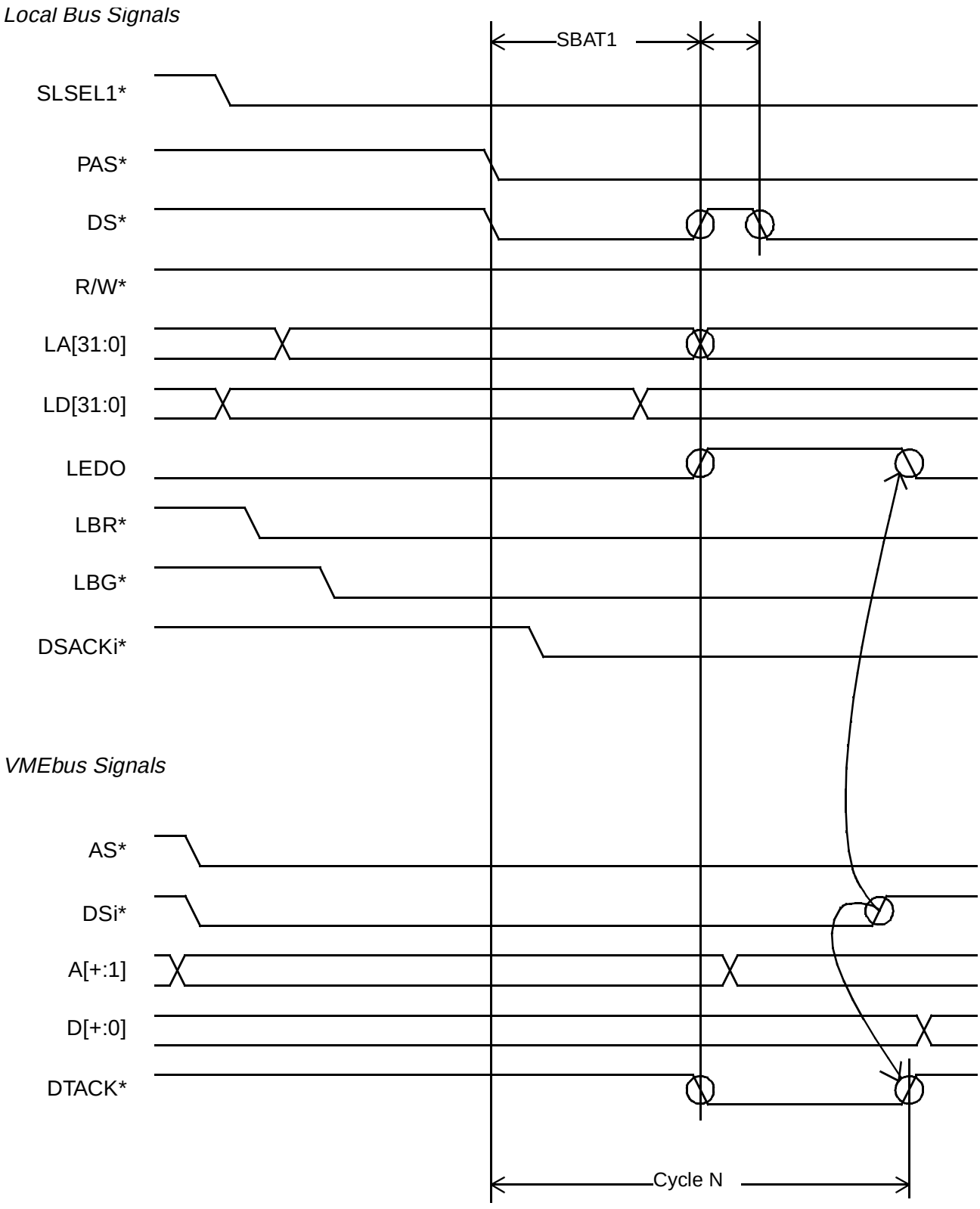


Figure 1-16. Slave Block Transfer—Read, First Cycle

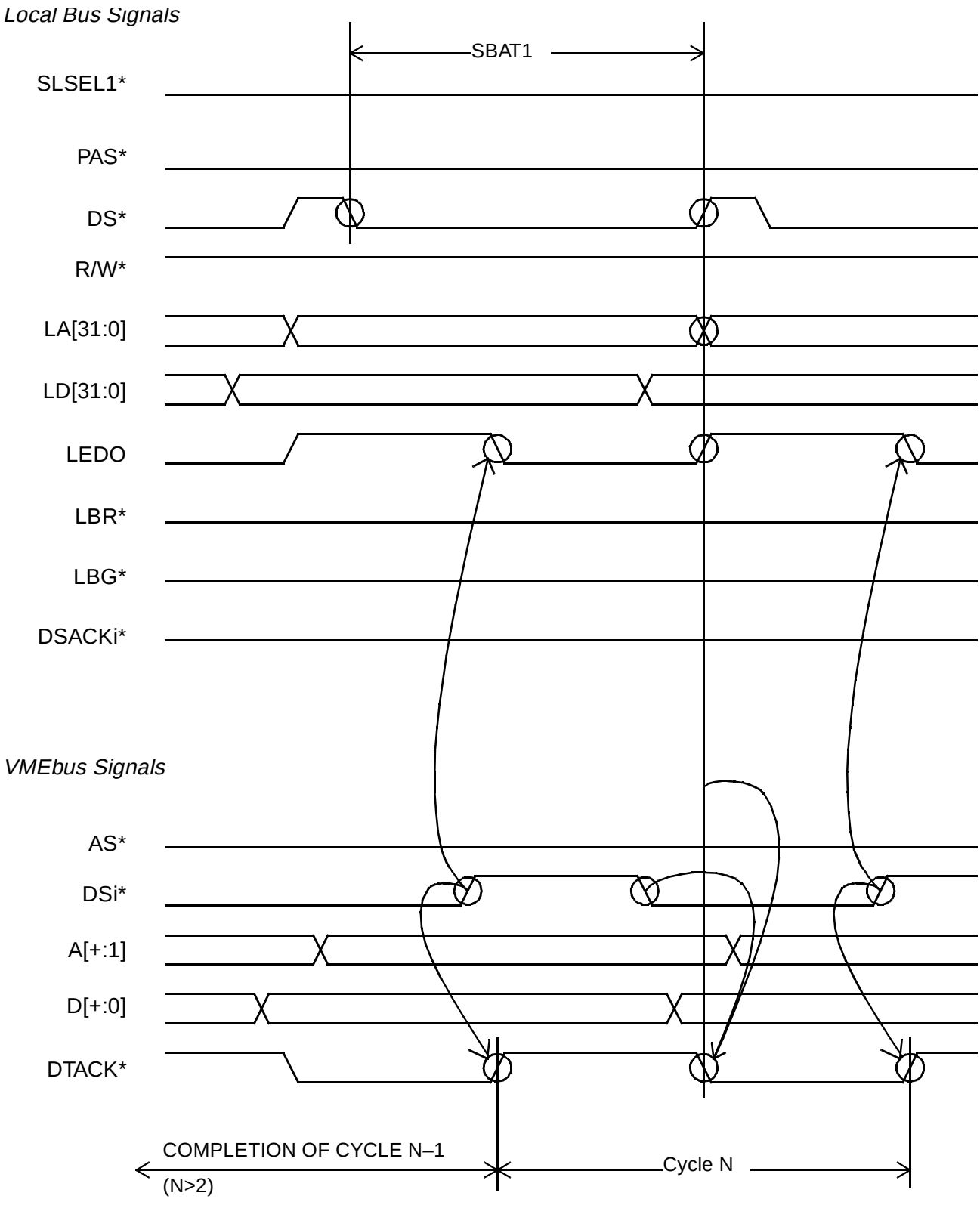


Figure 1-17. Slave Block Transfer—Read, Second and Subsequent Cycles

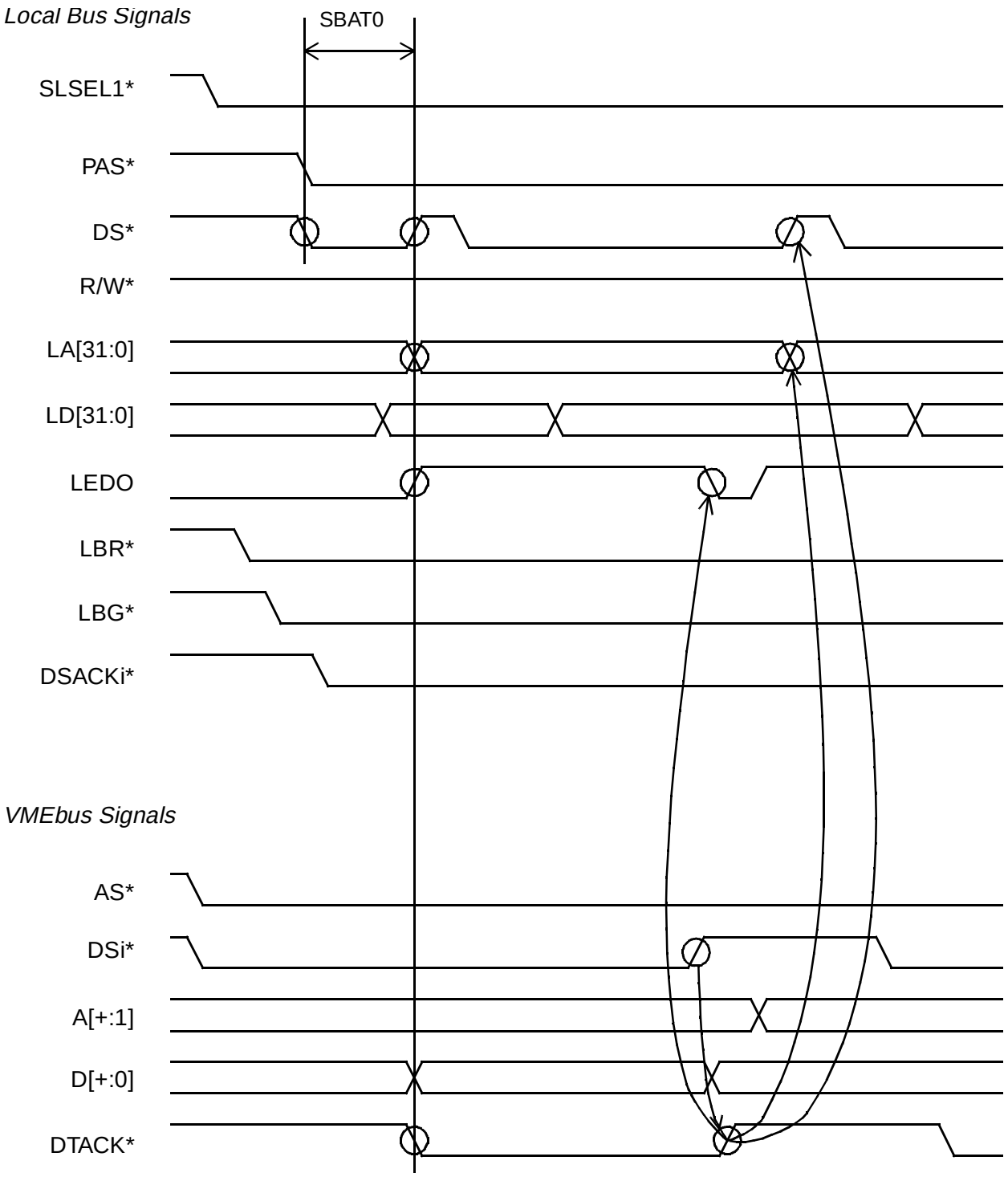


Figure 1-18. Slave Block Transfer—Read, Slow Master

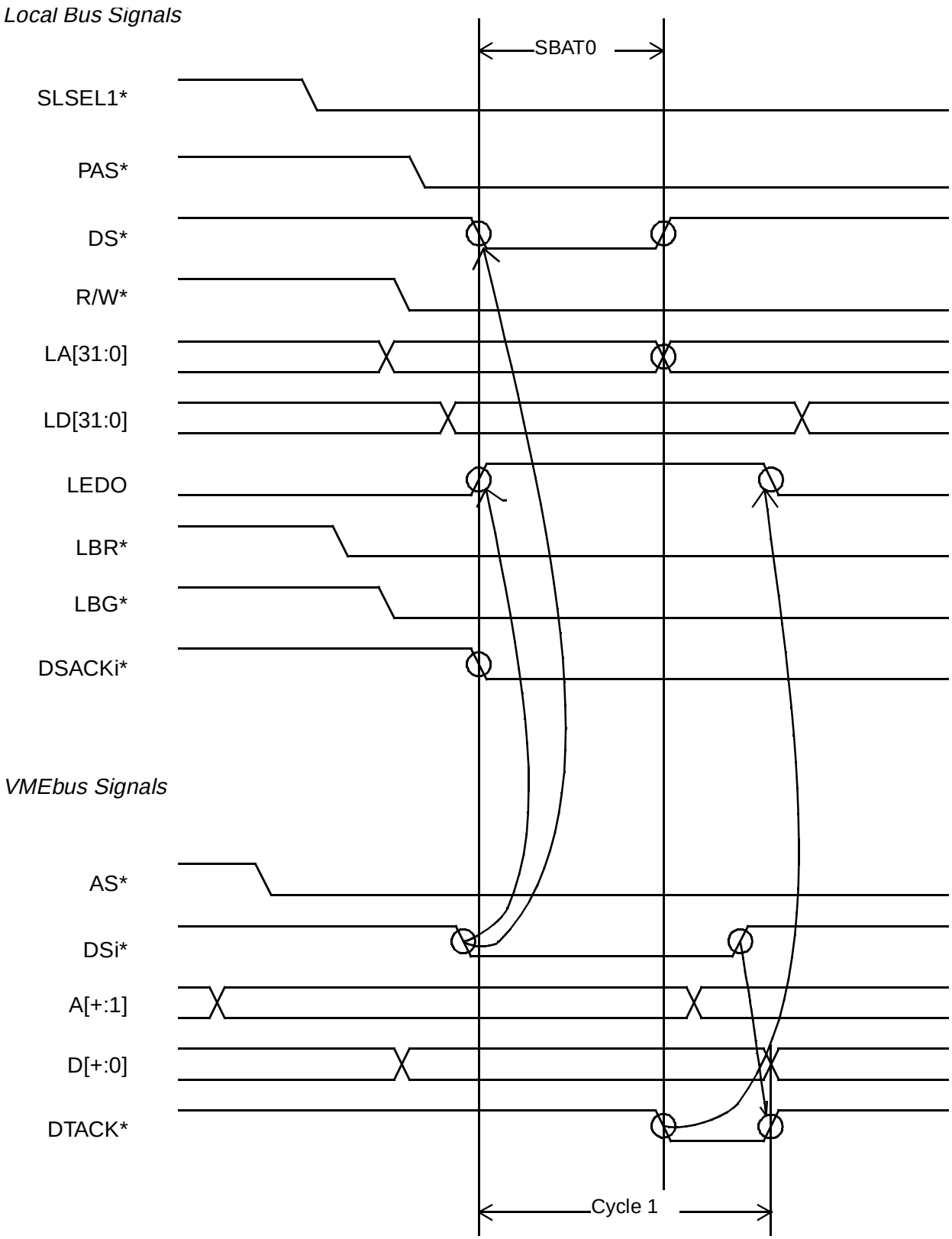


Figure 1-19. Slave Block Transfer—Write, First Cycle

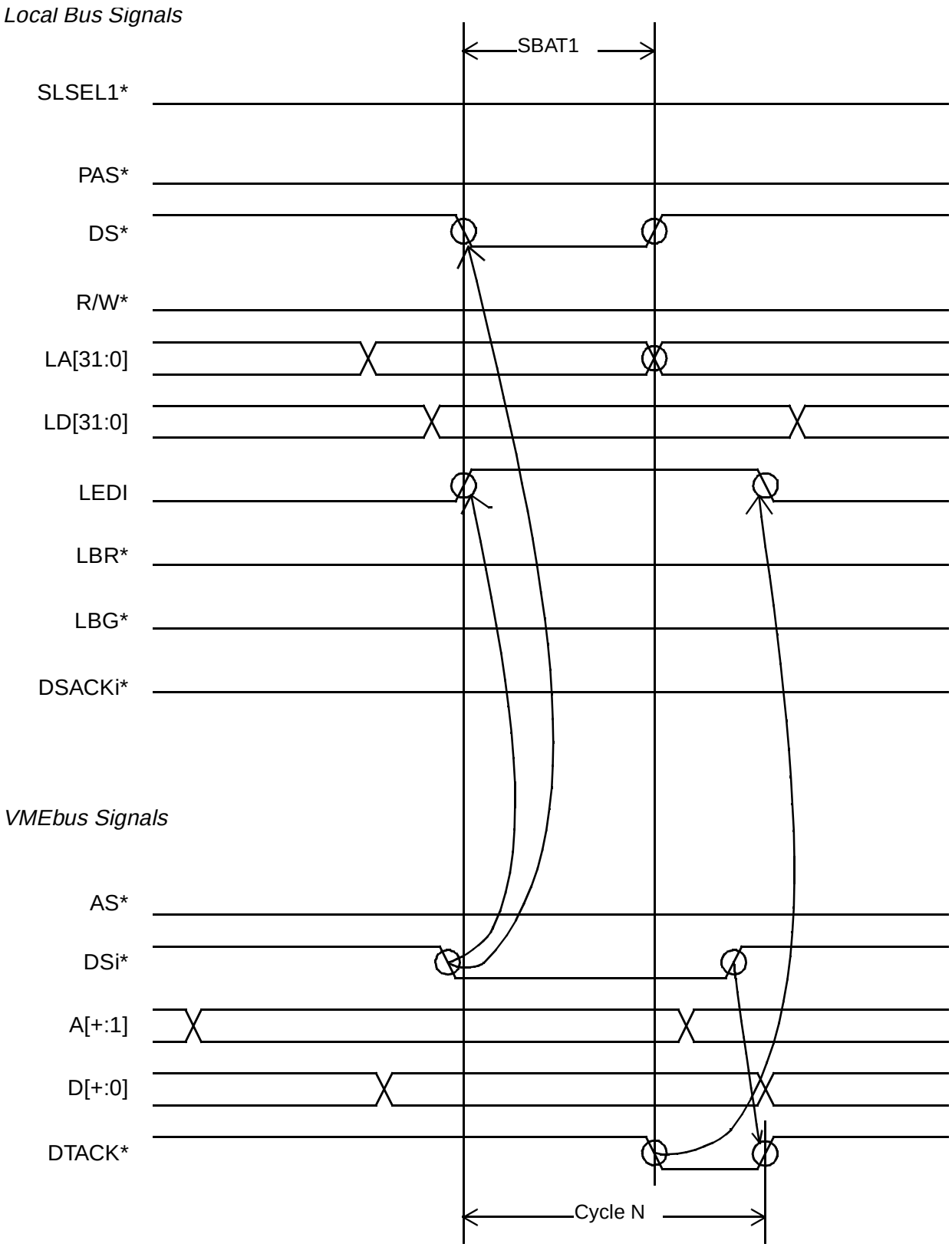


Figure 1-20. Slave Block Transfer—Write, Second and Subsequent Cycles

1.11
Miscellaneous Features

This chapter describes additional miscellaneous features of the VIC068A.

1.11.1 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

Upon completion of a VIC068A reset, ICR6[6] is set, thus causing SYSFAIL* to be asserted. To remove the SYSFAIL* set ICR7[7], which is cleared by a global or power-up reset.

IMPORTANT

It is vital to the proper operation of the VIC068A that a global reset be performed at power-up. It is not necessary to wait for a system reset (caused by an assertion of SYSRESET on the VMEbus) to complete before asserting a global reset. See section 1.11.1.2.*

1.11.1.1 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 ms 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.

1.11.1.2 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. A global reset must be issued at power-up to insure proper operation of the VIC068A.

The global reset is initiated by asserting the IPL0 signal after the IRESET* signal is asserted. It is important that IPL0 be asserted 16 ns (commercial) or 20 ns (industrial/military) 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 while IPL0 is being asserted.
- 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 first (i.e., before IRESET*), the reset continues until a 200-ms timeout expires.
- If the IRESET* signal is deasserted first (i.e., before IPL0), 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.

1.11.1.3 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 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.

If an internal or global reset is performed on a system controller VIC068A while SYSRESET* is being asserted by another board, the VIC068A will not assert SYSRESET*. This implies that any SYSRESET* asserted to a system controller VIC068A should be asserted for the 200 ms required by the VMEbus specification.

1.11.1.4 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 specified operating range (see Chapter 1.16).
- 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 stable, re-issue the global reset when the environment is stable.

Because a global reset causes certain VMEbus system controller operations to be suspended (see section 1.11.1.2), 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.

1.11.2 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 ms.

1.11.3 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.

1.11.4 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*	

1.11.5 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.

1.11.6 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.

1.12

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 1-16 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 1-16. 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
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

Table 1-16. VIC068A Register Values After Reset Operations (continued)

Address (hex)	Name	Description	Global Reset	Internal Reset	System Reset
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	BTLR1	Block Transfer Length Register 1	00000000	00000000	00000000
DF	BTLR0	Block Transfer Length Register 0	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 1s.

Bit 7: (1/1/1) VMEbus interrupt mask. When clear, the VIC068A signals a local interrupt at the acknowledgment 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: For interrupt:	$07, $0B, $0F, $13, $17, $1B, $1F 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 1s.
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 1s.
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.

Local Interrupt Control Registers 1–7 (continued)

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 clear (level sensitive) or on a rising edge if bit 5 is set (edge sensitive). When clear, the VIC068A responds to interrupts as active Low if bit 5 is clear (level sensitive) or on a falling edge if bit 5 is set (edge sensitive).
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.

ICMS Interrupt Control Register (continued)

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.
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.

Local Interrupt Vector Base Register (continued)

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: For registers:	$63, $67, $6B, $6F, $73 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 1.8 for details on accessing these registers from the VMEbus.
Bits 7–0: (0/0/0)	Data field.

Interprocessor 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 1.8 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 1.8 for details on accessing this register from the VMEbus.

Interprocessor Communication Register 6 (continued)

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 ms 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 1s.

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 1.8 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 must be cleared by the user (i.e., they are not cleared automatically). These bits can be read or written from the local bus or the VMEbus.

Interprocessor Communication Register 7 (continued)

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). This bit must be written after resetting the VIC068A to deassert SYSFAIL*.

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 automatically when the interrupt is serviced.

VMEbus Interrupt Vector Base Registers 1–7

Name:	VIVBR
Address: for IRQ:	$87, $8B, $8F, $93, $97, $9B, $9F 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.

Transfer Timeout Register (continued)

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 (ms)*
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 (ms)*
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.

Local Bus Timing Register

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 1.10 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 1s.

Interface Configuration Register

Name:	ICR
Address:	$AF
Description:	Controls various features of the VIC068A including RMCs, deadlock signaling, metastability 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.

Interface Configuration Register (continued)

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 1.11.5).

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 5 and 6 with the SIZ1/0 signals. If RMC control bits 5 or 6 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.

Arbiter/Requester Configuration Register (continued)

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 set.

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.

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.

Bus Error Status Register (continued)

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. 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 is cleared automatically by the VIC068A after completion of the local DMA operation, 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.

DMA Status Register (continued)

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 1s.
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.

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

Slave Select 0 Control Register 0 (continued)

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.

Slave Select 0 Control Register 1 (continued)

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)

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)

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.

Slave Select 1 Control Register 0 (continued)

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.

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*.

Slave Select 1 Control Register 1 (continued)

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

Slave Select 1 Control Register 1 (continued)

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)

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)	Special purpose. For normal operation set this bit to 0.

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

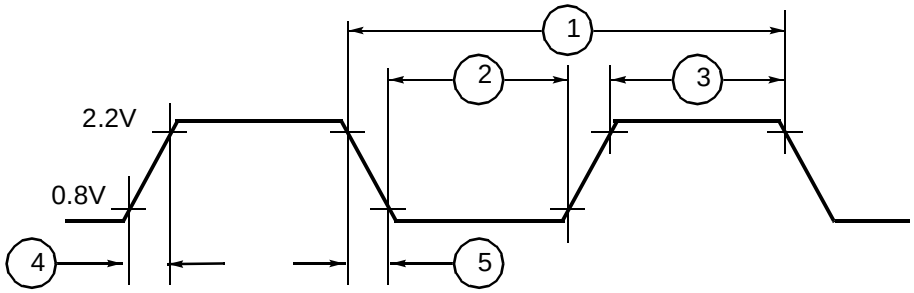
Name:	SRR
Address:	$E3
Description:	The system reset register provides the means to perform a VMEbus system reset (SYSRESET* asserted). Writing a value of $F0 causes this function to occur. A system reset is also performed within the VIC068A.
Bits 7–0: (1/1/1)	System reset field. Writing this bit field 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.

1.13

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, 4	2½T+5	3T+25	2½T+4	3T+26	2½T+4	3T+31⁄
A2	BRi*[0] to BBSY*[L]	4, 5	3T+8	3½T+28	3T+7	3½T+34	3T+7	3½T+35
A3	BRi*[0] to BGiOUT*[L]	4, 5	3T+4	4T+25	3T+4	4T+26	3T+3	4T+28
A4	BRi*[0] to BCLR*[L]	4	2	16	2	16	2	19
A5	BGiIN*[0] to BGiOUT*[L]	4	2	18	2	18	2	20
A6	BGiIN*[0] to BBSY*[L]	6	4	23	4	24	3	25
A7	BGiIN*[0] to BRi*[H]	4, 6	5	3T+26	4	3T+27	4	3T+31
A8	BGiIN*[1] to BGiOUT*[H]	4	3	20	2	21	2	23
A9	BBSY*[0] to BGiOUT*[H]	4, 5	4	21	3	22	3	24
A10	BBSY*[1] to BGiOUT*[L]	4	3T+5	4T+25	3T+4	4T+26	3T+3	4T+29
A11	BBSY*[1] to BCLR*[H]	4	1T+4	2T+24	1T+4	2T+25	1T+3	2T+27
MASTER ACCESSES								
B1	BGiIN*[0] to DENO*[L]	4, 6, 7	8	3T+36	7	3T+37	6	3T+42
B2	BGiIN*[0] to LADO[H]	4, 6	14	3T+59	13	3T+61	12	3T+67

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
B3	BGiIN*[0] to AS*[L]	4, 6	3T+5	6T+28	3T+5	6T+29	3T+4	6T+31
B4	BGiIN*[0] to A[7:1] Valid	4, 6	6	3T+31	6	3T+32	5	3T+37
B5	BGiIN*[0] to LWORD*[H/L]	4, 6	6	3T+31	6	3T+32	5	3T+37
B6	BGiIN*[0] to WRITE*[H/L]	4, 6	6	3T+31	6	3T+32	5	3T+37
B7	BGiIN*[0] to ABEN*[L]	4, 6	7	3T+34	6	3T+36	6	3T+38
B8	PAS*[0] & MWB*[0] to BRi*[L]	4	4	22	3	22	3	24
B9	PAS*[0] & MWB*[0] to ISOBE*[L]	4	4	22	3	23	3	25
B10	PAS*[0] & MWB*[0] to LADO[H]	4	15	60	13	62	12	68
B11	PAS*[0] & MWB*[0] to BB-SY*[L]	4, 8	7	32	5	33	5	36
B12	PAS*[0] & MWB*[0] to ABEN*[L]	4, 8	1½T+8	2½T+36	1½T+7	2½T+37	1½T+6	2½T+41
B13	PAS*[0] & MWB*[0] to A[7:1]	4, 8	1½T+7	2½T+36	1½T+6	2½T+37	1½T+5	2½T+41
B14	PAS*[0] & MWB*[0] to LWORD*[H/L]	4, 8	1½T+7	2½T+36	1½T+6	2½T+37	1½T+5	2½T+41
B15	PAS*[0] & MWB*[0] to WRITE*[H/L]	4, 8	1½T+7	2½T+36	1½T+6	2½T+37	1½T+5	2½T+41
B16	PAS*[0] & MWB*[0] & DS*[0] to DS1/0*[L]	4, 8	4½T+10	5½T+46	4½T+9	5½T+47	4½T+9	5½T+57
B17	PAS*[0] & MWB*[0] to SWDEN*[L]	4	7	36	4	12	3	14
B18	PAS*[0] & MWB*[0] to DE-NIN*[L]	4, 9	3	20	3	20	2	22
B19	PAS*[0] & MWB*[0] to DENIN1*[L]	4, 9	3	20	3	21	3	23
B20	PAS*[0] & MWB*[0] & DS*[0] to AS*[L]	4, 8	4½T+6	5½T+28	4½T+5	5½T+29	4½T+5	5½T+32
B21	R/W*[0] to DDIR[H]	4, 7	4	22	3	23	2	25
B22	R/W*[1] to DDIR[L]	4, 7	2	14	1	14	1	15
B23	D[7:0] to LD[7:0] Valid	4, 9	3	18	2	18	2	22
B24	DTACK*[0] to LEDI[H]	4, 9	3T+6	4T+28	3T+4	4T+29	3T+4	4T+32
B25	DTACK*[0] to DSACKi*[L]	4	4	30	3	31	3	36
B26	PAS*[1] & DS*[1] to DSACKi*[H]	4	2	19	2	20	2	27
B27	PAS*[1] to AS*[H]	4	6	30	5	31	5	41
B28	DS*[1] to ISOBE*[H]	4	4	23	3	24	3	26
B29	DS*[1] to SWDEN*[H]	4	4	10	3	10	2	13
B30	DS*[1] to DENIN1*[H]	4, 9	3	19	3	20	2	22
B31	DS*[1] to DENIN*[H]	4, 9	3	19	3	20	2	22
B32	DS*[1] to LD[7:0] Invalid	4, 9	3	20	2	22	2	28
B33	DS*[1] to LD[7:0] Hi-Z	4, 9	3	20	2	22	2	28
B34	DS*[0] to DSACKi*[L]	4, 10	6	T+30	5	T+32	5	T+35
B35	DS*[0] to LADO[H]	4, 10	8	38	7	39	7	43
B36	DS*[0] to LEDO[H]	4, 10	4	T+16	3	T+18	3	T+20

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
LOCAL BUS TIMING (VIC068A AS LOCAL BUS MASTER)								
C1	LBG*[0] to PAS*[L]	4	5T+6	6T+31	5T+5	6T+33	5T+5	6T+44
C2	LBG*[0] to LA[7:0] Valid	4	3T+8	4T+36	3T+7	4T+37	3T+6	4T+46
C3	LBG*[0] to SIZ[1:0] Valid	4	1T+3	2T+20	1T+3	2T+21	1T+2	2T+28
C4	LBG*[0] to FC[2:1] Valid	4	1T+3	2T+20	1T+3	2T+21	1T+2	2T+27
C5	LBG*[0] to LD[7:0] Driven	7	3T+8	4T+38	3T+7	4T+39	3T+7	4T+48
C6	LBG*[0] to LAEN[H]	4	3T+10	4T+43	3T+9	4T+44	3T+8	4T+48
C7	LBG*[0] to ISOBE*[L]	4	3T+8	4T+37	3T+7	4T+39	3T+7	4T+42
C8	LBG*[0] to SWDEN*[L]	4	3T+9	4T+39	3T+8	4T+41	3T+7	4T+45
C9	LBG*[0] to DDIR[H]	4, 7	3T+8	4T+37	3T+7	4T+39	3T+7	4T+42
C10	LBG*[0] to DENIN1*[L]	4, 7	3T+7	4T+36	3T+6	4T+38	3T+6	4T+42
C11	LBG*[0] to DENIN*[L]	4, 7	3T+7	4T+32	3T+6	4T+35	3T+5	4T+38
C12	LBG*[0] & DS1/0*[0] & WRITE*[0] to R/W*[L]	4, 7	3T+8	4T+38	3T+7	4T+40	3T+7	4T+47
C13	LBG*[0] & DS1/0*[0] to DS*[L]	4	5T+8	6T+39	5T+7	6T+42	5T+7	6T+56
C14	PAS*[0] to DS*[L]	4, 11	0	12	0	15	0	15
C15	LBR*[H] to LBG*[1]	4, 12		T		T		T
SLAVE ACCESSES								
D1	SLSELi*[0] & AS*[0] to LBR*[L]	4	7	35	6	36	6	40
D2	SLSELi*[0] & AS*[0] & DS1/0*[0] to LADI[H]	4	5	25	4	26	4	29
D3	LD[7:0] to D[7:0]	4, 9	2	16	2	16	2	18
D4	DSACKI*[0] to LEDO[H]	4, 9	SAT+8	SAT+½T +35	SAT+7	SAT+½T +36	SAT+6	SAT+½T +39
D5	DSACKi*[0] to DTACK*[L]	4	SAT+10	SAT+½T +45	SAT+9	SAT+½T +47	SAT+9	SAT+½T +53
D6	DS1/0*[0] to DTACK*[L]	4, 13	2T+5	3½T+28	2T+5	3½T+29	2T+4	3½T+33
D7	DS1/0*[0] to LEDI[H]	4, 13	9	41	8	43	8	47
D8	AS*[1] to LA[7:0], R/W* Invalid	4	5	38	4	42	4	55
D9	AS*[1] to LA[7:0], R/W* High-Z	4	5	38	4	42	4	55
D10	AS*[1] to FC2/1, Invalid	4	10	42	8	44	8	56
D11	AS*[1] & DSACKi*[1] to FC2/1, High-Z	4	10	42	8	44	8	56
D12	AS*[1] to SIZ1/0, Invalid	4	7	32	6	34	6	37
D13	AS*[1] & DSACKi*[1] to SIZ1/0, High-Z	4	3	1T+17	2	1T+19	2	1T+24
D14	AS*[1] to ISOBE*[H]	4	6	30	5	31	5	34
D15	AS*[1] to SWDEN*[H]	4	4	24	4	25	3	27
D16	AS*[1] to DENIN1*[H]	4, 7	5	27	4	28	4	30
D17	AS*[1] to DENIN*[H]	4, 7	5	27	4	28	4	30
D18	AS*[1] & DSACKi*[1] to LBR*[H]	4	5	26	4	27	4	30

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
D19	AS*[1] to LAEN[L]	4	9	40	8	43	7	56
D20	DS*1/0[1] to LD[7:0] Invalid	4, 7	2	27	2	30	2	39
D21	DS*1/0[1] to LD[7:0] High-Z	4, 7	2	27	2	30	2	39
D22	DSACKi*[0] to PAS*[H]	4	SAT+10	SAT+½T +44	SAT+9	SAT+½T +46	SAT+8	SAT+½T +56
D23	DSACKi*[0] to DS*[H]	4	SAT+9	SAT+½T +40	SAT+8	SAT+½T +41	SAT+7	SAT+½T +48
D24	DSi*[1] to DTACK*[H]	4	3	27	3	28	3	35
INTERRUPT								
E1	IACKIN*[0] to IACKOUT*[L]	4	2	16	2	17	2	18
E2	IACKIN*[1] to IACKOUT*[H]	4	3	18	2	19	2	20
E3	FCIACK*[0] & PAS*[0] to BRi*[L]	4	5T+9	6T+41	5T+8	6T+42	5T+7	6T+48
E4	FCIACK*[0] & PAS*[0] to IACK*[L]	4, 8	7½T+7	8½T+34	7½T+6	8½T+35	7½T+6	8½T+39
E5	FCIACK*[0] & PAS*[0] to LD[7:0] Driven	4, 14	5T+12	6T+50	5T+10	6T+52	5T+10	6T+57
E6	FCIACK*[0] & PAS*[0] to LD[7:0] valid	4, 15	9T+5	10T+29	9T+5	10T+33	9T+4	10T+37
E7	FCIACK*[0] & PAS*[0] to LIACK0*[L]	4, 16	5T+7	6T+32	5T+6	6T+33	5T+5	6T+36
E8	IRQi*[0] to IPL	4	5	33	5	34	4	37
E9	BGiIN*[0] to BBSY*[L]	4	7	32	5	33	5	36
E10	BGiIN*[0] to AS*[L]	4	3T+5	4T+27	3T+4	4T+28	3T+4	4T+31
E11	BGiIN*[0] to DS1/0*[L]	4	3T+10	4T+45	3T+9	4T+46	3T+8	4T+55
E12	BGiIN*[0] to IACK*[L]	4, 15	39	7	40	7	44	
E13	PAS*[0] to ISOBE*[L]	4	5T+9	6T+39	5T+7	6T+40	5T+7	6T+44
E14	PAS*[0] to SWDEN*[L]	4	5T+8	6T+37	5T+7	6T+38	5T+6	6T+42
E15	IPLi to IPLi	4, 11		10		12		12
MASTER BLOCK TRANSFER WITH LOCAL DMA (INITIATION CYCLE)								
F1	MWB*[0] & PAS*[0] & DS*[0] to BRi*[L]	4	T+7	2T+32	T+6	2T+33	T+5	2T+38
F2	BGiIN*[0] to LBR*[L]	4	4T+10	5T+42	4T+8	5T+44	4T+8	5T+50
F3	MWB*[0] & PAS*[0] & DS*[0] to LBR*[L]	4, 8	5T+10	6T+42	5T+8	6T+44	5T+8	6T+50
F4	MWB*[0] & PAS*[0] & DS*[0] to LADO[H]	4	T+7	2T+35	T+6	2T+36	T+6	2T+39
F5	MWB*[0] & PAS*[0] & DS*[0] to BLT*[L]	4	T+6	2T+28	T+5	2T+29	T+4	2T+37
MASTER BLOCK TRANSFER WITH LOCAL DMA (WRITE)								
** First cycle **								
G1	DSACKi*[0] and DS*[0] to DS*[H]	4	MBAT0+9	MBAT0+⁄T+41	MBAT0+8	MBAT0+½T+43	MBAT0+7	MBAT0+½T+52

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
G2	DSACKi*[0] and DS*[L] to LEDO[H]	4	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	4	MBAT0+T+11	MBAT0+1½T+32	MBAT0+T+9	MBAT0+1½T+36	MBAT0+T+9	MBAT0+1½T+40
G4	DSACKi*[0] and DS*[L] to DSi*[L]	4	MBAT0+3T+6	MBAT0+3½T+37	MBAT0+3T+5	MBAT0+3½T+39	MBAT0+3T+5	MBAT0+3½T+42
G5	DTACK*[0] to LEDO[L]	4	7	32	6	33	6	38
G6	DTACK*[0] to DSi*[H]	4	10	49	9	51	9	56
G7	DTACK*[0] to A[7:0] Valid	4	11	46	10	48	9	64
G8	DS*[H] to DS*[L]	4	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]	4	MBAT1+9	MBAT1+½T+41	MBAT1+8	MBAT1+½T+43	MBAT1+7	MBAT1+½T+52
G10	DSACKi*[0] and DS*[L] to LEDO[H]	4	MBAT1+8	MBAT1+½T+36	MBAT1+7	MBAT1+½T+37	MBAT1+6	MBAT1+½T+40
G11	DSACKi*[0] and DS*[L] to LA[7:0] Valid	4	MBAT1+T+11	MBAT1+1½T+32	MBAT1+T+9	MBAT1+1½T+36	MBAT1+T+9	MBAT1+1½T+40
G12	DSACKi*[0] and DS*[L] to DSi*[L]	4	MBAT1+3T+6	MBAT1+3⁄T+29	MBAT1+3T+5	MBAT1+3⁄T+30	MBAT1+3T+5	MBAT1+3⁄T+38
G13	DTACK*[0] to LEDO[L]	4	7	32	6	33	6	38
G14	DTACK*[0] to DSi*[H]	4	10	45	9	46	9	59
G15	DTACK*[0] to A[7:0] Valid	4	11	46	10	48	9	64
G16	DTACK*[0] to DS*[H]	4, 17	T + 15	1⁄T + 45	1⁄T + 14	1⁄T + 46	T + 13	1⁄T + 47
G17	LEDO[L] to LEDO[H]	4, 17	T + 11	1⁄T + 25	1⁄T + 10	1⁄T + 26	T + 9	1⁄T + 27
MASTER BLOCK TRANSFER WITH LOCAL DMA (READ)								
** First Cycle **								
H1	DTACK*[0] to LEDI[H]		2T+6	3T+23	2T+4	3T+25	2T+4	3T+27
H2	DTACK*[0] to DSi*[H]		2T+9	3T+28	2T+8	3T+30	2T+7	3T+32
H3	DTACK*[0] to A[7:0] Valid	4	1½T+10	2½T+44	1½T+9	2½T+45	1½T+8	2½T+53
H4	DTACK*[0] to DS*[L]	4	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+38	MBAT0+8	MBAT0+½T+40	MBAT0+7	MBAT0+½T+45
H6	DSACKi*[0] and DS*[L] to LEDI[L]		MBAT0+9	MBAT0+½T+48	MBAT0+8	MBAT0+½T+50	MBAT0+7	MBAT0+½T+55
H7	DSACKi*[0] and DS*[L] to LA[7:0] Valid	4	MBAT0+T+11	MBAT0+1½T+30	MBAT0+T+9	MBAT0+1½T+32	MBAT0+T+9	MBAT0+1½T+35
H8	DSACKi*[0] and DS*[L] to DSi*[L]	4	MBAT0+11	MBAT0+½T+74	MBAT0+10	MBAT0+½T+76	MBAT0+10	MBAT0+½T+83
** Second and subsequent cycles **								
H9	DTACK*[0] to LEDI[H]	4	2T+6	3T+23	2T+4	3T+25	2T+4	3T+27
H10	DTACK*[0] to DSi*[H]	4	2T+9	3T+28	2T+8	3T+30	2T+7	3T+32
H11	DTACK*[0] to A[7:0] Valid	4	10	44	9	45	8	53
H12	DTACK*[0] to DS*[L]	4	8	38	7	40	7	47
H13	DSACKi*[0] and DS*[0] to DS*[H]	4	MBAT1+9	MBAT1+½T+36	MBAT1+8	MBAT1+½T+38	MBAT1+7	MBAT1+½T+45

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
H14	DSACKi*[0] and DS*[0] to LEDI[L]	4	MBAT1+9	MBAT1+½T+44	MBAT1+8	MBAT1+½T+46	MBAT1+7	MBAT1+½T+55
H15	DSACKi*[0] and DS*[0] to LA[7:0] Valid	4	MBAT1+T+11	MBAT1+1½T+31	MBAT1+T+9	MBAT1+1½T+33	MBAT1+T+9	MBAT1+1½T+35
H16	DSACKi*[0] and DS*[0] to DSi*[L]	4	MBAT1+11	MBAT1+½T+72	MBAT1+10	MBAT1+½T+76	MBAT1+10	MBAT1+½T+83
MASTER BLOCK TRANSFER WITH LOCAL DMA (BOUNDARY CROSSING)								
J1	DS*[L] to BLT*[H]		2	30	2	32	2	35
J2	DS*[H] to BLT*[L]		2	17	2	19	2	21
J3	DSi*[L] to LEDO[H/L]		2	21	2	23	2	25
J4	DSi*[H] to LADO[L/H]	4	1	16	2	18	2	20
SLAVE BLOCK TRANSFER (WRITE)								
** First Cycle **								
See: Local Bus Timing (VIC068A as local bus master)								
** Second and subsequent cycles **								
K1	DSi*[0] to LEDI[H]	4	4	20	4	22	4	24
K2	DSi*[0] to DS*[L]		6	35	5	36	5	39
K3	DSACKi*[0] and DS*[L] to DS*[H]	4	SBAT+9	SBAT+½T+41	SBAT+8	SBAT+½T+42	SBAT+7	SBAT+½T+52
K4	DSACKi*[0] and DS*[L] to DTACK*[L]	4	SBAT+12	SBAT+½T+51	SBAT+11	SBAT+½T+53	SBAT+10	SBAT+½T+67
K5	DSACKi*[0] and DS*[L] to ISOBE*[H]	4	SBAT+13	SBAT+½T+54	SBAT+11	SBAT+½T+56	SBAT+11	SBAT+½T+62
K6	DSACKi*[0] and DS*[L] to SWDEN*[H]	4	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] Invalid	4	SBAT+T+10	SBAT+1½T+34	SBAT+T+8	SBAT+1½T+36	SBAT+T+8	SBAT+1½T+40
K8	DSACKi*[0] and DS*[L] to LEDI*[L]	4	SBAT+8	SBAT+½T+46	SBAT+7	SBAT+½T+48	SBAT+6	SBAT+½T+53
K9	DS1/0*[1] to DTACK*[H]	4	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]	4	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]	4	3	20	3	22	3	24
L4	DSACKi*[0] and DS*[0] to LEDO[H]	4	SBAT+8	SBAT+½T+36	SBAT+7	SBAT+½T+37	SBAT+6	SBAT+½T+41
L5	DSACKi*[0] and DS*[0] to DS*[H]	4	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+47	SBAT+9	SBAT+½T+48	SBAT+9	SBAT+½T+53
L7	DSACKi*[0] and DS*[0] to LA[7:0] Invalid	4	SBAT+T+9	SBAT+1½T+34	SBAT+T+8	SBAT+1½T+36	SBAT+T+8	SBAT+½T+40
L8	DS1/0*[1] to DENO*[H]	4	3	19	3	20	2	22

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
L9	DS1/0*[1] to DTACK*[H]	4	3	20	3	21	3	24
L10	LEDO[L] to LEDO[H]	4, 18	T + 11	1½ + 25	T + 10	1½ + 26	1½ + 9	1½ + 27
L11	DTACK*[0] to DS*[H]	4, 18	T + 15	1½ + 45	1½ + 14	1½ + 46	1½ + 13	1½ + 47
REGISTER ACCESS								
M1	PAS*[0] & DS*[0] & CS*[0] to DSACKi*[L]	4	4T+5	5T+34	4T+5	5T+35	4T+4	5T+38
M2	PAS*[0] & DS*[0] & CS*[0] to LD[7:0] Valid	4, 9	3T+5	4T+28	3T+5	4T+29	3T+4	4T+37
M3	AS*[0] & ICFSEL*[0] to DTACK*[L]	4	4T+6	4T+30	4T+5	4T+31	4T+5	4T+34
RESET								
N1	LBG*[0] to HALT*[L], RESET*[L]	4	8	36	7	37	6	48
N2	IRESET*[0] to LBR*[L]	4	6	29	5	30	5	33
N3	IRESET*[0] to IPL0[Z]	4	2	16	2	16	2	20
SET-UP TIMES								
P1	LA, ASIZ[1:0] Valid to PAS*[0]	4	–2T		–2T		–2T	
P2	SIZ[1:0], WORD*, FC[2:1] Valid to PAS*[0]	4	–2T		–2T		–2T	
P3	LD[7:0] Valid to DS*[0]	4	0		0		0	
HOLD TIMES								
Q1	PAS*[1] to LA, ASIZ[1:0] Invalid	4	0		0		0	
Q2	PAS*[1] to SIZ[1:0], WORD*, FC[2:1] Invalid	4	0		0		0	
Q3	DS*[1] to LD[7:0] Invalid	4	0		0		0	
Q4	DS1/0*[1] to DTACK*[H]	4	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. Timing specified but not tested.
5. While VMEbus system controller.
6. Synchronous delay depends on speed in which BGiIN* is returned. If BGiIN* is returned in zero time after request, synchronous delay will be maximum.
7. Write operation only.
8. While VMEbus master.
9. Read operation only.
10. Master write post only.
11. Skew.
12. Input requirement.
13. Slave write post only.
14. VMEbus interrupt only.
15. Local interrupt (LICR[4] = 1) only.
16. Local interrupt (LICR[4] = 0) only.
17. “Slow” Slave.
18. “Slow” Master.

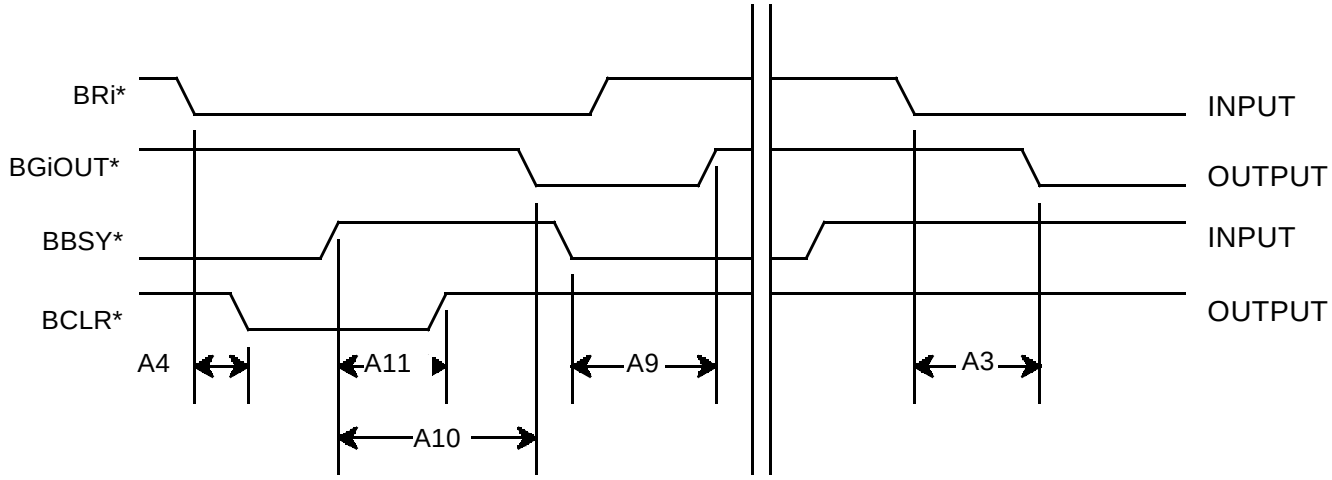


Figure 1-21. VMEbus Arbitration—VIC068A as Arbitor, priority interrupt

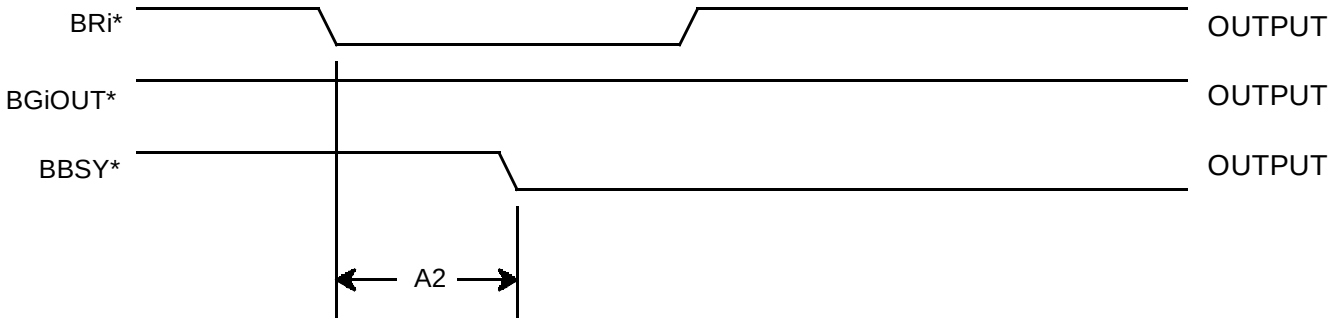


Figure 1-22. VMEbus Arbitration—VIC068A as System Controller (granting bus internally)

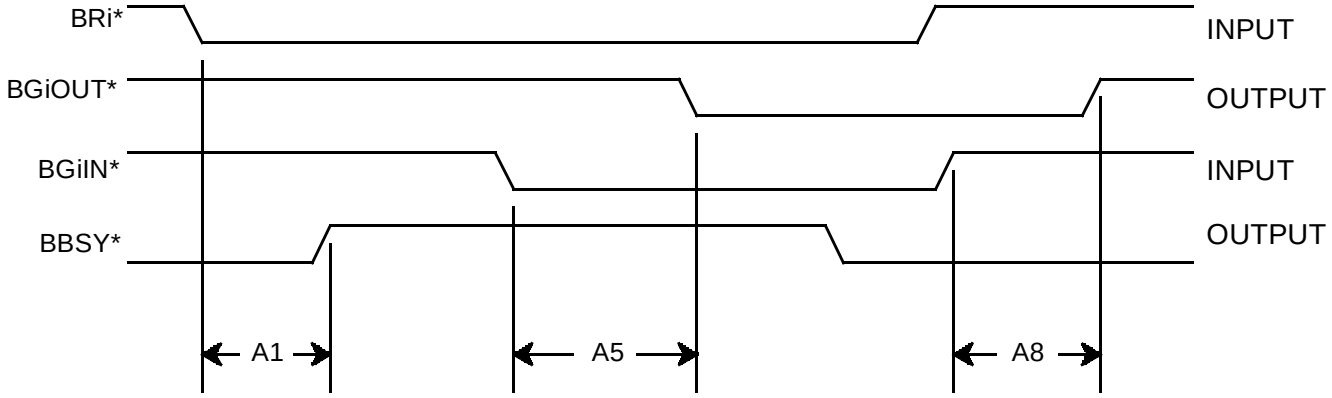


Figure 1-23. VMEbus Arbitration—VIC068A (not System Controller) Honoring ROR

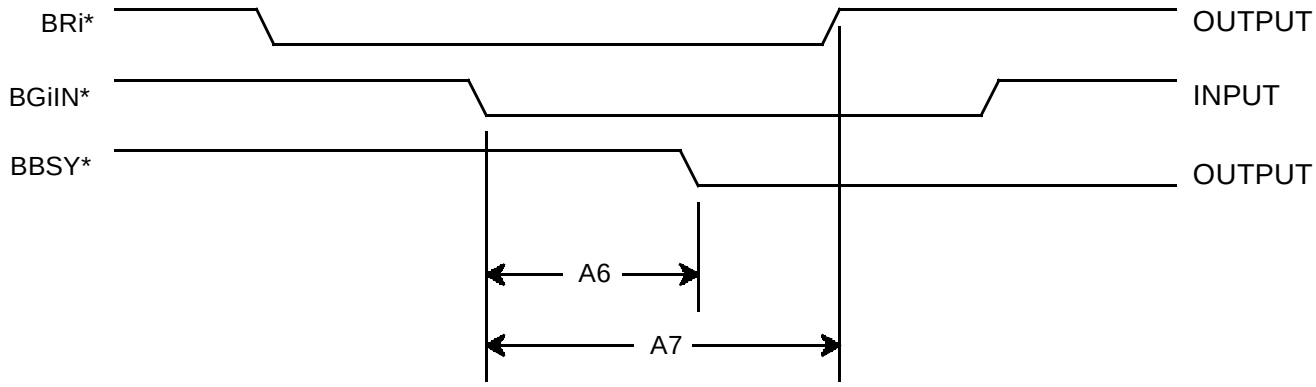


Figure 1-24. VMEbus Arbitration—VIC068A (not System Controller) Taking the VMEbus

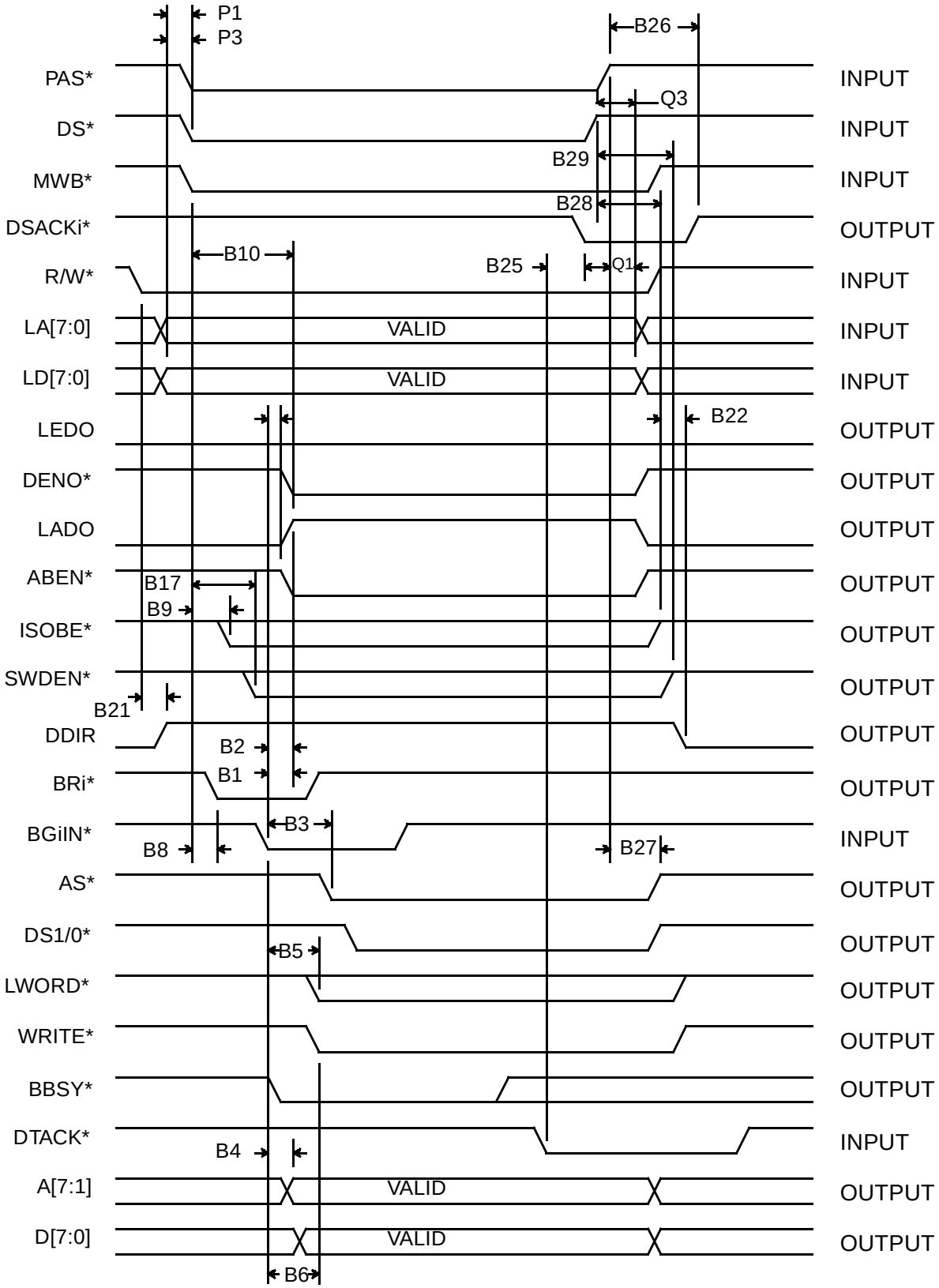


Figure 1-25. Master Write

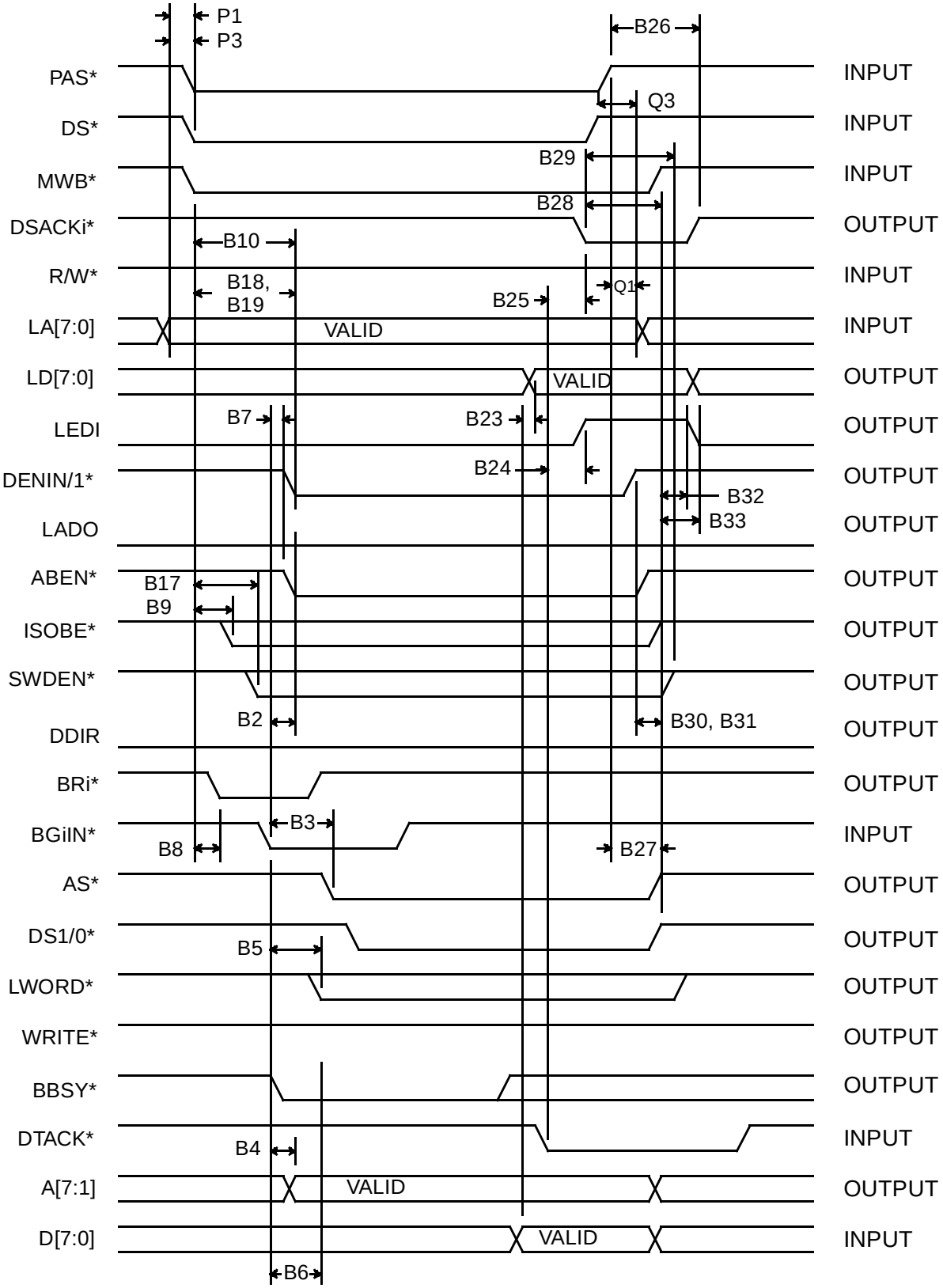


Figure 1-26. Master Read

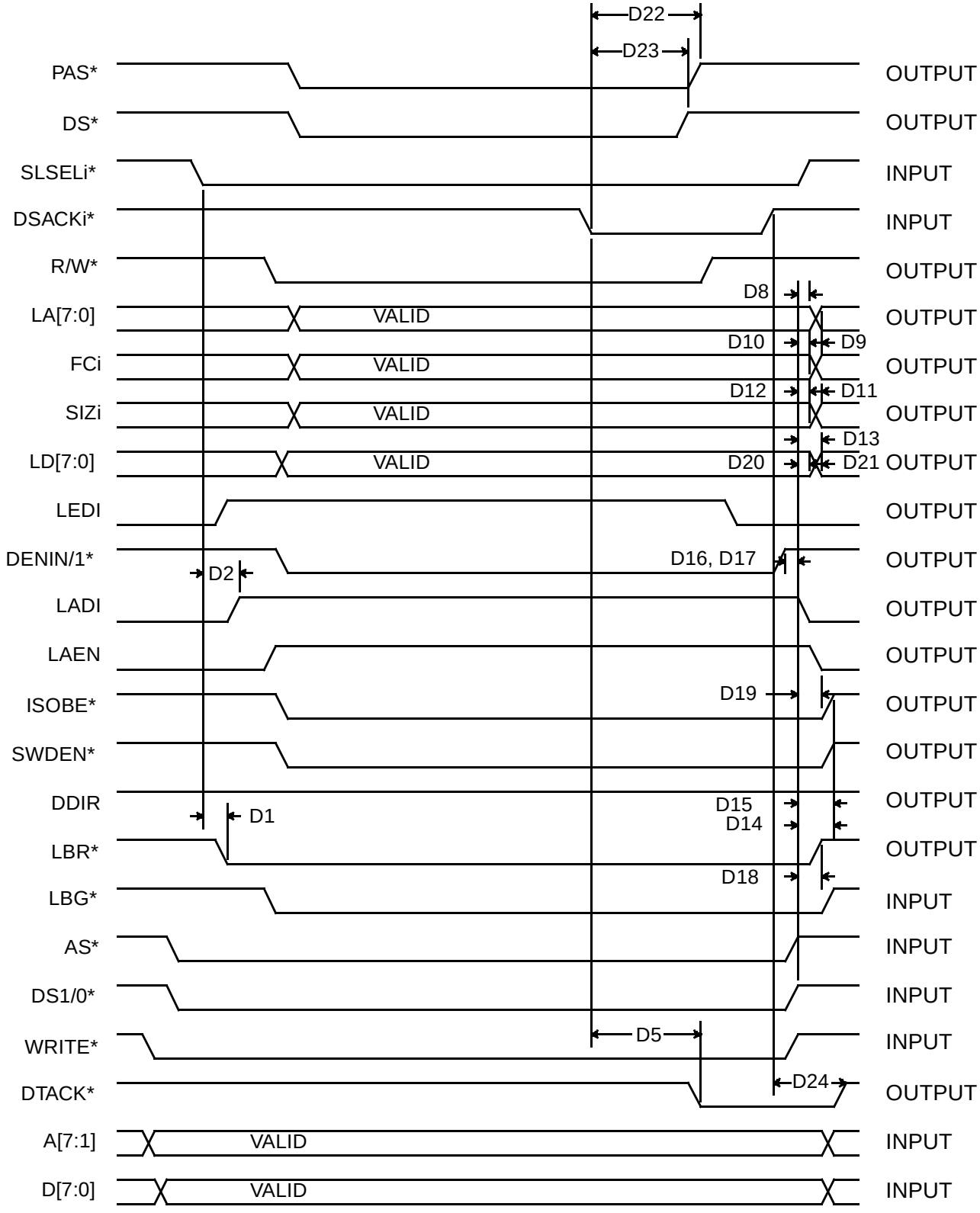


Figure 1-27. Slave Write

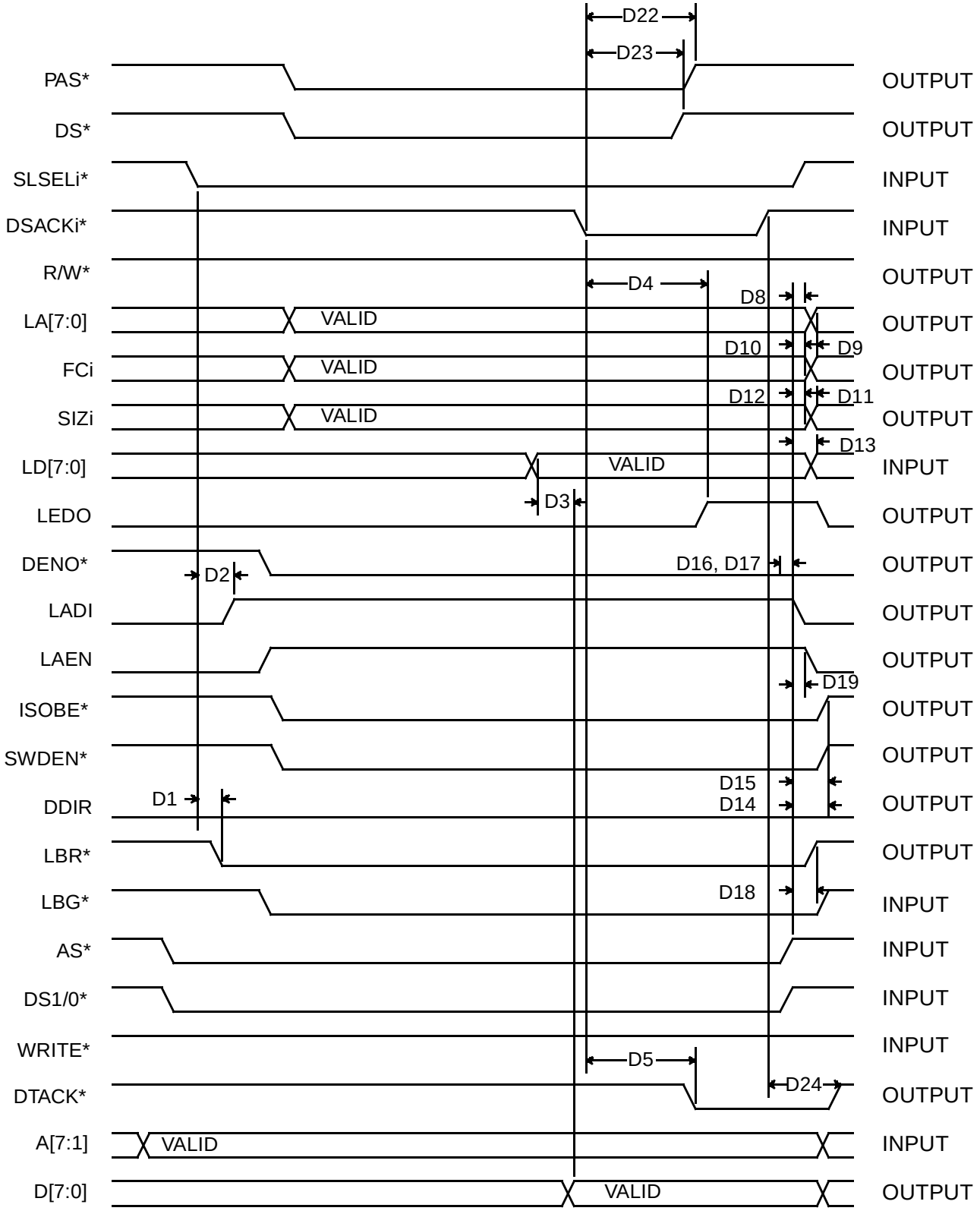


Figure 1-28. Slave Read

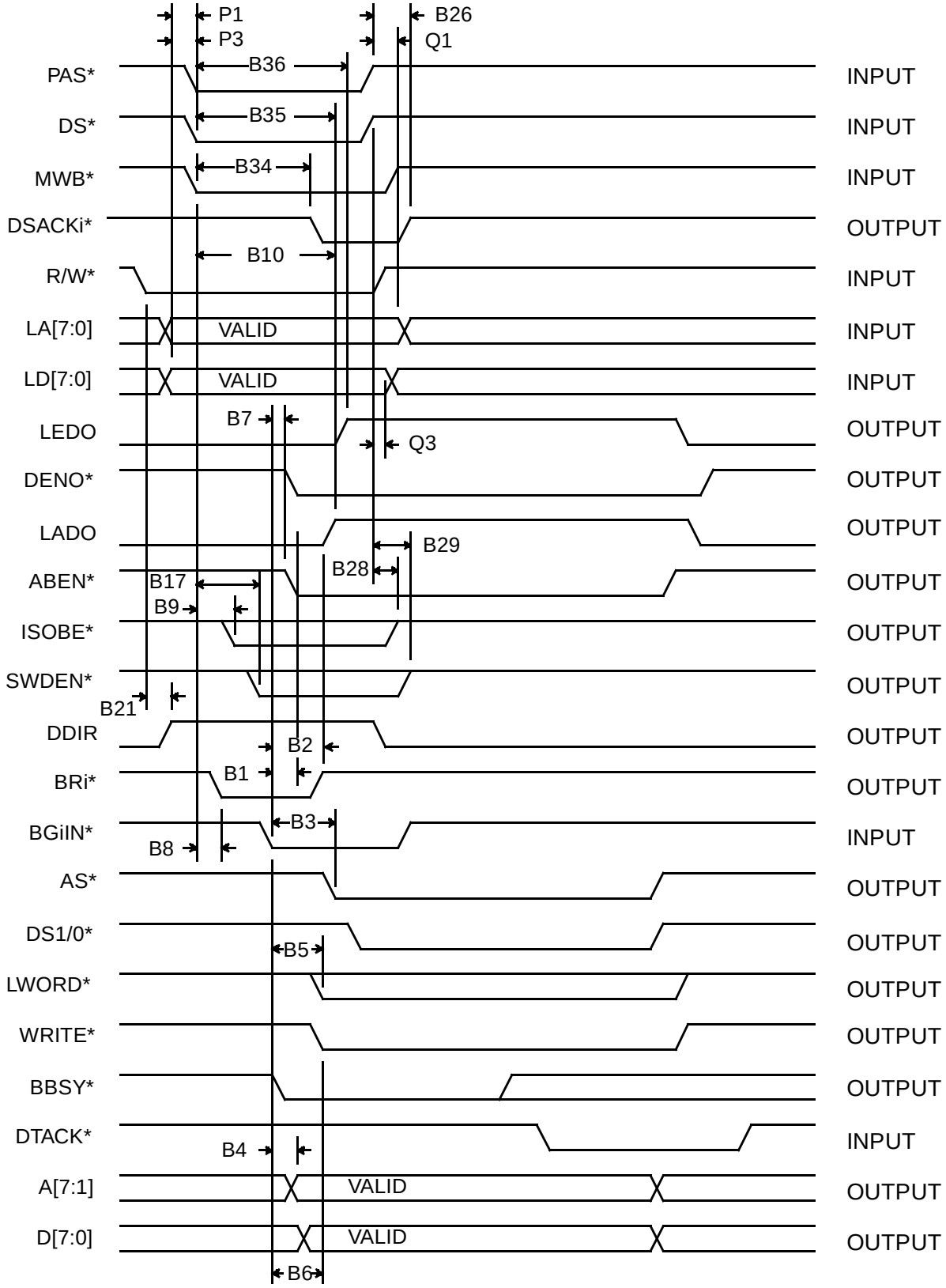


Figure 1-29. Master Write Post

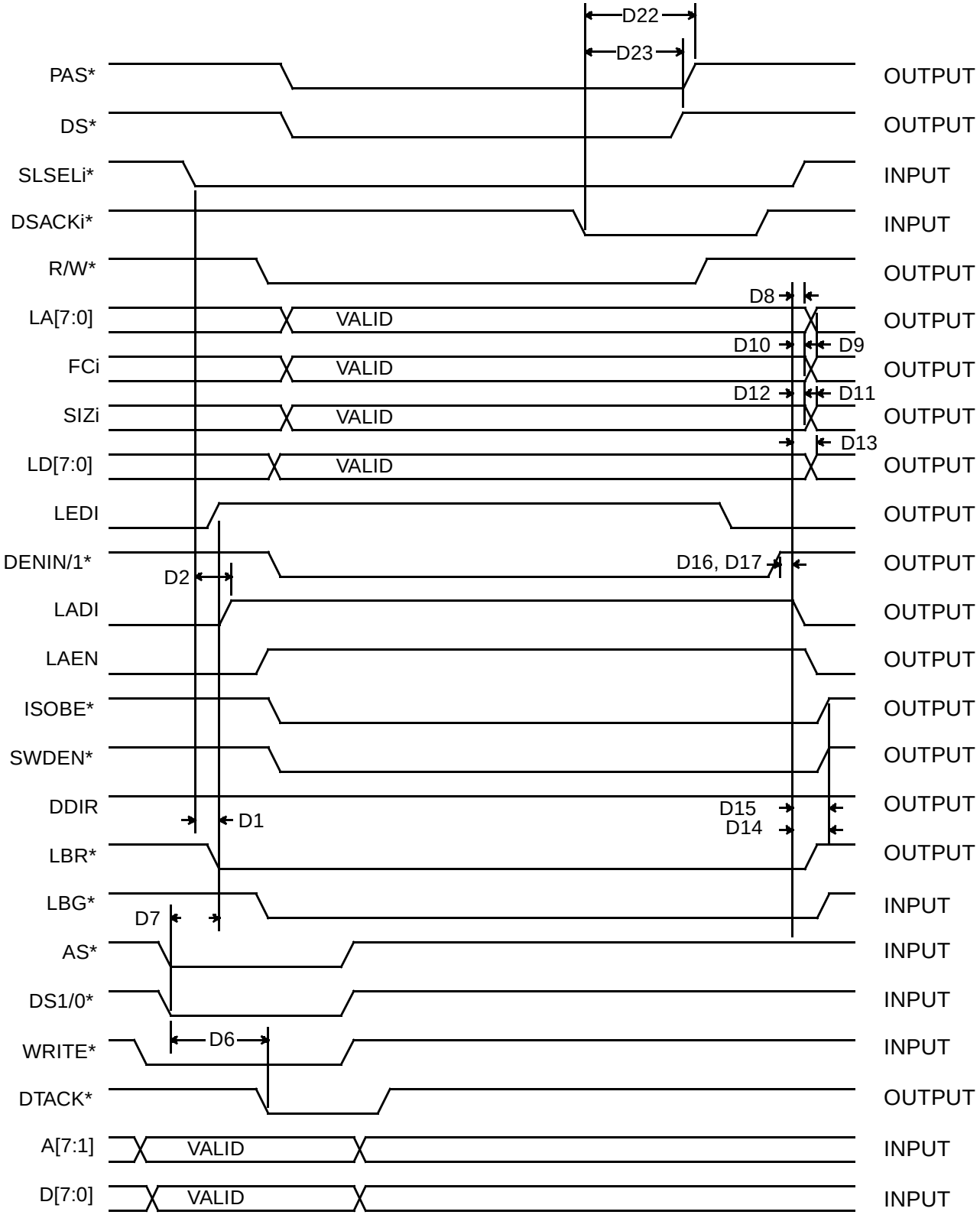


Figure 1-30. Slave Write Post

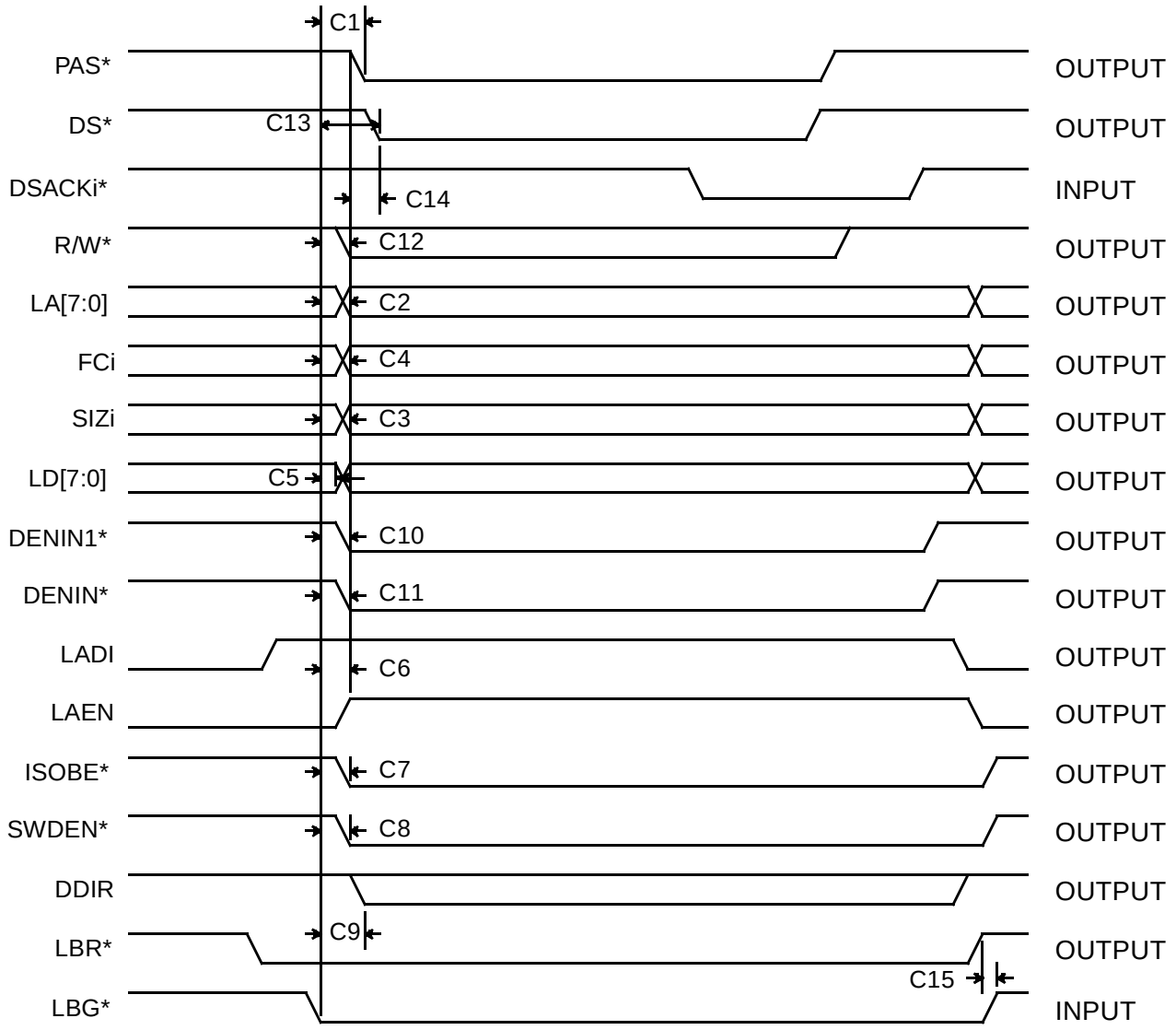


Figure 1-31. Local Bus

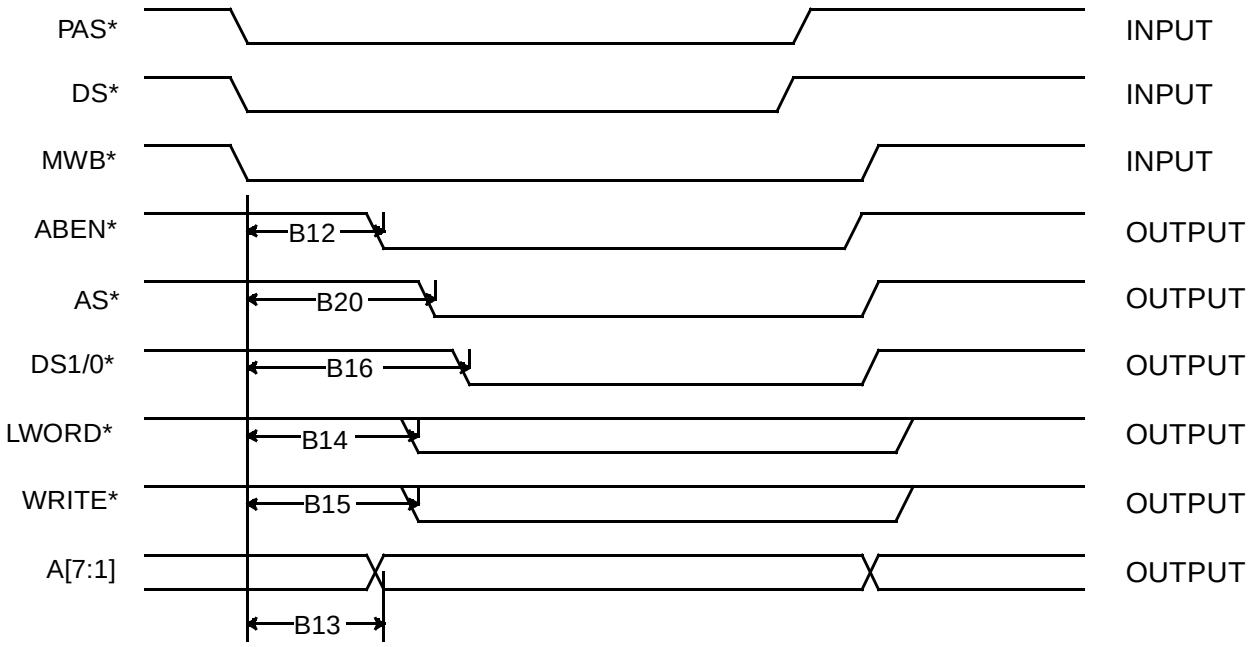


Figure 1-32. While VME Master

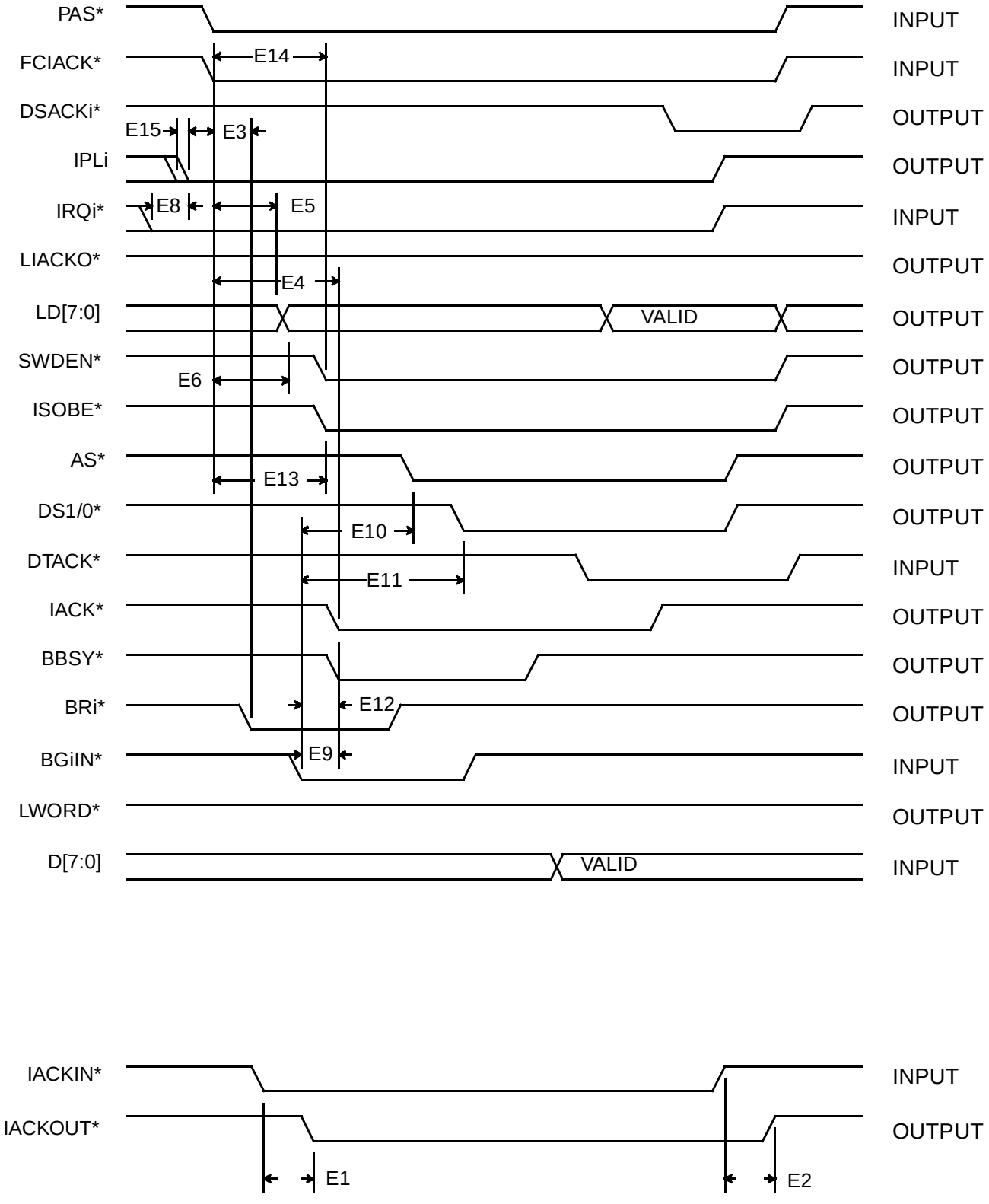


Figure 1-33. VME IACK

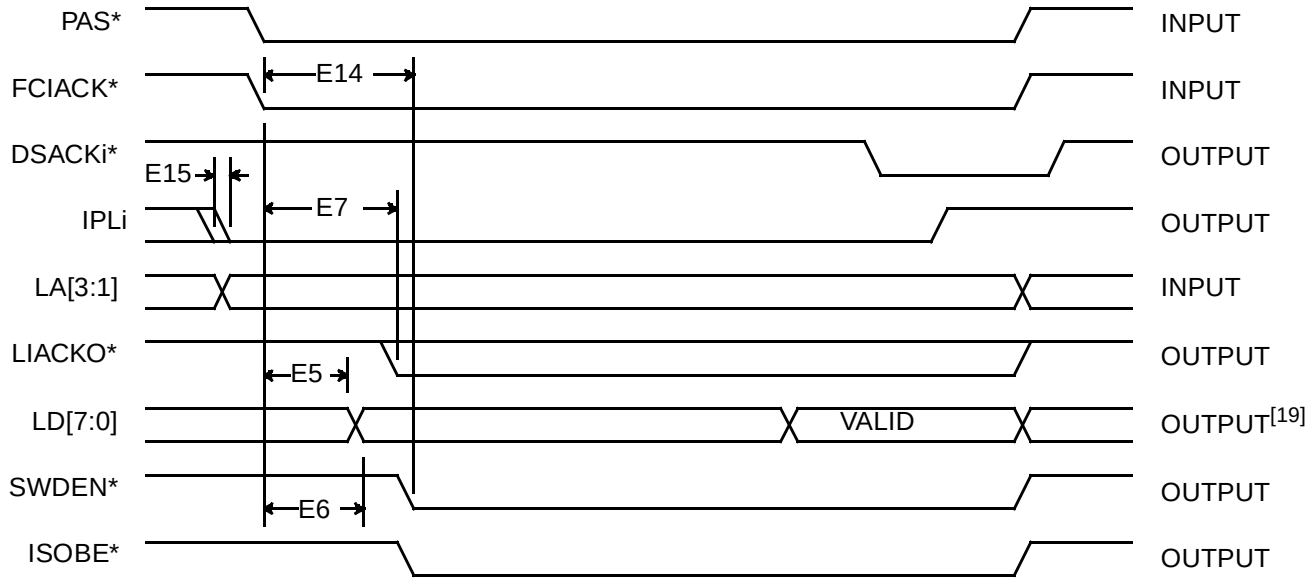


Figure 1-34. Local IACK

Note:

19. If VIC068A is configured to supply vector.

Initiation

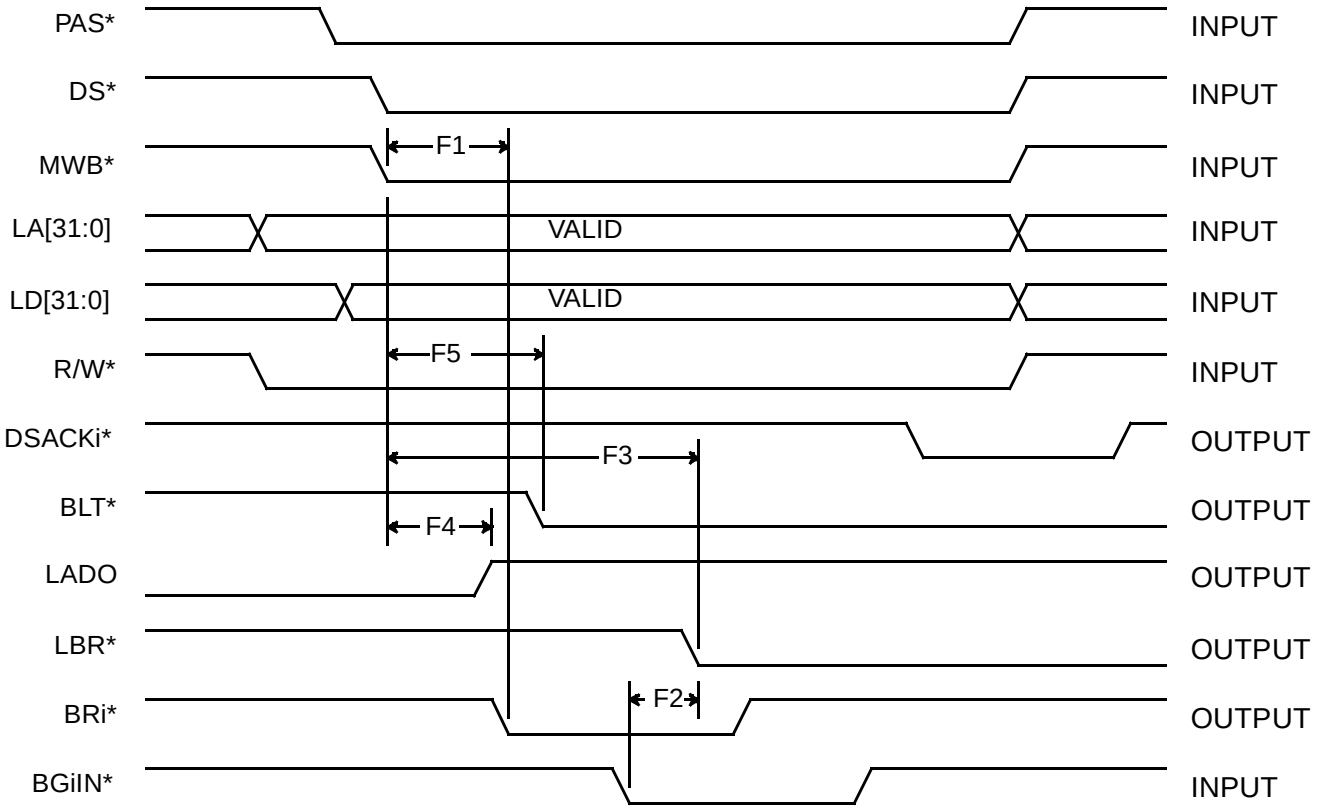


Figure 1-35. Initiation

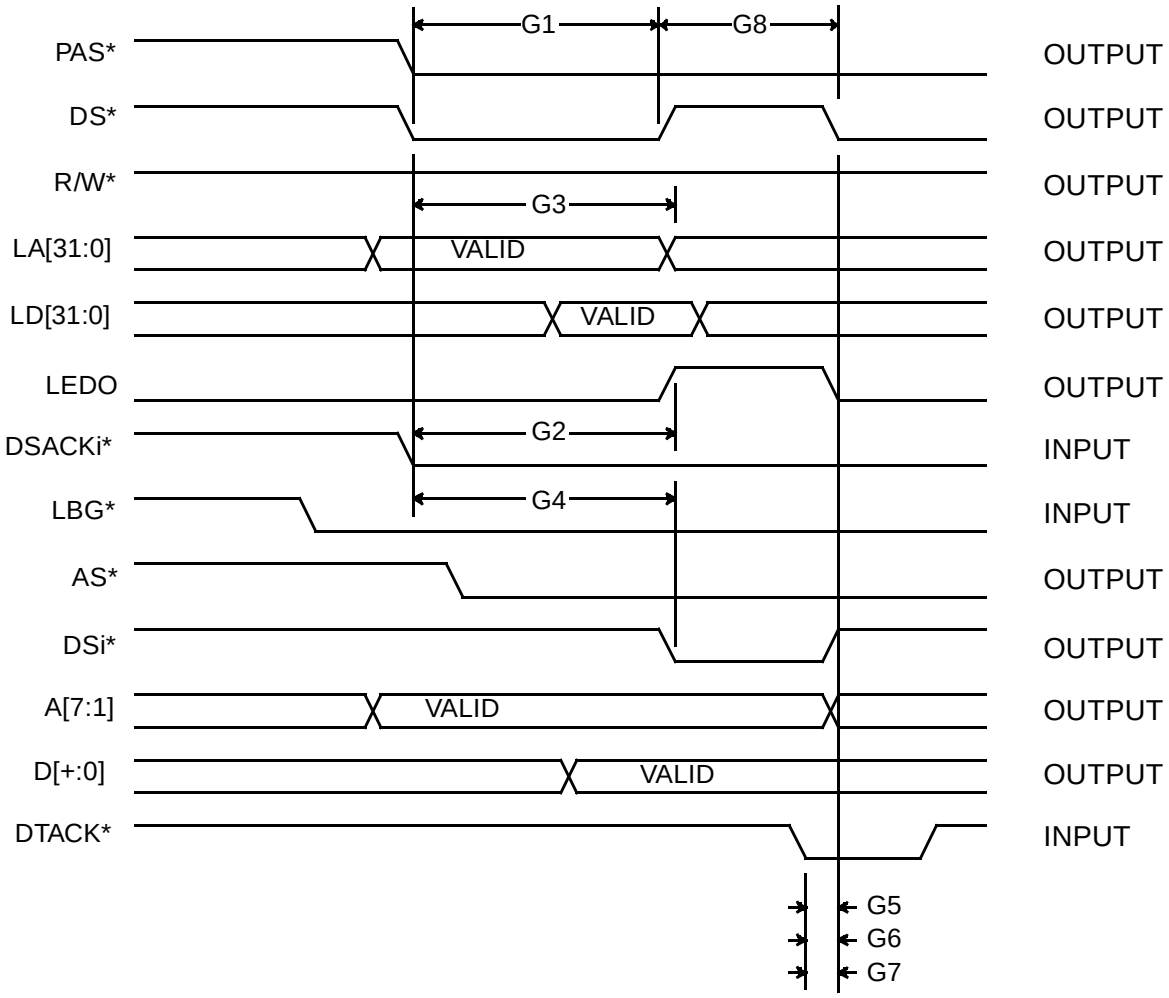


Figure 1-36. First MBLT Write

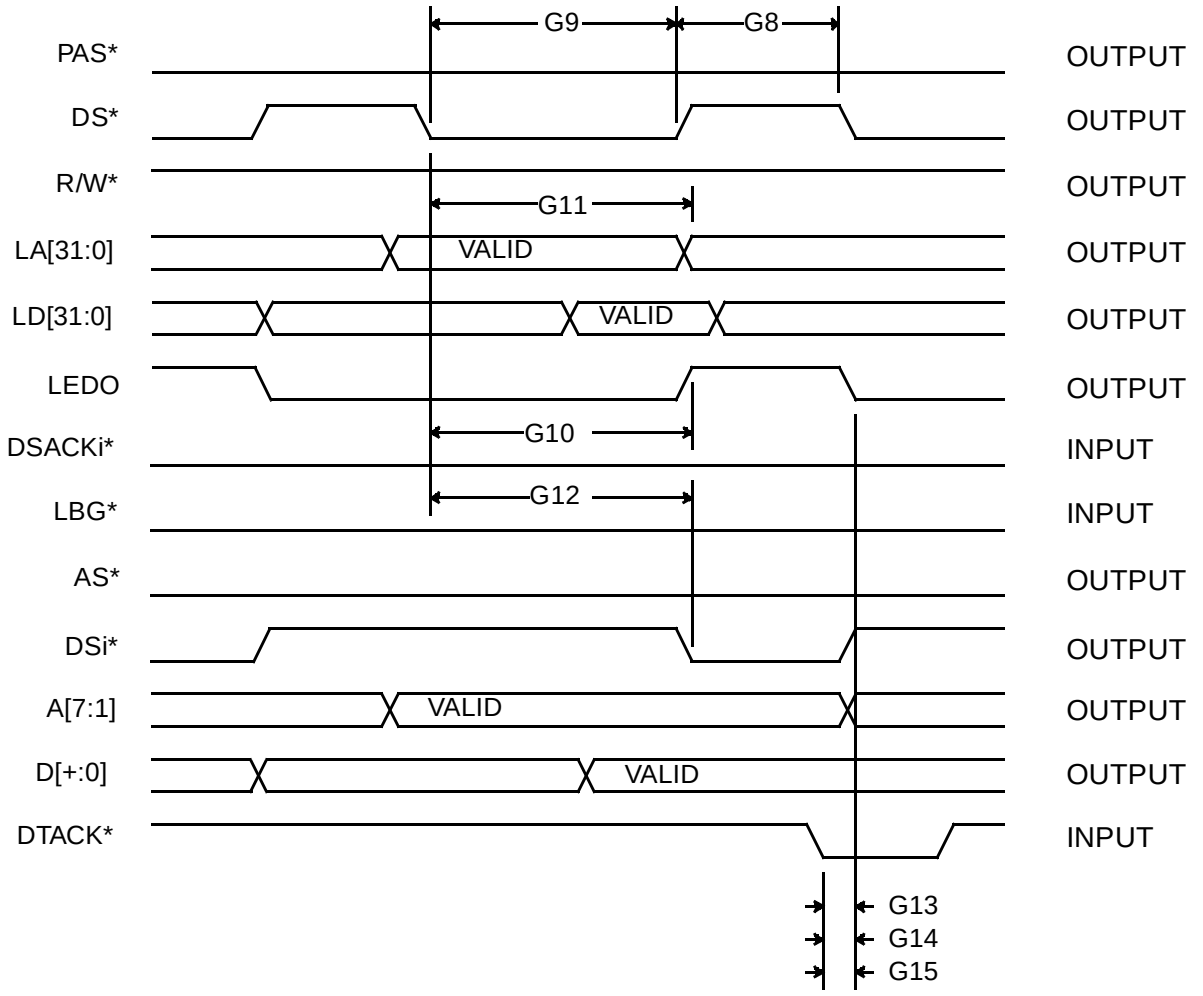


Figure 1-37. Second MBLT Write

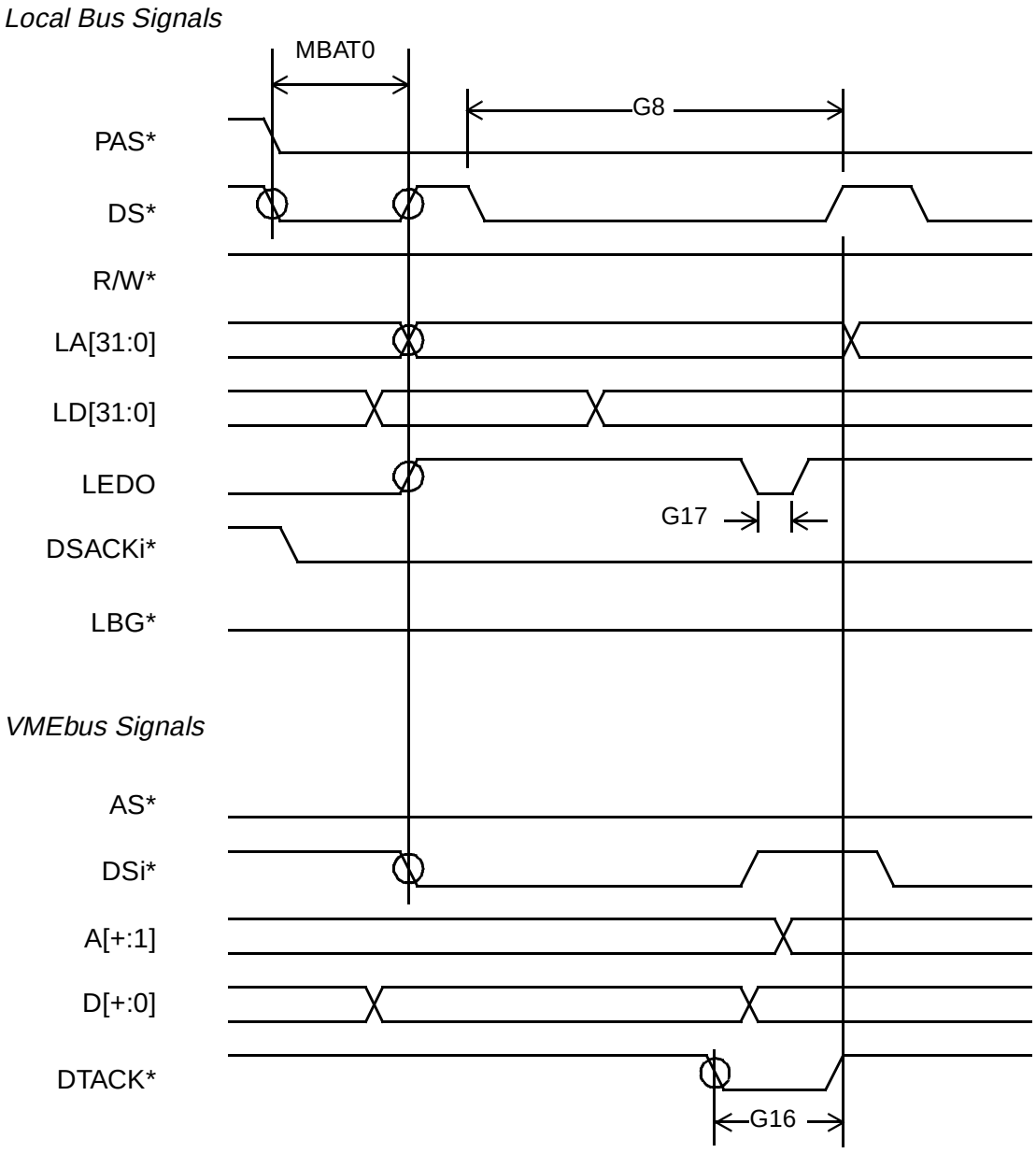


Figure 1-38. Master Block Transfer—Write (Slow Slave)

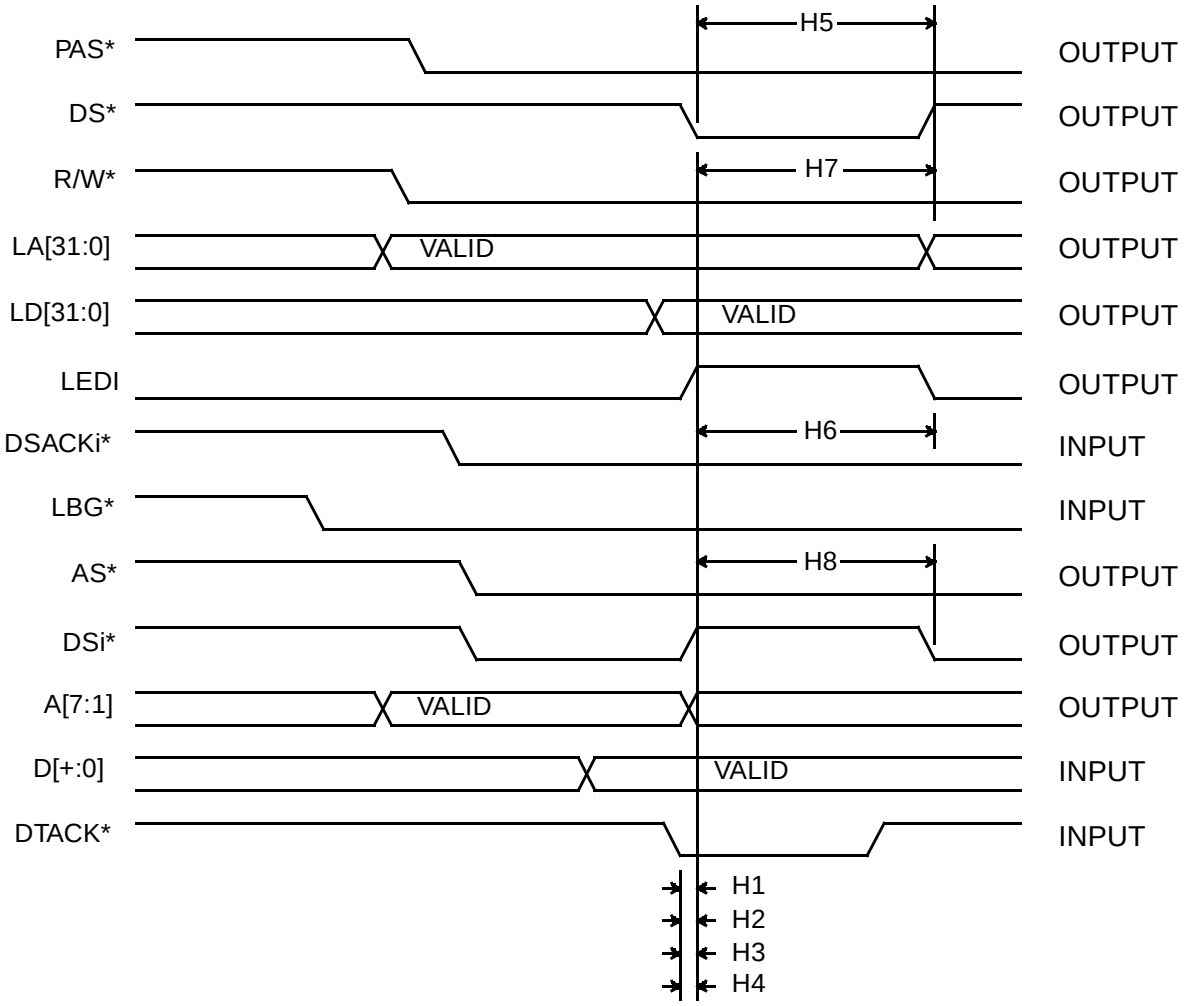


Figure 1-39. First MBLT Read

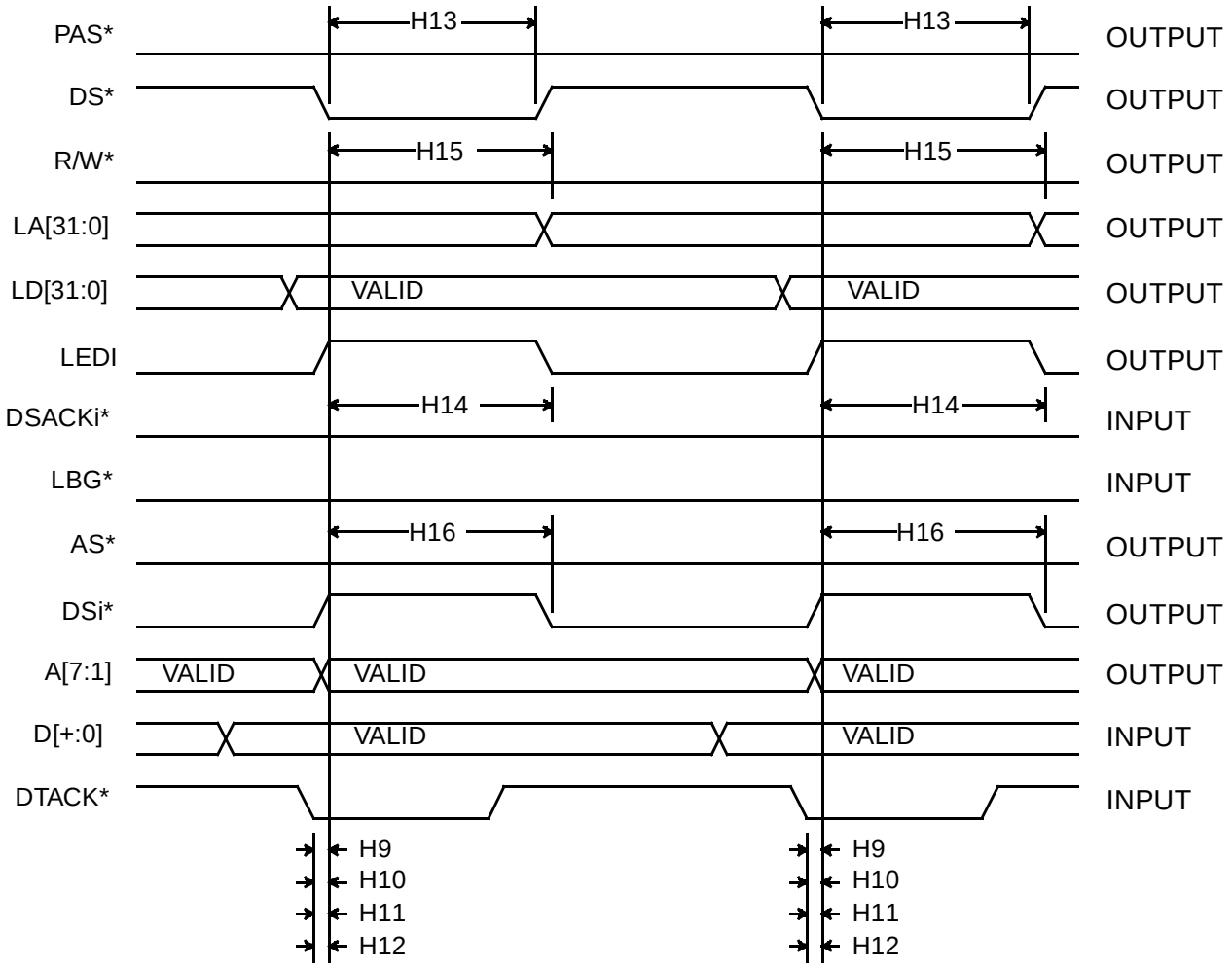
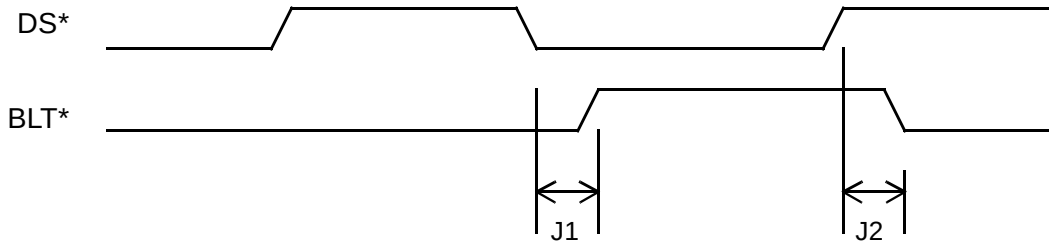


Figure 1-40. Second MBLT Read

Local Boundary Crossing



VMEbus Boundary Crossing

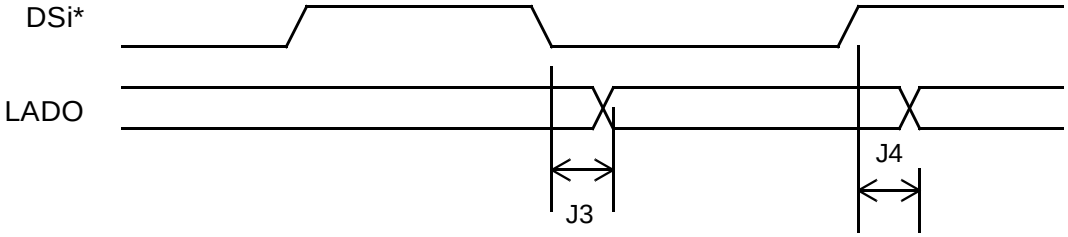


Figure 1-41. Boundary Crossing

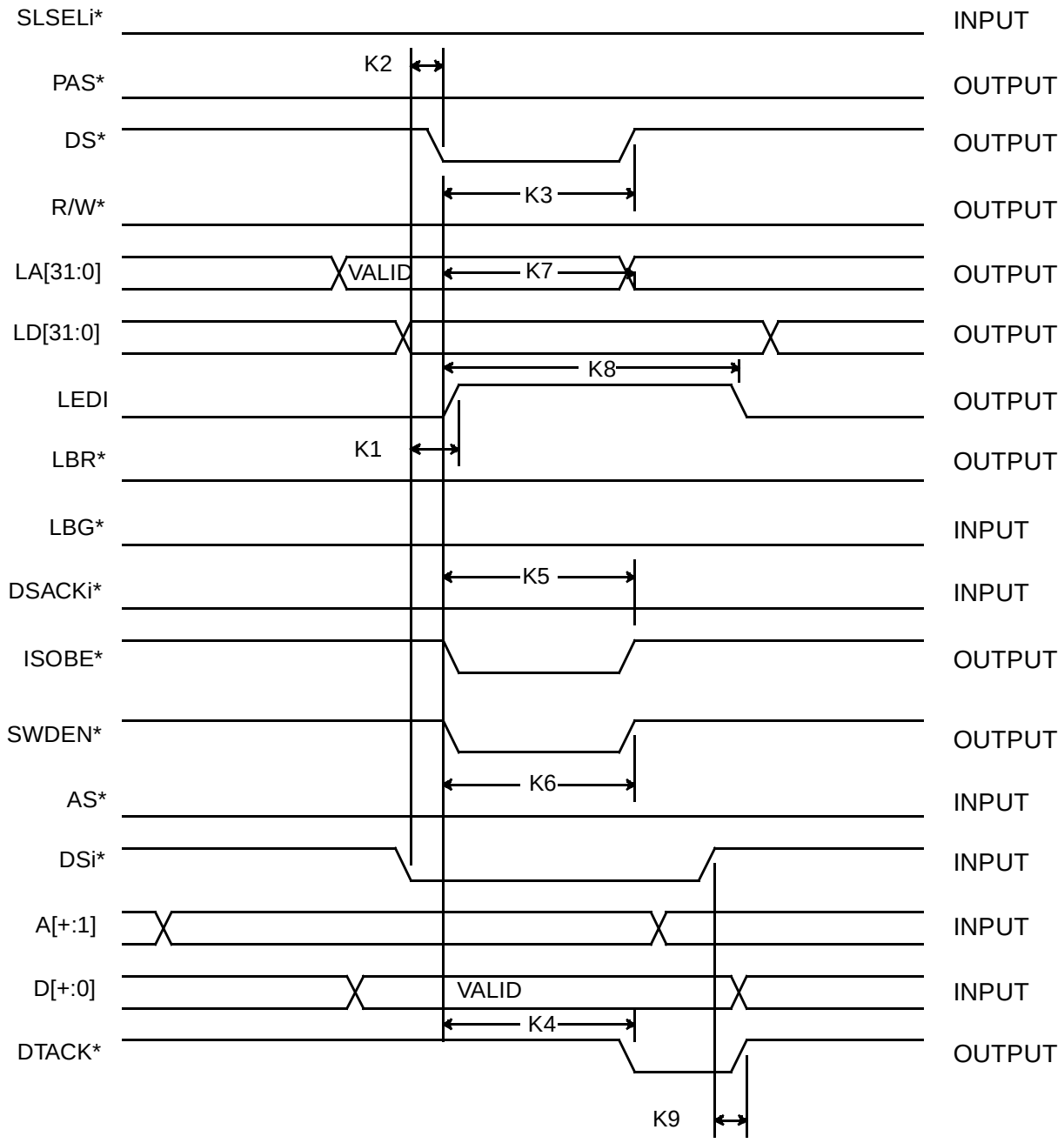


Figure 1-42. Slave Write BLT

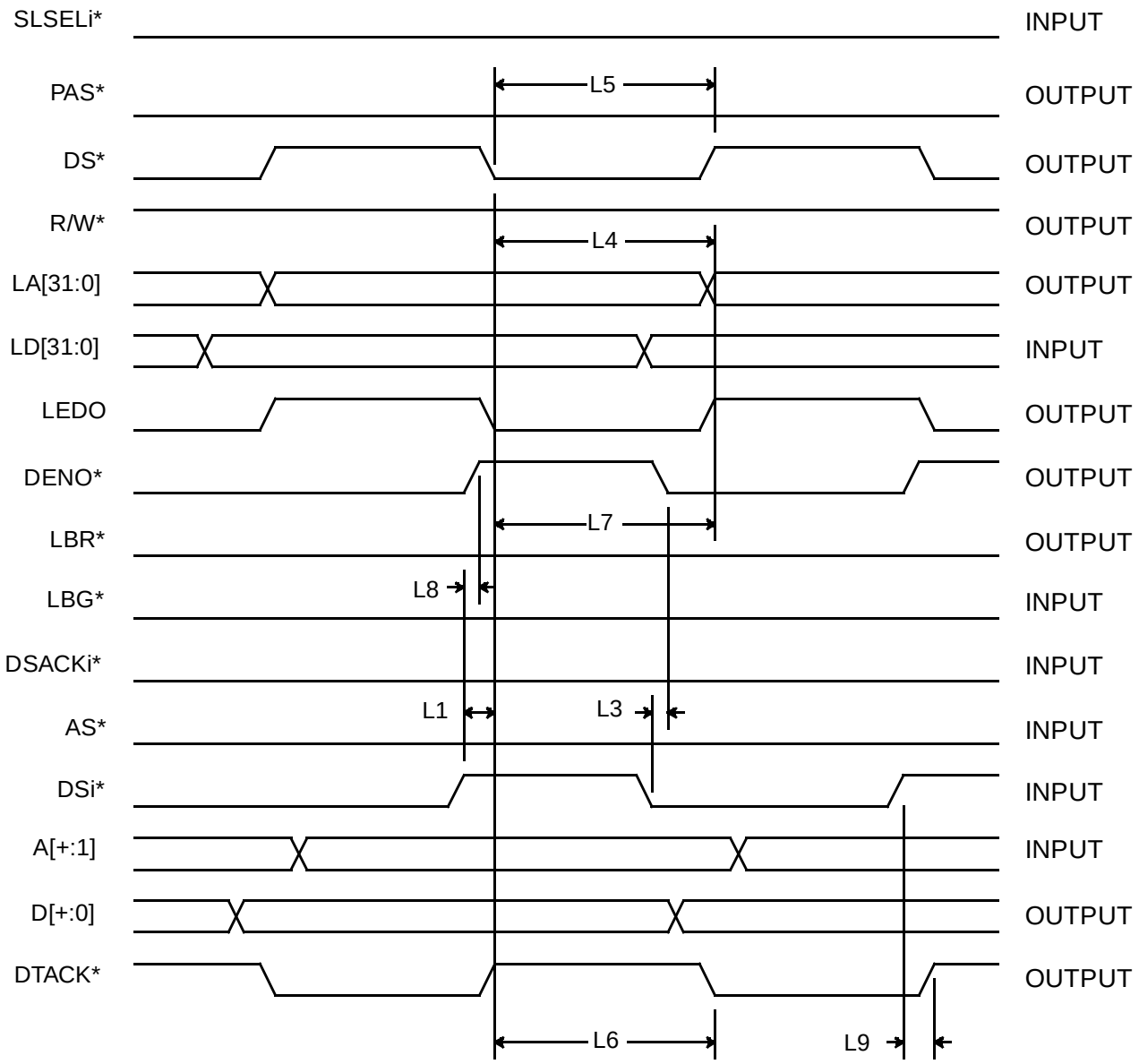


Figure 1-43. Slave Read BLT

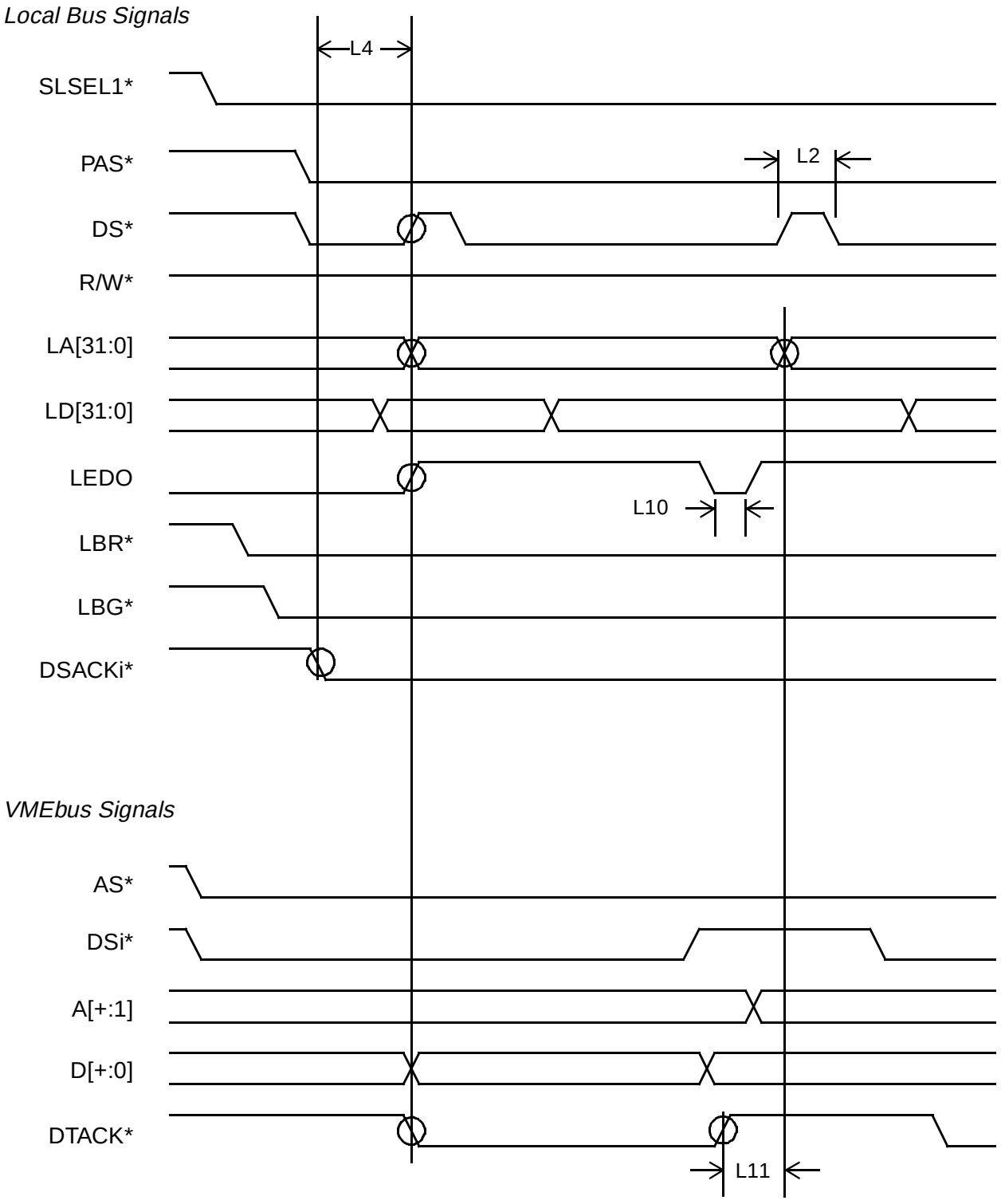
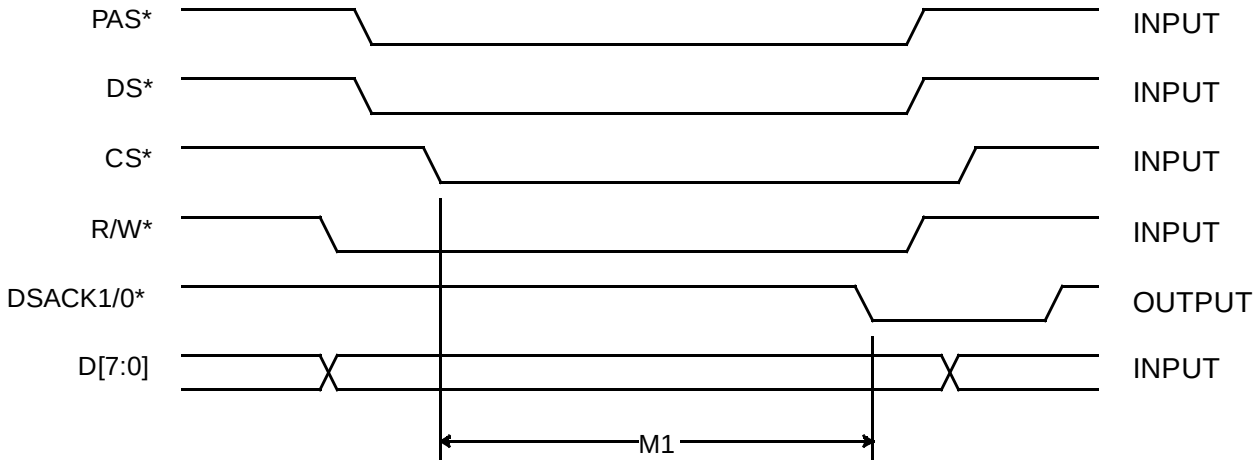
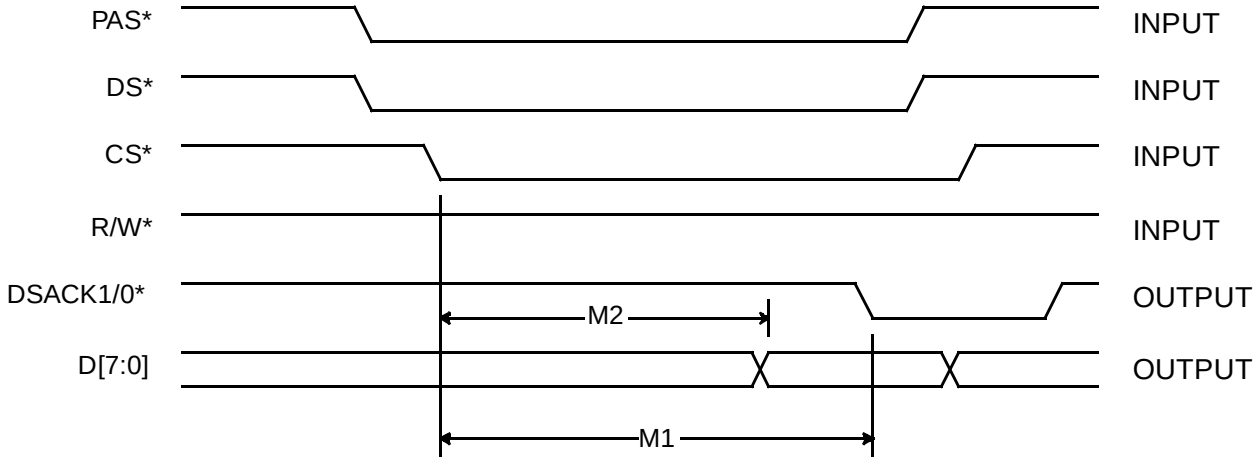


Figure 1-44. Slave Block Transfer—Read, Slow Master

Write



Read



ICF Select

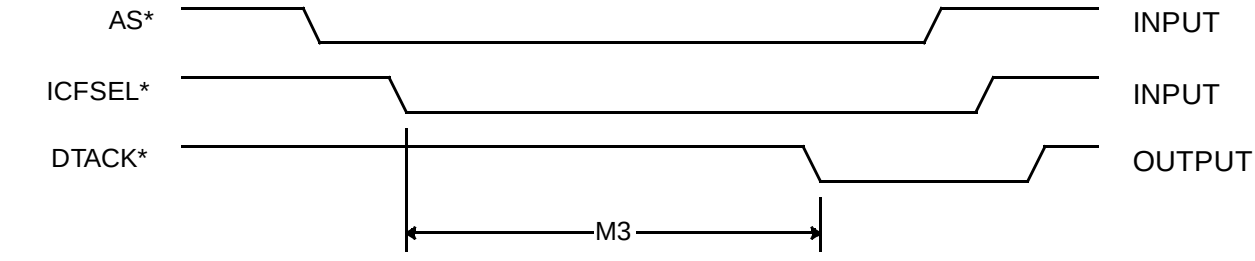


Figure 1-45. Register Operations

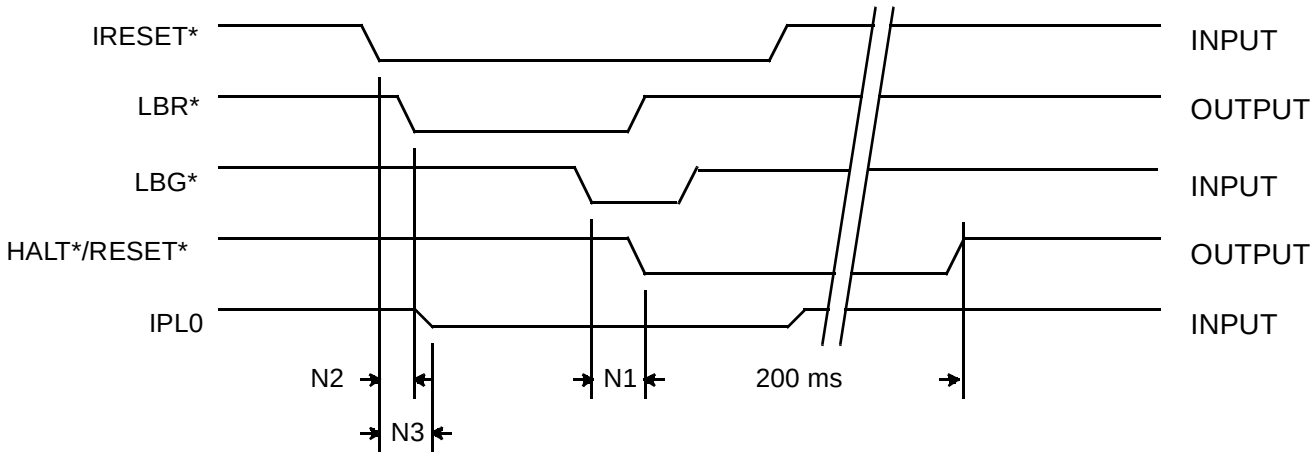


Figure 1-46. Global Reset

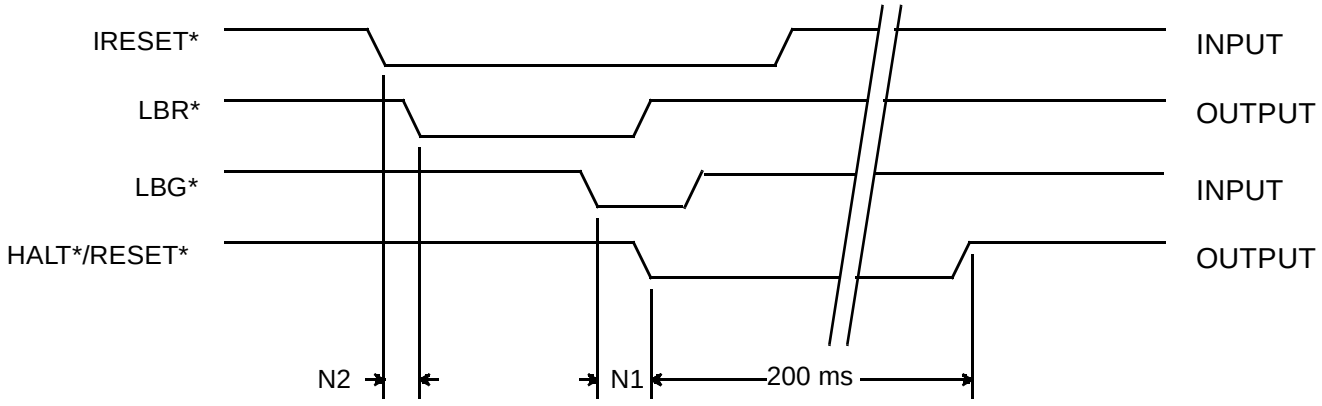


Figure 1-47. Internal Reset

1.14
VIC068A Signal List and Pinouts

Table 1-17. 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
BG1OUT*	L13	87	Output	VMEbus Bus Grant Output
BG0IN*	N13	76	Input	VMEbus Bus Grant Input
BG0OUT*	M14	86	Output	VMEbus Bus Grant Output

Table 1-17. VMEbus Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
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 1-18. 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	135	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
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

Table 1-18. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
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	9	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 1-19. 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
DENIN1*	E15	107	Output	Local Data Enable In (Upper Word)
DENIN*	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 1-20. 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 1-21*.

Table 1-21. 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	DENIN*
13	LIRQ5*	44	IRQ6*	75	BBSY*	106	V_{SS}
14	LIRQ6*	45	IRQ7*	76	BGIN0*	107	DENIN1*
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	SLSEL0*	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
125	SIZ1	134	DSACK1*	143	LA3	152	LD4

Table 1-21. Pinout for VIC068A Plastic and Ceramic Quad Flatpack (160-Pin): Cavity Up (continued)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
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*	FCIACK*	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*	SYSRE-SET*	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*	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*	$\overline{BR3}$*	VSS	13
SIZ1	IRESET*	LADO	LEDI	DDIR	DENIN*	DENO*	D06	D03	D01	VSS7	BG0OUT*	BG3IN*	BG1IN*	BCLR*	14
LBG*	ABEN*	VDD	LEDO	DENIN1*	SWDEN*	ISOBE*	D07	D05	D04	D02	BG3OUT*	BG2OUT*	SYSCLK	VSS	15

Figure 1-48. VIC068A Pin Grid Array (PGA), Bottom View

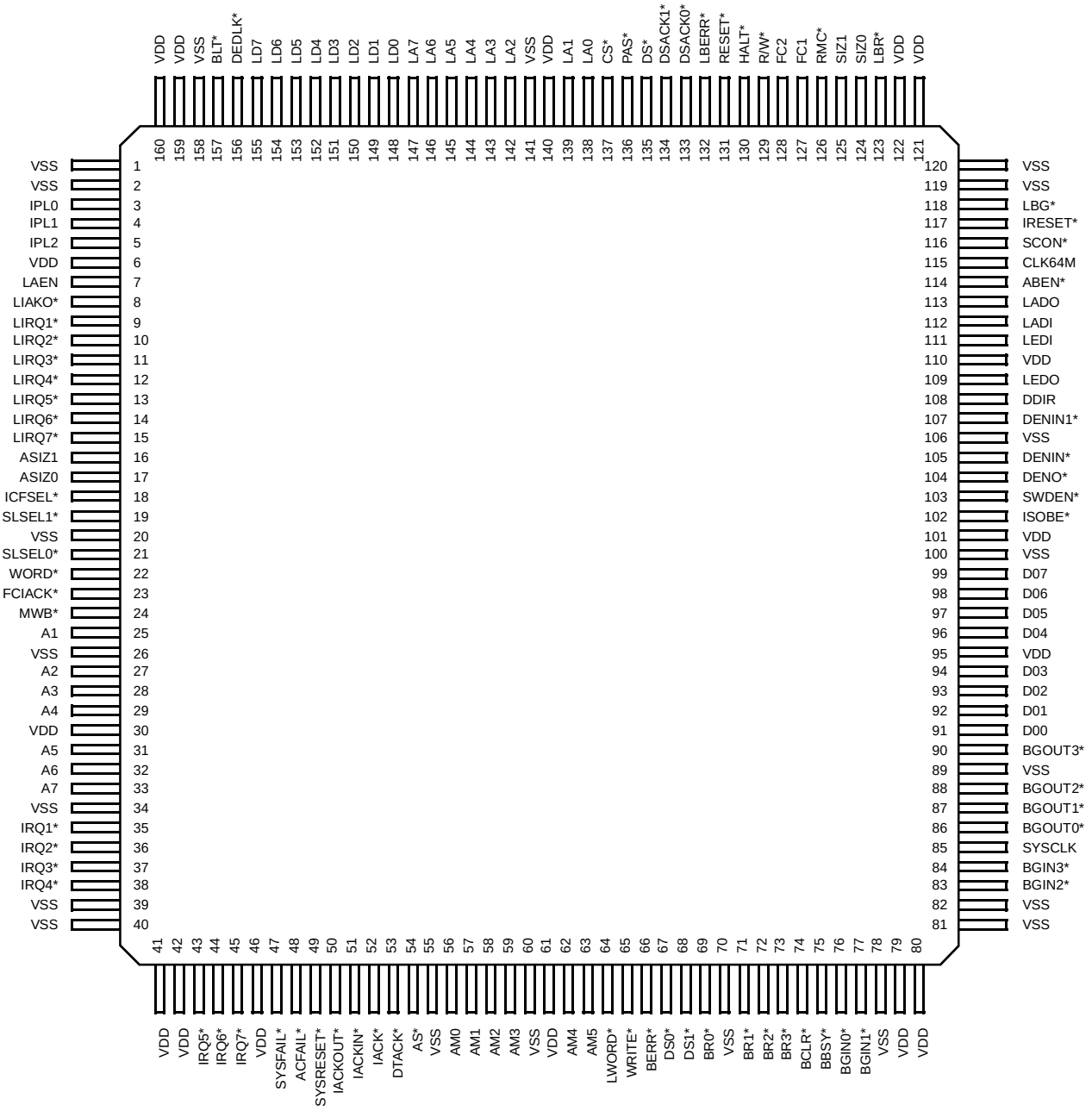


Figure 1-49. VIC068A Quad Flatpack (QFP), Top View

1.15

VIC068A Simulation Waveforms

Note: LWDENIN* is now called DENIN* and UWDENIN* is now called DENIN1*.

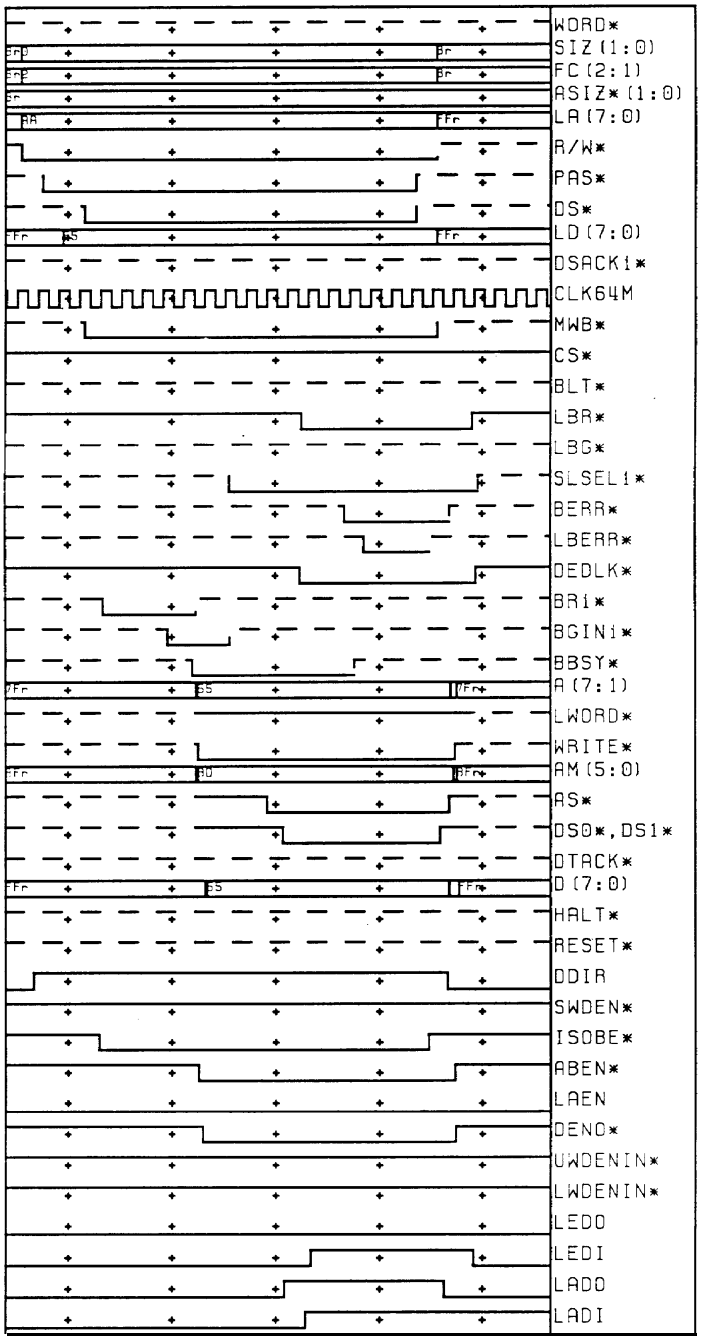


Figure 1-50. Master Self-Access

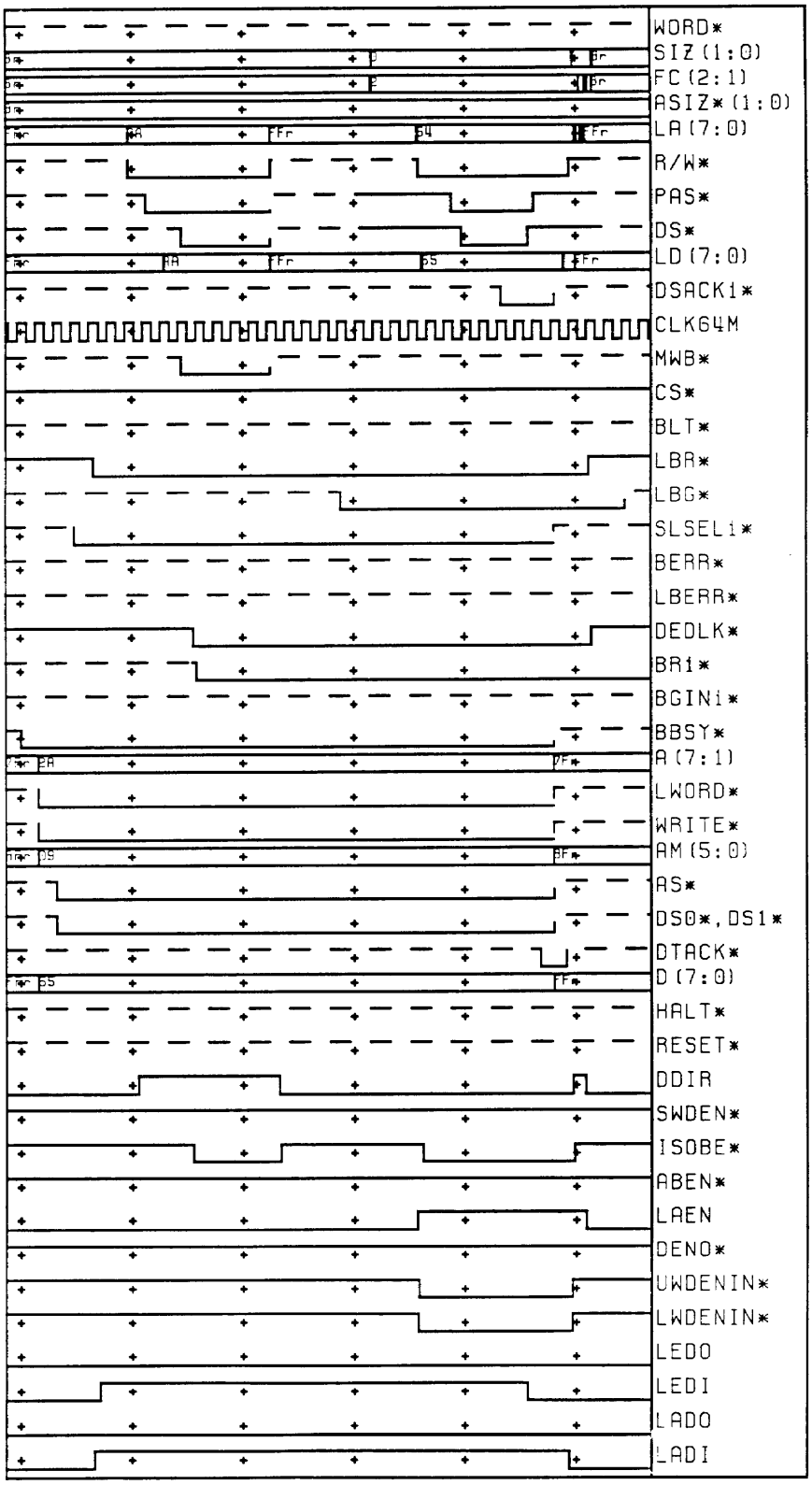


Figure 1-51. Master Deadlock Operation

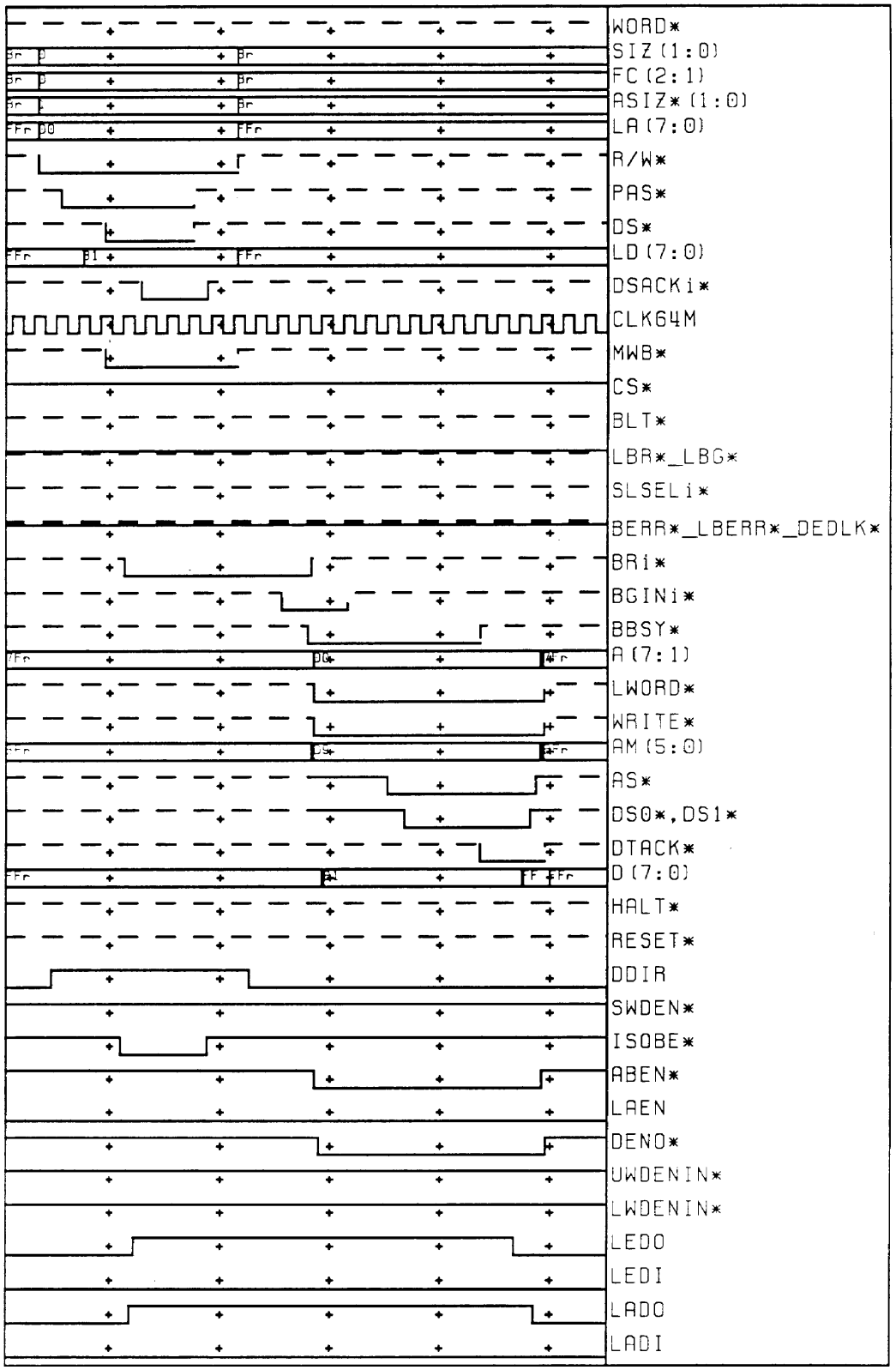


Figure 1-52. Master Write Post

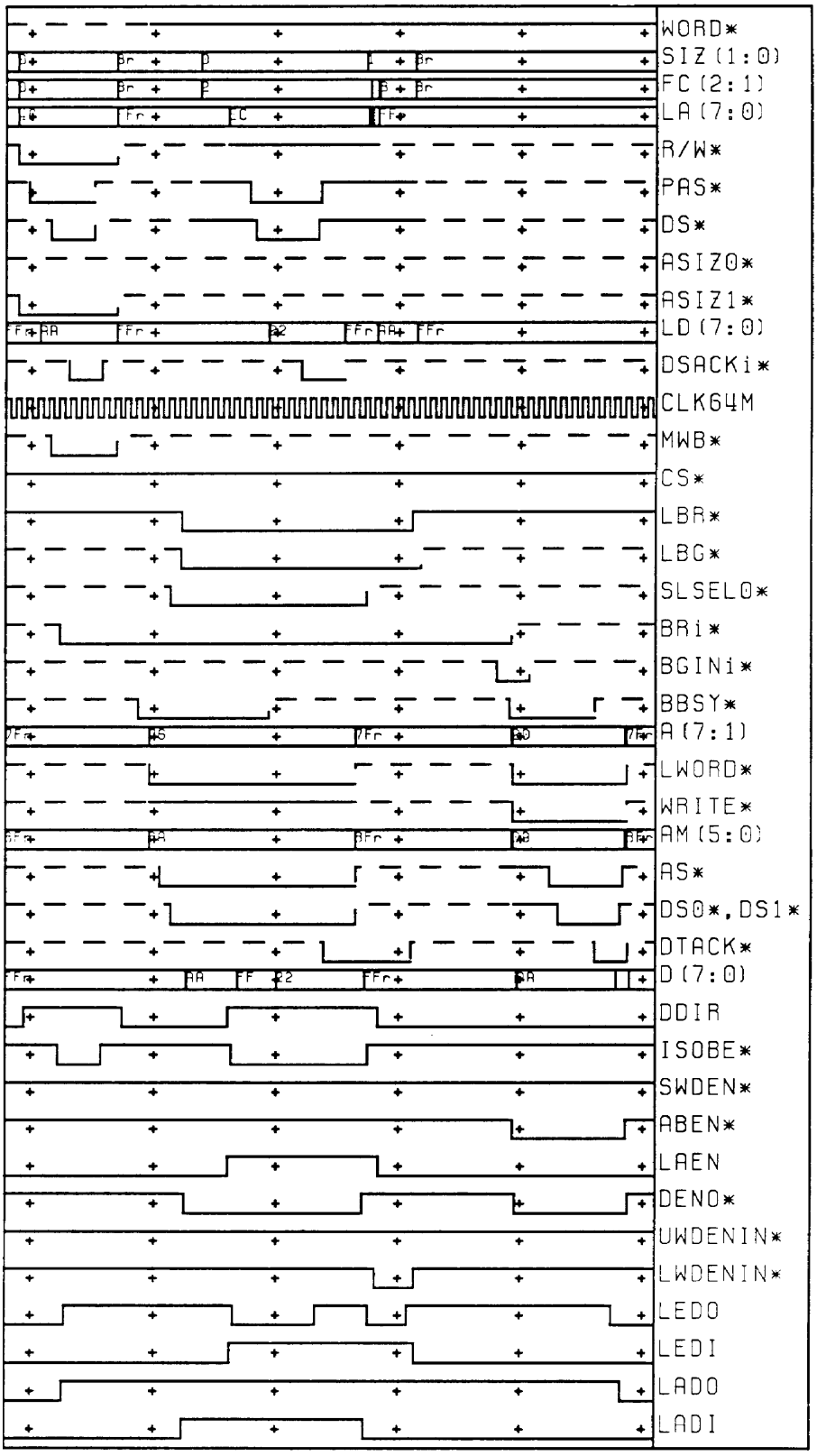


Figure 1-53. Master Write Post with Slave Read (shows data toggle)

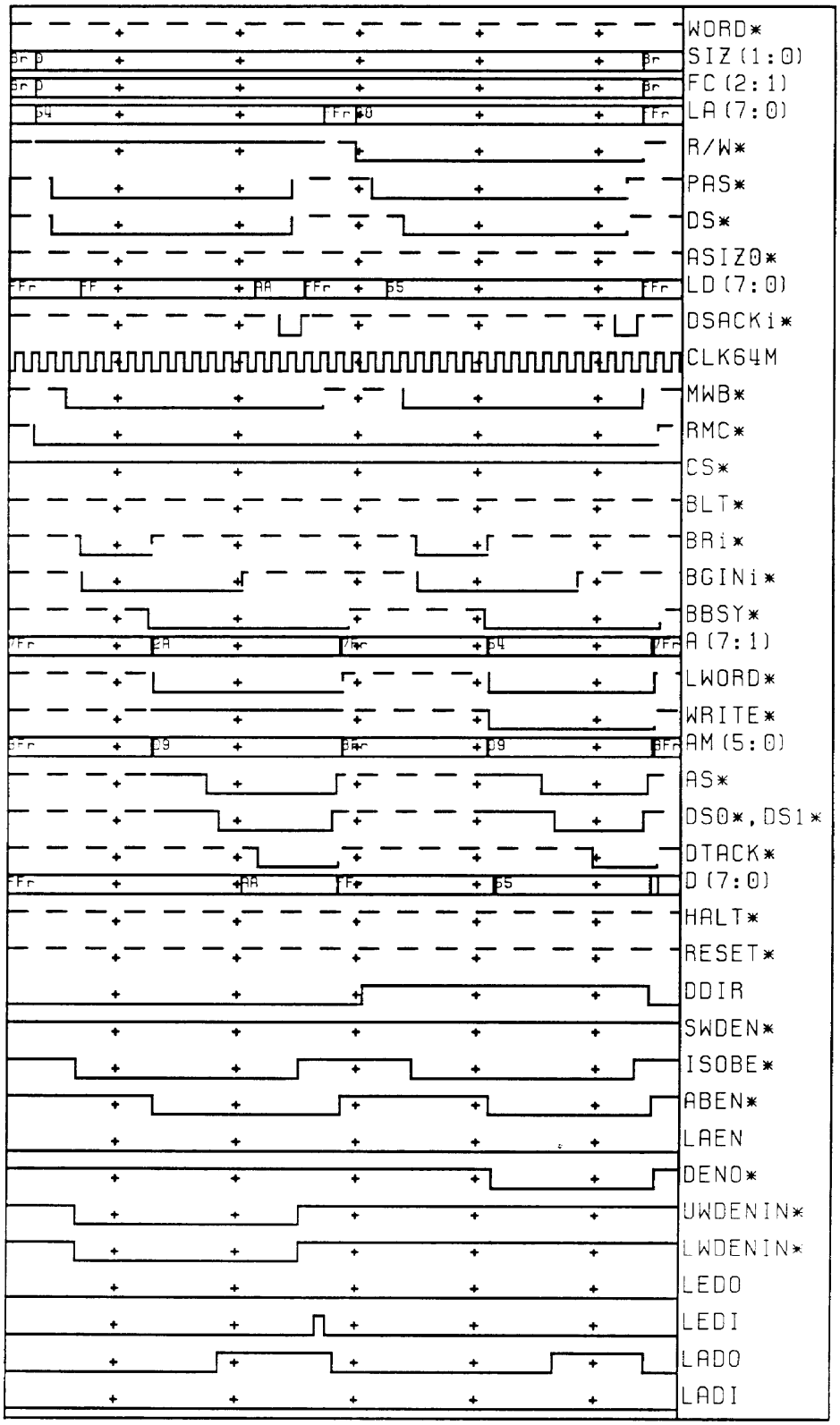


Figure 1-54. Master RMC1 ($AF[7:5] = 000)

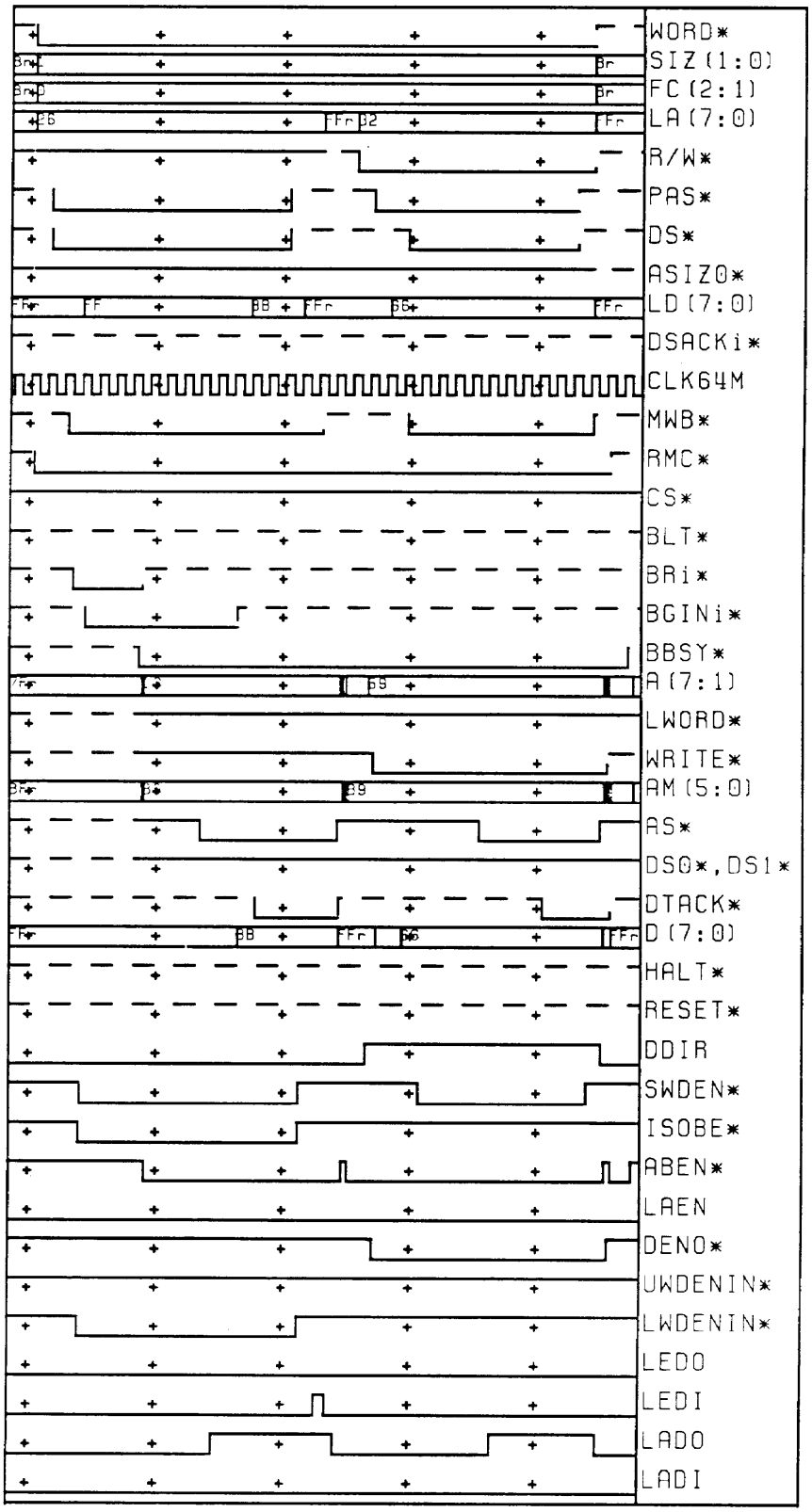


Figure 1-55. Master RMC2 ($AF[7:5] = 001)

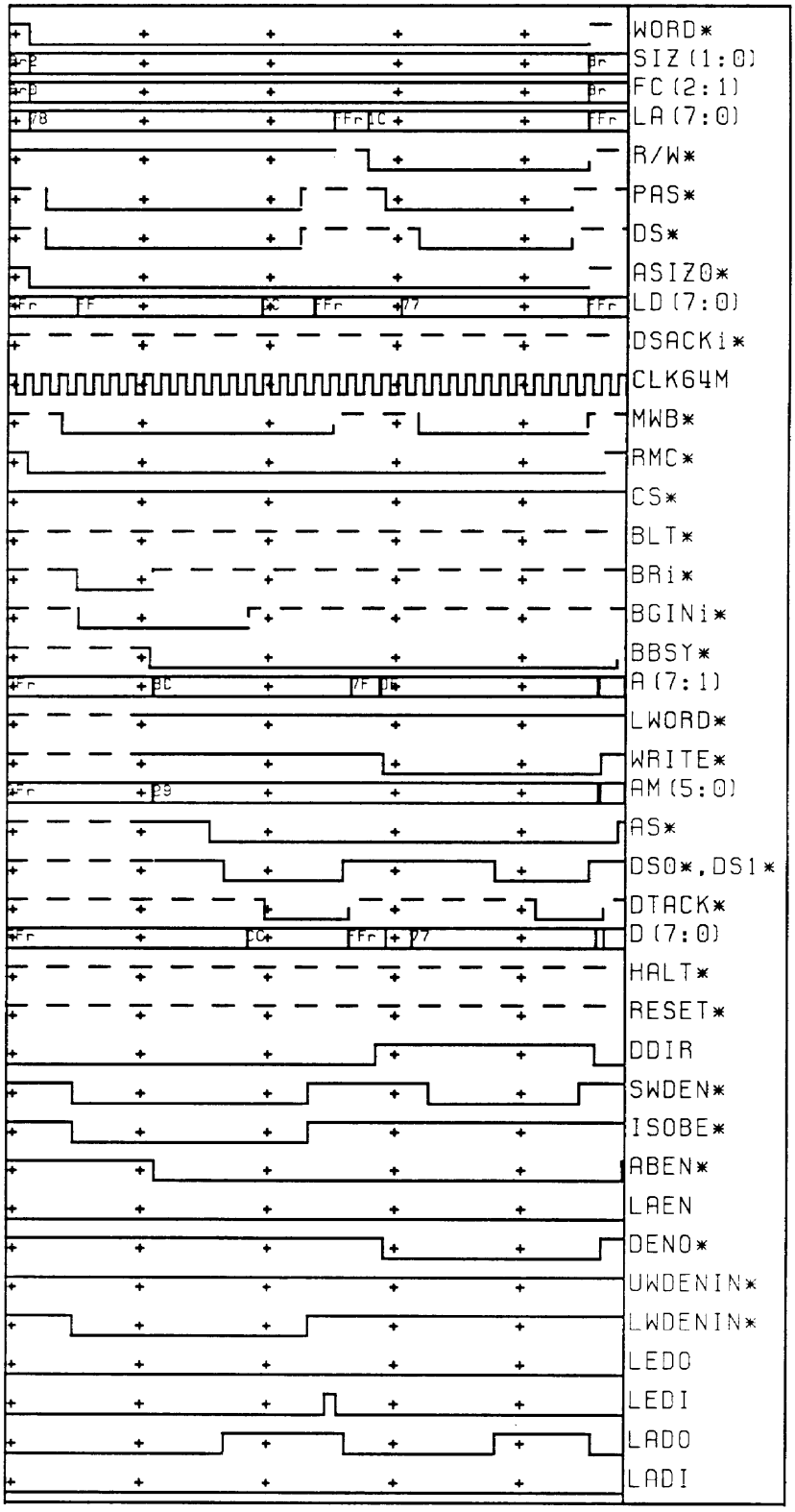


Figure 1-56. Master RMC3 ($AF[7:5] = 010)

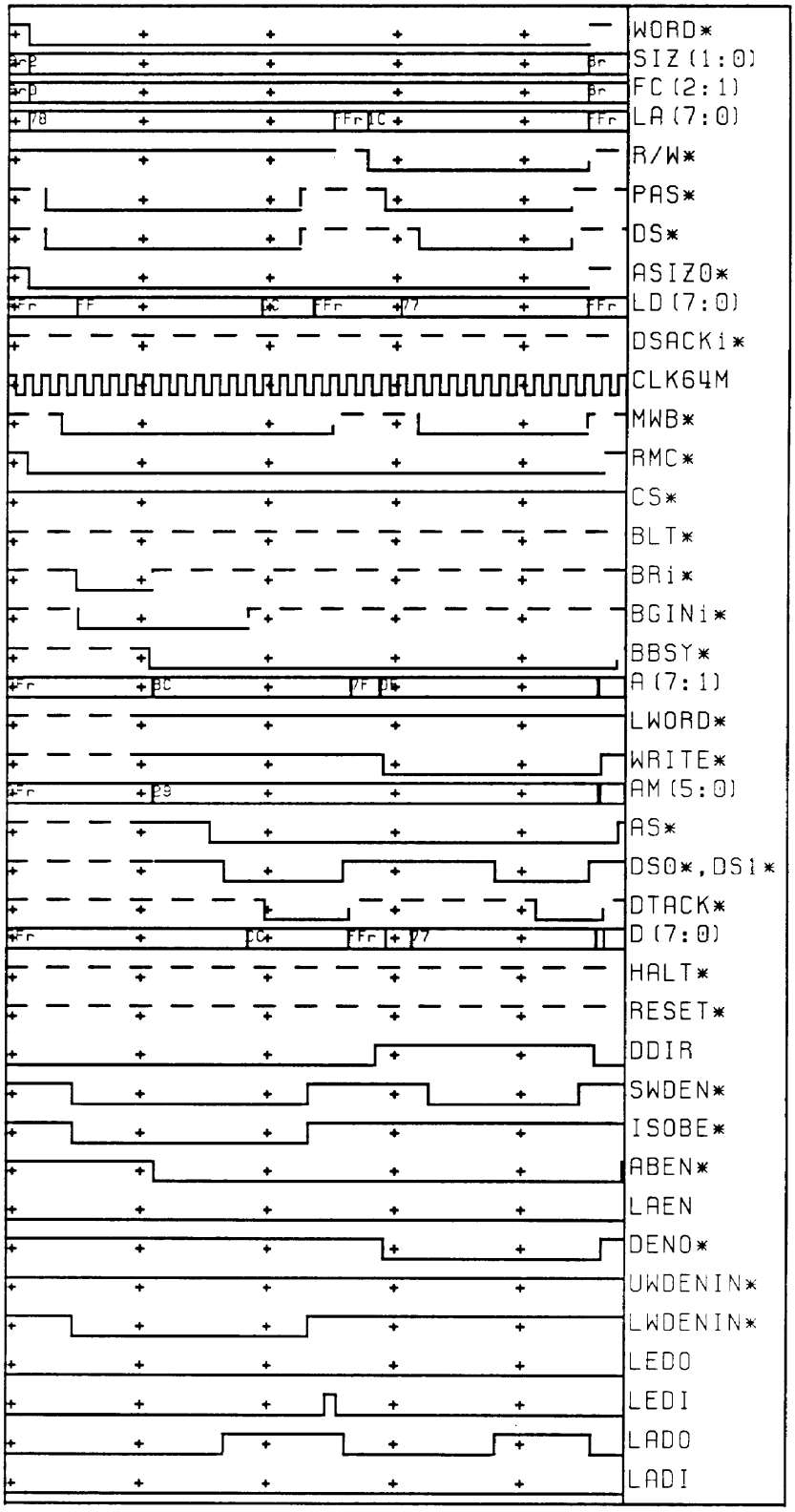


Figure 1-57. Master RMC4 ($AF[7:5] = 011)

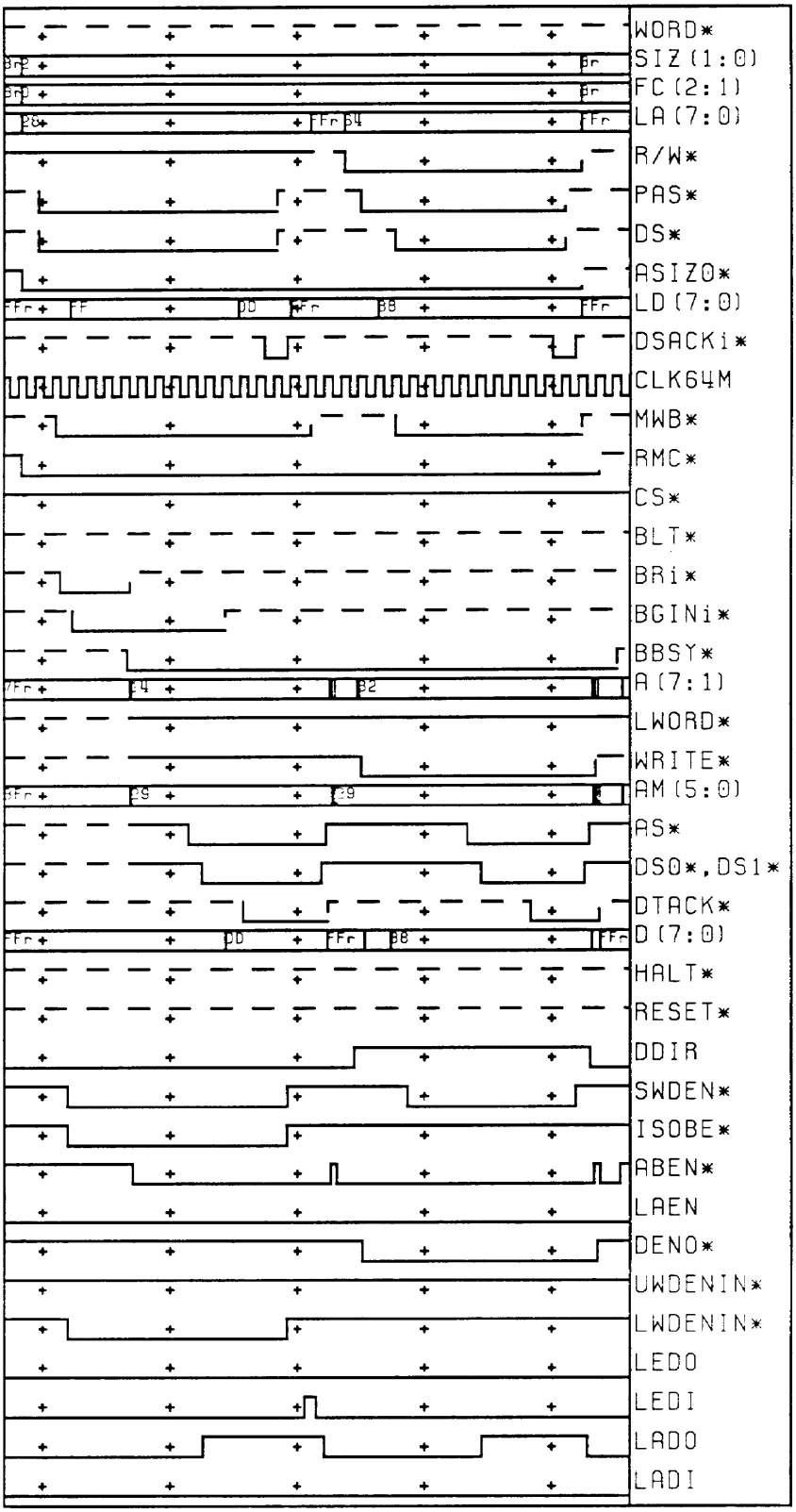


Figure 1-58. Master RMC5 ($AF[7:5] = 101)

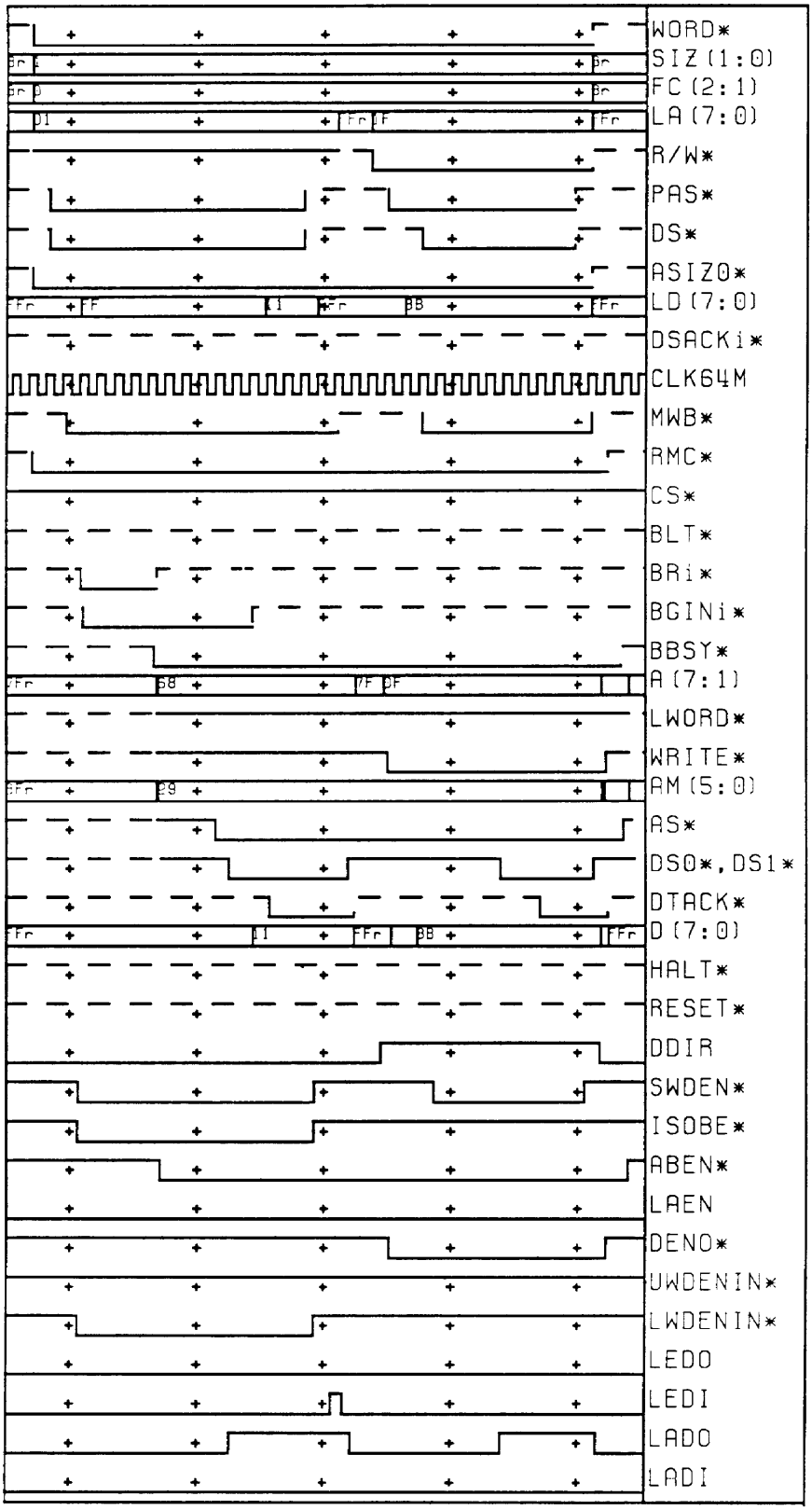


Figure 1-59. Master RMC6 Non-Byte ($AF[7:5] = 110)

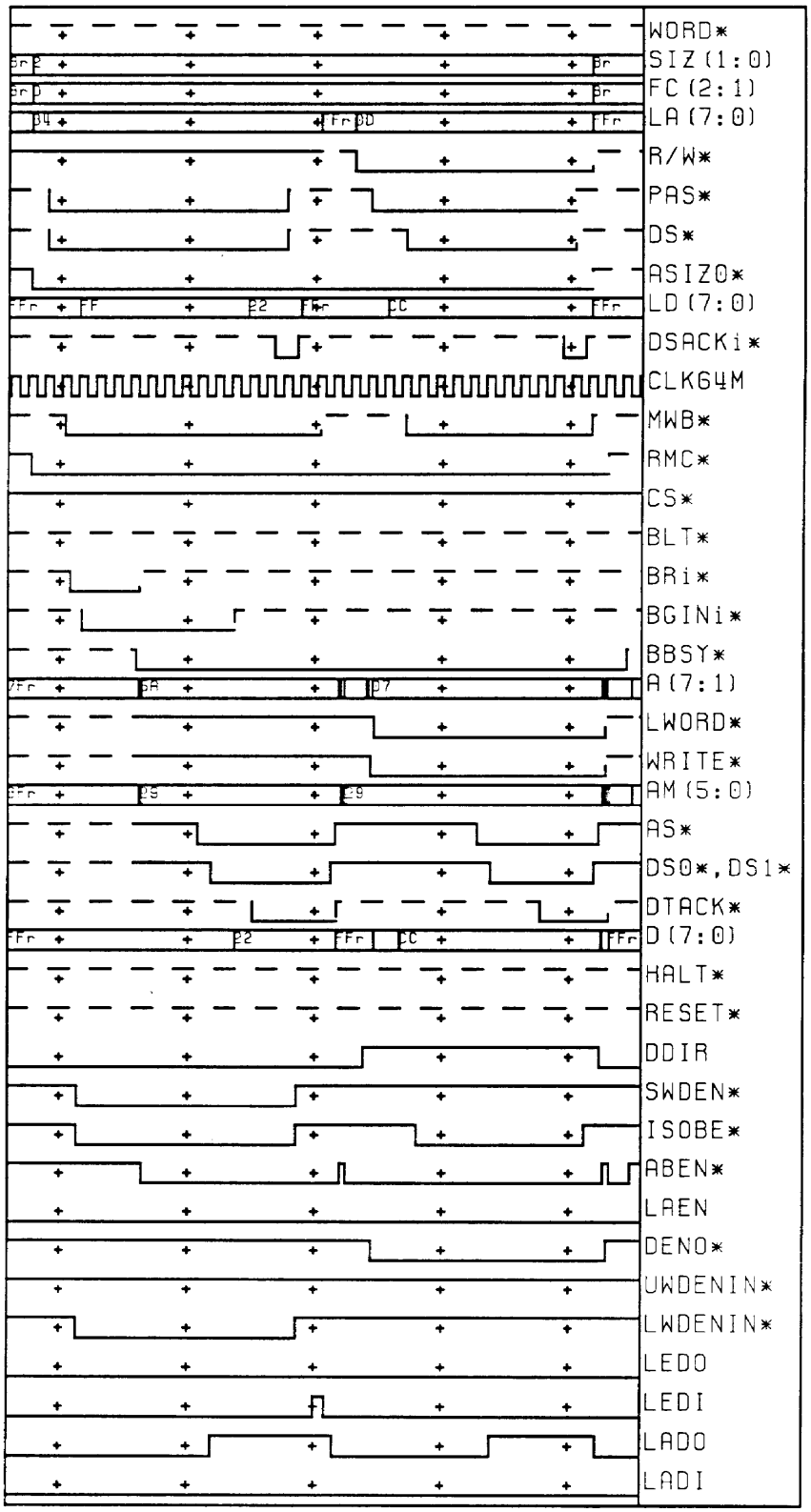


Figure 1-60. Master RMC7 ($AF[7:5] = 111)

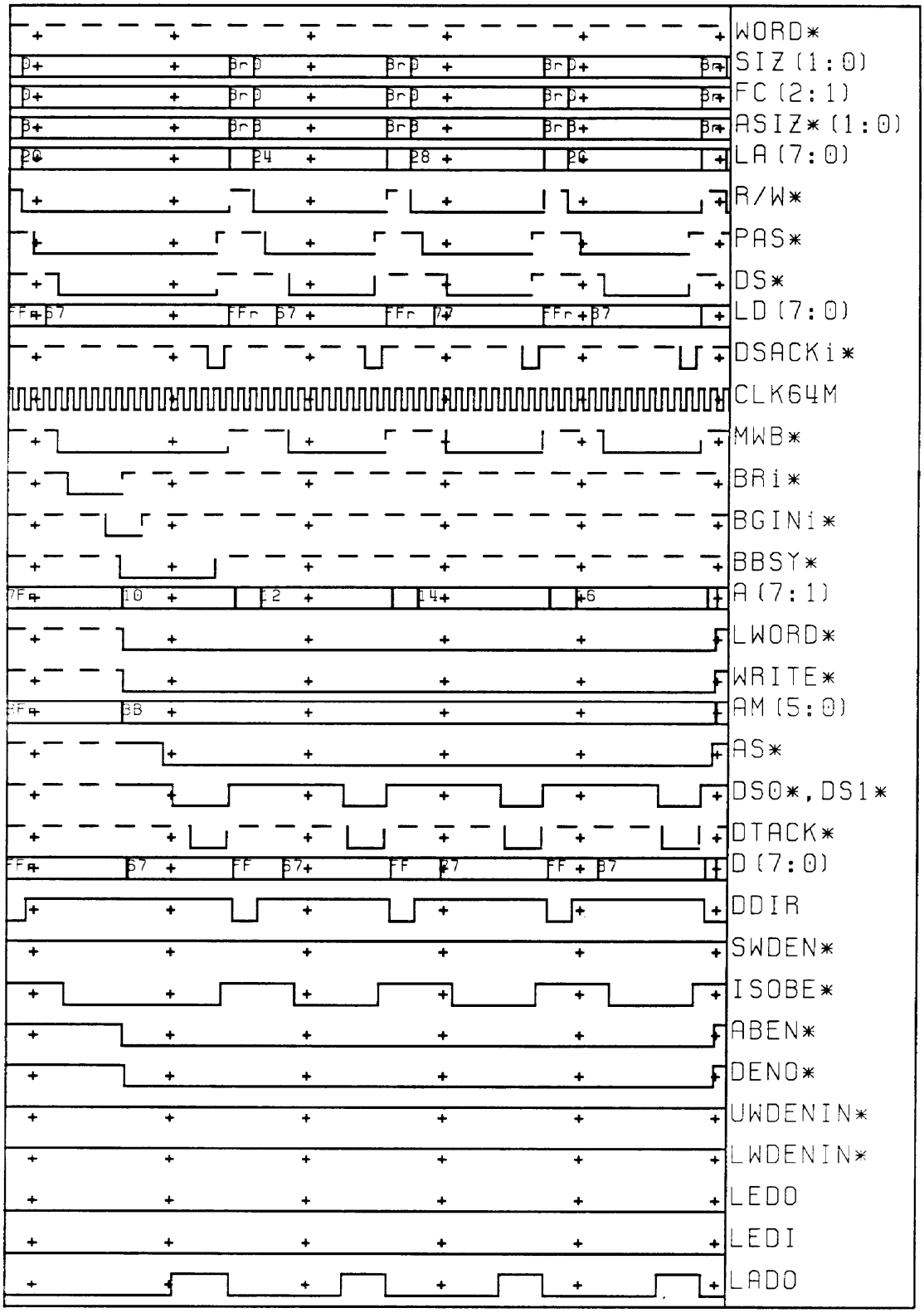


Figure 1-61. MOVEM Write Operation

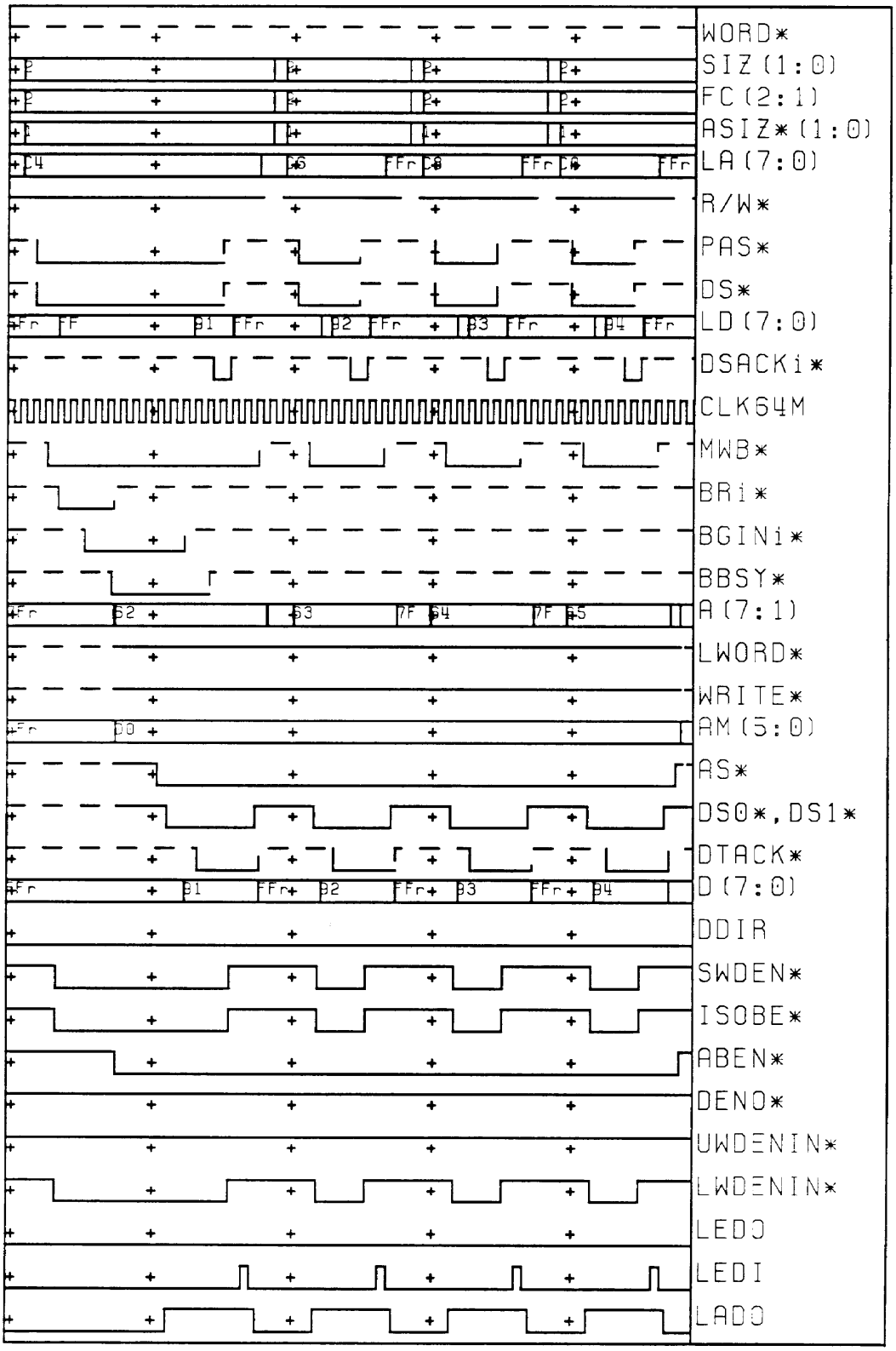


Figure 1-62. MOVEM Read Operation

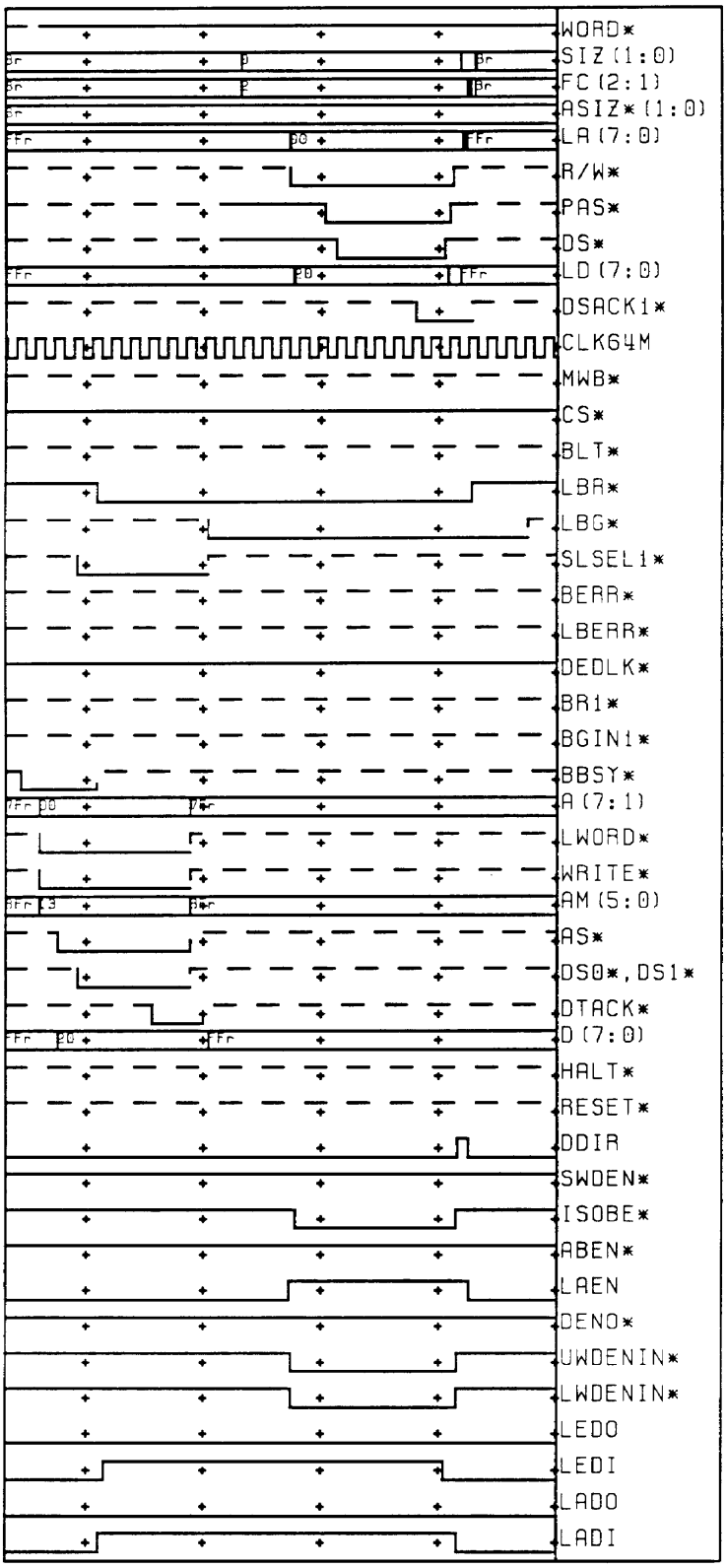


Figure 1-63. Slave Write Post

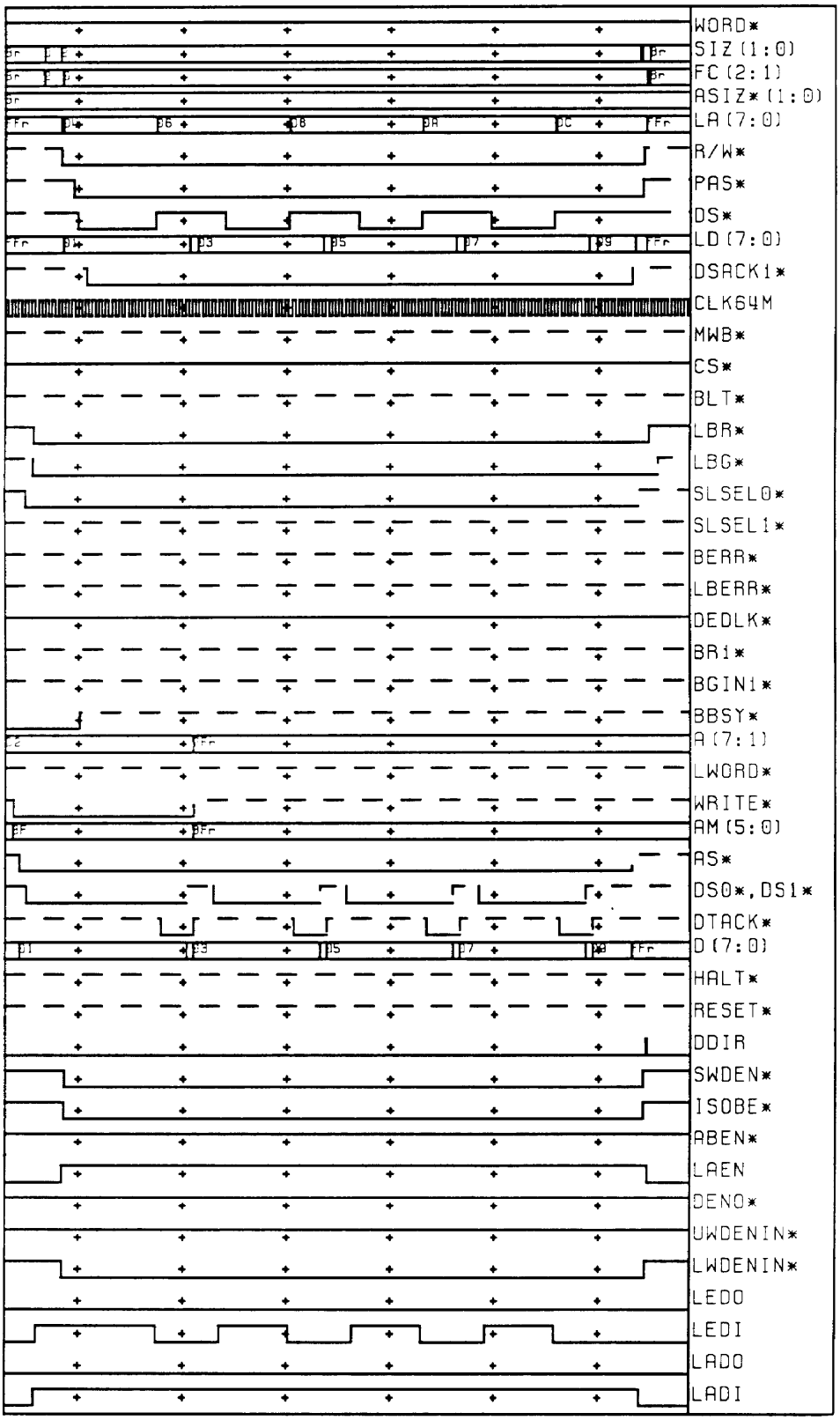


Figure 1-64. Slave Write Block Transfer Accelerated Mode ($C3[1:0]=10)

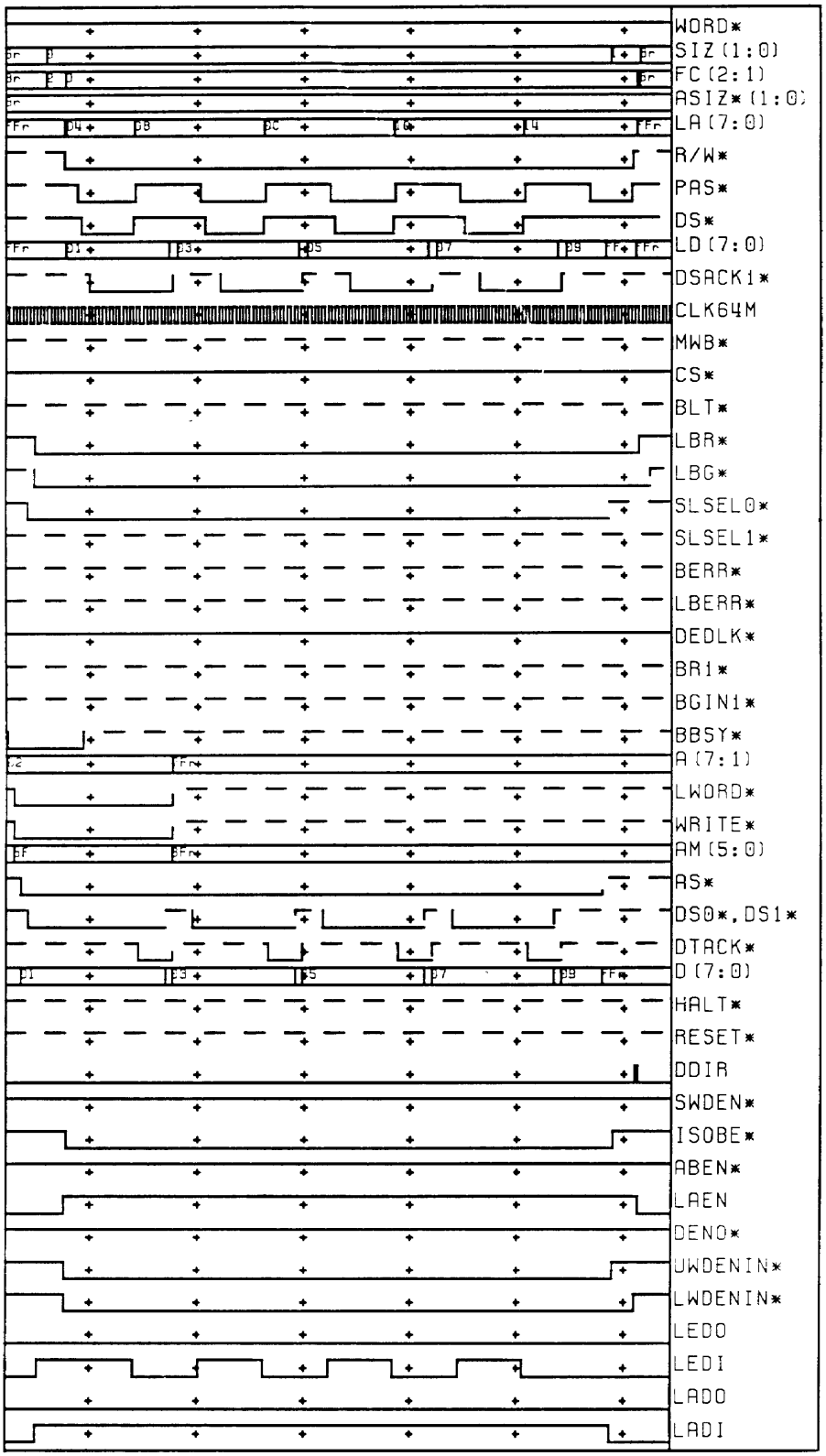


Figure 1-65. Slave Write Block Transfer Emulate Single-Cycle ($C3[1:0]=01)

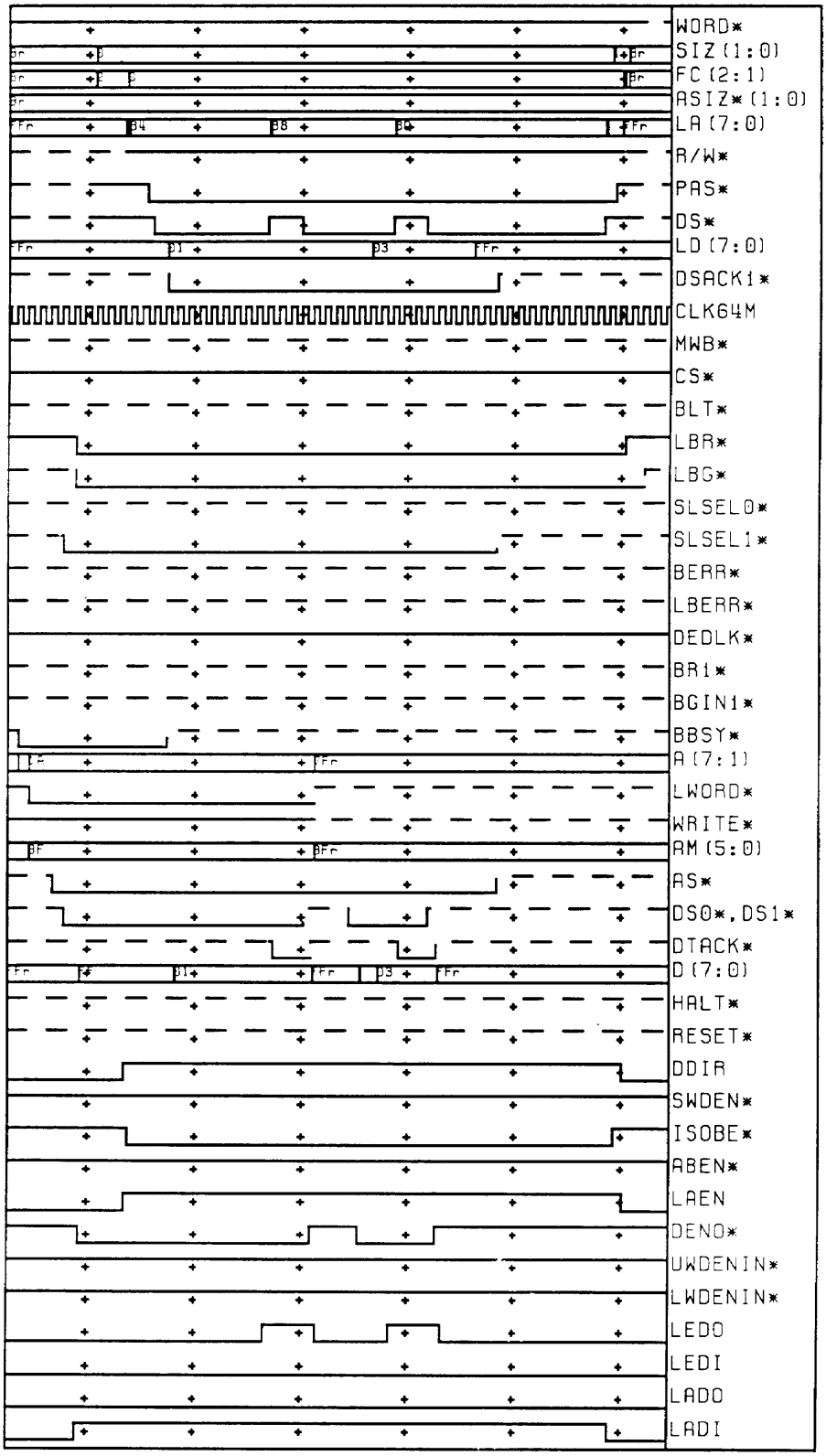


Figure 1-66. Slave Read Block Transfers Accelerated ($C3[1:0]=10)

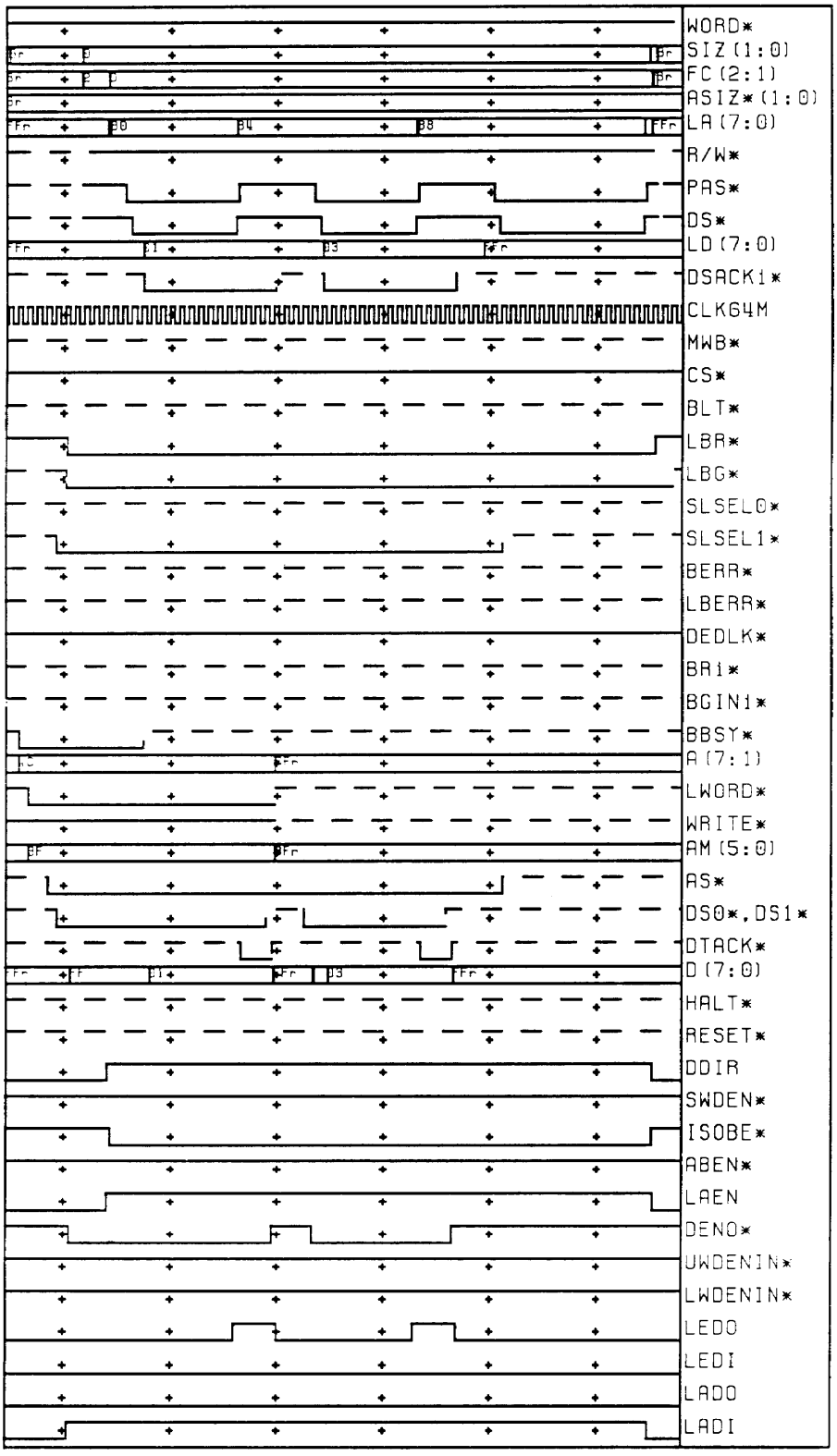


Figure 1-67. Slave Read Block Transfer Emulate Single-Cycle ($C3[1:0]=01)

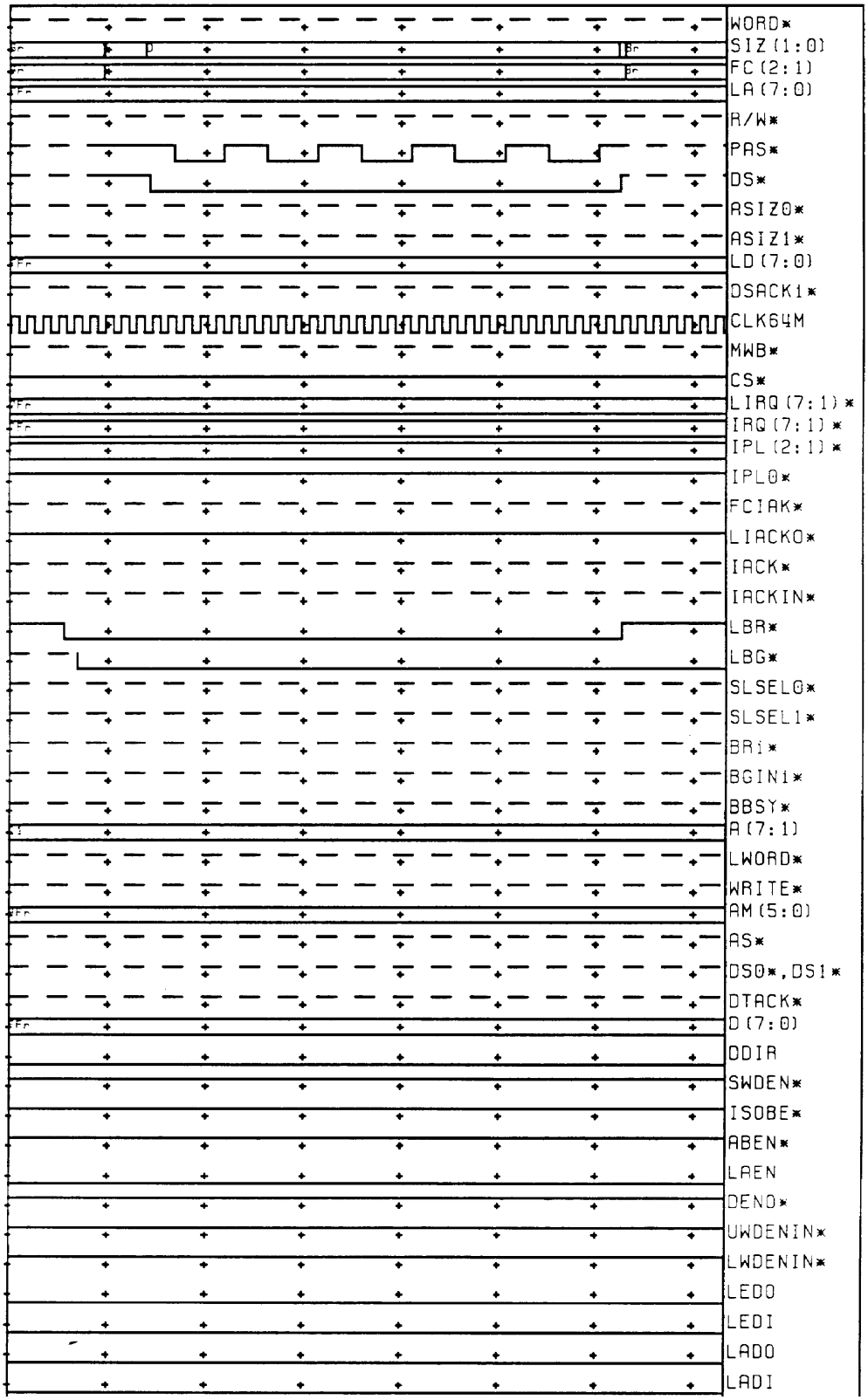


Figure 1-68. Refresh Timing

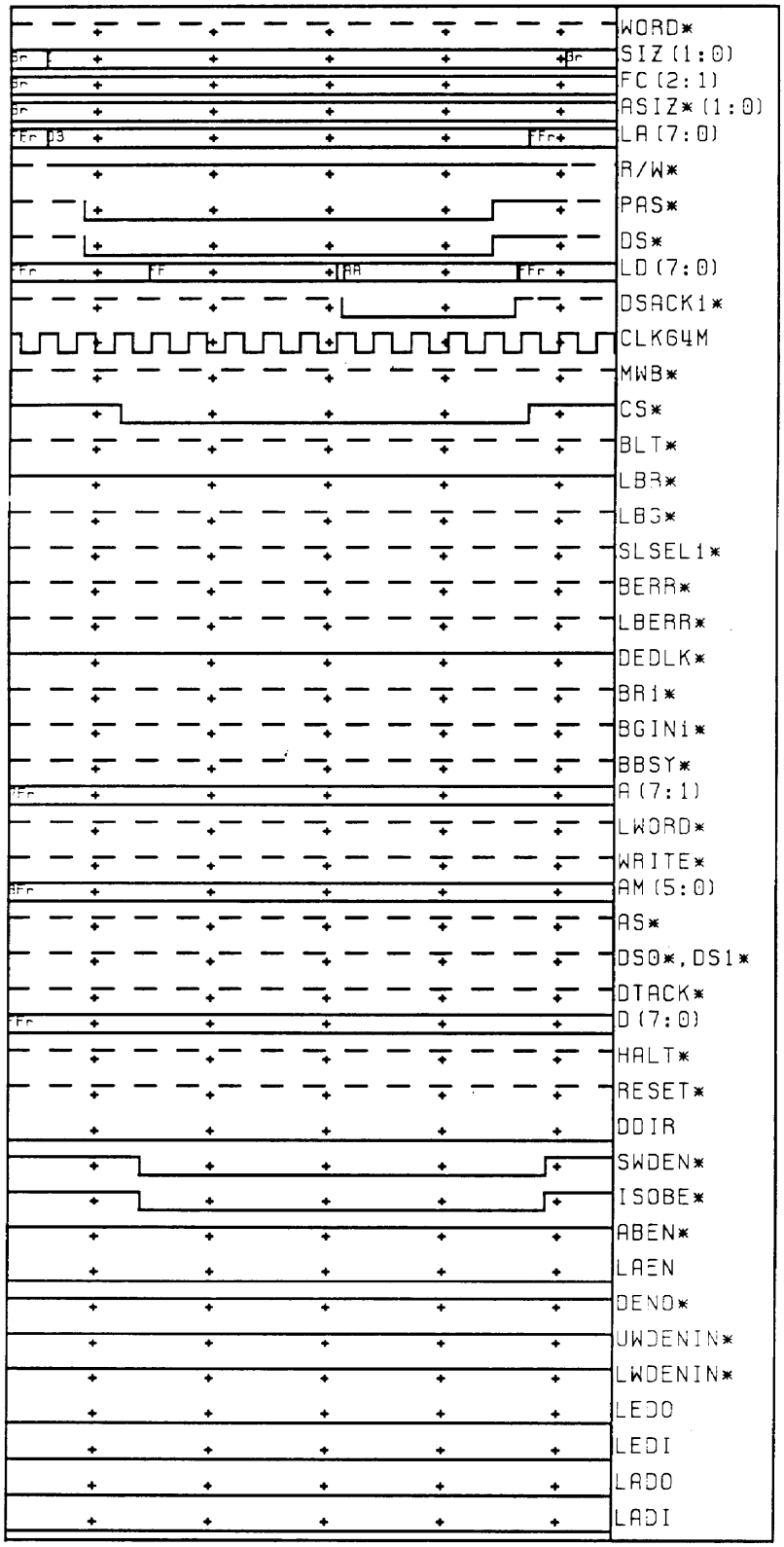


Figure 1-69. Register Read

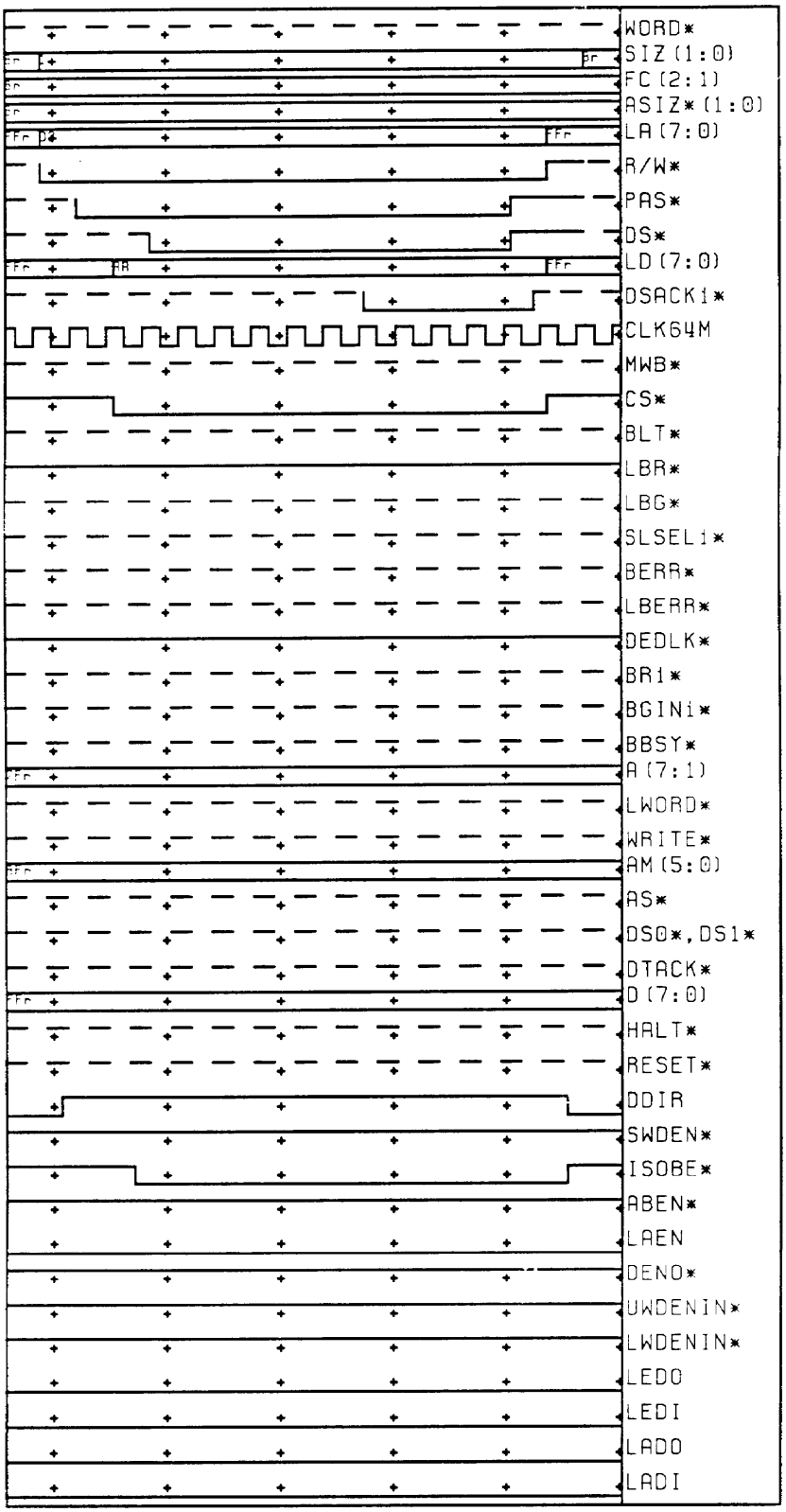


Figure 1-70. Register Write

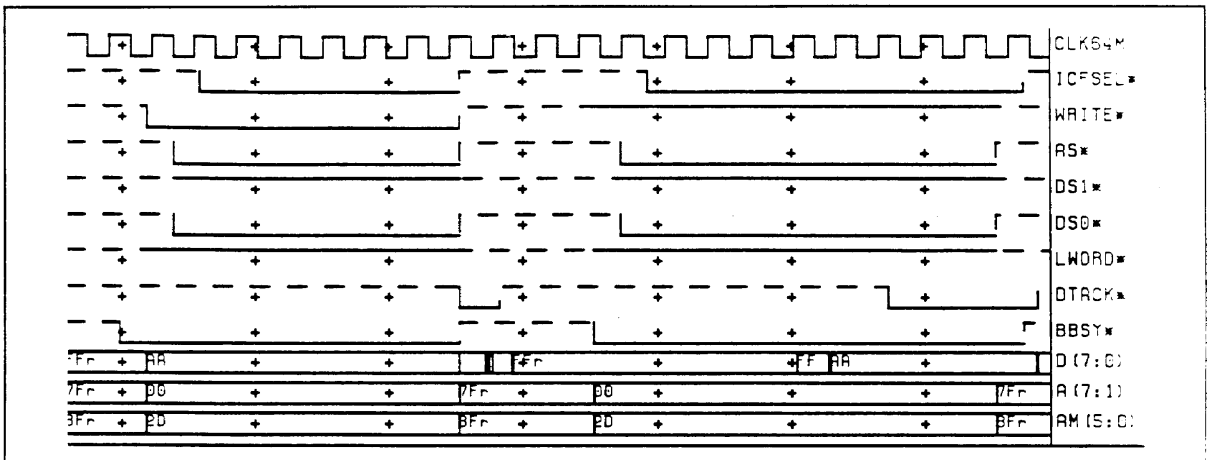


Figure 1-71. Interprocessor Communications Register Access Timing

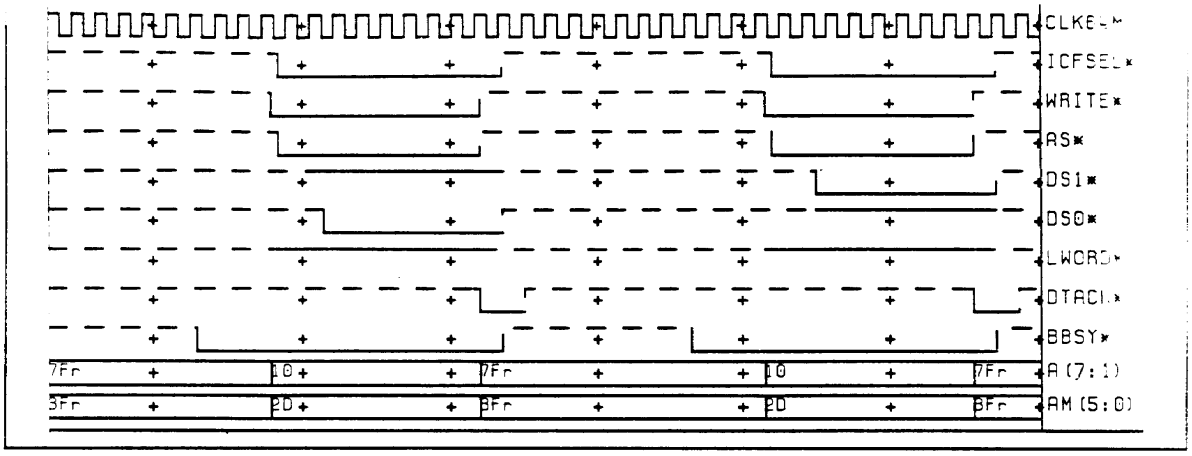


Figure 1-72. Interprocessor Communication Module Switch Access Timing

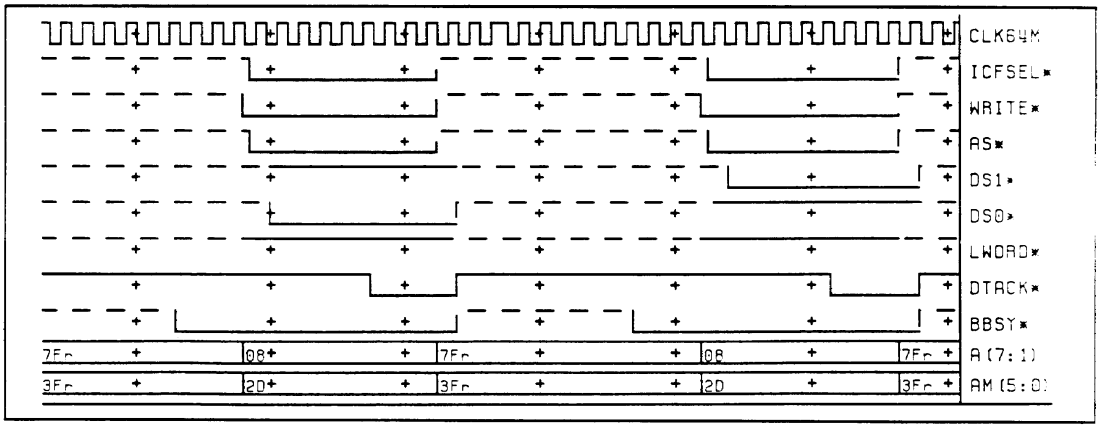


Figure 1-73. Interprocessor Communications Global Switch Access Timing

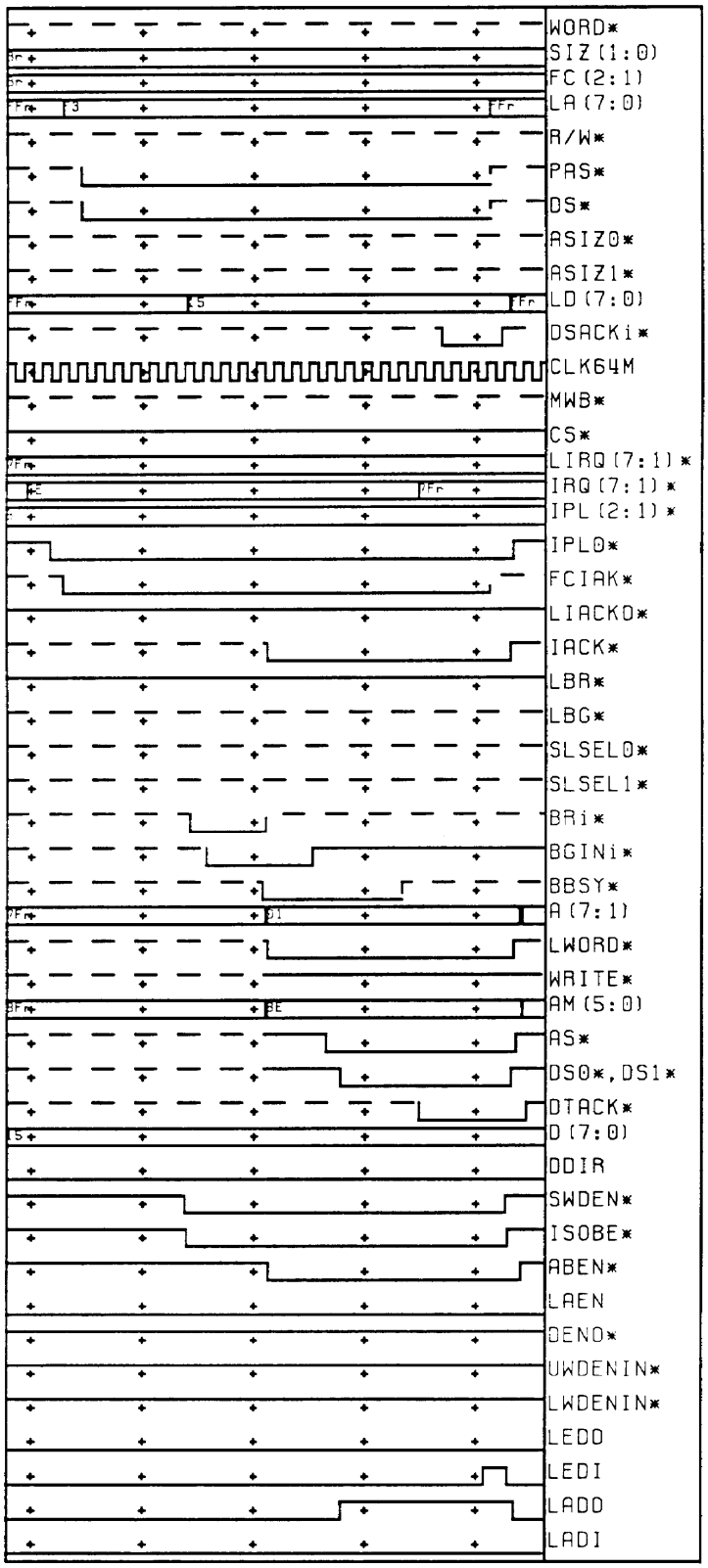


Figure 1-74. VMEbus Interrupt Acknowledge Cycle

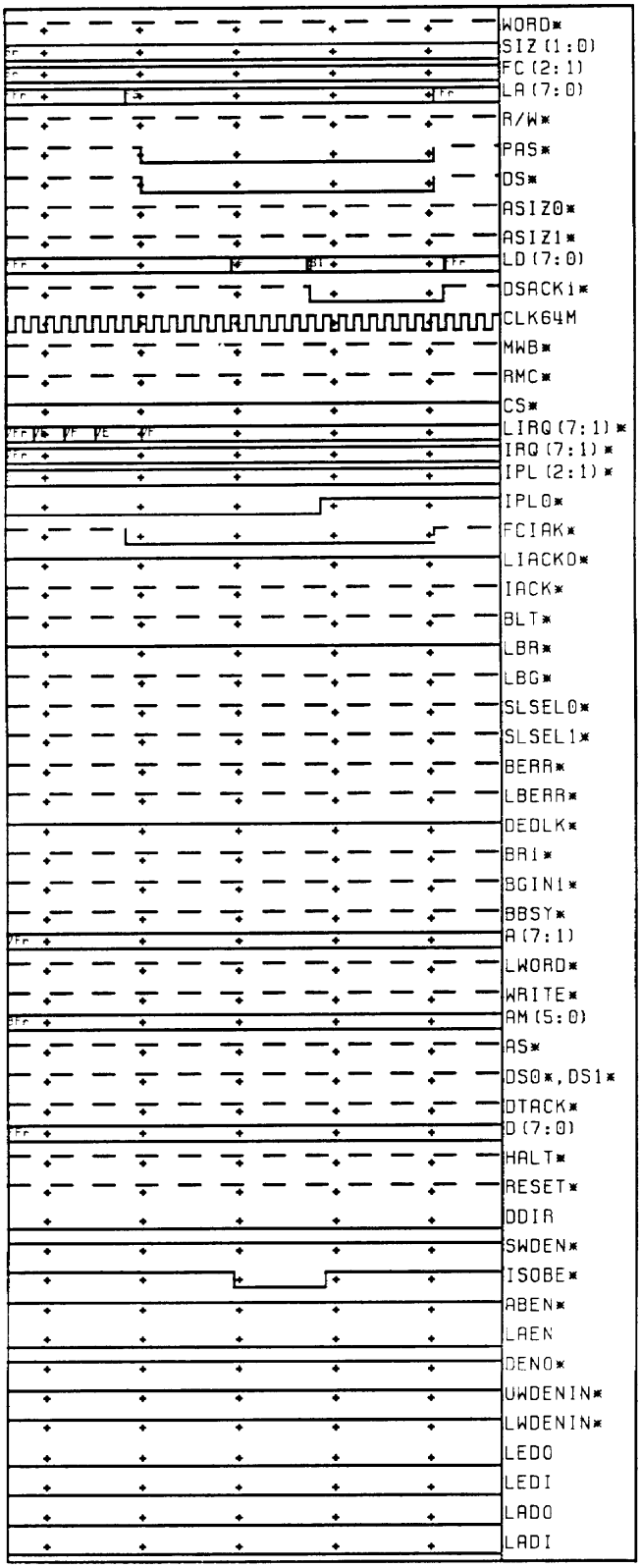


Figure 1-75. Local Interrupt Acknowledge Cycle

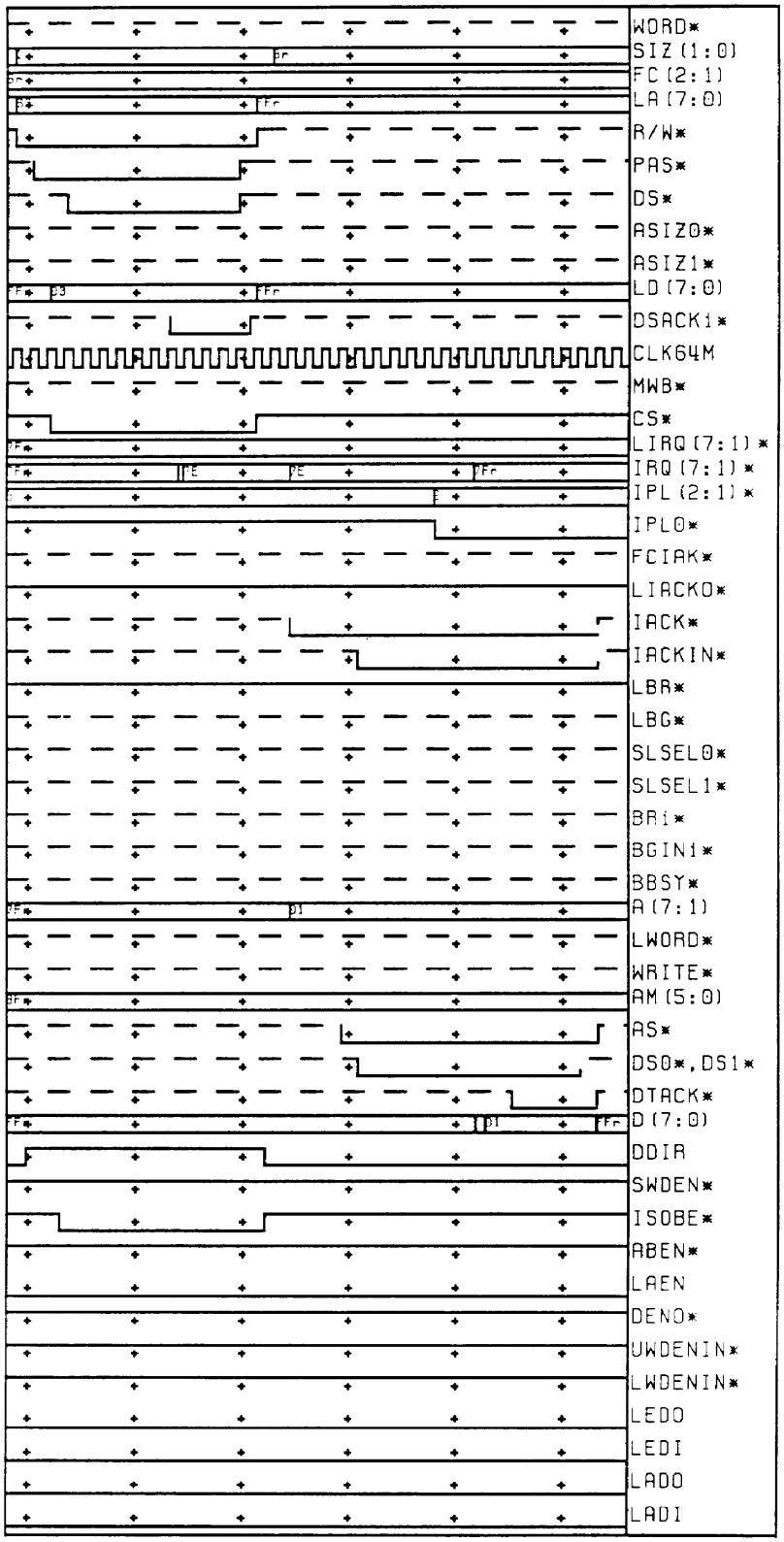


Figure 1-76. Interrupter Acknowledge Cycle

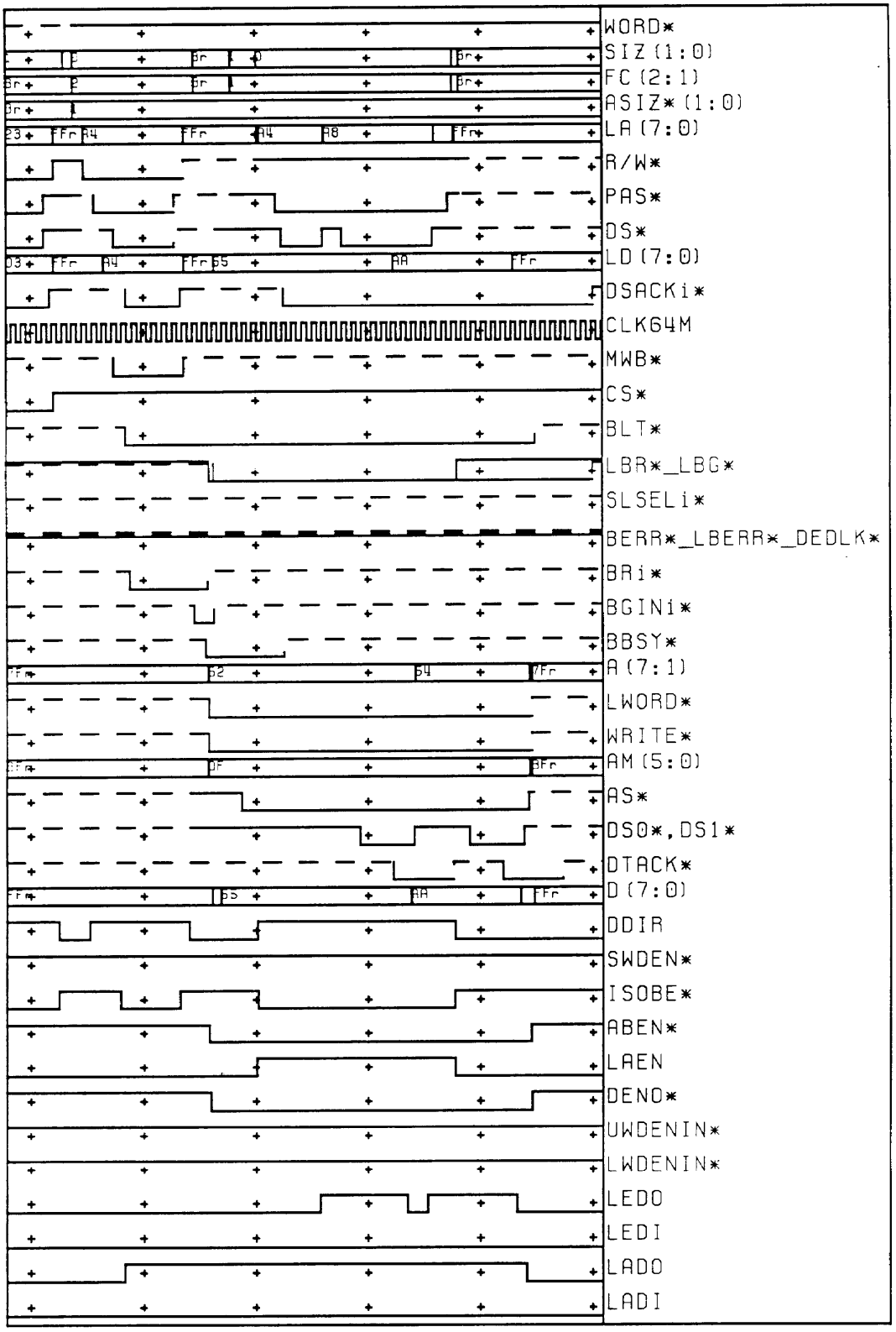


Figure 1-77. Block Transfer: VME Write: Burst of 2

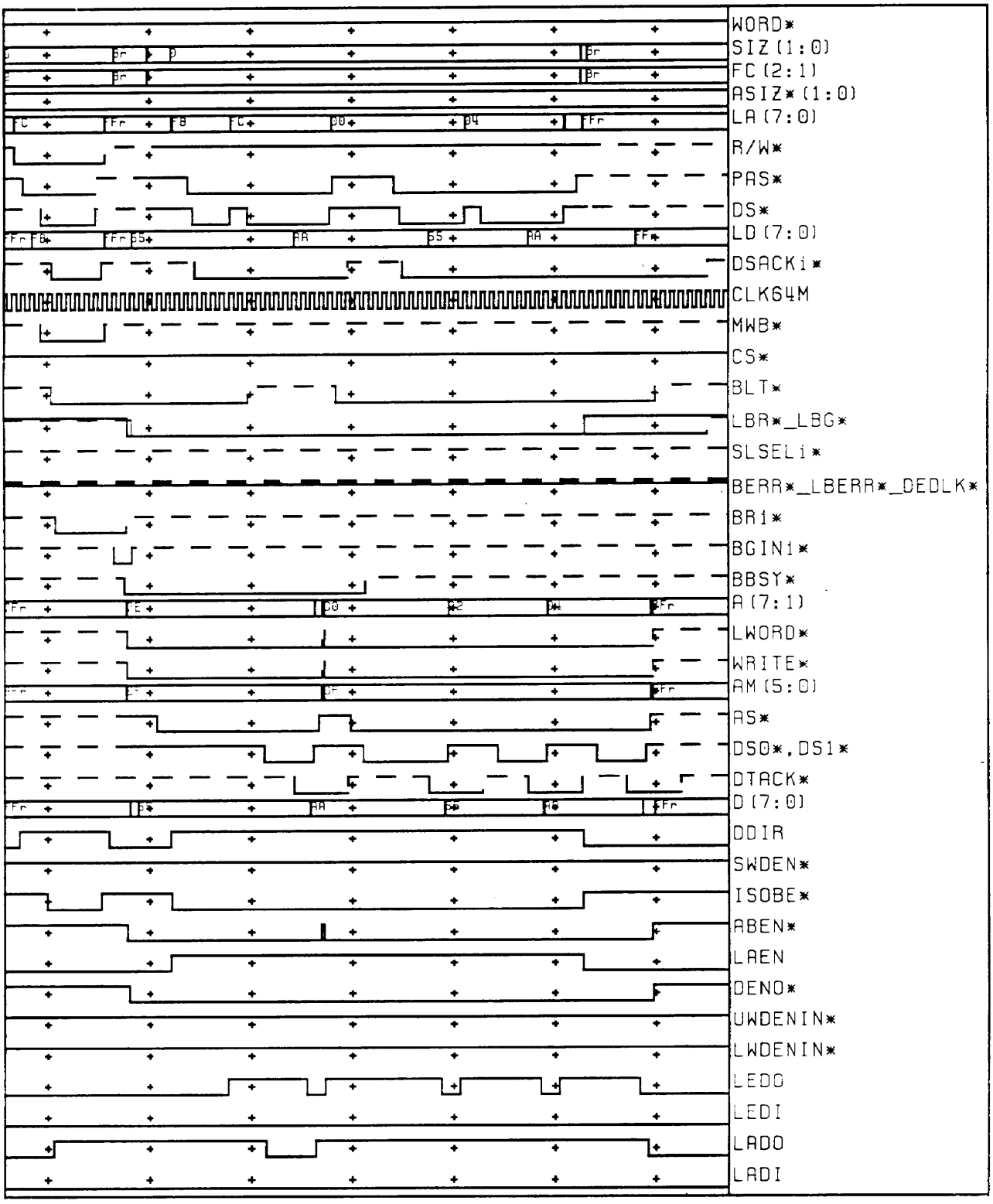


Figure 1-78. VME Boundary Crossing and Local Boundary Crossing

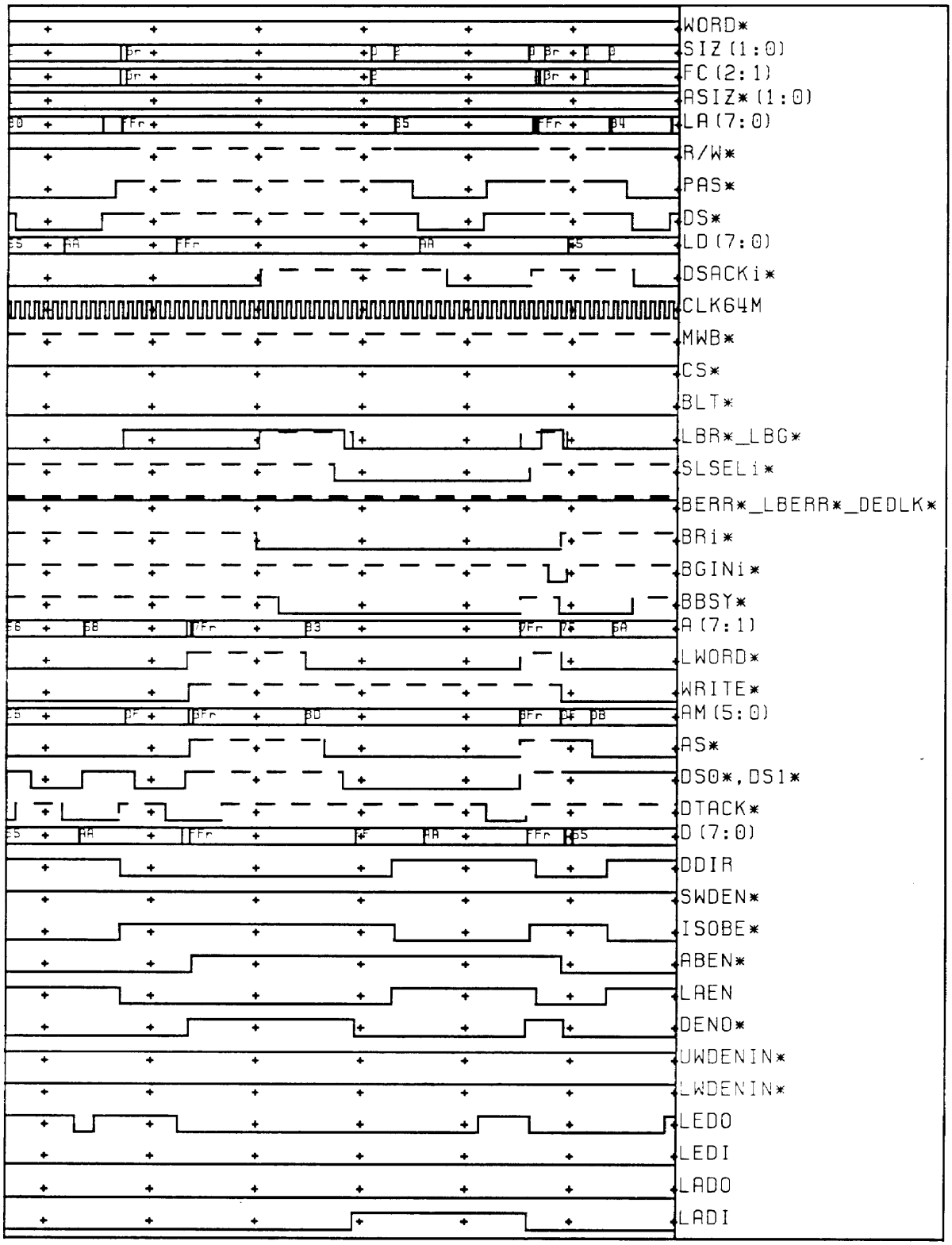


Figure 1-79. Slave Cycle During Interleave Period

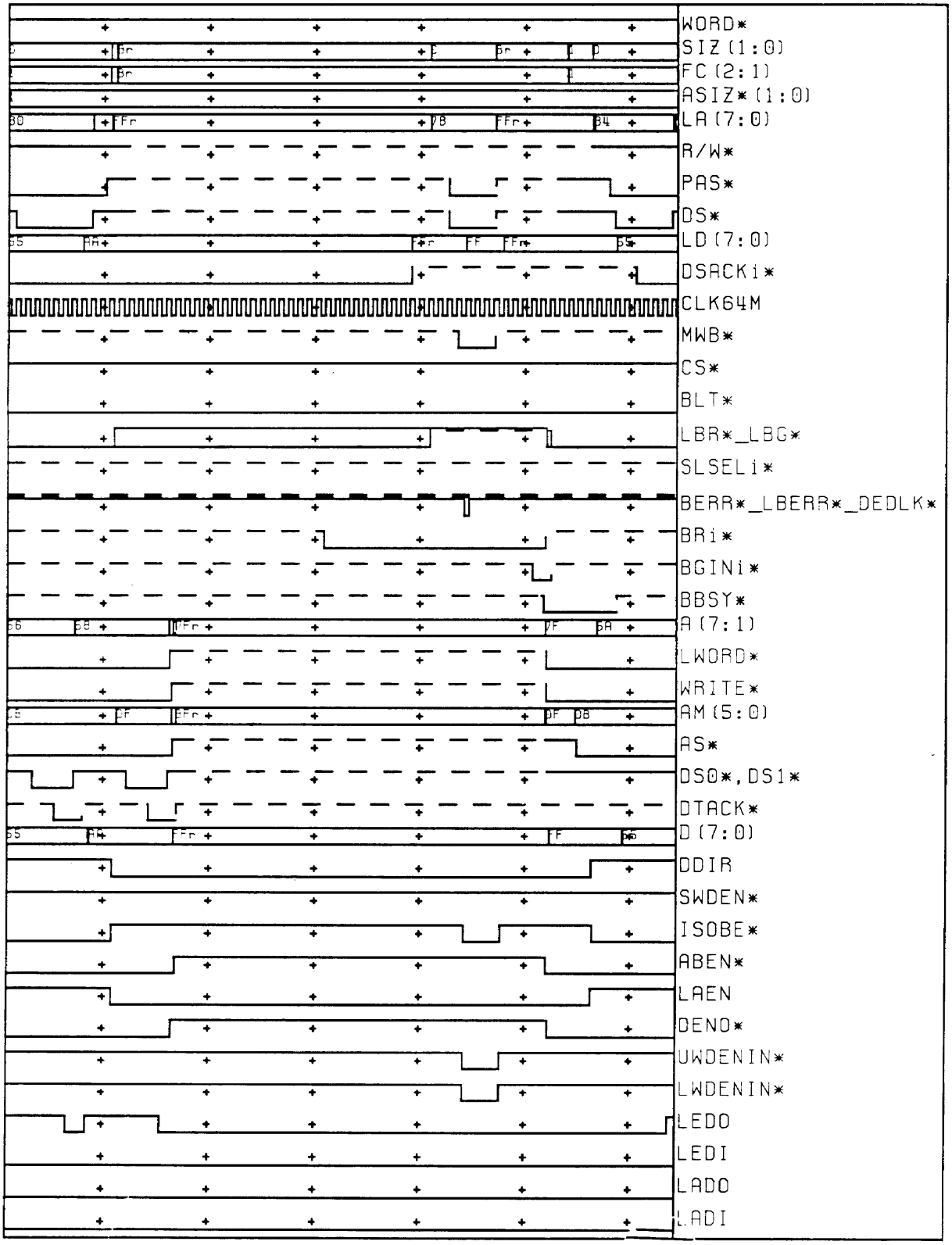


Figure 1-80. Master Cycle Deadlocked During Interleave

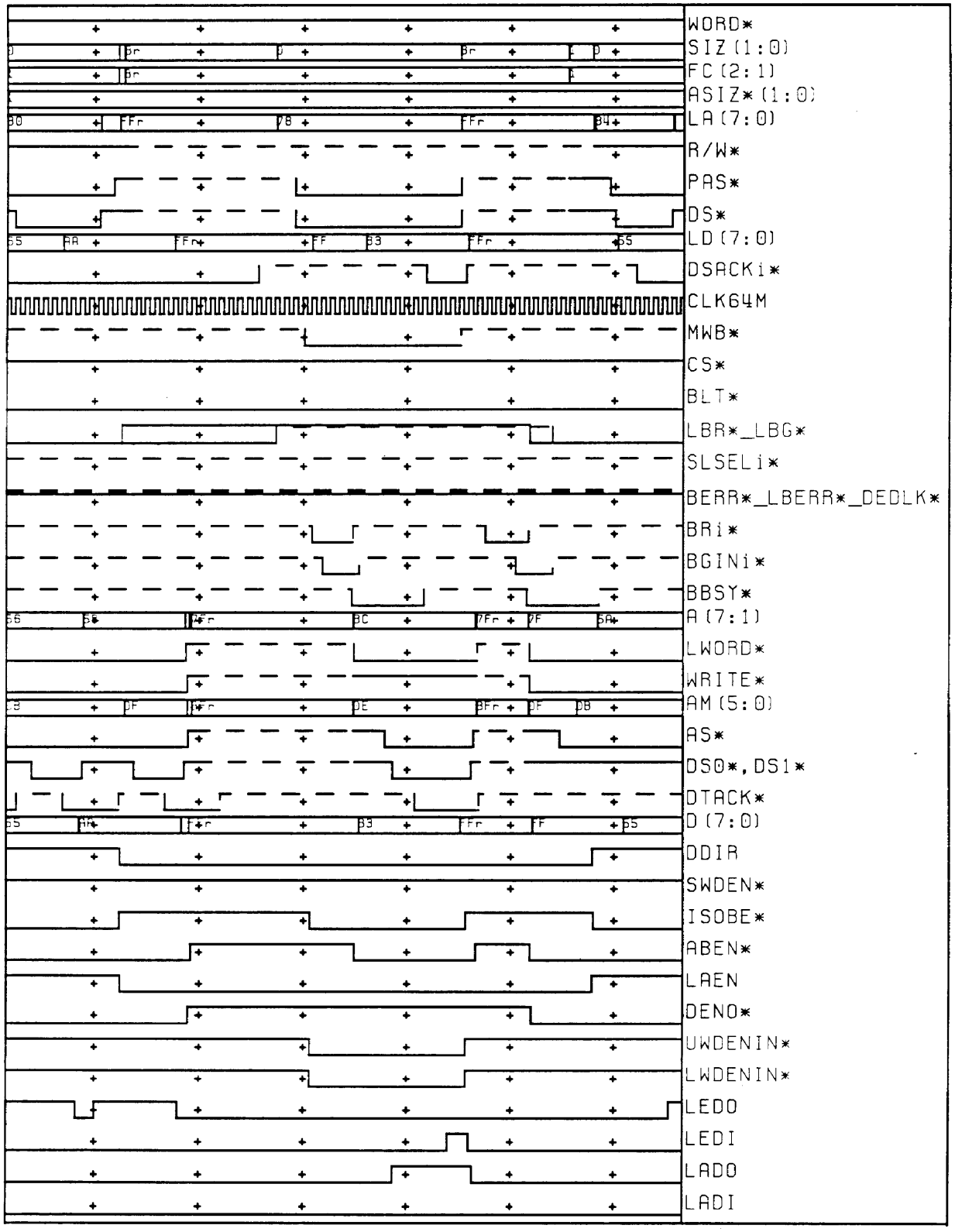


Figure 1-81. Master Cycle During Interleave with Dual-Path Option

1.16
DC Performance Specifications

Table 1-22. VMEbus Signals (AS*, DS1*, DS0*, BCLR*, SYSCLK)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 48 mA, 56 mA, 64 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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 1-23. VMEbus Signals (Low Drive. All VMEbus Daisy-Chain Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 1-24. VMEbus Signals (Medium Drive. All non-High, non-Low Drive Signals, All VAC068A VMEbus Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 1-25. Non-VMEbus Signals

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5		±10	µA

Table 1-26. Capacitance

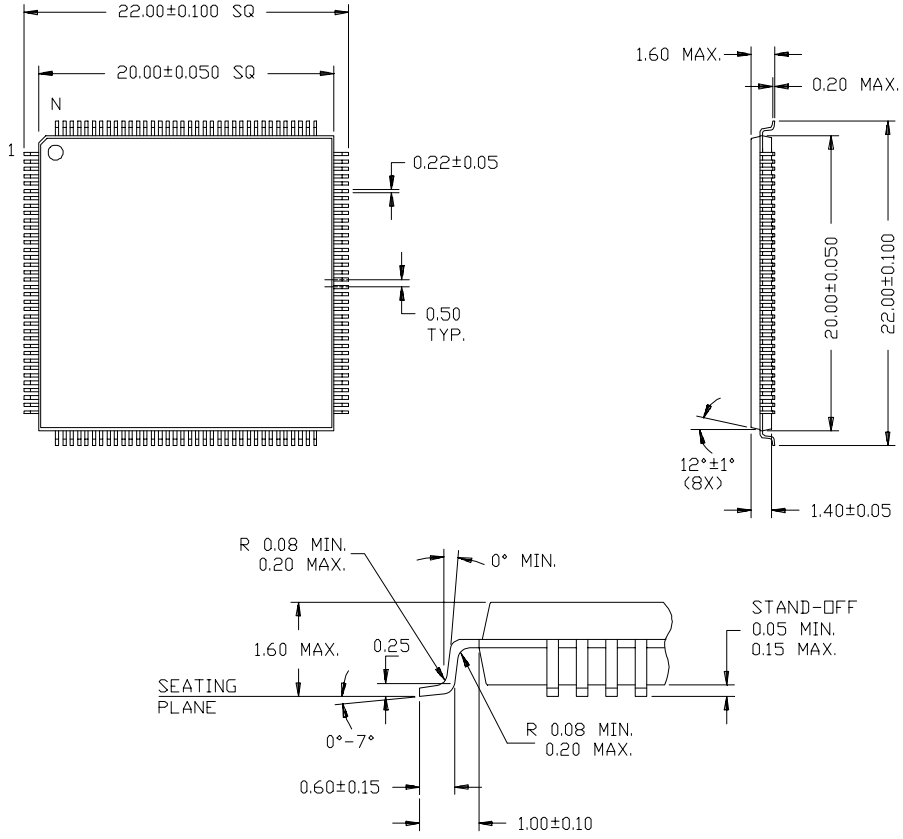
Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 64 MHz, V_{CC} = 5.0V	5	pF
C_{OUT}	Output Capacitance		7	pF

Table 1-27. Operating Current

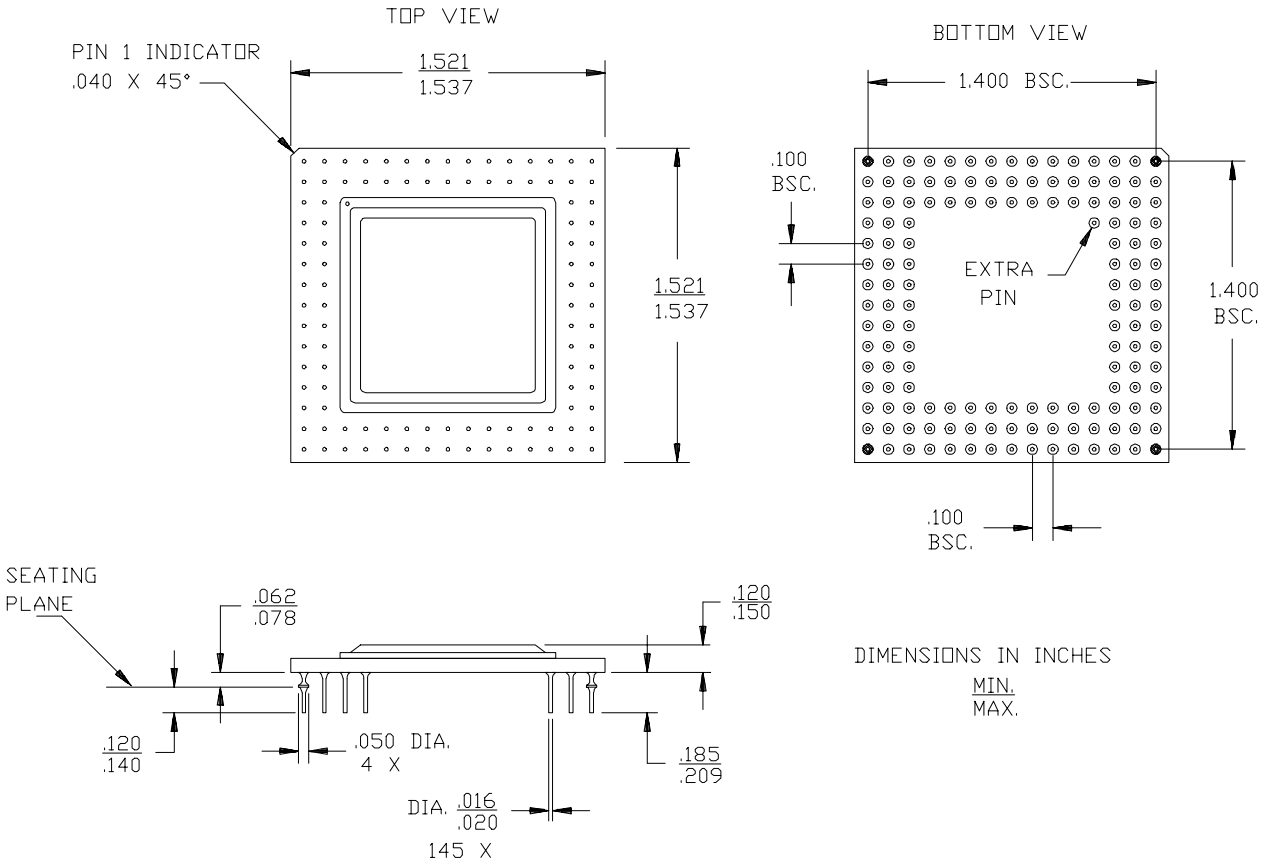
Parameters	Description	Test Conditions	Max.	Units
I_{DD}	Maximum Operating Current	No external DC load	150	mA

1.17
Package Diagrams

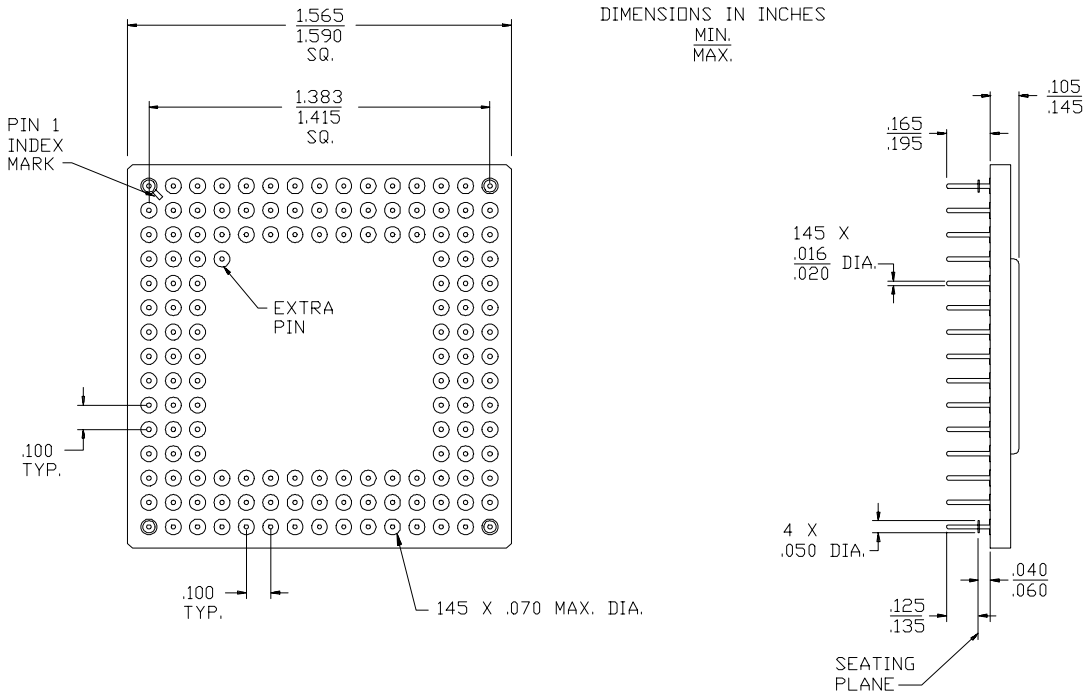
144-Pin Plastic Thin Quad Flat Pack (TQFP) A144



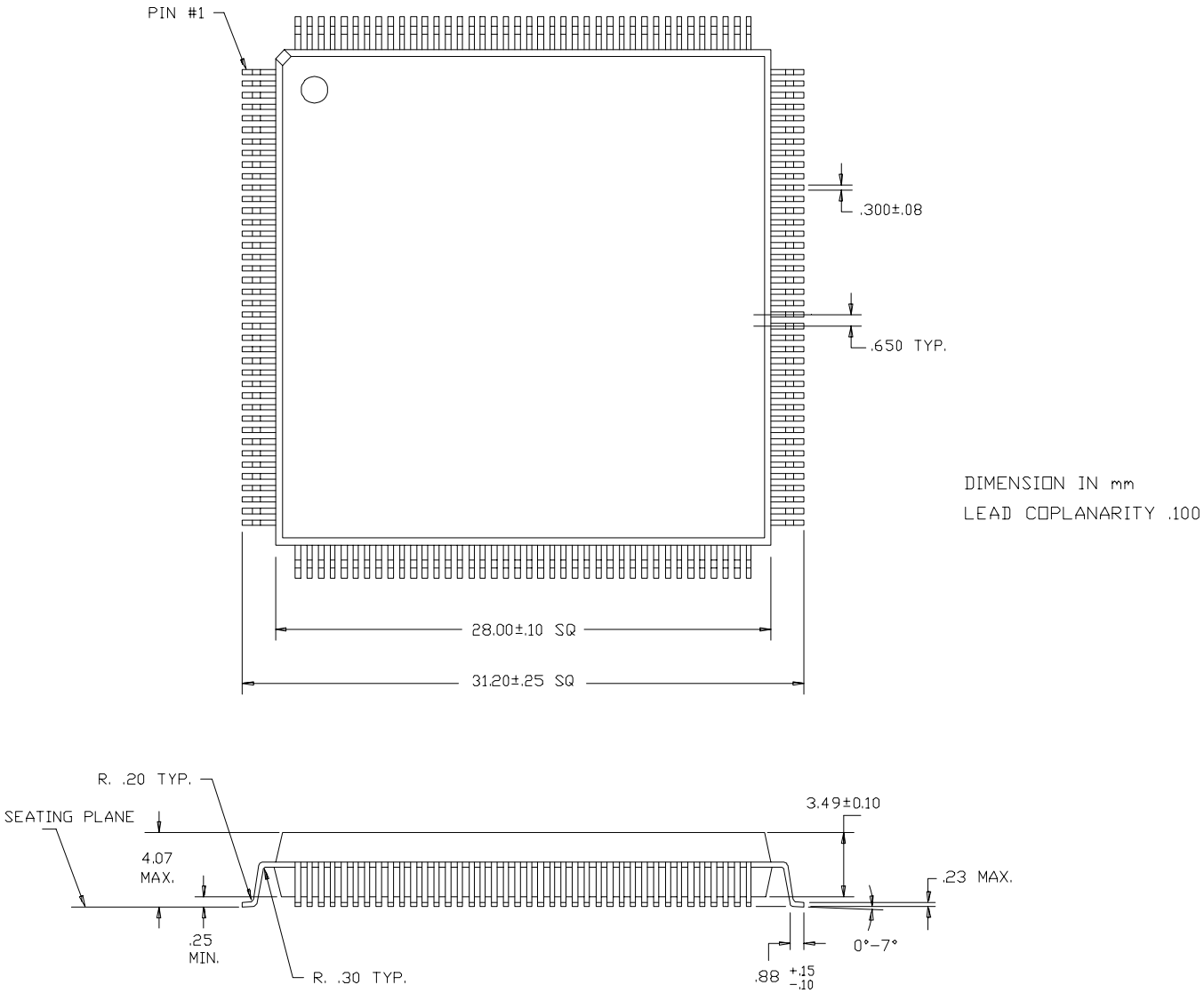
145-Pin Plastic Grid Array (Cavity Up) B144



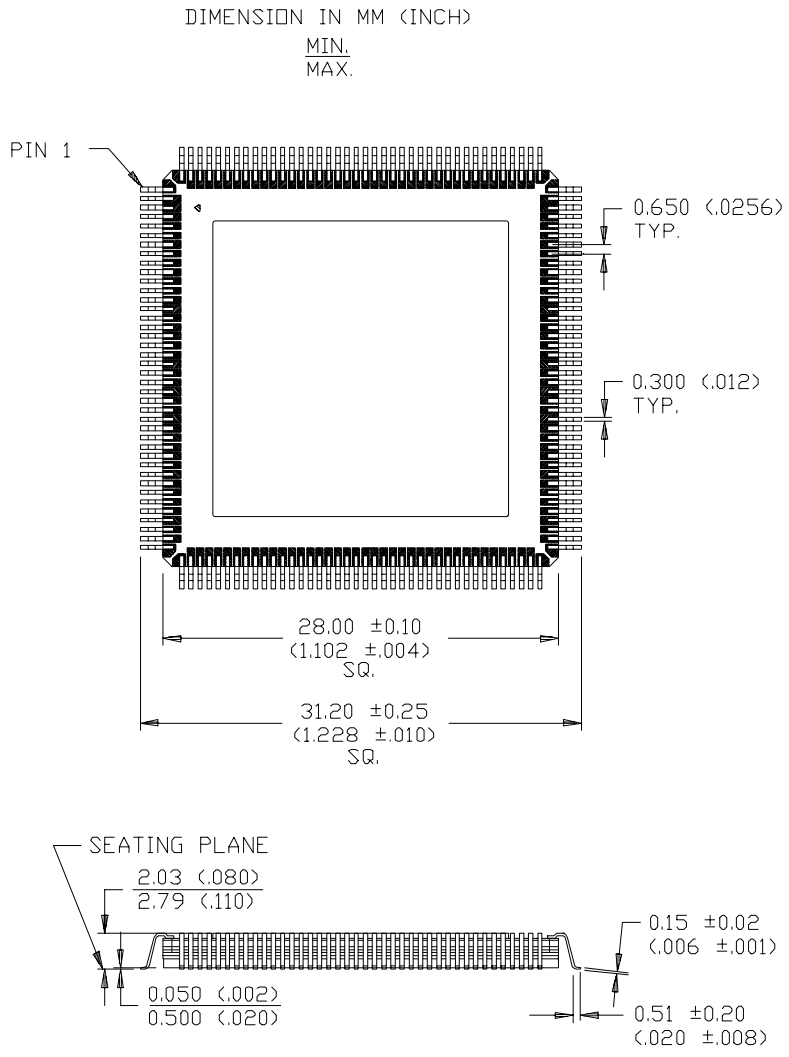
145-Pin Grid Array (Cavity Up) G145



160-Lead Plastic Quad Flatpack N160



160-Lead Ceramic Quad Flatpack (Cavity Up) U162



Section 2

The VIC64 VMEbus
Interface Controller

2.1 Introduction

The VIC64 is a member of the industry-standard VIC family of VMEbus interface products. The VIC64 implements 64-bit wide block transfers, in addition to 32- and 16-bit block transfers and 32-, 16-, and 8-bit single-cycle transfers, all using the same backplane hardware. VIC64 is software and hardware compatible with the VIC068A VMEbus Interface Controller.

This document provides the designer with the information needed to evaluate and develop VIC64-based boards. You should have already read the section on the VIC068A. This document provides information on the enhancements found within the VIC64. Another device, the CY7C964 Bus Interface Logic Chip, is described in Section 4 of this book.

Like the Cypress VIC068A, the VIC64 contains all the circuitry necessary to manage VMEbus transfers, either as a slave or a master. It can also be programmed as the VMEbus system controller. The VIC64 contains circuitry intended to minimize the problems associated with the development of a VMEbus interface such as an interrupt controller, a DMA controller, a DRAM refresh controller, and many other features normally on VMEbus boards. The VIC64 is a logical extension to the capabilities of the VIC068A, the industry-standard VMEbus Interface Controller chip, and it is fully compatible with the VIC068A.

The primary benefit of using the VIC64 is that you can perform 64-bit VMEbus transfers. The VIC64 also contains some enhancements to the VIC068A, including some performance improvements and additional features.

A board that has been designed to use the VIC068A is not likely to implement D64 VMEbus transfers, but there are several reasons why a user may choose to replace the VIC068A with a VIC64. For example, to take advantage of the enhancements of the VIC64, or to evaluate the device and use an existing board to speed the evaluation.

2.2
Compatibility

All pin assignments and register assignments are the same as those of the VIC068A, therefore the VIC64 will work flawlessly when used to replace a VIC068A. In fact, several of the VIC068's functions have been enhanced in the VIC64, allowing VIC068A applications to run faster in some cases. Naturally, some attention must be paid to the additional controls for the improved functionality to ensure that the original hardware and software supports the improvements. Therefore, several bits have been added to the VIC068A registers to control the enhancements. They map into the unused bits within the VIC068A register space; assuming that the VIC068A developer has not inadvertently set the bits, VIC068A code will run on the VIC64 without modification.

To add 64-bit functionality to the VIC family and still retain plug compatibility, the SCON* pin has been modified in the VIC64. Whereas previously it performed only the input function of selecting VIC068A to be the system controller, in VIC64 the pin is sensed and latched during reset to determine whether the system controller function is enabled. After reset the pin becomes an output to control external circuitry during 64-bit transfers. The new name for the pin is SCON*/D64. If you simply replace the VIC068A with the VIC64, the VIC64 will function in an identical manner to the VIC068A, whether it is the system controller or not. It is recommended that the VIC64 SCON*/D64 pin be connected via a resistive pull-down/up of greater than 4.7 kΩ to enable/disable the system controller function.

2.3
64-Bit Operations

2.3.1 VMEbus Specification

The primary reason for the development of the VIC64 was to provide users with the capability to perform 64-bit-wide data transfers in a manner consistent with the goals of the 64-bit VMEbus specification, more commonly known as the VME64 specification. The protocol for 64-bit MBLT (Multiplexed Block Transfer) is specified in an ANSI document VITA1-1994. The VIC64 implements this protocol.

2.3.2 Address Modifier Codes

VIC64 responds as a slave to the address modifier (AM) codes associated with MBLT transfers as follows:

$3C, $38, $0C, $08; Performs D64 operation as implied by the actual AM code, and the contents of the Slave configuration Registers $C3 and $CB, and Block Transfer Definition Register $AB.

$00-$07; No response: these codes are associated with 64-bit address operations, and the VIC64 does not support 64-bit address operation.

As a master, VIC64 will use the MBLT protocol to transfer data if the appropriate conditions occur:

AM codes $3C, $38, $0C, $08 are selected, and the appropriate bits of the configuration registers are set (see later for exact details).

In summary, VIC64 performs A32/D64 and A24/D64 operations in addition to the D8/16/32 single cycles and A16/24/32..D16/32 block transfers performed by VIC068A.

2.3.3 Boundary Crossing

There are several implications of the 64-bit VMEbus protocol and the requirement for compatibility that you should consider. The VIC64, being a 144-pin device, can connect to only the lower byte of the VMEbus address. For block transfers other than D64 transfers, the VMEbus specification requires that the VMEbus address be rebroadcast at 256-byte boundary crossings; this quantity maps neatly into the byte of address that the VIC64 can monitor. MBLT transfers, however, are required to rebroadcast the address only at 2-Kbyte boundaries. VIC64 has no means of determining how the starting address of a master block transfer

relates to the 2K boundary (it has access to only the lower 8 address bits), and therefore VIC64 rebroadcasts the address at every 256-byte boundary. This is still compatible with the specification, but has a small impact on the sustained transfer rate. If you wish to take advantage of the increased performance of 2-Kbyte boundaries, then VIC64 can be programmed to rebroadcast the address every 2048 bytes, and the starting address must then be aligned on a 2-Kbyte boundary.

2.3.4 External Circuit Complexity

The VIC64 is a flexible building block that can be used in many different configurations. The VMEbus specification is written to allow many levels of circuit complexity to conform to the specification. Such circuitry may include slave address decode circuitry, local DMA transfer, slave read modify cycles, and more. The VIC64 and the VIC068A provide dual-path operation, a mechanism whereby the local bus master can perform single-cycle VMEbus operations during the time that the VMEbus is between block transfer bursts (interleave period). They also provide a mechanism allowing master write-posting, and slave read modify cycles to occur concurrently without harming the posted data. All this circuitry must be duplicated externally for the higher-order data byes if you want these features.

You may choose to implement only those features that your system requires, thereby simplifying the necessary external circuitry. Alternatively, the user may decide to use the companion device, the CY7C964, and gain access to all the features using only three small devices. The CY7C964 is described in Section 4, The CY7C964 Bus Interface Logic Circuit.

2.4
VIC64: Additional Information

The following sections are related to Section 1, The VIC068A VMEbus Interface Controller. The chapter numbers are those chapters of Section 1 that require clarification or modification for the VIC64. All other information in Section 1 is applicable to the VIC64.

2.4.1 VIC64 Signal Description (Chapter 1.2)

All pins are identical to those of the VIC068A with the following exception:

SCON*/D64

Input: Yes
Output: Yes
Drive: 16 mA

This is the dual-function signal by which the VIC64 determines whether system controller functions are required, and by which the VIC64 controls external logic during 64-bit VMEbus transfers. During the time that RESET* is asserted, this pin is the SCON* input whose state is latched internally when RESET* goes inactive. A Low state causes VIC64 to become the VMEbus system controller. During the time that RESET* is inactive, this pin becomes the D64 output whose state is normally Low, becoming High only during the Data Phase of D64 transactions.

2.4.2 System Controller Operations (Chapter 1.4)

The VIC64 functions identically to VIC068A as a system controller, except that the SCON* pin of the VIC068A has been renamed to be SCON*/D64 on the VIC64. During the period that RESET* is asserted (Low), VIC64 assumes that the pin is an input whose state is latched on the rising edge of RESET*. The latched state is then used to determine whether the VIC64 is the system controller: if the state is Low, then the VIC64 is the system controller.

SCON*/D64 becomes an output after the rising edge of RESET*; the state of the output is used to enable 64-bit data transfers (Chapter 1.10, Block Transfer Functions contains information on this operation).

2.4.3 VMEbus Master Operations (Chapter 1.5)

The VIC64 does not perform single-cycle 64-bit transfers.

The VIC64 uses the same pins and register bits as the VIC068A to configure and select 64-bit block transfers. The release modes are identical, and the address broadcast phase is identical to the VIC068A, except that the AM code reflects the D64 transaction (see *Table 2-1*).

Table 2-1. Master Transfer AM Code Control Map for D64 Operations

VIC64 Master Access Inputs				VIC64 AM Code Output	
ASIZ1/0	**Address Size**	**Blk**	**FC2**	**Operation Type**	**AM[5:0]**
01	A32 addressing	Yes	0X	User block	$08
01	A32 addressing	Yes	1X	Supervisory block	$0C
11	A24 addressing	Yes	0X	User block	$38
11	A24 addressing	Yes	1X	Supervisory block	$3C

As the VIC64 has an identical local bus interface to that of the VIC068A, some mention must be made of the protocol used to transfer the 64-bit VMEbus data to the 32-bit local bus. First, it should be noted that Byte(0) is transferred on VMEbus address [A31:A24], and Byte(7) is transferred on VMEbus data [D7:D0]. Two local transactions are required for each VMEbus transaction. For maximization of performance, a pipelined architecture is used. The VIC64 provides the appropriate timing for latch controls to implement the pipe externally for those bytes that the VIC64 itself does not connect to.

2.4.3.1 D64 Master Write Cycles

In the case of master write cycles, the first local cycle fetches the first [Byte(0)–Byte(3)] longword and the VIC64 places it into a two-stage pipe: the next local cycle fetches the next longword and presents it to the VMEbus data bus, while the piped data is presented to the VMEbus address bus. Then the next local cycle can commence without waiting for the completion of the VMEbus cycle, as the first stage of the pipe is now free. See the timing diagrams for full details of this operation.

2.4.3.2 D64 Master Read Cycles

In the case of master read cycles, 64 bits of VMEbus data are latched under the control of VIC64. The [Byte(4)–Byte(7)] longword is placed into a two-stage pipe, while the [Byte(0)–Byte(3)] longword is presented to the local bus. After the local bus write cycle, the piped data is then presented to the local bus, and the next VMEbus cycle can commence as the first stage of the pipe is now free. See the timing diagrams for full details of this operation.

2.4.4 VMEbus Slave Operations (Chapter 1.6)

Upon detecting SLSEL0* or SLSEL1* asserted, the VIC64 behaves in an identical manner to the VIC068A except that if the AM code for the slave transaction is $08, $0C, $38, or $3C, the VIC64 configures itself for a D64 slave block transfer (see *Table 2-2*).

Table 2-2. Slave Transfer AM Code Control Map for D64 Operations

VIC64 AM Code Inputs		VIC64 Master Access Outputs		
Operation Type	**AM[5:0]**	**Address Size**	**Blk**	**FC2/1**
User block	$08	A32 addressing	Yes	00
Supervisory block	$0C	A32 addressing	Yes	00
User block	$38	A24 addressing	Yes	00
Supervisory block	$3C	A24 addressing	Yes	00

2.4.4.1 D64 Slave Read Cycles

As in the case of master write cycles, the first local cycle fetches the first [Byte(0)-Byte(3)] longword and the VIC64 places it into a two-stage pipe. The next local cycle fetches the next longword and presents it to the VMEbus data bus, while the piped data is presented to the VMEbus address bus. Then the next local cycle can commence without waiting for the completion of the VMEbus cycle, as the first stage of the pipe is now free.

2.4.4.2 D64 Slave Write Cycles

As in the case of master read cycles, 64 bits of VMEbus data are latched under the control of VIC64. The [Byte(4)–Byte(7)] longword is placed into a two-stage pipe, while the [Byte(0)–Byte(3)] longword is presented to the local bus. After the local bus write cycle, the piped data is then presented to the local bus, and the next VMEbus cycle can commence as the first stage of the pipe is now free.

2.4.5 Interrupts (Chapter 1.9)

The VIC64 can be programmed to perform either D8, D16, or D32 interrupt acknowledge cycles. The method by which this is performed is simply to drive the values on SIZ1/0, in a similar fashion to a master read or write operation. The SIZ1/0 lines are sensed by the VIC64 following the assertion of FCIACK* by the local processor. Note that no provision is made for non-aligned status/ID vector: The VIC64 enables the appropriate local bus drivers for either 8-, 16-, or 32-bit Status/ID.

Table 2-3. VIC64 Interrupt Acknowledge Cycle Selection

SIZ1/0	VMEbus Data Width
00	32
01	8
10	16
11	32

2.4.6 VIC64 Block Transfer Functions (Chapter 1.10)

As the VIC64 is a superset of the VIC068A, all the VIC068A block transfer functionality is reproduced in the VIC64. The additional features provided by the VIC64 are D64 transfers and performance enhancements.

2.4.6.1 D64 Transfers, VMEbus Boundary Crossing

The VME64 specification allows D64 block transfers to exceed the 256-byte boundary-crossing limitation that the original VMEbus specification contains. The new specification allows for 2-Kbyte boundaries. As the VIC64 can only discern 8 bits of address, it has no means of determining which 256-byte boundary is the 2048-byte boundary, and therefore the VIC64 rebroadcasts the address every 256 bytes unless BTDR[7] is set: this bit causes the address to be rebroadcast on 2-Kbyte boundaries, but the VIC64 then assumes that the transfer starts on the 2-Kbyte boundary.

2.4.7 Miscellaneous Features (Chapter 1.11)

2.4.7.1 Selection of System Controller Functionality

The VIC068A is configured to be system controller by strapping SCON* Low. In VIC64, the SCON"*/D64 pin performs this function: the state of the pin is latched during any of the possible Reset operations, and this state determines whether VIC64 is system controller. Following the Reset operation, the SCON*/D64 pin becomes an output whose state controls the external circuitry (such as the CY7C964) used during the data phase of D64 transfers. The detailed timing of this operation depends upon internal states such as DRAM refresh timing, in addition to the external stimuli such as SYSRESET*, IRESET*, and IPL0. Use of an external pull-up/pull-down resistor to determine the state of the SCON* pin during the Reset operation is all that is required to ensure correct operation.

2.4.7.2 Enhanced Turbo Mode

In addition to the use of ICR[1], another performance enhancement is possible in the VIC64. Setting BTDR[5] reduces the DSACK*-to-DTACK* time defined in the slave select control registers by 0.5 clock period for both master and slave block transfers. The reduced times are 0, 1.5, 2, 2.5, ..., 8.5 clock periods. See the AC Timing Parameters section for details on which times are affected by this bit.

2.4.8 Register Map and Descriptions (Chapter 1.12)

There are some differences between the VIC068A and the VIC64 register assignments and contents.

2.4.8.1 Interprocessor Communications Register 5

Name:	ICR5
Address:	$77
Description:	This register provides the VIC64 revision number.

2.4.8.2 Block Transfer Definition Register

Name:	BTDR
Address:	$AB
Description:	Configures master block transfers for boundary crossing, dual-path and user defined address modifiers. There are four additional bits defined for VIC64:
Bit 4 (0/0/0)	Enables D64 Master Operations when BTCR[6] is set
Bit 5 (0/0/0)	Enables Accelerated Block Transfer Operations as discussed above.
Bit 6 (0/0/0)	Enables D64 Slave Operations
Bit 7 (0/0/0)	Enables 2-Kbyte boundary crossing for D64 Master Operations. If this bit is set, VIC64 assumes that the transfer is aligned to a 2-Kbyte boundary.

2.4.8.3 Release Control Register

Name:	RCR
Address:	$D3
Description:	This register configures the VMEbus release mode, and the burst length for block transfers with local DMA.
Bits 5–0 (0/0/0)	For MBLT operations (D64 transfers), the burst length is 4 times the actual field contents. A value of 0 is interpreted to mean 4 x 64.

For non-MBLT operations, the burst length is simply the field contents. A value of 0 in this field is interpreted to mean 64.

2.4.8.4 Block Transfer Length Register 2

Name:	BTLR2
Address:	$E7
Description:	This register provides the most significant byte of the 24-bit value used to determine the byte count for block transfers with local DMA
Bits 7–0 (0/0/0)	Bits 23:16 of the block transfer length.

2.4.9 AC Performance Specifications (Chapter 1.13)

AC Timing for D64 Operations[1]

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
Master D64 Block Transfer with Local DMA (Initiation Cycle)[2]								
A1	MWB*[0] & PAS*[0] & DS*(0) to BRi*[L]		T+7	2T+32	T+7	2T+33	4T+5	5T+38
A2	MWB*[0] & PAS*[0] & DS*(0) to LADO[H]		T+9	2T+31	T+8	2T+32	T+8	2T+38
A3	MWB*[0] & PAS*[0] & DS*(0) to BLT*[L]		T+9	2T+26	T+8	2T+27	T+8	2T+30
A4	MWB*[0] & PAS*[0] & DS*(0) to DSACK1*[L]		T+11	2T+46	T+10	2T+48	T+10	2T+54
A5	MWB*[0] & PAS*[0] & DS*(0) to DSACK0*[L]		T+11	2T+46	T+10	2T+48	T+10	2T+54
Master D64 Block Transfer Address Broadcast Cycle[2]								
B1	DTACK*[0] to LBR*[L]		24	65	20	69	20	75
B2	DTACK*[0] to DSi*[H]		8	24	7	26	7	30
B3	DTACK*[0] to SCON*/ D64[H]		16	59	15	62	15	70
Master D64 Block Transfer with Local DMA (Write)								
** First Longword Fetch **								
C1	DSACKi*[0] and DS*[L] to DS*[H]	2, 3, 4	MBAT0+8	MBAT0+T+ 36	MBAT0+7	MBAT0+T+ 38	MBAT0+7	MBAT0+T+ 42
C2	DSACKi*[0] and DS*[L] to LEDO[H]	2, 3, 4	MBAT0+7	MBAT0+T+ 33	MBAT0+6	MBAT0+T+ 35	MBAT0+6	MBAT0+T+ 36
C3	DSACKi*[0] and DS*[L] to LA(7:0)	2, 3, 4	MBAT0+ .5T+9	MBAT0+2T+ 30	MBAT0+ .5T+8	MBAT0+2T+ 32	MBAT0+ .5T+8	MBAT0+2T+ 35
C4	DS*(H) to DS*[L]	2, 3, 4, 5	T+8	3T+31	T+7	3T+32	T+7	3T+35
** Second Longword Fetch **								
C5	DSACKi*[0] and DS*[L] to DS*[H]	2, 4	MBAT1+14	MBAT1+T+ 46	MBAT1+13	MBAT1+T+ 48	MBAT1+13	MBAT1+T+ 52
C6	DSACKi*[0] and DS*[L] to DENO*[L]	4	MBAT1+11	MBAT1+T+ 37	MBAT1+10	MBAT1+T+ 39	MBAT1+10	MBAT1+T+ 42
C7	DSACKi*[0] and DS*[L] to LA(7:0)	2, 4	MBAT1+ .5T+9	MBAT1+2T+ 31	MBAT1+ .5T+8	MBAT1+2T+ 32	MBAT1+ .5T+8	MBAT1+2T+ 35
C8	DSACKi*[0] and DS*[L] to DSi*[L]	6	MBAT1+ 3T+12	MBAT1+4T+ 32	MBAT1+ 3T+10	MBAT1+4T+ 33	MBAT1+ 3T+10	MBAT1+4T+ 36
C9	DSACKi*[0] and DS*[L] to LEDO[L]	4	MBAT1+16	MBAT1+T+ 56	MBAT1+15	MBAT1+T+ 57	MBAT1+15	MBAT1+T+ 64
C10	DS*(H) to DS*[L]	2, 4, 5	T+8	3T+31	T+7	3T+32	T+7	3T+35
C11	DTACK*[0] to DSi*[H]		7	22	6	23	6	25
C12	DTACK*[0] to DENO*[H]	2	8	24	7	26	7	30

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
Master Block Transfer with Local DMA (Read)								
** First Longword Write **								
D1	LBG*[0] to DENIN*[L]	2	2T+11	3T+41	2T+10	3T+43	2T+10	3T+48
D2	DTACK*[0] to LEDI[H]	7	2T+6	3T+23	2T+5	3T+24	2T+5	3T+27
D3	DTACK*[0] to DSi*[H]	2, 6	2T+9	3T+28	2T+8	3T+29	2T+8	3T+32
D4	DTACK*[0] to DS*[L]	6	2T+13	3T+36	2T+12	3T+37	2T+12	3T+41
D5	DSACKi*[0] and DS*[L] to DS*[H]	2, 3, 4	MBAT0+8	MBAT0+T+37	MBAT0+7	MBAT0+T+38	MBAT0+7	MBAT0+T+42
D6	DSACKi*[0] and DS*[L] to LEDI[L]	3, 4	MBAT0+13	MBAT0+T+52	MBAT0+12	MBAT0+T+53	MBAT0+12	MBAT0+T+60
D7	DSACKi*[0] and DS*[L] to DENIN1*[L]	3, 4	MBAT0+8	MBAT0+T+35	MBAT0+7	MBAT0+T+36	MBAT0+7	MBAT0+T+41
D8	DSACKi*[0] and DS*[L] to LA(7:0)	2, 3, 4	MBAT0+.5T+9	MBAT0+2T+29	MBAT0+.5T+8	MBAT0+2T+31	MBAT0+.5T+8	MBAT0+2T+35
D9	DSACKi*[0] and DS*[L] to DSi*[L]	3, 4	MBAT0+22	MBAT0+T+56	MBAT0+20	MBAT0+T+58	MBAT0+20	MBAT0+T+64
D10	DS*[H] to DS*[L]	2, 4, 5	T+8	3T+29	T+7	3T+31	T+7	3T+35
** Second Longword Write **								
D12	DSACKi*[0] and DS*[L] to DS*[H]	2, 4	MBAT1+8	MBAT1+T+36	MBAT1+7	MBAT1+T+38	MBAT1+7	MBAT1+T+42
D13	DSACKi*[0] and DS*[L] to DENIN1*[H]	2, 4	MBAT1+16	MBAT1+T+56	MBAT1+15	MBAT1+T+59	MBAT1+15	MBAT1+T+66
D14	DSACKi*[0] and DS*[L] to LA(7:0)	2, 4	MBAT1+.5T+9	MBAT1+2T+29	MBAT1+.5T+8	MBAT1+2T+31	MBAT1+.5T+8	MBAT1+2T+35
D15	DSACKi*[0] and DS*[L] to LD(7:0)	2, 4	MBAT1+.5T+12	MBAT1+2T+39	MBAT1+.5T+10	MBAT1+2T+42	MBAT1+.5T+10	MBAT1+2T+48
Master D64 Block Transfer with Local DMA (Boundary Crossing)[2]								
E1	DS*[0] to BLT*[H]		3	28	2	30	2	33
E2	DS*[1] to BLT*[L]		3	19	2	20	2	21
E3	DSi*[0] to LADO first transition		3	19	3	20	2	21
E4	DSi*[0] to LADO second transition		3	19	3	20	2	21
Slave D64 Block Transfer Address Broadcast Cycle[2]								
F1	DSi*[1] to LBR*[L]		11	36	10	39	10	42
F2	DSi*[0] to DTACK*[L]		2T+9	3T+28	2T+8	3T+29	2T+8	3T+32
F3	DSi*[1] to DTACK*[H]		9	28	8	35	8	39
F4	DSi*[1] to SCON*/D64[H]		10	33	9	35	9	39
F5	DSi*[0] and AS*[0] and SLSELi*[0] to LADI[H]		1.5T+5	2T+25	1.5T+4	2T+26	1.5T+4	2T+29
Slave D64 Block Transfer (Write)								
** First Longword Cycle **								
G1	DSi*[0] to DS*[L]	2, 4	3T+11	4T+38	3T+10	4T+40	3T+10	4T+44
G2	DSi*[0] to DTACK*[L]	6	2T+9	3T+23	2T+8	3T+24	2T+8	3T+26
G3	DSi*[0] to LEDI[H]	2, 4	T+11	2T+37	T+10	2T+39	T+10	2T+42

Operation		Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
G4	DSACKi*[0] and DS*[L] to DS*[H]	2, 4, 8	SBAT0+8	SBAT0+T+36	SBAT0+7	SBAT0+T+38	SBAT0+7	SBAT0+T+42
G5	DSACKi*[0] and DS*[L] to LEDI[L]	2, 4, 8	SBAT0+13	SBAT0+T+52	SBAT0+12	SBAT0+T+54	SBAT0+12	SBAT0+T+60
G6	DSACKi*[0] and DS*[L] to DENIN1*[L]	2, 4, 8	SBAT0+8	SBAT0+T+31	SBAT0+7	SBAT0+T+33	SBAT0+7	SBAT0+T+37
G7	DSACKi*[0] and DS*[L] to LA(7:0)	2, 4, 8	SBAT0+.5T+9	SBAT0+2T+29	SBAT0+.5T+8	SBAT0+2T+31	SBAT0+.5T+8	SBAT0+2T+35
G8	DS*[H] to DS*[L]	2, 4, 5	T+8	3T+31	T+7	3T+32	T+7	3T+35
** Second Longword Cycle[2, 4] **								
G9	DSACKi*[0] and DS*[L] to DENIN1*[H]		SBAT1+20	SBAT1+T+64	SBAT1+19	SBAT1+T+67	SBAT1+19	SBAT1+T+74
G10	DSACKi*[0] and DS*[L] to DS*[H]		SBAT1+11	SBAT1+T+34	SBAT1+10	SBAT1+T+36	SBAT1+10	SBAT1+T+42
G11	DSACKi*[0] and DS*[L] to LA(7:0)		SBAT1+.5T+9	SBAT1+2T+29	SBAT1+.5T+8	SBAT1+2T+31	SBAT1+.5T+8	SBAT1+2T+35
G12	DSACKi*[0] and DS*[L] to LD(7:0)		SBAT1+.5T+11	SBAT1+2T+40	SBAT1+.5T+10	SBAT1+2T+42	SBAT1+.5T+10	SBAT1+2T+48
Slave D64 Block Transfer (Read)								
** First Longword Cycle **								
H1	DSACKi*[0] and DS*[L] to LEDO[H]	4, 8	SBAT0+7	SBAT0+T+36	SBAT0+6	SBAT0+T+37	SBAT0+6	SBAT0+T+41
H2	DSACKi*[0] and DS*[L] to DS*[H]	4, 8	SBAT0+8	SBAT0+T+39	SBAT0+7	SBAT0+T+41	SBAT0+7	SBAT0+T+45
H3	DSACKi*[0] and DS*[L] to LA(7:0)	2, 4, 8	SBAT0+.5T+9	SBAT0+T+29	SBAT0+.5T+8	SBAT0+T+31	SBAT0+.5T+8	SBAT0+T+35
H4	DS*[H] to DS*[L]	2, 4, 5	T+8	3T+30	T+7	3T+32	T+7	3T+35
** Second Longword Cycle **								
H5	DSACKi*[0] and DS*[L] to LEDO[L]	4	SBAT1+19	SBAT1+T+64	SBAT1+18	SBAT1+T+67	SBAT1+18	SBAT1+T+74
H6	DSACKi*[0] and DS*[L] to DENO*[L]	4	SBAT1+11	SBAT1+T+37	SBAT1+10	SBAT1+T+39	SBAT1+10	SBAT1+T+42
H7	DSACKi*[0] and DS*[L] to DS*[H]	2, 4	SBAT1+24	SBAT1+T+72	SBAT1+23	SBAT1+T+75	SBAT1+23	SBAT1+T+85
H8	DSACKi*[0] and DS*[L] and DSi*[0] to DTACK*[L]	4	SBAT1+13	SBAT1+T+33	SBAT1+12	SBAT1+T+34	SBAT1+12	SBAT1+T+38
H9	DSACKi*[0] and DS*[L] to LA(7:0)	2, 4	SBAT1+.5T+9	SBAT1+2T+29	SBAT1+.5T+8	SBAT1+2T+31	SBAT1+.5T+8	SBAT1+2T+35
H10	DS1/0*[1] to DENO*[H]	2	6	21	5	22	5	25
Slave D64 Block Transfer (Boundary Crossing)								
H11	DS*[0] to LADI[L]	2	11	26	10	28	10	32
H12	DS*[1] to LADI[H]	2	6	13	5	14	5	15

Notes:

1. All minimum times are guaranteed, not tested.
2. These timings are specified for information, but not tested.
3. For second and all subsequent longword fetches, MBAT1 is used in the timing equations.
4. When the Enhanced Turbo Bit is set, all these times are reduced by 0.5T.
5. Min. and Max. Times are programmable: see Register Descriptions.
6. When the Enhanced Turbo Bit is set, these times become MBAT1+.5T+D min., MBAT1+1.5T+D max.
7. When the Enhanced Turbo Bit is set, all these items are reduced to 0.5T min., 1.0T max., plus appropriate asynchronous delay from the table. Minimum times reflect unloaded device pins. Actual in-system delays will be in accordance with the VMEbus specification.
8. For second and all subsequent longword fetches, SBAT1 is used in the timing equations.

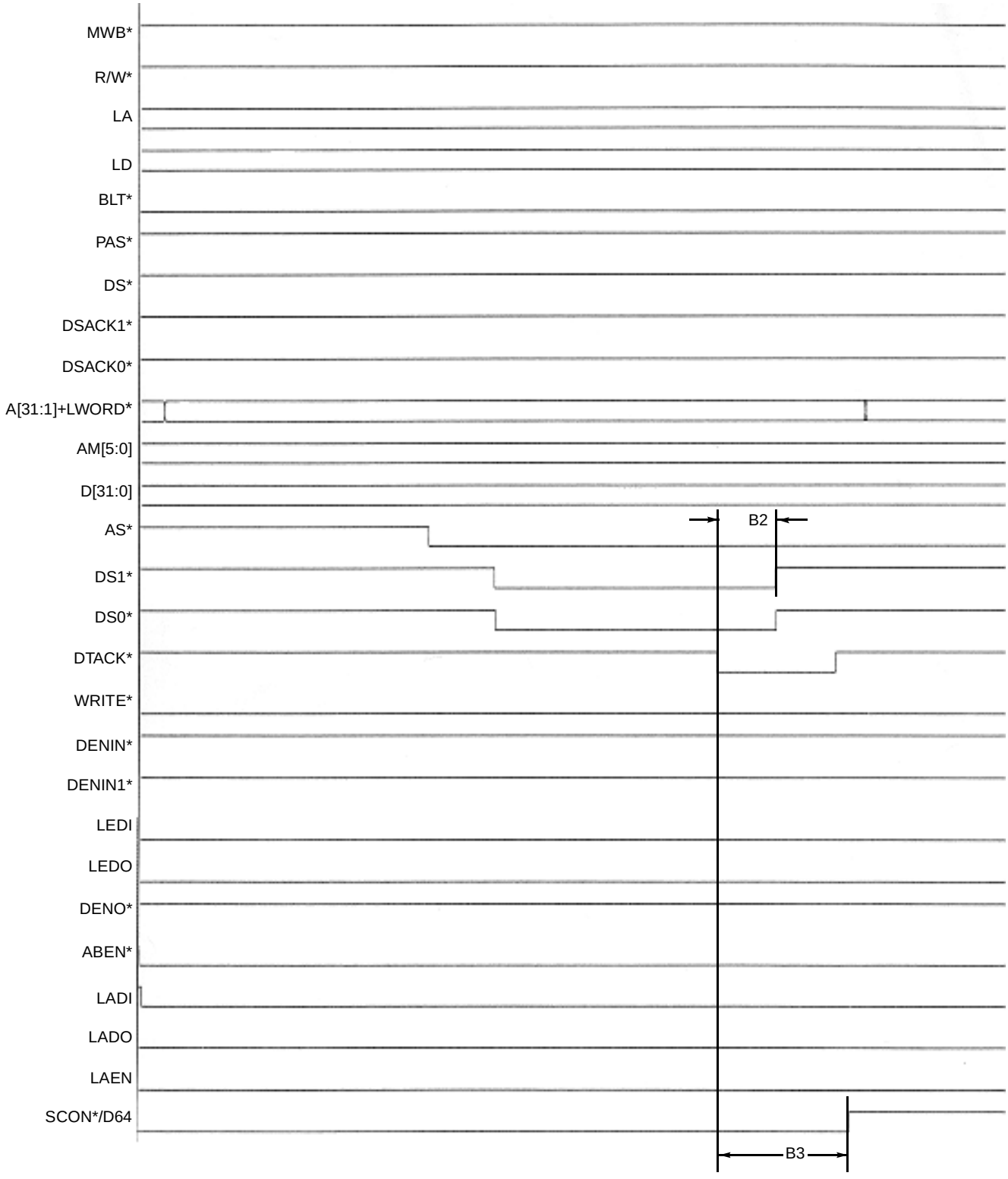


Figure 2-1. Master Address Broadcast Cycle

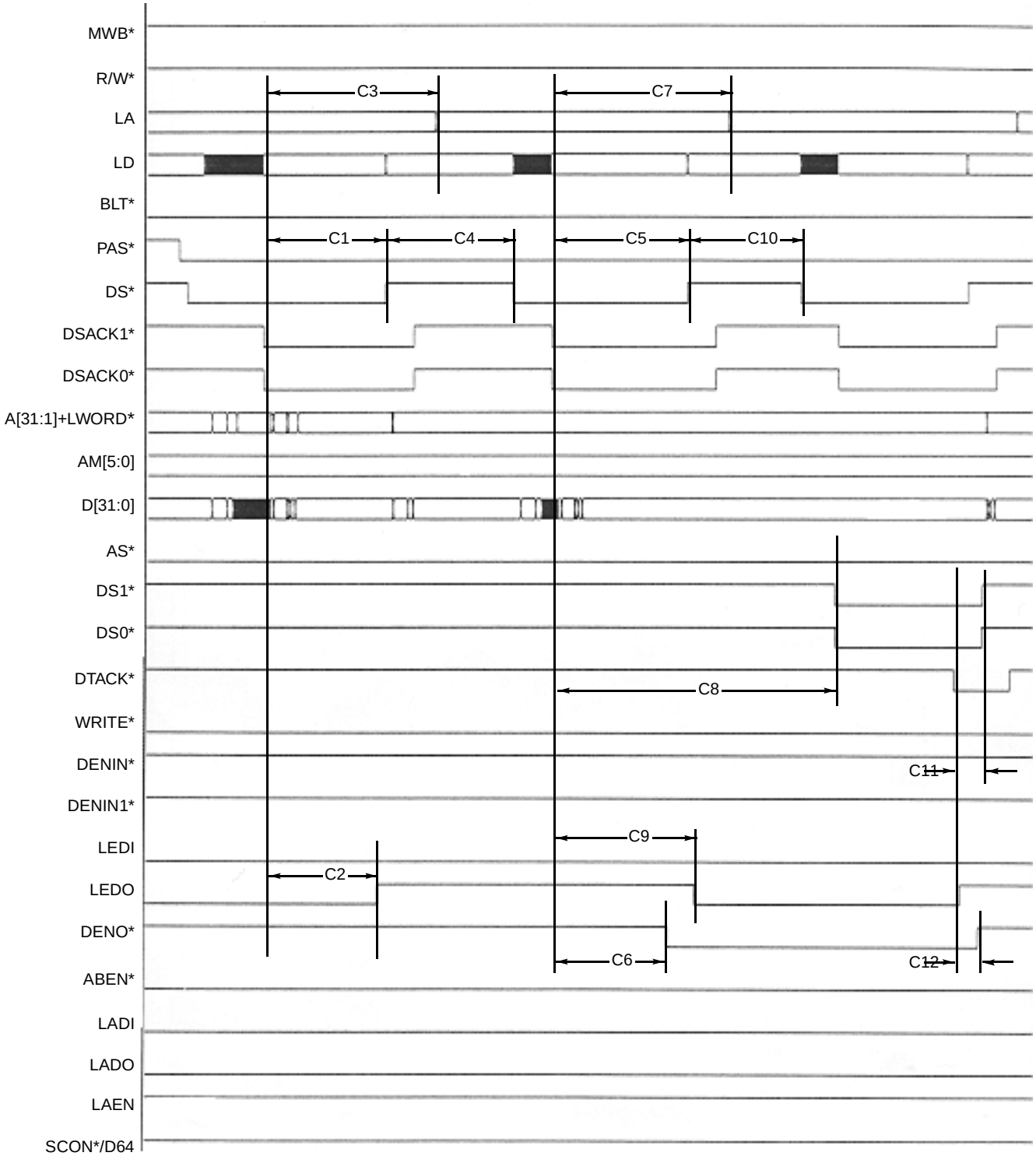


Figure 2-2. Master D64 Write Operation: Detail

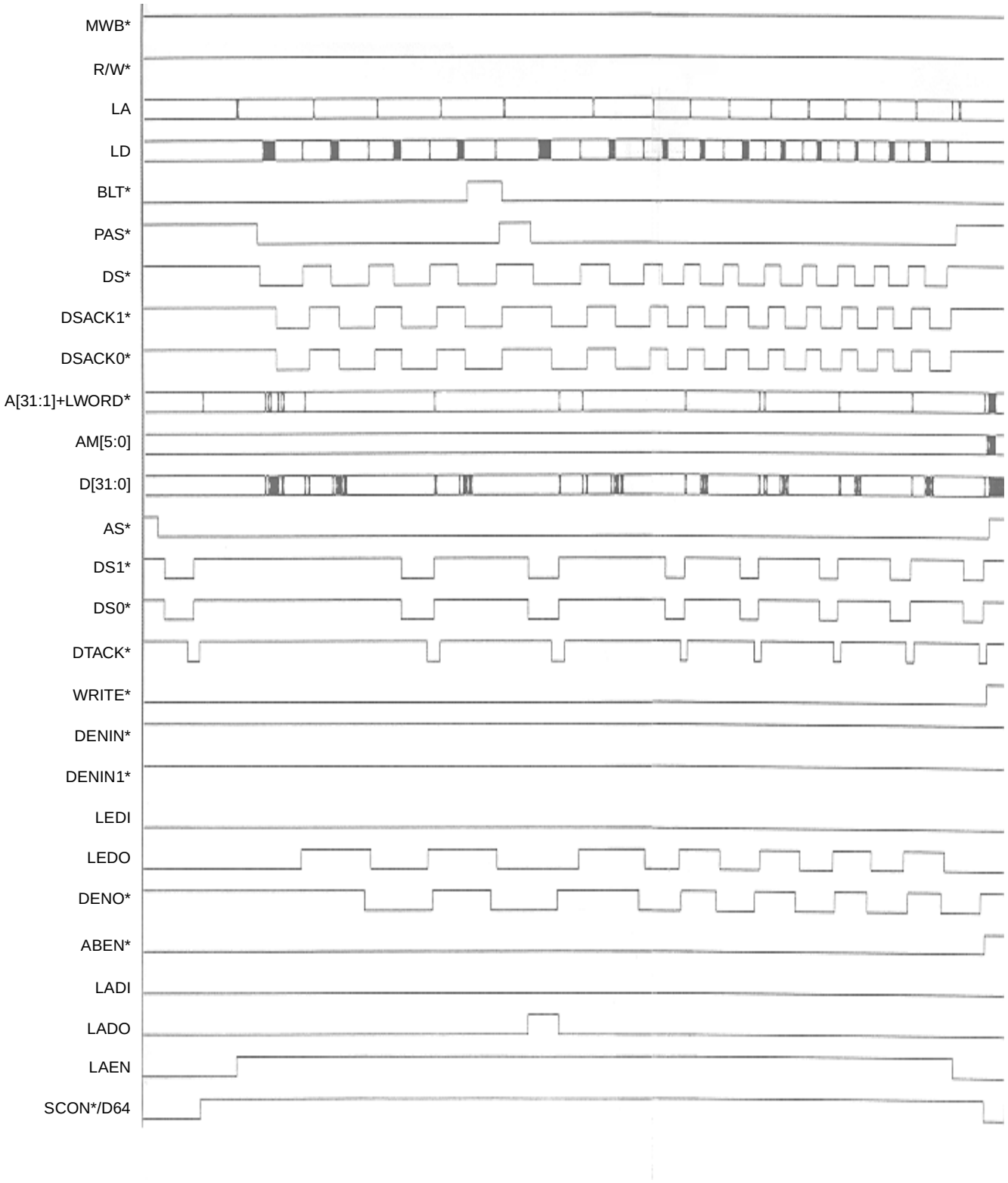


Figure 2-3. Master D64 Write Operation: Block Transfer

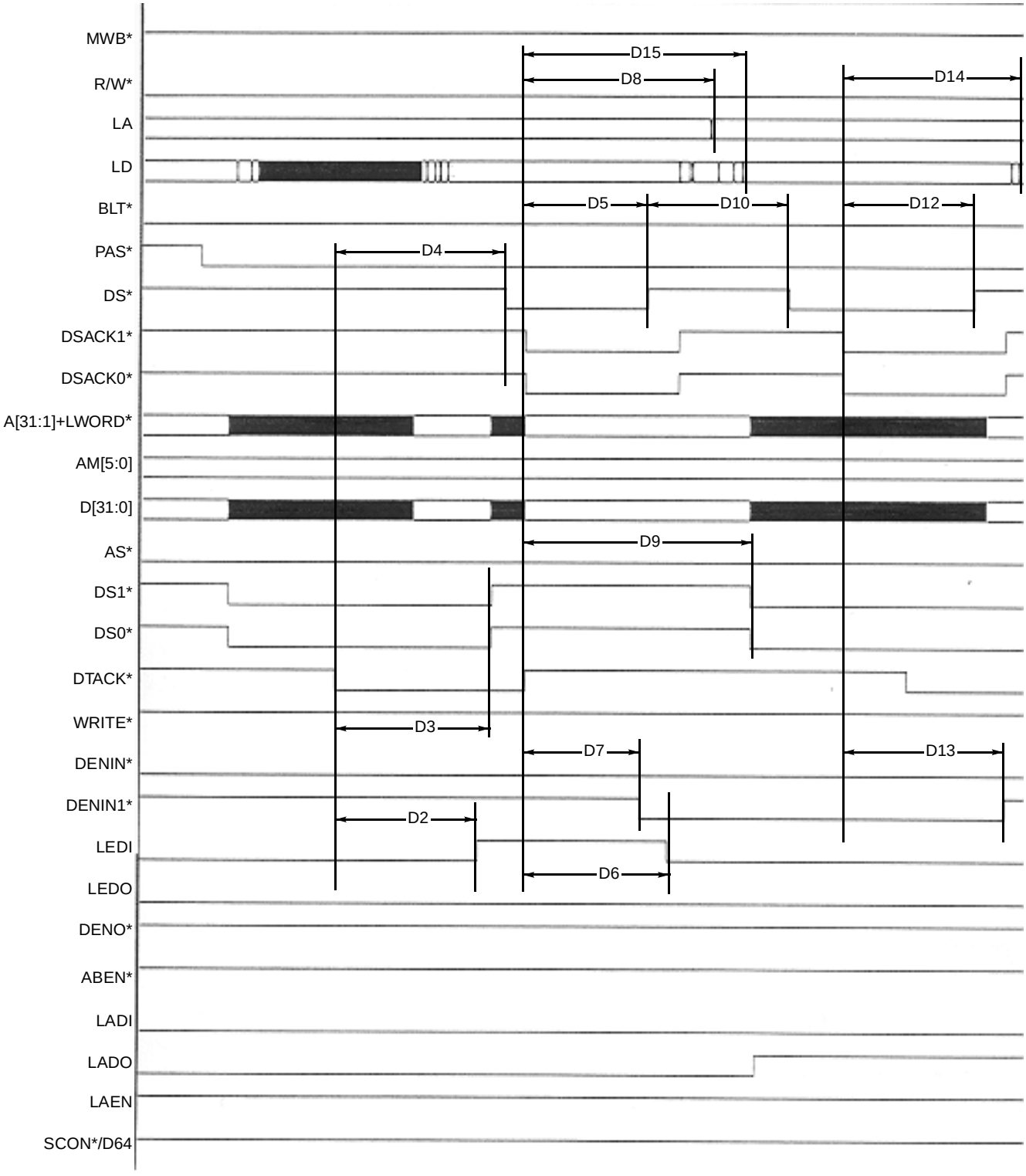


Figure 2-4. Master D64 Read Operation: Detail

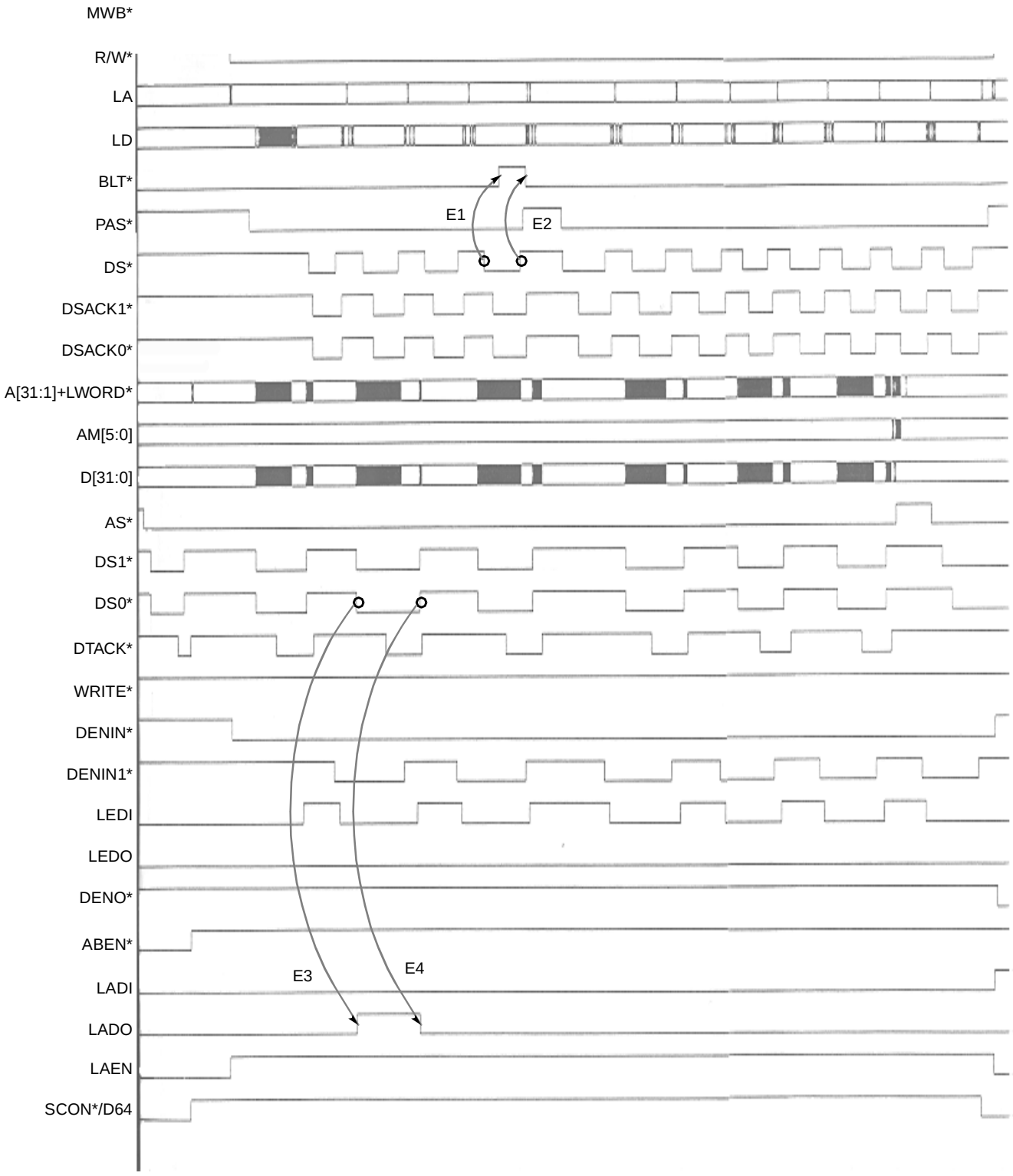


Figure 2-5. Master D64 Read Operation: Block Transfer

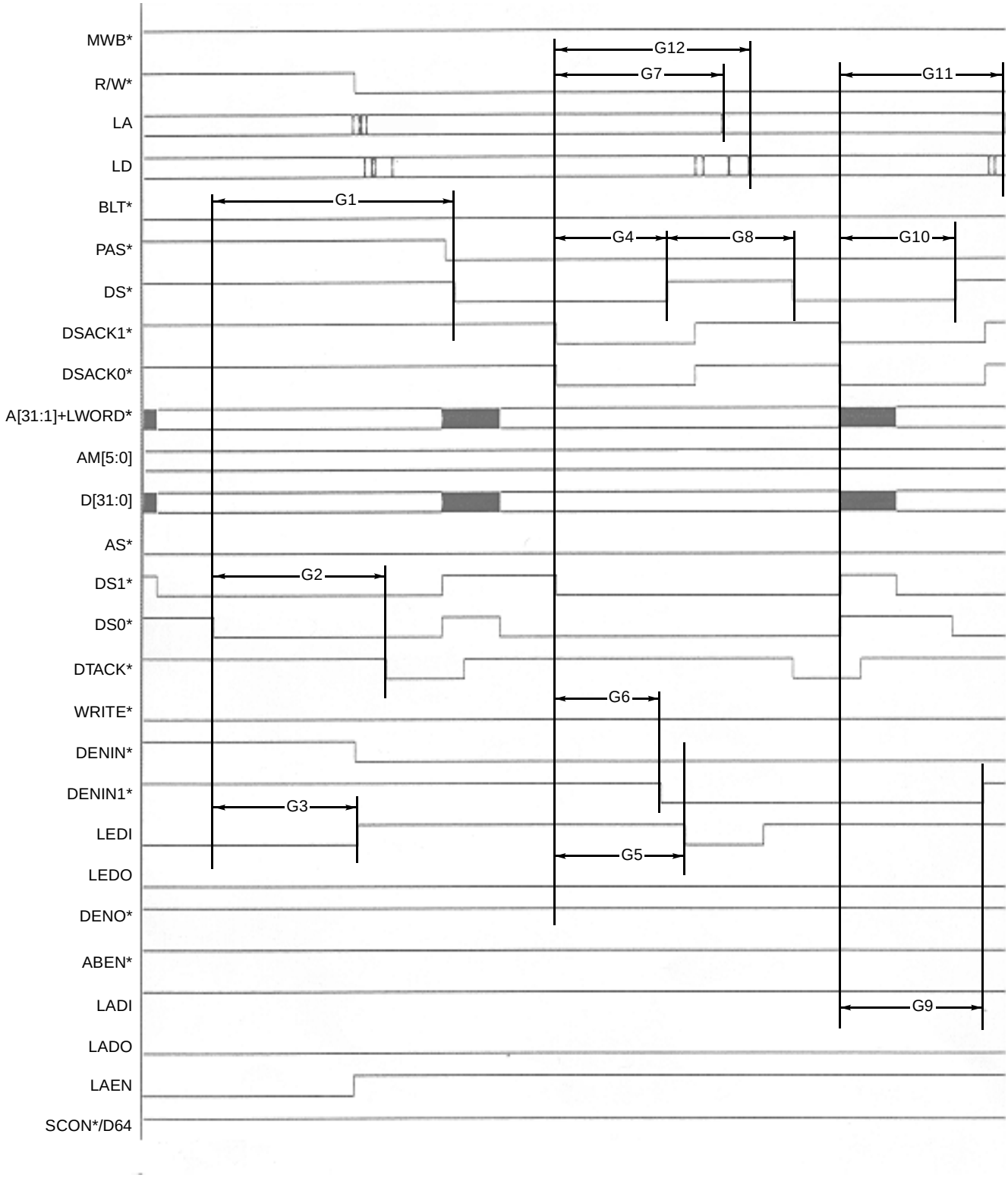


Figure 2-6. Slave D64 Write Operation: Detail

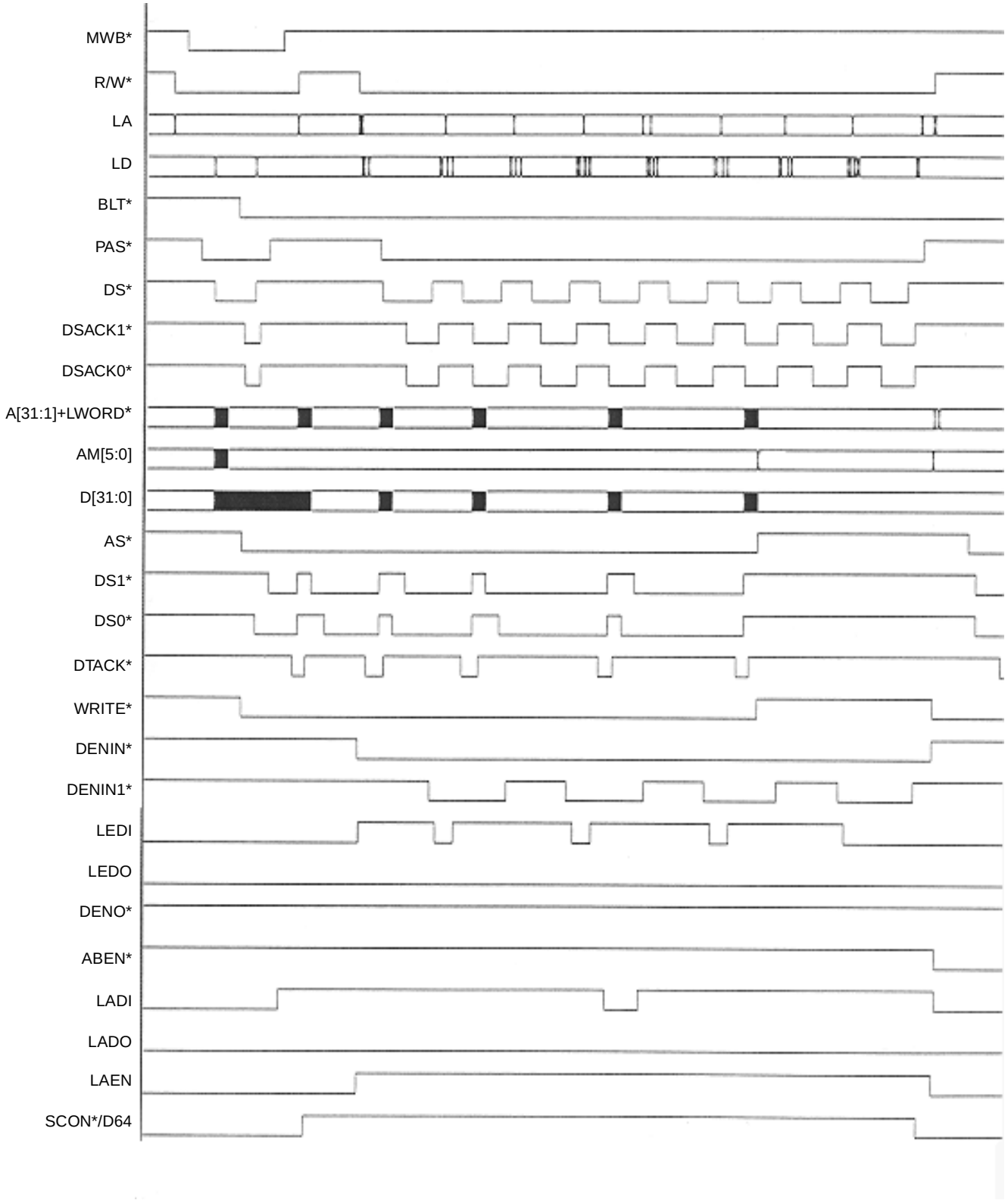


Figure 2-7. Slave D64 Write Operation: Block Transfer

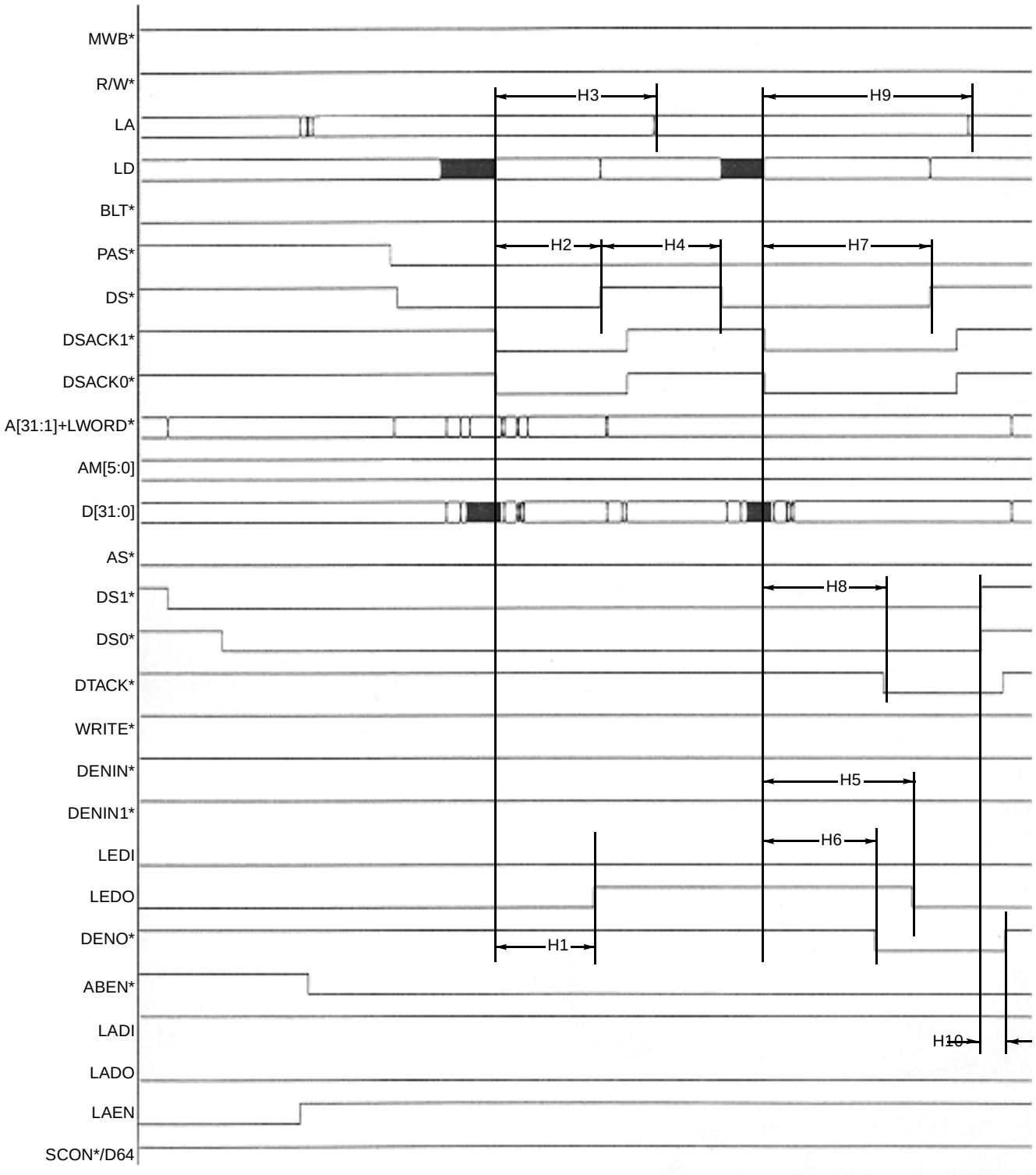


Figure 2-8. Slave D64 Read Operation: Detail

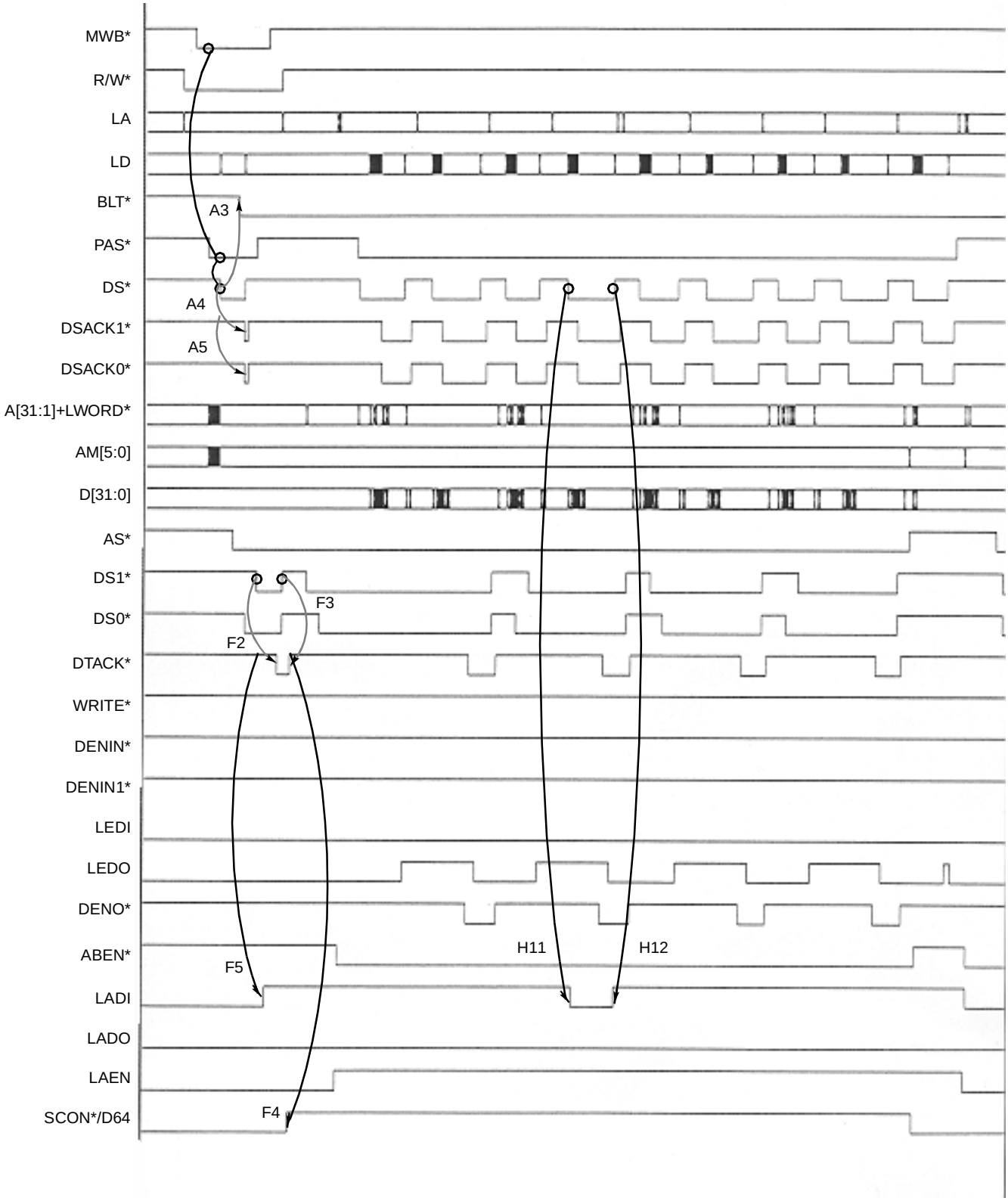


Figure 2-9. Slave D64 Read Operation: Block Transfer

2.5
DC Performance Specifications

Table 2-4. VMEbus Signals (AS*, DS1*, DS0*, BCLR*, SYSCLK)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 48 mA, 56 mA, 64 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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 2-5. VMEbus Signals (Low Drive. All VMEbus Daisy-Chain Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 2-6. VMEbus Signals (Medium Drive. All non-High, non-Low Drive Signals, All VAC068A VMEbus Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 2-7. Non-VMEbus Signals

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5		±10	µA

Table 2-8. Capacitance

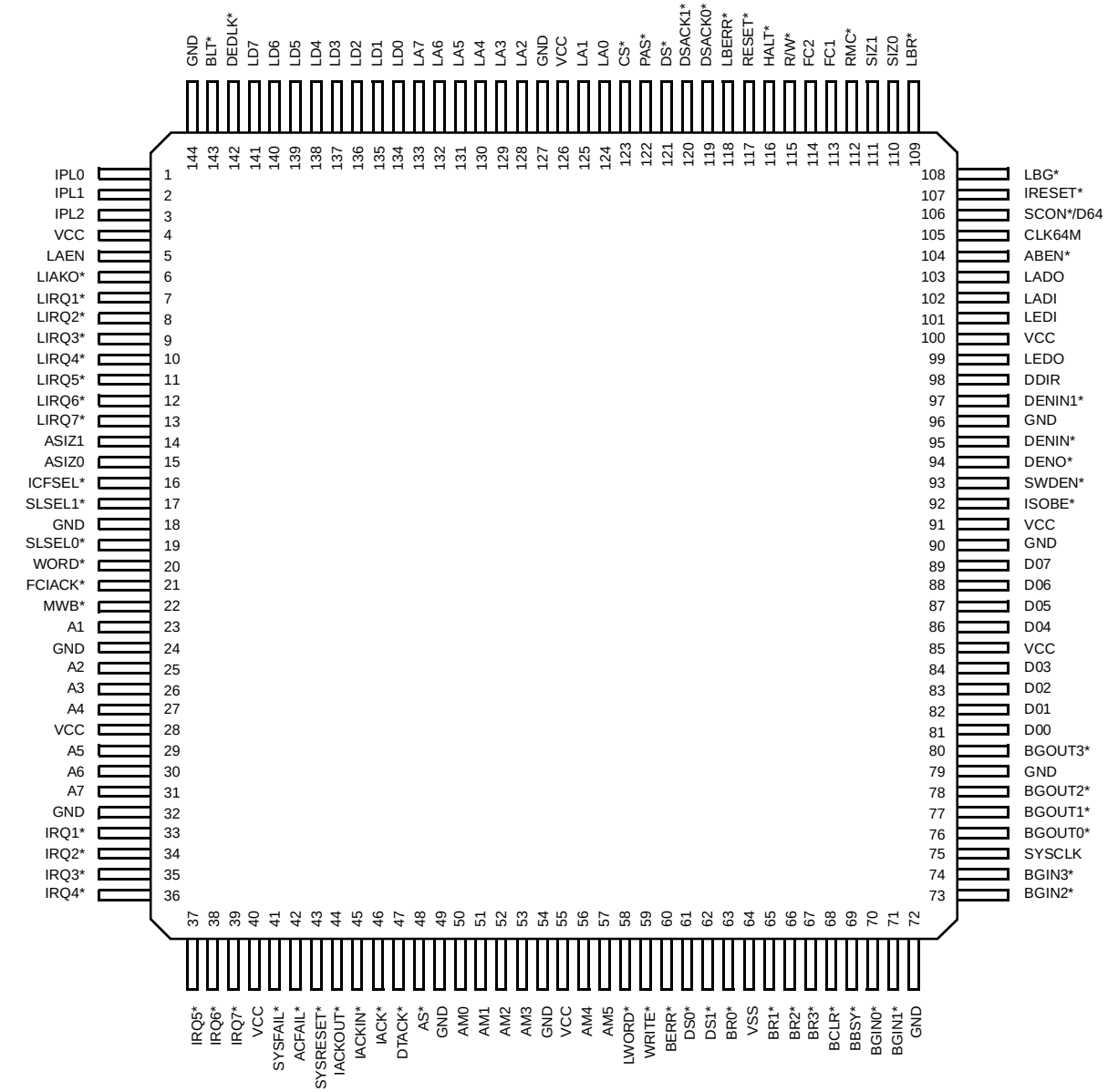
Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 64 MHz, V_{CC} = 5.0V	5	pF
C_{OUT}	Output Capacitance		7	pF

Table 2-9. Operating Current

Parameters	Description	Test Conditions	Max.	Units
I_{DD}	Maximum Operating Current	No external DC load	150	mA

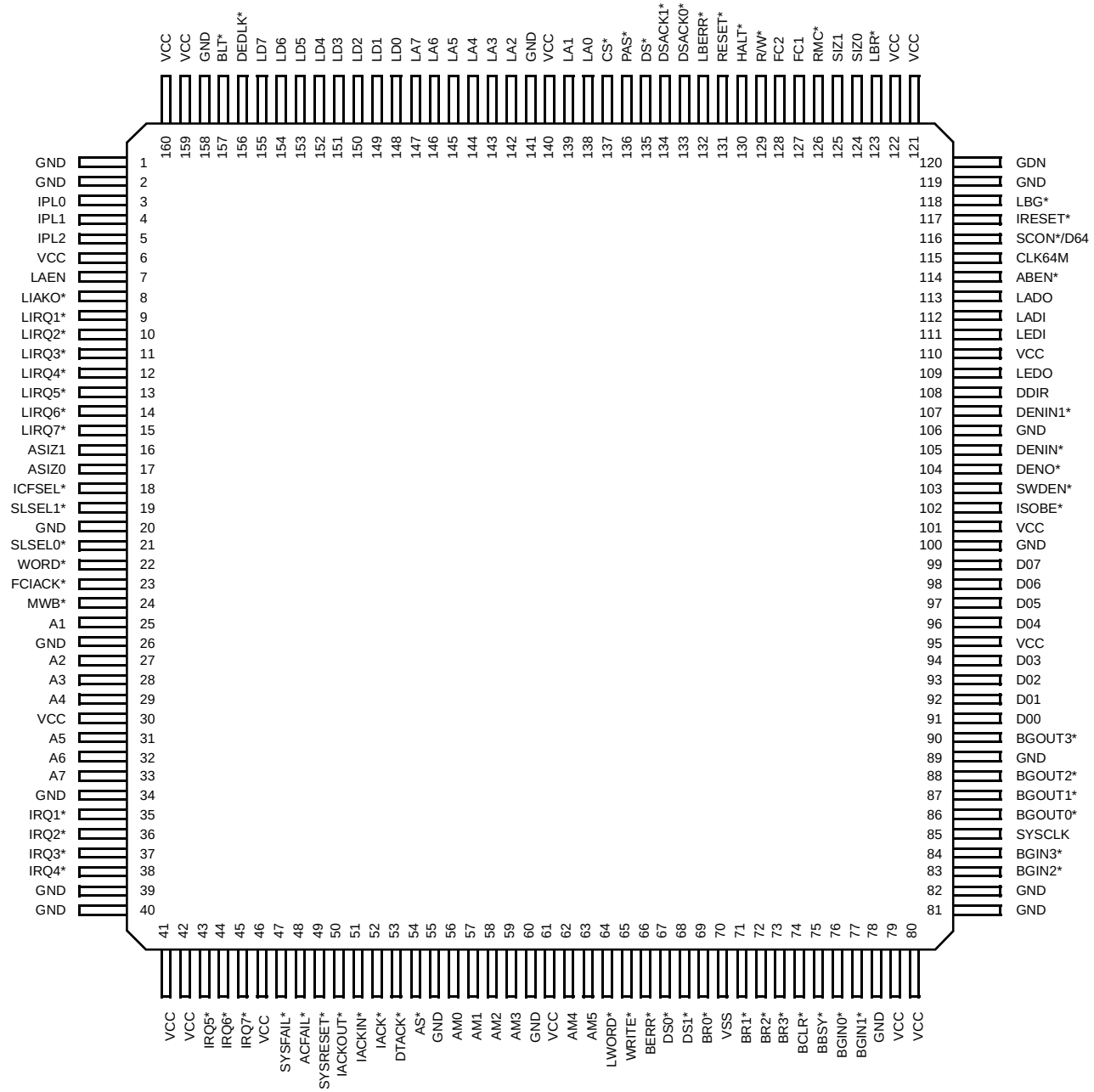
2.6
Pin Configurations

144-Pin Thin Quad Flatpack (TQFP)
Top View



160-Pin Quad Flatpack (QFP)

Top View

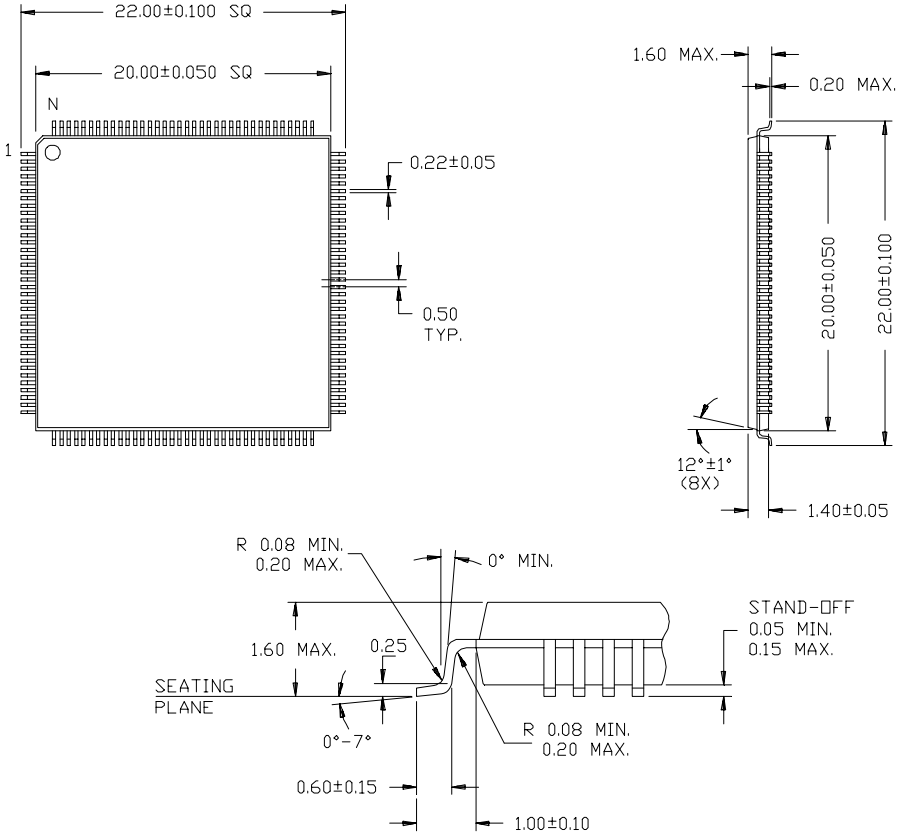


Pin Grid Array (PGA)
Bottom View

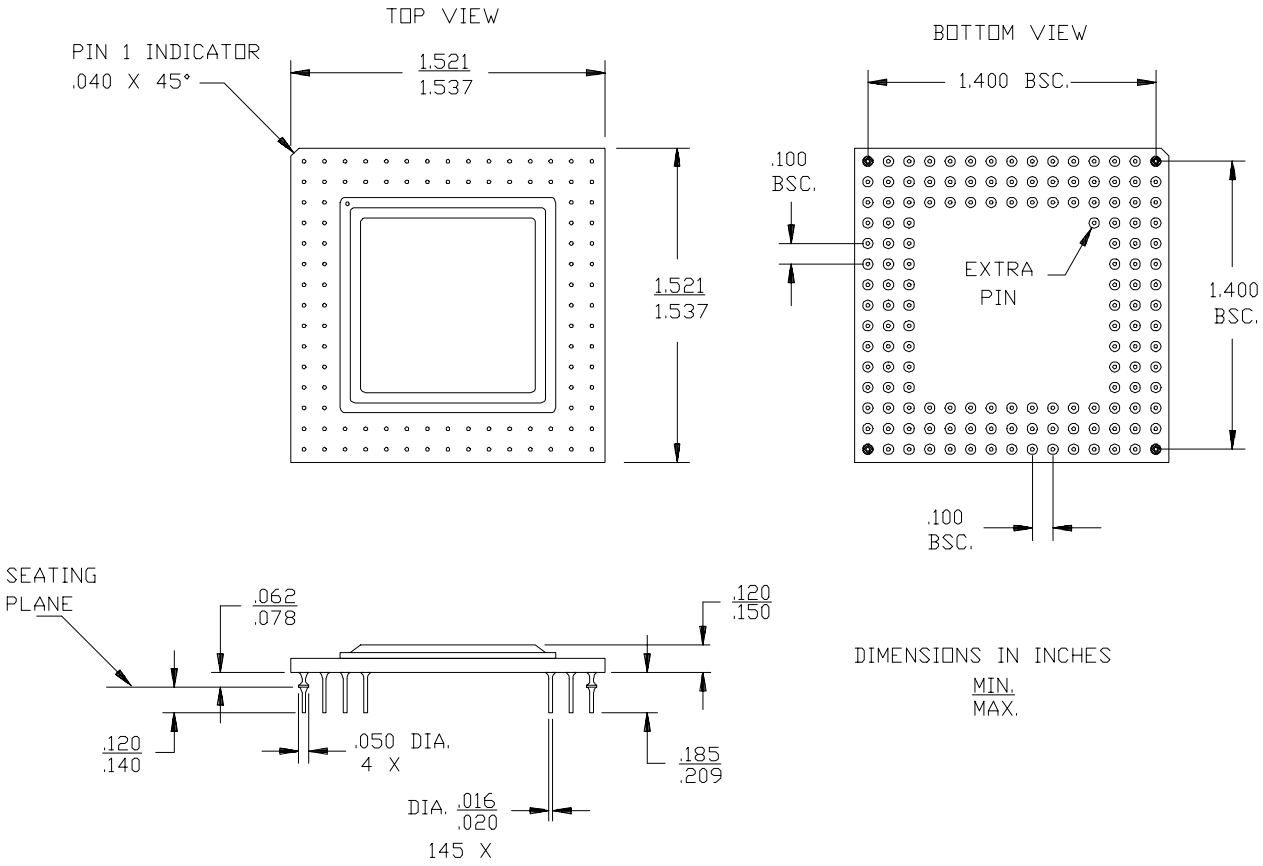
A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	
GND	IPL2	LIACKO*	LIRQ2*	LIRQ5*	ASIZ1	ASIZ0	SLSEL1*	WORD*	FCIACK*	A02	A04	VCC	GND	IRQ4*	1
LD6	BLT*	IPL1	VCC	LIRQ1*	LIRQ4*	LIRQ6*	ICFSEL*	MWB*	A01	A03	A05	A07	IRQ3*	IRQ7*	2
LD2	LD5	DEDLK*	IPL0	LAEN	LIRQ3*	LIRQ7*	GND	SLSEL0*	GND	A06	IRQ1*	IRQ2*	IRQ6*	ACFAIL*	3
LD1	LD3	LD7	LOCATOR									IRQ5*	VCC	IACKOUT*	4
LA7	LD0	LD4										SYSFAIL*	SYSRE-SET*	DTACK*	5
LA3	LA5	LA6										IACKIN*	IACK*	AM0	6
LA2	LA4	GND										GND	AS*	AM1	7
LA1	LA0	VCC										GND	AM2	AM3	8
CS*	DSACK1*	DS*										VCC	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*/D64	CLK64M	LADI	GND	VCC	GND	VCC	D00	BGOUT1*	BGIN2*	BGIN0*	BR3*	GND	13
SIZ1	IRESET*	LADO	LEDI	DDIR	DENIN*	DENO*	D06	D03	D01	GND	BGOUT0*	BGIN3*	BGIN1*	BCLR*	14
LBG*	ABEN*	VCC	LEDO	DENIN1*	SWDEN*	ISOBE*	D07	D05	D04	D02	BGOUT3*	BGOUT2*	SYSCLK	GND	15

2.7
Package Diagrams

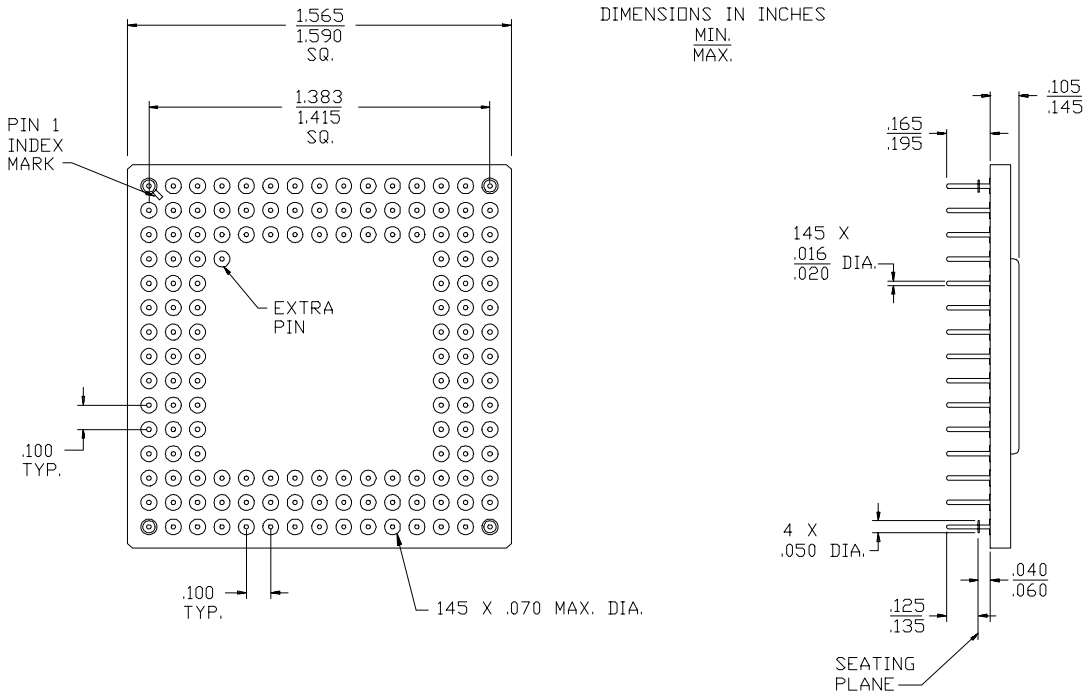
144-Pin Plastic Thin Quad Flat Pack (TQFP) A144



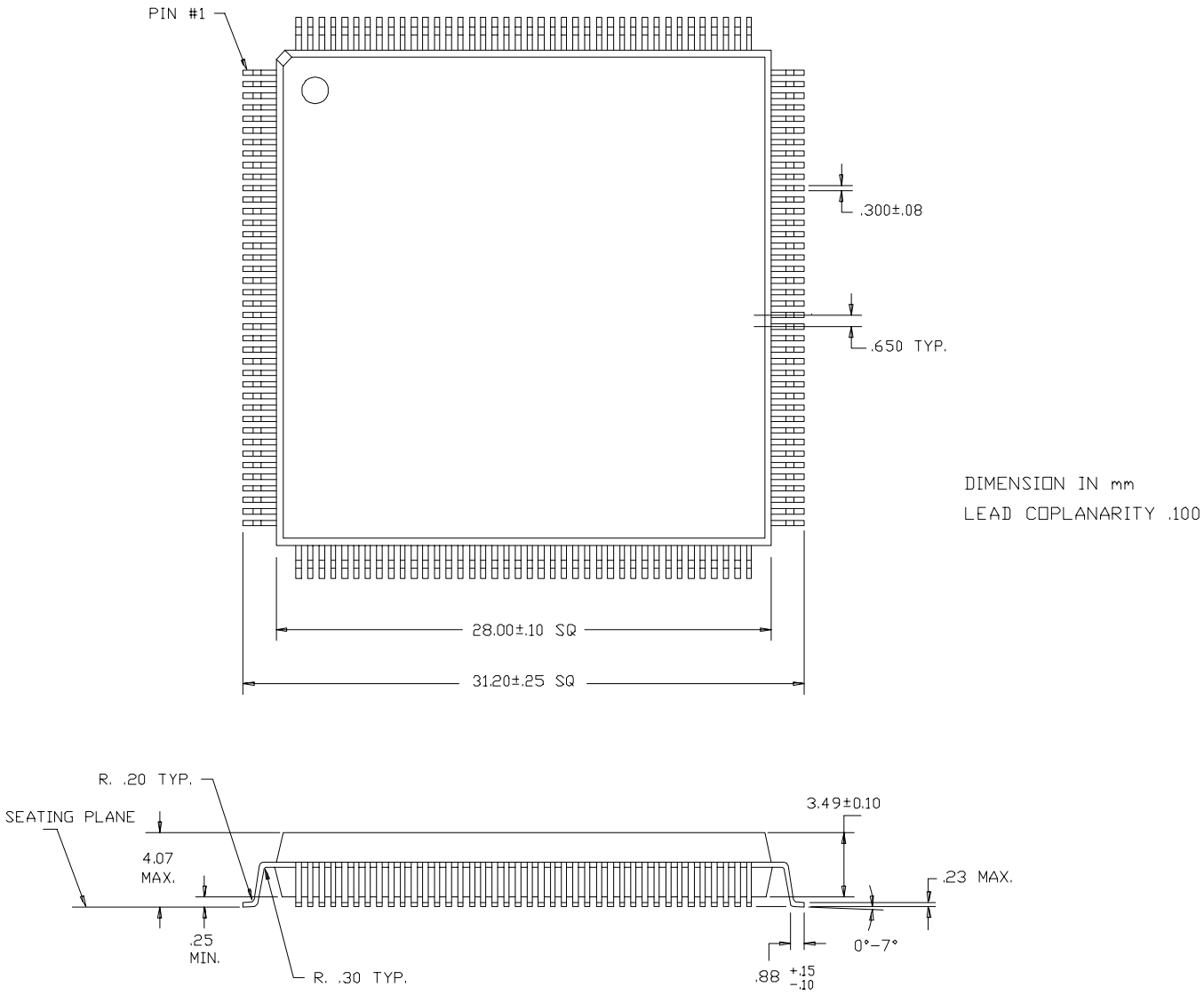
145-Pin Plastic Grid Array (Cavity Up) B144



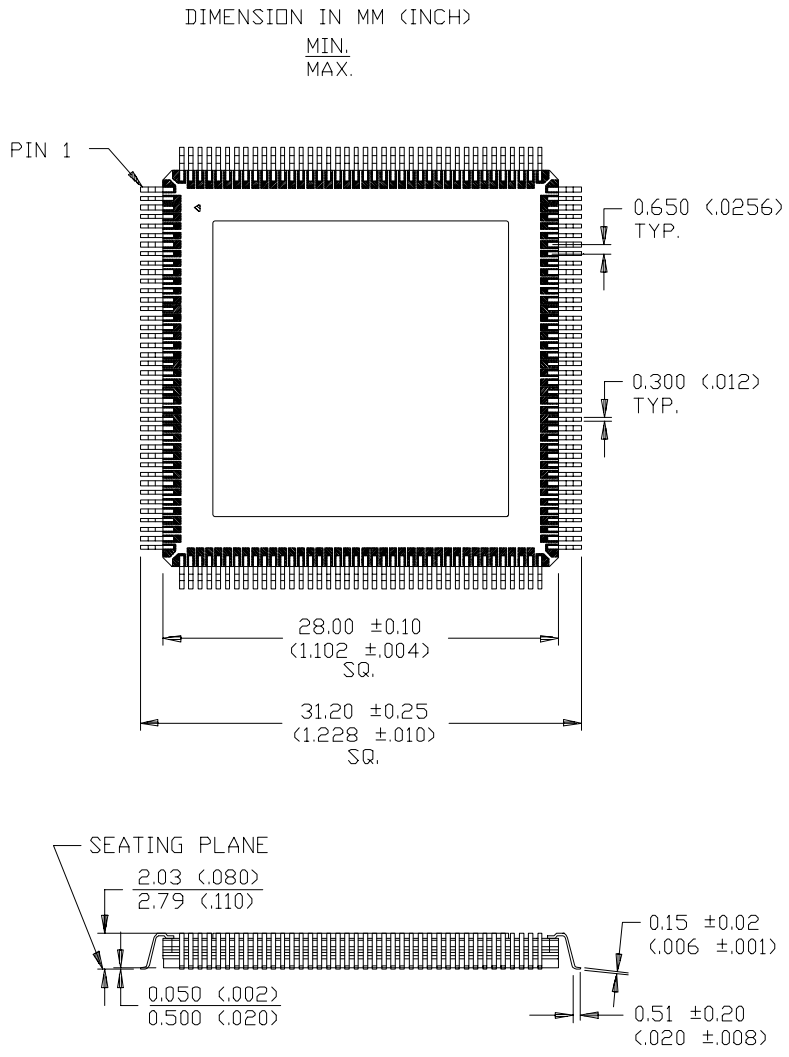
145-Pin Grid Array (Cavity Up) G145



160-Lead Plastic Quad Flatpack N160



160-Lead Ceramic Quad Flatpack (Cavity Up) U162



Section 3

The CY7C960/961 Slave VMEbus Interface Controllers

3.1 Introduction

3.1.1 Feature List

Optimal Performance:	80 Mbyte per second Block Transfer Rates
Next-Generation Product:	VME64 transactions, including A64/D64, A40/MD32 transfers, Auto Slot ID, CR/CSR space, LOCK cycles etc.
Backwards Compatible:	All standard VMEbus transactions implemented VMEbus Interrupter
Simple to Use:	No Local CPU required Programmable from VMEbus or serial PROM
Highly Integrated:	DRAM controller, including refresh timers, Local I/O controller,
Innovative Architecture:	Flexible VMEbus address scheme User-configured VMEbus Personality
Ultra-Small footprint:	TQFP Packaging, and other options.

3.1.2 Family Overview

The CY7C960 and the CY7C961 are members of Cypress's industry-standard range of VME-bus controllers. Although they are each low cost, they are highly flexible, and designed to meet the needs of high performance VMEbus board developers. The CY7C960 is intended for applications whose primary requirement is board space savings—it offers Slave-only features. The CY7C961 provides Master support in addition to the Slave features of the '960, and hence occupies a little more board space. Each device has been optimized for VMEbus performance, supporting the full 80-Mbyte/sec transfer rate of the VME64 specification.

The other members of the family are VIC068A (the industry's most popular 32-bit VMEbus interface controller), VIC64 (the 64-bit version of the VIC068A), VAC068A (VMEbus address controller), and CY7C964 (a useful companion for either VIC068A, VIC64, CY7C960, or CY7C961).

This Section of the book first describes the CY7C960, then extends that description to the CY7C961. It is necessary to understand the operation of the CY7C960 before reading the CY7C961 sections.

3.1.3 CY7C960 Architectural Overview

The CY7C960 Slave VMEbus Interface Controller provides the board designer with an integrated, full-featured VME64 interface. This 64 pin device can be programmed to handle every transaction defined in the VME64 specification. The CY7C960 contains all the circuitry needed to control large DRAM arrays and local I/O circuitry without the intervention of a local CPU. There are no registers to read or write, no complex command blocks to be constructed in memory. The CY7C960 simply fetches its own configuration parameters during the power-on reset period. After reset the CY7C960 responds appropriately to VMEbus activity and controls local circuitry transparently.

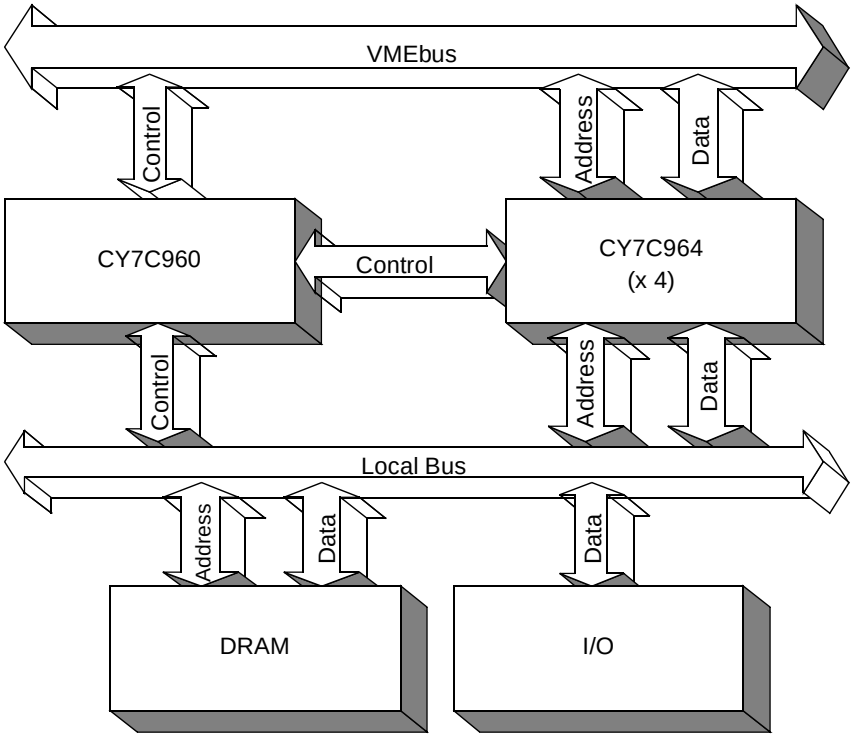


Figure 3-1. Typical System Block Diagram

The CY7C960 controls a bridge between the VMEbus and the local DRAM and I/O. Once programmed, the CY7C960 provides activities such as DRAM refresh and local I/O handshaking in a manner which requires no additional local circuitry. The VMEbus control signals are connected directly to CY7C960. The VMEbus address and data signals are connected to companion address/data transceivers which are controlled by CY7C960. The CY7C964 VMEbus Interface Logic Circuit is an ideal companion device: the CY7C964 provides a slice of data and address logic that has been optimized for VME64 transactions. In addition to providing the specified drive strength and timing for VME64 transactions, the CY7C964 contains all the circuitry needed to multiplex the address/data bus for multiplexed VMEbus transactions. It contains counters and latches needed during block transfer (BLT) operations. And it also contains address comparators which can be used in the board's Slave Address De-

coder. For a 6U or 9U application, four CY7C964 devices are controlled by a single CY7C960. For 3U applications, the CY7C960 controls two CY7C964 devices and an address latch.

If the application does not need VME64 transactions, then the user may choose to implement the companion logic using Cypress's inexpensive FCT family of interface devices. The CY7C960 also provides all of the timing and control signals needed for this application also. (See Chapter 3.7.)

The design of the CY7C960 makes it unnecessary to know the details of the VMEbus transaction timing and protocol. The complex VMEbus activities are translated by the CY7C960 to be simple local cycles involving a few familiar control signals. Similarly, it is not necessary to understand the operation of the companion device, CY7C964: all control sequences for the part are generated automatically by the CY7C960 in response to VMEbus or local activity. If more information is desired, consult Section 4, The CY7C964 Bus Interface Logic Circuit.

VMEbus Transactions supported by CY7C960 include all transactions defined by the VME64 specification. CY7C960 functions as a VMEbus Interrupter, and supports the new Auto Slot ID standard and CR/CSR space. CY7C960 also handles LOCK cycles, although full LOCK support is not possible within the constraints of the CY7C960 pinout. (For full LOCK support see the CY7C961 description.)

On the local side, no CPU is needed to program the CY7C960, nor to manage transactions. All programmable parameters are initialized through the use of either the VMEbus, or a serial PROM. As the CY7C960 incorporates a reliable power-on reset circuit, parameters are self-loaded by the device at power-up or after a system reset. If the VMEbus is used to provide parameters, a VMEbus Master provides the programming information using a protocol, described in a later section, which is compliant with the Auto Slot ID protocol from the VME64 specification.

Figure 3-2 shows the internal blocks that comprise the CY7C960. The architecture includes several functions that remove most of the VMEbus problems from the board designer's shoulders. All VMEbus control and response is automatic: the user loads the Region/AM table during configuration, and the CY7C960 then handles all appropriate VMEbus transactions. The CY7C964 controller works in lock step with the VMEbus Control Interface, providing the correct timing and control for the transaction in process. Local circuitry such as DRAM or I/O is simplified by the Refresh Controller, the DRAM Controller, and the Output Pattern Table. Block transfers are supported by the Local Address Controller together with the CY7C964 circuitry. Local timing is determined during configuration, and handshaking is available from the Data Byte Enable Controller. Local Interrupts are supported through the VME Interrupt Interface. The CY7C960 contains an internal Power-on Reset circuit, and responds also to a VMEbus SYSRESET*.

To keep the size of the package as small as possible, several signal pins carry multiplexed signals, and other pins have functions that are programmable.

The Pin Description section, and other parts of this document, provide full information on the definition and use of these multiplexed signals.

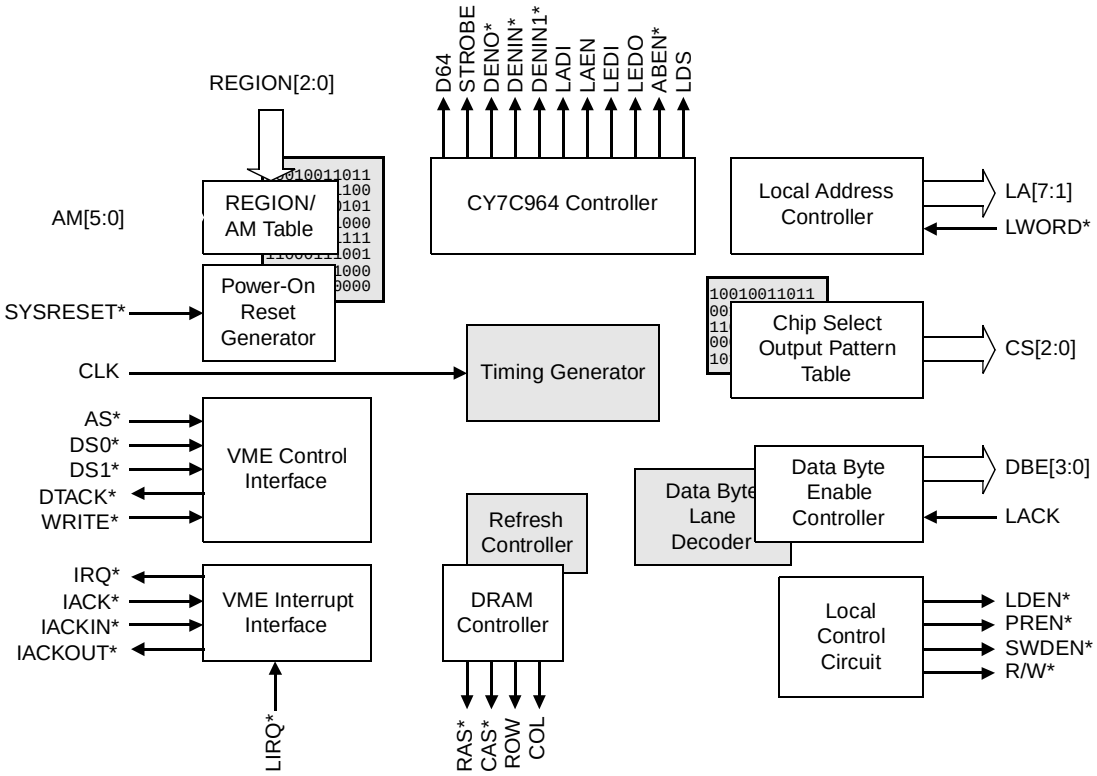


Figure 3-2. CY7C960 Block Diagram

3.1.4 Key Concepts

3.1.4.1 Local Bus Concepts

The CY7C960 has two modes of operation. The mode is selected during initialization. The user has a choice of DRAM/IO or I/O mode. If DRAM/IO mode is selected, then three Chip Select Outputs are provided along with the standard DRAM control signals. In I/O mode, six Chip Select Outputs are provided.

The CY7C960 introduces the concept of Region Mapping. This is a flexible method for selectively enabling VMEbus transactions to map to DRAM or I/O space. In DRAM/IO mode up to 8 Regions can be selectively mapped: in I/O mode up to 16 Regions are mappable. For example in DRAM/IO mode, Region 0 transfers could provide DRAM transactions while a transfer to Region 2 could disable DRAM, but provide Chip Select outputs.

The CY7C960 Chip Select Outputs are highly programmable: for each Region a different pattern can be driven from the available Chip Select Outputs. This allows external expansion if the three (six in I/O mode) outputs are insufficient. Additionally, the polarity of each chip select is programmable.

The VMEbus transfer mechanism allows for D8, D16, D32, D32UAT (Unaligned Transfer), MD32 (multiplexed), or D64 data transfers. The CY7C960 timing and control assume the use of the companion part CY7C964 that provides multiplexing control for the D64 (and for MD32) transfers. The CY7C964s are intended to connect to 32- or 16-bit local data buses: the CY7C960 provides appropriate signals to control byte-lane switching, pipelining and multiplexing as needed for the particular transaction in progress.

Four signals are provided for data byte enabling: DBE[3:0]. The CY7C960 controls these outputs as appropriate for the VMEbus transaction in progress. For example, in a three byte unaligned transfer on the VMEbus the three correct DBE pins go active. In a VMEbus transaction that has been mapped to access local DRAM; RAS*, CAS*, ROW and COL go active in the normal manner for a DRAM cycle, plus the appropriate DBE pins for the data size being transferred. In a transaction that has mapped to I/O space; the Chip Select patterns plus the appropriate DBE pins are driven.

VMEbus 8- and 16-bit transfers are carried on D[15:0]. To allow the use of 32-bit memory, these bytes need to be swapped onto LD[31:16] for those addresses with LA[1] = 0. CY7C960 drives SWDEN* at the appropriate time to facilitate this byte swapping.

The on-chip DRAM Refresh Controller automatically provides bursts of 4 CAS*-Before-RAS* refresh cycles at the appropriate times, in a manner that does not interfere with any ongoing VMEbus or local transaction. No external intervention is needed on behalf of posted data that was pre-empted by a refresh burst—the cycle completes automatically in all cases.

The CY7C960 can be configured to remain off the local bus upon assertion of a local bus holdoff signal (LACK). This facilitates connection of the CY7C960 to a local processor, or dual porting of local memory resources. This requires care in design, however, as long local bus holdoff time translates to VMEbus timeout. See section 3.5.4 for a full description of Local Bus Holdoff.

If the CY7C960 cannot provide a refresh burst to DRAM because of local bus holdoff, it will “remember” the number of missed bursts (up to 64) and “catch up” at the first opportunity. This also may translate to a VMEbus timeout, as a long DRAM refresh burst will delay the acknowledgment of a VMEbus transaction directed to DRAM space.

3.1.4.2 VMEbus Concepts

The CY7C964 devices contain address comparator circuitry that can form all or part of the user’s Slave Address Decoder. The CY7C960 provides timing and control to the CY7C964s that assumes the CY7C964s are indeed part of the Slave Address Decoder. The CY7C964s therefore play a key role during the initialization process, providing the base address of the CR/CSR (Configuration ROM/Control Status Registers) space that the CY7C960 occupies during configuration. The final step in configuration “moves” the CY7C964 base address to the intended position in the VMEbus address space. If the application does not use CY7C964s, the CY7C960 still provides control signals as if the CY7C964s were present. These signals can be used by external circuitry to accomplish the same tasks.

The CY7C960 recognizes Slave Address Regions which are selected by the use of the 4 REGION inputs (3 in DRAM/IO mode). It is the responsibility of the Slave Address Decoder circuitry external to the CY7C960 to drive these pins in response to VMEbus address changes. Thus 16 unique regions (8 in DRAM/IO mode) are recognized by the CY7C960, although normally at least one region would be used as the “Board Not Selected” region.

The CY7C960 is highly flexible in the manner by which it responds to VMEbus transfers. An Address Modifier (AM) Code table is configured during initialization which determines whether specific AM Codes will be recognized. If a particular code is disabled, then no VMEbus DTACK* is provided and the cycle is ignored. Furthermore, an independent AM Code table is available for each of the possible regions in the Slave Address Map. This provides great flexibility to the user: an I/O region could be configured to respond only to single cycle 8/16 bit transfers, while a DRAM region could be configured to respond only to A64 Block transfers.

3.1.5 Address Mapping

The CY7C960 supports many different schemes for address mapping. The function of the external decoder circuitry is to provide signals (REGION[3:0]) to the CY7C960 that define whether the board is addressed in VMEbus Address Space. One simple use of the CY7C960 is shown in *Figure 3-3*. In this simple case, the CY7C960 has been configured to respond only to A16 accesses in the VMEbus space 0h–ffffh; to A24 and CR/CSR accesses from 10000h–ff ffffh. From 100 0000h–ffff ffffh the CY7C960 responds to A32 accesses, and drives local circuitry to route the data to VSB (or Raceway) in the lower part, and to DRAM in the upper part. Above ffff ffffh, the CY7C960 is programmed to respond to A64 or A40 accesses, routing the data to DRAM.

Of course, this fictitious example is impractical due the size of the DRAM segments described. The CY7C960 can support simple contiguous schemes like this one, or arbitrarily complex decoders at the will of the designer.

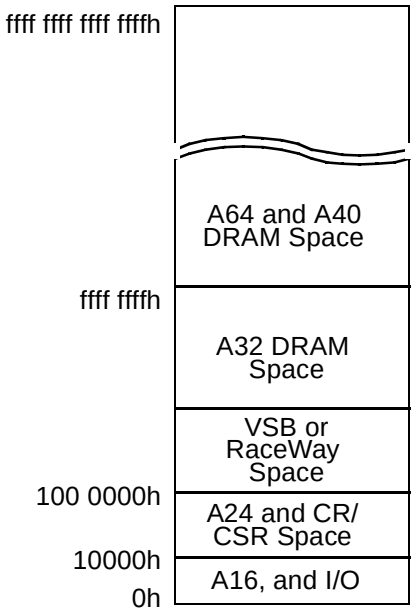


Figure 3-3. Possible VMEbus Slave Address Map

3.2
System Block Diagrams

Four examples of system diagrams are shown: a 6U example, a 3U example, a low cost 6U implementation, and a CY7C961 3U example.

Figure 3-4 shows an example of a 6U form factor board design. The details of the local circuitry are at the option of the board designer: this diagram illustrates the simplicity of the CY7C960 VMEbus interface. Note that in this example, DRAM/IO mode is selected so there are three region inputs; the VMEbus is used for configuration as no serial PROM is shown; if AM Code multiplexing is used, the LA[7:2] pins should be connected to the external address decoder and LADI is used to latch the code in the decoder. As CY7C964s are used, LAEN is required to be active High, and hence the internal pull-down resistor is sufficient to define this signal at power-on. No resistors are shown for clarity: see the Pin Description section (Chapter 3.3) for details of which signals are required to be pulled up or down.

The four CY7C964s are controlled by the CY7C960. For more details concerning the connection of the CY7C964 to the CY7C960, see the section on CY7C964 Interface (Chapter 3.6). The CY7C960 connects directly to the control signals of the VMEbus as shown. The CY7C964s connect directly to the VMEbus Data and Address bus. If it is desired to support 8- and 16-bit VMEbus data transfers to 32-bit memory or peripherals, then the Swap Buffer provides the data path switching needed to transfer the bytes on the appropriate byte lanes. All timing and control for this comes from the CY7C960. If only 32- and 64-bit transfers are required, the swap circuitry is not needed. The decoder block provides the function of Slave Address Map decoding, and is user-defined. The CY7C960 ensures that the VMEbus Address is available on the local address pins at the appropriate times, and the decoder compares the value with user-defined inputs. It is the responsibility of the decoder block to provide the REGION[2:0] inputs to the CY7C960. In very simple systems the address comparators within the CY7C964s could be used in place of the external decoder block. In *Figure 3-4* the CY7C964s are shown providing part of the decoder function (VCOMP* connections).

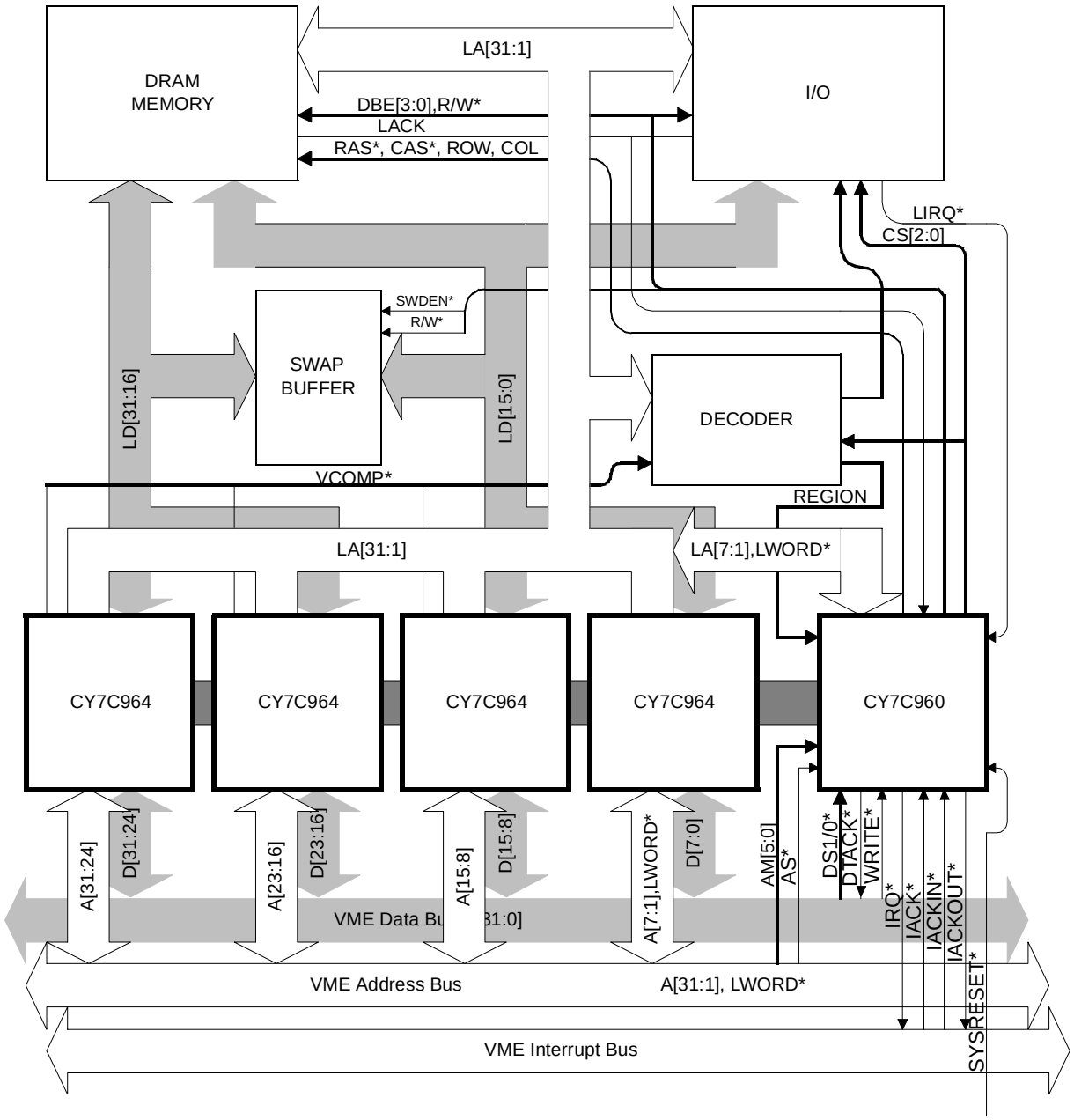


Figure 3-4. CY7C960 6U Block Diagram

The DRAM is controlled by the CY7C960, including refresh, and hence the standard connections, RAS*, CAS*, and R/W* are shown. The signals ROW, COL, and DBE, are additional signals provided by the CY7C960 to facilitate control of the array. The local handshake, LACK, is shown connected to the DRAM array. In some cases it may be desirable to delay the progress of a DRAM access—LACK can accomplish this. LACK is also used to handshake I/O cycles if desired. The DBE signals connected to the I/O block provide byte addressing for peripherals greater than 8 bits wide.

The I/O block is shown providing an interrupt, LIRQ*. This signal causes the CY7C960 to provide a VMEbus Interrupt. It is the responsibility of the Interrupt Handler and local circuitry to remove the LIRQ* signal at the appropriate time. The CY7C960 merely copies the value of the LIRQ* through to its IRQ* pin, and handles the IACK cycles appropriately.

The CY7C960 drives the Chip Select outputs in response to a VMEbus cycle. The pattern that is driven can be any binary pattern. In this example, the three available chip selects are programmed to provide a different binary pattern for each of the available eight Regions. They are connected to the decoder so as to provide a three-to-eight decoder. Thus up to eight peripherals could be selected in this manner.

Figure 3-5 shows a 3U system diagram. In this case there are only two CY7C964s needed, along with one 8-bit address buffer. If the AM Code is connected to the external decoder on pins LA[7:2], then LADI is used to latch the AM Code value in the decoder at the appropriate time. LAEN is required to be active-High as in the 6U example.

In order to add support for A40/MD32 operations the external decoder would use DENIN* and DENIN1* to enable the local data LD[15:0] into the decoder comparator, making up the upper 16 bits of A[39:0]. The CY7C964s provide the multiplexing for the upper 16 bits of the 32-bit data word. The controls for these transactions are provided automatically by the CY7C960. The DBE signals require combinational logic if the local memory or peripherals are 16 bits wide, as the CY7C960 is designed to control 32-bit local devices. Thus a VMEbus D16 block transfer would activate first DBE[1:0], then DBE[3:2], dependent upon the starting address. The DBEs should be logically combined to produce the desired result. Refer to section 3.6.3, Swap Buffer Control, for more information.

Figure 3-6 shows a low-cost 6U implementation. The VMEbus address and data are buffered by Cypress FCT parts as shown, controlled by the CY7C960 without additional logic being required. In this case the LAEN is required to be active-Low, accomplished at power on using the pull-up resistor externally as shown. As CY7C964s are not used, the functions that this schematic provides are limited. For example, no multiplexed data transactions, or A64/A40 transactions are possible. The external address decoder cannot take advantage of the CY7C964's on-chip comparators in providing the slave address decode signals REGION[3:0].

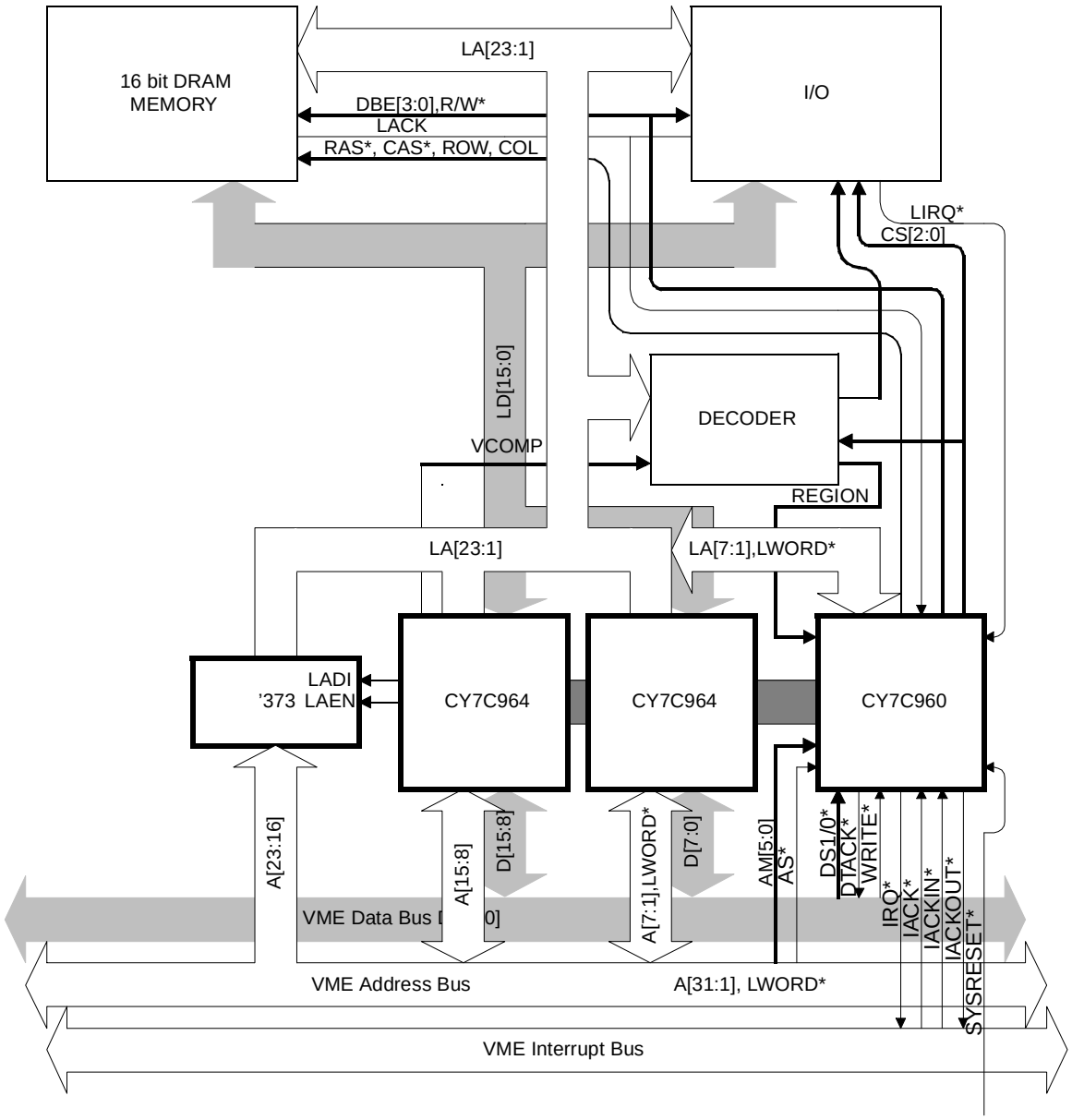


Figure 3-5. CY7C960 3U Block Diagram

Figure 3-7 shows the system diagram for a CY7C961 3U implementation. Note that there are additional connections to the VME arbitration bus. The block diagram shown supports A40 master operations through the use of an external upper address latch. In this implementation the local DRAM is 32 bits wide, though the VMEbus data connections are only 16 bits wide. A swap buffer comprised of bidirectional transceivers such as the CY74FCT162445 is used to drive the correct byte lanes of the DRAM, controlled by the CY7C961 automatically. Refer to section 3.6.3, Swap Buffer Control, for more information.

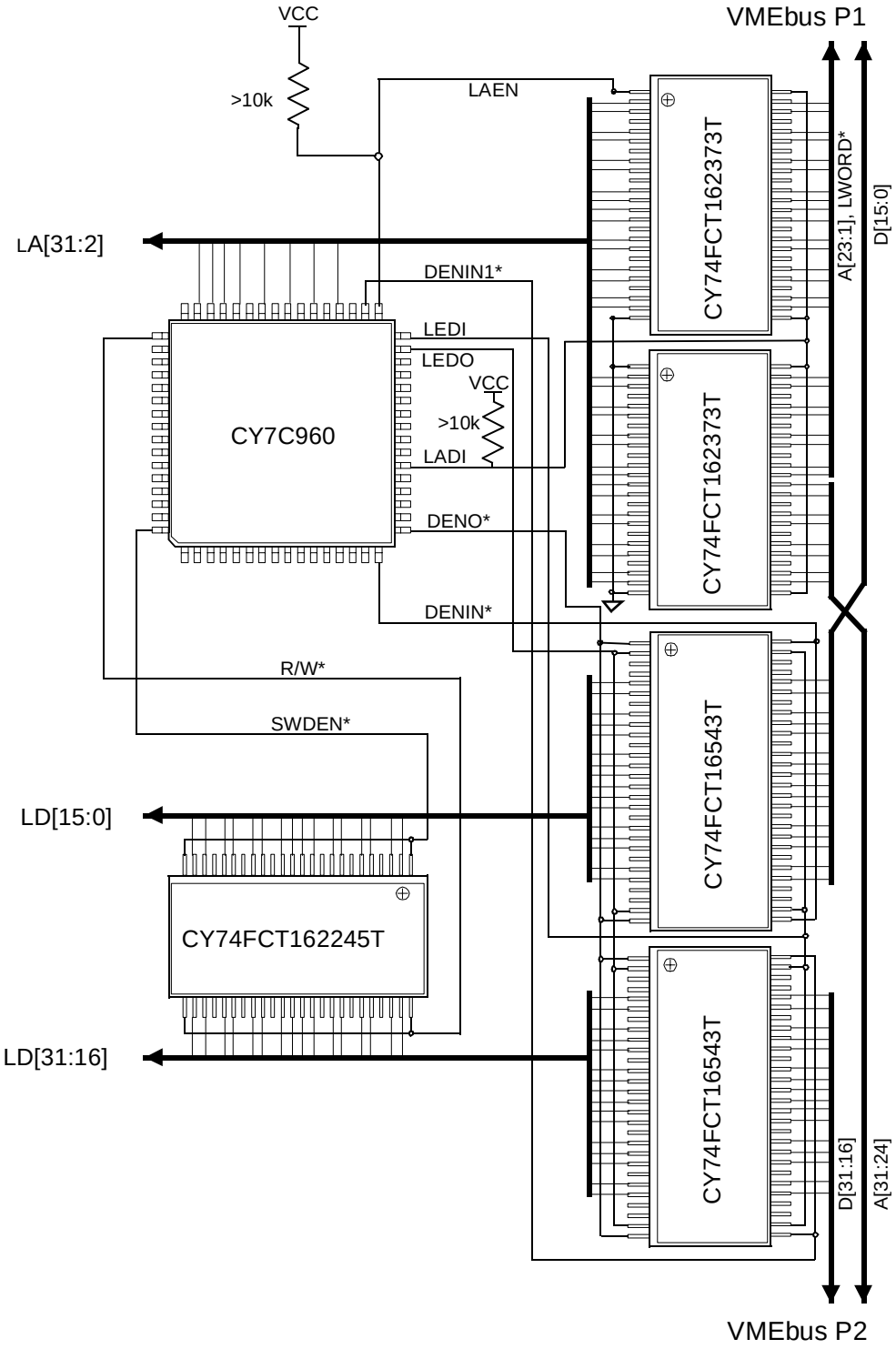


Figure 3-6. CY7C960 Low-Cost Block Diagram

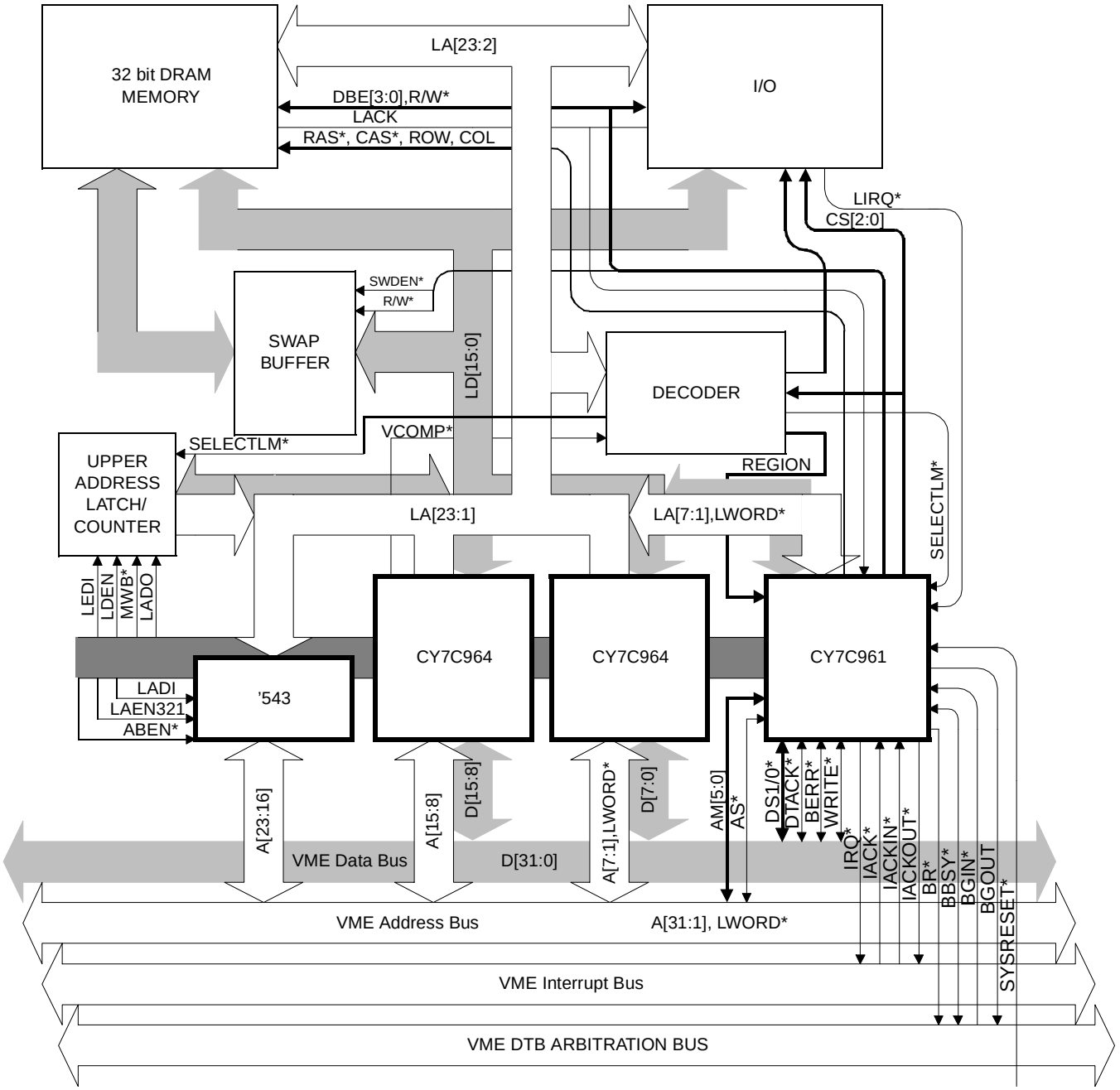


Figure 3-7. CY7C961 3U Block Diagram

These four examples illustrate the flexibility of the CY7C960 and CY7C961 in supporting many different VMEbus and local bus configurations.

3.3
Pin Description

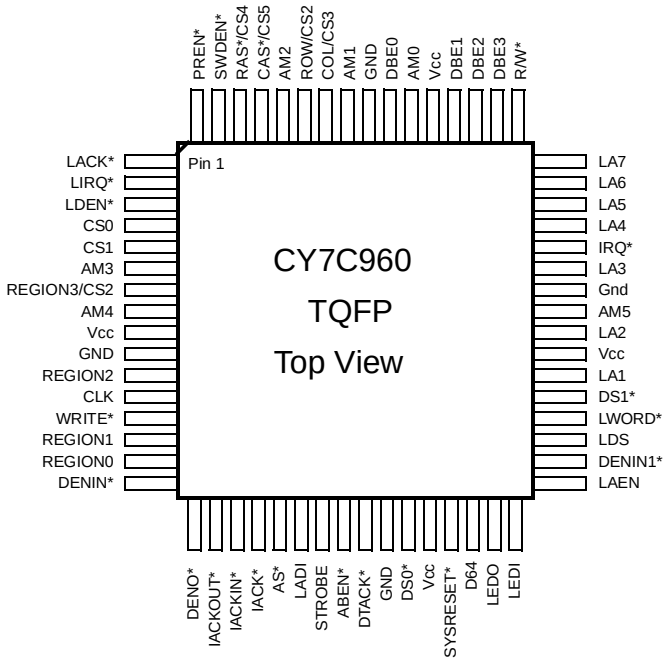


Figure 3-8. CY7C960 Pin Assignment (TQFP)

3.3.1 VMEbus Signals

AM[5:0] - VMEbus Address Modifier

Input: Yes
Output: No

Signals AM[5:0] are the VMEbus Address Modifier inputs. These input signals are used to decode the VMEbus transaction type. The CY7C960 provides support for all predefined and user-defined VMEbus Address Modifiers.

AS* - VMEbus Address Strobe

Input: Yes
Output: No
Active: Low

Address Strobe is the VMEbus signal that informs VMEbus slaves that a valid address is on the VMEbus. This signal is used by the CY7C960 to qualify the VMEbus Address Modifiers AM[5:0] and REGION[3:0] inputs to determine if a slave cycle should be performed.

DS0*, DS1* - VMEbus Data Strobe

Input: Yes
Output: No
Active: Low

DS0* and DS1* are the VMEbus Data Strobe inputs. These signals inform the CY7C960 that either an address broadcast or data phase of the VMEbus cycle has begun. These signals in conjunction with the VMEbus LWORD* and LA[1] signals encode the data transfer width or number of bytes, 1 through 4. This information is necessary to enable the appropriate CY7C964 data bytes.

IRQ* - VMEbus Interrupt Request

Input: No
Output: Yes
Drive: 48 mA
Active: Low

The IRQ* signal is driven by the CY7C960 to indicate that the local interrupt request signal is active. The IRQ* pin should be connected to one of the 7 VMEbus interrupt request signals IRQ[7..1]* in order to implement a standard VMEbus interrupt requester. This signal may also be used, following a power-on reset, as part of the configuration protocol using the VMEbus to load the device configuration registers. In this case it shall be connected to VMEbus IRQ2*.

IACK* - VMEbus Interrupt

Input: Yes
Output: No
Active: Low

The IACK* signal further qualifies the VMEbus address transaction. If this signal is asserted the CY7C960 decodes the VMEbus address transaction as an Interrupt Acknowledge cycle. During the interrupt acknowledge cycle the CY7C960 will match three bits of the VMEbus address (A[3:1]) against the user-programmed internal level. If the levels match, IACKIN* is received active, and a VMEbus interrupt is pending from the device, then the VMEbus interrupt acknowledge is completed by the enabling of a vector onto the VME data bus. This vector is user defined, and enabled using the LDEN* signal. CY7C960 assumes that it is an 8-bit interrupter, and hence responds to all interrupt acknowledge cycles without regard to the size of the STATUS/ID requested. If a VMEbus interrupt is not pending or IACKIN* is not asserted to the device this VMEbus address cycle will be disregarded.

IACKIN* - VMEbus Interrupt Acknowledge In

Input: Yes
Output: No
Active: Low

The IACKIN* signal informs the CY7C960 that no other up-stream (daisy chain) VMEbus device has responded to the VMEbus Interrupt Acknowledge cycle. When IACKIN* is asserted to the CY7C960 during a VMEbus Interrupt Acknowledge, cycle the device determines if a

VMEbus interrupt is pending of the same level being acknowledged. If so the CY7C960 generates a local interrupt acknowledge cycle to respond to the pending interrupt. If a local interrupt is not pending the device asserts IACKOUT*, passing the chain to the next interrupter on the VMEbus backplane.

IACKOUT* - VMEbus Interrupt Acknowledge Out

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The CY7C960 asserts the IACKOUT* signal during VMEbus interrupt acknowledge cycles if there is no VMEbus interrupt pending on the device, or if the VMEbus interrupt level being acknowledged does not match that being requested by the CY7C960.

SYSRESET* - VMEbus System Reset Input

Input: Yes
Output: No
Active: Low

The SYSRESET* input to the CY7C960 halts all local and VMEbus operations and causes the device to reinitialize all internal register bits. This reinitialization is either performed by reading the external serial PROM or by an Initialization Master on the VMEbus. When SYSRESET* goes Low, all outputs from the CY7C960 become three-state until the first rising edge on the CLK input.

WRITE* - VMEbus Write Input

Input: Yes
Output: No
Active: Low

The VMEbus WRITE* input encodes the type of VMEbus data cycle in progress. This signal is asserted when a VMEbus WRITE operation is in progress. During such a transaction, if the VMEbus address decodes properly, the CY7C960 responds by asserting the local R/W signal and performing the appropriate local cycle.

DTACK* - VMEbus Data Acknowledge Output

Input: No
Output: Yes
Drive: 64 mA
Active: Low

The DTACK* signal is asserted by the CY7C960 when a valid VMEbus transaction is in progress and the transaction has remained valid for the proper length of time. The assertion of this signal informs the VMEbus Master that the slave has either accepted the data during write operations or has sourced the data during read operations. This signal is a rescinding output.

3.3.2 Local Signals

LWORD* - Long Word

Input: Yes
Output: No
Active: Low

This is the local version of the VMEbus signal LWORD* which, when active, indicates either a D32 or D64 transfer over the VMEbus. The VMEbus LWORD* signal is connected directly to the A[0] bidirectional pin of the least significant CY7C964. The LA[0] bidirectional pin of the least significant CY7C964 is in turn connected directly to the LWORD input of the CY7C960. This signal in conjunction with the DS0*, DS1*, and LA[1] signals encode the data transfer width or number of bytes, 1 through 4. This information is necessary to enable the appropriate CY7C964 data bytes.

LA[1] / PCLK - Local Address Signal / Init PROM Clock

Input: Yes
Output: Yes
Drive: 8 mA

During CY7C960 initialization this pin sources or receives a serial-PROM-compatible clock. The state of this pin is sampled immediately after the power-on reset period expires. If it is sampled High, then the CY7C960 will source a PROM clock signal (PCLK) during local initialization. If it is sampled Low then the CY7C960 will accept an external clock which will advance data through its internal serial chain.

If the CY7C960 receives a High data bit on PDATA during the first initialization clock cycle, this signal becomes LA[1] and the device expects to be configured from the VMEbus. If the device receives a Low data bit on PDATA during the first initialization clock cycle, the CY7C960 will either source or receive a clock from this pin and proceed to load configuration data from the local serial data source.

After initialization, this pin carries the local address signal LA[1].

LA[2] / PDATA / AM[0] - Local Address / Init PROM Data / Local AM code

Input: Yes
Output: Yes
Drive: 8 mA

During device initialization the CY7C960 receives the serial data stream on this signal: if the local serial method is used, then the serial configuration data comes from the PROM; if the VMEbus method of configuration is used, then the VMEbus A[2] signal is used to carry the serially-encoded configuration data, and the appropriate CY7C964 is enabled by the CY7C960 to provide this data to PDATA. After system initialization this signal becomes local bidirectional address signal LA[2] or LA[2]/AM[0] depending on the programming.

If the CY7C960 is programmed for LA/AM multiplexing then the VMEbus signals AM[5:0] will be driven on LA[7:2] by the CY7C960 between VMEbus accesses. A rising edge on LADI is used by local external decoding circuitry to latch the AM codes before this bus changes from providing AM information to providing LA information. The CY7C960 will assert LAEN after LADI rises which disables the CY7C960 LA[7:1] drivers and at the same time enables the CY7C964 LA[7:0] drivers. See section 3.5.3.1.

LA[7:3] / AM[5:1] - Local Address Signals

Input: Yes
Output: Yes
Drive: 8 mA

LA[7:3] make up the remainder of the local bidirectional address bus. The CY7C960 only sources local addresses during VMEbus block transfer operations. During single cycle and interrupt acknowledge accesses the lowest order address byte is sourced from the respective CY7C964. The CY7C960 begins to source the least significant byte of local address information starting with the second cycle in a VMEbus block transfer. This is done to increment the local address by the proper amount, 1, 2, or 4, depending on the VMEbus word size of the transfer.

If the CY7C960 is programmed for LA/AM multiplexing, then the VMEbus signals AM[5:0] will be driven on LA[7:2] by the CY7C960 between VMEbus accesses. A rising edge on LADI is used by local external decoding circuitry to latch the AM codes before this bus changes from providing AM information to providing LA information. The CY7C960 will assert LAEN after LADI rises which disables the CY7C960 LA[7:1] drivers and at the same time enables the CY7C964 LA[7:0] drivers.

PREN* - Serial PROM Enable

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The PREN* signal enables the serial initialization PROM. The CY7C960 asserts this signal when it receives an active SYSRESET* input from the VMEbus or after a power-on reset period.

LDEN* - Latch Data Enable

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The LDEN* signal is used to select one of several potential latched-data sources. During the initialization sequence LDEN* is driven Low at the time that the Address Mask and Compare registers of the CY7C964s are to be loaded. After initialization, LDEN* driven low signifies that an Interrupt Status/ID or an AUTO ID Status/ID cycle is in progress.

RAS* / CS[4] - Row Address Strobe / Chip Select 4

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The RAS*/CS[4] output on the CY7C960 is a dual-purpose pin whose function is selected during the initialization period. When configured to do so, this output controls the row address strobe function for DRAM. In the general-purpose I/O configuration, this output is a user-programmable chip select.

CAS* / CS[5] - Column Address Strobe / Chip Select 5

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The CAS*/CS[5] output on the CY7C960 is a dual-purpose pin whose function is selected during the initialization period. When configured to do so, this output controls the column address strobe function for DRAM. In the general-purpose I/O configuration, this output is a user-programmable chip select.

ROW / CS[2] - Row Address Enable / Chip Select 2

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The ROW/CS[2] is a dual-purpose signal whose function is selected during the initialization period. When configured to do so, this output acts as a row address enable signal, to be used in conjunction with RAS. In the general-purpose I/O mode, this output becomes a user-programmable chip select.

When configured for DRAM operation, CY7C960 still provides CS[2] as an output from another pin: CS[2]/REGION[3].

COL / CS[3] - Column Address Enable / Chip Select 3

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The COL/CS[3] is a dual-purpose signal whose function is selected during the initialization period. When configured to do so, this output acts as a column address enable signal, to be used in conjunction with CAS. In the general-purpose I/O mode, this output becomes a user-programmable chip select.

DBE[3:0] - Data Byte Enables [3:0]

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

These four signals provide the byte enables for local circuitry, either DRAM or I/O. The size of the VMEbus data transaction is decoded by CY7C960 from the state of the DS0*/DS1*/LWORD*/A1 VMEbus signals, and the appropriate byte enable signals are driven. The active state of the signal is user-programmable during configuration. DBE3 represents LD[7:0]. DBE0 represents LD[31:24].

CS[1:0] - Chip Select [1:0]

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

These two signals are chip select outputs that are available whether the CY7C960 is configured for DRAM or for I/O operations. The behavior of these pins is determined during configuration.

R/W* - Read/Write

Input: No
Output: Yes
Drive: 8 mA

R/W* is the local signal that determines if the cycle in progress is a read operation or a write operation. The CY7C960 asserts this signal Low during write operations.

LIRQ* - Local Interrupt Request

Input: Yes
Output: No
Active: Low

LIRQ* is the local interrupt request input. Asserting this active Low input causes the CY7C960 to assert the VMEbus IRQ* output signal.

LACK* - Local Data Acknowledge/Local Bus Hold-off

Input: Yes
Output: No

The LACK input is used to acknowledge a local data transfer cycle. Asserting this active Low signal causes DTACK* to be driven on the VMEbus. The user may assert this signal continuously, which causes the CY7C960 to time data cycle acknowledgments: or the user may withhold assertion of LACK in order to handshake the acknowledge.

If the CY7C960 has been programmed for local bus hold-off mode, then LACK is also used to keep the CY7C960 off the local bus. If LACK is deasserted after LADI falls between VMEbus operations then the CY7C960 will three-state its local drivers and place itself in stand by until LACK is asserted again.

REGION[2:0] - Local Slave Decode Inputs

Input: Yes
Output: No

The REGION[2:0] inputs are user-programmable address decode inputs. External decoder circuitry, such as the CY7C964's, drives these signals when VMEbus addresses match user-defined values. CY7C960 uses these signals together with the REGION[3] and AM codes to determine its reaction to the VMEbus cycle. (See section 3.9.1, Region Mapping.)

CS[2] / REGION[3] - Chip Select [2] / Local Slave Decode Input

Input: Yes
Output: Yes
Drive: 8 mA
Active: Programmable

The CS[2]/REGION[3] signal pin has two user-configurable modes of operation. If CY7C960 is configured for I/O operation, the pin becomes the REGION[3] input, providing another decode input, a total of 4. If CY7C960 is configured for DRAM, then only three REGION inputs are required, and the pin becomes CS[2]. (See section 3.9.1, Region Mapping.)

CLK - Clock Input

Input: Yes
Output: No

The CY7C960 will operate with any input frequency in the range of 30-80 MHz. However, the VMEbus will suffer in performance if this clock is slower than 80MHz. All output events occur on the rising edge of the clock input. All internal states are timed by this signal.

3.3.3 Local Buffer Control Signals

LDS - Local Data Select

Input: No
Output: Yes
Drive: 8 mA

During multiplexed data VMEbus transactions this pin is used by the CY7C964s to select internal registers. During initialization, this signal determines which of the mask or compare registers is loaded within the CY7C964s.

LADI - Latch Address In

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The Latch Address In signal controls the address latches within the CY7C964 that contain address information to be written to the local bus. The LADI signal is asserted shortly after the CY7C960 detects an assertion of the VMEbus AS* signal and can be used externally to latch local AM codes or LA[31:1] values. In system designs that use the CY7C960 with the CY7C964s this signal can directly connect to the LADI inputs on the CY7C964s.

The state of this pin is sampled immediately after the power-on reset period expires. If it is sampled High, then the LADI signal will be active Low. If it is sampled Low, then the LADI signal will be active High.

LAEN - Local Address Enable

Input: No
Output: Yes
Drive: 8 mA
Active: Programmable

The Local Address Enable signal enables the local address output buffer within the CY7C964s. The state of this pin is sampled immediately after the power-on reset period expires. If it is sampled High, then the LAEN signal will be active Low. If it is sampled Low, then the LAEN signal will be active High.

In system designs that use the CY7C960 with four CY7C964s, this signal is connected to only the least significant device and is active High by default via a weak internal pull-down resistor. The LAEN inputs of the other three devices are tied High, or controlled by external logic allowing multi-port local bus accesses. This convention allows the CY7C960 to disable the address from the least significant CY7C964 so that it can source the least significant byte of local address during block transfer operations.

LEDI - Latch Enable Data In

Input: No
Output: Yes
Drive: 8 mA

LEDI is designed to control a transparent latch of the type used within the CY7C964. If this output is Low, data from the VMEbus flows through to the local data bus. Asserting this signal High closes the latch, maintaining the data present at the rising edge. System designs that use the CY7C960 with CY7C964s can tie this output directly to the associated LEDI input on all of the CY7C964s.

LEDO - Latch Enable Data Out

Input: No
Output: Yes
Drive: 8 mA

LEDO is designed to control a transparent latch similar to the type used within the CY7C964. If this output is Low, data from the local data bus flows through to the VMEbus. Asserting this signal High closes the latch, maintaining the data present at the rising edge. System designs that use the CY7C960 with CY7C964s can tie this output directly to the associated LEDO input on all of the CY7C964s.

DENO* - Data Enable Out

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The DENO* output controls the VMEbus data bus drivers in the CY7C964s. When data is to be driven onto the VMEbus, a Low level is driven from this output. A High level on this output signifies that the VMEbus drivers are high impedance. This pin should be connected to all the DENO* pins on the CY7C964s.

DENIN*, DENIN1* - Data Enable In Signals

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The DENIN* and DENIN1* outputs control the latching of sections of the VMEbus data. These two control signals in association with the local SWDEN* signal allow all VMEbus transfer widths to be transmitted to the local peripherals. These signals are intended to be connected to the DENIN* and DENIN1* inputs on all the CY7C964s to enable the local bus drivers. See section 3.6.3, Swap Buffer Control.

ABEN* - Address Bus Enable

Input: No
Output: Yes
Drive: 8 mA
Active: Low

ABEN* is the VMEbus address bus enable out signal. The CY7C960 asserts this signal Low during VMEbus D64 block transfer read operations, to enable the second longword of data information onto the VMEbus. This output should be connected to the corresponding ABEN* inputs on all four CY7C964s. A pull-up resistor is also needed on this pin for proper operation.

STROBE - CY7C964 Strobe Control

Input: No
Output: Yes
Drive: 8 mA

The STROBE output allows the CY7C960 to control the register loading on the CY7C964s. Loading the CY7C964 Address Mask and Compare registers requires assertion of the STROBE signal, with the LDS signal providing the signal that determines which of the two registers, mask or compare, is loaded. The STROBE output performs this function during system initialization. Designs that use four companion CY7C964s should tie this output to the associated STROBE input on all four CY7C964 devices.

D64 - CY7C964 D64 Control

Input: No
Output: Yes
Drive: 8 mA

The D64 output informs external hardware that a D64 VMEbus block transfer operation is in progress. When a D64 VMEbus address decode cycle is valid this signal asserts High. Designs that use four companion CY7C964s should tie this output to the associated D64 input on all four CY7C964 devices.

SWDEN* - Swap Data Enable

Input: No
Output: Yes
Drive: 8 mA
Active: Low

The SWDEN* signal controls a local swap buffer. This buffer is used to move data to the proper section of the local data bus during appropriate byte and word width transfers. The CY7C960 local bus ordering convention matches that of the MC68020. This signal asserts Low when a byte or word width transfer, which always are moved on D[15:0] of the VMEbus, must end up on D[31:16] of the local data bus. (See section 3.6.3, Swap Buffer Control.)

3.4
Programming the CY7C960

The VMEbus board that is designed to use the CY7C960 has what might be called a personality. This encompasses such things as DRAM (speed and amount), I/O circuitry, SRAM, and other matters under the control of the board designer. To a large extent, this personality has to be known to the CY7C960. Some portion of this personality is determined when the board is developed, and is not changed during the board's lifetime. Other aspects of the personality could conceivably change from time to time, such as the address to which the board responds, or the type of VMEbus transaction that it recognizes. All of this information must be provided to the CY7C960 each time the power is cycled or SYSRESET* is asserted. The designer might choose to provide switches that convey part of the personality, or he may rely upon some other VMEbus board to download the information. The CY7C960 has been designed to support these diverse programming requirements.

First consider the address on the VMEbus to which the board responds. This may be a very simple application which has only one address range within which to respond; or it may be a complex address map involving AM Code discrimination, and multiple address ranges. In either case, the external address comparator circuitry needs to be programmed with the slave address ranges in which the application will respond. The CY7C960 supports the loading of this address comparator circuitry from either local circuitry, or from the VMEbus.

For configuration of the other parameters of the user's application a serial bit stream is used. The CY7C960 contains approximately 380 programmable bits, each of which must be set to the right value for the user's application in order to ensure correct functionality (see *Figure 3-9*). There are two methods that can accomplish this: the VMEbus can be used to pass parameters from a VMEbus master to the CY7C960, or a local serial PROM can be used.

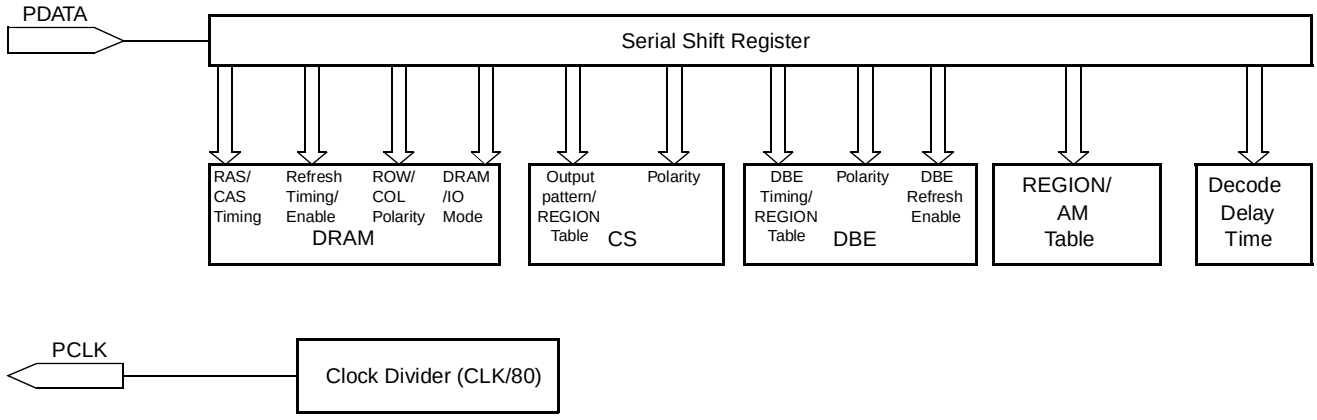


Figure 3-9. CY7C960 Initialization Block Diagram

Thus in summary, the slave address map can be obtained either from local circuitry or the VMEbus, and the configuration data can likewise be obtained either from the VMEbus or from local circuitry.

The CY7C960 is designed for use with the CY7C964 device. The CY7C964's address and mask registers can be used during the configuration process and the CY7C960 provides timing and control signals during initialization that are intended to load these address and mask registers. If the target application does not use the CY7C964's registers as part of the slave address decoder, these signals can be used by user-defined circuitry to configure the address decoder. Refer to Section 4, The CY7C964 Bus Interface Logic Circuit, for more information regarding the signals used to configure the address and mask registers.

The LAEN signal is programmed during the power-on reset period. This signal is used by the CY7C960 to control the external three-state buffer connected to LA[7:0], enabling the CY7C960 itself to drive LA[7:0] at certain times. If using the CY7C964, LAEN is required to be active High; other devices may require LAEN to be active Low. During the power-on reset period the LAEN pin is sampled to determine whether it is pulled up externally, and if so the LAEN output becomes an active Low signal.

3.4.1 Configuration Bit Stream

The 380 bits that make up the configuration data are provided to the CY7C960 by either a series of VMEbus transactions, or by a local serial device such as a serial PROM. The various fields within the bit stream are used to determine the behavior of the CY7C960, and hence the application, after configuration is complete. For the duration of the configuration process two pins, LA[2:1], assume the roles of serial clock and serial data signals. Another pin, PREN*, is used to enable the external serial device during configuration.

The CY7C960 expects the serial data to arrive on pin LA[2] so if the VMEbus configuration method is used, it is the responsibility of the remote VMEbus master to encode the desired configuration stream into 380 VMEbus cycles whose VME address A[2] is controlled by the desired configuration data bit. Likewise, if using a local serial device, the data output from the device shall be connected to LA[2].

To assist in the generation of the correct configuration data file, there is a configuration program available, called WinSvic, that runs on a PC using Windows. This program has extensive help menus which can be used to better understand the configuration bit stream. See section 3.4.6 for more details on this development tool.

For applications using a local serial device the serial clock may be generated by the CY7C960, or it may be provided to the CY7C960 by external circuitry.

The CY7C960 determines which method of configuration is desired, and whether to drive a serial clock or not, as part of the power-on process. This is described in the next section.

3.4.2 Operation at Power-On or Reset

The CY7C960 contains a power-on reset circuit. When power is applied to the device this circuit causes the internal state to be reset. All outputs become three-state until the first rising edge of the clock input is recognized after the internal power-on reset circuit terminates. (The time period of the on-chip power-on-reset circuit depends upon the characteristics of the power supply, but is guaranteed by design to be less than the time period of the rising power ramp.) Also, if a Low level is applied to the SYSRESET* input the CY7C960 immediately three-states its output drivers, resets its internal state and waits for the first clock edge before taking any action. SYSRESET* does not have to be High for the CY7C960 to commence the configuration operation, but must be High for normal operation to proceed after configuration.

PREN* is asserted Low four clocks after the assertion of SYSRESET*, or after a power-on reset. If an external serial device is connected, the first bit must be programmed to be zero. If the VMEbus is to be used, an external pull-up must provide a High level. The PDATA signal (LA[2]) is sensed 40 CLK periods after the assertion of PREN*. If PDATA is sensed to be High, PREN* is driven High one CLK period later.

Also, the CY7C960 senses the level of two pins: LAEN and PCLK (LA[1]). LAEN is sensed to determine whether the signal is desired to be high or low true. PCLK is sensed to determine whether the pin is to be an input or an output. The point at which these signals are sensed is 40 CLK periods after the assertion of PREN*. LAEN has an internal pull-down of approximately 470 kohms. PCLK has no internal resistor. Whichever level is sensed on LAEN is driven from the pin one clock period after it has been sensed. If PCLK is determined to be pulled High, then it is driven Low by the CY7C960 one CLK period after it was sensed: subsequently the CY7C960 will provide a clock output (of frequency CLK÷80). If PCLK is pulled Low, then external circuitry must provide the serial clock signal and must not drive the PCLK input High until at least 40 CLK periods following the falling edge of PREN*.

If the CY7C960 is providing the PCLK signal, the period of each half-cycle is 40 CLK periods, leading to a PCLK frequency of 1 MHz (CLK÷80). PDATA is required to be stable at each falling edge on PCLK, when the PDATA is clocked into the serial configuration register inside the CY7C960. Similarly, if the PCLK signal is provided externally, the PDATA input must be stable at the falling edge of PCLK. The frequency applied to the PCLK pin from an external source does not have to be synchronous to the CLK input as the PCLK input is synchronized internally. The minimum half cycle time for the PCLK input is 50 ns: there is no maximum time as the design is static.

If SYSRESET* is driven Low by a device on the VMEbus, then the CY7C960 reacts in a manner similar to the power-on reset: all outputs become three-state until the first rising edge of the clock (following the falling edge of SYSRESET*), then the PREN* signal is driven Low and the initialization cycle commences. Note that the configuration cycle commences prior to the SYSRESET* signal going inactive: this allows the CY7C960 to complete the configu-

ration cycle within the 200-ms VMEbus specification for SYSRESET* minimum active time if the Serial PROM method is used.

If a front panel reset switch is to be used, then external circuitry combines the two sources of reset (VMEbus SYSRESET* and switch reset), and connects the result to the SYSRESET* pin. The behavior of the CY7C960 is, of course, identical in the two different methods of applying reset. (Note, however, that if a manual reset is performed while VMEbus activity is taking place, the CY7C960 will ignore any VMEbus transactions to any and all Regions until the configuration has been loaded. This may cause the system timer to timeout, and the transaction will cause a BERR*.)

Following the reset and configuration sequence, the CY7C960 performs a DRAM refresh operation (if enabled to do so in the configuration sequence). This is illustrated in *Figure 3-10*. This allows a “warm” reset whereby data in DRAM is preserved across reset operations that do not involve power-down. The length of the refresh depends upon the DRAM configuration selected during initialization: the possible values are from zero bursts to 128 bursts, each burst being four CAS*-before-RAS* cycles whose cycle timing is taken from the configured values. During the long refresh operation, the CY7C960 monitors the VMEbus: if a transaction is directed to the CY7C960 which does not require the use of the DRAM circuitry (for example, a transaction addressed to a Region where DRAM is not enabled), the transaction is completed normally. If the DRAM access is required, the cycle cannot complete until the refresh burst is ended. The detail of the behavior depends upon the transaction: a read operation of any sort cannot commence until refresh is complete; for write operations, the first write cycle is DTACK’ed because the CY7C960 always posts write data, but subsequent write cycles will not be acknowledged until the DRAM refresh burst has completed and the posted data has been written to DRAM. In all cases, the CY7C960 completes the transaction correctly.

Following the power-on or reset configuration sequence, and the refresh burst (if enabled), the CY7C960 will respond to any VMEbus transfers that are enabled by the configuration.

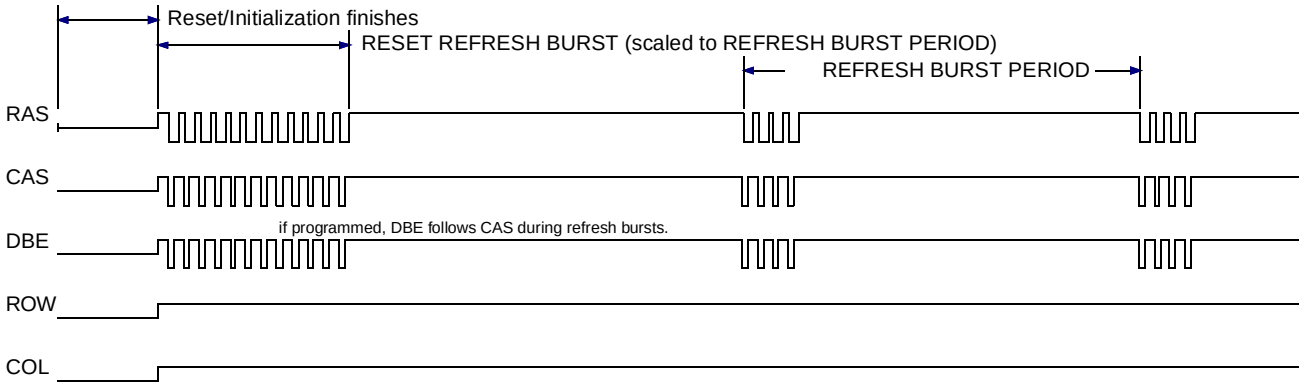


Figure 3-10. Refresh Burst after Initialization

3.4.3 VMEbus Method

After 40 clock periods, the CY7C960 senses the value of PDATA and PCLK. The VMEbus initialization method is automatically invoked by the CY7C960 if its PDATA pin is at logic 1 at power on or during a system reset. (This may be easily accomplished by using a pull-up resistor for the local address bus.) If PDATA is indeed High at this time, then after SYSRESET* is deasserted on the VME backplane, CY7C960 asserts its interrupt request line. This interrupt should be connected to IRQ2* of the VMEbus backplane as the initialization interrupter inside CY7C960 is hard-coded to respond to that level. (After configuration is complete, the user-programmed interrupt level is used, but this must match the actual hardware level connected on the backplane. Therefore if the user wishes to use a level other than 2, external circuitry is necessary to switch the IRQ output pin to the different VMEbus IRQ signal after initialization.) See section 3.5.6.

The VMEbus Interrupt Handler must provide a level 2 interrupt acknowledge cycle: the interrupt acknowledge daisy-chain will ensure that the CY7C960 closest to the handler will respond to the IACK cycle. When the IACK cycle is detected, CY7C960 asserts its LDEN pin, which provides a signal for enabling initialization interrupt status ID (user defined) onto the local data bus. This status comes from user-defined external circuitry such as jumpers or latches. The CY7C964s are controlled by the CY7C960, and hence pass the vector to the VMEbus. CY7C960 will acknowledge the next VMEbus master write qualified by the A16 (supervisory or nonprivileged) or CR/CSR AM code and write the VMEbus data to the address registers of the CY7C964s. Thus, the CY7C964s are now primed to react to that VMEbus address, unmasked, as the action of writing the compare register clears the mask register in the CY7C964. The interrupt acknowledge cycle and the following single master write cycle must be indivisible for guaranteeing system integrity (the board has yet to be positioned in the VMEbus slave address map, and therefore will respond to ANY A16 or CR/CSR write cycle). If this indivisibility is not possible, then the local method of configuration must be used.

From this point, CY7C960 expects a series of A16 (or CR/CSR) master write transactions to be written to the address just programmed. As the board address has now been set in A16 or CR/CSR space, there is no requirement for indivisibility. (There is a requirement that the AM code for each write transaction be the same as that used in the first transaction: start with A16, stay with A16; or start with CR/CSR, stay with CR/CSR.) Each transaction carries a single configuration bit (just like the serial PROM method). The bit must be encoded on the VMEbus A[2] signal. CY7C960 counts the transactions. After the last transaction of the serial stream is acknowledged, two more transactions are required, with the same address and AM code. The data from these transactions are loaded into the compare and mask registers of the CY7C964s to complete the initialization transaction sequence: the signals that control the loading of the CY7C964 registers are LDS and STROBE (driven by the CY7C960), and MWB* and BLT*, (pulled High).

The REGION[3:0] inputs to the CY7C960 should be driven by the user's external slave address map decoder. For the very first VMEbus write cycle following the IACK cycle, the REGION

inputs are ignored. Since the write cycle is not qualified with address (i.e., REGION), it must immediately follow the IACK cycle. For subsequent configuration cycles the REGION inputs must be driven with “xx00” if the CR/CSR AM Code is used; and with “xxx0” (x = don’t-care) if one of the two A16 AM Codes is used. Note that this system allows the user to connect the VCOMP (compare) outputs from the appropriate CY7C964 directly to the related REGION input as a simple configuration address map decoder. If the user requires a more complex address map, the configuration requirement is easily accomplished.

It should be understood that certain device behavior is not available during the configuration sequence. Obviously, no VMEbus slave operations can take place other than the configuration sequence itself. The CY7C960 does not provide AM Codes from the LA[7:2] pins until after the configuration sequence has completed as the bit that enables this operation is embedded in the configuration sequence. Similarly, the first IACK cycle is self timed because the bit which enables IACK cycles to be locally handshaked is embedded in the configuration data.

Figure 3-11 shows the timing of the start of the VMEbus initialization sequence, and *Figure 3-12* shows the end.

In summary, the sequence starts by setting the address of the CR/CSR space, then the CR/CSR is used to load the CY7C960, and finally the board’s Slave Address is loaded. CY7C960 exits initialization mode and, after the refresh burst if so enabled, is ready for service. (Note that this initialization sequence is compliant with the Auto Slot ID utility of the VME64 specification.)

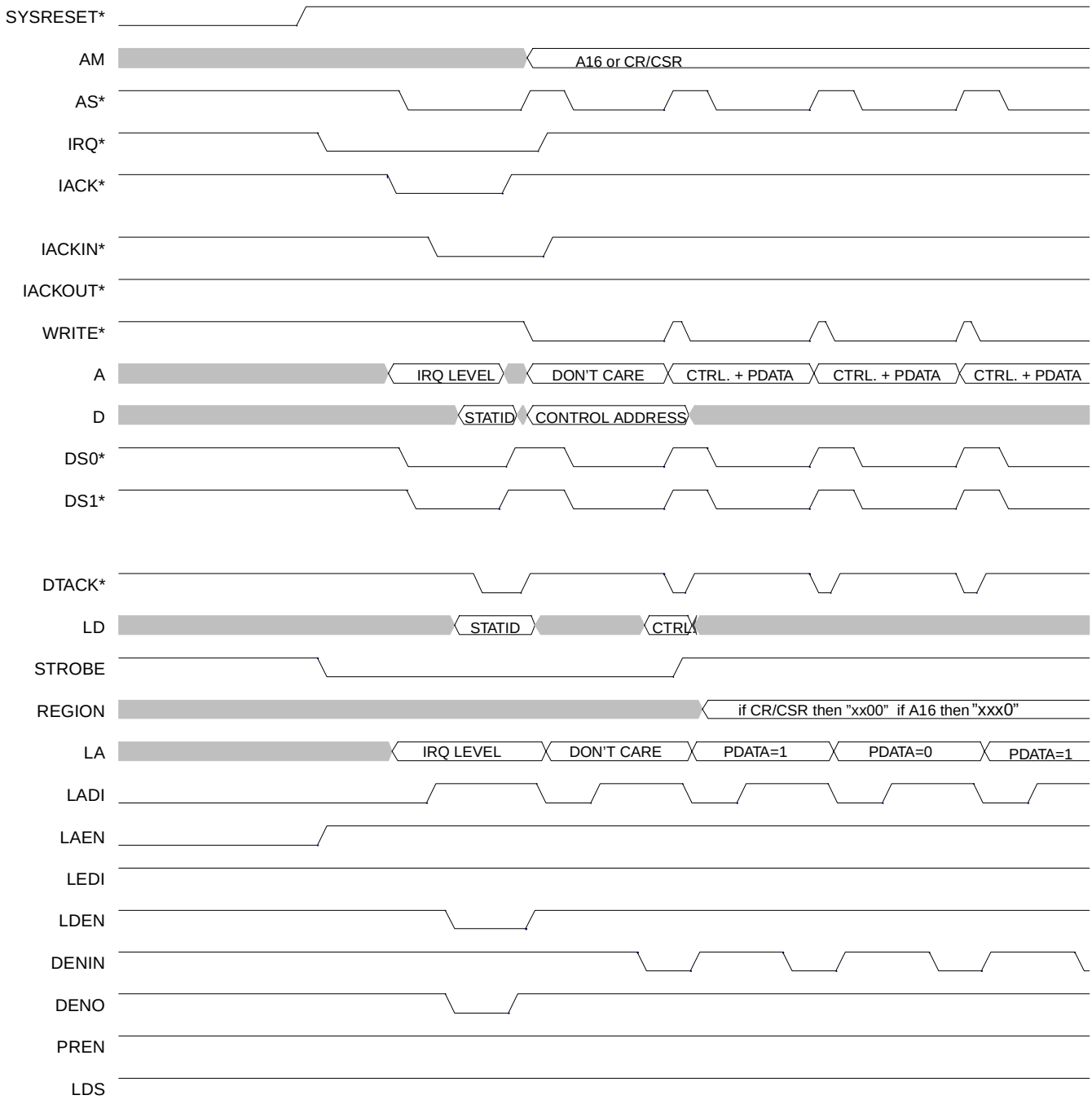


Figure 3-11. VMEbus Initialization Start

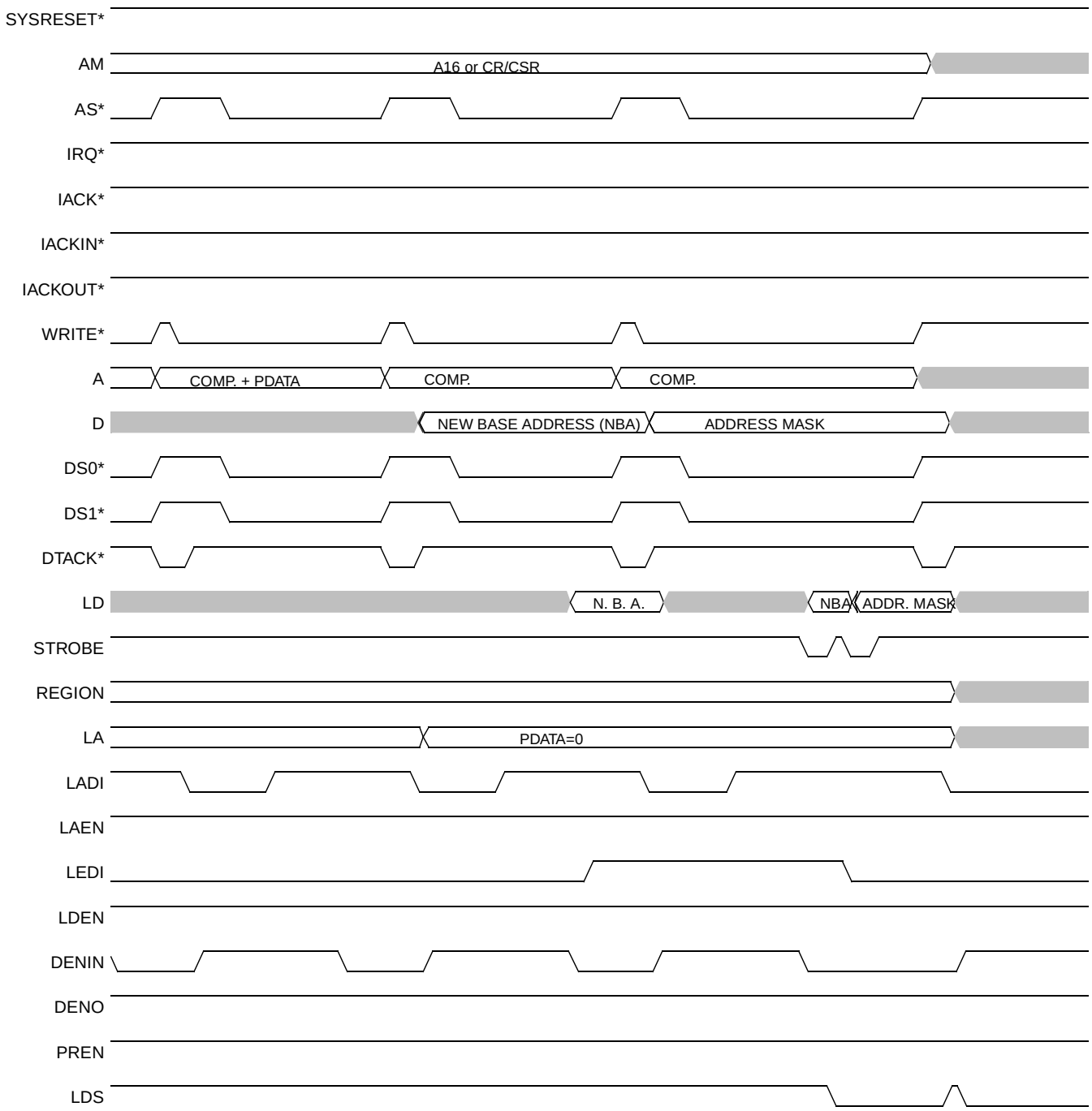


Figure 3-12. VMEbus Initialization End

3.4.4 Serial PROM Method

Following the assertion of SYSRESET*, or a power-on reset, CY7C960 drives PREN* Low and, after 40 CLK periods, senses the state of PDATA and PCLK. If the PDATA pin is Low, the serial method of configuration is used. The PCLK pin is sampled at this same point to determine clock direction. If PCLK is High (as controlled by an external resistor), CY7C960 will drive PCLK to strobe serial data into the CY7C960 from a serial PROM. If PCLK is sensed Low (as controlled by an external resistor), PCLK becomes the clock input of the CY7C960 serial program register. This option is to allow for programming the CY7C960 from a local microprocessor.

Once serial configuration has commenced, it continues for a predetermined number of clock pulses on the PCLK pin. If PCLK is sourced by the CY7C960, the clock frequency is CLK divided by 80 (maximum rate of 1 MHz), allowing industry-standard serial PROMS to be used. If PCLK is an input, the CY7C960 expects the PCLK signal to begin in the Low state, then toggle at a rate not to exceed 10 MHz. Regardless of PCLK direction, the rising edge on PCLK is assumed to clock the external serial device, and the falling edge is used by the CY7C960 to sample the PDATA input into the internal configuration data stream. PDATA is taken by the CY7C960 one CLK period after the High-to-Low transition of PCLK. When the correct number of data bits have been sampled by the CY7C960, PREN* is driven High. This PREN* transition can be used to disable external clock circuitry.

Once the entire stream of configuration data has been read into the CY7C960, the CY7C964s (or the external slave address decoders) are now to be configured. If the last bit of the serial bit stream was a 0 (Low), the CY7C960 assumes that the CY7C964s are to be configured using local resources. Therefore, the user-defined address and mask data (from latches or jumpers) are enabled onto the local bus by a combination of the LDEN*, STROBE, and LDS signals. LDEN* is driven Low when the CY7C960 detects that the serial PROM data has been loaded, and then LDS selects first the address register (High) then the mask register (Low). The CY7C960 controls the signal, STROBE, to ensure correct operation. Following the register load operation, LDEN* is driven High to disable the user's latches.

The CY7C960 performs the refresh burst, if so enabled, and is ready for service. The serial configuration portion of this operation takes approximately 380 µsec if the CY7C960 is providing the PCLK signal, and hence can be completed within the period of a normal VMEbus SYSRESET*. If the configuration sequence is invoked from a manual reset, it should be noted that the CY7C960 ignores any VMEbus activity until the entire configuration sequence is completed. VMEbus response during the long refresh burst is as described above.

Figure 3-13 shows the start of the Serial programming sequence, and *Figure 3-14* shows the end. SYSRESET* is Low as the sequence starts. This illustrates that serial programming can be completed before the VMEbus SYSRESET* period (200 ms) ends. There is no actual requirement that SYSRESET* be Low.

The first bit of PROM data after PREN* goes Low is 0: subsequent bits contain appropriate programming information. After the final bit of PROM Data has been read as a 0, the CY7C964s are programmed. First LDEN* and STROBE go Low, then STROBE goes High to load the Address Compare Register. While LDEN* remains Low, one clock after STROBE goes High, LDS is driven Low to select the Mask Data. A second STROBE pulse is issued to complete the process. The pulsewidth of STROBE assertion is 40 CLKs High/40 CLKs Low. See Section 4, The CY7C964 Bus Interface Logic Circuit, if using CY7C964s.

The CY7C960 board is now ready for service, once SYSRESET* goes High.

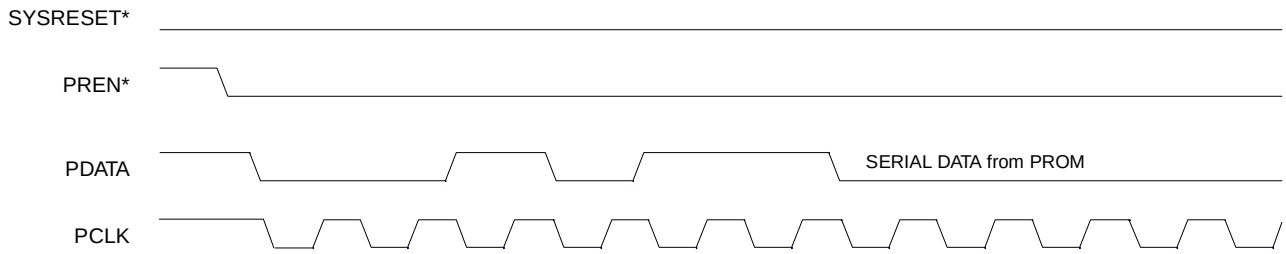


Figure 3-13. Serial Method Start

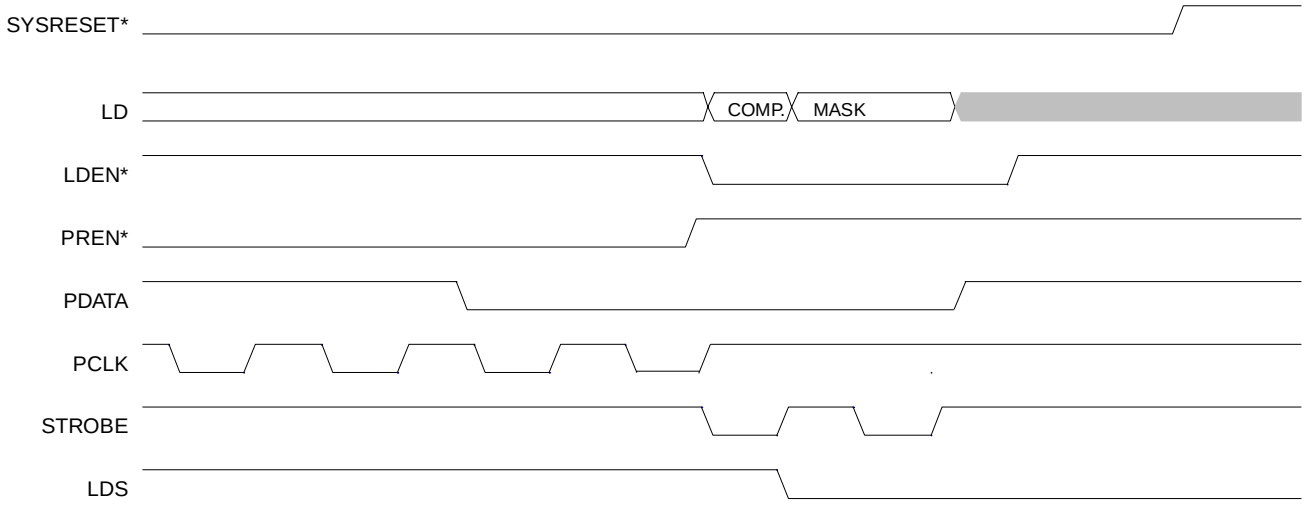


Figure 3-14. Serial Method End

3.4.5 Combination Method

The third method of initializing the CY7C960 is simply a combination of the above two methods. The serial PROM is used to load the configuration data stream, then VMEbus transactions are used to load the address and mask registers of the CY7C964s. The serial device operates as described above, except that the final bit is programmed to 1. This is a flag to the CY7C960 to use the VMEbus for the CY7C964 configuration. First the CY7C960 drives PREN* High and asserts the IRQ* output, which must be connected to the VMEbus IRQ line designated in the configuration data stream that was just loaded. This interrupt request must be interpreted by the interrupt handler in a similar manner to that described earlier for the VMEbus

method: the interrupt acknowledge cycle causes CY7C960 to enable the initialization status vector using LDEN*. CY7C960 then interprets the next A16 (or CR/CSR) write transaction to load the CY7C964 address register with the control address. Then it expects the next two A16 (or CR/CSR) write transactions (to the address just loaded) to be the address and mask data for the CY7C964s. See section 3.5.6.

Following receipt of these three transactions the CY7C960 initialization is complete.

Regarding the interrupt level used in this combination method: though any level can be used, if it is desired to be compliant with auto slot ID level 2 should be used.

Figure 3-15 shows the timing of the signals involved in the CY7C964 Address and Mask register programming that occurs at the end of the serial load sequence. Following the receipt of the final bit of serial data, CY7C960 waits for a High on SYSRESET* (if SYSRESET* is Low). Then IRQ* is driven Low. The resultant IACK cycle (AS* #1) causes CY7C960 to drive LDEN* Low, which causes local circuitry to drive a vector that is passed to the VMEbus. The interrupt handler must be programmed to then write the CR/CSR base address (AS* #2). This can use the CR/CSR AM Code, or one of the two A16 AM Codes. Note that, as the board has not yet been programmed, the VMEbus Address, and hence the REGION inputs to CY7C960 are not significant. The implication of this is that the IACK cycle and the first write cycle must be indivisible on the VMEbus. The CY7C964s are primed to receive the Address Compare data as STROBE previously went Low while LDS was High. DENIN* and DENIN1* go Low, which drives the latched VMEbus data onto the local bus. This is the Address Compare register value, the CR/CSR base address value. The CY7C960 drives STROBE High, latching the value into the CY7C964 Compare register (and clearing the Mask register in the process). Now the CY7C964s have a meaningful value to compare against VMEbus address, and the REGION inputs to the CY7C960 become significant. If A16 cycles are used the CY7C960 expects the value “xxx0” on REGION[3:0], or if CR/CSR cycles are used, “xx00”. The next two write cycles must be of the same AM Code as the first write cycle, but they do not have to be indivisible. AS* #3 provides the “final” address for the board’s slave address map, and AS* #4 provides the associated mask pattern. Note that during cycle #3 LEDI goes High: this latches the value of the VME data bus inside the CY7C964. Then during cycle #4, while the VMEbus is carrying the mask value, the data held from cycle #3 is loaded into the Address Compare register (STROBE going High), then LDS and STROBE toggle, selecting the Mask Register. Finally, LEDI goes Low allowing the Mask data to flow from the VMEbus, and STROBE goes High loading the Mask register. The CY7C964s are now primed to respond to the appropriate address.

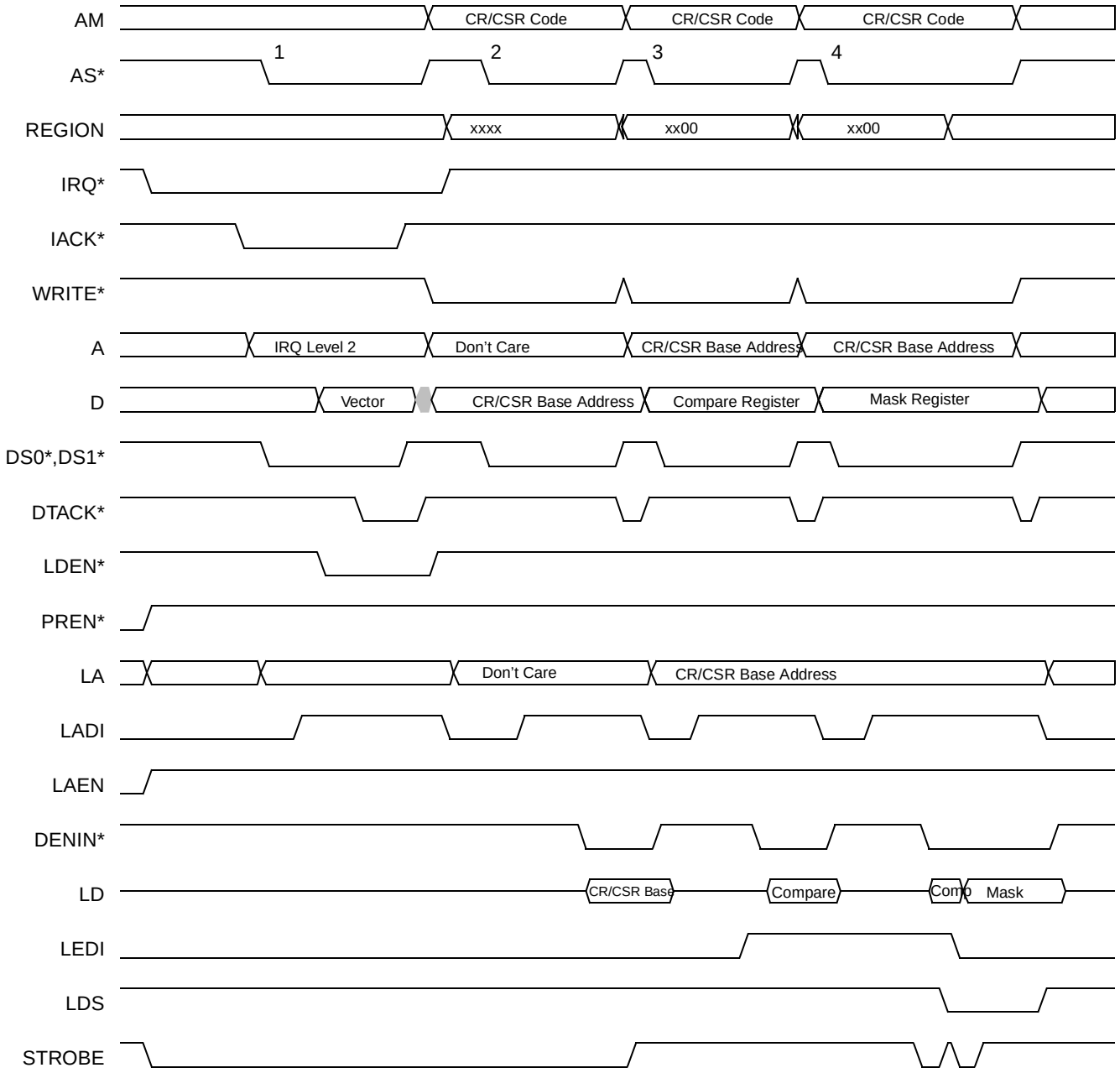


Figure 3-15. Combination Method Timing

3.4.6 Configuration Software

To assist in developing and debugging CY7C960 configurations for specific applications, Cypress has developed a Windows-based program called WinSvic. This program assists in configuration by leading the designer through all the steps needed to successfully generate the file that will be used to configure the CY7C960 during initialization

WinSvic presents device configuration graphically as a set of nested menu forms. The main screen provides for selection of basic options, such as DRAM/IO versus I/O mode, Decode Delay, and the method by which the CY7C960 is to be configured (Serial, Serial/VME, or VME). Selecting options changes the availability of submenu forms. For example, when DRAM/IO mode is selected, the I/O configuration submenu button is greyed out.

Selecting I/O mode allows the designer to move into the IO submenu. Here the polarity of the six available chip select outputs are individually specified, and for each Region, the pattern to be driven from the Chip Selects when that Region is addressed. Chip Select Assert Time is also specified for each individual Region.

A second submenu, the AM Code dialog box, allows specification of which AM codes are to be accepted as valid for each Region. The special AM codes, CR/CSR, USER1, USER2, LOCK, and SERIAL, are also enabled and assigned in this submenu.

If DRAM/IO mode is selected, the designer can access the DRAM submenu to specify various DRAM configuration options. DRAM timing parameters such as, RAS*/CAS* Delay, RAS* PreCharge, and CAS* Assert are specified in terms of CY7C960 CLK periods. Refresh controls, DRAM Region enables, Chip Select pattern and polarity, and ROW/COL control polarity are all selectable from this form.

Once the appropriate submenus have been negotiated, the main menu is revisited, and the configuration file can be written to disk using the appropriate button or menu choice.

Context sensitive help is available, and the program can write existing names for modification and adjustment. WinSvic guides the designer through the configurability of the CY7C960, making clear the relationship between operating modes and programmable features.

3.4.7 Programmable Features

During configuration the CY7C960 functionality is selected, and many timing parameters are set. For more information on the individual functions read the appropriate chapter of this document. The following two tables serve as a brief outline of the various functions.

The CY7C960 can be viewed as having two sections: Local Bus side, and VMEbus side. *Table 3-1* indicates the programmable functions that the user employs to affect the behavior of the local side of the chip. The part has two basic modes of operation: DRAM/IO and I/O. A bit in the configuration bit stream selects between the two modes. *Table 3-2* illustrates what control the user has over the VMEbus responses.

Table 3-1. Local Bus Programmable Functions

Programmable Function	DRAM Mode	I/O Mode
Polarity of data byte enable pins	All 4 either hi or low true	All 4 either hi or low true
Chip Select Output Pattern	3 signals	6 signals
DBE assert timing	If DRAM disabled in region, 3 to 18 clock periods	3 to 18 clock periods
Chip Select Output Polarity	Individually selected hi or low	Individually selected hi or low
Polarity of ROW/COL signals	Individually selected hi or low	Not Available
RAS/CAS Delay	2 to 9 clock periods	Not Available
RAS Precharge	5 to 12 clock periods	Not Available
CAS Assert Time	3 to 10 clock periods	Not Available
CAS Precharge	1 to 8 clock periods	Not Available
Refresh burst period	(1 to 255) * 256 clock periods	Not Available
Enable Refresh	Selectable	Not Available
Use DBE pins for refresh	Selectable	Not Available
Number of Regions	0 to 8	0 to 16
Enable DRAM Region access	Individually	Not Available

Table 3-2. VMEbus Programmable Functions

Function	Programmable Response
Decode delay from Address Strobe	2 to 5 clock periods
User group 1 (AM Code 18 - 1F hex)	Any 1 region, or none
User group 2 (AM Code 10 - 17 hex)	Any 1 region, or none
CR/CSR	Any 1 region, or none
Serial	Any 1 region, or none
Lock	Any 1 region, or none
Block Transfers, including MBLT	Any number of regions
Data Access	Any number of regions
Program Access	Any number of regions
Supervisory Access	Any number of regions
Non Privileged Access	Any number of regions
A64 Transfers	Any number of regions
A40 Transfers	Any number of regions
A32 Transfers	Any number of regions
A24 Transfers	Any number of regions
A16 Transfers	Any number of regions
D32/D64 Transactions	Any number of regions
Interrupt Level for IACK response	1 to 7

3.5 VMEbus Interface Description

3.5.1 Definition of Terms

DSi*	Either or both of DS0* and DS1*, the VMEbus data strobe signals
DSA*	The first of DS0* or DS1* to be asserted
DSB*	The second of DS0* or DS1* to be asserted
6U, 3U	The two most common sizes for VMEbus boards

3.5.2 Overview

The CY7C960 and its companion part, the CY7C964, provide a VMEbus interface that is fully compliant with IEEE 1014 rev C (original VMEbus specification) and the VME64 specification. *Table 3-3* lists the slave accesses that are supported on the VMEbus.

Table 3-3. VMEbus Transactions Recognized by the CY7C960

VMEbus Cycle Type	Description
D8, D16, D32	8-, 16-, and 32-Bit Single Cycle Accesses. The VMEbus Master asserts DSi*, and the CY7C960 responds with DTACK*. Data is transferred in one cycle.
ADO	Address Only. A VMEbus Master broadcasts an address on the bus without any data strobe assertion. There is no data phase and no local access associated with this cycle.
ADOH	Address Only With Handshake. A VMEbus Master broadcasts an address on the bus and asserts data strobes. The CY7C960 will handshake the address broadcast by asserting DTACK* Low in response to the assertion of data strobes. There is no data phase and no local access associated with this cycle. (The CY7C960 asserts Chip Selects, but not DBE.)
MD32	Multiplexed 32-Bit Single Cycle Access. A VMEbus Master broadcasts a 40-bit address using the address and data lines. The CY7C960 will handshake the address broadcast by asserting DTACK* Low in response to the assertion of data strobes. The Master then initiates another cycle that uses 16 bits of the data bus and 16 bits of the address bus to transfer 32 bits of data in single bus cycle. This transaction is typically used in 3U applications.
D8:BLT, D16:BLT, D32:BLT	8-, 16-, and 32-Bit Block Transfers. The VMEbus Master asserts DSi*, the CY7C960 handshakes with DTACK*. The Master continues with DSi* assertions. When AS* is deasserted, the block transfer is ended.

Table 3-3. VMEbus Transactions Recognized by the CY7C960 (continued)

VMEbus Cycle Type	Description
MD32:BLT	Multiplexed 32-Bit Block Transfer. A VMEbus Master broadcasts a 40-bit address using the address and data lines. The CY7C960 will handshake the address broadcast by asserting DTACK* Low in response to the assertion of data strobes. The Master then initiates a series of cycles that use 16 bits of the data bus and 16 bits of the address bus to transfer 32 bits of data in each cycle of the burst. This transaction is typically used in 3U applications.
D64:MBLT	Multiplexed 64-Bit Block Transfer. A VMEbus Master broadcasts a 32- or 24-bit address on the address lines or a 64-bit address on the address and data lines. The CY7C960 will handshake the address broadcast by asserting DTACK* Low in response to the assertion of data strobes. The Master then initiates a series of cycles that use 32 bits of the data bus and 32 bits of the address bus to transfer 64 bits of data in each cycle of the burst. When AS* is deasserted, the block transfer is ended.
D8:RMW, D16:RMW, D32:RMW	8-, 16-, and 32-Bit Read-Modify-Write Cycles. A VMEbus Master reads data from the slave board, modifies that data, and writes it back to the board in a single transaction. The CY7C960 provides DTACK* in response to each DSi* assertion.
MD32:RMW	Multiplexed 32-Bit Read-Modify-Write. A VMEbus Master broadcasts a 40-bit address using the address and data lines. The CY7C960 will handshake the address broadcast by asserting DTACK* Low in response to the assertion of data strobes. The Master then initiates a cycle that reads 16 bits of the data bus and 16 bits of the address bus, modifies that data, and writes the data back in a single VMEbus access. This transaction is typically used in 3U applications. The CY7C960 provides DTACK* in response to each DSi* assertion.
D32:UAT	32 Bit Unaligned Transfer. A VMEbus Master transfers 16 or 24 bits of data using a single cycle access. For example, a Master may transfer 16 bits of data using VMEbus data Byte 1 and Byte 2. The CY7C960 provides DTACK* in response to the DSi* assertion.
A16, A24, A32, A40, A64	The CY7960 can be programmed to respond to any of the address-based AM codes.
CR/CSR	Configuration ROM/Control and Status Register accesses. The CY7C960 can be programmed to respond to this AM Code.
SERIAL	Serial cycles. The extension to the VMEbus specification defines the use of certain reserved AM Codes for this serial data interface. The CY7C960 can be programmed to respond to these AM Codes.
LOCK	LOCK cycles. The CY7C960 provides limited support by acknowledging the address broadcast from the VMEbus Master. No local signals are driven. The CY7C961 device provides a complete LOCK facility.
USER	16 VMEbus AM codes grouped as USER1(18–1F) and USER2 (10–17). The CY7C960 can be programmed to respond to these transactions.
IACK	The VMEbus Interrupt Handler provides an IACK cycle in response to a VMEbus interrupt. The CY7C960 responds to IACK cycles as appropriate.

There are many features of the CY7C960 interface solution which can be combined in various ways and augmented with external decoder logic to configure a slave board to respond in any of 16 VMEbus address regions. The CY7C960 can be programmed to respond to a subset of the transaction types listed in *Table 3-3* when they occur within user-specified regions of VMEbus address space. (Refer to Programming the CY7C960 for complete details on how to configure the device.) This section will describe how the CY7C960 uses its AM code programming and REGION inputs to respond to certain transactions within certain address regions and how the CY7C960 is selected by a Master. As the CY7C960 is designed to handle the complexity of the VMEbus transaction processing without external assistance, it is not necessary to understand the internal circuitry to any great extent. The following two sections provide the basic knowledge needed to use the device: the rest of the chapter can be viewed as reference material.

3.5.3 Region Mapping

The CY7C960 receives VMEbus address decoder information on its REGION inputs. Slave selection begins by decoding these inputs. There are four REGION inputs available when the CY7C960 is configured as an IO controller and three REGION inputs when it is configured as a combination DRAM/IO controller. Refer to the Local Interface Description for details on these configurations. The IO controller configuration will be assumed for the purpose of this discussion.

The purpose for providing this large number of Regions is to allow a board to respond differently to the various VMEbus transactions that can be enabled. Each Region can have its own “personality.” For example some respond to block transfers, some do not. One or more of the 16 decoded Regions can be designated as the “unselected” regions. The way this is done is described in the next section.

The CY7C964s are configured to drive the VME address onto the local address bus. VMEbus addresses flow through and allow external decode logic to take advantage of the VMEbus buffering provided by the CY7C964 transceivers. In the case of A64 and A40 transactions, the CY7C960 provides two signals, DENIN* and DENIN1*, which can be used to instruct the external decoder circuitry when to sample the VMEbus data signals (available on the local data bus, buffered by the CY7C964 devices) for this multiplexed address information. Thus, the user may choose to implement a comparator externally of any complexity and drive the REGION inputs from this external circuit.

The amount of time the CY7C960 waits before sampling the REGION inputs is specified during initialization. The parameter is called Decode Delay. The CY7C960 starts counting the number of clocks that occur from the assertion of AS* by the VMEbus Master. When the number of clocks equal the Decode Delay, the CY7C960 samples the four REGION inputs to determine which one of 16 possible address regions the cycle is directed towards.

Decode delay starts with the sampling of DSB* instead of AS* in the case of multiplexed address cycles such as A64. This is to allow address information to arrive on the VME data bus.

If the minimum decode delay is specified, then the REGION inputs must be stable when AS* is sampled Low. If the maximum decode delay is specified, then the REGION inputs must be stable five clocks after AS* assertion. See *Figure 3-16* for an example of a Decode Delay of 3 clock periods.

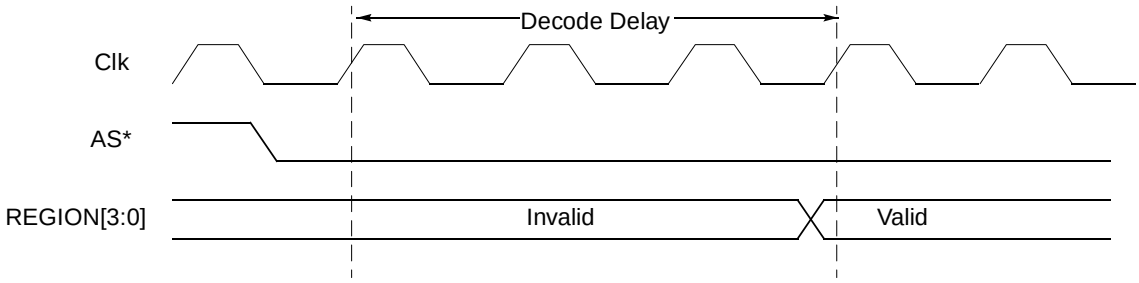


Figure 3-16. Decode Delay Example

3.5.3.1 AM/LA Multiplexing

The AM/LA Multiplexing feature of the CY7C960 is an extension to the Region Mapping capability described above and is enabled by setting a bit in the serial programming stream. This feature adds even more flexibility for mapping slave accesses to specific regions by providing VMEbus AM code information on LA[7:2] during the slave decode period of a VMEbus transaction. During this period the CY7C964s are driving upper address information on LA[31:8] and the CY7C960 is driving AM codes on LA[7:2]. With this feature enabled, external circuitry can be implemented to decode both address and AM code information in parallel during the decode delay period.

To demonstrate the power of this feature, consider a VME slave with one A24 CSR region and one A32 memory region with LOCK support. It is not sufficient to monitor just the VMEbus address lines because they may satisfy both the A24 and A32 decode conditions. For this application an external decoder must monitor the VMEbus AM lines in order to discriminate between A24 CSR and A32 LOCK cycles. This would require connecting AM[5:0] to both the CY7C960 and the external decoder, a violation of the VMEbus specification on signal loading. With the addition of local AM signalling, the CY7C960 and CY7C964s are the only load on the VMEbus and together they pass all available VMEbus addressing information to the local side of the interface for decode processing.

AM/LA multiplexing begins immediately after initialization. The address transceivers on the CY7C964s are enabled onto the upper bytes of the local address bus while the address transceivers on the CY7C960 are enabled onto the least significant byte. VMEbus addresses flow through the CY7C964s and VMEbus AM codes flow through the CY7C960 as shown in *Figure 3-17*.

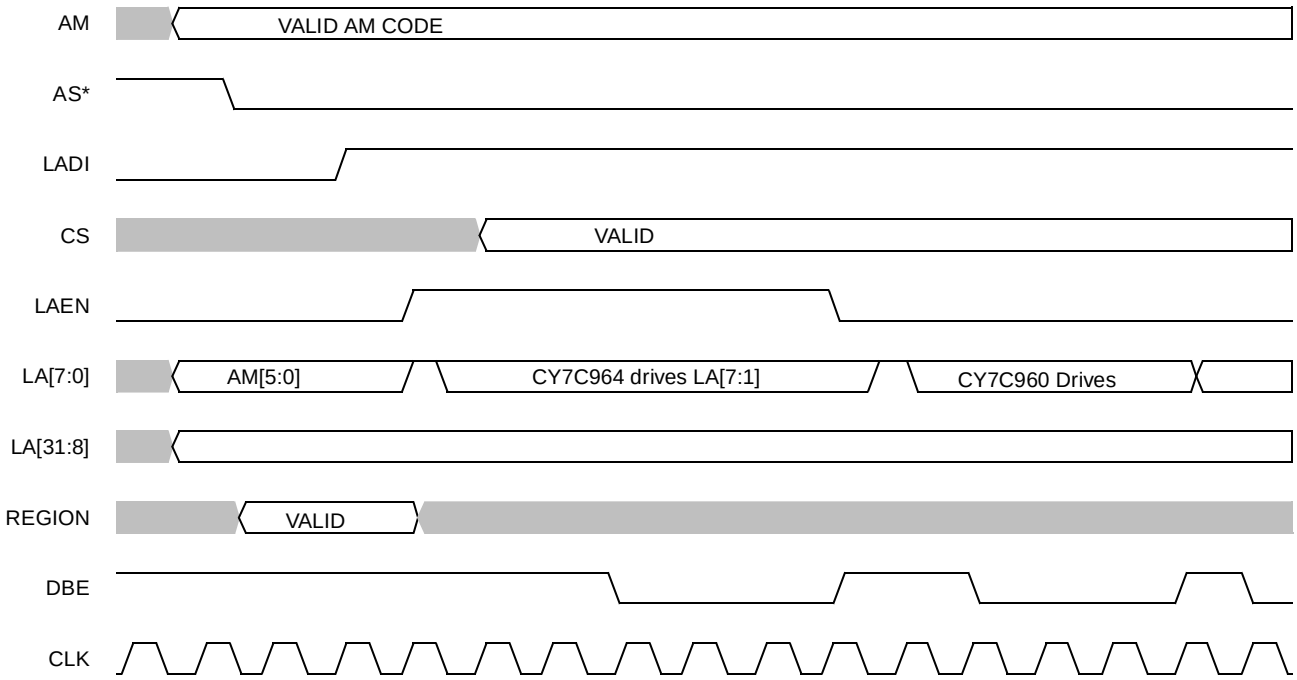


Figure 3-17. AM/LA Multiplexing with Decode Delay = 2T

The assertion of LADI is the event used to latch the local AM codes presented on LA[7:2] within an external decoder. The CY7C960 local address transceivers are disabled and the CY7C964 local address transceivers are enabled one clock after LADI assertion. The VMEbus signals A[7:1] and LWORD* will flow through to LA[7:0] of the least significant CY7C964 at this time. The CY7C960 then stores the values presented to it on LA[1] and LWORD* to complete the slave decode process. These values are used to compute the correct data byte enable pattern during the data phase of the access.

Figure 3-18 shows the effect of introducing Decode Delay to the address phase of a slave access when AM/LA Multiplexing is enabled. The timing of AS* sampling, LADI assertion, and local address bus transition from AM[5:0] to LA[7:1] and LWORD* remain the same as in *Figure 3-17* while the duration of LA[7:1], LWORD*, and the time to valid chip selects will increase according the number of Decode Delay clocks programmed at initialization.

Figure 3-19 represents the same slave access as in *Figure 3-17* except that AM/LA multiplexing is turned off. In this case, all four CY7C964s are enabled onto the local bus while the CY7C960 is waiting to be accessed by a Master. The CY7C960 will not drive the local address bus until the second and subsequent beats of a Block Transfer.

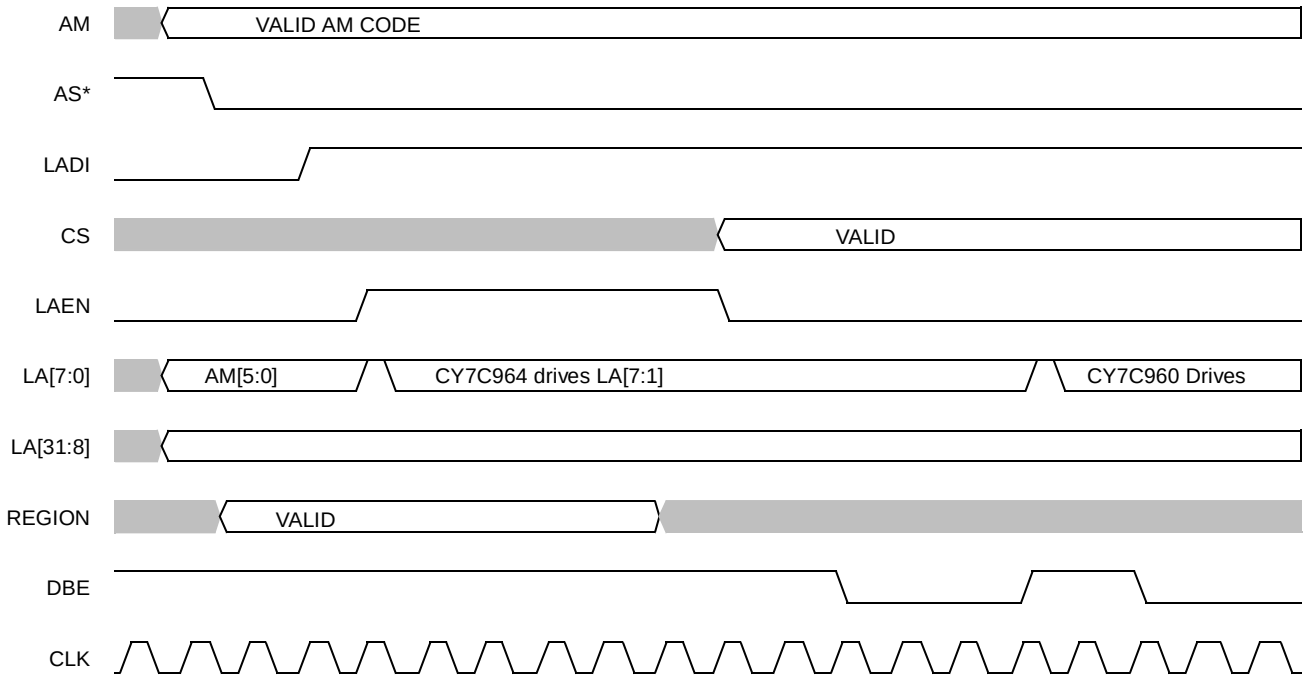


Figure 3-18. AM/LA Multiplexing with Decode Delay = 5T

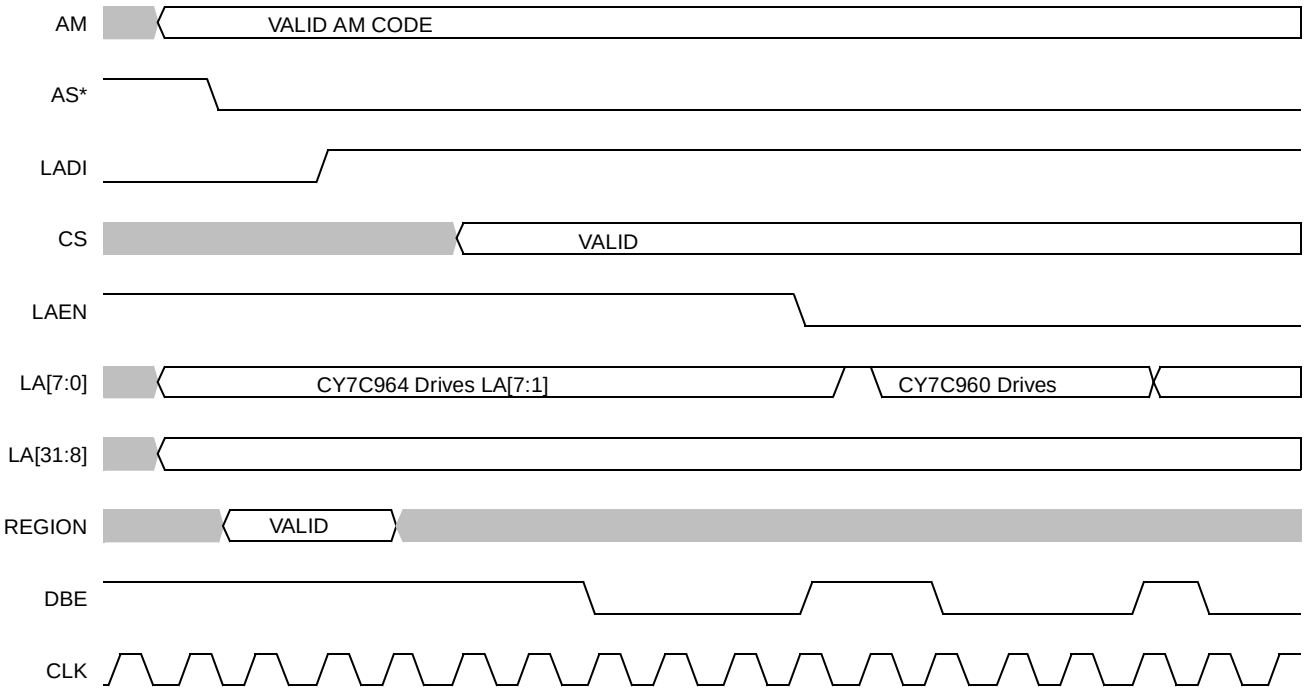


Figure 3-19. AM/LA Multiplexing Disabled with Decode Delay = 2T

3.5.4 Bus Holdoff

The Bus Holdoff feature of the CY7C960 provides the ability to three-state the slave interface chip set (CY7C960 and CY7C964s) and suspend slave decode activity in order to easily dual-port devices mapped as VMEbus slave resources. This feature also extends the functionality of the CY7C961 by allowing local configuration of its master DMA channel. The feature is enabled by setting a bit in the serial programming stream.

The mechanism for Bus Holdoff is overlaid on the CY7C960 LACK pin. When Bus Holdoff is enabled, LACK acquires a local bus grant function in addition to its previously described local transaction acknowledge function. By controlling LACK deassertion, it is possible to suspend all local response of the slave interface chip set, effectively delaying the next VMEbus slave access until LACK is reasserted.

While the CY7C960 is in its holdoff state, output signals DBE, RW, CS, and LA are three-state, and control signals DENIN[1:0]* and LAEN three-state the LD and LA pins of the least significant CY7C964 device. LAEN on the most significant CY7C964s must also be externally controlled during Bus Holdoff. This holdoff configuration is intended to allow simple design of address, data, and control multiplexing to achieve dual-ported local memory architectures.

The CY7C960 Bus Holdoff feature conceptually allows cycle stealing by other local bus masters. This is accomplished by using an output of the CY7C960, the LADI signal, as an interface busy signal. A local bus arbiter can monitor LADI and deassert LACK within two clocks after a deassertion edge on LADI. LADI deassertion represents the end of a VMEbus transaction regardless of whether it was decoded by the CY7C960 interface. If LACK is deasserted in the bus holdoff window, the local bus is three-state until LACK is again asserted. If LACK is deasserted outside the holdoff window, LACK performs its local acknowledge function.

3.5.4.1 Transaction Type Detection

There is another internal decoder: the CY7C960 decodes AM[5:0] and forms an internal Address Modifier Decode Word after the same Decode Delay interval that is used for sampling the REGION inputs. As the VMEbus specification dictates that AM Codes be stable prior to the assertion of AS* then the Decode Delay is unnecessary, but is used for convenience.

Two fields result from decoding the AM Codes: Address Size and Modifiers. *Table 3-4* provides a definition of the CY7C960's classification of AM Codes.

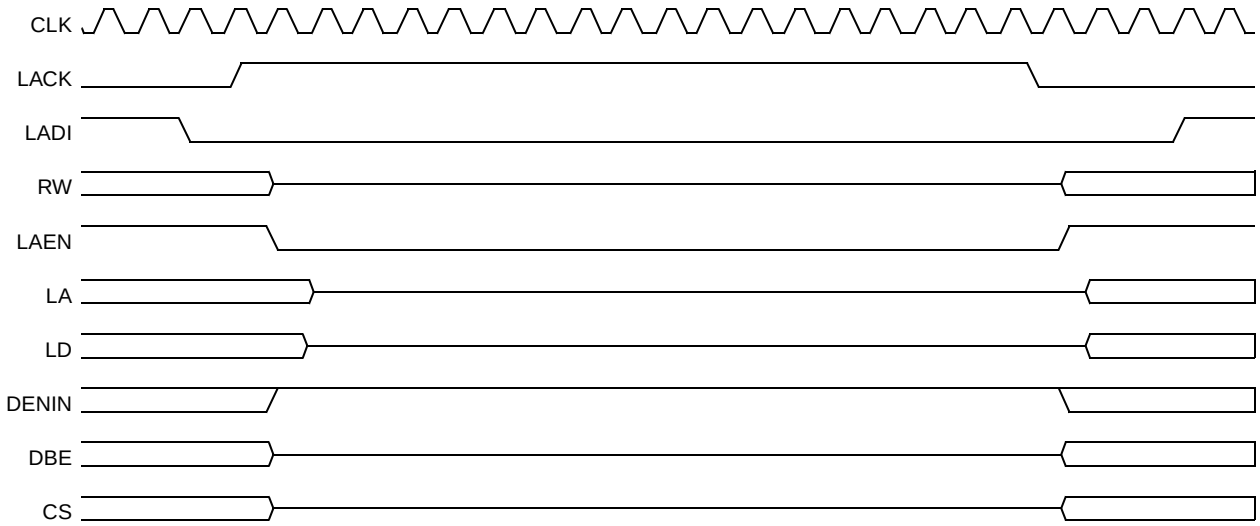


Figure 3-20. Local Bus Holdoff

Table 3-4. AM Code Decode

Address Size	Modifier	VMEbus Transactions
A64		A64 block transfer (BLT); A64 single cycle access; A64 64-bit block transfer (MBLT); A64 lock command (LCK); A64 serial access
A40		A40 block transfer (BLT); A40 single cycle access; A40 lock command (LCK); A40 serial access
A32		A32 supervisory block transfer (BLT); A32 supervisory program access; A32 supervisory data access; A32 supervisory 64-bit block transfer (MBLT); A32 non-privileged block transfer (BLT); A32 non-privileged program access; A32 non-privileged data access; A32 non-privileged 64-bit block transfer (MBLT); A32 lock command (LCK); A32 serial access
A24		A24 supervisory block transfer (BLT); A24 supervisory program access; A24 supervisory data access; A24 supervisory 64-bit block transfer (MBLT); A24 non-privileged block transfer (BLT); A24 non-privileged program access; A24 non-privileged data access; A24 non-privileged 64-bit block transfer (MBLT); A24 lock command (LCK); A24 serial access
A16		A16 supervisory access; A16 non privileged access; A16 lock command (LCK); A16 serial access
	SUPER	Supervisory access
	NPRIV	Non-Privileged access
	PROG	Program access
	BLT	Block transfer (inc. MBLT)
	LONG	32 or 64 bit access (not 8 or 16)
	LOCK	A64 lock command (LCK), A40 lock command (LCK), A32 lock command (LCK), A24 lock command (LCK), A16 lock command (LCK)
	SERIAL	A64 serial, A40 serial, A32 serial, A24 serial, A16 serial
	CR/CSR	Configuration ROM/Control and Status Register
	U1, U2	User Defined (U1 includes AM Codes 18–1F, U2 includes 10–17)

It should be noted that the parameter LONG is derived from a combination of DSi*, LWORD*, and AM Code. For more information on AM Code definitions, it is recommended that the VMEbus Specification be consulted.

The reason that the CY7C960 decodes the AM Code this way is to allow the user to enable VMEbus transactions selectively by AM Code classification. For example, a transaction with AM Code “0C” would result in the following settings: A32, SUPER, BLT, LONG. The CY7C960 would react to the transaction only if all four categories were enabled for the addressed Region. The CY7C960 consults a look-up table, configured during initialization, to determine whether the transaction is enabled or not. *Table 3-5* is an example of how this table could be programmed.

Table 3-5. Major AM-to-Region Mapping Example

REGION	A64	A40	A32	A24	A16	SUPER	NPRIV	PROG	DATA	BLT	LONG
0	✓					✓				✓	✓
1		✓				✓				✓	✓
2			✓			✓		✓			✓
3				✓			✓		✓		✓
4				✓	✓	✓		✓			
5			✓			✓	✓	✓	✓	✓	✓
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											

If the CY7C960 is configured according the table above, then it will respond as slave under the following conditions.

If the REGION inputs are “0000”, then the CY7C960 will respond to A64 block transfers of type BLT or MBLT.

If the REGION inputs are “0001”, then the CY7C960 will respond to A40 block transfers of type D8:BLT, D16:BLT, or MD32:BLT.

If the REGION inputs are “0010”, then the CY7C960 will respond to A32 supervisory program accesses of type D8, D16, or D32 (single cycle).

If the REGION inputs are “0011”, then the CY7C960 will respond to A24 non-privileged data accesses of type D8, D16, or D32 (single cycle).

If the REGION inputs are “0100”, then the CY7C960 will respond to A24 supervisory program, and A16 supervisory accesses of type D8 or D16 (single cycle). Note that D32s will not be responded to in this region because LONG is not set.

If the REGION inputs are “0101”, then the CY7C960 will respond to any A32 AM Code other than A32 LOCK and A32 SERIAL which must be enabled separately (see below).

If the REGION inputs are any other value, then the CY7C960 will assume that the cycle is intended for another board and will not respond.

LOCK, SERIAL, USER, and CR/CSR AM codes are treated somewhat differently from the “mainstream” codes. Each of these codes can be assigned to only a single specific REGION. Since these transactions are typically mapped to a single address space on a board, it was not necessary to include them in all 16 rows of the main REGION/AM decode lookup table. The user simply selects which, if any, of these transactions should be decoded and assigns a valid REGION pattern to them.

Table 3-6 shows an example: A32 LOCK transactions are enabled in Region 0; A40 and A16 SERIAL transactions are enabled in Region 8; User AM codes in group 1 are enabled in Region 10; the remaining two classifications are disabled (but had they been enabled they would have been enabled for Region 2).

Table 3-6. Minor AM-to-Region Mapping Example

AM Classification	A64	A40	A32	A24	A16	Enable	Region
LOCK			✓			✓	0
SERIAL		✓			✓	✓	8
USER1 (AM Codes 18 - 1F)	These do not have an associated address size					✓	10
USER2 (AM Codes 10 - 17)							2
CR/CSR							2

If the VMEbus transaction is determined by the above process to be enabled for the Region being addressed, then the CY7C960 internal flag DECODE is set, allowing local activity to commence. *Figure 3-21* shows a flowchart representation of the process of decoding a VME-bus Transaction.

A slave board that uses the CY7C960 can be programmed to respond in a wide variety of ways with a fine control on decode granularity by using the CY7C960’s Region Mapping features. The following sections and the AC timing diagrams at the end of this document are intended for users who need to know further details of the features described above.

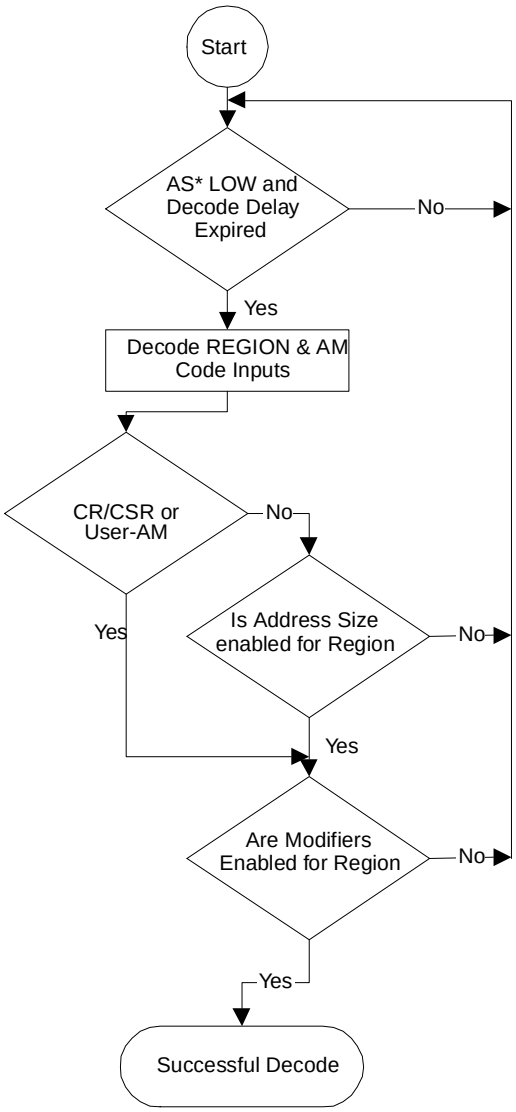


Figure 3-21. Flowchart for Decoding VMEbus Transactions

3.5.5 Decode Delay Timing

Slave decode delay timing is a process that conceptually begins whenever AS* is sampled low on the rising edge of CLK. However, at this time the CY7C960 slave decoder must wait for all local activity to clear before processing a VMEbus cycle. This is because it is possible for a local cycle to be in progress without a Master on the VMEbus.

For example, in the case of a posted slave write, the VMEbus Master is allowed to leave the bus with data still to be processed in the interface. If this is the case, then the local address bus will be holding an address from this posted write. Likewise, there are situations where the CY7C960 will perform a read-ahead cycle. This happens when block transfer reads do not terminate on a 256-byte boundary. The on-board slave decoder must wait for this address

to clear and allow enough time for the new address to arrive and become stable on the local bus.

The internal AM decoder then signals whether the first cycle will be an address broadcast cycle. Since MBLT and MD32 transactions use the address bus for data transfers, a Master initiating these transactions will first issue an address broadcast so the Slave can capture the starting address before the address bus changes into a data transfer bus. This address broadcast cycle also occurs for transactions that use the VME data bus as an address bus. These cycles include all A64 and A40 transactions since the width of the VME address bus is only 32 bits (24 bits for 3U boards).

If there will be no address broadcast cycle, then the decode delay timeout begins with AS*. An internal timer counts up to the programmed decode delay. If an address broadcast cycle is signalled, then the CY7C960 will wait for the DSB* event and then count up to the programmed decode delay.

In either case, after the decode delay has expired, the CY7C960 will determine if the addressing information for this cycle falls with its programmed address space. If it does, then an internal flag signals other internal circuitry that the cycle is a valid slave decode for the board. If it does not, then the internal slave decoder will wait until AS* is sampled High and then return to an idling state waiting for the next VMEbus cycle.

3.5.6 Slave Addressing Before Initialization

The CY7C960 can be programmed in one of three ways. The interface can be programmed from the VMEbus, the local bus, or a combination of the VME and local buses. If the CY7C960 is programmed using the VMEbus or the combination method, slave address information and device programming is loaded using a methodology compliant with the VMEbus Auto Slot ID specification using either a CR/CSR or A16 AM code. The configuration ROM/control and status register (CR/CSR) capability described by the latest revision of the VMEbus specification provide a mechanism for board identification and automatic initialization. Auto Slot ID is an optional method of assigning the CR/CSR base address to each VMEbus board. This Auto ID slave uses a level 2, D8 interrupter along with additional board specific hardware to obtain the CR/CSR base address. An Auto ID Master, called the Monarch, uses a level 2, D8 Interrupt Handler to acknowledge the Auto ID interrupt and assigns a base address to the slave board. Specifically, this Auto ID slave:

1. generates a level 2 interrupt if PDATA is sampled Low after power-on or a negative edge on SYSRESET*. (IRQ* must be connected to IRQ[2]* on the VMEbus.)
2. waits for and responds to the level 2 IACK cycle initiated by the Monarch.
3. asserts LDEN* thereby enabling a Status/ID byte that is sourced by external local hardware.
4. presents the Status/ID byte on the VMEbus. (all 32 bits of VMEbus data are driven)

5. drives DTACK* Low and releases IRQ*.

The Monarch must then:

1. write the base address of this board using a single cycle write with a CR/CSR or A16 AM code. If a CR/CSR AM code is used, then all subsequent programming cycles must use a CR/CSR AM code. Likewise, if an A16 AM code is used, then all subsequent programming cycles must use an A16 AM code. It is mandatory that this access be indivisible with the previous IACK cycle on the backplane to avoid collisions with
2. program the CY7C960 internal registers using a series of single-cycle writes to the base address of the slave if the VMEbus Initialization Method is used. These accesses require a bit pattern of ‘xx00’ on REGION[2:0] if a CR/CSR AM code is used and ‘xxx0’ if an A16 AM code is used.
3. initialize the CY7C964 compare and mask registers using two single writes to the base address of the slave. These two writes will reassign the CR/CSR base address of the slave to its new base address in the system.

3.5.7 Address and Data Strobe Event Processing

Another process begins in parallel with slave decoding. The following sequence of events occurs whenever AS* is sampled Low by the CY7C960:

- The internal AM decoder signals whether the first cycle will be an address broadcast cycle. Since MBLT and MD32 transactions use the address bus for data transfers, a Master initiating these transactions will first issue an address broadcast cycle so the Slave can capture the starting address before the address bus changes into a data transfer bus. This address broadcast cycle also occurs for transactions that use the VME data bus as an address bus. These cycles include all A64 and A40 transactions since the width of the VME address bus is only 32 bits.
- The value of LWORD* is latched and held for the duration of the transaction. LWORD* on the VMEbus must be connected to A[0] on the least significant CY7C964 because LWORD* is captured by the CY7C960 on LA[0]. This does not present a problem because A[0] on the VMEbus is encoded on the VMEbus data strobes.
- The local address that has been flowing through the CY7C964s will now be latched. An important consideration to note here is that the transaction may be intended for another slave on the backplane and that slave may respond very quickly on its DTACK* or BERR* lines. Since the Master is permitted to remove the address after the responding Slave drives DTACK* or BERR* Low, the CY7C960 will signal the CY7C964 to latch its local address lines at this time in order to capture the addressing information before it can be removed. This is accomplished by the assertion of LADI from the CY7C960 to the CY7C964.

The CY7C960 then waits for the DSB* and DECODE events. The DSB* event is defined by the VMEbus specification, which states that maximum skew between the assertions DS1* and DS0* Low by a Master shall be no more than 20 ns at the Slave. When the CY7C960 samples either DS1* or DS0* Low by CLK, it waits two clocks and then sets an internal flag called DSB*. This two clock waiting period is designed to accommodate any data strobe skew within specification by ensuring that both strobes have had time to arrive before the device decodes addressing information from the strobes.

The DECODE event is another internal flag that is set ‘true’ or ‘false’ depending on whether the CY7C960 has been selected. Refer to the previous sections on slave address decoding for a description of this process.

Several things happen when these two events occur. Data size information is decoded from the values on DS1*, DS0*, LA[1], and LWORD*. This information is used to determine local byte enable patterns (DBE[3:0]), data byte swapping control (SWDEN*), and local address counting. The CY7C960 also disables the drivers on the least significant CY7C964 and begins to drive the starting address for the transaction on LA[7:0].

If either of these two events fail to occur, then the CY7C960 will wait until it samples AS* High, deassert LADI thereby letting VME address flow through to local address on the CY7C964’s, and return to an idling state waiting for the next AS* Low event.

3.5.8 Slave Data Transfer Acknowledgment

Slave data transfers are acknowledged by a Low assertion of the DTACK* output. All assertions of DTACK* require the assertion of the internal DECODE and DSB* flags. If these two conditions are present, then DTACK* will be asserted Low on the backplane when

- the internal AM decoder signals that an address broadcast cycle is in progress. Address broadcast cycles do not generate local chip selects.
- the cycle is a slave read of any type. The CY7C960 will assert DTACK* Low when its local controller sets an internal flag that indicates data has been read from the local board and the data is being held in the interface (CY7C964s).
- the cycle is a slave block transfer write of any type. This is the case where data is posted in the interface and the VMEbus Master is allowed to start the next cycle or terminate the transaction. While the Master is starting the next cycle or terminating, the CY7C960 local controller writes the posted data to the local board. Slave write posting of block transfer cycles allow the CY7C960 to transfer data at a maximum rate of 80 Mbytes/s for MBLT bursts and 40 Mbytes/s for BLT bursts.
- the cycle is a single-cycle slave write of any type and the CY7C960 local controller has finished writing the data to the local board. This is the case where data is not posted in the interface and the VMEbus Master must wait for the write transfer to complete before starting the next cycle or terminating.

3.5.9 Slave Write Posting

Slave write posting is a means of obtaining maximum data transfer performance by decoupling the VMEbus from the local board. Every slave write data transaction is latched in the CY7C964s for further processing by the CY7C960 local controller and the VMEbus Master is allowed to begin its next cycle or terminate the transaction.

In other words, the VMEbus Master drives DSi*, and the CY7C960 drives DTACK* before the data has reached the local bus.

Data is latched in the interface when DSB* occurs during a slave write to a valid address region. At this time the CY7C960's local controller initiates a write cycle to the local board. This pipelined architecture allows fast VMEbus cycles that are not delayed by local affects such as DRAM refresh bursts, or resource contention. When the local cycle is complete, the data latches are opened and new data is allowed to flow into the interface. Also, LA[7:0] is incremented according to the data width of the transaction after the local board acknowledges that the posted data has been written.

3.5.10 Slave Read-Ahead Cycles

Slave read-ahead cycles are also used to achieve maximum data transfer rates for block transfer reads. When the CY7C960 completes a read cycle to a VMEbus Master during a block transfer operation, it will read ahead on the local bus in order to have data ready for the next cycle of the burst.

Data is read from the local board, latched in the interface, and driven on the VMEbus data lines when DSB* arrives to start a slave block transfer read from a valid address region. Note that all 32 bits of data are driven on the VMEbus data lines, regardless of data size. The CY7C960 asserts DTACK* Low to signal that the data is ready to be taken. When the VMEbus Master releases the data strobes, the CY7C960 stops driving the VMEbus data lines and LA[7:0] are incremented according to the data width of the transaction. The CY7C960 then performs another local read in preparation for the next cycle of the read burst.

Since there is no indication to the slave as to how long a block transfer will be or when it will terminate, it is possible to have a read-ahead cycle occur that does not get transferred to the Master. In other words, the Master could terminate before, during, or after the read-ahead and the local data that was read would not be transferred to the VMEbus. However, the CY7C960 will not read ahead at a 256 byte boundary even if 2k byte boundaries are in effect as is the case for some MBLT transactions. The read circuitry was implemented this way in order to provide a means of averting read-ahead cycles and is possible because the CY7C960 drives LA[7:0], which enables it to detect when the address is about to overflow.

Also, as in the case of MBLT and MD32 transactions, there can be two local accesses to be processed for one VMEbus access. The CY7C960 handles all the pipelining and multiplexing

control that must take place in the interface to accommodate reads or writes when the VME data bus is wider than the local data bus.

3.5.11 Interrupt Cycle Support

There are six pins associated with interrupt requesting on the VMEbus: IRQ*, IACK*, IACKIN*, IACKOUT*, LDEN*, and LIRQ*. There are two ways an interrupt can be signalled on the VMEbus backplane: a local interrupt request or a reset.

A power-on, system reset, or local reset condition causes the device to implement its initialization protocol. Refer to sections 3.4.3 and 3.4.5 for details on initialization.

LIRQ* is the input to the device from local circuitry demanding an interrupt cycle. If the LIRQ* signal is asserted Low, the CY7C960's IRQ* pin is asserted Low. The local circuitry should remove LIRQ* after the interrupt has been acknowledged: The CY7C960 simply drives the state of LIRQ* onto IRQ*. *Figure 3-22* illustrates the relationship between LIRQ* and IRQ*. s60 represents the assert and deassert delays.

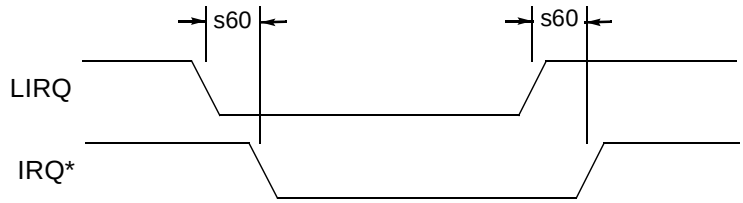


Figure 3-22. Interrupt Signal Timing

The CY7C960 actively drives the value of IACKIN* to IACKOUT* in conformance with the rules for VMEbus D8 Interrupt Acknowledge Cycles. If it is driving IRQ* and IACK* is sampled Low and IACKIN* is sampled Low and AS* is sampled Low, then LA[3:1] is compared with the programmed interrupt request level. If they are not equal then the Low IACKIN* is driven to IACKOUT*. If they are equal, then IACKOUT* remains High and LDEN* is driven Low to enable an external Status/ID vector onto the local data bus. This vector is in turn enabled onto the backplane and the VMEbus Interrupt Handler is acknowledged by a Low transition on the DTACK* line.

It is important to note that, while the CY7C960 is a D8 interrupter, it drives all 32 bits of VMEbus data on the backplane with the assertion of DENO* to the CY7C964s. If the user wishes to provide only 8 bits of status ID, then the value of the upper bytes should be set to FF hex to ensure full compliance with the VMEbus specification. If the user wishes to provide 16 or 32 bits of status ID, then the interrupt handler should be programmed to handle the provided width. The CY7C960 does not attempt to differentiate between 8-, 16-, or 32-bit interrupt acknowledge cycles.

3.5.12 Interrupt Handshake Support

The CY7C960 can be programmed at initialization to wait for a local acknowledge signal before terminating an IACK cycle. LACK is the local signal used to suspend completion of an IACK cycle until the local interrupter is ready to present a STATUS/ID vector to the VMEbus Interrupt Handler.

If the Handshake bit is set the CY7C960 will sample the state of the LACK pin three clocks after it samples a Low level on its IACKIN* pin during a VMEbus IACK cycle. If LACK is High at this time, then the CY7C960 will wait until LACK is Low before terminating the cycle with a falling edge on DTACK*.

If the Handshake bit is not set, the CY7C960 will ignore the state of the LACK pin and DTACK* the VMEbus Interrupt Handler four clocks after sampling a valid IACKIN*.

Handshake support is available immediately after local completion of the Combination Initialization Method or after full completion of the VMEbus or Local Initialization Methods. Refer to Chapter 3.4 for details on the three programming methods.

3.6
CY7C964 Interface

3.6.1 CY7C964 Overview

The CY7C960 is designed for use with the CY7C964 VMEbus Interface Logic Circuit. This device is fully described in Section 4, The CY7C964 Bus Interface Logic Circuit. The CY7C960 provides all the control and timing for the interface with the CY7C964. Interface timing as generated by the CY7C960 is designed for CY7C964 delay characteristics.

The CY7C964 contains a collection of counters, latches, and multiplexers used to facilitate data handling during any of the many VMEbus data transactions. Three-state high drive buffers allow direct connection to VME and local address and data busses. Additionally, the CY7C964 contains an address comparator with mask function to allow the CY7C964 to form part of the user's slave address decoder. Defined as a companion to the VIC068A/VIC64 devices, the CY7C964 has been designed into many high-performance VME64 CPU boards. In concert with the CY7C960, it performs the functions of address decoder, D64 data multiplexer, and local address counter, to facilitate the design of high-performance slave systems.

3.6.2 CY7C964 Connections

The signals that are provided by the CY7C960 for use by the CY7C964 are: LADI, LEDI, LEDO, DENO*, ABEN*, DENIN*, DENIN1*, LAEN, STROBE, LDS, and D64 using dedicated pins. These signals are described in the pin description section.

LADI (Latch ADdress In) controls a transparent latch which connects the VMEbus address to the local address. The default state of the signal ensures that the VMEbus address is driven to the local pins, allowing the implementation of VMEbus address decoder circuitry on the local address bus. The LAEN (Local Address ENable) input is tied High on all but the least significant CY7C964 to ensure that the local address is always enabled. The least significant CY7C964 has its LAEN pin controlled by the CY7C960 to allow the CY7C960 to source the LSB of address during block transfers when the address has to increment by 1, 2, or 4 (dependent on the transfer type: single, double, or quad byte). Note that in order to take advantage of the "local bus holdoff" feature of the CY7C960, it is necessary to control LAEN of the upper 3 CY7C964 devices. LAEN must be driven Low during holdoff.

LEDI (Latch Enable Data In) and LEDO (Latch Enable Data Out) control the VMEbus and local data transfer latches in the CY7C964. Data read from local resources must be set up to LEDO for burst transfers, while LEDI provides data hold time during burst transfers when data is captured from the CY7C964. These signals, by holding data in the interface allow "write posting" and "read-ahead" performance features.

DENO* (VMEbus Data ENable Out) and the pair of signals, DENIN* and DENIN1* (local Data ENable IN) are normally used for local and VMEbus data enable. DENIN* and DENIN1* are also active during address broadcast of A64 and A40 VMEbus transactions. They are used to inform an external address decoder when the data bus can be sampled for the high-order address information.

During D64 and MD32 data cycles, the function of DENO* and DENIN*/DENIN1* are identical to LEDO and LEDI respectively. A two-stage pipelined latch inside the CY7C964 is controlled in the data sourcing case by LEDO and DENO* working in concert, and in the data capture case by LEDI and DENIN*/DENIN1* working in concert.

Each VMEbus data cycle is served by two local data cycles. The second cycle for slave read must set data up to the falling edge of DENO*. For slave write, data hold on the second cycle is provided by the assertion of DENIN* (D64) or DENIN1* (MD32).

LDS is used during D64/MD32 slave write transfers to control the VMEbus address to local data demultiplexing, and during the initialization period to control the selection of the address mask and compare registers. During multiplexed slave data capture, LDS is the signal which controls data hold time with respect to the deassertion of CAS* and/or DBE.

ABEN* (VME Address Bus ENable) is active only during multiplexed data transactions to enable the VMEbus address pins during D64 or MD32 transfers.

D64 is the signal that informs the CY7C964s that the data phase of multiplexed data transfer is in progress and the on-chip pipelining should be used to multiplex or demultiplex the local data. The CY7C964 is expressly designed to demultiplex D64 and MD32 VME transactions to a non-multiplexed 32 bit local data bus.

STROBE is a timing signal used to load the on-chip address mask and compare registers of the CY7C964. It can be considered a latch enable signal which latches local data when High.

The on-chip counters of the CY7C964 are used for local address counting during slave D64 block transfers if the VMEbus master bursts to 2 Kbyte boundaries. Therefore the counter chains must be enabled by connecting the appropriate carry-out to carry-in pins. The VMEbus counters of the CY7C964 are not used by the CY7C960.

It is recommended that the following connections be made to ensure correct operation during the Reset period: Pull-up resistors attached to ABEN*, DENO*; pull-down resistors attached to LAEN and LADI. These connections prevent CY7C964 drivers from turning on during the brief periods of three-state operation during reset.

The CY7C964 has several signals which are not used for slave operations: BLT*, FC1, LADO, and MWB*. LADO and FC1 must be tied Low. BLT* and MWB* must be tied High for correct operation.

Figure 3-23 shows how the control signals for the CY7C964 are driven during initialization to load the Address Compare and Mask Registers. The significant signals are LDS, which selects

which register to load, and STROBE, which actually strobes the data on the local bus into the appropriate register.

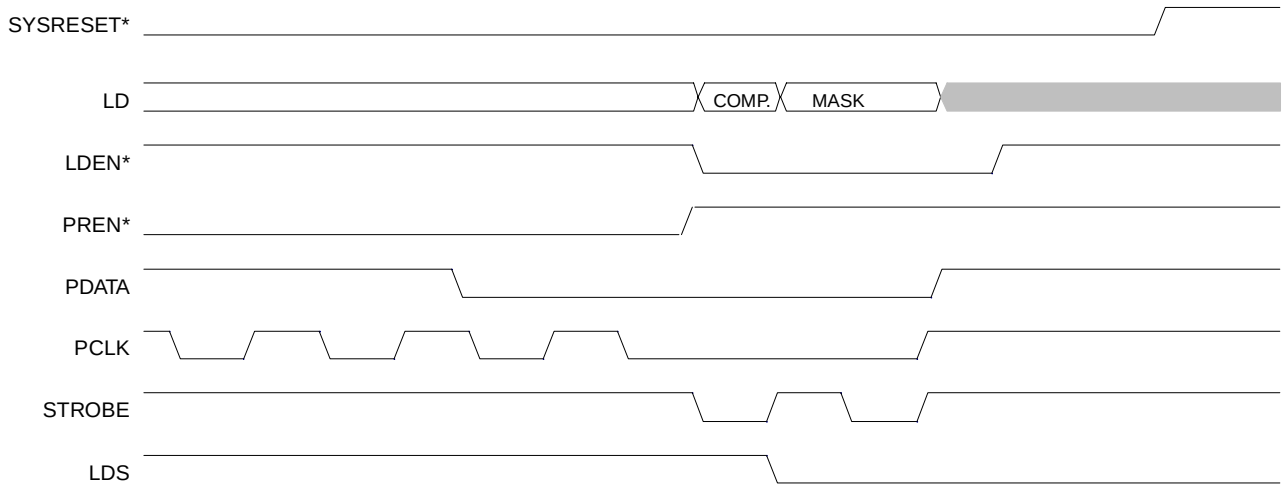


Figure 3-23. Example of CY7C964 Control Timing

3.6.3 Swap Buffer Control

The CY7C960 is designed to control the swap buffering necessary for handling D8 and D16 VMEbus transactions in combination with 32-bit local memory. *Figure 3-24* illustrates how VMEbus byte lanes relate to local memory. Note that the CY7C960 coordinates control of SWDEN* with DENIN*/DENIN1*. The CY7C964 Byte(0) and Byte(1) drivers are turned off during slave write transfers when SWDEN* is active. Because DENIN*/DENIN1* are used as latch control signals during multiplexed data transactions (D64 and MD32) DENIN* signaling to CY7C964 Byte(0) and Byte(1) must be modified by external logic IF (and only IF) and board is to handle BOTH D64 and A40/MD32 slave write transactions. The required circuitry is shown at the top of the figure below.

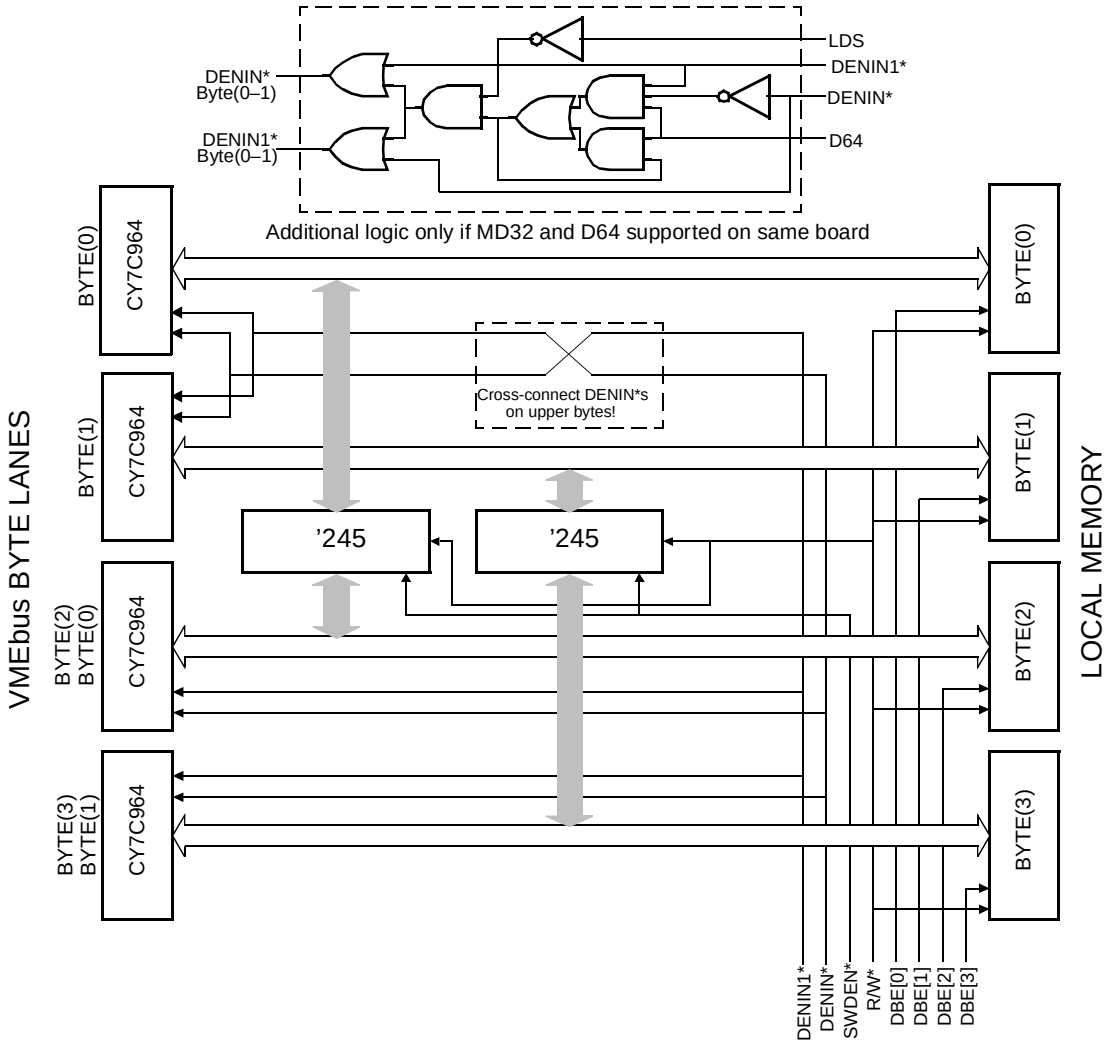


Figure 3-24. Swap Buffer Design

3.7 Interfacing without CY7C964

3.7.1 Reduced Cost, Fewer Features

Many board designs need only simple VMEbus slave interfaces. The CY7C960 used in combination with Cypress FCT Logic family devices meets this need perfectly. A complete interface can be built using the CY7C960 in combination with industry standard latches, transceivers, and latched transceivers.

This interface supports Rev C VMEbus: A16, A24, A32 D8, D16, D32, D32UAT, RMW, in other words, a complete interface excluding the performance and multiplexed address features of VME64.

Because the CY7C964 was designed as an integrated replacement for a pair of bidirectional transceivers controlling address and data in the VMEbus interface, the number and sense of signals controlling transceiver function have been preserved in the design of the CY7C960. This makes direct connection and control of discrete devices straightforward.

Figure 3-25 illustrates just such an implementation. One pair of CY74FCT16543T devices handles the bidirectional data path while a pair of CY74FCT162373T devices provides for latching and driving 32 bits of address from the VMEbus to the local side of the interface. A swap buffer, implemented with a CY74FCT162245T device is optional depending on the VMEbus transactions the slave interface is intended to accept.

The pull-up resistor shown is required to program the polarity of the LAEN output of the CY7C960. LAEN is three-stated at power-on or warm reset. The logic level sensed on LAEN during the first initialization cycle determines the deasserted state of LAEN during normal operation. In this case, the deasserted state is required to be High.

The FCT543 devices are rated 64 mA sink current and 32 mA source current over commercial temperature range. The FCT245 and FCT373 devices have balanced 24 mA output drive.

In *Figure 3-25* all components are drawn to relative scale assuming the CY7C960 in a 64-pin thin quad flat pack. The FCT Logic devices are depicted in thin shrunk small outline packages. The total footprint area of the interface as shown is only 1.08 sq. in. for the 6 chips. This compares to the slightly larger CY7C960/CY7C964 solution requiring 1.16 sq. in. using 5 identical packages. The CY7C961/CY7C964 chip set occupies 1.34 sq. in. All of these interface sets have thin package profiles suitable for mounting on the back side of a VMEbus board without violating mechanical clearance requirements.

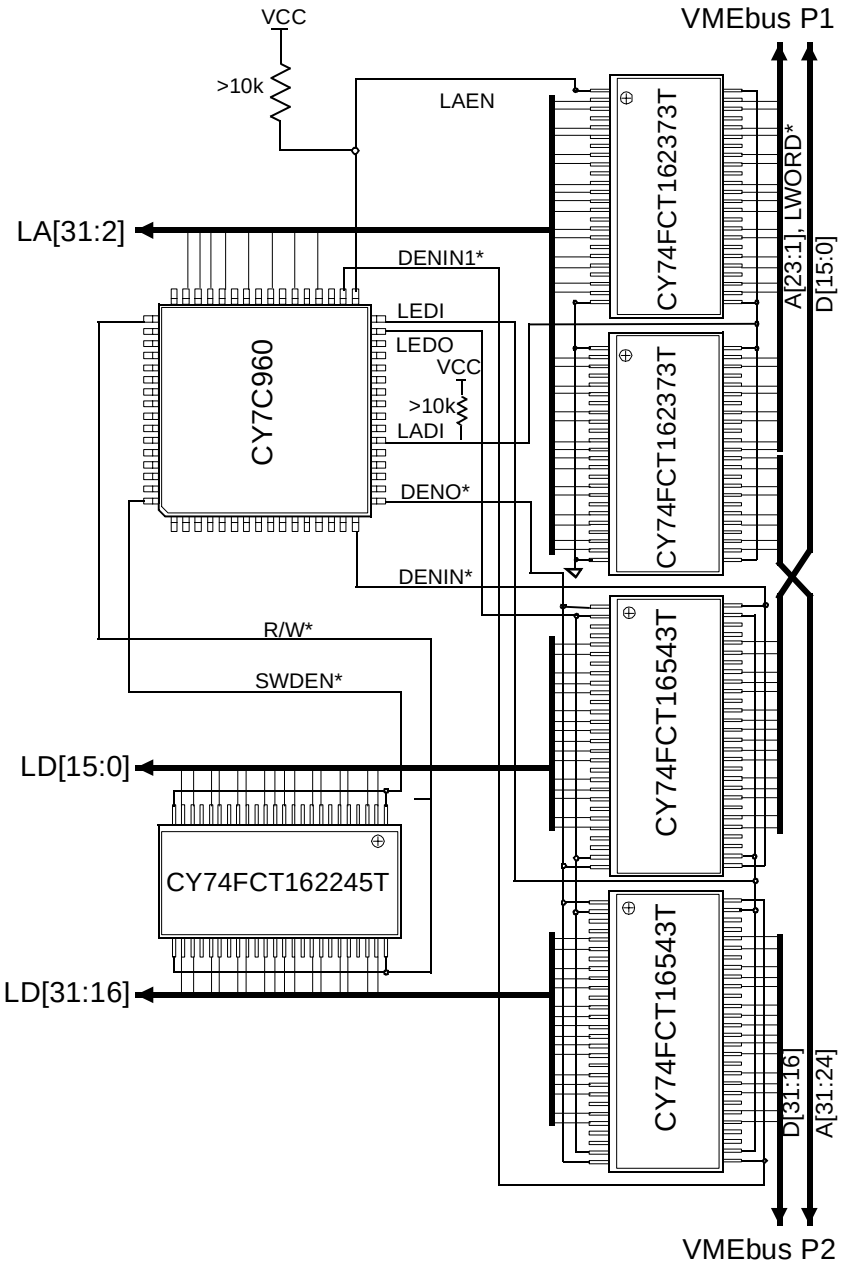


Figure 3-25. VMEbus Interface Implemented with CY7C960 and FCT Logic Family.

3.8
DRAM Control Description

3.8.1 Overview

The CY7C960 contains a high-performance DRAM controller, developed to take advantage of DRAMs operating in Fast Page Mode. By using Fast Page Mode devices, VMEbus MBLT transfer rates of 80 Mbyte per second are attainable with DRAM RAS*-access times of 70 ns. With appropriate buffering on the DRAM address and control signals, 16 Mbytes of DRAM can be directly accommodated (using 4-Mbit devices). With minimal external decoding logic generating multiple RAS*/CAS* signals, larger memory arrays can be controlled. When higher capacity DRAM devices become readily available, the memory arrays increase accordingly.

The CY7C960 is configured by the user to work with local DRAM during the initialization sequence. This configuration determines the function of several pins (RAS*, CAS*, ROW, COL, and REGION3/CS2). RAS* and CAS* provide the well-known address strobe signals for the DRAM array, ROW and COL provide address enable signals for strobing the row and column addresses into the DRAM array. If DRAM mode is selected during the configuration sequence, then the REGION3/CS2 pin becomes the CS2 output: 8 regions are available in DRAM mode requiring only three REGION decode inputs.

Figure 3-26 shows an example of the use of the DRAM control signals. ROW enables the upper bits of the local address bus to be latched in the DRAM by RAS*. COL enables the lower address bits to be latched by the use of the Data Byte Enable signals. In the example chosen, the DBE signals are connected to the CAS* inputs of the DRAMs, and the CY7C960 has been programmed to provide refresh timing on the DBE pins (DBE Refresh Enabled)

The LACK input may be used to acknowledge DRAM cycles. If LACK is tied Low, then the cycle is self timed using the programmed values for CAS* assert and CAS* precharge. The user may prevent DRAM CAS* bursts by the use of the LACK signal. This is done by maintaining LACK High until the DRAM CAS* burst is to be permitted. Once LACK is driven Low and CAS* is asserted by the CY7C960, the DRAM access is no longer affected by LACK until RAS* deasserts, signaling the end of the current access (useful in dual-port applications to hold off VMEbus accesses to DRAM).

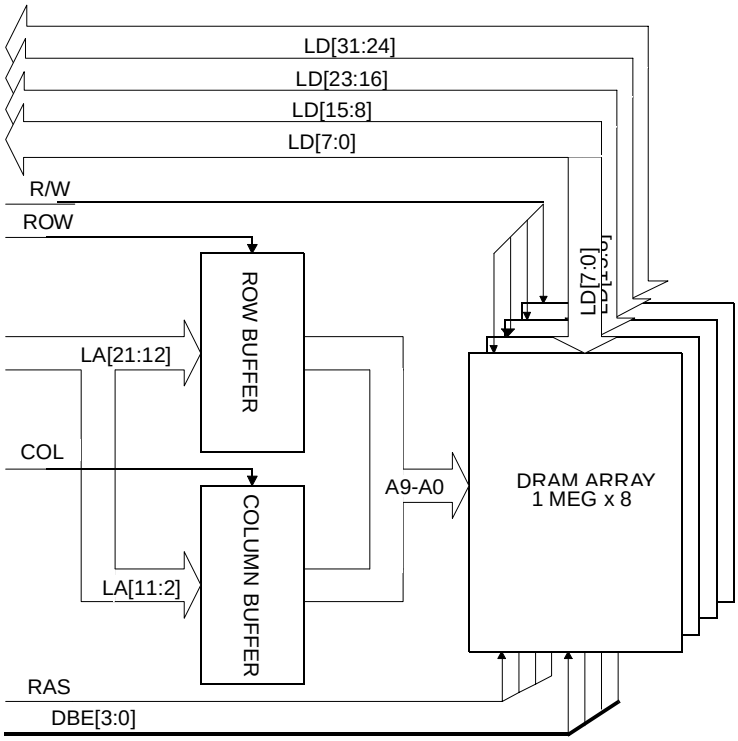


Figure 3-26. DRAM Signal Usage

3.8.2 Types of DRAM

The CY7C960 controls Fast Page Mode DRAMs. This is an industry standard mode of operation. For 80 Mbyte/second burst transfer rate, 70-ns (RAS*-access) devices, or faster, should be used. The CY7C960 can be configured to operate with a wide range of DRAM timings. For example, if an 80-MHz clock is applied to the device the range of timings is given below. If a lower frequency is used the times scale as shown in the AC Timings Specification (see Chapter 3.12).

The access time of a page-mode DRAM is determined by two factors: the time from application of RAS* to the data becoming available, and the time from application of CAS*. The fastest method for getting bursts of data out of or into memory is to maintain RAS* Low, and cycle CAS*. The access time for the first cycle is determined mainly by the RAS* access time, but subsequent cycles are shorter, determined by the CAS* access time. The amount of data that can be transferred in one burst is bounded by the size of the DRAM's row. When a row boundary is encountered it is necessary to reassert RAS, and hence another long cycle occurs.

3.8.3 VMEbus Implications

VMEbus transfers match DRAM Page Mode Accesses quite closely. When AS* is asserted the first cycle is generally longer than subsequent cycles, especially if the transaction involves an address broadcast cycle. For optimizing the slave's performance, RAS* is driven whenever AS* goes LOW. This starts the page mode cycle to the DRAMs. If the VMEbus transaction is not intended for the CY7C960, then RAS* is deasserted, and no DRAM cycle occurs. If the transaction is directed to the CY7C960, and the cycle type is enabled for the Region being addressed, then a DRAM cycle occurs. The programmable delays for DRAM cycle timing are then used as the cycle progresses. The first cycle therefore uses the RAS*/CAS* Delay parameter to ensure that the RAS* access time for the DRAM is met, and CAS* assert and precharge time parameters ensure that the shorter CAS* access time is used for subsequent cycles.

When a VMEbus boundary is encountered (256 byte for BLT, 2048 byte for MBLT), the address is rebroadcast by the VMEbus master. This causes AS* to cycle, which in turn causes RAS* to cycle. This ensures that no problems occur due to overflowing a page boundary in the DRAM, as the local address is aligned to the VME address. Put another way, the maximum number of transfers that can occur between RAS* cycles is 256. Even the smallest page mode DRAMs have page lengths that are a multiple of this number, so a page boundary in the DRAM is guaranteed to correspond to an address boundary on the VMEbus. Therefore RAS* is guaranteed to cycle at DRAM page boundaries.

3.8.4 Refresh Cycles

The CY7C960 contains a refresh controller. The refresh period and function enable are set during device configuration. The controller can be enabled to generate CAS* before RAS* Refresh cycles at preprogrammed intervals. The controller employs a hidden refresh strategy and is fully integrated into the response of the CY7C960 local interface. Each time a refresh interval expires, a burst of 4 CAS* before RAS* refresh cycles is scheduled. At the next opportunity a refresh burst will be generated. During DRAM accesses, refresh is allowed at double longword address boundaries. Refresh is blocked during burst accesses to I/O regions if the DBE refresh enable feature is enabled. Refresh is always allowed between slave accesses and does not interfere with transaction decoding. If refresh cannot occur before additional intervals expire, a proportional number of additional CAS* before RAS* cycles will be scheduled. Overflow and consequent loss of refresh control timing occurs if 256 intervals elapse without a refresh burst.

3.8.5 Refresh Timing

The CY7C960 DRAM control can be programmed by specifying 4 parameters which are part of the initialization bit stream. These 4 parameters are RAS*/CAS* Delay Time, CAS* Assert

Time, CAS* Precharge Time, and RAS* Precharge Time. Each is a 3-bit eight valued field. *Table 3-7* relates the programmed values for three of the parameters to the timing diagram of *Figure 3-27*. CAS* Precharge Time (not shown) is used to adjust the minimum deassertion time between CAS* assertions during CAS* bursting. The DRAM timing is designed to support 60-ns fast page mode DRAM.

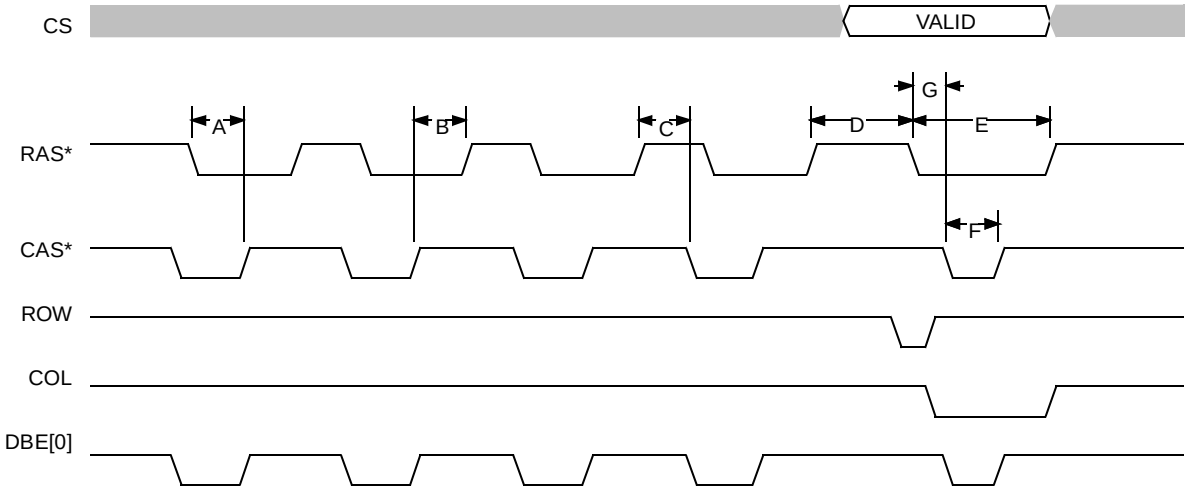


Figure 3-27. Programming DRAM Timing

Table 3-7. DRAM Timing

A	CAS* Assert Time – 1	2T – 9T
B	RAS* Precharge Time – 3	2T – 9T
C	RAS* Precharge Time – 3	2T – 9T
D	RAS* Precharge Time	5T – 12T
E	RAS* Precharge Time	5T – 12T
F	CAS* Assert Time	3T – 10T
G	RAS*/CAS* Delay Time	2T – 9T

3.8.6 DBE Refresh Enable Feature

Refresh control behavior is fundamentally affected by the DBE refresh enable feature of the CY7C960. If the feature is off, refresh is mutually exclusive with DRAM accesses, but is unrestricted when a DRAM access is not in progress. During I/O accesses, CAS* before RAS* refresh bursts can occur at any time with respect to access to I/O regions. R/W*, DBE, and CS* outputs are oblivious to activity on RAS* and CAS*. DRAM RAS* and CAS* signals can be connected directly to the CY7C960, while DRAM OE and WE signals must be gated with the CY7C960 COL signal to provide the distinction between DRAM access and DRAM refresh. The COL signal is inactive except during CAS* cycles of DRAM access.

If the DBE refresh enable feature is turned on, refresh will be mutually exclusive with DRAM accesses and I/O accesses. This is a necessary consequence of providing refresh CAS* behavior on the DBE signals. With the feature enabled, CS is deasserted and R/W* driven High during refresh bursts. This mode allows the use of CS signals to explicitly select DRAM and the use of DBE signals as byte addressable CAS signals. DRAM OE and WE can be gated by CY7C960 CS signaling. Where possible, the refresh occurs in parallel with user transactions such as I/O transactions with no DRAM enabled.

Figure 3-27 shows a refresh burst delaying a DRAM access with DBE Refresh Enable turned on. Note that R/W* (not shown) is High, and ROW and COL are inactive during the refresh burst. The four DBE signals (only one is shown) and CAS* toggle in concert with RAS* for four cycles. After a RAS* precharge time, row and column addresses are established and CS, R/W*, RAS*, CAS*, and DBE appropriate to the DRAM access are asserted.

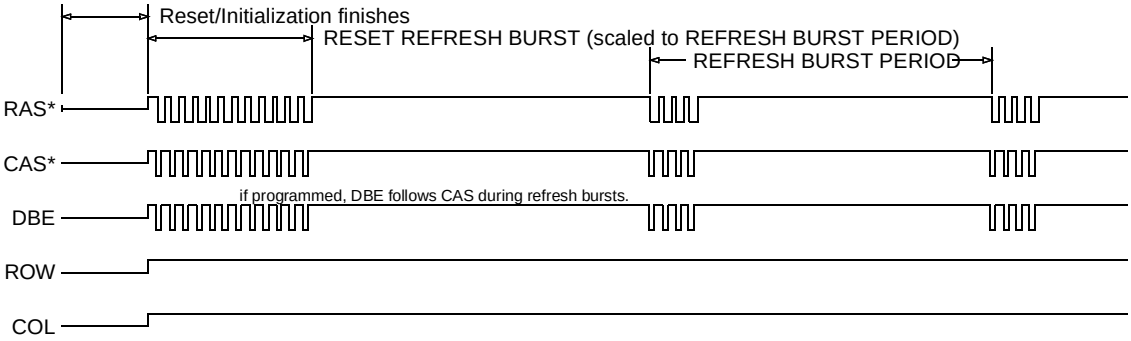


Figure 3-28. Refresh after Initialization

3.8.7 Refresh and Reset

If the CY7C960 is reset by a front panel switch, independently from the VMEbus SYSRESET*, a “Warm Reset” is supported whereby DRAM data is preserved across the reset period. The initialization sequence, whether Serial, Combo, or VMEbus method, is followed immediately by a DRAM Refresh burst of appropriate length. *Figure 3-28* illustrates this behavior. The actual length is calculated by the CY7C960 based upon the refresh timing parameters just loaded by the initialization sequence. The assumption is made that the DRAM was being refreshed normally up to the point that the local Reset signal was generated, so the CY7C960 calculates the number of refresh cycles needed based upon the time taken by the Serial Method of configuration (380 µsec). The reason for this is that the VMEbus method of configuration is faster under normal circumstances so the worst case for refresh purposes is Serial. Note however that if the VMEbus method is delayed substantially for any reason (for example Bus occupancy or bus tenure problems) then the refresh burst may not be sufficient to maintain DRAM contents. If warm reset is desired, the Serial configuration method is recommended.

During the extended refresh burst following a reset, the CY7C960 handles transactions normally. The user is cautioned that creating a local bus condition which blocks refresh after warm reset could overflow the refresh controllers hidden refresh counter which is limited to storing 256 refresh intervals.

Figure 3-29 shows the timing of the signals associated with a series of standard VMEbus transactions. The figure shows three back-to-back transactions: a LOCK cycle, an A64/D16 BLT, and an A24 Serial single cycle I/O transaction. The CY7C960 is configured as the responding slave to all three. The LOCK cycle is DTACKed, but no local activity takes place as the CY7C960 does not provide full LOCK support. The A64/D16 BLT write commences with an address broadcast cycle followed by 13 data beats. A refresh burst interrupts local bus activity, delaying the write of the last word. CY7C960 response to the I/O transaction is delayed until the local bus is ready. The A24 I/O transaction then completes.

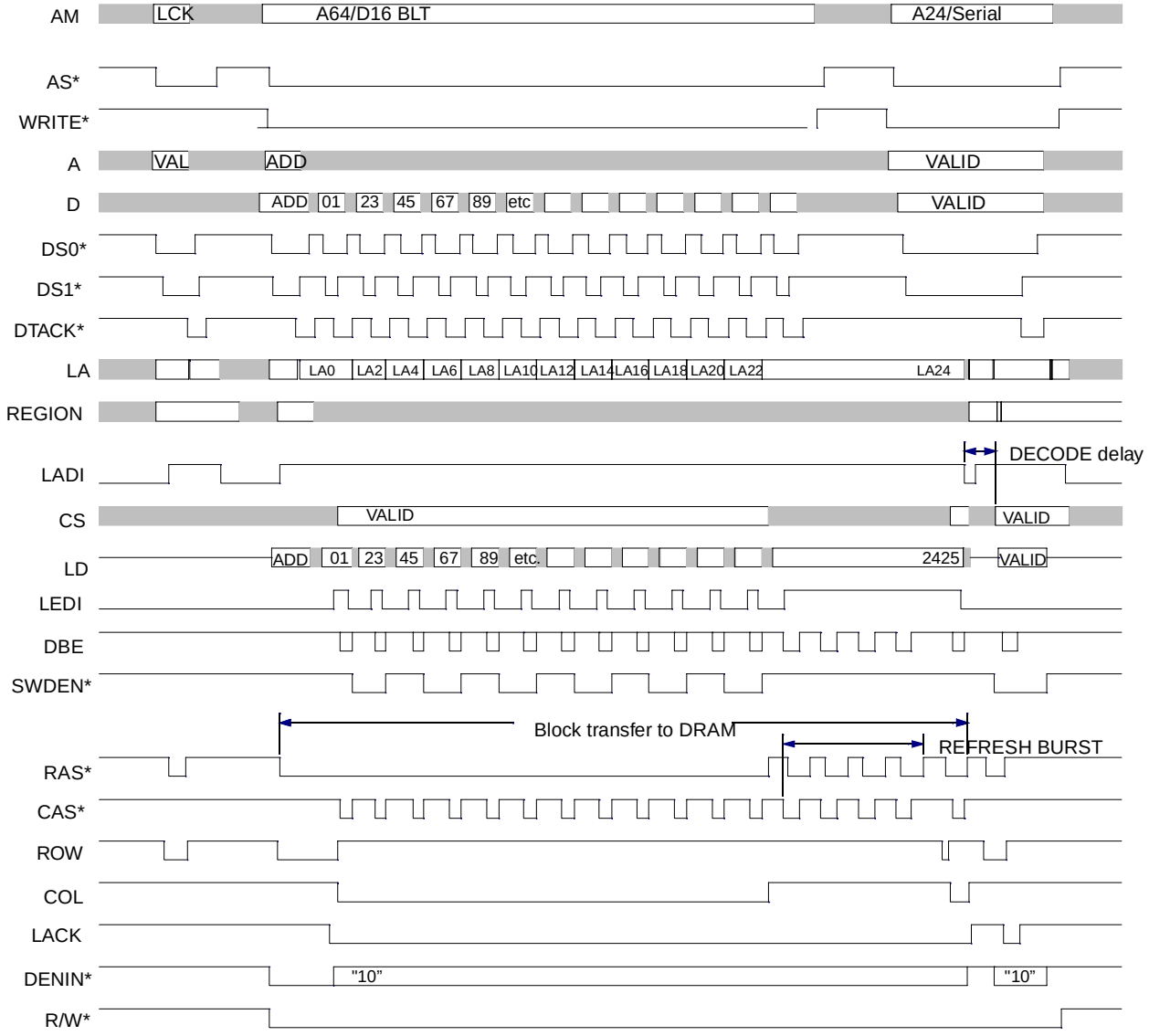


Figure 3-29. DRAM Transaction Timing

The LOCK cycle illustrates DRAM mode RAS* behavior. RAS* will always assert in response to VMEbus AS* assertion when CY7C960 is ready, regardless of transaction decode status. This behavior speeds up response for transactions that are decoded. It generates a harmless RAS*-only refresh signaling for transactions that are not decoded. Local bus hold-off always blocks RAS* assertion while LACK is deasserted. The CY7C961 should be used if full LOCK cycle support is desired.

The A64/D16 BLT write shows the posted nature of block writes. RAS* latches the flowthrough address during the address broadcast. The CY7C960 latches data in the interface with the LEDI signal and drives DTACK* as it begins local bus write activity. The CS outputs are driven with the pattern appropriate for the Region being addressed. CS signals are set up to the CAS* burst and remain valid through a burst. SWDEN* and DBE signals indicate the bytes valid for each data cycle. The BLT progresses, with the CY7C960 incrementing local addresses, and providing CAS*, DBE, and SWDEN* appropriate to each DRAM access.

In the example illustrated, a DRAM Refresh occurs at the end of the BLT, prior to the completion of the last local write. ROW and COL signals go inactive, and the four CAS* before RAS* cycles occur. Then the CY7C960 drives the original ROW address on the local address bus and drives ROW and RAS* to relatch the ROW address in the DRAMs. Next, COL, CAS*, DBE, and SWDEN* are driven appropriately, and the posted data in the CY7C964s is written to DRAM.

While this delayed write is going on, the A24 Serial VMEbus cycle has commenced that is also addressed to the CY7C960 board. Note that the cycle is extended until the previous local activity, refresh and posted data, has completed. Then the VME cycle is DTACK'ed, the data is posted (it is another write operation), and the local signals are driven appropriately. In this case the single cycle is addressed to a Region where DRAM is disabled, so no CAS* occurs The chip select pattern is driven by the CY7C960 along with the correct DBE and SWDEN*.

Those developers familiar with VMEbus transactions will know of the challenge associated with handling multiplexed, posted Write transactions together with address boundary crossing, and high priority refresh cycles. The CY7C960 handles all possible cases without intervention or monitoring being necessary. For example, a refresh burst may preempt the local writing of the last 64-bit MBLT transaction posted prior to a 2K boundary crossing. The CY7C960 completes the refresh burst, but in parallel, the rebroadcast VMEbus address is signalled by AS* toggling and DS* going Low. This address broadcast cycle cannot be completed immediately: first CY7C960 drives RAS* to relatch the PREVIOUS page address which is still held on the local address bus by the CY7C964s. Next, two CAS* cycles occur in order to write the two 32-bit longwords that were posted from the 64-bit write. CY7C960 then goes through a complete decode cycle to ascertain whether the newly-broadcast VMEbus address is still in an enabled Region: if so it cycles RAS* to latch the NEW page address, in parallel, DTACKing the VMEbus address broadcast cycle. Then the block transfer continues normally.

3.8.8 Local Acknowledge Behavior

In some applications it may be desirable to delay DRAM cycles. In these cases, the local signal LACK is used. If LACK is Low, CAS* is asserted at the appropriate time. If LACK is High when RAS*/CAS* delay time expires, then CAS remains deasserted: when LACK goes Low CAS* is asserted. This provides a mechanism for holding off DRAM accesses if desired. Note that this operation is limited to the FIRST CAS* after RAS* is asserted, not a cycle-by-cycle basis. Anytime RAS* is asserted the opportunity to delay the cycle occurs.

If LACK is High for a considerable period of time, the refresh burst will become due. CY7C960 initiates the refresh burst. After the refresh burst is complete, the part continues awaiting the arrival of LACK. If LACK has gone Low during the refresh burst the cycle completes without further delay.

Figure 3-30 shows the use of LACK. An A32/D16 BLT Read commences: the CY7C960 determines that it is addressed and the transfer is enabled, and so drives the local signals appropriately. ROW and RAS* latch the ROW address in the DRAMs, and soon after SWDEN* is driven because of the need to swap byte lanes for the addressed data bytes. But the local circuitry cannot, for some reason, allow the access, and LACK remains High. Therefore, CY7C960 maintains ROW active, and does not drive COL, CAS*, or DBE until LACK has been driven Low by the local circuitry. Once LACK has gone Low, the BLT commences.

3.8.9 DBE Signal Behavior

The Data Byte Enable signals are used to indicate which local data bytes are active for the current cycle. They do not necessarily reflect the VMEbus data byte lanes being used for the transfer. For example, if a VMEbus D16 transfer is selected, and the most significant two bytes, LD[31:16], are enabled onto the local bus by DBE[1:0], the VMEbus specification ensures that those two bytes are carried on D[15:0]. CY7C960 provides the SWDEN* signal for controlling a swap buffer if D8 or D16 transfers are required.

For DRAM cycles the timing of the DBE[3:0] signals is identical to CAS*. In some applications this means that the appropriate DBE pin can be connected to the CAS* input for one byte width of DRAM. For accesses to Regions where DRAM is not enabled, the DBE timing is taken from the programmed value for DBE Assert Time. See Chapter 3.9, I/O Control Description, for more details.

If DRAM refresh is enabled, then the DBE pins can be configured to provide the refresh burst if desired. If DBE Refresh is enabled during initialization, all four DBE pins carry the same refresh timing that is driven on the CAS* pin. If DBE refresh is disabled, then all four signals are not driven due to refresh.

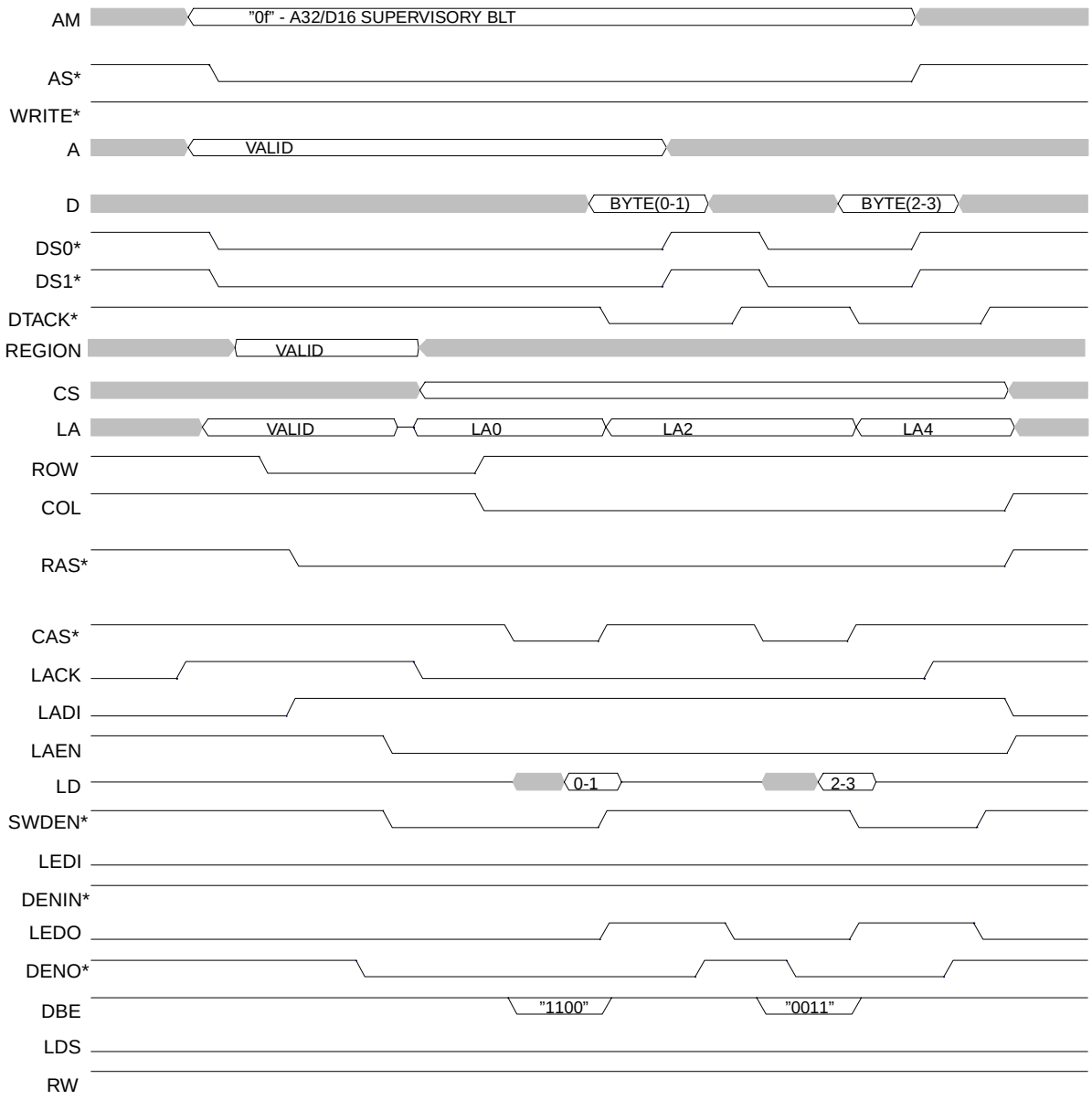


Figure 3-30. LACK in DRAM Cycles

3.8.10 Formal Signal Description

3.8.10.1 RAS*, CAS*, ROW, COL

RAS*, the DRAM Row Address Strobe, is an active Low signal that is used by the DRAM to latch the Row address internally. The falling edge of RAS* is timed with respect to the local address being driven by the CY7C964s so that there is sufficient set-up time prior to the falling edge for all commonly available DRAMs.

CAS*, the DRAM Column Address Strobe, is an active Low signal that is used by the DRAM to latch the column address internally. The falling edge of CAS* is timed with respect to the local address being driven by the CY7C964s so that there is sufficient set-up time prior to the falling edge for all commonly available DRAMs.

ROW, a signal provided by CY7C960 whose polarity is programmable, is intended to multiplex the row address signals to the address pins of the DRAM array prior to the falling edge of RAS*. ROW is set up 1 clock period before the falling edge of RAS*, and held 1 clock period beyond the falling edge of RAS* to guarantee that all common multiplexing circuitry has sufficient set-up and hold time.

COL, a signal provided by CY7C960 whose polarity is programmable, is intended to multiplex the column address signals to the address pins of the DRAM array prior to the falling edge of CAS*. COL is set up 1 clock period before the first falling edge on CAS*, and held active for the duration of the VMEbus transaction as appropriate.

Neither ROW nor COL become active for refresh bursts.

3.8.11 Programmable Features

3.8.11.1 Refresh Enable

A bit is set during initialization that enables refresh activity.

3.8.11.2 Cycle Timing

The following parameters are set during configuration: RAS* Precharge (min. 5, max 12 clocks); CAS* precharge (min. 1, max 8 clocks); CAS* assert (min. 3, max 10 clocks); RAS*/CAS* delay (min. 2, max 9 clocks)

With an 80-MHz clock, and 100-nsec RAS* access time DRAMs, an example would be RAS* precharge 7 clocks, CAS* precharge 1 clock, RAS*/CAS* delay 2 clocks, CAS* assert 3 clocks.

3.8.11.3 Refresh Period

CAS* before RAS* refresh cycles occur in bursts of 4. The timing is taken from the cycle timing just described. The time interval between bursts is programmable during configuration to be

$$(N \times 256) \times clockperiod$$

where N is a value between 1 and 255.

For example, if a DRAM was used with R rows requiring to be refreshed every t msec, then N would be found as follows

$$N = int\left(\frac{4}{R} \times \frac{t}{10^3} \times \frac{1}{256} \times \frac{1}{clockperiod}\right)$$

In the case of a 256 row device with a refresh period of 4 msec and a clock rate of 80 MHz, N turns out to be 19 (13hex). The period between bursts in this case would be seen to be about 61 μsec.

3.8.11.4 DBE Refresh

The DBE pins can be configured to provide the CAS* timing for refresh bursts. If this is not enabled, then the DBE pins do not toggle during refresh bursts.

3.8.11.5 DBE Polarity

The polarity of the DBE pins can be programmed as a group during initialization.

3.8.11.6 ROW, COL Polarity

The polarity of the ROW and COL pins can be programmed individually during initialization.

3.9
I/O Control Description

The CY7C960 has two basic modes of operation: DRAM mode and I/O mode. The user selects the mode during configuration. This section describes the I/O mode of operation.

In I/O mode the CY7C960 does not provide the timing signals required by DRAM, such as RAS*/CAS*. The pins that provide these signals in DRAM mode are therefore available for I/O functions in I/O mode, and are used as Chip Select Outputs.

The signals that form the Local I/O Mode Functionality are CS[5:0], LACK, and DBE[3:0].

Figure 3-31 shows the block diagram of the CY7C960 in I/O mode. It is, as may be expected, very similar to the block diagram of the DRAM/IO Mode (see Introduction). The DRAM control block is disabled, and hence does not appear in the figure, and the pins thus freed are reused as Chip Selects. One additional Region Input (REGION[3]) is provided in I/O mode, allowing 16 Regions to be individually configured.

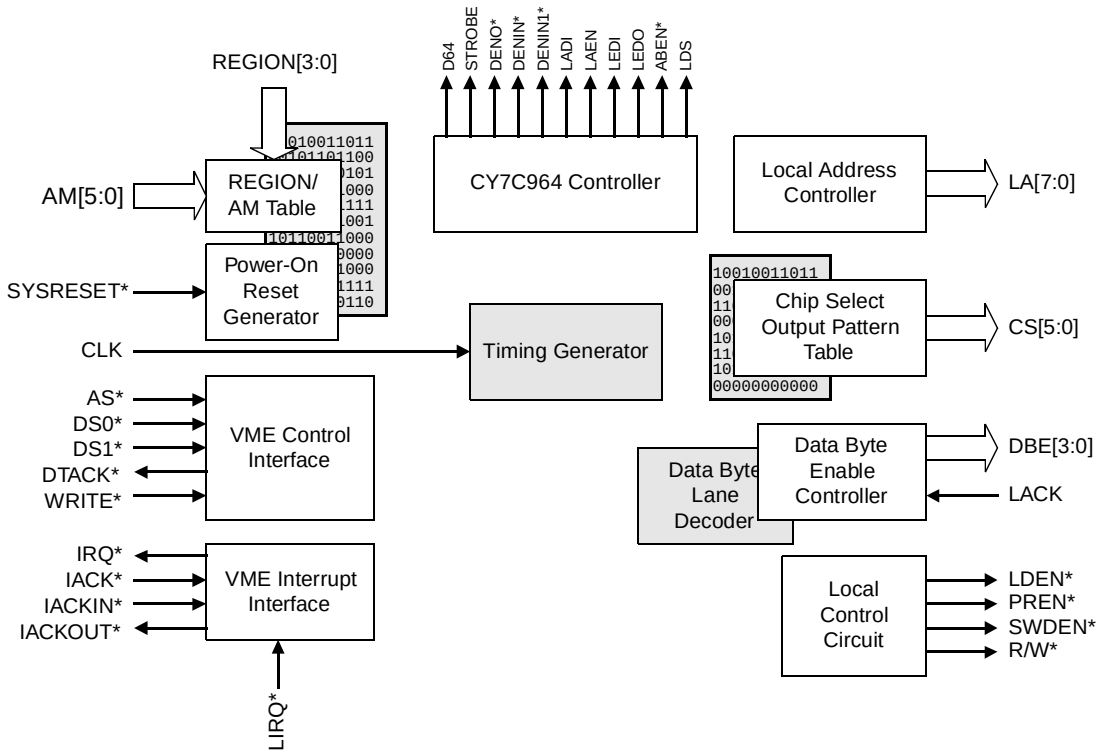


Figure 3-31. I/O Mode Block Diagram

3.9.1 Region Mapping

The CY7C960 introduces the concept of Region Mapping for I/O control. The CY7C960 provides up to 16 Regions when in I/O mode, selected by REGION[3:0]. The external Slave Address Map Decoder drives the REGION inputs in response to changes in the VMEbus address signals. When the CY7C960 detects the start of a valid transaction on the VMEbus (AS going Low) the REGION inputs are used to point into a table that contains entries for each possible VMEbus Transaction Type. The table entries are programmed during the initialization process, and are not adjustable after configuration is complete. If the VMEbus transaction is enabled in the table, then the CY7C960 proceeds. If the transaction is disabled then the CY7C960 takes no action and assumes that the VMEbus cycle is intended for another board. Further information is provided in the VMEbus Chapter.

This provides great flexibility in address mapping. For example, two separate VMEbus boards can occupy the same address in VMEbus space: one can react to 32/64 bit data transfers, the other can react to only 8/16 bit data transfers. Or one can selectively respond to block transfers, the other single cycles.

Each of the 16 Regions has a unique entry in the table, allowing for fine granularity in the Slave Address Map, and providing a wide range of options if the user cares to use them. If a simple Slave Address Map is desired, then the companion CY7C964 contains comparator circuitry which could be used, and the majority of the 16 possible regions could be selectively disabled.

Table 3-8 shows the use of the Region AM Code programming table. In this example, only transactions directed to Regions 0 through 4 could result in the CY7C960 driving DTACK*. Furthermore, only A64 supervisory block transfers, of any data width, are enabled in Region 0. The other Regions have different types of VME transaction enabled.

Table 3-8. Example of Region AM Code Enabling

REGION	A64	A40	A32	A24	A16	SUPER	NPRIV	PROG	DATA	BLT	LONG
0	✓					✓				✓	✓
1		✓					✓			✓	✓
2			✓			✓		✓			✓
3				✓			✓		✓		✓
4				✓	✓	✓					
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											

3.9.2 Chip Select Output Control

The six pins dedicated to I/O mode output, CS[5:0], can be configured during the initialization process in a flexible manner. Once the CY7C960 has detected that the VMEbus cycle is valid and enabled for the Region being addressed, then a pattern is driven from the Chip Select Outputs (CS[5:0]). The pattern is found in a table that contains entries for each possible Region, in a manner similar to the Region AM Code Enabling Table (*Table 3-8*). The entries are programmed during initialization and are not changeable after configuration is complete.

The user can select the polarity of each individual Chip Select Output to be High or Low true. This sets the output pattern when no VMEbus cycle has been decoded to drive the enabled Regions. In other words, this pattern is the relaxed state of the CS[5:0] pins. To effect a change in one or more of the 6 Chip Select pins, the appropriate bit or bits of the Chip Select Output Control Table (*Table 3-9*) entry (for the desired Region) are set. Any VMEbus transaction that is enabled for the Region it maps to will then cause that pattern to be driven from the CS[5:0] pins.

Table 3-9 provides an example of how the Chip Select Programming and Polarity may be set. The state of CS[5:0] after initialization becomes 000111, because the polarity field determines the unselected state of the signals. Any transaction directed to Regions 8, or 12 through 15, will not result in a CS[5:0] change as no CS programming has been entered for those Regions.

Regions 0 through 7 cause CS[2:0] patterned to be driven. Regions 9, 10, and 11 cause CS[5], CS[4], and CS[3] to change respectively.

Table 3-9. Example of Chip Select Output Control

REGION	CS[5]	CS[4]	CS[3]	CS[2]	CS[1]	CS[0]
0						
1				✓	✓	✓
2				✓	✓	
3				✓		✓
4				✓		
5					✓	✓
6					✓	
7						✓
8						
9	✓					
10		✓				
11			✓			
12						
13						
14						
15						
Polarity	L	L	L	H	H	H

An access to Region 0 therefore would cause the pattern 000000 to be driven. Region 9 would cause 100111.

3.9.3 Chip Select Output Timing

3.9.3.1 Overview

When a valid VMEbus transaction is decoded that maps to a Region with a Chip Select Pattern enabled, the pattern is driven from the CS[5:0] pins after the VMEbus DSi* signal goes true. Then the appropriate DBE signals are driven. After either the self-timed delay or the LACK handshake, the VMEbus DTACK* is driven and the DBE pins are disabled. If the VMEbus transaction is a block transfer, the CS[5:0] pattern is maintained and the DBE pins toggle interlocked with the VMEbus handshake mechanism. After the final transfer, signified by AS* going inactive, the CS[5:0] pins return to the relaxed state.

3.9.3.2 Read-Aheads

In VMEbus block read transactions, the CY7C960 optimizes performance by performing read-ahead cycles whenever possible. Therefore, if the AS* signal is slow going inactive, an extra DBE cycle may occur. The data read locally is not passed to the VMEbus and the cycle completes as soon as the AS* signal goes inactive.

To guarantee that a read-ahead cycle will not occur, it is necessary to deassert AS* at the same time that DSi* is deasserted. This of course is a VMEbus master transaction, not under the control of the CY7C960.

3.9.3.3 Local Acknowledge Timing

The CY7C960 provides the option of handshaking local accesses, or having the access be self-timed. The LACK pin provides the handshake. The time period for the self-timed access is programmed during configuration to be a value between 3 and 18 clock periods. When a VMEbus access causes the local cycle to commence, Chip Select goes active followed by the appropriate DBE pins. The width of the DBE signal is controlled: after the expiration of the access time, if LACK is Low the cycle terminates. If LACK is High, the cycle is extended until LACK goes Low.

Therefore, if the user ties LACK Low, the width of the DBE pulse is determined only by the programmed value for the Region's CS Access Time field. Each Region can have a unique value for this field. The user may also choose to Handshake the cycles with LACK, in which case the minimum width is set by the Access time field, and the maximum time is determined by LACK.

Figure 3-32 provides an example of one of the most challenging VMEbus Transactions—the A40/MD32 Read-Modify-Write Cycle. Once the CY7C960 determines that the transaction is directed to a Region for which it is enabled, the address broadcast cycle is DTACKed and the Read cycle commences. The appropriate pattern is driven on the Chip Select signals. Two 16-bit local accesses are needed to construct the MD32 VMEbus transaction, and SWDEN* and DBE signals are driven appropriate to the address requested. Once the 32 bits of data are latched into the CY7C964s and enabled onto the VMEbus signals (A[15:1], LWORD*, and D[15:0]) the VMEbus DTACK* is driven. The WRITE* signal is toggled by the VMEbus master, signifying an RMW cycle. The CY7C960 responds to the subsequent DSi* assertion by latching the state of A[15:1], LWORD*, and D[15:0] in the CY7C964s, thus posting the data, and completing the VMEbus cycle by driving and releasing DTACK*. The local cycle continues by reasserting the previously-latched lower 24 bits of address on the local address bus and driving SWDEN* and DBE signals appropriately for the address to be written. Two local cycles occur to write the MD32 data.

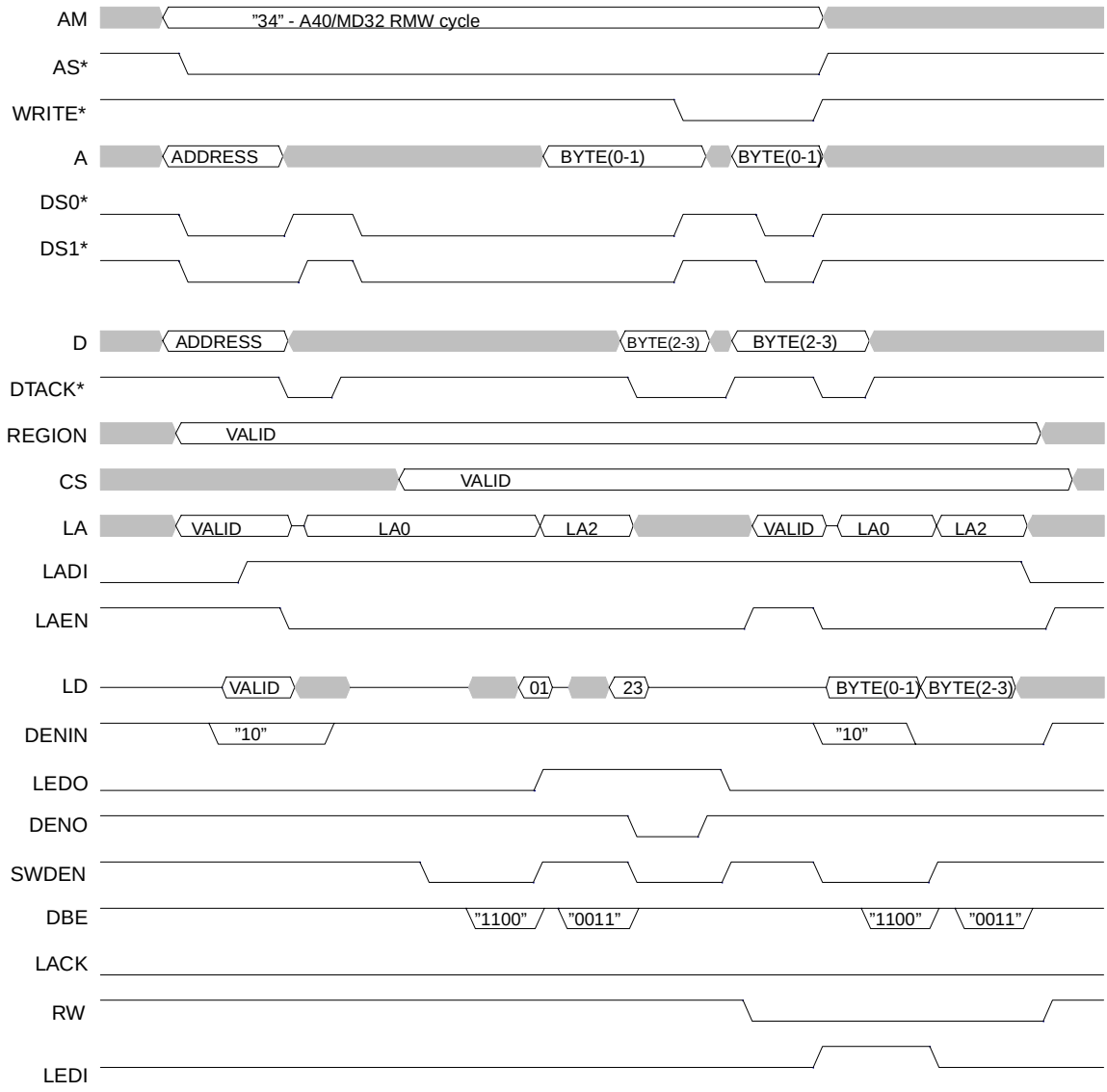


Figure 3-32. I/O Cycle Timing

Figure 3-33 shows an example of read-ahead operation. The transaction shown is an A32/D16 BLT, 2 transactions (4 bytes) long. In this example, DRAM is enabled for the Region being addressed so the DRAM control signals are shown in addition to the Chip Select signals. The two VMEbus Read Cycles are completed normally, and the local signals, SWDEN*, DBE, and Chip Selects are driven appropriately for the address being transferred. The 16-bit local data is driven on the correct VMEbus data signals (D[15:0]). After the second DTACK from the CY7C960, the VMEbus Master releases AS*, signifying the end of the block transfer. But meanwhile, the CY7C960 has provided another read sequence to the local circuitry. In this case, CAS* and DBE have been driven and the data has been presented by the local circuitry to the CY7C964 local data pins. When AS* is detected High, the CY7C960 simply drives all local controls inactive and terminates the local cycle with no ill effects.

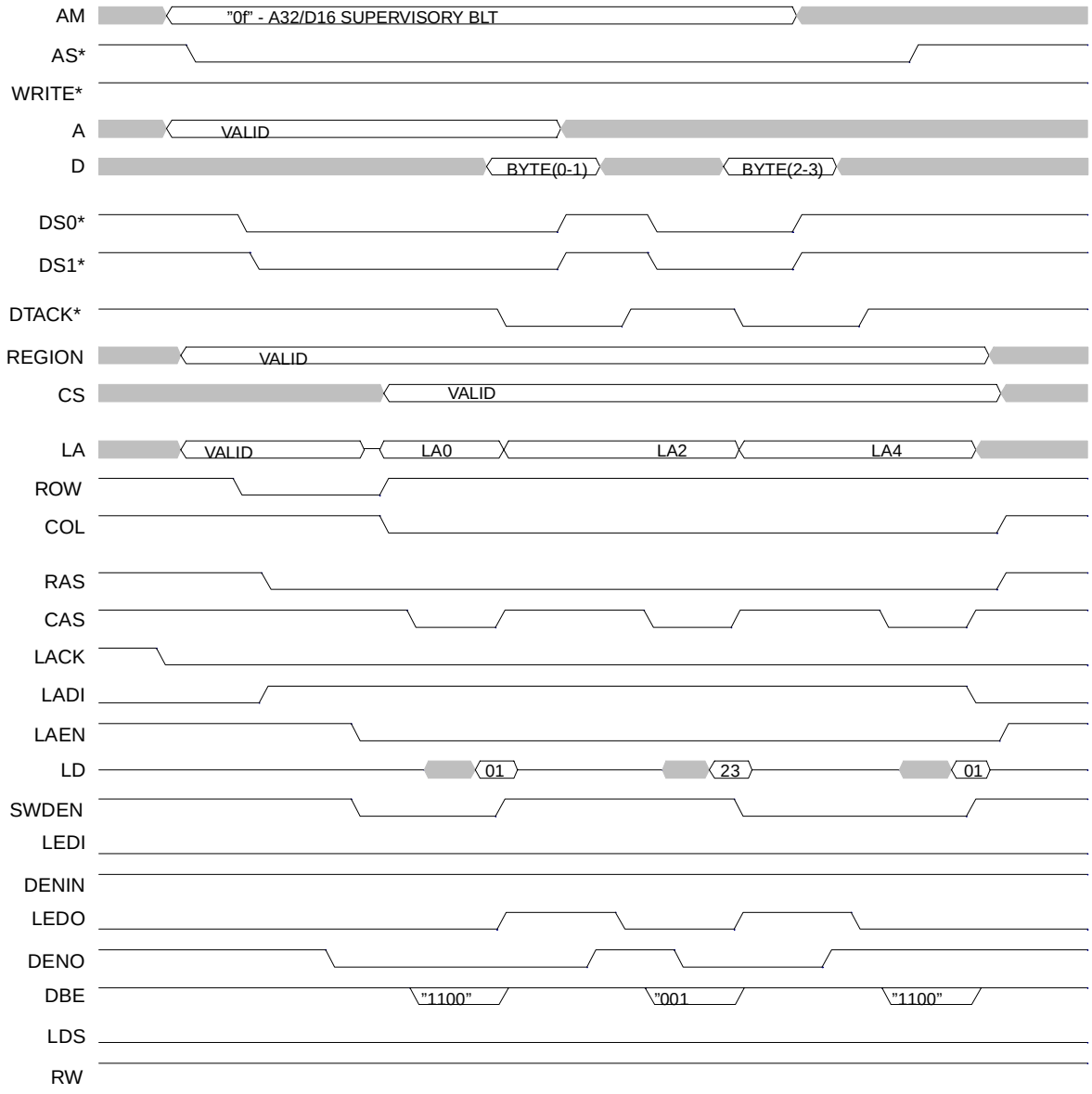


Figure 3-33. Example of Read-Ahead Timing

3.9.4 Data Byte Enable Usage

Table 3-10 indicates which Data Byte Enable signals are active for all possible VMEbus Transactions. The polarity assumed for the DBE pins is LOW true.

Table 3-10. DBE Signal Truth Table

	DS1*	DS0*	LA1	LA0 (LWORD*)	DBE3	DBE2	DBE1	DBE0
Byte	0	1	0	1	1	1	1	0
	1	0	0	1	1	1	0	1
	0	1	1	1	1	0	1	1
	1	0	1	1	0	1	1	1
Word	0	0	0	1	1	1	0	0
	0	0	1	1	0	0	1	1
Longword	0	0	0	0	0	0	0	0
Unaligned	0	1	0	0	1	0	0	0
	1	0	0	0	0	0	0	1
	0	0	1	0	1	0	0	1

3.9.5 Using I/O In DRAM Mode

When the CY7C960 is configured for DRAM, three pins remain for use as Chip Select Outputs, CS[2:0]. When a transaction takes place that maps to a Region where DRAM is enabled the Chip Select output pattern for that Region is driven from CS[2:0]. The DBE pins are driven with timing appropriate for a DRAM access (CAS* timing parameters configured during initialization). When a transaction takes place that maps to a Region where DRAM is disabled, then the Chip Select pattern for that Region is driven from CS[2:0], and the DBE pins are driven with timing appropriate for an I/O access (DBE assert time configured during initialization).

3.10 Design Considerations

3.10.1 Design Philosophy

The CY7C960 and the CY7C961 each share the basic VMEbus slave circuitry, and each has a common design philosophy. The most basic foundation for the design was that of achieving high performance in whatever system the devices were placed. For the CY7C960, that philosophy determined the hierarchical partitioning of the design into several independent blocks, each with its own state machine. The coexistence of these state machines demanded a fully-synchronous internal design in order to remove the necessity for block-to-block synchronization circuitry, or pipelining. Thus the internal operations are all designed to take place within one clock period, so that results from one state transition are available to other blocks in the cycle that they occur. No “analog” delays are employed in the design, and the standard VMEbus timings are determined by counting clock periods internally. The clock input frequency that the two parts expect to see is 80 MHz: at this frequency all the VMEbus timings are assured. The design is fully static, and the clock frequency must therefore be considered when specifying the configuration of the application. All parameters that are a function of clock period are specified as such (e.g., 3T + 4 nsec) in the AC timing tables (Chapter 3.12).

There is a comprehensive set of timing diagrams in Chapter that should be consulted to determine the relationship of the various signals. Where possible, there are AC timing parameters specified in *Table 3-16*. In general, however, the diagrams will give a better understanding of the operation of the parts than a table of numbers.

3.10.2 CY7C964 Interface

The operation of the interface is described in Chapter 3.6, and in Section 4, The CY7C964 Bus Interface Logic Circuit. The background to this section may be of interest.

The first controller produced by Cypress for the VMEbus community was the ubiquitous VIC068A—the current industry-standard controller. This device was intended to be used in concert with the state of the art in low-cost drivers—then “ACT” logic devices such as ‘543s and ‘245s. The control outputs from the VIC068A were crafted to interface cleanly with these devices. Thus LEDI, LEDO, DENIN*, DENO*, etc. from the VIC068A were signals whose timing was designed to turn the drivers on or off, and latch the data and address buses, at appropriate times for the VMEbus transfers. When the VIC64 was introduced it inherited the identical timings, because the VIC64 was socket-compatible with the VIC068A. The CY7C964 was designed as a glueless companion to VIC64, replacing the ACT logic with a higher level

of integration to allow the increased functionality of the VME64 specification to be implemented in similar board space. So it also inherited the control signal timing of the VIC068A.

Several of the control signals perform double duty, such as LDS and MWB*, which are used during the comparator load sequence in the CY7C964. But in general, the signals controlling the ‘964 are the original buffer control signals from the VIC068A.

Therefore, when the CY7C960 and ‘961 were designed, they had to conform to the original control timing imposed by the VIC068A in order to use the popular CY7C964. Therefore the control signals have the same names, functions, and timings as the original VIC068A and the VIC64. One advantage of this is that applications that do not require the sophistication of the VME64 specification are not forced into employing the sophisticated CY7C964—the control signals work with the FCT family of devices that are also offered by Cypress. These devices allow simpler applications to achieve cost goals without sacrificing performance. The limitation, as should be expected, is that the more complex VMEbus transactions are not implemented unless the CY7C964 is employed.

3.10.3 Local Bus Philosophy

The CY7C960 and ‘961 are very flexible in the way they deal with local circuitry. However, they are not intended to compete with their more sophisticated cousins, VIC64 and VIC068A. Whereas VIC068A and VIC64 have a local bus arbitration circuit, on-chip DMA engine, and local interrupt circuitry, the ‘960 and ‘961 are conceptually much simpler. On the surface, the pinout reveals a simple local acknowledge handshake, and a local interrupt input. The ‘961 also provides a local bus error function. Beneath the surface, there is another level of complexity that can be invoked by applications that need somewhat more local bus control.

The CY7C960 was intended to be the highest priority on the local bus. In other words, the ‘960 assumes that when a VMEbus slave transaction occurs, nothing will prevent it from reading or writing local transactions other than the local acknowledge handshake. The local cycle starts with no regard for other masters which may be accessing local resources. This optimizes VMEbus performance, preventing VMEbus cycles being extended by local bus contention. However, this philosophy is not beneficial in all cases. Some rudimentary control over the local bus may be needed from time to time by other devices.

To cater to this situation the CY7C960 can be prevented from starting a local cycle, or a refresh of local DRAM. To explain this operation, first consider the VMEbus activity. Whenever AS* is asserted by a VMEbus master, the CY7C960 will drive RAS* Low, and the VMEbus address is driven onto the local address bus by the CY7C964s. This happens whether the CY7C960 is addressed or not, in order to optimize performance. The problem for local bus ownership is that there is no way to predict when AS* will go active. Hence the only way to guarantee that AS* is not about to go Low, is to watch it go High. Then the AS* minimum High time provides a window of opportunity for the local master to capture the local bus from the CY7C960. The local signal which can be used to monitor AS* going High is LADI. This signal also indicates DRAM refresh activity, thus preventing a conflict between the local bus master

and a local DRAM refresh cycle. The local acknowledge signal is used to capture the local bus: if enabled to do so by a bit in the configuration sequence, the LACK pin prevents the start of a ‘960 local cycle. If LACK is driven High (or is High already) when the window described earlier occurs, then no local signals are driven by the ‘960 whatever happens on the VMEbus. Control signals to the ‘964 are driven to cause the least significant ‘964 to be three-stated and the local bus master can now perform local cycles without fear of intervention or DRAM refresh. LAEN of the remaining ‘964s must be controlled by external logic. The local bus is given back to the CY7C960 upon the assertion of LACK (Low).

In some applications, VMEbus traffic may not provide enough opportunities—AS* may not cycle frequently. In such a case, the local master can still acquire the local bus by driving LACK High, waiting a period of time (2T) then examining LADI. If LADI is still inactive then the local bus is available. If LADI is active, then the VMEbus transaction has begun and the ‘960 is about to start a local bus access.

Note that, while the local bus is not available to the CY7C960, any VMEbus accesses to the CY7C960 will hang until the local bus is again available. Also, the refresh engine will be unable to refresh the DRAM; when LACK is driven Low, the refresh engine has priority and will burst all the missed refresh cycles (256 bursts) before the CY7C960 will respond to a VMEbus cycle.

Note that this function is enabled by a bit in the configuration bit stream. If the bit is not enabled, then the CY7C960 cannot be prevented from DRAM refresh, or from starting a local cycle. The function of LACK is then simply to control the completion of local cycles allowing for slow or asynchronous local peripherals.

3.10.4 Read-Ahead Cycles

The CY7C960 does not read ahead single-cycle transfers, but it does perform read-ahead cycles when acting as a slave to a VMEbus BLT read operation. There are two ways to prevent CY7C960 performing the read-ahead, both involving the VMEbus Master: terminate the block transfer at a 256-byte address boundary; or remove AS* with the final DS* deassertion.

Read-ahead is implemented to allow a higher performance to be achieved, as otherwise the slave board will be unable to provide data within the specified timing for minimum transaction period.

There are considerations when designing FIFO-based slave circuitry in order to guarantee that the master will be given an error-free block transfer. The FIFO going empty during the progress of a block transfer Read will not be a problem, assuming that the circuitry external to CY7C960 intercepts the read strobe until the FIFO has data, and does not acknowledge the local cycle until the data has in fact been transferred to the CY7C960. The problem occurs when a data block is desired that is not aligned to 256-byte address boundaries. The final transfer of desired data will be followed by a read-ahead cycle which will, unless intercepted in some way, clock exactly one more data location from the FIFO.

In most applications this will not be a problem, as the system design is usually organized so that the FIFO goes empty after the final desired data byte has been read, and the following read-ahead will not be relevant. Once AS* has gone inactive on the VMEbus, the CY7C960 understands that the data transfer is complete. It drives all local signals inactive, whether or not the cycle has been locally acknowledged. This effectively discards the read-ahead.

In some cases the FIFO may contain relevant data at all times, and a read-ahead cycle would result in loss of data. These applications must either ensure that the data is aligned to 256 byte address boundaries, or ensure that AS* and DS* go inactive together, thereby ensuring that the final cycle is not followed by a read-ahead.

3.10.5 Write Posting

A similar consideration is inherent in the BLT Write operation. A common system configuration on VMEbus is that the slave “posts” the write data and acknowledges the VMEbus cycle prior to the local acknowledge. This optimizes performance: without BLT write posting, performance is substantially degraded, approximating that of a well-designed single cycle transfer.

During a BLT Write transfer, if the local circuitry detects an error or is unable to complete the transaction, then the local circuitry will not acknowledge the posted data. This causes the VMEbus to extend the current cycle (the one immediately AFTER the posted transaction) and the system timer will timeout, causing a BERR* and allowing the Master to perform error recovery.

In the event that the final local cycle of a block does not terminate correctly, there is no mechanism for communicating this error to the Master. The cycle has already been DTACKed, and the bus ownership has been relinquished. This is a problem common to all VMEbus Slave Write Posting applications.

The CY7C960 posts all BLT write transactions. If the system designer has a problem with write posting due to the last-cycle issue, then single cycle transfers must be used. CY7C960 does not post single cycle write data.

3.10.6 VMEbus Error Considerations

Because of pinout limitations the CY7C960 does not monitor or drive BERR*, the VMEbus error handshake. In the case where the VMEbus cycle terminates with a BERR*, indicating that the local data was in fact needed but was not provided before the VMEbus timer expired, CY7C960 cannot see BERR*, but still sees AS* go inactive. The CY7C960 always terminates the local cycle in response to AS* going inactive. In this case it is the responsibility of the Master to take action following the BERR*. The CY7C960 continues to operate normally.

3.11
CY7C961 Description

3.11.1 Introduction

The CY7C961 is a CY7C960 Slave VMEbus Interface Controller with the addition of a master block transfer capability. Full-featured Slave boards can be built, using the CY7C961, that offer a flexible Master block transfer facility for bursting data across the VMEbus. The CY7C961 can receive its instructions from a VMEbus Master or by programming registers locally. The CY7C961 interprets the instructions and then moves data accordingly as a VMEbus Master. This optimizes performance and bus utilization.

The CY7C961 is a true superset of the CY7C960. Signal pins have been added to control CY7C964 DMA functions. Unidirectional VMEbus pins have been changed to bidirectional. A few additional signals have been provided to complete a master interface, such as a data port and VMEbus requester signals. As a VMEbus Slave, the CY7C961 behaves in every respect like the CY7C960. It has more pins, a master block transfer facility, and (because of the addition of BBSY*) full lock cycle support.

From a system perspective, this CY7C961 master block transfer capability can be viewed as a DMA channel which resides on the slave card, and is controlled by a dual ported on-chip register file. It is possible to program the DMA channel from the VMEbus or from the local side of the interface, or both. Once programmed, the CY7C961 acquires the VMEbus and transfers data in one of 20 user-selected protocols.

Circuitry on the local side of the CY7C961 sees the same control signals as were described for the CY7C960. For example, the REGION inputs to the CY7C961 are driven by an external address decoder. The address decoder sits on the local address bus, which is driven from the CY7C964s. The local memory or I/O involved in the data transfer is enabled through the CS outputs and the DRAM control signals from the CY7C961, so they must be configured correctly for the REGION inputs being driven.

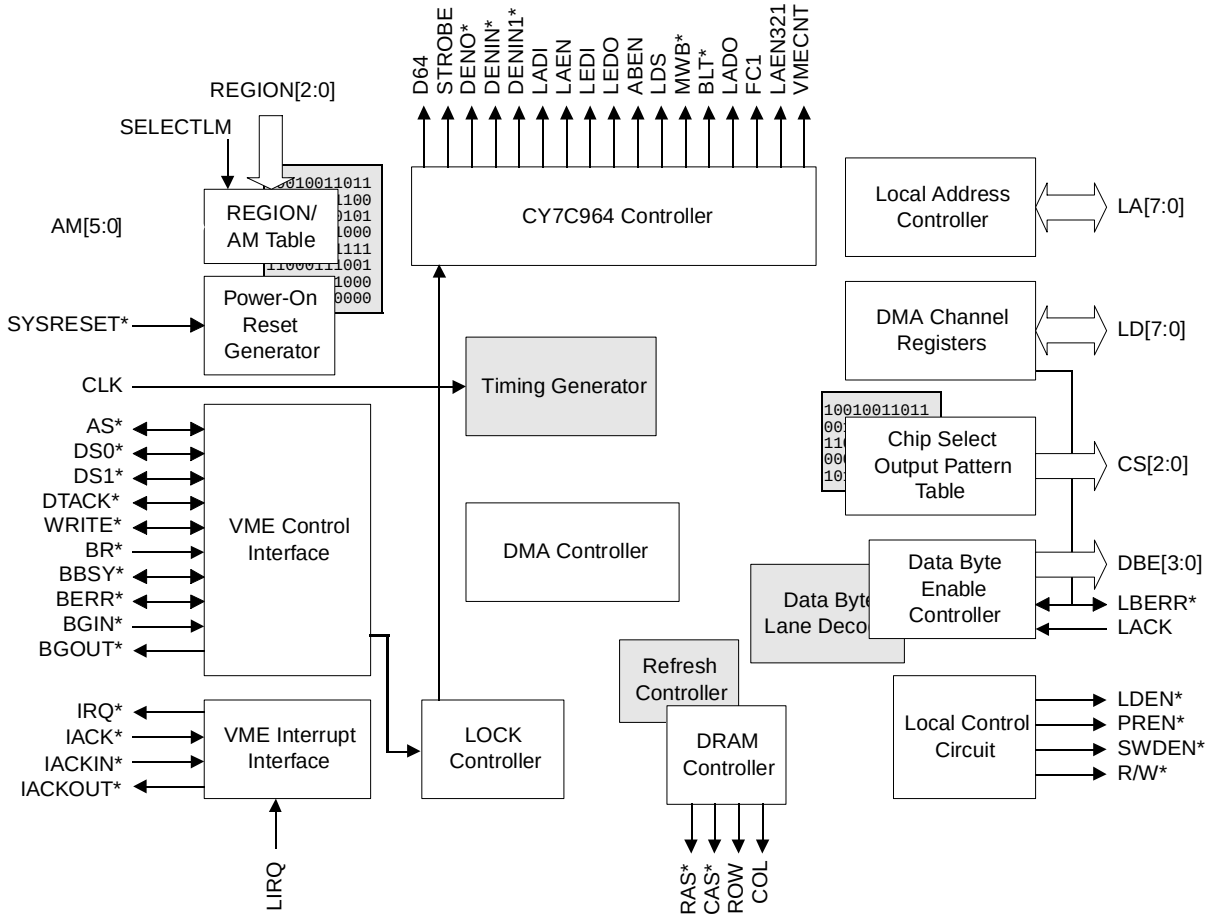


Figure 3-34. CY7C961 Block Diagram

This implies that the VCOMP* outputs from the CY7C964s cannot form part of the address decoder for master operations, because they are looking not at the local address, but at the VMEbus address, which by definition must be pointing somewhere other than the CY7C961's address space.

Figure 3-34 shows the block diagram of the CY7C961. It is very similar to the CY7C960 block diagram (see Introduction), with the addition of some CY7C964 control signals, signals needed to support VMEbus Master Transactions, and other minor changes.

3.11.2 CY7C961 Lock Cycle Support

3.11.2.1 Overview

Lock commands are Address-Only-Cycles-With-Handshake (ADOH) cycles that are used to lock the other port(s) of multiported resources, where one port is on VMEbus. Each Slave's resource that is addressed with a lock command is to lock out all other accesses to that resource. Use of a lock command signifies the start of a locked sequence of VMEbus cycles

which ends with the end of the current VMEbus Master's bus tenure. The CY7C961 allows locked cycles to be decoded, and drives a local lock indicator while the ensuing VMEbus locked sequence is in progress.

3.11.2.2 Description

Each of the lock commands (A16, A24, A32, A40, A64) consists of an address phase which is presented to the VMEbus and handshaken by the targeted Slave. The CY7C961 can be programmed to decode lock commands on any ONE of its 16 decode regions. Any combination of the five lock commands can be enabled for that region, but only one lock indicator signal is provided. When a lock command is decoded, the CY7C961 drives its MWB pin Low to indicate the beginning of the locked VMEbus sequence. The lock indication on MWB will be maintained as long as BBSY* or AS* remains asserted to the CY7C961. A decoded lock cycle will not cause chip selects or byte enables to become active. A decoded lock cycle will not interfere in any way with slave accesses to the CY7C961. *Figure 3-35* shows the timing of the various signals associated with LOCK cycles.

The MWB signal is active during block transfer register accesses and during CY7C961 master block transfers. MWB indicates a default lock condition during master block transfer. This should be considered when designing the local resource lock control circuitry in which MWB is used. If the lock default is not needed during master block transfers, the CY7C961 signal FC1 can be used to disable lock during master block transfers.

3.11.3 CY7C961 Master Block Facility

3.11.3.1 Overview

The CY7C961 Master Block Facility provides "block transfer on demand" capability for slave cards built around the Cypress CY7C961/CY7C964 chip set. This facility allows DMA control by writing a short series of commands to the CY7C961/CY7C964 chip set, telling it how much data to move, where to get it from, where to put it, and what transfer protocol to use while moving it. Blocks can be moved over the VMEbus as indivisible single cycles or BLTs. The protocol menu includes D8, D16, D32, MD32, or D64. A16, A24, A32, A40, and A64 address spaces can be specified. Burst lengths from 16 bytes to 8 megabytes can be requested. Eight registers accessible from the VMEbus or the local side of the interface make the facility simple to configure and simple to control. The facility has a busy semaphore, a VMEbus Interrupt on completion feature with a programmable statusID byte, and a built in requester and bus grant daisychain.

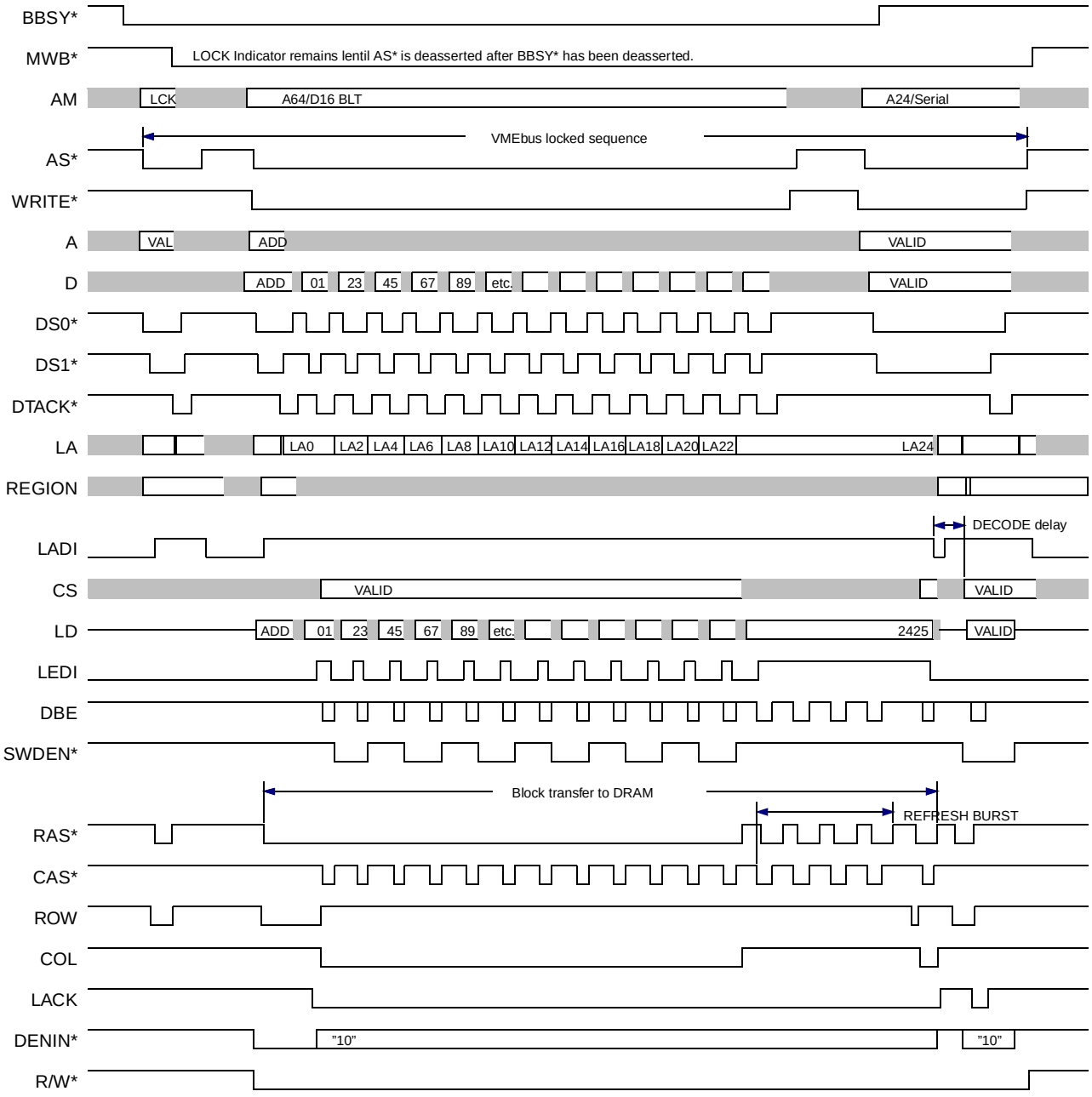


Figure 3-35. LOCK Cycle Timing

It is expected that the system designer will choose either local programming of the DMA channel or VMEbus programming and design decode and other support circuitry based on that choice. There is nothing in the design of the CY7C961 to prevent dual porting of the DMA control registers, but dual porting, like multi-master VMEbus control requires polling of the CY7C961 control register semaphore and/or added complexity in hardware and software design.

3.11.3.2 Master Block Transfer Control from VMEbus

All control of the master block facility is by register access. Eight registers are designed to be accessed over the VMEbus by single cycle VME masters. In order to reach the Master Block facility, the VMEbus address must cause the SELECTLM* signal and a valid REGION vector to be asserted to the CY7C961. Refer to *Figure 3-36* for the register access detail. The assertion of SELECTLM* blocks any local response by the CY7C961. Instead, internal gating and external buffer control signaling appropriate to the register access specified by LA[4:2] is substituted. The CY7C961 asserts DTACK* for valid register accesses or BERR* if an error access is attempted. Eight registers are defined, selected by LA[4:2].

3.11.3.3 Master Block Transfer Control from Local Side of Interface

All control of the master block facility is by register access. Eight registers are designed to be accessed directly through control of LA[4:2], LD[31:0], R/W*, and SELECTLM*. The “local bus holdoff” feature of the CY7C961 must be enabled and the CY7C961 must be in its “holdoff” state before register access is begun. (See section for a complete description of “Local Bus Holdoff.”) In order to reach the Master Block facility, LA[4:2] and R/W are set up to the assertion of the SELECTLM*. For register write cycles, LD[31:0] must also be set up to SELECTLM*.

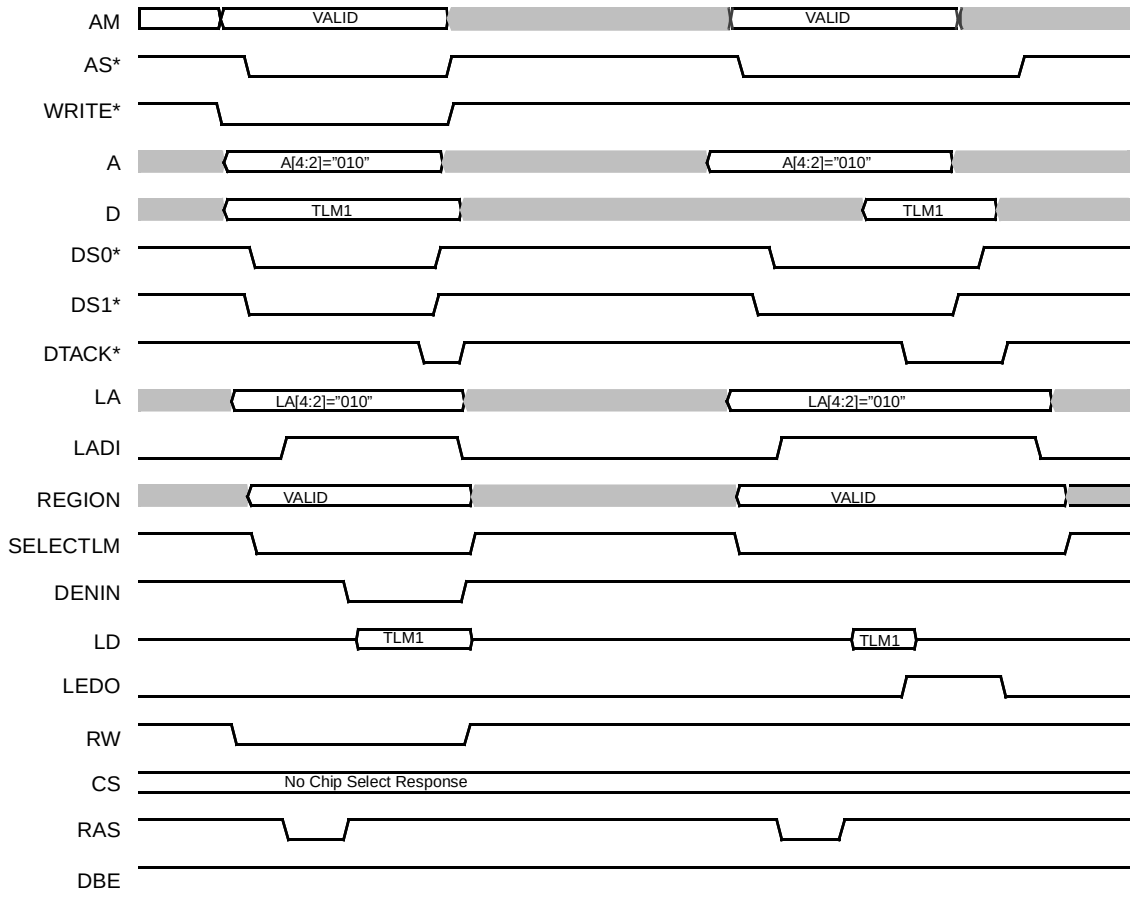


Figure 3-36. DMA Control Register Access from VMEbus

The response of the CY7C961 is self-timed and no acknowledge handshake is provided. SELECTLM* must be asserted for a minimum of 100 ns (eight CLK periods) and LA[4:2], R/W*, and LD[31:0] must be driven valid until SELECTLM* is deasserted. On register reads, CY7C961 drives LD[7:0] two clocks after SELECTLM* is sampled asserted and three-states LD[7:0] one clock after SELECTLM* is sampled deasserted. Refer to *Figure 3-37* for Local Register access signaling.

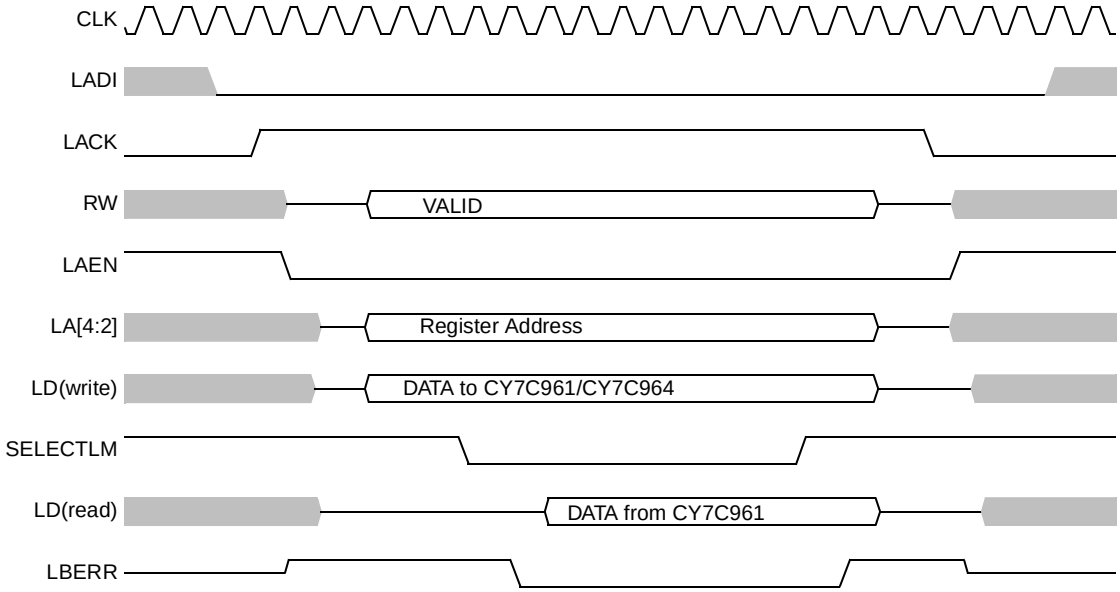


Figure 3-37. DMA Control Register Local Access

The CY7C961 provides control signals to the CY7C964 to control the loading of registers inside the device for operations such as establishing the local starting address and the VME-bus starting address, both of which could be 32-bit quantities stored inside the CY7C964s.

If an illegal register access is attempted, the CY7C961 will signal by driving LBERR* Low within two clocks after SELECTLM* is asserted. LBERR* deasserts one clock after SELECTLM* is sampled deasserted.

3.11.3.4 Programming the Master Block Facility

In general, the set-up programming is very simple. First the semaphore must be read. This register read unlocks the facility. Next, values are written to Transfer Length, Transfer Type, Upper Address (optional), and Master Block StatusID (optional). These parameters need not be refreshed each time the facility is used. After that, the VME Starting Address is written, then Local Starting Address and GO. This last register write operation starts the block transfer. The CY7C961 will assert BR* and, upon being granted the VMEbus, move data until the transfer length is exhausted.

Note that the CY7C961 can be programmed through the initialization bit stream to release and re-request the VMEbus at every 256-byte boundary as DMA transfer progresses. This

behavior can be useful in providing VMEbus access to high-priority traffic interleaved with the DMA transaction.

If more than one VMEbus master is to control the block transfer facility, or if the facility is dual-ported between a local and one or more VMEbus masters, the Semaphore Test & Set register can be polled by any master to determine if the facility is “idle” or “busy.” If the polling result is “idle”, that register read sets the semaphore to “busy”, eliminating the need for a read-modify operation. The semaphore can be written to “idle.” The semaphore is changed by the register write, but this operation is always acknowledged with an error response by the CY7C961.

The semaphore is reset by the CY7C961 on block transfer completion. To do another block transfer requires a minimum of three register accesses, namely, (1) read Semaphore Test & Set, (2) write VME Starting Address, and (3) write Local Starting Address and GO. Of course any of the set-up parameters can be read (with some limitations) or written before writing the GO register. Once the Local Starting Address and GO register is written, all register write attempts are greeted with error acknowledge, cancellation of the DMA operation, and reset of the semaphore. Register reads are always permitted.

The CY7C961 is equipped with BERR* and LBERR* inputs which can be used to interrupt a DMA transfer at any time after it is started (BBSY* is asserted). No consideration as to the timing of BERR* or LBERR* assertion with respect to the progress of the DMA needs to be given. These signals provide a mechanism for aborting DMA transfers in favor of higher priority VMEbus traffic.

Table 3-11. Master Block Transfer Control Registers

LA[4:0]	Register Function
000xx	Semaphore Test & Set (read only)
001xx	Transfer Length Multiplier 0 (TLM0)
010xx	Transfer Length Multiplier 1 (TLM1)
011xx	Transfer Type
100xx	Local Starting Address and GO
101xx	VME Starting Address
110xx	A40/A64 Upper Address
111xx	Master Block StatusID & Interrupt Enable

3.11.3.5 Register Definitions

The following descriptions apply equally to VMEbus register and local register access, but are written from the VMEbus perspective. Local access occurs in the context of “local bus holdoff” as previously described, and has as its error response LBERR* substituted for BERR*.

3.11.3.5.1 Semaphore Test & Set (read only)

Location 000xx is a semaphore test and set. It is a read cycle. CY7C961 responds with the current value of the semaphore. If the Master Block resource of the CY7C961 is idle, this read will return logic “1” on LD0 which is reflected on D0 of the VME data bus. This register read causes the semaphore to be set to busy. The CY7C961 will now respond to additional register accesses. Any subsequent read of the semaphore will return logic “0” on LD[0] and D[0] indicating that the block transfer resource is “busy.”

The semaphore is cleared by the CY7C961 upon completion of a block transfer or when an illegal combination of register values is held when a transfer is attempted. If the interrupt on completion feature is enabled, semaphore clear is delayed until the interrupt generated by block transfer completion is serviced. When BERR* is received during a block transfer, the transfer terminates and the semaphore is cleared. If the interrupt on completion feature is enabled, an interrupt will be signaled on termination and the semaphore clear is delayed until that interrupt is serviced.

Any write to location 000xx will cause the CY7C961 to drive BERR* and clear the semaphore. If access to any of the seven other defined registers is attempted before the semaphore is in the “busy” state, CY7C961 will BERR* those attempts and the semaphore will remain in the idle state.

The CY7C961 drives status conditions on LD[6:1] passed through to D[6:1] of the VMEbus during semaphore read. The normal value for these status bits is logic “1.” A logic “0” on any bit indicates a condition which would prevent a block transfer from starting (cause CY7C961 to BERR a write to Local Starting Address and GO). *Table 3-12* summarizes the semaphore Test and Set Status Bit functions.

Table 3-12. Semaphore Test and Set Status Bits

Status Bit	Error Indication
bit 0	Semaphore. (Logic “1” if idle)
bit 1	Master Block interrupt is pending.
bit 2	Transfer Length Multiplier registers are 0.
bit 3	Transfer Type is undefined.
bit 4	Data size is incompatible with VME or local starting address.
bit 5	Address alignment violated on multiplexed data BLT.
bit 6	VME starting address has not been updated.
bit 7	BERR* or LBERR* asserted during transaction.

A logic “0” on bit 1 means that interrupt on completion is enabled and a block transfer has terminated or completed, but the interrupt signalled on termination/completion has not yet been serviced by a VMEbus interrupt handler. Bit 0 is always logic “0” in this case since the

semaphore must be in state “busy.” Any attempt to access other registers of the block transfer facility while bit 1 is logic “0” will be BERRed by the CY7C961. This status bit will clear with the pending interrupt.

A logic “0” on bit 2 indicates that both registers Transfer Length Multiplier1 and Transfer Length Multiplier0 contain values of zero. The transfer length multiplier must be nonzero. To clear this status bit, a nonzero value must be written to either Transfer Length Multiplier register.

A logic “0” on bit 3 indicates the value in the transfer type register is undefined. To clear this condition, a valid code must be written to the Transfer Type register. (See Transfer Type below.)

A logic “0” on bit 4 means that either the local starting address or VME starting address is not compatible with the programmed data size. This error condition amounts to specifying unaligned starting addresses for a block transfer. For example, a local starting address with an LSB of 0x03 specified with a data size of D16 would result in an unaligned transfer request. Similarly, a VME starting address LSB of 0x02 specified for a D32 operation would start at an unaligned address. Either condition would cause this status bit to be logic “0.” This status bit is computed on the current values of VMEbus and local starting addresses. Since the Local Starting Address and GO register is written at the time a block transfer is started, an alignment error on the local starting address will always cause CY7C961 to assert BERR*. A subsequent read of the Semaphore Test and Set will indicate the error on this status bit.

A logic “0” on bit 5 indicates violation of a restriction placed on starting address when the transaction type is multiplexed data. The restriction is that VME starting address [7:0] be equal to local starting address [7:0]. This restriction is in addition to the address alignment requirement described with respect to status bit 4 above. The restriction applies expressly to transfer type codes: “0110011x”, “0110111x”, “0111010x”, “0101011x”, “0101111x”, and “0111111x”. This status bit is computed on the current values of VMEbus and local starting addresses. Since the Local Starting Address and GO register is written at the time a block transfer is started, a violation of this restriction may be created when a new local starting address is written causing CY7C961 to assert BERR*. A subsequent read of the Semaphore Test and Set will indicate the error on this status bit.

A logic “0” on bit 6 indicates that the VME starting address has not been written since the last master block transfer was started. This status bit is cleared by writing a new VME Starting Address.

3.11.3.5.2 Transfer Length Multiplier(1 & 0)

Two registers, Transfer Length Multiplier1 and Transfer Length Multiplier0 hold respectively the MSB and LSB of a 16-bit transfer length multiplier parameter. The transfer length of a block transfer in bytes can be computed by multiplying the transfer length multiplier by a block length factor for that transfer type. The block length factor is a function of the transfer data size as shown in *Table 3-13*.

Table 3-13. Transfer Length Calculation

Data Size	Block Length Factor	TLM1,TLM0	Transfer Length
D8	16 Bytes	X TLM[15:0]	= BYTES
D16	32 Bytes	X TLM[15:0]	= BYTES
D32	64 Bytes	X TLM[15:0]	= BYTES
D64	128 Bytes	X TLM[15:0]	= BYTES

For example, if TLM1= 4 and TLM0 = 6 and the data size is D32 (specified in the Transfer Type register), then the transfer length for the block transfer would be 64 X 1030 = 65,920 bytes. The number of data cycles is always 16 times the transfer length multiplier, so the data bytes transferred is obviously proportional to the width of the transaction. The transfer length multiplier registers are read/write with data sent/received on LD[7:0] and reflected on D[7:0] of the VMEbus. Bit 2 of the Semaphore Test and Set status word will be logic “0” if both TLM1and TLM0 are set to 0. This is an error condition which will block a transfer start. Both registers are cleared to 0 after power up or SYSRESET*.

3.11.3.5.3 Transfer Type

The Transfer type register specifies which of 22 possible block transfer operations is to be performed. The user can select from among A40BLT, D64MBLT, D32BLT, D16BLT, D8BLT, and single cycle block move options. The 8-bit hex codes shown in *Table 3-14* are valid contents of the transfer type register:

Table 3-14. Transfer Type Field—Block Transfer Operations

Transfer Type[7:0]	AmCode	Block Transfer Description
20,21,22,23,24,25	3F	A24 supervisory block transfer (D8BLT or D16BLT or D32BLT)
A8,A9,AA,AB,AC,AD	3E	A24 supervisory program access (D8 or D16 or D32)
B0,B1,B2,B3,B4,B5	3D	A24 supervisory data access (D8 or D16 or D32)
66,67	3C	A24 supervisory 64-bit block transfer (MBLT)
28,29,2A,2B,2C,2D	3B	A24 nonprivileged block transfer (D8BLT or D16BLT or D32BLT)
B8,B9,BA,BB,BC,BD	3A	A24 nonprivileged program access (D8 or D16 or D32)
A0,A1,A2,A3,A4,A5	39	A24 nonprivileged data access (D8 or D16 or D32)
6E,6F	38	A24 nonprivileged 64-bit block transfer (MBLT)
40,41,42,43	37	A40 block transfer (D8BLT or D16BLT)
74,75	37	A40 32-bit block transfer (MD32)
90,91,92,93,94,95	2D	A16 supervisory access (D8 or D16 or D32)
98,99,9A,9B,9C,9D	29	A16 nonprivileged access (D8 or D16 or D32)

Table 3-14. Transfer Type Field—Block Transfer Operations (continued)

Transfer Type[7:0]	AmCode	Block Transfer Description
10,11,12,13,14,15	0F	A32 supervisory block transfer (D8BLT or D16BLT or D32BLT)
C0,C1,C2,C3,C4,C5	0E	A32 supervisory program access (D8 or D16 or D32)
C8,C9,CA,CB,CC,CD	0D	A32 supervisory data access (D8 or D16 or D32)
56,57	0C	A32 supervisory 64-bit block transfer (MBLT)
18,19,1A,1B,1C,1D	0B	A32 nonprivileged block transfer (D8BLT or D16BLT or D32BLT)
D0,D1,D2,D3,D4,D5	0A	A32 nonprivileged program access (D8 or D16 or D32)
D8,D9,DA,DB,DC,DD	09	A32 nonprivileged data access (D8 or D16 or D32)
5E,5F	08	A32 nonprivileged 64-bit block transfer (MBLT)
48,49,4A,4B,4C,4D	03	A64 block transfer (D8BLT or D16BLT or D32BLT)
7E,7F	00	A64 64-bit block transfer (MBLT)

Bits [2:1] of the Transfer Type register specify the data size of the block transfer. *Table 3-15* defines this Transfer Type field. The CY7C961 checks for consistency between transfer type and data size. Illogical combinations will be reported on Semaphore Test and Set status bit 3 as an undefined transfer type and will prevent the start of a block transfer.

Table 3-15. Transfer Type Field—Data Size

Transfer Type [2:1]	Block Transfer Data Size
00	D8
01	D16
10	D32
11	D64

Bit 0 of the Transfer Type register specifies block transfer data direction. Logic “1” specifies master read with data moving to the CY7C961 interface, logic “0” specifies master write with data moving from the CY7C961interface to a VMEbus slave.

The transfer type register is read or written with data transmitted on LD[7:0] and reflected on D[7:0] of the VMEbus. Illegal codes cause block transfer to abort when the GO transaction is received. Both undefined transfer type fields and illogical transfer type data size combinations will BERR when Local Starting Address and GO is written.

3.11.3.5.4 Local Starting Address & GO

The Local starting address and GO register is used to write the local block address to be used in the DMA block transfer. This address can be up to 32 bits of address information provided as VME data when the register is written. Local starting address is loaded directly

into the CY7C964's local address counters C1 and the CY7C961 local address register. A read of the local starting address will yield only the LSB of the starting address on LD[7:0] reflected on D[7:0] of the VME data bus.

The CY7C961 signal BLT* is driven low for 2 CLK periods to load the local address into the CY7C964s from the LD bus. The value of LD on the rising edge of BLT* is stored in the CY7C964 counter C1. Refer to *Figure 3-38*.

The control function of this register is activated only for writes to the register. GO causes the start of the block transfer by signaling the CY7C961 requester to take the VMEbus and begin the transfer. Once CY7C961 acquires the VMEbus, it will not release the VMEbus until the block transfer finishes or is terminated by a BERR* during a transfer attempt. BBSY* is released when the transfer length count is exhausted, not at VMEbus boundary crossings. Any

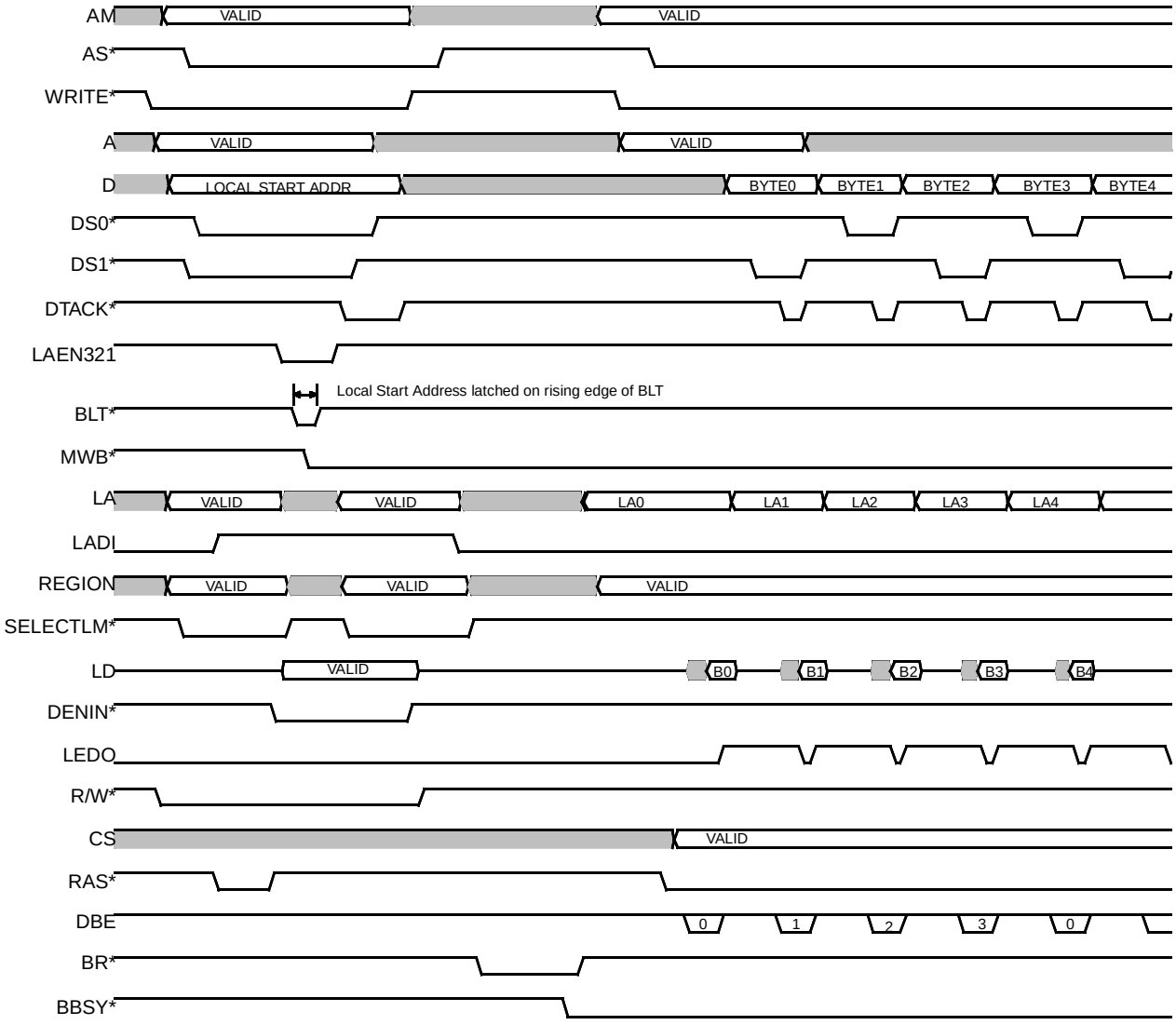


Figure 3-38. Local Starting Address and Go, Block Transfer Start Timing

write to a set-up register after the GO transaction will be BERRed by CY7C961, the block move aborted, and the semaphore reset.

3.11.3.5.5 VME Starting Address

The VME Starting Address register is used to write the VMEbus block address to be used in the DMA block transfer. This address can be up to 32 bits of address information written as VME data to the CY7C961. It is actually stored in latch L9 of the CY7C964s and the LSB of the address is also captured by the CY7C961 to facilitate counting to the VME address boundary. A read of the VMEbus Starting Address will yield only the LSB of the starting address on LD[7:0] reflected on D[7:0] of the VME data bus. Note that this specifier is a true byte address. The CY7C961 takes care of VMEbus LWORD* and DS1*/DS0* encoding. This register must be written each time a block transfer is executed. Bit 6 of the Semaphore Test and Set status register will indicate when the VME starting address has not been updated. This requirement guarantees that the CY7C964 and CY7C961 will be using the same VMEbus starting address.

3.11.3.5.6 A40/A64 Upper Address

A40/A64 Upper Address register is not a register residing in either the CY7C961 or CY7C964. Instead, it is a facility for reading/writing extended address information to external hardware on the slave card that is not part of the CY7C961/CY7C964 interface. Latch and enable signaling allows for reading and writing during register access as well as data enable during address broadcast as the DMA block transfer runs. A40 and A64 extensions are supported.

The latch signal for capturing extended address from the local data bus is (SELECTLM* & LEDI) where LEDI is a CY7C961 output. The active Low enable for driving the latched data onto the local data bus is (LDEN* | MWB*). If the extended address is to be incremented, VCOUT* of the appropriate CY7C964 and LADO are used as count enable and clock respectively for extended address counting. Refer to *Figure 3-39* for signaling.

The latch signal is designed to latch up to 32 bits of upper address from LD[31:0] passed through the interface from VME D[31:0] when Upper Address is written. The enable signal is designed to allow the stored address to be read from the VMEbus when Upper Data is read, as well as to enable the stored address onto LD when the master block transfer address broadcast requires it.

3.11.3.5.7 Master Block StatusID & Interrupt Enable

Master Block status ID & Interrupt Enable is a CY7C961 register which holds the Master Block statusID byte. This register can be read or written. Bits 7–1 of the register will be reflected in the statusID byte sourced by CY7C961 when the Master Block interrupt is serviced. Bit 0 controls enabling of interrupt on completion: logic “1” enables interrupt on completion.

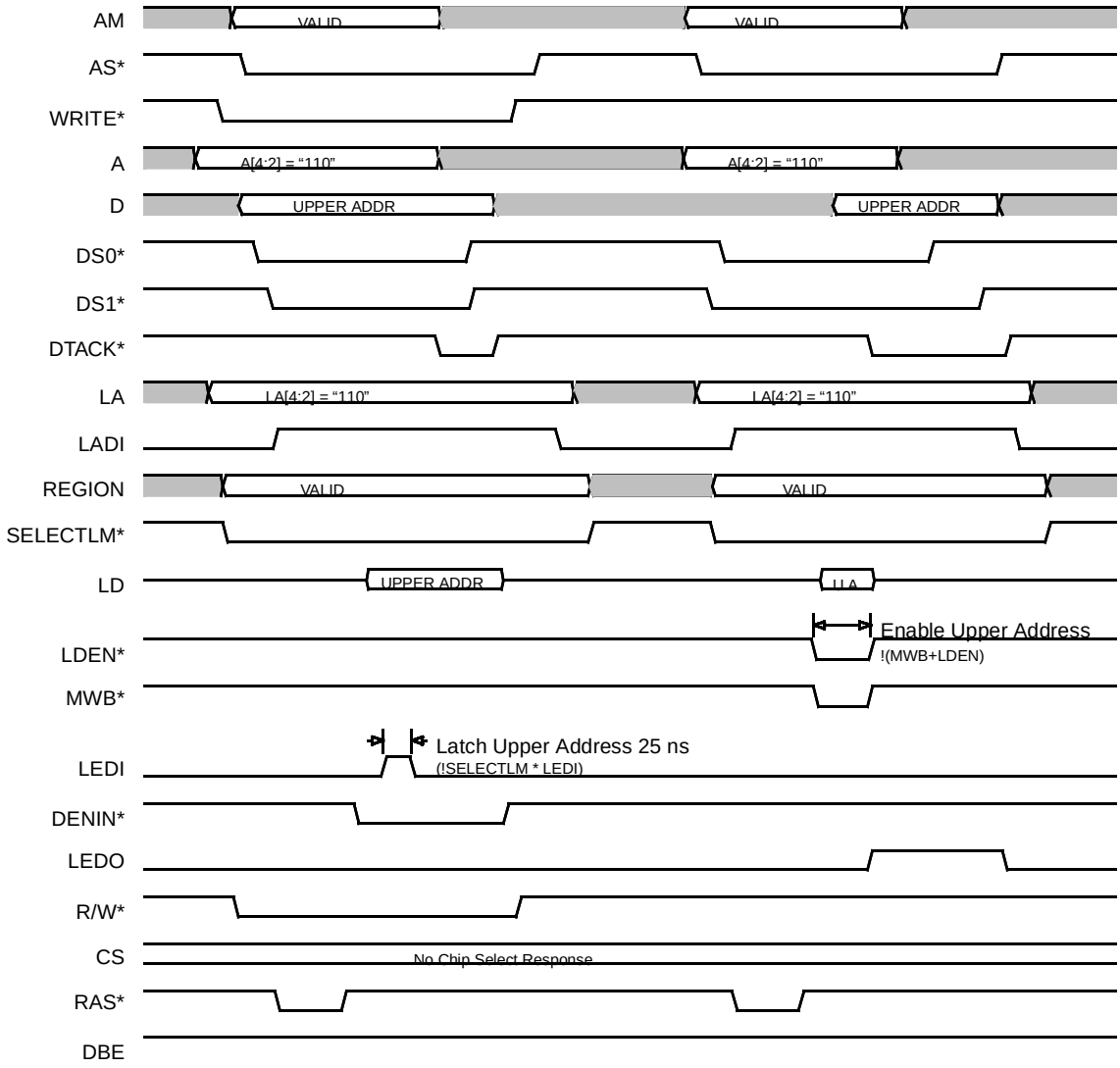


Figure 3-39. Upper Address Register Access Timing

When the interrupter status is read by the handler, bit 0 of the statusID indicates completion status. Logic “1” means normal completion. Logic “0” means abnormal termination (BERR* or LBERR* was received during block transfer). The interrupter is a Release on Acknowledge (ROAK) interrupter. If interrupt on completion is enabled, CY7C961 will reset its semaphore on interrupt acknowledge. If interrupt on completion is not enabled, CY7C961 will reset its semaphore with BBSY deassertion at the end of the DMA block transfer. Refer to *Figure 3-40* for Interrupter timing.

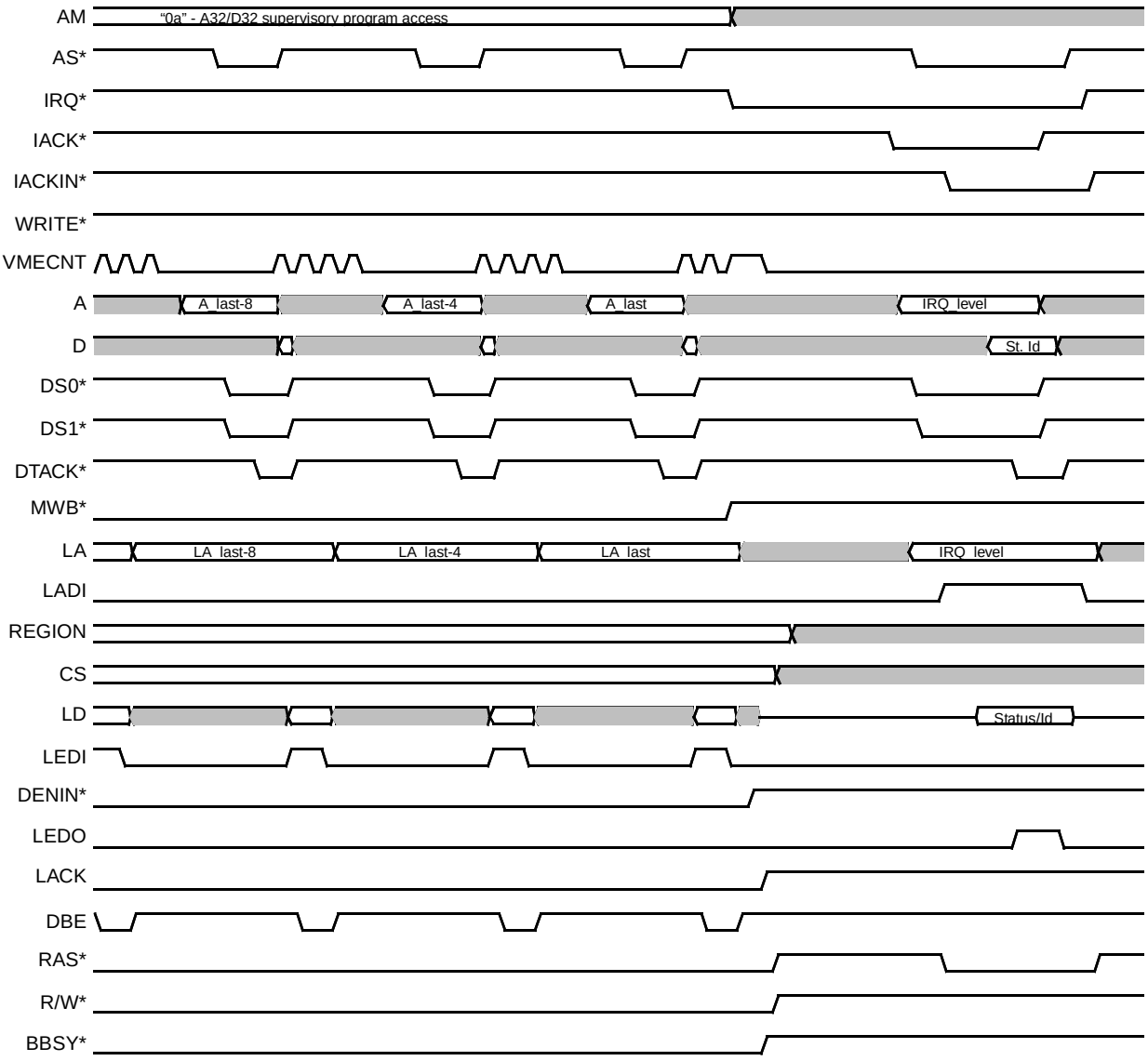
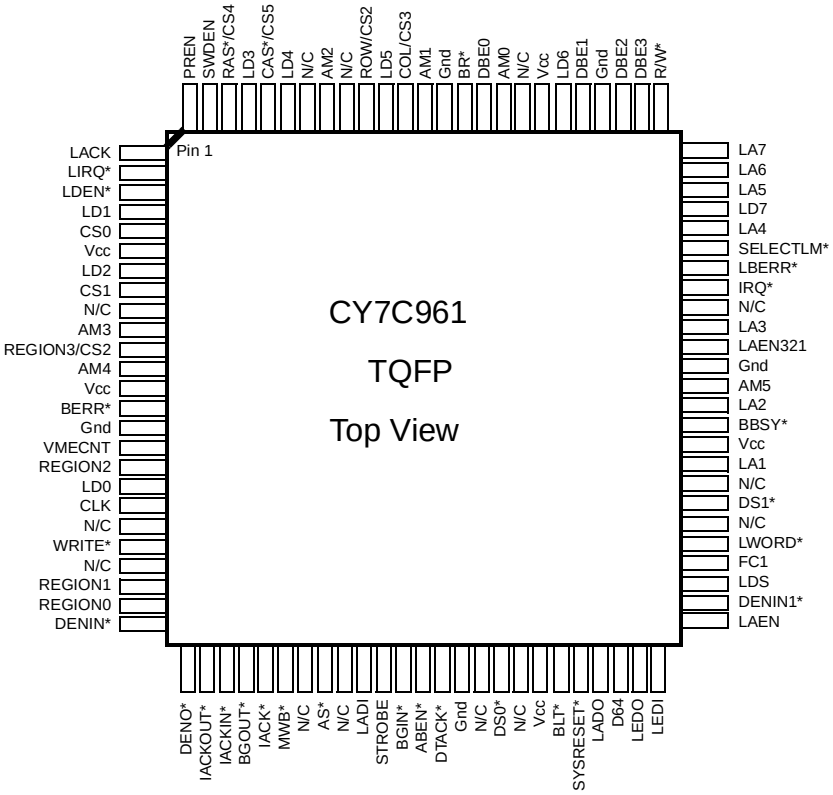


Figure 3-40. Interrupter Timing

3.11.4 Pin Description Addendum



The following signals are either additional to, or redefined from, the CY7C960 pin descriptions in Chapter 3.3.

3.11.4.1 VMEbus Signals

AM[5:0] - VMEbus Address Modifier

Input: Yes
Output: Yes
Drive: 48 mA

Signals AM[5:0] are the VMEbus Address Modifier I/Os. When inputs, these signals are used to decode the VMEbus data transaction type. The CY7C961 provides support for both pre-defined and user-defined VMEbus Address Modifiers. During master block transfers VMEbus AM codes are driven out on these signal pins.

AS* - VMEbus Address Strobe

Input: Yes
Output: Yes
Drive: 64 mA
Active: Low

Address Strobe is the VMEbus signal that informs VMEbus slaves that a valid address is on the VMEbus. This signal is used by the CY7C961 to qualify the VMEbus Address Modifiers AM[5:0] and REGION[3:0] inputs to determine if a valid slave cycle should be performed. During master block transfers VMEbus AS* is driven out on this signal pin.

DS0*,DS1* - VMEbus Data Strobes

Input: Yes
Output: Yes
Drive: 64 mA
Active: Low

DS0* and DS1* are the VMEbus Data Strobes. As inputs, these signals inform the CY7C961 that the data phase of the VMEbus cycle has begun. These signals in conjunction with the VMEbus LWORD* (connected to LA[0]) signal encode the data transfer width or number of bytes, 1 through 4. This information is necessary to enable the appropriate CY7C964 data bytes. During master block transfers VMEbus DS0* and DS1* are driven out on these signal pins.

WRITE* - VMEbus Write Line

Input: Yes
Output: Yes
Drive: 48 mA
Active: Low

The VMEbus WRITE* line specifies the data direction of the VMEbus data cycle in progress. If this signal is asserted then a VMEbus WRITE operation is in progress. During such a transaction, if the VMEbus address decodes properly, the CY7C961 responds by asserting the local R/W* signal and performing the appropriate local cycle. During master block transfers, VMEbus WRITE* is driven out on this signal pin indicating the direction of data for the DMA transfer.

DTACK* - VMEbus Data Acknowledge

Input: Yes
Output: Yes
Drive: 64 mA
Active: Low

The DTACK* signal is asserted by the CY7C961 when a valid VMEbus transaction is in progress and has remained valid for the proper length of time. The assertion of this signal informs the VMEbus Master that the slave has either accepted the data during write operations or has sourced the data during read operations. This signal is a rescinding output. During master block transfers, the CY7C961 receives VMEbus data acknowledge from the target slave.

BR* - VMEbus Request

Input: No
Output: Yes
Drive: 48 mA
Active: Low

Signal BR* is the VMEbus Request output. This output is open-collector with a low state sink current of 48 mA. It is asserted by the CY7C961 when the VMEbus is required for a block transfer.

BGIN* - VMEbus Bus Grant In

Input: Yes
Output: No
Active: Low

BGIN* is a VMEbus Bus Grant In signal. It is generated by the VMEbus arbiter and signals that CY7C961 may use the VMEbus. BGIN* and BGOUT* signals form a bus grant daisy chain.

BGOUT* - VMEbus Bus Grant Out

Input: No
Output: Yes
Drive: 8 mA
Active: Low

BGOUT* is a VMEbus Bus Grant Out signal. It is driven Low by the CY7C961 in response to an assertion of the BGIN* signal when CY7C961 does not want to use the VMEbus.

BBSY* - VMEbus Bus Busy

Input: Yes
Output: Yes
Drive: 64 mA
Active: Low

BBSY* is the VMEbus bus busy signal. It is driven Low by the CY7C961 to indicate the VMEbus is being used for a block transfer. When the block transfer operation is done, CY7C961 drives BBSY* High and then three-states the signal. In response to decoded VMEbus lock cycles, the CY7C961 monitors BBSY* to determine when a locked VMEbus sequence is ending.

BERR* - VMEbus Bus Error

Input: Yes
Output: Yes
Drive: 48 mA
Active: Low

BERR* is the VMEbus bus error signal. It is driven Low by CY7C961 in two cases. First, decoded slave accesses which attempt to illegally configure the CY7C961 block transfer facility will result in a BERR* acknowledge. Second, BERR* will be signaled to acknowledge a slave block data cycle if LBERR* is sampled asserted at the time VMEbus DSA* was received

by the CY7C961. BERR* is three-stated on the deassertion of VMEbus signals DS0* and DS1*. CY7C961 monitors BERR* during master block transfer operation, truncating the block transfer if a BERR* acknowledge is detected. BERR* assertion will also signal the end of any locked VMEbus sequence.

3.11.4.2 Local Buffer Control Signals

BLT*

Input: No
Output: Yes
Drive: 8 mA
Active: Low

BLT* is driven Low for 2 clock periods during the block transfer register access “Local starting address and GO.” The local starting address for block transfer is latched into CY7C964 internal counters on the High-going edge of this signal. The CY7C961 BLT* pin should be connected to the BLT* pin of each CY7C964 in the interface.

MWB*

Input: No
Output: Yes
Drive: 8 mA
Active: Low

MWB* is driven Low by CY7C961 during the block transfer register access “Local starting address and GO.” It remains in the Low state throughout master block transfer except at 256-byte local boundaries. When a 256-byte local boundary is crossed, MWB* will be pulsed High for one clock period. The High to Low transition on this pulse increments the interface CY7C964 local block counters. MWB* is deasserted on completion of a block transfer. MWB* also signals VMEbus lock status. MWB* is driven Low on decode of a VMEbus lock cycle. It is deasserted when the VMEbus lock sequence is complete. The CY7C961 MWB* pin should be connected to the MWB* pin of each CY7C964 in the interface.

LADO

Input: No
Output: Yes
Drive: 8 mA

LADO is pulsed High by CY7C961 at VMEbus 256-byte boundaries during master block transfers. LADO is driven High at the same time VMEbus data strobes are asserted for the last cycle of the address page. LADO is driven Low one clock after DTACK* is detected Low in that cycle. The High to Low transition on LADO increments the VMEbus block counters of the interface CY7C964s. The CY7C961 LADO pin should be connected to the LADO pins of all interface CY7C964s EXCEPT the one connected to VMEbus addresses A[7:1].

FC1

Input: No
Output: Yes
Drive: 8 mA
Active: High

FC1 is driven High by CY7C961 at the beginning of the block transfer's VMEbus tenure. It is deasserted at the end of the block transfer's VMEbus tenure. The CY7C961 FC1 pin should be connected to the FC1 pins of each CY7C964 in the interface. This signal can serve as a DMA complete indicator for local control purposes.

VMECNT

Input: No
Output: Yes
Drive: 8 mA

VMECNT is a clock signal driven by CY7C961 to the LADO pin of the CY7C964 connected to VMEbus addresses A[7:1]. This signal adjusts the VMEbus block counter of this CY7C964 to source LWORD correctly for D8 and D16 block transfers. VMECNT also burst clocks this block counter to $h00 in cases where the VMEbus block starting address is not aligned to a 256-byte boundary. The clock pulse width is one CLK period.

LAEN321

Input: No
Output: Yes
Drive: 8 mA
Active: High

LAEN321 is driven Low for four CLK periods during the block transfer register access "Local starting address and GO." The High to Low transition happens one clock before the High to Low transition on BLT*. LAEN321 is used in conjunction with BLT* and MWB* to place the interface CY7C964s in BLT_STATE. The CY7C961 LAEN321 pin should be connected to the LAEN pins of all interface CY7C964s EXCEPT the one connected to VMEbus addresses A[7:1].

3.11.4.3 Local Signals

LD[7:0] - Local Data Signals

Input: Yes
Output: Yes
Drive: 8 mA

LD[7:0] make up the local bidirectional data bus. These pins should be connected to LD[7:0] of the interface CY7C964 connected to VMEbus data D[7:0]. During block transfer register accesses and block transfer complete interrupt acknowledge accesses, data is read from or written to the CY7C961 via this data port.

SELECTLM* - Select Load Master Signal

Input: Yes
Output: No
Active: Low

SELECTLM* is a chip select for accessing the eight CY7C961 registers that control the block transfer facility. A decoded VMEbus slave cycle for which SELECTLM* is also asserted will be interpreted by CY7C961 as a block transfer register access. SELECTLM* timing should imitate that of the REGION input.

LBERR* - Local Bus Error Signal

Input: Yes
Output: Yes
Drive: 8 mA
Active: Low

LBERR* has three distinct functions. First, it provides a mechanism for aborting DMA transfers in progress. At any time after the GO register access completes, LBERR* may be asserted to cause DMA completion (with error status). If interrupt on completion is enabled, IRQ* will be asserted within two CLK periods after LBERR* assertion. A second function of LBERR* is as an error signaling mechanism for slave block transfer. Each time the VMEbus DSA* signal is asserted to the CY7C961, the state of the LBERR* signal is sampled. If it is asserted, that VMEbus cycle will receive a BERR* acknowledge instead of a DTACK* acknowledge. The third function of LBERR* is as an error acknowledge output for DMA register accesses during “local bus holdoff.” LBERR* will be asserted (driven low) by CY7C961 to indicate an error condition during a register access in progress. Note that LBERR* is always driving out of the CY7C961 during “local bus holdoff.”

R/W* - Read/Write

Input: Yes
Output: Yes
Drive: 8 mA

R/W* is the local signal that determines if the cycle in progress is a read operation or a write operation. The CY7C961 asserts this signal Low during write operations.

When accessing the Master Block Facility from the local side of the interface - R/W* is an input to the CY7C961. The local bus holdoff feature of the CY7C961 must be enabled before register access is initiated.

3.11.4.4 Master Block Transfer Performance

From the perspective of local bus timing, there is no difference between local bus signaling for master DMA and slave accesses. The master block facility uses the same circuits and the same self-timed acknowledge constants as the slave. Master block transfers sample REGION

at each 256-byte boundary. This allows a block transfer to, for example, start in an SRAM region and transparently cross a hardware boundary into a DRAM region.

The master block facility has performance limits set by CY7C961 internal synchronous state machines. 6 CLK periods is the minimum required per VMEbus data cycle for master write operations, 7 CLK periods for master read VMEbus data cycles. Local bus signaling limits D64 transfer rates to 80 megabytes per second, and MD32 to 40 megabytes/second. The performance of D32 block transfer is limited by VMEbus slave response, with an ideal slave response making block read performance of 40 megabytes/second and block write performance of 50 megabytes per second possible. These are, of course, burst maximums. Sustained D64 block transfer rate will be 73 megabytes/second to an ideal slave with local cycles set up for 50 ns.

The block transfers using single-cycle protocol will be considerably slower than the true block protocols because the CY7C961 must increment the CY7C964 after each data cycle. For D32 this represents an 8 CLK period overhead. Transfer rate will not exceed 20 megabytes/second. D16 single-cycle protocol will be down around 12 megabytes/second. D8 single-cycle will top out at 5.5 megabytes/second.

For BLT transfers, CY7C961 will cross VME address boundaries without releasing the VMEbus unless the Interleave function is programmed ON in the serial bit stream. Address strobe will be cycled and address rebroadcast. BBSY does not have an early release mode, and is asserted throughout the block transfer. D64MBLT will rebroadcast at 256-byte boundaries. If single-cycle accesses are specified, the block move will consist of an indivisible packet of single-cycle transfers sufficient to satisfy the transfer length parameter, transfer type dictating the AM code used.

3.11.5 Examples of Block Transfers

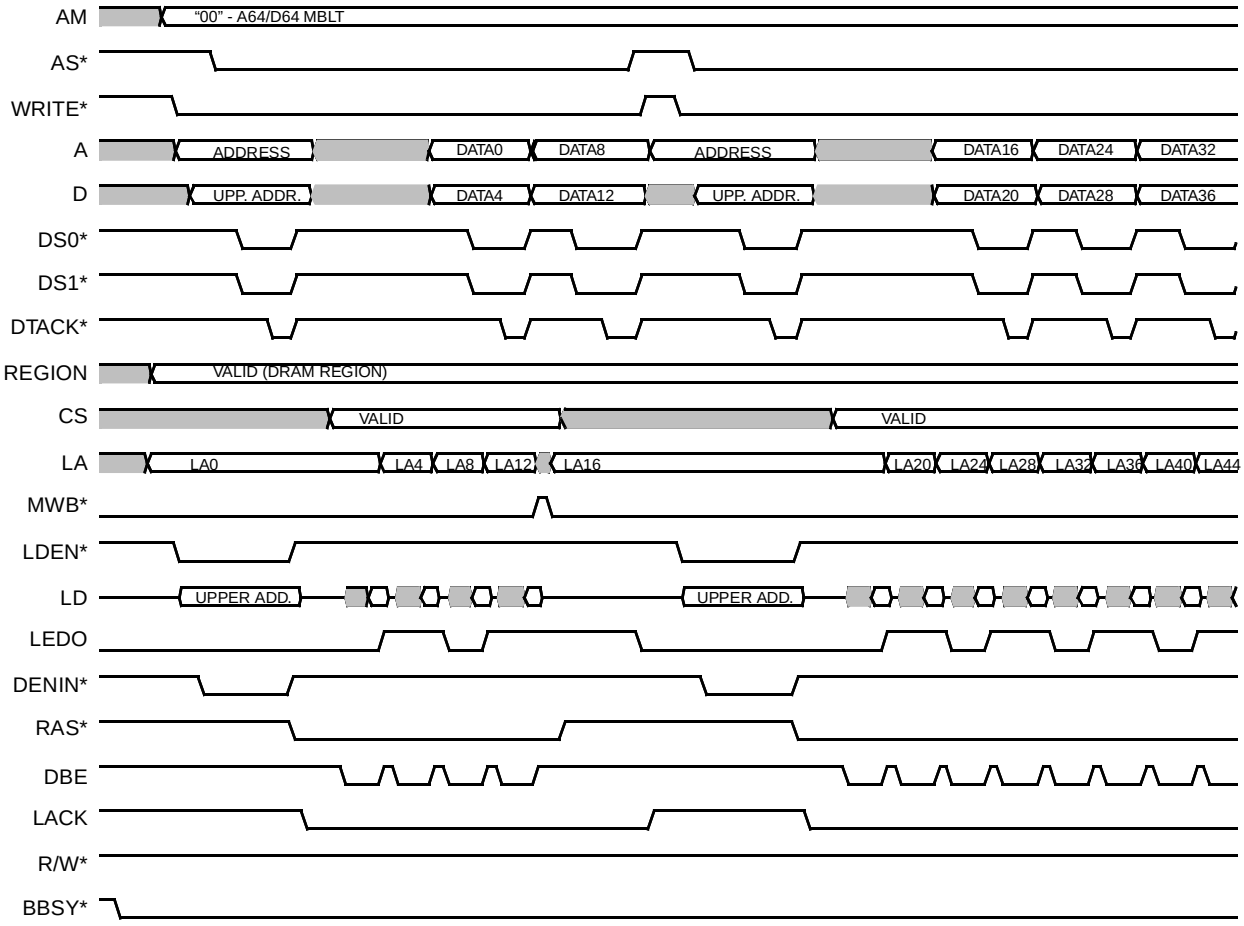


Figure 3-41. A64/D64 MBLT Master Write

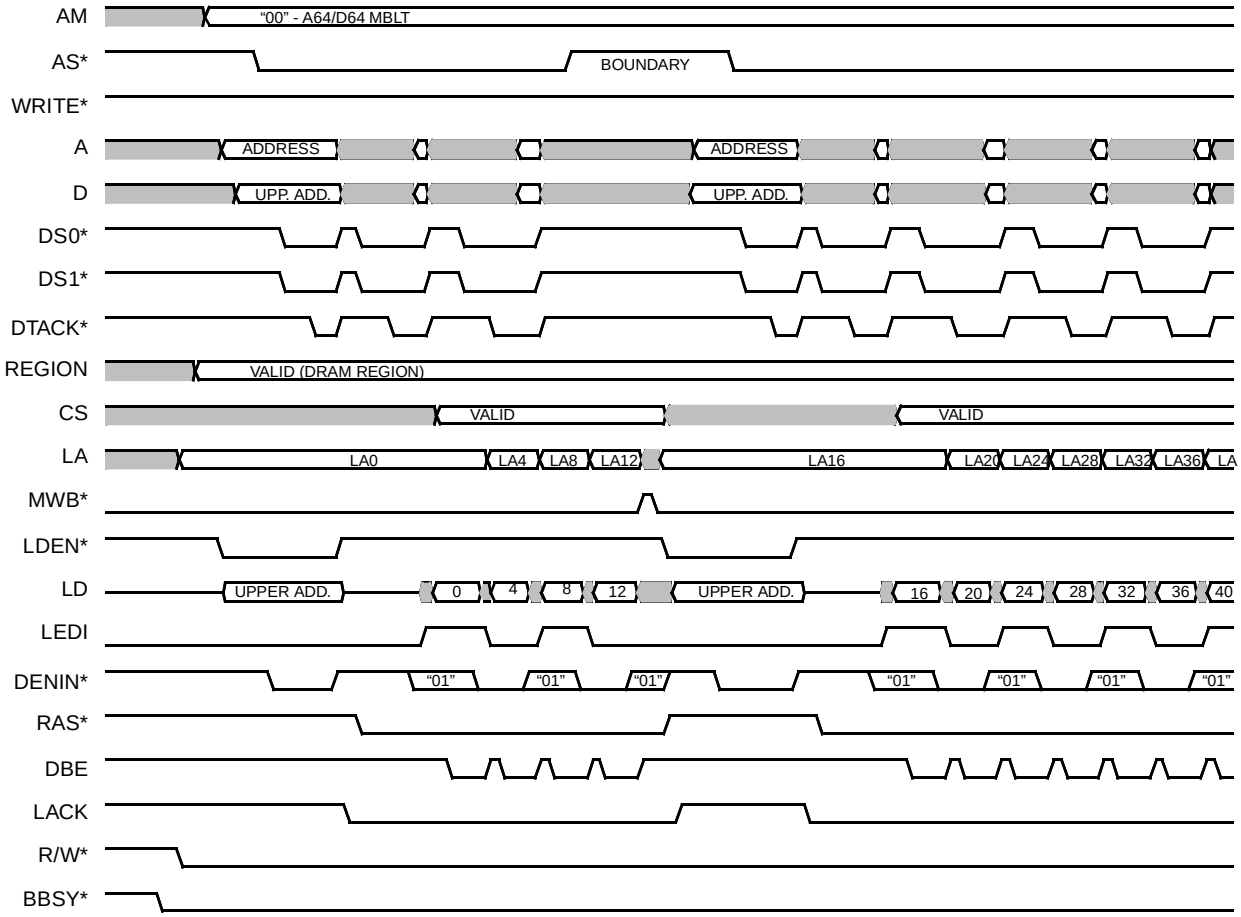


Figure 3-42. CY7C961 A64/D64 MBLT Master Read Example

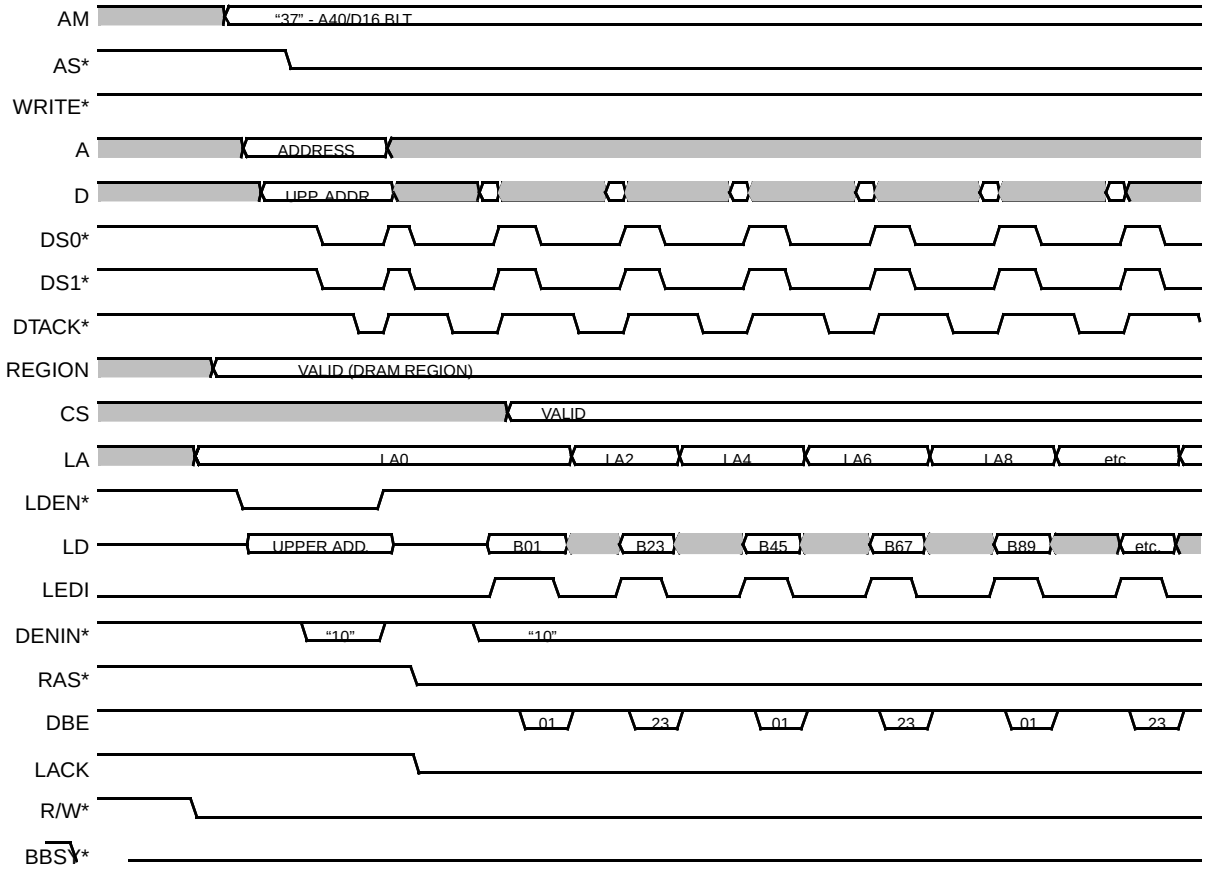


Figure 3-43. CY7C961 A40/D16 BLT Master Read Example

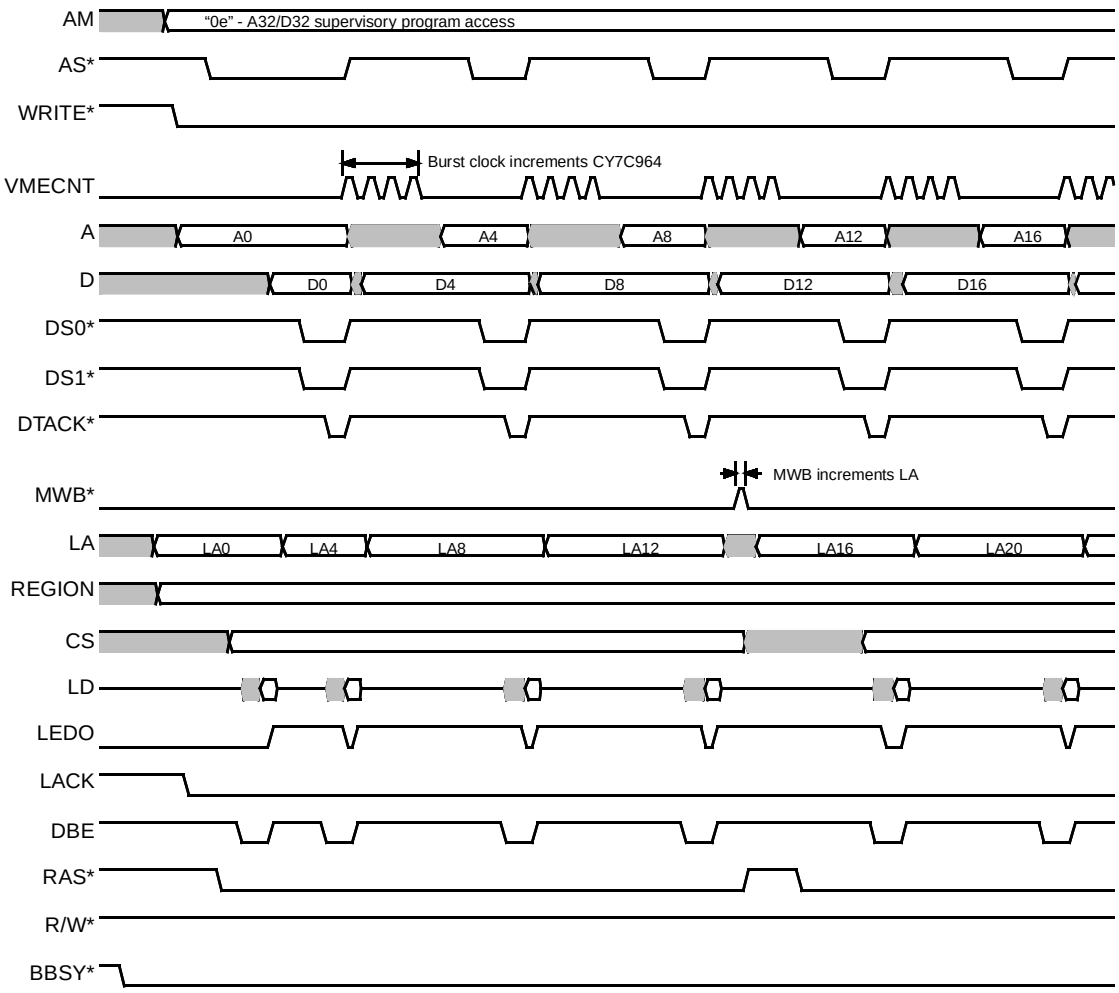
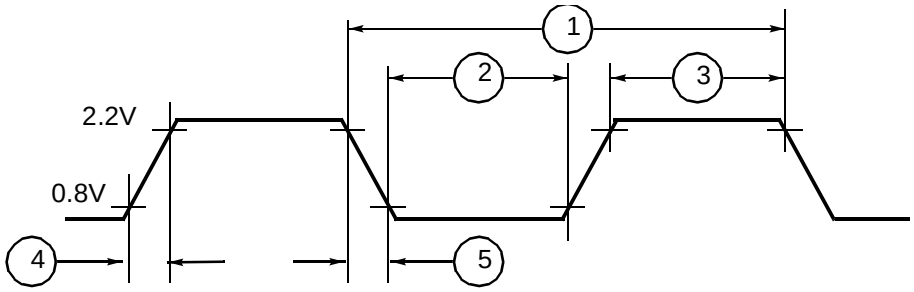


Figure 3-44. A32/D32 Single Cycle Write Example

3.12
AC Parameters

This addendum contains waveforms that describe the behavior of the CY7C960/961/ CY7C964 interface for a variety of transaction types and timings. A table of AC parameters is referenced to the waveforms. Note that “T” refers to the CLK input period. Note also that parameters are not indicated on the waveforms everywhere they apply. These waveforms were captured during in-system simulation and reflect accurately the behavior that can be expected.

Clock Input



Num.	Characteristic	Min.	Max.
	Frequency of Operation (MHz)	30	80
1	Cycle Time (ns)	12.5	33.3
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.

Table 3-16. AC Parameters

Parameter	Description	Min	Max	Reference	Notes
p1	DECODE Delay	2T	5T		2
p2	RAS*/CAS* Delay	2T	9T		2
p3	CAS* Width/DBE Width	3T	18T		2, 7
p4	CAS* Precharge	1T	8T		2
p5	RAS* Precharge	5T	12T		2
t2	LD to D		19 ns	Waveform 5	3
t3	A to LA		20 ns	Waveform 5	3
t4	D to LD		22 ns	Waveform 20	3

Table 3-16. AC Parameters (continued)

Parameter	Description	Min	Max	Reference	Notes
t5	LD to A Propagation Delay		22 ns	Waveform 11	3
t7	ABEN* falling to A Output Enable Delay		12 ns	Waveform 11	3
t8	DENO* falling to D Output Enable Delay		16 ns	Waveform 5	3
t9	ABEN* falling to D Output Enable Delay		17 ns	Waveform 11	3
t11	LAEN rising to LA		12 ns	Waveform 18	3
t12	DENIN* falling to LD Output Enable Delay		18 ns	Waveform 9	3
t13	DENIN1* falling to LD Output Enable Delay		21 ns	Waveform 23	3
t15	ABEN* rising to A High-Z		12 ns	Waveform 22	3
t16	DENO* rising to D High-Z		15 ns	Waveform 3	3
t17	ABEN* rising to D High-Z		15 ns	Waveform 22	3
t19	LAEN falling to LA High-Z		15 ns	Waveform 18	3
t20	DENIN* rising to LD High-Z		18 ns	Waveform 9	3
t23	A valid to VCOMP* falling		21 ns	Waveform 3	3
t25	LD set-up to LEDO rising	7 ns		Waveform 22	3
t26	LD hold to LEDO rising	0 ns		Waveform 22	3
t28	LD set-up to DENO* falling	0 ns		Waveform 22	3
t29	LD hold to DENO* falling	7 ns		Waveform 22	3
t37	A, D set-up to LEDI rising	7 ns		Waveform 23	3
t38	A, D hold after LEDI rising	0 ns		Waveform 23	3
t45	LD set-up to STROBE rising	5 ns			3
t46	LD hold after STROBE rising	5 ns			3
t74	LDS rising to LD valid		24 ns	Waveform 23	3
t75	LDS falling to LD valid		24 ns	Waveform 23	3
t138	LADI falling to LA[31:8] valid		18 ns	Waveform 18	3
s1	CLK period	12.5ns	33.3ns	Waveform 4	
s2	CLK pulse width	5 ns		Waveform 4	
s3	AS* falling edge to CS	p1 + 1T	p1 + 2T	Waveform 1	
s4	DSA* falling edge to CS	1T	2T	Waveform 4	
s5	AS* falling edge to REGION valid		p1 – 1T	Waveform 18	
s6	AS* falling edge to RAS* falling	2T	3T	Waveform 6	
s7	ROW set-up to RAS* falling	1T		Waveform 3	
s8	PREN* falling to PCLK rising	80T		Waveform 26	
s9	COL set-up to CAS*, DBE falling	1T		Waveform 3	
s10	ROW deassertion to CAS* falling, DBE	1T		Waveform 6	
s11	LACK* rising set-up to prevent CAS* falling		p2 – 1T	Waveform 25	4
s12	LACK* falling to CAS* falling,DBE	2T	3T	Waveform 25	
s13	DSA* falling to CAS* falling,DBE	3T	4T	Waveform 4	
s14	SWDEN* set-up to CAS* falling,DBE	1T		Waveform 12	

Table 3-16. AC Parameters (continued)

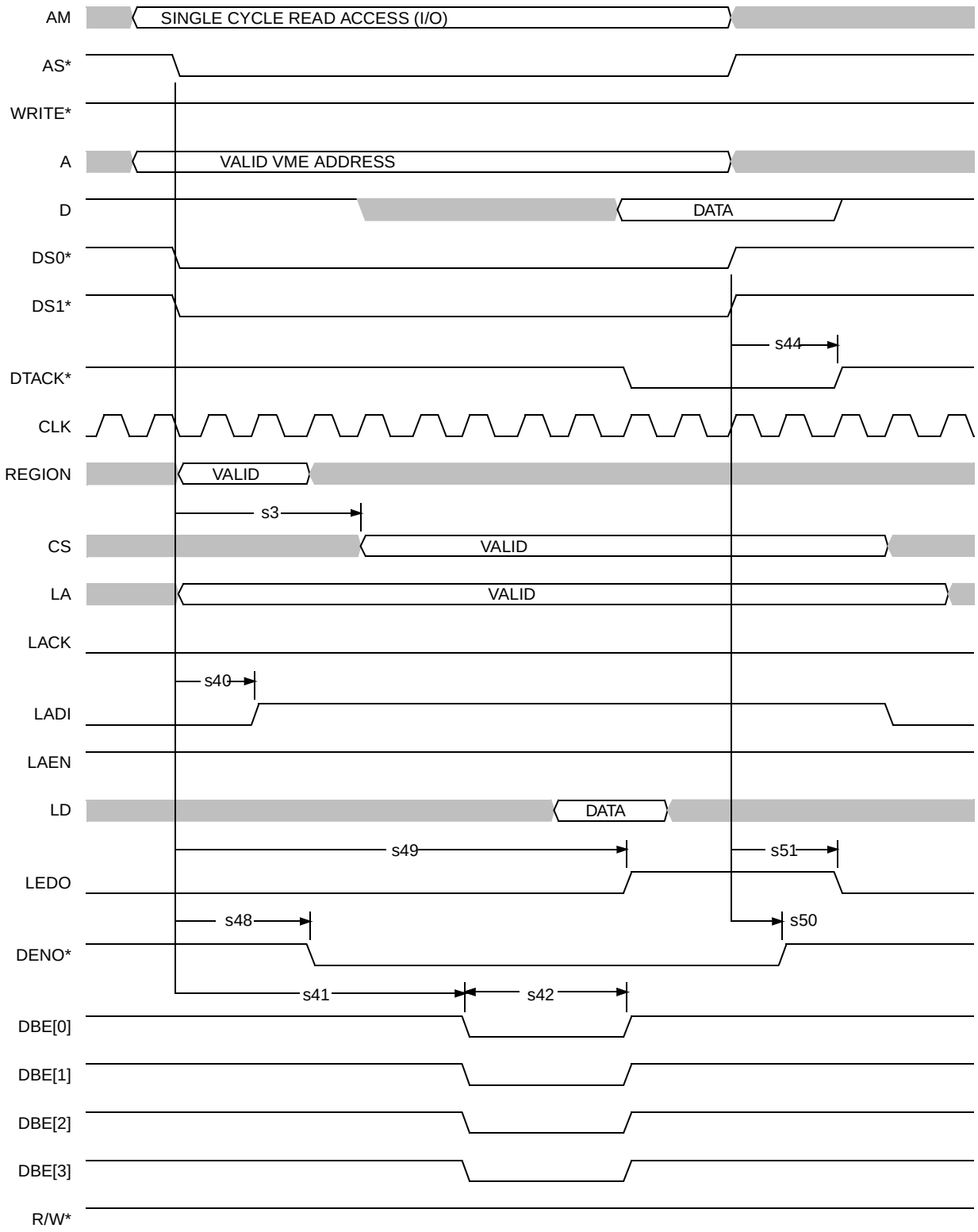
Parameter	Description	Min	Max	Reference	Notes
s15	SWDEN* hold after CAS* rising,DBE	0		Waveform 12	
s16	LA set-up to CAS* falling, DBE	1T		Waveform 12	
s17	LA hold after CAS* rising, DBE	0		Waveform 12	
s18	RAS*,COL,LAEN,R/W* hold after CAS* rising, DBE	1T		Waveform 12	
s19	DSA* falling to DTACK* falling	2T	3T	Waveform 20	
s20	RAS*,COL,LADI hold after AS* rising	2T	3T	Waveform 3	
s21	CAS* falling, DBE to DTACK* falling (read)	p3		Waveform 3	7
s22	CAS* falling, DBE to DTACK* falling (write)	p3 + 1T		Waveform 4	7
s23	DSA* rising to CAS* falling, DBE	2T	3T	Waveform 14	
s24	DSA* falling to CAS* falling,DBE	3T	4T	Waveform 16	
s25	WRITE* falling to R/W* falling		1T	Waveform 4	
s26	LACK* falling to DBE deassert	3T	4T	Waveform 24	
s27	LACK* set-up to DBE assert	0		Waveform 24	
s28	LD set-up to DBE deassert (read)	5 ns		Waveform 6	
s29	LD hold after DBE deassert (read)	5 ns		Waveform 6	
s30	LD set-up to DBE assert (write)	1T		Waveform 11	
s31	LD hold after DBE deassert (write)	0 ns		Waveform 11	6
s32	AM valid to DENIN*,DENIN1* falling	1T	2T	Waveform 9	
s33	DSA* rising to DENIN*,DENIN1* rising	1T	2T	Waveform 9	
s34	PREN* falling to PDATA valid		39T	Waveform 26	5
s35	PCLK period	1000T		Waveform 26	5
s36	PDATA set-up to PCLK falling	1T		Waveform 26	5
s37	PDATA hold after PCLK falling	3T		Waveform 26	5
s38	IACKIN* falling to LDEN* falling	2T	3T	Waveform 7	
s39	LDEN* falling to DTACK* falling	6T		Waveform 7	
s40	AS* falling to LADI rising	1T	2T	Waveform 1	
s41	AS* falling to DBE assertion	5T + p1	6T + p1	Waveform 1,2	
s42	DBE asserted width	p3		Waveform 1,2	7
s43	WRITE High to R/W* High		1T	Waveform 4	
s44	DSB High to DTACK* High	1T	2T	Waveform 1,2	
s45	AS* High to LADI High	1T	2T	Waveform 2	
s46	AS* High to CS deasserted	1T	2T	Waveform 2	
s47	AS* falling to DENIN* asserted	T + p1	2T + p1	Waveform 2	
s48	DSA* falling to DENO* asserted	2T + p1	3T + p1	Waveform 1	
s49	DSA* falling to LEDO rising	3T + p1 + p3	4T + p1 + p3	Waveform 1	7
s50	DSA* rising to DENO* rising	0	1T	Waveform 1	
s51	DSA* rising to LEDO falling	1T	2T	Waveform 1	

Table 3-16. AC Parameters (continued)

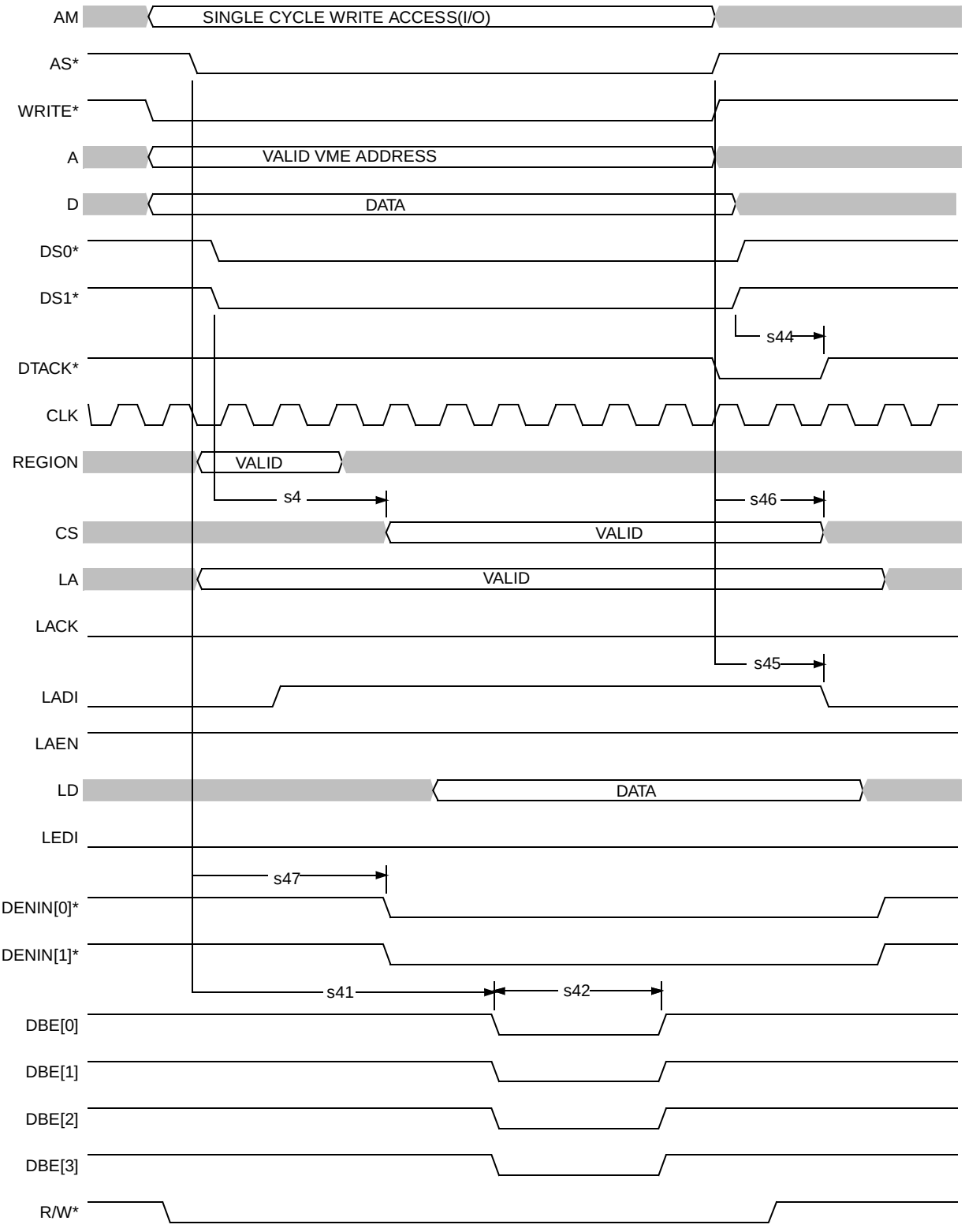
Parameter	Description	Min	Max	Reference	Notes
s52	RAS* rising to CAS* falling (Refresh)	(p5 – 3)T		Waveform 13	
s53	CAS* falling to RAS* falling (Refresh)	1T		Waveform 13	
s54	RAS* falling to CAS* rising (Refresh)	(p3 – 1)T		Waveform 13	
s55	CAS* rising to RAS* rising (Refresh)	(p5 – 3)T		Waveform 13	
s56	RAS* Precharge (Refresh)	(p5 – 2)T		Waveform 13	
s57	RAS* cycle time (Refresh)	(p3 + 2p5 – 5)T		Waveform 13	
s58	RAS* Precharge	p5		Waveform 13	
s59	AS* falling to REGION hold	3T		Waveform 18	
s60	LIRQ* to IRQ*	0	1T	Waveform 8	
s61	R/W*, LA, LD set-up to SELECTLM* falling	10 ns		Waveform 21	
s62	R/W*, LA, LD hold after SELECTLM* rising	0 ns		Waveform 21	
s63	SELECTLM* falling to LD valid	2T	3T	Waveform 21	
s64	SELECTLM* rising to LD Z	2T	3T	Waveform 21	
s65	LACK rising to LBERR* High	0	1T	Waveform 21	
s66	SELECTLM* falling to LBERR* falling	1T	2T	Waveform 21	
s67	SELECTLM* rising to LBERR* rising	1T	2T	Waveform 21	
s68	LACK falling to LBERR* Z	0	1T	Waveform 21	
s69	SELECTLM* minimum assertion	8T		Waveform 21	
s70	SYSRESET* falling to PREN falling			Waveform 19	
s71	PREN* pulse width			Waveform 19	
s72	SYSRESET* rising to IRQ* falling			Waveform 19	

Notes:

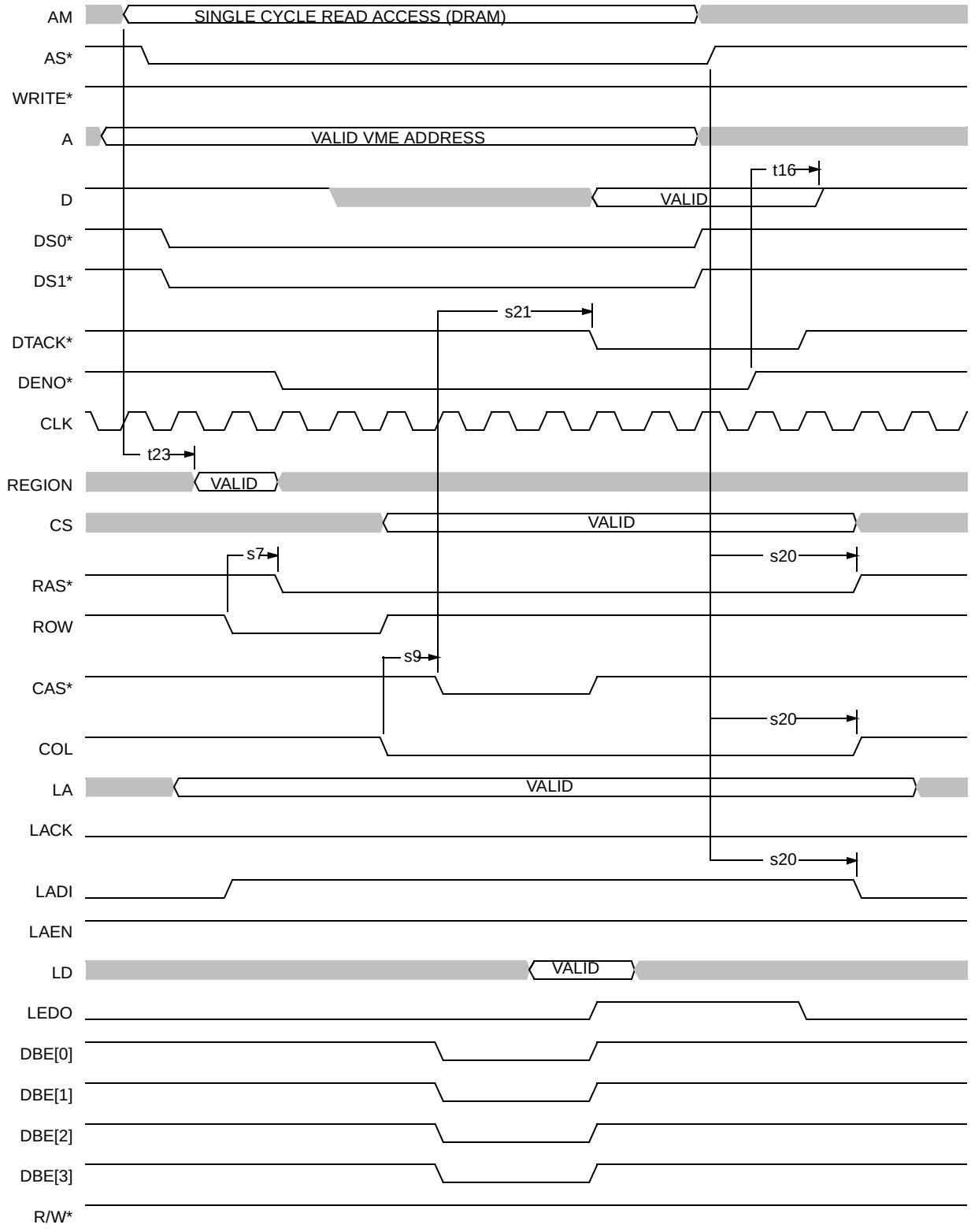
2. This parameter is a constant set when the CY7C960 is programmed at initialization.
3. This parameter is a CY7C964 performance parameter.
4. This parameter is timed from RAS falling and specifies the latest deassertion of LACK to inhibit the CAS burst.
5. This parameter applies when CY7C960 is sourcing PCLK.
6. Hold time is guaranteed by CY7C964 delay from LEDI, DENIN*, or LDS to LD. The CY7C960/CY7C964 design favors falling edge capture of data. LA, LD, and DBE differential delay must be carefully managed for rising edge capture.
7. p3 represents two independent programmable fields. CAS Assert width applies for DRAM access. DBE width is a program set. A unique value is associated with each “region” I/O access.



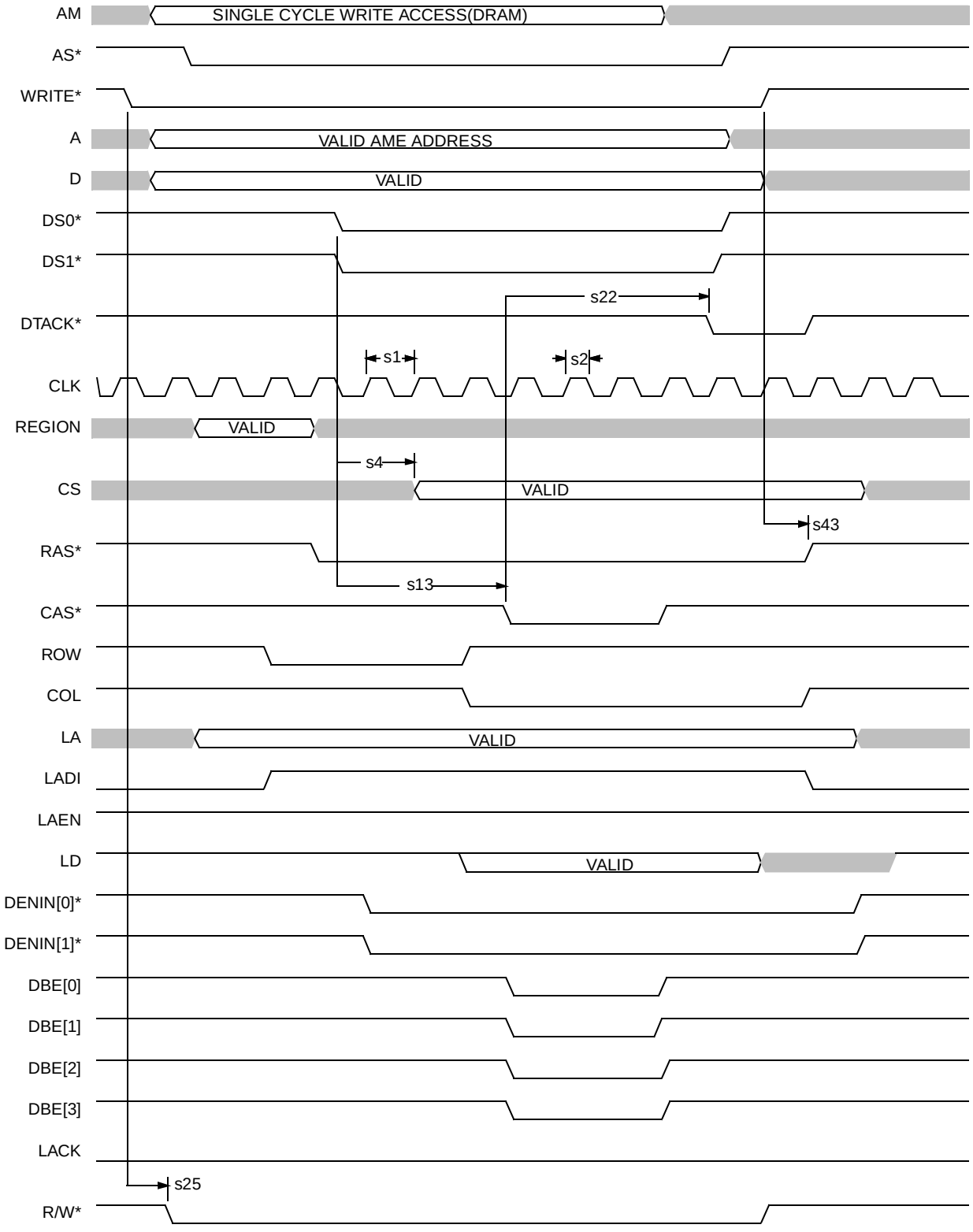
Waveform 1. Single Cycle Read Access (I/O Mode)



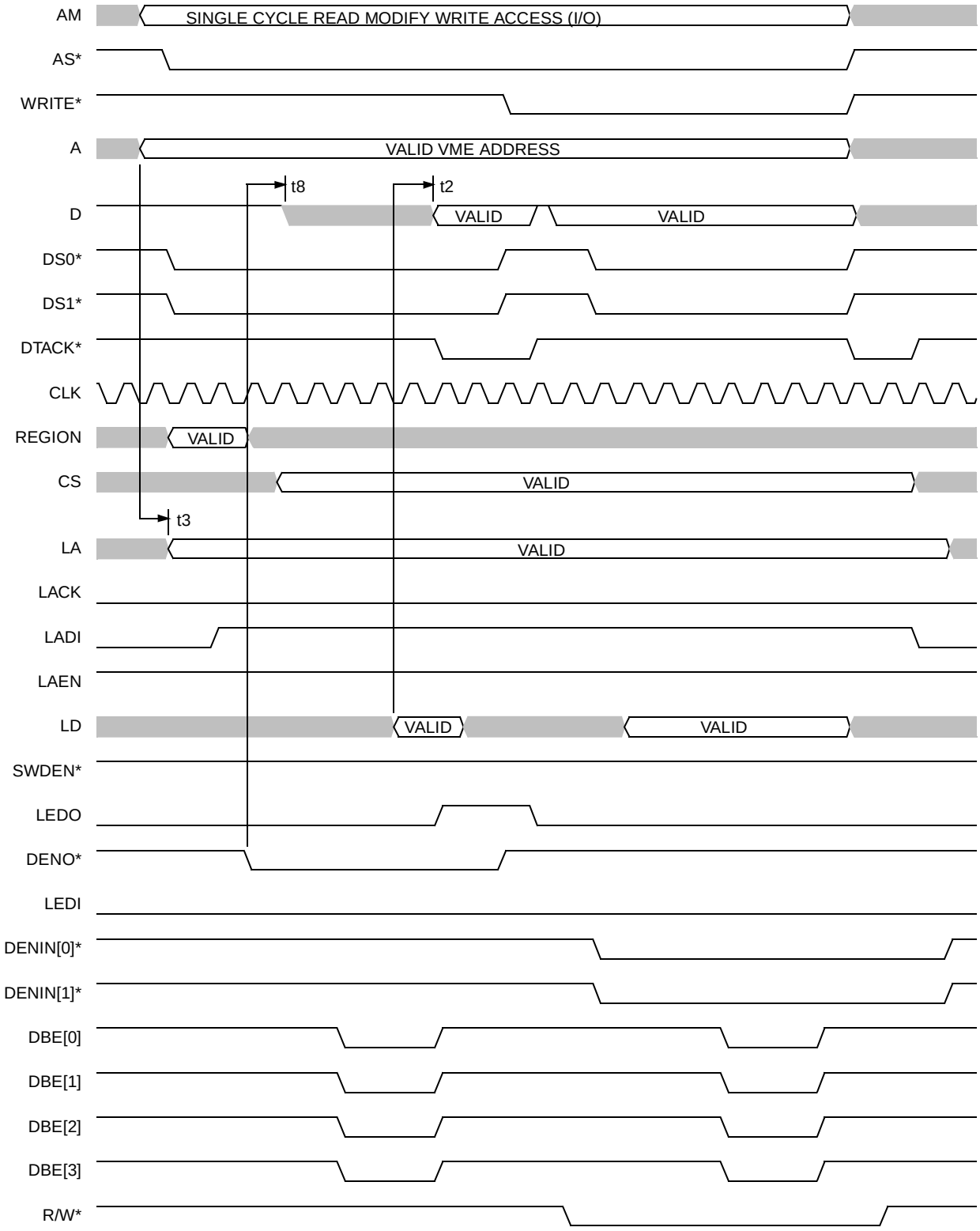
Waveform 2. Single Cycle Write Access (I/O Mode)



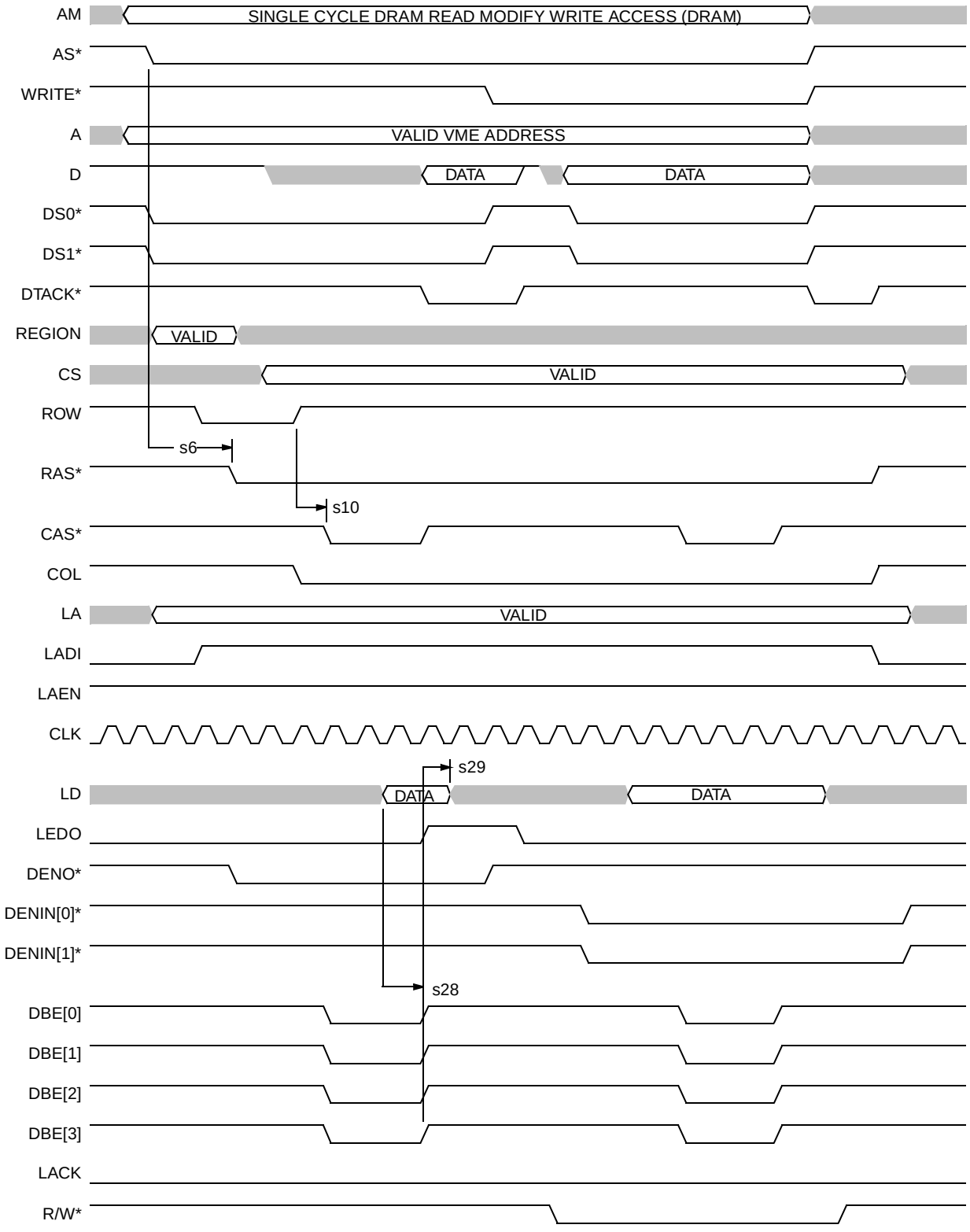
Waveform 3. Single-Cycle Read Access (DRAM Mode)



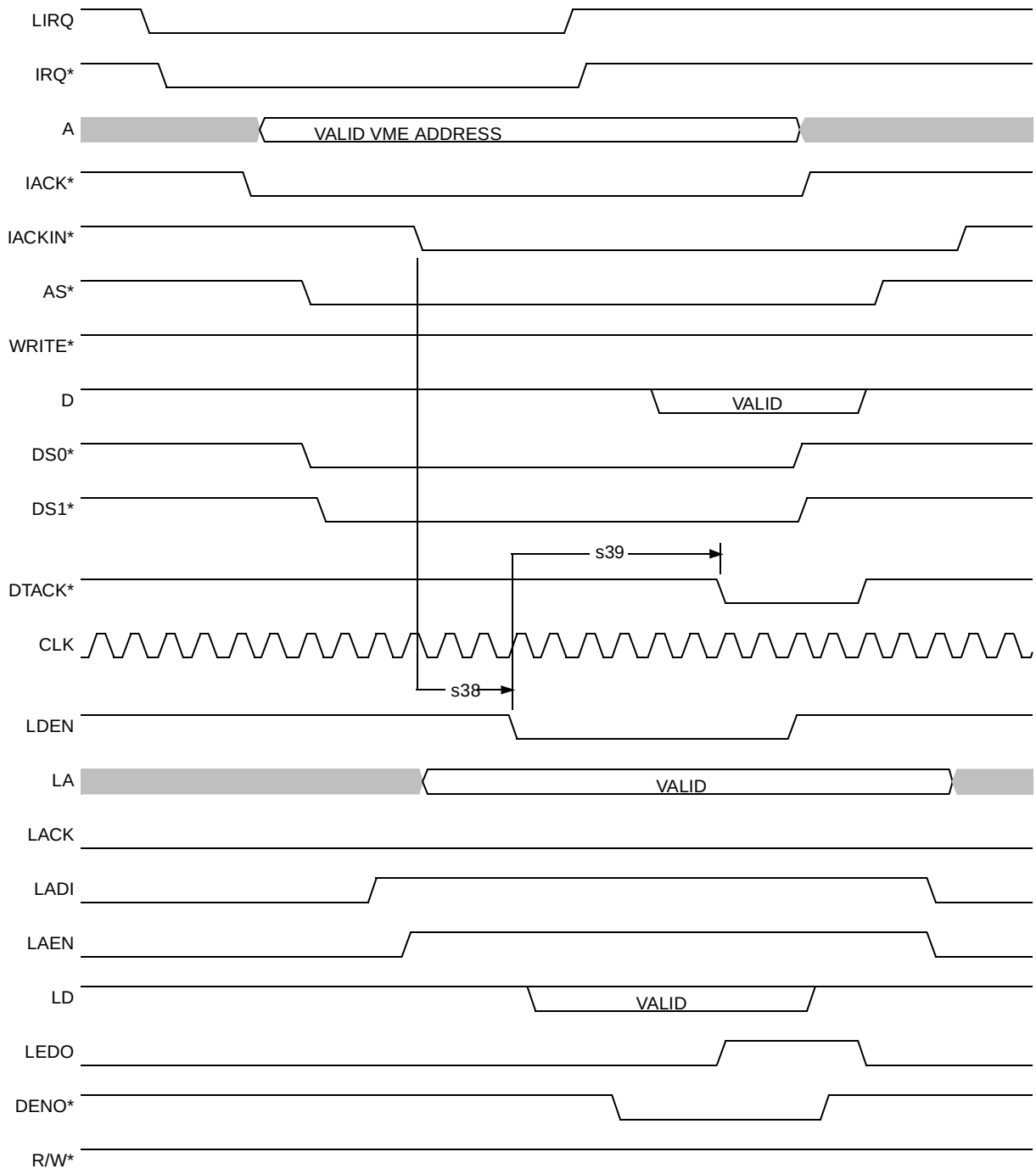
Waveform 4. Single Cycle Write Access (DRAM Mode)



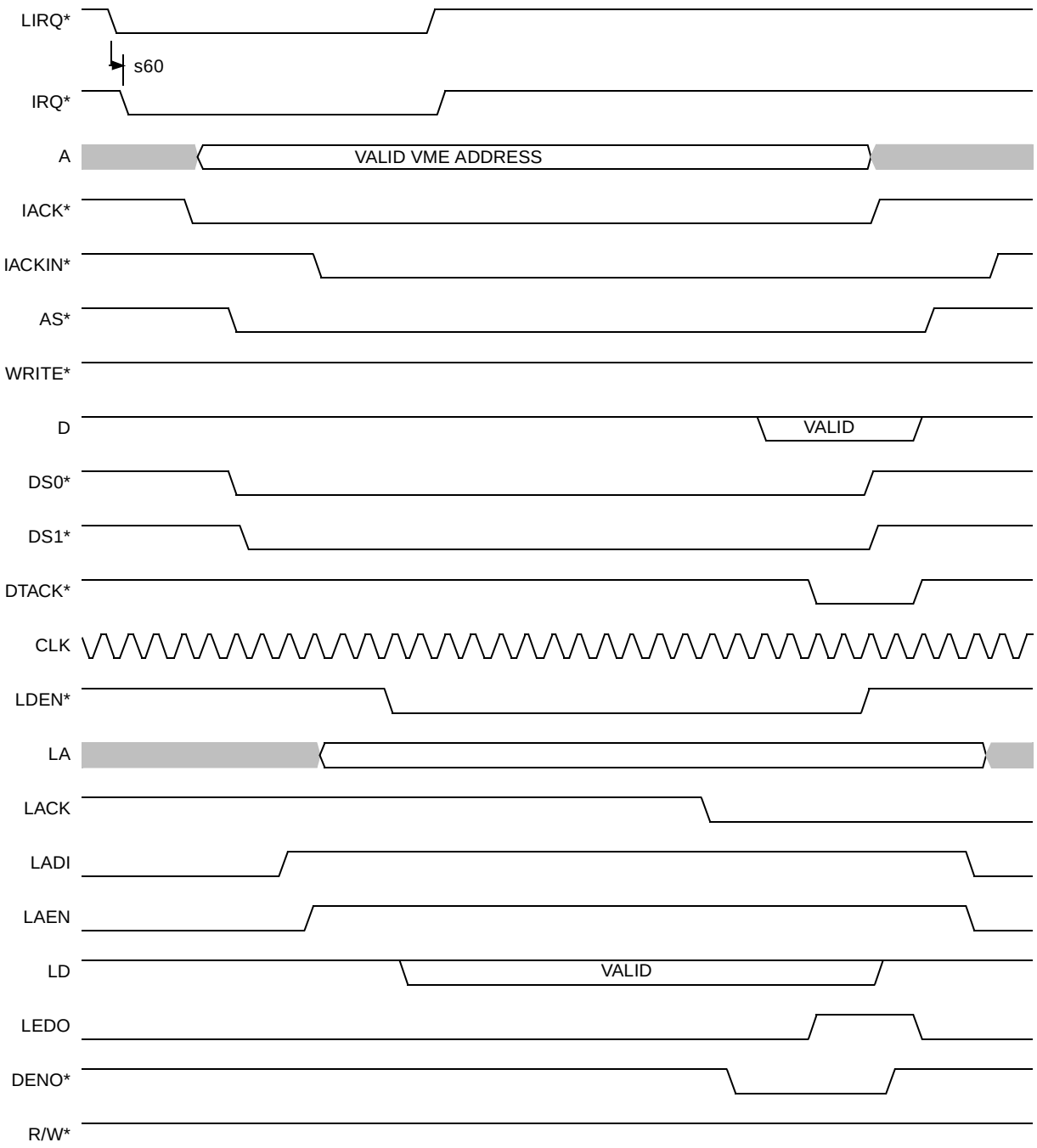
Waveform 5. Single-Cycle Read-Modify-Write Access (I/O Mode)



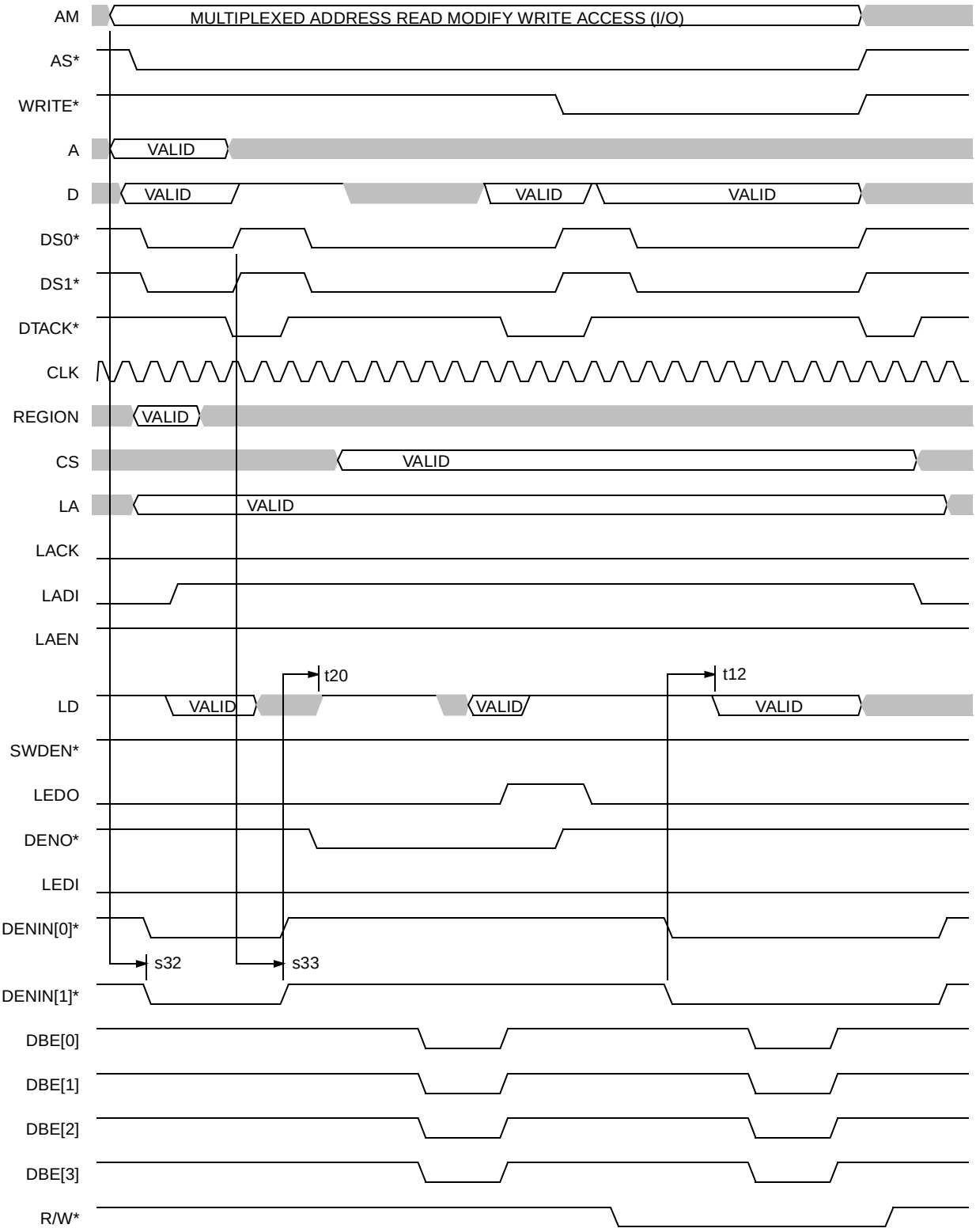
Waveform 6. Single-Cycle Read-Modify-Write Access (DRAM Mode)



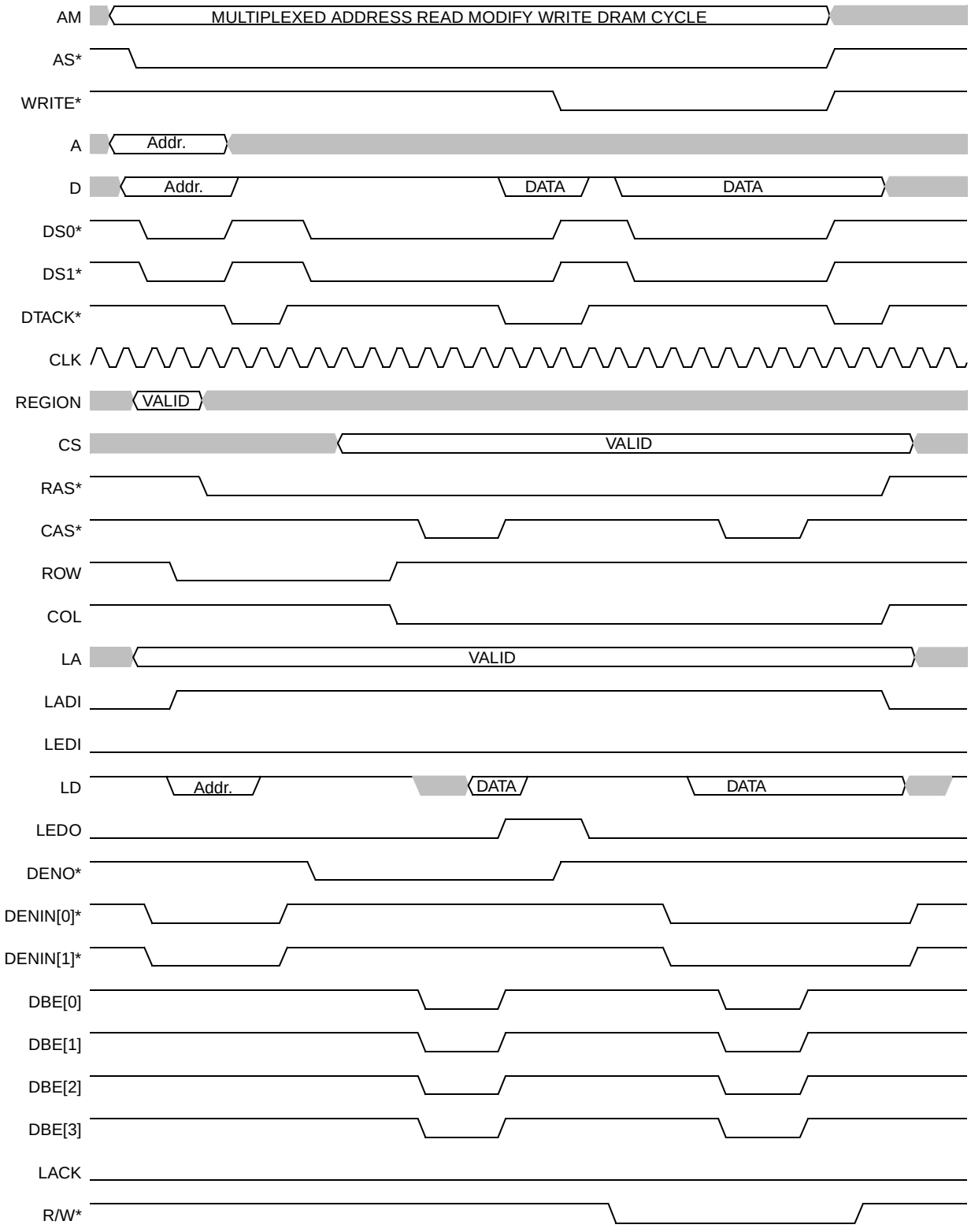
Waveform 7. IACK CYCLE Self-timed



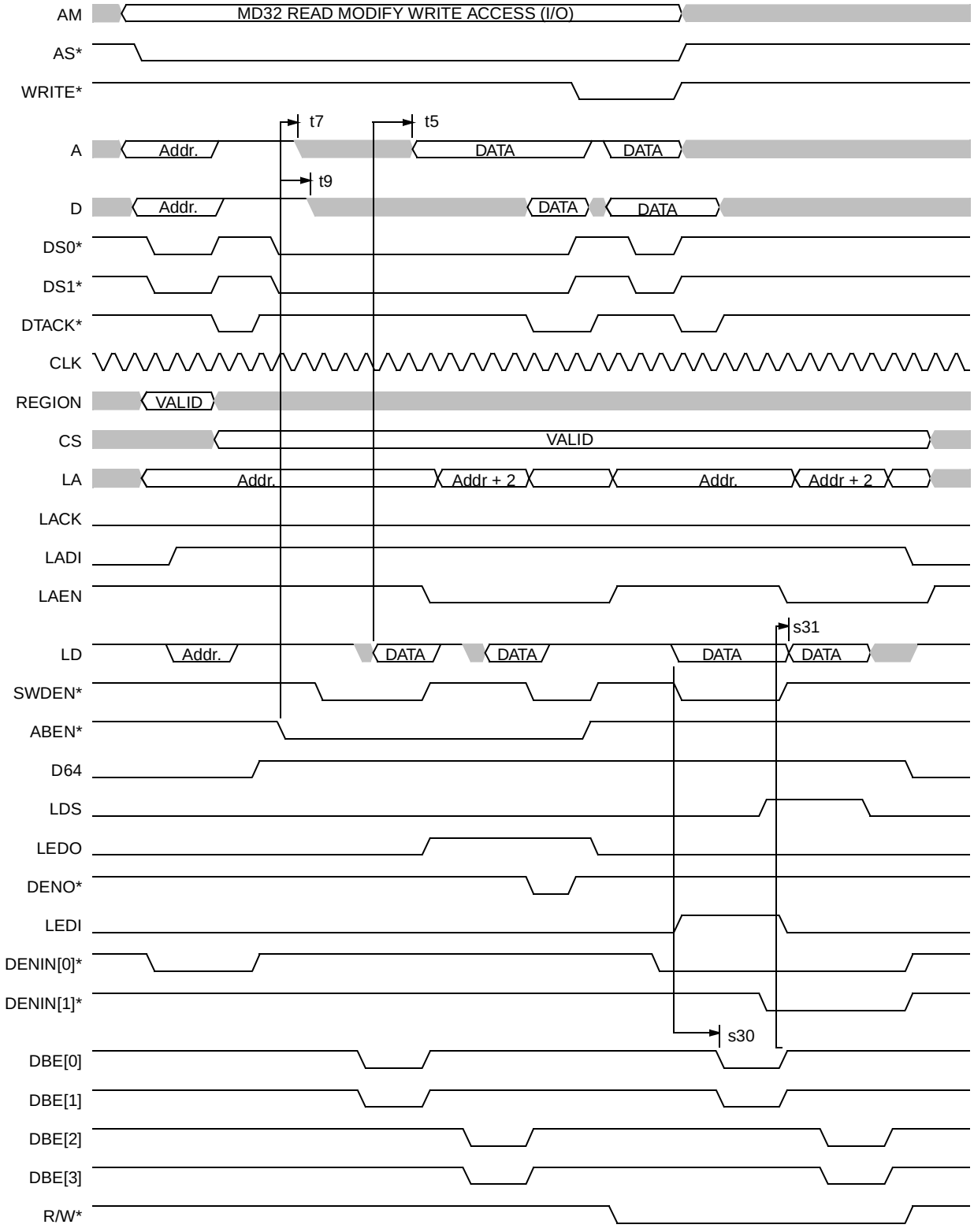
Waveform 8. IACK Cycle Handshake



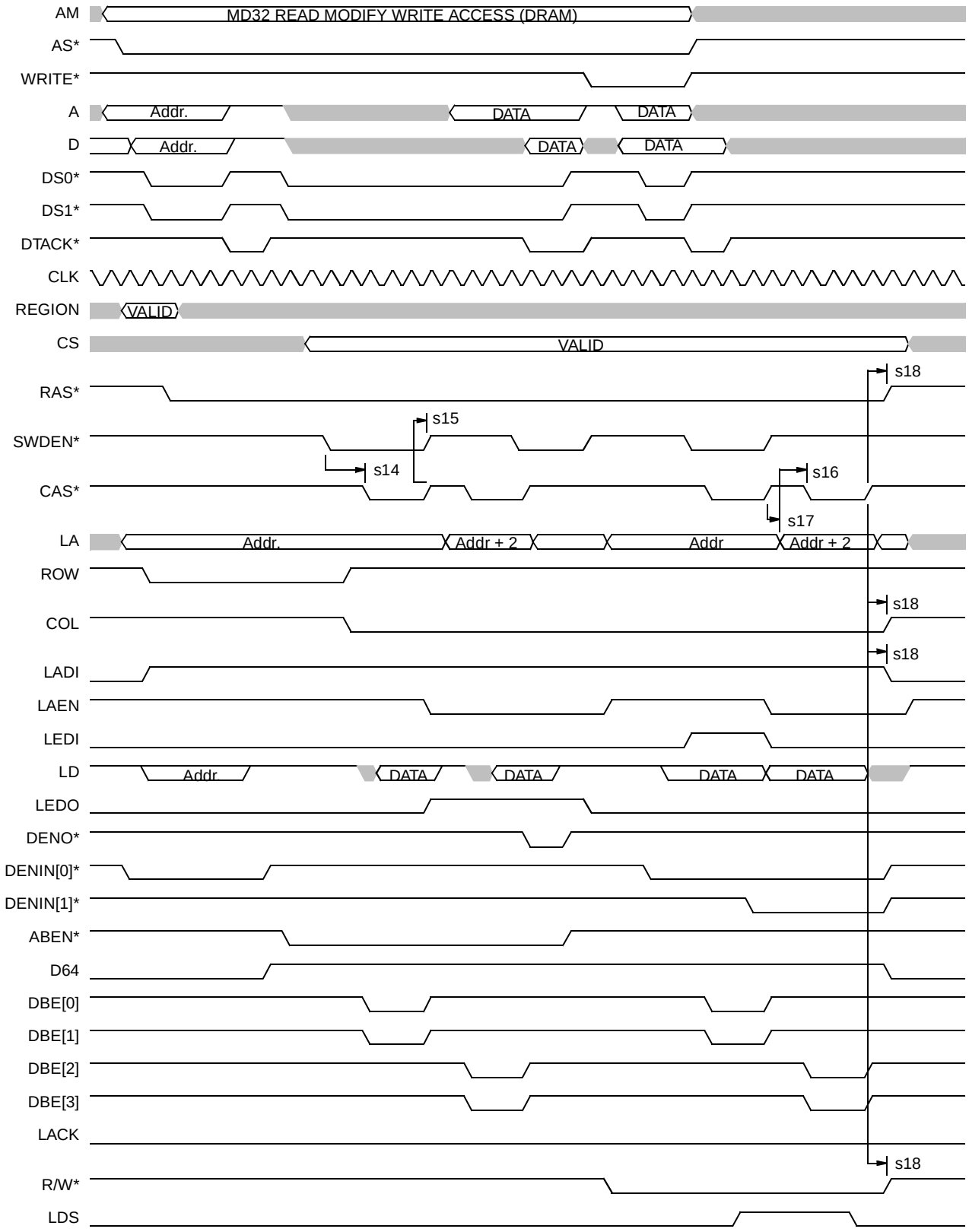
Waveform 9. Multiplexed Address Read-Modify-Write Access (I/O Mode)



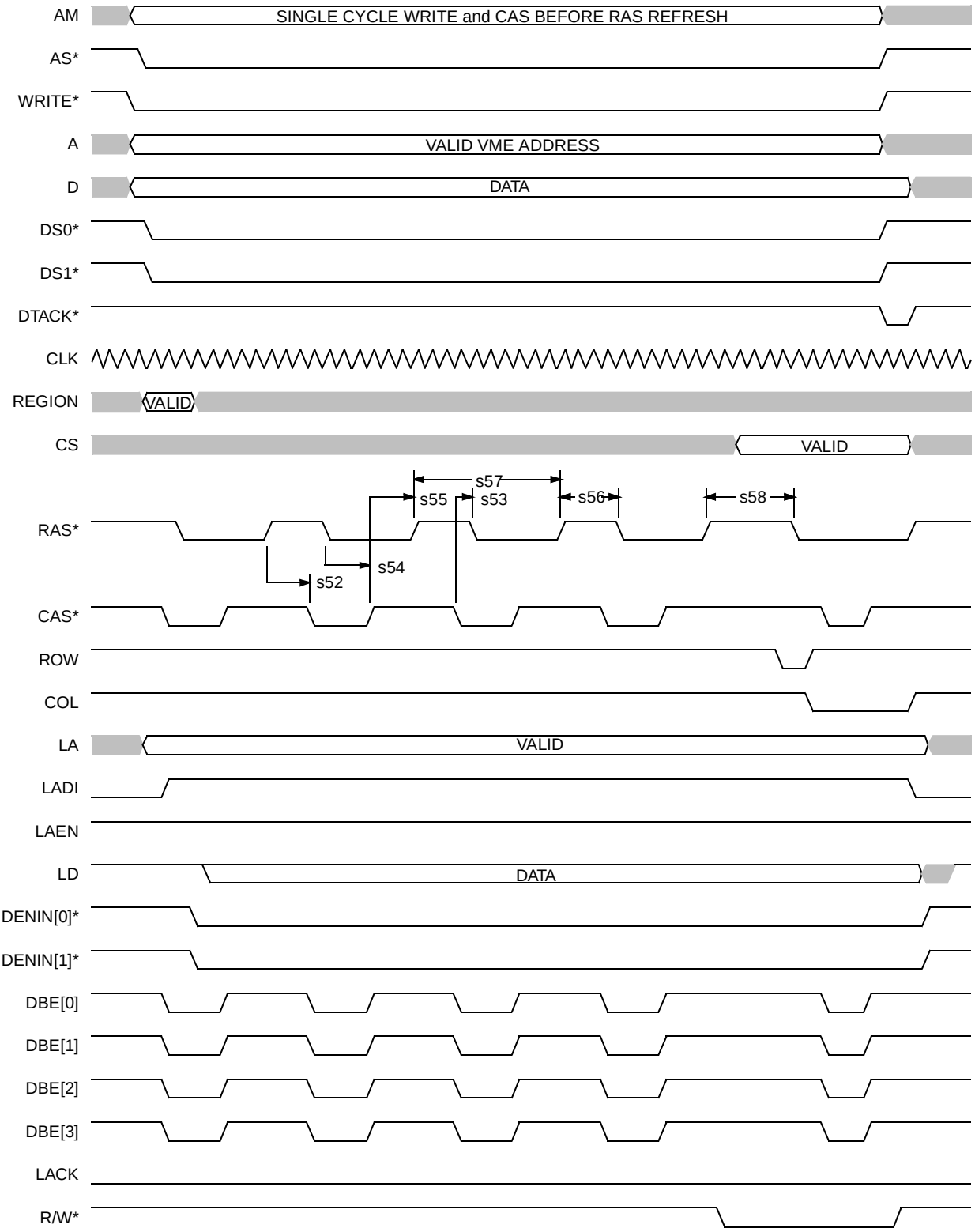
Waveform 10. Multiplexed Address Read-Modify-Write Access (DRAM Mode)



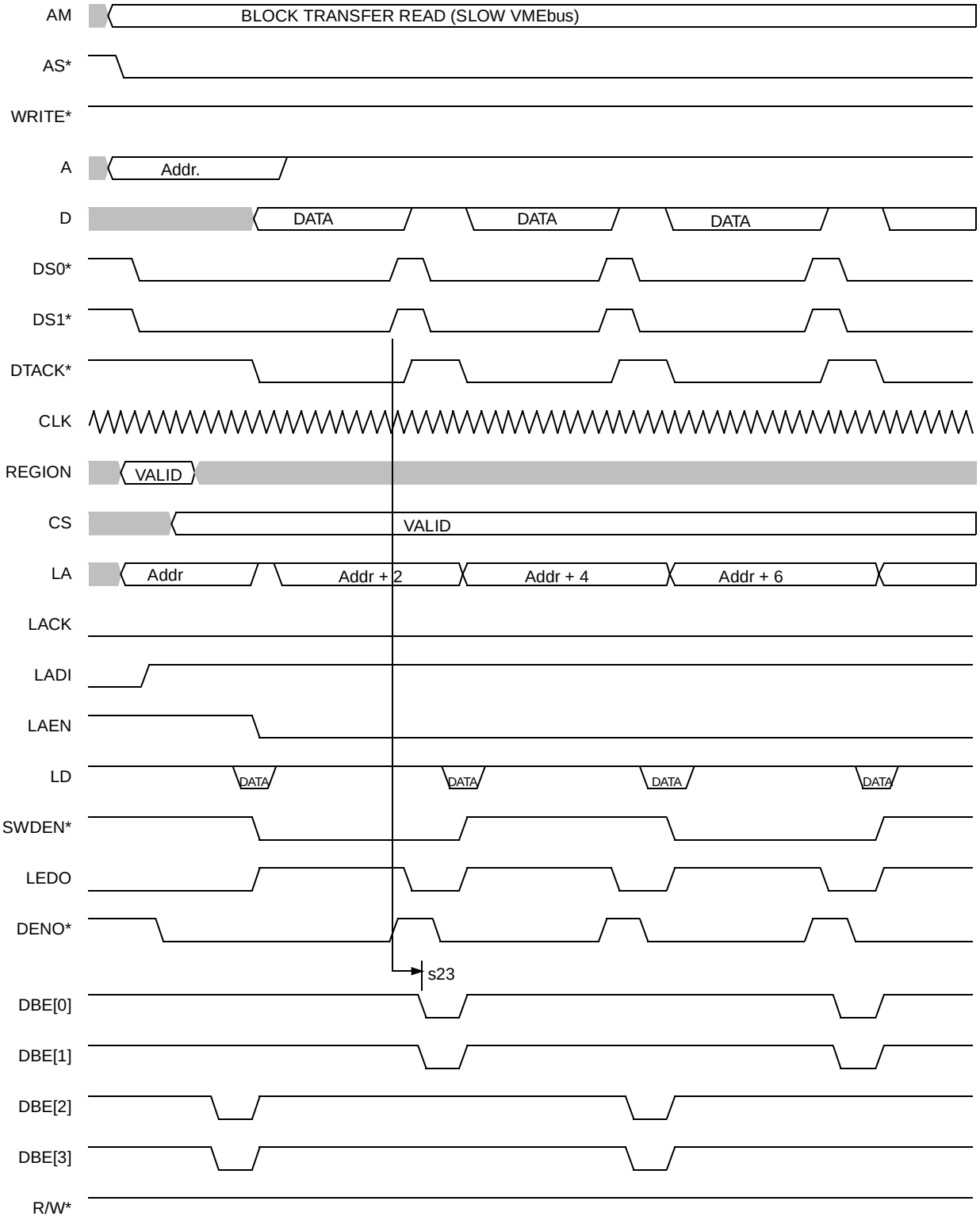
Waveform 11. MD32 Read-Modify-Write Access (I/O Mode)



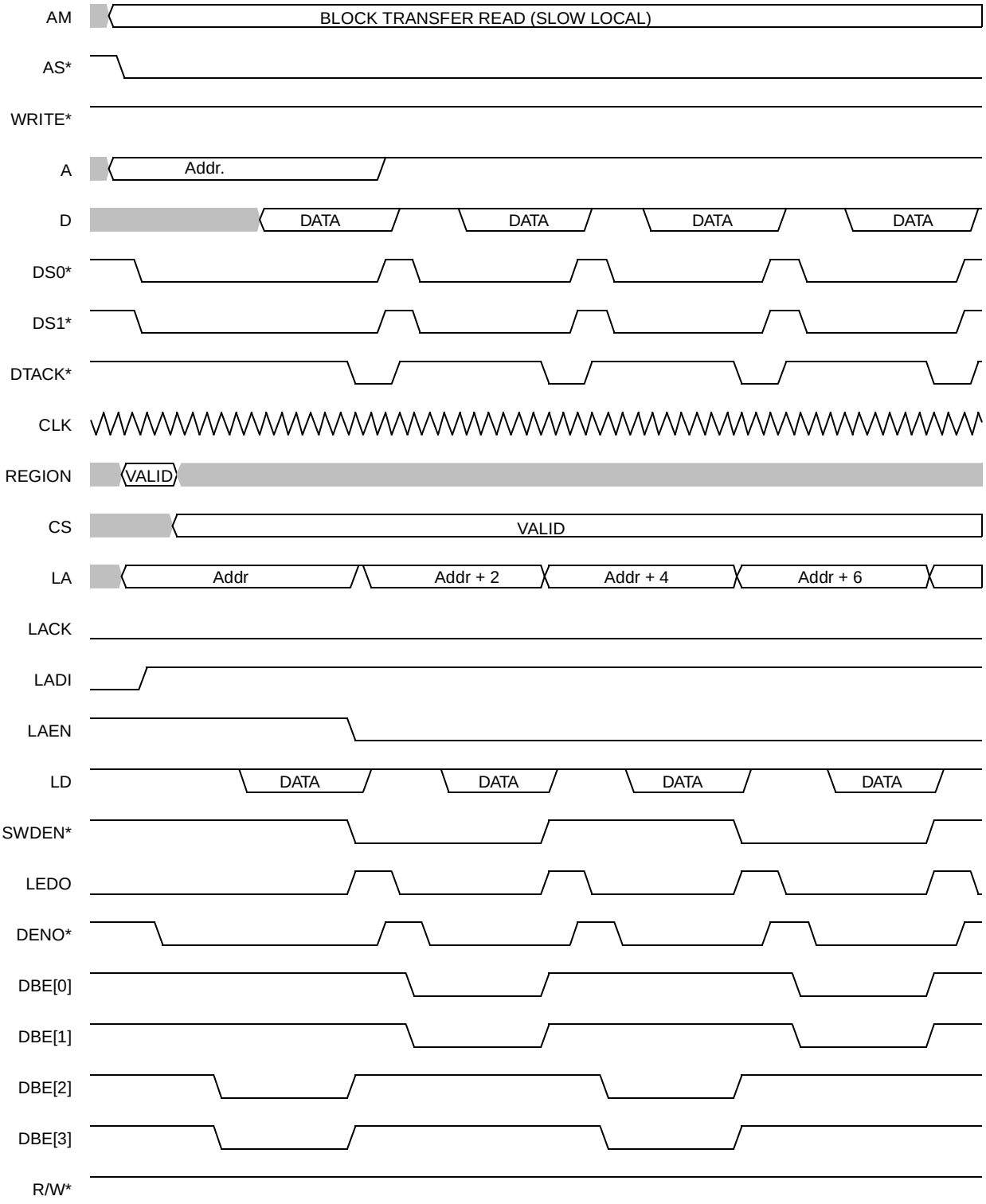
Waveform 12. MD32 Read-Modify-Write Access (DRAM Mode)



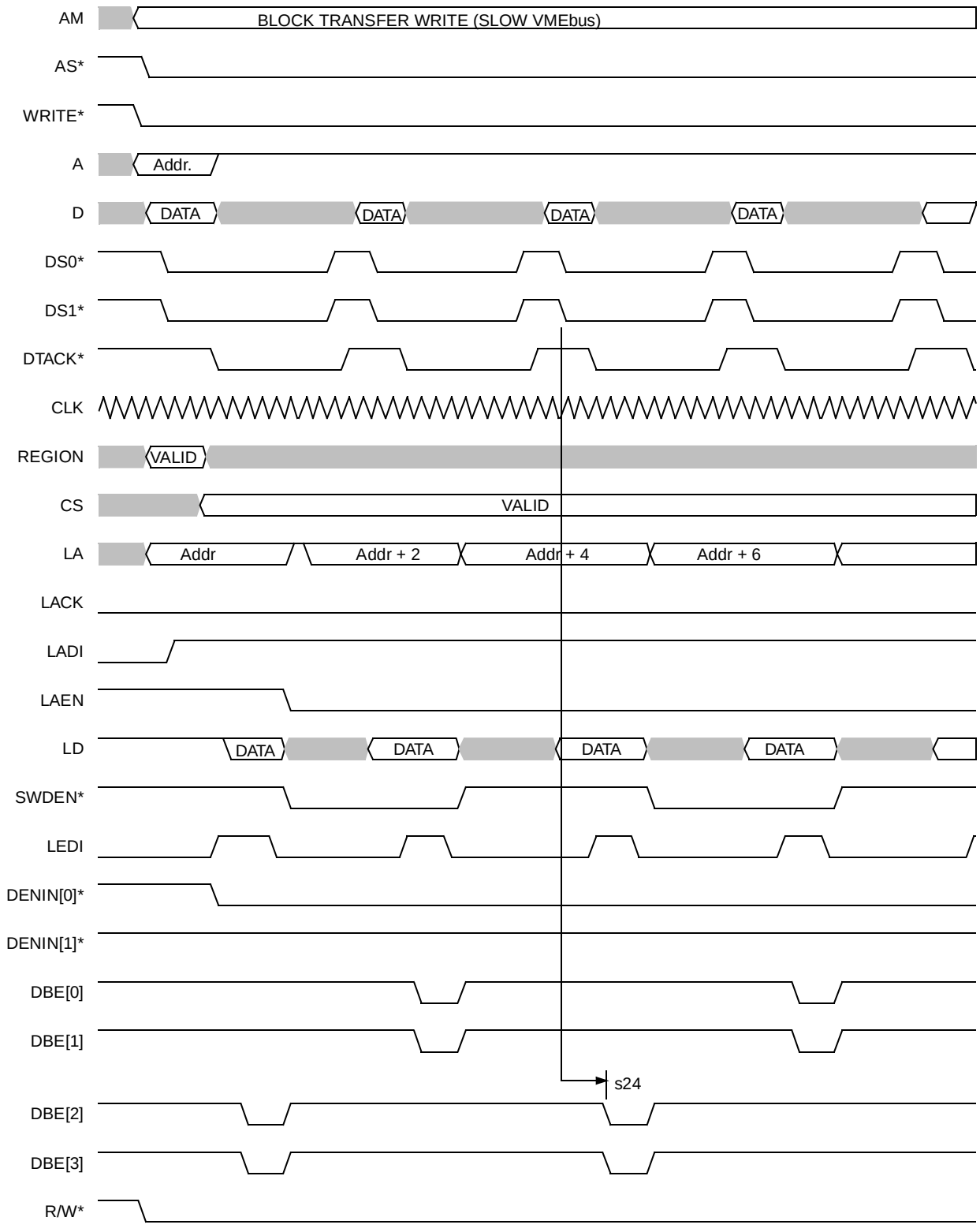
Waveform 13. Single-Cycle Write and CAS before RAS Refresh



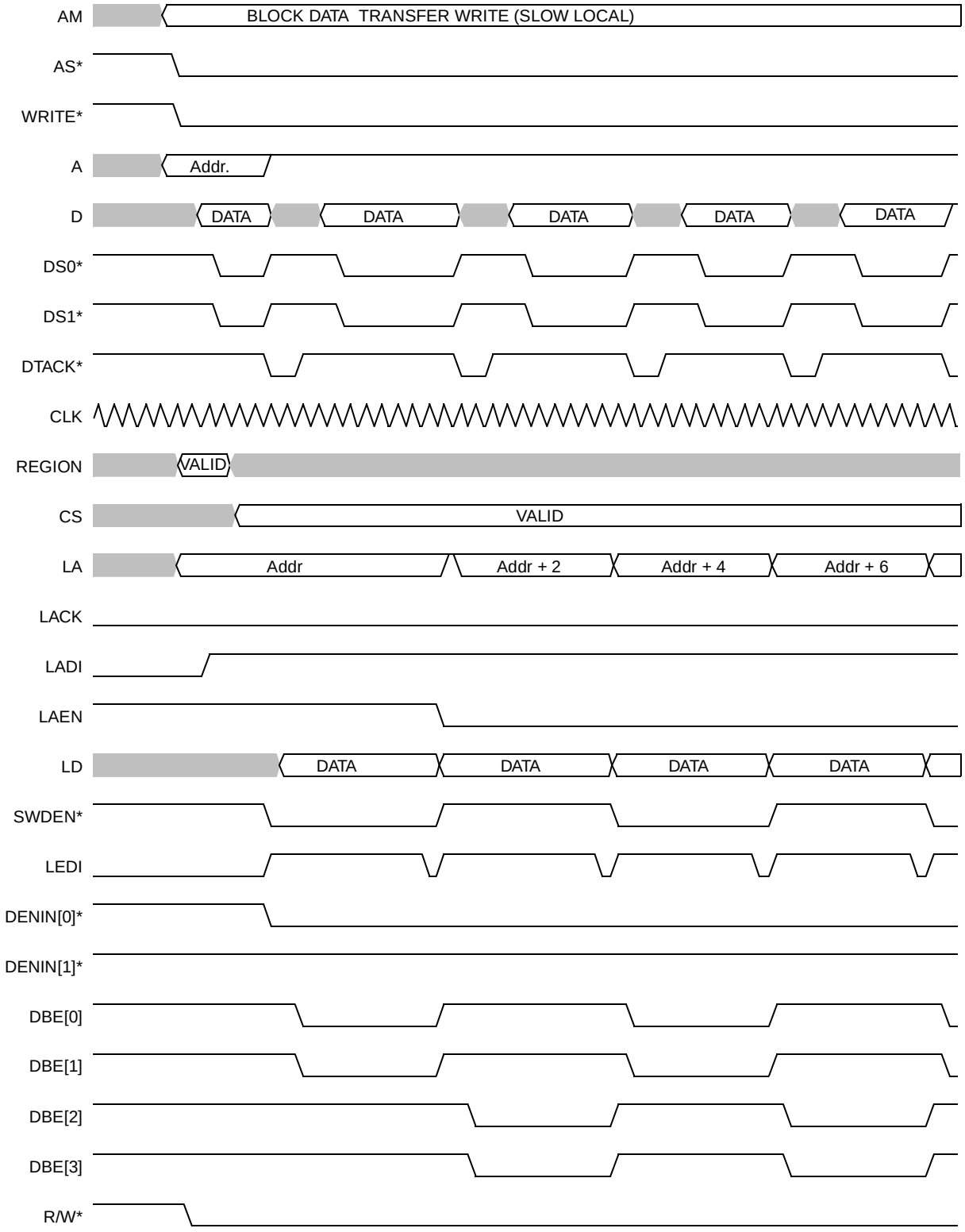
Waveform 14. Block Transfer Read (Slow VME)



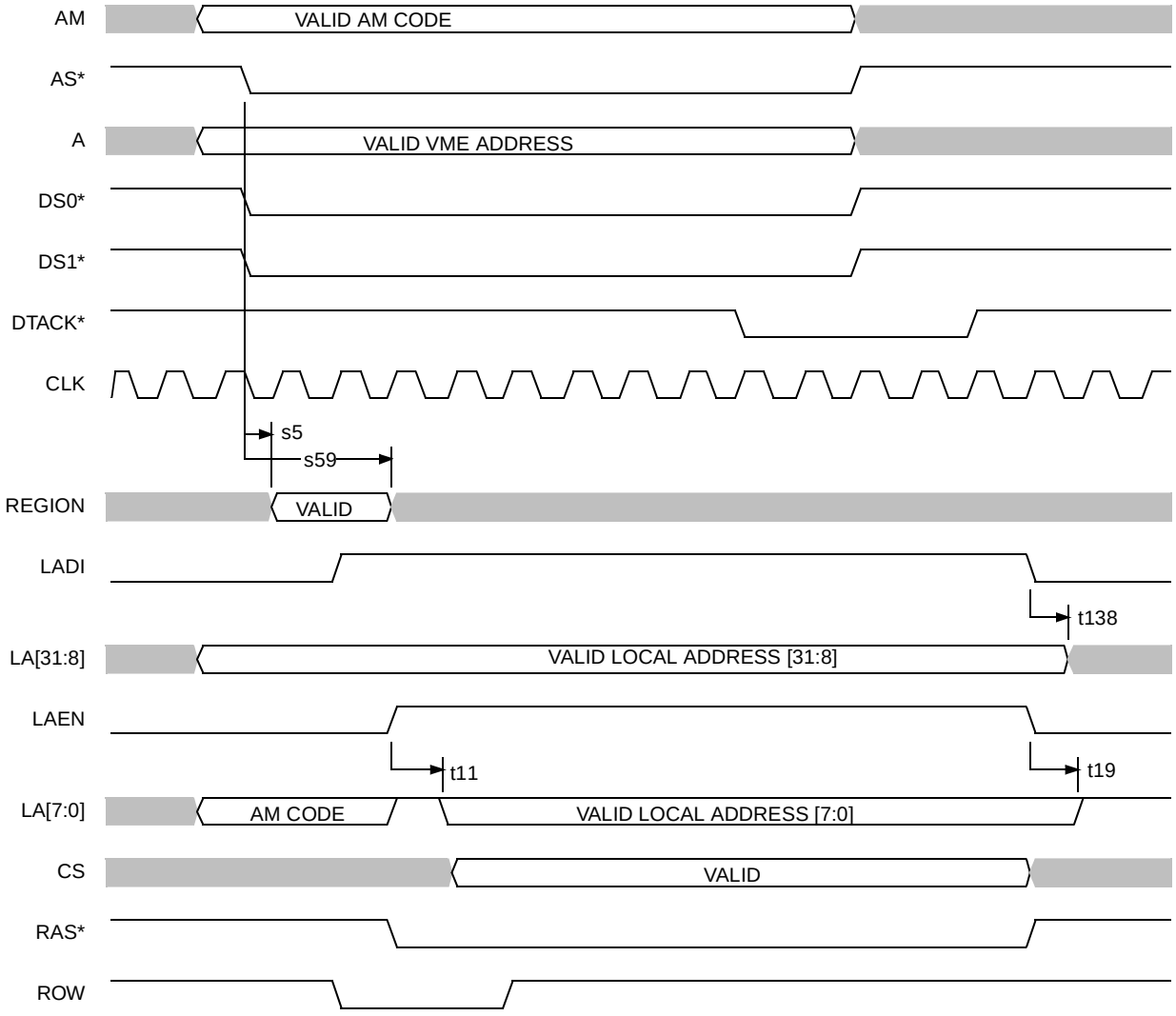
Waveform 15. Block Transfer Read (Slow Local)



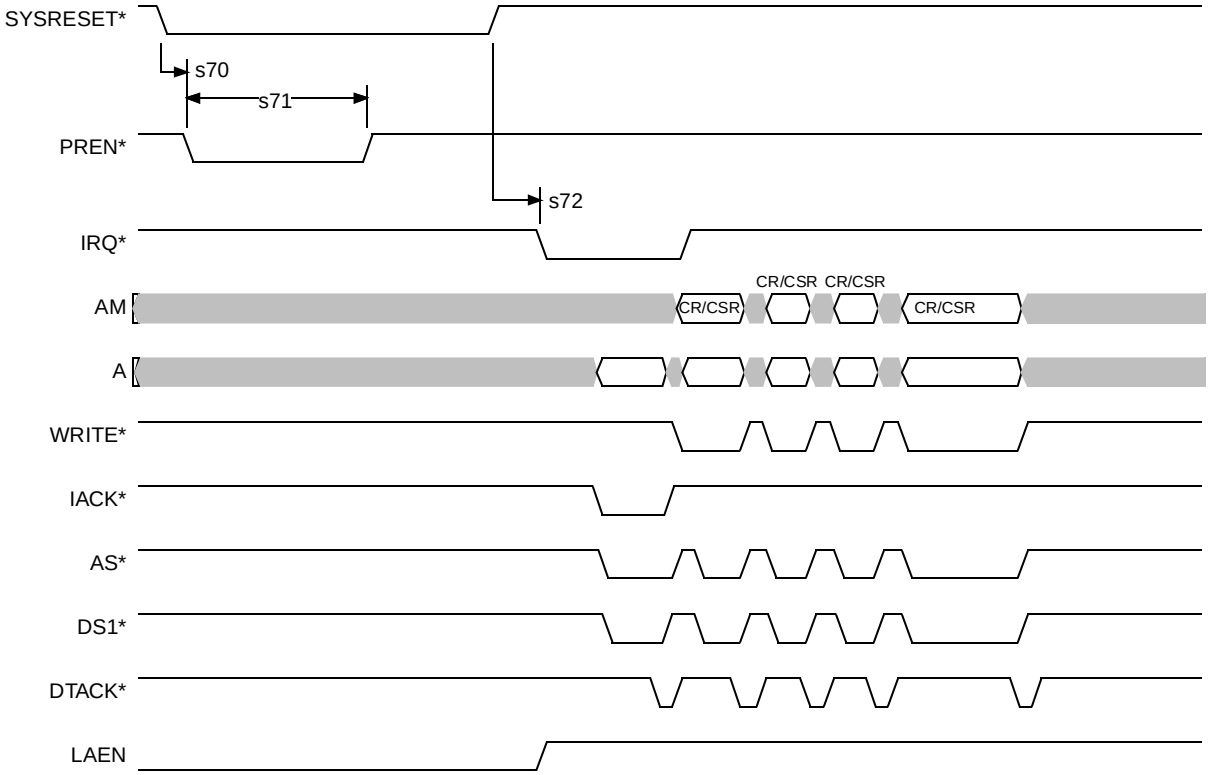
Waveform 16. Block Transfer Write (Slow VMEbus)



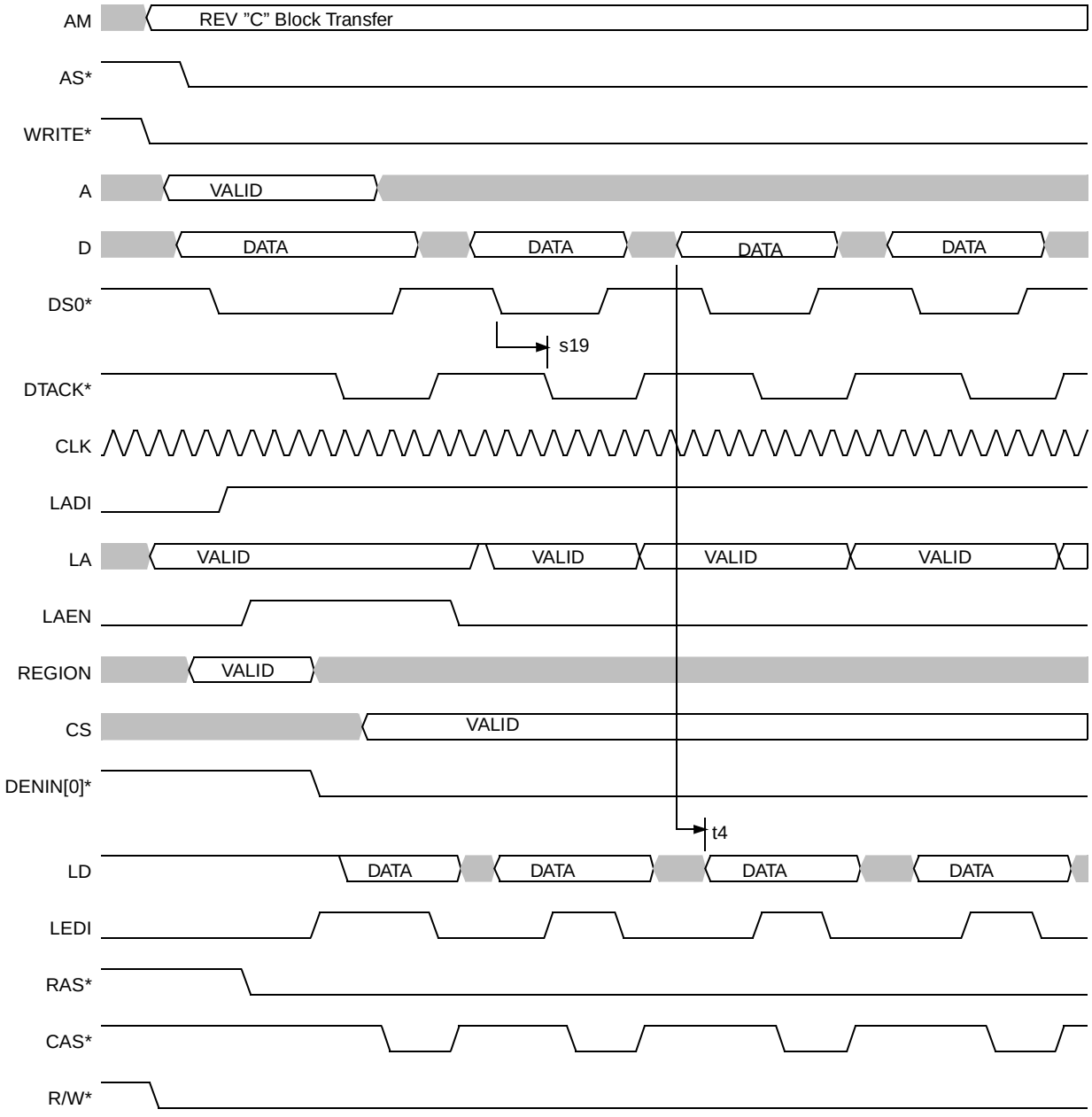
Waveform 17. Block Transfer Write (Slow Local)



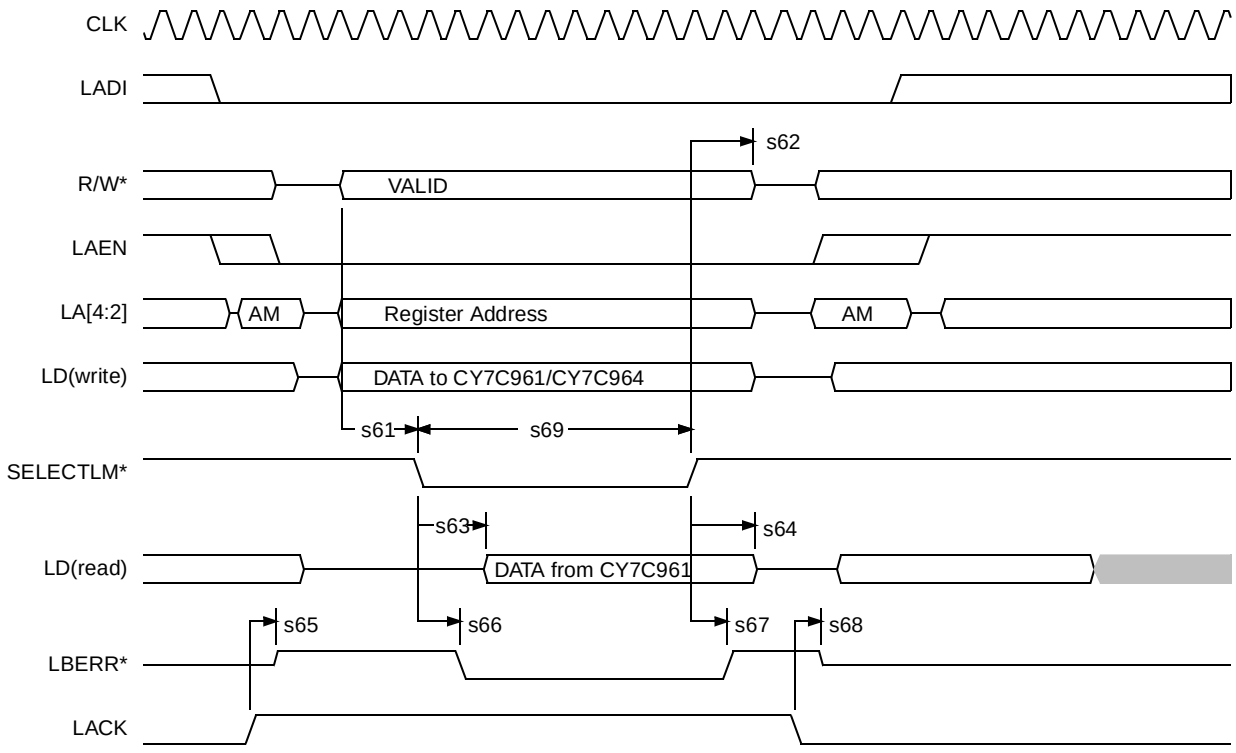
Waveform 18. AMCODE or LA and Decode Timing



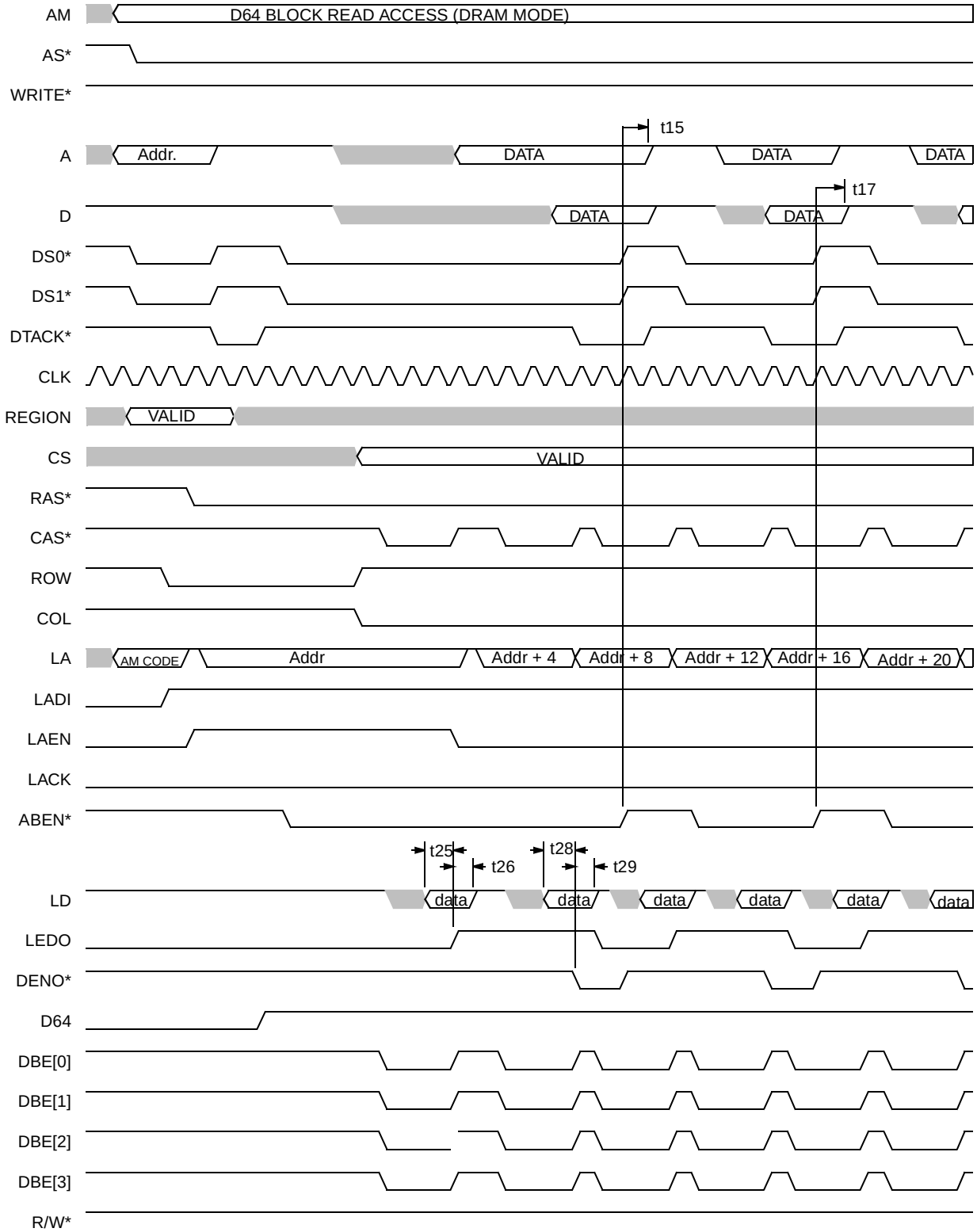
Waveform 19. VMEbus Initialization Method



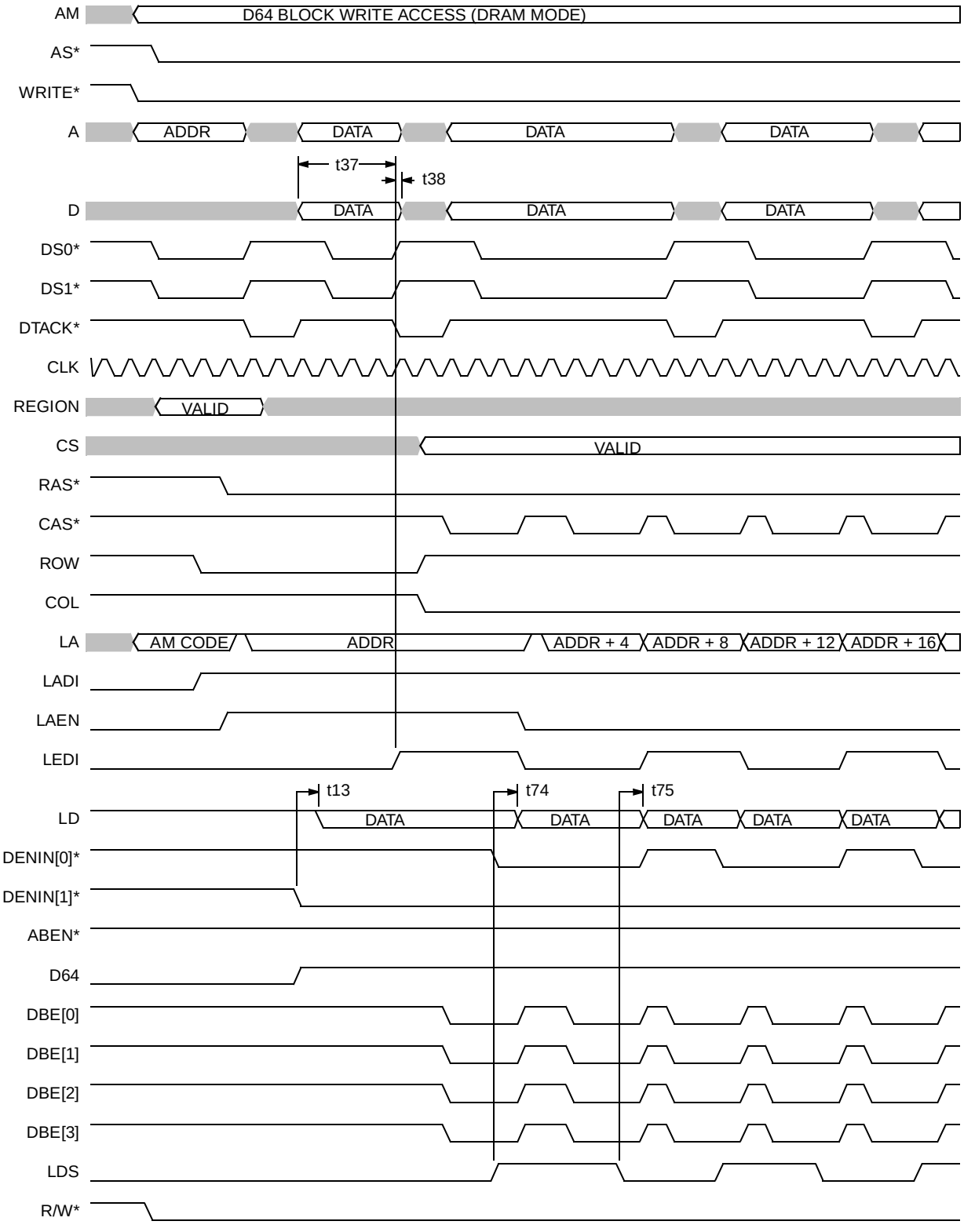
Waveform 20. Block Transfer Write (DRAM Mode)



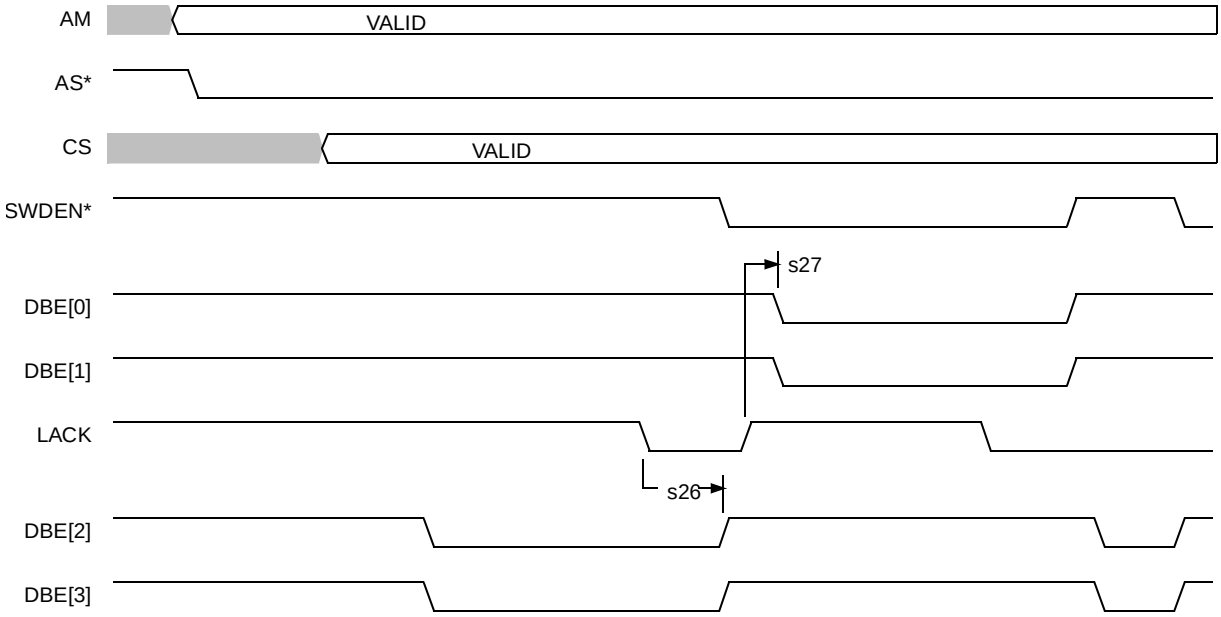
Waveform 21. Local Bus Holdoff (and CY7C961 Register Access)



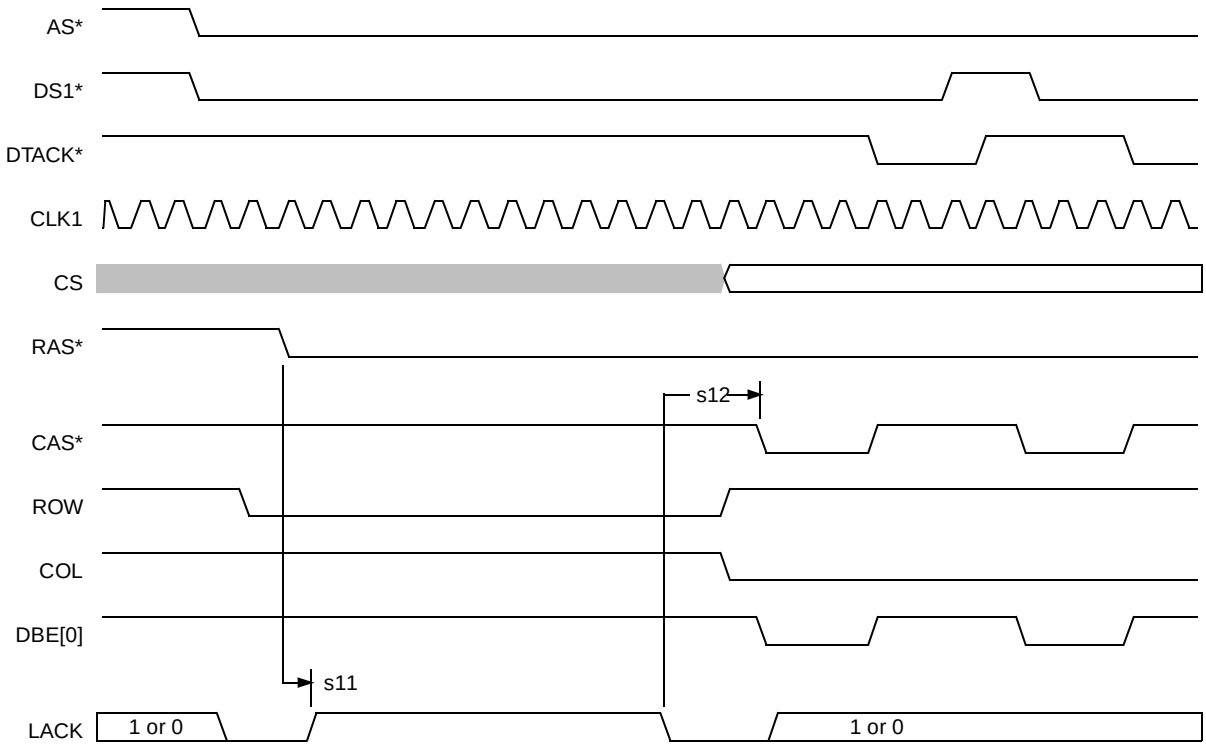
Waveform 22. D64 Block Read Access (DRAM Mode)



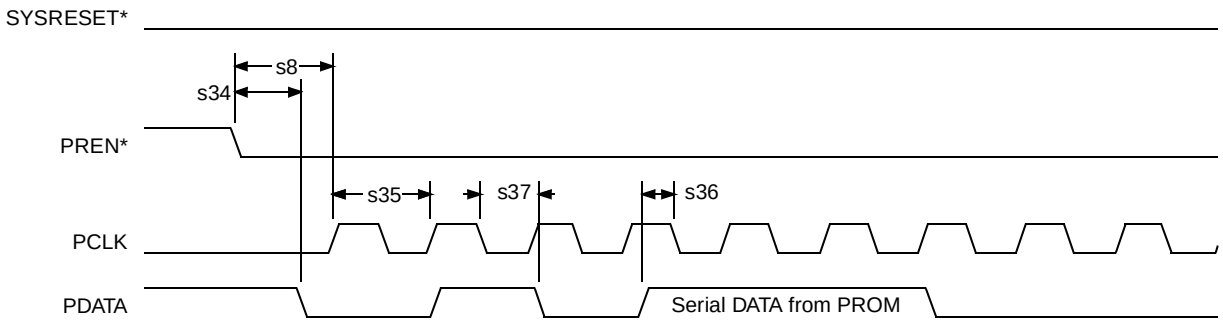
Waveform 23. D64 Block Write Access (DRAM Mode)



Waveform 24. LACK Handshake (I/O Mode)



Waveform 25. LACK Handshake (DRAM Mode)



Waveform 26. Local Initialization Method

3.13
DC Performance Specifications

Table 3-17. VMEbus Signals AS*, DS1*, DS0*, DTACK*

Parameter	Description	Test Conditions	Comm.	Industrial	Military	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} =	2.4 –16 mA	2.4 –10 mA	2.4 –9 mA	V
V_{OL}	Maximum Low-Level Output Voltage	V_{CC} = Min., I_{OL} =	0.6 64 mA	0.6 60 mA	0.6 52 mA	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., GND < V_{IN} < 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_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. GND ≤ V_{OUT} ≤ V_{CC} Outputs Disabled	±10	±10	±10	µA

Table 3-18. VMEbus Signals AM5, AM4, AM3, AM2, AM1, AM0, IRQ*, BERR*[1]

Parameter	Description	Test Conditions	Comm.	Industrial	Military	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} =	2.4 –16 mA	2.4 –10 mA	2.4 –9 mA	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} =	0.6 48 mA	0.6 44 mA	0.6 38 mA	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., GND < V_{IN} < 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_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. GND < V_{OUT} < V_{CC} Outputs Disabled	±5	±5	±10	µA

Note:

1. The BERR* signal has an on-chip pull-up resistor. For this signal the I_{OZ} value is modified by *Table 3-21*.

Table 3-19. All Other Output Signals[2]

Parameter	Description	Test Conditions	Comm.	Industrial	Military	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} =	2.4 –16 mA	2.4 –10 mA	2.4 –9 mA	V
V_{OL}	Minimum Low-Level Output Voltage	V_{CC} = Min., I_{OL} =	0.6 20 mA	0.6 18 mA	0.6 16 mA	V
I_L	Maximum Input Leakage Current	V_{CC} = Max., GND < V_{IN} < VCC	±5	±5	±5	µA
V_{IK}	Input Clamp Voltage	V_{CC} = Min., I_{IN} = –18 mA	–1.2	–1.2	–1.2	V
I_{OZ}	Maximum Output Leakage Current	V_{CC} = Max. GND < V_{OUT} < V_{CC} Outputs Disabled	±5	±5	±10	µA

Note:

2. Some signals have an on-chip pull-up or pull-down resistors. For these signals I_{OZ} value is modified.

Table 3-20. Capacitance - All Signals

Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 1 MHz, V_{CC} = 5.0V	10	pF
C_{OUT}	Output Capacitance		10	pF

Table 3-21. Pullup/Pulldown Current - All Signals

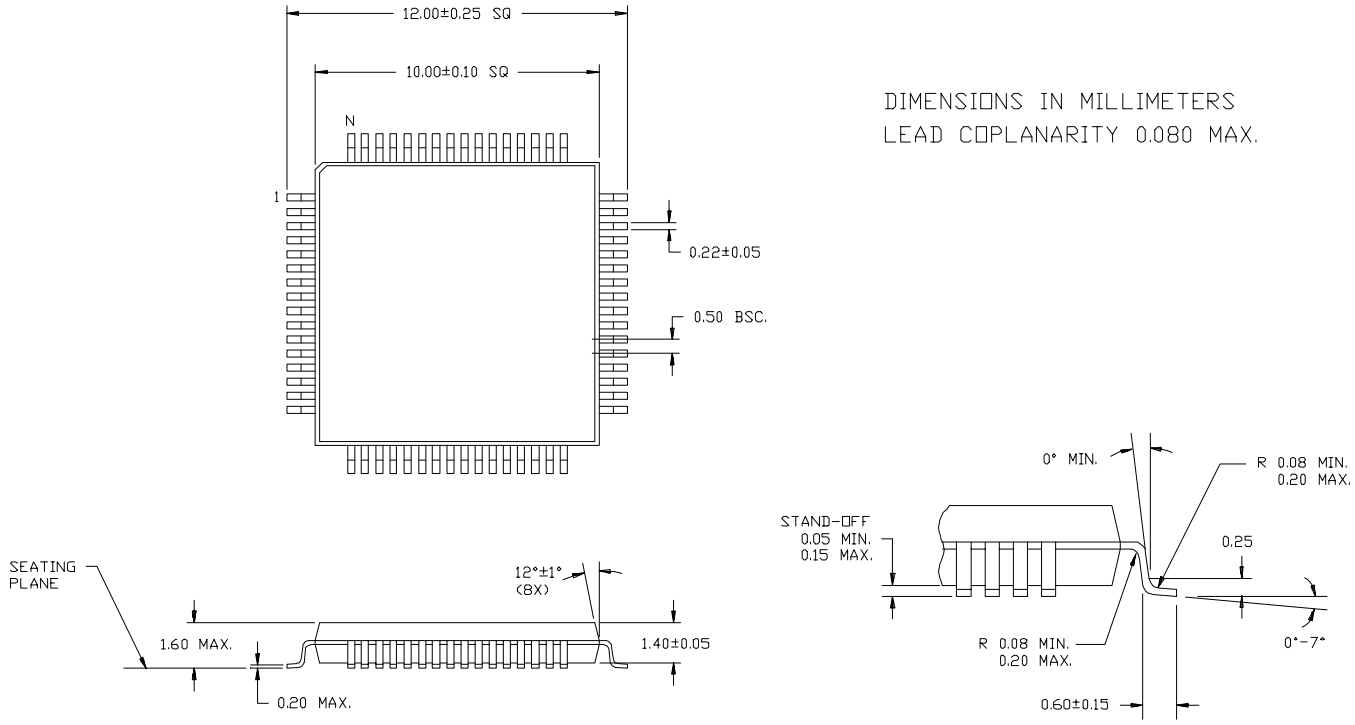
Parameters	Description	Test Conditions	Typ.	Max.
I_{PU}	Input Pullup Current	T_A =–55°C, V_{CC} = 5.5V V_{IN} = GND	100 mA	250 µA
I_{PD}	Input Pulldown Current	T_A = –55°C, V_{CC} = 5.5V V_{IN} = V_{CC}	–100 mA	–250 µA

Table 3-22. Operating Current (CY7C960/CY7C961)

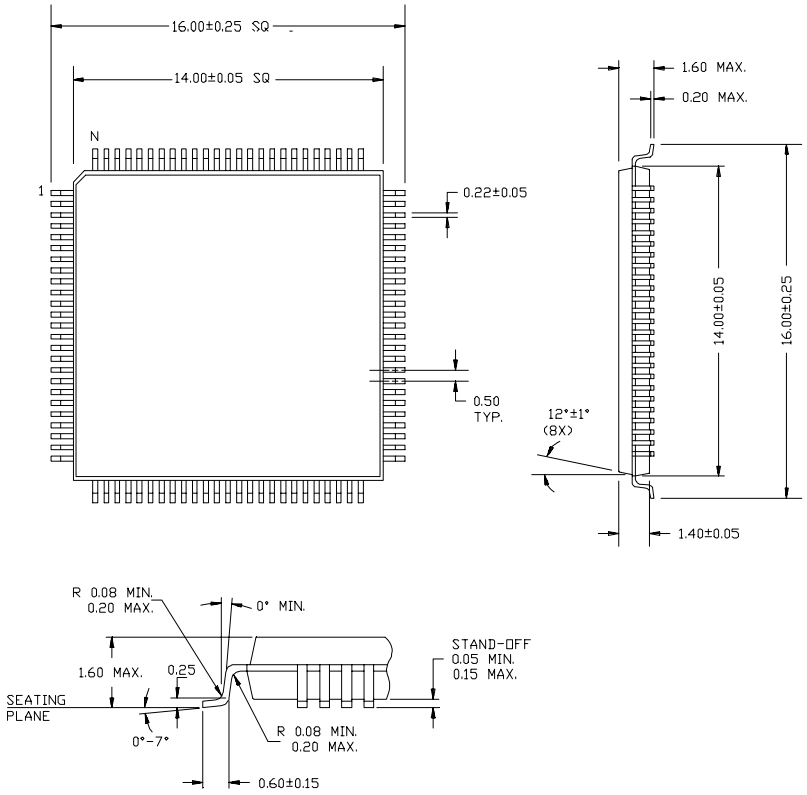
Parameters	Description	Test Conditions	Max.	Units
I_{DD}	Maximum Operating Current	No external DC load	100	mA

3.14
Package Diagrams

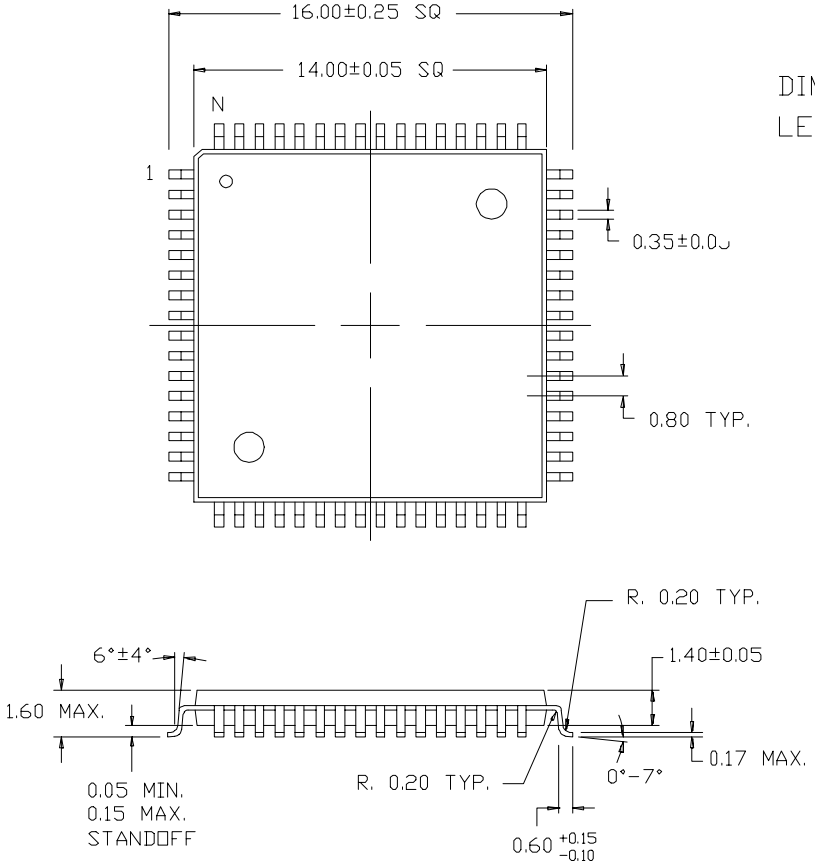
64-Pin Thin Quad Flat Pack A64



100-Pin Plastic Thin Quad Flat Pack (TQFP) A100



64-Lead Plastic Thin Quad Flatpack N65



Section 4

The CY7C964 Bus Interface Logic Circuit

4.1 Introduction

The CY7C964 is a flexible collection of byte (8-bit) wide transceivers, latches, counters, multiplexers, and comparators that provide bus interface designs with a low-cost alternative to PLDs, ASICs, or discrete logic devices. It is based on a standard cell design that incorporates patented line drivers for reduced ground bounce and high noise immunity. The CY7C964 is a companion part to Cypress's VIC068A, VIC64, CY7C960, and CY7C961 VMEbus interface controller devices. It is completely compatible with all operating modes of these devices, such as dual-address path, block transfer boundary crossing conditions, block transfer initialization cycle, local DMA control, and D64 VMEbus block transfers. Signal-naming conventions correspond directly to the VIC068A/VIC64/CY7C960/961 buffer control signals, and the CY7C964 can be directly connected to these corresponding signals.

The CY7C964 can also be used as a generic-interface building block. Some examples include low-cost slave VMEbus controllers, VSB interfaces, or other interface applications. CY7C964s are cascadable, allowing easy interfacing to buses of any width. By combining multiple logic functions into one discrete part, the CY7C964 saves board space and reduces power consumption, which is becoming increasingly important to designers.

4.2
Features

- Directly connects to VIC068A, VIC64, CY7C960, or CY7C961
- Has internal counters for block transfer and local DMA address control
- Has internal multiplexers for D64 64-bit VMEbus block transfers
- Has internal comparators for address decoding
- Supports VIC068A/VIC64 dual-address-path option
- Has cascadable operation
- Directly drives VMEbus address and data bus signals
- Directly drives local address and data bus signals
- Reduces components for a compact board design
- Has low power requirements
- Is available in a 64-pin TQFP or 68-pin PGA

4.3 Interfacing to Cypress VMEbus Interface Controllers

Previously, interfacing the VIC068A to the VMEbus required a significant number of LSI and MSI devices. With the advent of 64-bit VMEbus block transfers and the VIC64, the external discrete device count for a full functional interface has expanded. The CY7C964 has been developed to combat this problem by incorporating the functions of much of this external logic into a single package. Using the CY7C964 shortens system design, debug, and manufacturing cycle times. Design engineers are relieved from the burden of performing critical or worst-case timing analysis on the VMEbus and VIC buffer control signals. Local control signals other than those directly interfaceable to the interface controllers have been kept to a minimum.

A full function D64 VMEbus interface implemented using the CY7C964, VIC64, and all VME-bus interface local support logic includes the following features:

- block transfer support for D16, D32, and D64 VMEbus transfers
- dual-path address operation (allowing single-cycle transfers during master block transfer interleave periods)
- slave block transfers during master block transfer interleave periods
- fully programmable slave VMEbus address decoding
- write posting
- the VIC mail box interrupt message support

The interface can be broken into five functional sections for the purpose of discussion. These sections are:

- VMEbus signal group
- Buffer control signal group
- CY7C964 local signal group
- CY7C964 address comparator and local signal group
- Local data swap buffer logic

The focus of this section is the CY7C964. Each of the interface functional blocks are examined from this perspective. For additional information on the signals described within this section, consult the VMEbus Specification (IEEE 1014) and/or the section of this book that describes the controller you are connecting.

4.3.1 VMEbus Signal Group

This group includes the VMEbus address and data signals. Each CY7C964 provides support for 8 bits of VMEbus address and data. Three CY7C964s are necessary for 32-bit interface

applications when used with the VIC068A or VIC64. Four CY7C964s are used with CY7C960 and CY7C961 for full-function slave interfaces. The A[7:0] and D[7:0] transceivers on the CY7C964 furnish a high drive strength, allowing direct connection to the respective address and data signals on the VMEbus backplane. With any of Cypress's family of VMEbus interface controllers generating the control information, the CY7C964s meet VMEbus worst-case timing and drive-strength requirements for all forms of data transfers.

Table 4-1. VMEbus Signals

Signal	Description	Interface Comments
D[7:0]	VMEbus compatible data signals	Directly connect to VMEbus P1 and P2 connectors
A[7:0]	VMEbus compatible address signals	Directly connect to VMEbus P1 and P2 connectors

4.3.2 Buffer Control Signal Group

This group includes all of the address and data buffer control signals. A major design time savings is realized using the CY7C964s as all of these signals directly connect to the chosen controller or are wired to a steady state value. The buffer control interface is simple and straightforward with a few minor exceptions. *Table 4-2* specifies these connections for the VIC068A and VIC64 controllers.

Table 4-2. Buffer Control Signals Connection for VIC068A and VIC64

Signal	Description	Interface Comments
LADO	Latch address out	Directly connect to VIC LADO on all CY7C964s
LADI	Latch address in	Directly connect to VIC LADI on all CY7C964s
LEDO	Latch enable data out	Directly connect to VIC LEDO on all CY7C964s
LEDI	Latch enable data in	Directly connect to VIC LEDI on all CY7C964s
ABEN*	VMEbus address bus enable	Directly connect to VIC ABEN* on all CY7C964s
DENO*	Data enable output	Directly connect to VIC DENO* on all CY7C964s
D64	D64 block transfer mode enable	Directly connect to VIC64 SCON/D64 pin on all CY7C964s. Tie this input LOW on all CY7C964s when using VIC068A.
BLT*	Block transfer enable	Directly connect to VIC BLT* on all CY7C964s
LAEN	Local address enable	Directly connect to VIC LAEN on all CY7C964s
DENIN*	Primary data enable in signal	Directly connect to VIC DENIN1* on NMSB and MSB CY7C964s, directly connect to VIC DENIN* on LSB CY7C964
DENIN1*	Companion data enable in signal	Directly connect to VIC DENIN* on NMSB and MSB CY7C964s, directly connect to VIC DENIN1* on LSB CY7C964

4.3.3 CY7C964 Local Signal Group

The CY7C964 local signal group consists of the VMEbus and local block-transfer counter count-enable daisy-chains. These signals enable the local and VMEbus higher-order address counters; two local address counters (a master block transfer and a slave block transfer) and a single VMEbus address counter. The local address counters share the LCIN*/LCOUT* count-enable daisy-chain. These signals are multiplexed within the CY7C964 and enable counting for the proper counter depending on the current state of the interface. The VCIN*/ VCOUT* daisy-chain is dedicated to the VMEbus address counter on the device. When the VCIN* or LCIN* are held Low, counting is enabled for the appropriate counters within the device. The VCIN* or LCIN* signals do not advance the counters, they just enable counting. The counters increment when these signals are active and the proper increment count control logic sequence occurs. The controller advances the address counters at the proper time during VMEbus and local DMA block transfer operations. For further information on the counter advance control sequence, refer to Chapter 4.5, CY7C964 Operation.

Table 4-3. CY7C964 Local Signals for VIC068A and VIC64

Signal	Description	Interface Comments
LCIN*	Local address counter count enable	On LSB CY7C964, tie this input Low. On the NMSB device, directly connect to the LCOUT* of the LSB device. For the MSB CY7C964, connect to the LCOUT* of the NMSB device.
LCOUT*	Local address counter carry out	On LSB CY7C964, connect this output to the LCIN* input. On the NMSB CY7C964, connect this output to the MSB LCIN* input. For the MSB device do not connect this output.
VCIN*	VMEbus address counter count enable	On LSB CY7C964, tie this input Low. On the NMSB device, directly connect to the VCOUT* of the LSB device. For the MSB CY7C964, connect to the VCOUT* of the NMSB device.
VCOUT*	VMEbus address counter carry out	On LSB CY7C964, connect this output to the VCIN* input. On the NMSB CY7C964, connect this output to the MSB VCIN* input. For the MSB device do not connect this output.

4.3.4 CY7C964 Address Comparison and Local Signal Group

The implementation of this group of CY7C964 signals is application specific. The MWB* signal and FC1 signal have been included in this section because they are locally generated signals required by the VIC and output directly from the CY7C961. These two signals differ slightly on the VIC; MWB* is an input only, while FC1 is a bidirectional signal that can be driven by

the VIC. On the CY7C964 the MWB* and FC1 signals are inputs. These signals should be directly connected to the respective local signals on VIC or CY7C961. For CY7C960 designs, MWB* should be connected to VCC and FC1 should be connected to GND.

Table 4-4. CY7C964 Address Comparison and Local Signals for VIC068A and VIC64

Signal	Description	Interface Comments
FC1	Function code 1 signal	Directly connect on all CY7C964s to the same local signal that drives the FC1 signal on the VIC.
MWB*	Module wants VMEbus	Directly connect on all CY7C964s to the same local signal that drives the MWB* signal on the VIC.
LDS	Load register select signal	Should be directly connected to LA2 for VIC systems with 32-bit local bus. Refer to text below for further information.
STROBE	Latch register control signal	Chip select-like signal for CY7C964 internal comparator mask and comparison registers. See text below for further information.
VCOMP*	VMEbus address comparator output	Needs a small amount of external glue logic to validate an combine signals in a parallel high-performance fashion.

The CY7C964s contain a high-performance programmable VMEbus address equality comparator. The comparator is controlled by two internal, write-only registers: a mask and a compare register. The mask register enables and disables bits of the comparator, and the compare register stores the data pattern that inputs are compared against. VCOMP* is the active-Low comparator match output signal. VCOMP* is driven Low by the CY7C964 when the bit pattern on pins A[7:0] match enabled bits of the compare register. Setting mask register bits to 0 enables the corresponding bits of the compare register. Loading bits of the mask register with 1s places bits of the compare register in don't care or "match anything" state. Loading the mask register with all 0s forces the compare register to match all bits of the pattern on A[7:0]. Setting the mask register to all 1s effectively disables the on-board comparator. VCOMP* will always be Low.

These registers are loaded by supplying the proper data on LD[7:0] and the register address on MWB* and LDS signals. The STROBE input is used to qualify the address and latch the data into the proper internal register. *Figure 4-1* and *Figure 4-2* show the waveforms needed to load the compare and mask registers.

This load cycle operates as follows:

1. The state of LDS and MWB* are latched on the falling edge of STROBE.
2. The data is loaded into the selected register on the rising edge of the STROBE signal.
3. MWB* must be held inactive (High): the state of LDS selects the register to load.
4. If LDS is High at the falling edge of STROBE, the compare register will be loaded; if LDS is Low the data is written to the mask register.

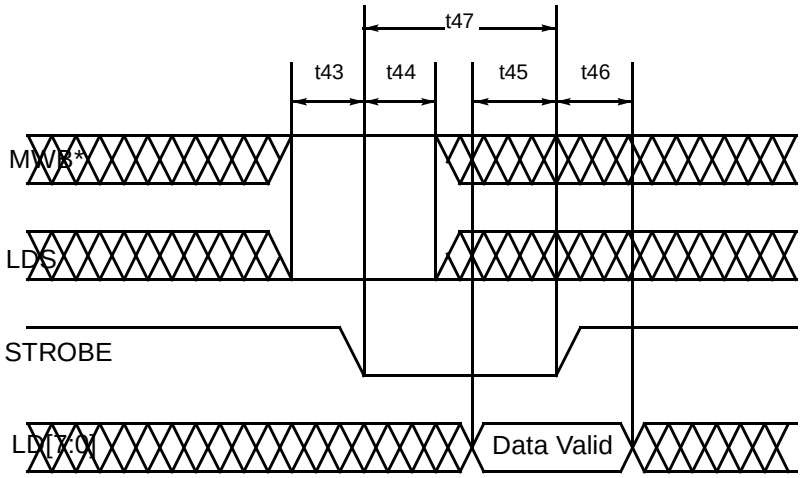


Figure 4-1. Mask Register Load Timing

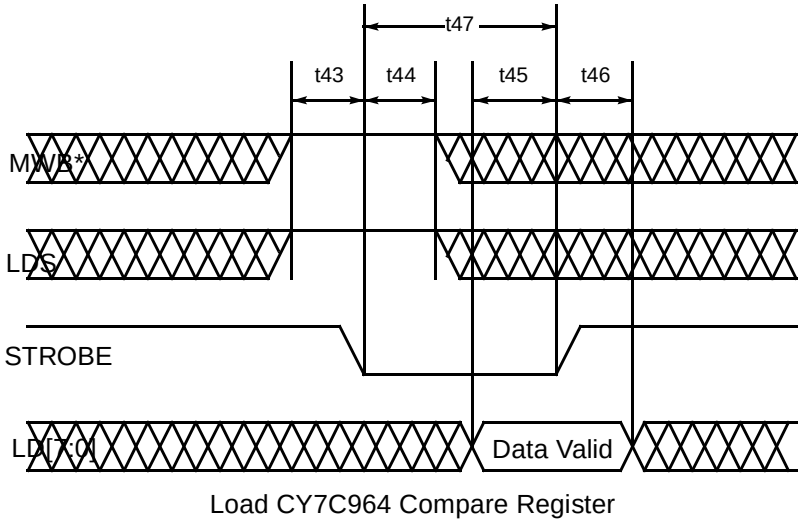


Figure 4-2. Compare Register Load Timing

This load cycle can be generated by decoding a separate address region or chip select signal for the CY7C964 comparator registers. In applications with a 32-bit local bus, it is desirable to load all three CY7C964s in parallel by having the host processor perform a 32-bit write cycle to the address region that will activate STROBE. The select signal for the address region is connected to the STROBE input on all CY7C964s. Boards that implement VIC068A or VIC64 interface should connect LDS to LA2, thereby decoding the mask register at the *base address* of the address region and the compare register at the *base address + 4*. LDS also controls the operation of the D64 block transfer data multiplexer/demultiplexer. If it is not connected to LA2, this logic will not operate properly. For CY7C960 and CY7C961 designs, LDS is connected directly to the controller.

The mask and compare registers can be set to select any contiguous address region on the VMEbus. These registers do not preload and therefore power up in an unknown state. It is advisable to initialize these registers as soon as possible in the system boot sequence. Note that the act of writing the compare register clears the mask register.

The CY7C964 comparator output signal VCOMP* supplies the result from the equality compare logic. VCOMP* drives Low when the input matches the loaded conditions. The CY7C964 VCOMP* signals are not directly compatible with the VIC SLSEL0* and SLSEL1* slave select signals. The short (10 ns) address set-up time to AS* active for VMEbus slave boards does not meet the worst case compare out delay of the CY7C964 VCOMP* signal. combining this with the potential output glitching that can occur with an asynchronous comparator can cause problems for the VIC. It is recommended that the VCOMP* signal be externally filtered prior to being used with the VIC SLSEL0* or SLSEL1* signal. Most applications will require some external comparison logic to combine VCOMP* signals from the NMSB and MSB device, furnishing finer grained VMEbus decoding: this logic can also be used to filter the CY7C964 VCOMP* signals. Note that CY7C960 and CY7C961 have a built-in decode delay feature, which eliminates the need for external filtering.

4.3.5 Local Data Swap Buffer Logic

Local Data Swap Buffer logic is a requirement for all 32-bit local bus designs that perform 8- or 16-bit transfers. The swap buffer moves data to and from the lower section of the VMEbus, D[15:0], to the upper segments of the local bus, D[31:16]. VMEbus requires that all 8- and 16-bit data transfers be performed on the D[15:0] section of the bus. The CY7C964s work properly with the VIC-controlled or CY7C960/961-controlled swap buffer. For the VIC case, if an isolation buffer is implemented, care should be taken to ensure that the local data bus is driven to the least-significant CY7C964 during address/mask register programming cycles. One way to ensure this is to assert the CS* and PAS* signals to the VIC068/VIC64, thus causing VIC to assert the ISOBE* signal to the isolation buffer.

4.4 Signal Descriptions

4.4.1 VMEbus Signals

A[7:0]

Input: Yes
Output: Yes, three-state
Drive: 48 mA

These are VMEbus-compatible address signal transceivers that can be directly connected to the VMEbus. A0 is the least-significant address bit. In flow-through modes of operation, these signals correspond one for one with local interface signals LA[7:0].

In VMEbus interface applications, including those using the VIC068A, VIC64, CY7C960, or CY7C961, these signals should be connected to the VMEbus address bus sequentially.

In CY7C960/961 designs, A0 on the least-significant CY7C964 should be connected to the VMEbus LWORD* signal.

D[7:0]

Input: Yes
Output: Yes, three-state
Drive: 48 mA

These are VMEbus-compatible data signal transceivers that can be directly connected to the VMEbus. D0 is the least-significant data bit. In flow-through modes of operation, these signals correspond one for one with local interface signals LD[7:0].

In VMEbus interface applications, including those using the VIC068A, VIC64, CY7C960, or CY7C961, these signals should be connected to the VMEbus data bus sequentially.

4.4.2 Local Signals

LA[7:0]

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

These are medium drive-strength local address transceivers that allow direct connection to memory, microprocessors and/or peripheral controllers. LA0 is the least-significant local address signal. In flow-through operating modes, these signals correspond one for one with the VMEbus signals A[7:0].

When implementing conventional VMEbus interfaces or using the CY7C964 with the VIC068A, VIC64, CY7C960, or CY7C961, these signals should be connected to the local address bus sequentially.

LD[7:0]

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

These are medium drive-strength local data transceivers that allow direct connection to memory, microprocessors and/or peripheral controllers. LD0 is the least-significant local data signal. In flow-through operating modes, these signals correspond one for one with the VMEbus signals D[7:0].

When implementing conventional VMEbus interfaces or using the CY7C964 with the VIC068A, VIC64, CY7C960, or CY7C961, these signals should be connected to the local data bus sequentially.

ABEN*

Input: Yes
Output: No

This is the VMEbus Address Bus ENable control signal. This signal controls the state of the VMEbus address transceivers A[7:0]. When asserted (driven Low), the transceivers are configured as outputs and are enabled.

When using the CY7C964 with the VIC068A, VIC64, CY7C960, or CY7C961 to implement VMEbus interfaces, this input should be connected to the ABEN* output of the controller.

BLT*

Input: Yes
Output: No

This signal controls the VIC-compatible block transfer operations that require local Direct Memory Access (DMA). If this input is driven Low, the CY7C964 will operate in the appropriate VIC-compatible block transfer mode (dependent on the states of the other buffer control logic).

When using the CY7C964 with the VIC068A, VIC64, or CY7C961, this signal can be directly connected to the controller BLT* pin. A rising edge of BLT* increments the local address counters if LCIN* is Low. Refer to the LCIN* signal description and Chapter 4.5, CY7C964 Operation, for the further information on the local address counter function.

D64

Input: Yes
Output: No

This signal is used to indicate to the CY7C964 that a VMEbus D64 block transfer is in progress. When High, the CY7C964 is instructed to use the high-performance two-state pipeline and multiplex or demultiplex 64-bit data to and from the VMEbus address bus.

When used in conjunction with the VIC64, CY7C960, or CY7C961, this pin can be directly connected to the SCON*/D64 signal. For applications that only support 32-bit block transfers, as is the case with the VIC068A, this input should be tied Low.

DENIN*

Input: Yes
Output: No

The Data ENable IN signal is used to control the three-state data transceivers LD[7:0]. If a logic Low level is presented to this input, LD[7:0] transceivers will be enable. In conventional VMEbus designs, this signal would need to be driven during master read or slave write operations.

When used in conjunction with the VIC068A, VIC64, CY7C960, or CY7C961, this input is typically connected to DENIN1* for CY7C964's controlling bus data lines D16 through D31, and DENIN* for the CY7C964's controlling bus data lines D8 through D15.

DENIN1*

Input: Yes
Output: No

The Data ENable IN 1 signal is used in conjunction with DENIN* to latch data from the VMEbus and provide a second enable control of the LD[7:0] transceivers for D64 transactions.

When used in conjunction with VIC64, CY7C960, or CY7C961, this input is typically connected to DENIN* for CY7C964's controlling bus data signals D16 through D31, and DENIN1* for the CY7C964's controlling bus data signals D8 through D15.

DENO*

Input: Yes
Output: No

The Data ENable Out signal is used to control the three-state transceivers D[7:0]. If a logic-Low level is presented to this input, the D[7:0] transceivers will be enabled. In conventional VMEbus design, the D[7:0] signals will be directly connected to the VMEbus. Used in this manner, this input must be asserted during master writes and slave read operations.

When used in conjunction with the VIC068A, VIC64, CY7C960, or CY7C961, this signal should be connected to the DENO* output on the controller.

FC1

Input: Yes
Output: No

The Function Code 1 input is used by the CY7C964 during block transfer operations to determine the source for the local address signals LA[7:0]. If the input is driven High, the internal Local DMA counter is selected as the source for LA[7:0]. If this input is Low, the Slave Block Transfer counter is the source for LA[7:0].

When used with the VIC068A, VIC64, or CY7C961, this signal can be directly connected to the FC1 pin of the controller. The controller will drive this pin to the proper level for slave or block transfer operations when it is master of the local system bus.

When used with the CY7C960, this signal should be tied to GND.

LCOUT*

Input: No
Output: Yes
Drive: 8 mA

The Local Carry Out signal is used by the CY7C964 for cascading the local address counter chains. This signal will drive Low when the local address counter has reached the maximum count (255). The signal generates a synchronous count enable for the next-most-significant CY7C964 in the cascade chain. This signal alone does not cause the local address counter to increment.

When cascading the CY7C964s, this signal should be connected to the LCIN* pin of the next most significant device. Refer to the description of the LCIN* pin for further information on the operation of the local address counters.

LDS

Input: Yes
Output: No

The Local Data Select input has two main functions: (1) as a control input for the data to address bus multiplexer during D64 VMEbus block transfers, (2) as a select bit for configuring the CY7C964 address comparison and mask registers. Refer to Chapter 4.5, CY7C964 Operation, for further information about how data is steered to and from the VMEbus address bus during block transfers. Typically this input will be connected to LA2 for VIC controllers and to the LDS output of CY7C960/961.

When configuring the CY7C964 internal address mask and address compare registers, this pin in conjunction with MWB* selects which registers to load. During a comparator register load cycle, LDS High will select the Address Compare register; otherwise the Address Mask register will be selected.

LADI

Input: Yes
Output: No

The Latch ADdress In signal controls a transparent VMEbus to local address latch within the CY7C964. When this input is High, the device will latch the data present on A[7:0]. This function is useful when building VMEbus interfaces for latching the VMEbus address during a slave access. If LADI is Low, the internal address latch will be in a flow-through mode. LADI is also used to increment the slave block transfer local address counter. LADI is used in conjunction with LAEN to control the operation of the VMEbus to local address section of the CY7C964. For more information, refer to the description of LAEN.

When using the CY7C964 to implement VMEbus interfaces, this signal is used to maintain the local address during slave read and write cycles. For VMEbus designs that use a Cypress controller, this input should be connected to the LADI output of the controller.

LAEN

Input: Yes
Output: No

The Local Address ENable signal controls the three-state enable for signals LA[7:0]. When this signal is High, LA[7:0] drive the address local bus. Driving the signal Low places LA[7:0] in the high-impedance/input state.

When using the CY7C964 to implement VMEbus interfaces, this signal is driven High to maintain the local address VMEbus during slave cycles. For VMEbus interfaces using the VIC068A or VIC64, this pin should be connected to the LAEN output on the VIC. For CY7C960/961 designs, only the least-significant CY7C964 is connected to the LAEN output of the controller.

LEDI

Input: Yes
Output: No

The Latch Enable Data In signal controls a transparent VMEbus-to-local-data-bus latch within the CY7C964. When this input is High, the device will latch the data present on D[7:0]. This function is useful when building VMEbus interfaces for latching the VMEbus data during a master or slave access. If LEDI is Low, the internal address latch will be in a flow-through mode.

LEDI is used in conjunction with DENIN* and DENIN1* to control the operation of the address VMEbus-to-local-data section of the CY7C964.

When implementing VMEbus interfaces with the CY7C964, LEDI can be used to maintain the local data during VMEbus master read and slave write cycles. For VMEbus designs that use a Cypress controller, this input should be connected to the LEDI output of the controller.

LEDO

Input: Yes
Output: No

The Latch Enable Data Out signal controls a transparent local-data-to-VMEbus-data latch within the CY7C964. When this input is High, the device will latch the data present on LD[7:0]. This function is useful when building VMEbus interfaces for latching the VMEbus data during a master or slave access. If LEDO is Low, the internal address latch will be in a flow-through mode.

LEDO is used in conjunction with DENO* to control the operation of the local-to-VMEbus-data section of the CY7C964.

When implementing VMEbus interfaces with the CY7C964, LEDO can be used to maintain the VMEbus data during VMEbus master write and slave read cycles. For VMEbus designs that use a Cypress controller, this input should be connected to the LEDO output of the controller.

LADO

Input: Yes
Output: No

The Latch Address Out signal controls a transparent local-to-VMEbus address latch within the CY7C964. When this input is High, the device will latch the data present on LA[7:0]. This function is useful when building VMEbus interfaces for latching the local address during a master transfer. If LADO is Low, the internal address latch will be in a flow-through mode.

LADO is used in conjunction with ABEN* to control the operation of the local-to-VMEbus address section of the CY7C964.

When using the CY7C964 to implement a VMEbus interface, this signal can be used to maintain the local address during VMEbus slave read and write cycles. For VMEbus designs that use a VIC068A, VIC64, or CY7C961, this input should be connected to the LADO output of the controller.

LCIN*

Input: Yes
Output: No

Local Carry IN is a synchronous count enable for both the local master block transfer and slave block transfer local address counters. LCIN* is multiplexed within the CY7C964 and can be routed to either local block transfer address counter. When connected to the master block transfer local address counter, if LCIN* is driven Low, a falling edge on the BLT* signal will increment the address count. When this input is connected to the slave block transfer counter and driven Low, a rising edge of LADI will increment the address count.

When cascading CY7C964s, this signal should be connected to the LCOUT* signal of the next-least-significant device.

MWB*

Input: Yes
Output: No

The Module Wants Bus is a decoding/control signal for the CY7C964 that allows the device to discern the cycle type. When MWB* is active (Low), the CY7C964 assumes that a block transfer initiation cycle or a single-cycle VMEbus transfer is pending. Subsequent assertion of BLT* allows the CY7C964 to enter block transfer mode.

MWB* is also used to decode accesses to the CY7C964 address Mask and Compare registers. To load these registers, MWB* must be inactive (High).

STROBE

Input: Yes
Output: No

The STROBE signal controls the loading of the internal Address Mask and Compare registers. This signal operates in the same manner as an active-Low chip select for these registers. When STROBE is Low and MWB* is High, data present on LD[7:0] will be loaded into either the Address Mask or Compare registers.

Refer to Chapter 4.5, CY7C964 Operation, for further information on the use of MWB* to load the Address Mask and Compare registers.

VCOMP*

Input: No
Output: Yes
Drive: 8 mA

The VMEbus COMPare signal indicates whether the address presented on A[7:0] matches the pattern of the internal Address Compare register. If the two values are determined to match, VCOMP* will drive Low. (Note: The Address Mask register can mask bits of the Compare register, causing these bits to match anything.)

This signal is the output of an asynchronous comparator and is therefore susceptible to glitching during address transitions on A[7:0]. When used in conjunction with the VIC64 or VIC068A for conventional VMEbus implementations, these signals should be de-glitched externally. The decode delay feature of the CY7C960/961 eliminates the need to de-glitch for slave controller designs. External logic is required to cascade VCOMP* comparison outputs.

VCIN*

Input: Yes
Output: No

VMEbus Carry IN is a synchronous count enable for the local address counters. When VCIN* is Low and the device is not operating in the Dual-Address-Path mode, a rising edge on the LADO signal will increment the VMEbus address counter.

When cascading CY7C964s, this signal should be connected to the VCOUT* signal of the next-least-significant device.

For more information on the Dual-Address-Path mode, refer to Chapter 4.5, CY7C964 Operation.

VCOUT*

Input: No
Output: Yes
Drive: 8 mA

The VMEbus Carry Out signal is used by the CY7C964 for cascading the VMEbus address counter chains. This signal will drive Low when the VMEbus address counter has reached the maximum count (255). The signal generates a synchronous count enable for the next-most-significant CY7C964 in the cascade chain. This signal alone does not cause the VMEbus address counter to increment.

When cascading the CY7C964s, this signal should be connected to the VCIN* pin of the next-most-significant device. Refer to the description of the VCIN* pin for further information on the operation of the VMEbus address counters.

4.5 CY7C964 Operation

4.5.1 Overview

The CY7C964 is a general-purpose bus interface device that provides seamless support for the entire family of VMEbus interface controllers. The part is also suitable for many other general-purpose bus interface applications. *Figure 4-3* is the block diagram of the device, showing the array of latches, multiplexers, and counters.

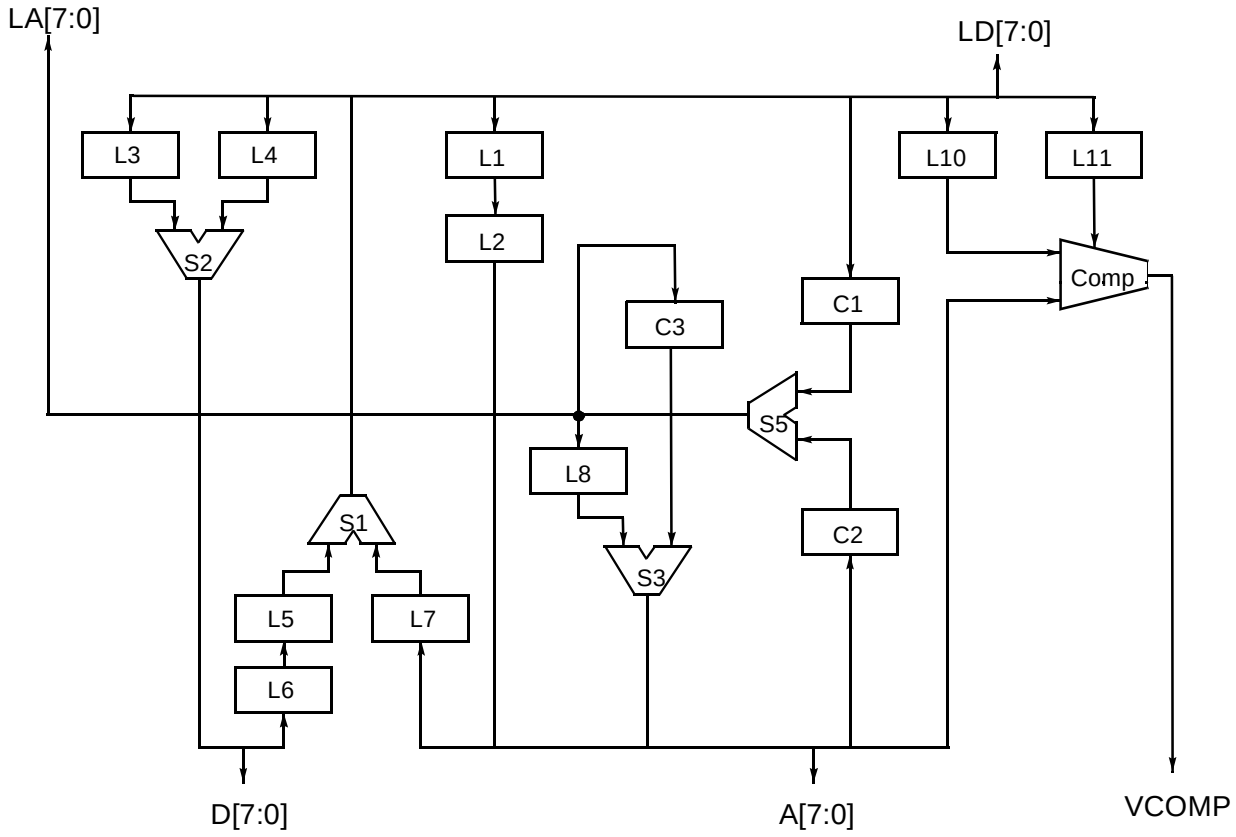


Figure 4-3. CY7C964 Block Diagram

This section of the document dissects the high-level block diagram into lower-level functional blocks. General operational and timing information is presented on a block-by-block basis. This information is provided for designers who wish to implement generically-controlled interfaces. The tables show the control signal logic sequence needed to operate or communicate with each of the functions. Timing parameters are included, which reference the switching characteristics listed later in this document.

The CY7C964 operation is controlled by the combination of external control signals and internal state logic. Three internal asynchronous state bit control the operating mode of the device. These bits are referred to as BLT_STATE, BLT_INIT, and DUAL_PATH. The BLT_STATE bit is set during block transfer operations. The block transfer initiation cycle generates a rising edge on the BLT_INIT signal. The DUAL_PATH signal is the output of a transparent latch within the device that latches the state of LADO. These internal state bits must be in the proper state to use and communicate with the internal logic of the device. The functional tables include references to these signals when their state is required for the operation. The designer must perform the appropriate cycle to the device to set or clear these latches as needed prior to the desired functional cycle. The internal latch signals and all other control signals that are not called out within the tables for a specific operation can be considered don't cares.

Table 4-5. Examples of References to Control Signals Within Functional Tables

Note 1: BLT_STATE=(/BLT* • /MWB*)+(BLT_STATE • (/BLT*+/MWB*+LAEN))
Note 2: BLT_INIT=(/BLT_STATE • /BLT* • /MWB*)+(BLT_INIT • /BLT* • /MWB*)
Note 3: DUAL_PATH=(LADO • BLT_INIT)+(DUAL_PATH • /BLT_INIT)

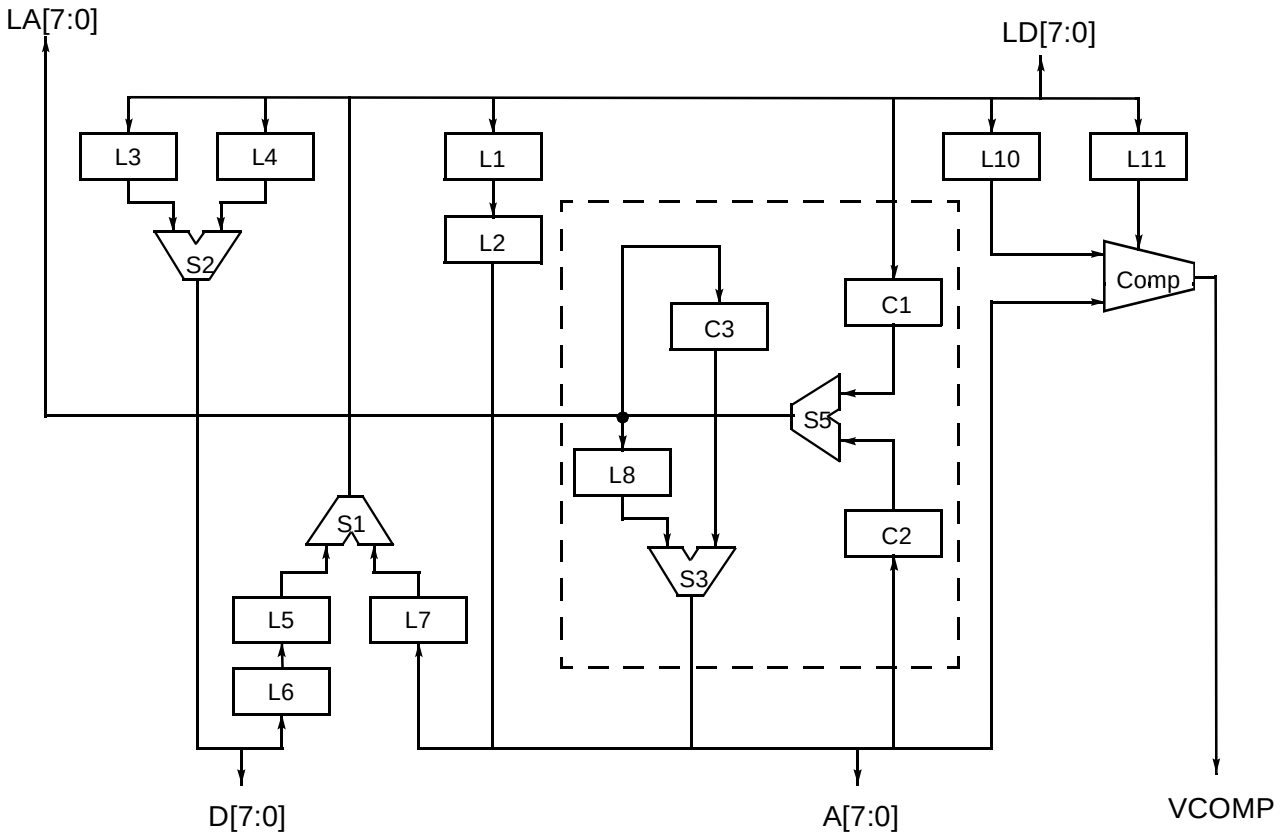


Figure 4-4. CY7C964 Block Diagram: Address Counters and Address Multiplexers

4.5.2 Master Block Transfer Local Address Counter (C1)

Master Block Transfer Local Address Counter supplies the local address to LA[7:0] during master block transfer operations. This 8-bit synchronous counter is cascadable using the LCIN*/LCOUT* daisy-chain. The counter powers up in an uninitialized state and must be initialized for predictable operation. The counter loads from LD[7:0] when both MWB* and BLT* control signals are active (Low). To enable the counter onto LA[7:0], an internal asynchronous latch (BLT_STATE) must be set and Local Address Multiplexer S5 must select counter C1. A falling edge on MWB* or BLT* increments C1. FC1 controls S5. If it is High, as shown in *Table 4-7*, C1 is selected. The internal latch and S5 multiplexer must also be in the proper state to increment the counter. For further information on the S5 Local Address Multiplexer, see section 4.5.3.

Table 4-6. Master Block Transfer Local Address Counter Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
C1	Load counter	LD[7:0] valid to falling edge of MWB*	BLT*=0, LAEN=0	Set-up t48
				Hold t49
		LD[7:0] valid to falling edge of BLT*	MWB*=0, LAEN=0	Set-up t50
				Hold t51
	Increment counter	MWB* falling edge to LA[7:0] valid	LAEN=1, FC1=1, BLT_STATE=1[1]	Prop t54
		BLT* falling edge to LA[7:0] valid	LAEN=1, FC1=1, BLT_STATE=1[1]	Prop t54
		LCIN* valid to MWB* falling edge	LAEN=1, FC1=1, BLT_STATE=1[1]	Set-up t52
				Hold t53
		LCIN* valid to BLT* falling edge	LAEN=1, FC1=1, BLT_STATE=1[1]	Set-up t52
				Hold t53
	Counter carry out at terminal count	MWB falling edge to LCOUT* valid	LAEN=1, FC1=1, BLT_STATE=1[1]	Prop t55
		BLT* falling edge to LCOUT* valid	LAEN=1, FC1=1, BLT_STATE=1[1]	Prop t55
		LCIN* valid to LCOUT* valid	LAEN=1, FC1=1, BLT_STATE=1[1]	Prop t56
	Minimum pulse widths	BLT*	LAEN=1, FC1=1, BLT_STATE=1[1]	t57
		MWB*		t57

4.5.3 Local Address Multiplexer (S5)

The Local Address Multiplexer S5 routes the outputs of counters C1 or C2 to signals LA[7:0]. The local address counter carry chain LCIN*/LCOUT* is also controlled by this multiplexer. If FC1 is High, counter C1 drives LA[7:0] and LCIN*/LCOUT* are visible/driven by C1, respectively. When FC1 is Low, C2 drives LA[7:0] and is attached to the LCIN*/LCOUT* daisy-chain.

Table 4-7. Local Address Multiplexer Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
S5	Select C1 counter	FC1 rising edge to LA[7:0] valid		Prop t85
	Select C2 counter	FC1 falling edge to LA[7:0] valid		Prop t86
	Select C1 carry chain	FC1 rising edge to LCOUT* valid		Prop t88
	Select C2 carry chain	FC1 falling edge to LCOUT* valid		Prop t87

4.5.4 Slave Block Transfer Local Address Counter/Latch (C2)

The Slave Block Transfer Local address counter provides two functions: a counter for slave block transfer operations and a transparent address latch for VMEbus slave operations. When the latch control signal LADI is held Low the counter is in a transparent mode: Logic levels present will flow through the device to the inputs of the local address multiplexer S5. FC1 controls the S5 multiplexer and must be Low to select counter C2 as the source for LA[7:0]. Driving either LADI or D64 High exclusively latches the data present on A[7:0]. The counter increments if LCIN* is Low, D64 is High, and a rising edge occurs on LADI. The contents of the counter/latch are enabled onto the local data bus when LADI and FC1 are Low and D64 is High. Counter C2 is not initialized at power-up; for predictable operation the counter should be loaded prior to use.

Table 4-8. Slave Block Transfer Local Address Counter/Latch Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
C1	Load counter	A[7:0] valid to D64 rising edge	LADI=0	Set-up t58
				Hold t59
		A[7:0] valid to LADI rising edge	D64=0	Set-up t60
				Hold t61
	Increment counter	LADI rising edge to LA[7:0]	D64=1, FC1=0	Prop t64
		LCIN* active to LADI rising edge	D64=1	Set-up t62
				Hold t63
	Counter carry out at terminal count	LADI rising edge to LCOUT*	D64=1, FC1=0	Prop t65
	Minimum pulse width	LADI		t66

4.5.5 Master Block Transfer VMEbus Address Counter (C3)

The VMEbus Master Block Transfer Address stores and increments the VMEbus address during master block transfer operations. The counter loads from LA[7:0] on the rising edge of MWB* provided that the internal asynchronous latch BLT_STATE is set. The contents of the counter are enabled onto the A[7:0] pins if the internal asynchronous latch bits BLT_STATE and multiplexer S3 are in the appropriate state. Depending on the state of DUAL_PATH, either the rising or the falling edge of LADO increments C3. Counter C3 uses the VCIN*/VCOUT* counter daisy-chain. This counter is uninitialized at power-up and should be initialized prior to use for predictable operation.

Table 4-9. Master Block Transfer VMEbus Address Counter Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
C3	Load counter	LA[7:0] valid to rising edge of MWB*	BLT_STATE=1[1] BLT_INIT=1	Set-up t67
				Hold t68
	Increment counter	LADO falling edge to A[7:0]	BLT_STATE=1[1] DUAL_PATH=1[3] BLT_INIT=0	Prop t69
		LADO rising edge to A[7:0]	BLT_STATE=1[1] DUAL_PATH=0 BLT_INIT=0	Prop t70
		VCIN* valid to LADO rising/falling edge		Set-up t134
				Hold t135
	Counter carry out	LADO falling edge to VCOUT* valid	BLT_STATE=1[1] DUAL_PATH=1[3] BLT_INIT=0	Prop t71
		LADO rising edge to VCOUT* valid	BLT_STATE=1[1] DUAL_PATH=0 BLT_INIT=0	Prop t72
	Minimum pulse width	LADO (High)		t73
		LADO (Low)		t73

4.5.6 VMEbus Address Latch (L8) and Multiplexer (S3)

The VMEbus Address Latch and Multiplexer selects the source for the VMEbus address signals A[7:0]. The information supplied to A[7:0] originates at one of three sources: the D64 block transfer data pipeline latch L2, the VMEbus master block transfer counter C3, or the VMEbus address latch L8. *Table 4-10* shows how to latch information into the VMEbus address latch L8 and control the selection of the source for signals A[7:0]. Latch L8 is uninitialized at power-up and for predictable operation should be loaded prior to use.

Table 4-10. VMEbus Address Latch and Multiplexer Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
S3	Select L8	D64 falling edge to A[7:0] valid	BLT_STATE=1[1]	Prop t83
S3	Select L8	ABEN* falling edge to A[7:0] valid	BLT_STATE=1[1]	Prop t84
S3	Select L8	D64 falling edge to A[7:0] valid	BLT_STATE=0	Prop t81
L8	Load L8	LA[7:0] valid to LADO rising edge		Set-up t40
L8	Load L8	LA[7:0] valid to LADO rising edge		Hold t41

4.5.7 VMEbus Address Comparator

The VMEbus Address Comparator is made up of three logic elements: an address mask register, address compare register, and a high-performance, 8-bit, equality comparator. The compare and mask registers control the compare logic. The mask register contains an 8-bit value that enables or disables bits of the comparator. The compare register contains an 8-bit pattern. The enabled bits of the compare register are matched against the value on A[7:0]. If a match is detected (all active bits equal), the VCOMP* output pin is driven Low. Neither the compare register nor the mask register are preset at power-up and must be initialized for predictable operation. The act of writing the compare register clears the mask register. This prevents any inadvertent address compares during the configuration process. See Chapter 4.3 for further information on the VMEbus address comparator.

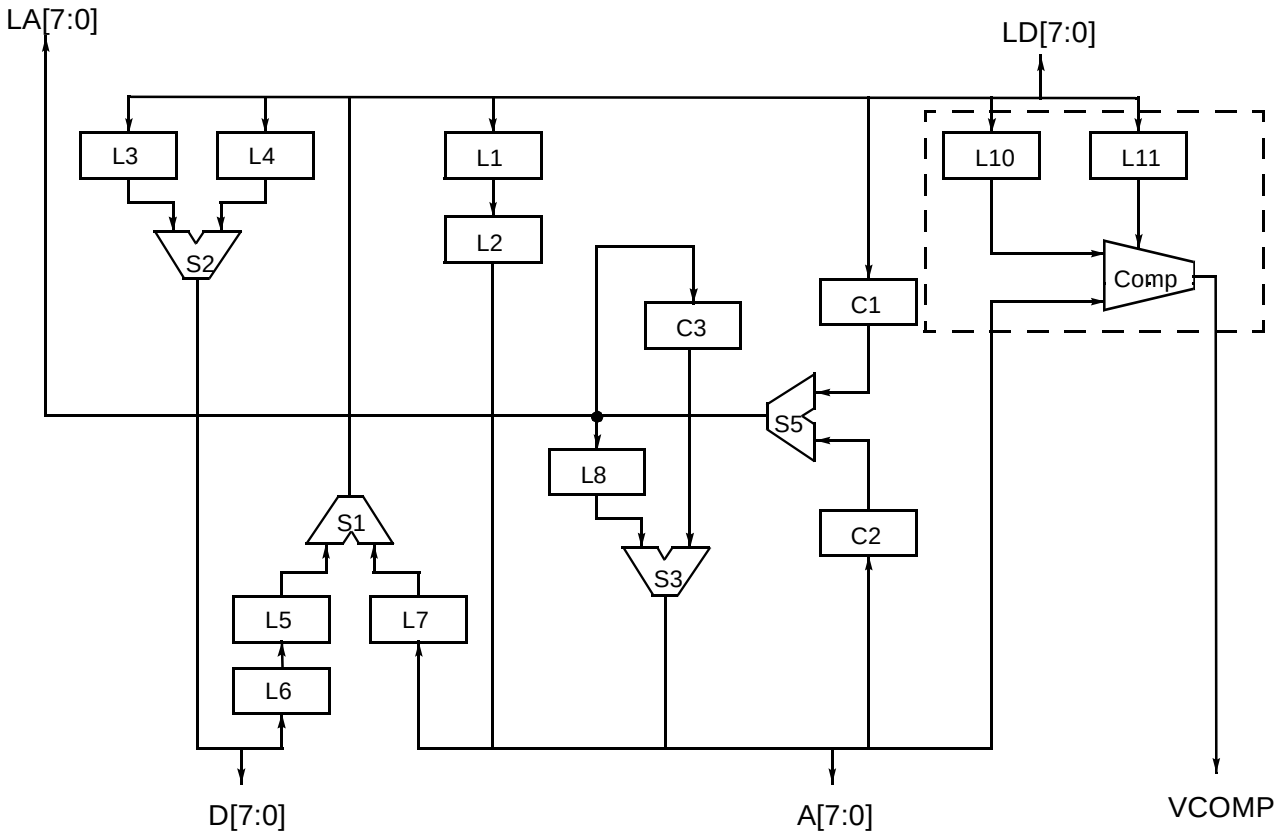


Figure 4-5. CY7C964 Block Diagram: VMEbus Address Comparator

Table 4-11. VMEbus Address Comparator Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
L10	Select compare register	LDS, MWB* valid to STROBE falling edge	LDS=1, MWB*=1	Set-up t43
				Hold t44
	Load compare register	LD[7:0] valid to STROBE rising edge		Set-up t46
				Hold t47
L11	Select mask register	LDS, MWB* valid to STROBE falling edge	LDS=0, MWB*=1	Set-up t43
				Hold t44
	Load mask register	LD[7:0] valid to STROBE rising edge		Set-up t46
				Hold t47
	Compare out	A[7:0] valid to VCOMP* valid		Prop t23
		A[7:0] valid to VCOMP* invalid		Prop t24
	Minimum pulse width	STROBE minimum pulse width		t47

4.5.8 VMEbus D64 Block Transfer Data Pipeline and Multiplexer

Latches L1 and L2 form a two-stage high-performance data pipeline for D64 block transfer operations. These latches load from the local signals LD[7:0], but drive VMEbus address signals A[7:0]. Latches L3 and L4 load from the local data signals LD[7:0] and in combination with multiplexer S2 drive D[7:0]. On the first cycle of a D64 block transfer, data on LD[7:0] is written to latch L1. During the second local data fetch of a D64 block transfer operation (D64=1), data from LD[7:0] is written to latch L3 and the data within latch L1 moves to L2. Two fetches must be performed to form the 64-bit block transfer data word. During non-D64 modes of operation (D64=0), data from LD[7:0] is written to latch L4. This is the normal data path from LD[7:0] to D[7:0] for all non-D64 operation. Because all the latches are implemented on transparent latches, L2 may be loaded from LD[7:0] when L1 is transparent (LEDO=0). None of the latches are initialized at power-up. Therefore, for predictable operation, these latches should be written prior to their use.

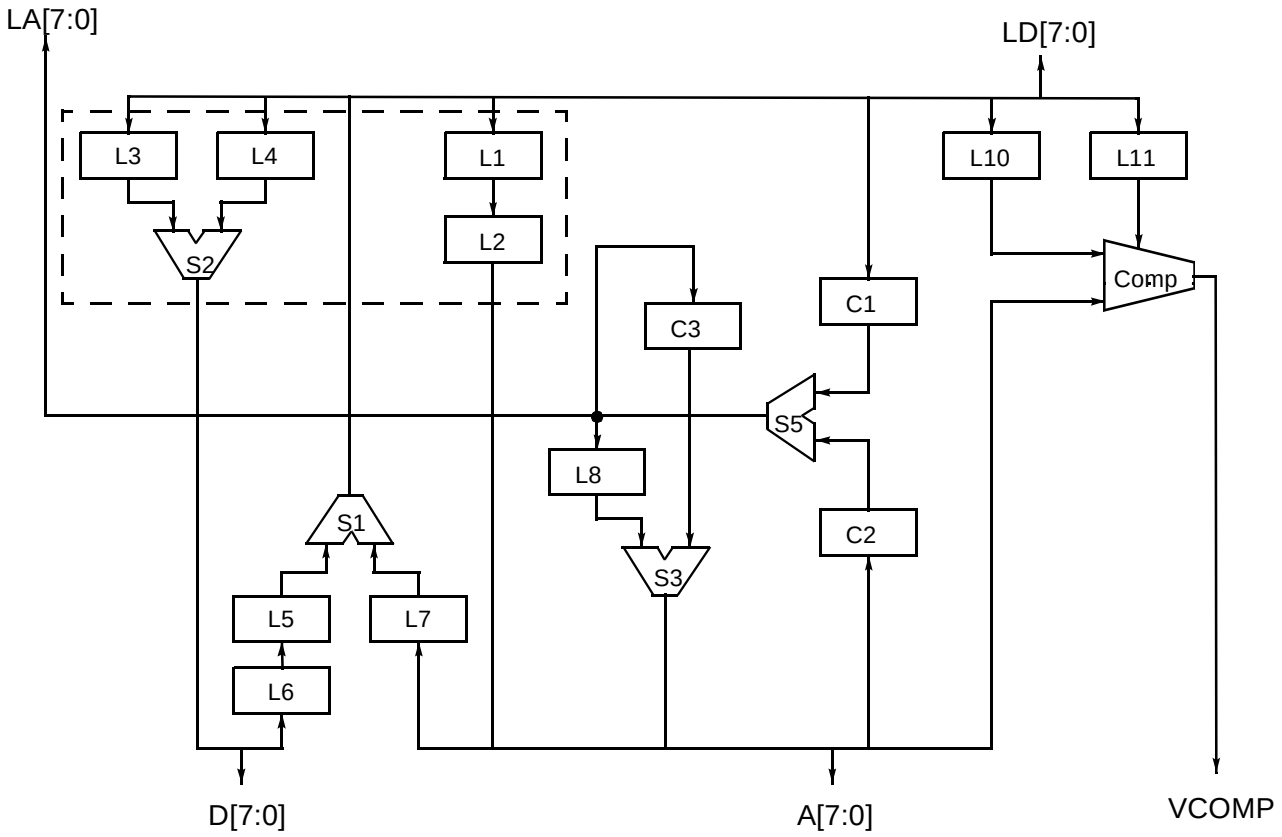


Figure 4-6. CY7C964 Block Diagram: D64 Block Transfer Data Pipeline and Multiplexer

Table 4-12. VMEbus D64 Block Transfer Data Pipeline and Multiplexer Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
L1	Load register	LD[7:0] valid to LEDO rising edge		Set-up t25
				Hold t26
L2	Load register	LD[7:0] valid to DENO* falling edge	LEDO=0	Set-up t28
				Hold t29
	Drive A[7:0]	D64 rising edge to A[7:0] valid	BLT_STATE=1	Prop t82
L3	Load register	LD[7:0] valid to DENO* rising edge		Set-up t25
				Hold t26
L4	Load register	LD[7:0] valid to LEDO rising edge		Set-up t131
				Hold t132
S2	Multiplexer selects L3 drive D[7:0]	D64 rising edge to D[7:0] valid		Prop t78
	Multiplexer selects L4 drive D[7:0]	D64 falling edge to D[7:0] valid		Prop t79
	Minimum pulse width	DENO*		t30
		LEDO		t27

4.5.9 VMEbus D64 Block Transfer Data Demultiplexer

The VMEbus D64 block transfer data demultiplexer moves data from D[7:0]/A[7:0] to LD[7:0]. The demultiplexer consists of three latches—L5, L6, and L7—and an output multiplexer, S1. During D64 block transfer operations (D64=1), data is written to latches L6 and L7 simultaneously on the rising edge of LEDI. Multiplexer S1 then selects either latch L6 or L7, depending on the state of LDS as a source for LD[7:0]. In most applications, LDS should be connected to LA2, showing that L7 contains even 32-bit words (addresses 0, 8, $10_{16...}$) and L6 contains odd 32-bit words (address 4, C, $14_{16...}$). Latch L6 is also used for non-D64 operating modes. None of these latches are initialized at power-up and for predictable operation should be initialized prior to use.

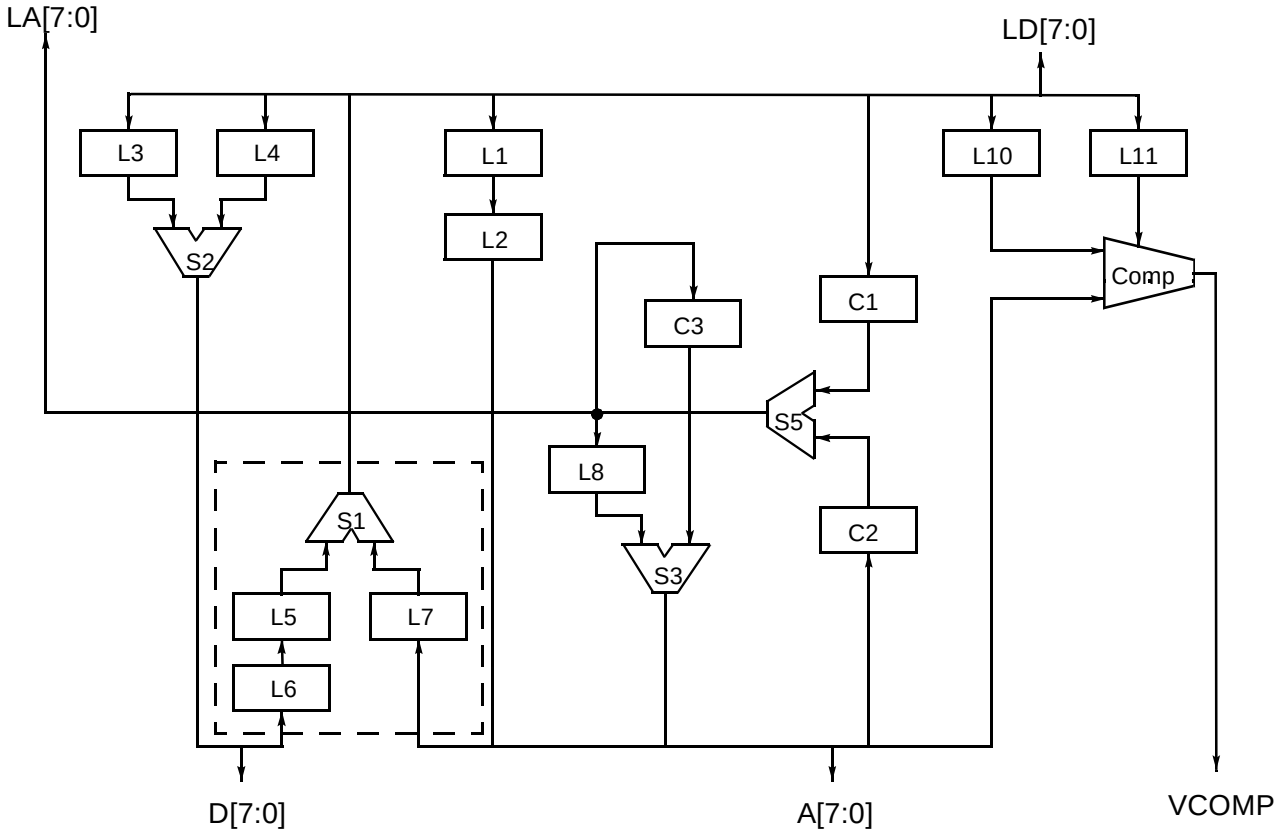


Figure 4-7. CY7C964 Block Diagram: D64 Block Transfer Data Demultiplexer

Table 4-13. VMEbus D64 Block Transfer Data Pipeline and Demultiplexer Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
L5	Load register	D[7:0] valid to DENIN* falling edge	DENIN1*=0, LEDI=0	Set-up t31
				Hold t32
	Load register	D[7:0] valid to DENIN1* falling edge	DENIN*=0, LEDI=0	Set-up t34
				Hold t35
L6	Load register	D[7:0] valid to LEDI rising edge	LEDO=0	Set-up t37
				Hold t38
L7	Load register	A[7:0] valid to LEDI rising edge	LEDO=0	Set-up t37
				Hold t38
S1	Select L5	LDS rising edge to LD[7:0] valid	D64=1	Prop t74
		D64 rising edge to LD[7:0] valid	LDS=1	Prop t76
	Select L7	LDS falling edge to LD[7:0] valid	D64=1	Prop t75
	Select L6	D64 falling edge to LD[7:0] valid		Prop t77
	Minimum pulse width	DENIN*		t33
		DENIN1*		t36
		LEDI		t39

4.6

CY7C964 Alternate BLT Initiation Operation for VIC068A and VIC64

Another method of loading the VMEbus block transfer address counters exists within the CY7C964. This method has been placed within its own section of the document because it is not completely compatible with the VIC block transfer initiation cycle.

The CY7C964 determines the source for loading the VMEbus master block transfer counter C3 by monitoring the arrival sequence of the MWB* and BLT* signals. For typical block transfer initiation cycles, the assertion of MWB* occurs prior to the assertion of the BLT*. The VMEbus master block transfer counter C3 loads from the local address pins, LA[7:0], as described within section 4.5.5.

Reversing the arrival order of these two signals changes the operation of the device. This is done at system design time by swapping the BLT* and MWB* inputs to the CY7C964. For proper operation, these signals must continue to operate in the same manner as they do on the VIC, even though they are no longer connected to the associated input pins on the CY7C964 which have the same name. Swapping these signals on the device changes the way that the VMEbus master block transfer counter C3 is loaded. In this mode it loads from the local data bus via latch L9. All other functions within the device operate in the same manner as described in Chapter 4.5.

Loading C3 is accomplished with a local cycle similar to the cycles needed to load the mask and compare registers. This cycle operates as follows: LDS is driven High, (most likely this signal is connected to LA2), the MWB* input pin of the CY7C964 is driven Low, (this pin is actually connected to the open collector BLT* output of the VIC), and STROBE is asserted. The local data bus should be driven to the appropriate value for the address to load into the C3 counters. STROBE is deasserted and the data is latched into L9 within the CY7C964. The local address decode signal used to assert MWB* on the CY7C964, (BLT* on the VIC), must be a three-state or an open-collector output. This signal must not be driven High or the VIC will be unable to perform block transfers.

A normal master block transfer initiation cycle is then performed, with one minor exception. The lower 8 bits of address LA[7:0] which are controlled by the VIC, must contain the desired lower address. This is needed because the VIC operates in the typical block transfer initiation mode. The upper address LA[31:8] will be ignored by the CY7C964s during the initiation cycle.

This mode of operation allows the VMEbus master block transfer address counters to be loaded independent of the VMEbus address. This has advantages in some designs, but C3 cannot be used to source single cycle transfer addresses. This limitation should be considered while performing the design analysis to use this mode.

Table 4-14. Master Block Transfer Local Address Counter Operation

Logic	Functional Description	Operational Description	Required Condition	Parameter
L9	Select register	LDS, MWB* valid to STROBE falling edge	LDS=1, MWB*=0	Set-up t43
				Hold t44
	Load register	LD[7:0] valid to STROBE rising edge		Set-up t45
				Hold t46
	Minimum pulse width	STROBE		t46

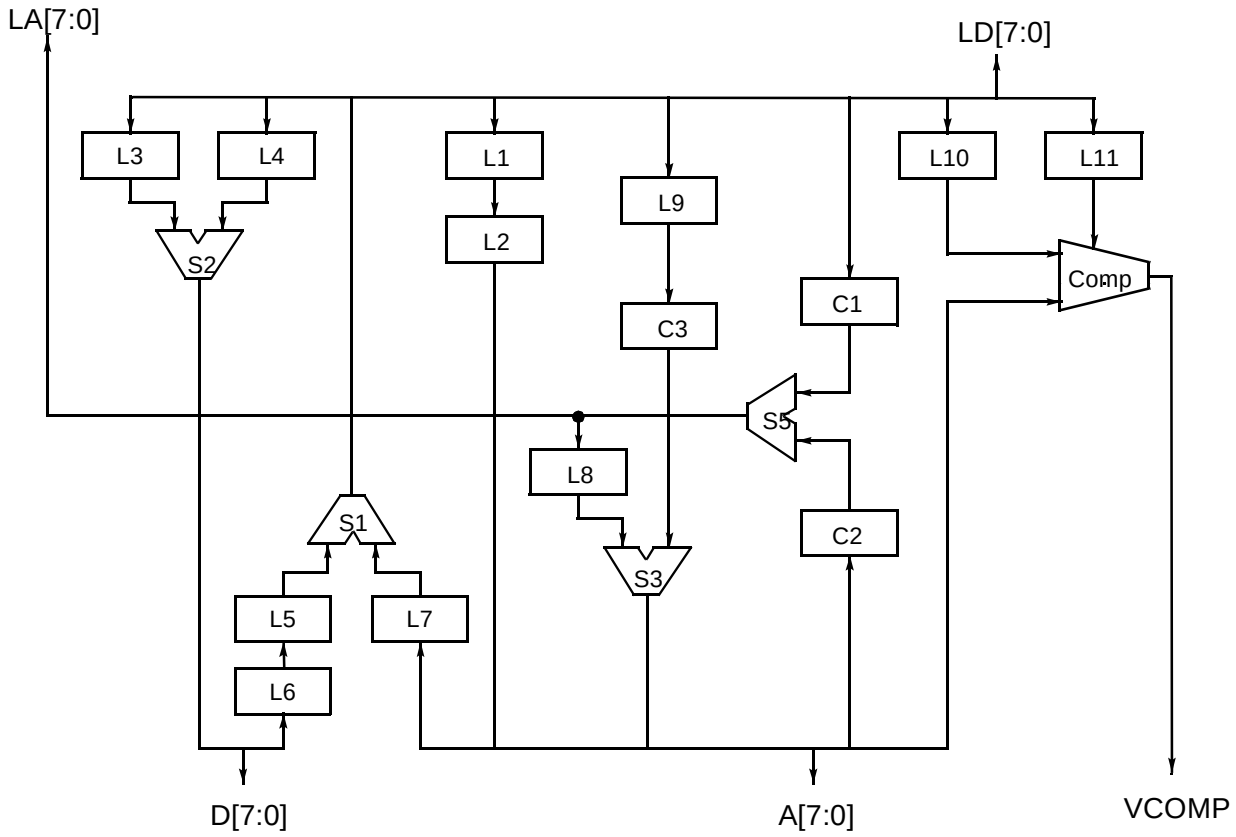


Figure 4-8. CY7C964 Alternate BLT Operation Block Diagram

4.7

DC Performance Specifications

Table 4-15. VMEbus Signals (A[7:0], D[7:0])

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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 4-16. Non-VMEbus Signals

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±5	µA
I_{CC}	V_{CC} Maximum Operating Supply Current	V_{CC} = Max., All Outputs Disabled		25	25	25	mA

Table 4-17. Operating Current

Parameters	Description	Test Conditions	Max.	Units
I_{DD}	Maximum Operating Current	No external DC load	50	mA

4.8

AC Performance Specifications

Parameter	Description	Min. All Grades[1]	Max. (Com'l)	Max. (Ind)	Max. (Mil)	Unit	Comment
t1	LA[7:0] to A[7:0] propagation delay[2]		15	16	18	ns	
t2	LD[7:0] to D[7:0] propagation delay		16	17	19	ns	
t3	A[7:0] to LA[7:0] propagation delay		17	18	20	ns	
t4	D[7:0] to LD[7:0] propagation delay[2]		19	20	22	ns	
t5	LD[7:0] to A[7:0] propagation delay[2]		19	20	22	ns	D64=1
t6	A[7:0] to LD[7:0] propagation delay[2]		19	20	22	ns	D64=1
t7	ABEN* active to A[7:0] output enable delay[2]		11	11	12	ns	
t8	DENO* active to D[7:0] output enable delay[2]		14	15	16	ns	
t9	ABEN* active to D[7:0] output enable delay		14	15	17	ns	D64=1
t10	D64 active to D[7:0] output enable delay[2]		13	14	15	ns	ABEN*=0
t11	LAEN active to LA[7:0] output enable delay[2]		10	11	12	ns	
t12	DENIN* active to LD[7:0] output enable delay		15	16	18	ns	
t13	DENIN1* active to LD[7:0] output enable delay[2]		18	19	21	ns	D64=1
t14	D64 active to LD[7:0] output enable delay[2]		18	19	21	ns	DENIN1*=0
t15	ABEN* inactive to A[7:0] High-Z output disable delay[2]		8	10	12	ns	
t16	DENO* inactive to D[7:0] High-Z output disable delay[2]		12	14	15	ns	
t17	ABEN* inactive to D[7:0] High-Z output disable delay[2]		12	14	15	ns	D64=1
t18	D64 inactive to D[7:0] High-Z output disable delay[2]		14	15	16	ns	ABEN*=0
t19	LAEN inactive to LA[7:0] High-Z output disable delay[2]		13	14	15	ns	
t20	DENIN* inactive to LD[7:0] High-Z output disable delay[2]		14	16	18	ns	
t21	DENIN1* inactive to LD[7:0] High-Z output disable delay[2]		14	16	18	ns	D64=1
t22	D64 inactive to LD[7:0] High-Z output disable delay[2]		19	21	24	ns	DENIN1*=0

Parameter	Description	Min. All Grades[1]	Max. (Com'l)	Max. (Ind)	Max. (Mil)	Unit	Comment
t23	A[7:0] to VCOMP* High-to-Low propagation delay		18	19	21	ns	
t24	A[7:0] to VCOMP* Low-to-High propagation delay		16	17	19	ns	
t25	LD[7:0] set-up time to LEDO rising edge	7				ns	
t26	LD[7:0] hold time after LEDO rising edge	0				ns	
t27	LEDO minimum pulse width	7				ns	
t28	LD[7:0] set-up time to DENO* falling edge	0				ns	LEDO=0
t29	LD[7:0] hold time after DENO* falling edge	7				ns	LEDO=0
t30	DENO* minimum pulse width	10				ns	
t31	D[7:0] set-up time to DENIN* falling edge	5				ns	DENIN1*=0, LEDI=0
t32	D[7:0] hold time after DENIN* falling edge	5				ns	DENIN1*=0, LEDI=0
t33	DENIN* minimum pulse width	10				ns	
t34	D[7:0] set-up time to DENIN1* falling edge	5				ns	
t35	D[7:0] hold time after DENIN1* falling edge	5				ns	
t36	DENIN1* minimum pulse width	10				ns	
t37	D[7:0], A[7:0] set-up time to LEDI rising edge	7				ns	
t38	D[7:0], A[7:0] hold time after LEDI rising edge	0				ns	
t39	LEDI minimum pulse width	10				ns	
t40	LA[7:0] set-up time to LADO rising edge	5				ns	
t41	LA[7:0] hold time after LADO rising edge	5				ns	
t42	LADO minimum pulse width	10				ns	
t43	MWB*, LDS set-up time to STROBE falling edge	0				ns	
t44	MWB*, LDS hold time after STROBE falling edge	5				ns	
t45	LD[7:0] set-up time to STROBE rising edge	5				ns	
t46	LD[7:0] hold time after STROBE rising edge	5				ns	
t47	STROBE minimum pulse width	10				ns	
t48	Local master block transfer address counter C1 LD[7:0] set-up time before MWB* falling edge	0				ns	BLT*, LAEN=0
t49	Local master block transfer address counter C1 LD[7:0] hold time after MWB* falling edge	5				ns	BLT*, LAEN=0
t50	Master block transfer local address counter C1 LD[7:0] set-up time to BLT* falling edge	0				ns	MWB*, LAEN=0
t51	Master block transfer local address counter C1 LD[7:0] hold time after BLT* falling edge	5				ns	MWB*, LAEN=0

Parameter	Description	Min. All Grades[1]	Max. (Com'l)	Max. (Ind)	Max. (Mil)	Unit	Comment
t52	LCIN* set-up time to MWB* falling edge	5				ns	BLT_STATE, LAEN, FC1=1
t53	LCIN hold time after MWB* or BLT* falling edge	0				ns	BLT_STATE, LAEN, FC1=1
t54	MWB*, BLT* falling edge to LA[7:0] propagation delay		21	22	25	ns	BLT_STATE, LAEN, FC1=1
t55	MWB*, BLT* falling edge to LCOUT* propagation delay		24	25	28	ns	BLT_STATE, LAEN, FC1=1
t56	LCIN* to LCOUT* propagation delay[2]		17	18	20	ns	BLT_STATE, LAEN, FC1=1
t57	BLT*, MWB* minimum pulse width	10				ns	
t58	Slave block transfer local address counter C2, A[7:0] set-up time to D64 rising edge	5				ns	LADI=0
t59	Slave block transfer local address counter C2, A[7:0] hold time after D64 rising edge	5				ns	LADI=0
t60	Slave block transfer local address counter C2, A[7:0] set-up time to LADI rising edge	5				ns	D64=0
t61	Slave block transfer local address counter C2, A[7:0] hold time after LADI rising edge	5				ns	D64=0
t62	LCIN* set-up time to LADI rising edge	8				ns	D64=1
t63	LCIN* hold time after LADI rising edge	0				ns	D64=1
t64	LADI rising edge to LA[7:0] valid propagation delay		15	16	18	ns	D64=1, FC1=0
t65	LADI rising edge to LCOUT* valid propagation delay		20	21	24	ns	D64=1, FC1=0
t66	LADI minimum pulse width	10				ns	
t67	Master block transfer VMEbus address counter LA[7:0] set-up time to MWB* rising edge	5				ns	BLT_STATE=1, BLT_INIT=1
t68	Master block transfer VMEbus address counter LA[7:0] hold time after MWB* rising edge	5				ns	BLT_STATE=1, BLT_INIT=1
t69	LADO falling edge to A[7:0] valid propagation delay		23	24	27	ns	BLT_STATE, DUAL_PATH=1, BLT_INIT=0
t70	LADO rising edge to A[7:0] valid propagation delay[2]		24	25	28	ns	BLT_STATE=1, DUAL_PATH=0, BLT_INIT=0
t71	LADO falling edge to VCOUT* valid propagation delay		26	27	30	ns	BLT_STATE, DUAL_PATH=1, BLT_INIT=0
t72	LADO rising edge to VCOUT* valid propagation delay[2]		30	33	36	ns	BLT_STATE, DUAL_PATH=0, BLT_INIT=0
t73	LADO minimum High or Low pulse width	10				ns	
t74	LDS rising edge to LD[7:0] valid propagation delay[2]		20	21	24	ns	D64=1
t75	LDS falling edge to LD[7:0] valid propagation delay		20	21	24	ns	D64=1

Parameter	Description	Min. All Grades[1]	Max. (Com'l)	Max. (Ind)	Max. (Mil)	Unit	Comment
t76	D64 rising edge to LD[7:0] valid propagation delay[2]		21	22	24	ns	
t77	D64 falling edge to LD[7:0] valid propagation delay		21	22	24	ns	
t78	D64 rising edge to D[7:0] valid propagation delay[2]		16	16	18	ns	
t79	D64 falling edge to D[7:0] valid propagation delay[2]		19	19	21	ns	
t80	D64 rising edge to A[7:0] valid propagation delay[2]		15	16	18	ns	BLT_STATE=0
t81	D64 falling edge to A[7:0] valid propagation delay[2]		16	17	19	ns	BLT_STATE=0
t82	D64 rising edge to A[7:0] valid propagation delay[2]		19	19	22	ns	BLT_STATE=1
t83	D64 falling edge to A[7:0] valid propagation delay		20	21	23	ns	BLT_STATE=1
t84	ABEN* falling edge to A[7:0] valid propagation delay[2]		11	11	12	ns	BLT_STATE=1
t85	FC1 rising edge to LA[7:0] valid propagation delay[2]		14	15	17	ns	
t86	FC1 falling edge to LA[7:0] valid propagation delay		17	18	20	ns	
t87	FC1 falling edge to LCOUT* valid propagation delay[2]		17	18	20	ns	
t88	FC1 rising edge to LCOUT* valid propagation delay		17	18	20	ns	
t89	LADO set-up time to MWB*, BLT* rising edge	0				ns	BLT_STATE=1
t136	LADO hold time after MWB*, BLT* rising edge	7				ns	BLT_STATE=1
t137	BLT* set-up time to MWB* falling edge	5				ns	
t90	BLT* hold time after MWB* falling edge	5				ns	
t131	LD[7:0] to DENO* falling edge set-up	5				ns	
t132	LD[7:0] after DENO* falling edge	5				ns	
t133	LCIN* hold time after BLT* falling edge	5				ns	

Notes:

1. All minimum times guaranteed by design, not tested.
2. Guaranteed, not tested.

4.9 Pin Description

4.9.1 Pin Definitions

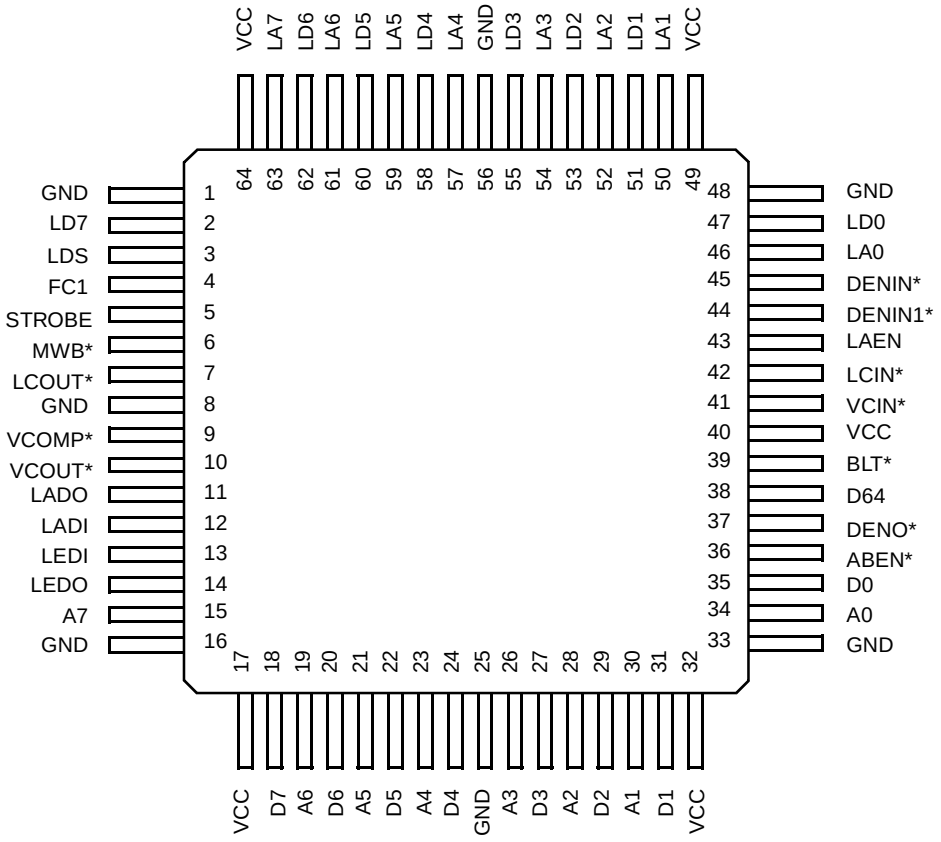
QFP Pin No.	PGA Pin No.	Signal Name	Type	Description
1	E1	GND	Power	Ground
2	L2	LD7	Three-state I/O	Local data transceiver 7
3	K3	LDS	Input	Register select bit
4	L3	FC1	Input	Function code 1 control signal
5	K4	STROBE	Input	Comparator register load control signal
6	L4	MWB*	Input	Module wants VMEbus control signal
7	K5	LCOUT*	Output	Local address counters carry out signal
8	L5	GND	Power	Ground
9	K6	VCOMP*	Output	VMEbus address comparator out signal
10	L6	VCOUT*	Output	VMEbus address counter carry out signal
11	K7	LADO	Input	Latch address out control signal
12	L7	LADI	Input	Latch address in control signal
13	K8	LEDI	Input	Latch enable data in control signal
14	L8	LEDO	Input	Latch enable data out control signal
15	K9	A7	Three-state I/O	High drive address transceiver 7
16	A3	GND	Power	Ground
17	A2	V_{CC}	Power	V_{CC}
18	K11	D7	Three-state I/O	High drive data transceiver 7
19	J10	A6	Three-state I/O	High drive address transceiver 6
20	J11	D6	Three-state I/O	High drive data transceiver 6
21	H10	A5	Three-state I/O	High drive address transceiver 5
22	H11	D5	Three-state I/O	High drive data transceiver 5
23	G10	A4	Three-state I/O	High drive address transceiver 4
24	G11	D4	Three-state I/O	High drive data transceiver 4
25	L9	GND	Power	Ground
26	F11	A3	Three-state I/O	High drive address transceiver 3
27	E10	D3	Three-state I/O	High drive data transceiver 3
28	E11	A2	Three-state I/O	High drive address transceiver 2

QFP Pin No.	PGA Pin No.	Signal Name	Type	Description
29	D10	D2	Three-state I/O	High drive data transceiver 2
30	D11	A1	Three-state I/O	High drive address transceiver 1
31	C10	D1	Three-state I/O	High drive data transceiver 1
32	K10	V_{CC}	Power	V_{CC}
33	B10	GND	Power	Ground
34	A10	A0	Three-state I/O	High drive address transceiver 0
35	B9	D0	Three-state I/O	High drive data transceiver 0
36	A9	ABEN*	Input	High drive address bus enable signal
37	B8	DENO*	Input	High drive data bus enable signal
38	A8	D64	Input	D64 mode enable control signal
39	B7	BLT*	Input	Block transfer control signal
40	B2	V_{CC}	Power	V_{CC}
41	B6	VCIN*	Input	VMEbus address counter count enable signal
42	A6	LCIN*	Input	Local address counter count enable signal
43	B5	LAEN	Input	Local address enable control signal
44	A5	DENIN1*	Input	Data enable in 1 control signal
45	B4	DENIN*	Input	Data enable in control signal
46	A4	LA0	Three-state I/O	Local address transceiver 0
47	B3	LD0	Three-state I/O	Local data transceiver 0
48	F10	GND	Power	Ground
49	A7	V_{CC}	Power	V_{CC}
50	B1	LA1	Three-state I/O	Local address transceiver 1
51	C2	LD1	Three-state I/O	Local data transceiver 1
52	C1	LA2	Three-state I/O	Local address transceiver 2
53	D2	LD2	Three-state I/O	Local data transceiver 2
54	D1	LA3	Three-state I/O	Local address transceiver 3
55	E2	LD3	Three-state I/O	Local data transceiver 3
56	L10	GND	Power	Ground
57	F2	LA4	Three-state I/O	Local address transceiver4
58	F1	LD4	Three-state I/O	Local data transceiver 4
59	G2	LA5	Three-state I/O	Local address transceiver 5
60	G1	LD5	Three-state I/O	Local data transceiver 5
61	H2	LA6	Three-state I/O	Local address transceiver 6
62	H1	LD6	Three-state I/O	Local data transceiver 6

QFP Pin No.	PGA Pin No.	Signal Name	Type	Description
63	J2	LA7	Three-state I/O	Local address transceiver 7
64	J1	V_{CC}	Power	V_{CC}
	K2	GND	Power	Ground
	K1	V_{CC}	Power	V_{CC}
	B11	GND	Power	Ground
	C11	V_{CC}	Power	V_{CC}

4.9.2 Pin Configurations

PQFP/CQFP/TQFP
Top View



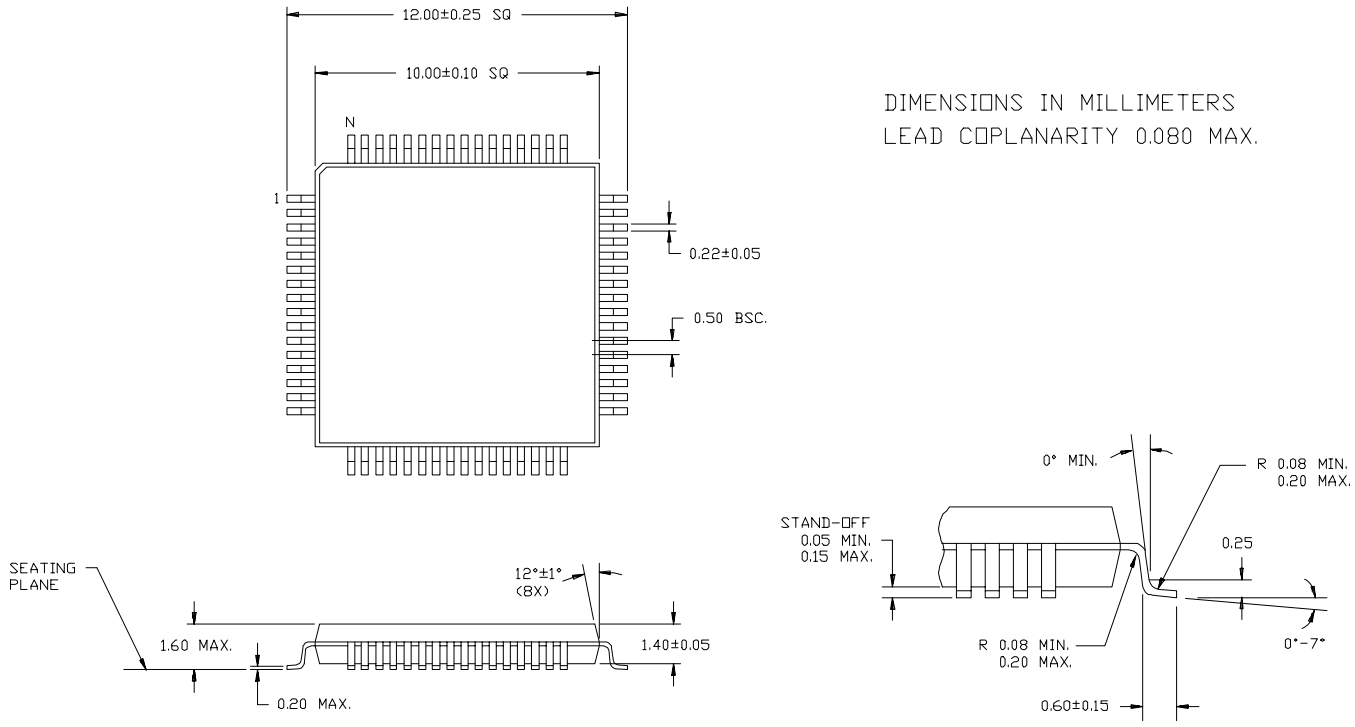
68-Pin Ceramic PGA
Bottom View

11	10	9	8	7	6	5	4	3	2	1	
	A0	ABEN*	D64	V_{CC}	LCIN*	DENIN1*	LA0	GND	V_{CC}		A
GND	GND	D0	DENO*	BLT*	VCIN*	LAEN	DENIN*	LD0	V_{CC}	LA1	B
V_{CC}	D1								LD1	LA2	C
A1	D2								LD2	LA3	D
A2	D3								LD3	GND	E
A3	GND								LA4	LD4	F
D4	A4								LA5	LD5	G
D5	A5								LA6	LD6	H
D6	A6								LA7	V_{CC}	J
D7	V_{CC}	A7	LEDI	LADO	VCOMP*	LCOUT*	STROBE	LDS	GND	V_{CC}	K
	GND	GND	LEDO	LADI	VCOUT*	GND	MWB*	FC1	LD7		L

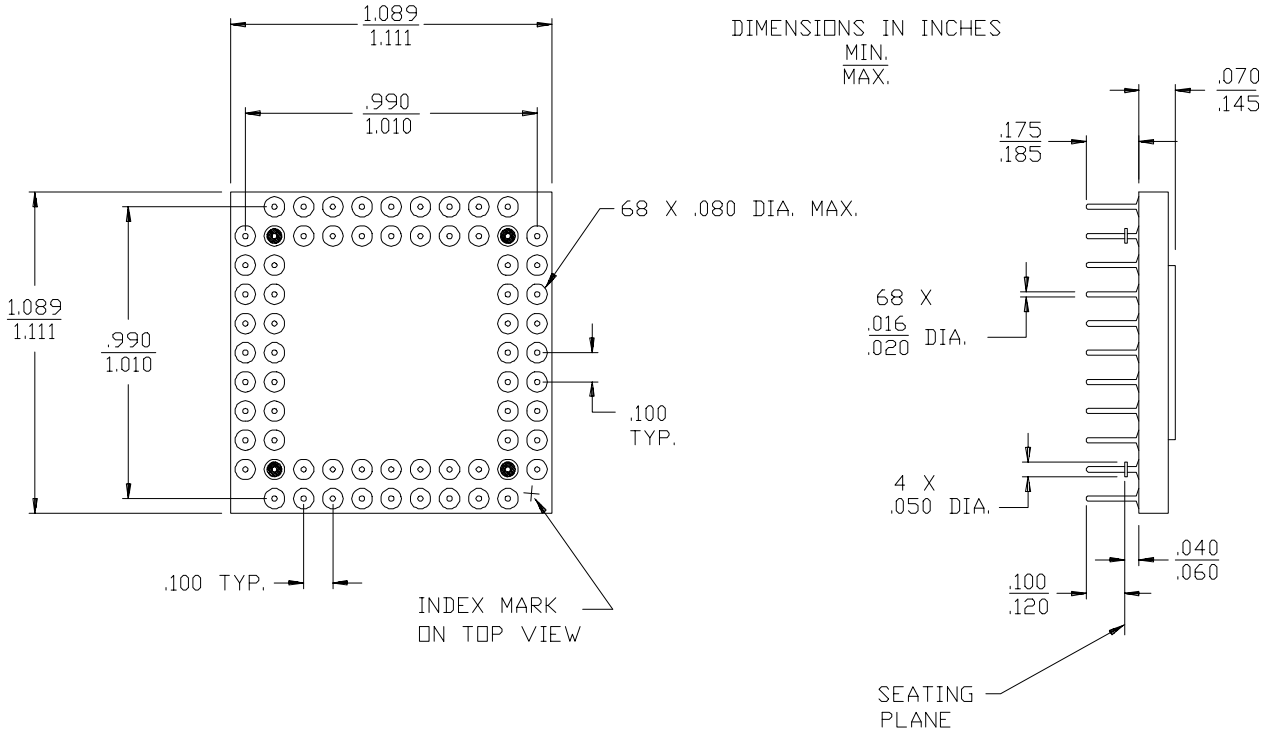
Index Mark
On Top

4.10
Package Diagrams

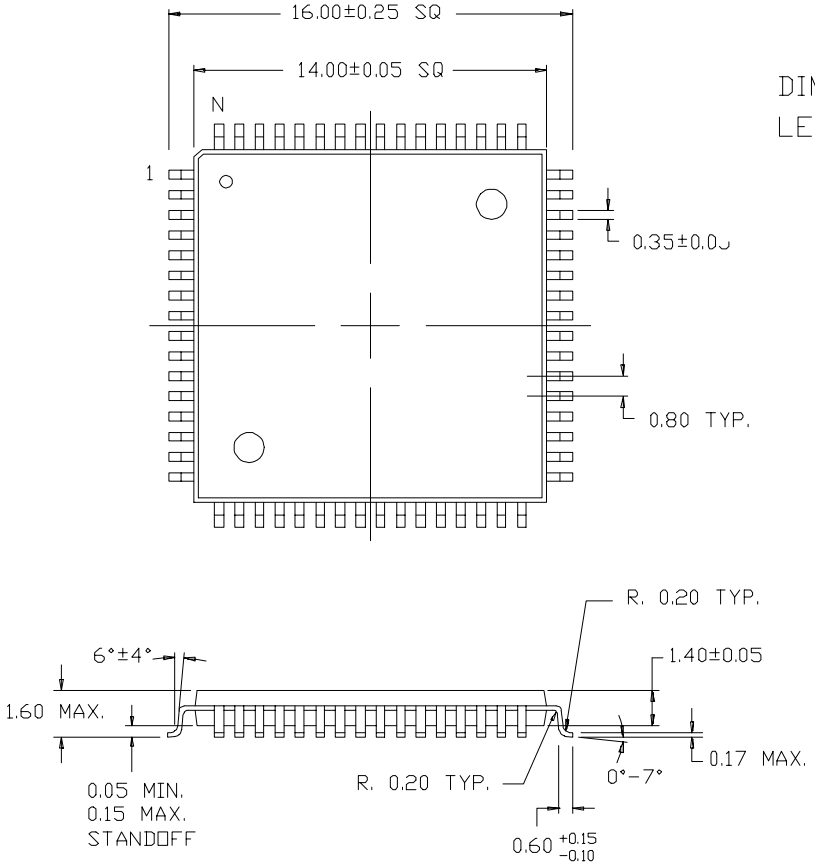
64-Pin Thin Quad Flat Pack A64



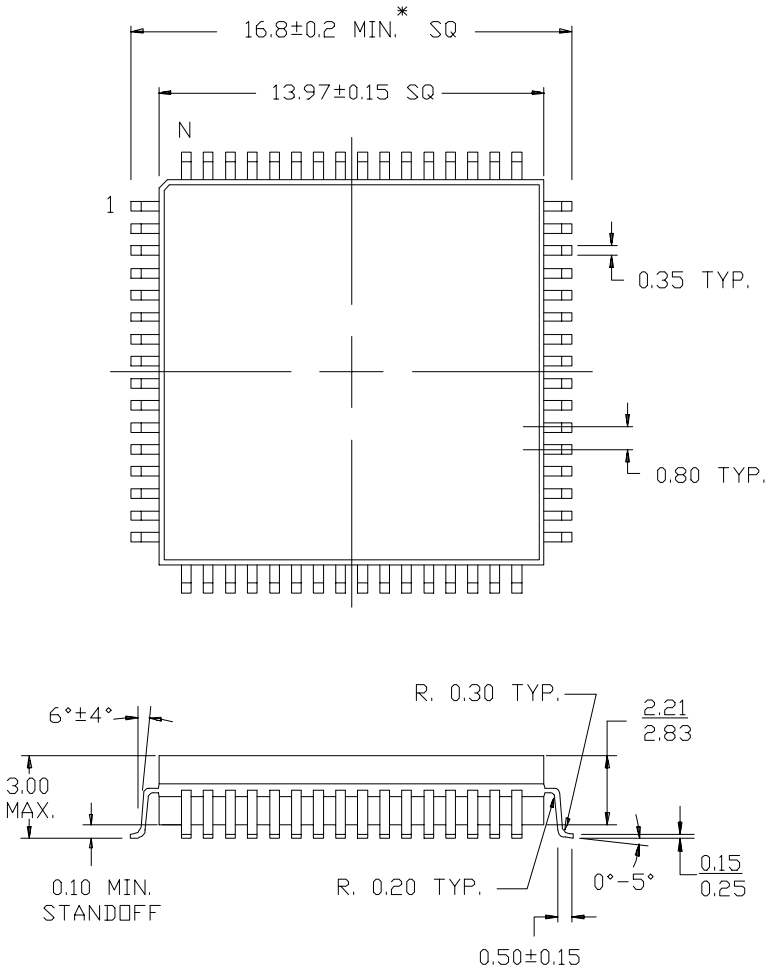
68-Pin Grid Array (Cavity Up) G68



64-Lead Plastic Thin Quad Flatpack N65



64-Lead Ceramic Quad Flatpack (Cavity Up) U65



DIMENSIONS IN MILLIMETERS
LEAD COPLANARITY 0.102 MAX.

DIMENSION	MIN. / MAX.

Section 5

The VAC068A VMEbus
Address Controller

5.1 Introduction to the VAC068A

5.1.1 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, quad-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

5.1.2 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.

5.2
VAC068A Signal Descriptions

5.2.1 VMEbus Signals

A[31:8]

Drive: 64 mA (all)
Type: Three-state I/O

These are the VMEbus address signals.

AS*

Type: Input

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

ID[15:8]

Drive: 16 mA
Type: Three-state I/O

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.

5.2.2 CPU/Local Interface Signals

LD[31:16]

Drive: 16 mA
Type: Three-state I/O

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].

LA[31:8]

Drive: 16 mA
Type: Three-state I/O

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.

PAS*

Type: Input

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.

R/W*

Type: Input

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.

RESET*

Type: Input

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.

WORD*

Drive: 16 mA
Type: Input/Three-state output

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.

ASIZ1, ASIZ0

Drive: 16 mA
Type: Three-state output

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

DSACK1/0*

Drive: 16 mA
Type: Three-state I/O (rescinding)

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.

FC2/0

Type: Inputs

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 1.2.2 of this Handbook)

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

MWB*

Drive: 16 mA
Type: Output

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.

FCIACK*

Drive: 16 mA
Type: Output

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.

DRAMCS*

Drive: 16 mA
Type: Output

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.

EPROMCS*

Drive: 16 mA
Type: Output

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.

FPUCS*

Drive: 16 mA
Type: Output

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.

VSBSEL*

Drive: 16 mA
Type: Output

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.

REFGT*

Drive: 16 mA
Type: Output

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).

LBR*

Type: Input

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.

CS*

Drive: 16 mA
Type: Output

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*).

BLT*

Type: Input

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.

CACHINH*

Drive: 16 mA
Type: Open Collector Output

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.

LDMACK*

Drive: 16 mA
Type: Output

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).

CPUCLK

Type: Input

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

SLSEL0*

Drive: 16 mA
Type: Output

This is the slave select 0 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.

SLSEL1*

Drive: 16 mA
Type: Output

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.

ICFSEL*

Drive: 16 mA
Type: Output

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.

IOSEL1/0*

Drive: 16 mA
Type: Output

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.

5.2.3 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

Drive: 16 mA
Type: Input/Three-state output

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.

PIO1–RXDA

Drive: 16 mA
Type: Input/Three-state output

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.

PIO2–TXDB

Drive: 16 mA
Type: Input/Three-state output

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.

PIO3-RXDB

Drive: 16 mA
Type: Input/Three-state output

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.

PIO4–IORD*

Drive: 16 mA
Type: Input/Three-state output

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).

PIO5–IOWR*

Drive: 16 mA
Type: Input/Three-state output

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).

PIO6–IOSEL3*

Drive: 16 mA
Type: Input/Three-state output

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).

PIO7–Interrupt Request

Drive: 16 mA
Type: Input/Three-state output

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).

PIO8–IOSEL4*

Drive: 16 mA
Type: Input/Three-state output

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.

PIO9–IOSEL5*

Drive: 16 mA
Type: Input/Three-state output

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.

PIO10–Interrupt Request

Drive: 16 mA
Type: Input/Three-state output

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.

PIO11–Interrupt Request

Drive: 16 mA
Type: Input/Three-state output

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.

PIO12–SHRCS*

Drive: 16 mA
Type: Input/Three-state output

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.

PIO13–IOSEL2*

Drive: 16 mA
Type: Input/Three-state output

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).

5.2.4 Data Flow Control Signals

These signals are inputs to VAC068A and are connected to outputs from VIC068A.

SWDEN*

Type: Input

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.

DDIR

Type: Input

This is the data direction signal. It is typically connected to the VIC068A.

LADO

Type: Input

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.

LADI

Type: Input

This is the latch address in signal. It is used to latch the local address from the VMEbus.

LAEN

Type: Input

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.

ABEN*

Type: Input

This is the VMEbus address enable signal. It is used to indicate that the VIC068A is driving the VMEbus address bus.

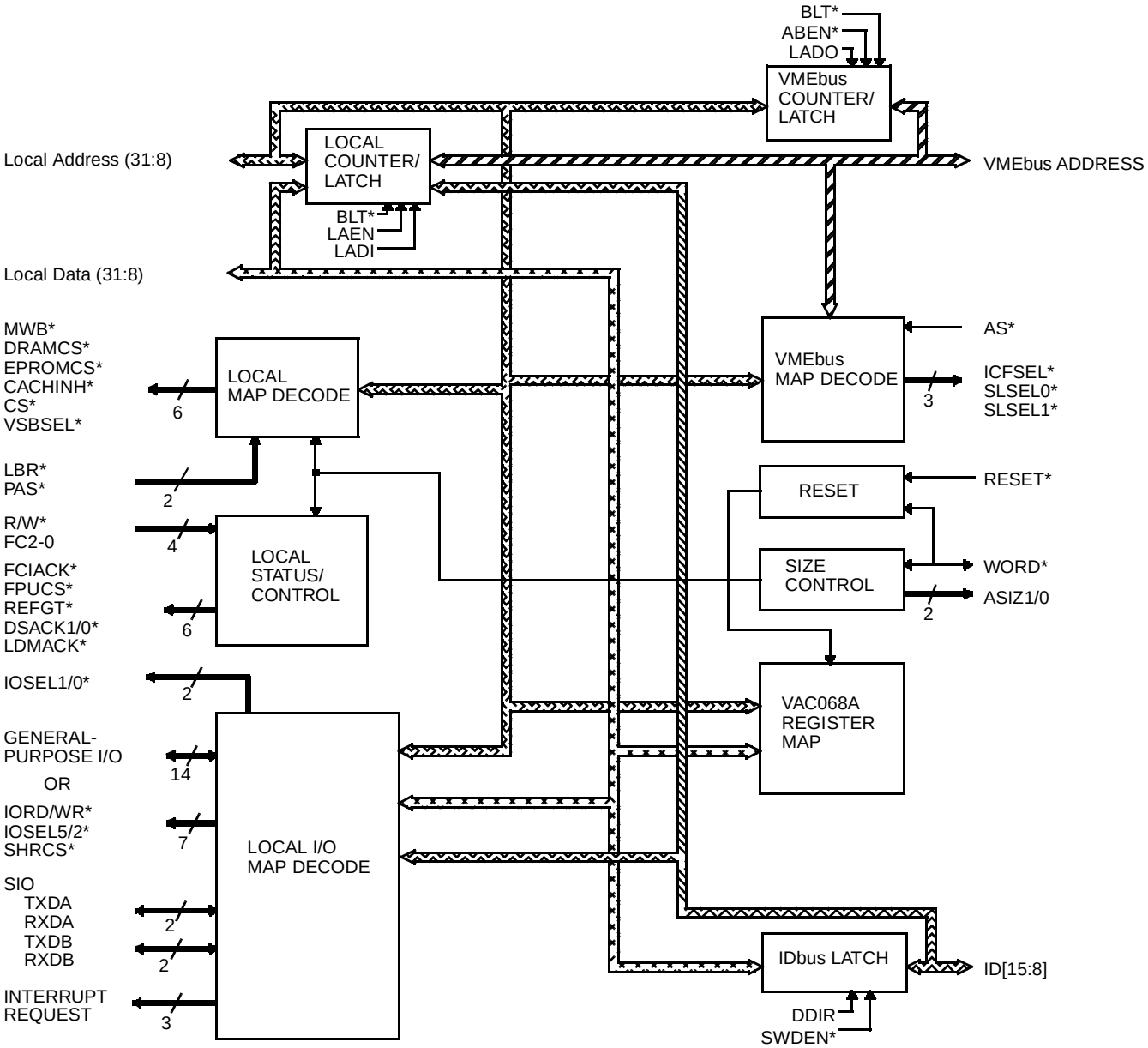


Figure 5-1. VAC068A Block Diagram

5.3
VAC068A Overview

5.3.1 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. Section 1 of this book 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 two '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 Device 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 correspond-

ing 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 1s
- bit 30 of the PIO Direction register is set and an access is made to $FFFF FFxx and function code inputs are NOT all set to 1s
- 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 VME-bus.

5.3.2 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.

5.3.2.1 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

5.3.2.2 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 Section 1 of this book for details on VMEbus master transfers.

5.3.2.3 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 datapath 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.

5.3.2.4 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.

5.3.3 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., 0s 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. Section 1 of this user's guide should be referenced when accessing specific switches and registers.

5.3.4 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 *Figure 5-2* for graphical representation).

5.3.4.1 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.

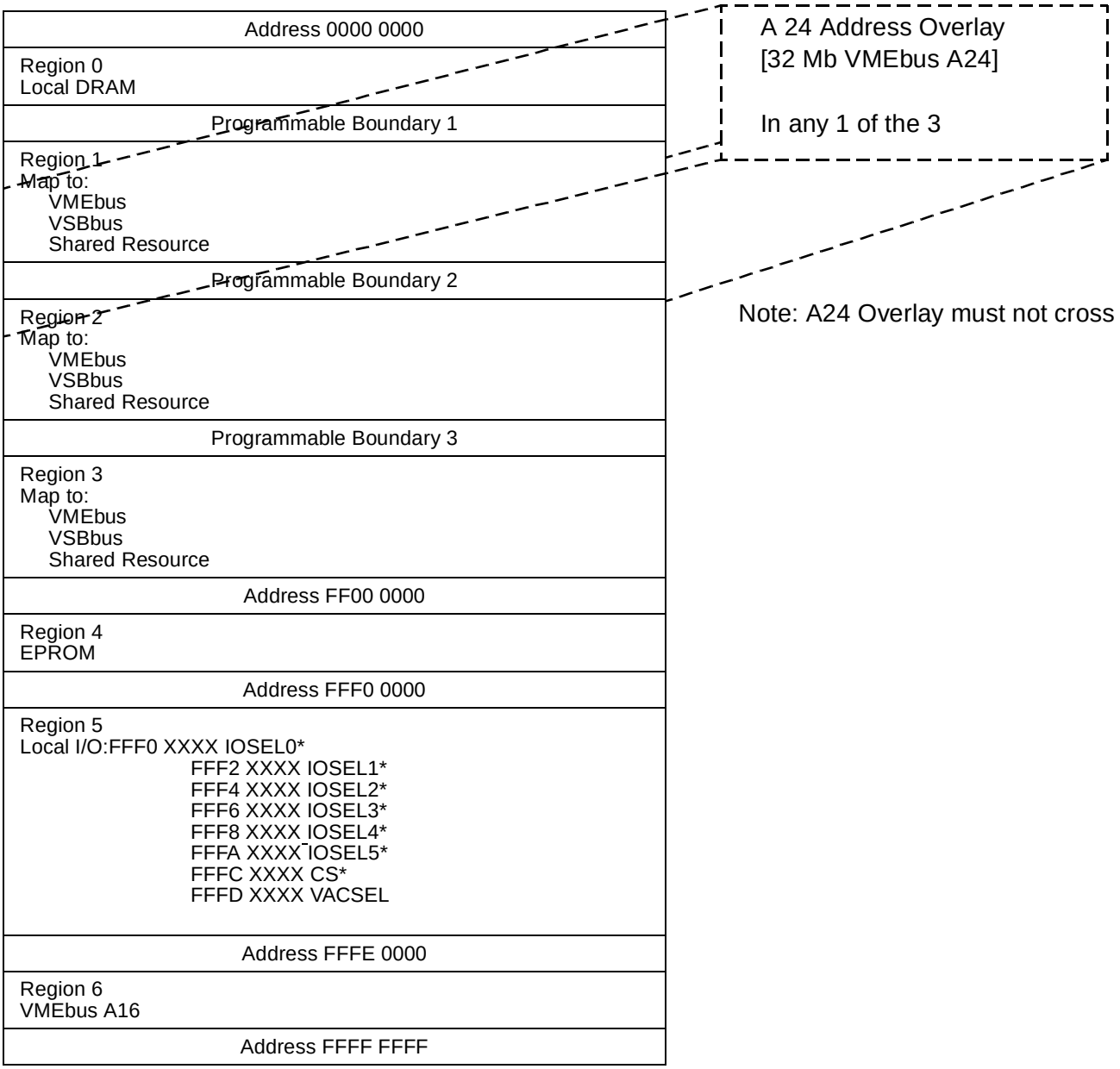


Figure 5-2. VAC068A Memory Map

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.

5.3.4.2 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.

5.3.4.3 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).

5.3.4.3.1 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.

5.3.4.4 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 1s. When all registers are configured, the VAC068A ID register must be written to enable decode and control functions.

5.3.5 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

5.3.5.1 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 Section 1 of this book) 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).

5.3.6 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 5.3.8.1). 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*.

5.3.6.1 Serial I/O

The VAC068A contains a dual, full-duplex UART. Each channel is double-buffered on transmit and quad-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

5.3.6.2 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 functions 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.

5.3.7 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.

5.3.7.1 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.

5.3.7.2 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.

5.3.7.2.1 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.

5.3.7.2.2 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.

5.3.7.2.3 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.

5.3.8 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.

5.3.8.1 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.

5.3.8.2 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.

5.3.8.3 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-resolution DSACKi*-to-DTACK* delay timer in the VIC068A to delay until data is valid. (See VIC068A registers $C7 and $A7.)

5.3.8.4 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*.

5.3.8.5 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.

5.3.8.6 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.

5.3.8.7 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.

5.3.8.8 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.

5.4
VAC068A Operation

5.4.1 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.

5.4.1.1 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.

5.4.1.1.1 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.

5.4.1.2 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.

5.4.1.3 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 DSACK*s 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.

5.4.2 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.

5.4.3 Configuring the Local Memory Map

Sections 5.4.3 and 5.4.4 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.

5.4.3.1 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.

5.4.3.2 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.

5.4.3.3 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.

5.4.3.4 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.

5.4.3.5 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*.

5.4.4 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.

5.4.4.1 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*.

5.4.4.2 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.

5.4.4.3 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.

5.4.4.4 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.

5.4.4.5 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.

5.4.4.6 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*.

5.4.5 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.

5.4.6 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.

5.4.6.1 Slave Transfer Sequence

The following sequence of events occurs 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*, DENIN*, DENIN1*, 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.

5.4.7 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 LD[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*, DENIN*, DENIN1*, 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*.
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.

5.4.8 VIC068A/VAC068A Interconnect Diagram

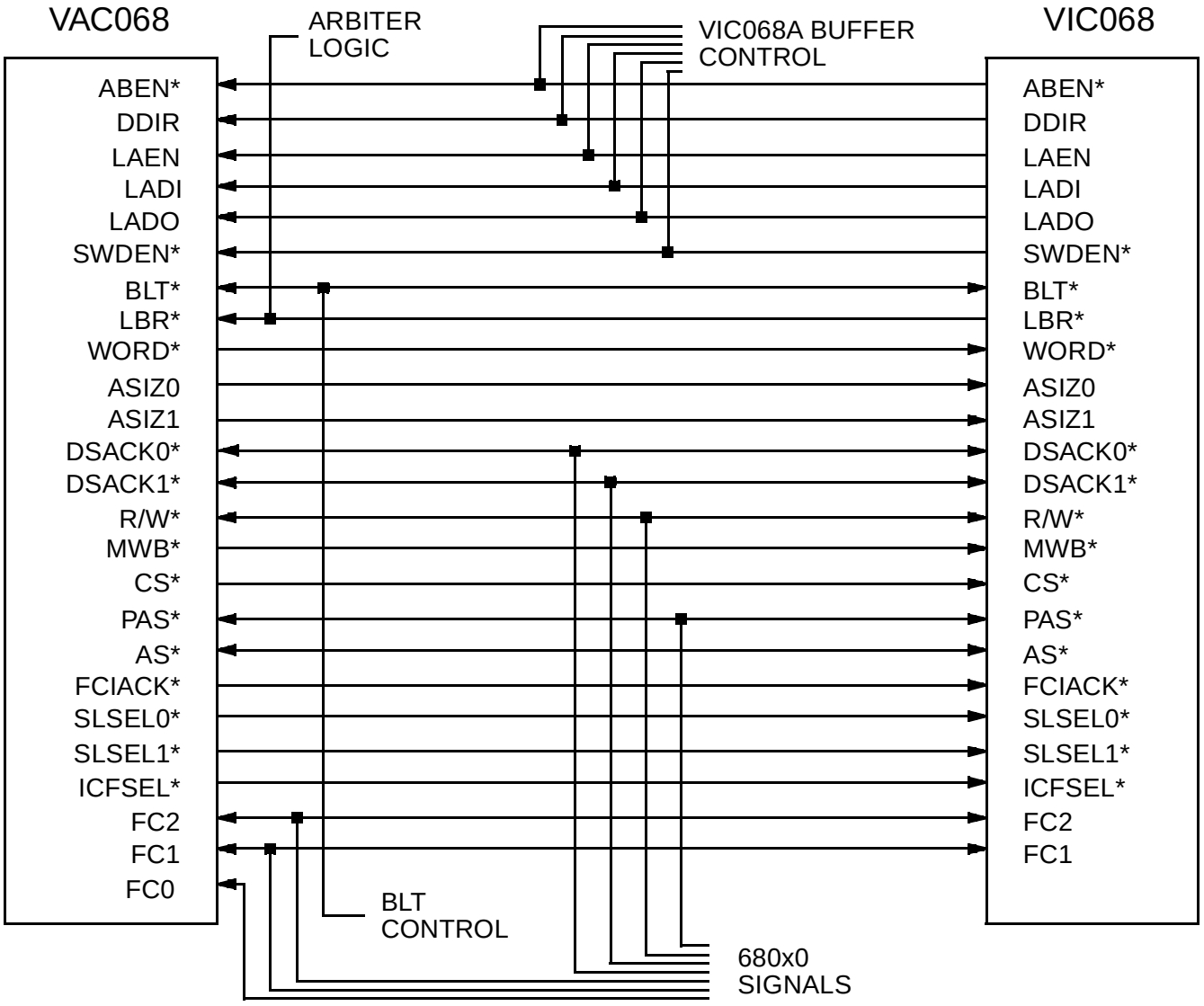


Figure 5-3. VIC068A/VAC068A Interconnect Diagram

5.5

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 1s. Register values are listed in *Table 5-1*.

Table 5-1. 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* Address Mask Register	16 bits
FFFD 03xx	SLSEL0* Base Address 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
FFFD 16xx	Interrupt Control Register	16 bits
FFFD 17xx	Device Location Register	6 bits

Table 5-1. Register Values (continued)

Local Address	Register List	Size
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 (Boundary 1)

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.

A24 Base Address Register

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).

DSACKi* Control Registers (continued)

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*. When cleared, DSACK1* is inactive.
Bit 27	When set, this bit enables DSACK0*. 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).

Decode Control Register (continued)

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*. 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.
Bit 31	When set, assert DSACKi* on slave accesses to DRAM during VIC068A local bus cycles (except DRAM Refresh). When cleared, three-state

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.
Bit 26 (Bit 26 below)	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

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* address 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 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	*CPU Clock Divisor Register*
16 MHz	$96
16.67 MHz	$93
20 MHz	$7C
25 MHz	$5B
30 MHz	$3B
33 MHz	$27

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 enabled. When cleared, parity generate and check are disabled.

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.

UART Channel A and B Mode Register (continued)

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. Enable respective transmitters BEFORE loading these registers.

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.

UART Channel A and B Interrupt Mask Register (continued)

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 dependent 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

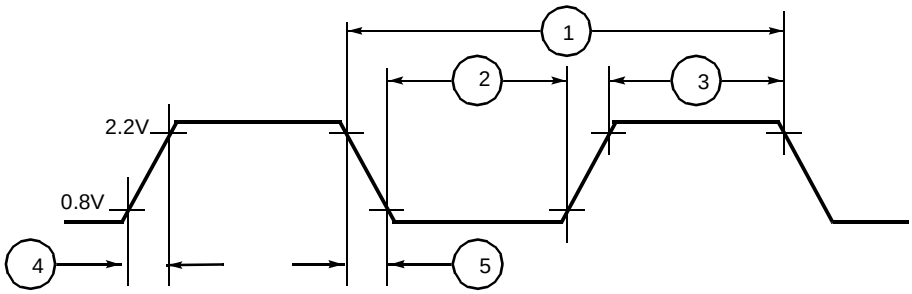
Note: After a global reset and the completion of loading all other registers, this register must be written in order for the VAC068A to enable its decode and compare functions.

5.6

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]	1	5T		5T		5T	
2	WORD*[0] to RESET* High, WORD*[H]	1	10T		10T		10T	
REGISTER WRITE								
1	LA[31:8], FCi, R/W* Valid to PAS*[L] (Set-Up Time)	1	10		10		10	
2	LD[31:16] Valid to DSACKi*[L] (Set-Up Time)	1	5		5		5	
3	PAS*[0] to DSACKi*[L]	1	6 + 1T	35 + 2T	5 + 1T	36 + 2T	5 + 1T	40 + 2T
4	PAS*[1] to DSACKi*[H]	1	5	29	4	30	4	33
5	PAS*[1] to LA[31:8], FCi, R /W* (Hold Time)	1	5		5		5	
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)	1	10		10		10	

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
2	PAS*[0] to DSACKi*[L]		6	35 + 1T	5	36 + 1T	5	40 +1T
3	PAS*[0] to LD[31:16] Valid		9	51 + 1T	8	53 + 1T	7	58 +1T
4	PAS*[1] to DSACKi*[H]		5	17	4	17	5	18
5	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)	1	5		5		5	
6	PAS*[0] to LD[31:16] Invalid	2	6 + 3T	44 + 35T	5 + 3T	44 + 3.5T	5 + 3T	45 + 3.5T
LOCAL ACCESS VIA LOCAL BUS								
1	LA[31:8], FCi, R/W* to PAS*[0] (Set-Up Time)	1	10		10		10	
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	33	6	34	6	37
3	PAS*[0] to Chip Select[L]	3, 4	6	33	5	34	5	38
4	PAS*[0] to DSACKi*[L]	5	PI1 + 6	PI1+24+1T	PI1 + 5	PI1+25+1T	PI1 + 5	PI1+28+1T
5	PAS*[0] to IORD*/IOWR*[L]	6	PI3 + 6	PI3+47+1T	PI3 + 6	PI3+48+1T	PI3 + 5	PI3+53+1T
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)	1	5		5		5	
7	PAS*[1] to ASIZ1/0 WORD* Invalid		7	37	7	38	6	42
8	PAS*[1] to Chip Select*[H]	3	4	34	3	35	3	39
9	PAS*[1] to DSACKi*[H]		4	16	4	17	4	18
10	PAS*[1] to IORD*[H] / IOWR*[H]	6	PI3 + 4	PI3+21+1T	PI3 + 3	PI3+22+1T	PI3 + 3	PI3+25+1T
LOCAL ACCESS VIA VMEbus								
1	PAS*[0] to ASIZ1/0, WORD* Valid		7	33	6	34	6	37
2	PAS*[0] to Chip Select[L]	7	6	33	5	34	5	38
3	PAS*[0] to DSACKi*[L]	8	PI2 + 6	PI2+29+1T	PI2 + 5	PI2+30+1T	PI2 + 5	PI2+33+1T
4	PAS*[0] to IORD*[L], IOWR*[L]	6	PI3 + 6	PI3 + 47+1T	PI3 + 6	PI3+48+1T	PI3 + 5	PI3+53+1T
5	PAS*[1] to ASIZ1/0, WORD* Invalid		7	37	7	38	6	42
6	PAS*[1] to Chip Select[H]	6	4	34	3	35	3	39
7	PAS*[1] to DSACKi*[H]	9	5	17	4	17	4	18
8	PAS*[1] to IORD*[H], IOWR*[H]	6	PI3 + 4	PI3+21+1T	PI3 + 3	PI3+22+1T	PI3 + 3	PI3+25+1T
VMEbus SLAVE/SLAVE BLOCK ACCESS								
1	AS*[0] to SLSELi*[L] or ICFSEL*[L]		3	24	3	25	2	27
2	LAEN[1] to LA[31:8] Valid		4	24	3	25	3	27
3	AS*[1] to SLSELi*[H] or ICFSEL*[L]		6	20	5	21	5	23
4	LAEN[0] to LA[31:0] Invalid		12	30	10	31	10	33
VMEbus MASTER ACCESS								
1	LA[31:8], FCi, R/W* to PAS*[0] (Set-Up Time)	1	10		10		10	

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	33	6	34	6	37
3	PAS*[0] to MWB*[L]		11	30	10	32	10	35
4	ABEN*[0] to A[31:8] Valid		3	17	2	18	2	20
5	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)	1	5		5		5	
6	PAS*[1] to ASIZ1/0, WORD* Invalid		7	37	7	38	6	42
7	PAS*[1] to MWB*[L]		3	31	2	32	2	36
8	ABEN*[1] to A[31:8] Invalid		2	10	1	10	1	11
MASTER BLOCK TRANSFER INITIATION CYCLE								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)	1	10		10		10	
2	PAS*[0] to ASIZ1/0, WORD* Valid		7	33	6	34	6	37
3	PAS*[0] to DSACKi*[L]		6 + 1T	35 + 2T	5 + 1T	36 + 2T	5 + 1T	40 + 2T
4	PAS*[0] to MWB*[L]		11	30	10	32	10	35
5	PAS*[1] to ASIZ1/0, WORD* (Hold Time)		7	37	7	38	6	42
6	PAS*[1] to DSACKi*[H]		5	17	4	17	4	18
7	PAS*[1] to MWB*[H]		3	31	2	32	2	36
8	ABEN*[0] to A[31:8] Valid		3	17	2	18	2	20
9	LAEN[1], FCi Valid to LA[31:8] Valid		4	24	3	25	3	27
10	LAEN[1], FCi Valid to LDMACK*[1]		1	27	1	28	1	31
BOUNDARY CROSSING								
1	BLT*[1] to LA[31:8] Incremented		1	27	1	28	1	33
2	LADO[0] to A[31:8] Incremented		7	28	5	29	4	33
PIO OUTPUT								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)	1	10		10		10	
2	LD[31:16] to PAS*[0] (Set-Up Time)	1	5		5		5	
3	PAS*[0] to PIO[13:0] Valid		4 + 1T	54 + 2T	3 + 1T	56 + 2T	3 + 1T	63 + 2T
4	PAS*[0] to DSACKi*[L]		6 + 1T	35 + 2T	5 + 1T	36 + 2T	5 + 1T	40 + 2T
5	PAS*[1] to DSACKi*[H]		4	16	4	17	4	18
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)	1	5		5		5	
7	PAS*[1] to LD[31:16] Invalid		6	44	5	44	5	45
PIO INPUT								
1	LA[31:8], FCi, R/W* Valid to PAS*[0] (Set-Up Time)	1	10		10		10	

	Operation	Notes	Commercial		Industrial		Military	
			Min.	Max.	Min.	Max.	Min.	Max.
2	PIO[13:0] to PAS*[0] (Set-Up Time)	1	5		5		5	
3	PAS*[0] to DSACKi*[L]		6	35 + 1T	5	36 + 1T	5	40 + 1T
4	PAS*[0] to LD[31:16] Valid		9	51 +1T	8	53 + 1T	7	58 + 1T
5	PAS*[1] to DSACKi*[H]		4	16	4	17	4	18
6	PAS*[1] to LA[31:8], FCi, R/W* (Hold Time)	1	5		5		5	
7	PAS*[1] to LD[31:16] Invalid		6	44	5	44	5	45

Notes:

1. Guaranteed, but not tested.
2. Maximum time to LD[31:16] invalid is PAS*[0] + 3.5T or PAS*[1], whichever occurs first.
3. Chip select can be any of DRAMCS*, EPROMCS*, SHRCS*, VSBSEL*, FPUCS*, CS*, or IOSELi*.
4. The Decode Control register provides facilities to condition DRAMCS* or boundary decodes with the assertion of PAS*.
5. PI1 is the programmable interval for EPROMCS*, SHRCS*, and IOSELi* in the DSACKi* Control register.
6. PI3 is the programmable interval for IORD* and IOWR* in the Decode Control register.
7. Chip select can be any of DRAMCS*, EPROMCS*, SHRCS*, or VSBSEL*.
8. PI2 is the programmable interval for EPROMCS*, SHRCS*, DRAMCS*, or VSBSEL* in the Decode Control register.
9. SLSELi* redirection is enabled in the Decode Control register.

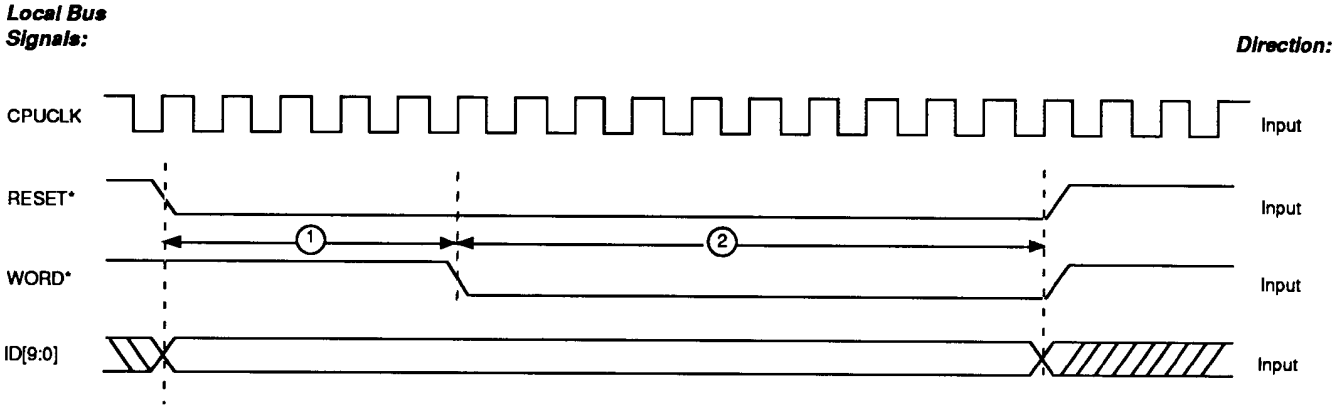


Figure 5-4. VAC068A Global Reset

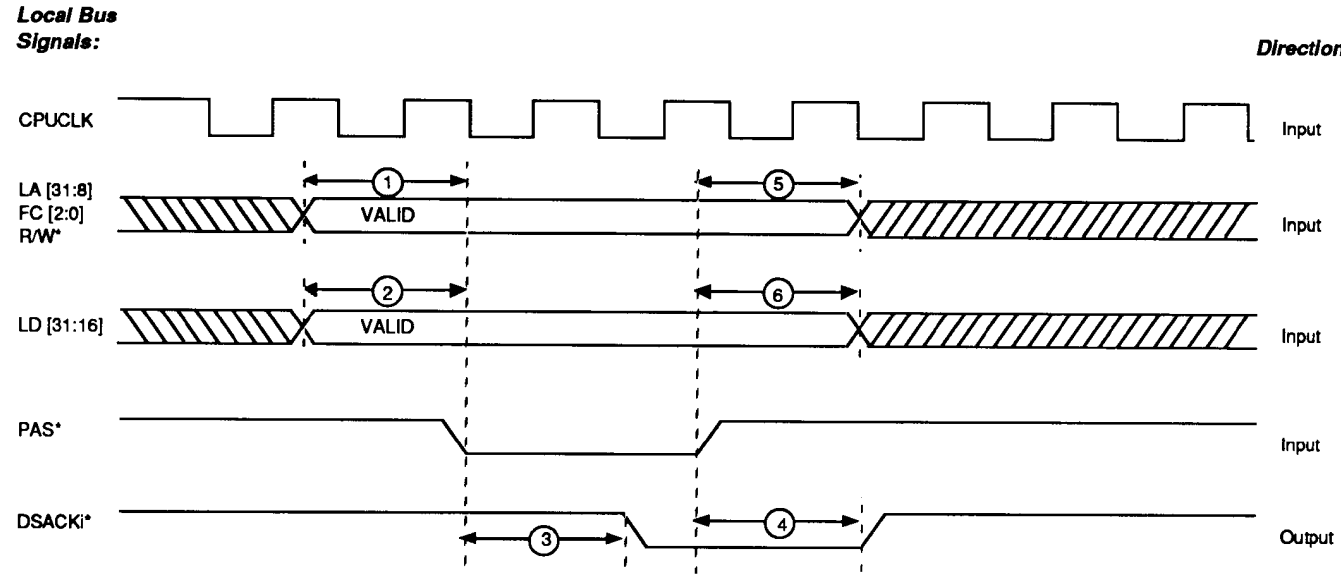


Figure 5-5. VAC068A Register Write

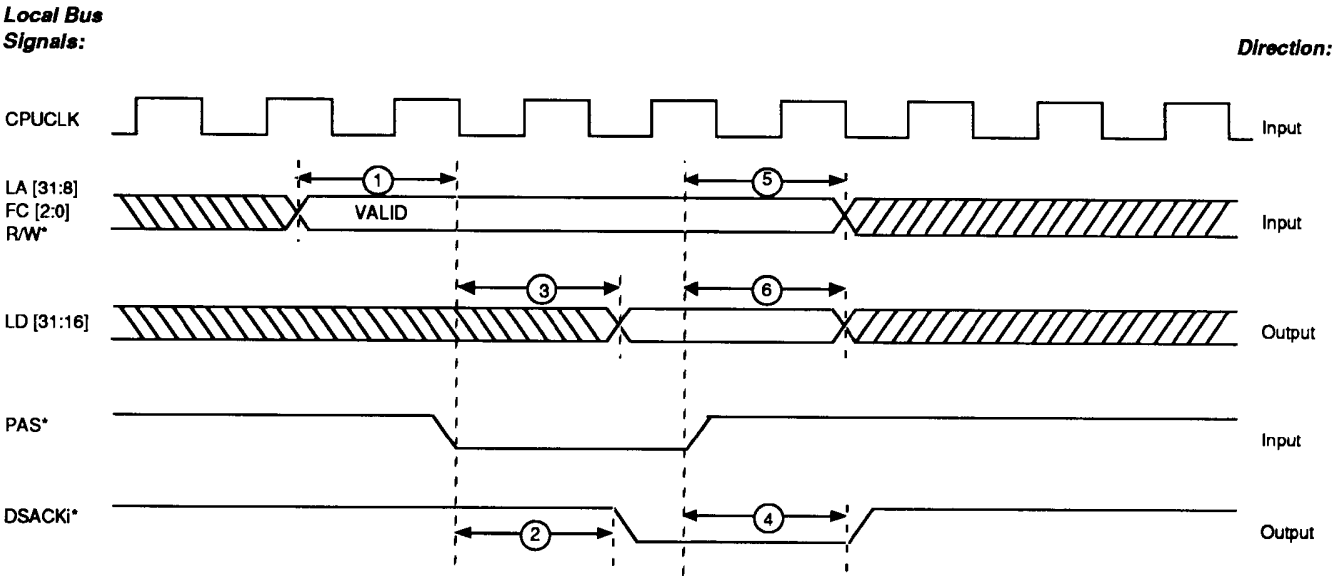


Figure 5-6. VAC068A Register Read

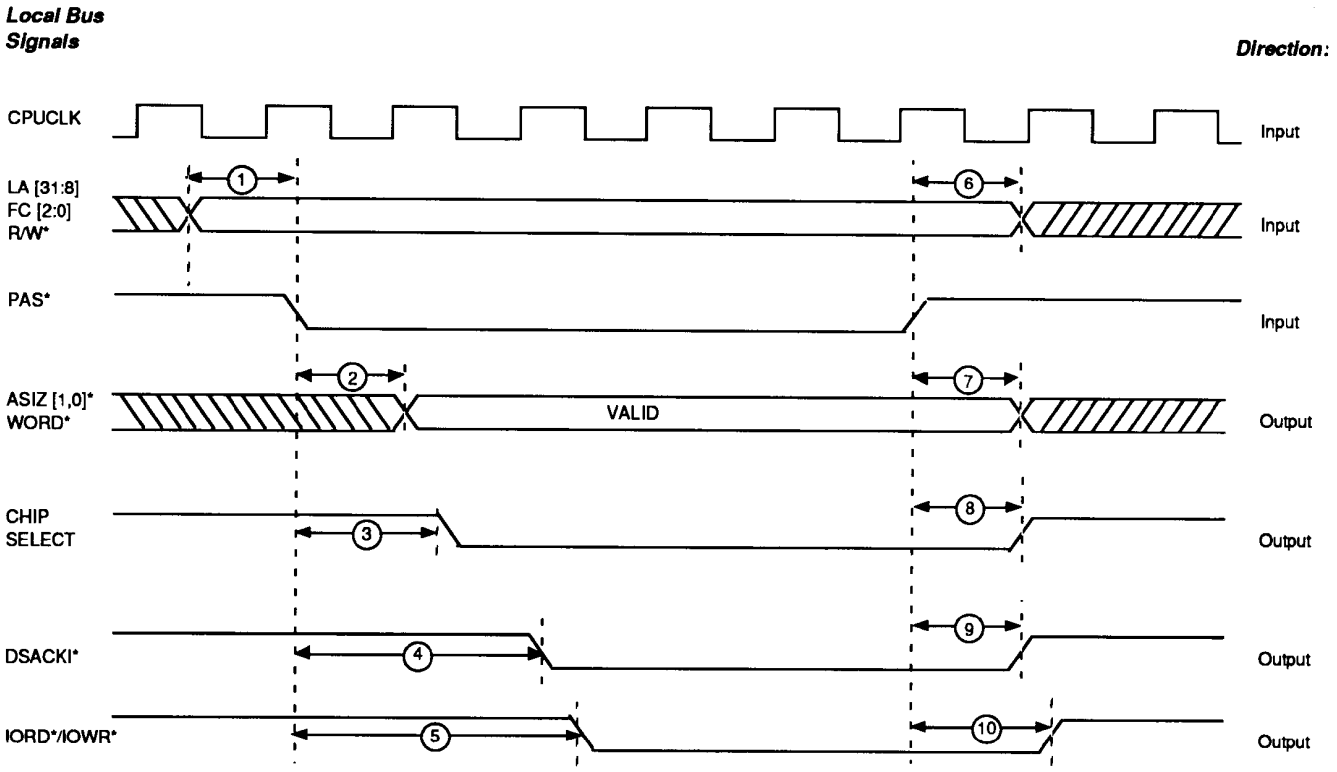


Figure 5-7. Local Resource Access via Local Bus

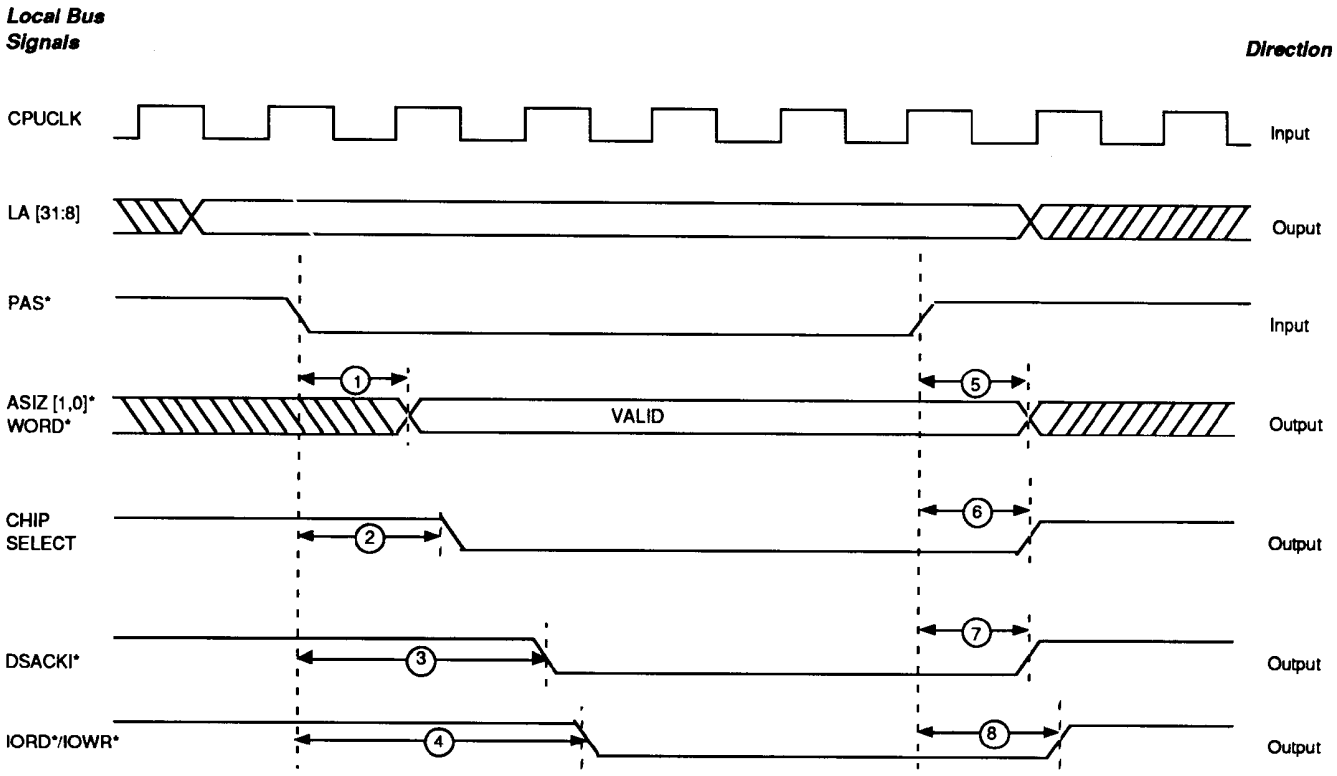


Figure 5-8. Local Resource Accesses via VMEbus

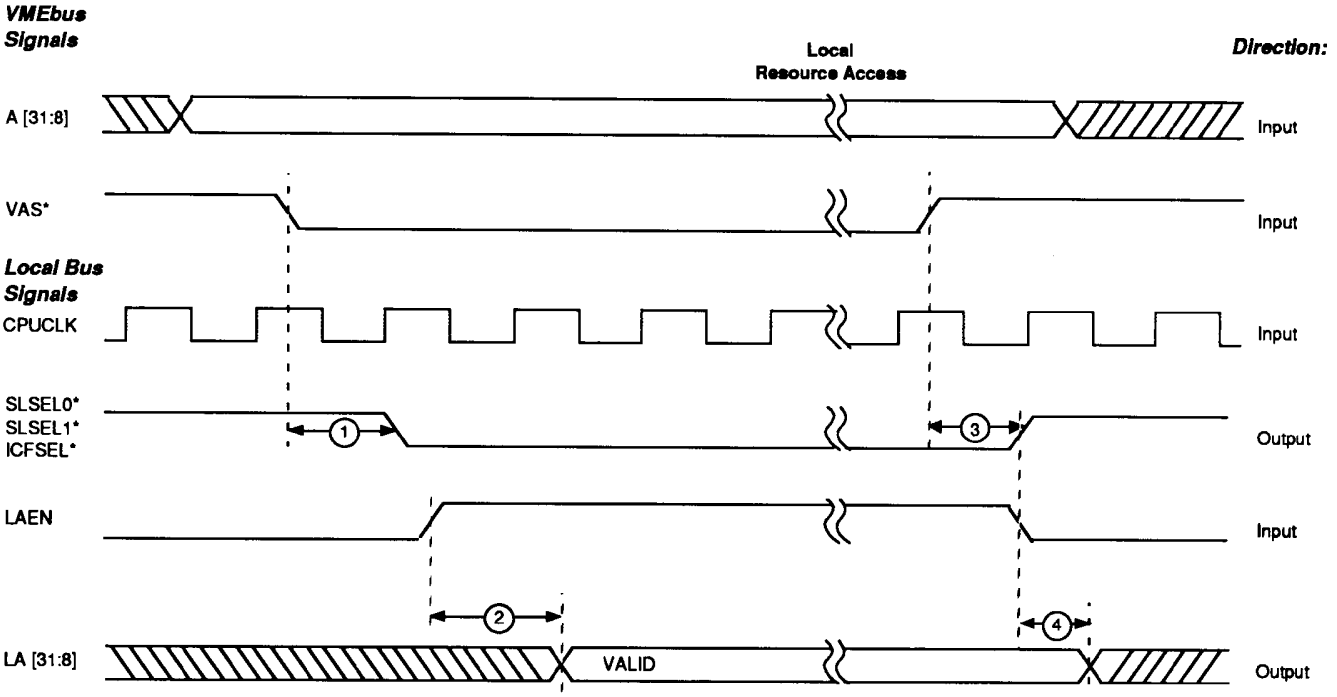


Figure 5-9. VMEbus Accesses – Slave

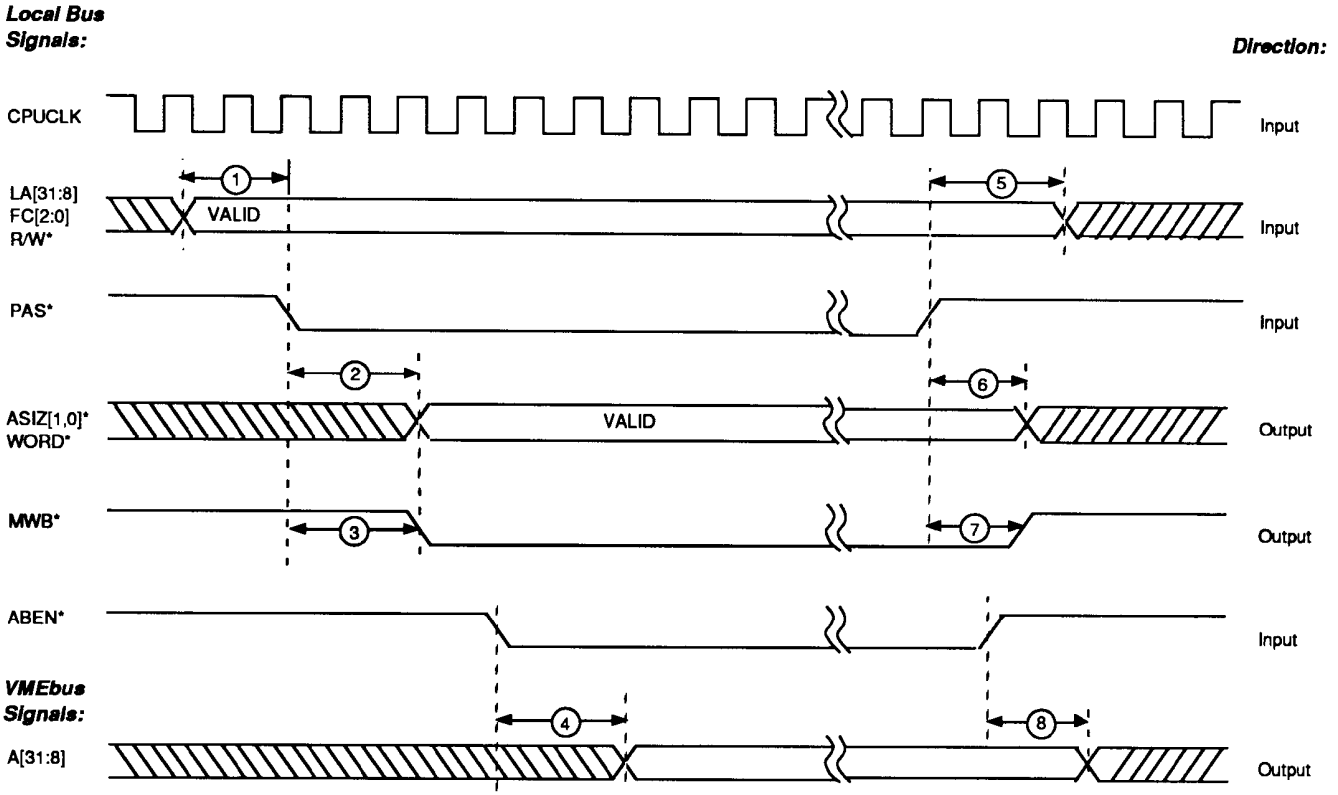


Figure 5-10. VMEbus Accesses – Master

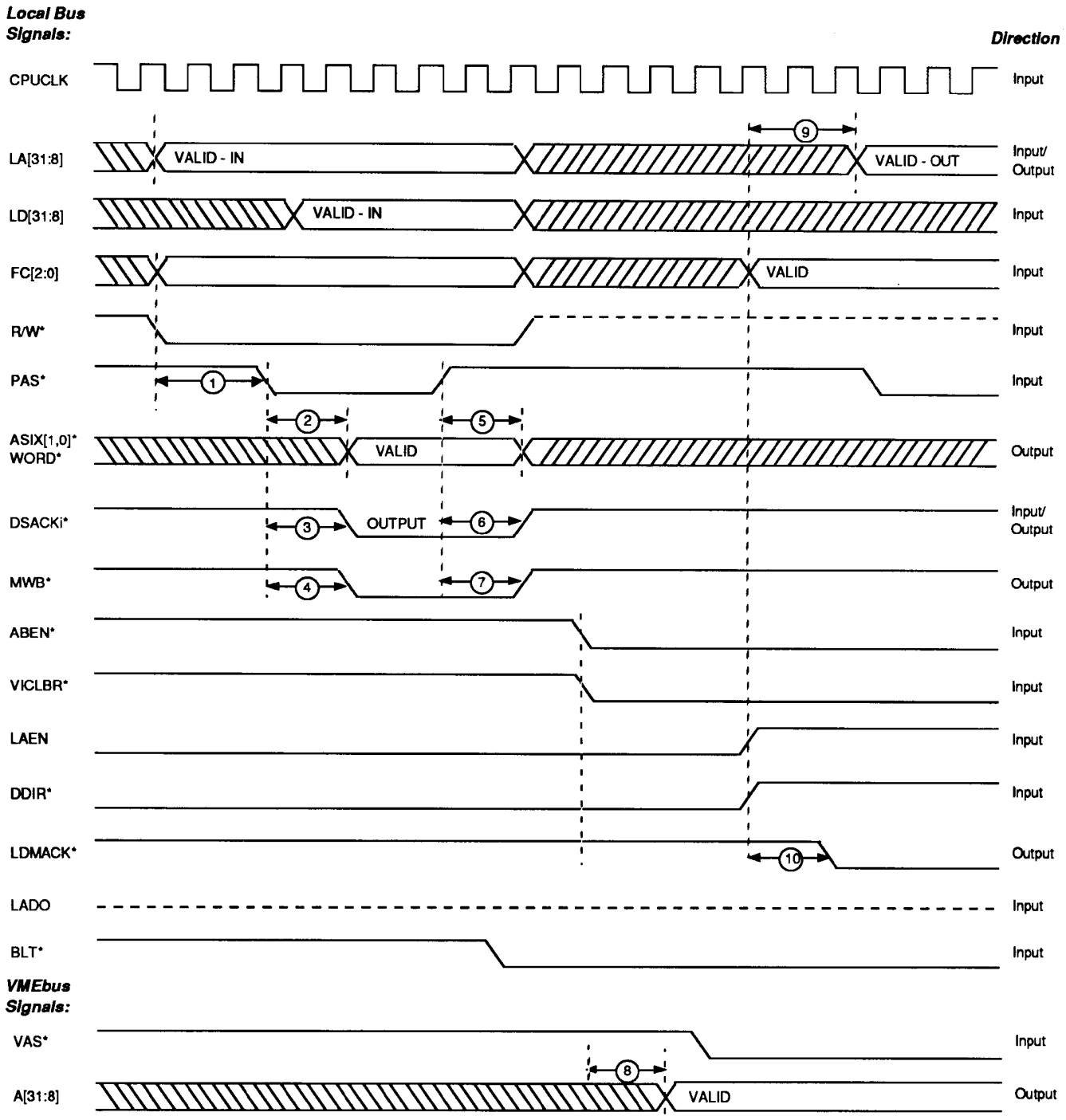


Figure 5-11. Master Block Transfer – Initialization Cycle

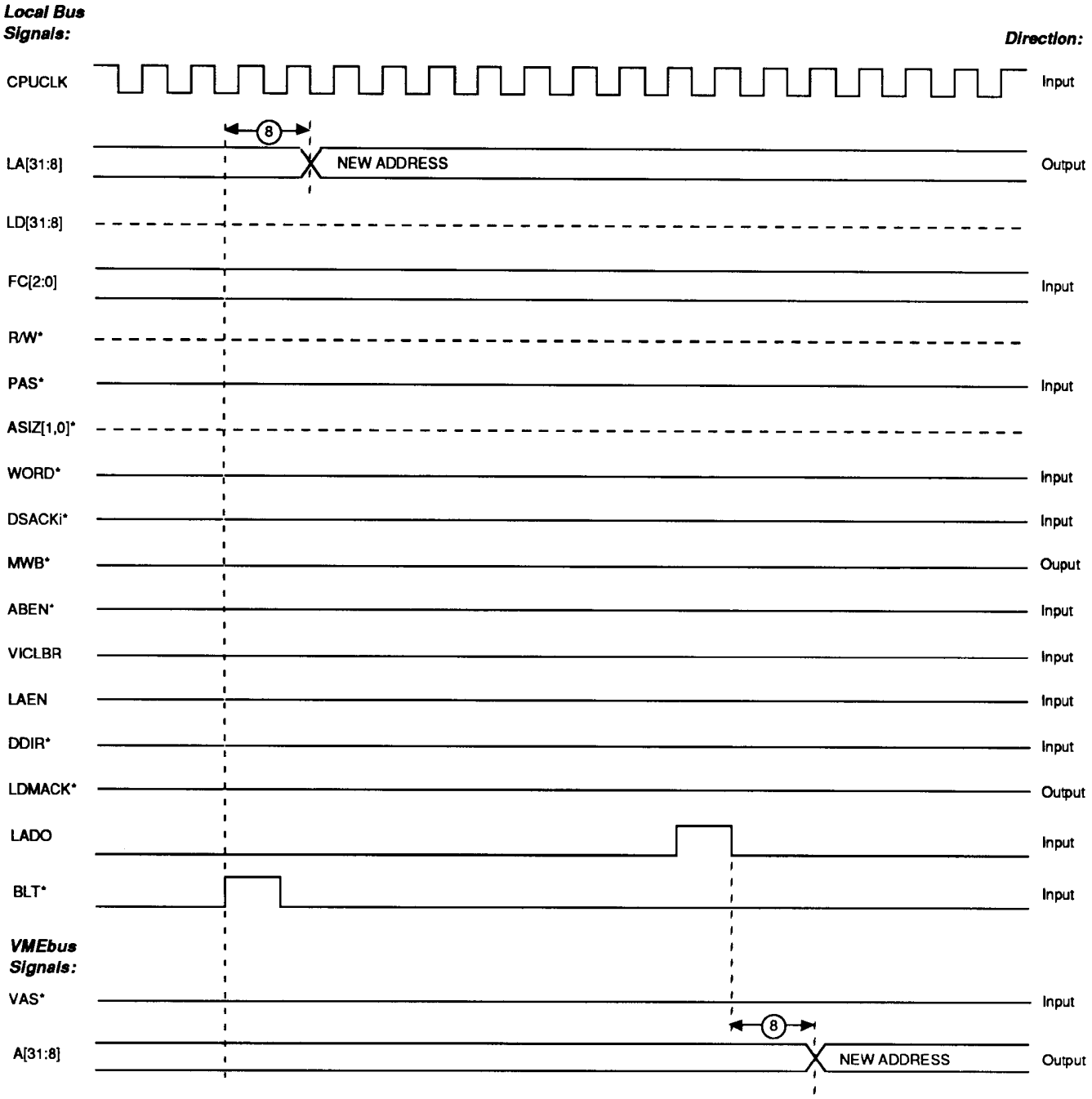


Figure 5-12. Master Block Transfer – Local and VME Boundary Crossing

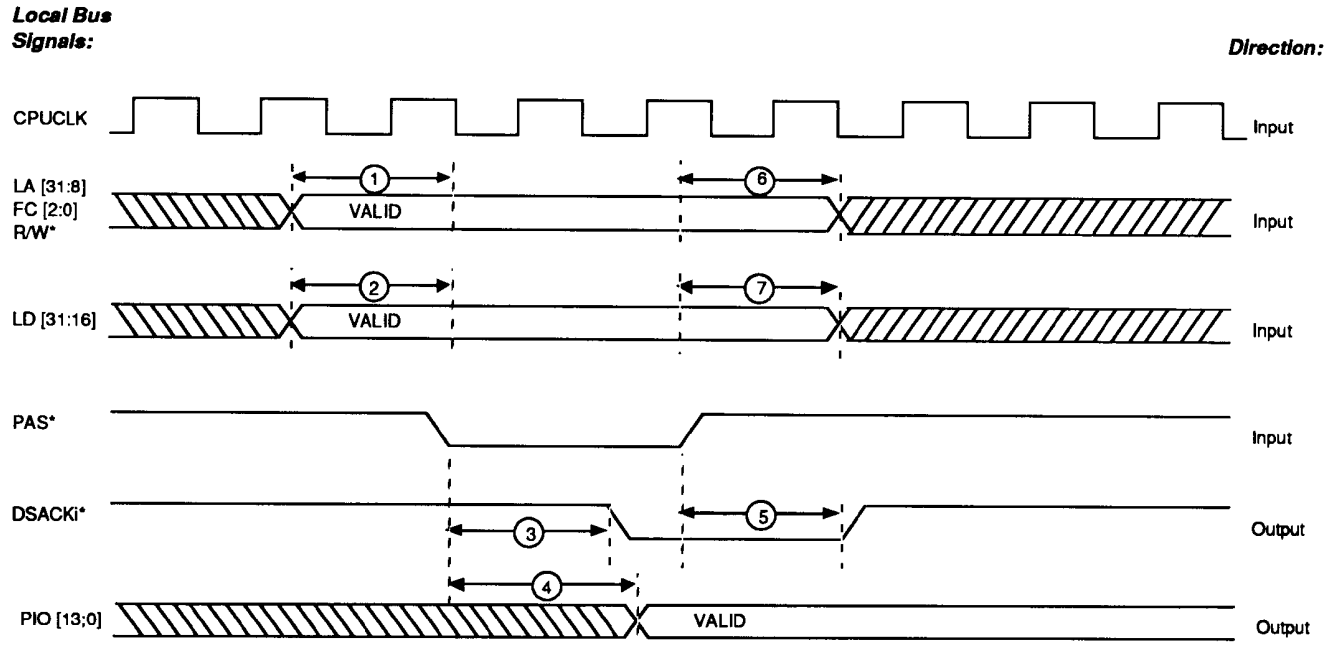


Figure 5-13. PIO Operation – Output

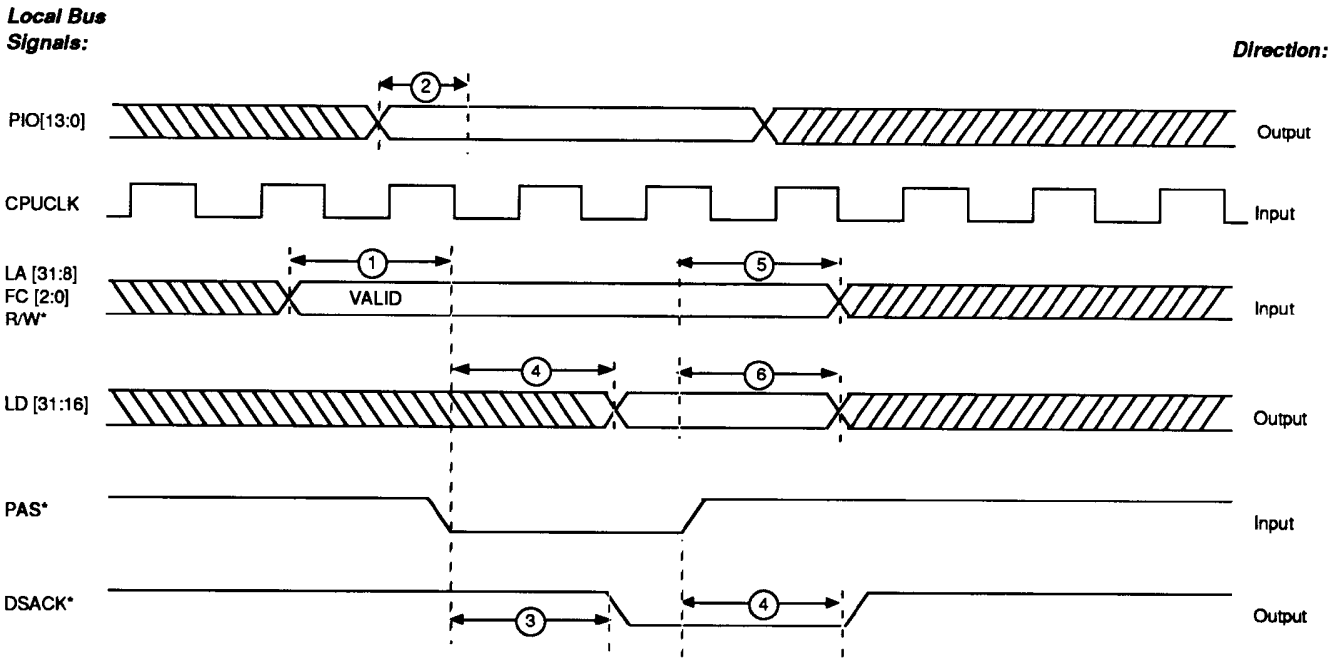


Figure 5-14. PIO Operation – Input

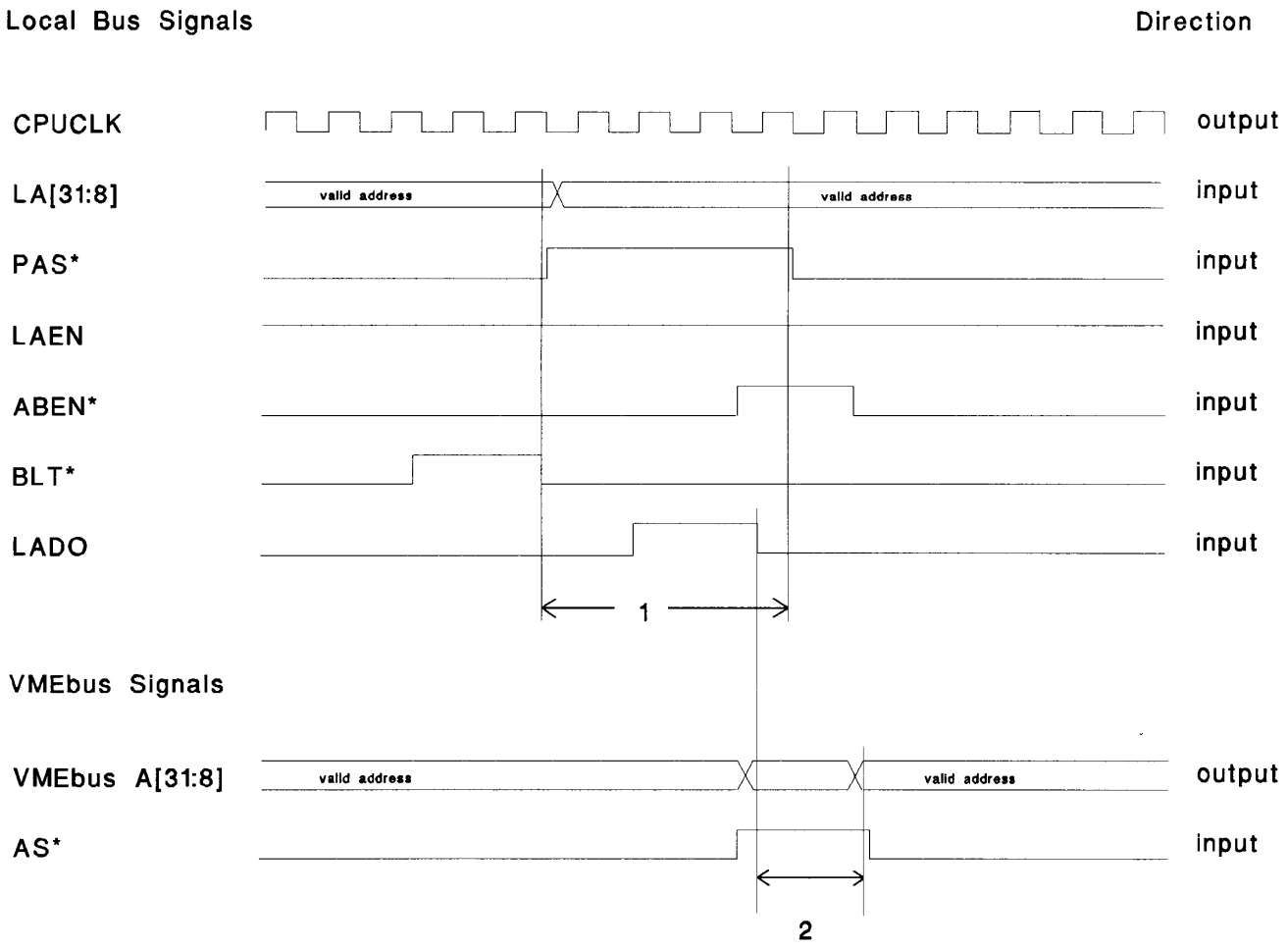


Figure 5-15. Master Block Transfer – Local and VMEbus Boundary Crossing (DMA Write Cycle)

5.7
VAC068A Signal List and Pinout

Table 5-2. 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 5-3. 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
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

Table 5-3. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
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
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

Table 5-3. Local Signals (continued)

Name	PGA Pin	QFP Pin	Type	Description
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
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 5-4. 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
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

Table 5-4. Power Supply Signals[1] (continued)

Name	PGA Pin	Type	Description
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 5-5. 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 5-5. Pinout for VAC068A Plastic and Ceramic Quad Flatpack (160-Pin): 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*	IDI0	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 5-16. VAC068A Pin Grid Array (PGA), Bottom View

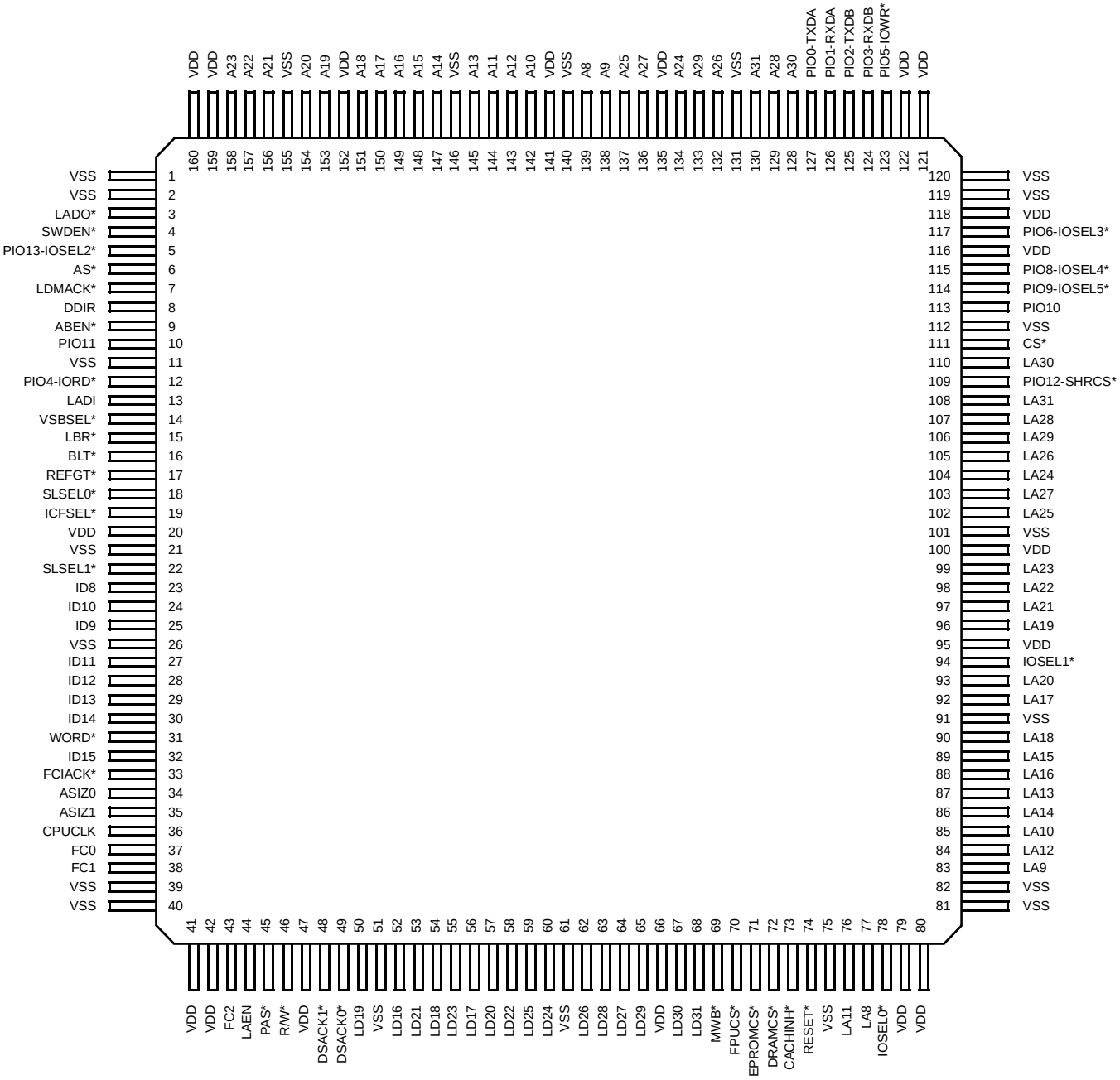


Figure 5-17. VAC068A Quad Flatpack (QFP), Top View

5.8

DC Performance Specifications

Table 5-6. VMEbus Signals (AS*, DS1*, DS0*, BCLR*, SYSCLK)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum Low-Level Output Voltage	V_{CC} = Min., I_{OL} = 48 mA, 56 mA, 64 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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 5-7. VMEbus Signals (Low Drive. All VMEbus Daisy-Chain Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 5-8. VMEbus Signals (Medium Drive. All non-High, non-Low Drive Signals, All VAC068A VMEbus Signals.)

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5	±5	±10	µA

Table 5-9. Non-VMEbus Signals

Parameter	Description	Test Conditions		Comm.	Industrial	Military	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} = –8 mA		2.4	2.4	2.4	V
V_{OL}	Maximum 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
V_{IK}	Input Clamp Voltage	V_{CC} = Min.	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		±5		±10	µA

Table 5-10. Capacitance

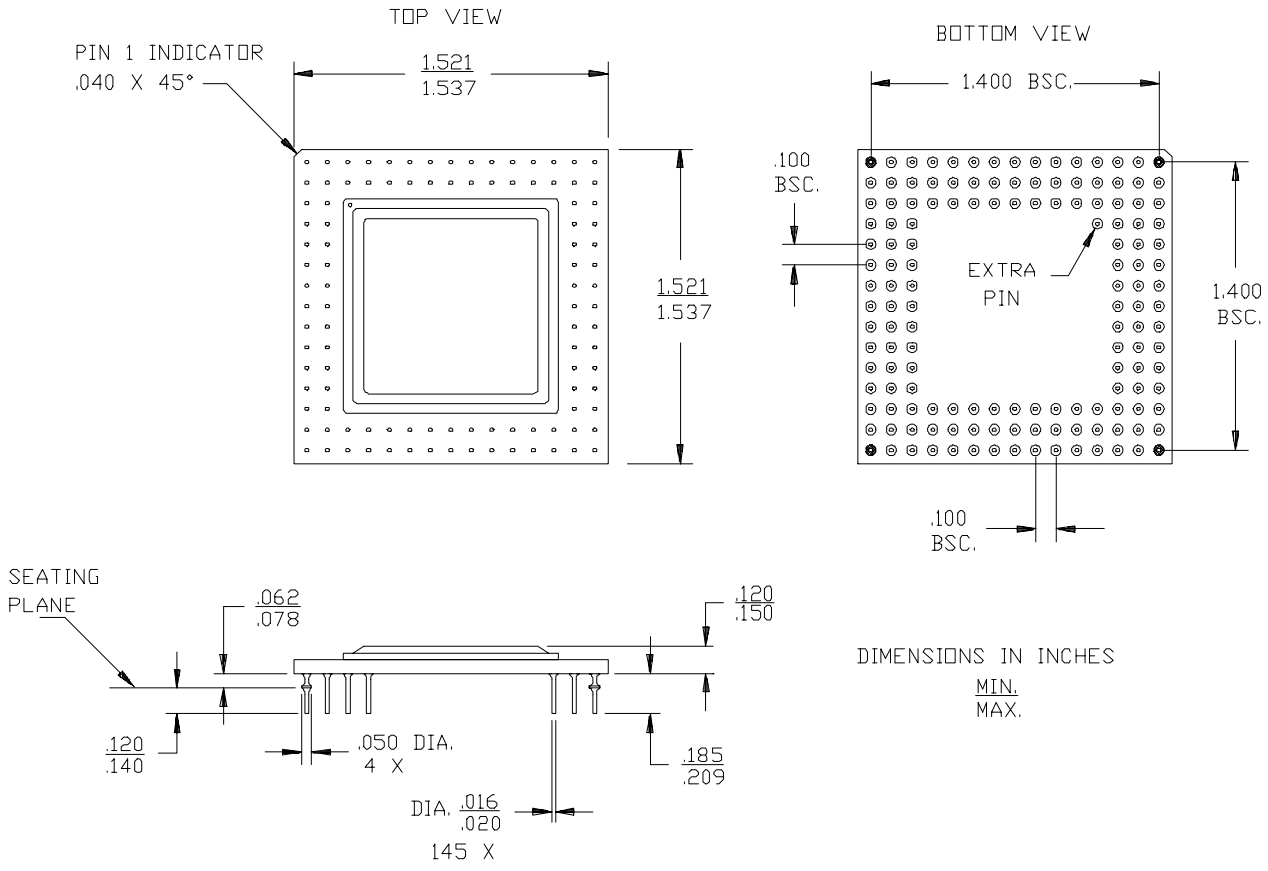
Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 64 MHz, V_{CC} = 5.0V	5	pF
C_{OUT}	Output Capacitance		7	pF

Table 5-11. Operating Current

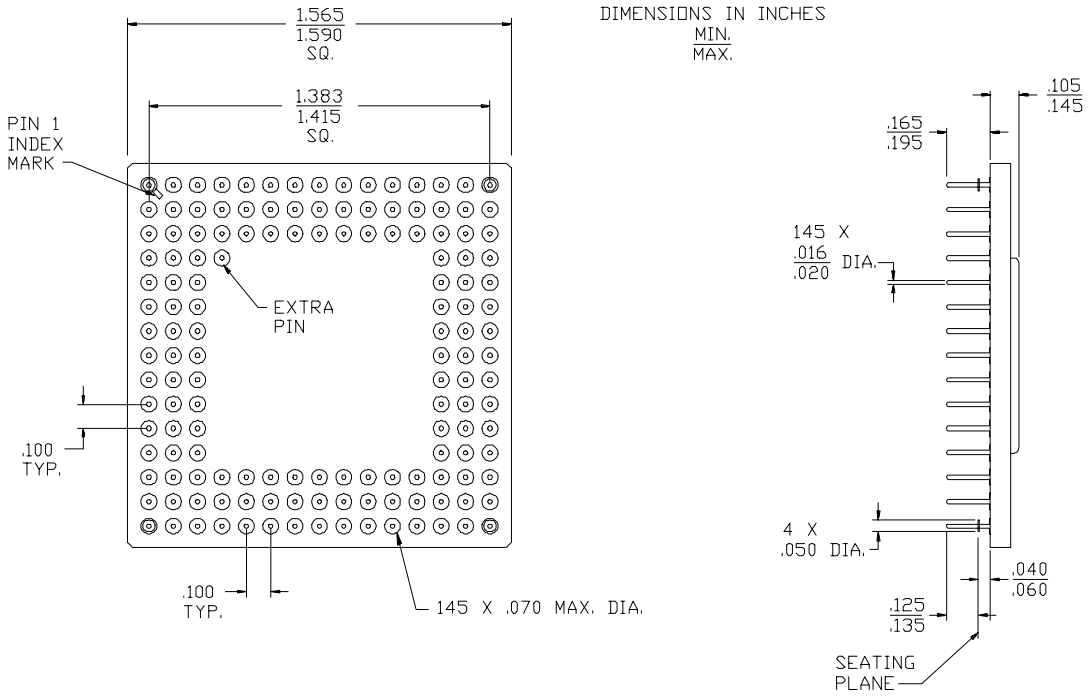
Parameters	Description	Test Conditions	Max.	Units
I_{DD}	Maximum Operating Current	No external DC load	150	mA

5.9
Package Diagrams

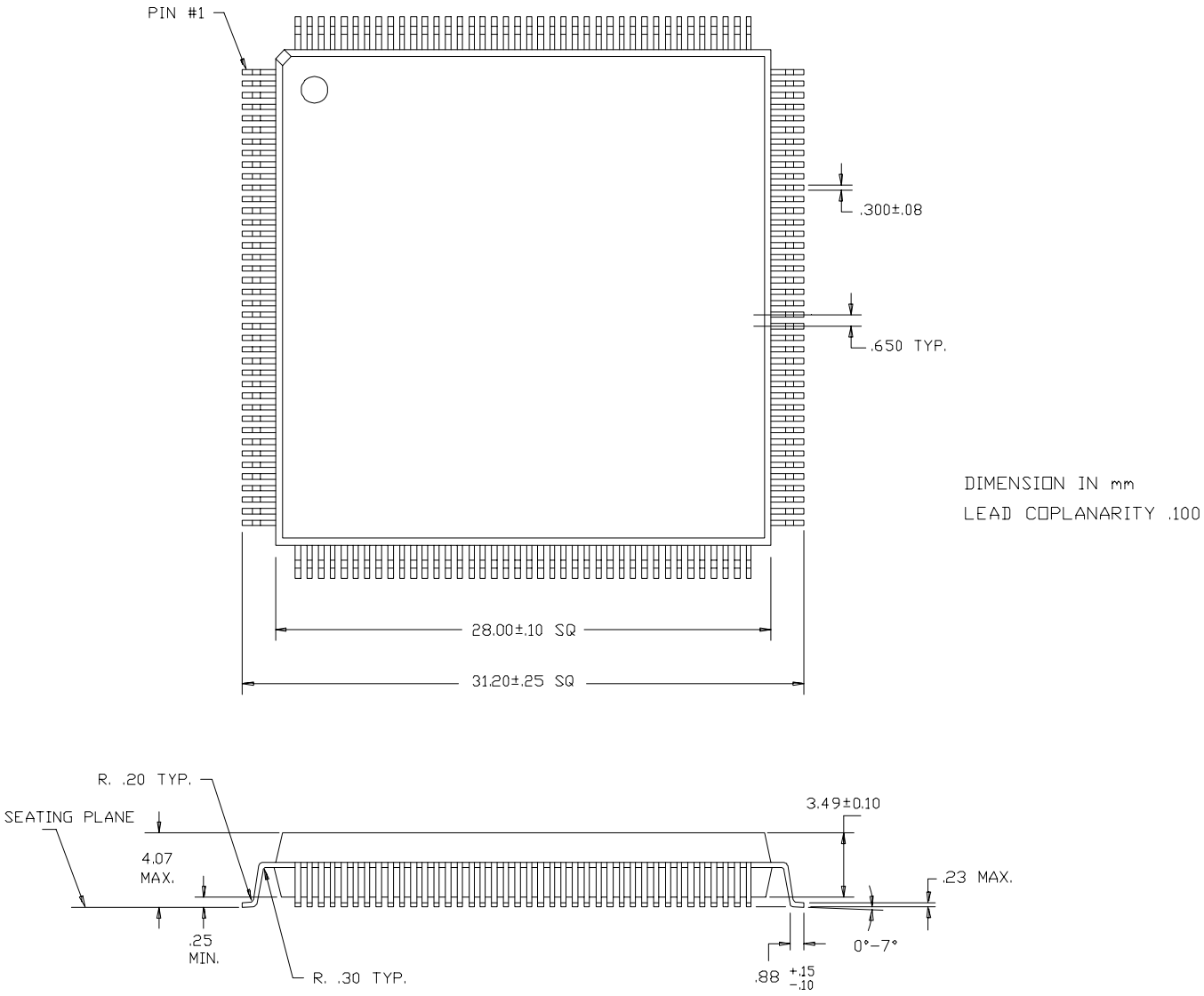
145-Pin Plastic Grid Array (Cavity Up) B144



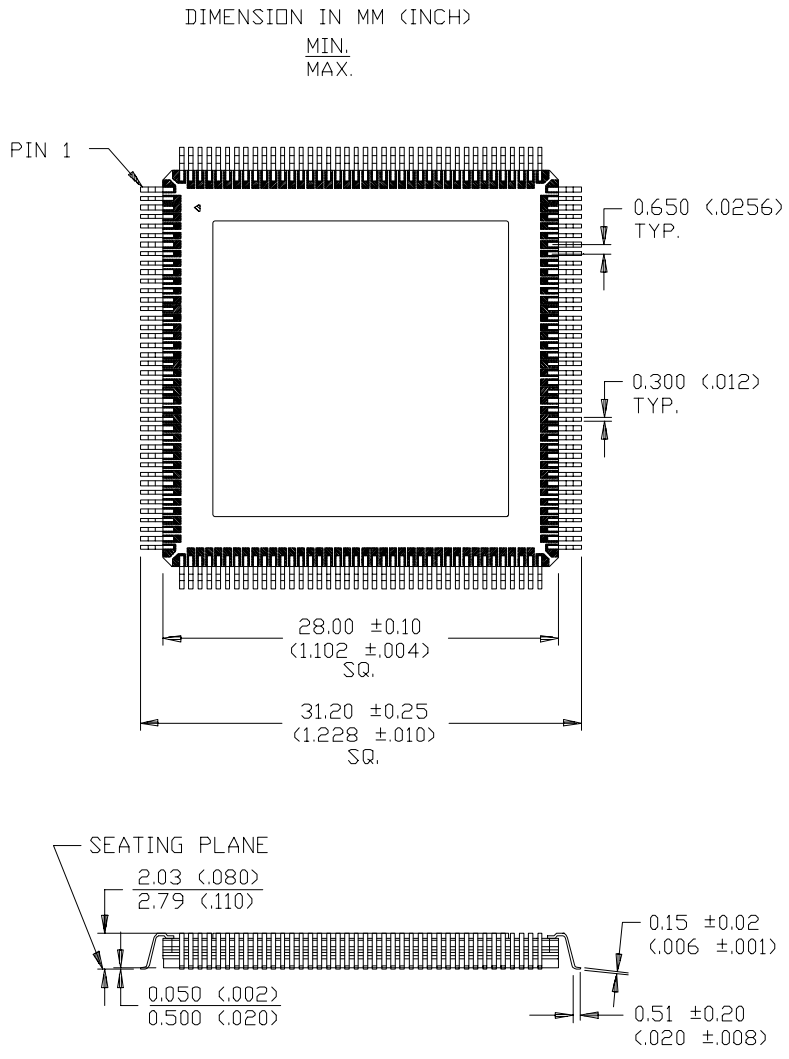
145-Pin Grid Array (Cavity Up) G145



160-Lead Plastic Quad Flatpack N160



160-Lead Ceramic Quad Flatpack (Cavity Up) 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.